CLASSIFICATION HIÉRARCHIQUE FLOUE BASÉE SUR LE SVM ET SON APPLICATION POUR LA CATÉGORISATION DES DOCUMENTS

par

Taoufik Guernine

Mémoire présenté au Département d'informatique
en vue de l'obtention du grade de maître ès sciences (M.Sc.)

FACULTÉ DES SCIENCES

UNIVERSITÉ DE SHERBROOKE

Sherbrooke, Québec, Canada, 14 avril 2010

Your file Votre référence
ISBN: 978-0-494-61414-3
Our file Notre référence
ISBN: 978-0-494-61414-3

Le 14 avril 2010

le jury a accepté le mémoire de Monsieur Guernine Taoufik dans sa version finale.

Membres du jury

Professeur Kacem Zeroual
Directeur de recherche
Département d'informatique

Professeur Djemel Ziou
Membre
Département d'informatique

Professeure Marie-Flavie Auclair-Fortier
Membre
Département d'informatique

Professeur Abdelhamid Benchakroun
Président rapporteur
Département d'informatique

Sommaire

La croissance exponentielle des moyens de communication durant ces dernières années et en particulier l'Internet a contribué à l'augmentation du volume de données traitées via les réseaux informatiques. Cette croissance a poussé les chercheurs à penser à la meilleure façon de structurer ces données pour faciliter leur accès et leur classification. À ce problème de classification, plusieurs techniques ont été proposées. Dans la pratique, nous constatons deux grandes familles de problèmes de classification, les problèmes binaires et les problèmes multi-classes. Le premier constat ayant attiré notre attention est l'existence du problème de confusion de classes lors de la classification. Ce phénomène rend les résultats ambigûs et non interprétables. Le deuxième constat est la difficulté de résoudre ces problèmes par les méthodes existantes surtout dans le cas où les données ne sont pas linéairement séparables. En outre, les méthodes existantes souffrent des problèmes de complexité en temps de calcul et d'espace mémoire. Afin de remédier à ces problèmes, nous proposons une nouvelle méthode de classification qui s'articule autour de trois principaux concepts : la classification hiérarchique, la théorie de la logique floue et la machine à vecteur de support (SVM). À cet égard et vu l'importance accordée au domaine de classification des textes, nous adaptons notre méthode pour faire face au problème de la catégorisation des textes. Nous testons la méthode proposée sur des données numériques et des données textuelles respectivement. Les résultats expérimentaux ont démontré une performance considérable comparativement à certaines méthodes de classification.

Remerciements

Tout d'abord, je remercie le bon Dieu de m'avoir aïdé à mener à terme ce travail.

Je tiens à remercier mon directeur de recherche, le professeur Kacem Zeroual, pour ses conseils judicieux, ses encouragements et son suivi permanent de l'évolution de ce travail. Je tiens à lui exprimer ma profonde gratitude pour son soutien technique et moral qu'il m'a prodigué durant la période du projet. Grâce à lui, j'ai repris confiance en moi et j'ai découvert un domaine de recherche qui aujourd'hui me passionne.

J'adresse mes remerciements aux professeurs qui m'ont fait l'honneur d'être membres de jury de ce mémoire.

Je désire également remercier le personnel du département d'informatique pour les services fournis. Je remercie également toutes les personnes qui m'ont apporté, de près ou de loin, leur précieuse contribution.

Il me parait opportun de dire un grand merci à mon père et à ma mère. Toutes les bonnes choses que j'ai pu réaliser dans cette vie, c'est grâce à vous. Je ne peux pas vous exprimer ma reconnaissance envers tout ce que vous avez fait pour moi. Je vous dis merci pour tout et je vous dédie ce travail.

Je réitère mes vifs remerciements à mes frères et mes sœurs pour leur encouragement et leur soutien.

J'adresse un remerciement particulier à ma femme et à ma petite fille Selsabil pour leur sacrifice et leur soutien moral durant les moments difficiles. Merci d'avoir rendu ma vie très agréable.

Mes vifs remerciements vont à mes amis, mes collègues et mes proches, qui m'ont toujours soutenu et encouragé au cours de la réalisation de ce mémoire et pour tout, je leur dédie ce travail.

Table des matières

Table des figures

Liste des tableaux

Introduction

Les grands projets faisant appel au forage de données représentent aujourd'hui, pour les entreprises, un des leviers les plus performants et surtout les mieux adaptés pour prendre des décisions. Le forage de données est une discipline récente relevant de l'analyse des données et qui s'intéresse à l'extraction de l'information cachée dans la donnée. Il représente de nos jours un outil opérationnel très important, touchant pratiquement une multitude de domaines à savoir l'industrie, la médecine et la bio-informatique.

Selon le MIT (*Massachussets Institute of Technology*), le forage de données est devenu durant ces dernières années l'une des dix technologies émergentes qui vont bouleverser le monde. Le forage de données se résume en deux grandes familles, celle des techniques descriptives et celle des techniques prédictives. Les techniques descriptives sont basées sur la réduction du volume de données de sorte qu'il n'existe pas une variable cible à prédire. Dans cette famille, nous citons à titre d'exemple l'analyse factorielle et la classification. À l'inverse, les techniques prédictives consistent à extraire des nouvelles connaissances en visant une variable cible à prédire. Cette famille englobe plusieurs techniques de prédiction. Nous citons à titre d'exemple les réseaux de neurones. Dans la pratique, le forage de données doit accomplir plusieurs tâches [55]. La classification est parmi les tâches descriptives les plus répandues dans le domaine du forage de données. Dans notre mémoire nous nous intéressons aux techniques descriptives et en particulier à la classification.

La tâche de classification fait partie de la statistique exploratoire multidimensionelle [69]. Elle est l'une des problématiques qui existent depuis longtemps. Plusieurs chercheurs ont traité ce problème de diverses façons. Nous citons à titre d'exemple les travaux de Duda et Hart [24], Aggrawal et Yu [4] et Benzécri [12]. La tâche de classification est subdivisée en deux familles : les méthodes supervisées et non supervisées. Dans la classification supervisée, les objets à classer sont étiquetés et le nombre de classes est connu *a priori*. Dans

le cas contraire, il s'agit d'une classification non supervisée. Quelle que soit la famille de classification, son principe consiste à regrouper les objets similaires dans la même classe selon une métrique de similarité. Le regroupement des données en classes est basé sur la notion de la pureté des classes [59].

La classification constitue une tâche fondamentale dans les approches d'apprentissage. La classification est considérée comme une tâche d'apprentissage supervisé lorsqu'elle permet de regrouper les objets en entrée en classes en fonction de certaines variables cibles, par exemple catégoriser un document par rapport à certains sujets pré-déterminés : sport, politique ou finance. La classification est, par contre, considérée une tâche d'apprentissage non supervisé lorsqu'elle ne dispose d'aucune information sur les variables cibles pour classer les objets en entrée. Ces tâches d'apprentissage (supervisé et non supervisé) peuvent être classées dans deux méthodologies d'apprentissage : apprentissage passif et apprentissage actif. Dans l'apprentissage passif [88], le choix de l'ensemble d'apprentissage est fait aléatoirement, sans aucune technique de sélection. Dans la méthodologie de l'apprentissage actif [16, 18, 76], le processus de classification nécessite une préparation de l'ensemble de données avant de le présenter au classificateur. L'utilisateur doit posséder une information *a priori* sur l'ensemble de données d'apprentissage. Dans notre travail, nous nous intéressons à l'apprentissage actif, en proposant une nouvelle technique de sélection de l'ensemble de données d'apprentissage du classificateur SVM. Le choix de l'ensemble de données d'apprentissage est basé sur la logique floue.

La croissance du volume des données dans les bases de données nécessite des techniques efficaces pour structurer et accéder à l'information en réduisant leur temps de traitement. En outre, cette croissance du volume de données rend la discrimination des données difficile dans le cas où les données ne sont pas linéairement séparables.

Plusieurs techniques ont été développées pour faire face aux problèmes de la classification. Nous citons à titre d'exemple les techniques de bases, à savoir les algorithmes hiérarchiques par agglomération, K-Moyennes et K-Médoïdes. D'autres travaux utilisent les méthodes hybrides pour accomplir la tâche de classification. L'objectif de l'hybridation consiste à concevoir un système performant, en exploitant les avantages de chacune des techniques combinées. Cette combinaison a donné lieu à la création d'un système hybride dont chacune des techniques possède des points forts lui permettant d'être bien adaptée à résoudre certains types de problèmes.

INTRODUCTION

Dans notre mémoire nous nous intéressons aux méthodes hybrides pour résoudre le problème multi-classes. Nous combinons les trois concepts suivants : (i) la classification hiérarchique, (ii) la théorie des ensembles flous, (iii) la machine à vecteur de support (SVM). Premièrement, la classification hiérarchique permet de structurer et d'organiser les classes à séparer en une structure hiérarchique facile à interpréter. Une succession de classes est obtenue tout au long de la hiérarchie. Ensuite, la classification floue permet de préciser le degré d'appartenance de chaque objet. Dans la classification floue, l'objet peut appartenir à plusieurs classes en même temps, en précisant son degré d'appartenance. Finalement, nous introduisons le classificateur SVM à chaque niveau de la hiérarchie dans le but de discriminer les classes. Notre méthode SVM-CHF consiste à construire dynamiquement une structure hiérarchique floue à partir des données d'apprentissage. Le choix de la structure hiérarchique floue basée sur le SVM est motivé, d'une part, par la rareté des travaux menés dans le domaine de la classification introduisant le SVM comme un classificateur et, d'autre part, par le besoin de convertir le problème de classification original en sous-problèmes binaires simples à résoudre.

La résolution des problèmes multi-classes est devenue une préoccupation majeure des chercheurs dans la communauté de classification. Le but visé dans ce mémoire est de mettre en place une méthode de classification pour résoudre le problème multi-classes, en minimisant le taux d'erreurs de notre classificateur. Le premier constat qui nous a poussé à développer cette méthode est la difficulté de discriminer les nombreuses classes existantes dans le monde réel. Deuxièmement, la résolution des problèmes multi-classes par les méthodes de classification existantes souffrent des problèmes de confusion de classes et de complexité en temps d'exécution et d'espace mémoire.

Étant donné, d'une part, que les données peuvent prendre plusieurs formes, à savoir l'image, le son ou le texte, et vu, d'autre part, la croissance rapide des données textuelles dans les bases de données, nous avons adapté notre méthode SVM-CHF pour le problème de catégorisation des textes. La représentation vectorielle des documents textuels a facilité l'adaptation de notre méthode pour manipuler les données textuelles. La nouvelle méthode adaptée SVM-CHF-Text consiste à regrouper les documents selon leur degré de similarité : document de forte, moyenne, faible et peu de similarité. Ce regroupement facilite et accélère la recherche et l'indexation des documents dans les bases de données.

Les résultats expérimentaux démontrent que notre nouvelle méthode SVM-CHF propo-

sée ainsi que la méthode adaptée SVM-CHF-Text sont très efficaces et performantes aussi bien pour les données numériques que textuelles.

Notre mémoire est organisé comme suit :

- **chapitre 1 :** Dans ce chapitre nous décrivons, dans un premier temps, les concepts fondamentaux de la classification en présentant les différents algorithmes de classification, à savoir les algorithmes hiérarchiques, par partition, les algorithmes utilisés dans la catégorisation des textes, et nous mettons l'accent sur les approches spécifiques de la classification. Dans un second temps, nous définissons le concept de la classification floue, en décrivant les concepts des ensembles flous et en présentant les algorithmes appartenant à cette catégorie de classification.
- **chapitre 2 :** Ce chapitre est subdivisé en trois grandes parties. La première partie est consacrée à la présentation de l'aspect théorique du SVM. Dans la deuxième partie, nous décrivons les différentes méthodes multi-classes basées sur le SVM. Dans la troisième partie, un accent particulier est mis sur les travaux connexes de la catégorisation des textes.
- **chapitre 3 :** Dans ce chapitre nous détaillons notre nouvelle méthode SVM-CHF développée pour le problème multi-classes. Ensuite, nous présentons notre algorithme développé pour les données numériques. La deuxième partie est consacrée à la présentation de la méthode adaptée pour les données textuelles SVM-CHF-Text.
- **chapitre 4 :** Ce chapitre est subdivisé en deux parties. La première partie est consacrée à la présentation des cas pratiques appliqués sur des données numériques et des données textuelles. Les résultats expérimentaux sont présentés dans la deuxième partie de ce chapitre.

  Finalement, nous concluons notre travail et nous évoquons nos perspectives de recherche.

Chapitre 1

Principe de la classification

Ce chapitre présente essentiellement les techniques fondamentales de la classification qui sont liées à notre champ d'étude. Nous commençons tout d'abord par décrire le contexte de la classification automatique d'une manière générale, en présentant les opérations à effectuer sur les objets, avant de commencer le processus de classification et nous présentons les différentes mesures de similarité utilisées dans le domaine de la classification : entre objets, entre classes et entre documents. Ensuite, nous présentons les différents algorithmes de classification, les approches spécifiques de la classification et nous mettons l'accent sur les algorithmes de base de la catégorisation de documents. Un accent particulier est mis sur la classification floue en définissant les concepts de base des ensembles flous.

1.1 Contexte de la classification

La classification automatique des données (*clustering* en anglais) est parmi les tâches les plus répandues dans le domaine du forage de données. Elle consiste à extraire de façon automatique les connaissances cachées dans les données en affectant chaque objet à sa propre classe. En fait, la classification porte sur le regroupement des individus ou d'observations similaires dans la même classe en respectant la variation inter-classes et la variation intra-classe. La figure 1.1 illustre le principe de la classification automatique.

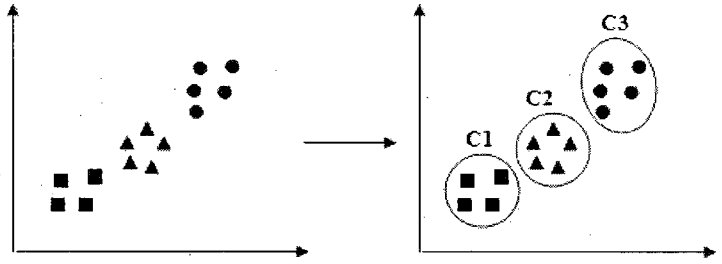


Figure 1.1 – Principe de la classification

D'une manière générale, la similarité entre les individus de la même classe doit être élevée et la similarité entre les individus qui appartiennent à des classes différentes doit être faible.

La tâche de classification comprend deux phases : la phase d'apprentissage et la phase de décision. Premièrement, la phase d'apprentissage consiste à prélever de l'ensemble X un sous ensemble Ω, appelé ensemble d'apprentissage à partir duquel nous construisons une fonction Φ qui permet de classer le plus correctement possible les éléments de Ω (voir la figure 1.2). Ensuite, la phase de décision consiste à classer de nouveaux objets qui n'appartiennent pas à l'ensemble d'apprentissage Ω (voir la figure 1.3).

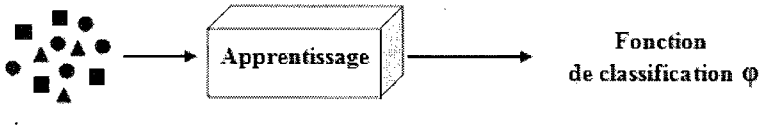


Figure 1.2 – Phase d'apprentissage du processus de classification

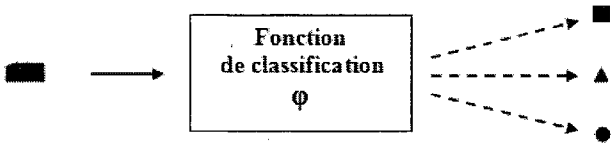


Figure 1.3 – Phase de décision du processus de classification

Selon Jain *et al.* [42], les méthodes de classification sont regroupées sous une forme hiérarchique, comme nous le montre la figure 1.4. Dans les méthodes exclusives, l'objet doit appartenir à une seule classe. Par contre, dans les méthodes non exclusives, l'objet peut appartenir à plusieurs classes en même temps avec un degré d'appartenance : c'est le cas de la classification floue. Dans les méthodes exclusives, le problème de classification est traité de deux façons : la classification non supervisée et la classification supervisée.

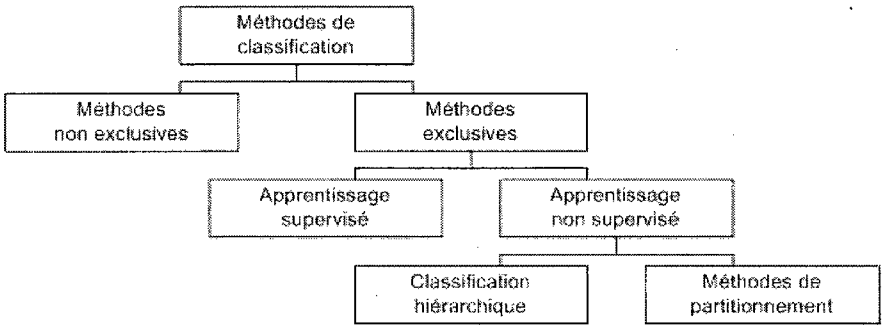


Figure 1.4 – Subdivision des méthodes de classification [42]

1.1.1 Domaine d'application de la classification

La classification est en pratique appliquée dans la plupart des domaines du monde réel. Nous la trouvons à titre d'exemple dans :

1. le Web, pour la classification des documents en fonction de leurs sujets et le filtrage des spams (spam/non spam) ;
2. le secteur médical, pour la classification des patients en fonction de leurs maladies ;
3. la bio-informatique, pour la classification des gènes quand une grande quantité de gènes peuvent montrer des comportements similaires ;
4. le marketing, pour la classification des entreprises en fonction de leurs productions.

1.1.2 Méthodologie adoptée

Supposons un ensemble d'observation $X = \{x_1, x_2, ..., x_n\}$ dans un espace $\mathbf{R^d}$, où n représente le nombre d'observations et d représente le nombre des attributs de chaque observation : $x_i = (x_{i1}, x_{i2}, ..., x_{id})$. Supposons, de plus, $W = \{w_1, w_2, ..., w_k\}$ un ensemble

de classes, où k représente le nombre de classes. Le but de la classification est d'affecter automatiquement chaque observation de l'ensemble X à une classe de l'ensemble W. La classification consiste à trouver un modèle (fonction objective) qui permet, à partir d'un ensemble d'apprentissage, de classer de nouveaux objets. La figure 1.5 est un exemple qui illustre le processus d'une classification automatique.

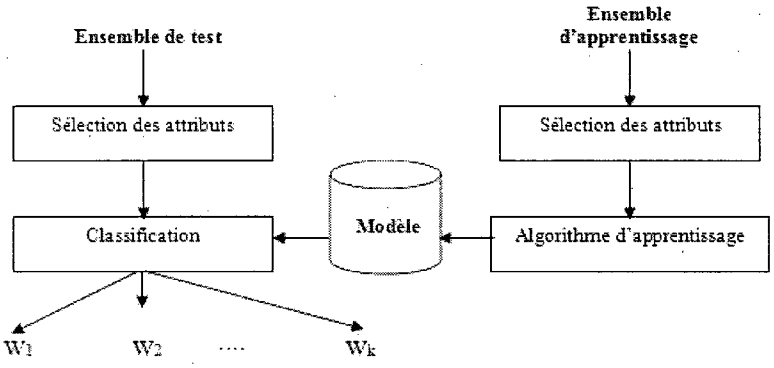


Figure 1.5 – Processus d'une classification automatique supervisée

Étant donné n attributs dans l'espace $\mathbf{R^n}$, la sélection des attributs consiste à choisir à partir de l'espace $\mathbf{R^n}$ les m attributs les plus discriminants, où $m < n$. La sélection des attributs permet de réduire le nombre d'attributs pour améliorer les performances de la classification. Plusieurs techniques ont été développées pour accomplir cette tâche de sélection [20, 27, 43].

L'application de l'algorithme d'apprentissage nécessite la spécification du nombre de données d'apprentissage et de la mesure de similarité. Plusieurs mesures de similarité peuvent être utilisées [23, 68]. La phase d'apprentissage consiste à concevoir un modèle de classification qui est appliqué sur de nouvelles données.

1.1.3 Évaluation des classificateurs

Il est nécessaire d'évaluer le degré de performance du classificateur. Pour ce faire, plusieurs mesures sont utilisées, à savoir le taux de classification, la matrice de confusion et la validation de l'ensemble d'apprentissage et de test.

Taux de classification

L'estimation du taux de classification est déterminée par le calcul de l'ensemble de test bien classé (respectivement mal classé) par rapport à tout l'ensemble de données :

$$Taux = \frac{TP}{N} \tag{1.1}$$

où TP représente l'ensemble de test bien classé et N représente tout l'ensemble de données.

Matrice de confusion

La matrice de confusion est une matrice carrée $[k \times k]$, où k représente le nombre de classes. Elle détermine l'erreur de classification de l'ensemble de test par rapport aux différentes classes. Chaque composante a_{ij} de la matrice de confusion indique le nombre d'exemples d'apprentissage dont la véritable classe est C_i alors qu'ils sont affectés à la classe C_j.

Validation de l'ensemble d'apprentissage et de test

Dans cette technique, l'ensemble de données est subdivisé en deux sous-ensembles séparés : un ensemble d'apprentissage et un ensemble de test. L'ensemble d'apprentissage est construit pour déterminer le bon classificateur. L'ensemble de test est construit pour évaluer la performance du classificateur choisi, en calculant le taux d'erreurs sur les nouvelles données. La technique la plus utilisée est celle de la validation croisée. Elle consiste à subdiviser l'ensemble de données en k sous-ensembles. Durant la classification, un ensemble parmi les k est choisi pour le test et le reste de l'échantillon est utilisé pour l'apprentissage. L'opération est répétée k fois et le taux d'erreur est déterminé à partir des k itérations.

1.1.4 Critères d'une partition : Inertie inter-classes et intra-classe

Pour regrouper les objets homogènes dans une même classe, nous avons besoin de deux critères, à savoir l'inertie inter-classes et l'inertie intra-classe. Dans cette section, nous décrivons ces deux critères de qualité d'une partition en détails.

Supposons $M = \{m_1, m_2, ..., m_n\}$, un ensemble de n objets, et ζ le centre de gravité du nuage des objets. Soit une classification de k classes $\{C_1, C_2,...,C_k\}$ d'effectifs $\{n_1, n_2,...,n_k\}$ et $\{c_1, c_2,...,c_k\}$ leurs centres de gravité.

– **Inertie inter-classes :** Elle est donnée par la formule suivante :

$$I_{inter} = \frac{1}{k}\sum_{i=1}^{k} n_i(c_i - \zeta)^2 \tag{1.2}$$

– **Inertie intra-classe :** Elle est donnée par la formule suivante :

$$I_{intra} = \frac{1}{n}\sum_{i=1}^{k}\sum_{m\in C_i}(m - c_i)^2 \tag{1.3}$$

Pour obtenir une meilleure classification, il faut que l'inertie inter-classes (I_{inter}) soit élevée et que l'inertie intra-classe (I_{intra}) soit faible. La figure 1.6 illustre les notions d'inerties inter-classes et intra-classe.

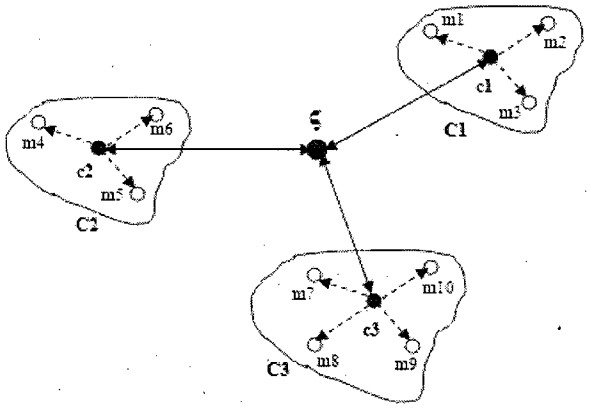


Figure 1.6 – Inerties inter-classes et intra-classe

1.2 Représentation des objets à classer

Afin de pouvoir accomplir la tâche de classification, il faut spécifier le format de l'objet à classer, à savoir : donnée numérique, texte, image ou son. Dans cette section, nous présentons les différentes tâches qui doivent être réalisées avant de présenter les objets

à classer aux algorithmes de classification. Ces tâches dépendent essentiellement du type d'objets à classer. Nous nous intéressons dans cette section à la représentation des données numériques et à la représentation des textes.

1.2.1 Représentation des données numériques

Dans un espace, la donnée est représentée par un vecteur de n dimensions qui est fonction du nombre d'attributs de chaque objet. Chaque attribut représente une dimension de l'espace de représentation. Avant de présenter ces données aux algorithmes de classification, certaines opérations doivent être réalisées :

- **Préparation des données** : La plupart des données qui existent dans les bases de données ne sont pas préparées. Il est nécessaire de transformer ces données dans un format adéquat afin de faciliter leur traitement. La préparation des données consiste à identifier les valeurs aberrantes, les valeurs manquantes et les valeurs redondantes. Des méthodes statistiques telle que la méthode d'amplitude interquartile et des méthodes graphiques telles que les méthodes des histogrammes et du nuage de points, sont utilisées pour préparer ces données [55].
- **Normalisation des données** : Une grande différence dans les amplitudes des objets conduit sans aucun doute à une mauvaise classification. Pour cela, la tâche de normalisation est nécessaire. Parmi les techniques les plus répandues dans le domaine de la classification, nous citons à titre d'exemple la normalisation min-max. Cette technique consiste à mesurer l'écart entre une donnée quelconque par rapport à la valeur minimale, pondérée par l'amplitude. Elle est exprimée par la formule suivante :

$$x^* = \frac{X - min(X)}{max(X) - min(X)} \tag{1.4}$$

où $X = (x_1, x_2, ..., x_n)$ est un vecteur de n dimensions.
Il existe d'autres méthodes de normalisation, nous citons à titre d'exemple la normalisation par le *test* Z. Cette technique consiste à mesurer l'écart entre la valeur de la variable et la moyenne des valeurs $\bar{X}$ de tous les objets, pondérée par l'écart type $\sigma(X)$ de l'ensemble des variables.

$$x^* = \frac{X - \bar{X}}{\sigma(X)} \tag{1.5}$$

Il faudra noter que la phase de préparation des données et la phase de normalisation sont des tâches primordiales et importantes pendant le processus de classification.

1.2.2 Représentation du texte

Bien que le but de ce mémoire soit de présenter une nouvelle méthode pour traiter le problème multi-classes et son application pour la classification des données numériques, nous mettons l'accent dans cette section sur les différentes techniques de la représentation des textes.

- **Prétraitement du texte** : La phase de prétraitement du texte est une phase importante dans le processus de catégorisation du texte. Elle consiste à normaliser et à nettoyer le texte à classer afin de faciliter son exploitation. La segmentation (*tokenization*) [1] et la radicalisation [2] sont les techniques les plus utilisées lors de la phase de prétraitement [67].
- **Représentation des documents** : Comme toute sorte de données, que ce soit image ou son, le texte nécessite une représentation durant la phase du traitement automatique. Dans le domaine de la recherche d'information, la représentation sans aucun doute la plus connue est la représentation vectorielle de *G. Salton* [74]. Dans cette symbolisation, chaque terme du vecteur représente une dimension de l'espace et chaque composante du vecteur représente le poids du terme dans le document. Les termes des vecteurs peuvent être des mots simples ou des séquences de mots. Pour associer un poids à un terme dans un document donné, nous disposons de trois méthodes de présentation :

  1. **Vecteur binaire** : La représentation par vecteur binaire est la représentation la plus ancienne et la plus simple parmi les représentations utilisées dans le domaine de la recherche d'information. L'association du poids au terme est simplement binaire : (1 si le mot est présent dans le texte, 0 sinon).
  2. **Vecteur fréquentiel** (*Term Frequency*) : Cette représentation donne une importance au nombre de termes qui apparaissent dans un document. Elle est basée

1. La segmentation consiste à découper le texte en morceau en éliminant les mots inutiles dans le processus de classification tels que les stop words, les mots rares, la ponctuation.
2. La radicalisation consiste à rendre le mot qui a subi plusieurs transformations (conjugaison, dérivation) à sa forme initiale (racine).

sur le nombre de fois (fréquence) du terme dans un document.

Selon Denoyer [22], certains problèmes surviennent dans le cas où la taille d'un document dans un corpus est supérieur à celle d'un autre document, ce qui influe sur l'interprétation de la représentation des documents lors de la recherche documentaire. Pour remédier à ce genre de problème, une normalisation des termes est exigée. La normalisation fréquentielle est obtenue en fractionnant le nombre de fréquences d'un mot par le nombre total des mots du même document.

3. **Vecteur** $TfIdf$ (*Term Frequency, Inverse Document Frequency*) : La représentation $TfIdf$ est la plus répandue dans le domaine de la recherche d'information [46]. Cette représentation décrit une relation entre les termes et leur fréquence dans le document [52]. Elle est basée sur la loi de *Zipf*[3] [22] (voir l'annexe A). La représentation vectorielle $TfIdf$ est donnée par la formule :

$$TfIdf = Tf \times Idf = Tf \times log(\frac{|D|}{df}) \tag{1.6}$$

où Tf représente le nombre d'apparitions du terme dans le document, $|D|$ représente le nombre global de documents dans le corpus et df représente le nombre de documents contenant le terme. Généralement, et afin d'éviter les problèmes liés à la longueur des documents, les vecteurs doivent subir une normalisation.

– **Réduction de la dimension du vocabulaire** : La taille immense du vocabulaire de la langue naturelle par rapport au nombre de documents influe sans aucun doute négativement sur le traitement automatique du texte. Pour cela, la réduction de l'espace vocabulaire devient une tâche indispensable pour éliminer les termes inutiles et non informatifs. En effet, plusieurs techniques ont été développées dans le but de réduire l'espace de représentation. Selon Sebastiani [78], la sélection des attributs (*features selection*)[4] et l'extraction des attributs (*features extraction*) sont parmi les techniques les plus utilisées dans le domaine du *Text Mining*. La technique la plus répandue pour

3. La loi de Zipf illustre l'importance du terme en fonction de sa fréquence dans un corpus. Un mot est considéré important s'il n'est ni trop fréquent ni trop rare.

4. Cette technique consiste à obtenir une cardinalité de vocabulaire inférieure à la cardinalité initiale : $|V'| < |V|$.

l'extraction des termes est la *Latent Semantic Indexing* (LSI) [21]. Dans cette représentation, chaque colonne correspond à un document d_j et chaque ligne correspond à un terme t_i. La relation terme/document est exprimée par la fréquence d'apparitions du terme t_i dans le document d_j.

Principe de LSI : Supposons une matrice d'occurrences A de taille $[M \times N]$, où M représente le nombre de mots et N représente le nombre de documents. LSI consiste en une décomposition en valeurs singulières de la matrice d'occurrences A. La décomposition en valeurs singulières permet de mesurer la similarité entre les documents où chaque colonne de la matrice V correspond à un document. Nous obtenons les matrices U, Σ et V :

$$A = U \times \Sigma \times V \tag{1.7}$$

- U est une matrice de taille $[M \times N]$;
- V et Σ sont de taille $[N \times N]$.

Selon la propriété de la décomposition en valeurs singulières :

1. $U^T U = V^T V = I_N$: $I_N[N \times N]$ représente la matrice identité ;
2. Σ = diag(σ_1,σ_2,...,σ_N).
   (a) $\sigma_i > 0$: Pour $0 \leq i \leq r$ où r est le rang de la matrice M ;
   (b) $\sigma_i = 0$: Pour $i \geq r$.
3. Les $\{\sigma_i\}$ représentent les racines carrées des valeurs propres de $M^T M$ et $M M^T$;
4. Les vecteurs propres de $M M^T$ constituent les colonnes de U (les vecteurs singuliers à gauche de M) ;
5. Les vecteurs propres de $M^T M$ constituent les colonnes de V (les vecteurs singuliers à droite de M).

1.3 Mesure de similarité

1.3.1 Mesure de similarité entre objets

La mesure de similarité consiste à définir une métrique pour regrouper les objets similaires dans la même classe. Cette métrique de similarité doit satisfaire les conditions suivantes [37] :

1. dist(x_i, x_j) $\geq 0, \forall\ (x_i, x_j) \in X$ (positive) ;
2. dist(x_i, x_j) = dist(x_j, x_i), $\forall\ (x_i, x_j) \in X$ (symétrique) ;
3. $\forall\ (x_i, x_j) \in X$: dist(x_i, x_j) = $0 \Leftrightarrow x_i = x_j$;
4. dist(x_i, x_k) $\leq$ dist(x_i, x_j) + dist(x_j, x_k), $\forall\ x_i, x_j, x_k \in X$ (inégalité triangulaire).

– **Mesure de distance :** Soient X_i et Y_j deux objets dans un espace R^n, représentés respectivement par leurs vecteurs $\vec{X_i} = (x_{i1}, x_{i2}, ..., x_{in})$ et $\vec{Y_j} = (y_{j1}, y_{j2}, ..., y_{jn})$ de longueur n. La similarité entre ces deux objets est calculée par plusieurs mesures de distance.

– Distance de Manhattan

$$dist(X_i, Y_j) = \|\vec{X_i} - \vec{Y_j}\|_1 = \sum_{k=1}^{n} |x_{ik} - y_{jk}| \tag{1.8}$$

– Distance Euclidienne

$$dist(X_i, Y_j) = \|\vec{X_i} - \vec{Y_j}\|_2 = \sqrt{\sum_{k=1}^{n} (x_{ik} - y_{jk})^2} \tag{1.9}$$

– Distance de Minkowski

$$dist(X_i, Y_j) = \|\vec{X_i} - \vec{Y_j}\|_m = \sqrt[m]{\sum_{k=1}^{n} |x_{ik} - y_{jk}|^m} \tag{1.10}$$

Où m est un entier positif :

– Pour $m = 1$: distance de Manhattan ;

– Pour $m = 2$: distance Euclidienne ;

– Pour $m \rightarrow \infty$: distance de Chebychev :

$$dist(X_i, Y_j) = \|\vec{X_i} - \vec{Y_j}\|_\infty = \max_{1 \leq k \leq n} |x_{ik} - y_{jk}| \tag{1.11}$$

- **Le cosinus :** Le cosinus est parmi les mesures de similarité les plus répandues dans le domaine de la classification. La similarité entre deux objets X et Y dépend de l'angle existant entre les deux vecteurs représentatifs de ces objets. Deux objets sont plus similaires si l'angle formé entre les deux vecteurs est petit. Formellement, le cosinus de deux vecteurs est obtenu par :

$$Cos(\vec{X_i}, \vec{Y_j}) = \frac{\vec{X_i} \cdot \vec{Y_j}}{\|\vec{X_i}\| \cdot \|\vec{Y_j}\|} = \frac{\sum_{k=1}^{n} x_{ik} \cdot y_{jk}}{\sqrt{\sum_{k=1}^{n} x_{ik}^2} \cdot \sqrt{\sum_{k=1}^{n} y_{jk}^2}} \tag{1.12}$$

où $(\vec{X_i} \cdot \vec{Y_j})$ représente le produit scalaire de $\vec{X_i}$ et $\vec{Y_j}$, et $\|\vec{X_i}\|$ et $\|\vec{Y_j}\|$ représentent les normes de $\vec{X_i}$ et $\vec{Y_j}$ respectivement.

1.3.2 Mesure de similarité entre classes

Plusieurs méthodes ont été proposées pour calculer la mesure de similarité entre les classes. Elles sont largement utilisées par les algorithmes hiérarchiques. L'ensemble de données est remplacé par des classes. Chaque classe représente un sous-ensemble de données dans lequel les objets partagent les mêmes caractéristiques. L'objectif de l'utilisation de ces méthodes est de réduire la complexité des algorithmes de classification et leur temps de calcul. Supposons deux classes C_p et C_q de n et m objets respectivement et supposons g_p et g_q leurs centroïdes. Considérons x_i un élément de C_p et x_j un élément de C_q :

- **Technique du centroïde :** La mesure de similarité entre deux classes est obtenue par la mesure de distance entre leurs centroïdes. La distance entre deux classes C_p et C_q est donnée comme suit :

$$Distance(C_p, C_q) = dist(g_p, g_q) \tag{1.13}$$

où $g_i = \frac{\sum_{x_j \in C_i} x_j}{n_i}$.

- **Technique du médoïde :** Contrairement à la technique du centroïde, la mesure de similarité entre deux classes C_p et C_q est obtenue en mesurant la distance entre leurs médoïdes m_p et m_q respectivement. La distance entre deux classes C_p et C_q est donnée comme suit :

$$Distance(C_p, C_q) = dist(m_p, m_q) \tag{1.14}$$

où $m_i = arg \min_{x_j \in C_i} \sum_{x_k \in C_i} dist(x_j, x_k)$.

- **Technique du plus proche voisin :** La distance entre deux classes est définie comme la distance minimale entre toutes les paires d'objets de deux classes. Elle est définie comme suit :

$$Distance(C_p, C_q) = \min_{\substack{x_i \in C_p \\ x_j \in C_q}} dist(x_i, x_j) \tag{1.15}$$

- **Technique du voisin le plus éloigné :** La distance entre deux classes est définie comme la distance maximale entre toutes les paires d'objets de deux classes. Elle est définie comme suit :

$$Distance(C_p, C_q) = \max_{\substack{x_i \in C_p \\ x_j \in C_q}} dist(x_i, x_j) \tag{1.16}$$

- **Technique de la moyenne de groupe :** La distance entre deux classes est définie comme la distance moyenne entre toutes les paires d'objets de deux classes. Elle est définie comme suit :

$$Distance(C_p, C_q) = \frac{\sum_{x_i \in C_p} \sum_{x_j \in C_q} dist(x_i, x_j)}{n_p \cdot n_q} \tag{1.17}$$

où n_p et n_q représentent le nombre d'objet de C_p et C_q respectivement.

1.3.3 Mesure de similarité entre documents

La classification des documents consiste à regrouper les documents les plus similaires dans la même classe en se basant sur leur contenu contextuel. La définition d'une mesure de similarité est nécessaire pour déterminer le degré de similarité entre les documents.

Plusieurs mesures de similarité sont utilisées dans le domaine de la catégorisation du texte. Les mesures de distance et de cosinus que nous avons étudiées dans la section (1.3.1) sont utilisées pour mesurer la similarité entre des documents qui sont considérés comme des objets représentés par des vecteurs. Nous définissons dans cette section les coefficients d'association et le coefficient de corrélation qui sont aussi utilisés pour mesurer la similarité entre les documents :

- **Coefficients d'association :** La technique des coefficients d'association est utilisée pour la mesure de similarité entre les documents. Les coefficients les plus répandus dans le domaine de la recherche d'information sont le coefficient de Jaccard et le coefficient de Dice qui sont basés sur les caractéristiques communes entre deux documents. Les deux métriques sont exprimées par les deux formules suivantes :

  1. **Coefficient de Jaccard :**

$$Jaccard(\vec{d_i}, \vec{d_j}) = \frac{\sum_{k=1}^{n} d_{ik} d_{jk}}{\sum_{k=1}^{n} d_{ik}^2 + \sum_{k=1}^{n} d_{jk}^2 - \sum_{k=1}^{n} d_{ik} d_{jk}} \tag{1.18}$$

  2. **Coefficient de Dice :**

$$Dice(\vec{d_i}, \vec{d_j}) = \frac{2\sum_{k=1}^{n} d_{ik} d_{jk}}{\sum_{k=1}^{n} d_{ik}^2 + \sum_{k=1}^{n} d_{jk}^2} \tag{1.19}$$

  Le coefficient de Jaccard normalise le produit scalaire de deux vecteurs en divisant par la somme des composantes non nulles de l'union de deux vecteurs. Le coefficient de Dice normalise le produit scalaire de deux vecteurs en divisant par la somme des composantes non nulles de deux vecteurs.

- **Coefficients de corrélation :** Le coefficient de corrélation de Pearson est parmi les mesures de similarité les plus répandues dans le domaine de la recherche d'information. Il consiste à ressortir le rapport existant entre deux documents. Supposons deux documents d_i et d_j représentés par leur vecteur $\vec{d_i}$= $(d_{i1}, d_{i2}, ..., d_{in})$ et $\vec{d_j}$= $(d_{j1}, d_{j2}, ..., d_{jn})$. Formellement, le coefficient de corrélation de Pearson est exprimé par la formule suivante :

$$Pearson(\vec{d_i}, \vec{d_j}) = \frac{\sum_{k=1}^{n}(d_{ik} - \bar{d_i})(d_{jk} - \bar{d_j})}{\sqrt{\sum_{k=1}^{n}(d_{ik} - \bar{d_i})^2}\sqrt{\sum_{k=1}^{n}(d_{jk} - \bar{d_j})^2}} \quad (1.20)$$

où $\bar{d_i} = \frac{\sum_{k=1}^{n} d_{ik}}{n}$ et $\bar{d_j} = \frac{\sum_{k=1}^{n} d_{jk}}{n}$.

Dans les travaux de Strehl *et al.* [84], une étude comparative a été effectuée entre les différentes mesures de similarité. Ils ont constaté que le cosinus et le coefficient de Jaccard sont les mesures de similarité les plus performantes dans le domaine de la catégorisation de texte.

1.4 Algorithmes généraux de classification

Nous présentons en détails dans cette section les algorithmes de classification qui sont largement utilisés dans le domaine de la classification. Ces algorithmes utilisent les différentes mesures de similarité que nous avons déjà présentées dans les sections précédentes. Dans la littérature, nous distinguons essentiellement deux grandes familles d'algorithmes : les algorithmes hiérarchiques et les algorithmes par partition. Nous présentons ces deux familles d'algorithmes dans les sections qui suivent.

1.4.1 Algorithmes hiérarchiques

Les algorithmes hiérarchiques consistent à produire une succession hiérarchique de classes. Cette représentation hiérarchique s'appelle dendrogramme. La figure 1.7 représente un exemple illustratif d'un dendrogramme.

Les techniques utilisées pour construire cette hiérarchie sont des techniques par agglomération (combinaison) ou des méthodes divisive (descendante). Dans la technique d'agglomération, à chaque niveau de la hiérarchie, une diminution du nombre de classes par la fusion de deux classes les plus similaires dans une seule classe, jusqu'à la vérification d'un critère d'arrêt. Dans la technique divisive, l'ensemble de données est regroupé dans un seul groupe contenant toutes les classes et, à chaque niveau de la hiérarchie, les deux classes les

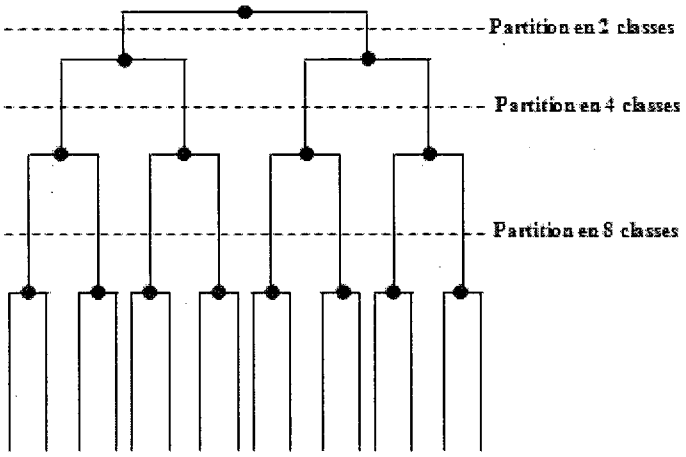


Figure 1.7 – Exemple de dendrogramme

plus dissemblables sont séparées. De cette façon, le nombre de classes augmente jusqu'à ce qu'un critère d'arrêt soit vérifié. L'algorithme hiérarchique par agglomération est décrit dans (Algorithme 1). Le paramètre ξ représente généralement l'erreur de classification au carré.

Algorithme 1 Algorithme de Classification Hiérarchique Agglomérative

Entrée : Ensemble $X = \{x_1, x_2,...,x_n\}$, ξ : critère d'arrêt, β : indice de similarité ;
Sortie : Ensemble de classes $C = \{C_1, C_2,...,C_k\}$;
1. Pour chaque paire (C_i,C_j), calculer la similarité $Sim(C_i,C_j)$;
2. Si $Sim(C_i,C_j) \leq \beta$ **Alors** $\{C_i, C_j\} \in C$;
3. Répéter les étapes (1) et (2) jusqu'à ce que ξ soit vérifié.

D'un autre coté, plusieurs indices de similarité sont utilisés pour mesurer la similarité entre les classes. Les définitions les plus usuelles :

Association simple : L'association simple est basée sur la mesure du plus proche voisin, exprimée par l'équation (1.15). À chaque niveau de la hiérarchie, les deux classes les plus similaires sont fusionnées en une seule classe. Cette technique est sensible à l'effet de chaîne, c'est-à-dire que dans certains cas nous risquons de regrouper deux classes éloignées qui sont reliées par une chaîne d'objets.

Association complète : L'association complète est basée sur la mesure du voisin le plus

éloigné, exprimée par l'équation (1.16). Elle nécessite un temps de calcul très élevé.

Association moyenne : L'association moyenne est basée sur la distance moyenne entre tous les objets de deux classes. Elle se situe entre l'association simple et l'association complète. Elle est exprimée par l'équation (1.17). Les résultats expérimentaux [83] ont montré que les techniques qui utilisent l'association moyenne donnent de meilleurs résultats que les techniques qui utilisent l'association simple et l'association complète.

La complexité en temps de calcul des algorithmes hiérarchiques est de $O(n^2)$, où n représente le nombre des objets dans la base de données. L'espace mémoire pour l'exécution des algorithmes hiérarchiques est proportionnel au nombre des classes. L'un des avantages des algorithmes hiérarchiques est que le nombre de classes n'est pas fixé *a priori*. Cela permet de choisir un nombre de classes de façon optimale. D'un autre coté, l'utilisation des métriques de distance telle que la distance euclidienne rend l'interprétation des résultats des algorithmes hiérarchiques très simple. Un autre avantage de ces algorithmes est que les algorithmes hiérarchiques permettent de trouver des classes de formes diverses, selon la distance choisie [89]. À l'opposé, la complexité en temps de calcul et en espace représente l'inconvénient majeur des algorithmes hiérarchiques. Pour passer de k classes à $(k-1)$ classes, il faut calculer $(\frac{k(k-1)}{2})$ distances afin de trouver les classes les plus proches.

1.4.2 Algorithmes par partition

Contrairement aux méthodes hiérarchiques, les méthodes par partition consistent à regrouper les données dans un nombre de classes fixé *a priori* par l'utilisateur. Les algorithmes par partition les plus connus dans le domaine de la classification sont présentés comme suit :

Algorithme K-Moyennes

L'algorithme K-Moyennes a été introduit par Mac Queen [66]. L'idée de base de K-Moyennes est d'affecter tous les objets dans des classes fixées *a priori* en minimisant l'erreur de classification des objets. Il commence d'abord par choisir aléatoirement les centres initiaux de classes. Chaque objet est affecté à la classe la plus proche. Le centre de gravité g_i d'une classe est calculé à chaque itération comme suit :

$$g_i = \frac{1}{C_i} \sum_{j=1}^{C_i} m_j^{(i)} \tag{1.21}$$

où $m_j^{(i)}$ représente le $j^{ème}$ objet affecté à la classe C_i. À chaque itération, une erreur de classification ξ liée à l'affectation des objets à une classe donnée est exprimée comme suit :

$$\xi = \sum_{i=1}^{k} \sum_{m_j \in C_i} d^2(m_j, C_i) \tag{1.22}$$

Il existe plusieurs façons pour choisir les k centroïdes :

- utiliser les k premiers objets ;
- choisir les k objets aléatoirement ;
- sélectionner aléatoirement une partition de k classes ensuite calculer son centroïde.

Un pseudo-code de l'algorithme K-Moyennes est présenté dans (Algorithme 2).

Algorithme 2 Algorithme de classification K-Moyennes

Entrée : Ensemble $X = \{x_1, x_2,...,x_n\}$, k : nombre de classes ;
Sortie : Ensemble de classes $C = \{C_1, C_2,...,C_k\}$ et leurs centres de gravité $\{g_1, g_2,...,g_k\}$;
1. Assigner aléatoirement k objets comme étant les centres initiaux ;
2. Pour chaque $x_{i_{(i:1,n)}}$ de l'ensemble X, calculer la distance entre chaque x_i et tous les centres initiaux puis assigner x_i à la classe la plus proche ;
3. Mettre à jour les nouveaux centres de gravité g_j associés aux nouveaux objets ;
4. Répéter les étapes (1), (2) et (3) jusqu'à ce que les centres de gravité ne changent plus.

Le K-Moyennes termine lorsqu'il n'y a plus de changement dans les composants de chaque classe, c'est-à-dire que les objets de la même classe restent dans leur classe. De plus, nous pouvons arrêter l'algorithme s'il n'y a pas vraiment une diminution significative de la somme des erreurs au carré.

La complexité en temps de calcul de l'algorithme K-Moyennes est calculée en fonction des itérations effectuées. Elle est de l'ordre $O(n \cdot k \cdot i)$, où n représente le nombre des objets à classer, k représente le nombre de classes et i représente le nombre des itérations effectuées jusqu'à la convergence. L'avantage majeur du K-Moyennes est qu'il est applicable aux bases de données de grande taille sans aucun effet négatif sur l'espace mémoire et le temps de calcul. Malgré sa simplicité, le K-Moyennes a des inconvénients. La spécification *a priori* du nombre de classes est un inconvénient puisque dans la plupart des

cas, l'utilisateur ne détient pas cette information. En outre, la classification finale obtenue par le K-Moyennes dépend du choix des centres initiaux, ce qui implique que l'algorithme converge dans la plupart des cas vers une solution locale qui n'est pas optimale.

Algorithme K-Médoïdes

Afin d'améliorer la performance du K-Moyennes, un autre algorithme de classification appelé K-Médoïdes a été développé [51]. Dans K-Médoïdes, les médoïdes sont utilisés au lieu des centroïdes. L'inconvénient majeur de cet algorithme est sa complexité qui est de l'ordre de $O(i \cdot k \cdot (n-k)^2)$, où i représente le nombre d'itérations pour assigner tous les objets à leurs classes, k représente le nombre de classes et n représente le nombre d'objets.

Il existe d'autres extensions du K-Moyennes. Nous citons à titre d'exemple l'algorithme Bisecting K-Moyennes [83]. Pour obtenir k classes, l'algorithme Bisecting K-Moyennes regroupe en premier lieu tous les objets dans une seule classe. Ensuite, l'algorithme sélectionne une classe pour la subdiviser en deux classes en appliquant le K-Moyennes de base. La sélection de la classe à subdiviser à chaque itération dépend de plusieurs critères : la classe qui contient le plus grand nombre d'objets et la somme des erreurs au carré. Lorsque le nombre de classes k est obtenu, l'algorithme s'arrête.

1.4.3 Les classificateurs utilisés dans la catégorisation des textes

Dans la pratique, plusieurs types de classificateurs ont été mis en place. Parmi les classificateurs existants, nous distinguons les classificateurs probabilistes qui utilisent l'ensemble d'apprentissage pour estimer les paramètres de la distribution de probabilité des termes par rapport aux catégories. Dans cette famille, nous citons entre autres le classificateur bayésien naïf. Il existe aussi une famille de classificateurs basés sur l'exemple. Dans cette famille, les nouveaux documents à classer font l'objet de comparaison avec l'ensemble d'apprentissage. L'algorithme K-NN est le plus connu de cette famille. Nous trouvons aussi des classificateurs linéaires. Dans cette famille, le document est représenté par un vecteur de termes pondérés. Parmi les algorithmes de cette famille, nous citons le classificateur SVM qui permet de séparer l'hyperplan en deux parties, en trouvant l'hyperplan optimal qui sépare les deux parties. Dans ce qui suit, nous exposons en détails, pour chaque famille, un algorithme qui lui correspond.

Classificateur bayésien naïf

Ce classificateur se base sur le théorème de Bayes qui consiste à calculer les probabilités conditionnelles d'une cause sachant la présence d'un effet. Son principe dans la catégorisation du texte est de calculer les probabilités qu'un nouveau document appartient à une catégorie sachant la proportion des documents d'apprentissage appartenant à cette catégorie. En outre, il permet de calculer la probabilité qu'un mot soit présent dans un texte sachant que ce texte appartient à telle catégorie. La probabilité à estimer est :

$$p(c_j/d_i) = \frac{p(a_1, a_2, ..., a_n/c_j)p(c_j)}{p(a_1, a_2, ..., a_n)} \tag{1.23}$$

où c_j représente la catégorie, a_i représente le terme et d_i représente le document.

Le calcul de $p(a_i/c_j)$ dépend du modèle de génération des exemples. Dans le cas de la catégorisation de documents, les modèles les plus utilisés sont le modèle multivarié de Bernoulli et le modèle multinomial [57].

Algorithme K plus proches voisins (K-NN)

L'algorithme K-NN [19] est parmi les classificateurs les plus utilisés dans le domaine de la catégorisation des documents. Le K-NN consiste à assigner un nouveau document au K plus proche voisin. Les textes sont représentés sous forme vectorielle. Le K-NN calcule la similarité entre les documents en utilisant la distance euclidienne entre les composants des vecteurs.

$$dist(d_i, d_j) = \sqrt{\sum_{t \in T}(p_t(d_i) - p_t(d_j))^2} \tag{1.24}$$

où T représente l'ensemble des attributs, $p_t(d_i)$ représente le poids du terme t dans le document d_i et $p_t(d_j)$ représente le poids du terme t dans le document d_j.

Plusieurs travaux ont pour objectif l'amélioration de l'algorithmes K-NN. Nous citons à titre d'exemple les travaux de Baoli *et al.* [9] qui ont permis d'optimiser la valeur de k pour chaque catégorie. Han *et al.* [38] ont modifié la mesure de similarité en attribuant un poids à chaque terme par validation croisée. Malgré ces recherches, la version originale de l'algorithme de base de K-NN demeure la plus utilisée dans le domaine de la catégorisation de documents.

1.5 Approches spécifiques de classification

Dans la pratique, il existe d'autres approches spécifiques que les algorithmes hiérarchiques et les algorithmes par partition. Ces approches sont développées spécialement pour des bases de données spécifiques. Elles incluent des méthodes pour obtenir des formes arbitraires, des méthodes fondées sur la densité et des méthodes fondées sur un quadrillage de l'espace de données. Dans cette section, nous détaillons ces méthodes spécifiques en démontrant leurs avantages et leurs inconvénients.

1.5.1 Méthodes pour obtenir des formes arbitraires

Pour faire face aux problèmes de la difficulté de la détection des classes de formes arbitraires dans les bases de données volumineuses, de nouveaux algorithmes ont été développés [35, 36, 49, 102]. Dans cette partie, nous décrivons quelques algorithmes qui appartiennent à cette famille.

L'algorithme CURE (*«Clustering Using REpresentatives»*) [35], a été développé pour obtenir des classes de formes arbitraires d'une part, et pour faire face aux problèmes de la complexité en temps de calcul et en espace mémoire des algorithmes hiérarchiques d'autre part. Dans cet algorithme, un ensemble de données représentatif de taille p de l'ensemble initial n est utilisé pour réduire en même temps la complexité en temps de calcul ainsi que l'espace mémoire. Ensuite chaque classe est représentée par un nombre d'individus $(\frac{p}{k})$ où k est le nombre de classes fixé par l'utilisateur. Les classes ayant un nombre faible d'objets sont éliminées. Les individus représentatifs de chaque classe sont rapprochés de leur centre par un facteur $\alpha : 0 < \alpha < 1$, en utilisant la notion de *shrinking*[5] [80]. La méthode CURE fusionne les deux classes dont les individus représentatifs sont les plus proches. Sa complexité en temps de calcul est de $O(m^2 \cdot log\ m)$ où m représente le nombre de données de l'échantillon extrait de l'échantillon total.

L'algorithme ROCK (*«Robust Clustering for Categorical Data»*) [36] a été développé spécialement pour la classification des données catégorielles. Il est basé sur le concept de *lien*. ROCK utilise les notions suivantes :

5. La technique de *shrinking* permet de déplacer les objets d'une classe vers leur centre pour obtenir une densité plus grande.

1. Deux objets p et q sont voisins si la similarité dépasse un certain seuil θ fixé par l'utilisateur : $sim(p,q) \geq \theta$;
2. Le lien entre deux objets $link(p,q)$ est le nombre de voisins communs semblables simultanémemt aux deux objets p et q ;
3. Le lien entre deux classes C_1 et C_2 est donné par tous les liens qui existent entre leurs objets : $link(C_1,C_2) = \sum_{p \in C_1} \sum_{q \in C_2} link(p,q)$;
4. Le nombre de liens dans une classe C est donné par : $n^{1+2f(\theta)}$, où $f(\theta) = \frac{1-\theta}{1+\theta}$ et θ représente le seuil de similarité, fixé par l'utilisateur ;
5. La mesure d'aggrégation de deux classes est définie comme suit :

$$Agg(C_1,C_2) = \frac{link(C_1,C_2)}{(n_1+n_2)^{1+2f(\theta)} - n_1^{1+2f(\theta)} - n_2^{1+2f(\theta)}} \tag{1.25}$$

L'algorithme ROCK fusionne les deux classes ayant le nombre de voisins le plus grand. Sa complexité en temps de calcul est de $O(n \cdot m_m \cdot m_a \cdot log\ n)$ où n représente le nombre total des objets, m_m et m_a représentent le nombre maximum et le nombre moyen de voisins d'un objet respectivement.

L'algorithme BIRCH (*«Balanced Iterative Reducing and Clustering using Hierarchies»*) [102] est basé sur le principe CF (*Clustering Feature*) et l'arbre CF, en regroupant les informations des objets de la classe dans un triplet : $CF = (n, LS, SS)$, où n est le nombre des objets dans la classe, LS défini par $\sum_{x_i \in C} x_i$ est la somme linéaire dans la classe C et SS est la somme quadratique $\sum_{x_i \in C} x_i^2$.

L'algorithme CHAMELEON [49] permet d'estimer la densité intra-classe et inter-classes à partir du graphe du K-plus proche voisin. Son principe se résume en deux phases :

1. Construire des sous-classes de petite taille à partir d'un graphe de partitionnement ;
2. Une agrégation est appliquée en fonction de la proximité relatives ($PR(C_1,C_2)$) et l'inter-connectivité ($IR(C_1,C_2)$). La proximité et l'inter-connectivité relatives sont définies respectivement par les formules :

$$PR(C_1, C_2) = \frac{(\sum_{x_{1i} \in C_1} x_{1i} + \sum_{x_{2i} \in C_2} x_{2i}) \cdot Dist(C_1, C_2)}{\sum_{x_{1i} \in C_1} x_{1i} \cdot Dist(C_1) + \sum_{x_{2i} \in C_2} x_{2i} \cdot Dist(C_2)} \tag{1.26}$$

où $Dist(C_1, C_2)$ représente la distance moyenne des poids des arrêtes reliant les deux classes C_1 et C_2 ;

$$IR(C_1, C_2) = \frac{2|EC(C_1, C_2)|}{|EC(C_1)| \cdot |EC(C_2)|} \tag{1.27}$$

où $|EC(C_1, C_2)|$ représente l'ensemble des arrêtes qui relient C_1 et C_2 et $E(C)$ représente le plus petit ensemble d'arrêtes qui partitionne C en deux classes.

Sa complexité en temps de calcul est de $O(n^3)$. L'inconvénient majeur de l'algorithme CHAMELEON est qu'il est sensible aux données aberrantes.

1.5.2 Méthodes fondées sur la densité

Les algorithmes basés sur la densité consistent à trouver des régions homogènes de haute densité entourées par un nombre m d'objets faibles, en respectant les deux paramètres :

Eps : Rayon maximum de voisinage.

MinPts : Nombre minimum de points dans le voisinage Eps d'un point.

Le voisinage d'un point est défini par : $V_{Eps}(p) : \{q \in D / dist(p, q) \leq Eps\}$. Les classes obtenues sont les régions de forte densité, séparées par des régions de faible densité. La figure 1.8 illustre un exemple de classification basée sur la densité.

Dans la pratique, DBSCAN (*«Density-Based Spacial Clustering of Applications with Noise»*) [28] est la méthode la plus connue qui est basée sur la densité. L'algorithme DBSCAN commence par sélectionner aléatoirement un point p. Ensuite, il rassemble dans une classe C tous les points qui vérifient le critère de la densité (**Eps** et **MinPts**) et vérifie si p représente un noyau par rapport à l'ensemble des points. Si c'est le cas, C est une classe, sinon un autre point est choisi de l'ensemble de données, en répétant le même processus. Sa complexité en temps de calcul est de $O(m \cdot log\ m)$ où m représente le nombre de points. L'un des avantages majeurs du DBSCAN est qu'il peut découvrir des classes de formes

diverses et isoler les bruits (points atypiques). L'algorithme DBSCAN manipule les données de grandes dimensions. Il est utilisé souvent dans l'analyse des données satellitaires et les données géographiques. Il existe aussi d'autres méthodes dérivées de DBSCAN, à savoir GDBSCAN (*Generalized* DBSCAN) [75] et OPTICS (*«Ordering Points To Identify the Clustering Structure»*) [7].

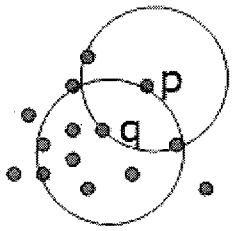


Figure 1.8 – Exemple de classification basée sur la densité

1.5.3 Méthodes fondées sur un quadrillage de l'espace de données

Parmi les algorithmes appartenant à cette famille, nous citons CLIQUE (*«Clustering In QUest»*) [5] et STING (*«Statistical Information Grid-based method»*) [94]. L'idée de base de ces algorithmes est de découper l'espace en segments. Ensuite, les segments denses proches sont fusionnés. La figure 1.9 illustre le principe de ces méthodes.

Dans la pratique, il existe plusieurs algorithmes qui font partie des méthodes fondées sur un quadrillage de l'espace de données. Nous citons à titre d'exemple les algorithmes WaveCluster [79], CACTUS (*«Clustering Categorical Data Using Summaries»*) [30] et PROCLUS [3].

L'inconvénient majeur de ces algorithmes est la spécification de la taille des cellules. Quand les cellules sont petites, ils conduisent à une estimation bruitée de la densité et quand les cellules sont grandes, ils conduisent à une estimation faible de la densité [15].

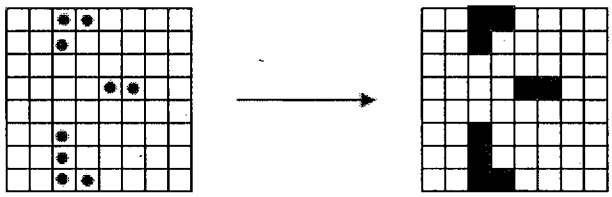


Figure 1.9 – Classification fondée sur quadrillage de l'espace de données

1.6 Classification floue

La théorie des ensembles flous a été introduite par Zadeh [98]. Un ensemble flou est un ensemble dont les bords ne sont pas définis [29]. Le degré d'appartenance d'un individu à un ensemble varie dans l'intervalle $[0, 1]$. À l'inverse de la classification conventionnelle, dans la classification floue l'objet x_i peut appartenir à plusieurs classes C_j en même temps, en spécifiant un degré d'appartenance μ_{ij} qui varie entre 0 et 1. Le concept flou est utilisé pour représenter les connaissances imprécises. Cela correspond à des situations où il est difficile de décider d'une manière précise l'appartenance d'un objet à une classe spécifique.

1.6.1 Principe de la classification floue

Plusieurs approches de la classification classique permettent d'affecter un objet à une seule classe. À l'inverse de ces approches, la classification floue utilise le degré d'appartenance $\mu_{ij} \in [0, 1]$ pour affecter un objet à une classe.

Supposons l'ensemble $X = \{x_1, x_2, ..., x_n\}$ dans un espace $\mathbf{R^d}$, où n représente le nombre d'observations et d représente le nombre d'attributs de chaque observation $x_i = (x_{i1}, x_{i2}, ..., x_{id})_{(i=1,n)}$. Supposons un ensemble de classes $C_j = \{C_1, C_2, ..., C_k\}_{(j=1,k)}$. Supposons que μ_{ij} est le degré d'appartenance d'un objet x_i à une classe C_j. Une classification est dite classification floue si et seulement si elle satisfait les deux conditions suivantes :

1. Pour chaque objet x_i :

$$\sum_{j=1}^{k} \mu_{ij} = 1 \tag{1.28}$$

2. Chaque classe C_j contient au minimum un objet :

$$0 < \sum_{i=1}^{n} \mu_{ij} < 1 \tag{1.29}$$

Caractéristiques d'un ensemble flou

Étant donné un sous-ensemble A de l'ensemble X, le support de A, noté $supp(A)$, est la partie de X sur laquelle la fonction d'appartenance de A n'est pas nulle :

$$supp(A) = \{x \in X / \mu_A(x) \neq 0\} \tag{1.30}$$

où $\mu_A(x)$ représente la fonction d'appartenance de l'élément x à l'ensemble A.

La hauteur de A, notée $h(A)$, est la plus grande valeur prise par la fonction d'appartenance :

$$h(A) = Sup_{x \in X}\{\mu_A(x)\} \tag{1.31}$$

Le noyau de A, noté $noy(A)$, est la partie de X sur laquelle la fonction d'appartenance de A est égal à 1.

$$noy(A) = \{x \in X / \mu_A(x) = 1\} \tag{1.32}$$

Nous disons que X est un ensemble fini si et seulement si :

$$|A| = \sum_{x \in X} \mu_A(x) = 1 \tag{1.33}$$

Les opérateurs flous

1. **L'opérateur négation (complément) :** Étant donné un sous-ensemble A de l'ensemble X. Le complément de A, noté $\bar{A}$, est défini par les éléments de X qui n'appartiennent pas à A. La fonction d'appartenance de l'ensemble complémentaire est donnée par :

$$\forall x \in X : \mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{1.34}$$

La figure 1.10 illustre un exemple d'appartenance du complément.

2. **L'opérateur ET (intersection) :** Étant donné deux sous-ensembles flous A et B, la fonction d'appartenance de l'intersection de A et B, utilisée par Zadeh, est réalisée par le minimum des fonctions d'appartenances μ_A et μ_B :

$$\forall x \in X : \mu_{A \cap B}(x) = min\{\mu_A(x), \mu_B(x)\} \tag{1.35}$$

La figure 1.11 illustre un exemple de l'opérateur **ET** de deux fonctions d'appartenances gaussiennes.

3. **L'opérateur OU (union) :** Étant donné deux sous-ensemble flous A et B, la fonction d'appartenance utilisée par Zadeh de l'union de A et B est réalisée par le maximum des fonctions d'appartenances μ_A et μ_B :

$$\forall x \in X : \mu_{A \cup B}(x) = max\{\mu_A(x), \mu_B(x)\} \tag{1.36}$$

La figure 1.12 illustre un exemple de l'opérateur **OU** de deux fonctions d'appartenances gaussiennes.

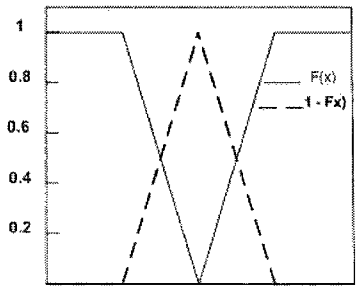


Figure 1.10 – Exemple de la fonction d'appartenance du complément

1.6.2 Algorithmes de classification floue

Dans la pratique, il existe plusieurs algorithmes qui sont basés sur le principe de la partition floue. Nous citons, à titre d'exemples, l'algorithme C-Moyennes floues [26] et l'algorithme Maximisation de la vraisemblance par une méthode floue (FMLE) [13]. Les

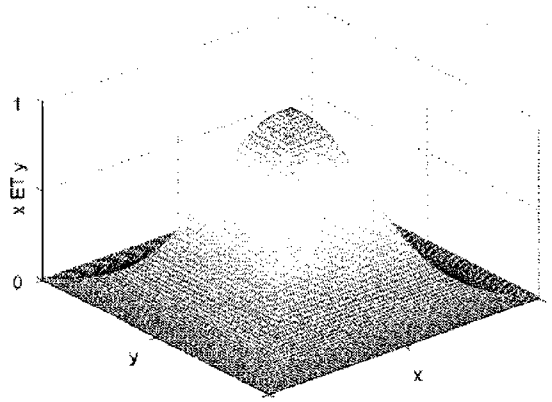


Figure 1.11 – Exemple de deux fonctions d'appartenance gaussiennes (opérateur **ET**)

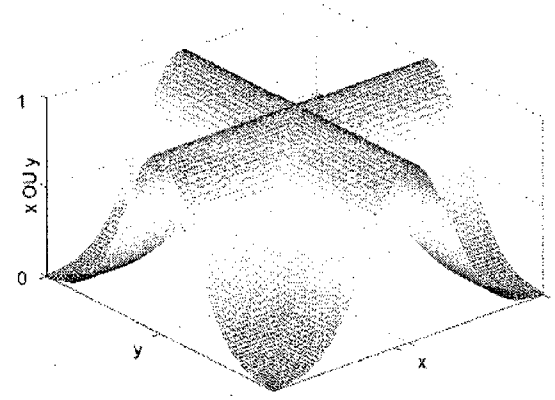


Figure 1.12 – Exemple de deux fonctions d'appartenance gaussiennes (opérateur **OU**)

algorithmes de classification floue sont caractérisés par la fonction du degré d'appartenance. Chaque objet appartient à plusieurs classes en même temps. Nous mettons un accent particulier sur l'algorithme C-Moyennes floues.

C-moyennes floues : FCM

L'algorithme C-moyennes floues est basé sur le même principe que celui du K-Moyennes. Les centres remplacent l'ensemble des objets. Le C-moyennes floues consiste à minimiser la fonction objective suivante :

$$SSE(C_1, C_2, ..., C_k) = \sum_{i=1}^{n} \sum_{j=1}^{k} \mu_{ij}^m dist(x_i - c_j)^2 \tag{1.37}$$

L'algorithme C-moyennes floues débute par l'initialisation du degré d'appartenance μ_{ij}. Ensuite, il calcule les centres de classes. Une mise à jour des centres est faite à chaque itération. L'algorithme s'arrête s'il n'y a plus aucun changement des centres. L'algorithme FCM nécessite la spécification de trois paramètres en entrée : ξ, le coefficient flou m et le nombre de classes k. L'algorithme C-moyennes floues est moins sensible au bruit grâce à la pondération par le degré d'appartenance. Il utilise généralement la distance euclidienne comme une mesure de similarité. Cette métrique limite les recherches pour les formes sphériques seulement. L'algorithme C-moyennes floues conserve les mêmes avantages et inconvénients du K-Moyennes. Le pseudo code de l'algorithme FCM est décrit dans (Algorithme 3).

Algorithme 3 Algorithme C-moyennes floues

Entrée : $X = \{x_1, x_2, ..., x_n\}$, k le nombre de classes, $C^{(0)}$ l'ensemble initial des classes, $dist$: la métrique de similarité, m : coefficient flou, ξ : seuil de convergence de l'algorithme.
Sortie : La matrice de degrés d'appartenance D et les centres de classes C.
1. $t = 0$;
2. Mise à jour des degrés d'appartenance :

$$\mu_{ij} = \frac{(1/dist(x_i, c_j)^2)^{\frac{1}{m-1}}}{\sum_{q=1}^{k} (1/dist(x_i, c_q)^2)^{\frac{1}{m-1}}} ;$$

3. Mise à jour des centres de classes :

$$c_j^{t+1} = \frac{\sum_{i=1}^{n} \mu_{ij}^m x_i}{\sum_{i=1}^{n} \mu_{ij}^m} ;$$

4. Test de convergence :
$|c_j^{t+1} - c_j^t = | < \xi$: Fin de l'algorithme.

1.7 Conclusion

Dans ce chapitre nous avons présenté les différents concepts de la classification. Nous avons détaillé le principe des algorithmes de classification hiérarchique, par partition et les algorithmes de la catégorisation du texte. Nous avons présenté les approches spécifiques de classification lorsqu'il s'agit de formes arbitraires. Un accent particulier a été mis sur la classification floue. Ces algorithmes souffrent généralement de problèmes de précision et de complexité en temps de calcul lorsqu'il s'agit d'un nombre élevé de données. Dans ce contexte, il existe d'autres techniques qui ont été développées pour faire face au problème de la classification. Nous citons à titre d'exemple le SVM que nous allons détailler dans le prochain chapitre.

Chapitre 2

Classification basée sur le SVM

2.1 Introduction

Étant donné la croissance rapide des bases de données, plusieurs recherches ont été effectuées dans le domaine de la classification afin de catégoriser, structurer et faciliter l'indexation des données. Dans la littérature, le problème de classification a été traité à l'aide de différentes techniques. Dans toutes les techniques développées, l'idée de base de la classification reste toujours la même : regrouper les objets qui partagent les mêmes caractéristiques dans une même classe en se basant sur une mesure de similarité. Le problème de la classification dépend essentiellement du nombre de classes à séparer. Nous distinguons deux types de problèmes de classification : les problèmes binaires et les problèmes multi-classes. Dans les problèmes multi-classes, le nombre de classes k est supérieur à deux. Plusieurs méthodes ont été développées pour résoudre ce genre de problème, à savoir les arbres de décision, les réseaux de neurones et la machine à vecteur de support (SVM). Le SVM est une technique d'apprentissage supervisé, développée par Vladimir Vapnik [91]. Vu la solidité de sa base théorique, le SVM est devenu un outil opérationnel important dans le domaine de la classification [40, 60]. Ce chapitre est subdivisé en trois parties. La première partie est consacrée à la présentation de l'aspect théorique du SVM. Dans la deuxième partie, nous présentons les différentes techniques introduisant le SVM pour résoudre les problèmes multi-classes. Dans la troisième partie, nous mettons un accent

particulier sur les travaux connexes du SVM pour la catégorisation des textes.

2.2 Base théorique du SVM

L'efficacité du SVM est due à sa base théorique solide qui constitue l'un de ses atouts. Le SVM s'articule autour de deux principes : le premier vise à minimiser l'erreur empirique de classification et le deuxième vise à trouver un hyper-plan optimal qui maximise la marge entre deux ou plusieurs classes. Le SVM trouve un hyper-plan qui permet de regrouper les données similaires dans une même classe et de séparer les données hétérogènes. La figure 2.1 illustre le principe du SVM dans lequel il existe une infinité d'hyper-plans séparant deux classes. L'idée de base est de déterminer ces hyper-plans optimums [86].

Dans la figure 2.1, (A) et (B) sont des hyper-plans séparant la classe positive et la classe négative avec des marges δ_1 et δ_2 respectivement. (B) représente l'hyper-plan optimal qui sépare ces deux classes, en préservant une marge maximale.

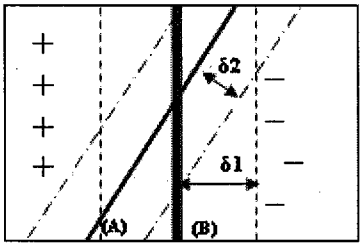


Figure 2.1 – Détermination d'un hyper-plan optimal

2.2.1 Discrimination linéaire

Définition

Supposons un ensemble de données $(x_i, y_i)_{(i=1,n)}$ où x_i correspond à l'ensemble de données dans l'espace et $y_i \in \{-1, +1\}$. L'idée est de trouver une fonction linéaire qui permet de séparer l'ensemble de données :

$$f(x) = wx + b \tag{2.1}$$

où w et b sont les paramètres du modèle.

Il est nécessaire de déterminer w et b qui satisfont les conditions (2.2) et (2.3) :

$$f(x) = wx_+ + b \tag{2.2}$$

où x_+ représente l'ensemble des données appartenant à la classe positive.

$$f(x) = wx_- + b \tag{2.3}$$

où x_- représente l'ensemble des données appartenant à la classe négative.

Marge de l'hyper-Plan

Un problème linéairement séparable signifie l'existence d'un séparateur linéaire capable de séparer les données en deux ou plusieurs classes dont la marge est maximale. Géométriquement, nous avons :

$$x_+ = x_- + \delta w \tag{2.4}$$

où δ représente la distance perpendiculaire entre une donnée de la classe positive et celle de la classe négative. En remplaçant x_+ de l'équation (2.4), dans l'équation (2.2), et en comparant avec l'équation (2.3), nous obtenons la marge qui sépare ces deux classes :

$$\delta = \frac{2}{\|w\|^2} \tag{2.5}$$

Donc, l'hyper-plan est optimal si et seulement si w est minimale. Le problème d'optimisation est exprimé par :

$$(P) \begin{cases} \min \frac{1}{2}\|w\|^2 \\ y_i(wx_i + b) \geq 1 : i \in \{1, ..., n\} : \forall x \in \mathbf{R^n} \end{cases} \tag{2.6}$$

Cela nous conduit à déterminer la marge δ. Cette marge est maximale si w est minimale. Puisque la fonction d'optimisation est quadratique avec des contraintes linéaires, le problème d'optimisation est convexe. La solution de ce genre de problèmes nécessite les multiplicateurs de Lagrange. La nouvelle fonction objective est :

$$L_p = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(wx_i + b) - 1] \tag{2.7}$$

où α_i sont les multiplicateurs de Lagrange.

La solution du problème d'optimisation exprimé par l'équation (2.7) est :

- $\frac{dL}{dw} = 0$:

$$f_1(x) = \sum_{i=1}^{n} \alpha_i y_i x_i \tag{2.8}$$

- $\frac{dL}{db} = 0$:

$$f_2(x) = \sum_{i=1}^{n} \alpha_i y_i \tag{2.9}$$

Puisque le problème (2.7) manipule les trois variables α_i, w et b en même temps, il peut être simplifié par la transformation du problème de Lagrange en un problème dual. La formulation du problème dual est donnée par :

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{ij}^{n} \alpha_i \alpha_j y_i y_j x_i x_j \tag{2.10}$$

2.2.2 Cas d'un problème linéaire avec des erreurs de classification

Dans certains cas, des erreurs de classification ξ_i surviennent lors de l'ajout de nouveaux enregistrements. Ce cas ressemble au cas linéaire présenté dans la section précédente. La seule différence entre les deux cas est l'existence de données mal classées (données bruitées). La figure 2.3 montre un cas linéaire avec des erreurs de classification.

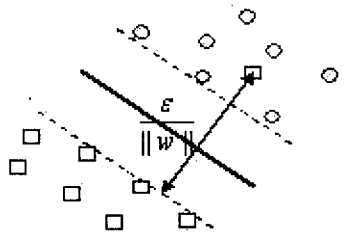


Figure 2.2 – Cas linéaire avec des erreurs de classification

Dans ce cas, la formulation du problème est la suivante :

$$wx_i + b \leq (1 - \xi_i) : y_i = -1 \tag{2.11}$$

$$\forall i, \xi_i > 0 \tag{2.12}$$

La fonction objective est exprimée comme suit :

$$f(w) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i)^k \tag{2.13}$$

où C et ξ_i représentent les paramètres de pénalités de mauvaise classification, spécifiés par l'utilisateur et k le nombre de classes à séparer.

Le problème d'optimisation est donné par :

$$L_p = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}(\xi_i) - \sum_{i=1}^{n}\alpha_i[y_i(wx_i + b) - 1 + \xi_i] - \sum_{i=1}^{n}\mu_i\xi_i \tag{2.14}$$

Son problème dual est le suivant :

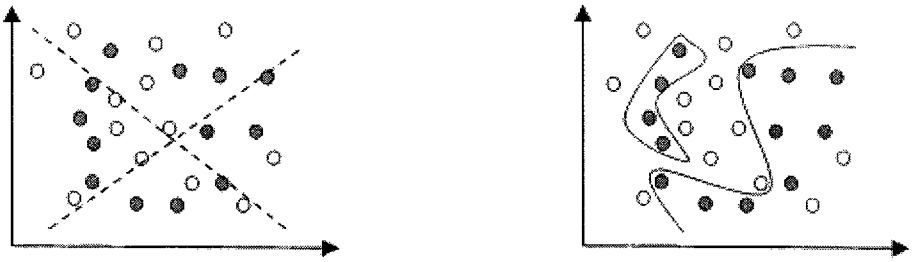
$$L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{ij}^{n}\alpha_i\alpha_j y_i y_j x_i x_j \tag{2.15}$$

Le problème (2.15) devient identique au problème linéaire exprimé par l'équation (2.10).

2.2.3 Discrimination non linéaire

Les deux cas présentés aux sections (2.2.1) et (2.2.2), nécessitent une fonction linéaire pour la séparation des données.

Dans cette section, nous décrivons le cas des données non séparables par une fonction linéaire et comment le SVM se comporte devant ce genre de problèmes. Le cas (a) de la figure 2.3 montre que le séparateur linéaire n'est pas le bon choix pour la séparation des données, par contre le cas (b) montre que la fonction non linéaire représente le meilleur séparateur des données. Dans le cas des données non linéairement séparables dans un espace initial $\mathbf{R^d}$, il est nécessaire de trouver un espace de dimension plus grande R^D qui rend la séparation des données assez facile : $\Phi(X) : \mathbf{R^d} \rightarrow \mathbf{R^D} : D > d$. Après la transformation (*mapping*) vers l'espace $\mathbf{R^D}$, nous obtenons : $wx + b = 0 \rightarrow w\Phi + b = 0$.



(a) Séparateur linéaire (b) Séparateur non linéaire

Figure 2.3 – Données non linéairement séparables

2.2.4 Les fonctions de noyaux

Les fonctions de noyaux interviennent lorsque le problème de classification est non linéaire. Nous utilisons une fonction non linéaire pour classer les données. Géométriquement, cela nous conduit à construire un hyper-plan séparant les deux classes positive et négative (figure 2.4). Un exemple d'un cas non séparable est décrit dans l'annexe B. Les fonctions de noyaux les plus utilisées sont :

1. Fonction polynomiale : $k(x, y) = (x \cdot y + 1)^d$ où d représente le degré choisi par l'utilisateur ;
2. Fonction radiale de base (RBF) : $k(x, y) = exp(-\gamma|x - y|^2)$;
3. Fonction de Neuron : $k(x, y) = tan\, h(ax \cdot y + b)$ où les paramètres a et b sont adaptés par l'utilisateur ;
4. Fonction gaussienne : $k(x, y) = exp^{\frac{||x-y||^2}{2\sigma^2}}$ où σ représente l'écart-type entre x et y.

Condition de Mercer

Peut-on utiliser n'importe quelle fonction de noyau ?

Les fonctions de noyaux $k(x, y)$ doivent satisfaire la condition de Mercer pour être utilisées. Pour toute fonction $g(x)$:

$$\int \int k(x, y)g(x)g(y)\, dxdy \geq 0 \tag{2.16}$$

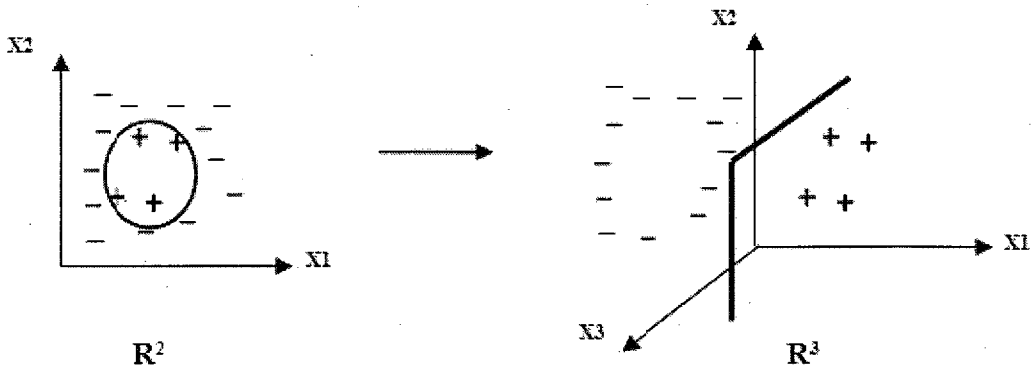


Figure 2.4 – Technique de transformation (*mapping*)

Si cette condition est vérifiée, le problème quadratique possède une solution. Supposons ainsi la matrice G, appelée la matrice de *Gram*, qui définit les similarités entre les éléments de l'ensemble d'apprentissage :

$$G = \begin{pmatrix} k_{11} & k_{12} & \dots & k_{1n} \\ k_{21} & k_{22} & \dots & k_{2n} \\ \vdots & \vdots & \vdots & \\ k_{n1} & k_{n2} & \dots & k_{1n} \end{pmatrix}$$

La fonction $k(x, y)$ est une fonction de noyau si $G = (k(x_i, y_j))_{i,j=1}^{n}$ est définie positive. Une matrice définie positive possède les trois propriétés suivantes :

1. positive : $k(x_i, x_i) > 0$;
2. symétrie : $k(x_i, y_j) = k(y_j, x_i)$;
3. inégalité de Cauchy-Shwartz : $|k(x_i, y_j)| \leq ||k(x_i)|| \cdot ||k(y_j)||$.

2.3 Les problèmes multi-classes basés sur le SVM

Les classificateurs SVM de base sont binaires [92] et leur extension au cas multi-classes reste un domaine de recherche ouvert [39, 61]. Cependant, pour résoudre les problèmes multi-classes, deux approches principales ont été développées. La première approche consiste à résoudre le problème d'optimisation lié aux problèmes multi-classes [6, 41, 85]. Les problèmes liés à cette approche sont assez complexes puisque cette approche

prend en considération tout l'ensemble de données, ce qui nécessite un temps énorme pour résoudre le problème d'optimisation. Malgré l'amélioration de cette approche par [82], le problème de complexité persiste toujours, vu le nombre élevé de données traitées. La deuxième approche consiste à introduire le SVM pour résoudre le problème multi-classes en convertissant le problème original en sous-problèmes binaires. Plusieurs résultats expérimentaux ont prouvé que la combinaison des SVM-binaires est efficace pour résoudre les problèmes multi-classes [17, 48, 81, 100]. Dans notre travail, nous nous intéressons à l'étude de la deuxième approche puisqu'elle s'apparente à la méthode que nous avons développée dans ce projet.

Ainsi, plusieurs méthodes basées sur le SVM ont été proposées pour résoudre ce genre de problèmes. Dans ce mémoire, nous détaillons quatre méthodes principales : «Un-contre-un» [41, 54, 100], «Un-contre-tous» [73, 81], Graphe Acyclique Orienté (DAG) [72] et la méthode hiérarchique descendante basée sur SVM (DHSVM) [11]. Les trois premières méthodes sont les plus répandues dans le domaine de la classification et sont largement utilisées. La quatrième méthode utilise une structure hiérarchique qui ressemble à notre méthode.

2.3.1 Méthode «Un-contre-un»

Étant donné k classes, dans la méthode «Un-contre-un» un seul SVM_{ij} est construit pour chaque paire de classes (i, j). Le nombre de SVM_{ij} requis pour un problème de k classes est $\frac{k(k-1)}{2}$. La figure 2.5 illustre le principe de «Un-contre-un».

Lors de la phase de test, l'échantillon est évalué par tous les SVM_{ij}. La décision est obtenue par un mécanisme de votes [1].

Formulation du problème : Étant donné k classes, supposons un ensemble de n données d'apprentissage (x_i, y_j) où $x_i \in \mathbf{R^n}$, $k > 2$ et $y_j \in \{1, ..., k\}$. Le problème d'optimisation lié à la construction du classificateur SVM_{ij} pour la séparation des deux classes C_i et C_j est défini comme suit :

1. La stratégie de vote la plus utilisée est Max-Wins

$$(P)\begin{cases}\min\limits_{w^i,b^i,\xi^i} \frac{1}{2}(w^{ij})^T w^{ij} + C\sum\limits_t \xi_t^{ij} \\ (w^{ij})^T\phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij} : y_j = 1 \\ (w^{ij})^T\phi(x_t) + b^{ij} \leq -1 + \xi_t^{ij} : y_j \neq 1 \\ \xi_t^{ij} \geq 0 : j = 1, ..., k.\end{cases} \tag{2.17}$$

Pour k classes, la phase d'apprentissage de la méthode «Un-contre-un» nécessite $(\frac{k(k-1)}{2})$ SVM_{ij}. Pour la phase de test, la stratégie Max-Wins (nommée la stratégie de vote) est utilisée.

- **Stratégie de vote :** La stratégie de vote est décrite comme suit :
  Supposons SVM_{ij} un classificateur pour les deux classes i et j et la fonction de décision est $f_{ij}(x) = Sgn(w_{ij} \cdot x + b_{ij})$.

$$Sgn(x) = \begin{cases}+1 : x > 0 \\ -1 : x \leq 0\end{cases} \tag{2.18}$$

$$x \in \begin{cases}i : f_{ij}(x) = 1 \\ j : f_{ij}(x) = -1\end{cases} \tag{2.19}$$

Donc le processus de la stratégie de Max-Wins pour les k SVM_{ij} est définie comme suit :

1. Pour l'ensemble de test x_i, calculer :

$$f_{ij}(x) = \sum_{j\neq i, j=1}^{k} Sgn(f_{ij}(x)) \tag{2.20}$$

2. Classer x dans la classe :

$$arg \max_{i:1,...,k} f_i(x) \tag{2.21}$$

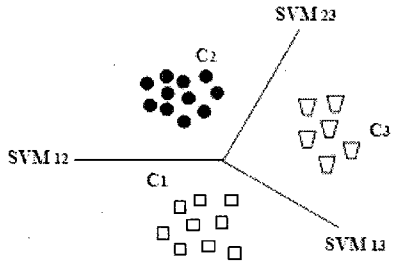


Figure 2.5 – Principe de la méthode «Un-contre-un»

2.3.2 Méthode «Un-contre-tous»

Contrairement à la méthode «Un-contre-un», la méthode «Un-contre-tous» consiste à séparer à chaque étape une seule classe par rapport au reste des classes en minimisant l'erreur de classification. La classe séparée est considérée comme une classe positive et le reste est considéré comme une classe négative. Nous répétons le processus de la classification sur le reste des objets en éliminant de l'ensemble de données les objets des classes positives déjà classées dans les étapes précédentes. La méthode «Un-contre-tous» nécessite k classificateurs SVM_i pour un problème de k classes. Lors de la phase de test, l'échantillon de test est évalué par les $SVM_{i,(i=1,n)}$ et le SVM_i qui montre la valeur de décision la plus élevée est choisi. La figure 2.6 illustre le principe de la méthode «Un-contre-tous».

Formulation du problème : Étant donné un ensemble de données (x_i, y_j) de taille n où $x_i \in \mathbf{R^n}$ et $y_j \in \{1, ..., k\}$ représente la classe de x_i. Le problème d'optimisation est :

$$(P) \begin{cases} \min\limits_{w^i, b^i, \xi^i} \dfrac{1}{2}(w^i)^T w^i + C \sum\limits_{j=1}^{l} \xi_j^i \\ (w^i)^T \phi(x_j) + b^i \geq 1 - \xi_j^i : y_j = 1 \\ (w^i)^T \phi(x_j) + b^i \leq -1 + \xi_j^i : y_j \neq 1 \\ \xi_j^i \geq 0 : j = 1, ..., k. \end{cases} \tag{2.22}$$

où la fonction ϕ représente la fonction de transformation (*mapping*) dans une dimension plus élevée et C représente le paramètre de pénalité de classification.

Après la résolution du problème (2.22), nous obtenons k fonctions de décision :

$$\begin{cases} (w^1)^T\phi(x) + b^1, \\ . \\ . \\ (w^k)^T\phi(x) + b^k. \end{cases} \tag{2.23}$$

La classe de l'élément x est la plus grande valeur de la fonction de décision :

$$Class(x) = \arg \max_{(i=1,\ldots,k)} ((w^i)^T\phi(x) + b^i). \tag{2.24}$$

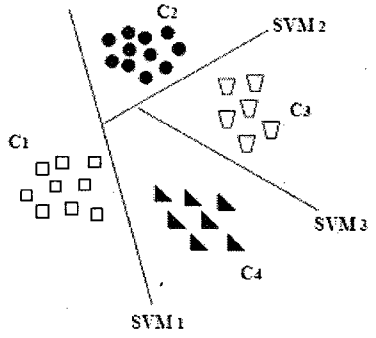


Figure 2.6 – Principe de la méthode «Un-contre-tous»

2.3.3 Méthode «Graphe Acyclique Orienté» (DAGSVM)

La méthode DAGSVM est basée sur la notion du graphe acyclique. Un graphe acyclique orienté est un graphe qui ne possède pas de cycle et dont les arcs sont orientés. Dans le graphe, les feuilles représentent les classes à séparer. Chaque nœud contient 0 ou 2 arcs en sortie. Dans la phase d'apprentissage, la méthode DAGSVM nécessite $(\frac{k(k-1)}{2})$ SVM_{ij}. Cette méthode suit le principe des opérations sur les listes. À chaque niveau de la hiérarchie, une classe est éliminée de la liste. La figure 2.7 illustre le principe de la méthode DAGSVM.

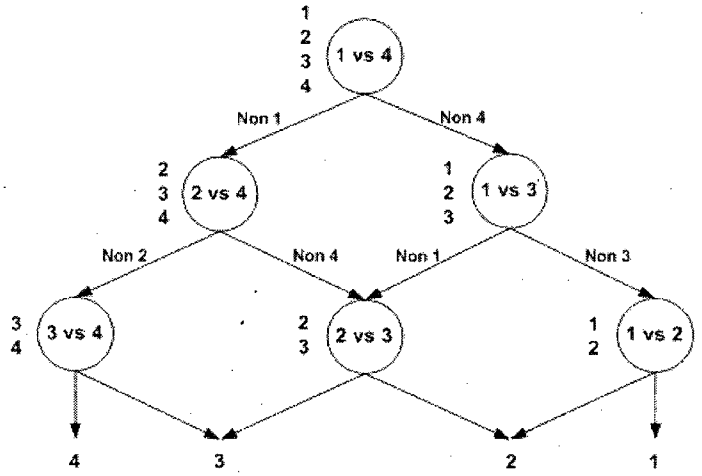


Figure 2.7 – Principe de la méthode DAGSVM

2.3.4 Méthode hiérarchique descendante DHSVM

Dans [11], une nouvelle méthode intitulée DHSVM pour traiter le problème multi-classes a été développée. La méthode DHSVM consiste à subdiviser le problème original en sous problèmes binaires afin de faciliter sa résolution. Lors de la phase de test, un nombre optimal de SVM est obtenu à partir de la racine jusqu'aux feuilles. DHSVM nécessite $(k-1)$ SVM_i pour résoudre un problème de k classes. Elle consiste en premier lieu à réduire le nombre de données, en calculant les classes représentatives de l'ensemble de données. Ensuite, le SVM est introduit à chaque niveau pour établir une classification binaire. La figure 2.8 illustre le principe de la méthode DHSVM. L'inconvénient majeur de cette méthode est que le choix des données d'apprentissage s'effectue aléatoirement sans l'utilisation d'une métrique. Lors de la phase d'apprentissage, plusieurs itérations sont effectuées pour déterminer l'ensemble d'apprentissage qui réduit le temps de classification. Ce choix diminue la performance du classificateur SVM.

Une comparaison des méthodes : «Un-contre-un», «Un-contre-tous» et DAGSVM, a été réalisée par Hsu *et al.* [41]. Les résultats expérimentaux ont montré que la méthode «Un-contre-un» et DAGSVM sont plus performantes que «Un-contre-tous». Les trois méthodes en question souffrent du problème de complexité en temps de calcul. Nous constatons aussi, lors de la phase d'apprentissage, qu'il n'existe pas de définition d'une métrique

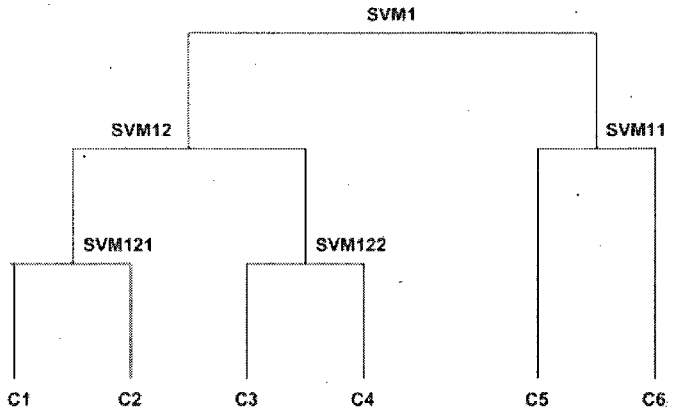


Figure 2.8 – Principe de la méthode DHSVM

de similarité lors du choix de la classe parmi les n classes. Le choix de la classe est basé sur la minimisation des erreurs de classification. L'itération pour la sélection de la classe nécessite un temps d'exécution très élevé.

Une autre comparaison a été réalisée dans [11] entre «Un-contre-un», «Un-contre-tous» et DHSVM. Les travaux de [11] ont démontré la performance de DHSVM en matière de précision et de temps d'apprentissage *versus* «Un-contre-un» et «Un-contre-tous».

En outre, il existe dans la littérature d'autres méthodes qui ont été développées pour résoudre le problème multi-classes. Zhang *et al.* [101] ont proposé une architecture des arbres binaires dans le but de convertir le problème original en sous-problèmes binaires. Pour regrouper les objets homogènes dans une seule classe, les auteurs ont introduit le concept de coefficient de ressemblance à chaque niveau de l'arbre. Le calcul du coefficient de ressemblance de deux classes permet de spécifier leur degré de corrélation. Li *et al.* [59] ont développé un algorithme qui permet de regrouper les données. L'algorithme nommé C-SVM est basé sur le DBSCAN (*«Density-Based Spatial Clustering of Applications with Noise»*). L'algorithme C-SVM consiste à déterminer les classes homogènes et à supprimer les données qui n'appartiennent à aucune classe. L'algorithme C-SVM s'articule autour de la notion de pureté des classes. Lorsque la pureté est élevée, la séparation des classes devient assez facile. Dans cette technique, chaque classe est caractérisée par son rayon qui est déterminé par le théorème de la surface de l'hyper-sphère. Dans leurs travaux, les auteurs ont pu diminuer le temps d'apprentissage du SVM.

2.4 Problème de la catégorisation des textes basée sur SVM

Devant la croissance rapide des données textuelles dans les bases de données, on a constaté que la classification manuelle des documents n'est pas en mesure de suivre cette croissance. La mise en place des méthodes automatiques pour structurer et classifier ces données devient une tâche indispensable [56]. La catégorisation automatique des documents consiste à regrouper les documents similaires dans la même classe en se basant sur leur contenu contextuel. Imaginons la présence d'une base de données volumineuse de documents qui seraient plus facilement accessibles si les documents étaient répartis dans un ensemble de catégories en fonction de leur sujet. Évidemment, le traitement manuel est faisable. Mais la tâche s'avère compliquée lorsqu'on est en présence des milliers de documents. Jusqu'à présent, la catégorisation des documents demeure parmi les préoccupations majeures dans le domaine de la classification [10]. Plusieurs recherches ont été effectuées dans le but de concevoir une meilleure représentation des textes [78, 90].

Plusieurs techniques ont été proposées à savoir les arbres de décision [95], le plus proche voisin [97], le modèle bayésien [53, 58], le modèle de régression [77] et le SVM [45, 78] qui a démontré une performance élevée dans ce domaine [39, 47, 62, 63].

Dans notre travail, nous nous intéressons au problème de la catégorisation des documents basée sur le SVM. Cette problématique est principalement liée au domaine de la classification hiérarchique. La plupart des recherches récentes introduisent la structure hiérarchique pour résoudre le problème de la catégorisation des documents [25, 39, 44, 45, 53, 62, 71, 93].

Dumais *et al.* [25] ont utilisé la classification hiérarchique pour classifier les collections hétérogènes du contenu du Web. Ils ont introduit le SVM à chaque niveau de la hiérarchie pour distinguer les catégories. Dans [39], une classification hiérarchique basée sur le SVM a été proposée. La méthode proposée par Hao *et al.* [39] consiste à subdiviser le problème original en sous-problèmes binaires afin de faciliter la discrimination entre un grand nombre de catégories. Les résultats obtenus dans [39] ont démontré une performance plus élevée que celle du SVM de base. Joachims [44] a proposé le classificateur Tf-Idf, en introduisant l'aspect probabiliste qui est donné par :

$$P(d|c_j) = \sum_{w \in (d \cap c_j)} \frac{P(w|c_j) \cdot P(c_j)}{\sum_{c \in C} P(w \backslash c_j) \cdot P(c)} \cdot P(w \backslash d) \tag{2.25}$$

où c et c_j représente les catégories, $P(d \backslash c_j)$ représente la probabilité que le document d appartient à la catégorie c_j et $P(w \backslash d)$ représente la probabilité que l'attribut w appartient au document d.

Dans les travaux de Joachims [45], on a découvert les avantages de l'utilisation du SVM pour le problème de la catégorisation des documents par rapport aux méthodes standards : Bayes, Rocchio, C4.5 et K-NN. Les résultats dans [45] ont démontré une performance très élevée. Koller *et al.* [53] ont utilisé la structure hiérarchique en conjonction avec le naïve Bayes naïf. Ils ont constaté que la sélection d'un nombre minimum d'attributs permet la discrimination entre les catégories à chaque niveau de la hiérarchie. Cette structure hiérarchique a démontré une performance très élevée. Liu *et al.* [62] ont évalué le SVM dans le Web. Dans leurs travaux [62], ils ont développé un nouveau système pour la catégorisation des textes. Les auteurs ont utilisé la méthode «Un-contre-tous» pour la séparation des catégories à chaque niveau de la hiérarchie. Les résultats obtenus ont démontré l'avantage de la structure hiérarchique pour la catégorisation des textes. Peng *et al.* [71] ont proposé un nouvel algorithme construisant automatiquement une structure hiérarchique. Le système de classification hiérarchique proposé permet d'ajouter de nouvelles catégories selon le besoin de l'utilisateur, d'organiser les pages web dans une structure d'arbre et de classifier les pages web sous une forme hiérarchique. Wang *et al.* [93] ont proposé un système de catégorisation de documents pour résoudre le problème multi-classes. Le système proposé contient les modules suivants : (i) module de prétraitement des données textuelles, et (ii) module de classification. Dans [93], la logique floue a été introduite au niveau du module de classification.

2.5 Conclusion

Ce chapitre nous a permis de présenter les principales méthodes utilisées dans la résolution du problème de classification multi-classes pour les types de données numériques et textuelles. Ces méthodes souffrent principalement du problème de complexité en temps de calcul et de performance lorsqu'il s'agit d'un grand nombre de données à traiter. La

plupart des méthodes de classification que nous avons étudiées dans ce chapitre utilisent un ensemble de données aléatoire lors de la phase d'apprentissage, ce qui influe, sans aucun doute, sur le temps de traitement. En outre, nous avons constaté, en consultant les travaux connexes de la catégorisation de texte, que la structure hiérarchique, en raison de sa simplicité, est considérée comme la structure la plus utilisée pour résoudre les problèmes multi-classes. Aussi, afin de remédier à ces problèmes, nous proposons une nouvelle méthode de classification qui introduit une structure hiérarchique floue basée sur le SVM. La nouvelle méthode développée, SVM-CHF, ainsi que sa version adaptée SVM-CHF-Text pour la catégorisation des textes sont présentées en détails dans le chapitre qui suit.

Chapitre 3

La méthode SVM-CHF et sa version adaptée SVM-CHF-Text

Le classificateur SVM est conçu à la base pour résoudre les problèmes binaires et son extension pour résoudre le problème multi-classes reste un domaine de recherche ouvert. Aussi, pour traiter le problème multi-classes, nous proposons une nouvelle méthode pour construire dynamiquement une structure hiérarchique floue à partir des données d'apprentissage. Notre méthode est basée sur trois principaux concepts : la classification hiérarchique, la théorie des ensembles flous et le SVM. Premièrement, la classification hiérarchique consiste à faire ressortir les relations entres les classes de la racine jusqu'aux feuilles. La logique floue consiste à définir les relations floues entre les objets, en trouvant les chemins possibles qui existent entre les objets. Nous appliquons la technique de fermeture transitive pour découvrir les relations floues entre les objets. Finalement, une fois les données préparées, nous introduisons le SVM à chaque niveau de la hiérarchie pour accomplir la tâche de classification. Le SVM est utilisé pour subdiviser le problème original en sous-problèmes binaires. Tout au long de la hiérarchie, de la racine jusqu'aux feuilles, nous combinons les SVM binaires obtenus pour résoudre le problème original. En outre, nous combinons les techniques de classification hiérarchique, de classification par partition, de la logique floue et le SVM pour améliorer la qualité des résultats. L'idée de notre méthode consiste à simplifier la phase d'apprentissage du SVM en rendant l'ensemble des données le plus possible séparable à l'aide d'une spécification de distance floue entre les objets.

En outre, nous avons adapté notre méthode pour traiter le problème de la catégorisation des textes. Nous introduisons la technique d'indexation sémantique latente (LSI) dans la phase de prétraitement pour réduire la dimension de l'espace. Nous testons, à travers ce travail, l'impact de la combinaison de la logique floue avec le classificateur SVM sur le problème de la catégorisation des textes.

Afin de simplifier la lecture de ce chapitre, nous l'avons subdivisé en deux grandes parties. La première partie est consacrée à la présentation de la méthode SVM-CHF de base que nous avons développé pour traiter les données numériques. Dans la deuxième partie, nous présentons sa version adaptée SVM-CHF-Text pour traiter le problème de catégorisation des textes.

3.1 Présentation de la méthode SVM-CHF

3.1.1 Processus

Dans cette section nous détaillons notre méthode intitulée classification hiérarchique floue basée sur le SVM (SVM-CHF) pour traiter les données numériques. La méthode SVM-CHF offre une solution alternative aux quatre méthodes existantes : «Un-contre-un», «Un-contre-tous», DAGSVM et DHSVM. Elle est basée sur une technique de classification floue que nous avons développée pour la réutilisation des spécifications logicielles [70]. La méthode SVM-CHF se résume en trois étapes : (1) la compression des données d'apprentissage par l'algorithme K-Moyennes ; (2) la classification hiérarchique floue ; (3) l'introduction du SVM à chaque niveau de la hiérarchie. L'avantage de notre méthode est que l'ensemble d'apprentissage obtenu *a priori* par la technique de la fermeture transitive Min-Max assure la bonne séparation entre les classes positives et les classes négatives ce qui permet une meilleure classification.

1. **La compression des données d'apprentissage :** Plusieurs recherches appliquent la technique de compression de données avant de procéder à la classification [96]. Dans notre méthode nous avons appliqué l'algorithme K-Moyennes pour la compression des données d'apprentissage. L'algorithme K-Moyennes permet de regrouper les données homogènes partageant des caractéristiques similaires dans la même classe. Donc, au lieu de traiter tout l'ensemble de données, il est suffisant de traiter unique-

ment les centres des classes. Cette technique permet de réduire l'ensemble initial des données d'une part, et d'autre part, de réduire le temps de traitement. Pour ce faire, nous varions la valeur de k en utilisant la formule (1.21) qui permet de regrouper l'ensemble des données en des classes. La valeur de k retenue est celle qui minimise l'erreur de classification ξ exprimée par l'équation (1.22).

2. **La classification hiérarchique floue :** Notre méthode est basée sur le principe d'équivalence des classes [50, 87] qui consiste à regrouper les objets similaires dans la même classe. Nous utilisons la relation floue pour mesurer la similarité entre les classes. Les classes sont structurées dans une hiérarchie floue qui permet de déterminer et de faire ressortir les objets qui sont similaires. Dans cette phase, nous introduisons la notion de mesure de similarité, les opérations sur l'ensemble flou, la hiérarchie et la fermeture transitive de la relation floue. Pour mieux comprendre ces aspects, nous allons les expliquer dans ce qui suit.

   – **La mesure de similarité**

   Supposons K un univers de discours et A un sous-ensemble de K : $A \subset K$. Supposons de plus, $\mu_A(x)$ la fonction caractéristique dont la valeur indique l'appartenance ou non de l'élément x à A :

$$\mu_A(x) = \begin{cases} \mu_A(x) = 1 & \text{si} \quad x \in A \\ \mu_A(x) = 0 & \text{sinon.} \end{cases} \tag{3.1}$$

   La fonction d'appartenance $\mu_A(x)$ prend ses valeurs dans l'intervalle $[0, 1]$.

   La mesure de distance entre x et y, notée $d(x, y)$, a été utilisée dans plusieurs contextes comme une mesure de similarité entre les objets. La fonction d est définie comme une distance dans K, s'il existe une opération, par exemple $(*)$, qui est, $\forall\, x, y, z \in K$:

   (a) réflexive :

$$\forall x \in K : d(x, x) = 0 \tag{3.2}$$

   (b) non négative :

$$\forall x, y \in K : d(x, y) \geq 0 \tag{3.3}$$

   (c) symétrique :

$$\forall x, y \in K : d(x, y) = d(y, x) \tag{3.4}$$

(d) transitive :

$$\forall x, y, z \in K : d(x, z) \leq d(x, y) * d(y, z) \tag{3.5}$$

Dans ce travail, nous définissons la distance relative de Hamming δ pour mesurer la similarité entre les classes obtenues dans la phase (1). Cette distance est définie comme suit :

$$\delta(C_i, C_j) = \frac{1}{n} \times d(C_i, C_j) = \frac{1}{n} \sum_{i=1}^{n} |\mu_{C_i}(x_i) - \mu_{C_j}(x_i)| \tag{3.6}$$

où n représente le nombre des attributs x_i de chaque classe ; $d(C_i, C_j)$ représente la distance de hamming entre chaque paire de classes C_i et C_j et $\mu_{C_i}(x_i)$, $\mu_{C_j}(x_i) \in [0, 1], \forall\, i = 1, .., n$. De plus :

$$0 \leq \delta(C_i, C_j) \leq 1. \tag{3.7}$$

– Les opérations sur l'ensemble flou

L'ensemble flou d'un univers donné est défini comme étant une application : $\mu_A(x) : K \rightarrow [0, 1]$. La valeur zéro représente la non appartenance complète alors que la valeur un représente l'appartenance complète. Étant donné K un ensemble et $\mu_A(x) = [0, 1]$ ses degrés d'appartenance, les opérations de base sur les ensembles flous A et B de K sont données comme suit :

(a) égalité

$$\forall A, B \subset K \Leftrightarrow \forall x \in K, \mu_A(x) = \mu_B(x) \tag{3.8}$$

(b) complément

$$\forall x \in K : \mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{3.9}$$

(c) intersection

$$A \cap B \Leftrightarrow \forall x \in K, \mu_{A \cap B}(x) = MIN(\mu_A(x), \mu_B(x)) \tag{3.10}$$

(d) union

$$A \cup B \Leftrightarrow \forall x \in K, \mu_{A \cup B}(x) = MAX(\mu_A(x), \mu_B(x)) \tag{3.11}$$

(e) inclusion

$$A \subset B \Leftrightarrow \forall x \in K, \mu_A(x) \leq \mu_B(x) \tag{3.12}$$

– La hiérarchie

Étant donné A un ensemble où $A \subset K$ et x est un élément de K, la classification floue est définie par la fonction caractéristique d'appartenance $\mu_A(x)$ qui prend ses valeurs dans l'intervalle $[0, 1]$. Nous obtenons une hiérarchie d'un ensemble H dans K si les conditions suivantes sont vérifiées :

(a) Tout singleton A appartient à H :

i. $Card(A) = 1$ et $A \subset K \Rightarrow A \in H$

ii. Un singleton est une classe avec un seul élément.

(b) $\forall\, A, B \in H, A \cap B \neq \varnothing \Rightarrow (A \subset B) \vee (B \subset A)$.

La figure 3.1 illustre le principe de la hiérarchie utilisé dans notre méthode.

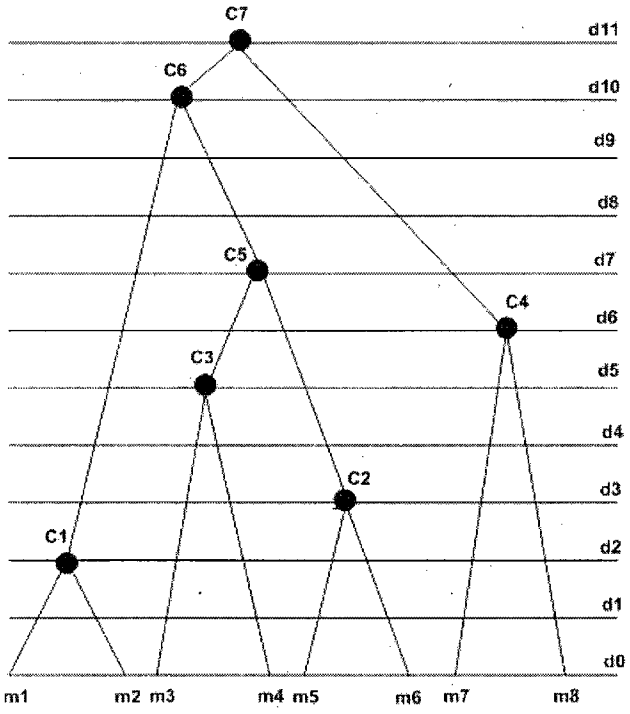


Figure 3.1 – Principe de la hiérarchie

– La hiérarchie dirigée

Cette hiérarchie devient une hiérarchie dirigée H sur un ensemble K s'il existe une fonction $\phi : H \rightarrow \mathrm{R}^+$ (voir figure 3.2) telle que :

(a) Pour chaque singleton, $\phi = 0$, c'est-à-dire :
- $(A \in H$ et $Card(A) = 1) \Rightarrow \phi(A) = 0$.

(b) Étant donné A et B des éléments de la hiérarchie H, si $B \subset A$ alors $\phi(A)$ est supérieure à $\phi(B)$, c'est-à-dire :
- $\forall\, A, B \in H, B \subset A$ et $A \neq B \Rightarrow \phi(A) > \phi(B)$.

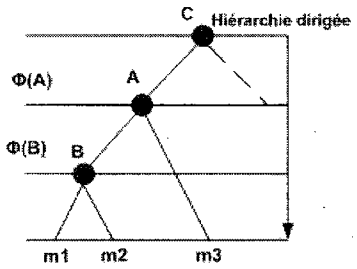


Figure 3.2 – Concept de la hiérarchie dirigée

– La fermeture transitive de la relation floue

La théorie des ensembles flous a été introduite par [98, 99]. La relation floue consiste à déterminer un chemin indirect qui existe entre les classes. Ce chemin représente le chemin le plus court qui lie ces classes. Dans notre travail, nous utilisons la fermeture transitive Min-Max pour déterminer la relation floue entre les classes.

Étant donné deux classes $C_p = \{x_{i1}, x_{i2}, ..., x_{in}\}$ et $C_q = \{x_{j1}, x_{j2}, ..., x_{jn}\}$, où x_{ij} représente le degré de similarité entre chaque paire de classes C_i et C_j, obtenu à partir de la matrice de similarité, la relation floue entre C_i et C_j est définie comme suit :

$$\Gamma_{ij} = Min[Max(x_{i1}, x_{1j}), Max(x_{i2}, x_{2j}), ..., Max(x_{in}, x_{nj})] \tag{3.13}$$

Nous itérons sur le calcul de la fermeture transitive Min-Max exprimée par l'équation (3.13) jusqu'à ce que nous obtenions la fermeture transitive Γ : $R^{\kappa-1}$=R^{κ}. Cette

égalité, d'une part, assure l'existence d'une hiérarchie dirigée et, d'autre part, détermine le niveau de chaque classe dans la hiérarchie. La figure 3.3 illustre un exemple de calcul de la fermeture transitive entre deux classes.

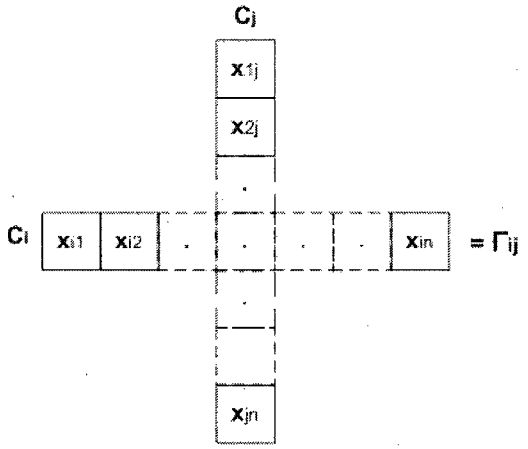


Figure 3.3 – Exemple de calcul de la fermeture transitive entre deux classes

3. **L'introduction du SVM au niveau de la hiérarchie :** Dans cette étape, nous introduisons le classificateur SVM à chaque niveau de la hiérarchie pour décomposer le problème original en sous-problèmes binaires. Pour cela, nous procédons à la préparation de l'ensemble d'apprentissage du SVM. Premièrement, nous calculons la similarité moyenne entre toutes les classes à partir de la matrice de fermeture transitive obtenue dans la deuxième étape :

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{n}\Gamma_{ij} \tag{3.14}$$

où Γ_{ij} représente la similarité floue entre C_i et C_j et n représente le nombre de Γ_{ij} dans la matrice de fermeture transitive.

Deuxièmement, nous calculons la similarité moyenne de chaque classe C_i par rapport aux autres classes :

$$\bar{v}_i = \frac{1}{k}\sum_{j=1}^{k}\Gamma_{ij}, j = 1, ..., n : i \neq j \tag{3.15}$$

où k représente le nombre des classes.

Nous obtenons l'ensemble d'apprentissage $E = \{(C_1, y_1, \upsilon_1), ..., (C_k, y_k, \upsilon_k)\}$.

Chaque classe C_i est affectée à $y_i \in \{-1, +1\}$ avec un degré d'appartenance υ_i, où $C_i \in \mathbf{R^k}$ et $0 \leq \upsilon_i \leq 1$. L'ensemble d'apprentissage est construit comme suit :

$$SVM = \begin{cases} \{C_i\} \cup SVM_{ij}^{+} & : \upsilon_i > \bar{X} \\ \{C_i\} \cup SVM_{ij}^{-} & : \upsilon_i \leq \bar{X} \end{cases} \tag{3.16}$$

Dans ce travail, nous avons introduit les valeurs floues υ_i dans la phase d'apprentissage du SVM pour améliorer la performance du SVM. Chaque ligne i de la matrice de fermeture transitive définit la similarité floue entre la classe C_i et le reste des classes. La construction de l'ensemble flou nous conduit au problème d'optimisation :

$$\begin{cases} \min \frac{1}{2} w^T w + C \sum_{i=1}^{k} \upsilon_i \xi_i \\ y_i(wx_i + b) \geq 1 - \upsilon_i \xi_i \\ \upsilon_i \xi_i \geq 0 : i = 1, k \end{cases} \tag{3.17}$$

où C et $\upsilon_i \xi_i$ représentent la pénalité de l'erreur de classification.

En utilisant les multiplicateurs de Lagrange, le problème d'optimisation devient :

$$\begin{cases} \max : w(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha_i \alpha_j y_i y_j K(C_i, C_j) \\ \sum_{i=1}^{k} y_i x_i = 0, 0 \leq \alpha_i \leq \upsilon_i C : i = 1, k \end{cases} \tag{3.18}$$

Nous répétons le processus de classification à chaque niveau de la hiérarchie de la racine jusqu'aux feuilles. Par conséquent, nous obtenons une classification hiérarchique descendante représentée par une succession de classes. Chaque classe regroupe les objets similaires.

La structure hiérarchique obtenue par notre méthode SVM-CHF offre l'avantage de classer de nouveaux objets. La figure 3.4 illustre un exemple de classification d'un nouvel objet. Supposons que nous voulons classifier le nouvel objet X. Premièrement, nous appliquons le SVM_1 au niveau de la racine sur l'ensemble $\{C_1, C_2, ..., C_6\}$.

Le résultat obtenu par SVM_1 est $X \in \{C_1, C_2, C_3, C_4\}$. Ensuite, nous appliquons le SVM_{12} sur $\{C_1, C_2, C_3, C_4\}$, le résultat obtenu par SVM_{12} est $X \in \{C_1, C_2\}$. Finalement, nous appliquons le SVM_{122}, le résultat obtenu est $X \in \{C_1\}$. La méthode SVM-CHF fournit le chemin suivi lors de la classification d'un nouvel objet $X : SVM_1 \rightarrow SVM_{12} \rightarrow SVM_{122}$.

À partir de l'exemple présenté, il semble bien que notre méthode fournit un nombre optimal de SVM_i. Pour un nombre de six classes, nous avons construit cinq classificateurs SVM_i. Par contre, les méthodes «Un-contre-un» et DAGSVM ont besoin d'un nombre de $\frac{k(k-1)}{2} = 15$ classificateurs et la méthode «Un-contre-tous» a besoin de $k = 6$ classificateurs.

En outre, notre méthode offre une architecture mixte (figure 3.4) qui nous permet d'établir la tâche de classification (descendant) et la tâche de *clustering* (ascendant) en même temps.

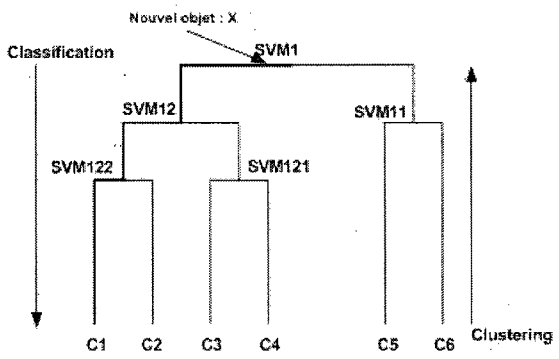


Figure 3.4 – Classification d'un nouvel objet

3.1.2 Présentation de l'algorithme SVM-CHF

Notre algorithme nécessite deux paramètres d'entrée. Le premier paramètre, n, représente le nombre de données et le deuxième paramètre, β, représente le seuil de similarité entre deux classes. Le choix du nombre de classes k est basé sur l'erreur de classification. La valeur de k qui minimise l'erreur ξ est maintenue. Le seuil de similarité β représente le critère d'arrêt pour les subdivisions du SVM au niveau de la hiérarchie. Ce seuil est obtenu à partir de la matrice de fermeture transitive et il représente la distance minimale entre

deux classes. Une fois ce seuil atteint, les deux classes $\{C_i, C_j\}$ sont automatiquement regroupées dans la même classe. Ce seuil assure le regroupement des classes homogènes et permet de diminuer le nombre de SVM appliqués tout au long de la hiérarchie, ce qui diminue automatiquement le temps d'apprentissage. Dans l'exemple de la figure 3.4, le nombre maximal de SVM construit est égal à 5. Dans le cas où la similarité entre les classes au niveau de la hiérarchie atteint un seuil inferieur ou égal à β, nous obtenons un nombre de SVM inférieur. Le détail de notre algorithme est décrit dans (Algorithme 4).

Algorithme 4 Algorithme SVM-CHF

Entrée : n : ensemble de données, β : seuil de similarité ;
Sortie : Classification hiérarchique floue des classes ;
1. Trouver les centres de classes C_i en utilisant K-Moyennes ;
2. Calculer la similarité entre chaque paire de classes (C_i,C_j) avec l'équation (3.6) ;
3. Calculer la fermeture transitive en utilisant l'équation (3.13) ;
4. À partir de la matrice de similarité, calculer, pour toutes les classes C_i, la similarité moyenne en utilisant l'équation (3.14) ;
5. À partir de la matrice de similarité, pour chaque classe C_k, calculer la similarité moyenne en utilisant l'équation (3.15) ;
6. Si $Sim(C_i, C_j) < \beta$ **Alors** $\{C_i, C_j\} \in C_k$;
7. Sinon aller à (8) ;
8. Appliquer SVM pour la phase d'apprentissage :
Si $\bar{v}_i > \bar{X}$: $SVM_{l^+} = (C_i \cup SVM_{l^+})$;
Sinon : $SVM_{l^-} = (C_i \cup SVM_{l^-})$.
9. Répéter (4), (5) et (6) à chaque niveau de la hiérarchie à partir de la racine jusqu'aux feuilles.

3.2 Présentation de la méthode SVM-CHF-Text

Dans cette section, nous présentons la méthode SVM-HCF-Text adaptée au traitement du problème de la catégorisation des textes. La figure 3.5 illustre les étapes de la méthode SVM-CHF-Text. La méthode adaptée comprend deux tâches : (i) subdivision du problème original en sous-problèmes binaires, et (ii) classification et recherche des documents pertinents. La subdivision du problème original en sous-problèmes binaires se compose de trois phases : (a) la phase de prétraitement pour réduire la dimension de l'espace, (b) la phase de la catégorisation hiérarchique floue des documents, et (c) la phase d'introduction du

SVM au niveau de la hiérarchie. En outre, la recherche des documents pertinents consiste à trouver les documents similaires qui satisfont toutes les requêtes lancées par les utilisateurs.

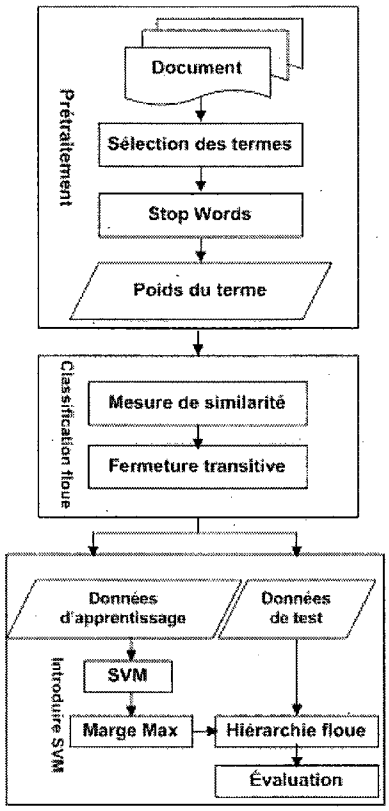


Figure 3.5 – Processus de SVM-CHF-Text pour la catégorisation des textes

1. **Subdivision du problème original**

   – **Phase de prétraitement**

   Le but principal de cette phase est de représenter les données textuelles sous une forme vectorielle facile à manipuler. Pour ce faire, nous éliminons d'abord les formats SGML (*Standard Generalized Markup Language*) [1]. Le format SGML est utilisé pour définir la structure électronique des documents. Ensuite, nous éliminons les termes qui sont inutiles dans la classification et qui ne fournissent aucune information supplémentaire sur le contenu du document. L'élimination de ces termes permet de réduire la dimension de l'espace [65]. En outre, pour extraire les attributs (*features*)

pertinents, nous appliquons la représentation ($TfIdf$) qui consiste à sélectionner les termes fréquents dans le document. Dans notre travail, nous n'avons sélectionné que les termes t_j apparaissant au moins trois fois dans le même document d_i.

– **Phase de la catégorisation hiérarchique floue des documents**

(a) **Représentation des documents :** Supposons l'ensemble de documents $D = \{d_1, d_2, ..., d_n\}$, où n représente le nombre de documents dans le corpus et $T = \{t_1, t_2, ..., t_m\}$ l'ensemble des termes où m représente le nombre des termes. La figure 3.6 illustre la représentation vectorielle de chaque document. La variable μ_{ij} représente la fréquence d'apparition du terme j dans le document i.

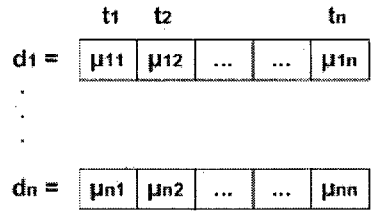


Figure 3.6 – Représentation vectorielle des documents : phase de prétraitement

$$\begin{cases} \mu_{ij} \geq 1 : t_j \in d_i \\ \mu_{ij} = 0 : \text{sinon.} \end{cases} \tag{3.19}$$

Nous appliquons la technique LSI sur la matrice d'entrée obtenue, $A[M \times N]$. La matrice A est décomposée en trois matrices comme suit : $A = U \times \Sigma \times V$. La décomposition en valeurs singulières (SVD) nous conduit à extraire les relations entre les documents à partir de la matrice V dans laquelle chaque colonne représente un document. En outre, nous éliminons de la matrice V les vecteurs qui ont les mêmes composantes. Ces vecteurs représentent les documents ayant le même contenu contextuel, c'est-à-dire, les documents traitant du même sujet. Cela nous permet de réduire le nombre de documents à catégoriser.

(b) **Processus d'agrégation hiérarchique :** L'agrégation hiérarchique consiste à trouver un pré-ordre flou sur les n documents en se basant sur leur contenu contextuel pour extraire les documents qui sont similaires. Le processus d'agrégation hiérarchique est décrit comme suit :

i. Pour chaque paire de documents (d_p,d_q) : $p \neq q$, nous calculons le facteur de similarité par l'équation (1.12). Les valeurs de cette matrice varient dans l'intervalle $[0, 1]$.

ii. Nous calculons la fermeture transitive Min-Max avec l'équation (3.13) jusqu'à l'obtention de la relation Γ égale : $R^{k-1} = R^k$. Cette égalité assure l'existence d'une hiérarchie. La distance obtenue définit le niveau de chaque classe en regroupant les documents similaires dans la même classe.

– **Introduction du SVM à chaque niveau de la hiérarchie**

À cette étape, nous appliquons la technique du SVM à chaque niveau de la hiérarchie. La préparation des données d'apprentissage et des données de test est la même que celle déjà exprimée pour la méthode SVM-CHF.

2. **Classification et recherche des documents pertinents**

Il est très utile de mettre en place un système automatique qui est capable d'affecter les documents à un ensemble prédéfini de catégories. C'est le but de la classification automatique des documents. Le processus de classification des nouveaux documents que nous proposons est similaire à celui servant à la classification des objets.

Dans cette section, nous détaillons le nouveau modèle de recherche des documents pertinents. Ce type de recherche que nous utilisons est similaire à celui de la recherche des attributs dans les grandes bases de données relationnelles. Les documents sont décrits sous forme d'attributs (termes). Le modèle de recherche proposé basé sur la logique floue consiste à utiliser une hiérarchie, à construire les sous-ensembles $F^i_{\lambda_i}$, à évaluer la requête et à appliquer les critères de sélection en utilisant l'entropie floue.

– **Construire les sous-ensembles $F^i_{\lambda_i}$**

Étant donnée λ_i, la valeur seuil appartenant à l'interval $[0, 1]$, nous affirmons qu'un document d_i appartient à une classe C_j si et seulement si $\mu_{ij} \geq \lambda_i$, où μ_{ij} représente le degré d'appartenance du document d_i à la classe C_j.

Soit $D = \{d_1, d_2, ..., d_n\}$ un ensemble de documents et $F = \{F_1, F_2, ..., F_n\}$ un ensemble des termes (caractéristiques) contenus dans chaque document d_i de l'ensemble D. Soit le document $d_j \in D$ où d_j contient au moins un attribut F_i avec un degré d'appartenance $\mu_{ij} \in [0, 1]$. Le degré d'appartenance μ_{ij} représente la fréquence normalisée de l'attribut F_i dans le document d_j. Nous obtenons donc pour

chaque document d_j de l'ensemble D, $\mu_{F_i}(d_j) = \mu_{ij}$.

Soit F_i un sous-ensemble flou de F (figure 3.7).

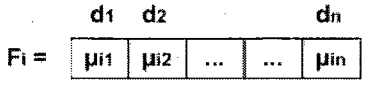


Figure 3.7 – Représentation floue des termes

À partir de l'ensemble F nous construisons les sous-ensembles $F^i_{\lambda_i}$ exprimés par leurs fonctions caractéristiques d'appartenance :

$$\mu_{F_{\lambda_i}}(x) \begin{cases} \mu_{F_{\lambda_i}}(x) < \lambda_i \Rightarrow \mu_{F_{\lambda_i}}(x) = 0 \\ \mu_{F_{\lambda_i}}(x) \geq \lambda_i \Rightarrow \mu_{F_{\lambda_i}}(x) = 1 \end{cases} \tag{3.20}$$

où λ_i sont des valeurs seuils pour les attributs F_i, fixés par l'utilisateur.

Étant donné $F = \{F_1, F_2, F_3\}$, un ensemble de trois termes et $D = \{d_1, d_2, d_3, d_4\}$ un ensemble de quatre documents, la construction des sous-ensembles $F^i_{\lambda_i}$ se fait comme suit :

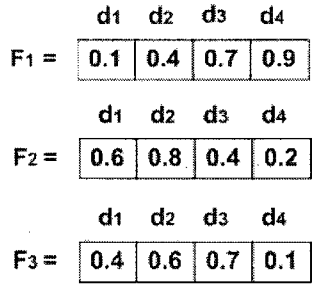


Figure 3.8 – Définition des documents : ensemble flou des termes

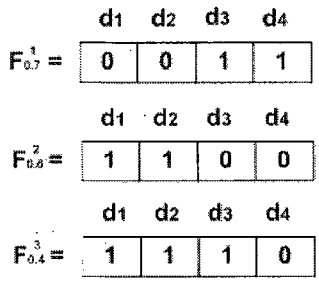


Figure 3.9 – Construction de l'ensemble flou $F^i_{\lambda_i}$

– Évaluation d'une requête

Dans notre modèle, nous supposons que les requêtes lancées par l'utilisateur sont composées des opérateurs logiques ($\wedge$: et) et ($\vee$: ou). Dans nos travaux, les opérateurs ($\wedge$) sont remplacés par l'opération d'intersection ($\cap$) et les opérateurs ($\vee$) sont remplacés par l'opération d'union ($\cup$). Supposons que nous nous intéressons à trouver le document satisfaisant l'expression :

$E_1 = [(F_1 \vee F_2) \wedge F_3]$.

Cette expression se transformera comme suit :

$E_2 = [(F_1 \cup F_2) \cap F_3]$.

À partir du cas de la figure 3.9, nous obtenons :

- $E_2 = (F^1_{0.7} \cup F^2_{0.6}) \cap F^3_{0.4}$.
- $E_2 = [\{d_3, d_4\} \cup \{d_1, d_2\}] \cap \{d_1, d_2, d_3\}$.
- $E_2 = \{d_1, d_2, d_3\}$.

Nous obtenons trois documents qui répondent à notre requête. Dans certains cas, le résultat de notre phase d'évaluation des requêtes peut contenir des documents qui appartiennent à des classes différentes. Donc, la question qui se pose est comment faire pour choisir la classe qui conserve beaucoup d'informations ?

La réponse à cette question est expliquée dans la partie qui suit.

– Critères de sélection

Lorsque nous obtenons un ensemble de documents qui appartiennent à des classes différentes, notre but est de localiser la classe la plus pertinente dans la hiérarchie.

Pour cela, nos critères de sélection de la classe pertinente consistent à :

(a) évaluer la similarité entre les documents résultant de la requête et choisir la classe la plus basse dans la hiérarchie à laquelle appartiennent ces documents ;

(b) évaluer l'entropie floue externe des classes en choisissant l'entropie la plus faible (respectivement gain de l'information).

Chaque hauteur obtenue dans la hiérarchie définit le degré de similarité entre les documents. Cette mesure est calculée à partir de la matrice de fermeture transitive donnée comme suit : $Sim(d_i, d_j)$ = Min-Max(d_i, d_j). Pour déterminer la classe qui répond à notre requête, nous utilisons la mesure de l'entropie.

– Entropie

L'entropie est une mesure de probabilité qui mesure la dégradation de l'énergie d'un état initial à un état final. Cette mesure représente une fonction d'état qui exprime toutes les transformations possibles. L'entropie d'un système mesure le degré de désordre de ses composants à partir des probabilités d'états. L'entropie d'une variable X est exprimée par la formule :

$$E(X) = \sum_j p_j log_2(p_j) \tag{3.21}$$

où les p_j représentent les probabilités possibles de déplacement d'un état à un autre état. Pour une variable avec plusieurs états, nous utilisons la somme pondérée des probabilités (p_j).

Luca *et al.* [64] proposent une autre façon d'exprimer l'entropie. Cette nouvelle entropie est liée aux sous-ensembles flous. Elle est définie comme suit :

$$E(A) = -k \sum_{i=1}^{N} \{\mu_A(x_i) \ln(\mu_A(x_i)) + (1 - \mu_A(x_i) \ln(1 - \mu_A(x_i)))\} \tag{3.22}$$

où $\mu_A(x_i)$ représente la fonction caractéristique d'appartenance de l'élément x_i au sous-ensemble flou A et $(1 - \mu_A(x_i))$ représente la fonction caractéristique d'appartenance du complément de A. Dans notre cas, nous utilisons la définition de l'entropie floue. Chaque niveau de la hiérarchie représente un état d'une classe. L'entropie consiste à choisir dans la hiérarchie la classe qui conserve la plus grande quantité

d'informations en se déplaçant d'un état à un autre (respectivement l'entropie la plus faible). Cette mesure est définie comme suit :

$$E(C_i) = -\frac{1}{\ln(N)}\Pi(h_i)\ln(\Pi(h_i)) \tag{3.23}$$

où N représente le nombre des niveaux dans la hiérarchie, h_i représente la distance entre deux niveaux de la hiérarchie et $\Pi(h_i)$ représente les valeurs relatives des distances de la hiérarchie :

$$\Pi(h_i) = \frac{h_i}{\sum_{i=1}^{N} h_i} \tag{3.24}$$

3.2.1 Présentation de l'algorithme SVM-CHF-Text

L'algorithme SVM-CHF-Text nécessite deux paramètres d'entrée : n, le nombre de documents et β, le seuil de similarité qui représente le critère d'arrêt lors de l'application du SVM. Le regroupement de deux documents se fait si la similarité $Sim(d_j, d_{j+1}) \leq \beta$. Les valeurs de β sont obtenues à partir de la matrice de fermeture transitive. Le détail de l'algorithme de notre méthode adaptée est décrit dans (Algorithme 5).

Algorithme 5 Algorithme SVM-CHF-Text

Entrée : n : nombre de documents, β : seuil de similarité ;
Sortie : structure hiérarchique floue des documents ;
1. Trouver les centres de classes C_i par K-Moyennes ;
2. Appliquer la représentation $TfIdf$;
3. Appliquer la technique LSI ;
4. À chaque niveau de la hiérarchie, calculer la similarité moyenne en utilisant les équations (3.14) et (3.15) ;
Si $(Sim(d_j, d_{j+1})) \leq \beta$ **alors** $(d_j, d_{j+1}) \in C$
Sinon aller à (5) ;
5. Appliquer SVM ;
Si $\bar{v}_i > \bar{X}$: $SVM_{l^+} = (d_i \cup SVM_{l^+})$;
Sinon $SVM_{l^-} = (d_i \cup SVM_{l^-})$;
6. Répéter les étapes (4) et (5) à chaque niveau de la hiérarchie à partir de la racine jusqu'aux feuilles. Sélectionner le nombre de SVM obtenus.

3.3 Conclusion

Dans ce chapitre, nous avons présenté en détail les étapes de notre méthode SVM-CHF pour résoudre le problème multi-classes ainsi que sa version adaptée pour la catégorisation des textes. En outre, nous avons présenté un nouveau modèle de recherche des documents basé sur la logique floue. Le modèle développé permet de satisfaire les requêtes des utilisateurs. Dans le chapitre suivant et afin d'appuyer nos méthodes, nous présentons deux cas d'applications. Le premier cas traite les données numériques et le deuxième cas traite les données textuelles.

Chapitre 4

Cas d'application et résultats expérimentaux

4.1 Introduction

L'objectif de ce chapitre est d'illuster le déroulement de notre méthode SVM-CHF ainsi que sa version adaptée à l'aide de données réelles. Une étude comparative est réalisée entre les méthodes que nous avons développées et les méthodes de classification existantes. Ce chapitre est subdivisé en deux parties. La première partie présente deux cas d'application sur deux types différents de données (données numériques et textuelles) afin d'illuster le déroulement de nos algorithmes. La deuxième partie présente nos résultats expérimentaux.

4.2 Cas d'application

Dans cette section, nous présentons les différentes étapes de nos algorithmes sur deux cas d'application différents. Le premier cas est celui de la base de données Iris [14] qui contient trois classes à séparer. Dans ce premier cas d'application, nous démontrons les étapes du déroulement de l'algorithme SVM-CHF. Dans le deuxième cas d'application qui est extrait des travaux de Baldi *et al.* [8], nous démontrons le déroulement de l'algorithme SVM-CHF-Text.

4.2.1 Cas d'application IRIS : données numériques

Afin d'illuster le déroulement de notre algorithme SVM-CHF, nous présentons un cas pratique sur les données de la base de données Iris [14]. Premièrement, nous normalisons les données traitées en utilisant la technique min-max, exprimée par la formule (1.4). Deuxièmement, nous appliquons l'algorithme K-Moyennes sur l'ensemble des données normalisées, pour regrouper les données dans des classes. À ce point, nous faisons varier le nombre de classes ($k \in \{2, 3, 4, 6, 8, 10\}$), en sélectionnant le nombre de classes qui minimise la somme des erreurs au carré (SSE). Les résultats de l'application de l'algorithme K-Moyennes sont exprimés dans le tableau 4.1.

Tableau 4.1 – Application de K-Moyennes sur Iris

	Différentes valeurs de k					
Valeurs (k)	2	3	4	6	8	10
SSE	62.143	**7.817**	16.382	9.131	11.852	33.678

Les trois classes obtenues par K-Moyennes sont : $C_1 = \{0.195, 0.592, 0.077, 0.04\}$, $C_2 = \{0.463, 0.320, 0.55, 0.50\}$ et $C_3 = \{0.63, 0.419, 0.765, 0.71\}$.

Tableau 4.2 – Résultat de K-Moyennes sur Iris

C_1	0.195	0.592	0.077	0.04
C_2	0.463	0.320	0.55	0.50
C_3	0.63	0.419	0.765	0.71

La troisième étape consiste à mesurer la similarité entre les trois classes obtenues dans la deuxième étape. Nous appliquons la distance relative de Hamming, exprimée par la formule (3.6). Les résultats sont exprimés dans le tableau 4.3.

Tableau 4.3 – Matrice de similarité du problème Iris

Classe	C_1	C_2	C_3
C_1	0.000	0.37	0.49
C_2	0.37	0.000	0.17
C_3	0.49	0.17	0.000

Une fois la matrice de similarité construite, nous appliquons la formule (3.13) pour calculer la fermeture transitive Min-Max qui consiste à trouver le chemin le plus court

entre les classes C_1, C_2 et C_3. Nous répétons le calcul de la fermeture transitive jusqu'à l'obtention d'une égalité : $R^{k-1} = R^k$, où k représente le niveau de chaque étape. Dans notre cas, nous arrêtons le calcul à la deuxième étape : $(k = 2)$. Le résultat de calcul de la fermeture transitive est décrit dans le tableau 4.4.

Tableau 4.4 – Calcul de la fermeture transitive Min-Max

Classe	C_1	C_2	C_3
C_1	0.00	0.37	0.37
C_2	0.37	0.00	0.17
C_3	0.37	0.17	0.00

À partir du tableau 4.4, nous constatons que C_2 et C_3 sont plus proches l'une de l'autre que de C_1. Ce constat nous donne une information *a priori* que ces deux classes peuvent se regrouper en une seule classe au niveau de la structure hiérarchique. L'étape suivante consiste à calculer la similarité totale de toutes les classes et la similarité moyenne de chaque classe par rapport aux autres classes, en utilisant les deux formules (3.14) et (3.15) respectivement.

1. Similarité globale entre les trois classes :
   $Sim(C_1, C_2, C_3) = 0.22$
2. Similarité de chaque classe par rapport aux autres classes :
   $-Sim(C_1, (C_2, C_3)) = 0.25$
   $-Sim(C_2, (C_1, C_3)) = 0.18$
   $-Sim(C_3, (C_2, C_3)) = 0.18$

La dernière étape consiste à préparer l'ensemble d'apprentissage pour le classificateur SVM. Les résultats obtenus par les formules (3.14) et (3.15) nous permettent de construire les deux classes, positive et négative, du SVM, de telle sorte que : $SVM_{l+} = \{C_1\}$ et $SVM_{l-} = \{C_2, C_3\}$. Ensuite, nous appliquons le SVM à chaque niveau de la hiérarchie. Dans le cas de la base de données Iris, nous avons fixé les paramètres $(C = 1, \xi = 0.107)$. Nous avons utilisé la fonction polynomiale d'ordre 2 comme fonction de noyau. La figure 4.1 illustre le déroulement de notre algorithme sur la base de données Iris.

Pour tester la performance de notre méthode, nous prenons un échantillon de 50 enregistrements de la classe positive et 50 enregistrements de la classe négative. Les résultats

sont décrits dans le tableau 4.5 (matrice de confusion).

Selon les résultats du tableau 4.5, nous avons obtenu une précision de 98%, c'est-à-dire une erreur de 0.02. Ce qui signifie que la précision est très élevée.

Dans ce cas, le déroulement de l'algorithme SVM-CHF a conduit à la construction de deux classificateurs SVM. Nous appliquons SVM_1 sur tout l'ensemble des classes $\{C_1, C_2, C_3\}$ au niveau de la racine. Nous obtenons deux sous-ensembles : $\{C_1\}$ et $\{C_2, C_3\}$. Ensuite, nous appliquons SVM_2 sur le sous-ensemble $\{C_2$ et $C_3\}$ (figure 4.1). Tout au long de la hiérarchie, nous retenons deux classificateurs SVM.

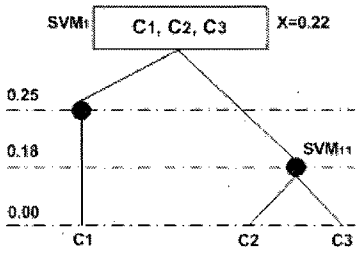


Figure 4.1 – Cas illustratif du déroulement de SVM-CHF sur les données Iris

Tableau 4.5 – Matrice de confusion du problème Iris

Classe	$Classe(+)$	$Classe(-)$
$Classe(+)$	49	1
$Classe(-)$	1	49

4.2.2 Cas d'application de dix classes : données textuelles

Dans cette section, nous présentons un cas pratique de données textuelles dans le but d'illustrer le déroulement de notre algorithme SVM-CHF-Text. Le cas pratique abordé dans cette section fait partie des travaux de Baldi *et al.* [8]. L'échantillon est composé de dix documents de sujets différents, dont les cinq premiers sont relatifs au système d'exploitation Linux et les cinq autres sont reliés au domaine de la génétique. Selon [8], chaque document appartient à un seul sujet. Nous démontrons par notre méthode que chacun de ces documents peut appartenir aux deux sujets en même temps avec un degré d'appartenance et que ces documents peuvent faire l'objet d'une étude de classification floue et non seulement d'une classification classique.

Dans notre cas, nous nous intéressons aux termes qui apparaissent au moins deux fois dans le document. Les termes qui nous intéressent sont soulignés.

- d_1 : Indian government goes for open-source software ;
- d_2 : Debian 3.0 Woody released ;
- d_3 : Wine 2.0 released with fixes for Gentoo 1.4 and Debian 3.0 ;
- d_4 : gnu POD released : iPod on Linux... with GPLed software ;
- d_5 : Gentoo servers running an open-source mySQL database ;
- d_6 : Dolly the sheep not totally identical clone ;
- d_7 : DNA news : introduced low-cost human genome DNA chip ;
- d_8 : Malaria-parasite genome database on the Web ;
- d_9 : UK sets up genome bank to protect rare sheep breeds ;
- d_{10} : Dolly's DNA Damaged.

La matrice $A^T[11 * 10]$ (terme/document) est présentée comme suit :

Tableau 4.6 – Représentation terme/document

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
Open-source	1	0	0	0	1	0	0	0	0	0
Software	1	0	0	1	0	0	0	0	0	0
Linux	0	0	0	1	0	0	0	0	0	0
Released	0	1	1	1	0	0	0	0	0	0
Debian	0	1	1	0	0	0	0	0	0	0
Gentoo	0	0	1	0	1	0	0	0	0	0
Database	0	0	0	0	1	0	0	1	0	0
Dolly	0	0	0	0	0	1	0	0	0	1
Sheep	0	0	0	0	0	1	0	0	0	0
Genome	0	0	0	0	0	0	1	1	1	0
DNA	0	0	0	0	0	0	2	0	0	1

Nous appliquons la décomposition en valeur singulières (SVD), en utilisant la formule (1.7). Nous obtenons les matrices U, Σ et V^T :

$$\mathbf{U} = \begin{pmatrix} -0.047 & 0.226 & 0.517 & 0.313 & -0.020 & -0.278 & 0.418 & -0.189 & -0.332 & -0.428 & 0.000 \\ -0.022 & 0.287 & 0.028 & 0.637 & -0.301 & 0.113 & 0.126 & -0.124 & 0.431 & 0.437 & 0.000 \\ -0.011 & 0.198 & -0.118 & 0.310 & -0.187 & 0.182 & -0.396 & 0.388 & -0.242 & -0.289 & 0.577 \\ -0.042 & 0.652 & -0.351 & -0.045 & -0.041 & 0.138 & -0.143 & -0.027 & -0.224 & -0.151 & -0.577 \\ -0.030 & 0.4541 & -0.2334 & -0.356 & 0.147 & -0.045 & 0.252 & -0.415 & 0.018 & 0.139 & 0.577 \\ -0.056 & 0.398 & 0.294 & -0.232 & 0.203 & -0.292 & 0.045 & 0.6298 & 0.4154 & 0.0257 & 0.000 \\ -0.137 & 0.160 & 0.582 & -0.122 & 0.082 & 0.076 & -0.570 & -0.244 & -0.232 & 0.389 & 0.000 \\ -0.208 & -0.032 & -0.106 & -0.100 & -0.458 & -0.596 & -0.373 & -0.273 & 0.297 & -0.261 & 0.000 \\ -0.160 & -0.015 & 0.056 & -0.321 & -0.693 & 0.006 & 0.304 & 0.270 & -0.347 & 0.318 & 0.000 \\ -0.525 & -0.023 & 0.205 & -0.191 & -0.115 & 0.598 & 0.082 & -0.102 & 0.345 & -0.372 & 0.000 \\ -0.793 & -0.105 & -0.242 & 0.231 & 0.316 & -0.229 & 0.0527 & 0.109 & -0.204 & 0.201 & 0.000 \end{pmatrix}$$

$$\mathbf{\Sigma} = \begin{pmatrix}
\mathbf{0.2566} & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & \mathbf{0.2397} & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & \mathbf{0.1936} & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & \mathbf{0.1705} & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.1681} & 0.000 & 0.000 & -0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.1540} & 0.000 & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.1022} & 0.000 & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.0782} & 0.000 & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.0384} & 0.000 \\
0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & \mathbf{0.0107}
\end{pmatrix}$$

$$\mathbf{V^T} = \begin{pmatrix}
-0.0267 & 0.2137 & 0.2814 & 0.5570 & -0.1911 & -0.1074 & 0.5325 & -0.3997 & 0.2582 & 0.0780 \\
-0.0281 & 0.4613 & -0.3019 & -0.2352 & 0.0629 & 0.0605 & 0.1070 & -0.5648 & -0.5379 & -0.1119 \\
-0.0498 & 0.6273 & -0.1499 & -0.3712 & 0.1833 & -0.1290 & 0.1509 & 0.2406 & 0.5448 & 0.1268 \\
-0.0291 & 0.4740 & -0.2279 & 0.5292 & -0.3148 & 0.2806 & -0.4040 & 0.3031 & -0.0929 & -0.0311 \\
-0.0932 & 0.3269 & 0.7196 & -0.0241 & 0.1570 & -0.3207 & -0.1051 & 0.2519 & -0.3855 & -0.1241 \\
-0.1432 & -0.0194 & -0.0258 & -0.2472 & -0.6848 & -0.3826 & -0.0676 & -0.0035 & -0.1296 & 0.5298 \\
-0.8224 & -0.0969 & -0.1445 & 0.1583 & 0.3076 & 0.0915 & 0.1837 & 0.1476 & -0.1608 & 0.2897 \\
-0.2578 & 0.0572 & 0.4064 & -0.1836 & -0.0196 & 0.4375 & -0.4778 & -0.4425 & 0.2966 & 0.1659 \\
-0.2668 & -0.0158 & 0.1351 & -0.3006 & -0.4808 & 0.3924 & 0.3782 & 0.2150 & -0.0034 & -0.4956 \\
-0.3899 & -0.0569 & -0.1802 & 0.0764 & -0.0843 & -0.5351 & -0.3136 & -0.2101 & 0.2435 & -0.5579
\end{pmatrix}$$

Dans notre exemple, selon la matrice V^T, nous conservons le même nombre de vecteurs de la matrice d'entrée. Cela veut dire qu'il n'y a pas de réduction de l'espace. La conservation de la même dimension de l'espace a deux explications possibles. La première est due au nombre limite de documents traités. La deuxième est liée à la longueur des documents eux mêmes qui ne sont pas assez longs, ce qui ne permet pas de faire ressortir exactement son contenu contextuel. Donc, nous traitons notre matrice d'entrée telle quelle sans aucune élimination de document.

Nous mesurons la similarité entre les dix documents, en appliquant l'équation (3.6). Les résultats sont présentés dans le tableau 4.7. Une fois que la matrice de similarité entre les documents est construite, nous appliquons la fermeture transitive Min-Max dans le but de trouver le chemin le plus court entre les documents en suivant les mêmes étapes suivies dans le cas d'application pour les données numériques. Nous répétons la même procédure du calcul de la fermeture transitive Min-Max trois fois de suite jusqu'à l'obtention de la relation $R^{k-1} = R^k$ (dans notre cas $k = 3$). Les résultats sont représentés dans le tableau 4.8. La dernière étape consiste à introduire le SVM à chaque niveau de la hiérarchie. Cette étape nécessite la préparation des données d'apprentissage à chaque niveau de la hiérarchie en utilisant les formules (3.14) et (3.15). Les résultats sont présentés dans le tableau 4.9.

Au niveau de la racine, l'ensemble de données contient tous les documents $\{d_1, ..., d_{10}\}$. La similarité globale entre les documents est égale à $\bar{X} = \mathbf{0.364}$. Les résultats du SVM_1

Tableau 4.7 – Cas pratique : matrice de similarité des données textuelles

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
d_1	**0.000**	0.300	0.500	0.300	0.300	0.400	0.500	0.400	0.400	0.400
d_2	0.300	**0.000**	0.100	0.300	0.400	0.400	0.500	0.400	0.400	0.400
d_3	0.500	0.100	**0.000**	0.400	0.400	0.500	0.600	0.500	0.500	0.500
d_4	0.300	0.300	0.400	**0.000**	0.600	0.500	0.600	0.500	0.500	0.500
d_5	0.300	0.400	0.400	0.600	**0.000**	0.500	0.600	0.300	0.500	0.500
d_6	0.400	0.400	0.500	0.500	0.500	**0.000**	0.500	0.400	0.200	0.200
d_7	0.500	0.500	0.600	0.600	0.600	0.500	**0.000**	0.300	0.300	0.300
d_8	0.400	0.400	0.500	0.500	0.300	0.400	0.300	**0.000**	0.200	0.400
d_9	0.400	0.400	0.500	0.500	0.500	0.200	0.300	0.200	**0.000**	0.400
d_{10}	0.400	0.400	0.500	0.500	0.500	0.200	0.300	0.400	0.400	**0.000**

Tableau 4.8 – Cas pratique : la fermeture Min-Max pour les données textuelles

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}
d_1	**0.000**	0.400	0.400	0.300	0.300	0.500	0.500	0.400	0.500	0.500
d_2	0.400	**0.000**	0.100	0.400	0.500	0.500	0.500	0.500	0.500	0.500
d_3	0.400	0.100	**0.000**	0.400	0.400	0.500	0.500	0.500	0.500	0.500
d_4	0.300	0.400	0.400	**0.000**	0.300	0.500	0.500	0.500	0.500	0.500
d_5	0.300	0.400	0.400	0.300	**0.000**	0.500	0.400	0.300	0.500	0.500
d_6	0.500	0.500	0.500	0.500	0.500	**0.000**	0.500	0.500	0.300	0.300
d_7	0.500	0.500	0.500	0.500	0.400	0.500	**0.000**	0.200	0.300	0.300
d_8	0.400	0.500	0.500	0.500	0.300	0.500	0.200	**0.000**	0.400	0.500
d_9	0.500	0.500	0.500	0.500	0.500	0.300	0.300	0.400	**0.000**	0.200
d_{10}	0.500	0.500	0.500	0.500	0.500	0.300	0.300	0.500	0.200	**0.000**

sont les deux sous-ensembles : $\{d_2, d_3, d_6, d_7, d_9, d_{10}\}$ et $\{d_1, d_4, d_5, d_8\}$. À chaque niveau, nous appliquons le SVM en construisant les deux classes et nous retirons de la liste l'ensemble des documents déjà classés lors de l'itération précédente. La figure 4.2 illustre les étapes de notre méthode sur l'ensemble des documents.

Finalement, nous obtenons une structure hiérarchique floue, constituée d'une succession de classes. Chaque classe regroupe les documents qui partagent un degré de similarité. À l'inverse de la classification classique, qui regroupe les dix documents en deux classes, notre méthode subdivise l'ensemble des documents en six classes : C_1 regroupe $\{d_2, d_3\}$, C_2 regroupe $\{d_1, d_4, d_5\}$, C_3 regroupe $\{d_8, C_2, C_4\}$, C_4 regroupe $\{d_6, d_7, C_1\}$, C_5 regroupe $\{d_9, d_{10}\}$ et C_6 regroupe $\{C_4, C_5\}$. La figure 4.3 décrit les résultats obtenus par notre méthode. Selon la figure 4.3, nous constatons que les deux classes C_2 et C_4 sont regroupées en une seule classe. Mais selon [8], il n'existe aucun lien entre les deux ensembles $\{d_1, d_2, d_3, d_4, d_5\}$ et $\{d_6, d_7, d_8, d_9, d_{10}\}$ (classification classique). Par contre, selon notre méthode, nous avons pu trouver un lien indirect reliant les deux sous-ensembles

Tableau 4.9 – Cas pratique : l'ensemble d'apprentissage des données textuelles

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_{10}	**Similarité**
d_1	**0.00**	0.40	0.40	0.30	0.30	0.50	0.50	0.40	0.50	0.50	**0.37**
d_2	0.40	**0.00**	0.10	0.40	0.40	0.50	0.50	0.50	0.40	0.40	**0.36**
d_3	0.40	0.10	**0.00**	0.40	0.40	0.50	0.50	0.50	0.40	0.40	**0.36**
d_4	0.30	0.40	0.40	**0.00**	0.30	0.50	0.50	0.50	0.50	0.50	**0.39**
d_5	0.30	0.40	0.40	0.30	**0.00**	0.50	0.50	0.40	0.50	0.50	**0.38**
d_6	0.50	0.50	0.50	0.50	0.50	**0.00**	0.30	0.30	0.30	0.20	**0.36**
d_7	0.50	0.50	0.50	0.50	0.50	0.30	**0.00**	0.20	0.30	0.30	**0.35**
d_8	0.40	0.50	0.50	0.50	0.40	0.30	0.20	**0.000**	0.40	0.50	**0.38**
d_9	0.50	0.40	0.40	0.50	0.50	0.30	0.30	0.40	**0.00**	0.20	**0.35**
d_{10}	0.50	0.40	0.40	0.50	0.50	0.20	0.30	0.50	0.20	**0.00**	**0.35**

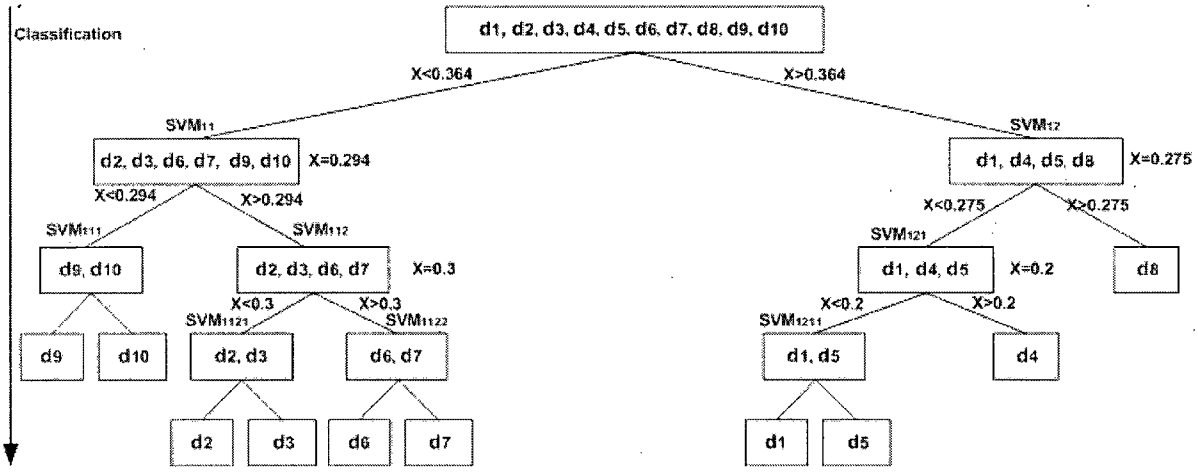


Figure 4.2 – Cas illustratif du déroulement de SVM-CHF-Text sur les données textuelles

$C_2 = \{d_1, d_4, d_5\}$ et $C_4 = \{d_6, d_7, C_1\}$. Le lien trouvé entre la classe C_2 et C_4 est dû à la similarité qui existe entre les documents d_5 et d_8. Les documents d_5 et d_8 partagent le terme (*database*) qui n'a pas été pris en considération dans les travaux de la classification de [8]. Grâce à ce chemin indirect, nous avons regroupé la classe $C_1 = \{d_2, d_3\}$ et l'ensemble $\{d_6, d_7\}$ en une même classe C_4. Ce chemin permet aussi de trouver une similarité entre les classes C_1 , C_4 et C_5. Nous avons démontré, grâce à ce cas d'application, qu'un document peut appartenir à plusieurs classes en même temps. Il suffit de trouver des chemins indirects entre les documents.

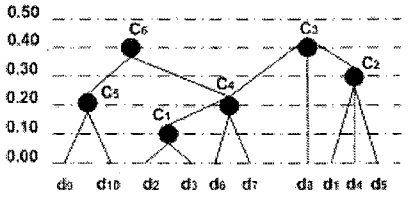


Figure 4.3 – Structure hiérarchique floue obtenue par SVM-CHF-Text (cas pratique)

4.3 Résultats expérimentaux

4.3.1 Données de tests

Notre objectif dans cette partie consiste à valider les deux méthodes que nous avons présentées dans le chapitre précédent et à établir une comparaison par rapport aux méthodes les plus répandues dans le domaine de la classification. Les deux méthodes développées, à savoir SVM-CHF et SVM-CHF-Text, sont testées sur deux types différents de données. Pour bien organiser la partie expérimentale, nous l'avons subdivisé en deux parties. La première partie concerne les données numériques et la deuxième partie concerne les données textuelles.

Il faut noter que les mesures d'évaluation choisies sont la précision, le rappel et F-Mesure. En outre, nous avons calculé le nombre des SVM retenus et le facteur temps qui est très important pour l'évaluation de la performance des classificateurs. Ces critères sont exprimés comme suit :

$$\text{Précision} = \frac{\text{TP}}{\text{TP+FP}} \tag{4.1}$$

$$\text{Rappel} = \frac{\text{TP}}{\text{TP+FN}} \tag{4.2}$$

$$\text{F-Mesure} = \frac{2 \times \text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}} \tag{4.3}$$

De plus, la Macro-Moyenne et la Micro-Moyenne sont deux mesures utilisées pour évaluer la performance moyenne des classificateurs binaires. Dans nos travaux, nous nous intéressons à la Micro-Moyenne-Précision et à la Micro-Moyenne-Rappel, qui sont exprimées respectivement comme suit :

$$\pi^{\mu} = \frac{\sum_{i=1}^{m} \mathrm{TP}_i}{\sum_{i=1}^{m} (\mathrm{TP}_i + \mathrm{FP}_i)} \tag{4.4}$$

$$\rho^{\mu} = \frac{\sum_{i=1}^{m} \mathrm{TP}_i}{\sum_{i=1}^{m} (\mathrm{TP}_i + \mathrm{FN}_i)} \tag{4.5}$$

où m représente le nombre des catégories et μ représente la Micro-Moyenne.

Ces critères de mesure de performance sont obtenus à partir de la matrice de confusion en calculant les quantités suivantes : le vrai positif (TP) qui représente les enregistrements positifs classés positifs, le vrai négatif (TN) qui représente les enregistrements négatifs classés négatifs, le faux positif (FP) qui représente les enregistrements négatifs classés positifs et le faux négatif (FN) qui représente les enregistrements positifs classés négatifs.

– **Données numériques**

Nous avons appliqué notre méthode SVM-CHF sur trois bases de données largement utilisées dans le domaine de la classification, qui sont Iris, Glass et Lettre [14].

Iris : C'est une base de données qui contient 150 enregistrements regroupés en trois classes : *Setosa*, *Versicolor*, et *Virginica*. Chaque classe contient 50 enregistrements. Chaque enregistrement est caractérisé par quatre attributs : longueur de sépale, largeur de sépale, longueur de pétale et largeur de pétale.

Glass : C'est une base de données qui contient 214 enregistrements, regroupés en six classes : *building windows float processed*, *building windows non float processed*, *vehicle windows float processed*, conteneurs, arts de la table et projecteurs.

Lettre : C'est une base de données qui contient 20000 enregistrements, regroupés en 26 classes. Chaque classe représente une lettre de l'alphabet (A-Z) de la langue française. Chaque enregistrement est caractérisé par 16 attributs.

Le détail des trois problèmes est décrit dans le tableau 4.10.

– **Données textuelles**

Dans nos expérimentations, nous avons utilisé deux collections de texte largement utilisées dans la communauté de la catégorisation des textes, qui sont : Reuters-21578 et 20 Newsgroups.

Tableau 4.10 – Détails statistiques des problèmes : Iris, Glass et Lettre

	Problèmes		
Données	**Iris**	**Glass**	**Lettre**
# Données d'apprentissage	100	142	13334
# Données de test	50	72	6666
# Class	3	6	26
# Attributs	4	9	16

Reuters-21578 : Cette collection des textes contient 21578 articles. Chaque article est indexé selon un nombre variable de thèmes (135), de personnes (267), de places (175), d'organisations (56) et de bourses (39). Dans nos expérimentations, nous nous intéressons seulement au thème et au corps du document. Nous avons sélectionné seulement les dix thèmes les plus fréquents : *Earn*, *Acq*, *Money-fx*, *Grain*, *Grude*, *Trade*, *Interest*, *Ship*, *Wheat* et *Corn* [1].

20 Newsgroups : Une autre collection de textes largement utilisée dans le domaine de la classification est 20 Newsgroups qui contient 20000 documents distribués sur 20 groupes d'articles. Dans nos expérimentations nous avons utilisé la collection **20NG**. Pour chaque groupe, nous avons sélectionné 100 documents qui peuvent appartenir à une ou plusieurs classes en même temps.

Le détail des deux collections est donné dans le tableau 4.11.

Tableau 4.11 – Détails statistiques de Reuters-21578 et 20 Newsgroups

	Problèmes	
Données	**Reuters-21578**	**20 Newsgroups**
# Données d'apprentissage	9603	1200
# Données de test	3299	800
# Données supprimées	8676	-
# Classes	10	20

1. Le corpus Reuters 21578, utilisé pour nos expérimentations, est disponible à l'adresse (http ://www.research.att.com/ lewis/reuters21578.html)

4.3.2 Mesure de performance de SVM-CHF

Afin de jusifier l'importance de la phase de compression des données d'apprentissage dans notre méthode, nous avons effectué deux expériences. La première expérience consiste à introduire l'algorithme K-Moyennes dans la phase de la compression des données. Les données traitées sont les centres des classes. Dans la deuxième expérience, nous traitons tout l'ensemble des données d'apprentissage sans appel à la phase de compression. Dans ces expériences, trois mesures de performance ont été prises en considération, à savoir le nombre de SVM obtenus, le temps d'apprentissage nécessaire et la précision. Les résultats sont décrits dans le tableau 4.12.

Tableau 4.12 – Influence de la phase de compression sur la performance de SVM-CHF

	Avec K-Moyennes			Sans K-Moyennes		
Données	# SVM	Temps d'apprentissage	Précision	# SVM	Temps d'apprentissage	Précision
Iris	**2**	**0.021**	98.00	3	0.05	**98.23**
Glass	**4**	**0.05**	77.63	6	11	**78.10**
Lettre	**21**	**110**	98.35	24	255	**98.45**

Selon les résultats obtenus par les deux tests effectués, nous constatons que la phase de compression améliore la performance de notre méthode. Elle permet de reduire le nombre de SVM obtenus et le temps d'apprentissage. Cette performance est due à la réduction du nombre de données d'apprentissage traitées. La classification s'est effectuée seulement sur les centres de classes obtenus par l'algorithme K-Moyennes. Par contre, nous constatons que la précision est légèrement meilleure dans le deuxième test. Cela est dû, sans aucun doute, à la nature des données traitées dans ce test. En effet, la classification s'est effectuée sur l'ensemble des données réelles et non pas sur leurs représentants (centres de classes), ce qui influe sur la précision.

Vu l'importance des deux premiers critères sur la performance de notre méthode, nous avons décidé d'introduire le K-Moyennes qui nous permet d'accélérer la tâche de classification et de réduire le nombre de SVM tout au long de la hiérarchie, malgré une légère perte de précision.

Fonction de noyau utilisée : Nous avons effectué plusieurs tests afin de choisir la meilleure fonction de noyau. Nous avons testé la fonction polynomiale et la fonction RBF,

en faisant varier leurs paramètres. Les résultats sont décrits dans le tableau 4.13. Nous constatons que pour les deux problèmes où le nombre de classes k varie entre 3 et 6, la fonction polynomiale donne des meilleurs résultats par rapport à la fonction RBF. Pour le problème lettre ($k = 26$), la fonction RBF donne de meilleurs résultats. La seule explication est que les fonctions polynomiales sont performantes lorsqu'il s'agit d'un nombre de classes faible et que la discrimination entre les classes devient difficile lorsque le nombre de classes augmente. En outre, pour le problème Iris, nous savons *a priori* que la classe 1 est séparable linéairement par rapport aux autres classes. Donc, la discrimination peut se faire par une fonction polynomiale.

Tableau 4.13 – Précision obtenue par les différentes fonctions de noyaux

	données numériques										
	$SVM_{Poly:d}$					$SVM_{RB:\gamma}$					
	2	4	6	8	10	0.1	0.2	0.4	0.6	0.8	1.0
Iris	**0.98**	0.95	0.94	0.94	0.93	0.97	0.96	0.90	0.89	0.89	0.96
Glass	0.66	**0.77**	0.76	0.69	0.74	0.66	0.69	0.72	0.67	0.63	0.65
Lettre	0.54	0.67	0.88	0.87	0.75	0.78	0.93	**0.98**	0.96	0.94	0.92

Ensuite, nous avons comparé la performance de notre méthode à celle des méthodes suivantes : «Un-contre-un», «Un-contre-tous» et DHSVM [11]. Les résultats sont décrits dans le tableau 4.14.

Tableau 4.14 – Étude comparative de la précision de chaque méthode

	Méthodes de classification			
Données	**Un-contre-un**	**Un-contre-tous**	**DHSVM**	**SVM-CHF**
Iris	97.33	96.67	97.61	**98.00**
Glass	71.49	71.96	76.76	**77.63**
Lettre	97.98	97.88	98.01	**98.35**

Nous avons calculé aussi les critères de sensibilité et de spécificité :

$$\text{sensibilité} = \frac{\text{TP}}{\text{TP+FN}} \tag{4.6}$$

$$\text{spécificité} = \frac{\text{TN}}{\text{TN+FP}} \tag{4.7}$$

La sensibilité représente une proportion d'individus positifs effectivement bien détectés lors de la phase de test. Le classificateur est parfait pour l'ensemble de données positives lorsque la sensibilité vaut 1. La spécificité représente une proportion d'individus négatifs effectivement bien détectés. Le classificateur est parfait pour l'ensemble de données négatives lorsque la spécificité vaut 1.

Pour le problème Iris, la sensibilité et la spécificité sont entre 95 et 100% et entre 92 et 100% respectivement. Pour le problème Glass, la sensibilité et la spécificité sont entre 86 et 100% et entre 81 et 100% respectivement. Pour le problème Lettre, la sensibilité et la spécificité sont entre 86 et 100% et entre 85 et 100% respectivement. L'étude comparative entre SVM-CHF et la méthode DHSVM [11] montre que nos résultats en terme de sensibilité et spécificité sont meilleurs. Les résultats de cette étude comparative sont décrits dans le tableau 4.15.

Tableau 4.15 – Étude comparative entre SVM-CHF et DHSVM

SVM-CHF *versus* DHSVM				
	DHSVM		SVM-CHF	
	Sensibilité	Spécificité	Sensibilité	Spécificité
Iris	92-100%	**94-100%**	**95-100%**	92-100%
Glass	84-100%	68-100%	**86-100%**	**81-100%**
Lettre	**94-100%**	75-100%	86-100%	**85-100%**

– Nombre de SVM et temps d'apprentissage

Nous avons calculé le nombre des SVM obtenus tout au long de la hiérarchie par notre méthode ainsi que le temps d'apprentissage nécessaire pour chaque problème. Les tableaux 4.16 et 4.17 montrent les résultats obtenus par SVM-CHF avec les résultats expérimentaux des méthodes «Un-contre-un», «Un-contre-tous» et DHSVM.

Tableau 4.16 – Nombre de SVM obtenus (données numériques)

Nombre de SVM				
Données	**Un-contre-un**	**Un-contre-tous**	**DHSVM**	**SVM-CHF**
Iris	3	3	2	**2**
Glass	15	6	5	**4**
Lettre	325	26	25	**21**

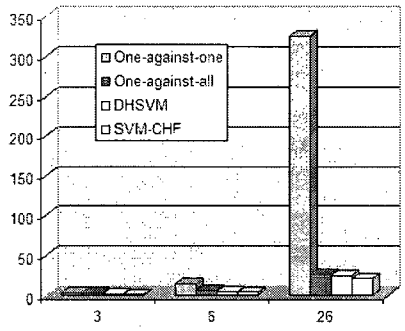


Figure 4.4 – Comparaison de nombre de SVM obtenus pour chaque méthode

Tableau 4.17 – Temps d'apprentissage obtenu (données numériques)

	Temps d'apprentissage			
Données	**Un-contre-un**	**Un-contre-tous**	**DHSVM**	**SVM-CHF**
Iris	0.04	0.10	0.03	**0.021**
Glass	2.42	10	0.09	**0.05**
Lettre	298.08	4500	140	**110**

Les résultats obtenus par SVM-CHF sont meilleurs que les résultats des trois méthodes étudiées dans ce travail. Cette performance est due à :

- l'application de l'algorithme K-Moyennes qui converge rapidement pour trouver les classes représentatives de l'ensemble des données. Nous traitons un nombre de classes qui est inférieur au nombre de l'ensemble initial des données, ce qui diminue le temps de traitement ;
- l'utilisation de la fermeture transitive Min-Max qui assure de trouver le chemin le plus court entre les classes. Cette technique permet de préparer l'ensemble d'apprentissage pour le SVM ;
- la réduction du nombre de SVM au niveau de la hiérarchie est due à l'introduction du seuil de similarité au niveau de la hiérarchie. Si le seuil est atteint, nous regroupons les deux classes en question sans l'application du SVM. Le seuil est extrait de la matrice de fermeture transitive Min-Max.
- **Influence du seuil de similarité sur la performance de SVM-CHF**

Nous avons varié les valeurs du seuil de similarité pour tester son influence sur la performance de notre méthode SVM-CHF. Selon la figure 4.5, nous constatons que la perfor-

mance de notre méthode atteint son maximum lorsque le seuil est égal à 0.17, 0.15 et 0.2 pour les problèmes Iris, Glass et Lettre respectivement. Ces valeurs représentent les valeurs minimales de la matrice de fermeture transitive. Les résultats obtenus montrent que le choix du seuil de similarité à partir de la matrice de fermeture transitive Min-Max améliore la performance de notre méthode SVM-CHF.

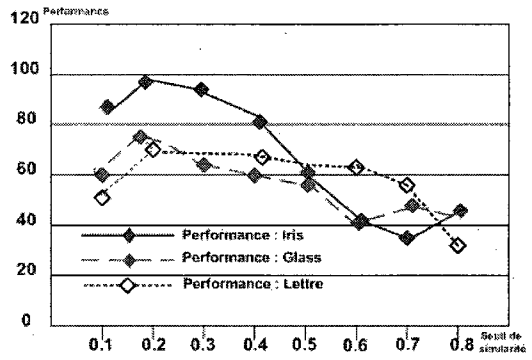


Figure 4.5 – Influence du seuil de similarité sur la performance de SVM-CHF

4.3.3 Mesure de performance de SVM-CHF-Text

Dans nos expérimentations, nous avons effectué une comparaison avec trois techniques standards de la catégorisation de textes : SVM [45], K-plus-proche-voisin (K-NN) [2] et les arbres de décision [95]. Pour évaluer la performance de notre méthode, nous avons utilisé les trois critères précision, rappel et F-mesure.

Dans un premier temps, nous avons comparé SVM-CHF-Text avec le SVM de base sur la collection Reuters-21578. Les résultats sont résumés dans le tableau 4.18. En plus, et sur la même collection Reuters-21578, nous avons comparé SVM-CHF-Text aux techniques K-NN et arbres de décision. Les résultats sont résumés dans le tableau 4.19.

Dans un deuxième temps, nous avons comparé notre méthode SVM-CHF-Text avec le SVM de base sur la collection 20 Newsgroups. Nous avons évalué notre méthode en utilisant la fonction polynomiale et RBF. Pour SVM-CHF-Text, la fonction RBF ($\gamma = 0.6$) donne de meilleurs résultats sur la collection **20NG**. Pour le SVM de base, nous avons testé sa performance en utilisant les deux fonctions de noyau suivantes : la fonction polynomiale (d=2,3,...,8) et RBF ($\gamma = 0.1, 0.2, ..., 1$). Nous avons sélectionné uniquement les meilleurs

résultats du SVM. Les résultats sont exprimés dans le tableau 4.20. En outre, le tableau 4.21 résume nos résultats avec les techniques les plus connues dans le domaine de la catégorisation des textes, à savoir Naive Bayes, Rochio, C4.5 et K-NN. Pour l'algorithme Rochio, la performance est obtenue avec $\beta = 0.5$. Pour l'algorithme K-NN, la performance est atteinte avec $k = 35$. Dans tous les cas, notre méthode démontre une meilleure performance. Cette performance est due essentiellement à la préparation de données d'apprentissage et de test lors de la phase de la catégorisation.

Tableau 4.18 – SVM-CHF-Text *versus* SVM de base (Reuters-21578)

SVM-CHF-Text *versus* SVM (Reuters-21578)			
Classes	$SVM_{Poly:d=4}$	$SVM_{RB:\gamma=0.8}$	SVM-CHF-Text
Earn	98.4	98.5	**98.7**
Acq	95.2	95.3	**99.2**
Money-fx	74.9	75.4	**100**
Grain	91.3	91.9	**94**
Crude	88.6	89.0	**95.4**
Trade	77.3	78.0	**98**
Interest	73.1	75.0	**96.1**
Ship	86.5	86.5	**98.8**
Wheat	85.9	85.9	**87.4**
Corn	85.7	85.7	**88.6**
μ	86.2	86.5	**96.6**

Tableau 4.19 – SVM-CHF-Text *versus* K-NN et arbres de décision (Reuters-21578)

Classes	K-NN			Arbre de décision			SVM-CHF-Text		
	Rappel	Précision	F_1	Rappel	Précision	F_1	Rappel	Précision	F_1
Earn	95	92	94	99	97	98	**98**	**98.7**	**98**
Acq	100	91	95	96	95	96	**99**	**99.2**	**99**
Money-fx	92	65	76	77	76	76	**99**	**100**	**99**
Grain	96	70	81	95	92	93	**98**	**94**	**96**
Crude	82	75	78	93	85	89	**92**	**95.4**	**94**
Trade	89	66	76	81	70	75	**95**	**98**	**97**
Interest	80	71	75	65	83	73	**98**	**96.1**	**97**
Ship	85	77	81	76	86	81	**96**	**98.8**	**98**
Wheat	69	73	71	97	83	89	**88**	**87.4**	**88**
Corn	35	76	48	98	82	98	**94**	**88.6**	**91**
μ	84.2	85.1	85.9	86.2	85.9	86.4	**96**	**95.6**	**95**

Tableau 4.20 – SVM-CHF-Text *versus* SVM de base (collection **20NG**)

	$SVM_{Poly:d=4}$			$SVM_{RB:\gamma=0.6}$			SVM-CHF-Text		
	Rappel	Précision	F_1	Rappel	Précision	F_1	Rappel	Précision	F_1
μ	86.75	85.9	86.29	85.4	85.6	85.5	**91.31**	**91.27**	**91.29**

Tableau 4.21 – SVM-CHF-Text *versus* les méthodes de catégorisation de textes (**20NG**)

	Bayes	Rochio	C4.5	K-NN	SVM-CHF-Text
Rappel	72.4	78	83.4	76.4	**91.31**
Précision	74.7	81.2	85	79.8	**91.27**
F-Mesure	73.53	79.57	84.19	78.06	**91.29**

4.4 Conclusion

Ce chapitre est subdivisé en deux parties. Dans la première partie, nous avons présenté deux cas pratiques pour illuster le déroulement de notre méthode. Des données numériques et textuelles ont été utilisées. La deuxième partie a été consacrée aux mesures de la performance de SVM-CHF et SVM-CHF-Text. Notre méthode a été comparée aux techniques principales de la classification et la catégorisation des textes. Nous avons étudié l'influence du seuil de similarité sur la performance de SVM-CHF. Notre méthode tire ses avantages de : (i) l'utilisation de la mesure de similarité pour extraire la similarité entre les objets ; (ii) la fermeture transitive pour trouver le chemin le plus court entre les objets ou les documents ; (iii) l'application du SVM à chaque niveau de la hiérarchie.

Conclusion

Dans notre travail, nous nous somme intéressés à la résolution du problème multi-classes basée sur la machine à vecteur de support (SVM). Dans la littérature, plusieurs recherches ont été réalisées pour faire face à ce genre de problème qui reste jusqu'à présent un domaine de recherche ouvert. Pour résoudre ce problème, plusieurs méthodes ont été développées : «Un-contre-un», «Un-contre-tous», DAGSVM et DHSVM.

Afin d'éclaicir nos apports, nous avons organisé ce mémoire autour de trois parties dont la première consiste à situer le lecteur dans le contexte de la classification. La deuxième partie a été consacrée aux détails du classificateur SVM avec la description de travaux récents relatifs à la classification de données numériques et à la catégorisation de textes. Dans la troisième partie, nous avons détaillé notre méthode qui vise l'amélioration de la phase d'apprentissage du SVM. L'idée de base consiste, d'une part, à simplifier la phase d'apprentissage en rendant les exemples linéairement séparables par la définition d'un chemin minimal entre les exemples à l'aide de la fermeture transitive Min-Max et d'autre part, à diminuer le nombre de SVM obtenus tout au long de la hiérarchie.

Dans le cadre de ce projet, nous avons proposé une nouvelle méthode de classification, basée sur le SVM, pour traiter les problèmes multi-classes. Notre méthode SVM-CHF consiste à construire dynamiquement une structure hiérarchique floue à partir des données d'apprentissage. Notre méthode s'articule autour de trois principaux concepts : la classification hiérarchique, la logique floue et le SVM. Elle tire ses avantages de la combinaison des techniques suivantes : (i) la classification par partition ; (ii) la classification hiérarchique ; (iii) la classification floue et (iv) la classification basée sur le SVM. De plus, la structure hiérarchique obtenue par SVM-CHF nous permet d'accomplir les tâches de classification et de *clustering* en même temps. La tâche de classification s'effectue dans le sens descendant et la tâche de *clustering* s'effectue dans le sens ascendant.

Étant donné le volume élevé de données textuelles circulant dans les réseaux informatiques, nous avons adapté notre méthode SVM-CHF afin de traiter le problème de la catégorisation de textes. La nouvelle méthode adaptée est intitulée SVM-CHF-Text. La structure hiérarchique de SVM-CHF-Text permet la conversion du problème original en sous-problèmes binaires simples à résoudre ainsi que la classification de nouveaux documents.

La nouvelle méthode proposée a pour but d'améliorer la performance du classificateur SVM à l'aide de la diminution du taux d'erreurs de classification ainsi que le temps d'apprentissage. Notre méthode SVM-CHF ainsi que sa version adaptée SVM-CHF-Text ont été développées également pour simplifier la recherche et la classification des objets et des documents dans les bases de données de grande dimension.

Pour appuyer notre méthode, nous avons effectué des tests d'évaluation sur deux types différents de bases de données. Le premier type concerne les données numériques et le deuxième type concerne les données textuelles. Les résultats obtenus ont démontré que notre méthode est performante.

Un article intitulé "**SVM Fuzzy Hierarchical Classification Method for multi-class problems**", qui décrit la première partie de ce travail, a été publié dans la conférence internationale : *The IEEE* 23^{rd} *Advanced Information Networking and Applications (AINA 2009)*[33].

Un article intitulé "**A new fuzzy hierarchical classification based on SVM for text categorization**", qui décrit la deuxième partie de ce travail, a été publié dans la conférence internationale : *International Conference on Image Analysis and Recognition (ICIAR 2009)*[32].

En outre, nous avons proposé un nouveau modèle de recherche basé sur la logique floue pour la recherche de documents pertinents à une requête lancée par l'utilisateur. Ce modèle flou consiste à faire ressortir les documents qui répondent à une requête en utilisant la notion du gain d'information dans la hiérarchie.

Un article intitulé "**SVM fuzzy hierarchical document categorization and retrieval method**", qui décrit cette partie a été publié dans la conférence internationale : *The* 5^{th} *International Conference on Data Mining (DMIN'09)*[34].

Un article intitulé "**A multi-class method using fuzzy hierarchical structure to train SVM**", qui synthétise le travail de ce mémoire, a été soumis à la revue : *International*

Journal of Intelligent Computing and Cybernetics [31].

Une extension de ce travail consiste à enrichir notre méthode par la création d'une nouvelle fonction de noyau dynamique dans le but de traiter automatiquement la classification des objets ainsi que la catégorisation des documents.

Une autre direction de recherche consiste à adapter notre méthode pour traiter les séquences vidéos dans les bases de données de grande dimension afin de faire ressortir la similarité floue existante entre ces séquences.

Annexe A

Loi de Zipf

La loi de Zipf stipule que les termes les plus informatifs d'un corpus de documents ne sont ni les mots qui apparaissent le plus dans le corpus ni ceux les moins fréquents. La figure A.1 illustre de manière graphique la loi de Zipf.

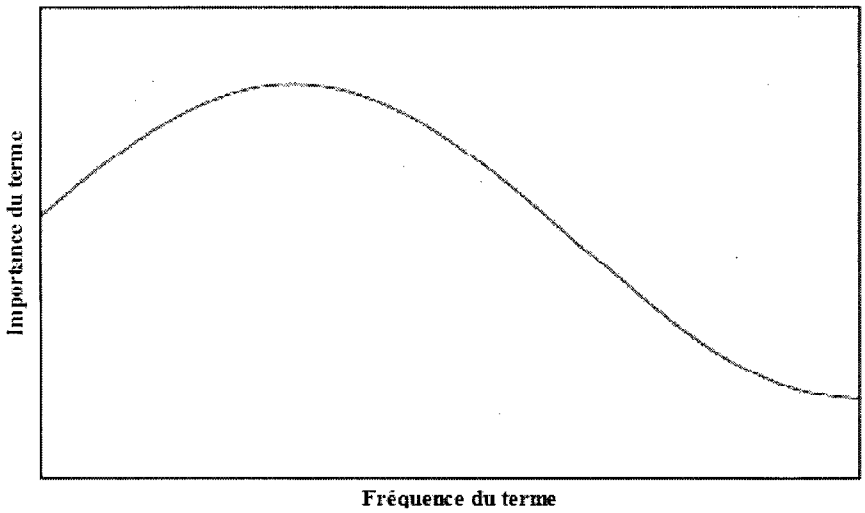


Figure A.1 – Loi de Zipf.

Annexe B

Séparation de la fonction XOR

La fonction **XOR** dans un espace à deux dimensions n'est pas séparable et pour qu'elle soit séparable, il faut une application de $\mathbf{R^2}$ vers $\mathbf{R^3}$ (figure B.1).

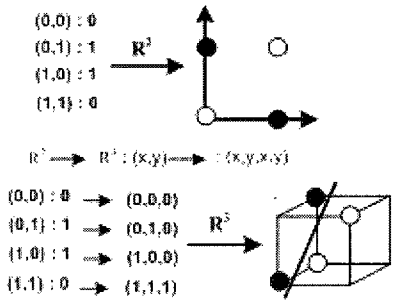


Figure B.1 – Transformation de la fonction XOR de $\mathbf{R^2}$ vers $\mathbf{R^3}$.

Bibliographie

[1] « http ://www.ga.gov.au/anzmeta/sgmlwhy.html ».

[2] K. AES et L. EIKVIL. « Text categorization : a survey ». Report No 941, Norwegian Computing Center, pages 1–38, 1999.

[3] C. AGGARWAL, C. PROCOPIUS, J. WOLF, P. YU et J. PARK. « Fast algorithms for projected clustering ». Proceedings of the ACM SIGMOD Conference, Philadelphia, PA, pages 61–72, 1999.

[4] C. AGGARWAL et P. YU. « Finding generalized projected clusters in high dimensional spaces ». Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pages 70–81, 2000.

[5] R. AGRAWAL, J. GEHRK, D. GUNOPULOS et P. RAGHAVAN. « Automatic subspace clustering of high dimensional data for data mining applications ». Proceedings of the ACM SIGMOD Conference, Seattle, WA, pages 94–105, 1998.

[6] F. AIOLLI et A. SPERDUTI. « Multi-class classification with multi-prototype support vector machines ». Journal of Machine Learning Research, 6 :817–850, 2005.

[7] M. ANKERST, H. KRIEGEL et J. SANDER. « OPTICS : ordering points to identify the clustering structure ». In Proceedings of the ACM SIGMOD Conference, Philadelphia, PA, USA, pages 49–60, 1999.

[8] P. BALDI, P. FRASCONI et P. SMYTH. Modeling the internet and the Web : probabilistic methods and algorithms. John Wiley and Sons, 2003.

[9] L. BAOLI, Y. SHIWEN et L. QIN. « An improved K-nearest neighbor algorithm for text categorization ». Proceeding of the 20^{th} International Conference on Computer Processing of Oriental Languages, China, 2003.

[10] F. BELLOMI et M. CRISTANI. « Supervised document classification based upon domain-specific term taxonomies ». International J.Metadata, Semantics and Ontologies, pages 37–46, 2006.

[11] K. BENABDESLEM. « Descendant hierarchical support vector machine for multi-class problems ». International Joint Conference on Neural Networks, Vancouver, BC, Canada, July 2006.

[12] J. BENZÉCRI. « Rappel : construction d'une classification ascendante hiérarchique par la recherche en chaîne de voisins réciproques ». Les cahiers de l'analyse des données, 22(2) :191–198, 1997.

[13] J. BEZDEK et J. DUNN. « A heuristic for estimating the parameters in a mixture of normal distibutions ». IEEE Transactions on Computers, 24(8) :835–838, 1975.

[14] C. BLAKE et C. MERZ. « UCI repository of machine learning databases. Available at ftp.ics.uci.edu/pub/machine-learning-databases ». 1998.

[15] M. BOUGUESSA. « Une approche objective pour déterminer le nombre de clusters dans le cadre de la classification floue non supervisée ». Mémoire de maîtrise, Université de sherbrooke, Département d'informatique, Québec, Canada, 2005.

[16] C. CAMPBELL, N. CRISTIANINI et A. SMOLA. « Query learning with large margin classifiers ». Proceedings of ICML-00, pages 111–118, 2000.

[17] D. CASASENT et Y. WANG. « A hierarchical classifier using new support vector machines for automatic target recognition ». Neural Networks, 18 :541–548, 2005.

[18] D. COHN, L. ATLAS et R. LADNER. « Improving generalization with active learning ». Machine Learning, (15) :201–221, 1994.

[19] T. COVER et P. HART. « Nearest neighbour pattern classification ». IEEE Transactions on Information Theory, 13(1) :1179–1184, 1967.

[20] M. DASH et H. LIU. « Feature selection for classification ». Intelligent Data Analysis, 1(3) :131–156, 1997.

[21] S. DEERWESTER, S. DUMAIS, T. LANDAUER, G. FURNAS et R. HARSHMAN. « Introduction to modern information retrieval ». Journal of the American Society of Information Science, 416(6) :391–407, 1990.

[22] L. DENOYER. « Apprentissage et inférence statistique dans les bases de documents structurés : application aux corpus de documents textuels ». Thèse de doctorat, Université Paris 6, Paris, France, 2004.

[23] E. DIDAY et J. SIMON. « Clustering analysis ». Digital Pattern Recognition, NW, USA, 10 :47–94, 1976.

[24] R. DUDA et P. HART. Pattern classification and scene analysis. Wiley, 1973.

[25] S. DUMAIS et H. CHEN. « Hierarchical classification of web content ». Proceedings of the 23^{rd} Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Athens, Greece, pages 256–263, 2000.

[26] J. DUNN. « A fuzzy relative of the ISODATA process and its use in detecting compact, well-separated clusters ». Journal of Cybernetics, 3(3) :32–57, 1973.

[27] J. DY et C. BRODLEY. « Feature subset selection and order identification for unsupervised learning ». In Proceedings of 17^{th} International Conference on Machine Learning, San Francisco, California, USA, pages 247–254, 2000.

[28] M. ESTER, H. KRIEGEL, J. SANDER et X. XU. « A density-based algorithm for discovering clusters in large spatial databases with noise ». In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, pages 226–231, 1996.

[29] S. FOLIGUET. « Segmentation d'images en régions floues ». Rencontres francophones sur la logique floue et ses applications, LFA 2000, La Rochelle, pages 189–196, 2000.

[30] V. GANTI, J. GEHRKE et R. RAMAKRISHNAN. « CACTUS-clustering categorical data using summaries ». In Proceedings of the 5^{th} ACM SIGKDD, San Diego, CA, pages 73–83, 1999.

[31] T. GUERNINE et K. ZEROUAL. « A multi-class method using fuzzy hierarchical structure to train support vector machine ». Submitted to International Journal of Intelligent Computing and Cybernetics, October 2009.

[32] T. GUERNINE et K. ZEROUAL. « A new fuzzy hierarchical classification based on SVM for text categorization ». International Conference on Image Analysis and Recognition (ICIAR 2009), Halifax, Canada, pages 865–874, July 2009.

[33] T. GUERNINE et K. ZEROUAL. « SVM fuzzy hierarchical classification method for multi-class problems ». The IEEE 23^{rd} International Conference on Advanced Information Networking and Applications (AINA-2009), pages 691–696, may 2009.

[34] T. GUERNINE et K. ZEROUAL. « SVM fuzzy hierarchical document categorization and retrieval method ». The 5^{th} International Conference on Data Mining (DMIN'09), Las Vegas, Nevada, USA, pages 441–447, July 2009.

[35] S. GUHA, R. RASTOGI et K. SHIM. « An efficient clustering algorithm for large databases ». International Conference on Management of Data (SIGMOD 1998), Seattle, Washington, USA, pages 73–84, June 1998.

[36] S. GUHA, R. RASTOGI et K. SHIM. « ROCK : A robust clustering algorithm for categorical attributes ». In Proceedings of the 15^{th} ICDE, Sydney, Australia, pages 512–521, 1999.

[37] J. HAN et M. KAMBER. Data mining : concepts and techniques. Morgan Kaufmann Publishers, 2001.

[38] E. HANG, G. KARYPIS et V. KUMAR. « Text categorization using weight adjusted K-nearest neighbour classification ». Proceeding of the 5^{th} Pacific-Asia Conference on knowledge discovery and Data Mining, 2035 :53–65, 2001.

[39] P. HAO, J. CHIANG et Y. TU. « Hierarchically SVM classification based on support vector clustering method and its application to document categorization ». Elsevier, 33 :627–635, 2007.

[40] M. HASAN et F. BORIS. « Machines à vecteurs de support ou séparateurs à vastes marges ». BD Web, ISTY3, France, 2006.

[41] C. HSU et C. LIN. « A comparison of methods for multi-class support vector machines ». IEEE Trans Neural Networks, 13 :415–425, June 2002.

[42] A. JAIN, M. MURTY et P. FLYNN. « Data clustering : a review ». ACM Computing Surveys, September 1999.

[43] A. JAIN et D. ZONGKER. « Feature selection : evaluation, application and small sample performance ». IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(2) :152–157, 1997.

[44] T. JOACHIMS. « A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization ». In International Conference on Machine Learning (ICML), pages 143–151, 1997.

[45] T. JOACHIMS. « Text categorization with support vector machines : learning with many relevant features ». Proceedings of the 10^{th} European conference on machine learning, pages 137–142, 1998.

[46] T. JOACHIMS. « Transductive inference for text classification using support vector machine ». In proceedings of 16^{th} International Conference on Machine Learning (ICML 1999), San Francisco, USA, pages 200–209, 1999.

[47] T. JOACHIMS. Learning to classify text using support vector machines : methods, theory and algorithms. Kluwer Academic. University of Rome Tor Vergata, 2002.

[48] L. KAHSAY, F. SCHENKER et G. PALM. « Comparaison of multi-class SVM decomposition schemes for visual object recognition ». Lecture Notes in Computer Science, 3663 :334–341, 2005.

[49] G. KARYPIS, E. HAN et V. KUMAR. « CHAMELEON : a hierarchical clustering algorithm using dynamic modeling ». IEEE Computer, 32(8) :68–75, 1999.

[50] A. KAUFFMANN. Introduction to theory of fuzzy subsets. Academic, New York, USA, 1975.

[51] L. KAUFMAN et P. ROUSSEEUW. « Finding groups in data : an introduction to cluster analysis ». John Wiley and Sons, New York, USA, 1990.

[52] A. KELAIAIA. « Classification non supervisée de textes arabes appliquée à la recherche documentaire ». Mémoire de maîtrise, Université Guelma, Algérie, 2007.

[53] D. KOLLER et M. SAHAMI. « Hierarchically classifying documents using very few words ». Proceedings of the 14^{th} International Conference on Machine Learning (ML-97), Nashville, Tennessee, pages 170–178, 1997.

[54] U. KRE. « Pairwise classification and support vector machine ». Advances in Kernel Methods-Support Vector Learning, MIT Press, 12 :185–208, 1999.

[55] T. LAROSE. Des données à la connaissance : une introduction au data mining. Vuibert Informatique, Paris, France, 2005.

[56] A. LEHMAM et P. BOUVET. « Évaluation, rectification et pertinence du résumé automatique de texte pour une utilisation en réseaux internet et intranet ». $3^{ème}$ colloque du chapitre français de l'ISKO, Paris, France, pages 111–124, 2001.

[57] D. LEWIS. « Naive Bayes at forty the independence assumption in information retrieval ». Proceedings of ECML-98, 10^{th} European Conference on Machine Learning, pages 4–15, 1998.

[58] D. LEWIS et M. RINGUETTE. « A comparison of two learning algorithms for text categorization ». In Third annual symposium on document analysis and information retrieval (SDAIR 1994), pages 81–93, 1994.

[59] D. LI et Y. FANG. « An algorithm to cluster data for efficient classification of support vector machines ». Expert Systems with Applications : An International Journal, 34(3) :2013–2018, 2008.

[60] P. LINGRAS et C. BUTZ. « Rough set based 1-v-1 and 1-v-r approaches to support vector machine multi-classification ». Elsevier, 177 :3782–3798, 2007.

[61] S. LIU, H. YI, L. CHIA et D. RAJAN. « Adaptive hierarchical multi-class SVM classifier for texture-based image classification ».

[62] T. LIU, Y. YANG, H. WAN, H. ZENG, Z. CHEN, et W. MA. « Support vector machines classification with a very large-scale taxonomy ». SIGKDD Explorations, 7 :36–43, 2005.

[63] H. LODHI, C. SAUNDERS, J. TAYLOR, N. CRISTIANINI et C. WATKINS. « Text classification using string kernels ». Journal of Machine Learning Research, pages 419–444, 2002.

[64] A. LUCA et S. TERMINI. « Entropy of L-fuzzy sets ». Academic Press Inc, 24 :55–73, 1974.

[65] D. MANNING, P. RAGHAVAN et H. SCHUTZE. An introduction to information retrieval. Cambridge UP, Cambridge University, England, 1998.

[66] J. MCQUEEN. « Some methods of classification and analysis of multivariate observations ». In Proceedings of 5^{th} Berkley Symposium on Mathematical Statistics and Probability, University of California, pages 281–297, 1967.

[67] A. MEGHAOUI. « Exploitation des arbres fréquents de dépendance pour la représentation et la classification automatique de textes ». Mémoire de maîtrise, Université de sherbrooke, Département d'informatique, Québec, Canada, 2008.

[68] R. MICHALSKI, R. STEPP et E. DIDAY. « Automated construction of classifications : conceptual clustering versus numerical taxonomy ». IEEE Transactions on Pattern Analysis and Machine Intelligence, 5(4) :396–409, 1983.

[69] J. NAKACHE et J. CONFAIS. Approche pragmatique de la classification. Technip, Paris, France, 2005.

[70] J. NKOGHE, K. ZEROUAL et W. SHENGRUI. « Specification reuse : a fuzzy approach ». International Joint Conference on Artificial Intelligence, 1995.

[71] X. PENG et B. CHOI. « Automatic web page classification in a dynamic and hierarchical way ». IEEE International Conference on Data Mining, pages 386–393, 2002.

[72] J. PLATT, N. CRISTIANINI et J. SHAWE-TAYLOR. « Large margin DAG's for multi-class classification ». Advances in Neural Information Processing Systems, Cambridge, MA : MIT Press, 12 :547–553, 2000.

[73] R. RIFKIN et A. KLAUTAU. « In defence of one-vs-all classification ». Journal of Machine Learning Research, 5 :101–141, 2004.

[74] G. SALTON et C. BUCKLEY. « Term-weighting approaches in automatic text retrieval ». Information Processing and Management, 24(5) :513–523, 1988.

[75] J. SANDER, M. ESTER, H. KRIEGEL et X. XU. « Density-based clustering in spatial databases : the algorithm GDBSCAN and its application ». In Data Mining and Knowledge Discovery, pages 169–194, 1998.

[76] G. SCHOHN et D. COHN. « Less is more : active learning with support vector machines ». Proceedings of ICML-00, pages 839–846, 2000.

[77] H. SCHUTZE, D. HULL et J. PEDERSEN. « A comparison of classifiers and document representations for the routing problem ». The 18^{th} ACM International Conference on Research and Development in Information Retrieval, New York, USA, pages 229–237.

[78] F. SEBASTIANI. « Machine learning in automated text categorization ». ACM Computing Surveys, 34 :1–47, 2002.

[79] G. SHEIKHOLESLAMI, S. GHATTERJEE et A. ZHANG. « WaveCluster : a multi-resolution clustering approach for very large spatial databases ». In Proceedings of the 24^{th} Conference on VLDE, New York, NY, pages 428–439, 1998.

[80] Y. SHI, Y. SONG et A. ZHANG. « A shrinking-based approach for multi-dimensional data analysis ». In Proceeding of the 29^{th} VLDB Conference, Berlin, 2003.

[81] F. SHWENKER et G. PALM. « Tree structured support vector machines for multi-class pattern recognition ». Lecture Notes in Computer Science, 2096 :409–417, 2001.

[82] K. SOMAN, R. LOGANATHAN, M. VIJAYA, V. AJAY et K. SHIVSUBRAMANI. « Fast single shot multi-class proximal support vector machines and perceptrons ». Proceedings of the International Conference on Computing : Theory and Applications (ICCTA), Kolkata, India, pages 294–298, 2007.

[83] M. STEINBACH, G. KARYPIS et V. KUMAR. « A comparison of document clustering techniques ». Technical Report No : 00-034, Department of Computer Science and Egineering, University of Minnesota, 2000.

[84] A. STREHL, J. GHOSH et R. MOONEY. « Impact of similarity measures on web-page clustering ». In Proceedings of the 17^{th} National Conference on Artificial Intelligence : Workshop of Artificial Intelligence for Web Search (AAAI 2000), Austin, Texas, USA, pages 58–64, 2000.

[85] S. SZEDMAK et J. SHAWE-TAYLOR. « Multi-class learning at one-class complexity ». Technical Report No : 1508, School of Electronics and Computer Science, UK, 2005.

[86] P. TAN, M. STEINBACH et V. KUMAR. Introduction to data mining. Addison-Wesley, 2006.

[87] T. TERANO, K. ASAI et M. SUGENO. Fuzzy systems theory and its application. Academic Press, 1992.

[88] S. TONG. « Active learning : theory and applications ». Thèse de doctorat, Stanford University, 2001.

[89] S. TUFFERY. Data mining et statistique décisionnelle : l'intelligence des données. Technip, Paris, France, 2007.

[90] M. VALDIVIA, L. LOPEZ et M. VEGA. « The learning vector quantization algorithm applied to automatic text classification tasks ». ACM Computing Surveys, 20 :748–756, 2007.

[91] V. VAPNIK. « The nature of statistical learning theory ». Springer Verlag, New York, 1995, 1995.

[92] V. VAPNIK. Statistical learning theory. John Wiley and Sons, 1998.

[93] T. WANG et H. CHIANG. « Fuzzy support vector machine for multi-class text categorization ». Information Processing and Management, 43 :914–929, 2007.

[94] W. WANG, J. YANG et R. MUNTZ. « STING : a statistical information grid approach to spatial data mining ». In Proceedings of the 23^{th} Conference on VLDB, pages 186–195, 1997.

[95] S. WEISS, C. APTE, F. DAMERAU, D. JOHNSON, F. OLES, H. GOETZ et AL. « Maximizing text mining performance ». IEEE Intelligent Systems, 14 :2–8, 1999.

[96] B. XIAOJUAN, S. QILONG, S. HAO et T. XUYAN. « Compression method based on training dataset of SVM ». Journal of Systems Engineering and Electronics, 19(1) :198–201, 2008.

[97] X. XIE et G. BENI. « A validity measure for fuzzy clustering ». IEEE Transactions Pattern analysis and Machine Intelligence, 13 :841–847.

[98] L. ZADEH. « Fuzzy sets : information and control ». Elsevier, 8 :338–353, June 1965.

[99] L. ZADEH. « Knowledge representation in fuzzy logic ». IEEE Transaction on Knowledge and Data Engineering, 1(1) :89–100, 1989.

[100] G. ZHANG. « Support vector machine with huffman tree architecture for multi-class classification ». Lecture Notes in Computer Science, 3773 :24–33, 2005.

[101] G. ZHANG et W. JIN. « Automatic construction algorithm for multi-class support vector machines with binary tree architecture ». IJCSNS International Journal of Computer Science and Network Security, 6(2A) :119–126, February 2006.

[102] T. ZHANG, R. RAMAKRISHNAN et M. LIVNY. « A new data clustering algorithm and its applications ». Data Mining and knowledge Discovery, pages 141–182, 1997.