Sur l'utilisation active de la diversité dans la construction d'ensembles de classifieurs. Application à la détection de fumées nocives sur site industriel

David Gacquer

Université de Valenciennes et du Hainaut-Cambrésis

École Doctorale Sciences de l'Ingénieur, Mathématiques, Humaines, Juridiques et de Gestion
Institut des Sciences et Techniques de Valenciennes

Sur l'utilisation active de la diversité dans la construction d'ensembles de classifieurs. Application à la détection de fumées nocives sur site industriel

THÈSE

présentée et soutenue publiquement le 5 décembre 2008

pour l'obtention du

Doctorat de l'université de Valenciennes et du Hainaut-Cambrésis

(mention informatique)

par

David GACQUER

Composition du jury

Rapporteurs :	Eric Gaussier, Professeur à l'Université Joseph Fourier (Grenoble 1) Philippe Leray, Professeur à l'Université de Nantes
Examinateurs :	Jean-Paul Haton, Professeur à l'Université Henri Poincaré (Nancy 1) Benjamin Quost, Maître de Conférences à l'Université de Technologie de Compiègne
Directeur :	Sylvain Piechowiak, Professeur à l'Université de Valenciennes
Co-Directeurs :	Véronique Delcroix, Maître de Conférences à l'Université de Valenciennes François Delmotte, Maître de Conférences HDR à l'Université de Valenciennes

Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines
UMR CNRS 8530

LAMIH
LABORATOIRE D'AUTOMATIQUE DE MECANIQUE ET D'INFORMATIQUE INDUSTRIELLES ET HUMAINES

cnrs
dépasser les frontières

Mis en page avec la classe thloria.

Université de Valenciennes et du Hainaut-Cambrésis

École Doctorale Sciences de l'Ingénieur, Mathématiques, Humaines, Juridiques et de Gestion
Institut des Sciences et Techniques de Valenciennes

Sur l'utilisation active de la diversité dans la construction d'ensembles de classifieurs. Application à la détection de fumées nocives sur site industriel

THÈSE

présentée et soutenue publiquement le 5 décembre 2008

pour l'obtention du

Doctorat de l'université de Valenciennes et du Hainaut-Cambrésis

(mention informatique)

par

David GACQUER

Composition du jury

Rapporteurs :	Eric Gaussier, Professeur à l'Université Joseph Fourier (Grenoble 1) Philippe Leray, Professeur à l'Université de Nantes
Examinateurs :	Jean-Paul Haton, Professeur à l'Université Henri Poincaré (Nancy 1) Benjamin Quost, Maître de Conférences à l'Université de Technologie de Compiègne
Directeur :	Sylvain Piechowiak, Professeur à l'Université de Valenciennes
Co-Directeurs :	Véronique Delcroix, Maître de Conférences à l'Université de Valenciennes François Delmotte, Maître de Conférences HDR à l'Université de Valenciennes

Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines
UMR CNRS 8530

LAMIH
LABORATOIRE D'AUTOMATIQUE DE MECANIQUE ET D'INFORMATIQUE INDUSTRIELLES ET HUMAINES

cnrs
dépasser les frontières

Mis en page avec la classe thloria.

Remerciements

Je tiens tout d'abord à remercier mes encadrants, Sylvain Piechowiak, Véronique Delcroix et François Delmotte, pour m'avoir fait profiter de leur sagesse et de leur expérience durant ces trois années de thèse et qui ont montré de manière expérimentale qu'un ensemble d'encadrants diversifié permettait, au même titre qu'un ensemble de classifieurs divers, d'améliorer la qualité de l'apprentissage (du doctorant en l'occurence). Je remercie également Philippe Leray et Eric Gaussier pour avoir rapporté cette thèse, ainsi que Jean-Paul Haton et Benjamin Quost pour avoir accepté d'en être les examinateurs. Je remercie également Philippe Bourrier, directeur de la société Aloatec, pour avoir initié les travaux de recherche présentés dans ce mémoire.

J'adresse également un remerciement particulier à René Mandiau pour m'avoir initié à la rédaction de documents en LaTeXet qui m'a prouvé à maintes reprises que l'investissement initial destiné à la prise en main de ce langage était très vite rentabilisé aussi bien en terme de temps qu'en terme de qualité.

Cette thèse n'aurait pu être menée à son terme sans le soutiens de mes $N \rightarrow +\infty$ amis et collègues de l'Université de Valenciennes, dont la liste est trop longue pour être parcourue de manière exhaustive mais qui se reconnaitront très certainement. Un grand merci donc à mes amis du Bureau 210, du Bâtiment Jonas ainsi qu'à ceux du Bâtiment LAMIH-ROI, dont l'isolement géographique ne nuit en rien à la chaleur de leur accueil.

Enfin, je tiens à remercier mes parents, pour m'avoir accompagné durant ces longues années de travail, ainsi que ma *Grande Soeur*, pour m'avoir fait partager son expérience et ses connaissances de doctorante et qui a toujours su me réconforter dans les moments difficiles, par son écoute, ses conseils mais également ses talents culinaires.

Cette thèse a été réalisée grâce au soutien du Campus International sur la Sécurité et l'Intermodalité des Transports, de la Région Nord Pas de Calais, de la Communauté Européenne, de la Délégation Régionale à la Recherche et à la Technologie, du Ministère de l'Enseignement Supérieur et de la Recherche et du Centre National de la Recherche Scientifique.

So, yes, we know a lot
and, no, we don't have the all-explaining theory.
This makes our field of research what it is
challenging and entertaining.
L. I. Kuncheva

Table des matières

Table des figures

1

Introduction

1.1 Le contrôle de la pollution atmosphérique

La préservation de l'environnement est devenue une question récurrente ces dernières années. Qu'il s'agisse de la réduction de l'émission des gaz à effet de serre, du recyclage des déchets, ou du retraitement des eaux usées, toutes ces activités font l'objet de réglementations à l'échelle nationale ou internationale. En France, c'est le code de l'environnement, un ouvrage comprenant un ensemble de textes juridiques relatifs aux droits de l'environnement, qui réglemente l'élimination et le recyclage des déchets, l'utilisation de produits chimiques biocides ou plus particulièrement, dans le cas qui nous intéresse ici, la mise en place du principe communément appelé du *pollueur-payeur*. Ce principe consiste à faire prendre en compte par chaque acteur économique ou industriel les conséquences négatives de son activité en lui imputant les coûts associés à la lutte contre la pollution. Ce principe fait partie des mesures essentielles sur lesquelles est basée la politique environnementale des pays développés, mais son application repose notamment sur la possibilité de quantifier et chiffrer ces conséquences négatives pour l'environnement. Pour pouvoir mettre en pratique ce principe, il est donc nécessaire de procéder à un contrôle des activités présentant un risque pour l'environnement.

Nous nous intéressons plus particulièrement à la pollution atmosphérique, une forme de pollution correspondant à une altération de la pureté de l'air. Cette altération est provoquée par une ou plusieurs substances ou particules présentes à des concentrations et durant des temps suffisants pour créer un effet toxique présentant des risques pour l'environnement et la population. Les manifestations les plus courantes de ce phénomène sont les risques d'alertes relayées par les médias concernant la pollution de l'air, dans des zones à forte densité urbaine ou pour des régions où l'activité industrielle est très importante. Ces alertes surviennent plus particulièrement en cas de fortes chaleurs, qui ont tendance à accentuer ce phénomène. Des concentrations trop fortes de polluants atmosphériques et notamment d'ozone ou de dioxyde de souffre peuvent entraîner des manifestations pathologiques sur la population : conjonctivites, rhinite, toux, essoufflements, voire malaises dans certains cas. Certaines catégories de la population y sont davantage sensibles, comme les enfants, les personnes âgées ou les personnes souffrant de problèmes respiratoires, la pollution de l'air abaissant le seuil de déclenchement des crises chez les asthmatiques.

Ces alertes sont en grande majorité provoquées par les rejets de l'industrie et de l'activité humaine : les industries de la chimie et de la pétrochimie notamment rejettent dans l'air de nombreux types de produits, résidus de processus de transformation. Les installations du secteur de la sidérurgie et de la métallurgie émettent également de nombreux polluants en grande quantité, notamment dans des processus de combustion incomplète qui sont fréquents dans les cokeries, ou

de refonte de matériaux comme les aciéries électriques. Pour prévenir la population des risques encourus lorsque survient une alerte, il existe des observatoires dont la principale tâche consiste à contrôler et à recenser les activités des sites industriels jugés à risque. S'il est commun d'avoir recours de manière intuitive à des disciplines issues de la chimie ou de la biologie pour réaliser cette activité de contrôle, l'emploi de méthodes appartenant au traitement d'image et à l'informatique pour répondre à cette problématique est beaucoup moins répandu. C'est pourtant sous cet angle que cette thèse aborde le problème de la pollution atmosphérique.

Les travaux de recherche présentés dans ce manuscrit concernent l'exploitation d'outils issus de la reconnaissance des formes et de l'apprentissage automatique pour la détection et la classification de phénomènes locaux et ponctuels de pollution atmosphérique. Nous assimilons le rejet des fumées industrielles à un problème d'apprentissage automatique et de classification de scènes à risque, à partir de données non pas mesurées par des analyses chimiques classiques, mais extraites par traitement d'image sur des scènes enregistrées par caméra.

Ces scènes à risque peuvent prendre différentes formes en fonction de l'activité industrielle que l'on cherche à contrôler : raffineries, usines pétrochimiques ou sidérurgiques, et bien que certaines scènes soient caractéristiques d'un rejet à haut risque pour l'environnement sur un site donné, il n'est pas possible de définir à l'avance un système expert qui serait en mesure d'évaluer la gravité des différentes scènes observées de manière exhaustive. Le concept d'une machine capable d'apprendre directement à reconnaître différents types de scènes à risque à partir d'exemples serait d'autant plus attrayant qu'il permettrait à la fois de faciliter l'utilisation du système, en évitant à un opérateur humain son paramétrage, tout en améliorerant sa portabilité.

1.2 L'apprentissage Automatique

Parmi les nombreuses disciplines représentées en informatique, l'Apprentissage Automatique, ou *Machine Learning* en anglais, désigne la recherche et l'étude d'algorithmes permettant à une machine de réaliser l'induction automatique de règles à partir d'un ensemble d'exemples. Ce domaine de l'intelligence artificielle est proche des statistiques, de la fouille de données et de la reconnaissance de formes. Pour résoudre des problèmes issus du monde réel, la machine doit apprendre à produire la sortie désirée lorsqu'on lui présente un vecteur d'entrées particulier. On distingue en général les problèmes de régression et les problèmes de classification. Pour les problèmes de régression, la sortie à calculer a une valeur continue. Réaliser une estimation du chiffre d'affaire d'une entreprise à partir d'un ensemble de variables concernant ses activités est une illustration possible de ce type de problèmes. Les problèmes de classification, quant à eux, cherchent à exprimer une sortie correspondant à une étiquette ou à un groupe, à valeur dans un ensemble fini. La reconnaissance de caractères manuscrits, où l'étiquette correspond alors à l'une des vingt-six lettres de l'alphabet, est l'une des nombreuses applications usuelles de ce type de problème.

Cette thèse concerne plus particulièrement le problème de la classification. De manière empirique, il s'agit de déterminer un modèle à partir des échantillons disponibles pour le problème posé, de manière à appliquer par la suite ce modèle pour classer automatiquement de nouveaux échantillons jusqu'alors inconnus. Pour les applications usuelles, le nombre d'exemples dont on dispose est limité, et ne couvre pas nécessairement l'ensemble des formes possibles que peuvent prendre les objets que l'on cherche à classer. En ce sens, l'apprentissage de la machine présente une forte analogie avec l'apprentissage humain. L'élève qui apprendrait par coeur les exercices donnés par un professeur se verrait dans l'incapacité de résoudre de nouveaux exercices inconnus. Pour être efficace, l'élève doit effectuer un travail de synthèse, de manière à extraire les concepts

généraux à partir des exemples qui lui ont été fournis. Cet apprentissage lui permet ensuite de trouver la solution à de nouveaux exercices plus ou moins proches de ceux sur lesquels il s'est entrainé. Au même titre que l'élève que nous venons de décrire, la machine apprenant à reconnaître différents objets doit construire un modèle en adéquation avec les exemples qui lui ont été présentés au cours du processus d'entraînement, mais ce modèle doit rester suffisamment général pour identifier des exemples s'éloignant des prototypes sur lequel a été réalisé l'apprentissage.

Le domaine de l'apprentissage automatique a acquis une connaissance robuste de ce problème, depuis le premier modèle de perceptron décrit par Rosenblatt jusqu'à la récente théorie de l'apprentissage statistique proposée par Vapnik. La littérature traitant du sujet est particulièrement vaste, et la plupart des algorithmes existant ont été appliqués avec succès à une grande variété de problèmes issus du monde réel.

Le second chapitre de cette thèse est d'ailleurs consacré au formalisme de l'apprentissage et de la classification supervisée, et à la description des principales familles d'algorithmes permettant de réaliser cette tâche. Ce choix d'algorithmes a été fait non pas en fonction de leur efficacité, mais en raison des différentes approches dont ils sont issus, de manière à illustrer la grande variétés des méthodes disponibles. Cependant, il n'est pas possible de désigner un algorithme qui soit toujours meilleur que les autres, le paramétrage du classifieur étant le plus souvent spécifique à l'application. De plus, il demeure toujours une part irréductible d'aléatoire dans les protocoles mis en place pour évaluer un modèle. La question revenant le plus souvent au sein de la communauté de l'apprentissage automatique est maintenant de savoir comment combiner l'information produite par plusieurs prédicteurs de manière à tirer parti des avantages de chacun.

1.3 Vers une approche ensembliste

Ces dernières années, les recherches dans le domaine de l'apprentissage automatique ont montré que l'utilisation d'un ensemble de modèles permettait d'améliorer de manière significative les performances obtenues en classification par rapport à l'utilisation d'un classifieur unique. Intuitivement, un ensemble de classifieurs permet de réduire le nombre d'exemples qui seraient mal classés par un prédicteur unique si ses membres commettent des erreurs différentes. La comparaison entre les algortithmes basés sur le Bagging et le Boosting en est l'illustration la plus évidente. Fusionner les décisions provenant de plusieurs classifieurs spécialisés induit généralement une réduction de l'erreur plus importante que celle obtenue avec un ensemble de modèles créés aléatoirement. La justification formelle de ce résultat intuitif repose sur la notion de diversité inter-classifieurs. Cependant, si la connaissance dont on dispose sur les mécanismes de l'apprentissage automatique sont relativement solides, il n'en est pas de même concernant le rôle de la diversité dans les ensembles de classifieurs. Il est clairement reconnu que le seul fait d'optimiser la diversité de l'ensemble n'offre pas une garantie suffisante pour minimiser le risque d'erreur de l'ensemble obtenu, mais la relation entre la précision individuelle des classifieurs et leur diversité demeure encore floue.

Dans cette thèse, nous effectuons donc une synthèse des travaux sur les ensembles de classifieurs et de leurs liens avec la diversité. Nous étendons ces travaux en proposant un algorithme génétique permettant d'aborder la construction d'ensembles de classifieurs selon le paradigme *Surproduction et Sélection*. Notre objectif principal est d'étudier plus en détail le compromis entre la précision et la diversité lorsque cette dernière est utilisée explicitement pour construire un ensemble de classifieurs.

1.4 Ce qu'apporte cette thèse

Cette thèse présente les travaux de recherches sur le thème principal de l'apprentissage supervisé et plus particulièrement sur le rôle joué par la diversité dans les ensembles de classifieurs. Le présent mémoire n'a pas la prétention d'effectuer un recensement exhaustif de tous les algorithmes existant en apprentissage supervisé, mais plutôt de faire le point sur l'état d'avancement actuel des théories et connaissances en Apprentissage Automatique.

Cette thèse propose également une application de l'Apprentissage Automatique à un problème concret et d'actualité : le contrôle de la pollution de l'air. Nous proposons une adaptation des algorithmes d'Apprentissage Automatique usuels dans un contexte de classification supervisée des rejets de fumées dangeureuses survenant sur des sites industriels et enregistrés par caméras. Ces travaux de recherche sont destinés à l'intégration d'un module d'apprentissage dans un système de surveillance vidéo existant, dans le but de diminuer le nombre de détections incorrectes et de faciliter le paramétrage du système.

1.5 Comment est organisée cette thèse

Le présent mémoire est organisé de la manière suivante :

Le chapitre 2 propose un formaliste de la notion d'apprentissage automatique, et introduit un ensemble de notations qui seront utilisées tout au long du présent mémoire. Ce chapitre permettra également de réaliser un état de l'art des algorithmes de classifications usuels, de leurs limitations, et de la manière avec laquelle ils abordent le problème de la classification supervisée.

Le chapitre 3 permet de répondre aux limitations des méthodes décrites dans le second chapitre en introduisant la notion d'ensemble de classifieurs. Nous ferons la distinction entre l'utilisation d'ensembles dans le but de décomposer un problème complexe en sous-problèmes élémentaires, en décrivant les schémas de décomposition usuels, et par conséquent les différentes techniques permettant de fusionner les décisions provenant d'un ensemble d'experts, et la volonté d'utiliser un ensemble en vue d'améliorer la précision des résultats. Cette dernière approche nous permettra de formaliser la motivation intuitive selon laquelle un comité d'experts permet en général une meilleure décision en nous appuyant sur le concept de diversité dans un ensemble de classifieurs. Nous présenterons les motivations théoriques en nous appuyant sur la décomposition biais-variance du risque dans un contexte de régression, ainsi que les différentes adaptations applicables dans un contexte de classification. Nous établirons un état de l'art sur les principaux algorithmes ensemblistes, en nous intéressant plus particulièrement à la diversité induite durant l'apprentissage des classifieurs et son impact sur la précision des résultats obtenus.

Le chapitre 4 présente une approche génétique pour la création d'ensemble de classifieurs, basée sur l'optimisation d'un critère prenant en compte la diversité de l'ensemble. Les résultats de l'implémentation proposée sont comparés à ceux obtenus avec d'autres algorithmes ensemblistes sur plusieurs bases de l'UCI Repository.

Le chapitre 5 décrit la mise en oeuvre de la méthode proposée sur une application réelle, à savoir la classification de scènes à risque sur site industriel. Nous détaillerons dans ce chapitre le problème de la classification de scènes évoqué précédemment dans l'introduction et présenterons le système de détection tel qu'il est actuellement utilisé dans le nord de la France pour procéder à la détection et à la classification de fumées diffuses survenant sur des usines pétrochimiques ou des acieries.

Enfin, le sixième et dernier chapitre résume la contribution, les résultats obtenus ainsi que les limitations de l'approche que nous proposons dans cette thèse.

2

Apprentissage Automatique : formalisme et état de l'art

Dans le chapitre précédent, nous avons introduit le concept d'Apprentissage Automatique pour définir un ensemble d'algorithmes permettant à une machine de déterminer une procédure de classification à partir d'un ensemble d'exemples. Dans ce chapitre, une définition plus formelle du concept d'Apprentissage Automatique est donnée, et une présentation des notations usuelles de la littérature dans le domaine est réalisée. Nous introduisons les concepts inhérents à la classification supervisée et nous abordons l'évaluation, la validation des modèles et les phénomènes de sur-apprentissage. Ce second chapitre dresse également un état de l'art des méthodes et algorithmes usuels en Apprentissage Automatique, qui traitent ce problème sous différents aspects en fonction du domaine dont ils sont issus.

2.1 Qu'est-ce-qu'apprendre ?

L'apprentissage désigne, au sens large, le processus d'acquisition de connaissances, de compétences, de comportements ou plus généralement de savoir-faire, obtenu par différentes méthodes comme l'observation, l'association ou l'expérience. Les psychologues béhavioristes donnent une définition plus formelle du concept d'apprentissage : *l'individu qui apprend acquiert ou modifie une représentation de son environnement.* S'il est facile de concevoir la notion d'apprentissage en ces termes lorsque l'on parle d'un être vivant, cela devient beaucoup plus abstrait lorsque l'apprenant est une machine. L'Apprentissage Automatique, ou *Machine Learning* en anglais [146], est un domaine d'étude de l'Intelligence Artificielle ayant pour objectif le développement et l'implémentation d'algorithmes permettant à une machine d'apprendre à réaliser une tâche donnée. De manière formelle, nous disons que la machine doit apprendre à fournir la réponse correcte lorsqu'elle est face à une situation spécifique. Nous limitons notre approche de l'Apprentissage Automatique au domaine de l'Informatique, mais il est concerne d'autres disciplines telles que les sciences cognitives ou les neurosciences.

Pour illustrer cette définition, nous allons considérer une problème usuel auquel le lecteur, qu'il soit informaticien ou non, a déjà été confronté. Imaginons que l'on désire développer un logiciel ou système permettant de différencier automatiquement le courrier électronique provenant d'expéditeurs de confiance du courrier indésirable, à but publicitaire, ou au contenu frauduleux. Une solution partielle consiste à définir manuellement un ensemble de règles permettant à un ordinateur d'effectuer le classement du courrier. Cette solution ne peut filtrer la totalité des courriers indésirables car il est très certainement impossible de définir un ensemble de règles

exhaustives. En effet, le courrier indésirable peut prendre différentes formes : publicités, pièces jointes ou liens frauduleux. Il n'est pas possible de disposer, à chaque instant, d'un ensemble d'échantillons parfaitement représentatif de la forme que peut prendre le courrier indésirable. De plus, la définition manuelle d'un ensemble complet de règles de filtrage constituerait une tâche extrêmement laborieuse pour la personne en charge du développement ou pour l'utilisateur lui-même. Enfin, une autre source de difficulté provient du fait que le courrier indésirable est en constante évolution. Ceci implique une mise à jour régulière des règles de filtrage afin que le système s'adapte à de nouvelles formes de courriers indésirables. Ces limitations amènent à rechercher des moyens plus adaptés au filtrage du courrier électronique. Les méthodes d'apprentissage automatique permettent de concevoir un système qui, à partir du courrier présent dans la messagerie de l'utilisateur et préalablement scindé en deux catégories correspondant aux deux classes de courrier. Ce système pourrait apprendre lui-même des règles de filtrage, et pourrait identifier le courrier indésirable. Une bonne méthode de classification doit présenter un taux de reconnaissances correctes élevé même avec du courrier différant légèrement des échantillons utilisés lors de l'apprentissage. Ce type de problème est connu sous le nom de *classification*, et s'inscrit de manière plus générale dans le domaine de l'Apprentissage Automatique et de la Reconnaissance de Formes [224]. Le filtrage du courrier électronique décrit ici est un problème de classification binaire car le courrier est classé en deux catégories : acceptable ou inacceptable. Bien que certains algorithmes soient dédiés à des cas de classification binaire, nous verrons que l'on peut, dans la plupart des cas, généraliser une procédure de classification pour des problèmes comportant plus de deux classes distinctes. On pourrait également définir un système de filtrage du courrier qui, au lieu d'affecter une catégorie à chaque message, déterminerait une valeur réelle, entre 0 et 1 par exemple, et qui correspondrait au degré de confiance associé au message, par exemple $0,12$ pour un courrier au contenu frauduleux. C'est ce qu'on appelle un problème de régression.

Plus particulièrement, on parlera dans le cas qui nous intéresse d'*apprentissage supervisé*, car les classes auxquelles appartiennent les exemples sont connues *a priori*. Lorsque le nombre et la nature des classes entre lesquelles se répartissent les échantillons ne sont pas connus à l'avance, on parle d'apprentissage *non supervisé*. Dans ce cas, l'apprentissage réalisé par le système n'est pas guidé par la décision d'un expert ou superviseur, mais il est fonction uniquement des similitutes ou des différences entre les échantillons disponibles. La contribution apportée par cette thèse concerne l'approche supervisée de l'Apprentissage Automatique. Cependant nous présentons brièvement les différents aspects de l'apprentissage automatique, pour éviter toute confusion éventuelle, mais également pour définir clairement les limites des travaux de recherche décrits dans le présent mémoire.

Le filtrage du courrier électronique est l'un des nombreux problèmes qui peuvent être traités grâce à la classification supervisée. Parmi les domaines d'application importants de la classification supervisée, on peut citer l'identification de la parole, la classification automatique de documents, la reconnaissance de caractères manuscrits ou encore le diagnostic médical et plus généralement la bio-informatique. La première partie de cette thèse est consacrée à la présentation de différentes familles d'algorithmes d'Apprentissage Automatique issus de différents courants, mais ayant pour objectif commun de résoudre les problèmes de classification. Avant de présenter en détail chacune de ces méthodes, il est nécessaire de formaliser le problème de la classification et d'introduire certaines notations qui seront utilisées au cours de ce mémoire.

2.2 Formalisme de l'apprentissage automatique

2.2.1 Définitions conceptuelles

Soit $Z = \{(X_1, Y_1), (X_2, Y_2), \dots (X_N, Y_N)\}$ un ensemble de N objets, avec $X_i \in \Re^n$ le vecteur contenant les n attributs caractérisant l'objet i et $Y_i \in \Omega = \{\omega_1, \omega_2, \dots \omega_c\}$ la classe à laquelle appartient l'objet i, où Ω désigne un ensemble fini d'étiquettes ou classes. Un classifieur peut être vu comme une fonction C à valeurs dans Ω permettant d'associer à chaque objet défini par ses attributs la classe à laquelle il appartient :

$$\begin{aligned} C :\ & \Re^n \rightarrow \Omega \\ & C(X_i) \mapsto Y_i \end{aligned} \tag{2.1}$$

L'Apprentissage Automatique est un procédé inductif consistant à déterminer cette fonction C à partir des exemples disponibles. Une généralisation de cette définition est proposée dans [50], en considérant un classifieur comme un ensemble fini G de c fonctions discriminantes :

$$G = \{g_1(X), \dots, g_c(X)\} \quad avec \quad g_i : \Re^n \rightarrow \Re \quad i = 1, \dots, c \tag{2.2}$$

Le résultat réel de chacune de ces fonctions peut être vu comme un degré associé à la classe qu'elle caractérise. La classe associée à l'objet considéré est celle dont le degré est maximal :

$$C(X) = \omega_{i^*} \in \Omega \Leftrightarrow g_{i^*} = \max_{i=1,\dots,c} \{g_i(X)\} \tag{2.3}$$

Cette définition convient plus particulièrement à certains types de classifieurs. C'est le cas, par exemple, des réseaux de neurones, dont la structure donne lieu le plus souvent à plusieurs fonctions d'activation à valeur dans $[-1,\ 1]$. C'est le cas également des réseaux bayésiens, où les fonctions discriminantes peuvent être vues comme les probabilités *a posteriori* de chacune des classes. Pour des problèmes de régression, les fonctions discriminantes sont plus généralement à valeurs dans $\Re$. Pour des problèmes de classification, on parle indifféremment de fonctions à valeurs dans un ensemble fini de valeurs discrètes Ω ou de frontières de décisions.

Les fonctions discriminantes forment une partitions de l'espace des attributs, de dimension n, en c régions de décision, où c désigne le nombre de fonctions discriminantes. La region $\mathcal{R}_i$ correspondant à la classe ω_i est le sous-espace dans lequel la fonction discriminante $g_i(X)$ a une valeur maximale :

$$\mathcal{R}_i = \left\{ X \mid X \in \Re^n, g_i(X) = \max_{k=1,\dots,c} g_k(X) \right\}, \quad i = 1, \dots, c \tag{2.4}$$

Cependant, pour les problèmes réels, il est rare que les objets appartenant à une classe ω_i soient regroupés dans une même région $\mathcal{R}_i$ car les distributions des différentes classes dans l'espace des attributs présentent des recouvrements partiels. Ces recouvrements sont au coeur des principales difficultées rencontrées en Apprentissage Automatique. Il est délicat de déterminer des frontières de décision optimales à partir d'un ensemble fini d'exemples, car cet ensemble n'offre en général qu'une représentation réduite du problème posé. Dans la section suivante, nous montrons que le classifieur optimal selon la théorie des probabilités présente une erreur irréductible induite par le recouvrement partiel entre les classes.

2.2.2 Décision de Bayes, mesure d'erreur et sur-apprentissage

Pour les applications réelles, les classes en présence ne sont en général pas clairement séparables et présentent un recouvrement partiel. La théorie de la décision selon Bayes fournit un cadre particulièrement explicite pour décrire la difficulté de l'apprentissage supervisé en termes d'approximation des probabilités *a posteriori* de chaque classe. Considérons un problème à c classes $\omega_1, \ldots, \omega_c$ pour lequel les probabilités *a priori* $P(\omega_i)$ sont supposées connues :

$$0 \leq P(\omega_i) \leq 1 \quad avec \quad \sum_{i=1}^{c} P(\omega_i) = 1 \tag{2.5}$$

Supposons que les exemples appartenant à chacune des classes ω_i du problème se répartissent dans l'espace des attributs $\Re^n$ suivant une densité de probabilité $P(X|\omega_i)$:

$$P(X|\omega_i) \geq 0 \quad avec \quad \int_{\Re^n} P(X|\omega_i)\, dX = 1, \quad i = 1, \ldots, c \tag{2.6}$$

Le théorème de Bayes nous permet de déterminer les probabilités *a posteriori* des différentes classes $P(\omega_i|X)$ que nous cherchons à estimer :

$$P(\omega_i|X) = \frac{P(\omega_i)\, P(X|\omega_i)}{P(X)} = \frac{P(\omega_i)\, P(X|\omega_i)}{\sum_{j=1}^{c} P(\omega_j)\, P(X|\omega_j)} \tag{2.7}$$

L'équation 2.7 montre pourquoi les probabilités *a posteriori réelles* ne sont en général pas calculables. En effet, les probabilités à priori $P(\omega_i)$ sont inconnues ou difficiles à estimer en pratique, et la distribution des paramètres $P(X|\omega_i)$ doit être déterminée par apprentissage à partir d'un ensemble d'exemples non nécessairement représentatif du problème posé.

A partir des probabilités *a posteriori* réelles, il est possible de définir un classifieur optimal en reprenant la définition donnée dans l'équation 2.2 :

$$C^* = \{g_i(X) = P(\omega_i|X), \quad i = 1, \ldots, c\} \tag{2.8}$$

qui assigne à chaque exemple la classe dont la probabilité *a posteriori* est maximale :

$$C^*(X) = \omega_i^* \in \Omega \Leftrightarrow P(\omega_i^*|X) = \max_{i=1,\ldots,c} \{P(\omega_i|X)\} \tag{2.9}$$

Ce classifieur optimal présente une quantité d'erreur minimale et incompressible induite par le recouvrement partiel entre les différentes classes du problème. Cette erreur, appelée *Erreur de Bayes*, est notée $P_e(C^*)$ et correspond à la somme des erreurs commises par le classifieur optimal sur chaque région de décision R_i^* :

$$P_e(C^*) = \sum_{i=1}^{c} \int_{R_i^*} [1 - P(\omega_i|X)]\, P(X)\, dX \tag{2.10}$$

La figure 2.1 illustre le calcul de l'erreur de bayes sur un problème à deux classes $\Omega = \{\omega_1, \omega_2\}$ pour lequel les exemples sont caractérisés par une variable unique continue $x \in \Re$ $(n = 1)$ dont les valeurs sont réparties selon une distribution normale. L'equation 2.10 s'écrit dans ce cas particulier :

$$P_e(C^*) = \int_{R_1^*} [1 - P(\omega_1|x)]\, P(x)\, dx + \int_{R_2^*} [1 - P(\omega_2|x)]\, P(x)\, dx \tag{2.11}$$

Pour un problème à deux classes, on peut écrire $P(\omega_1|x) = 1 - P(\omega_2|x)$ et l'équation 2.11 devient :

$$P_e(C^*) = \int_{\mathcal{R}_1^*} P(\omega_2|x)\, P(x)\, dx + \int_{\mathcal{R}_2^*} P(\omega_1|x)\, P(x)\, dx \tag{2.12}$$

D'après le thèorème de Bayes donné dans l'équation 2.7 nous obtenons finalement :

$$P_e(C^*) = \int_{\mathcal{R}_1^*} P(\omega_2)\, P(x|\omega_2)\, dx + \int_{\mathcal{R}_2^*} P(\omega_1)\, P(x|\omega_1)\, dx \tag{2.13}$$

La figure 2.1 représente la frontière de décision optimale pour un problème à deux classes obtenue en posant $g_i(x) = P(\omega_i)\, P(x|\omega_i)$. Cet ensemble de fonctions discriminantes est équivalent à $g_i(x) = P(\omega_i|x)$. En effet, la quantité $P(X) = \sum_{j=1}^{c} P(\omega_j)\, P(X|\omega_j)$ étant égale pour chacune des $g_i(x)$, l'ordre de grandeur reste inchangé, et la frontière de décision calculée par la règle du maximum n'est pas modifiée. La zone grisée, obtenue en sommant les erreurs commises sur chacune des régions de décision, représente l'erreur de Bayes et correspond à la quantité d'erreur minimale et incompressible obtenue avec le classifieur optimal.

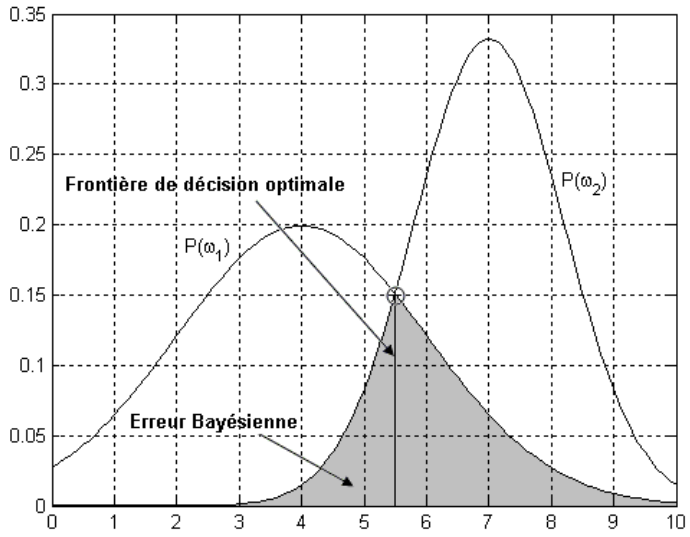

FIG. 2.1 – Erreur de Bayes obtenue pour un problème à deux classes

En pratique, on cherche à estimer les probabilités *a posteriori* des différentes classes, il n'est pas possible de savoir dans quelle mesure les frontières de décisions déterminées par apprentissage sont proches de la frontière optimale. Plusieurs méthodes existent pour évaluer la robustesse d'un classifieur. La section suivante présente les plus répandues d'entre elles.

2.2.3 Evaluation et validation des modèles

Pour les applications réelles, on ne dispose en général que d'un ensemble d'échantillons de taille réduite, qui n'est pas forcément représentatif du problème posé. Il est toutefois nécessaire d'estimer le risque d'erreur du classifieur appris. Une solution consiste à évaluer un modèle sur

les mêmes données que celles utilisées pour apprendre le modèle. Cette méthode par *substitution* serait fortement biaisée. En effet, on se trouve en présence d'un classifieur spécialisé sur les données ayant servies à son apprentissage mais dont on ignore totalement les facultés prédictives sur de nouveaux échantillons. Pour évaluer l'erreur en généralisation d'un classifieur, la méthode communément adoptée consiste à partitionner l'ensemble de données disponible Z en deux sous-ensembles Z_{tra} et Z_{tes}, dédiées respectivement à l'apprentissage (*training*) du classifieur et à l'évaluation (*test*) de ses performances. Z_{tes} est utilisé comme un ensemble de nouveaux échantillons de test inconnus lors de l'apprentissage. L'erreur obtenue sur chacun des partitionnements est donnée par :

$$Erreur(C) = \frac{1}{N_{tes}} \sum_{j=1}^{N_{tes}} \left\{ 1 - \mathcal{I}\left(\widehat{Y}_j, Y_j\right) \right\} \tag{2.14}$$

où N_{tes} représente la taille de l'ensemble de test Z_{tes}, $\widehat{Y}_j = C\left(X_j\right)$ désigne la classe calculée par le classifieur pour l'exemple j et Y_j est sa classe réelle. La fonction $\mathcal{I}$ est définie par :

$$\mathcal{I}\left(\widehat{Y}, Y\right) = \begin{cases} 1 & si \quad \widehat{Y} = Y \\ 0 & si \quad \widehat{Y} \neq Y \end{cases} \tag{2.15}$$

Pour évaluer les performances finales du classifieur, on considère l'erreur moyenne obtenue sur l'ensemble des partitionnements réalisés. Cette méthode présente l'avantage de fournir une bonne estimation des facultés prédictives des modèles obtenus, en évitant une dépendance du résultat au choix particulier des exemples d'apprentissage et de test. Le choix du partitionnement dépend principalement de la quantité de données disponibles. Une quantité trop faible d'échantillons d'apprentissage risque d'entraîner une situation de sous-apprentissage du modèle, qui se traduit par un modèle incorrect et peu précis. A l'inverse, un ensemble d'apprentissage de taille trop importante risque de provoquer un sur-apprentissage du modèle, dont les facultés prédictives sur de nouveaux exemples sont diminuées. Une étude des différentes méthodes utilisées pour la validation d'un modèle est présentée dans [108]. Nous en présentons quelques unes.

La méthode *Hold-Out* partitionne aléatoirement l'ensemble des données Z en deux parties, généralement de tailles égales. Une partie est utilisée pour apprendre le modèle, et l'autre pour calculer son erreur. Il est possible de permuter les deux parties (méthode *split-half*) et d'évaluer le modèle par la moyenne des deux estimations ainsi obtenues, mais pour estimer de manière plus robuste l'erreur en généralisation du modèle, il est nécessaire de réitérer sur plusieurs partitionnements.

La validation croisée en k blocs (*k-fold cross validation*) partitionne aléatoirement l'ensemble Z en k sous-ensembles de même taille appelés blocs. Le classifieur est entrainé sur les données contenues dans $k-1$ blocs, et évalué sur le bloc restant. Le processus est répété k fois, en changeant de bloc pour tester le classifieur. L'évaluation des qualités prédictives du modèle est obtenue en moyennant les performances obtenues à chaque itération.

Un cas particulier de cette méthode, appelé *leave-one-out cross validation*, consiste à prendre k égal au nombre d'échantillons disponibles N. Chaque classifieur est alors entrainé sur l'ensemble des données auquel a été retiré un unique exemple, utilisé pour tester le modèle. Cette méthode, plus longue à mettre en oeuvre du fait de la taille importante de l'ensemble d'apprentissage et du nombre d'itérations nécessaires, convient plus particulièrement aux problèmes pour lequel l'ensemble d'échantillons N disponibles est faible. D'autre part, cette méthode donne une bonne estimation de l'erreur pour des modèles cherchant à approximer des fonctions continues, mais la validation croisée en k blocs se révèle plus adaptée pour évaluer l'erreur d'un classifieur.

Des méthodes basées sur le *Bootstrap*, introduites par Efron et Tibshirani [53], sont également utilisées pour évaluer les performances d'un modèle. Contrairement aux méthodes décrites précédemment, elles sont basées non plus sur un partitionnement des données initiales en plusieurs sous-ensembles, mais sur un rééchantillonnage des données initiales. Ces méthodes consistent à générer b ensembles de taille N, appelés *bootstrap samples*, à partir des données initiales, en effectuant des tirages aléatoires avec remise. La procédure la plus simple qui en résulte consiste à apprendre un modèle différent sur chacun des b ensembles de données ainsi formés, et de calculer leur erreur prédictive sur l'ensemble de données initial $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$. La moyenne des erreurs de chaque modèle constitue l'estimation par bootstrap ordinaire [53]. Cette méthode a tendance à sous-estimer l'erreur en généralisation induite par un recouvrement partiel entre les données d'apprentissage et de test, du fait que plusieurs échantillons sont utilisés à la fois pour construire et évaluer le classifieur. Plusieurs méthodes dérivées de cette procédure sont décrites dans [102]. Cependant, l'estimateur ayant la plus grande robustesse est l'estimateur *bootstrap .632* [51, 52]. Les b ensembles de test obtenus par rééchantillonage des données d'origine sont générés par tirage avec remise. La probabilité qu'un échantillon n'ait pas été sélectionné au terme des N tirages aléatoires est donnée par $\left(1 - \frac{1}{N}\right)^N \approx e^{-1} \approx 0.368$. Par conséquent, le nombre d'échantillons distincs des données d'origine apparaissant dans les données de test est de $0.632N$. L'estimation de l'erreur prédictive du classifieur s'exprime par :

$$Pre_{boot} = \frac{1}{b} \sum_{i=1}^{b} (0,632 \cdot \epsilon 0_i + 0,368 \cdot Pre_{\mathcal{S}}) \tag{2.16}$$

où $\epsilon 0_i$ désigne la précision obtenue en testant le modèle sur le rééchantillonage i et $Pre_{\mathcal{S}}$ représente la précision obtenue par substitution de l'ensemble de test par les données d'apprentissage d'origine. Les méthodes basées sur le *bootstrap* sont plus difficiles à mettre en œuvre que la validation croisée, du fait du nombre important de rééchantillonages requis pour appromixer correctement l'erreur en généralisation d'un classifieur. Elles sont principalement utilisées lorsque l'on se trouve en présence d'un nombre limité d'échantillons, en particulier pour des données médicales, lorsque le nombre de patients est limité.

Il est parfois nécessaire de définir un ensemble additionnel de données Z_{val}, appelé *ensemble de validation*, utilisé en parallèle des données d'entraînement pour détecter d'éventuelles situations de sur-apprentissage. Le modèle est entrainé sur l'ensemble d'apprentissage Z_{tra} et son erreur en généralisation est évaluée simultanément sur Z_{val}. Une situation de sur-apprentissage survient lorsque le modèle apprend parfaitement les données d'entraînement, mais généralise très mal sur de nouveaux exemples. Cette situation correspond au phénomène d'apprentissage *par coeur* et elle est illustré dans la figure 2.2.

L'ensemble de validation est utilisé pour évaluer l'erreur en généralisation au cours du processus d'entraînement, de manière à arrêter l'apprentissage dès que les facultés prédictives du modèle diminuent. Cette méthode est souvent utilisée pour l'entraînement des réseaux de neurones. Dans ce cas, l'apprentissage consiste à modifier la structure du modèle de manière itérative jusqu'à ce qu'il converge vers un état où l'erreur d'apprentissage est minimale mais au détriment de ses propriétés de généralisation. L'utilisation d'un ensemble de validation additionnel suppose que l'on dispose d'un nombre important d'exemples. En effet, une diminution trop importante du nombre d'échantillons dédiés à l'apprentissage du modèle entraîne paradoxalement un sous-apprentissage du classifieur.

Le choix de la méthode permettant d'évaluer un modèle est fortement lié à l'application : problème de régression ou de classification, taille de l'ensemble de données, importance du nombre n d'attributs considérés relativement au nombre d'échantillons disponibles sont autant de pa-

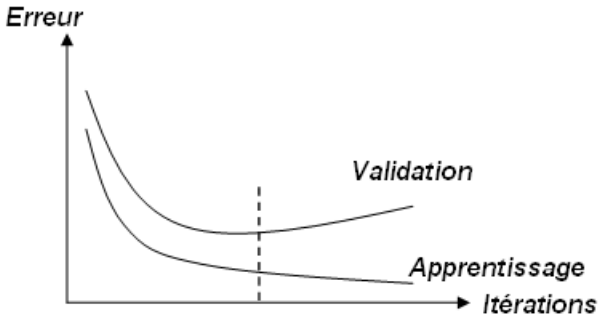

FIG. 2.2 – Variation de l'erreur prédictive et de l'erreur d'apprentissage d'un modèle en situation de sur-apprentissage : passé un certain seuil, l'écart entre les deux erreurs a tendance à augmenter

ramètres à prendre en compte lors du choix du protocole à mettre en place. Des comparaisons concernant ces différentes méthodes sont décrites dans [217, 213, 16, 14].

2.3 Algorithmes de classification usuels

Nous présentons dans cette partie les principaux algorithmes de classification supervisée proposés dans la littérature. Nous ne retenons que les méthodes utilisées pour des problèmes non linéairement séparables, qui représentent la majeur partie des applications réelles. Cette partie aborde le problème de l'apprentissage supervisé selon différentes approches.

2.3.1 Réseaux Bayésiens

Principes et notations

L'apprentissage bayésien repose sur le principe de minimisation du risque empirique. Ce principe considère que l'hypothèse minimisant le risque d'erreur est celle qui représente le mieux les données observables, en supposant que ces dernières sont représentatives du monde général. Ce type d'apprentissage définit sur l'espace des fonctions hypothèses des distributions de probabilités qui vont être modifiées suivant l'information "contenue" dans l'ensemble d'apprentissage. Au début de ce chapitre, nous avons montré comment la théorie de la décision selon Bayes définit un classifieur optimal qui attribue à chaque exemple la classe pour laquelle la probabilité *a posteriori* $P(\omega_i|X)$ est maximale. L'apprentissage bayésien est basé sur cette approche probabiliste pour déterminer la classe d'un objet par inférence à partir d'un ensemble d'observations. Le problème de l'apprentissage d'une règle de classification serait résolu si l'on connaissait les probabilités *a priori* $P(\omega_i)$ et les densités de probabilités $P(X|\omega_i)$. En pratique, les probabilités *a priori* $P(\omega_i)$ sont supposées égales, ou alors elles sont estimées à partir des fréquences d'apparition dans l'ensemble d'apprentissage. Les densités de probabilités $P(X|\omega_i)$ sont apprises à partir d'un nombre fini d'observations supposées représentatives du problème considéré, ou définies par un expert. Les méthodes paramétriques supposent que ces densités de probabilité possèdent une forme analytique. Pour une distribution gaussienne par exemple, il suffit d'estimer la moyenne et la variance de cette distribution. Lorsque la variable considérée est une valeur discrète possédant un nombre fini de modalités, on parle de table de probabilités conditionnelles. A l'inverse, les méthodes non-paramétriques, comme l'algorithme de Parzen ou les k plus proches

voisins se contentent d'estimer localement ces densités de probabilités en observant l'ensemble d'apprentissage dans le voisinage de l'objet à classer.

L'apprentissage bayésien revient, en général, à approximer le classifieur optimal de Bayes à partir des exemples disponibles. Cette approximation entraine l'ajout d'une quantité d'erreur du fait de la non optimalité des frontières de décision obtenues. Dans la figure 2.1 donnée au début de ce chapitre, nous avons présenté le classifieur optimal selon Bayes, en supposant que la distribution des données $P(X|\omega_i)$ est connue. Nous avons présenté un exemple dans lequel il n'y a qu'une seule variable, que nous avons noté arbitrairement $P(X)$. En pratique, les applications réelles font généralement intervenir un grand nombre de variables. $P(X)$ correspond à une loi de probabilité jointe $P(x_1, \ldots, x_n)$ que l'on peut factoriser sous la forme :

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i|x_1, \ldots, x_{i-1}) \tag{2.17}$$

Dans le cas général, la distribution jointe dépend d'un trop grand nombre de variables pour être exploitable en pratique. De plus, des relations d'indépendance conditionnelle peuvent exister entre certaines variables du problème. En terme de probabilités, on définit l'indépendance entre deux variables ou ensembles de variables A et B conditionnellement à un ensemble C, que l'on note $A \perp B|\ C$, par :

$$P(A\ |B, C) = P(A|\ C) \tag{2.18}$$

Ces indépendances permettent de simplifier la factorisation donnée dans l'équation 2.17, difficilement calculable en pratique. Les réseaux bayésiens [100, 161, 151, 149] permettent de répondre à la difficulté de ce calcul en combinant l'approche probabiliste et la théorie des graphes. Ils représentent la structure de la connaissance et rendent l'inférence possible localement en s'appuyant sur la condition de Markov, que nous décrivons ci-dessous.

Soit un graphe orienté $G = (N, A)$, où N désigne un ensemble fini de nœuds et A représente un ensemble d'arcs reliant deux éléments distincts de N. On appelle chemin entre deux nœuds N_1 et N_2 un ensemble d'arcs permettant de connecter ces deux nœuds. Ce graphe est dit acyclique si pour tout élément de N, il n'existe aucun chemin reliant cet élément à lui même. On définit une terminologie particulière sur un graphe acyclique. S'il existe un arc orienté de N_1 vers N_2, on dit que N_1 est un parent de N_2, et inversement, que N_2 est un descendant de N_1.

Supposons que l'on dispose d'une distribution de probabilité jointe P sur un ensemble de variable $X = \{x_1, \ldots, x_n\}$, que l'on modélise sous la forme d'un graphe orienté acyclique $G = (X, A)$. On dit que le couple (G, P) satisfait la *condition de Markov* si, pour toute variable $x_i \in X$, x_i est conditionnellement indépendante de l'ensemble de ses non descendants connaissant l'état de ses parents, que l'on note :

$$I_P(x_i, nondescendants(x_i)|parents(x_i)) \tag{2.19}$$

La condition de Markov permet de ramener le calcul des probabilités jointes à un niveau local. En effet, si le couple (G, P) satisfait la condition de Markov, alors la probabilité jointe d'un ensemble de variables $P(x_1, \ldots, x_n)$ est égale au produit des probabilités conditionnelles de ces variables conditionnellement à la valeur de leur parents :

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid parents(x_i)) \tag{2.20}$$

L'exploitation de l'équation 2.20 permet de résoudre les problèmes comportant un nombre important de variables en réduisant le nombre de probabilités à estimer. Soient P une distribution de probabilités sur un ensemble de variables $X = (x_1, \ldots, x_n)$ et G un graphe orienté acyclique. Le couple (G, P) est appelé réseau bayésien s'il satisfait la condition de Markov. Il est cependant important de noter que pour une même distribution de probabilité P, il existe plusieurs graphes G pour lesquels que le couple (G, P) satisfait la condition de Markov. On dit qu'un graphe G est fidèle pour une distribution de probabilité P lorsque G encode toutes les indépendances conditionnelles de P. Les réseaux bayésiens, parfois appelés *bayesian belief networks* ou *causal networks*, permettent de représenter des situations de raisonnement probabiliste à partir de connaissances incertaines.

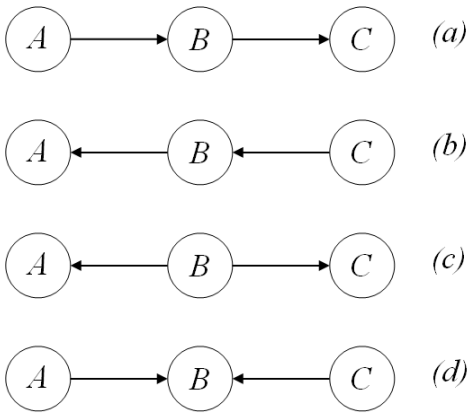

FIG. 2.3 – Indépendances conditionnelles encodées dans la structure du graphe

L'indépendance conditionnelle entre deux ensembles de variables peut être traduite en terme de propriétés d'un graphe par la notion de *d-séparation* [161], illustrée dans la figure 2.3. On dit que deux ensembles de variables X et Z sont *d-séparés* par un troisième ensemble Y si chaque chemin du graphe allant de X à Z est *bloqué* par Y. Une chemin est dit *bloqué* s'il vérifie l'une des trois conditions suivantes :

- Il existe une variable V appartenant à Y sur ce chemin, telle que les arcs incidents à V soient de la forme $X \ldots \leftarrow V \rightarrow \ldots Z$
- Il existe une variable V appartenant à Y sur ce chemin, telle que les arcs incidents à V soient de la forme $X \ldots \rightarrow V \rightarrow \ldots Z$
- Il existe une variable V n'appartenant pas à Y sur ce chemin, et dont les descendants n'appartiennent pas à Y, et telle que les arcs incidents à V soient de la forme $X \ldots \rightarrow V \leftarrow \ldots Z$

Si deux ensembles X et Z sont d-séparés par Y, alors X et Z sont conditionnellement indépendants connaissant Y. De plus, il est possible de redéfinir la notion de fidélité d'un graphe G à partir de la notion de d-séparation. On dit alors que G est fidèle pour une distribution de probabilité P s'il vérifie la condition inverse, c'est à dire que pour trois ensembles de variables données X, Y et Z, si $X \perp Z | Y$ alors X et Z sont d-séparés par Y dans G, car un graphe fidèle pour P encode toutes les indépendances conditionnelles de P.

Si nous reprenons les exemples de la figure 2.3, nous pouvons dire que B d-sépare A et C suivant la première condition dans le cas *(c)*, et suivant la deuxième condition dans les cas *(a)* et *(b)*. Ainsi, les graphes *a*, *b* et *c*, munis d'une distribution de probabilité P telle que

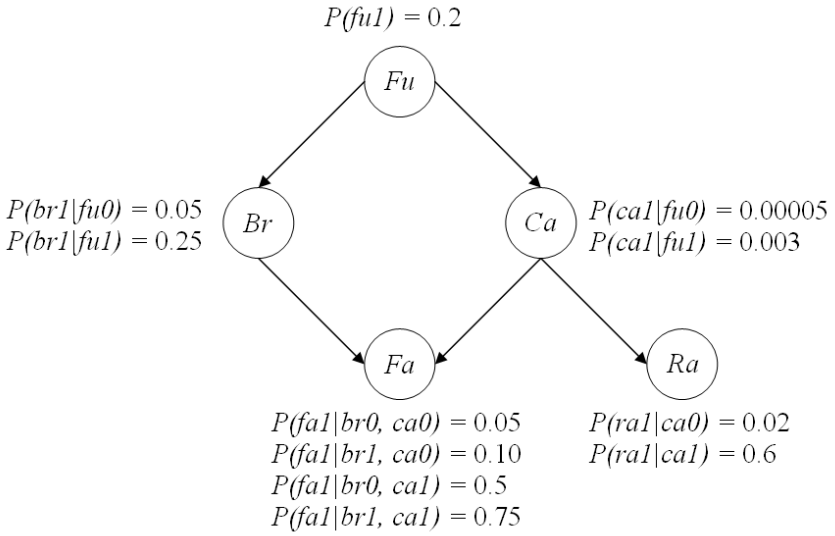

FIG. 2.4 – Exemple de réseau bayésien utilisé pour le diagnostic médical

$P(A|\ B,C) = P(A|\ C)$ constituent 3 réseaux bayésiens possibles pour cette distribution. Dans ce cas, les trois graphes encodent la même indépendance conditionnelle. Dans le cas général, il est fréquent que certains graphes encodent plus d'indépendances conditionnelles que d'autres. En revanche, dans le cas *(d)*, B ne d-sépare pas A et C.

Nous résumons les mécanismes mis en jeu par les réseaux bayésiens à travers un exemple issu du domaine médical et décrit dans [151].

On considère un ensemble de variables binaires $X = (Fu, Br, Ca, Fa, Ra)$ muni d'une distribution de probabilités, et dont les dépendances conditionnelles sont modélisées sous la forme du graphe causal donné dans figure 2.4. Les variables étant binaires, nous ne donnons qu'une seule valeur dans les tables de probabilités de chaque nœud, de manière à alléger la notation. La signification des variables est donnée dans la table 2.1.

TAB. 2.1 – Descriptif des variables utilisés dans notre exemple récapitulatif.

Variable	**Valeur**	**Signification**
Fu	*fu0*	Le patient ne présente aucun antécédent au tabac
	fu1	Le patient présente des antécédents au tabac
Br	*br0*	Absence de bronchite
	br1	Présence de bronchite
Ca	*ca0*	Absence de cancer du poumon
	ca1	Présence de cancer du poumon
Fa	*fa0*	Absence de fatigue
	fa1	Présence de fatigue
Ra	*ra0*	Radiographie des poumons négative
	ra1	Radiographie des poumons positive

On observant le graphe, on peut voir, par exemple, que le fait de fumer a une influence directe sur le risque de bronchite ou de cancer du poumon. De manière similaire, la présence d'un cancer

a une influence directe sur le résultat d'une radiographie des poumons. En revanche, sachant qu'un patient souffre de bronchite, obtenir une information sur ses antécédents de fumeur n'aura aucune influence sur la présence de signes de fatigue, car Br d-sépare Fu et Fa et induit une indépendance conditionnelle. Ce réseau pourrait être utilisé par exemple pour déterminer les probabilités qu'un patient souffre de bronchite sachant qu'il fume, souffre de fatigue et que la radiographie de ses poumons montre la présence d'une lésion. Les propriétés des réseaux bayésiens permettent de limiter le nombre de calculs à effectuer pour répondre à ce type de question.

De l'inférence probabiliste à la classification supervisée

Le réseau que nous proposons dans la figure 2.4 est réalisé manuellement par un expert du domaine, qui définit la structure du graphe causal et les tables de probabilités associées. Pour des réseaux de grande taille, il est difficile pour l'expert de construire ce réseau et il convient de faire réaliser cette tâche par la machine. L'apprentissage du réseau bayésien à partir des données se compose de deux tâches distinctes : l'apprentissage de la structure causale du graphe et l'estimation des densités de probabilités conditionnelles qui sont associées à cette structure. Cooper et Herskovits [29] proposent un algorithme permettant l'estimation des densités de probabilités des différentes variables en utilisant leurs fréquences relatives pour une structure donnée.

Les algorithmes permettant de proposer la structure d'un graphe à partir des données s'organisent autour de deux grandes stratégies. La première s'appuie sur le fait que le graphe causal modélise une distribution de probabilité jointe sur les données d'entraînement. Une structure sera considérée comme acceptable si elle modélise correctement les données d'apprentissage. Les heuristiques exploratoires basées sur l'utilisation de scores sont une illustration d'une telle stratégie. Elles s'appuient sur une fonction de coût que l'on cherche à maximiser, à partir d'un graphe vide auquel on ajoute de manière successive des arcs permettant une augmentation de la fonction de coût utilisée [88].

La seconde stratégie repose sur le principe selon lequel la structure du graphe doit encoder les relations d'indépendance entre les différentes variables. Ce principe favorise donc les graphes causaux prenant le mieux en compte les indépendances conditionnelles entre les attributs considérés. Cette approche regroupe les algorithmes basés sur un critère d'information qui utilisent des tests statistiques pour élaguer un graphe complétement connecté en supprimant les arcs entre deux variables présentant une indépendance conditionnelle [26].

Friedman *et al.* [63] montrent que la seconde stratégie est en général plus efficace en classification que les méthodes basées sur l'utilisation d'un score pour les applications pratiques.

Intéressons nous à une structure particulière, appelée classifieur bayésien naif. Ce modèle, bien que reposant sur l'hypothèse forte selon laquelle les attributs sont statistiquement indépendants, se montre en général efficace pour la classification [231, 115, 178, 48]. Ce type de classifieur est représenté par un graphe en étoile, au centre duquel se trouve la variable pour laquelle on cherche à estimer les probabilités *a posteriori* (2.5).

L'hypothèse d'indépendance conditionnelle entre les attributs $P(x_i|C, x_j) = P(x_i|C)$ nous permet de redéfinir les probabilités *a posteriori* :

$$P(C = \omega_i|x_1, \ldots, x_n) = \frac{1}{Z} P(C = \omega_i) \prod_{j=1}^{n} P(x_j|C = \omega_i) \tag{2.21}$$

La classe w^* attribuée par ce classifieur est alors donnée par :

$$\omega^* = argmax_c \; P(C = \omega_c) \prod_{i=1}^{n} P(x_i|C = \omega_c) \tag{2.22}$$

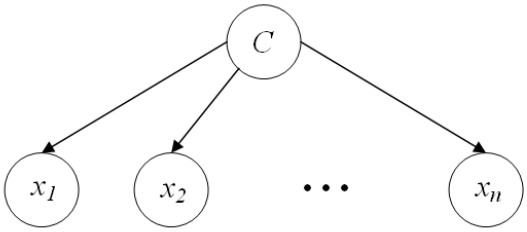

FIG. 2.5 – Graphe causal du classifieur bayésien naif

Chaque valeur $P(x_i|w_i)$ est estimée par comptage dans un intervalle. Une variation du classifieur bayésien naif, appelée réseau naif augmenté (*Tree Augmented Naive ou TAN*), est également utilisée pour lever l'hypothèse d'indépendance conditionnelle entre les attributs du modèle. Ce type de classifieur étend les propriétés du réseau naif en autorisant l'existence de liens entre les différents attributs sous une forme arborescente. Friedman *et al.* [63] montrent que ces deux types de structure relativement simple permettent d'obtenir des résultats comparables aux arbres de décision pour des problèmes de classification supervisée. Les réseaux bayésiens ont été appliqués avec succès à la classification de textes [139], au filtrage du courrier électronique [194], à la reconnaissance d'expressions faciales [163] et au diagnostic [71].

2.3.2 Arbres de décision

Les arbres de décision, souvent désignés par l'appréviation CART [18] pour *Classification And Regression Trees*, font partie des méthodes dites non paramétriques. Aucune hypothèse n'est faite sur la distribution conditionnelle de probabilité du modéle. C'est à partir d'une approche structurelle des données que les frontières de décisions dans l'espace des attributs sont déterminées.

La procédure de classification est apprise de manière hiérarchique, sous la forme d'un arbre où les nœuds correspondent aux attributs, et les feuilles portent les différentes classes du problème. La méthodologie employée pour contruire l'arbre de décision consiste à séparer un nœud en plusieurs branches par rapport à un attribut donné, de manière à maximiser la proportion d'objets appartenant à une même classe dans chacune des branches formées. Ce procédé est répété de manière récursive dans chacun des nœuds fils obtenus, jusqu'à obtention de sous-régions de l'espace des attributs dans lesquelles les exemples appartiennent tous à une même classe. Ce nœud est alors remplacé par une feuille étiquetée par la classe à laquelle appartiennent les objets y figurant. La division d'un nœud en plusieurs branches s'effectue généralement de façon binaire. Pour une variable continue x donnée, cette division consiste à séparer les objets par rapport à une valeur seuil $x_{\mathcal{S}}$ prédéfinie ou déterminée par apprentissage. Pour un attribut discret possedant m modalités, on préférera en général utiliser plusieurs séparations binaires successives plutot que de former m branches différentes. Un exemple d'arbre de décision est présenté dans la figure 2.6. Cet arbre a été appris sur le jeu de données *Iris* de l'UCI Repository, utilisé pour la première fois par Fisher [56]. Il modélise un problème de classication à 3 classes dans $\Re^4$. L'une des classes est linéairement séparable des deux autres, ce qui se traduit par l'obtention d'une feuille dès le second niveau. Les deux classes restantes présentent un recouvrement partiel, et nécessitent donc plusieurs séparation successives avant l'obtention d'une feuille. Remarquons que différentes feuilles peuvent correspondre à une même classe (ce qui est le cas dans l'exemple de la figure 2.6

pour les classes *iris-virginica* et *iris-versicolor*).

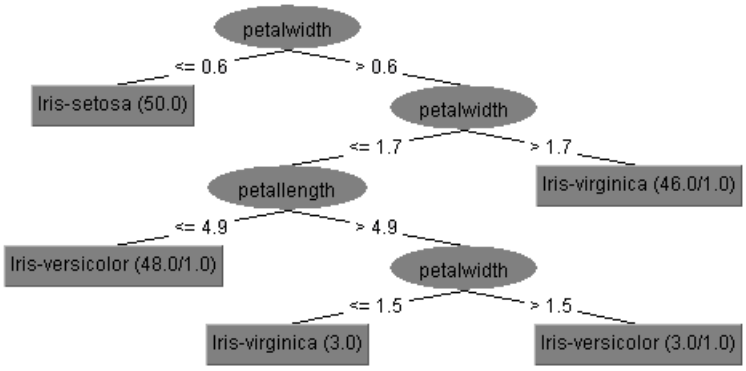

FIG. 2.6 – Arbre de décision obtenu par l'algorithme *C4.5*sur les données d'iris de Fisher.

Un exemple est classé en parcourant la structure arborescente modélisant la répartition des données dans $\Re^n$, ce qui revient à effectuer une séquence de décisions formulées à l'aide de règles élémentaires, depuis la racine de l'arbre jusqu'à ses feuilles, qui correspondent à la classe de l'objet. On peut considérer que chaque division d'un nœud en deux branches distinctes par rapport à un attribut donné revient à séparer l'espace des attributs de manière linéaire. La structure de l'ensemble de l'arbre permet donc de déterminer des régions de décision non linéairement séparables en effectuant une approximation linéaire par morceaux. La figure 2.7 représente les surfaces de décision obtenues pour un problème de classification dans $\Re^2$ en utilisant l'algorithme *C4.5* [170].

Pour déterminer l'attribut suivant lequel doit être divisé l'espace de représentation, on calcule pour chaque attribut un indicateur $Ind\,(n = x_i)$ permettant d'évaluer la répartition des objets dans les différentes branches qui seraient obtenues. On choisit l'attribut le plus discrimant, c'est à dire celui permettant de se rapprocher au maximum d'une branche dans laquelle une classe serait majoritairement représentée, voir unique. Soit un nœud de l'arbre n, pour lequel deux choix de variables x_1 et x_2 sont possibles pour éclater ce nœud. Supposons que les indicateurs $Ind\,(n = x_1)$ et $Ind\,(n = x_2)$ donnent une répartition des classes ω_1 et ω_2 en deux branches avec des proportions respectives égales à 50/50 et 90/10. On choisira alors d'éclater le nœud n par rapport à la variable x_2 qui maximise les chances d'obtention d'une feuille. Kuncheva [113] souligne que la méthode utilisée pour sélectionner les nœuds à éclater peut être assimilée à un algorithme de type *recherche gloutonne*. En effet, on effectue à chaque nœud un choix optimal selon un certain critère, de manière locale, sans pour autant garantir que l'arbre global obtenu sera optimal également. Les principaux indicateurs utilisés sont :

- le critère d'impureté de Gini [18] :

$$Ind\,(n = x) = 1 - \sum_{i=1}^{c} f(n, \omega_i)^2 \tag{2.23}$$

où $f(n, \omega_i)$ désigne la proportion d'exemples de l'ensemble d'entraînement de classe ω_i ayant atteint le nœud n. Cet indicateur, utilisé par l'algorithme CART, peut être vu comme une estimation de l'erreur de classification attendue au nœud n pour une classe choisie aléatoirement parmi Ω.

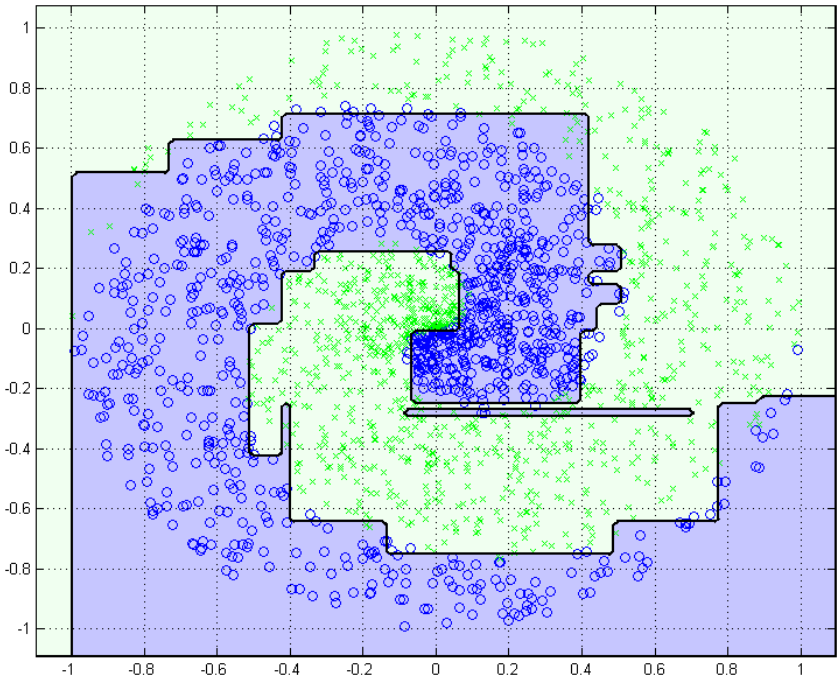

FIG. 2.7 – Frontières de décisions déterminées par l'algorithme *C4.5* pour deux classes non linéairement séparables.

- le gain d'information, basé sur la mesure de l'entropie de Shannon [201] :

$$Ind\,(n = x) = \sum_{i=1}^{c} f(n, \omega_i) \log f(n, \omega_i) \tag{2.24}$$

  La valeur minimale $Ind\,(n = x) = 0$ est obtenue pour une feuille. La plus mauvaise valeur possible $Ind\,(n = x) = \log c$ correspond à un nœud où chacune des classes possède une répartition uniforme. Le gain d'information est utilisé par les algorithmes *ID3* [169] et *C4.5* [170].
- l'indice d'erreur en classification :

$$Ind\,(n = x) = 1 - argmax_c f(n, \omega_c) \tag{2.25}$$

  Cet indice correspond à l'erreur qui serait obtenue dans le cas où le nœud n serait remplacé par une feuille portant la classe majoritairement représentée à ce nœud.

Duda *et al.* [172] soulignent que le choix du critère utilisé pour diviser chaque nœud a une influence limitée sur la précision de la procédure de classification obtenue comparativement au choix du critère d'arrêt ou de la méthode d'élagage utilisée pour affiner la structure de l'arbre.

La procédure décrite jusqu'à présent permet la construction de l'arbre de décision de manière descendante. Il est possible, dans certains cas de déterminer un arbre parfait, tel que les objets se rapportant à chacune des feuilles appartiennent à une seule et même classe. Un tel abre présente néanmoins un risque de sur-apprentissage élevé, puisqu'il caractériserait parfaitement l'ensemble

d'entraînement. Pour éviter d'induire un sur-apprentissage des exemples lors de la création de l'arbre, deux approches peuvent être envisagées : modifier le critère d'arrêt de l'algorithme, pour éviter une trop grande majoritée d'une classe au niveau des feuilles, ou appliquer des méthodes d'élagages (*Pruning*) de l'arbre après sa construction.

Différents critères d'arrêt sont proposés dans [172], en étudiant localement chaque nœud ou en considérant la structure de l'arbre dans son ensemble. Il est par exemple possible de définir un nombre minimal d'exemples à chaque nœud. Si pour un nœud donné, le nombre d'exemples ayant atteint ce nœud dans l'arbre est inférieur à cette valeur minimale, alors aucun éclatement n'est effectué, et le nœud concerné est remplacé par une feuille. Une variante de cette méthode consiste à étudier le critère utilisé pour l'éclatement d'un nœud. Si pour un nœud n donné, ce nœud vérifie $argmin_{x_i}\ Ind\,(n = x_i) \leq \beta$, le nœud est transformé en une feuille étiquetée par la classe majoritairement représentée. La structure globale de l'arbre peut également être pénalisée suivant sa compléxité, en cherchant à minimiser un critère de la forme :

$$\alpha \times taille(a) + \sum_{f(a)} Ind\,(n = f(a)) \tag{2.26}$$

où $f(a)$ désigne les feuilles de l'arbre a et $taille(a)$ est une valeur caractérisant sa compléxité, en termes de nœuds ou d'arcs. Cette méthode dépend du paramètre α à définir lors de l'entraînement du classifieur pour assurer un compromis entre la compléxité de la structure et son erreur en généralisation. Il est également possible d'utiliser en ensemble de validation pour estimer l'erreur prédictive de l'arbre à chaque étape de sa construction, et d'empêcher l'éclatement d'un nœud lorsque cette erreur tend à augmenter. Cette méthode suppose cependant qu'on dispose de suffisamment d'exemples d'entraînement pour apprendre correctement la structure de l'arbre. Un critère d'arrêt mal paramétré risque d'entraîner une situation de sous-apprentissage en limitant la structure apprise. Une autre solution consiste à construire un arbre complet, puis à affiner sa structure au cours d'une phase d'élagage, durant laquelle les branches non significatives et pouvant induire un sur-apprentissage des données d'entrainement sont supprimées. Plusieurs algorithmes ont été proposés dans [55]. Cette solution présente néanmoins un désavantage dans le cas où la dimension de l'ensemble d'apprentissage est très importante. Construire l'arbre complet pour ensuite lui superposer un phase d'élagage peut dans ce cas être prohibitif en terme de complexité.

Trois principaux algorithmes d'apprentissage sont cités dans la littérature : l'algorithme *CART*, introduit par Breiman *et al.* [18], ainsi que *ID3* [169] et *C4.5* [170], qui en sont des améliorations. *ID3* traite uniquement les attributs nominaux, et sépare un nœud en autant de branches que de modalités pour la variable considérée. Le problème se pose quant à l'utilisation d'attributs continus, qui doivent être préalablement discrétisés ce qui se entraîne une perte importante d'information. L'algorithme *C4.5* propose une alternative à la discrétisation d'attributs continus, en combinant les approches de *CART* et *ID3* respectivement pour les attributs continus et les variables nominales. Cet algorithme est l'un des plus utilisés en apprentissage automatique.

Les arbres de décision présentent de nombreux avantages et sont fréquemment utilisés pour des applications du monde réel. Voici ce qu'on peut notamment remarquer à propos de cette famille d'algorithmes :

- Ils offrent une interprétation simple et intuitive de la procédure de classification apprise sous forme de règles élémentaires, comparativement aux réseaux de neurones, qui s'apparentent davantage à un apprentissage de type *boîte noire*, où la sortie calculée est une fonction des entrées non explicitement définie. Leur structure arborescente permet également de mieux comprendre la structure des données en indiquant leurs liens.

- Ils sont relativement robustes face à la forme des données d'entrée. En effet, ils peuvent manipuler indépendamment des attributs qualitatifs ou quantitatifs, continus ou discrets. Les arbres de décision permettent également de manipuler des ensembles d'exemples pour lesquels certaines données sont manquantes. Ils offrent donc une plus grande souplesse quant aux données manipulées, comparativement à d'autres familles d'algorithmes qui nécessitent un pré-traitement des entrées tel que la normalisation des attributs ou la transformation de variables nominales en plusieurs variables binaires.
- L'apprentissage est relativement peu coûteux et cette famille d'algorithmes permet de traiter des données de grande dimension en un temps acceptable.
- Ces classifieurs sont néanmoins instables : de faibles changements dans les données utilisées pour construire la structure arborescente peuvent entraîner d'importantes modifications au niveau des règles de décision élémentaires apprises. Leur instabilité font des arbres de décision un choix efficace pour l'utilisation d'ensembles de classifieurs. Ce point sera développé plus en détail dans le chapitre suivant.

2.3.3 Réseaux de neurones

Formalisme et notations

Les réseaux de neurones (*Artificial Neural Network* ou *ANN* en anglais) [86, 87] sont issus d'un formalisme mathématique de l'organisation des neurones biologiques. La figure 2.8 représente le neurone formel tel qu'il est décrit à l'origine par McCulloch et Pitts [140].

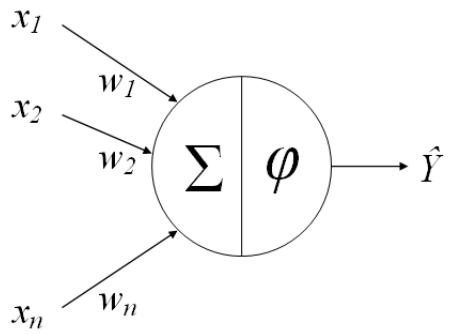

FIG. 2.8 – Représentation du neurone formel

Ce neurone est constitué des entrées observées $x_i \in \Re$, des poids synaptiques w_i associés à chaque entrée du neurone et d'une fonction d'activation, notée φ, utilisée pour calculer la sortie $\hat{Y} \in \Re$ du neurone. Cette sortie peut être vue comme l'activation du neurone en fonction de l'importance de sa stimulation. Cette stimulation, notée $\mathcal{S}$, correspond à la somme des entrées du neurone pondérée par leur poids synaptique :

$$\mathcal{S} = \sum_{i=1}^{n} w_i x_i \tag{2.27}$$

et la sortie du neurone s'écrit alors $\hat{Y} = \varphi(\mathcal{S})$. Les fonctions d'activation communément utilisées par le neurone formel sont :

- la fonction seuil, ou fonction de heaviside :

$$\varphi(\mathcal{S}) = \begin{cases} 1 & si \;\; \mathcal{S} > \theta \\ 0 & sinon \end{cases} \tag{2.28}$$

- la fonction sigmoïde :

$$\varphi(\mathcal{S}) = \frac{1}{1 + e^{-\mathcal{S}}} \tag{2.29}$$

- la fonction tangente hyperbolique :

$$\varphi(\mathcal{S}) = \frac{e^{\mathcal{S}} - e^{-\mathcal{S}}}{e^{\mathcal{S}} + e^{-\mathcal{S}}} \tag{2.30}$$

L'intérêt que présentent ces différentes fonctions d'activation est leur dérivabilité, une propriété nécessaire à la mise en oeuvre de l'algorithme d'apprentissage utilisé par les réseaux de neurones, comme nous le verrons plus loin. Pour une fonction d'activation donnée, les entrées x_i étant des données définies par le problème posé, seuls le vecteur des poids $W = (w_1, \ldots, w_n)$ doit être déterminé. Le perceptron, décrit dans [181], utilise un algorithme d'apprentissage pour déterminer les poids w_i à partir de l'observation de l'ensemble Z. Les poids, initialisés aléatoirement, sont modifiés lorsque la sortie calculée $\hat{Y} = \varphi(\mathcal{S})$ et la sortie réelle Y sont différentes, selon la règle :

$$W \leftarrow W + \alpha \cdot \mathcal{I}\left(\hat{Y}, Y\right) \cdot X \tag{2.31}$$

où $X = (x_1, \ldots, x_n)$ désigne le vecteur des entrées correspondant à un exemple de $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ et $\alpha \in [0, 1]$ est un paramètre appelé facteur d'apprentissage permettant d'ajuster la variabilité des poids et de stabiliser la convergence du modèle. La principale limitation de cette approche est qu'elle ne peut pas traiter les familles de problèmes où les classes sont non linéairement séparables [145]. En effet, si l'on considère la fonction d'activation de Heaviside, il est possible d'interpréter la sortie du perceptron comme l'équation d'un hyperplan dans $\Re^n$:

$$\sum_{i=1}^{n} w_i x_i - \theta = 0 \tag{2.32}$$

qui sépare l'espace des attributs en deux sous-régions correspondant à chacune des deux classes. Pour des problèmes faisant intervenir plus de deux classes non linéairement séparables, on utilise un réseau d'unités interconnectées. Dans la suite, nous nous intéressons à un modèle particulier de réseaux de neurones. La figure 2.9 présente l'architecture générale de ce type de classifieur : le perceptron multi-couches.

Les différentes unités sont organisées en couches successives. Les unités correspondant aux n attributs observés forment la couche d'entrée. La couche de sortie est constituée des neurones représentant les différentes fonctions discriminantes $g_i(X)$ données dans l'équation 2.2 et caractéristiques de chacune des classes du problème. Les couches intermédiaires sont appelées couches cachées. Leur nombre et leur taille sont définis au cours de la modélisation et constituent un degré de liberté de la structure du réseau. Cette architecture particulière est également appelée réseau sans rétroaction ou *feedforward network* : les unités d'une couche donnée sont complètement connectées avec les unités de la couche suivante. Ce type de réseau est fréquemment utilisé en classification du fait de sa faculté à approximer n'importe quelle fonction avec une précision donnée [94].

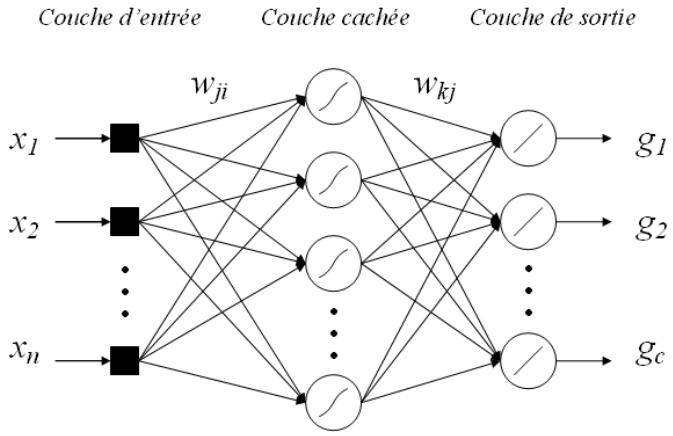

FIG. 2.9 – Architecture d'un réseau de neurones multi-couches sans rétroaction comportant une couche cachée

Apprentissage dans les réseaux de neurones

Une autre raison justifiant l'emploi intensif de ce type de classifieur repose sur l'algorithme d'apprentissage mis en oeuvre. L'entrainement du perceptron multi-couches s'effectue de manière empirique selon le paradigme professeur-élève. Il s'agit d'un processus adaptatif, consistant à présenter de manière itérative au réseau une série d'exemples tout en évaluant son erreur. La procédure de classification apprise est modifiée à chaque itération de manière à minimiser l'erreur obtenue sur cet ensemble d'exemples par une descente de gradient. Le modèle est modifié selon l'importance des éléments ayant participé à la réalisation des erreurs. Dans le cas des réseaux de neurones, les poids synaptiques qui contribuent à engendrer une erreur importante sont modifiés de manière plus significative que les poids qui ont engendré une erreur moindre. Il faut déterminer la contribution de chaque poids synaptique à cette erreur, et propager la modification de ces poids synaptiques depuis les unités de sorties jusqu'aux unités se trouvant dans la couche d'entrée, d'où l'appellation de rétropropagation du gradient.

La fonction d'erreur utilisée est définie par :

$$E(w) = \frac{1}{2} \sum_k (Y_k - \hat{Y}_k) \tag{2.33}$$

où Y_k et $\hat{Y}_k$ désignent respectivement la sortie attendue et la sortie calculée par le réseau au niveau de la k-ième unité. La modification des poids est donnée par :

$$w \leftarrow w + \Delta w \tag{2.34}$$

avec

$$\Delta w = -\alpha \frac{\partial E(w)}{\partial w} \tag{2.35}$$

Par exemple, pour une architecture sans rétroaction comportant une seule couche cachée comme pour le réseau de la figure 2.9, la modification des poids est donnée par :

$$\Delta w_{kj} = -\alpha \frac{\partial E}{\partial w_{kj}} = \alpha (Y_k - \hat{Y}_k) f'(a_k) \hat{Y}_j \tag{2.36}$$

$$\Delta w_{ji} = -\alpha \frac{\partial E}{\partial w_{ji}} = \alpha \left(\sum_k w_{kj} (Y_k - \hat{Y}_k) f'(a_k) \right) f'(a_j) x_i \tag{2.37}$$

avec $\hat{Y}_k = f(a_k) = f\left(\sum_j w_{kj} y_j - w_{k0}\right)$ et $\hat{y}_j = f(a_j) = f\left(\sum_j w_{ji} x_i - w_{j0}\right)$ comme fonctions discriminantes respectivement pour les unités de sorties et les unités cachées.

La généralisation de ce calcul pour un réseau comportant plus d'une couche cachée fonctionne de manière analogue, et est décrite dans [182, 24]. Remarquons que l'algorithme présenté ci-dessus est de type *online*, c'est-à-dire que les poids synaptiques sont mis à jour pour chaque exemple d'apprentissage ayant entrainé une erreur. Pour des raisons de convergence et de temps de calcul, on utilise en pratique des algorithmes de type *batch*, qui mettent à jour les poids en calculant la fonction de coût sur l'ensemble des exemples d'apprentissage. Différentes modifications relatives à la convergence de cet algorithme sont décrites dans [79].

Un autre type de neurone formel, appelé neurone sphérique peut également être utilisé. Ce type de neurone utilise une fonction d'activation s'appliquant à la distance entre le vecteur d'entrée X et le neurone : $\varphi(||X - W||)$ où X désigne le vecteur des entrées et W représente de centre de la fonction de base. Ce type d'unité est également appelé neurone à fonction de base radiale. Un réseau de neurones sphérique comporte généralement une seule couche cachée. Différents algorithmes d'apprentissage pour ce type de classifieur sont proposés dans [156, 185].

Un perceptron multi-couches sépare les classes en formant des hyperplans dans l'espace des attributs $\Re^n$. Un réseau de neurones sphériques détermine des régions de décisions par agrégation d'hypersphères dans ce même espace. La figure 2.10 présente de manière simplifiée comment ces deux types de réseaux neurones permettent d'approximer des surface de séparations non linéaires.

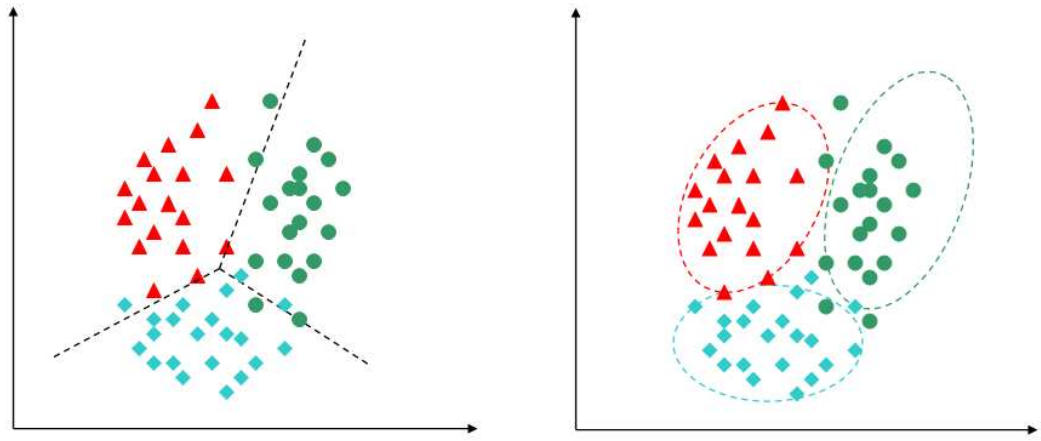

FIG. 2.10 – Régions de décisions obtenues à l'aide d'un réseau de neurones à fonction d'activation linéaire (à gauche) et à fonction de base radiale (à droite)

Leurs propriétés d'approximateurs universels font des réseaux de neurones un outil particulièrement utilisé en classification supervisée [197, 81, 106]. Ces classifieurs sont dits instables du fait qu'un changement au niveau des données d'apprentissage peut induire d'importantes modifications au niveau de l'approximation des fonctions cibles. Cette instabilité peut cependant être

mise à profit lorsque plusieurs réseaux sont utilisés conjointement dans un ensemble de classifieurs, comme nous le verrons dans le chapitre suivant. Le principal désavantage de ce type de classifieur réside dans leur tendance à sur-apprendre les exemples d'entraînement. Tetko *et al.* [211] définissent deux principales formes de sur-apprentissage, en distingant le sur-entrainement et la sur-paramétrisation d'un réseau de neurone. Comme il a été expliqué précédemment, l'apprentissage par propagation de l'erreur est une méthode itérative, consistant à modifier les poids à estimer jusqu'à ce que l'erreur d'apprentissage atteigne un seuil prédéfini, ou qu'un nombre maximal d'itérations soit atteint. Le sur-entrainement survient généralement lorsque le nombre d'itérations de l'algorithme est trop important. Le réseau apprend alors parfaitement la structure des données d'entraînement, au détriment de sa faculté à généraliser et à classer correctement de nouveaux exemples. Ce phénomène peut être évité en réservant une partie de l'ensemble d'apprentissage pour arrêter l'entraînement à son point optimal, lorsque l'erreur en généralisation tend à augmenter [186]. La sur-paramétrisation est liée à la compléxité du réseau. Ce type de classifieurs possède un grand nombre de degrés de liberté. Le choix de la strucure, de la taille et du nombre de couches cachées, ou de la valeur du coefficient d'apprentissage doivent être paramétrés en fonction du problème posé. Les réseaux de neurones étant capables d'approximer n'importe quelle fonction non linéaire des entrées, une structure trop complexe conduit le réseau à apprendre le bruit contenu dans les données d'entraînement. Il n'existe pas de moyen prédéfini pour estimer automatiquement la taille et la structure à adopter. Le choix des paramètres s'effectue généralement en utilisant les méthodes de validation croisée présentées dans la section précédente. Plusieurs alternatives à l'initialisation aléatoire des poids du réseau sont proposées dans [152, 160] pour améliorer l'apprentissage. Des algorithmes permettant de déterminer la structure du réseau parallèlement à l'estimation des poids sont également recensés dans [38, 216].

2.3.4 Machines à Vecteur de Support

La théorie de l'apprentissage statistique

La théorie de l'apprentissage statistique proposée par Vapnik [220] rejette l'hypothèse selon laquelle la minimisation des erreurs sur l'ensemble d'apprentissage, tel le principe du maximum de vraisemblance, induit implicitement une bonne généralisation du modèle. Cette théorie définit l'apprentissage comme étant la recherche d'un principe d'induction qui conduit explicitement à une capacité de généralisation du modèle la plus grande possible. L'apprentissage statistique s'appuie sur de solides fondements mathématiques, qu'il n'est pas possible de détailler de manière exhaustive dans ce mémoire. Vapnik propose une formulation du risque effectif d'erreur d'un classifieur permettant un contrôle des propriétés de généralisation de ce dernier. Ce risque est garanti avec une probabilité au moins égale à $1 - \eta$:

$$R(C) = R_{emp}(C) + \Delta \quad avec \quad \Delta = (b-a)\sqrt{\frac{h(\ln(2N/h)+1) - \ln(\eta/4)}{N}} \tag{2.38}$$

où Δ est une quantitée appelée *intervalle de confiance* représentant l'écart entre le risque effectif $R(C)$, que l'on cherche à minimiser pour garantir les facultés prédictives du modèle sur des échantillons inconnus, et le risque empirique $R_{emp}(C)$, mesurable à partir de l'ensenble d'apprentissage. La quantité Δ est fonction de la taille N de l'ensemble d'entraînement, et d'une donnée sur la structure du modèle, notée h dans l'équation 2.38, et appelée *dimension de Vapnik-Chervonenkis VC-dimension* [221]. La *VC-dimension* d'un modèle correspond, de manière simplifiée, au nombre maximum d'exemples que ce modèle est capable de distinguer.

Remarquons que :

$$\lim_{N\to\infty} \Delta = 0 \tag{2.39}$$

pour une valeur arbitrairement grande de N, le terme prédominant dans la borne du risque effectif est le risque empirique. Dans ce cas, une minimisation du risque empirique permet à elle seule de contrôler la capacité de généralisation du modèle et donne une estimation fiable de ses facultés prédictive sur de nouveaux exemples. A l'inverse, lorsque le nombre d'échantillons disponibles est faible, ce qui est le cas pour la plupart des applications réelles de l'apprentissage automatique, l'intervalle de confiance peut prendre une valeur importante. La minimisation du risque empirique n'offre donc pas de garantie suffisante pour estimer l'erreur en généralisation du modèle.

La formulation donnée dans l'équation 2.38 illustre le principe fondamental de la théorie de l'apprentissage statistique. Le nombre N d'exemples disponibles étant, pour la plupart des applications usuelles, une constante fournie par le problème posée, la quantité Δ n'est fonction que de la VC-dimension h, propre à la structure du modèle. La seule façon de garantir la capacité de généralisation du modèle consiste alors à contrôler sa structure durant l'apprentissage. Ce principe inductif est appelé minimisation du risque structurel et consiste à minimiser conjointement le risque empirique et l'intervalle de confiance Δ au cours de l'apprentissage.

Les machines à vecteurs de supports

Les Machines à Vecteurs de Support ou *SVM* [30, 21] constituent une famille d'algorithmes d'apprentissage permettant de contrôler la capacité de généralisation des modèles qu'ils produisent. En s'appuyant sur le principe de minimisation du risque structurel, cette famille d'algorithmes formule l'apprentissage sous la forme d'un problème d'optimisation cherchant à miniser l'intervalle de confiance Δ.

Soit un ensemble d'apprentissage $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ avec $X_i \in \Re^n$ et $Y_i \in \Omega = \{-1,\ 1\}$ pour lequel les exemples sont linéairement séparables par une fonction $f(X) = W.X + b$. Un hyperplan *admissible* permettant de séparer ces exemples dans l'espace des attributs a une équation de la forme $W.X + b = 0$, et vérifie :

$$(W.X_i + b) = +1 \quad si \quad Y_i = +1 \tag{2.40}$$
$$(W.X_i + b) = -1 \quad si \quad Y_i = -1 \tag{2.41}$$

L'hyperplan admissible pour lequel la distance aux points les plus proches est maximale est dit optimal, car il maximise la marge entre les deux sous régions de l'espace ainsi formées. La marge étant égale à $2/||W||$, maximiser la marge revient à minimiser la norme $||W||$ sous les contraintes données dans l'équation 2.40. La reformulation du problème d'optimisation s'exprime :

$$\min \frac{1}{2}||W||^2 \tag{2.42}$$
$$(W.X_i + b)Y_i \geq 1 \quad \forall i \in \{1, \ldots, N\} \tag{2.43}$$

Une telle approche minimise le risque empirique tout en cherchant à améliorer les propriétés en généralisation du modèle appris par maximisation de la marge géométrique entre les deux régions de décision ainsi formées. Une illustration est donnée par la figure 2.11. Intuitivement, le fait de maximiser la marge permet une séparation des exemples plus générale, c'est à dire

qui minimise l'intervalle de confiance. C'est pour cette raison que les SVM reçoivent parfois l'appelation de *Séparateurs à Vaste Marge*.

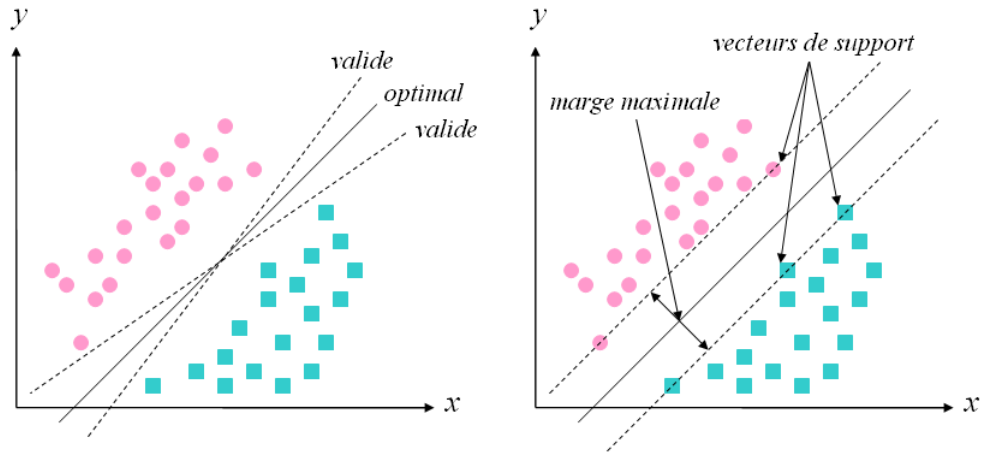

FIG. 2.11 – Hyperplan séparateur optimal maximisant la marge entre les classes.

Le problème d'optimisation donné dans l'équation 2.42 est résolu en introduisant des multiplicateurs de Lagrange dans sa formulation duale :

$$\max \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j Y_i Y_j X_i \cdot X_j \tag{2.44}$$

$$\forall i,\ \alpha_i \geq 0 \tag{2.45}$$

$$\sum_{i=1}^{N} \alpha_i Y_i = 0 \tag{2.46}$$

Déterminer l'hyperplan optimal revient à résoudre le problème d'optimisation quadratique donné dans l'équation 2.44, dont la dimension est égale au nombre de contraintes, c'est à dire à la taille N de l'ensemble d'entraînement. Les α_i^* solutions de ce problème sont non nuls et vérifient l'équation :

$$W^* = \sum_{i=1}^{N} \alpha_i^* Y_i X_i \tag{2.47}$$

Les α_i^* représentent les vecteurs de support donnés sur la figure 2.11 et correspondent aux points les plus proches de l'hyperplan séparateur. La résolution du problème d'optimisation donné dans l'équation 2.44 permet de déterminer l'équation de l'hyperplan optimal :

$$\sum_{i=1}^{N} \alpha_i^* Y_i X_i \cdot X + b = 0 \tag{2.48}$$

Pour déterminer la classe d'un nouvel exemple, il suffit de calculer le signe de l'expression donnée par le membre de gauche de l'équation 2.48 et ainsi savoir dans quelle sous-région de l'espace se trouve ce nouvel exemple.

Lorsqu'il n'existe pas d'hyperplan séparateur satisfaisant l'ensemble des contraintes du problème d'optimisation, il est possible de reformuler les contraintes données dans les équations 2.40 pour trouver un hyperplan maximisant la marge mais pour lequel il existe des exemples situés dans le mauvais demi-espace formé. Cortes et Vapnik [30] définissent le concept de marges souples, en introduisant des variables ressort ξ_i au niveau des contraintes et en pénalisant la violation de ces dernières par un coût de dépassement C. La reformulation du problème s'écrit :

$$\min \frac{1}{2}||W||^2 + C\sum_{i=1}^{N}\xi_i \tag{2.49}$$

$$(W.X_i + b)Y_i \geq 1 - \xi_i \quad \forall i \in \{1, \ldots, N\} \tag{2.50}$$

Une grande valeur de C signifie une forte pénalisation du dépassement de contrainte. Ce nouveau problème est également résolu sous sa forme duale :

$$\max \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j Y_i Y_j X_i \cdot X_j \tag{2.51}$$

$$\forall i, 0 \leq \ \alpha_i \leq C \tag{2.52}$$

$$\sum_{i=1}^{N}\alpha_i Y_i = 0 \tag{2.53}$$

La formulation donnée par l'équation 2.51 ne diffère de l'équation 2.44 que par la borne supérieure sur les α_i.

La méthodolgie décrite jusqu'à présent n'est applicable que dans le cas où les exemples sont linéairement séparables. Pour adapter aux cas les plus généraux, l'idée de base des SVM est qu'il est possible de représenter les vecteurs d'entrée dans un espace de description suffisamment grand pour qu'ils puissent être linéairement séparables par un hyperplan maximisant la marge, indépendamment de la dimension de cet espace. Cet espace est généralement appelé espace de *redescription* et est obtenu par une transformation non linéaire de l'espace des attributs notée Φ :

$$\begin{aligned} \Phi : \Re^n &\rightarrow \Re^p \quad p > n \\ X &\mapsto \Phi(X) \end{aligned} \tag{2.54}$$

Le fait qu'un problème puisse devenir linéairement séparable lorsqu'on augmente la dimension de l'espace de représentation interne des objets peut trouver une justification théorique dans des travaux concernant les réseaux de neurones multi-couches décrits dans [75]. Une illustration de ce principe est donnée dans la figure 2.12.

Le problème d'optimisation à résoudre dans l'espace de redescription est donné par :

$$\max \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j Y_i Y_j \Phi(X_i) \cdot \Phi(X_j) \tag{2.55}$$

$$\forall i, 0 \leq \ \alpha_i \leq C \tag{2.56}$$

$$\sum_{i=1}^{N}\alpha_i Y_i = 0 \tag{2.57}$$

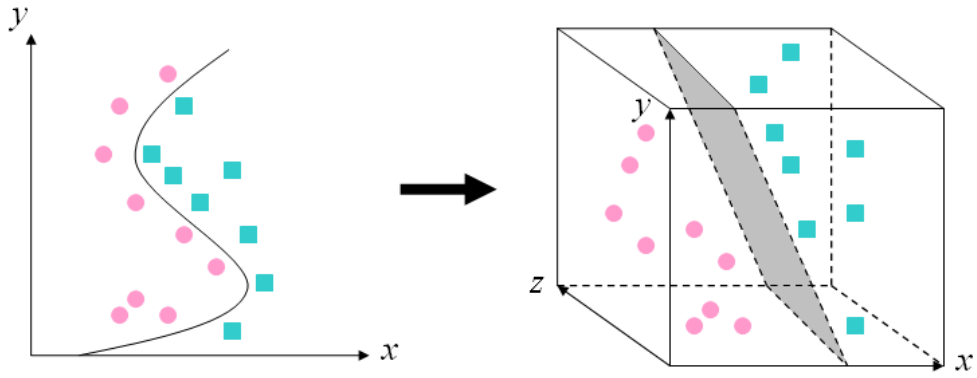

FIG. 2.12 – Hyperplan séparateur obtenu dans l'espace de redescription. A gauche, dans l'espace des attributs, les exemples sont non linéairement séparables. Dans l'espace de redescription, à droite, il est possible de séparer ces mêmes exemples à l'aide d'un hyperplan.

et admet une solution de la forme :

$$\sum_{i=1} N\alpha_i^* Y_i \Phi\left(X_i\right) \cdot \Phi\left(X\right) + b = 0 \tag{2.58}$$

La solution du problème d'optimisation dans l'espace de redescription ne dépend que des produits scalaires $\Phi\left(X_i\right) \cdot \Phi\left(X\right)$. Les travaux de recherche exposés dans [11] s'appuient sur cette constatation pour proposer une méthode permettant de déterminer l'hyperplan séparateur sans avoir à définir de manière explicite la fonction de transformation Φ. La justification théorique est qu'il est possible d'exprimer un produit scalaire à l'aide d'une fonction noyau $K(X, X') = \Phi(X) \cdot \Phi(X')$, symétrique, et vérifiant les conditions de Mercer (à savoir la positivité et la symétrie) [193]. Lorsque la fonction noyau est bien choisie pour représenter le produit scalaire dans l'espace de redescription, il est possible de calculer directement les hyperplans séparateurs dans ce nouvel espace. On désigne parfois l'utilisation des fonctions noyaux pour déterminer l'hyperplan de marge maximale par l'appellation *kernel trick* [192].

En pratique, il n'est pas toujours facile de vérifier qu'une fonction vérifie les conditions de Mercer. Les fonctions noyaux usuelles manipulées par les machines à vecteurs de support sont :

- la fonction noyau linéaire :

$$K\left(X, X'\right) = X \cdot X' \tag{2.59}$$

- la fonction noyau polynomiale :

$$K\left(X, X'\right) = \left(X \cdot X' + c\right)^d \tag{2.60}$$

- la fonction noyau Gaussienne :

$$K\left(X, X'\right) = \exp(-\frac{1}{\sigma}||X - X'||^2) \tag{2.61}$$

Le choix de la fonction noyau,de ses paramètres et de la valeur du coût de dépassement des contraintes C sont fixés avant de résoudre le problème d'optimisation. Les noyaux polynomiaux peuvent donner des temps de résolution excessivement longs. On préfèrera en général utiliser un noyau Gaussien pour obtenir plus rapidement la solution optimale du problème. Une faible

valeur pour C facilite la résolution du problème d'optimisation au détriment de l'optimalité de la solution obtenue. A l'inverse, la complexité du problème étant fonction du nombre de contraintes, une valeur trop importante pour C augmente de manière significative le temps de calcul.

La principale limitation des SVM vient de leur incapacité à traiter directement les problèmes comportant plus de deux classes. En effet, l'implémentation décrite jusqu'à présent n'est applicable que pour des problèmes de classification binaires, comportant deux classes notées arbitrairement $\omega_1 = +1$ et $\omega_2 = -1$, du fait de la formulation des contraintes dans le problème d'optimisation décrit dans l'équation 2.40. Pour étendre l'utilisation des machines à vecteurs de supports à des applications comportant plus de deux classes, le problème initial est décomposé en un ensemble de sous-problèmes binaires, résolus par la méthode décrite précédemment. Différentes adaptations possibles de SVM multi-classes sont proposées dans [23, 43, 2]. Nous reviendrons plus en détail sur ces différentes décompositions dans le chapitre suivant, dédié aux ensembles de classifieurs. Les SVM constituent une approche récente au problème de la classification, et ont été utilisées avec succès à des applications issues du monde réel [61, 155]. Par ailleurs, différentes adaptations des SVM ont été proposées dans le cas de la régression [49, 219].

2.3.5 K plus proches voisins

Les k plus proches voisins [200, 34] désignent une famille d'algorithmes traitant le problème de l'apprentissage automatique comme l'identification de prototypes [1]. Utilisé conjointement à une mesure de distance pour définir la notion de proximité, ce type d'algorithme permet de classer un échantillon en identifiant les prototypes présentant le plus de similitudes. Les k plus proches voisins considèrent les prototypes contenus dans Z comme autant de points dans l'espace des attributs R^n. Pour déterminer la classe d'un nouvel exemple X', les distances $d(X',\ X_i),\quad i = 1, \ldots, N$ entre ces echantillons et les N prototypes contenus dans Z sont calculées. Plus cette distance $d(X',\ X_i)$ est faible, plus la similarité entre X' et le prototype X est grande. Les classes des k plus proches prototypes sont ainsi identifiées et l'exemple à classer se voit attribuer la classe majoritairement représentée parmi eux :

$$C(X) = argmax_c \sum_{i=1}^{k} k_i |\ k_i = \omega_c \tag{2.62}$$

La figure 2.13 illustre l'attribution d'une classe à un échantillon par utilisation des algorithmes 3-ppv et 7-ppv.

Remarquons que les méthodes basées sur l'utilisation de prototypes ne réalisent pas d'apprentissage au même titre que les algorithmes décrits précédemment. En effet, contrairement aux algorithmes approximant des régions de décision en calculant les fonctions discrimantes $g_i(X)$ à partir d'exemples, les k plus proches voisins se contentent de mémoriser les exemples d'entraînement, le calcul des distances et l'identification des prototypes sont différés et systématiquement réévalués pour classer chaque nouvel exemple. Il est cependant possible de formaliser le fonctionnement des méthodes basées sur l'utilisation de prototypes selon l'estimation locale de densités de probabilités. Les méthodes d'apprentissage bayésiennes déterminent les fonctions discriminantes $g_i(X) = \hat{P}(\omega_i|X)$ de manière globale, à partir des données contenues dans l'ensemble de données Z. Les méthodes basées sur l'utilisation de prototypes s'en distinguent par le fait qu'elles doivent réestimer localement les fonctions discriminantes pour chaque exemple à classer, ce qui est d'ailleurs leur principal inconvénient. En effet, ce mode de fonctionnement peut rendre l'utilisation des méthodes basée sur l'identification de prototypes particulièrement coûteuses en temps lorsque la dimension de l'ensemble d'apprentissage Z est importante.

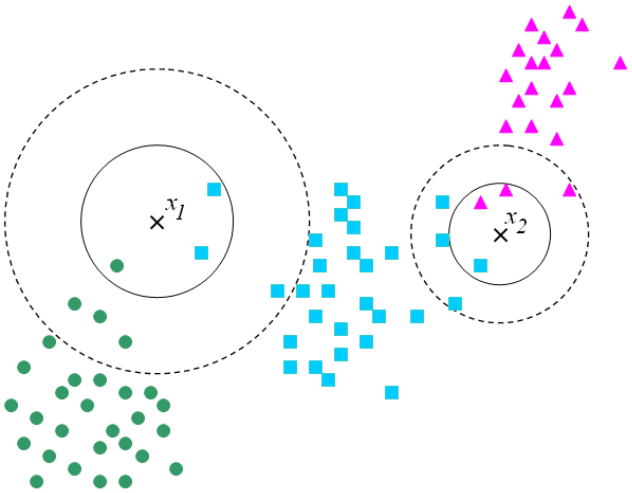

FIG. 2.13 – Classification de deux objets x_1 et x_2 par l'algorithme des k plus proches voisins. Dans le cas où $k = 3$ (cercle continu), les classes affectées aux objets x_1 et x_2 sont respectivement *carré* et *triangle*. Pour $k = 7$ (cercle discontinu), x_1 et x_2 se voient attribuer les étiquettes respectives *rond* et *carré*.

Aha *et al.* [1] décrivent d'autres inconvénients caractéristiques de ce type d'algorithme. Les k-ppv sont sensibles au bruit présent dans les données d'apprentissage. Ce phénomène se ressent encore davantage pour des valeurs de k très faibles. Ce type d'algorithme présente également une faible tolérance au niveau des données en entrées. En effet, il n'existe pas de méchanisme naturel permettant la prise en compte de données manquantes et de variables nominales dans le calcul des distances euclidiennes. Un problème survient également pour classer un nouvel objet en cas d'absence d'une classe majoritaire parmi les k plus proches voisins considérés.

L'algorithme des k plus proches voisins peut être vu comme une forme locale d'apprentissage bayésien. Lorsqu'on désire déterminer la classe d'un nouvel exemple X, on estime les densités de probabilités de chaque classe $\hat{P}(\omega_i|X)$ localement, en comptant le nombre $k_i, \quad i = 1, \ldots, c$ d'éléments de chaque classe dans un voisinage de X :

$$k = \sum_{i=1}^{c} k_i \tag{2.63}$$

Le nombre k de voisins considérés définit dans l'espace des attributs $\Re^n$ un voisinage, notée R, centrée en X et de volume V_R. Evaluer k_i dans R revient à estimer les probabilités *a posteriori* de chaque classe dans le voisinage R :

$$\hat{P}(\omega_i|X) = \frac{k_i N}{V_R} \tag{2.64}$$

où N désigne le nombre d'échantillons total. Le théorème de Bayes donné dans l'équation 2.7 nous permet d'écrire :

$$P(\omega_i|X) \approx \frac{k_i}{k} \tag{2.65}$$

Le risque d'erreur des k plus proches voisins tend vers le risque d'erreur du classifieur optimal au sens de la théorie de Bayes pour un nombre de voisins k considéré très grand :

$$\lim_{N\to\infty,\ k\to\infty,\ k/N\to 0} P_{k-ppv} = P_e \tag{2.66}$$

Le choix du voisinage R peut être effectué selon deux approches :

- En fixant arbitrairement le volume V_R de la région considérée de manière à ce qu'il contienne un nombre fixé k d'éléments : c'est la méthode des k plus proches voisins usuelle décrite précédemment.
- Soit en définissant la forme du voisinage R à l'aide d'une fonction croissante de l'espace des attributs : c'est l'algorithme de Parzen [159, 99].

Remarquons que pour le cas particulier où $k = 1$, l'erreur du classifieur 1-ppv est bornée par deux fois l'erreur optimale de Bayes :

$$P_{1-ppv} \leq 2P_e \tag{2.67}$$

Il est possible de représenter les surfaces séparatrices définies par le classifieur 1-ppv à l'aide d'un diagramme de Voronoi tel que celui décrit dans la figure 2.14.

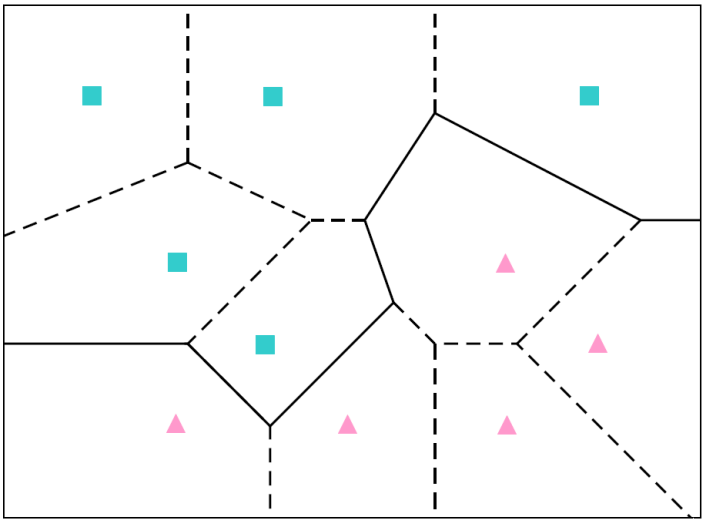

FIG. 2.14 – Zone de Voronoi obtenue par application du classifieur *1-ppv* dans un espace de dimension 2.

On appelle zone de Voronoi $V(X_i)$ d'un exemple d'entrainement X_i le sous-espace dans lequel tout objet $X' \in \Re^n$ est plus proche de X_i que de tout autre exemple :

$$V(X_i) = \left\{ X' \mid X' \in \Re^n,\ d(X_i, X') = \min_{X_j \in Z} d(X_j, X') \right\} \tag{2.68}$$

Dans le cas d'un espace de dimension 2, on obtient les régions de décision illustrées dans la figure 2.14. La frontière de décision entre deux exemples d'apprentissage de classes différentes se situe sur la médiatrice qui sépare ces deux exemples. Pour un ensemble de N exemples d'apprentissage, le diagramme de Voronoi se construit donc en déterminant les médiatrices qui séparent chaque couple de points. Les zones de Voronoi des différents exemples d'entrainement partitionnent alors l'espace des attributs en N polyèdres convexes d'intersections nulles. La surface

séparatrice entre deux classes (représentée par une ligne continue sur la figure 2.14)est la surface séparatrice entre les deux volumes obtenus en faisant l'union des surfaces de Voronoi des exemples de chaque classe.

L'algorithme des k-ppv dépend principalement du nombre k de voisins considérés. Augmenter la valeur de k permet en général de diminuer l'influence du bruit contenu dans les données. A l'inverse, une valeur exagérément grande entraîne le plus souvent l'apparition de frontières de décision imprécises, et tend à augmenter l'erreur en généralisation de l'algorithme, comme le montre la figure 2.15. Une valeur communément utilisée pour assurer un compromis efficace du nombre de voisins considéré est donné par :

$$k = \left\lfloor \sqrt{\frac{N}{c}} \right\rfloor \tag{2.69}$$

où $\frac{N}{c}$ désigne le nombre moyen d'exemples d'entraînement par classe. La valeur optimale de k pour un problème peut également être estimée en utilisant des méthodes de validation croisées.

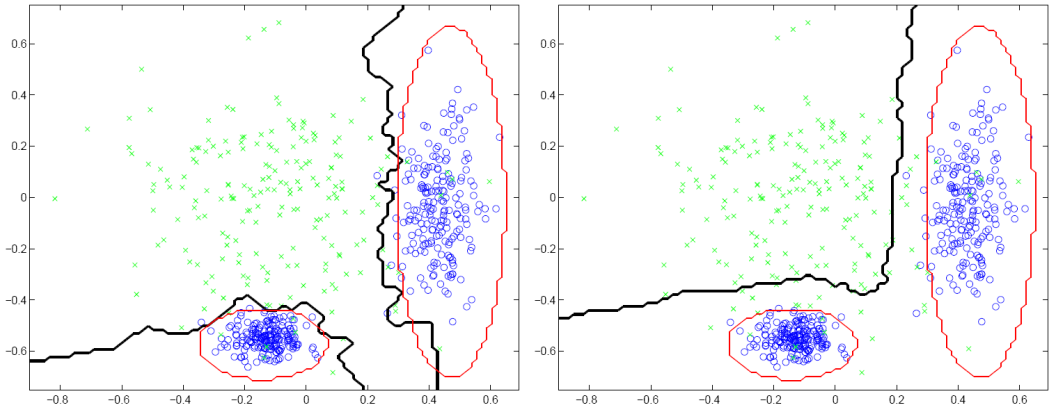

FIG. 2.15 – Régions de décision obtenues pour k=3 (à gauche) et k=50 (à droite). L'augmentation du nombre de voisins considérés engendre des frontières de décisions moins précises.

Le choix de la distance utilisée peut également être considéré comme un degré de liberté de l'algorithme. Pour résoudre l'impossibilité des k-ppv à manipuler des distances pour des variables nominales, Aha *et al.* [1] proposent l'utilisation de la métrique suivante :

$$d(X, X') = -\sqrt{\sum_{i=1}^{n} f(x_i, x'_i)} \tag{2.70}$$

avec

$$f(x_i, x'_i) = \begin{cases} (x_i - x'_i) & \text{si le i-ème attribut est continu} \\ 1 - \mathcal{I}(x_i, x'_i) & \text{si le i-ème attribut est nominal} \end{cases} \tag{2.71}$$

et où $\mathcal{I}(x_i, x'_i)$ désigne la fonction définie dans l'équation 2.15. Cette métrique propose en outre une solution partielle au problème des données manquantes, en supposant qu'une valeur manquante est toujours significativement différente de celle disponible. Pour remédier à l'absence de classe majoritaire lors de l'attribution de la classe d'un nouvel exemple, il est possible de

remplacer le vote majoritaire usuel donné dans l'équation 2.62 par un vote pondéré par l'inverse de la distance aux exemples d'entrainement :

$$C(X) = argmax_c \sum_{i=1}^{k} \frac{1}{d(X, k_i)} | \ k_i = \omega_c \tag{2.72}$$

Ainsi, plus un exemple d'entraînement est proche du point à classer, plus sa contribution au vote est importante.

Les principales variantes de l'implémentation originale des k-ppv portent sur l'édition de l'ensemble d'entraînement et la sélection de prototypes. En effet, la compléxité de l'algorithme est de l'ordre de $O(Nn)$, N étant le nombre d'exemples disponibles et n le nombre d'attributs caractérisant chaque exemple, ce qui représente son principal handicap pour un ensemble d'apprentissage de grande dimension. On va donc chercher à réduire l'ensemble d'apprentissage par condensation pour accélérer la décision sans changer le résultat des futurs décisions de l'algorithme. Deux méthologies permettent de procéder à une réorganisation de l'ensemble d'apprentissage : par sélection de prototypes à partir de l'ensemble d'entrainement, ou par le calcul de nouveaux prototypes à partir des données initiales. Différentes implémentations pour l'édition de Z sont proposées dans [34]. Elles sont le plus souvent basées sur des heuristiques permettant d'extraire un sous-ensemble de Z sans entrainer de diminution des facultés prédictives de l'algorithme. Les stratégies permettant de construire des prototypes à partir des données d'entraînement sont nombreuses, et il n'est pas possible dans le cadre de ce chapitre dans dresser une liste exhaustive. Nous pouvons néanmoins citer les algorithmes de clustering compétitif [7], les techniques basées sur la descente du gradient [36] et les méthodes utilisant le bootstrap [82].

2.3.6 Identification par modèle flou à partir des données

Les travaux de Zadeh [230] sur la théorie des ensembles flous ont défini les bases de la modélisation et de la commande floue utilisées en automatique et en informatique. Cette théorie propose de traiter l'incertitude dans le raisonnement et la conception de modèles, en définissant des ensembles *flous* avec une inclusion graduelle, par opposition à l'utilisation d'ensembles et de valeurs numériques précis (*crisp*). La théorie des ensembles flous est actuellement utilisée dans de nombreux domaines tels que la commande de systèmes automatiques, le traitement du signal, la classification de données ou les systèmes à base de connaissances. Bien qu'ils soient moins fréquemment utilisés comparativement aux méthodes issues de l'Apprentissage Automatique décrites dans ce chapitre, les modèles flous peuvent être utilisés dans un contexte de classification supervisée et de régression. Nous présentons dans cette section une description générale des modèles flous, et nous nous intéressons plus particulièrement à l'identification neuro-floue de données entrées-sorties, inspirée du modèle de Takagi-Sugeno (TS) [210], et utilisée dans un contexte de classification supervisée.

Modèles flous : description générale

De manière générale, un modèle flou est défini par un ensemble de régles permettant de représenter le comportement d'un système à l'aide de relations entrées-sorties de la forme :

Si premisse **Alors** conséquence

La *prémisse*, également appelé antécédent, décrit l'état dans lequel se trouve le système que l'on cherche à modéliser. Il est constitué par les différentes variables en entrée. La conséquence, ou conclusion, représente le comportement ou sortie du système correspondant à l'état décrit par la

prémisse. Cette modélisation s'apparente aux systèmes experts, à la différence majeure que les règles utilisées sont dites *floues*, car elles associent aux variables modélisant l'état du système un degré d'appartenance à différents ensembles flous. Pour illustrer cette distinction, considérons par exemple le modèle décrit par les deux règles suivantes :

R_1 : **Si** cholesterol est élevé **et** activité sportive est faible **alors** risque d'infarctus est important
R_2 : **Si** cholesterol est faible **et** activité sportive est régulière **alors** risque d'infarctus est faible

L'état du système est décrit par les deux variables *cholesterol* et *activité sportive* dont les valeurs sont définies par les termes linguistiques *élevé*, *faible* et *régulière*. De la même manière, la conclusion du système est définie par une variable de sortie *risque d'infarctus* définie par un autre terme linguistique. Un système expert classique modélise chacune des conditions de la prémisse par une variable binaire, qui reflète l'appartenance totale (ou nulle) des variables d'entrée à un domaine précis. Les modèles flous, en revanche, autorisent l'appartenance partielle des variables des prémisses aux différents ensembles correspondant aux termes linguistiques employés. Cette différence est illustrée dans la figure 2.16. Dans le cas d'un ensemble *crisp*, l'appartenance d'une variable à cet ensemble, notée $\mu(x)$, est totale ($\mu(x) = 1$) ou nulle ($\mu(x) = 0$). Dans le cas d'un ensemble flou, ce degré d'appartenance est nul pour des valeurs de x inférieures à x_a et supérieure à x_d, total sur l'intervalle $[x_b, x_c]$ mais graduel sur les intervalles $[x_a, x_b]$ et $[x_c, x_d]$, où $\mu_A(x)$ est compris entre 0 et 1.

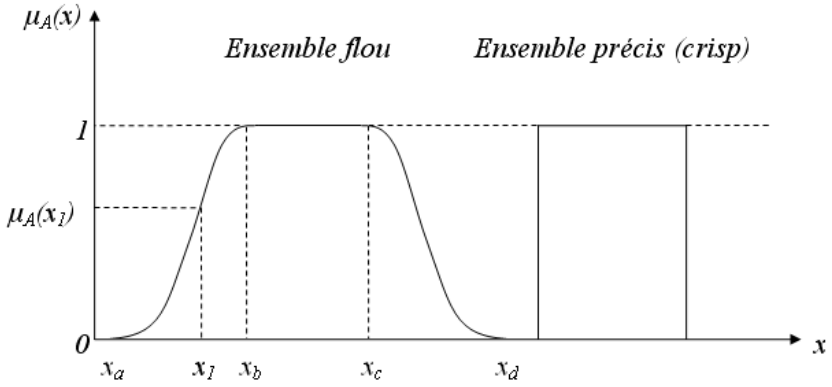

FIG. 2.16 – Comparaison entre l'appartenance d'une variable à un ensemble *crisp*(à droite) et un ensemble flou (à gauche). Dans ce dernier cas, l'appartenance est partielle pour une valeur de x appartenant aux intervalles $[x_a, x_b]$ et $[x_c, x_d]$.

Nous adoptons à présent la notation formelle d'un système représenté par un ensemble de règles floues. Dans cette notation, un modèle tel que celui décrit dans l'exemple précédent est exprimé sous la forme :

$$R_i : \textbf{Si } x_1 \text{ est } A_{i1} \textbf{ et } \ldots \textbf{ et } x_p \text{ est } A_{ip} \textbf{ alors } y_i \text{ est } B_i, \quad \text{i=1,...,r}$$

où r désigne le nombre de règles utilisées par le modèle. Les éléments x_j du vecteur $X \in \Re^p$ représentent les variables d'entrée du système dont on cherche à modéliser le comportement. Les termes linguistiques A_{ij} et B_i correspondent à des ensembles flous décrits par des fonctions d'appartenance, notées μ, permettant de caractériser l'appartenance des variables d'entrée x_j à

ces ensembles de manière graduelle. Nous considérons ici un espace de sortie dont la dimension est égale à un, comme c'est le cas dans un contexte de classification supervisée où l'on cherche à prédire l'étiquette des exemples en entrées.

Les fonctions d'appartenance $\mu_{A_{ij}}$ et μ_{B_i} établissent une correspondance entre les termes linguistiques utilisés dans les règles floues et l'intervalle $[0, 1]$:

$$\mu_{A_{ij}}(x_j) : \Re \rightarrow [0, 1], \mu_{B_i}(y) : \Re \rightarrow [0, 1] \tag{2.73}$$

Plusieurs formes de fonctions d'appartenance sont utilisables en pratique. La figure 2.17 représente trois des principales formes de fonctions d'appartenance utilisées dans la définition d'un modèle flou : gaussienne, trapézoïdale et triangulaire.

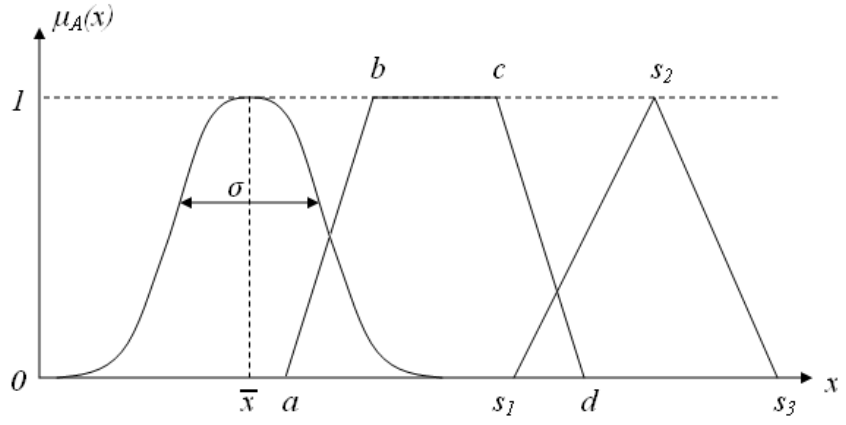

FIG. 2.17 – . Principales fonctions d'appartenance utilisée pour caractériser les ensembles flous d'un modèle : gaussienne, trapézoïdale et triangulaire, respectivement de gauche à droite.

Ces fonctions d'appartenance peuvent être définies par le concepteur du modèle, au même titre que la structure d'un réseau bayésien par exemple, ou apprises à partir des données entrées-sorties du problème. Le nombre et la forme des fonctions d'appartenance influencent la précision du modèle obtenu.

La mise en œuvre de l'inférence dans un modèle flou dépend du type de modèle utilisé. On distingue communément trois types de modèles flous en fonction de la forme de la conséquence des règles utilisées :

- *Le modèle flou Mamdani* [135, 136], dans lequel la variable de la conséquence est décrite par son degré d'appartenance à l'ensemble flou B_i.
- *Le modèle flou relationnel* [162], qui peut être vu comme un modèle Mamdani plus général, dans lequel il est possible d'associer une prémisse spécifique avec plusieurs conséquences différents via une relation floue.
- *Le modèle flou Takagi-Sugeno* [210], dans lequel la variable de la conséquence est numérique. Cette variable est exprimée sous la forme d'une constante, d'un polynome ou plus généralement par une fonction ou une équation différencielle des variables associées à la prémisse.

L'exemple proposé au début de cette section correspond à une modèle flou Mamdani, car la variable de la conséquence *risque d'infarctus* est caractérisée par son degré d'appartenance aux ensembles flous *faible* et *important*. A l'inverse, des règles de la forme :

R_1 : **Si** cholesterol est élevé **et** activité sportive est faible **alors** risque d'infarctus = 50%

R_2 : **Si** cholesterol est faible **et** activité sportive est régulière **alors** risque d'infarctus = 10%

correspondent à celles utilisées par un modèle flou Takagi-Sugeno, où la variable de la conséquence est exprimée comme une fonction des variables en entrée du système (nous avons ici pris le cas particulier d'une constante). Dans la section suivante, nous décrivons plus en détail une implémentation inspirée du modèle flou Takagi-Sugeno proposée par Bontempi et Birattari [10] permettant de déterminer un ensemble de règles floues à partir de données entrées-sorties.

Modélisation neuro-floue pour l'analyse de données entrées-sorties

De manière à appliquer la théorie des ensembles flous dans un contexte d'apprentissage automatique et de classification supervisée, où les données en entrée du modèle correspondent aux vecteurs des attributs caractérisant un objet, et où la sortie du modèle désigne l'étiquette de cet objet, nous utilisons la formulation proposée par Bontempi et Birattari [10], inspirée du modèle flou Takagi-Sugeno. Dans cette formulation, un modèle flou est décrit par un ensemble de règles de la forme :

$$R_i : \textbf{Si } x_1 \text{ est } A_{i1} \textbf{ et } \ldots \textbf{ et } x_p \text{ est } A_{ip} \textbf{ alors } y_i = f_i(X),\ i = 1, \ldots, r$$

Les fonctions f_i utilisées possèdent la même structure pour chacune des règles du modèle et seuls leurs paramètres varient. Les fonctions utilisées dans la formulation proposée par Bontempi et Birattari sont des fonctions affines de la forme :

$$y_i = A_i^T \cdot X + d_i \tag{2.74}$$

où A_i^T désigne le vecteur des coefficients a_{ij} et d_i le biais de la fonction. Lorsque les sorties des différentes règles du modèle sont exprimées sous cette forme, elles définissent des hyperplans séparateurs. La prémisse de chaque règle correspond à une region floue de validité pour le sous-modèle correspondant à la conséquence. Le modele global, composé des modèles élémentaires linéaires correspondant aux prémisses des différentes règles utilisées, peut être vu comme une approximation par morceaux d'une surface non lineaire correspondant à la sortie globale Y du modèle final.

Dans le cas particulier où les coefficients a_{ij} sont tous nuls, la conséquence s'exprime par une fonction constante :

$$y_i = d_i \tag{2.75}$$

Supposons que l'on dispose d'un tel modèle, pour lequel les différentes fonctions d'appartenance ont préalablement été déterminées. Avant de pouvoir inférer la sortie globale du système, il faut d'abord calculer le degré d'accomplissement (*degree of fulfillment*), que l'on note β_i, de la prémisse de chacune des règles du système. Dans le modèle que nous présentons dans cette section, la prémisse est décomposée en un ensemble de propositions floues élémentaires définies sur les composantes x_j du vecteur d'entrée $X = (x_1, \ldots, x_p)$ auxquelles on associe des sous-ensembles flous unidimensionnels.

Le degré d'accomplissement de la prémisse est calculé en combinant les degrés d'appartenance des propositions floues élémentaires. Dans l'implémentation proposée dans le cadre de ce mémoire, seul l'opérateur logique de conjonction (*et*) est utilisé. Pour représenter cet opérateur,

nous utilisons le produit des degrés d'appartenance des entrées aux ensembles flous unidimensionnels de chaque proposition élémentaire. Le degré d'accomplissement de la prémisse est donné par :

$$\beta_i(X) = \prod_{j=1}^{p} \mu_{A_{ij}}(x_j) \tag{2.76}$$

Ce degré d'accomplissement peut être vu comme une mesure reflétant à quel point le vecteur X en entrée du système satisfait chaque règle du modèle utilisé. L'idée sous-jacente est que plus le vecteur d'entrée X satisfait la prémisse d'une règle R_i, et plus la contribution de la sortie correspondante y_i dans la sortie globale Y du système est importante.

Dans le cas où les données en entrée-sortie définissent un problème de régression, ou de classification pour lequel les étiquettes sont des variables ordinales, la sortie du modèle est obtenue par defuzzification barycentrique. Ce procédé revient à calculer la moyenne des sorties correspondant aux différentes règles, pondérées par le degré d'accomplissement normalisé λ_i de leur prémisse :

$$Y(X) = \sum_{i=1}^{r} \lambda_i y_i(X) \quad avec \quad \lambda_i = \frac{\beta_i(X)}{\sum_{i=1}^{r} \beta_i(X)} \tag{2.77}$$

Dans le cas plus général où les classes sont représentées par des étiquettes n'obéissant aucune relation d'ordre, on considère que la sortie globale du modèle correspond à la sortie y_i de la règle r_i dont le degré d'accomplissement β_i de la prémisse est maximal :

$$Y(X) = y^*(X), \quad \beta^*(X) = argmax_{i=1}^{r}\ \beta_i(X) \tag{2.78}$$

Les mécanismes d'inférence décrits dans cette section permettent de déterminer la sortie Y correspondant à un vecteur X en entrée d'un modèle flou, de la même manière qu'un réseau de neurones est capable de prédire la sortie désirée face à ce même vecteur d'entrée. Cependant, même si l'utilisateur définit la structure du réseau, ce dernier n'est utilisable qu'une fois les poids synaptiques déterminés par apprentissage, à partir d'exemples entrées-sorties. Nous présentons maintenant la démarche mise en oeuvre pour déterminer un modèle flou à partir des données, à l'image des méthodes de l'Apprentissage Automatique décrites précédemment dans ce mémoire.

L'apprentissage du modèle à partir des données

Les mécanismes de modélisation et d'inférence dans un modèle flou tels que nous les avons décrits précédemment supposent que l'utilisateur dispose d'un tel modèle, pour lequel les fonctions d'appartenance $\mu_{A_{ij}}$ et les fonctions f_i caractérisant les conséquences des différentes règles sont connues. Dans le cadre de cette thèse, nous nous intéressons aux méthodes permettant l'induction d'un tel modèle à partir de données d'apprentissage. L'identification basée sur la commande floue entrée-sortie décrite dans [10] propose un algorithme permettant l'induction d'un modèle flou à partir d'un ensemble d'apprentissage.

Dans la plupart des algorithmes manipulés par la communauté de l'apprentissage automatique, un certain nombre de paramètres sont définis manuellement par l'utilisateur, et d'autres sont déterminés au cours de l'apprentissage. Considérons le cas d'un réseau de neurone dont la structure est définie par l'utilisateur. Ce dernier spécifie le nombre et la taille de chacune des couches du réseau, ainsi que la fonction d'activation de chaque unité, qui peut être une fonction sigmoïde, linéaire ou radiale, dans le cas de réseaux à unités sphériques. Les poids synaptiques, en revanche, sont déterminés au cours de l'apprentissage, de manière adaptative, en utilisant un algorithme basé sur la rétropropagation de l'erreur, qui peut également être considéré comme

un des paramètres du réseau. L'identification d'un modèle flou fonctionne de manière similaire, et l'analogie que nous effectuons avec l'apprentissage d'un réseau de neurones n'a rien de fortuite, puisque qu'il est commun de parler d'identification neuro-floue dans le cas de modèles flous utilisés pour prédire la sortie associée au comportement d'un système [6, 9].

Dans l'implémentation inspirée du modèle Takagi-Sugeno présentée dans la section précédente, l'utilisateur définit un certain nombre de paramètres : la forme des fonctions d'appartenance (gaussienne, triangulaire ou trapèzoïdale), le type de la fonction caractérisant la conséquence de chaque règle (constante ou affine) ainsi que le nombre de règles utilisées dans le modèle. Un algorithme d'apprentissage détermine les paramètres de chacune des fonctions d'appartenance $\mu_{A_{ij}}$, comme la moyenne $\bar{x}$ et l'écart type σ dans le cas de fonctions d'appartenance de forme gaussienne, ainsi que les coefficients a_{ij} des fonctions de sorties f_i de la conséquence de chaque règle du modèle.

Dans un premier temps, une implémentation des *k-means* est utilisée pour initialiser les paramètres des fonctions d'appartenance. La prémisse et la conséquence de chaque règle doivent alors être optimisés. Leur optimisation simultanée est un problème non linéaire de complexité importante et très couteux en temps. De manière à rendre l'utilisation d'un modèle flou compétitive par rapport à l'apprentissage d'un réseau de neurone par exemple, Bontempi propose un algorithme d'apprentissage en deux phases. La première phase consiste à fixer les conséquences des différentes règles et à optimiser les prémisses. Au cours de la seconde phase, les paramètres optimaux des différentes prémisses déterminés précédemment sont conservés, et ce sont les paramètres des fonctions de sorties qui sont alors optimisés. Ces deux étapes sont répétées durant un nombre d'itérations prédéfini, jusqu'à obtention d'un état du système minimisant l'erreur, à la manière des algorithmes basés sur la rétropropagation du gradient de l'erreur.

Remarquons que dans le cadre de la classification supervisée, l'identification de l'étiquette basée sur l'utilisation d'un modèle neuro-flou utilise les variables d'entrée sans passer préalablement par une étape de *fuzzification*. Le degré d'appartenance des éléments du vecteur des attributs $X = (x_1, \ldots, x_p)$ est évalué directement pour déterminer le degré de satisfaction de chaque règle, sans traitement préalable, hormis une normalisation des données dans l'intervalle $[0, 1]$ ou $[0, 1]$ telle que celle mise en oeuvre dans le cas des réseaux de neurones.

2.4 Conclusion du chapitre

Dans ce chapitre, nous avons défini le formalisme de l'apprentissage automatique et de la classification supervisée, et présenté différents algorithmes de la litterature, plus particulièrement ceux permettant de traiter les problèmes non linéairement séparables. Cependant, il n'existe pas de *solution miracle* au problème de la classification. Comme expliqué dans ce chapitre, bien que de nombreuses méthodes de validation existent pour évaluer de manière rigoureuse un classifieur, il demeure toujours un facteur aléatoire lors de l'évaluation d'une méthode de classification, et une même expérience répétée deux fois dans des conditions similaires donnera lieu la plupart du temps à deux résultats légèrement différents. Ceci est dû à l'absence de contrôle sur cet aspect aléatoire survenant lors l'évaluation d'un classifieur. Les *degrés de liberté* inhérents à la comparaison de différentes méthodes de classification sont exposés dans [39] et peuvent être divisés en deux groupes : ceux relatifs aux données et ceux propres au choix de l'algorithme utilisé. Le choix de l'ensemble de test peut influer sur l'évaluation des performances prédictives d'un même modèle. Le partitionnement des données d'origine étant le plus souvent effectué de façon aléatoire, il est difficile de contrôler le résultat obtenu, même en répétant plusieurs fois le partitionnement. De manière analogue, différents ensembles d'apprentissage peuvent donner lieu

à des procédures de classification différentes, en particulier dans le cas de classifieurs instables, tels que les arbres de décisions ou les réseaux de neurones. Une classe insuffisamment représentée dans l'ensemble d'entraînement se traduit en général par une fonction discriminante associée très imprécise. La présence d'exemples mal classés dans les données d'apprentissage et de test peut également être considérée comme un facteur de variabilité, au même titre que la présence de bruit dans les données de manière générale : certains algorithmes d'apprentissage y sont sensibles. La seconde cause de variations de l'estimation de l'erreur concerne le nombre de degrés de liberté de l'algorithme utilisé. Certains modèles, comme les k plus proches voisins, ne dépendent que d'un nombre limité de paramètres. A l'inverse, les réseaux de neurones présentent un grand nombre de degrés de liberté, comme le choix de la structure ou de l'algorithme d'apprentissage. Certains algorithmes présentent même des paramètres aléatoires. A titre d'exemple, nous pouvons citer l'initialisation des poids dans le cas des réseaux de neurones, où tout algorithme basé sur des heuristiques exploratoires, tels les algorithmes génétiques. Il n'existe pas de configuration optimale et universelle des paramètres mis en jeu. Ces derniers dépendent du problème posé et doivent le plus souvent être estimés par les méthodes de validation exposées au début de ce chapitre.

Il n'existe pas de consensus pour regrouper les différents algorithmes de classification de manière formelle. Lippman [126] et Holmström [93] proposent différents regroupement des algorithmes d'apprentissage usuels, en distingant les méthodes locales et globales, paramétriques et non-paramétriques, ou encore probabilistes et structurelles. Kuncheva [113] propose une taxonomie arborescente des principaux algorithmes de classification présents dans la littérature, en soulignant toutefois qu'un même classifieur peut être positionné de manière arbitraire dans différentes catégories. C'est le cas par exemple des k plus proches voisins, habituellement considérés comme une approche non-paramétrique bien qu'ils posent une hypothèse sur des densités de probabilités locales. De manière similaire, les réseaux de neurones peuvent indépendamment être considérés comme des approximateurs de fonctions, ou comme des méthodes modélisant la structure des données, au même titre que les arbres de décisions. Xu *et al.* [227] proposent une topologie des classifieurs basée sur la nature de leur sortie. Ils sont regroupés en trois types, selon que la sortie produite correspond respectivement à une valeur discrète, à une liste d'étiquettes classées par ordre de préférence ou à une mesure de confiance pour chacune des classes possibles. Cette topologie est souvent reprise dans les ouvrages traitant de la fusion d'information provenant de différentes sources, comme nous le verrons au chapitre suivant.

Il n'est pas possible de dire qu'un algorithme de classification est meilleur qu'un autre. Le choix d'un classifieur dépend du problème posé. La question qui se pose actuellement en Apprentissage Automatique n'est pas *quel classifieur utiliser ?* mais plutôt *comment combiner les décisions provenant de différents algorithmes ?*. C'est à cette question que répondent les ensembles de classifieurs, que nous présentons dans le chapitre suivant, et qui s'appuient sur le concept de la diversité.

3

Ensembles de classifieurs

Quand des décisions importantes doivent être prises, la société préfère souvent avoir recours à un comité de décideurs plutôt que de confier cette décision à un unique expert. Les comités de décision de grandes entreprises, les commissions d'enquête, les assemblées pénales ou encore les jurys universitaires décidant de l'admission d'étudiants en année supérieure sont autant de cas de figure où la décision est prise de manière consensuelle entre plusieurs membres. Ce choix de placer la confiance dans un groupe plutôt que dans un décideur unique se justifie de deux manières différentes. Une mauvaise décision individuelle se traduit en général par de lourdes conséquences, on cherche donc à obtenir la décision la plus objective possible en ayant recours à des groupes de décideurs pour avoir plusieurs points de vue. La seconde raison pour laquelle on a également recours à un comité de décideurs est que les différents intervenants peuvent n'avoir qu'une vision partielle du problème posé. C'est le cas par exemple d'un jury universitaire, où les différents membres présents ne sont en général chargés que d'une ou deux disciplines d'enseignement. Le passage en année supérieure est alors décidé à la majorité parmi les différents intervenants, de sorte qu'une éventuelle lacune de l'élève dans une des disciplines soit compensée par ses résultats dans les autres. Depuis plusieurs années, l'Apprentissage Automatique tend à s'inspirer de cette vision pour rechercher des algorithmes permettant de construire des ensembles de classifieurs (*Ensemble Learning*, *Commitee Learning*) [113]. L'objectif est d'obtenir un nouveau classifieur, constitué d'un ensemble de prédicteurs de base, de manière à diminuer le nombre d'exemples mal classés tout en conservant un temps de calcul raisonnable. Nous nous intéressons plus particulièrement dans ce mémoire au cas où des classifieurs différents sont entrainés à résoudre le même problème, à l'image de l'audience où l'on cherche à maximiser les chances de prendre la bonne décision en diversifiant les points de vue. Ce chapitre présente un état de l'art des travaux sur les ensembles de classifieurs. Nous faisons la distinction entre cette approche et celle consistant à décomposer un problème complexe, comportant généralement plus de deux classes, de manière à faciliter sa résolution. Cependant, nous commençons par présenter les différents schémas de décomposition d'un problème complexe en sous-problème élémentaires résolus par des classifieurs indépendants [2, 23], ainsi que les principales méthodes de fusion décrites dans la littérature, car cette approche constitue un thème de l'Apprentissage Automatique à part entière. Nous abordons ensuite le problème de la construction d'ensembles de classifieurs différents entrainés à résoudre un même problème en nous appuyant sur la comparaison entre les approches basées sur le *Bagging* et le *Boosting*. Cette comparaison nous permet d'introduire la notion de diversité dans les ensembles de classifieurs ainsi que son influence sur la qualité des prédictions réalisées par l'ensemble de classifieurs. Nous terminons ce chapitre par la présentation des principaux algorithmes existant dans la littérature et permettant une utilisation active de la diversité au

cours du processus de création de l'ensemble.

3.1 Decomposition de problèmes et fusion de décisions

En Apprentissage Automatique, on distingue parfois les problèmes de classification binaires des problèmes comportant plusieurs classes. En effet, certains algorithmes d'apprentissage ne sont applicables que pour des problèmes binaires. C'est le cas des Machines à Vecteur de Support par exemple, décrites dans le précédent chapitre. L'expression des contraintes dans la formulation du problème d'optimisation ne rend son application possible que pour des problèmes comportant deux classes, négatives et positives, notées respectivement -1 et $+1$. L'implémentation originale de l'algorithme Adaboost, présenté plus loin dans ce chapitre, est également destinée à la résolution de problèmes de classification binaire, bien que des implémentations permettant de traiter les problèmes comportant plus de deux classes ont été proposés [2].

De manière générale, il est intéressant d'avoir recours à des décompositions de manière à diminuer la compléxité du problème initial. Nous présentons maintenant les trois principales méthodes permettant de décomposer un problème de classification comportant plus de deux classes en un ensemble de sous problèmes binaires : la décomposition *un contre un* (*One versus One*), la décomposition *un contre tous* (*One versus All, One versus Rest*) et l'utilisation de codes correcteurs d'erreur.

3.1.1 Schémas de décomposition usuels

Les deux principales méthodes de décomposition d'un problème multi-classes consistent soit à opposer une classe particulière au reste des autres classes soit à réaliser toutes les dichotomies envisageables en considérant chaque paire de classes possible pour le problème posé. La première méthode est appelée décomposition *un contre tous* (*One versus All, One versus Rest*). Ce schéma de décomposition considère un nombre de classifieurs égal au nombre de classes $|\Omega|$ du problème considéré (un classifieur par classe). Chaque classifieur C_i est entrainé à séparer les exemples de la classe ω_i, dite positive, de ceux faisant partie de l'ensemble des autres classes $\omega_j, j \neq i$ et formant la classe négative. La figure 3.1 illustre le schéma de décomposition *un contre tous* dans le cas d'un problème à trois classes.

Dans le cas d'une décomposition *un contre un* (*One versus One*), $C^2_{|\Omega|}$ dichotomies sont formées, de manière à obtenir un ensemble de classifieurs binaires entrainés à distinguer deux classes ω_i et ω_j, les autres classes $\omega_k,\ k \neq i, j$ étant ignorées pour le sous-problème considéré [84, 110]. Une illustration de ce type de décomposition est donnée dans la figure 3.2, en considérant le même exemple que celui donné dans le cas du schéma *un contre tous*.

Dans [2, 96, 144], les auteurs discutent des avantages et des inconvénients de chacune de ces décompositions. Les résultats décrits dans [2] donnent un avantage à la décomposition *un contre un* en terme de précision alors que les travaux décrits dans [177] montrent que les deux schémas de décomposition ont une précision équivalente lorsque les SVM binaires employés sont correctement paramétrés. En termes d'erreur de classification, ces deux décompositions semblent donc équivalentes, mais il apparait que le modèle *un contre un* est plus avantageux en terme de temps de traitement bien que le nombre de classifieurs binaires soit plus important dans ce cas [96, 144]. En effet, dans le cas d'une décomposition *un contre tous*, les exemples appartenant aux classes $\omega_j, j \neq i$ sont regroupés au sein d'une même *super classe* négative et chaque classifieur C_i est entrainé sur la totalité des exemples d'apprentissage. Avec le schéma *un contre un*, chacune des dichotomies ne fait intervenir que deux classes, le nombre d'exemples d'apprentissage utilisés pour chacun des sous-problèmes binaires est donc réduit de manière significative. Bien que le

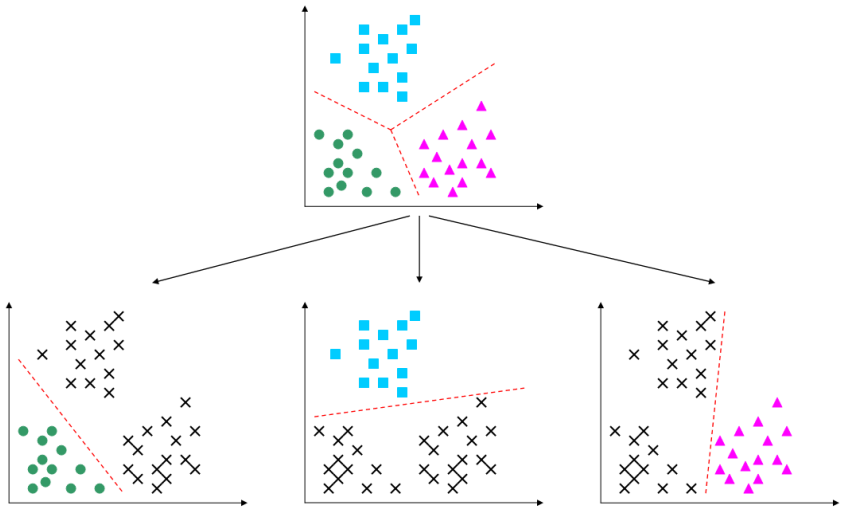

FIG. 3.1 – Décomposition d'un problème de classification comportant trois classes suivant un schéma de décomposition *un contre tous*

nombre de dichotomies réalisées soit plus important que dans le cas d'une décomposition *un contre un* ($|\Omega|$ classifieurs binaires contre $C_{|\Omega|}^{|2|}$ dans le cas d'une décomposition *un contre tous*), le temps requis pour résoudre le problème global est en général diminué. Cette propriété est souvent vérifiée lorsque les classifieurs binaires utilisés sont des SVM, dont le coût d'apprentissage dépend en grande parties du nombre de contraintes, c'est à dire du nombre d'exemples considérés.

Remarquons que dans l'exemple des figures 3.2 et 3.1, les sous-problèmes binaires considérés peuvent être résolus à l'aide d'un hyperplan séparateur. Dans le cas général, les frontières de décision obtenues par une décomposition *un contre tous* sont plus complexes que dans le cas du schéma *un contre un*.

Le principal inconvénient que présente une décomposition *un contre un* est que chacune des dichotomies réalisées considère qu'une partie du problème initial, les exemples qui n'appartiennent pas aux deux classes considérées sont ignorés. Ce phénomène apparait également dans le cas de la décomposition *un contre tous*, mais dans une moindre mesure. Chaque classifieur binaire C_i est en effet entrainé sur la totalité des exemples disponibles. Cependant, lorsqu'un échantillon appartient à la classe négative, le classifieur est incapable de déterminer de quelle classe $\omega_j, j \neq i$ cet échantillon est le plus proche.

Une méthode de décomposition plus générale d'un problème multi-classes est proposée par Dietterich et Bakiri [43, 2] et repose sur l'utilisation de codes correcteurs d'erreur (*Error-Correcting Output Codes* ou *ECOC*). Cette approche considère que chaque classe du problème initial peut être représentée par une chaine binaire de taille fixe L appelée signature et notée $mathcalS_i$, $i = 1, \ldots, |\Omega|$. La résolution du problème initial se fait à l'aide d'une matrice de codage $\mathcal{M} = (b_{ij})$, avec $i = 1, \ldots, |\Omega|$ et $j = 1, \ldots, L$. Les lignes de cette matrice correspondent à l'expression des différentes classes suivant le codage binaire choisi. La table 3.1 décrit l'exemple présenté dans [43] utilisé pour décomposer un problème de reconnaissance de chiffres manuscrits à l'aide de six classifieurs binaires. La signification de chacun des bits utilisé dans le codage est donné dans la table 3.2.

Remarquons que dans cet exemple, chaque classe possède une signature $\mathcal{S}_i = \{b_{i1}, \ldots, b_{iL}\}$

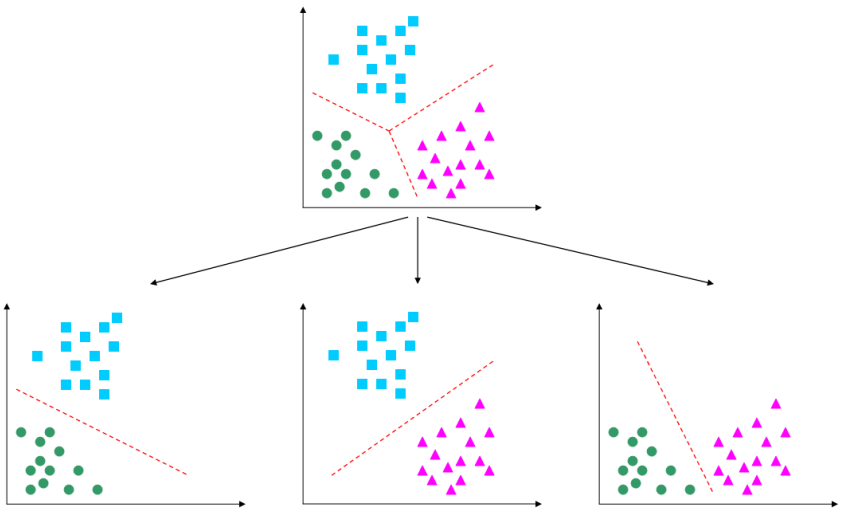

FIG. 3.2 – Application du schéma de décomposition *un contre un* sur un problème de classification à trois classes

unique, $i = 1, \ldots, |\Omega|$, représentée par le vecteur binaire de taille L correspondant à la ligne i de la matrice de codage. Les étiquettes discrètes correspondant aux différentes classes du problème sont remplacées par les signatures associées suivant le code utilisé. Les L classifieurs binaires (un pour chaque bit du code) sont construits à partir des exemples d'apprentissage. Pour évaluer un nouvel exemple X, sa signature $\mathcal{S}_X$ est calculée à partir des sorties des classifieurs binaires $C_1, \ldots, C_L$. Le nouvel exemple est affecté à la classe dont la signature est la plus proche selon la distance de Hamming, qui correspond au nombre de bits différents entre deux signatures.

Le principe d'utilisation de codes correcteurs d'erreurs repose sur la tolérance que présente la matrice de codage face aux erreurs individuelles des classifieurs binaires. L'idée principale est que si le code est bien choisi, certains bits érronnés peuvent être corrigés par le code de manière à ce que la signature la plus proche soit toujours celle de la classe correcte. La qualité d'un code correcteur d'erreur peut être mesurée par la distance de Hamming minimale entre deux classes du problème considéré. Si l'on note d^*_H cette distance, alors le code choisi pourra corriger au minimum $\left\lfloor \frac{d^*_H - 1}{2} \right\rfloor$ erreurs individuelles des classifieurs binaires. Dans l'exemple donné dans la table 3.1, les signatures des chiffres 0 et 9 ne différent que de 1 bit, ce qui limite le nombre d'erreurs pouvant être corrigées par ce code. Remarquons que pour cet exemple, la définition de la matrice de codage repose sur la signification bien particulière donnée à chaque bit. De manière générale, le code correcteur est construit de manière à maximiser la distance de Hamming minimale entre les lignes et les colonnes de la matrice de codage. Une distance de Hamming élevée entre les lignes de la matrice se traduit par une meilleure distinction des signatures de chaque classe et permet d'augmenter le nombre d'erreurs que le code peut corriger. De la même manière, une distance élevée entre les colonnes de la matrice $\mathcal{M}$ renforce l'indépendance entre les classifieurs binaires utilisés pour décomposer le problème initial.

La décomposition par codes correcteurs d'erreur initialement proposée par Dietterich et Bakiri [43] et décrite jusqu'à présent définit une matrice de codage $\mathcal{M}$ dont les éléments b_{ij} ont valeur dans $\{0, 1\}$. Allwein *et al.* [2] définissent une approche plus générale reposant sur l'utilisation de code correcteurs d'erreur. Cette fois, les éléments b_{ij} prennent leurs valeurs dans

TAB. 3.1 – Matrice de codage utilisée pour la reconnaissance de chiffres manuscrits.

Classe	Code correspondant					
	lv	lh	ld	cf	og	od
0	0	0	0	1	0	0
1	1	0	0	0	0	0
2	0	1	1	0	1	0
3	0	0	0	0	1	0
4	1	1	1	0	0	0
5	1	1	0	0	1	0
6	0	0	0	1	0	1
7	0	1	1	0	0	0
8	0	0	1	1	0	0
9	0	0	0	1	1	0

TAB. 3.2 – Signification de différents bits utilisés dans le code de la table 3.1.

Abbréviation	Signification
lv	Présence de lignes verticales
lh	Présence de lignes horizontales
ld	Présence de lignes diagonales
cf	Présence de courbes fermées
og	Présence de courbes ouvertes sur la gauche
od	Présence de courbes ouvertes sur la droite

$\{1, 0, -1\}$. Certains termes de la matrice $\mathcal{M}$ peuvent être nuls, ce qui signifie qu'il est possible d'ignorer certains bits du code pour caractériser la signature d'une classe.

Remarquons que selon cette approche plus générale, les schémas de décomposition *un contre tous* et *un contre un* peuvent être considérés comme des cas particuliers de l'utilisation de code correcteurs d'erreurs. Par exemple, les matrices de codage associées pour un problème à quatre classes sont données respectivement dans les tables 3.3 et 3.4. On peut remarquer que la distance de Hamming minimale dans le cas d'une décomposition *un contre un* est plus importante, ce qui traduit une meilleure séparation des classes et une plus grande indépendance des classifieurs binaires utilisés. Une analyse détaillée du calcul de distances de Hamming est décrite dans la section suivante, dédiée à la fusion de classifieurs binaires.

TAB. 3.3 – Matrice de codage correspondant à une décomposition de type *un contre tous* pour un problème à quatre classes.

	C_1	C_2	C_3	C_4
ω_1	+1	-1	-1	-1
ω_2	-1	+1	-1	-1
ω_3	-1	-1	+1	-1
ω_4	-1	-1	-1	+1

Le nombre d'exemples mal classés dépend de la qualité du code choisi. Plusieurs solutions

TAB. 3.4 – Matrice de codage correspondant à une décomposition de type *un contre un* pour un problème à quatre classes.

	C_{12}	C_{13}	C_{14}	C_{23}	C_{24}	C_{34}
ω_1	+1	+1	+1	0	0	0
ω_2	-1	0	0	+1	+1	0
ω_3	0	-1	0	-1	0	+1
ω_4	0	0	-1	0	-1	-1

ont été proposées pour la génération de codes correcteurs d'erreur [5, 32, 167]. Leur utilisation pour la classification de textes est décrite dans [74]. Les auteurs y soulignent que l'augmentation de la longueur du code permet de réduire de manière significative le nombre d'exemples mal classés. Cependant, un code de grande taille augmente également la compléxité du problème du fait du nombre plus important de classifieurs à construire. Les résultats sont obtenus en utilisant des classifieurs bayésiens naïfs bien que ce type d'algorithme puisse traiter directement des problèmes multi-classes. Une décomposition par codes correcteurs peut donc être utilisée dans l'unique objectif de tirer partie des propriétés de correction de la matrice de codage $\mathcal{M}$ afin de réduire le nombre d'exemples mal classés.

L'étude de nouvelles méthodes permettant de décomposer efficacement un problème multi-classes est encore un sujet de recherche actif. Platt *et al.* [164] proposent une alternative aux trois principales méthodes décrites précédemment en utilisant un graphe de décision (*Decision Directed Acyclic Graph* ou *DDAG*) pour décomposer un problème en sous-problèmes binaires. Comme dans le schéma *un contre un*, $C^2_{|\Omega|}$ classifieurs binaires sont construits de manière à réaliser toutes les dichotomies possibles entre les classes. Ces classifieurs sont ensuite combinés à un graphe triangulaire, où chaque nœud interne est associé à un classifieur binaire, et où les feuilles correspondent aux classes du problème initial. L'illustration du graphe de décision obtenu pour un problème à quatre classes est décrit dans la figure 3.3.

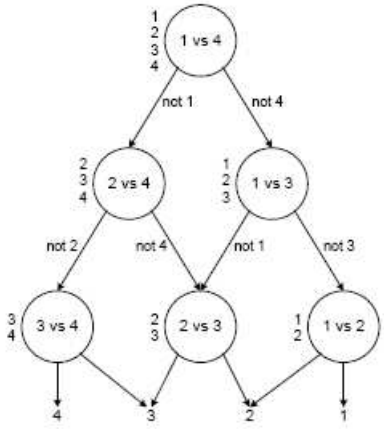

FIG. 3.3 – Graphe de décision obtenu par l'algorithme DDAGSVM [164] dans le cas d'un problème à quatre classes. La liste des étiquettes potentielles pouvant être attribuées à chaque exemple est écrite sur la gauche des nœuds.

Les auteurs soulignent que l'évaluation d'un nouvel exemple à l'aide de cette méthode est plus rapide que dans le cas d'un décomposition *un contre un* car seuls les classifieurs situés sur le chemin reliant la racine du graphe à une feuille contribuent à la classification d'un exemple. L'algorithme proposé par les auteurs dans le cas de SVM binaires, appelé DDAGSVM, permet également de borner l'erreur en généralisation.

Une autre approche descendante est proposée dans [127, 123]. Contrairement aux autres implémentations multi-classes des SVM, cette approche repose sur une décomposition hiérarchique et récursive du problème initial jusqu'à l'obtention de classifieurs binaires. Le schéma de décomposition présenté dans [123], appelé décomposition *Half versus Half*, s'appuie également sur une représentation arborescente du problème mais seules les feuilles de l'arbre correspondent à des sous-problèmes binaires opposant deux classes ω_i et ω_j du problème initial. Les noeuds internes du graphe représentent des sous-problèmes binaires opposant deux *super classes* résultant de l'agrégation des classes initiales. La figure 3.4 présente deux décompositions hiérarchiques d'un problème comportant six classes. Dans le premier cas (à gauche) la décomposition est réalisée de manière intuitive. Dans le deuxième cas, la décomposition est basée sur un regroupement hiérarchique de la distance séparant les classes dans l'espace des attributs.

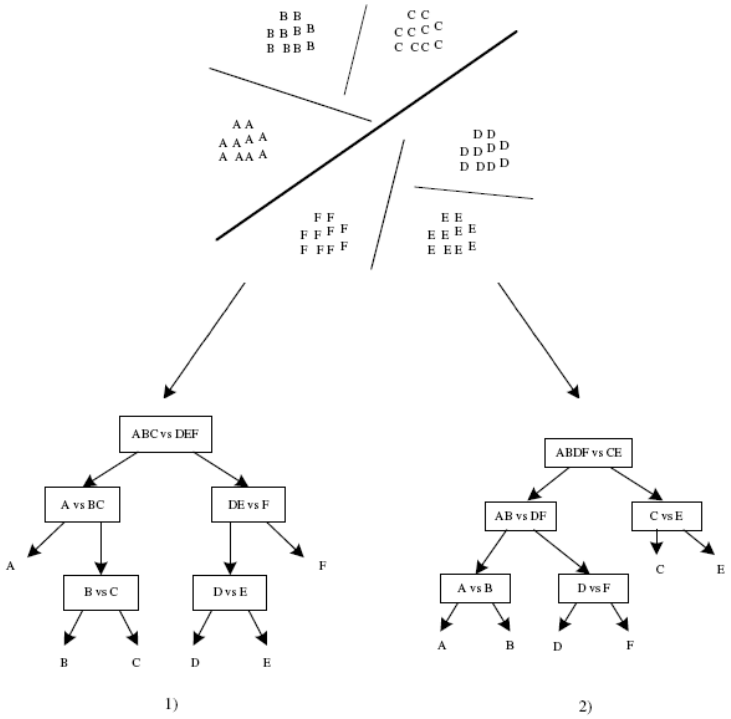

FIG. 3.4 – Décomposition d'un problème multi-classes selon l'approche Half vs Half : 1) cas d'une hiérarchisation intuitive 2) cas d'une décomposition par regroupement hiérarchique des distances séparant les classes dans l'espace des attributs

Les résultats obtenus montrent que ce type de décomposition présente une précision comparable à celle des méthodes usuelles *un contre un* et *un contre tous*. Cependant ce type de décomposition est obtenu dans des temps plus courts et nécessite moins d'espace mémoire pour sa mise en œuvre. La question concernant le choix d'une hiérarchisation des classes optimale est cependant laissée sans réponse par les auteurs.

La plupart des travaux relatifs à la décomposition de problèmes multi-classes que nous citons dans ce mémoire sont dédiés aux SVM. Le formalisme des SVM limite leur application à des problèmes binaires comme nous l'avons déjà énoncé. Ce type de classifieur rencontre un franc succès, il est donc naturel de chercher à étendre ses propriétés de généralisation face aux problèmes de classification concrets et comportant généralement plus de deux classes. L'utilisation des schémas de décomposition usuels n'est cependant pas limitée au cas unique des SVM. La décomposition en sous-problèmes binaires peut également s'appliquer à la régression logistique comme le proposent Hosmer et Lemeshow [95]. Une adaptation des codes correcteurs pour l'algorithme Adaboost présenté ultérieurement dans ce chapitre est également proposée dans [209].

3.1.2 Fusion de décisions

Les méthodes décrites dans la section précédente permettent de décomposer un problème complexe en un ensemble de sous-problèmes binaires. L'attribution d'une étiquette à un nouvel exemple est obtenue par agrégation des sorties de chaque classifieur binaire au moyen d'un règle décisionnelle. Nous présentons dans cette section les méthodes de fusion usuelles pour chacun des différents schémas de décomposition décrits précédemment.

La méthode la plus couramment utilisée dans le cas d'une décomposition *un contre un* est la règle du vote majoritaire [110]. Chaque classifieur C_{ij} entrainé à distinguer les classes ω_i et ω_j produit un vote V_{ij} défini par :

$$V_{ij}(X) = \begin{cases} 1 & si \quad C_{ij}(X) = \omega_i \\ 0 & si \quad C_{ij}(X) = \omega_j \end{cases} \tag{3.1}$$

La classe qui sera attribuée à un nouvel exemple est celle qui reçoit le plus grand nombre de votes. Cette classe, notéeω^*, est définie par :

$$\omega^* = max_{i=1}^{|\Omega|} \sum_{j \neq i} V_{ij}(X) \tag{3.2}$$

Le vote majoritaire peut également être appliqué dans le cas de décompositions *un contre tous* mais le risque de voir deux classes différentes recevoir un même nombre de votes est plus important que dans le cas *un contre un*. Dans le cas de SVM binaires utilisés dans une décomposition *un contre tous*, il est possible de considérer la valeur de la fonction de sortie $f(X)$ associée à un exemple X plutôt que son signe :

$$f(X) = \sum_{i=1}^{N} \alpha_i^* Y_i X_i \cdot X + b \tag{3.3}$$

Un SVM binaire C_i produit une sortie f_i positive lorsque $C_i(X) = \omega_i$ et négative lorsque $C_i(X) = \omega_j,\ j \neq i$. La valeur de sortie $f_i(X)$ peut être considérée comme une mesure de confiance que X appartienne à la classe ω_i. Un vecteur X est attribué à la classe pour laquelle cette valeur de confiance est maximale :

$$\omega^* = max_{i=1}^{|\Omega|} f_i(X) \tag{3.4}$$

Une limitation majeure des méthodes basées sur le vote majoritaire est liée à l'hypothèse selon laquelle les différents classifieurs possèdent une fiabilité similaire. Afin de prendre en compte la précision individuelle de chaque prédicteur dans la décision finale, il est possible de leur associer un poids w_i. Plus un prédicteur est précis, plus son poids est élevé. Ces poids sont choisis de manière à ce que $\sum_{i=1} w_i = 1$. Pour un exemple donné X, la fonction discriminante associée à

chaque classe $g_j(X),\ j = 1, \ldots, c$ est égale à la somme des poids des prédicteurs ayant attribué cette classe à l'exemple X :

$$g_j(X) = \sum_{i,\ C_i(X)=\omega_j} w_i \tag{3.5}$$

La règle de vote utilisée, appelée vote majoritaire pondéré, s'exprime alors :

$$\omega^* = max_{j=1}^{c} g_j(X) \tag{3.6}$$

Remarquons que le poids attribué à chaque prédicteur est fixé quel que soit le vecteur à classer, ce qui laisse supposer que les classifieurs ne sont pas spécialisés sur certains types d'exemples. Partant de ce constat, Liu et Zheng [132] proposent une implémentation multi-classes des SVM basée sur le schéma *un contre tous* mais tenant compte d'une mesure de confiance calculée pour chacun des classifieurs binaires obtenus. Deux types de mesures de fiabilité sont proposés en reformulant le problème d'optimisation à résoudre pour chaque SVM binaire. La comparaison entre une mesure de fiabilité statique et une mesure dynamique définie sur un voisinage de l'élément à classer donne un très net avantage à la prise en compte de la spécialisation des classifieurs dans l'espace des attributs pour effectuer là fusion de prédicteurs.

La méthode de fusion proposée par Dietterich et Bakiri [43] dans le cadre des codes correcteurs d'erreurs est basée sur le calcul de la distance de Hamming. Pour une matrice de codage $\mathcal{M} = (b_{ij}) \in \{-1, 1\}$ avec $i = 1, \ldots, c$ et $j = 1, \ldots, L$, la distance de Hamming entre la signature $\mathcal{S}_i$ de la classe ω_i et la signature $\mathcal{S}_X = \{C_1(X), \ldots, C_L(X)\}$ de l'exemple X est donnée par :

$$d_H(\mathcal{S}_i, \mathcal{S}_X) = \sum_{j=1}^{L} 1 - (b_{ij} C_j(X)) \tag{3.7}$$

où $C_j(X)$ désigne la sortie du classifieur associé au *j-ème* bit du code pour l'exemple X. La valeur de d_H traduit le nombre de bits qui diffèrent entre la signature de X et la signature de la classe ω_i. La classe qui sera affectée à l'exemple X est celle dont la signature est la plus proche selon la distance de Hamming :

$$\omega^* = min_{i=1}^{c} d_H(\mathcal{S}_i, \mathcal{S}_X) \tag{3.8}$$

Dans la formulation plus générale proposée par Allwein et al. [2], où $\mathcal{M} = (b_{ij}) \in \{-1, 0, 1\}$, le calcul de la distance de Hamming est donné par :

$$d_H(\mathcal{S}_i, \mathcal{S}_X) = \sum_{j=1}^{L} \frac{1 - signe(b_{ij} C_j(X))}{2} \tag{3.9}$$

De ce fait, si un bit b_{ij} a pour valeur 0, il ne contribue au calcul de la distance que de $\frac{1}{2}$. Le calcul de la distance de Hamming tel qu'il est proposé ne prend pas en compte la valeur de la fonction de sortie des classifieurs associés aux différents bits (par exemple la valeur de la fonction donnée dans l'équation 2.48 dans le cas de SVM binaires), qui peut être vue comme une mesure de confiance. Une formulation de la distance de Hamming permettant de répondre à cette limitation, appelée *loss based decoding*, est proposée dans [2]. Elle s'appuie sur l'utilisation d'une fonction l permettant de pénaliser l'attribution de la classe i à un exemple X :

$$d_l(\mathcal{S}_i, \mathcal{S}_X) = \sum_{j=1}^{L} l(b_{ij}, C_j(X) \tag{3.10}$$

L'attribution d'une classe à un exemple X s'effectue d'une manière similaire à celle utilisée dans le cas de la distance de Hamming, en sélectionnant la classe dont la valeur de la fonction de pénalisation est minimale :

$$\omega^* = min_{i=1}^{c} d_l(\mathcal{S}_i, \mathcal{S}_X) \tag{3.11}$$

Les auteurs proposent également une unification des codes correcteurs d'erreur et de la théorie des marges, et soulignent l'importance du code choisi concernant la réduction de l'erreur pour les SVM et l'algorithme Adaboost, décrit plus en détail dans ce même chapitre.

Remarquons que la décomposition *Half versus Half* ainsi que l'utilisation d'un graphe de décision ne nécessitent pas de superposer un opérateur de fusion à l'ensemble de classifieurs binaires. L'évaluation d'un nouvel exemple peut cependant être assimilée à un processus de sélection. En effet, pour ces deux types de décomposition, l'évaluation d'un exemple X est réalisée récursivement par le sous-ensemble de classifieurs binaires situés sur le chemin reliant le nœud racine à la feuille réalisant la dernière dichotomie et permettant d'attribuer l'étiquette définitive de X. La résolution du sous-problème binaire associé à un nœud de niveau t définit la sélection du classifieur binaire intervenant au niveau $t+1$, depuis le niveau $t = 0$ correspondant à la racine jusqu'au dernier niveau du graphe où se situent les feuilles.

Il apparait que la méthode utilisée pour fusionner des classifieurs binaires dépend en grande partie du type de leur sortie. Xu *et al.* [227] regroupent les algorithmes de classification en trois types différents en fonction de la sortie qu'ils produisent :

- type I : Chaque classifieur C_i produit une étiquette discrète $\omega_i \in \Omega$. Aucune mesure de confiance n'est disponible quant à l'appartenance d'un exemple à la classe ω_i. Les arbres de décision sont un exemple de classifieur de type I.
- type II : Chaque classifieur C_i produit une liste des c classes du problème ordonnées par ordre de vraisemblance.
- type III : Chaque classifieur C_i produit une liste de valeurs de confiance $[p(\omega_i), \ldots, p(\omega_c)]$ pour chaque classe possible. Un réseau bayésien donnant les probabilités $P(\omega_j|X)$ avec $j = 1, \ldots, c$ est un classifieur de type III.

Les classifieurs de type I constituent le type le plus universel. En effet, Kuncheva [113] souligne qu'il est toujours possible de se ramener à un classifieur de ce type. Un réseau de neurones par exemple, dont les unités de sortie produisent une valeur de confiance pour chaque classe du problème, est un classifieur de type III mais il peut se ramener à un classifieur de type I en sélectionnant la classe pour laquelle l'unité de sortie fournit la valeur maximale.

Dans [171], Quost propose une formulation de la fusion de classifieurs binaires de type I et III dans le cadre de la théorie des fonctions de croyance [199, 208]. Pour chacun des schémas de décomposition précédemment évoqués (décomposition *un contre un*, *un contre tous* et Codes Correcteurs d'Erreur), des opérateurs d'agrégation sont proposés pour fusionner les décisions de classifieurs binaires.

Une méthode de fusion plus générale et applicable dans la plupart des cas consiste à utiliser un classifieur additionnel, également appelé classifieur d'ordre supérieur. Cette méthode, connue sous le nom de *Stacked Generalisation* ou plus généralement *Stacking*, est introduite par Wolpert [225] et décrite plus en détail dans [214, 215]. Un ensemble de prédicteurs $C_1, \ldots, C_L$ est construit à partir d'un ensemble d'apprentissage $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$. Les N vecteurs $\left[\widehat{Y}_{i,1}, \ldots, \widehat{Y}_{i,L}\right]$ contenant les étiquettes attribuées par les L classifieurs pour chacun des exemples d'apprentissage sont utilisés pour entraîner un classifieur jouant le rôle de superviseur, noté $C_{\mathcal{S}}$.

L'ensemble d'apprentissage de ce superviseur est donné dans la table 3.5. La fusion des différents classifieurs est réalisée par le superviseur, qui attribue la classe d'un nouvel échantillon $C_{\mathcal{S}}(X)$.

TAB. 3.5 – Ensemble d'apprentissage d'un classifieur additionnel jouant le rôle de superviseur

C_1	C_2	...	C_L	Classe
$\widehat{Y}_{1,1}$	$\widehat{Y}_{1,2}$	...	$\widehat{Y}_{1,L}$	Y_1
$\widehat{Y}_{2,1}$	$\widehat{Y}_{2,2}$	...	$\widehat{Y}_{2,L}$	Y_2
...	...	...	...	...
$\widehat{Y}_{N,1}$	$\widehat{Y}_{N,2}$	...	$\widehat{Y}_{N,L}$	Y_N

Les méthodes de fusion qui ont été présentées dans cette section sont fréquemment utilisées dans le cadre d'une approche *Diviser pour Régner*. Les techniques de vote sont utilisables dans un cadre plus général, où chaque prédicteur est directement entrainé à résoudre un problème multi-classes. Par exemple, l'agrégation d'un ensemble d'arbres de décisions, auquel nous ferons référence sous le terme *Bagging* dans la suite de ce mémoire, est obtenue par un vote majoritaire similaire à celui que nous avons décrit, mais dont la formulation diffère du fait que les arbres de décisions utilisés sont entrainés à résoudre un même problème multi-classes.

3.2 Vers une approche ensembliste du problème de la classification

Dans la section précédente, plusieurs classifieurs sont utilisés pour résoudre un même problème selon l'approche *Diviser pour Régner*, qui consiste à décomposer un problème complexe en sous-problèmes plus simples. Une telle approche est nécessaire lorsque les classifieurs utilisés, notamment les SVM, ne peuvent résoudre directement les problèmes comportants plus de deux classes. Cette approche présente l'avantage de diminuer la compléxité du problème initial afin d'en faciliter la résolution. Dans le cas des codes correcteurs d'erreurs par exemple, initialment proposés dans le cadre de la transmission d'information, la décomposition en sous-problèmes binaires permet de corriger les erreurs individuelles des classifieurs.

Dans ce mémoire, nous nous intéressons à l'approche inverse, qui consiste à produire un modèle de classification plus précis en construisant plusieurs classifieurs différents pour résoudre le problème initial. Celà se fait au détriment de la compléxité. Cette approche est appelée de manière générale *Ensemble de classifieurs* (*Classifier Ensemble, Multiple Classifier system,Ensemble learning, Committee Learning*). Bien que ce terme puisse préter à confusion par rapport à la décomposition d'un problème donnant lieu également à un ensemble de prédicteurs binaires, la principale différence est l'absence de notion de décomposition. En effet, la réponse au problème initial est obtenue non pas en le divisant, mais en le *répliquant*. Les répliques sont résolues par des classifieurs différents. Pour illustrer cette approche, considérons l'exemple d'un jury pénal comportant plusieurs membres. Chaque membre dispose de la totalité des informations disponibles du procès : antécédents familiaux, casier judiciaire, déclarations des témoins et autres pièces à conviction. Les membres du jury n'ont donc pas une vision partielle du problème posé, comme c'est le cas avec une décomposition *un contre un* par exemple, et doivent tous résoudre le même problème. La décision rendue à l'issue de l'audience est basée sur le consentement mutuel, à savoir un vote, des membres du jury. Breiman denote cette approche par l'expression *Combine and Conquer* [15], littéralement *Combiner pour Régner*. C'est sur cette approche que nous nous focalisons à présent. Il est cependant important de noter que l'utilisation d'un ensemble

de classifieurs n'améliore pas systématiquement la précision obtenue par rapport à un classifieur unique. Dans le cas où la fusion de décisions est réalisée par un vote majoritaire, l'utilisation d'un ensemble peut dégrader la précision obtenue si la probabilité d'erreur des différents classifieurs qui le constituent est trop importante [114].

3.2.1 Topologie générale

Une topologie possible de la construction d'un ensemble de classifieurs est proposée dans [113, 222] et illustrée dans la figure 3.5.

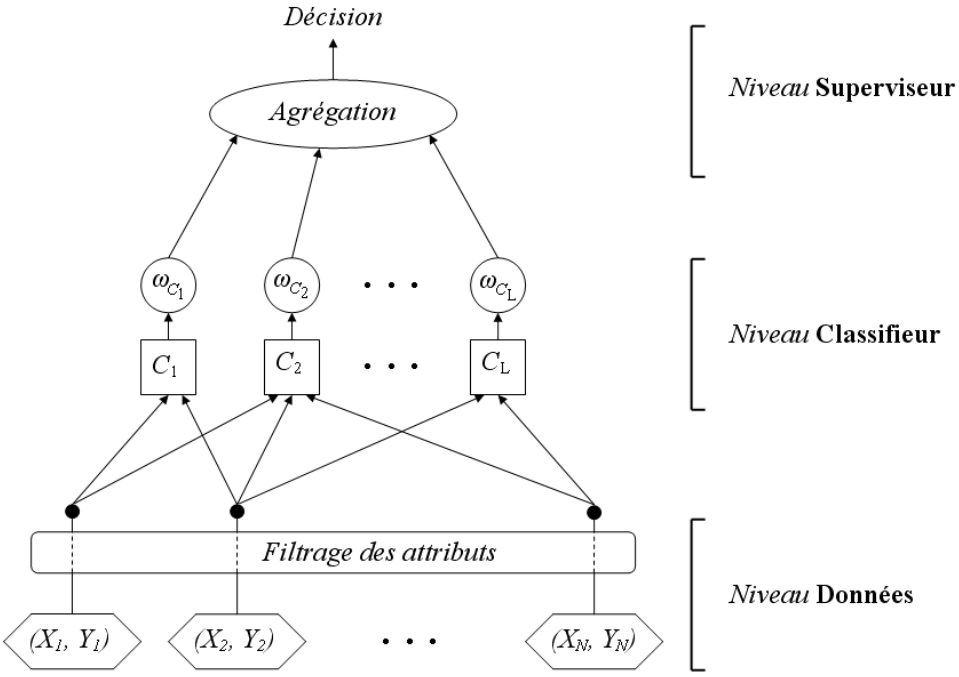

FIG. 3.5 – Schéma de construction général d'un ensemble de classifieurs.

La décision est prise par un ensemble de classifieurs, entrainés à résoudre un même problème. Il est évident que si les classifieurs sont en tout point identiques, l'utilisation de l'ensemble ne présente aucun avantage au niveau de la réduction de l'erreur. Les membres de l'ensemble doivent présenter des différences, que nous désignerons par le terme de *diversité*, et que nous détaillerons plus loin dans ce chapitre. Pour obtenir des classifieurs différents, on peut introduire volontairement des modifications aux trois niveaux de la construction. Le niveau *Données* désigne l'ensemble des traitements pouvant être appliqués à l'ensemble d'apprentissage. Ces traitements peuvent concerner la sélection d'exemples, comme le rééchantillonage ou l'application de filtres au niveau des attributs. Par exemple, l'algorithme Rotation Forest [179] partitionne l'espace des attributs en sous-ensembles sur lesquels une analyse en composantes principales est effectuée. Chaque classifieur de l'ensemble est entrainé sur les données obtenues. Le niveau *Classifieur* concerne l'algorithme d'apprentissage utilisé en lui-même. Les modifications intervenant à ce niveau peuvent concerner les paramètres de l'algorithme d'apprentissage. Li *et al.* [125] utilisent un ensemble de SVM à noyau gaussien entrainés pour différentes valeurs du paramètre du noyau σ. Enfin, le dernier niveau, que nous appelons niveau *Superviseur*, concerne la règle d'agrégation

utilisée pour prendre une décision. Kuncheva [113] considère trois grands types d'agrégation :

- ceux qui nécessitent une phase d'apprentissage
- ceux qui sont utilisables directement
- ceux qui consistent à sélectionner un classifieur ou un sous-ensemble de classifieurs utilisés pour classer un nouvel exemple

La décomposition en sous-problèmes binaires peut être vue comme un cas particulier du schéma de construction proposé dans la figure 3.5. Dans ce cas, les différences entre les classifieurs s'effectuent uniquement au niveau *Données*, et plus particulièrement sur les exemples d'apprentissage eux-mêmes.

L'utilisation de ce schéma pour résoudre les problèmes de classification permet en pratique une importante diminution de l'erreur [40, 154, 42]. Une justification intuitive de ce résultat est proposée par Dietterich [40] : supposons que l'on dispose d'un ensemble de L arbres de décision entrainés sur différents sous-ensemble d'attributs, et présentant chacun une erreur apparente (ou *resubstitution error* calculée sur l'ensemble d'apprentissage) nulle. Bien que ces classifieurs soient similaires en terme d'erreur de substitution, ils peuvent potentiellement présenter différentes facultés de généralisation. Il est alors plus sûr de prendre en considération l'ensemble des arbres de décisions disponibles et d'agréger leurs décisions plutôt que de retenir la classe attribuée par un seul de ces classifieurs. Plusieurs cadres formels basés sur la décomposition de l'erreur permettent de justifier l'utilisation d'ensembles de classifieurs. Les mécanismes mis en jeu sont en général plus complexes dans le cas de la classification que pour des problèmes de régression. La section 3.3.1 de ce mémoire sera dédiée à l'étude de ces fondements théoriques.

3.2.2 Bagging et Boosting

Les deux familles d'algorithmes ensemblistes les plus étudiées en Apprentissage Automatique sont le *Boosting* et le *Bagging*. Ces deux types d'algorithmes permettent la construction d'un ensemble de classifieurs de même type différenciés au niveau des exemples manipulés pendant l'apprentissage. Nous présentons en parallèle ces deux types d'algorithmes de manière à montrer comment la spécialisation des classifieurs de base sur certaines régions de l'espace des attributs permet, au détriment de leur erreur prédictive individuelle, d'améliorer la précision globale de l'ensemble obtenu. Dans les chapitres suivants, nous ferons référence à cette propriété sous le terme de *diversité*.

Le terme *Bagging* [12, 40, 168, 42] est la contraction de *Bootstrap Aggregating*. Cet algorithme consiste à construire un ensemble de classifieurs à partir de différents rééchantillonages d'un même ensemble de données d'apprentissage. Les prédictions de chaque classifieur peuvent ensuite être combinées par un vote majoritaire ou tout autre méthode de fusion. Le pseudo-code est donné dans l'algorithme 1.

Idéalement, les différents ensembles d'apprentissage utilisés pour construire chaque classifieur devraient être générés de manière aléatoire à partir de la distribution des données pour le problème posé. Pour des applications pratiques, on dispose en général d'un nombre d'exemples d'apprentissage limité. La génération aléatoire de différents ensembles d'apprentissage pour les classifieurs individuels est donc simulée par tirages aléatoires avec remise d'exemples à partir des données initiales de manière à construire L ensembles d'entraînement différents. Ces rééchantillonnages des données d'apprentissage sont appelés *bootstrap samples* ou *bootstrap replicates*. La figure 3.6 décrit le fonctionnement de cet algorithme.

La diversité de l'ensemble de classifieurs est donc assurée par les différences au niveau des données d'apprentissage. Les tirages avec remise étant aléatoires, certains exemples peuvent appararaître plusieurs fois parmi les données d'entrainement d'un classifieur donné, alors que

Algorithme 1 L'algorithme Bagging

entrée :

un ensemble d'apprentissage $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ avec $X_i \in \Re^n$ et $Y_i \in \Omega = \{\omega_1, \ldots, \omega_c\}$
un algorithme d'apprentissage C
un entier L spécifiant le nombre de classifieurs à construire

sortie :

un ensemble de classifieurs E

$E = \emptyset$
pour $i = 1$ à L **faire**
- Construire un rééchantillonnage Z_i à partir de l'ensemble d'apprentissage initial Z
- Appeler l'algorithme C de manière à entrainer un classifieur C_i sur Z_i
- Ajouter C_i à l'ensemble $E = E \cup C_i$

fin pour
retourner l'ensemble de classifieurs E

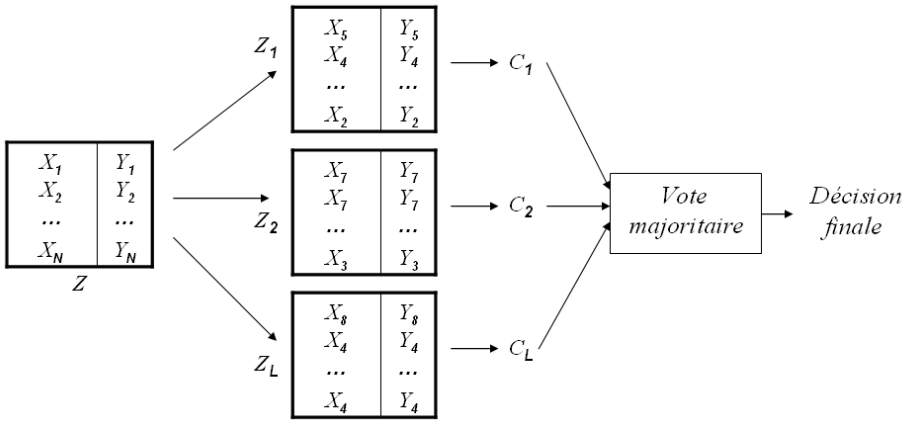

FIG. 3.6 – Description schématique de l'apprentissage d'un ensemble de classifieurs par Bagging et de l'agrégation usuelle par vote majoritaire.

d'autres peuvent ne pas être utilisés. Pour tirer parti de ces différences au niveau des données d'apprentissage, le type de classifieur de base doit être instable. L'instabilité d'un algorithme de classification traduit le fait qu'une légère modification des données d'apprentissage entraine des différences importantes au niveau de l'estimation des frontières de décision. Les arbres de décision et les réseaux de neurones, décrits au chapitre précédent, sont des exemples de classifieurs instables souvent utilisés dans la littérature. Remarquons que la construction des différents classifieurs se fait de manière indépendante, ce qui rend possible une parallélisation de l'algorithme, tant au niveau de l'apprentissage que pour l'évaluation d'un nouvel exemple. Les seuls paramètres du Bagging sont le nombre et la nature des classifieurs utilisés ainsi que la taille des ensembles d'apprentissage obtenus par rééchantillonnage.

Le *Boosting* [57, 187, 189, 141] désigne une méthode générale permettant d'améliorer la précision d'un algorithme d'apprentissage donné. Son principe repose sur l'hypothèse qu'il est plus facile de déterminer un grand nombre de règles décisionnelles grossières (*rules of thumb*)

et modérément précises et de les combiner, plutôt que de chercher à construire directement un classifieur complexe et minimisant le nombre d'exemples mal classés. Ces règles élémentaires sont construites de manière itérative. Le nombre d'itérations T est un paramètre de l'algorithme. L'idée fondamentale du Boosting est que les règles de décisions élémentaires sont déterminées en focalisant l'apprentissage sur les exemples *difficiles*, c'est à dire les exemples mal classés de manière répétée au cours des itérations successives. Cette modification de l'importance donnée aux exemples est réalisée en maintenant une distribution de poids sur l'ensemble d'apprentissage au cours des différentes itérations. Les exemples mal classés voient leur poids augmenté, de sorte que la règle élémentaire suivante se concentre en priorité sur ces exemples difficiles. Les différentes règles sont alors combinées par un vote pondéré par l'erreur d'apprentissage sur la distribution de poids. La figure 3.7 illustre ce principe de manière simplifiée en décrivant la construction et l'agrégation de trois règles élémentaires apprises sur des distributions de poids différentes.

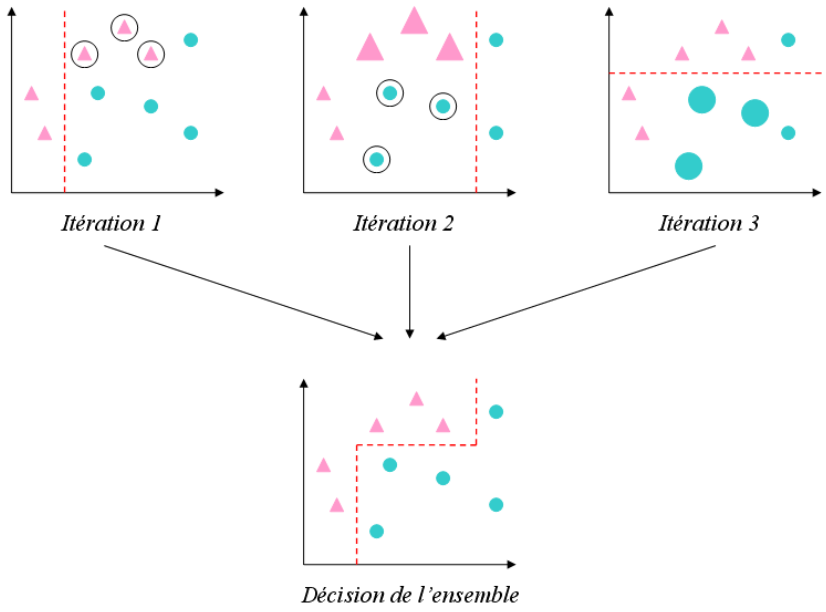

FIG. 3.7 – Apprentissage et agrégation de classifieurs par la méthode du Boosting

Une première règle est construite à partir d'une distribution de poids uniforme. Les exemples mal classés voient leur importance augmentée lors de l'itération suivante, ce qui se traduit par une frontière de décision différente. Ce procédé est itéré plusieurs fois, et les règles élémentaires apprises sont agrégées de sorte que le classifieur global possède une précision bien plus grande que celle des règles individuelles qui le constituent.

Le premier algorithme exploitant ce principe, Adaboost, pour *Adaptative Boosting*, est décrit dans [57, 190]. Le pseudo-code correspondant est donné dans l'algorithme 2.

L'algorithme Adaboost requiert un nombre d'itérations T correspondant au nombre de règles élémentaires qui doivent être construites. L'algorithme d'apprentissage ou apprenant faible (*weak learner*) est un paramètre d'entrée. L'implémentation proposée par Freund [57] manipule un ensemble d'apprentissage $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ où Y_i est à valeur dans $\{-1, +1\}$. Les implémentations multi-classes de cet algorithme seront décrites plus en détail à la fin de ce chapitre. Adaboost définit sur cet ensemble d'apprentissage une distribution de poids D_t. Cette distribution, uniforme à l'origine, est modifiée à chaque itération. Le poids des exemples mal

Algorithme 2 L'algorithme Adaboost

entrée :

un ensemble d'apprentissage $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ avec $X_i \in \Re^n$ et $Y_i \in \{-1,\ +1\}$
un algorithme d'apprentissage C
un entier T spécifiant le nombre d'itérations

sortie :

la règle de classification globale H

Initialiser la distribution $D_1(i) = \frac{1}{N}$, $i = 1, \ldots, N$
pour $t = 1$ à T **faire**

- Appeler l'algorithme C et déterminer l'hypothèse faible h_t sur la distribution D_t
- Calculer l'erreur de h_t : $\epsilon_t = \sum_{i=1}^{N} D_t(i)\,[Y_i \neq h_t(X_i)]$
- Choisir $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- Mettre à jour la distribution D_{t+1} : $D_{t+1}\,(i) = \frac{D_t(i) \times e^{-\alpha_t Y_i h_t(X_i)}}{K_t}$

fin pour
retourner la procédure de classification globale $H(X) = signe\left(\sum_{t=1}^{T} \alpha_t h_t(X)\right)$

classés est augmenté alors que les exemples correctement étiquetés voient leur importance diminuée. De cette manière, la règle apprise à l'itération suivante va chercher en priorité à classer correctement les exemples ayant entrainé des erreurs.

Un algorithme d'apprentissage est utilisé à chaque itération pour déterminer une hypothèse faible h_t. Ces hypothèses faibles correspondent à des classifieurs ou apprenants faibles C_t, mais nous adoptons la notation h_t de manière à respecter la terminologie utilisée dans la littérature relative au Boosting. En effet, le principe du Boosting consiste à *affaiblir* volontairement la précision des classifieurs de manière à les spécialiser sur certains exemples. C'est pourquoi on fait parfois référence au Boosting sous le terme d'Apprentissage Faible (*Weak Learning*). L'erreur de chaque hypothèse faible, notée ϵ_t, est évaluée sur sa distribution D_t :

$$\epsilon_t = \sum_{i,\ h_t(X_i) \neq Y_i} D_t(i) \tag{3.12}$$

où $D_t(i)$ désigne le poids donné à l'exemple i au cours de l'itération t. Le calcul de ϵ_t suppose que le classifieur de base utilisé soit capable de manipuler les poids donnés aux différents exemples. En pratique, il est possible de créer un rééchantillonnage de l'ensemble d'apprentissage d'origine tenant compte de cette distribution de sorte que les exemples de poids plus important apparaissent plusieurs fois alors que les exemples possédant les poids les plus faibles sont ignorés.

La distribution est alors mise à jour en tenant compte de l'erreur de la règle précédente :

$$D_{t+1}(i) = \frac{D_t(i)}{K_t} \times \begin{cases} e^{-\alpha_t} & si \quad h_t(X_i) = Y_i \\ e^{\alpha_t} & si \quad h_t(X_i) \neq Y_i \end{cases} \tag{3.13}$$

où K_t désigne une constante de normalisation choisie de manière à ce que $\sum_{i=1}^{N} D_{t+1}(i) = 1$. La modification de la distribution dépend directement de l'erreur de la règle précédente. Cette modification est donnée par le coefficient α, exprimé par :

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{3.14}$$

L'expression de α implique que chaque hypothèse faible doit avoir une erreur ϵ_t inférieure strictement à 0.5 de manière à conserver la cohérence lors de la mise à jour de la distribution D_t.

L'évaluation d'un nouvel exemple X est réalisée par l'hypothèse forte notée $H(X)$, obtenue par agrégation de l'ensemble des hypothèses faibles h_t déterminées au cours de l'apprentissage, chacune de ces hypothèses faibles étant pondérée par α_t. La classe attribuée à X est exprimée par :

$$\omega^* = H(X) = signe\left(\sum_{t=1}^{T} \alpha_t h_t(X)\right) \tag{3.15}$$

L'algorithme décrit dans [57] effectue une agrégation de classifieurs de type *décision stump* pour résoudre un problème binaire. Ce type de classifieur correspond à un arbre de décision binaire comportant un seul nœud éclaté en deux feuilles étiquetées par des classes distinctes. La définition du boosting peut cependant être étendue à tout type de classifieur, celui-ci étant un paramètre de l'algorithme. Le boosting a été appliqué avec succès aux réseaux de neurones [195, 147], aux arbres de décisions [168, 42], aux réseaux bayésiens naïfs [150, 54], et plus récemment aux SVM [125].

Le Bagging et le Boosting sont parfois regroupés sous le terme de meta-algorithmes d'apprentissage (*meta-learning algorithms*), du fait de leur capacité à améliorer les facultés prédictives d'un algorithme de classification indépendamment de sa nature. Remarquons que la construction des classifieurs de base pour les approches basées sur le boosting s'effectue de manière incrémentale. La distribution des poids D_t nécessaire à l'apprentissage du classifieur à l'itération t suppose que le classifieur déterminé à l'itération $t-1$ soit connu, puisque son erreur contribue directement à l'évaluation de D_t. Les algorithmes basés sur le boosting sont donc de nature séquentielle et l'apprentissage des classifieurs individuels ne peut pas être réalisé en parallèle.

Les études comparatives décrites dans [40, 154, 42] ont montré que le Boosting constitue la méthode d'agrégation de classifieurs la plus efficace en pratique. Contrairement au Bagging, qui constitue une technique expérimentale pour la création d'ensemble de classifieurs, le Boosting a donné lieu à de nombreux travaux théoriques notamment sur la vitesse de convergence de l'algorithme, sur l'existence de bornes relatives à l'erreur, sur sa résistance au sur-apprentissage [101] ou encore sur le lien existant entre le Boosting et la théorie des marges. Cependant, il nous semble que le Boosting présente plusieurs inconvénients majeurs. La présence d'une quantité importante de bruit dans les données altère grandement la précision de l'ensemble et entraine généralement une tendance au sur-apprentissage [40]. De nombreuses contributions ont été apportées pour proposer des solutions aux faiblesses de l'algorithme Adaboost. Elles seront décrites plus en détail dans la suite de ce chapitre.

Wang et Wang [222] proposent une unification des approches basées sur la Bagging et le Boosting. Les auteurs suggèrent que ces deux approchent sont basées sur une distribution de poids définie sur l'ensemble d'apprentissage. Le Boosting et le Bagging ne diffèrent que par la manière dont est déterminée cette distribution. Dans le cas du Bagging, cette distribution est déterminée aléatoirement en raison des tirages avec remise. Les approches basées sur le Boosting, en revanche, calculent cette distribution de manière à spécialiser les classifieurs élémentaires sur certains exemples difficiles. Cette unification entre les deux approches nous laisse penser qu'une plus grande diversité de l'erreur entre les classifieurs doit certainement être un facteur important sur la précision globale de l'ensemble. Il nous semble donc important d'approfondir notre étude de la diversité dans les ensembles de classifieurs en y consacrant une section dans la suite de ce chapitre.

3.2.3 Mixture d'experts

Un autre type d'approche ensembliste est désigné sous le terme *Mixture d'experts* [98, 105, 8]. L'architecture employée consiste en un ensemble de prédicteurs muni d'une entité additionnelle appelée *Gating Network*, auquel nous faisons référence sous le terme de *Réseau Superviseur*. Le réseau superviseur est généralement un réseau de neurones sans couche cachée, et dont la couche de sortie comporte un nombre d'unités égal à la taille L de l'ensemble. Il se comporte comme un classifieur additionnel, auquel sont présentés les différents exemples d'apprentissage, mais dont la tâche consiste à déterminer la contribution de chaque expert dans la décision globale de l'ensemble, comme décrit dans la figure 3.8. Lors de l'évaluation d'un nouvel exemple X, les unités de sortie du réseau superviseur produisent un ensemble $\{w_1(X), \ldots, w_L(X)\}$ correspondant aux poids que possède chaque classifieur dans la décision globale de l'ensemble. L'idée principale de cette architecture est d'entraîner simultanément les prédicteurs C_i et le superviseur durant le processus de création de l'ensemble.

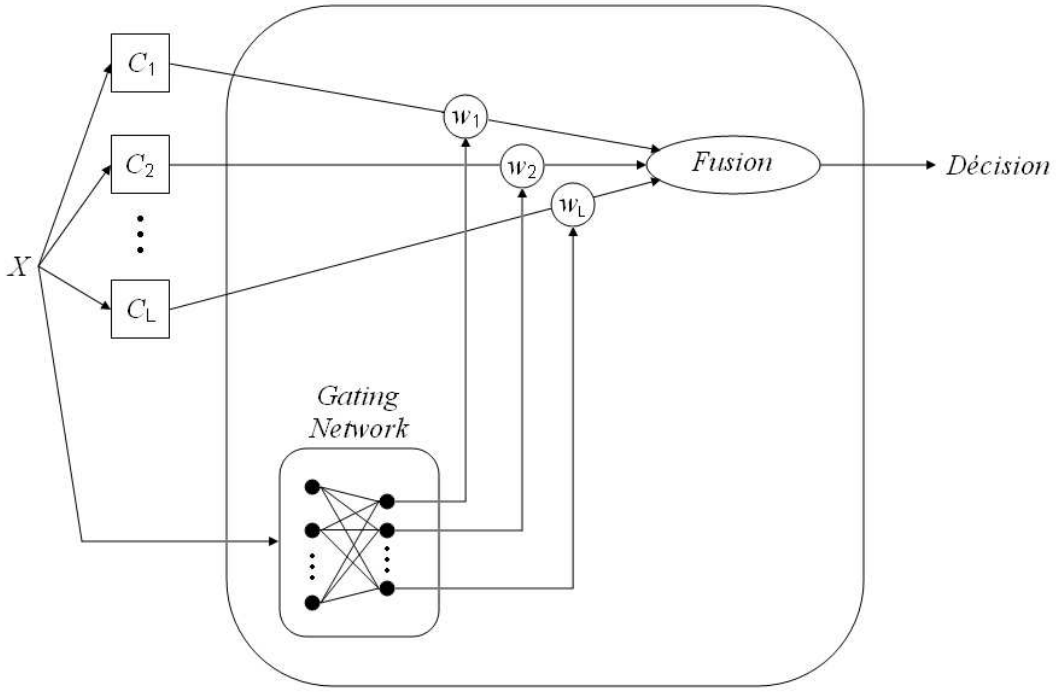

Fig. 3.8 – Classification par une mixture d'experts

La justification de cette architecture repose sur l'hypothèse que certains des classifieurs de l'ensemble sont spécialisés sur une région de l'espace des attributs. On peut considérer les sorties $w_i(X)$ produites par le superviseur comme les probabilités pour que le classifieur C_i soit le plus compétent pour évaluer l'exemple X. Dans le cas usuel où les experts correspondent à des réseaux de neurones, la littérature propose deux principales approches pour l'apprentissage d'une mixture d'experts. La première consiste à utiliser l'algorithme de rétropropagation présenté dans le chapitre précédent. L'apprentissage revient alors à réaliser une descente de gradient pour un réseau possédant une structure complexe. La seconde méthode consiste à utiliser l'algorithme EM (*expectation maximization*) [103, 104], plus rapide que la première solution. Cependant, aucune contrainte particulière n'est fixée concernant l'ensemble de classifieurs en lui-même. Avnimelech

et Intrator [4] suggèrent de construire l'ensemble par des méthodes basées sur le Bagging et le Boosting et de lui adjoindre une stratégie de sélection pour combiner les sorties produites par chaque classifieur. Kuncheva [113] propose trois stratégies différentes pour combiner les décisions d'un ensemble de classifieurs dans le cadre d'une mixture d'experts :

- Le vote pondéré des experts : pour tout vecteur X à classer, les sorties du réseau superviseur $w_i(X)$ sont utilisées en tant que coefficients de pondération associés à chacun des classifieurs de l'ensemble.

$$f_{mix}(X) = \sum_{i=1}^{L} w_i(X) C_i(X) \tag{3.16}$$

- Considérer uniquement l'expert de poids maximal (*Winner-takes-all*) : seul le classifieur possédant le poids maximal est utilisé pour évaluer X

$$f_{mix}(X) = C_i(X),\ w_i(X) = \max_{j=1}^{L} w_j(X) \tag{3.17}$$

- la sélection stochastique : l'évaluation du vecteur X est effectuée en rééchantillonnant l'ensemble des prédicteurs $\{C_1, \ldots, C_L\}$ selon la distribution définie par $w_1(X), \ldots, w_L(X)$.

Brown [20] considère quant à lui la mixture d'experts comme une approche *modulaire* pour la création d'un ensemble de classifieurs, par opposition au Bagging et au Boosting où la contribution des différents éléments de l'ensemble n'est pas réévaluée pour chaque vecteur exemple X à classer. Kuncheva [113] définit la mixture d'experts comme faisant partie d'une approche plus large appelée *sélection de classifieurs* consistant à sélectionner et combiner les prédictions d'une sous-partie de l'ensemble des experts, cette sous-partie pouvant ne contenir qu'un unique élément. L'idée qui consiste à réaliser l'apprentissage du réseau superviseur parallèlement à celui des membres de l'ensemble permet également de considérer la mixture d'experts comme une forme de stacking. Le réseau superviseur joue ici le rôle d'un classifieur d'ordre supérieur qui combine les prédictions des membres de l'ensemble. Une architecture dérivant de celle décrite dans la figure 3.8, appelée mixtures d'experts hiérarchiques [173, 223], consiste à organiser de manière arborescente les experts en utilisant plusieurs réseaux superviseurs. L'apprentissage par mixture d'experts est utilisé en grande majorité pour l'approximation de fonctions [22], bien que l'on trouve dans la littérature des applications à des problèmes de classification [25] ou au traitement de séries chronologiques [137].

3.3 La diversité dans les ensembles de classifieurs

La supériorité de l'algorithme Adaboost, s'appuyant sur l'agrégation de règles volontairement peu précises, sous-entend que l'utilisation d'algorithmes ensemblistes est une technique efficace lorsque les classifieurs constituant l'ensemble présentent une certaine diversité. Dans cette section, nous présentons une approche formelle de cette intuition en décrivant comment la diversité intervient dans le cadre de la décomposition de l'erreur. Nous donnons ensuite différents moyens issus de la littérature pour quantifier cette diversité. Nous terminons enfin par une étude de différents algorithmes ensemblistes, tels que les méthodes basées sur les Forêts Aléatoires ou l'apprentissage par Corrélation Négative, en précisant leurs liens avec la diversité.

3.3.1 Décomposition Biais-Variance-Bruit

De nombreux chercheurs se sont intéressés à la justification formelle du principe intuitif selon lequel un ensemble de prédicteurs permet de réduire l'erreur par rapport à l'utilisation d'un unique classifieur de même type. Dans le cas de la régression, la décomposition de l'erreur a été initialement proposée par Geman *et al.* [73], dans le cas des réseaux de neurones. Elle consiste à décomposer l'expression de l'erreur en trois termes, correspondant respectivement au biais, à la variance et à la covariance. La décomposition selon l'ambiguité, proposée par Krogh et Vedelsby [111], montre que pour un exemple donné, l'erreur d'un ensemble de classifieurs est inférieure ou égale à la moyenne des erreurs commises par chacun de ses membres. Brown [20] établit un lien entre ces deux décompositions, en soulignant qu'elles ne sont cependant pas applicables dans le cas où la fonction de coût est en 0-1, fonction qui est pourtant la plus souvent utilisée pour des problèmes de classification.

Dans le cadre de la classification, une justification formelle des performances obtenues par les méthodes du Bagging et du Boosting repose sur la décomposition de l'erreur en trois termes, correspondant respectivement au biais, à la variance et au bruit contenu dans les données. Considérons un ensemble de classifieurs $E = \{C_1, \ldots, C_L\}$. On note $P(\omega_i|X)$ la probabilité *a posteriori* de chacune des c classes du problème et $P_C(\omega_i|X)$ la probabilité qu'un classifieur C choisi aléatoirement dans E attribue la classe ω_i à l'exemple X.

La probabilité d'erreur, pour un exemple X donné, est définie par :

$$P(Err|X) = 1 - \sum_{i=1}^{c} P(\omega_i|X) P_C(\omega_i|X) \tag{3.18}$$

et correspond à la probabilité qu'un classifieur choisi aléatoirement attribue la mauvaise classe à l'exemple X. Kohavi et Wolpert [109] décomposent cette erreur sous la forme :

$$P(Err|X) = biais + variance + bruit \tag{3.19}$$

avec

$$biais = \frac{1}{2} \sum_{i=1}^{c} \left(P(\omega_i|X) - P_C(\omega_i|X)\right)^2 \tag{3.20}$$

$$variance = \frac{1}{2} \left(\sum_{i=1}^{c} 1 - P_C(\omega_i|X)^2 \right) \tag{3.21}$$

$$bruit = \frac{1}{2} \left(\sum_{i=1}^{c} 1 - P(\omega_i|X)^2 \right) \tag{3.22}$$

Le biais correspond de manière générale à une estimation moyenne de la différence entre la classe prédite et la classe réelle. Dans la formulation proposée dans [109], cette estimation correspond à la différence entre la distribution réelle $P(\omega_i|X)$ et la distribution estimée $P_C(\omega_i|X)$. L'expression associée à la variance exprime la variabilité de la classe calculée indépendamment de la classe réelle. A l'inverse, le bruit désigne la variabilité de la classe réelle indépendamment de celle calculée par le classifieur.

Une autre décomposition de l'erreur est proposée par Breiman [15]. Soit ω^* la classe ayant la plus importante probabilité *a posteriori* $P(\omega|X)$ et $\widehat{\omega}^*$ la classe dont la probabilité $P_C(\omega_i|X)$ pour un classifieur choisi aléatoirement dans E est maximale. La décomposition de l'erreur selon Breiman s'écrit :

$$biais = (P(\omega^*|X) - P(\widehat{\omega}^*|X))P_C(\widehat{\omega}^*|X) \tag{3.23}$$

$$variance = \sum_{\omega_i \neq \widehat{\omega}^*} (P(\omega^*|X) - P(\omega_i|X))P_C(\omega_i|X) \tag{3.24}$$

$$bruit = 1 - P(\omega^*|X) \tag{3.25}$$

Dans cette formulation, la variance est également appelée *dispersion* car elle exprime les variations de la distribution de probabilité estimée. Remarquons que le terme associé au bruit correspond à l'erreur bayésienne décrite dans le chapitre précédent, qui correspond à la quantité d'erreur incompressible et inhérente au problème posé.

Domingos [46, 47] propose une unification de la décomposition de l'erreur en utilisant une fonction de coût notée l. La formulation obtenue dans le cas d'une fonction de coût en 0-1 est donnée par :

$$biais = l(\widehat{\omega}^*, \omega^*) \tag{3.26}$$

$$variance = \sum_{\omega_i \neq \widehat{\omega}^*} P_C(\omega_i|X) = 1 - P_C(\widehat{\omega}^*|X) \tag{3.27}$$

$$bruit = 1 - P(\omega^*|X) \tag{3.28}$$

Le biais ne dépend pas de la classe donnée par un classifieur choisi aléatoirement mais seulement de la classe majoritairement attribuée par l'ensemble et de la valeur optimale de la classe. Il ne peut prendre que les valeurs 0 et 1, correspondant respectivement à une classification correcte et incorrecte.

Ces décompositions de l'erreur ont toutes pour objectif de fournir un cadre théorique pour expliquer comment l'utilisation d'un ensemble de prédicteurs permet en général de réduire de manière significative l'erreur de classification.

Considérons le cas d'un ensemble d'arbres de décisions entrainés sur différents ensembles d'apprentissage. Ce type de classifieur est instable, les éléments de l'ensemble auront donc un biais relativement faible mais une variance importante. Kuncheva [113] exprime cette possibilité d'augmenter le biais d'un type de classifieur de manière à diminuer sa variance sous le terme de *dilemme biais-variance*. Une illustration de ce dilemme est illustré dans [125]. Un ensemble de SVM est construit en utilisant l'algorithme Adaboost, bien que les SVM soient un type de classifieur relativement précis, ce qui va à l'encontre du principe de *weak learning* sur lequel repose le Boosting. Pour rendre ce type de classifieur exploitable dans un ensemble, les auteurs proposent l'utilisation de SVM gaussien pour lesquels le paramètre σ est volontairement dégradé, de manière à augmenter la variance au prix d'un biais accru. Kuncheva [113] assimile la variance au sur-apprentissage. Le classifieur modélise avec précision les données d'apprentissage, exhibant une variance importante lorsqu'il généralise ce modèle sur de nouveaux exemples. A l'inverse, le biais est associé au sous-apprentissage. Le modèle appris est éloigné de l'optimal, ce qui entraîne une erreur importante. Ce cadre théorique suggère que le Bagging permet d'améliorer les performances d'un unique classifieur en réduisant la variance tout en maintenant un biais constant [62, 13]. Le lien entre la décomposition de l'erreur et le Boosting est davantage sujet à discussion. Dietterich [40] montre que les premières itérations de l'algorithme tendent à réduire le biais, avant de contribuer à une diminution de la variance. A l'inverse, les travaux présentés

dans [78, 191, 47, 218] suggèrent que le succès du Boosting s'explique par la théorie des marges. Il semble cependant que l'efficacité d'un ensemble de classifieurs est intimement liée à la notion de diversité des erreurs commises par les membres de l'ensemble.

3.3.2 Quantification de la diversité

Ce sont en grande partie les travaux réalisés par Kuncheva [120, 121, 117, 118, 119, 205, 207] qui ont permis d'étudier et de quantifier la diversité d'un ensemble de classifieurs. Ces travaux ont ouvert la voie à d'autres travaux sur l'utilisation de la diversité de manière active au cours du processus de construction d'ensembles de classifieurs. Les approches qui introduisent la diversité de manière intuitive se révèlent particulièrement performantes d'un point de vue expérimental, à l'image de l'algorithme Adaboost, qui diminue la précision des membres de l'ensemble au profit d'une plus grande diversité. Paradoxalement, les approches proposant une utilisation active de cette diversité ne donnent pas toujours les performances attendues et la connaissance des méchanismes de la diversité dont on dispose à l'heure actuelle n'est pas encore bien établie [112].

De manière empirique, l'évaluation de la diversité d'un groupe d'individus soulève plusieurs questions. Doit-on considérer la diversité de la population de manière globale, au risque d'ignorer les disparités internes importantes que peuvent présenter deux individus, ou à l'inverse, doit-on évaluer la différence entre toutes les paires d'individus possibles dans la population. Ces questions permettent de distinguer deux groupes de mesures : les mesures dites *pairwise*, qui évaluent la distance entre deux prédicteurs en fonction de leur désaccord sur les mêmes exemples, et les mesures globales, qui quantifient la diversité de la population dans son ensemble.

De manière à quantifier le désaccord entre deux classifieurs et ainsi définir différentes mesures de diversité, on peut utiliser une abstraction des sorties des classifieurs, en se ramenant à un niveau binaire appelé niveau oracle. Pour un exemple X_j donné, on cherche uniquement à savoir si le classifieur C_i détermine la bonne classe ou si l'étiquette attribuée est incorrecte :

$$C_i(X_j) = \begin{cases} 1 & si \quad C_i(X_j) = Y_j \\ 0 & si \quad C_i(X_j) \neq Y_j \end{cases} \tag{3.29}$$

Quel que soit le nombre de classes, il est toujours possible de se ramener au niveau oracle. On peut alors définir une matrice d'incidence permettant d'évaluer les probabilités de désaccord entre deux classifieurs C_i et C_j, comme le montre la table 3.6.

Tab. 3.6 – Matrice d'incidence définie pour deux classifieurs au niveau oracle.

	$C_j(X)$ est correcte	$C_j(X)$ est erronnée
$C_i(X)$ est correcte	a	b
$C_i(X)$ est erronnée	c	d

Les termes de la matrice correspondent aux probabilités pour lesquelles C_i et C_j sont en accord ou en désaccord concernant la classe attribuée pour les mêmes exemples, avec $a+b+c+d = 1$. Ainsi, les entrées a et d dénotent les exemples pour lesquelles les deux classifieurs considérés attribuent la même étiquette. Plusieurs mesures peuvent être calculées sur cette matrice pour deux classifieurs donnés.

Yule [229] définit une mesure appelée *Q Statistic* pour deux classifieurs C_i et C_j :

$$Q_{i,j} = \frac{ad - bc}{ad + bc} \tag{3.30}$$

Pour deux classifieurs statistiquement indépendants, $Q_{i,j} = 0$. Plus généralement, Q varie entre -1 et 1. Deux classifieurs ayant tendance à classer correctement les mêmes exemples ont une valeur Q positive.

La corrélation ρ entre deux classifieurs peut également être calculée sur la matrice décrite dans la table 3.6. L'expression de cette corrélation est :

$$\rho_{i,j} = \frac{ad - bc}{\sqrt{(a+b)(c+d)(a+c)(b+d)}} \tag{3.31}$$

Pour deux classifieurs donnés, Q et ρ ont le même signe et $|\rho| \leq |Q|$. La mesure de désaccord [206] entre deux classifieurs, notée $D_{i,j}$, peut être calculée sur la matrice d'incidence décrite précédemment. Elle correspond à la probabilité que deux classifieurs C_i et C_j attribuent des étiquettes différentes à un même exemple :

$$D_{i,j} = b + c \tag{3.32}$$

Un autre choix intuitif pour mesurer la diversité entre deux classifieurs est la mesure de *double faute* DF [183] :

$$DF_{i,j} = d \tag{3.33}$$

Cette mesure correspond à la probabilité que deux classifieurs C_i et C_j fournissent simultanément une étiquette incorrecte à un même exemple. Le principe sur lequel repose cette mesure de diversité est qu'il est plus important de tenir compte des erreurs simultanées commises par les membres de l'ensemble plutôt que d'évaluer la probabilité qu'ils soient en accord.

Fleiss et al. [97] proposent l'utilisation du κ en tant que critère permettant d'évaluer le désaccord respectif entre deux prédicteurs. Si l'on considère ces deux prédicteurs au niveau oracle, l'expression du κ est donnée par :

$$\kappa_{i,j} = \frac{2(ac - bd)}{(a+b)(c+d) + (a+c)(b+d)} \tag{3.34}$$

Une définition plus générale du κ peut être donnée dans le cas où les membres de l'ensemble attribuent à un exemple une étiquette discrète parmi c classes possibles. Soit $\mathcal{M} = m_{ks}$ la matrice d'incidence $c \times c$ correspondant à deux classifieurs C_i et C_j. Chaque terme m_{ks} de la matrice correspond à la proportion d'exemples affectés aux classes ω_k et ω_s respectivement par C_i et C_j. L'expression de la diversité entre C_i et C_j est donnée par :

$$\kappa_{i,j} = \frac{\sum_k m_{kk} - ABC}{1 - ABC} \tag{3.35}$$

où m_{kk} désigne l'accord observé et ABC est appelé *accord par chance* :

$$ABC = \sum_k \left(\sum_s m_{ks} \right) \left(\sum_s m_{sk} \right) \tag{3.36}$$

Une valeur faible de $\kappa_{i,j}$ signifie un grand désaccord entre C_i et C_j, et par conséquent une diversité accrue. Une extension du κ proposée par Margineantu et Dietterich [138] consiste à utiliser des diagrammes *kappa-erreur* pour étudier l'influence de la diversité sur les performances obtenues par deux algorithmes emsemblistes. Un tel diagramme, tiré de [179], est donné dans la figure 3.9.

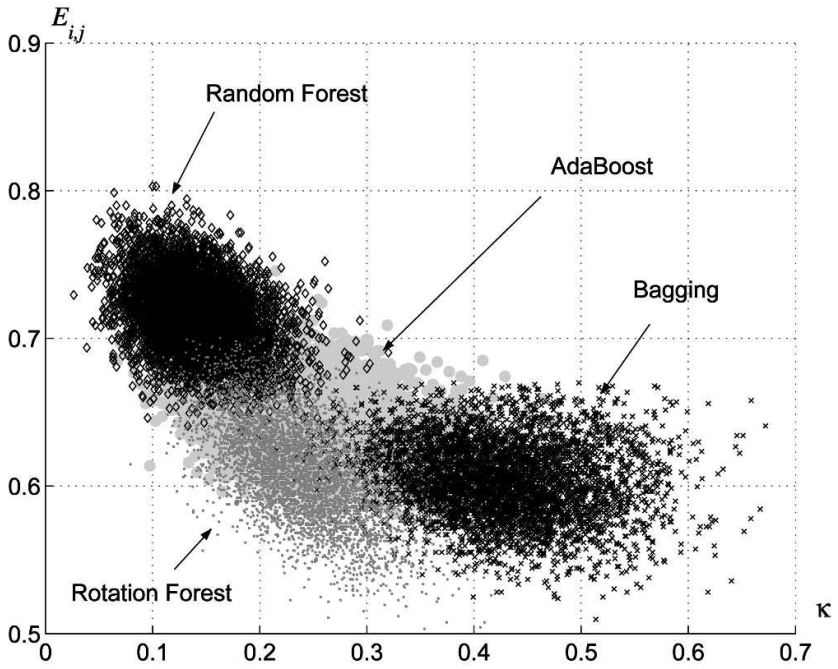

FIG. 3.9 – Un exemple de diagramme Kappa-Erreur

Pour un ensemble comportant L classifieurs, il existe $\frac{L(L-1)}{2}$ paires possibles pour lesquelles on peut calculer les valeurs $\kappa_{i,j}$. Chaque point figurant sur le diagramme correspond à une paire de classifieurs $(C_i,\ C_j)$ ayant respectivement pour abscisse et ordonnées la diversité entre ces deux classifieurs $\kappa_{i,j}$ et l'erreur moyenne définie par $E_{i,j} = \frac{E_i+E_j}{2}$. Un ensemble de L classifieurs est représenté par le nuage de $\frac{L(L-1)}{2}$ points correspondants à toutes les paires qui le constituent. La forme et la position de ces nuages de points dans le plan renseigne sur le comportement de différents algorithmes ensemblistes vis à vis de la diversité et la précision individuelle de ses membres. Sur l'exemple donné dans la figure 3.9, l'ensemble obtenu par l'algorithme *Random Forest* possède une diversité plus importante que l'ensemble de classifieurs résultant du Bagging, au détriment d'une précision moindre. Idéalement, l'ensemble le plus satisfaisant se situe dans la partie inférieure gauche du diagramme, ce qui traduit le fait que les classifieurs qui le constituent possèdent une grande précision et présentent une diversité importante.

Il est possible de quantifier la diversité de manière globale relativement à un ensemble de classifieurs. La mesure d'entropie [33] repose sur le constat qu'un ensemble de L classifieurs peut être considéré *divers* si pour un exemple donné, $\left\lfloor \frac{L}{2} \right\rfloor$ classifieurs attribuent la bonne classe et les $L - \left\lfloor \frac{L}{2} \right\rfloor$ classifieurs restants sont incorrects (ou inversement). L'expression de l'entropie est alors donnée par :

$$Ent = \frac{1}{N} \times \frac{2}{L-1} \sum_{j=1}^{N} min \left\{ \left(\sum_{i=1}^{L} y_{j,i} \right), \left(L - \sum_{i=1}^{L} y_{j,i} \right) \right\} \tag{3.37}$$

où N désigne le nombre d'exemples d'apprentissane et $Ent \in [0, 1]$, 1 indiquant une diversité maximale de l'ensemble. Kuncheva [113] propose une mesure de diversité de l'ensemble dérivant de la définition de la variance proposée par Kohavi et Wolpert [109] dans le cas de la

décomposition de l'erreur. Cette mesure, notée KW, est définie par :

$$KW = \frac{1}{NL^2} \sum_{j=1}^{N} H(X_j) \times (L - H(X_j)) \tag{3.38}$$

où $H(X_j)$ désigne le nombre de classifieurs ayant correctement étiqueté l'exemple X_j. La valeur κ peut également être calculée globalement pour l'ensemble. Soit p_{moy} la précision individuelle moyenne des classifieurs de l'ensemble. Le κ s'exprime alors :

$$\kappa = 1 - \frac{\frac{1}{L}\sum_{j=1}^{N} H(X_j) \times (L - H(X_j))}{N(L-1) \times p_{moy} \times (1 - p_{moy})} \tag{3.39}$$

Remarquons que cette expression est différente de celle obtenue en moyennant $\kappa_{i,j}$ sur toutes les paires de classifieurs possibles.

La *diversité généralisée* DG, proposée par Partridge et Krzanowski [157], est définie par :

$$DG = 1 - \frac{p(2)}{p(1)} \tag{3.40}$$

avec

$$p(1) = \sum_{i=1}^{L} \frac{i}{L} p_i \quad et \quad p(2) = \sum_{i=1}^{L} \frac{i}{L} \times \frac{(i-1)}{(L-1)} p_i \tag{3.41}$$

où le terme p_i correspond à la probabilité que i classifieurs choisis aléatoirement dans l'ensemble classent de manière incorrecte un exemple donné et on note $p(i)$ la probabilité pour que i classifieurs choisis aléatoirement attribuent une mauvaise classe à un exemple tiré au hasard. La diversité est maximale lorsque l'attribution d'une classe incorrecte à un exemple par un classifieur donné est accompagnée d'un étiquettage correct d'un autre classifieur de l'ensemble. Dans ce cas, on a $p(2) = 0$ et $DG = 1$. A l'inverse, lorsqu'un classifieur commet une erreur pour un exemple donné et que les autres membres de l'ensemble commettent cette même erreur, la probabilité d'échec simultané $p(2)$ et égale à la probabilité $p(1)$ qu'un classifieur aléatoire attribue la mauvaise classe à un exemple, et on a $DG = 0$.

Avec ces notations, Partridge et Krzanowski [157] définissent la diversité par *coïncidence d'échec* :

$$CFD = \begin{cases} 0 & si \quad p_0 = 1 \\ \frac{1}{1-p_0} \sum_{i=1}^{L} \frac{L-i}{L-1} p_i & si \quad p_0 < 1 \end{cases} \tag{3.42}$$

La diversité maximale au sens de cette mesure est atteinte lorsque l'attribution d'une classe incorrecte à un exemple donné est unique, c'est à dire qu'au plus un unique classifieur est incorrect pour un exemple choisi aléatoirement. Dans ce cas on a $CFD = 1$.

Remarquons qu'il est toujours possible de ramener une mesure de diversité calculée pour deux classifieurs au niveau de l'ensemble en calculant la moyenne sur l'ensemble des paires de classifieurs possibles. Notons $DIV_{i,j}$ la mesure de diversité calculée pour deux classifieurs C_i et C_j et L la taille de l'ensemble. La diversité globale de l'ensemble est définie par :

$$DIV_E = \frac{2}{L(L-1)} \times \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} DIV_{i,j} \tag{3.43}$$

3.3.3 Diversité : comment la générer

Les travaux de Brown [19] fournissent une catégorisation possible des méthodes permettant de générer la diversité lors de la construction d'un ensemble de classifieurs. Une distinction majeure peut être effectuée entre les méthodes tenant compte de manière explicite de la diversité durant le processus de construction de l'ensemble et celles qui en font un usage implicite. Le Bagging fait partie des méthodes implicites. Des réchantillonages aléatoires de données d'apprentissage initiales sont utilisés pour construire un ensemble de classifieurs mais aucune mesure n'est utilisée pour assurer la diversité de l'ensemble obtenu. A l'inverse, Adaboost maintient une distribution de poids sur les exemples d'apprentissage calculée de manière à ce que les différents membres de l'ensemble présentent une certaine forme de diversité. Adaboost fait donc partie des méthodes permettant de manipuler explicitement la diversité. Brown [19] souligne cependant qu'il n'y a aucune garantie sur la forme de la diversité générée.

On peut considérer qu'au cours de l'apprentissage, un classifieur suit une certaine trajectoire dans l'espace des hypothèses possibles. Prenons le cas d'un réseau de neurones dont la structure a été préalablement fixée. Ce réseau ne dépend que de ses poids, déterminés au cours de l'apprentissage par rétropropagation du gradient par exemple. Si l'on considère l'espace des poids possibles pour ce réseau, l'apprentissage revient à faire converger ce réseau de neurones, depuis sa position initiale jusqu'au point de l'espace où l'erreur sur l'ensemble d'apprentissage est minimale. Brown [20] définit trois classes principales de méthodes qui permettent de générer la diversité, selon que ces méthodes modifient l'état initial du classifieur (*starting point in hypothesis space*), l'ensemble des états accessibles par le classifieur au cours de l'apprentissage (*set of accessible hypothesis*) ou la manière dont cet espace des états accessibles est exploré (*traversal of hypothesis space*). Cette classification des méthodes est illustrée dans la figure 3.10 dans laquelle on considère un réseau de neurones ne dépendant que de deux poids.

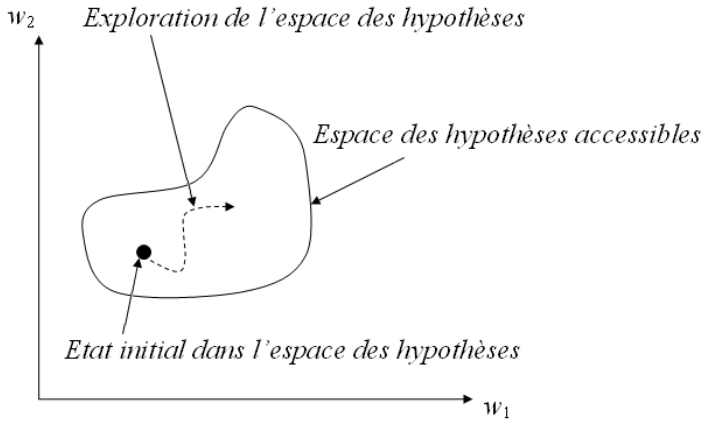

FIG. 3.10 – Classification des méthodes permettant de générer la diversité dans les ensembles de classifieurs. Illustration pour un réseau de neurones simplifié caractérisé par deux poids w_1 et w_2.

Nous donnons ici quelques exemples permettant d'illustrer chacune de ces trois méthodes de génération de la diversité. Le lecteur désirant un recensement plus exhaustif des algorithmes de ces méthodes est invité à consulter les références [20, 19] qui décrivent en détail la taxonomie résumée dans cette section.

La modification de l'état initial d'un classifieur s'applique dans le cas des réseaux de neurones,

où une valeur de départ doit être fixée pour chacun des poids w du réseau. Une distinction est établie à ce niveau entre l'initialisation aléatoire ou *randomization* des poids du réseau [204] et les solutions plus explicites maximisant la distance entre les hypothèses initiales de manière à augmenter l'ensemble des optimum locaux accessibles par rétropropagation [134]. Les comparaisons effectuées par Partridge et Yates [158, 228] montrent que de manière générale, la modification de l'état initial d'un ensemble de réseaux de neurones est moins efficace que des manipulations de la structure ou des données d'apprentissage.

La définition de l'espace des classifieurs accessibles peut être modifiée de deux manières distinctes, selon que l'algorithme effectue des manipulations sur les données ou sur l'architecture du classifieur. Par exemple, limiter la structure d'un réseau de neurones à celle d'un perceptron sans couche cachée restreint l'espace des hypothèses accessibles à l'ensemble des classifieurs permettant de résoudre les problèmes linéairement séparables. Les techniques consistant à travailler sur les données sont les plus fréquemment rencontrées dans la littérature. Comme décrit dans la figure 3.5, les modifications intervenant aux niveaux des données peuvent concerner les exemples eux mêmes ou tout type de filtrage appliqué aux attributs du problème. Le Bagging et le Boosting manipulent les exemples d'apprentissage respectivement de manière implicite et explicite dans le but de spécialiser les classifieurs obtenus dans certaines régions de l'espace des entrées. L'algorithme DECORATE [142] décrit dans la section suivante s'appuie sur la génération d'exemples artificiels étiquetés de manière à maximiser la diversité de l'ensemble obtenu. L'algorithme Random Forest [17] désigne une méthode générale consistant à construire un ensemble d'arbres de décisions entrainés sur des données provenant d'une même distribution mais sur des exemples, attributs ou paramètres différents, de manière à engendrer la diversité entre les classifieurs obtenus. Plus récemment, l'algorithme Rotation Forest [179] propose d'accroître la diversité entre arbres de décision en appliquant une analyse en composantes principales sur des sous-ensembles aléatoires d'attributs, et d'entrainer les arbres de décision sur les données ainsi transformées. Les méthodes relatives aux manipulations de l'architecture du classifieur sont moins répandues. L'utilisation d'un ensemble de réseaux de neurones ayant un nombre différent d'unités cachées est décrite dans [158]. Li *et al.* [125] proposent également de faire varier la structure de SVM à noyaux gaussiens en modifiant la valeur du paramètre γ. Un certain nombre de publications font également référence à l'utilisation d'architectures hybrides, dans lesquelles l'ensemble est constitué par différents types de classifieurs [122, 226].

L'algorithme *NC-Learning* [133, 130] fait partie des méthodes permettant de modifier l'exploration de l'espace des hypothèses dans le cadre d'ensemble de réseaux de neurones dont l'apprentissage s'effectue en parallèle. Un terme de régularisation est ajouté dans la formulation de l'erreur de chacun des prédicteurs, de manière à explorer l'espace des poids possibles tout en maximisant la diversité entre les différents éléments de l'ensemble.

D'autres catégorisations de la diversité sont disponibles dans la littérature [90, 203]. Nous estimons que celle proposée par Brown [19] et résumée dans cette section convient davantage à une analyse empirique des moyens disponibles pour générer la diversité dans un ensemble de classifieurs, en raison de son niveau d'abstraction important. Nous terminons ce chapitre par la description de quelques algorithmes ensemblistes, en mettant en relief la manière dont ils génèrent la diversité durant le processus de construction de l'ensemble.

3.4 Les algorithmes ensemblistes et leurs liens avec la diversité

Dans cette section, nous présentons quelques algorithmes ensemblistes. Nous effectuons une distinction entre ces algorithmes selon qu'ils exploitent la diversité de manière *implicite* ou *expli-*

cite. Nous émettons cependant quelques réserves concernant cette distinction également faite par Brown [19]. En effet, bien qu'Adaboost et ses différentes variantes construisent naturellement des ensembles de classifieurs spécialisés et diversifiés, aucune métrique permettant de quantifier cette diversité n'est utilisée au cours du processus d'apprentissage. La diversité de l'ensemble obtenu est donc évaluée *a posteriori* sur l'ensemble de test de la même manière que les algorithmes inspirés du Bagging, par opposition à l'algorithme DECORATE qui évalue la diversité de l'ensemble à chaque étape du processus de construction. Nous préférons donc désigner les approches dérivant du Boosting sous le terme *pseudo-explicite* de manière à souligner la différence avec l'algorithme DECORATE qui utilise explicitement une mesure pour quantifier la diversité d'un ensemble de classifieurs.

3.4.1 Améliorations du Boosting

L'algorithme Adaboost proposé par Freund [57] a suscité un grand nombre de travaux visant à l'améliorer et également de comprendre ses mécanismes. Les premières modifications apportées à l'algorithme concernent son adaptation à des problèmes multi-classes. Les implémentations multi-classes les plus abouties sont appelées Adaboost.M1 et Adaboost.M2. [59, 60]. Des implémentations basées sur l'utilisation de codes correcteurs d'erreurs sont également proposées dans [209]. Plus récemment, les adaptations SAMME [232] et Adaboost.Mv [174] ont été conçues pour la résolution de problèmes multi-classes.

Le boosting est reconnu comme étant l'une des meilleures méthodes d'agrégation de classifieurs. L'application d'Adaboost à des arbres de décision est d'ailleurs considérée comme la méthode de classification la plus efficace en pratique [85]. Une explication de cette efficacité est issue de la théorie des marges [191]. Il a été montré de manière expérimentale que sur des données dépourvues de bruit, Adaboost est résistant au sur-apprentissage [78, 101].

Cependant, pour des données fortement bruitées, Adaboost présente une très nette tendance au sur-apprentissage et plus particulièrement lorsque le nombre d'exemples d'entrainement est réduit [188, 141, 143]. Cette sensibilité de l'algorithme au bruit s'explique par le fait que les exemples incorrects, où *outliers*, voient leur poids augmenté à tort et de manière significative. Ce phénomène engendre également une perte de vitesse de convergence de l'algorithme car un grand nombre d'itérations est inutilement dédié à l'apprentissage de ces *outliers*. Ces principales faiblesses d'Adaboost sont résumées dans [196]. Partant de ce constat, une part importante des travaux relatifs au Boosting visent à contrôler le poids attribué aux exemples bruités de manière à diminuer le nombre d'itérations consacrées à ces exemples. Ratsch et al. [175] proposent une version régularisée d'Adaboost.

L'utilisation du concept de marges souples est également décrite dans [176]. Nous pouvons également citer différentes adaptations du boosting ayant toutes pour objectif d'assurer le contrôle de la croissance des poids attribués aux exemples difficiles : BrownBoost [58], MadaBoost [45, 44], SmoothBoost [198], LPBoost [37], NadaBoost [148] et iAdaboostsebba04.

Contrairement au Bagging, qui désigne une méthode pratique pour la construction d'ensembles de classifieurs, le Boosting est devenu au cours de ces dernières années une véritable théorie à part entière reposant sur des fondements solides et faisant encore l'objet de nombreuses études à l'heure actuelle [196, 77]. Il suffit pour s'en convaincre de recenser le nombre de travaux relativement récents dédiés à l'étude de cet algorithme. Brown [19] considère que le Boosting contribue de manière active à la génération de la diversité dans la mesure où la distribution maintenue sur l'ensemble d'apprentissage est guidée par la difficulté des exemples. Les résultats décrits dans [179] tendent à conforter cette hypothèse : ils montrent une diminution de la précision individuelle des membres de l'ensemble au profit d'une diversité accrue constitue une

approche très efficace.

3.4.2 Random Forest et Rotation Forest

Les Forêts Aléatoires (*Random Forest*) [17, 91], désignent une extension du Bagging s'appliquant au cas particulier des arbres de décision. La définition proposée par Breiman [17] considère qu'une Forêt Aléatoire correspond à un ensemble d'arbres de décision entrainés sur des ensembles d'apprentissage indépendants et provenant de la même distribution. Kuncheva [113] souligne que cette définition permet d'interpréter de plusieurs façon cet algorithme. Tout ensemble d'arbres de décision obtenu en effectuant des manipulations au niveau des échantillons, des attributs ou des paramètres utilisés pour construire un arbre donné peut être considéré comme conforme à cette définition. L'une des implémentations les plus efficaces de cet algorithme repose sur une sélection aléatoire des attributs [89]. Chaque arbre de décision est entrainé sur un rééchantillonage aléatoire des exemples disponibles, tel qu'il est décrit dans l'algorithme 1 à la page 70. La différence avec le Bagging se situe au niveau de l'apprentissage individuel des arbres de décision. Pour un nœud donné de l'arbre, un sous ensemble d'attributs $\mathcal{S}'$ est sélectionné aléatoirement parmi les n attributs du problème. L'éclatement du nœud s'effectue par rapport à un attribut $x' \in \mathcal{S}'$. Un nouveau sous-ensemble aléatoire $\mathcal{S}'$ est sélectionné de cette manière pour chaque nœud de l'arbre. Cet algorithme présente l'avantage, au même titre que le Bagging dont il est issu, de pouvoir être aisément parallélisé puisque l'apprentissage de chaque arbre de décision est indépedant des autres.

L'algorithme Rotation Forest [179] s'appuie sur l'heuristique précédante consistant à manipuler les attributs. A chaque itération, un sous ensemble aléatoire d'attributs est sélectionné comme précédemment. Une analyse en composante principale est utilisée de manière à appliquer une transformation aux données utilisées pour réaliser l'apprentissage des classifieurs. Cette transformation est appliquée à chaque fois qu'un arbre de décision est ajouté à l'ensemble. En comparant cette implémentation aux autres méthodes ensemblistes à l'aide de diagrammes kappa-erreur, les auteurs concluent que bien qu'Adaboost produise généralement des ensembles plus divers, les performances obtenues par l'algorithme Rotation Forest sont dues en grande partie à la précision individuelle des arbres de décisions obtenus. Cette constatation renforce l'idée intuitive que la diversité n'est pas le seul facteur déterminant la précision d'un ensemble. Ces algorithmes dérivant du Bagging sont considérés comme des méthodes implicites, car les modifications effectuées au niveau des données sont effectuées de manière aléatoire, sans évaluation d'une métrique quelconque.

3.4.3 L'apprentissage par corrélation négative

Dans le cadre de la construction d'ensembles de réseaux de neurones, Liu et Yao [129, 128, 133] proposent une méthode dite d'apprentissage par corrélation négative (*NC-Learning*) [131, 130]. Le pseudo-code correspondant est donné dans l'algorithme 3.

Utilisé dans un contexte de régression, cet algorithme peut en théorie être appliqué à tout type de classifieur permettant de minimiser une fonction d'erreur. Le principe de l'algorithme consiste à introduire un terme de régularisation dans la fonction d'erreur calculée pour chaque réseau de manière à contrôler la diversité durant l'apprentissage en parallèle des éléments de l'ensemble. Une étude approfondie du rôle joué par la diversité lors de l'apprentissage par corrélation négative est proposée dans [20]. Pour des problèmes de régression, l'auteur montre que ce type d'algorithme permet de contrôler de manière explicite la diversité en influant sur le terme relatif à la covariance dans la décomposition de l'erreur. Cependant, pour des problèmes de classification, le lien entre

Algorithme 3 L'algorithme par corrélation négative NC-Learning

entrée :

un ensemble d'apprentissage $Z = \{(X_1, Y_1), \ldots, (X_N, Y_N)\}$ avec $X_i \in \Re^n$ et $Y_i \in \{-1,\ +1\}$
un ensemble de L réseaux de neurones dont les poids ont été initialisés

sortie :

un ensemble de L réseaux de neurones entrainés

pour chaque exemple d'apprentissage $n = 1, \ldots, N$ **faire**
- Calculer la sortie moyenne $\overline{f} = \frac{1}{L}\sum_{i=1}^{L} f_i(X_n)$

pour chaque réseau de neurones $l = 1, \ldots, L$ **faire**
- calculer l'erreur de la fonction de sortie $e_i = \frac{1}{2}\left(f_i(X_n) - Y_n\right)^2 - \lambda\left(f_i(X_n) - \overline{f}\right)^2$
- mettre à jour les poids selon $\frac{\partial e_i}{\partial f_i} = (f_i(X_n) - Y_n) - 2\lambda\frac{L-1}{L}\left(f_i(X_n) - \overline{f}\right)$

fin pour
fin pour
retourner l'ensemble de L réseaux de neurones entrainés

la diversité et l'algorithme *NC-Learning* est encore mal défini.

3.4.4 l'algorithme DECORATE

L'algorithme DECORATE (*Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples*) proposé par Melville et Mooney [142] se distingue des autres algorithmes ensemblistes précédemment décrits. En effet, ce méta-algorithme d'apprentissage est l'une des rares approches proposant une évaluation et une utilisation explicite de la diversité à chaque étape de création de l'ensemble de classifieurs. L'architecture globale de DECORATE est proche de celle d'Adaboost, où l'ensemble de classifieurs est construit de manière incrémentale en modifiant l'ensemble d'apprentissage à chaque itération. DECORATE se distingue cependant d'Adaboost au niveau des modifications apportées aux données d'entrainement de chaque classifieur.

A chaque itération, un classifieur est entrainé sur l'ensemble d'apprentissage d'origine, auquel sont ajoutées des données artificielles, appelées données diversifiées. Elles sont constituées d'exemples générés à partir de la distribution du problème mais étiquetés de manière à ce que les différences avec les prédictions de l'ensemble courant soient maximales. Le nombre d'exemples artificiels devant être générés à chaque itération est un paramètre de l'algorithme. Un nouveau classifieur est entrainé sur les données d'origine et ces données diversifiés, avec l'hypothèse selon laquelle ce classifieur augmente la diversité globale de l'ensemblede classifieurs courant. Pour maintenir un niveau de précision raisonnable, seuls les classifieurs qui augmentent les facultés prédictives de l'ensemble de classifieurs courant sont ajoutés. Ce procédé est répété jusqu'à ce que l'ensemble comporte le nombre d'éléments désiré ou lorsque qu'un nombre d'itérations maximal est atteint. Le risque est qu'il y ait trop de rejets, ce qui entraine la construction d'un ensemble de taille inférieure à celle initialement souhaitée.

La méthode employée pour étiqueter un vecteur X est la suivante : chaque élément C_i de l'ensemble de classifieurs définit un ensemble de probabilités $\widehat{P}_{C_{i,\omega_j}}(X)$ dénotant le degré d'appartenance de X à chacune des classes ω_j. Ces probabilités d'appartenance de X aux différentes classes sont calculées de manière globale pour l'ensemble :

$$\widehat{P}_{w_j}(X) = \frac{\sum_{C_i \in E} \widehat{P}_{C_{i,\omega_j}}(X)}{|E|} \tag{3.44}$$

et le vecteur X se voit attribuer la classe dont le degré d'appartenance est maximal :

$$\omega^* = max_{w_j \in \Omega} \ \widehat{P}_{w_j}(X) \tag{3.45}$$

La diversité de l'ensemble est maintenue lors de l'étiquetage des R exemples artificiels. Notons $\widehat{P}_i(X_r)$ la probabilité d'appartenance d'un exemple artificiel X_r aux différentes classes du problème déterminées par l'ensemble à l'itération t. Pour maximiser la diversité de l'ensemble de classifieurs, cet exemple artificiel sera étiqueté de sorte que les probabilités d'appartenance de cet exemple aux différentes classes soient inversement proportionnelles aux prédictions de l'ensemble courant E_t. L'étiquette de l'exemple artificiel X_r sera choisie selon des fonctions d'appartenance $\widehat{P'}_i(X_r)$ définies par :

$$\widehat{P'}_i(X_r) = \frac{1/\widehat{P}_i(X_r)}{\sum_{i=1}^{c} 1/\widehat{P}_i(X_r)} \tag{3.46}$$

Les approches ensemblistes basées sur une redistribution ou un rééchantillonnage des exemples limitent la quantité de diversité exploitable au cours de la création de l'ensemble. Melville et Mooney s'appuient sur cet argument pour valider les performances obtenues, en montrant que leur approche présente une meilleure erreur en généralisation lorsque l'ensemble d'apprentissage initial est de faible dimension. Les auteurs soulignent également qu'à la différence d'Adaboost, qui requiert l'utilisation d'un algorithme d'apprentissage faible, DECORATE nécessite un classifieur de base performant afin d'éviter que l'ajout des exemples artificiels ne perturbe l'apprentissage des éléments de l'ensemble.

3.5 Conclusion

Dans ce chapitre, nous avons établi un état de l'art non exhaustif des différentes méthodes qui abordent le problème de l'Apprentissage Automatique selon une approche ensembliste. Nous distinguons les ensembles de classifieurs binaires ayant pour objectif de décomposer un problème complexe des ensembles constitués de prédicteurs redondants entrainés à résoudre un même problème. Nous avons présenté différents algorithmes ensemblistes en nous appuyant sur le cadre théorique de la décomposition biais-variance-bruit de l'erreur d'un classifieur. Il apparait que s'il est possible d'expliquer dans quelle mesure l'utilisation d'un ensemble de prédicteurs contribue à une réduction de l'erreur dans le cadre de la régression et de l'approximation de fonction, cette justification est plus difficile à donner lorsque l'on s'intéresse à des problèmes de classification. La manière dont certains algorithmes, tel Adaboost, se situent dans le cadre de la décompostion de l'erreur est également sujet à de nombreuses discussions. Nous avons développé le concept de diversité dans un ensemble de classifieurs, et il apparait qu'à l'heure actuelle peu de méthodes proposent d'évaluer la diversité et de l'utiliser de manière explicite durant l'apprentissage et la construction de l'ensemble. Les mécanismes mis en jeu et l'influence de la diversité sur les performances d'un ensemble de classifieurs sont encore mal connus. Il semble y avoir un compromis à trouver entre la diversité entre classifieurs et leur précision individuelle. Les recherches décrites dans [179] montrent que la diversité n'est pas le seul élément à prendre en compte pour construire un ensemble performant. Il est donc naturel de chercher à approfondir l'étude et la compréhension de ce compromis entre précision et diversité. Nous allons maintenant tenter d'apporter des éléments de réponse au cours du chapitre suivant, en proposant un algorithme qui utilise de manière constructive la diversité durant le processus de création d'un ensemble de classifieurs.

4

Sur l'influence de la diversité pour la construction d'ensembles de classifieurs

Dans le chapitre précédent, nous avons présenté l'approche ensembliste du problème de la classification vers laquelle tend à s'orienter la recherche en Apprentissage Automatique. Au terme de cet état de l'art, nous nous trouvons face à un paradoxe. D'un point de vue empirique et expérimental, la supériorité de l'approche ensembliste comparativement à l'utilisation d'un classifieur unique n'est plus à démontrer en terme de réduction de l'erreur en généralisation. Cependant, la justification de ce succès fait encore l'objet, à l'heure actuelle, de nombreuses questions dont les réponses sont parfois contradictoires [112]. Dans le cas de la régression, on dispose de fondements théoriques et de preuves formelles permettant d'exprimer le lien entre la diversité et la précision d'un ensemble. Pour des problèmes de classification, en revanche, l'existence d'un tel lien est encore actuellement beaucoup plus intuitive. Pour illustrer l'existence de ce lien, nous comparons dans la figure 4.1 les diagrammes kappa-erreur [138] décrits dans le chapitre précédent et obtenus sur deux benchmarks provenant de l'UCI Repository [3] fréquemment utilisés dans la littérature. Pour chacun de ces benchmarks, nous comparons les nuages de points correspondant à un ensemble de 40 classifieurs construits en utilisant Adaboost.M1 (en noir) et l'implémentation usuelle du Bagging (en rouge). Rappelons que les points représentant les couples de classifieurs ayant la plus grande diversité sont situés à gauche du diagramme et les couples de classifieurs possédant la plus grande précision sont situés dans la partie inférieure du diagramme.

Le nuage de points correspondant au Bagging (en rouge) possède un aspect compact. Les classifieurs obtenus possèdent une précision individuelle importante mais la diversité de l'ensemble est extrêmement réduite. A l'inverse, le nuage de points obtenu par Boosting possède une surface beaucoup plus étendue. La diversité de l'ensemble résultant est beaucoup plus importante mais l'erreur moyenne de chaque paire de classifieurs est plus élevée. De plus, le diagramme correspondant à l'algorithme Adaboost.M1 met en évidence l'existence du dilemme entre diversité et précision. Le nuage possède une forme diagonale, et les paires de classifieurs possédant la plus grande diversité sont généralement les plus imprécises. Nous donnons dans la table 4.1 la précision obtenue par vote majoritaire (pondéré par les α_i dans le cas d'Adaboost.M1) par ces deux ensembles de classifieurs pour chacun des benchmarks utilisés. La précision est évaluée sur 5 validations croisées en 10 blocs et les valeurs données correspondent à la précision moyenne calculée sur les 50 valeurs obtenues.

Sur les deux benchmarks considérés dans cet exemple, il apparait que l'ensemble obtenu par application de l'algorithme Adaboost.M1 présente des facultés prédictives supérieures à celle du Bagging (de l'ordre de 1,5% d'erreur), ce qui semble montrer que la diversité joue un rôle

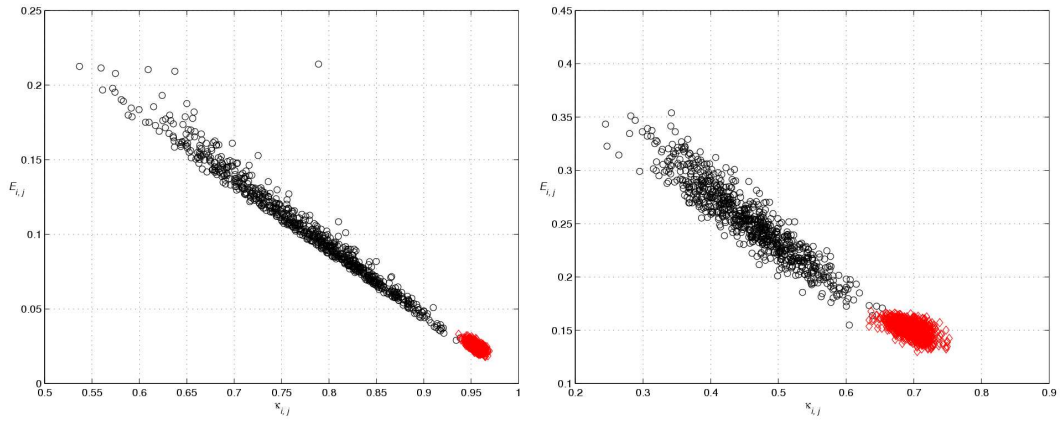

FIG. 4.1 – Comparaison des diagrammes kappa-erreur correspondant à deux ensembles de classifieurs obtenus par Boosting (en noir) et par Bagging (en rouge). Les figures correspondent respectivement aux diagrammes obtenus sur les benchmark *segment* (à gauche) et *vehicle* (à droite) de l'UCI Repository.

TAB. 4.1 – Précision obtenue par validation croisée sur les benchmarks *segment* et *vehicle* de l'UCI Repository.

	Bagging	Boosting
segment	96.76 ± 1.02	98.16 ± 0.70
vehicle	72.93 ± 4.19	74.43 ± 5.07

important dans la construction d'un ensemble de classifieurs. Cependant, les travaux menés par Rodriguez *et al.* [179] portant sur l'étude de l'algorithme Rotation Forest suggèrent que la diversité n'est pas le facteur le plus important sur la réduction de l'erreur en généralisation d'un ensemble de classifieurs. Les ensembles obtenus à l'aide de cet algorithme présentent en général une diversité moindre comparativement à Adaboost, mais la précision globale du diagramme kappa-erreur correspondant est plus importante. De manière générale, l'algorithme Rotation Forest possède un comportement proche du Bagging, donnant lieu à des ensembles légèrement plus divers et plus précis que ce dernier. Certaines interrogations subsistent donc concernant le dilemme précision-diversité.

Dans ce chapitre, nous cherchons à clarifier le lien existant entre la précision et la diversité d'un ensemble de classifieurs, en abordant sa construction selon une approche évolutionniste. Plus précisément, nous abordons la construction d'ensembles de classifieurs selon le paradigme *Surproduction et Sélection* (*overproduce and choose*) [138, 180] qui consiste à entraîner un grand nombre de classifieurs et à contruire l'ensemble en sélectionnant ceux qui favorisent la précision, la diversité, ou toute combinaison de ces paramètres. Nous commençons par décrire le principe général des méthodes évolutionnistes et plus particulièrement des algorithmes génétiques, de manière à établir le lien avec l'approche que nous proposons dans la cadre de la construction d'ensembles de classifieurs. Nous introduisons également différentes heuristiques proposées dans la littérature permettant d'illustrer le paradigme *Surproduction et Sélection*, avant de présenter notre approche qui consiste à explorer l'espace des sous-ensembles de classifieurs à l'aide d'un algorithme génétique. Nous terminons ce chapitre par l'étude de notre approche et l'analyse des

résultats obtenus sur différents benchmarks de l'UCI Repository de manière à apporter certaines réponses quant au rôle tenu par la diversité dans la construction d'ensembles de classifieurs.

4.1 Classification et Algorithmes Génétiques

Les algorithmes génétiques [92, 76] désignent un ensemble de méthodes heuristiques appartenant à la famille des algorithmes évolutionnistes, qui s'inspirent de la théorie de l'évolution pour résoudre des problèmes d'optimisation. Ces algorithmes font évoluer une population de solutions, représentées par un ensemble de *gènes*, en utilisant des mécanismes inspirés de la biologie du vivant et jusqu'à l'obtention d'une solution satisfaisante. Le principe sur lequel repose ce type d'algorithme est que l'héritage des gènes permettant une meilleure adaptation devrait conduire à une évolution progressive de la population vers de bonnes solutions. Les mécanismes du vivant mis en jeu sont généralement simplifiés. Par exemple, la notion de dominance et de récessivité des gènes n'est en général pas prise en compte dans la représentation des solutions, qui sont le plus souvent codées de manière binaire.

Les algorithmes génétiques sont utilisés pour obtenir une solution, non nécessairement optimale, à un problème donné lorsque l'on ne dispose pas d'une méthode déterministe permettant de résoudre ce problème, ou lorsque l'espace des solutions possibles est trop important pour être parcouru de manière exhaustive. Un algorithme génétique comprend en général 5 étapes : l'initialisation de la population, l'évaluation des individus, la sélection, la reproduction et le remplacement de la population.

Initialisation Cette étape consiste à définir une population de solutions initiales. Dans le cas des algorithmes génétiques, on parle indifféremment de solutions, d'individus ou de chromosomes. Chaque solution est codée sous la forme d'un ensemble de gènes. Il n'existe pas de méthode prédéfinie permettant de réaliser le codage de chaque solution. Dans le cas d'un problème de *sac à dos*, par exemple, qui constitue un problème d'optimisation classique, chaque solution peut être codée de manière binaire. Dans ce type de problème, on dispose d'un ensemble d'objets, caractérisés par une valeur appelée profit. L'objectif et de sélectionner un sous-ensemble d'objets permettant de maximiser le profit total sans excéder la capacité du sac, celui-ci ne pouvant contenir qu'un nombre limité d'objets. Chaque solution est représentée par une chaine binaire puisque chaque objet est soit retenu, soit écarté du sac (on considère ici le cas où chaque objet n'est disponible qu'en un seul exemplaire). Le nombre de bits dans le génome de chaque solution correspond au nombre d'objets disponibles. La population initiale, dans ce cas, correspond à un ensemble de chaines binaires obtenues aléatoirement.

Evaluation Pour évaluer chaque individu présent dans la population, on associe à chaque solution une valeur appelée *fitness* reflétant son niveau de satisfaction pour le problème considéré. La définition de la fonction de fitness dépend du problème posé et du choix du concepteur. Cependant, la valeur de cette fonction doit dépendre directement du génome de chaque individu. Si l'on reprend le problème précédent du sac à dos, le fitness de chaque solution correspond à la somme des profits associés à chaque objet retenu. Les solutions possédant un degré de satisfaction important sont celles pour lesquelles ce fitness est élevé.

Sélection Au cours de cette étape, les individus les mieux adaptés au problème posé sont sélectionnés pour engendrer la génération de solutions suivante. Il existe plusieurs méthodes de sélection utilisables dans le cadre des algorithmes génétiques. La plus répandue consiste à associer à chaque solution une probabilité de sélection proportionnelle à son degré de

satisfaction, donné par la fonction de fitness. Ce principe est parfois désigné sous le nom de *roue de la fortune biaisée* en raison de son mode de fonctionnement. On effectue plusieurs tirages au sort homogènes sur une roue où la surface correspondant à chaque solution est proportionnelle à son niveau d'adaptation. Remarquons qu'avec une telle méthode, les solutions possédant un fitness important peuvent être sélectionnées plusieurs fois. La sélection proportionnelle à l'adaptation est la méthode qui correspond le mieux à l'évolution du vivant. Le principe sur lequel elle repose est que les individus possèdant les gènes les mieux adaptés auront davantage de chances de les transmettre aux générations suivantes, tout en autorisant un aspect aléatoire dans la sélection. La sélection par rang, au contraire, ne tient pas compte du hasard et sélectionne systématiquement les individus ayant une adaptation maximale. A l'inverse, la sélection uniforme attribue à chaque individu une probabilité de sélection identique, indépendamment de son fitness.

Reproduction Les solutions retenues durant l'étape de sélection sont croisées de manière à générer une nouvelle population, qui hérite des gènes favorisant l'adaptation au problème posé. La reproduction fait intervenir deux opérateurs : Le croisement et la mutation. Le croisement, ou *cross-over*, consiste à former de nouvelles solutions en échangeant des portions de génomes entre les individus. Dans le cas d'un cross-over simple, les chromosomes représentant deux solutions sont scindés en un même point choisi aléatoirement et les parties ainsi obtenues sont échangées pour former deux nouvelles solutions. Lorsque les génomes des différents individus sont fragmentés en plus de deux parties, on parle de cross-over multiple. Le processus de mutation intervient généralement à la suite des croisements entre solutions. Les individus de la population obtenue à l'issue des différents croisements peuvent subir des mutations aléatoires au niveau de leur génome. Dans le cas d'un codage sous forme de chaîne binaire, chaque bit ayant une valeur égale à 1 est susceptible d'être remplacé par un 0, et inversement. La probabilité qu'a une valeur de changer spontanément (encore appelée taux de mutation) est un paramètre de l'algorithme. Ces mutations permettent d'éviter une convergence prématurée de l'algorithme génétique vers un optimum local. A l'inverse, un taux de mutation trop important risque d'entraîner une instabilité de l'algorithme et d'empécher sa convergence.

Remplacement La nouvelle population obtenue à l'issue de la reproduction succède aux solutions dites *parents*. Ce remplacement peut être mis en place de différentes façons. Dans le cas où la taille de la population reste constante durant le processus d'évolution (chaque paire de solutions parents produit exactement deux descendants), la population est entièrement remplacée par sa descendance qui est globalement constituée de solutions plus proches de l'optimal. Il est également possible de pratiquer une politique élitiste de manière à conserver le ou les individus possédant la plus grande adaptation lors du remplacement de la population.

Les étapes d'évaluation, sélection, reproduction et remplacement sont répétées jusqu'à ce qu'une condition d'arrêt donnée soit satisfaite. Par exemple, il est possible d'arrêter le processus d'évolution lorsqu'un nombre maximum d'itérations est atteint (on parle aussi de générations), où lorsque l'accroissement du fitness au cours de plusieurs itérations successives est inférieur à un certain seuil.

Le fonctionnement des algorithmes génétiques est résumé dans la figure 4.2. Nous donnons ici une description générale de ce type d'algorithme. En effet, chacune des étapes précédemment décrites comporte un grand nombre de variantes, que ce soit au niveau de la représentation des solutions ou des mécanismes mis en jeu pour réaliser l'évolution. En outre, les mécanismes sur lesquels reposent les algorithmes évolutionnistes laissent une part importante au hasard et

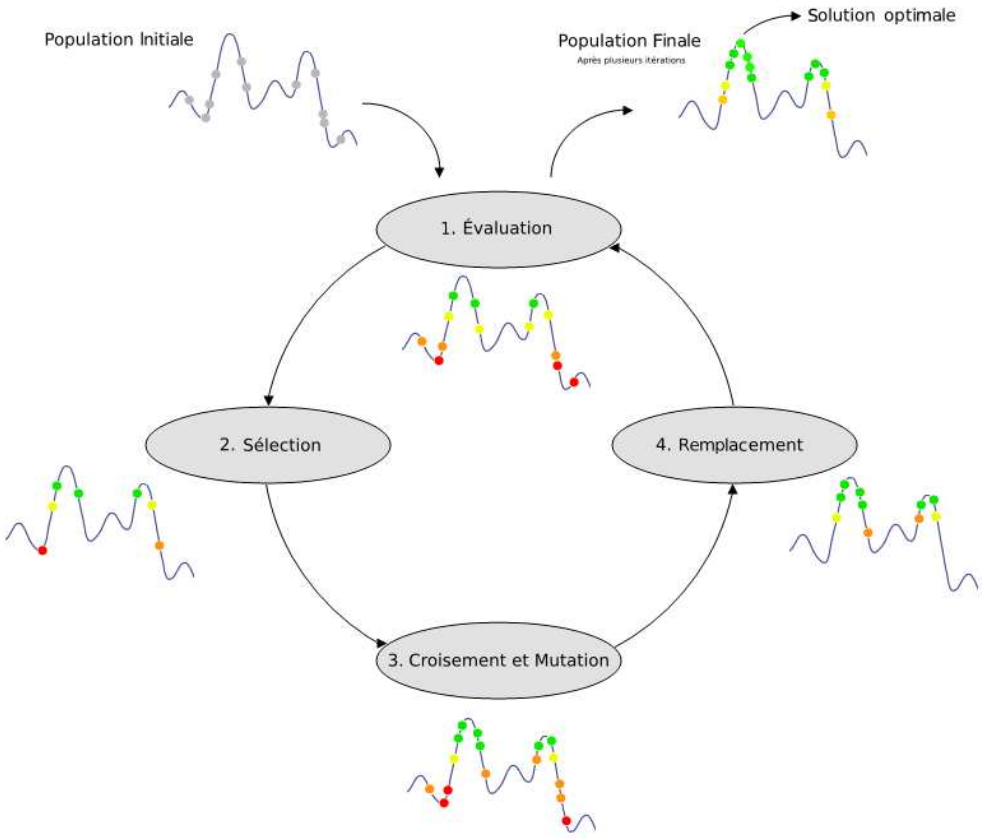

FIG. 4.2 – Principe de l'algorithme génétique. A partir d'une population initiale, des opérations de sélection, reproduction et remplacement sont répétés jusqu'à obtention d'une solution jugée optimale.

n'offrent aucune garantie concernant l'optimalité de la solution obtenue. Leur utilisation est donc conditionnée par certains facteurs, comme la dimension de l'espace des solutions à analyser.

Les algorithmes génétiques sont mis en œuvre principalement pour résoudre des problèmes d'optimisation mais on dénombre plusieurs cas d'utilisation dans le cadre de sélection de classifieurs [184, 28]. Wang et Wang [222] proposent l'utilisation d'un algorithme évolutionniste dans le cadre d'une unification entre le Bagging et le Boosting. Les auteurs suggèrent que pour ces deux algorithmes, la construction d'ensembles de classifieurs soit basée sur une exploration de l'espace des distributions de poids possibles sur les exemples d'apprentissage. Adaboost explore cet espace de manière séquentielle : la distribution utilisée à l'itération t dépend directement de la distribution et de l'erreur associée lors de l'itération $t-1$. A l'inverse, le Bagging explore de manière aveugle cet espace, en déterminant des distributions aléatoires pour entraîner chaque membre de l'ensemble. Les auteurs proposent d'utiliser un algorithme génétique pour explorer cet espace en considérant chaque chromosome comme une distribution de poids utilisée pour l'apprentissage d'un classifieur, à la manière d'Adaboost.

Dans le cadre d'ensembles de classifieurs entrainés sur différents sous ensembles d'attributs, Kuncheva et Jain [116] proposent de déterminer ces sous-ensembles à l'aide d'un algorithme génétique. Les auteurs proposent différents codages chromosomiques permettant de construire un

ensemble de classifieurs sur des sous-ensembles d'attributs disjoints ou présentant un recouvrement partiel et minimisant l'erreur en classification. Une extension de cette méthode est décrite dans [64]. Les auteurs considèrent que la construction d'ensembles de classifieurs est fonction de trois paramètres : le choix du classifieur, le choix des attributs considérés lors de l'apprentissage et le choix de la règle d'agrégation utilisée pour fusionner les décisions des membres de l'ensemble. Le codage proposé n'est donc plus basé sur une chaîne binaire, mais sur un hypercube où chaque élément correspond à un triplet *classifieur-attribut-agrégation*. L'utilisation d'un algorithme génétique permet d'explorer l'espace des combinaisons possibles entre ces trois degrés de liberté de manière à construire un bon ensemble de classifieurs sans procéder à une recherche exhaustive.

Le recours à un algorithme génétique dans le cas de la construction d'ensembles de classifieurs est justifié par la dimension importante de l'espace des solutions possibles. Roli et al. [180] proposent d'aborder ce problème par des méthodes heuristiques mais qui ne garantissent pas nécessairement l'obtention de la solution optimale. Dans le cas d'une heuristique basée sur une approche évolutionniste, le choix du critère d'adaptation utilisé permet d'explorer l'espace des solutions en favorisant certains paramètres. Cet aspect modulaire constitue la principale motivation de l'approche que nous présentons maintenant.

4.2 Description de l'approche proposée

4.2.1 Choix d'utilisation de la diversité

Kuncheva [112] distingue quatre utilisations possibles de la diversité pour étudier les ensembles de classifieurs :

- La diversité utilisée en tant qu'outil théorique.
- La diversité utilisée en tant qu'outil de visualisation. Il est possible d'utiliser la diversité comme un outil de visualisation permettant d'étudier la relation entre la diversité et la précision d'un ensemble de classifieurs. Les diagrammes kappa-erreur proposés par [138] et représentés dans la figure 3.9 permettent de comparer l'influence de la diversité pour différents algorithmes ensemblistes. Remarquons que dans ce cas, la diversité est évaluée *a posteriori* sur les exemples de test.
- La diversité utilisée en tant que critère de sélection pour construire un ensemble de classifieurs. Cette méthode est également appelée *overproduce and select*. Elle consiste à entraîner un pool de classifieurs, et à sélectionner ceux qui maximisent la diversité et/ou la précision de l'ensemble. Margineantu et Dietterich [138] proposent de calculer la diversité pour toutes les paires de classifieurs possibles dans l'ensemble et d'ajouter successivement les paires dont la diversité est maximale jusqu'à atteindre la taille désirée pour l'ensemble. Cette méthode est appelée *Kappa Pruning*. Une méthode basée sur le clustering des classifieurs de base est également proposée dans [180]. Une approche graphique, proposée dans [138], consiste à utiliser l'enveloppe convexe du diagramma kappa-erreur pour sélectionner les paires de classifieurs maximisant à la fois la précision et la diversité. Une approche similaire consistant à remplacer l'enveloppe convexe par le front de Pareto est décrite dans [112]. On désigne également cette forme d'utilisation de la diversité par le terme *ensemble pruning* [138, 212]. Dans ce cas, la diversité est évaluée au cours de la construction de l'ensemble, sur les exemples d'apprentissage.
- La diversité utilisée directement pour l'apprentissage des classifieurs de base de l'ensemble. Cette méthode est la moins explorée à ce jour. Les algorithmes les plus connus appartenant à cette catégorie sont l'algorithme NC-Learning [19, 130, 131] et l'algorithme DECORATE [142] présentés dans le chapitre précédent, et dans une moindre mesure Adaboost

et ses dérivés, d'une manière que nous qualifions de *pseudo-explicite*.

4.2.2 Construction d'ensembles de classifieurs selon l'approche Surproduction et Sélection

L'approche que nous décrivons à présent est de type *Surproduction et Sélection* [138, 180]. Cette approche s'organise en deux grandes phases. La première phase, dite de *surproduction*, consiste à entraîner un grand nombre de classifieurs. Le seconde étape consiste à sélectionner parmi cet ensemble de grande taille appelé *pool de classifieurs* un sous-ensemble permettant d'obtenir une bonne précision. On espère ainsi réduire la taille du sous-ensemble de classifieurs tout en conservant une précision proche de celle de l'ensemble initial, voir plus importante dans certains cas. Margineantu et Dietterich [138] désignent également cette approche sous le terme *ensemble pruning* dans le cas de la sélection d'un sous ensemble de classifieurs construits en utilisant Adaboost. Roli *et al.* [180] suggèrent cependant que tout algorithme ensembliste est utilisable durant la phase de surproduction.

L'étape de sélection met en jeu différentes stratégies de recherche pour explorer l'espace des sous-ensembles de classifieurs possibles. Il serait naturel de procéder à une énumération exhaustive de toutes les solutions candidates, cependant, pour un nombre L de classifieurs obtenus à l'étape de surproduction, le nombre de sous-ensembles possibles est $\sum_{i=1}^{L} \binom{L}{i}$. L'approche énumérative n'est donc pas applicable en pratique du fait du grand nombre de solutions à traiter. Pourtant, Sharkey *et al.* [202] proposent un algorithme exact utilisable uniquement lorsque le nombre de solutions possibles est réduit. De manière à surmonter le problème de la dimension importante de l'espace des sous-ensembles de classifieurs possibles, différentes heuristiques de sélection sont proposées dans la littérature. Partridge et Yates [158] proposent une heuristique gloutonne permettant de contrôler le nombre de classifieurs dans le sous-ensemble solution en sélectionnant les classifieurs ayant la plus grande précision. Cette sélection ne prend pas en compte la diversité présente entre les éléments sélectionnés. A l'inverse, Margineantu et Dietterich [138] proposent d'utiliser le kappa comme critère de sélection. Cette heuristique, appelée *kappa pruning*, consiste à calculer la diversité entre chaque paire de classifieurs de l'ensemble de départ, et à ajouter de manière successive les paires de classifieurs par ordre de diversité décroissante, jusqu'à ce que le nombre d'éléments désiré soit atteint. Ces méthodes ne tiennent pas compte du lien existant entre la précision individuelle des classifieurs et leur diversité. Les heuristiques basées sur l'enveloppe convexe et le front de Pareto du diagramme kappa-erreur proposent de prendre en compte simultanément la précision et la diversité de l'ensemble, mais ne permettent pas de contrôler le nombre de classifieurs dans le sous-ensemble construit. Différentes heuristiques basées sur l'utilisation de la diversité de manière ascendante ou descendante sont également décrites dans [180].

Dans le cadre des travaux présentés dans ce mémoire, nous proposons l'utilisation d'un algorithme génétique pour réaliser l'étape de sélection dans le cadre de la construction d'ensembles de classifieurs selon une approche de type *overproduce and select*. Le choix de cette approche pour la construction de classifieurs repose en partie sur les résultats présentés dans [138] qui montrent que pour des applications pratiques il peut être pénalisant d'utiliser un ensemble comportant un grand nombre de classifieurs et qu'il est alors intéressant de réduire le nombre de classifieurs utilisés tout en conservant des performances proches de celles de l'ensemble d'origine. Nous cherchons à tirer partie de la flexibilité des algorithmes génétiques pour étudier le dilemme précision-diversité lors de l'exploration de l'espace des sous ensembles candidats. Durant la phase de surproduction, nous utilisons l'algorithme adaboost.M1 pour construire un ensemble d'arbres de décisions. Si l'on se réfère aux diagrammes kappa-erreur présentés dans la figure 4.1, on

constate qu'il est plus intéressant d'avoir recours au Boosting pour construire l'ensemble initial, en raison de la surface plus étendue du diagramme kappa-erreur qui en résulte. Nous verrons par la suite que notre approche peut être vue comme une exploration du diagramme kappa-erreur guidée par l'importance donnée à la précision et à la diversité durant l'étape de sélection. L'utilisation d'un diagramme compact tel que celui obtenu par Bagging risque de limiter l'analyse de la diversité que nous cherchons à effectuer.

4.2.3 Description de l'algorithme génétique utilisé

Nous redéfinissons à présent chacune des étapes de l'algotihme génétique utilisé dans le cadre de la sélection d'un sous-ensemble de classifieurs.

Codage et initialisation des solutions Plusieurs solutions sont envisageables concernant le codage des solutions. La manière la plus intuitive serait de coder chaque solution par une chaîne binaire de taille égale au nombre de classifieurs dans l'ensemble de départ. Lorsqu'un gène situé à la position i a pour valeur 1, cela signifie que le i-ème classifieur dans le pool de départ est retenu dans le sous-ensemble solution correspondant. Cette notation suppose de maintenir un ordre sur les classifieurs de départ, par exemple en les numérotant suivant l'ordre dans lequel l'algorithme Adaboost.M1 construit ces classifieurs. Ce codage permet également d'éviter de sélectionner plusieurs fois un même classifieur, chaque gène correspondant à un élément différent dans l'ensemble de départ. Ce type de représentation possède cependant un inconvénient majeur car il ne permet pas de contrôler le nombre de classifieurs retenus dans chaque solution. En effet, les mécanismes de recombinaison sont indépendants des valeurs portées par chaque gène ce qui rend impossible tout contrôle sur le niveau d'élagage de l'ensemble initial. De manière similaire, il est envisageable de représenter chaque paire de classifieurs par un gène différent. Pour un ensemble de départ comportant L classifieurs, le nombre de paires distinctes est égal à $\frac{L(L-1)}{2}$. Chaque solution peut donc être représentée par une chaine binaire de cette taille, et les gènes ayant une valeur égale à 1 correspondent aux paires retenues dans la solution correspondante. Un classifieur pouvant apparaître dans plusieurs paires différentes, cette méthode limite encore davantage le contrôle sur l'élagage de l'ensemble d'origine. Le codage que nous utilisons permet de paramétrer la taille de chaque solution et ainsi étudier l'influence de la diversité sur le niveau de réduction de l'ensemble de départ. Nous adoptons une représentation à l'aide d'une chaîne non binaire de taille prédéfinie notée L^* qui représente la taille de l'ensemble de classifieurs à construire par sélection. Chaque gène porte une valeur comprise entre 1 et L correspondant au numéro du classifieur dans l'ensemble de départ. Cette représentation garantit une taille constante des solutions durant le processus d'évolution. En revanche, elle ne permet pas de gérer directement les apparitions multiples d'un même classifieur dans la construction des sous-ensembles.

La construction de la population initiale est illustrée dans la figure 4.3. Cette étape consiste à former P sous-ensembles de taille L^*, ces deux valeurs étant des paramètres de l'algorithme. Ces sous-ensembles sont construits en effectuant des tirages aléatoires sans remise pour une même solution, de manière à éviter les occurences multiples d'un même classifieur lors de l'initialisation. Nous verrons par la suite que les mécanismes relatifs au cross-over et à la mutation risquent de générer des occurences multiples d'un classifieur dans un même ensemble et nous décrirons comment les gérer.

Evaluation des solutions Le critère d'évaluation des sous-ensembles est fondamental dans

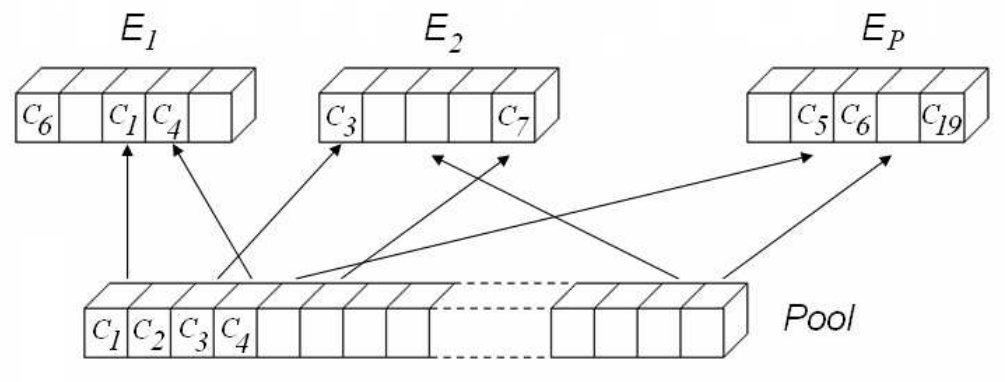

FIG. 4.3 – Initialisation d'une population d'ensembles de classifieurs, ici de taille 5, à partir d'un pool. Un même classifieur peut apparaître dans plusieurs ensembles différents mais pas dans un même ensemble.

l'approche que nous proposons. Dans le cadre de ce mémoire, nous nous focalisons sur le couple précision-diversité dans la construction d'ensembles de classifieurs. La fonction de fitness doit donc prendre en compte ces deux paramètres pendant l'évaluation de chaque solution. Cependant, la discussion que nous avons introduite au début de ce chapitre nous amène à nous intéresser à l'importance donnée à chacun de ces deux facteurs. Nous introduisons donc dans la fonction de fitness un terme, noté α, permettant de modifier l'importance donnée à la précision et à la diversité lors de l'exploration de l'espace des solutions. Le niveau d'adaptation ou fitness d'un ensemble de classifieurs E est défini dans sa forme générale par :

$$Fitness(E) = \alpha \times Prec(E) + (1 - \alpha) \times Div(E) \tag{4.1}$$

où $Prec(E)$ et $Div(E)$ désignent respectivement la précision et la diversité de l'ensemble de classifieurs E. Le terme utilisé pour caractériser la diversité de l'ensemble peut faire intervenir chacune des mesures de diversité décrites dans le chapitre précédent. Dans le cas où la mesure de désaccord $D_{i,j}$, décrite dans le chapitre précédent (page 79), est utilisée pour évaluer la diversité entre deux classifieurs, on a :

$$Div(E) = \frac{2}{L^*(L^* - 1)} \sum_{i=1}^{L^*-1} \sum_{j=i+1}^{L^*} D_{C_i,C_j} \tag{4.2}$$

où C_i et C_j désignent respectivement les classifieurs correspondant aux gènes i et j de l'ensemble E. Comme nous utilisons l'algorithme Adaboost pour construire le pool de départ, chaque hypothèse faible ou classifieur C_i est associé à un poids α_i. La précision de l'ensemble de classifieurs est évaluée en comparant la classe réelle des exemples à la classe calculée en combinant les sorties des différents classifieurs par un vote majoritaire pondéré tel qu'il est décrit dans l'algorithme 2 page 72. La précision de chaque ensemble de classifieurs est calculée sur un ensemble de validation : les données du problème sont découpées en trois parties. La première partie est utilisée pour l'apprentissage des classifieurs de l'ensemble initial, la seconde pour calculer la précision des sous-ensembles obtenus par sélection et la troisième en tant que données de test. Ce choix a été adopté pour éviter d'effectuer

une sélection des classifieurs en fonction de l'erreur apparente de l'ensemble (calculée sur les exemples d'apprentissage) et qui est fortement biaisée.

Dans la formule du fitness, une valeur de α supérieure à 0.5 privilégie la précision de l'ensemble de classifieurs obtenu. A l'inverse, pour une valeur inférieure à 0.5, c'est la diversité qui est favorisée. Le paramètre α permet donc de contrôler le compromis entre précision et diversité. Remarquons que dans le cas particulier où la diversité est calculée au moyen du kappa, la formulation du fitness donnée dans l'équation 4.1 n'est plus adaptée. En effet, deux classifieurs sont fortement divers lorsque la valeur du kappa est faible, contrairement à la mesure de désaccord par exemple, qui est égale à 1 lorsque la diversité est maximale. Dans ce cas particulier, le terme relatif à la diversité est donné par l'équation 4.3 et la formulation de la fonction d'évaluation donnée dans l'équation 4.1 reste valide.

$$Div(S) = 1 - \frac{2}{L^*(L^*-1)} \sum_{i=1}^{L^*-1} \sum_{j=i+1}^{L^*} \kappa_{S(i),S(j)} \tag{4.3}$$

Sélection des solutions La sélection des ensembles de classifieurs est effectuée par rapport à la valeur de leur fitness selon la méthode de *la roue de la fortune biaisée*. Plus un ensemble de classifieurs possède un fitness élevé plus il a de chances d'être sélectionné pour engendrer de nouveaux ensembles. Nous tenons cependant à préciser que cette méthode implique une faible part de hasard dans la sélection. Pour une valeur de α égale à 0, par exemple, le fitness devient égal à la diversité de l'ensemble de classifieurs. Notre algorithme génétique se comporte alors de manière similaire aux différentes heuristiques proposées par Roli *et al.* [180], consistant à sélectionner les classifieurs qui maximisent la diversité de l'ensemble résultat. Une différence majeure subsiste cependant dans la façon dont sont construits ces ensembles de classifieurs car la sélection basée sur *la roue de la fortune biaisée* laisse aux ensembles peu diversifiés une chance d'être sélectionnés. Remarquons également qu'en utilisant cette méthode, les ensembles de classifieurs possédant le fitness le plus élevé sont susceptibles d'être sélectionnés plusieurs fois.

Croisement et mutation Les P solutions retenues durant l'étape de sélection sont regroupées par paires, ou couples de parents. Chaque couple produit exactement deux successeurs en utilisant le mécanisme de cross-over décrit dans la figure 4.4. En adoptant le codage par une chaîne de taille prédéfinie L^* décrit précédemment, le cross-over garantit la création de nouvelles solutions de tailles identiques. Les gènes portés par les chaînes binaires des ensembles parents sont brassés aléatoirement de manière à éviter que certains classifieurs soient systématiquement privilégiés dans les différents ensembles de classifieurs manipulés.

L'appariement des solutions, préalable au cross-over, étant effectuée de manière aléatoire, il est possible qu'une paire de solutions soit constituée d'un même sous-ensemble. Dans ce cas particulier, le mécanisme de cross-over produit deux solutions identiques au sous-ensemble parent. Pour éviter de voir apparaître des doublons inutiles dans les ensembles de classifieurs successeurs, tout appariement d'un sous-ensemble de classifieurs avec lui-même est rendu impossible.

Le croisement employé s'effectue comme suit : les chromosomes correspondants aux sous-ensembles parents sélectionnés sont scindés en un même point choisi aléatoirement. Les portions ainsi obtenues sont échangées de manière à former deux nouveaux sous-ensembles héritant des gènes de leurs parents. A l'issu de ces croisements, les chromosomes de chacune des nouvelles solutions peuvent subir des mutations aléatoires.

Cette façon de faire permet de maintenir la taille de la population durant le processus

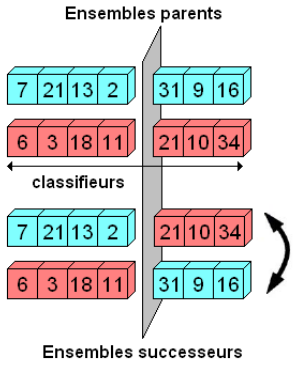

FIG. 4.4 – Mécanisme de cross-over utilisé dans le cadre d'une approche génétique pour la construction d'ensembles de classifieurs.

d'évolution et rend possible un contrôle direct sur le nombre de classifieurs de chaque sous-ensemble solution. Cependant, elle n'offre aucune garantie concernant les occurences multiples d'un classifieur dans une même solution. Nous modifions en conséquence les mécanismes relatifs à la reproduction de sorte que si un même classifieur apparaît de manière répétée dans une même solution suite à un croisement, ces occurences multiples sont remplacées par un classifieur choisi aléatoirement dans l'ensemble de départ mais n'apparaissant pas dans le codage de la solution considérée. La mutation s'effectue de manière analogue : lorsqu'un gène subit une mutation aléatoire, le classifieur correspondant à ce gène est remplacé par un élément ne figurant pas déjà dans la solution correspondante.

Remplacement de la population Sachant que la taille de la population est égale à P, le nombre de paires de solutions parents est égal à $P/2$. Chaque paire donnant naissance à exactement deux successeurs, P nouvelles solutions sont donc créées à l'issue de l'étape de reproduction de notre implémentation. Cette nouvelle population remplace entièrement les sous-ensembles de classifieurs utilisés lors de l'itération précédente. Nous appliquons cependant une politique élitiste, en conservant la meilleure solution obtenue durant le processus d'évolution. A chaque itération, préalablement à l'étape de recombinaison des ensembles de classifieurs, l'ensemble possédant le fitness le plus élevé est sauvegardé et l'ensemble de classifieurs possédant le fitness minimal dans la nouvelle population obtenue est remplacé par l'ensemble ayant le fitness le plus élevé. Cette mesure permet d'éviter une dégradation prématurée des solutions obtenues qui pourrait survenir compte tenu que plusieurs mécanismes aléatoires entrent en jeu.

Nous donnons dans l'algorithme 4 le pseudo-code de l'implémentation génétique proposée.

4.3 Etude expérimentale

Nous mettons en pratique l'algorithme décrit précédemment pour évaluer l'importance de la diversité dans la construction d'un ensemble de classifieurs, selon une approche Surproduction et Sélection. Nous comparons les résultats obtenus par notre algorithme avec ceux obtenus à l'aide de différentes heuristiques de sélection de classifieurs à partir d'un ensemble initial pour différents jeux de données provenant de l'UCI Repository [3]. De manière à pouvoir comparer équitablement les différentes heuristiques permettant d'effectuer la sélection des classifieurs et

Algorithme 4 Pseudo-code de l'algorithme GenDiv proposé

entrée :

P	: la taille de la population
L^*	: la taille désirée pour l'ensemble
$Pool$	: le pool constitué des L classifieurs construits durant la phase de surproduction
$MaxGen$	: le nombre d'itérations
α	: le coefficient utilisé dans la fonction de fitness

sortie :

E^* : L'ensemble dont le fitness est maximal sur les exemples de validation au terme de l'évolution

// **Initialisation aléatoire de la population**

pour $i = 1$ to P **faire**

- soit E_i un ensemble de L^* classifieurs distincts choisis aléatoirement dans le $Pool$ de départ
- calculer le fitness de E_i sur les exemples de validation

fin pour

// **Sélection, reproduction et mutation des différentes générations**

$iter \leftarrow 1$

pour $iter = 1$ à $MaxGen$ **faire**

- déterminer l'ensemble E^+ dont le fitness est maximal
- sélectionner P ensembles pour la reproduction en fonction de leur fitness
- grouper ces ensembles pour former une liste CP de couples de classifieurs distincts E_iE_j
- **pour chaque** (couple de classifieurs E_iE_j dans CP) **faire**
  - construire 2 nouveaux ensembles $E_{ij}^{'}$ and $E_{ji}^{'}$ par cross-overs et mutations
  - calculer le fitness de $E_{ij}^{'}$ et $E_{ji}^{'}$ de ces ensembles sur les exemples de validation
- **fin pour**
- remplacer l'ancienne population par la nouvelle
- déterminer l'ensemble E^- de la nouvelle population dont le fitness est minimal
- $E^- \leftarrow E^+$

fin pour

retourner E^* l'ensemble de classifieurs dans la dernière génération dont le fitness est maximal sur les exemples de validation

apporter des réponses concernant le rôle de la diversité durant cette étape, nous commençons par détailler le protocole utilisé pour les tests présentés dans cette section.

4.3.1 Description du protocole utilisé

Les différents jeux de données utilisés sont décrits dans la table 4.2. Ces jeux de données proviennent tous de l'UCI Repository [3] et représentent des problèmes de classification supervisée fréquemment utilisés dans la littérature du domaine.

TAB. 4.2 – Caractéristiques des différents jeux de données de l'UCI Repository utilisés.

Fichier	#classes	#exemples	#attributs	#app	#val	#test	#essais
balance-scale	3	625	4	405	110	110	30
german-credit	2	1000	20	600	200	200	30
glass	7	214	9	114	50	50	30
ionosphere	2	351	34	201	75	75	30
letter	26	20000	16	2000	500	500	10
pendigits	10	10992	16	4000	1000	1000	10
pima-diabetes	2	768	8	568	100	100	30
segment	7	2310	19	1310	500	500	30
sonar	2	208	60	108	50	50	30
spam	2	4601	57	1000	1000	1000	10
vehicle	4	846	18	546	150	150	30
waveform	3	5000	40	1000	1000	1000	10

Les trois premières colonnes de la table 4.2 désignent respectivement le nombre de classes, le nombre d'exemples disponibles et le nombre d'attributs caractérisant chaque exemple. Nous donnons ensuite la taille respective des ensembles d'apprentissage, de validation et de test que nous utilisons. Les exemples d'apprentissage sont utilisés pour construire un ensemble de 50 arbres de décision à l'aide de l'algorithme Adaboost.M1. Cette étape de construction de l'ensemble initial de classifieurs est identique pour toutes les méthodes de sélection que nous comparons dans la suite. L'ensemble de validation est utilisé pour effectuer la sélection de classifieurs à l'aide de différentes heuristiques de la littérature. Les données de test sont utilisées pour calculer la précision des ensembles de classifieurs construits par application de ces différentes heuristiques. La dernière colonne correspond aux nombres d'essais sur lesquels sont calculés les résultats présentés dans cette section pour chaque jeu de données.

Sur ces différents jeux de données, nous comparons la précision des ensembles construits par application de différentes heuristiques selon le paradigme Surproduction et Selection référencées dans la littérature. Ces heuristiques de sélection sont les suivantes :

L'heuristique *Choose Best* [158] consiste à sélectionner les classifieurs ayant la plus grande précision individuelle. Le nombre de classifieurs sélectionnés est un paramètre qu'il est possible de contrôler. Cette heuristique ne tient pas compte de la diversité pour la construction de l'ensemble de classifieurs.

L'heuristique *Kappa Pruning* [138] sélectionne les classifieurs correspondant aux paires les plus diverses (dont la valeur du Kappa est minimale). La sélection de classifieurs basée sur cette heuristique ne prend pas en compte la précision individuelle des classifieurs. Nous pouvons voir les heuristiques *Choose Best* et *Kappa Pruning* comme deux méthodes de

sélection duales, car elles ne prennent respectivement en compte que la précision ou la diversité.

L'heuristique génétique GenDiv [70] que nous proposons dans ce chapitre permet de prendre en compte à la fois la précision et la diversité de l'ensemble de classifieurs, en faisant varier l'influence de ces deux critères par modification du paramètre α dans la fonction de fitness. L'heuristique que nous proposons permet également de paramétrer le nombre de classifieurs sélectionnés. Les comparaisons effectuées dans ce chapitre sont réalisées avec une valeur de α égale à 0.8 dans la fonction de fitness de manière à privilégier la précision de l'ensemble plutôt que sa diversité, comme le suggère Kuncheva.

L'heuristique *Kappa Convex Hull Pruning* [138] est basée sur l'utilisation du diagramme Kappa-erreur dont nous donnons une illustration dans la figure 4.1 au début de ce chapitre. Cette heuristique identifie les paires de classifieurs situées sur la partie inférieure gauche de l'enveloppe convexe du diagramme et sélectionne les classifieurs constituant ces paires. La sélection de classifieurs basée sur l'enveloppe convexe du diagramme Kappa-erreur prend en compte le compromis entre la précision et la diversité des paires de classifieurs, en sélectionnant les classifieurs les plus précis et les plus divers. Il n'est pas possible, dans ce cas, de contrôler le taille de l'ensemble résultant car tous les classifieurs figurant sur l'enveloppe convexe sont ajoutés à l'ensemble.

L'heuristique *Pareto Pruning* [112] est basée sur le même principe que la sélection de classifieurs à partir de l'enveloppe convexe à laquelle sont ajoutés les paires de classifieurs *non-dominées*. Le nombre de classifieurs sélectionnés ne peut pas être fixé à l'avance. Nous donnons une illustration de l'enveloppe convexe et de l'ensemble de Pareto calculés sur le diagramme kappa-erreur du jeu de données *glass* dans la figure 4.5. Remarquons que les paires de classifieurs appartenant à l'enveloppe convexe (en rouge) sont inclues dans l'ensemble de Pareto (en bleu), la taille de l'ensemble de classifieurs obtenu par l'application de cette heuristique est donc supérieure ou égale à la taille de l'ensemble construit en utilisant l'enveloppe convexe du diagramme Kappa-erreur.

L'heuristique *Early Stopping* [138] consiste à sélectionner les L^* premiers classifieurs construits en utilisant l'algorithme Adaboost.M1. Ce nombre L^* est un paramètre de l'heuristique.

L'heuristique de sélection aléatoire (*Random Selection*) conduit à un ensemble de classifieurs de taille donnée en sélectionnant aléatoirement des éléments de l'ensemble de classifieurs initial. Nous utilisons cette sélection aléatoire de manière à situer les performances des autres heuristiques précédemment citées.

Pour comparer ces différentes heuristiques de façon pertinente, il est indispensable de se pencher sur la façon dont les données sont exploitées dans les différentes étapes de chaque méthode de sélection. C'est ce que nous détaillons maintenant.

Les heuristiques *Kappa Pruning* et *Convex Hull Pruning* proposées par Margineantu et Dietterich dans [138] sont mises en œuvre sur l'ensemble d'apprentissage. A l'inverse, l'heuristique *Reduce Error Pruning* proposée par les mêmes auteurs consiste à effectuer la sélection des membres de l'ensemble sur une autre partie des données appelée ensemble de validation ou *Pruning Set*. La comparaison des résultats obtenus par ces heuristiques s'avère délicate car c'est une partie des données d'apprentissage qui est utilisée pour constituer l'ensemble de validation. Les classifieurs de base sont donc différents puisqu'ils ne sont pas entrainés sur les mêmes exemples. La comparaison des ensembles de classifieurs obtenus peut donc être faussée. Ce risque est encore plus grand lorsque les classifieurs de base utilisés sont instables et que le nombre d'exemples disponibles est réduit.

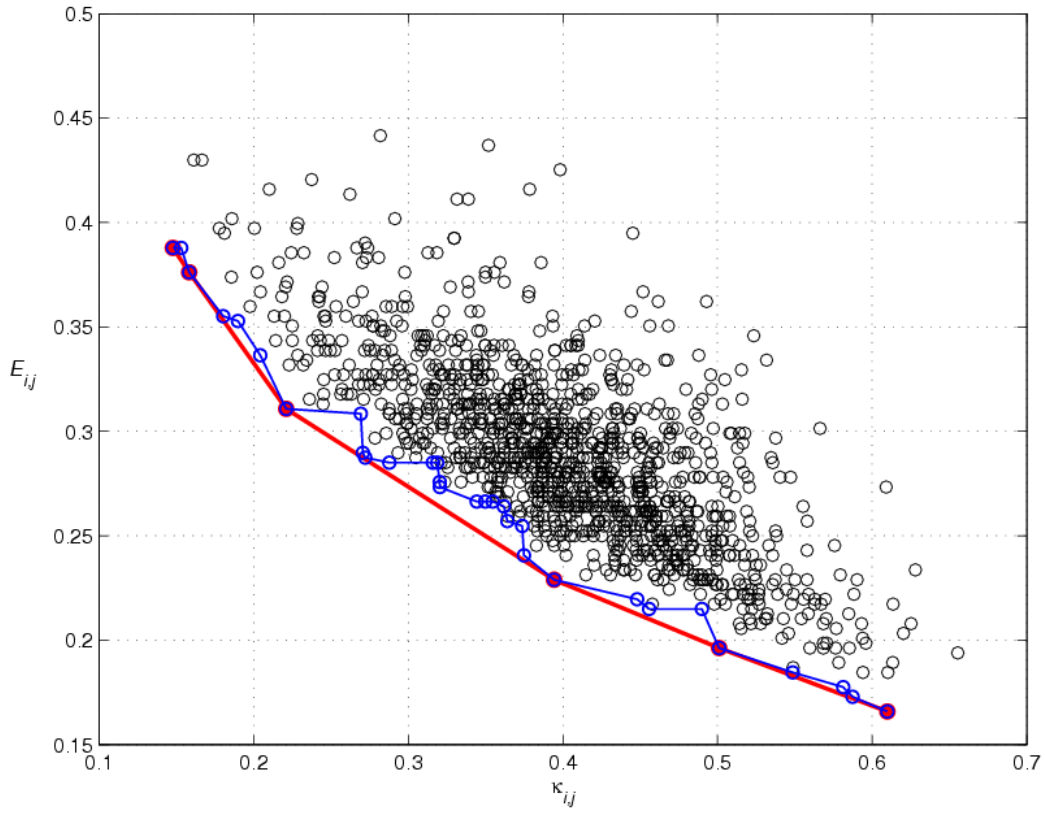

FIG. 4.5 – Diagramme kappa-erreur obtenu pour le jeu de données *glass* de l'UCI Repository. Nous représentons respectivement en rouge et en bleu les sous-ensembles de classifieurs obtenus par application des heuristiques *Convex Hull Pruning* et *Pareto Pruning*

Roli *et al.* proposent également d'utiliser différentes heuristiques de sélection sur un ensemble de validation additionnel pour éviter les problèmes de sur-apprentissage. En revanche, les heuristiques *Early Stopping* et *Random Selection* ne dépendent d'aucun critère (si ce n'est l'ordre de création des classifieurs de base pour former l'ensemble initial) et de ce fait ne nécessitent pas d'ensemble de validation.

Pour comparer efficacement les différentes heuristiques utilisées, nous cherchons à limiter au maximum les variations de précision qui pourraient être induites par des différences au niveau des données manipulées. Nous estimons donc qu'il est nécessaire de découper systématiquement les jeux de données utilisés en trois parties distinctes de manière à ce que les classifieurs de base utilisés soient les mêmes pour chaque heuristique (en d'autres termes, de manière à ce que les exemples d'apprentissage soient les mêmes quelle que soit l'heuristique employée) et également pour que les critères de sélection des différentes heuristiques utilisées soient évalués sur les mêmes données de validation.

La proportion d'exemples de validation utilisée est également sujet à discussion. Si nous utilisons un ensemble de validation de taille réduite, nous courons le risque que ces exemples ne soient pas représentatifs du problème posé. Dans ce cas, les heuristiques de sélection risquent de produire un ensemble de classifieurs en situation de sur-apprentissage de ces exemples de validation. A l'inverse, diminuer la taille de l'ensemble de test se traduit par des résultats instables

et un ensemble d'apprentissage insuffisant entraîne une diminution de la précision des classifieurs de base.

Le découpage des données constitue un réel problème en soi, c'est pourquoi nous définissons manuellement et pour chaque jeu de données utilisé une répartition des exemples en trois ensembles, respectivement pour l'apprentissage, la validation et les tests, dont la taille est donnée dans la table 4.2. Cette répartition des données est mise en œuvre de manière à éviter des perturbations éventuelles qui pourraient être générées par les différents problèmes énoncés ci-dessus.

4.3.2 Etude du rôle de la diversité dans une approche Surproduction et Sélection

De manière à étudier le comportement de l'algorithme génétique que nous proposons, nous comparons différentes heuristiques de sélection de classifieurs selon le cadre proposé par Margineantu et Dietterich [138]. Celui-ci consiste à évaluer le gain relatif obtenu par différents ensembles de classifieurs de taille donnée par rapport à l'utilisation d'un seul classifieur de même type. Nous donnons dans la figure 4.6 une illustration de cette méthode de comparaison. Nous utilisons ensuite les courbes pour présenter les résultats obtenus sur différents jeux de données de l'UCI Repository.

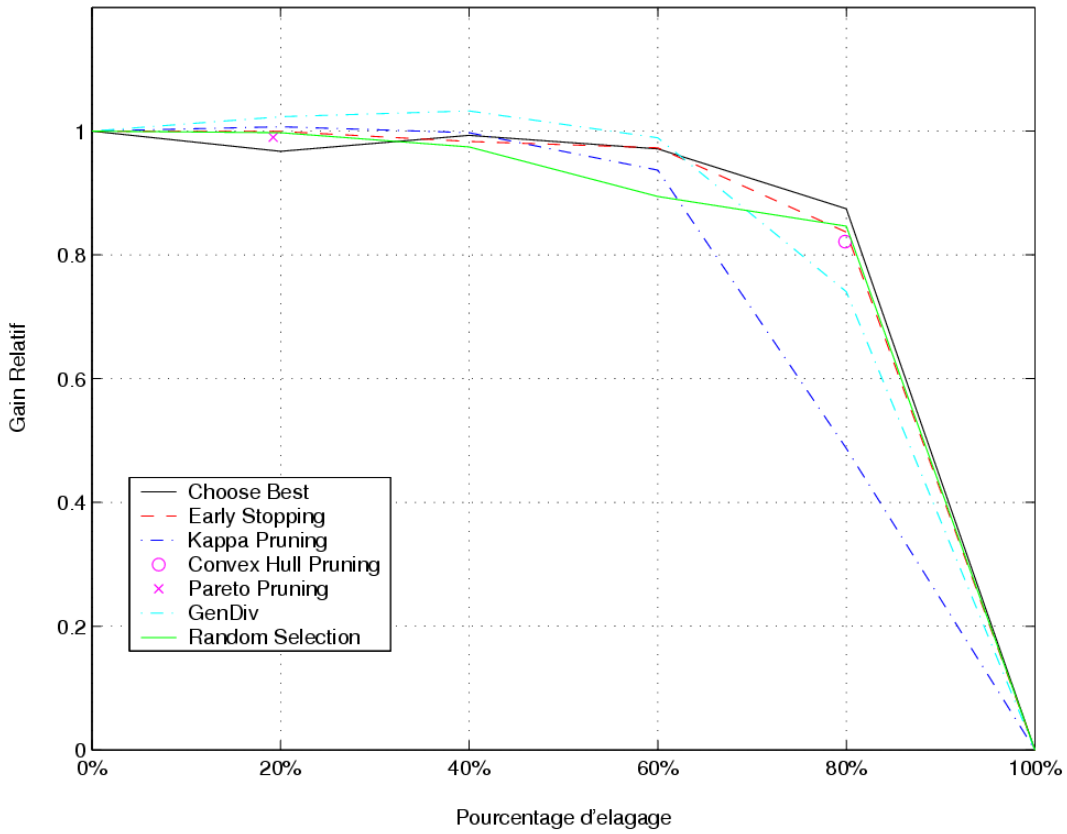

FIG. 4.6 – Illustration des courbes permettant de comparer les performances relatives des différentes heuristiques de sélection utilisées pour contruire des ensembles de classifieurs de différentes tailles. Un gain relatif supérieur à 1 signifie que l'ensemble élagué est plus précis que l'ensemble de classifieurs initial.

Les courbes présentées dans la figure 4.6 sont obtenues de la manière suivante : pour chaque jeu de données utilisé, les données sont découpées aléatoirement en trois parties dont la taille est données dans la table 4.2. L'ensemble d'apprentissage est utilisé pour construire un ensemble de 50 arbres de décision par application de l'algorithme Adaboost.M1. Nous utilisons la modification proposée par Breiman [12] consistant à réinitialiser la distribution de poids sur les exemples d'apprentissage lorsqu'un classifieur dont la précision est inférieure à 0.5 est construit. Cette modification de l'algorithme Adaboost.M1 garantit que l'ensemble de classifieurs initial est systématiquement constitué de $L = 50$ arbres de décision. Un unique arbre de décision est également construit sur la totalité des exemples d'apprentissage.

Le gain absolu correspond à la différence entre la précision de l'ensemble initial de classifieurs et un arbre de décision unique :

$$gainA = prec(Pool) - prec(A) \tag{4.4}$$

où $Pool$ désigne l'ensemble initial de classifieurs et A représente un arbre de décision entrainé sur tous les exemples d'apprentissage. Il peut arriver que le gain absolu soit négatif si l'ensemble de classifieurs initial possède une précision inférieure à celle de de l'arbre de décison unique. Dans ce cas, aucune sélection de classifieurs n'est effectuée et nous rejetons l'ensemble initial. Les données sont divisées à nouveau, et un nouvel ensemble de classifieurs est construit. Les courbes que nous présentons dans cette partie sont calculées sur la moyenne des essais pour lesquels le gain absolu est positif. Nous comptabilisons cependant le nombre de rejets survenant pour chaque jeu de données.

Lorsque le gain absolu est positif, nous élaguons l'ensemble initial de classifieurs par l'application des différentes heuristiques précédemment décrites en modifiant la taille de l'ensemble élagué lorsque l'heuristique le permet. Le ou les critères manipulés par les différentes heurisitiques de sélection de classifieurs sont systématiquement calculés sur l'ensemble de validation, sauf dans le cas des heuristiques *Early Stopping* et *Random Selection* qui ne reposent sur aucun critère particulier, et pour lesquelles l'ensemble de validation n'est pas utilisé.

La précision de l'ensemble de classifieurs obtenu par élagage est ensuite calculée sur l'ensemble de test et moyennée sur le nombre d'essais donné dans la table 4.2. Le gain relatif $gainR(E_i)$ associé à un sous-ensemble de classifieurs E_i correspond à la différente entre la précision $prec(E_i)$ du sous-ensemble de classifieurs et la précision $prec(A)$ d'un arbre de décision unique, le tout divisé par le gain absolu $gainA$:

$$gainR(E_i) = \frac{prec(E_i) - prec(A)}{gainA} \tag{4.5}$$

Un gain relatif égal à 1 indique que le sous-ensemble de classifieurs obtenu par application d'une heuristique de sélection est aussi bon que l'ensemble de classifieurs initial. Un gain relatif nul signifie que la précision du sous-ensemble de classifieurs est identique à celle d'un classifieur unique.

Remarquons qu'un gain relatif supérieur à 1 indique que l'heuristique de sélection a permis d'améliorer les performances de l'ensemble initial tout en diminuant sa taille. Ce phénomène est rarement observé et survient généralement pour un faible pourcentage d'élagage. La suppression d'un grand nombre de classifieurs entraine le plus souvent une nette diminution du gain relatif.

De manière similaire, il est possible que le gain relatif soit négatif lorsque le sous-ensemble de classifieurs obtenu par élagage est moins précis qu'un arbre de décision unique entrainé sur toutes les données d'apprentissage. Nous verrons par la suite que cette situation survient le plus souvent lorsque le critère de sélection ne prend en compte que la diversité de l'ensemble de classifieurs.

Les courbes que nous présentons dans la figure 4.6 caractérisent l'évolution moyenne du gain relatif des ensembles de classifieurs obtenus par les différentes heuristiques de sélection pour différents pourcentages d'élagage. Lorsque l'heuristique utilisée le permet, nous construisons des ensembles de classifieurs dont les tailles respectives sont égales à 40, 30, 20 et 10. Ces tailles d'ensembles correspondent à un niveau d'élagage respectivement égal à 80%, 60%, 40% et 20%. Le gain obtenu pour un elagage de 0% correspond au gain absolu obtenu en utilisant l'ensemble de classifieurs initial alors qu'un élagage de 100% est assimilé à l'utilisation de l'arbre de décision unique entrainé sur toutes les données d'apprentissage.

Nous présentons à présent les courbes obtenues suivant la cadre décrit ci-dessus pour les douze bases de données de l'UCI Repository présentées dans la table 4.2. Nous donnons également pour chaque courbe d'élagage un histogramme permettant de visualiser la précision moyenne obtenue par l'ensemble de départ et la précision d'un arbre de décision unique. Ces valeurs moyennes sont calculées sur les essais n'ayant pas engendré de rejets, c'est à dire pour lequel le gain absolu est positif. Nous reportons également dans chacun des histogrammes le nombre de rejets, en pourcentage du nombre d'essais donné dans la table 4.2, car ce nombre est un bon indicateur pour savoir s'il est pertinent d'utiliser un ensemble de classifieurs.

Nous commençons par étudier cinq jeux de données pour lesquels le nombre d'exemples est important, ce qui permet d'avoir des ensembles d'apprentissage, de validation et de test suffisamment larges pour que les courbes obtenues soient stables : ceci se traduit par un taux de rejet nul. Les figures 4.7 à 4.11 concernent respectivement les jeux de données *segment*, *letter*, *spam*, *waveform* et *pendigits*. Il est intéressant de noter que pour ces cinq bases de données, où les courbes se croisent peu, on peut établir un ordre sur les performances des différentes heuristiques utilisées. Les meilleurs résultats reviennent en général aux heuristiques *Choose Best* et *Early Stopping*, suivies par l'algorithme génétique que nous proposons. Viennent ensuite la sélection aléatoire, que nous utilisons en tant qu'heuristique de référence, qui est de manière générale meilleure que l'heuristique *Kappa Pruning* qui se base uniquement sur la diversité.

La figure 4.7 compare les performances des différentes heuristiques utilisées sur le jeu de données *segment*. Nous constatons que pour ce jeu de données, il est possible d'élaguer l'ensemble de classifieurs de départ jusqu'à 60% tout en conservant un gain relatif proche de 1. L'utilisation de l'algorithme génétique proposé permet même d'améliorer sensiblement la précision des sous-ensembles obtenus pour des valeurs d'élagage de 20% et 40 %. Pour une valeur d'élagage supérieure à 60% cependant, la précision diminue de manière plus importante si l'on cherche à maintenir trop de diversité dans les sous-ensembles de classifieurs obtenus, et une sélection aléatoire peut même se révéler plus performante qu'une heuristique dont le critère de sélection est basé sur la diversité. Pour des ensembles de classifieurs de petite taille, la précision des classifieurs individuels offre la meilleure garantie pour que l'ensemble obtenu possède également une bonne précision, comme le montre la courbe relative à l'heuristique *Choose Best*.

Une tendance similaire peut être observée pour le jeu de données *letter* pour lequel les résultats sont donnés dans la figure 4.8. Jusqu'à 60% d'élagage, l'heuristique génétique que nous proposons permet d'obtenir un ensemble sensiblement plus précis. Lorsque que la taille des sous-ensembles diminue, la précision individuelle des classifieurs reste le critère de sélection à privilégier par rapport à la diversité.

Les résultats obtenus sur le jeu de données *spam* sont représentés dans la figure 4.9. Pour ce jeu de données, l'heuristique *Choose Best* est celle qui permet d'obtenir le meilleur gain relatif de manière générale. Nous remarquons également que la diversité se révèle être un mauvais critère de sélection pour ce problème. En effet, pour un élagage peu important, les ensembles de classifieurs obtenus par l'application de l'heuristique *Kappa Pruning* (construits en sélectionnant les classifieurs les plus divers) donnent des résultats moins bons que ceux obtenus à l'aide des autres

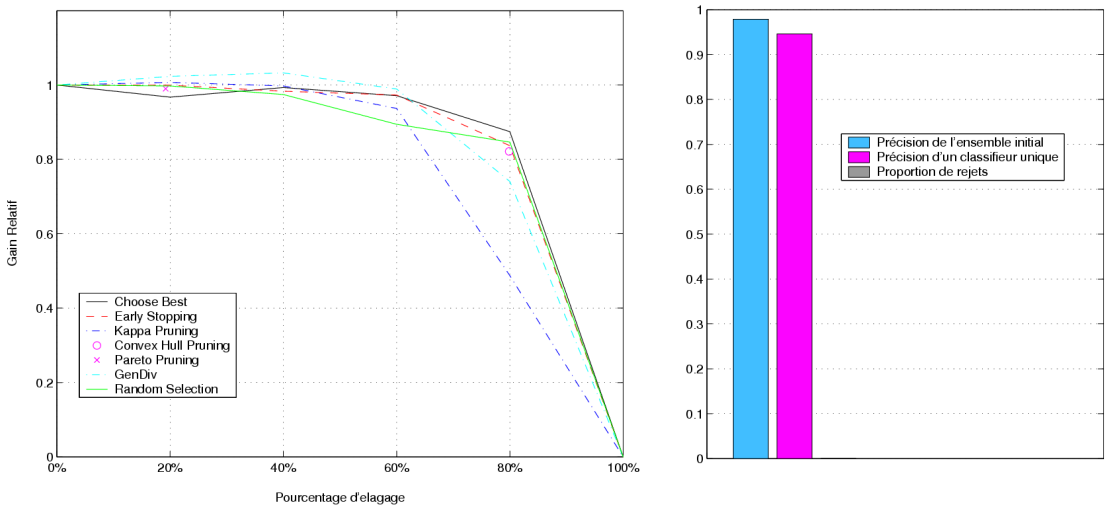

FIG. 4.7 – Résultats obtenus sur le jeu de données *segment*.

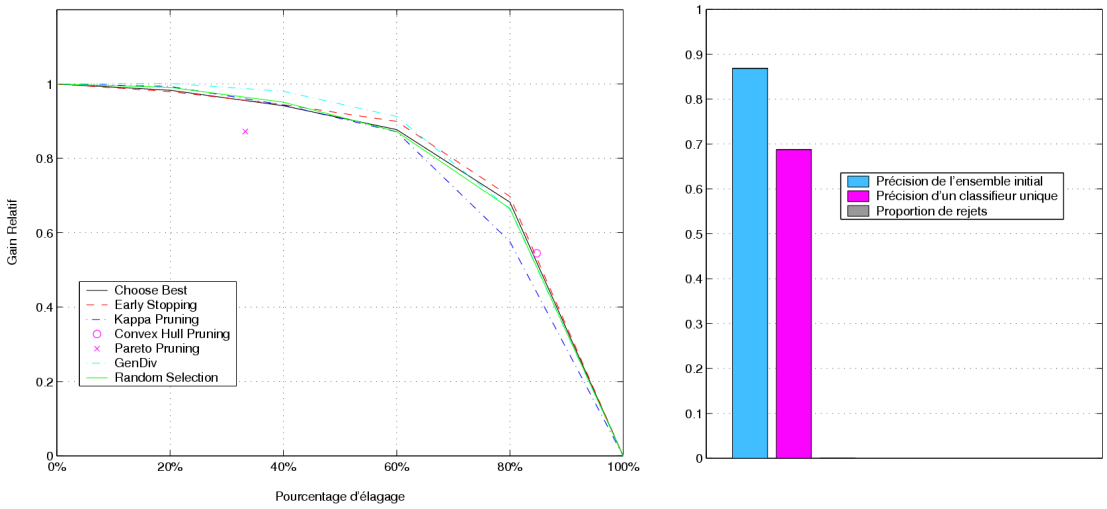

FIG. 4.8 – Résultats obtenus sur le jeu de données *letter*.

méthodes de sélection. De plus, pour un élagage important, les ensembles obtenus sont moins bons qu'un classifieur unique de même type et entrainé sur tous les exemples d'apprentissage : ce qui se traduit par un gain négatif.

Les courbes obtenues pour le jeu de données *waveform* et présentées dans la figure 4.10 illustrent fidèlement la notion d'ordre énoncée précédemment sur les performances relatives aux différentes méthodes de sélection de classifieurs. La diversité ne semble pas contribuer grandement à la précision des ensembles de classifieurs obtenus pour ce problème.

Nous constatons que pour la base *pendigits*, dont les résultats sont donnés dans la figure 4.11, il est possible de diminuer la taille de l'ensemble initial de 80% tout en conservant un gain proche de 1. Les résultats obtenus par les différentes heuristiques sont d'ailleurs très proches pour ce jeu de données.

Remarquons que pour les jeux de données que nous venons d'étudier, le nombre de rejets est nul, ce qui signifie que pour les tests effectués, l'ensemble de classifieurs initial est toujours plus

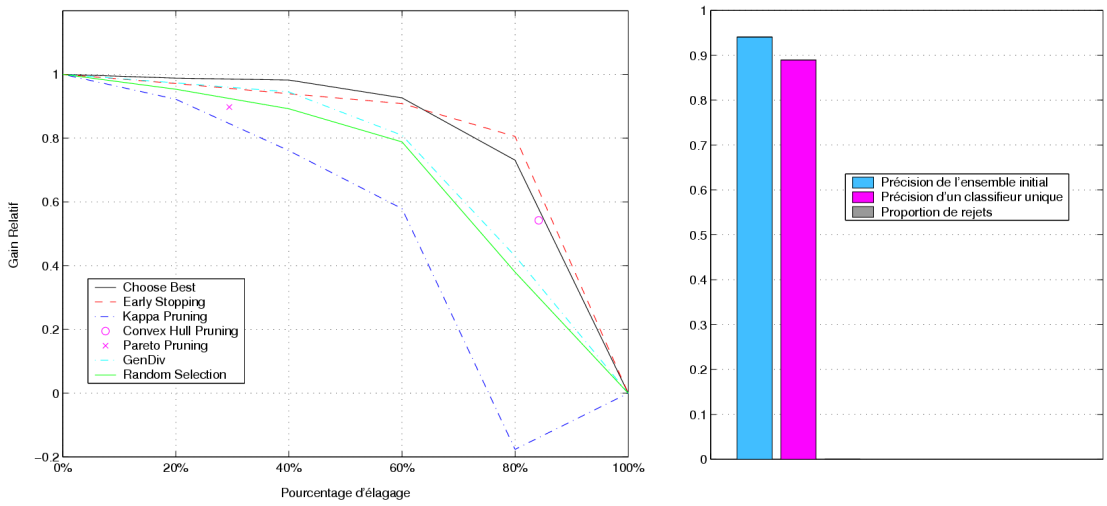

FIG. 4.9 – Résultats obtenus sur le jeu de données *spam*.

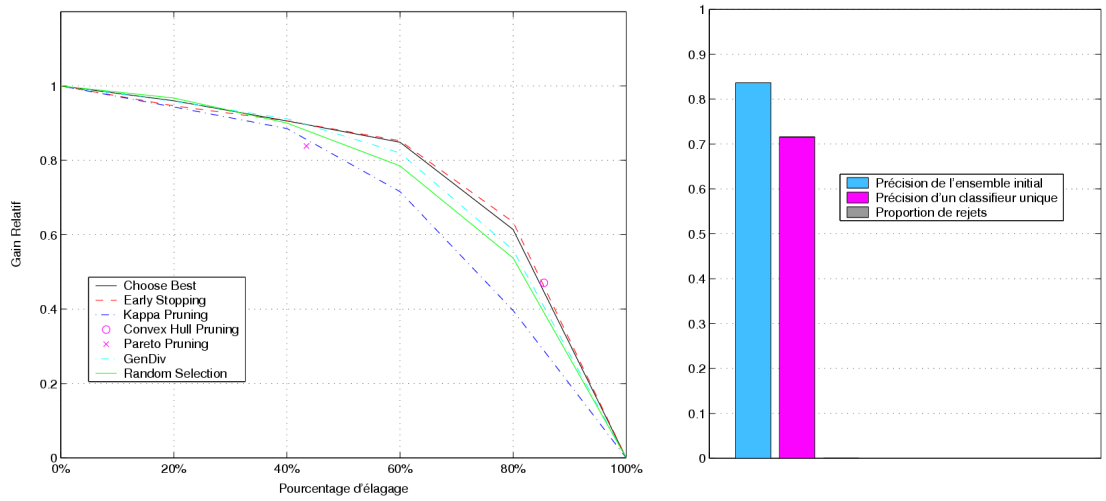

FIG. 4.10 – Résultats obtenus sur le jeu de données *waveform*.

précis qu'un classifieur unique. Il est possible d'évaluer cet écart de précision dans les différents histogrammes donnés pour chaque problème de l'UCI étudié.

Ces premiers résultats obtenus sur différents problèmes de référence nous permettent d'établir certaines conclusions préliminaires. Lorsque le nombre d'exemples pour le problème posé est important et qu'il est possible de disposer d'un ensemble de validation de taille suffisante pour mettre en œuvre les différentes heuristiques de sélection, les résultats observés sont relativement stables. Pour un niveau d'élagage allant jusqu'à 60%, prendre en compte la diversité lors de la sélection des classifieurs peut améliorer la précision de l'ensemble résultant, voir même dans certains cas améliorer la précision par rapport à l'ensemble initial. Cependant, lorsqu'on désire construire des ensembles de taille réduite, la précision individuelle des classifieurs reste le critère à privilégier. Chercher à maximiser la diversité se traduit généralement par un ensemble de classifieurs moins précis. Nous expliquons cette tendance par le fait que les classifieurs les plus divers sont généralement les moins précis, comme le montrent les différents diagrammes kappa-

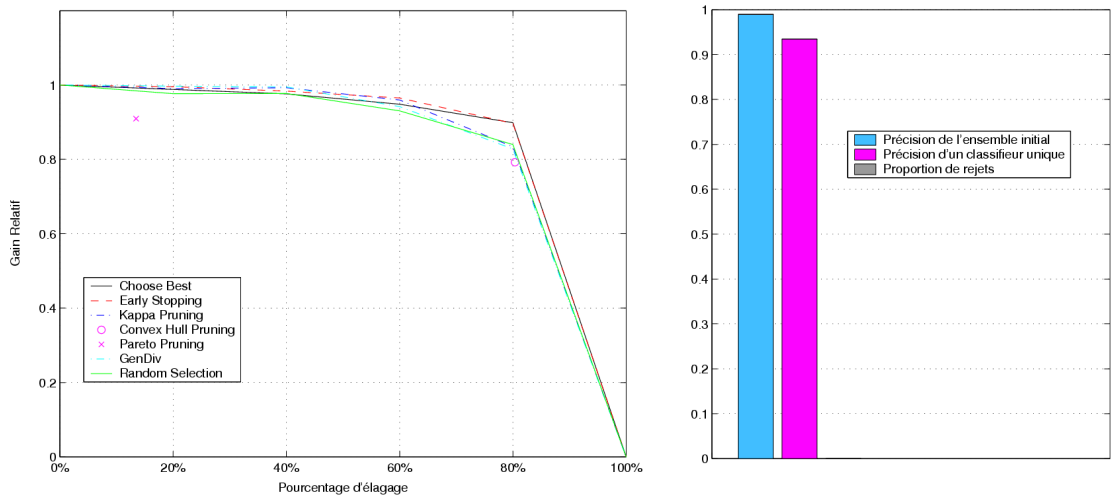

FIG. 4.11 – Résultats obtenus sur le jeu de données *pendigits*.

erreur présentés dans ce mémoire. En sélectionnant ces classifieurs, le vote majoritaire ne garantit plus une amélioration de la précision mais au contraire contribue à sa diminution.

Pour des jeux de données possédant un faible nombre d'exemples et présentant une grande instabilité, l'influence de la diversité, et plus généralement la comparaison d'heuristiques de sélection, sont beaucoup plus discutables. Pour les figures 4.12 à 4.18 concernent des bases de données pour lesquelles le nombre d'exemples disponibles varie entre 200 et 1000. Globalement, on observe une plus grande instabilité des résultats malgré un nombre d'essais plus important, et une décroissance plus rapide du gain relatif avec la diminution de la taille de l'ensemble sélectionné.

Nous présentons dans la figure 4.12 les résultats obtenus pour la base de données *german-credit*

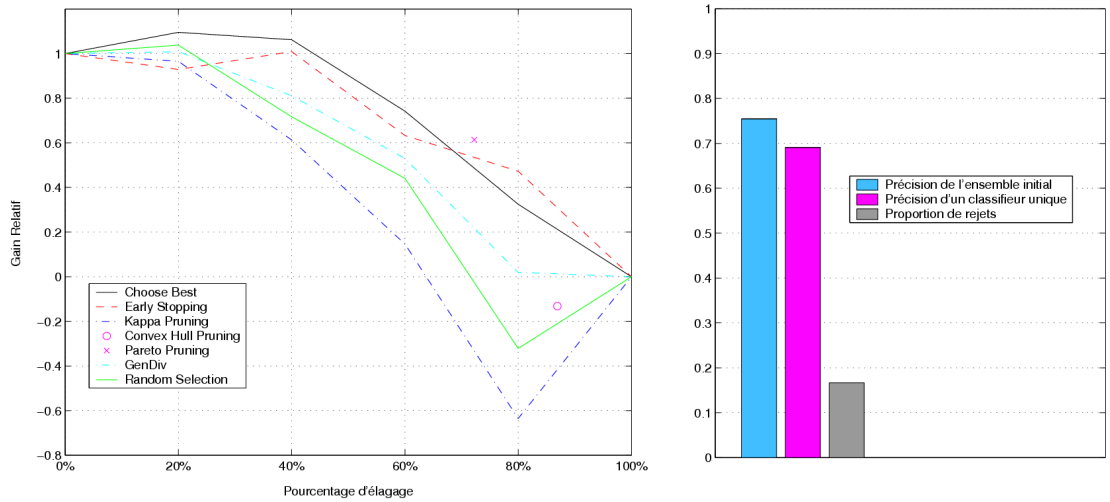

FIG. 4.12 – Résultats obtenus sur le jeu de données *german-credit*.

Nous constatons que la suppression d'un faible nombre de classifieurs permet d'améliorer légèrement la précision de l'ensemble obtenu en écartant les classifieurs les moins précis. Privilégier

la précision lors de la sélection des classifieurs semble être le moyen le plus efficace pour améliorer la précision de l'ensemble obtenu comme le montre les gains obtenus par l'heuristique *Choose Best*. Nous constatons également que la sélection de classifieurs basée sur la diversité donne ici de mauvais résultats, pouvant aller jusqu'à l'obtention d'un gain négatif. De manière générale, les ensembles de classifieurs construits par l'application de l'heuristique *Kappa Pruning* ont une précision inférieure à ceux construits par sélection aléatoire ce qui montre que la diversité est un critère de sélection inadapté pour ce problème.

Le comportement des différentes heuristiques de sélection pour la base *pima-diabetes* est donné dans la figure 4.13.

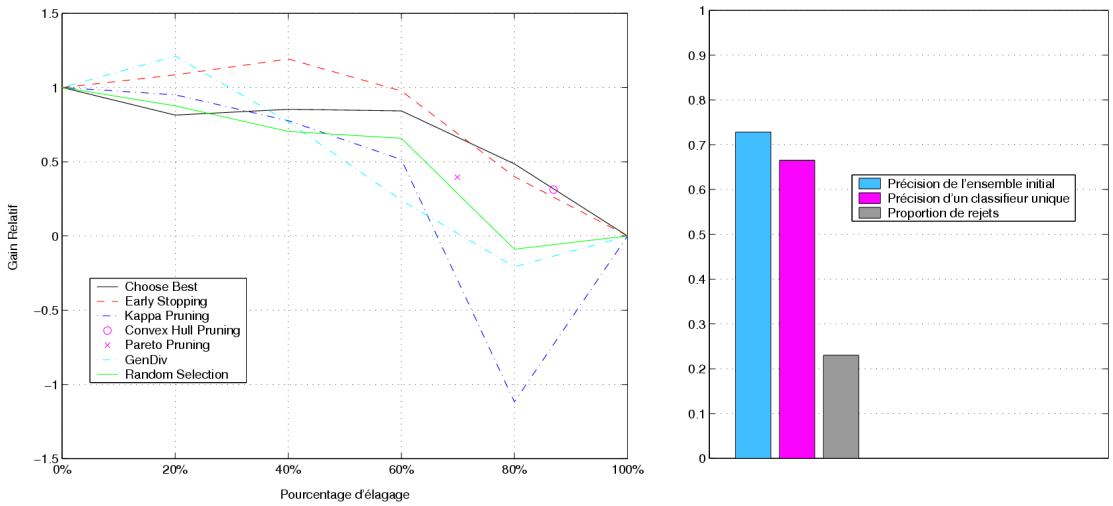

FIG. 4.13 – Résultats obtenus sur le jeu de données *pima-diabetes*.

L'utilisation de la diversité pour ce problème permet d'améliorer la précision des ensembles de classifieurs obtenus pour des valeurs d'élagage inférieures à 20%. Cependant, à partir d'une valeur de 40%, la précision semble de nouveau être le critère offrant la meilleure garantie pour la robustesse des résultats ; la diversité, quant à elle, produit des ensembles de classifieurs moins précis qu'un classifieur unique.

Pour la base de données *segment* dont les résultats sont présentés dans la figure 4.14, la diversité semble également être un critère inadapté pour la construction d'ensemble de classifieurs.

Bien que le problème soit plus stable et présente un taux de rejets nul, l'algorithme génétique proposé ne permet pas d'améliorer la précision des ensembles de classifieurs obtenus. Sélectionner les classifieurs les plus divers se traduit par des ensembles bien moins précis que ceux qu'il est possible d'obtenir en utilisant l'heuristique *Choose Best*.

Les résultats obtenus pour le jeu de données *sonar* à l'inverse semblent montrer que la diversité est un bon critère de sélection pour ce problème.

Les courbes présentées dans la figure 4.15 indiquent que privilégier la diversité pour un ensemble de 40 classifieurs (20% d'élagage) permet d'améliorer la précision de l'ensemble initial. Nous émettons cependant certaines réserves dues à l'instabilité des données. En effet, le faible nombre d'exemples, le taux de rejets pour ce problème ainsi que les variations des performances relatives à la sélection aléatoires (que nous utilisons en tant qu'heurisque *témoin*) limitent les conclusions qu'il est possible de tirer des résultats obtenus.

Un comportement similaire peut être observé dans les figure 4.16 et 4.17 qui montrent respectivement les résultats obtenus pour les bases de données *ionosphere* et *glass*.

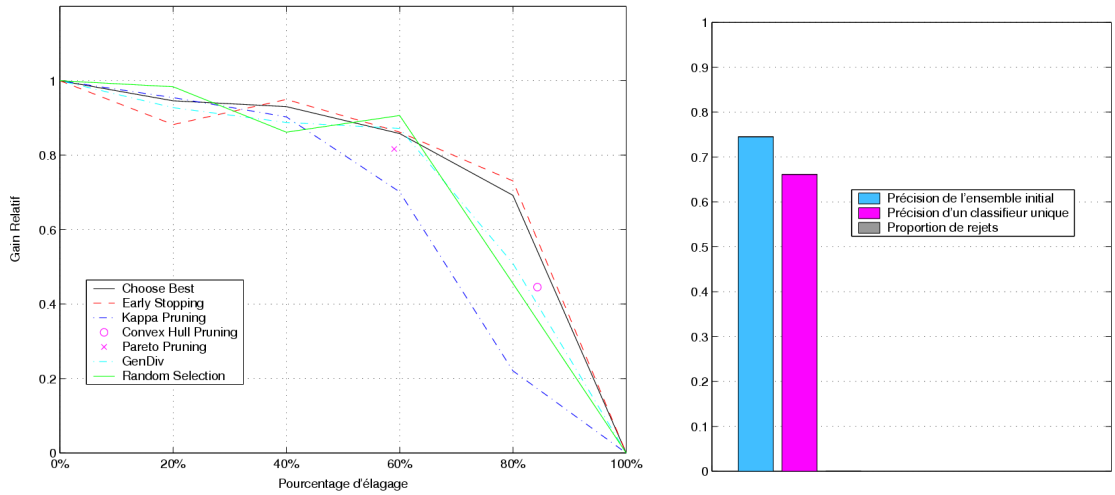

FIG. 4.14 – Résultats obtenus sur le jeu de données *vehicle*.

Pour ces problèmes, la précision individuelle des classifieurs semble à nouveau être le meilleur critère de sélection pour garantir les performances de l'ensemble obtenu.

Nous terminons par le jeu de données *balance-scale* dont les résultats sont présentés dans la figure 4.18. Ce problème est celui pour lequel nous avons observé le plus grand nombre de rejets. Un tiers des essais effectués pour évaluer les performances des différentes heuristiques de sélection ont produit un ensemble de 50 arbres de décision dont la précision était inférieure à un arbre unique entrainé sur toutes les données d'apprentissage. Ce problème est donc particulièrement instable, comme le montrent les courbes obtenues.

Dans cette situation, il est difficile de comparer les performances des différentes heuristiques utilisées. L'heuristique *Choose Best* semble être le choix le plus judicieux pour se prémunir de cette instabilité, en choisissant les classifieurs possédant la plus grande précision individuelle. Il apparaît d'ailleurs que réduire la taille de l'ensemble de départ en privilégiant la précision permet systématiquement d'obtenir un gain supérieur à 1. A l'inverse, privilégier la diversité se traduit par un ensemble de classifieurs très imprécis, comme le montre les gains négatifs observés pour l'heuristique *Kappa Pruning*. Cependant, nous nous gardons d'établir des conclusions définitives en raison de la forte instabilité de ce jeu de données.

Certaines interrogations subsistent également à propos des performances relatives à l'heuristique *Early Stopping*. Les résultats obtenus par Margineantu et Dietterich [138] montrent que cette heuristique de sélection est celle qui permet d'obtenir les ensembles les moins précis en pratique. Les expériences décrites dans ce chapitre montrent cependant que cette heuristique donne des résultats souvent proches, en terme de gain, de l'heuristique consistant à sélectionner les classifieurs possédant la plus grande précision individuelle. Une explication possible concerne la différence au niveau du protocole utilisé pour séparer les exemples en apprentissage, en validation et en test. Les résultats obtenus tendent à montrer que pour nos expériences, l'algorithme Adaboost a construit les classifieurs de l'ensemble initial par ordre de précision décroissante, ce qui expliquerait le comportement similaire des heuristiques *Choose Best* et *Early Stopping*.

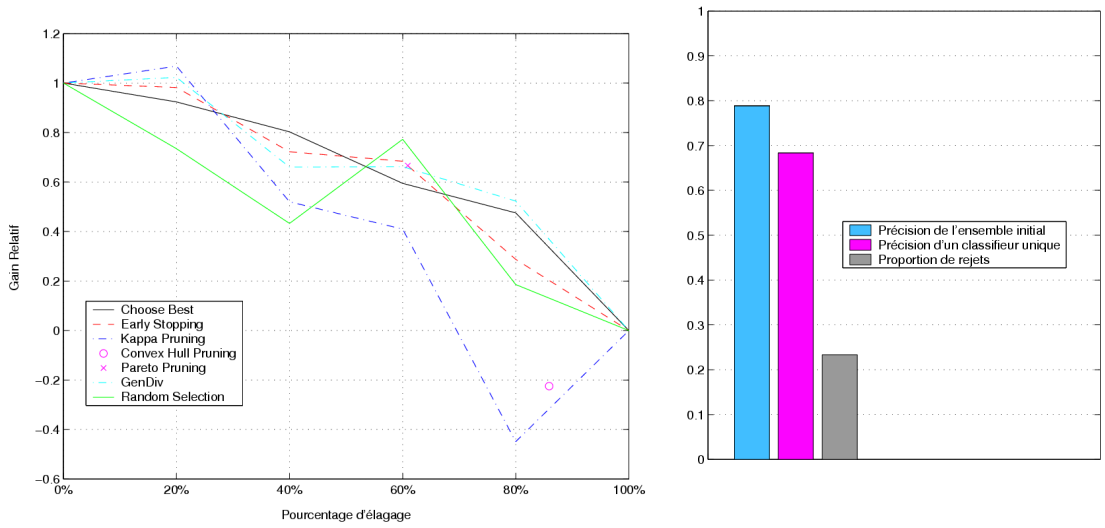

FIG. 4.15 – Résultats obtenus sur le jeu de données *sonar*.

4.3.3 Vers une possible explication de l'instabilité des résultats observés

Pour les premiers jeux de données que nous étudions, pour lesquels il est possible d'effectuer la sélection de classifieurs sur un ensemble de validation comportant un grand nombre d'exemples, il est possible de faire des observations relativement stable. Cependant, pour les autres bases de données de l'UCI, il est beaucoup plus difficile d'établir des conclusions.

Pour expliquer ce phénomène, nous observons la réduction de l'erreur en parallèle sur l'ensemble de validation et sur l'ensemble de test pour deux bases de l'UCI. La figure 4.19 décrit les variations de l'erreur obtenue sur la base *letter*. Pour obtenir cette figure, nous séparons les exemples en suivant le protocole décrit dans la table 4.2. Nous fixons le nombre maximum d'itérations de notre algorithme génétique à 100 et pour chaque itération, nous évaluons la précision obtenue par le meilleur ensemble de classifieurs de la génération courante sur l'ensemble de validation (représentée par la courbe bleue) et sur l'ensemble de test (représentée par la courbe mauve) ainsi que la précision moyenne obtenue par les 10 ensembles de classifieurs constituant la population de la génération courante sur l'ensemble de validation (représentée par la courbe rouge). Pour la base *letter* qui est relativement stable, ce calcul est répéré dix fois, et nous donnons l'évolution moyenne de l'erreur.

Nous effectuons une expérience similaire sur le jeu de données *balance-scale*, qui est instable, et nous donnons dans la figure 4.20 l'évolution moyenne de l'erreur calculée sur 30 essais différents.

Sur la figure 4.19, qui correspond à un cas où le nombre d'exemples de validation est suffisant, l'optimisation réalisée pour la sélection des classifieurs s'applique aussi bien à l'ensemble de validation qu'à l'ensemble de test. La diminution de l'erreur est cependant nettement plus importante sur les exemples utilisés pour construire l'ensemble par sélection des classifieurs. En revanche, sur la figure 4.20 qui correspond à un cas où l'ensemble de validation est de taille réduite, la diminution de l'erreur obtenue par application de l'algorithme génétique proposé ne se généralise pas à l'ensemble de test. Dans ce cas, nous n'avons aucune garantie que l'ensemble de classifieurs possédant la plus grande précision sur l'ensemble de validation soit également celui minimisant l'erreur sur l'ensemble de test, ce qui explique l'instabilité des résultats observés.

Ce constat semble général sur toutes les bases testées avec des variations d'amplitude selon la base considérée : autrement dit, les bonnes performances d'un ensemble de classifieurs

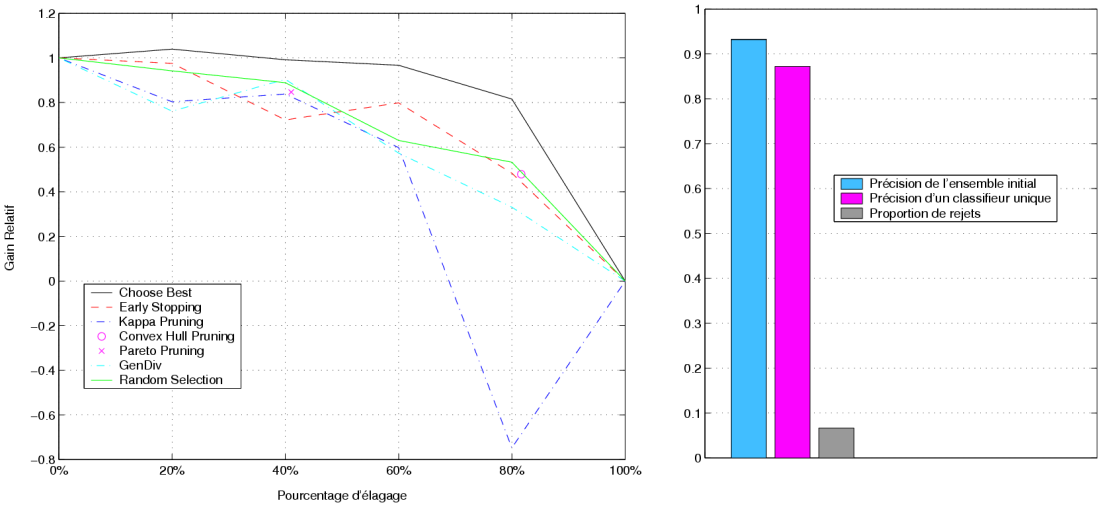

FIG. 4.16 – Résultats obtenus sur le jeu de données *ionosphere*.

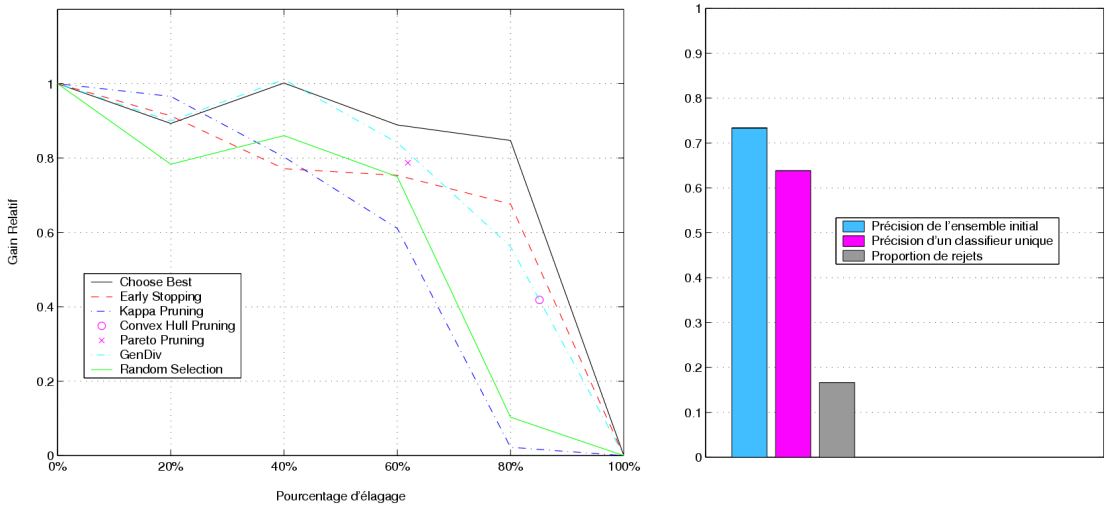

FIG. 4.17 – Résultats obtenus sur le jeu de données *glass*.

sur les données de validation se généralisent mal ou peu sur l'ensemble de test. En revanche, lorsqu'on se place dans le cas d'un classifieur unique, la précision d'un classifieur calculée sur un ensemble d'exemples donnés se répercute de manière beaucoup plus systématique sur de nouveaux exemples, à l'exception d'un cas de sur-apprentissage.

4.4 Conclusion du chapitre

Au vu des résultats de ce chapitre, une première conclusion rapide indique que la précision semble être le facteur prépondérant lors de la sélection d'un ensemble de classifieurs. Cette première conclusion vient de l'observation des résultats exposés ci-dessus, dans lesquels l'heuristique *Choose Best* qui ne tient pas compte de la diversité est globalement meilleure. Que penser alors des conclusions connues qui établissent qu'un ensemble obtenu par boosting est généralement meilleur qu'un ensemble obtenu par bagging ? Ces conclusions sont d'ailleurs mises

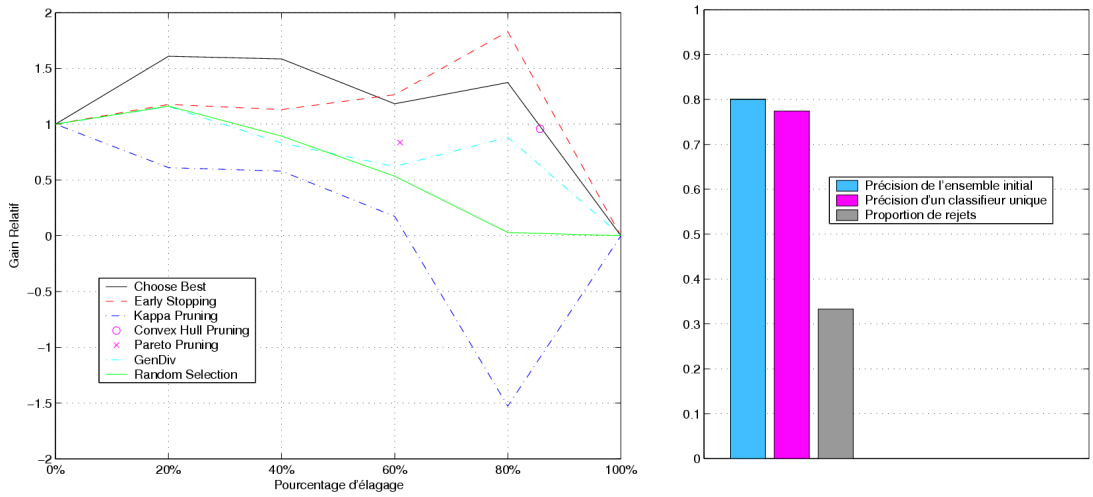

FIG. 4.18 – Résultats obtenus sur le jeu de données *balance-scale*.

en évidence au début de ce chapitre dans la figure 4.1, lorsque nous comparons le compromis précision-diversité dans les diagrammes kappa-erreur correspondant au bagging et au boosting. Si un ensemble de classifieurs globalement plus précis mais peu divers est souvent moins bon qu'un ensemble composé de classifieurs plus divers, c'est que la diversité a une influence sur la précision de l'ensemble.

Concernant le rôle de la diversité dans la construction d'ensembles de classifieurs, si nous devions formuler de manière simpliste les conclusions que nous en tirons et répondre à la question *faut-il oui ou non prendre en compte la diversité lors de la sélection des classifieurs* alors la réponse serait : *la diversité oui, mais pas à n'importe quel prix et pas dans n'importe quelles conditions.*

Par *n'importe quel prix*, nous sous-entendons l'importance donnée à la diversité lors de l'utilisation d'une heuristique de sélection des classifieurs. En effet, au vu des courbes obtenues, il apparaît que donner une trop grande importance à la diversité limite le gain de précision qu'il est possible d'obtenir par rapport à un classifieur unique entrainé sur le même ensemble d'apprentissage. Dans certains cas, nous pouvons même observer un gain négatif, c'est à dire que l'ensemble élagué, pour lequel ce sont les classifieurs les plus divers qui ont été sélectionnés, possède une précision inférieure à celle obtenue à l'aide d'un arbre de décision unique.

Cette constatation peut s'expliquer. En effet, nous savons qu'il existe un compromis entre la précision et la diversité de deux classifieurs. En favorisant la diversité lors de la sélection des classifieurs de l'ensemble, nous nous exposons inévitablement à la construction d'un ensemble constitués de classifieurs peu précis. Ceci nous conduit à une situation dans laquelle le vote majoritaire ne garantit plus une amélioration de la précision, situation que Kuncheva désigne sous le terme de *Pattern of Failure* [113].

Dans ce sens, les résultats que nous présentons dans ce chapitre sont en adéquation avec ceux obtenus par Rodriguez *et al.* [179], qui montrent que la précision individuelle des classifieurs reste le facteur offrant la meilleure garantie pour construire d'ensembles de classifieurs également précis.

Cependant, même lorsque la diversité contribue de manière effective à la précision d'un ensemble de classifieurs, nous nous devons de l'utiliser avec parcimonie. En effet, pour pouvoir observer l'influence de la diversité dans les expériences réalisées au cours de ce chapitre, nous

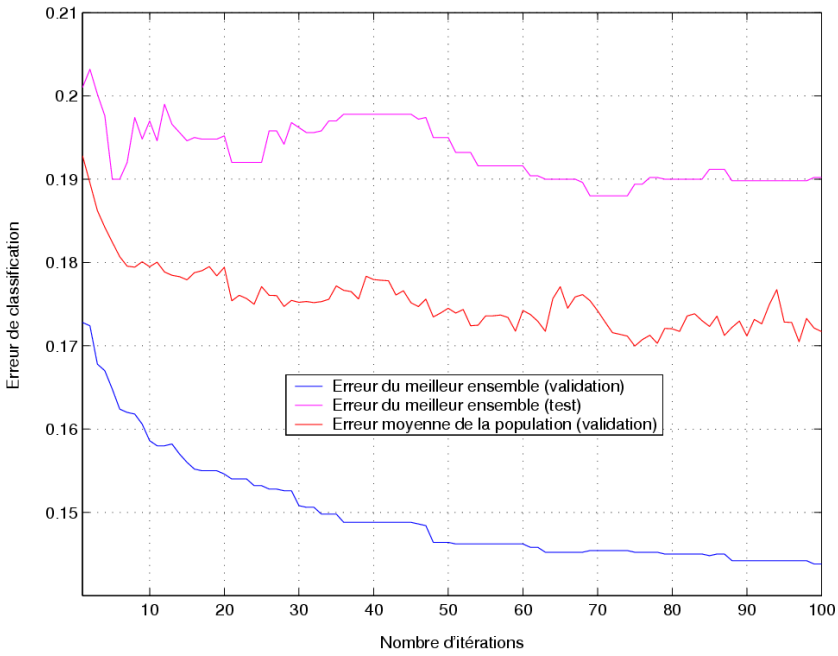

FIG. 4.19 – Variations de l'erreur en classification sur le jeu de données *letter*. La diminution de l'erreur sur l'ensemble de validation (en bleu pour le meilleur ensemble de classifieurs à l'itération courante, en rouge pour l'erreur moyenne des ensembles de classifieurs constituant la population courante) s'accompagne d'une diminution plus faible de l'erreur sur l'ensemble de test (en mauve).

avons du réunir plusieurs conditions. Prenons l'exemple d'une situation dans laquelle l'utilisation d'un ensemble de 50 arbres de décision ne permet d'améliorer la précision que de 1% par rapport à un classifieur unique. Dans une telle situation, il peut être difficile de justifier l'emploi d'un ensemble de classifieurs, plus particulièrement si l'écart type de la précision obtenue est important. Si nous devions maintenant discuter de l'importance de la diversité pour sélectionner un sous-ensemble de ces classifieurs, nous risquerions alors de fonder nos conclusions sur de simples effets de bords.

Il est utile de comparer différentes heuristiques de sélection des classifieurs lorsque le gain absolu, c'est à dire la différence de précision entre un classifieur unique et l'ensemble de classifieurs de départ (ici 50 arbres de décision) est importante. Dans ce cas, nous constatons que prendre en compte la diversité lors de l'élagage de l'ensemble de départ peut permettre d'améliorer légèrement la précision de l'ensemble obtenu pour un niveau d'élagage de 80% à 60%. Au delà de ce seuil, la précision reste le facteur à privilégier pour la sélection des classifieurs.

Nous précisons cependant que les discussions présentées dans ce chapitre à propos de la diversité ne concernent que la construction d'ensembles de classifieurs selon l'approche Surproduction et Sélection. Dans le cas où l'ensemble est construit de manière incrémentale, comme c'est le cas pour les algorithmes DECORATE ou Rotation Forest par exemple, la diversité est utilisée dans un contexte différent.

Au terme de ce chapitre, nous avons mis en regard plusieurs méthodes permettant de construire un ensemble de classifieurs selon le paradigme Surproduction et Sélection. S'il est difficile de tirer des conclusions définitives sur le rôle de la diversité pour construire un ensemble par sélection de classifieurs, notre étude nous a permi de mieux relativiser l'apport des différentes heuristiques

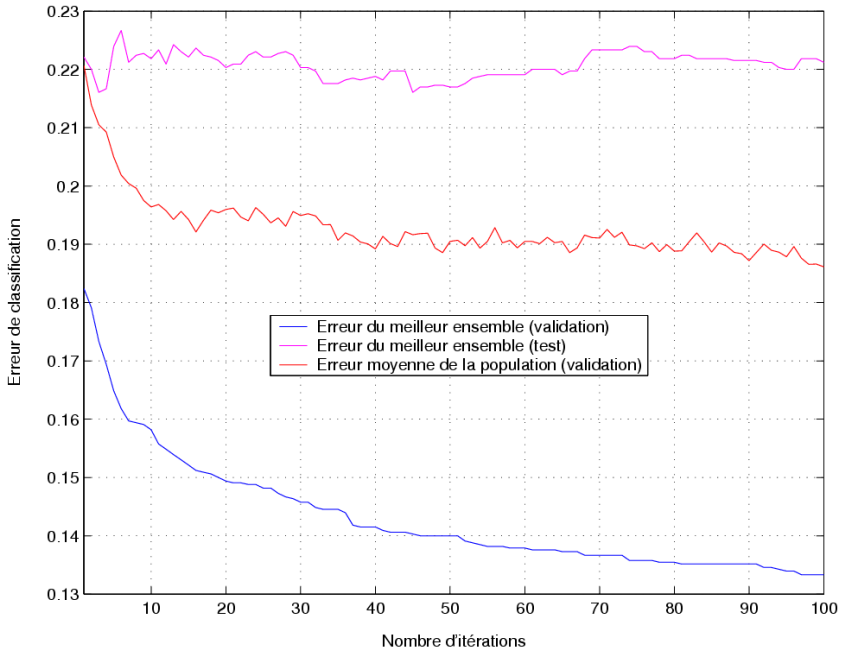

FIG. 4.20 – Variations de l'erreur en classification sur le jeu de données *balance-scale*. Bien que l'erreur calculée sur l'ensemble de validation diminue en fonction du nombre d'itérations, cette diminution ne s'applique pas à l'ensemble de test, du fait du nombre limité d'exemples en validation.

de sélection existantes.

Nous pouvons cependant apporter des réponses partielles, qui rejoignent des conclusions établies par d'autres travaux. Roli *et al.* [180] montrent qu'il n'est pas possible, dans le cadre d'une approche Surproduction et Sélection, de dire qu'une heuristique est meilleure qu'une autre car le choix de l'heuristique employée dépend du problème posé. Nous confirmons cette conclusion, et proposons certaines conditions permettant de guider le choix d'une heuristique particulière, en fonction du nombre d'exemples à disposition et du gain de précision offert par l'ensemble comparativement à un classifieur unique. Kuncheva souligne également le caractère *insaisissable* de la diversité [112] en tant que critère de sélection des classifieurs. Les travaux présentés dans cette thèse constituent un pas supplémentaire vers la compréhension de la diversité.

5

Application à la détection des fumées

Nous présentons dans ce chapitre la mise en pratique des algorithmes de classification issus de l'Apprentissage Automatique pour le contrôle de la pollution de l'air [67, 69, 68, 65, 66]. Cette contribution pratique s'inscrit dans le contexte plus général de l'amélioration de la surveillance de la pollution atmosphérique. L'objectif de ce chapitre concerne le développement et l'utilisation d'une plateforme logicielle et matérielle permettant un contrôle en temps réel des rejets de fumées survenant dans la zone proche d'une source de pollution potentielle, et de leur impact sur l'environnement. Le but sur le long terme est de contrôler et dimuner de manière significative la pollution atmosphérique en Europe.

La demande à l'échelle Européenne de moyens pour contrôler et réguler en temps réel la pollution atmosphérique est en constante augmentation. La législation en vigueur depuis plusieurs années relative à la régulation de la pollution de l'air concerne aussi bien les rejets ponctuels que les émissions diffuses. Depuis 1999, la société ALOATEC a mis en place le développement d'un système de caméras assistées par ordinateur permettant d'effectuer une surveillance en temps réel des rejets de fumées polluantes survenant au niveau de sources diffuses ou ponctuelles durant la journée. Les figures 5.1 et 5.2 présentent respectivement un exemple d'émissions diffuses observables sur des sites sidérurgiques de type acierie et un rejet ponctuel filmé au niveau des torchères d'une usine pétrochimique. En complément de ce système, la société ALOATEC étudie également une solution basée sur l'utilisation de caméras infrarouges permettant la détection nocturne d'événements à risque.

L'objectif du projet ayant initié les recherches présentées dans ce mémoire est la conception d'un système de contrôle temps réel des rejets de pollution atmosphérique basé sur l'utilisation de caméras intelligentes. Ce choix peu conventionnel d'utiliser des caméras est dû aux limitations des systèmes de surveillance traditionnels qui se basent sur des relevés chimiques. En effet, les capteurs utilisés pour mesurer la présence de particules ou de matières polluantes dans l'air sont spécifiques à l'activité industrielle. L'installation de ces capteurs sur de hautes cheminées est difficile et coûteux. De plus, ces capteurs placés de manière ponctuelle ne permettent pas de détecter des émissions diffuses pouvant survenir partout sur le site.

Notre contribution concerne plus particulièrement le développement d'un module d'apprentissage reposant sur le traitement d'image de scènes à risque préalablement enregistrées et de la décision associée provenant d'un expert humain, de manière à améliorer les performances du système de détection et de classifcation temps réel de ces scènes à risque. L'évaluation de ce module est réalisé par les clients d'ALOATEC, directement concernés par les rejets de fumées polluantes.

Les travaux de recherche présentés dans ce mémoire ont été réalisés en partenariat avec la

FIG. 5.1 – Exemples d'émissions de fumées diffuses survenant sur des sites industriels sidérurgiques de type acierie.

FIG. 5.2 – Rejet nocif ponctuel survenant au niveau des torchères d'usines pétrochimiques.

société ALOATEC. Cette start-up française, fondée en 1998 et située à Calais, développe des systèmes de contrôle innovants reposant sur l'utilisation de caméras assistées par ordinateur. C'est la société ALOATEC qui est à l'origine du développement du système de détection de pollution atmosphérique présenté plus haut. Ce système intègre des caméras numériques superposées à une couche logicielle destinée à analyser les scènes filmées par traitement d'image et à émettre un signal d'alarme sur le réseau de l'entreprise en cas de problème. Le principe d'un tel système a été validé dans des conditions d'utilisation réelles sur différents sites industriels. Cependant, le système de détection dans sa conception initiale nécessite des adaptations pour améliorer sa fiabilité, sa portabilité et son paramétrage. Nous présentons maintenant un état de l'art des systèmes de surveillance existant et décrivons en détail les spécifications de la solution mise au point par la société ALOATEC pour la surveillance des fumées polluantes.

5.1 Spécifications et limitations du système initial

5.1.1 Etat de l'art des solutions existantes

Pour le contrôle de la pollution atmosphérique, la plupart des systèmes existants se limitent à un traitement différé et ne permettent pas une analyse en temps réel. Les principales sources de rejets industriels proviennent généralement de cheminées dont l'activité est contrôlée par des relevés effectués directement à l'origine de l'émission. Cependant, pour des cheminées de grande taille, ou pour des sources diffuses, il n'existe généralement aucun moyen permettant une surveillance des émissions polluantes.

Les solutions existantes basées sur l'utilisation de caméras sont en nombre très limité. La plupart des systèmes permettant une surveillance visuelle de l'activité reposent sur l'utilisation d'un écran de contrôle par un opérateur humain. Un grand nombre de sites sidérurgiques et pétrochimiques sont d'ailleurs équipés d'écrans de contrôle à cette fin.

Cette solution présente cependant des limitations : la plus importante étant sa fiabilité. En effet, l'opérateur humain affecté à la surveillance des écrans de contrôle effectue souvent plusieurs tâches en parallèle et il lui est difficile, voir impossible, de relever la totalité des événements à risque survenant sur le site.

Les rares solutions de détection temps-réel basées sur l'utilisation de caméras qui existent sont très spécifiques ce qui rend impossible leur portabilité à différents contextes. De plus, le coût de ces systèmes est très élevé.

En réponse aux limitations des solutions existantes pour des besoins industriels, la société ALOATEC a développé un système de surveillance innovant basé sur l'utilisation de caméras assistée par ordinateurs. Associées à des programmes spécifiques au traitement d'image et de détection des événements à risque, ce système constitue une solution fiable et à moindre coût pour la surveillance de la pollution atmosphérique dans l'industrie. Nous présentons maintenant en détail l'architecture matérielle et logicielle de ce système.

5.1.2 Architecture globale du système

Le système développé par la société ALOATEC depuis 1999 est un système de surveillance video de scènes à risque destinés à l'industrie et permettant de pallier à la plupart des limitations exposées précédemment. L'utilisation de caméras constitue une solution simple et économique, permettant en outre une surveillance en temps réel de l'ensemble d'un site industriel. Les cheminées à l'origine de rejets ponctuels comme les émissions diffuses peuvent ainsi être contrôlées par un système unique.

Pour la surveillance des rejets de fumées, la société ALOATEC et les industriels avec lesquels elle collabore ont adopté une convention permettant permettant d'associer à chaque événement une classe, ou degré de gravité. Il existe quatre classes ou niveaux de gravité distincts pour un événement. Le niveau 0, le plus faible, correspond généralement à un événement de faible nocivité, voire inoffensif dans le cas où la fumée présente dans la scène est constituée de vapeur d'eau. Ce phénomène est fréquemment observé sur des sites sidérurgiques, lorsque du métal en fusion est refroidi par des quantités d'eau importantes. A l'inverse, les rejets de niveau 3 présentent un risque élevé et sont généralement constitués de polluants tels que des particules de métaux lourds. Un événement appartenant à cette classe provoque souvent une alarme destinée à alerter les superviseurs, qui prennent alors la décision de stopper temporairement le processus de fabrication.

Dans ce dernier cas, l'émission d'un signal d'alarme et la prise de décision doivent s'effectuer de manière rapide. Le système développé par ALOATEC permet d'analyser la scène filmée en

temps réel et d'attribuer à un événement la classe correspondante dès que celui-ci est détecté. Lorsque qu'une pollution de niveau 3 est détectée, le système émet un signal d'alarme sur le réseau intranet du site de production de manière à permettre une réaction rapide des superviseurs.

Grâce à ce système, la présence en continu d'un expert au poste de surveillance des fumées n'est plus nécessaire. Son intervention reste cependant indispensable pour *calibrer* le système, c'est à dire définir manuellement les règles de classification permettant de déterminer la gravité des événements filmés. Pour cela, l'ensemble des scènes filmées par caméra est enregistré dans une base de données et consultable de manière différée. Ceci permet également à l'expert de corriger la classe attribuée par le système de détection lorsqu'un événement est incorrectement classé, ce qui arrive généralement lorsque la scène filmée est pertubée par des phénomènes extérieurs.

L'architecture matérielle du système de surveillance telle qu'elle est mise en place sur différents sites industriels est donnée dans la figure 5.3. Les caméras filmant la scène sont reliées à des armoire de terrain chargées d'analyser les séquences vidéos. Les images et les signaux calculés en temps réel sont enregistrés dans une base de données et stockées sur un serveur de manière à pouvoir être consultées sur le réseau interne de l'entreprise.

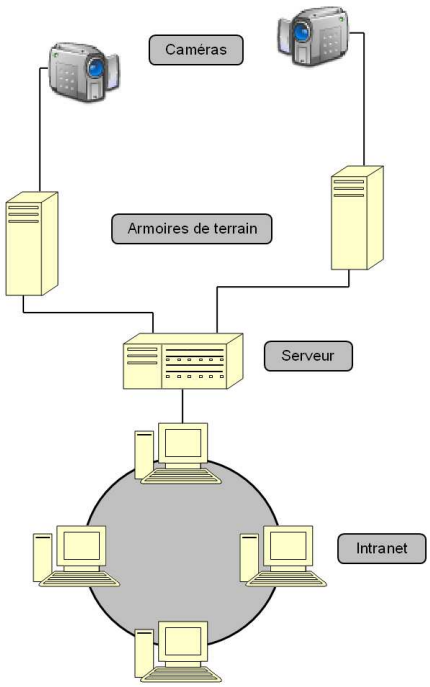

Fig. 5.3 – Architecture matérielle du système de surveillance développé par ALOATEC.

Les caméras sont placées à distance raisonnable du site de production de manière à ce que l'ensemble des zones où des rejets de fumées qui sont susceptibles d'être observés soient visibles dans le champ de la caméra. Les caméras sont également disposées dans des box, derrière des parois vitrées, pour les protéger des intempéries. Ces caméras sont raccordées à des armoires de terrain qui sont dédiées au traitement d'images en temps réel de la scène observée ainsi que de l'attribution des niveaux de pollution des différents événements détectés. L'ensemble des séquences vidéos enregistrées ainsi que les signaux calculés par traitement d'image sont ensuite stockées sur une base de données SQL mise à disposition sur une machine serveur. Les opérateurs présents sur le site de production peuvent ensuite visionner les événements, en temps réels ou de

manière différée, depuis leur poste situé sur le réseau interne du site.

Le système de caméras assistées par ordinateur a pour objectif principal de reproduire la décision fournie par l'expert humain concernant le niveau des rejets de fumées observés. Ce problème comprend deux étapes importantes : l'obtention des signaux, à partir des caméras (et éventuellement d'autres types de capteurs), et représentatifs du phénomène observé par l'expert et la reproduction du raisonnement et de la décision établie par l'expert à partir de ces signaux. Nous présentons à présent les deux composants logiciels permettant de réaliser ces deux étapes de manière automatique. Rappelons encore que nos travaux de recherche concernent uniquement le second composant dedié à la classification des rejets de fumées.

5.1.3 Description du module de traitement d'image

L'opérateur humain affecté à la surveillance des écrans de contrôle prend ses décisions sur un panache de fumées en tenant compte de son aspect (par exemple une fumée dense accompagnée d'une flamme à l'extrémité d'une cheminée). La connaissance qu'il utilise pour décider est acquise très rapidement.

Par conséquent, la détection d'un événement à risque est le résultat combiné d'un processus de déduction rapide basée sur l'observation et l'expérience d'événements antérieurs et d'une connaissance préalable de l'activité industrielle sous surveillance. Par exemple, la localisation de la source du rejet, ainsi que l'étape du processus de production durant laquelle est détecté le rejet en question, peuvent conforter ou affaiblir la décision de l'expert.

La société ALOATEC a développé des procédés de traitement d'images spécifiques à la surveillance en temps réel de la pollution atmosphérique. Le module de traitement d'image manipule deux types de signaux. Le premier type est utilisé pour la pré-détection c'est à dire l'attribution d'un top de début et d'un top de fin à chaque événement. Lorsqu'un événement est pré-détecté, une fenêtre dynamique se positionne dans la zone de l'image où a lieu le rejet de fumées. C'est dans cette zone dynamique que sont calculés les signaux du second type qui concernent les caractéristiques du rejet de fumées enregistrées entre le top de début et le top de fin. Nous donnons ci-dessous deux des signaux utilisés pour la pré-détection des événements et quatre exemples de signaux utilisés pour mesurer les caractéristiques des fumées. Cependant, pour des raisons de confidentialité, certains détails de conception ainsi que certains des signaux utilisés ne peuvent pas être présentés dans ce mémoire.

FIG. 5.4 – Disposition des zones de détection du système sur une usine de type acierie. A gauche, les zones de détection sont positionnées de façon à analyser l'évolution d'un rejet survenant au niveau d'une source donnée. A droite, il y a autant de fenêtres de détection que de zones où pouvent survenir des rejets de fumée.

FIG. 5.5 – Disposition des zones de détection du système pour contrôler l'activité d'une usine pétrochimque. Les zones de détection sont disposées de manière à analyser l'évolution et la dispersion d'un rejet de fumées ponctuel survenant au niveau d'une torchère.

Analyse locale de la scène Le système de surveillance permet de positionner des fenêtres de détection dans certains zones de l'image où peuvent survenir des rejets de fumées polluantes. Les figures 5.4 et 5.5 montrent le positionnement de ces zones en différents points de la scène filmée. Ce traitement local est utilisé pour la *pré-détection* d'un événement.

Analyse fréquentielle L'analyse fréquentielle des variations de luminosité et de couleur des pixels présent dans les fenêtres de détection est utilisée pour attribuer les marqueurs de début et de fin des différents événements. Cette analyse permet également de déterminer les contours du panache de fumée.

Analyse de la luminosité et de la coloration des pixels La prise en compte de la couleur des fumées est due au fait que certaines des particules rejetées dans l'air possèdent des propriétés colorimétriques spécifiques. Des exemples de fumées polluantes colorées émises par les acieries sont visibles dans la figure 5.1. La luminosité est également utilisée pour isoler un rejet de fumées nocives dans la scène filmée, par contraste avec la luminosité de l'arrière plan.

Analyse de la surface du rejet Un composant dédié réalise une estimation de la surface du nuage de fumée en effectuant un comptage des pixels détectés par les composants précédants. Cette surface peut être évaluée en dimensions absolues lorsque la distance entre la source d'émission et la caméra enregistrant la scène est connue, ce qui est le cas dans le contexte de la surveilance des fumées industrielles.

Analyse de la densité du rejet La densité du panache est calculée par un algorithme spécifique permettant de déterminer la concentration de la pollution. La densité d'un pixel est évaluée par le recensement des pixels adjacents ayant été marqués comme appartenant au nuage de fumée. Ce paramètre est fortement lié à la gravité du rejet de fumées polluantes.

Analyse de la forme du rejet La forme du nuage de pollution est basée sur le calcul en temps réel de ses délimitations avec la source d'émission. C'est l'un des paramètres les plus pertinents pour réduire le doute lors de la prise de décision.

Le module de traitement d'image du système est résumé dans la figure 5.6. Associé à un programme permettant l'enregistrement et le visionnage de scènes antérieures, stockées dans une base de données, il permet une évaluation rapide de sa pertinence, par comparaison entre l'évolution des signaux calculés et de l'image visualisée simultanément. Les algorithmes de traitement

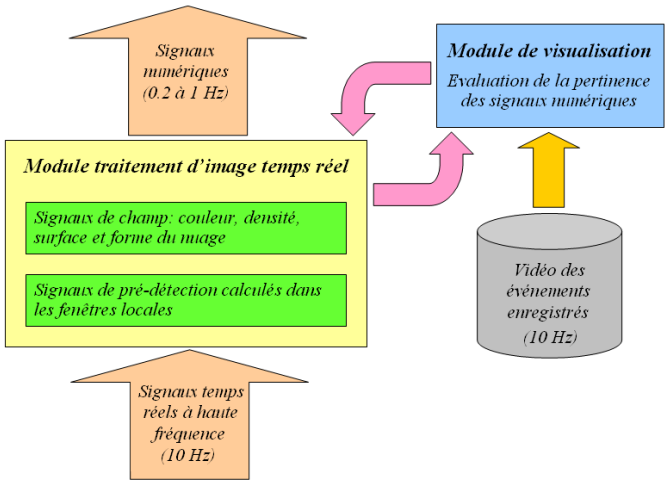

FIG. 5.6 – Module de traitement d'image du système de surveillance développé par ALOATEC. Des informations visuelles à haute fréquence enregistrées par la caméra sont converties en signaux numériques utilisable par le système de détection.

d'image que nous avons décrits dans cette section ont été mis au point par la societé ALOATEC, qui bénéficie d'une grande expérience dans ce domaine. Les travaux présentés dans ce mémoire ne portent en aucun cas sur l'amélioration de ces signaux, mais sur leur utilisation dans le but d'améliorer la prise de décision, réalisée par le module de détection que nous décrivons à présent.

5.1.4 Description du premier module de classification

Avant de présenter plus en détail le module de classification du système développé par la société ALOATEC, nous clarifions certains des termes employés qui peuvent préter à confusion. Pour les industriels utilisant le système, la détection consiste à reproduire la décision de l'expert humain à partir de l'observation de la scène filmée. C'est pourquoi ce module est parfois appelé module de détection par l'industriel. Cette décision consiste à attribuer au rejet de fumée une note reflétant le degré de gravité du nuage. Comme nous l'avons précisé auparavant, cette note correspond à une variable discrète et ordinale, variant entre 0 pour un nuage inoffensif constitué de vapeur d'eau par exemple, et 3, qui correspond à un événement critique donnant lieu généralement à un signal d'alarme. C'est pourquoi nous qualifions ce composant de module de classification supervisée. La première version de ce module de classification supervisée était réalisée par un système expert, et non par des algorithmes d'apprentissage automatique. Ce système expert était basé sur des règles définies manuellement, à la fois par l'opérateur humain affecté à la surveillance des fumées, et par les concepteurs du système travaillant pour la société ALOATEC. Dans cette version, aucun apprentissage n'était réalisé par le système de manière à alléger sa calibration particulièrement laborieuse. C'est sur ce point particulier que porte notre travail, qui sera exposé en détail par la suite.

Le fait d'isoler des événements discrets d'un processus continu, c'est à dire attribuer à ces événements discrets un top de début et un top de fin, est désigné par les industriels sous le terme de *pré-detection*. Comme nous l'avons expliqué précédemment, cette pré-détection est réalisée par l'analyse de fenêtres spéficiques de l'image, positionnées de manière stratégique sur les zones où peuvent survenir d'éventuels rejets.

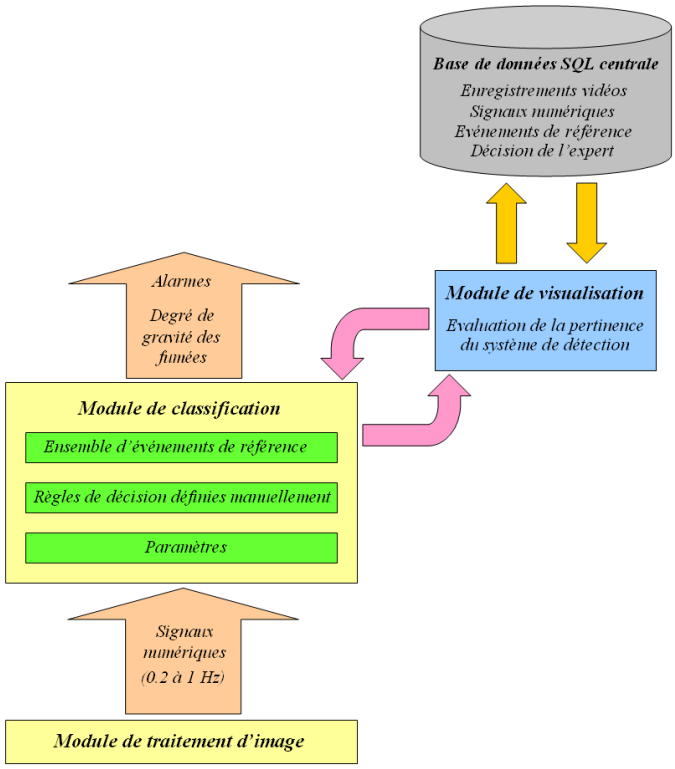

FIG. 5.7 – Schéma descriptif du système de classification initialement mis au point par ALOATEC.

Le module de classification consiste à reproduire le plus fidèlement possible la décision de l'expert humain affecté à la surveillance de l'activité. Cette prise de décision s'effectue à partir des différents signaux calculés par le module dédié au traitement d'image. Pour la machine, reproduire la décision de l'humain est un processus difficile. En effet, la présence de perturbations éventuelles dans la scène filmée risque de fausser la décision. Ces perturbations sont engendrées le plus souvent par :

- L'ensoleillement et la luminosité ambiante, qui peuvent faire varier la manière dont est perçue la scène.
- La direction du vent, qui peut réduire la surface visible du nuage de fumées. Dans le cas d'un vent latéral, cette surface sera beaucoup plus importante que dans le cas d'un vent venant face à la caméra par exemple.
- La présence de pluie ou de brouillard pouvant réduire la visibilité.
- Les nuages situés en arrière plan, pouvant être confondus avec des rejets de fumées nocives, et accessoirement altérer le fonctionnement des algorithmes permettant d'analyser la densité, la forme, et la surface du panache de fumées.
- La présence du soleil dans le champ de vision de la caméra, qui perturbe grandement la visibilité et modifie les couleurs de la scène.
- La présence d'un arrière plan coloré lorsque le soleil est situé à une position inférieure à 7°au dessus de l'horizon. De manière plus familière, ce phénomène correspond à un ciel de teinte rosée, survenant généralement au lever et au coucher du soleil, lorsque ce dernier est très bas.

L'architecture du module de classification initialement développé par ALOATEC est présentée dans la figure 5.7. Le module de traitement d'image permet d'extraire des signaux numériques à partir des scènes filmées par la caméra. Un système expert dont les règles sont définies manuellement par les concepteurs du programme, utilise ces valeurs numériques pour reproduire la décision de l'expert qui surveille les écrans de contrôle. La définition des règles de décision est effectuée à partir d'un ensemble d'événements de référence préalablement enregistrés par la caméra. Le module de détection attribue à chaque événement une classe correspondant à son degré de gravité, et émet une alarme sur le poste de contrôle dans le cas où un événement est jugé critique. Toutes les informations utilisées durant ce processus sont enregistrées dans une base de données contenant les séquences vidéos des événements filmés par les caméras, les signaux numériques associés ainsi que les degrés de gravité attribués par le système et l'expert humain. Un module d'évaluation permet de visionner les événements antérieurs enregistrés dans cette base de données, de manière à estimer l'efficacité des règles de décision utilisées et permettre une éventuelle correction par l'expert humain de la classe attribuée par le système.

La solution proposée par ALOATEC constitue un premier pas important dans le développement d'un système pouvant détecter automatiquement des rejets de fumées polluantes avec un taux de fausses alarmes variant entre 10% et 60% lorsque des perturbations sont présentes dans la scène filmée. Plusieurs industriels utilisent ce système depuis plusieurs années pour contrôler leur activité. Ce système présente cependant certaines limites. En effet, il est dépourvu de module d'apprentissage permettant de définir automatiquement une procédure de classification des événements à risque. Le paramétrage du système est défini de manière manuelle par les concepteurs, qui doivent établir les règles d'un système expert à chaque utilisation sur un nouveau site. Cette tâche se révèle particulièrement lourde, puisqu'elle nécessite le déplacement et l'intervention des concepteurs sur les sites correspondants. De plus, les paramètres du système ne sont valables que pour un site précis et ne sont pas génériques. Ce système présente également un taux de fausses alarmes élevé du fait du grand nombre de perturbations qui peuvent survenir dans la scène enregistrée par les caméras.

Notre travail consiste à développer un module d'apprentissage automatique destiné à remplacer le système expert dont les règles ont été définies manuellement. L'utilisation d'algorithmes d'apprentissage permet d'avoir un outil utilisable sur différents sites. Pour installer le système sur un nouveau site, il suffit de relancer la procédure d'apprentissage sur les données recueuillies sur ce nouveau site. L'utilisation est également simplifiée puisque le système apprend lui-même à reconnaître la pollution sans nécessiter l'intervention d'un expert. Pour améliorer précision du système, il est possible de sélectionner le type de classifieur le plus fiable face aux fausses alarmes et de non-détections. Nous présentons dans le paragraphe suivant le module de classification développé dans le cadre de cette thèse.

5.2 Etude du module de classification proposé

La première étape dans la conception d'un module d'apprentissage consiste à réunir un ensemble d'entraînement Z tel qu'il a été formellement introduit au début de ce mémoire. Le système mis au point par la société ALOATEC réalise une surveillance continue de l'activité industrielle. L'isolation des événements ponctuels correspondant à des pics de pollution potentiels est réalisée par l'algorithme de pré-détection du système. Celui-ci attribue un début et une fin à ces événements. Cette étape, que nous qualifions de discrétisation de la surveillance, permet d'obtenir une base d'événements constituée de différents panaches de fumées dont la classe correspond au niveau de gravité attribué par l'opérateur humain. Une seconde étape, dite de discrétisation

des données, est nécessaire pour obtenir un ensemble d'apprentissage sous sa forme usuelle. En effet, à l'issue de la première étape, chaque exemple est caractérisé par un ensemble de signaux temporels représentés par des courbes caractérisant l'évolution de la densité, de la forme, ou de la surface du panache Elles sont mesurées entre le début et la fin de l'événement. Pour appliquer les algorithmes de classification présentés au cours des chapitres précédents, il est nécessaire de disposer de variables non temporelles, qu'elles soient discrètes ou continues. Cette étape, que nous qualifions de discrétisation des signaux numériques, est imposée par ALOATEC de manière à associer un sens physique aux signaux discrets obtenus. Par exemple, la discrétisation de la densité est réalisée par la moyenne des valeurs observées au cours du rejet. Pour la couleur des fumées, c'est la valeur maximale sur le diagramme cholorimétrique qui est considérée. La figure 5.8 présente un exemple de cette discrétisation des signaux numériques.

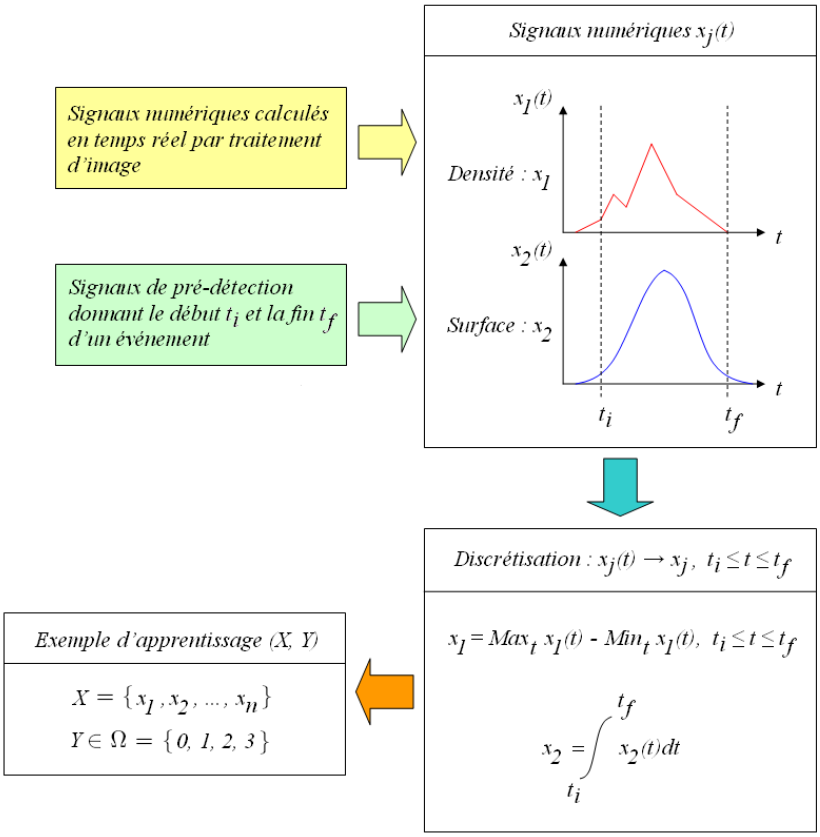

FIG. 5.8 – Schématisation de l'étape de discrétisation des signaux temporels. Deux signaux, arbitrairement associés à la densité et la surface d'un panache de fumée dans notre exemple, sont convertis de manière à modéliser cet événement sous la forme d'un exemple d'apprentissage utilisé pour réaliser l'apprentissage d'un classifieur.

A partir des signaux temps réels calculés par traitement d'image et des signaux de pré-détection permettant de définir le début et la fin d'un événement, des opérateurs prédéfinis effectuent une convertion de l'ensemble des événements contenus dans une base de données provenant du terrain en un ensemble d'apprentissage utilisable dans le cadre de la classification supervisée des fumées.

La mise en place de ce module d'apprentissage s'effectue en deux temps. L'apprentissage d'un

classifieur à partir d'un ensemble d'événements antérieurs s'effectue hors ligne (*offline*). Cette étape peut demander un temps relativement important, en fonction du type de classifieur utilisé, de ses paramètres, et du nombre d'événements utilisés pour réaliser l'apprentissage mais elle n'a besoin d'être effectuée qu'une seule fois. Cependant, si l'on dispose de nouvelles données (par exemple, l'ajout d'une nouvelle caméra ou de nouveaux signaux) cette opération doit être réalisée à nouveau pour mettre à jour le classifieur.

L'utilisation de ce classifieur pour déterminer en temps réel le niveau de gravité des fumées s'effectue en ligne. Lorsque les signaux de pré-détection isolent un événement à risque, un composant spécifique développé dans le cadre cette thèse effectue la discrétisation des signaux calculés au cours de cet événement. Le classifieur détermine ensuite le degré de gravité de cet événement. La classifieur utilisé en temps réel par le système doit avoir un temps de réponse court. Cette contrainte peut soulever certains problèmes selon le type de classifieur sélectionné. Par exemple, dans le cadre d'un classifieur basé sur les k plus proches voisins lorsque la base d'apprentissage comporte un grand nombre d'événements, le temps requis pour classer un nouvel événement peut être trop long.

La structure du système de surveillance basé sur le module de classification développé dans le cadre de cette thèse est donné dans la figure 5.9.

L'ajout d'un module d'apprentissage au système de surveillance existant permet de simplifier grandement son paramétrage. En effet, la définition manuelle des règles de décision est une étape particulièrement lourde pour les concepteurs, qui doivent se déplacer sur le site industriel et consacrer un temps long pour le paramétrage de la détection. L'apprentissage du classifieur est une étape indépendante, qui ne requiert pas de connaissance *a priori* sur l'activité observée, et qui peut être réalisée indépendamment par l'opérateur ou les concepteurs du système. La mise en place de ce module d'apprentissage suppose cependant que l'on dispose d'un nombre suffisant d'événements significatifs pour construire un classifieur. Pour valider notre travail sur le secteur industriel, nous avons testé ce module avec différents classifieurs et nous avons comparé les résultats obtenus avec les taux d'événements correctement classés par le système initialement développé par ALOATEC. Nous consacrons la section suivante aux résultats obtenus sur des données industrielles par différents algorithmes issus de l'Apprentissage Automatique.

5.3 Etude expérimentale

5.3.1 Protocole utilisé

La base d'apprentissage fournie par la société ALOATEC est constituée d'environ 2900 événements enregistrés durant une période de plusieurs mois sur le site d'une société sidérurgique. Cette base d'apprentissage ne comporte pas de valeurs manquantes. Un événement correspond à un rejet de fumée dont la classe ou niveau de gravité a préalablement été établi par un expert humain. Chaque événement est caractérisé par 12 attributs numériques et une variable entière correspondant à son niveau de gravité. Les variables numériques correspondent aux signaux calculés par le module de traitement d'image, tels que la densité, la forme et la surface du panache de fumée. Le détail de ces 12 attributs numériques ne peut pas être donné pour des raisons de confidentialité.

La classe de chaque événement est donnée par le niveau de gravité, une variable discrète à valeur dans $\Omega = \{0, 1, 2, 3\}$, caractérisant respectivement les pollutions très faibles ou inoffensives telles que les rejets de vapeur d'eau et les événements critiques donnant lieu le plus souvent à un signal d'alerte.

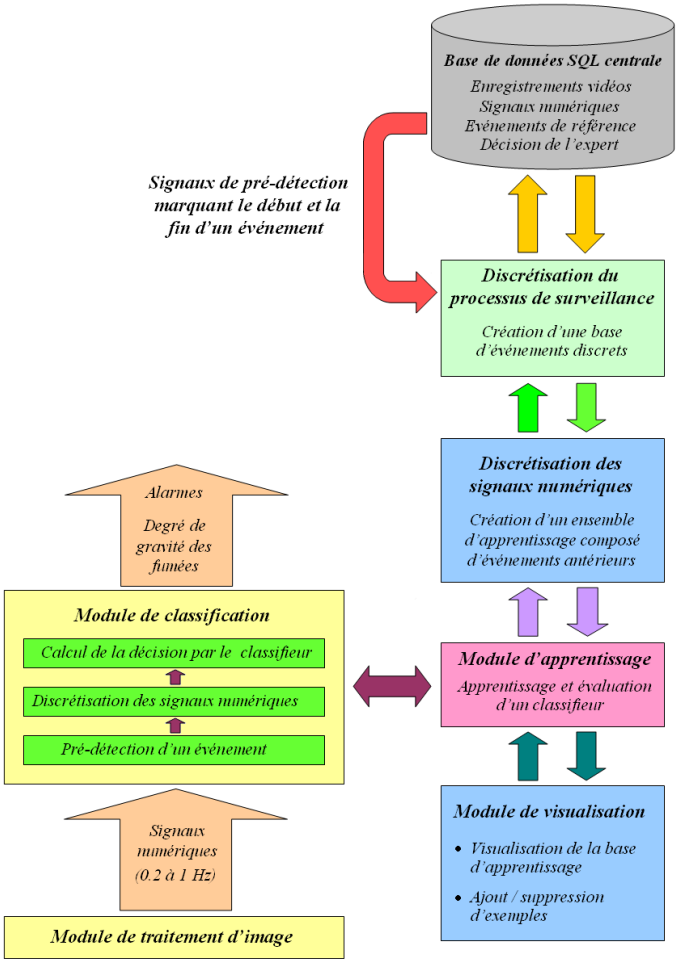

FIG. 5.9 – Module de classification développé dans le cadre de cette thèse pour le système de surveillance des fumées polluantes. Les règles de décisions définies manuellement sont remplacées par l'utilisation d'un classifieur préalablement construit par apprentissage, à partir d'événements antérieurs.

Les différentes classes du problème ne sont pas distribuées de manière identique dans la base fournie, le nombre d'événements de niveau 0 étant relativement important. La distribution des classes pour la base de données utilisée est donnée dans la table 5.1. Le faible nombre de rejets critiques rend l'apprentissage plus difficile par rapport aux pollutions de moindre importance, plus nombreuses.

De plus, les données fournies par la société ALOATEC présentent une quantité de bruit non négligeable, due à un désaccord entre le système de surveillance et le superviseur humain survenant lors de l'attribution d'une classe pour certains événements. Le module de pré-détection décrit précédemment dans ce chapitre isole des événements discrets dont la durée ne peut excéder cinq minutes. Si nous considérons un rejet critique, de niveau 3, d'une durée de 12 minutes, il est subdivisé par le module de pré-détection en trois événements successifs, de durées respectives égales à cinq, cinq et deux minutes. Le module de classification comptabilise alors trois événements distincts de classe 3. Cependant, le superviseur affecté à la surveillance des écrans de

TAB. 5.1 – Distribution des classes dans la base d'événements fournie par ALOATEC

	Niveau 0	Niveau 1	Niveau 2	Niveau 3
Nombre d'événements	2085	421	143	235
Proportion	72 %	15 %	5 %	8 %

contrôle considère qu'il n'y a eu qu'un seul événement de niveau 3, et *dégrade* les deux événements restants en niveau 0. La présence de ces objets mal classés dans la base d'apprentissage rend la construction du classifieur plus difficile. Tout rejet de fumée volontairement *dégradé* par l'opérateur se comporte comme un événement aberrant et génère du bruit dans la base d'apprentissage, ce qui perturbe le calcul des frontières de décisions.

Certains événements étiquetés de manière incorrecte par le système ont une incidence plus importante sur son utilisabilité dans des situations réelles. Les fausses alarmes (événements de niveau 0 identifiés par le système comme étant de niveau 3) et les non-détections (événements critiques de niveau 3 étiquetés en tant que niveau 0) ont une influence directe pour des applications industrielles. Une fausse alarme se traduit généralement par un ralentissement non justifié, voir une interruption, de l'activité industrielle. Les non-détections quant à elles limitent l'évaluation des risques de l'activité pour l'environnement et la population environnante. A l'inverse, si le système identifie un événement de niveau 1 comme étant de niveau 0, les répercussions de cette erreur sont moindres. De manière à prendre en compte ce phénomène, nous utilisons un critère d'évaluation appelé *Efficacité*. Ce critère est défini par la société ALOATEC et ses partenaires industriels pour caractériser la robustesse du système vis à vis des non-détections et fausses alarmes. Il est donné par :

$$Efficacite = \frac{0.3 \cdot (N_{13} + N_{31}) + 0.8 \cdot (N_{23} + N_{32}) + N_{33}}{\sum_{i=0}^{3} N_{i3} + \sum_{j=0}^{2} N_{3j}} \tag{5.1}$$

où N_{ij} désigne le nombre d'exemples de classe i pour lesquels le système a attribué l'étiquette j. Ce critère peut être vu comme la précision du système pondérée par l'importance de l'événement détecté, à l'image d'un problème de classification binaire où les faux négatifs sont plus pénalisant que les faux positifs. Pour des applications médicales par exemple, diagnostiquer la présence d'une tumeur cancéreuse pour un patient sain (faux positif) présente des conséquences beaucoup moins lourdes que de ne pas détecter la tumeur sur une personne atteinte d'un cancer (faux négatif).

5.3.2 Influence des paramètres des classifieurs sur la qualité de la détection

Comme nous l'avons vu dans le second chapitre de ce mémoire, il n'est pas possible, d'un point de vue général, de dire qu'un algorithme de classification est meilleur qu'un autre. Chaque classifieur possède un certain nombre de paramètres ayant une grande influence sur les performances obtenues. Le choix de ces différents paramètres dépend du problème posé, et bien que l'utilisation d'un ensemble d'arbres de décision obtenu à l'aide de l'algorithme Adaboost.M1 soit considéré comme la meilleure solution *clé en main* [85], ce choix peut se révéler inadapté lorsque les données sont fortement bruitées.

Cette dernière considération nous amène à nous interroger sur certains facteurs inhérents au problème en lui-même, et non au type de classifieur choisi. Prenons par exemple la base *iris* de l'UCI Repository, fréquemment citée dans la littérature. Le nombre d'exemples étant

relativement limité (150), les résultats obtenus par une validation croisée en 10 blocs sont calculés sur seulement 15 échantillons de test. Dans une telle configuration, attribuer la mauvaise classe à un seul exemple se traduit par une variation de la précision de l'ordre de 6.6%. Dietterich souligne ce problème en qualifiant la base iris d'instable [138]. Pour ce type de base, le choix du meilleur classifieur peut se révéler difficile en raison des variations des performances qui ne dépendent pas de l'algorithme choisi, mais de phénomènes externes tels que le partitionnement entre données d'apprentissage et de test.

Dans le cas particulier des données concernant la détection de fumées polluantes, nous cherchons à déterminer le classifieur le mieux adapté vue la distribution très inégale des classes et la présence de bruit.

De manière à pouvoir comparer les performances des différents algorithmes d'apprentissage présentés dans ce mémoire pour la classification des fumées, il est nécessaire de déterminer leurs paramètres optimaux à travers une première étape de validation. Nous faisons la distinction entre les paramètres définis par l'utilisateur lors de la conception du classifieur, les paramètres déterminés au cours d'une étape de validation croisée et ceux calculés durant l'apprentissage.

Pour certains algorithmes, le nombre de paramètres peut devenir très important. Nous limitons notre étude en imposant certains choix concernant la structure des classifieurs utilisés. La table 5.2 liste pour chacun des algorithmes utilisés les paramètres que nous avons définis manuellement.

Tab. 5.2 – Liste des paramètres définis manuellement.

Algorithme de classification	**Paramètres définis manuellement**
k-PPV	distance Euclidienne
Réseau Bayésien Naïf	graphe en étoile noeuds discrets
Arbre de décision	Eclatement selon le critère de Gini Eclatement à partir de 10 exemples minimum
Réseau de neurones	réseau sans rétroaction une couche cachée un neurone de sortie par classe apprentissage : algorithme de Levenberg-Marquardt
SVM	noyau Gaussien décomposition multi-classes *One Versus One*
Modèle Flou	prémisses : fonctions d'appartenance gaussiennes conséquences : fonctions affines

Nous donnons également dans la table 5.3 les paramètres déterminés au cours de l'apprentissage en fonction du classifieur utilisé. Rappelons que l'algorithme k-PPV ne réalise pas d'apprentissage à proprement parlé.

Nous discutons à présent du choix des paramètres des différents classifieurs que nous déterminons par validation croisée. Pour les classifieurs avec un seul paramètre à déterminer, les figures 5.10, 5.11 et 5.13 présentent l'évolution moyenne de la précision et de l'efficacité ainsi que son écart type. Le réseau bayésien naïf ne requiert pas de sélection de paramètres. La structure utilisée est la structure en étoile décrite dans la figure 2.5 et les différents attributs sont discrétisés en utilisant la méthode présentée dans [153] et appliquées aux réseaux bayésiens par Leray et François [124]. L'apprentissage des tables de probabilités de chaque nœud est réalisé par maximum de

TAB. 5.3 – Paramètres déterminés au cours de l'apprentissage.

Algorithme de classification	**Paramètres appris**
Réseau Bayésien Naïf	Distributions de probabilités $P(x_i \mid \omega = \omega_j)$
Arbre de décision	Structure de l'arbre
Réseau de neurones	Poids synaptiques w_{ij}
SVM	Vecteurs de supports
Modèle flou	Premisses : moyenne ($\bar{x}$) et écart type (σ) des fonctions d'appartenance Conséquences : coefficients a_{ij} et biais b_i des fonctions de sortie $f_i(X)$

vraisemblance [151], les données fournies ne comportant aucune valeur manquante. L'étiquetage d'un nouvel exemple est réalisé en lui attribuant la classe dont la probabilité *a posteriori* est maximale. L'absence de sélection de paramètre concerne également les arbres de décision. Nous utilisons le critère de Gini pour l'éclatement des nœuds et nous appliquons une étape d'élagage une fois l'arbre construit.

Les résultats présentés dans la figure 5.10 donnent la précision et l'efficacité de l'algorithme k-PPV en fonction du nombre k de voisins considérés. Une valeur trop faible pour k diminue la précision obtenue, du fait de l'influence des événements dégradés dans la base d'apprentissage. Pour un nombre de voisins supérieur à 10, l'influence du bruit est diminuée et la précision obtenue varie peu. Pour un nombre de voisins supérieur à 15 cependant, la précision n'augmente plus puis diminue à partir de $k = 30$ car les frontières de décision deviennent imprécises. Cette tendance est accentuée sur les variations de l'efficacité. Pour une valeur de k supérieure à 15, l'efficacité diminue progressivement. Nous justifions ces observations par le fait que le nombre d'événements de niveau 3 est faible dans la base d'apprentissage par rapport aux autres classes. En prenant une valeur de k trop importante, il est alors plus difficile de faire apparaître une majorité d'événements de cette classe pour étiqueter un nouvel exemple lui appartenant. L'efficacité étant calculée principalement sur l'identification des pollutions les plus importantes en terme de risque, elle est plus sensible au choix du paramètre k. Par la suite, nous considérons qu'une valeur de k égale à 15 offre le meilleur compromis entre précision et efficacité.

Le réseau de neurones utilisé est un perceptron multi-couches tel qu'il est décrit dans la figure 2.9. L'apprentissage des réseaux de neurones peut être influencé par un grand nombre de paramètres. C'est pourquoi nous limitons notre étude aux réseaux sans rétroaction comportant une seule couche cachée, dont nous faisons varier le nombre d'unités entre 5 et 35 de manière à étudier l'influence de la compléxité du réseau. Les poids sont initialisés aléatoirement et l'apprentissage du réseau est réalisé en appliquant l'algorithme de Levenberg-Marquardt [80] basé sur la rétropropagation du gradient. De manière à éviter un sur-apprentissage du réseau, nous utilisons un tiers des exemples d'entraînement en tant qu'ensemble de validation. Les variations de la précision et de l'efficacité obtenues sont décrites la figure 5.11.

Pour notre application, le nombre d'unités cachées a peu d'influence sur la précision et l'efficacité du réseau de neurones. En augmentant le nombre d'unités cachées, il est possible d'améliorer légèrement les performances obtenues. Au delà de 15 unités dans la couche cachée cependant, les performances diminuent car la structure du réseau devient trop complexe. Nous constatons qu'un réseau comportant 15 unités dans la couche cachée permet d'obtenir les performances optimales

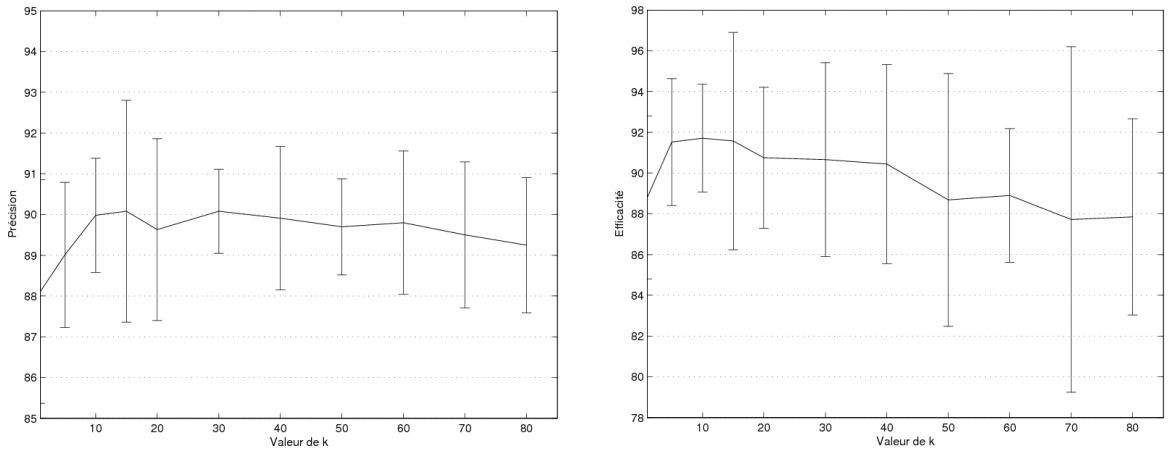

FIG. 5.10 – Influence du nombre de voisins considérés sur la précision du classifieur k-PPV (à gauche) et sur son efficacité (à droite).

pour la détection des fumées.

Les SVM à noyaux gaussiens dépendent de deux paramètres : le coût de dépassement des contraintes du problème d'optimisation, noté C, et le paramètre du noyau gaussien σ. L'influence de ces paramètres sur les performances obtenues est décrite dans la figure 5.12.

Contrairement à l'algorithme des k plus proches voisins, où le paramètre k possède peu d'influence sur les performances, les résultats obtenus par l'utilisation de SVM gaussiens sont sensibles au choix du paramètre σ. Une valeur inadaptée de ce paramètre entraîne une diminution de la précision de près de 20% et peut rendre l'efficacité quasiment nulle. Des observations similaires sont décrites par Li *et al.* [125], qui tirent parti de cette sensibilité des SVM gaussiens au paramètre σ pour l'utiliser conjointement à l'algorithme Adaboost. En revanche, le coût C de dépassement des contraintes a moins d'influence sur les résultats obtenus. Augmenter la valeur de C permet d'obtenir un hyperplan séparateur plus général et diminue l'influence du bruit contenu dans les données. Cependant, au delà d'un certain seuil, la valeur de C augmente beaucoup le temps requis pour déterminer les vecteurs de support sans pour autant apporter d'amélioration significative au niveau des performances. Ceci est dû au fait que le problème d'optimisation permettant de déterminer l'hyperplan séparateur est plus difficile à résoudre. Nous constatons que le meilleur compromis est obtenu en fixant la valeur de σ à 10^{-9} avec une valeur de C égale à 10^5.

Le modèle flou dépend notamment du nombre de règles utilisées. Les courbes présentées dans la figure 5.13 décrivent l'influence de ce paramètre sur sa précision et son efficacité. De manière générale, un nombre de règles trop important diminue les performances obtenues. Toutefois, les valeurs obtenues présentent une variance importante même en utilisant la validation croisée. Nous constatons que les meilleures performances sont obtenues par un modèle constitué de 7 règles.

Nous terminons cette partie concernant la sélection de paramètres en récapitulant dans la table 5.4 les paramètres retenus pour chaque classifieur. Dans la partie suivante, nous comparons les différents algorithmes de classification présentés dans ce mémoire pour la détection des fumées en utilisant ces valeurs de paramètres.

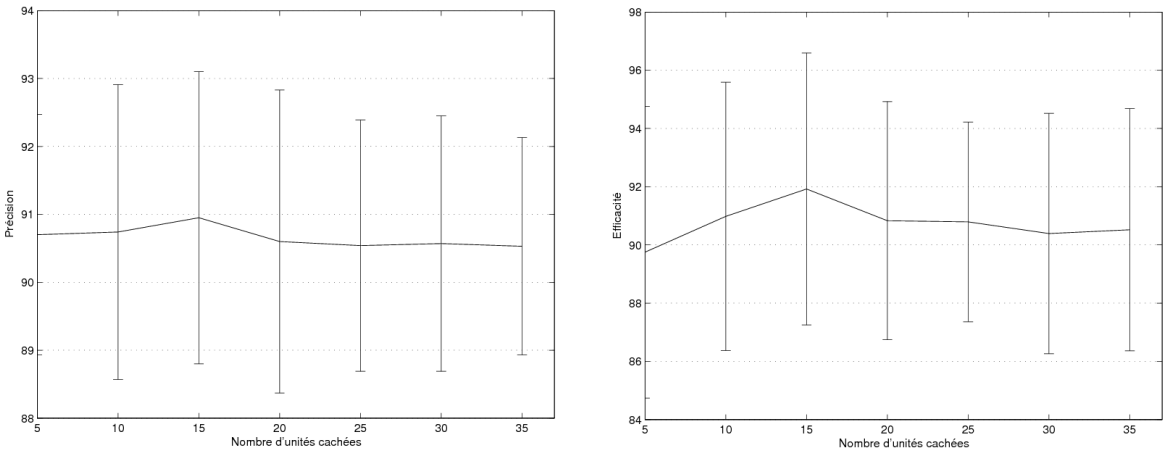

FIG. 5.11 – Influence du nombre d'unités cachées sur la précision du réseau de neurones sans rétroaction (à gauche) et sur son efficacité (à droite).

TAB. 5.4 – Paramètres déterminés par validation croisée.

Algorithme de classification	Paramètres	Valeur
k-PPV	nombre de voisins k considérés	15
Réseau de neurones	nombre d'unités cachées	15
Modèle Flou	nombre de règles du modèle	7
SVM	coût de dépassement C des contraintes	10^5
	paramètre du noyau gaussien σ	10^{-9}

5.3.3 Performances des algorithmes de classification usuels pour la surveillance des fumées

Nous nous intéressons à présent aux performances des différents algorithmes présentés dans ce mémoire appliqués à la détection des fumées. Pour chaque algorithme de classification, les paramètres utilisés sont ceux reportés dans la table 5.4 et donnant les meilleurs résultats pour le problème considéré. Les valeurs reportées dans la table 5.5 ont été calculées en effectuant la moyenne des taux de précision et d'efficacité obtenues sur 5 validations croisées en 10 blocs. Dans le cas du système expert initialement développé par la société ALOATEC, les valeurs de précision et d'efficacité ont été calculées sur la matrice de confusion opposant la décision de l'expert et celle du système d'origine, basé sur un système de règles définies manuellement, sur la totalité des événements disponibles.

En terme de précision, le réseau de neurones utilisé est le classifieur qui présente le meilleur taux de reconnaissance. Nous expliquons ce résultat par le nombre important d'exemples disponibles qui nous permet de disposer d'un ensemble de validation suffisant pour éviter une situation de sur-apprentissage de ce classifieur.

Remarquons que le réseau bayésien naïf, qui semble assez mal classé en terme de précision, est au contraire le classifieur qui présente la meilleure efficacité sur l'ensemble des tests effectués. Ce type de classifieur semble bien adapté au contexte de la classification des fumées pour lequel les données sont bruitées et les classes très inéquitablement représentées.

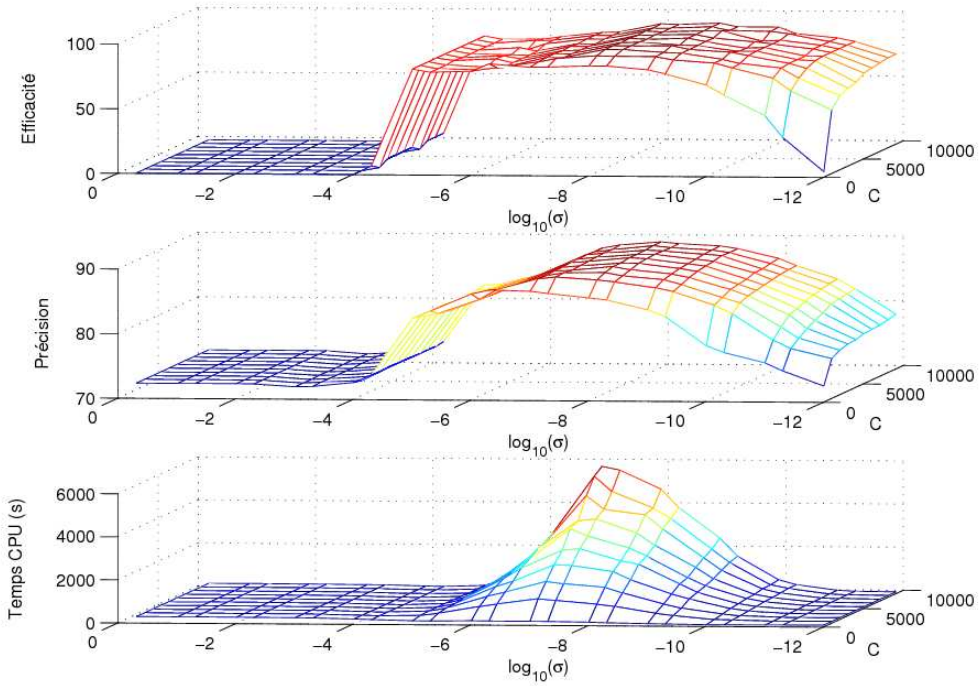

FIG. 5.12 – Influence des paramètres des SVM à noyau gaussien sur la précision, l'efficacité et la durée requise pour identifier les vecteurs de support.

D'un point de vue général, les résultats obtenus par les différents algorithmes de classification sont proches mis à part pour le modèle flou utilisé qui présente des performances légèrement inférieures. La différence avec le système initialement conçu par ALOATEC est cependant beaucoup plus importante, surtout en terme d'efficacité. Ceci montre que les pollutions importantes sont mieux classées par le système et que le nombre de fausses alarmes émises par le système est grandement réduit.

Nous donnons également la précision et l'efficacité obtenues par aggrégation de classifieurs selon les méthodes du Bagging et du Boosting. Pour ces deux algorithmes ensemblistes, les classifieurs de base utilisés sont des arbres de décisions. Dans le cas du Boosting, nous utilisons l'algorithme Adaboost.M1. Pour le Bagging, les différents arbres de décisions sont entrainés sur

TAB. 5.5 – Performances des algorithmes utilisés pour la classification des fumées.

Algorithme de classification	Précision	Efficacité
k-PPV	89.88 ± 1.89	91.34 ± 4.00
Réseau Bayésien Naïf	88.40 ± 1.57	**92.20 ± 4.13**
Arbre de décision	89.15 ± 1.59	91.64 ± 4.00
Réseau de neurones	**91.06 ± 1.63**	91.50 ± 3.77
SVM	88.87 ± 1.80	91.13 ± 4.50
Modèle flou	88.09 ± 1.79	88.66 ± 4.67
Système original ALOATEC	70.90	48.00

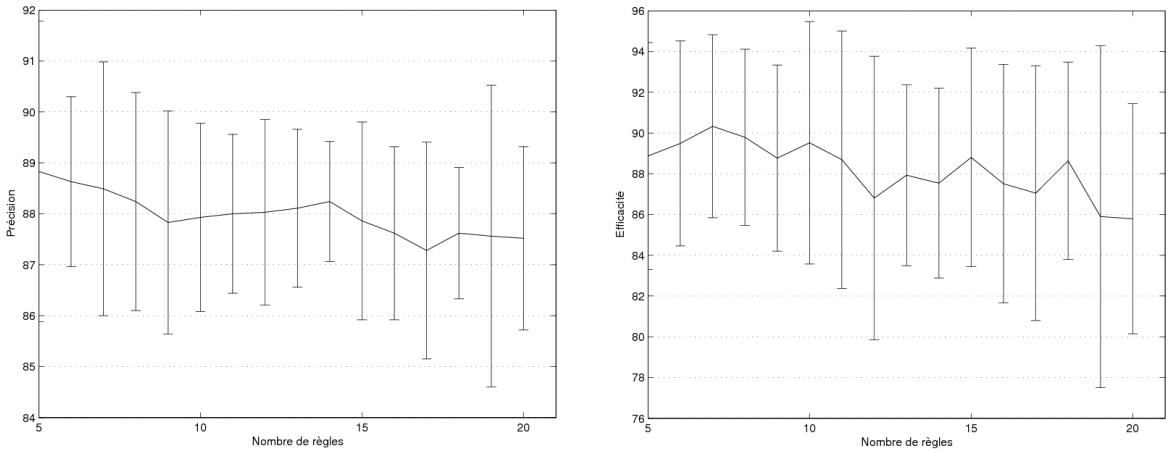

FIG. 5.13 – Influence du nombre de règles utilisées sur l'efficacité du modèle flou (à gauche) et sur son efficacité (à droite).

des rééchantillonnages aléatoires dont la taille est égale à 75% de l'ensemble d'apprentissage initial. L'agrégation des sorties des différents classifieurs est réalisée par vote majoritaire. Les taux de précision et d'efficacité reportés dans la table 5.6 ont été calculés comme précédemment en effectuant la moyenne des performances obtenues sur 5 validations croisées en 10 blocs.

TAB. 5.6 – Performances obtenues par agrégation de classifieurs pour la classification des fumées. Les chiffres entre parenthèses correspondent à la taille de l'ensemble utilisé.

Algorithme Ensembliste	Précision	Efficacité
Adaboost.M1 (10)	90.07 ± 1.74	92.88 ± 3.36
Adaboost.M1 (20)	90.15 ± 1.64	92.81 ± 3.29
Adaboost.M1 (30)	90.04 ± 1.72	92.36 ± 3.84
Adaboost.M1 (40)	90.17 ± 1.74	92.73 ± 3.66
Adaboost.M1 (50)	90.35 ± 1.72	92.99 ± 3.41
Bagging (10)	90.83 ± 1.59	92.65 ± 3.27
Bagging (20)	90.81 ± 1.64	92.35 ± 4.04
Bagging (30)	91.01 ± 1.66	92.77 ± 3.06
Bagging (40)	91.05 ± 1.56	92.56 ± 3.19
Bagging (50)	$\mathbf{91.16 \pm 1.85}$	$\mathbf{93.01 \pm 3.39}$

La présence de bruit dans les données d'apprentissage, due à l'intervention de l'opérateur qui diminue le degré de gravité des événements critiques consécutifs en considérant qu'il s'agit d'un même et unique événement, limite le gain de performances obtenu en utilisant l'algorithme Adaboost.M1. En effet, pour des ensembles de taille similaire, l'agrégation de classifieurs obtenue par bagging donne des performances supérieures à celle de l'ensemble construit par Adaboost.M1, aussi bien en termes de précision que d'efficacité. Si nous nous référons à la table 5.5, nous constatons cependant que l'utilisation d'un ensemble d'arbres de décision permet d'améliorer la précision de 1% dans le cas du Boosting et de 1% à 2% dans le cas du Bagging par rapport à l'utilisation d'un arbre de décision unique.

L'utilisation des méthodes de l'Apprentissage Automatique permet d'améliorer la précision

de détection des fumées de 19% environ par rapport au système d'origine. La différence notable se situe au niveau de l'efficacité. En effet, le système de détection original présente un grand nombre de fausses alarmes, qui sont en grande partie corrigées en utilisant le module d'apprentissage présenté dans ce chapitre. De manière à illustrer cette observation, nous donnons dans la table 5.7 les matrices de confusion permettant de comparer les classes attribuées par le système original et un ensemble de 50 arbres de décisions obtenus par bagging.

Tab. 5.7 – Matrices de confusion obtenues par le système original et un ensemble de 50 arbres de décisions construits par bagging.

Système original ALOATEC

Superviseur				
	59.60 %	1.19 %	2.84 %	6.00 %
	9.60 %	6.52 %	1.01 %	0.37 %
	2.48 %	0.69 %	1.47 %	4.27 %
	0.37 %	0.01 %	1.83 %	3.35 %

Bagging (50)

Superviseur				
	68.00 %	2.08 %	0 %	0 %
	0.97 %	14.42 %	0 %	0 %
	0.42 %	3.74 %	1.39 %	0.82 %
	0.14 %	0.41 %	0.41 %	7.77 %

Remarquons que les chiffres reportés dans ces matrices sont données sous forme de pourcentages car dans le cas du système d'origine, qui ne réalise aucun apprentissage, tous les événements sont utilisés en tant qu'exemples de test. Les résultats reportés dans la table 5.7 montrent une très nette diminution du nombre de fausses alarmes, et une meilleure détection des événements de niveau 3. Cependant, la distinction entre les événements de niveau 1 et 2 est plus difficile pour le classifieur car le nombre d'exemples de niveau 2 est faible.

Les performances des différents algorithmes présentés dans cette section étant relativement proches, le choix d'un classifieur dans le cadre de la détection des rejets de fumées relève davantage des contraintes propres à l'application. En effet, les expériences décrites dans cette section ont été réalisées avec une base d'apprentissage relativement importante, comportant un grand nombre d'événements enregistrés sur une période de plusieurs mois. Même si les rejets de fumées de niveau 0 sont prédominants dans cette base d'événements, le nombre d'exemples pour chaque classe est suffisant pour éviter une situation de sous-apprentissage du classifieur. Lorsque le système est déployé sur un nouveau site industriel, le nombre d'événements dans la base d'apprentissage est très limité, ce qui peut rendre l'estimation des densités de probabilités difficiles si l'on choisit d'utiliser un classifieur Bayésien Naïf par exemple. Pour ce type de situation, un classifieur basé sur les plus proches voisins est davantage conseillé. A l'inverse, lorsque la base d'événements est importante, il est préférable de choisir un classifieur dont l'apprentissage est réalisé de manière hors ligne. Le classifieur k-PPV ne réalise pas d'apprentissage proprement dit, mais attribue en temps réel une étiquette à chaque exemple en effectuant des comparaisons avec chaque événement de la base d'apprentissage. La solution des k-PPV est donc à éviter dans le cas d'une utilisation en temps réel, plus particulièrement si la base d'apprentissage est conséquente. Une bonne solution consiste à utiliser un réseau bayésien naïf ou un ensemble de classifieurs, qui présentent un fort taux de reconnaissance des pollutions importantes.

5.3.4 Utilisation de la diversité pour la classification des fumées dangereuses

Nous terminons notre étude de la classification des rejets de fumées polluantes en comparant les performances obtenues par les différentes heuristiques de sélection de classifieurs utilisées dans le chapitre précédent. Nous répartissons les exemples disponibles en trois parties dédiées respectivement à la construction de l'ensemble initial d'arbres de décision (1889 exemples), à la sélection des classifieurs (500 exemples) et à l'évaluation de la précision des ensembles obtenus (500 exemples). Nous calculons le gain relatif moyen obtenu par les différentes heuristiques en répétant aléatoirement ce découpage des données 50 fois. Les courbes et l'histogramme présentant les résultats obtenus sont donnés dans la figure 5.14.

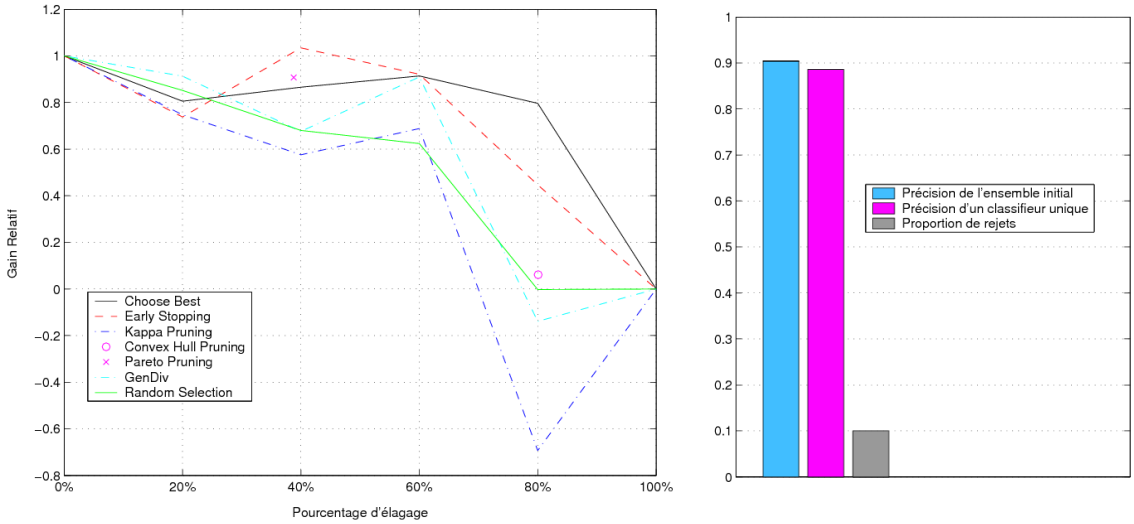

FIG. 5.14 – Permormances des différentes heuristiques de sélection de classifieurs pour la classification des rejets de fumées polluantes.

De manière générale, la répartition inégale des classes, le faible gain absolu obtenu en moyenne (1.8%) ainsi que la présence de bruit dans les données se répercutent en terme d'instabilité sur les résultats obtenus. Pour un niveau d'élagage faible (20%), la sélection de classifieurs par l'approche génétique proposée offre un gain relatif supérieure aux autres méthodes de sélection. Au delà de cette valeur, nous observons une dégradation rapide de la précision lorsque l'on diminue le nombre de classifieurs sélectionnés. Cette diminution de la précision est cependant beaucoup moins importante dans le cas de l'heuristique *Choose Best*, comme nous avons pu l'observer au chapitre précédent sur plusieurs benchmarks provenant de l'UCI Repository.

Nous observons que pour ce problème, la diversité constitue un mauvais critère de sélection car pour un ensemble de taille donnée, l'heuristique *Kappa Pruning* est celle donnant les plus mauvais résultats en terme de gain. Pour un niveau d'élagage de 80%, nous constatons que l'application de l'implémentation génétique proposée se traduit par un ensemble moins précis qu'un classifieur unique. Ce classifieur unique est d'ailleurs pratiquement aussi performant que l'ensemble obtenu par sélection des classifieurs appartenant à l'enveloppe convexe.

Pour le problème de la classification des fumées, le nombre d'événements de niveau 3 est relativement faible. De plus, parmi ces exemples de niveau 3, il existe une part non négligeable d'événements mal classés par l'opérateur, qui diminue le niveau de gravité des événements consécutifs de manière à n'en comptabiliser qu'un seul. Certains des classifieurs individuels construits

par l'algorithme Adaboost vont concentrer leur apprentissage sur ces événements mal classés, également appelés *outliers*, et se comporter de manière très diverses vis à vis des classifieurs entrainés sur les bons exemples. En nous basant sur la diversité, nous nous exposons donc à sélectionner ces mauvais classifieurs. L'usage de la diversité pour ce problème est donc fortement déconseillé.

En utilisant l'heuristique *Choose Best*, l'approche Surproduction et Sélection présente cependant l'avantage pratique de diminuer le nombre de classifieurs requis pour la surveillance en ligne des fumées. En effet, il est possible en appliquant cette heuristique d'élaguer à 60% l'ensemble initial tout en conservant 90% du gain relatif, ou 80% si on ne sélectionne que les 10 classifieurs possédant la plus grande précision individuelle.

5.4 Conclusion du chapitre

Dans ce chapitre, nous avons présenté une application des méthodes de l'apprentissage automatique pour la classification de la pollution atmosphérique. Les travaux de recherche présentés répondent à deux principaux objectifs : améliorer la qualité des détections effectuées par le système d'origine, qui présente un nombre important de fausses alarmes et de non-détections, et rendre le paramétrage de ce système autonome vis à vis de ses concepteurs.

Le module d'apprentissage proposé dans ce chapitre rend possible l'utilisation du système de surveillance des fumées indépendamment de l'intervention des concepteurs, tout en offrant des performances supérieures à celles du système de détection original. Par rapport au module de classification initialement développé par la société ALOATEC, nous pouvons constater une amélioration significative de la qualité des détections, et ce quel que soit le type de classifieur choisi. Cependant, les résultats obtenus à l'aide des différents algorithmes utilisés sont relativement proches et il est difficile de dire, du point de vue des performances uniquement, qu'un classifieur est meilleur que les autres.

Si nous nous intéressons aux performances obtenues par agrégation de classifieurs, nous pouvons remarquer que la présence de nombreux exemples dégradés (de niveau 3 en niveau 0) dans la base d'apprentissage diminue les facultés prédictives des ensemble construits par application de l'algorithme Adaboost.M1. Les résultats obtenus sont donc en accord avec les différentes études relatives au boosting présentées dans [40, 42, 196]. Néanmoins, l'utilisation d'un ensemble de classifieurs permet d'améliorer les résultats obtenus comparativement à un classifieur unique, à la fois en termes de précision et d'efficacité.

La discussion relative au choix d'un classifieur pour la détection des rejets de fumée est donc dépendante de plusieurs facteurs. L'industriel privilégie la réduction du nombre de fausses alarmes et de non-détections, illustrée par la valeur d'efficacité de la solution sélectionnée. Les valeurs les plus importantes pour ce critère sont obtenues par construction d'un ensemble de classifieurs, bagging et boosting offrant des résultats similaires en terme d'efficacité, bien que la précision obtenue par bagging soit supérieure. Remarquons également que l'efficacité étant liée aux événements critiques, peu nombreux dans la base d'apprentissage fournie, cette dernière possède une variance plus importante que la précision sur l'ensemble des tests effectués.

Un autre facteur important concernant le choix d'un algorithme de classification pour l'application proposée est le temps requis pour déterminer la classe d'un nouvel événement. Mis à part pour le classifieur k-PPV, l'apprentissage s'effectue hors ligne. Le classifieur est ensuite mis en ligne de manière à déterminer la gravité des différents événements en temps réel. L'utilisation des k-PPV dans le contexte de la détection des fumées implique que le superviseur effectue des opérations de filtrage régulières sur la base d'événements enregistrés, de manière à limiter leur

nombre tout en conservant les plus caractéristiques d'entre eux.

Le dernier point à considérer concerne la paramétrisation des différents classifieurs. Les performances des SVM gaussiens sont sensibles au choix du paramètre σ comme le montrent les expériences décrites dans ce chapitre. Pour un superviseur ne disposant pas de notions en Apprentissage Automatique, il est préférable de proposer une solution *clé en main* directement exploitable à des fins industrielles, telle qu'un ensemble d'arbre de décisions ou un réseau bayésien naïf si on préfère utiliser un seul classifieur.

6

Conclusion et perspectives

Nous résumons à présent les travaux de recherche présentés dans ce mémoire sur l'Apprentissage Automatique et nous donnons les conclusions que nous en avons tirées. Nous présentons également plusieurs perspectives sur les travaux menés à propos de la diversité et sur l'application des algorithmes d'apprentissage à la classification des fumées.

6.1 Sur l'avancement de l'Apprentissage Automatique

Dans cette thèse, nous nous sommes intéressés à l'Apprentissage Automatique et plus particulièrement à la classification supervisée. Les recherches présentées dans ce mémoire ont permis de dresser un état de l'art non exhaustif des principales méthodes permettant de construire un classifieur à partir d'exemples. Nous nous sommes plus particluièrement intéressés à l'approche ensembliste du problème de l'Apprentissage Automatique, en étudiant comment l'utilisation d'un ensemble de classifieurs permet d'améliorer les résultats obtenus. Nous avons recensé les algorithmes ensemblistes les plus fréquemment cités dans la littérature et fait le point sur les contributions les plus récentes en la matière.

Nous avons également effectué une synthèse des travaux existants sur les différents aspects de la diversité dans les ensembles de classifieurs. L'objectif était de savoir comment mesurer la diversité à partir des différentes erreurs commises par les membres de l'ensemble et de savoir comment la générer. Pour cela, nous nous sommes appuyés sur la taxonomie proposée par Brown [19] et sur les algorithmes ensemblistes de la littérature.

Au terme de cette étude, certaines interrogations sont apparues. De nombreuses études empiriques montrent que les ensembles de classifieurs présentant une grande diversité, tels que ceux obtenus par l'algorithme Adaboost, sont généralement plus précis que les ensembles de classifieurs issus de techniques comme le bagging. Dans un contexte de régression, la décomposition de l'erreur permet même de justifier de manière théorique l'apport de la diversité, par exemple dans le cas de l'apprentissage par corrélation négative [20]. Cependant, lorsque l'on cherche à utiliser explicitement cette diversité dans le cas de la classification, les résultats obtenus par un ensemble de classifieurs en terme de précision ne sont pas toujours ceux attendus.

Nous nous posons donc la question de savoir s'il est possible de faire en sorte que la diversité, qui est utilisée implicitement par l'algorithme Adaboost, par exemple, puisse être utilisée explicitement pour guider la création d'un ensemble de classifieurs et améliorer ses performances.

Ce caractère *insaisissable* de la diversité, pour reprendre le terme utilisé par Kuncheva [113], nous a amené à étudier plus en détail le compromis existant entre la précision et la diversité dans un ensemble de classifieurs et à proposer un moyen permettant de comprendre dans quelle

mesure la diversité doit être explicitement utilisée durant le processus de création de l'ensemble.

6.2 Bilan et perspectives concernant le rôle de la diversité dans les ensembles de classifieurs

De manière à apporter des réponses concernant l'influence de la diversité sur les performances d'un ensemble de prédicteurs, nous avons proposé un algorithme génétique *GenDiv* pour la construction d'ensembles de classifieurs selon le paradigme Surproduction et Sélection. Cette approche consiste à construire un grand nombre de classifieurs élémentaires et à sélectionner individuellement les membres de l'ensemble à partir de différents critères prédéfinis. Ce choix a été effectué pour pouvoir étudier le compromis existant entre la précision et la diversité et repose sur une fonction de fitness paramétrable. Il apparait notamment deux familles principales d'algorithmes ensemblistes, selon qu'ils construisent l'ensemble de manière ascendante, c'est à dire par construction et ajout successifs des classifieurs dans l'ensemble, ou au contraire de manière descendante selon le paradigme Surproduction er Sélection. Nous nous sommes intéressés dans le cadre de cette thèse à la seconde famille d'algorithmes, en comparant différentes méthodes pour effectuer la sélection de classifieurs.

Nous avons choisi d'évaluer la diversité en utilisant la valeur du kappa entre deux classifieurs. L'heuristique génétique proposée est cependant indépendante de la mesure de diversité utilisée. De manière similaire, la sélection de classifieurs basée sur l'enveloppe convexe ou l'ensemble de Pareto est initialement proposée pour les diagrammes kappa-erreur. Des expériences analogues à celles présentées dans le chapitre 4 sont donc envisageables pour différentes mesures de diversité. Il faut cependant noter que les différentes mesures proposées dans la littérature ne quantifient pas la diversité de manière identique. Une diversité importante se traduit par une valeur du kappa faible, alors que si l'on utilise la mesure de désaccord, par exemple, la diversité entre deux classifieurs sera importante pour une valeur du désaccord proche de 1. Il est possible d'étendre les diagrammes kappa-erreur à différentes mesures de diversité. Dans ce cas, l'interprétation de ces diagrammes devra être revue. De même, la fonction de fitness utilisée dans l'approche génétique proposée devra être adaptée en conséquence. Le principe de sélection des classifieurs restera le même d'un point de vue général. La comparaison des précisions des ensembles de classifieurs obtenus pour différentes mesures de diversité constitue une perspective de notre travail.

La fonction de fitness utilisée dans l'implémentation proposée permet également de faire varier le compromis entre la précision et la diversité de l'ensemble. Dans le cadre de cette thèse, nous avons fixé le paramètre α de la fonction de fitness à 0.8 de manière à privilégier la précision de l'ensemble construit. Une poursuite des travaux est de faire varier ce paramètre, est d'observer son influence sur la qualité des résultats obtenus.

Une autre perspective intéressante serait de comparer les courbes obtenues dans le cas où l'ensemble initial de classifieurs est obtenu par bagging. Les classifieurs de base seraient alors beaucoup plus précis mais la diversité existante entre deux classifieurs serait globalement plus faible. Il serait alors intéressant de comparer la stabilité des courbes obtenues. Pour combiner les avantages du boosting et du bagging, il serait également intéressant de construire le pool de départ par des classifieurs issus à la fois de ces deux méthodes. La comparaison de différentes méthodes d'agrégation des classifieurs de base constitue une autre perspective de nos travaux.

Le problème auquel nous nous trouvons confrontés en adoptant une approche Surproduction et Sélection est que des données de validation (*pruning set*) sont requises pour effectuer la sélection des classifieurs. De manière générale, le gain de précision obtenu sur cet ensemble de validation se généralise peu ou mal sur des nouvelles données, comme le montrent les courbes 4.20

et 4.19 données à la fin du chapitre 4. Lorsque l'on se place dans le cas d'un classifieur unique, la précision obtenue sur des données de test permet d'avoir une bonne estimation des performances de ce classifieur sur de nouvelles données inconnues. En revanche, les bonnes performances d'un ensemble de classifieurs sur les exemples de validation ne permettent pas d'affirmer systématiquement que cet ensemble offrira une précision similaire sur de nouveaux exemples. Ce phénomène explique pourquoi l'heuristique *Choose best* permet d'avoir la sélection la plus robuste et la plus stable en pratique pour les tests que nous avons effectués. Cet aspect dominant de la précision dans un bon ensemble de classifieurs a également été montré dans les travaux de Rodriguez *et al.* [179]

Notre sentiment final concernant la diversité rejoint celui exprimé par Kuncheva [112]. Les connaissances dont on dispose à l'heure actuelle concernant la diversité sont encore incomplètes. Nous ne disposons pas de moyen d'analyse théorique solide telle que la décomposition de l'erreur qui s'applique dans le cas de la régression pour étudier l'influence de la diversité sur la précision d'un ensemble de classifieurs. De plus, rien ne permet d'affirmer que les différentes mesures de diversité proposées dans la littérature constituent le moyen le plus adapté pour étudier son comportement.

Les résultats que nous obtenons sur différents benchmarks de l'UCI montrent que dans certains cas, incorporer une part *raisonnable* de diversité durant le processus de création de l'ensemble se traduit par une augmentation de la précision. Cependant, ce cas de figure ne survient que pour des problèmes stables où le nombre d'exemples est important. Pour les problèmes où le nombre d'exemples est limité et où il n'est pas possible de réserver une partie de ces exemples en tant que données de validation, la précision individuelle des classifieurs reste la meilleure garantie dont on dispose pour obtenir un ensemble performant. Ces conclusions sont similaires à celles établies par Roli *et al.* [180].

Au travers des tests que nous avons présenté, il apparait que les problèmes de classifications sont très divers et ne présentent pas tous les mêmes difficultés. La *taille* du problème est un des paramètres à prendre en compte pour exploiter des données en vue d'effectuer l'apprentissage d'un classifieur (ou d'un ensemble de classifieurs). Cette taille dépend notamment du nombre de classes et du nombre d'exemples dont on dispose pour l'apprentissage. De nouveaux problèmes sont aujourd'hui abordés dans le domaine du data mining concernant une problématique de classification avec un très grand nombre de classes, qui de plus sont en évolution constante [72]. Nous pouvons nous interroger sur ce que devient la problématique de l'apprentissage dans une telle situation.

6.3 Apport applicatif pour la surveillance des rejets de fumées industrielles

Dans le cadre de l'application des méthodes de l'Apprentissage Automatique au contrôle de la pollution atmosphérique, nous avons mis au point un système de reconnaissance des événements à risque à partir d'exemples. Nous avons étudié les performances obtenues par différents algorithmes de classification supervisée et amélioré la qualité du système de reconnaissance original, en diminuant de manière significative le nombre de fausses alarmes et de mauvaises classification des rejets de fumée les plus dangereux.

Les travaux de recherche présentés ne concernent que la partie du système dédiée à la construction d'un classifieur à partir d'exemples. Les algorithmes de traitement d'images ainsi que le calcul des variables utilisées ont été définis par la société ALOATEC.

Une amélioration directement liée au mode de fonctionnement du système actuel peut être

introduite. La principale caractéristique du système développé par la société ALOATEC consiste à traiter sous la forme d'événements discrets les rejets de fumées dangereuses. Un événement est considéré comme un exemple que le système doit classer automatiquement. Une discrétisation des signaux continus est imposée par les concepteurs du système original de manière à associer à chaque attibut discret un sens physique relatif aux caractéristiques des panaches de fumées enregistrés par les caméras. Le fait de discrétiser la surveillance des rejets de fumée en événements indépendants entraîne une perte d'information importante car l'aspect dynamique de l'activité industrielle n'est pas pris en compte. Certaines méthodes de l'Apprentissage Automatique dédiées au traitement des séries chronologiques ou de la parole, par exemple, sont plus particulièrement adaptées lorsque l'apprentissage comporte un aspect dynamique. Une amélioration envisagée du système de surveillance des fumées est de ne plus traiter des événements discrets possédant un début et une fin dans le temps, mais de considérer la surveillance des fumées directement comme un processus de surveillance continu.

Une telle modification implique cependant une refonte du système dans son état actuel puisque la classification des rejets de fumées s'effectue selon des niveaux de gravité discrets. De plus, le dispositif matériel (caméras) devra être adapté pour permettre l'analyse de scène filmée avec une fréquence égale à celle des enregistrements réalisés par les caméras.

Le travail présenté ici constitue finalement une contribution au domaine de l'Apprentissage Automatique et, en particulier, sur la compréhension de la diversité dans l'élaboration d'ensembles de classifieurs. Il a été réalisé avec un soucis pragmatique à partir d'un problème environnemental d'actualité : la surveillance des fumées industrielles.

Bibliographie

[1] W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1) :37–66, 1991.

[2] Erin L. Allwein, Robert E. Schapire, and Yoram Singer. Reducing multiclass to binary : a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1 :113–141, 2001.

[3] A. Asuncion and D. J. Newman. Uci machine learning repository, 2007. http ://www.ics.uci.edu/∼mlearn/MLRepository.html.

[4] R. Avnimelech and N. Intrator. Boosted mixture of experts : an ensemble learning scheme. *Neural Computation*, 11(2) :483–497, February 1999.

[5] A. Berger. Error-correcting output coding for text classification. In *IJCAI'99 : Workshop on Machine Learnng for Information Filtering*, 1999.

[6] H. Bersini, G. Bontempi, and C. Decaestecker. Comparing rbf and fuzzy inference systems on theoretical and practical basis. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 169–174, Paris, 1995.

[7] J. C. Bezdek, T. R. Reichherzer, G. S.Lim, and Y. Attikiouzel. Multiple-prototype classifier design. *IEEE Transactions on Systems, Man, and Cybernetics, Part C : Applications and Reviews*, 28(1) :67–79, February 1998.

[8] C. M. Bishop. Mixture density networks. Technical Report NCRG/94/004, Aston University, 1994.

[9] G. Bontempi and H. Bersini. Identification of a sensor model with hybrid neuro-fuzzy methods. In *Proceedings of the 1997 International Conference on Engineering Applications of Neural Networks*, pages 325–328, Stockolm, Sweden, 1997.

[10] G. Bontempi and M. Birattari. Matlab software tool for neuro-fuzzy identification and data analysis. http ://www.ulb.ac.be/di/map/gbonte/software/Local/FIS.html.

[11] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, 1992. ACM.

[12] L. Breiman. Bagging predictors. *Machine Learning*, 24(2) :123–140, August 1996.

[13] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California, 1996.

[14] L. Breiman. Heuristics of instability and stabilization in model selection. *Annals of Statistics*, 24 :2350–2383, 1996.

[15] L. Breiman. Combining predictors. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets*, chapter 2, pages 31–50. Springer-Verlag, 1999.

[16] L. Breiman and P. Spector. Submodel selection and evaluation in regression : The x-random case. *International Statistical Review*, 60 :291–319, 1992.

[17] Leo Breiman. Random forests. *Machine Learning*, 45(1) :5–32, 2001.

[18] Leo Breiman, Jerome Friedman, Charles J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman & Hall/CRC, January 1984.

[19] G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods : a survey and categorisation. *Information Fusion*, 6(1) :5–20, 2005.

[20] Gavin Brown. *Diversity in Neural Network Ensembles*. PhD thesis, University of Birmingham, January 2004.

[21] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2) :121–167, 1998.

[22] W. S. Chaer, R. H. Bishop, and J. Ghosh. A mixture-of-experts framework for adaptive kalman filtering. *IEEE Transactions on Systems, Man, and Cybernetics*, 27(3) :452–464, 1997.

[23] C.-C. Chang, C.-W. Hsu, and C.-J. Lin. The analysis of decomposition methods for support vector machines. *IEEE Transactions on Neural Networks*, 11(4) :1003–1008, July 2000.

[24] Y. Chauvin and D. E. Rumelhart. *Back Propagation : Theory, Architectures, and Applications*. Lawrence Erlbaum Associates, 1995.

[25] K. Chen, D. H. Xie, and H. S. Chi. Speaker identification based on the time-delay hierarchical mixture of experts. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 2062–2066, November 1995.

[26] J. Cheng, D. Bell, and W. Liu. An algorithm for bayesian belief network construction from data. In *Proceedings of AI & STAT'97*, pages 83–90, Ft. Lauderdale, Florida, 1997.

[27] Jie Cheng and Russell Greiner. Learning bayesian belief network classifiers : Algorithms and system. *Lecture Notes in Computer Science*, 2056 :141–151, 2001.

[28] S.-B. Cho. Pattern recognition with neural networks combined by genetic algorithms. *Fuzzy Sets and Systems*, 103(2) :339–347, April 1999.

[29] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4) :309–347, October 1992.

[30] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3) :273–297, 1995.

[31] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2 :265–292, 2001.

[32] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Machine Learning*, 47(2-3) :201–233, 2002.

[33] P. Cunningham and J. Carney. Diversity versus quality in classification ensembles based on feature selection. In *Proceedings of the 11th European Conference on Machine Learning*, pages 109–116, Barcelona, Catalonia, Spain, May 2000.

[34] B. V. Dasarathy. *Nearest Neighbor (NN) Norms : NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, 1990.

[35] B. V. Dasarathy, J. S. Sanchez, and S. Townsend. Nearest neighbour editing and condensing tools-synergy exploitation. *Pattern Analysis & Applications*, 3(1) :19–30, February 2000.

[36] C. Decaestecker. Finding prototypes for nearest neighbour classification by means of gradient descent and deterministic annealing. *Pattern Recognition*, 30(2) :281–288, 1997.

[37] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46 :225–254, 2002.

[38] Jan Depenau. *Automated Design of Neural Network Architecture for Classification*. PhD thesis, DAIMI, Computer Science Department, Aarhus University, 1995.

[39] T. G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7) :1895–1923, October 1998.

[40] T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science. Proceedings of the First International Workshop on Multiple Classifier Systems*, 1857 :1–15, 2000.

[41] T. G. Dietterich. Ensemble methods in machine learning. *Lecture Notes in Computer Science*, 1857 :1–15, 2000.

[42] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees : bagging, boosting and randomization. *Machine Learning*, 40(2) :139–157, 2000.

[43] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2 :263–286, 1995.

[44] C. Domingo and O. Watanabe. Madaboost : A modification of adaboost. In *Proceedings of the 13th Annual Conference on Computational Learning Theory*, pages 180–189. Morgan Kaufmann, San Francisco, 2000.

[45] C. Domingo and O. Watanabe. Scaling up a boosting-based learner via adaptive sampling. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, pages 317–328, 2000.

[46] P. Domingos. A unified bias-variance decomposition and its applications. In *Proceedings of 17th International Conference on Machine Learning*, pages 231–238. Morgan Kaufmann, June 2000.

[47] P. Domingos. A unified bias-variance decomposition for zero-one and squared loss. In *Proceedings of the 17th National Conference on Artifcial Intelligence*, pages 564–569, Austin, Texas, 2000. AAAI Press.

[48] Pedro Domingos and Michael J. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3) :103–130, 1997.

[49] Harris Drucker, Chris J. C. Burges, Linda Kaufman, Alex Smola, and Vladimir Vapnik. Support vector regression machines. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 155–161. The MIT Press, 1997.

[50] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, N.Y., 1973.

[51] B. Efron. Estimating the error rate of a prediction rule : improvement on crossvalidation. In *Journal of the American Statistical Association*, volume 78, pages 316–331, 1983.

[52] B. Efron. Improvement on cross-validation : the .632+ bootstrap method. In *Journal of the American Statistical Association*, volume 92, pages 548–560, 1997.

[53] B. Efron and R. Tibshirani. *An introduction to the bootstrap*. Chapman & Hall, 1994.

[54] C. Elkan. Boosting and naive bayesian learning. Technical Report No. CS97-557. University of California, San Diego, September 1997.

[55] Floriana Esposito, Donato Malerba, and Giovanni Semeraro. A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5) :476–491, 1997.

[56] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7 :179–188, 1936.

[57] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2) :256–285, 1995.

[58] Y. Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3) :293–318, June 2001.

[59] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning*, pages 148–156, 1996.

[60] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1) :119–139, 1997.

[61] E. Frias-Martinez, A. Sanchez, and J. Veleza. Support vector machines versus multi-layer perceptrons for efficient off-line signature recognition. *Engineering Applications of Artificial Intelligence*, 19(6) :693–704, September 2006.

[62] J. H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Datamining and Knowledge Discovery*, 1(1) :55–77, 1997.

[63] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29 :131–163, 1997.

[64] B. Gabrys and D. Ruta. Genetic algorithms in classifier fusion. *Applied Soft Computing*, 6(4) :337–347, August 2006.

[65] D. Gacquer. Méthodes de classification appliquées à la détection de fumées polluantes. In *Manifestation des Jeunes Chercheurs STIC MajecSTIC'2006*, Lorient, France, Novembre 2006.

[66] D. Gacquer. Réseaux bayésiens et autres outils pour la classification de panaches de fumées polluantes. In *3ème Journées Francophones sur les Réseaux Bayésiens JFRB'06*, Valenciennes, France, Mai 2006.

[67] D. Gacquer, F. Delmotte, V. Delcroix, and S. Piechowiak. Détection automatique de fumées polluantes. Salon POLLUTEC, Paris, Novembre 2005.

[68] D. Gacquer, F. Delmotte, V. Delcroix, and S. Piechowiak. Comparison of bayesian classifiers to detect pollution. In *Proceedings of the 25th Edition of the European Annual Conference on Human Decision-Making and Manual Control EAM'06*, Valenciennes, France, Septembre 2006.

[69] D. Gacquer, F. Delmotte, V. Delcroix, and S. Piechowiak. Comparison of several classifiers for the detection of polluting smokes. In *International Conference on Computational Intelligence for Modelling, Control and Automation CIMCA'2006*, Sydney, Australia, November 2006. IEEE Computer Society.

[70] D. Gacquer, F. Delmotte, V. Delcroix, and S. Piechowiak. A genetic approach for training diverse classifier ensembles. In *Proceedings of the 12th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2008)*, Malaga, Spain, June 2008.

[71] S. F. Galán, F. Aguado, F. J. Díez, and J. Mira. Nasonet, modeling the spread of nasopharyngeal cancer with networks of probabilistic events in discrete time. *Artificial Intelligence in Medicine*, 25(3) :247–264, July 2002.

[72] E. Gaussier and F. Pacull. Distributed categorizer for large category system. In *NATO Advanced Study Institute on Mining Massive Data Sets for Security*, Gazzada, Italy, September 2007.

[73] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1) :1–58, January 1992.

[74] R. Ghani. Using error-correcting codes for text classification. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 303–310, 2000.

[75] G. J. Gibson. Exact classification with two-layer neural nets. *Journal of Computer and System Sciences*, 52(2) :349–356, April 1996.

[76] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.

[77] A. Golestani, K. A. Ali Amiri, and M. R. Jahed Motlagh. A novel adaptive-boost-based strategy for combining classifiers using diversity concept. In *6th IEEE/ACIS International Conference on Computer and Information Science*, pages 128–134, July 2007.

[78] A. J. Groves and D. Schuurmans. Boosting in the limit : Maximizing the margin of learned ensembles. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence*, pages 692–699, Madison, Wisconsin, USA, 1998.

[79] M. T. Hagan, H. B. Demuth, and M.H. Beale. *Neural Network Design*. PWS Publishing, Boston, MA, 1996.

[80] M. T. Hagan and M. Menhaj. Training feed-forward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(6) :989–993, 1994.

[81] T.B. Haley. Applying neural networks to automatic active sonar classification. In *Proceedings of the 10th International Conference on Pattern Recognition*, volume 2, pages 41–44, June 1990.

[82] Y. Hamamoto, S. Uchimura, and S. Tomita. A bootstrap technique for nearest neighbor classifier design. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1) :73–79, 1997.

[83] T. Hamamura, H. Mizutani, and B. Irie. A multiclass classification method based on multiple pairwise classifiers. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 809–813, Edinburgh, Scotland, August 2003.

[84] T. Hastie and R. Tibshirani. Classification by pairwise coupling. *The Annals of Statistics*, 26 :451–471, 1998.

[85] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Verlag, New York, 2001.

[86] J.-P. Haton. Neural network for speech recognition. In C.H. CHEN, editor, *Fuzzy Logic and Neural Network Handbook*. Mac Graw Hill, 1995.

[87] J.-P. Haton. Neural network for automatic speech recognition : A review. In G. Chollet, M. Di Benedetto, A. Esposito, and M. Marinaro, editors, *Speech Processing, Recognition and Artificial Neural Networks*. Springer, 1999.

[88] D. Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington, 1995.

[89] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8) :832 – 844, 98 1998.

[90] T. K. Ho. Data complexity analysis for classifier combination. In *Proceedings of the International Workshop on Multiple Classifier Systems*, pages 53–67, Cambrifge, UK, 2001.

[91] T. K. Ho. A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis & Applications*, 5(2) :102–112, June 2002.

[92] J. H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[93] L. Holmstrom, P. Koistinen, J. Laaksonen, and E. Oja. Neural and statistical classifiers - taxonomy and two case studies. *IEEE Transactions on Neural Networks*, 8(1) :5–17, January 1997.

[94] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5) :359–366, 1989.

[95] D.W. Hosmer and S. Lemeshow. *Applied Logistic Regression.* Wiley Series in Probability and Mathematical Statistics, 2000.

[96] C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE transactions on Neural Networks*, 13 :415–425, 2002.

[97] B. Levin J. L. Fleiss and M. C. Paik. *Statistical Methods for Rates and Proportions, 3rd Edition.* Wiley, 2003.

[98] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1) :79–87, 1991.

[99] A.K. Jain and M.D. Ramaswami. Classifier design with parzen windows. In *Pattern Recognition and Artificial Intelligence, E.S. Gelsema and L.N. Kanal, eds. North-Holland : Elsevier Science Publishers B.V.*, pages 211–228, 1988.

[100] Finn V. Jensen. *Bayesian Networks and Decision Graphs.* Information Science and Statistics. Springer, July 2001.

[101] W. Jiang. Process consistency for adaboost. Technical Report. Department of Statistics, Northwestern University, 2000.

[102] W. Jiang and R. Simon. A comparison of bootstrap methods and an adjusted bootstrap approach for estimating the prediction error in microarray classification. *Statistics in Medicine*, July 2007.

[103] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6(2) :181–214, 1994.

[104] M. I. Jordan and L. Xu. Convergence results for the em approach to mixtures of experts architectures. *Neural Networks*, 8(9) :1409–1431, 1995.

[105] K. Kang and J. Oh. Statistical mechanics of the mixture of experts. In *Advances in Neural Information Processing Systems*, pages 183–189, 1996.

[106] W. Kim, H. Lee, J. Park, and K. Yoon. Multi class adult image classification using neural networks. *Lecture Notes in Computer Science. Advances in Artificial Intelligence*, 3501 :222–226, 2005.

[107] Y. W. Kim and I. S. Oh. Classifier ensemble selection using hybrid genetic algorithms. *Pattern Recognition Letters*, 29(6) :796–802, April 2008.

[108] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *IJCAI*, pages 1137–1145, 1995.

[109] R. Kohavi and D. H. Wolpert. Bias plus variance decomposition for zero-one loss functions. In *Proceedings of the 13th International Conference on Machine Learning*, pages 275–283. Morgan Kaufmann, 1996.

[110] U. Kreßel. *Advances in Kernel Methods : Support Vector Learnings*, chapter Pairwise classification and support vector machines, pages 255–268. MIT Press, 1999.

[111] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In *Advances in Neural Information Processing Systems*, volume 7, pages 231–238. The MIT Press, 1995.

[112] L. I. Kuncheva. That elusive diversity in classifier ensembles. In *Proceedings of the 1st Iberian Conference on Pattern Recognition and Image Analysis*, pages 1126–1138, 2003.

[113] L. I. Kuncheva. *Combining Pattern Classifiers : Methods and Algorithms*. Wyley-Interscience, 2004.

[114] L. I. Kuncheva. *Combining Pattern Classifiers : Methods and Algorithms*, chapter Fusion of label outputs, pages 117–123. Wyley-Interscience, 2004.

[115] L. I. Kuncheva. On the optimality of naïve bayes with dependent binary features. *Pattern Recognition Letters*, 27(7) :830–837, 2006.

[116] L. I. Kuncheva and L. C. Jain. Designing classifier fusion systems by genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 4(4) :327–336, November 2000.

[117] L. I. Kuncheva and R. K. Kountchev. Generating classifier outputs of fixed accuracy and diversity. *Pattern Recognition Letters*, 23(5) :593–600, March 2002.

[118] L. I. Kuncheva and C. Whitaker. Using diversity with three variants of boosting : aggressive, conservative, and inverse. In *Proceedings of the International Workshop on Multiple Classifier Systems*, pages 81–90, Cagliari, Italy, June 2002.

[119] L. I. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2) :181–207, May 2003.

[120] L. I. Kuncheva, C. Whitaker, C. Shipp, and R. P. W. Duin. Is independence good for combining classiers. In *Proceedings of the 15th International Conference on Pattern Recognition*, pages 168–171, Barcelona, Spain, 2000.

[121] L. I. Kuncheva and C. J. Whitaker. Ten measures of diversity in classifier ensembles : Limits for two classifiers. In *Workshop on Intelligent Sensor Processing*, Birmingham, UK, February 2001.

[122] W.B. Langdon, S.J. Barrett, and B.F. Buxton. Combining decision trees and neural networks for drug discovery. In *Proceedings of the 5th European Conference on Genetic Programming*, pages 60–70, Kinsale, Ireland, 2002.

[123] H. Lei and V. Govindaraju. Half-against-half multi-class support vector machines. In *Multiple Classifier Systems*, pages 156–164, 2005.

[124] P. Leray and O. François. Bnt structure learning package, documentation and experiments. Technical Report, Laboratoire PSI - INSA Rouen- FRE CNRS 2645, 2004.

[125] X. Li, L. Wang, and E. Sung. Adaboost with svm-based component classifiers. *Engineering Applications of Artificial Intelligence*, 21 :785–795, 2008.

[126] R. P. Lippmann. A critical overview of neural network pattern classifiers. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 266–275, Princeton, NJ, USA, September 1991.

[127] X. Liu, H. Xing, and X. Wang. A multistage support vector machine. In *Proceedings of the 2nd International Conference on Machine Learning and Cybernetics*, pages 1305–1308, 2003.

[128] Y. Liu and X. Yao. A cooperative ensemble learning system. In *Proceedings of the International Joint Conference on Neural Networks*, pages 2202–2207, 1998.

[129] Y. Liu and X. Yao. Negatively correlated neural networks can produce best ensembles. *Australian Journal of Intelligent Information Processing Systems*, 4 :176–185, 1998.

[130] Y. Liu and X. Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B : Cybernetics*, 29(6) :716–725, December 1999.

[131] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4) :380–387, November 2000.

[132] Y. Liu and Y. F. Zheng. One-against-all multi-class svm classification using reliability measures. In *Proceedings of the International Joint Conference on Neural Networks*, pages 849–854, August 2005.

[133] Yong Liu. *Negative Correlation Learning and Evolutionary Neural Network Ensembles*. PhD thesis, The University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 1998.

[134] R. Maclin and J. W. Shavlik. Combining the predictions of multiple classifiers : using competitive learning to initialize networks. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 524–530, 1995.

[135] E. H. Mamdani. Application of fuzzy algorithms for the control of a dynamic plant. In *Proceedings of eht IEE*, volume 121, pages 1585–1588, December 1974.

[136] E. H. Mamdani. Application of fuzzy logic to approximate reasoning using a linguistic system. *IEEE Transaction on Computers*, 26 :1182–1191, 1977.

[137] M. Mangeas and A. S. Weigend. First experiments using a mixture of nonlinear experts for time series analysis. In *Proceedings of the World Congress on Neural Networks (WCNN'95)*, pages 104–109, Washington, DC, USA, July 1995.

[138] D. D. Margineantu and T. G. Dietterich. Pruning adaptive boosting. In *Proceedings 14th International Conference on Machine Learning*, pages 211–218, 1997.

[139] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification. AAAI/ICML-98 Workshop on Learning for Text Categorization. Technical Report WS-98-05. AAAI Press, 1998.

[140] W. S. McCullcoch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5 :115–133, 1943.

[141] R. Meir and G. Ratsch. An introduction to boosting and leveraging. *Lecture Notes In Artificial Intelligence*, pages 118–183, 2003.

[142] P. Melville and R. J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, 6 :99–111, 2005.

[143] P. Melville, N. Shat, L. Mihalkova, and R. J. Mooney. Experiments on ensembles with missing and noisy data. In *Proceedings of the 5th International Workshop on Multiple Classifier Systems*, pages 293–302, Cagliari, Italy, June 2004.

[144] J. Milgram, M. Cheriet, and R. Sabourin. One against one or one against all : Which one is better for handwriting recognition with svms ? In *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule, France, October 2006.

[145] M. L. Minsky and S. A. Papert. *Perceptrons*. MIT Press, 1969.

[146] T. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[147] Y. L. Murphey, C. Zhihang, and G. Hong. Neural learning using adaboost. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 1037–1042, July 2001.

[148] M. Nakamura, H. Nomiya, and K. Uehara. Improvement of boosting algorithm by modifying the weighting rule. *Annals of Mathematics and Artificial Intelligence*, 41(1) :95–105, may 2004.

[149] P. Naïm, P-H. Wuillemin, P. Leray, O. Pourret, and A. Becker. *Réseaux bayésiens*. Eyrolles, Paris, 2004.

[150] H. D. Navone, D. Cook, T. Downs, and D. Chen. Boosting naive-bayes classifiers to predict outcomes for hipprostheses. In *Proceedings of the International Joint Conference on Neural Networks*, volume 5, pages 3622–3626, Washington, DC, USA, July 1999.

[151] Richard E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall series in Artificial Intelligence, 2004.

[152] D. Nguyen and B. Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptative weights. In *Proceedings of the International Joint Conference on Neural Networks*, volume 3, pages 21–26, San Diego, Calif., June 1990.

[153] C. Olivier, O. Colot, P. Courtellemont, and A. El Matouat. Information criteria and abrupt changes of probability laws. *Signal Processing VII : Theorie and Applications*, 7(3) :1855–1858, 1994.

[154] D. Opitz and R. Maclin. Popular ensemble methods : An empirical study. *Journal of Artificial Intelligence Research*, 11 :169–198, 1999.

[155] S. Osowski and K. Garanty. Forecasting of the daily meteorological pollution using wavelets and support vector machine. *Engineering Applications of Artificial Intelligence*, 20(6) :745–755, September 2007.

[156] J. Park and I. W. Sandberg. Universal approximation using radial-basis-function networks. *Neural Computation*, 3(2) :246–257, 1991.

[157] D. Partridge and W. Krzanowski. Software diversity : practical statistics for its measurement and exploitation. *Information and Software Technology*, 39(10) :707–717, 1997.

[158] D. Partridge and W. B. Yates. Engineering multiversion neural-net systems. *Neural Computation*, 8(4) :869–893, 1996.

[159] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3) :1065–1076, September 1962.

[160] A. Pavelka and A. Procházka. Algorithms for initialization of neural network weights. Technical Report. Institute of Chemical Technology, Prague. Department of Computing and Control Engineering, 2004.

[161] Judea Pearl. *Causality : Models, Reasoning, and Inference*. Cambridge University Press, March 2000.

[162] W. Pedrycz. An identification algorithm in fuzzy relational systems. *Fuzzy Sets and Systems*, 13(2) :153–167, 1984.

[163] T. V. Pham, M. Worring, and A. W. M. Smeulders. Face detection by aggregated bayesian network classifiers. *Lecture Notes in Computer Science*, 2123 :249–262, 2001.

[164] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin dags for multiclass classification. In *In Advances in Neural Information Processing Systems*, volume 12, pages 547–553, 2000.

[165] J. G. postaire. *De l'image à la décision : Analyse des images numériques et théorie de la decision*. Dunod, 1987.

[166] D. Price, S. Knerr, L. Personnaz, and G. Dreyfus. Pairwise neural network classifiers with probabilistic outputs. *Neural Information Processing Systems*, 7 :1109–1116, 1994.

[167] O. Pujol, P. I. Radeva, and J. Vitria. Discriminant ecoc : A heuristic method for application dependent design of error correcting output codes. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(6) :1007–1012, 2006.

[168] J. R. Quinlan. Bagging, boosting, and c4.5. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence.*, pages 725–730, 1996.

[169] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1) :81–106, 1986.

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

[171] B. Quost. *Combinaison de classifieurs binaires dans le cadre de la théorie des fonctions de croyance*. PhD thesis, Université de Technologie de Compiègne, Novembre 2006.

[172] P. E. Hart R. O. Duda and D. G. Stork. *Pattern Classification (2nd Edition)*. John Wiley & Sons, New York, 2001.

[173] V. Ramamurti and J. Ghosh. Advances in using hierarchical mixture of experts for signal classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3569–3572, 1996.

[174] G. Ratsch. Robust multi-class boosting. In *Eurospeech 2003, 8th European Conference on Speech Communication and Technology*, pages 997–1000, Geneva, Switzerland, September 2003.

[175] G. Ratsch, T. Onoda, and K. R. Muller. Regularizing adaboost. In *Proceedings of the 1998 conference on Advances in neural information processing systems*, pages 564–570, Denver, Colorado, USA, December 1998. MIT Press.

[176] G. Ratsch, T. Onoda, and K. R. Muller. Soft margins for adaboost. *Journal of Machine Learning*, 42(3) :287–320, March 2001.

[177] R. Rifkin and A. Klautau. In defence of one-vs-all classification. *Journal of Machine Learning Research*, 5 :101–141, 2004.

[178] I. Rish. An empirical study of the naïve bayes classifier. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2001.

[179] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest : A new classifier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(10) :1619–1630, October 2006.

[180] F. Roli, G. Giacinto, and G. Vernazza. Methods for designing multiple classifier systems. In *Proceedings of the 2nd International Workshop on Multiple Classifier Systems*, pages 78–87, Cambridge, UK, July 2001.

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

[182] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323 :533–536, 1986.

[183] D. Ruta and B. Gabrys. Analysis of the correlation between majority voting errors and the diversity measures in multiple classifier systems. In *Proceedings of the International Symposium on Soft Computing SOCO'2001*, Paisley, Scotland, 2001.

[184] D. Ruta and B. Gabrys. Application of the evolutionary algorithms for classifier selection in multiple classifier systems with majority voting. In *Proceedings of the Second International Workshop on Multiple Classifier Systems*, pages 399–408, Cambridge, UK, July 2001.

[185] R. M. Sanner and J.-J. E. Slotine. Gaussian networks for direct adaptive control. *IEEE Transactions on Neural Networks*, 3 :837–863, 1992.

[186] W. Sarle. Stopped training and other remedies for overfitting. In *In Proceedings of the 27th Symposium on Interface*, pages 352–360, 1995.

[187] R. E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2) :197–227, 1990.

[188] R. E. Schapire. Theoretical views of boosting and applications. In *Proceedings of the 10th International Conference on Algorithmic Learning Theory*, pages 13–25, 1999.

[189] R. E. Schapire. The boosting approach to machine learning : an overview. In *Proceedings of the MSRI Workshop on Nonlinear Estimation and Classification*, 2002.

[190] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3) :297–336, December 1999.

[191] R. E. Schapire, Y. Singer, P. Bartlett, and W. Lee. Boosting the margin : a new explanation for the effectiveness of voting methods. *Annals of Statistics*, 26(5) :1651–1686, 1998.

[192] B. Schölkopf. The kernel trick for distances. In T. K. Leen, T. G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems*, volume 13, pages 301–307. The MIT Press, 2000.

[193] B. Schölkopf and A. J. Smola. *Learning with Kernels*. The MIT Press, 2002.

[194] Karl-Michael Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics*, volume 1, pages 307–314, 2003.

[195] H. Schwenk and Y. Bengio. Boosting neural networks. *Neural Computation*, 12 :1869–1887, 2000.

[196] M. Sebban and H.-M. Suchier. Etude sur l'amélioration du boosting : Réduction de l'erreur et accélération de la convergence. *Journal Electronique d'Intelligence Artificielle*, 6, 2004.

[197] T. J. Sejnowski and C. R. Rosenberg. Parallel networks that learn to pronounce english text. *Journal of Complex Systems*, 1(1) :145–168, February 1987.

[198] R. A. Servedio. Smooth boosting and learning with malicious noise. *The Journal of Machine Learning Research*, 4 :633–648, December 2003.

[199] G. Shafer. *A Mathematical Theory of Evidence*. Princeton Univ. Press, 1976.

[200] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-Neighbor Methods in Learning and Vision : Theory and Practice (Neural Information Processing)*. The MIT Press, March 2006.

[201] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27 :379–423, 623–656, July and October 1948.

[202] A. J. C. Sharkey, N. E. Sharkey, U. Gerecke, and G. O. Chandroth. The test and select approach to ensemble combination. In *Proceedings of the First International Workshop on Multiple Classifier Systems*, pages 30–44, Cagliari, Italy, June 2000.

[203] A.J.C. Sharkey. Types of multinet system. In *Proceedings of the International Workshop on Multiple Classifier Systems*, pages 108–117, Calgiari, Italy, 2002.

[204] N. Sharkey, J. Neary, and A. Sharkey. Searching weight space for backpropagation solution types, current trends in connectionism. In *Proceedings of the Swedish Conference on Connectionism*, pages 103–120, 1995.

[205] C. A. Shipp and L. I. Kuncheva. Relationships between combination methods and measures of diversity in combining classifiers. *Information Fusion*, 3(2) :135–148, June 2002.

[206] D. B. Skalak. The sources of increased accuracy for two proposed boosting algorithms. In *Proceedings of the American Association for Artificial Intelligence. Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, 1996.

[207] M. Skuruchina, L. I. Kuncheva, and R. P. W. Duin. Bagging and boosting for the nearest mean classifier : Effects of sample size on diversity and accuracy. In *Proceedings of the International Workshop on Multiple Classifier Systems*, pages 62–71, Cagliari, Italy, June 2002.

[208] P. Smets. Belief functions : the disjunctive rule of combination and the generalized bayesian theorem. *International Journal of Approximate Reasoning*, 9 :1–35, 1993.

[209] Y. Sun, S. Todorovic, J. Li, and D. Wu. Unifying the error-correcting and output-code adaboost within the margin framework. In *Proceedings of the 22nd international conference on Machine learning*, pages 872–879, Bonn, Germany, 2005.

[210] T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics*, 15(1) :116–132, 1985.

[211] I V. Tetko, D. J. Livingstone, and A. I. Luik. Neural network studies. 1. comparison of overfitting and overtraining. *In Journal of chemical information and computer sciences*, 35 :826–833, 1995.

[212] S. G. Thompson. Pruning boosted classifiers with a real valued genetic algorithm. *Knowledge-Based Systems*, 12(5) :277–284, October 1999.

[213] R. Tibshirani. A comparison of some error estimates for neural network models. *Neural Computation*, 8 :152–163, 1996.

[214] K. M. Ting and I. H. Witten. Stacked generalizations : When does it work ? In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 2, pages 866–873, 1997.

[215] K. M. Ting and I. H. Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research*, 10 :271–289, 1999.

[216] J. M. Torres-Moreno. *Apprentissage et généralisation par réseaux de neurones : étude de nouveaux algorithmes constructifs*. PhD thesis, Institut National Polytechnique de Grenoble, 1997.

[217] G. T. Toussaint. Bibliography on estimation of misclassification. *IEEE Transactions on Information Theory*, 20 :472–479, July 1974.

[218] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[219] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation and signal processing. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 281–287. The MIT Press, 1997.

[220] V. N. Vapnik. *The nature of Statistical Learning Theory*. Springer, 1995.

[221] V. N. Vapnik and A. J. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2) :264–281, 1971.

[222] X. Wang and H. Wang. Classification by evolutionnary ensembles. *Pattern Recognition*, 39(4) :595–607, April 2006.

[223] S. R. Waterhouse and A. J. Robinson. Classification using hierarchical mixtures of experts. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 177–186, Long Beach, CA, 1994. IEEE Press.

[224] S. M. Weiss and C. A. Kulikowski. *Computer systems that learn : classification and prediction methods from statistics, neural nets, machine learning, and expert systems*. Morgan Kaufmann, 1991.

[225] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5 :241–259, 1992.

[226] K. Woods, W. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4) :405–410, April 1997.

[227] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3) :418–435, May 1992.

[228] W. Yates and D. Partridge. Use of methodological diversity to improve neural network generalization. *Neural Computing and Applications*, 4(2) :114–128, 1996.

[229] G. U. Yule. On the association of attributes in statistics. *Philosophical Transactions of the Royal Society of London. Series A*, 194 :257–319, 1900.

[230] L. Zadeh. Fuzzy sets. *Information and Control*, 8(3) :338–353, 1965.

[231] Harry Zhang. The optimality of naive bayes. In *FLAIRS Conference*. AAAI Press, 2004.

[232] Ji Zhu, Saharon Rosset, Hui Zou, and Trevor Hastie. Multi-class adaboost. Technical report, Department of Statistics, University of Michigan, Ann Arbor, MI 48109, 2006.

Résumé

L'influence de la diversité lors de la construction d'ensembles de classifieurs a soulevé de nombreuses discussions au sein de la communauté de l'Apprentissage Automatique ces dernières années. Une manière particulière de construire un ensemble de classifieurs consiste à sélectionner individuellement les membres de l'ensemble à partir d'un pool de classifieurs en se basant sur des critères prédéfinis. La littérature fait référence à cette méthode sous le terme de paradigme Surproduction et Sélection, également appelé élagage d'ensemble de classifieurs.

Les travaux présentés dans cette thèse ont pour objectif d'étudier le compromis entre la précision et la diversité existant dans les ensembles de classifieurs. Nous apportons également certains éléments de réponse sur le comportement insaisissable de la diversité lorsqu'elle est utilisée de manière explicite lors de la construction d'un ensemble de classifieurs.

Nous commençons par étudier différents algorithmes d'apprentissage de la littérature. Nous présentons également les algorithmes ensemblistes les plus fréquemment utilisés. Nous définissons ensuite le concept de diversité dans les ensembles de classifieurs ainsi que les différentes méthodes permettant de l'utiliser directement lors de la création de l'ensemble.

Nous proposons un algorithme génétique permettant de construire un ensemble de classifieurs en contrôlant le compromis entre précision et diversité lors de la sélection des membres de l'ensemble. Nous comparons notre algorithme avec différentes heuristiques de sélection proposées dans la littérature pour construire un ensemble de classifieurs selon le paradigme Surproduction et Sélection.

Les différentes conclusions que nous tirons des résultats obtenus pour différents jeux de données de l'UCI Repository nous conduisent à la proposition de conditions spécifiques pour lesquelles l'utilisation de la diversité peut amener à une amélioration des performances de l'ensemble de classifieurs. Nous montrons également que l'efficacité de l'approche Surproduction et Sélection repose en grande partie sur la stabilité inhérente au problème posé.

Nous appliquons finalement nos travaux de recherche au développement d'un système de classification supervisée pour le contrôle de la pollution atmosphérique survenant sur des sites industriels. Ce système est basé sur l'analyse par traitement d'image de scènes à risque enregistrées à l'aide de caméras. Son principal objectif principal est de détecter les rejets de fumées dangereux émis par des usines sidérurgiques et pétro-chimiques.

Mots-clés: Apprentissage Automatique, Classification Supervisée, Ensemble de classifieurs, Diversité, Sélection de classifieurs

Abstract

Discussions about the influence of diversity when designing Multiple Classifier Systems has been an active topic in Machine Learning overs recent years. One possible way of considering the design of Multiple Classifier Systems is to select the ensemble members from a large pool of classifiers focusing on predefined criteria, which is known as the Overproduce and Choose paradigm, also called Ensemble Pruning.

The objective of this PhD Thesis is to study the trade-off between accuracy and diversity which exists in multiple classifier systems and bring some elements of response on the elusive behavior of diversity when using it explicitly in ensemble learning algorithms.

We start by reviewing some well known Machine Learning algorithms and ensemble learning techniques from the literature. We then present in details the concept of diversity and the way it is used by certain ensemble learning algorithms.

We propose a genetic heuristic to design multiple classifier systems by controlling the trade-off between diversity and accuracy when selecting individual classifiers. We compare the proposed genetic selection with several heuristics described in the literature to build multiple classifier systems under the Overproduce and Choose paradigm.

The different observations we draw from several experiments on UCI datasets lead us to propose certain specific conditions where it might be worth using diversity explicitly during the design stage of multiple classifier systems. We also show that effectiveness of the Overproduce and Choose paradigm mainly relies on the stability of a given problem.

The application of our research work concerns the development of a supervised classification system to control atmospheric pollution around industrial complexes. This system is based on the analysis of visual scenes recorded by cameras and aims at detecting dangerous smoke trails rejected by steelworks or chemical factories.

Keywords: Machine Learning, Supervised Classification, Multiple Classifier Systems, Diversity, Classifier Selection, Ensemble Pruning