REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU-BEKR BELKAID - TLEMCEN

THÈSE

Présentée à :

FACULTE DES SCIENCES – DEPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

DOCTORAT EN SCIENCES

Spécialité: Informatique

Par :

Mme Saidi Meryem

Sur le thème

Traitement des données biologiques par les méthodes ensemblistes

---

Soutenue publiquement à l'université de Tlemcen devant le jury composé de :

Chikh Azeddine	Professeur	Université de Tlemcen	Président
Chikh Med Amine	Professeur	Université de Tlemcen	Directeur de thèse
Benamrane nacéra	Professeur	Université d'Oran	Examinateur
Abderrahim Med El Amine	MCA	Université de Tlemcen	Examinateur
Adjoudj Reda	MCA	Université de Sidi Bel Abbes	Examinateur

Laboratoire de Génie Biomédical (GBM)

Remerciements

Je tiens à remercier Monsieur Chikh Mohammed El Amine, Directeur de recherche, d'avoir accepté de diriger ma thèse. Sans la qualité de ses conseils ce travail n'aurait jamais vu le jour.

Je remercie le Professeur Chikh Azeddine, de m'avoir fait l'honneur de présider le jury, et de l'intérêt qu'il a porté à mes travaux.

Je remercie le Dr. Abderrahim Med El Amine, le Pr. Benamrane Nacéra et le Dr. Adjoudj Reda pour l'intérêt qu'ils ont porté à ce travail et l'honneur qu'ils m'ont fait en acceptant d'en être les rapporteurs.

Je remercie le Dr. Settouti Nesma et le Dr. Bechar Mohammed El Amine pour leurs aides diverses, leurs encouragements dans les moments difficiles et l'intérêt qu'ils m'ont toujours témoigné.

Une sincère pensée à mes parents, mon mari et tous mes proches pour leur compréhension et leurs encouragements.

Résumé

Le développement des modèles de classification est l'une des principales tâches dans le domaine de data mining. Toutefois, le volume élevé de données générées par différents domaines de recherche, allant du séquençage du génome humain, qui permet d'obtenir des niveaux d'expressions de plusieurs milliers de gènes, aux millions d'informations circulant sur internet rend l'utilisation des méthodes d'apprentissage automatique un vrai défi. D'où la nécessité d'une étape de prétraitement afin de préparer la base aux algorithmes d'apprentissage.

L'induction d'un modèle de classification pour le diagnostic avec autant d'instances et de variables est un défi majeur dans le domaine de l'apprentissage statistique. D'où la nécessité de réduire ce nombre. Parmi les processus de prétraitements applicables sur une base, nous trouvons les méthodes de réduction : les algorithmes de sélection d'instances et de variables.

Le sujet de cette thèse est orienté vers la recherche de méthodes efficaces de traitements des données médicales et biologiques. Nous nous sommes principalement intéressés à l'application d'une méthode de sélection d'instances pour nettoyer et réduire la base d'apprentissage avant la conception du classifieur.

Au cours de nos recherches, nous avons pu étudier les différentes approches existantes ainsi que leur avantages et limitations. Nous nous sommes intéressés aux méthodes ensemblistes afin de pallier les problèmes rencontrés par les méthodes de sélection classiques. Les méthodes ensemblistes sont un ensemble d'algorithmes qui s'inspire du principe « *l'union fait la force* », en effet ces méthodes combinent les décisions individuelles de plusieurs algorithmes de classification faibles afin d'améliorer leurs performances pour classer de nouveaux exemples.

Donc si on décide *«à la majorité»*, alors on se trompe si et seulement si plus de la moitié du *«comité»* se trompe. En effet, la décision prise en groupe ne peut être fausse que si la majorité du groupe se trompe. Ceci rend les individus sur lesquels les classifieurs sont le plus en désaccord, les plus intéressants à traiter au cours de la sélection d'instance.

Un algorithme de sélection d'instances basé sur les algorithmes ensemblistes et notamment sur l'algorithme Forêt aléatoire a été implémenté. Nous avons testé notre proposition sur plusieurs problèmes de classification de UCI Machine Learning Repository ainsi que sur une base d'image cytologique afin d'optimiser la segmentation automatique de globules blancs. Les résultats obtenus démontrent

que notre proposition est aussi performante que les méthodes existantes tout en étant moins coûteuse.

Mots clés
Sélection d'instances, méthodes ensembliste, forêt aléatoire, marge ensembliste, reconnaissance automatique des globules blancs, images cytologique

Abstract

Classification algorithm development is one of the main processes in data mining. However, the high volume of data generated by different areas of research, from the sequencing of the human genome, which provides levels of expression of several thousand genes, to millions of information circulating on the Internet makes the use of machine learning methods a real challenge. Hence there is a need for a pretreatment step to prepare the basis for learning algorithms.

The induction of a classification model for diagnosis with this huge amount of instances and variables is a major challenge in the field of statistical learning. So, it is necessary to reduce this number. Among the pretreatment processes applicable on a base, we find the methods of reduction : the algorithms of instance selection.

This thesis is devoted to the search for efficient methods of medical and biological data processing. We were mainly interested in applying an instance selection method to clean up and reduce the learning dataset before building the classifier.

We were able to study the different existing approaches as well as their advantages and limitations. We are interested in the set-up methods in order to overcome the problems encountered by classical selection methods.

Ensemble methods are a set of algorithms inspired by the precept "union is strength", because these methods combine the individual decisions of several weak classification algorithms to improve their performance to classify new examples. So, if we decide "by majority", then we are wrong if and only if more than half of the "committee" is wrong. Indeed, the fact that the group decision can only be false if the majority of the group is mistaken makes the individuals on which the classifiers are most at odds are those who are the most interesting to deal with in the selection process.

A selection algorithm based on the ensemble algorithms and in particular the Random Forest algorithm has been implemented. We have tested our method on several UCI Machine Learning Repository classification issues as well as on a cytological image base in order to optimize the automatic segmentation of white blood cells. The results obtained demonstrate that our approach is as efficient as the existing methods while being less expensive.

Keywords

Instances selection, Ensemble methods, Random Forest, Ensemble margin, automatic recognition of white blood cells, cytological images.

Table des matières

Table des figures

Liste des tableaux

Introduction

1 Motivation

Les nouvelles technologies d'acquisition et de stockage de données, ont conduit à une forte augmentation des données collectées quotidiennement. Par exemple, l'accélérateur de particules du CERN génère chaque année à lui seul 1 petabyte de données, un problème auquel doivent faire face la plupart des domaines de recherche comme l'astronomie au séquençage du génome humain [1]. Par conséquent, il est devenu très difficile d'extraire manuellement des informations utiles à partir des énormes quantités de données disponibles.

L'apprentissage artificiel est un domaine qui s'intéresse au développement de modèles dits intelligents et qui simule le raisonnement humain en tirant des connaissances à partir d'un ensemble de données. Grâce à ce processus, une machine peut répondre correctement à un problème réel ou artificiel en utilisant l'expérience accumulée durant l'apprentissage des exemples. A chaque exemple de la base d'apprentissage est associée une réponse, ce qui permet aux algorithmes de réagir correctement aux nouveaux exemples.

Le processus d'identification des patterns descriptifs sur une large base de données est appelé *Extraction de connaissance et Fouille de données (Knowledge discovery and data mining (KDD))*. Parmi les approches du KDD, nous trouvons le clustering qui permet de générer les patterns sans supervision ou l'apprentissage supervisé qui prédit la classe de nouvelles données à partir de données existantes fournies avec les classes. Les instances inconnues forment l'ensemble de tests tandis que les instances labellisées forment l'ensemble d'apprentissage. La prédiction des classes se fait par un algorithme de classification suivant deux étapes : [2] :

— Entrainement du classifieur (en général)
— Test du classifieur.

Malheureusement, l'application des algorithmes d'apprentissage artificiel a des problèmes de grandes dimensions présentes certaines limites étant donné que la plupart des algorithmes classiques (SVM, K-ppv, réseaux de neurones, ...) ont été développés avec des bases de petites dimension. En effet, si la taille (nombre d'instances) ou la dimension (nombre de variables) de la base augmentent, les approches existantes font face à plusieurs problèmes. Parmi ces problèmes nous citons, l'augmentation du coût de labellisation, de l'espace mémoire et du temps d'apprentissage.

En effet, il n'y a aucun intérêt majeur lorsque nous utilisons un algorithme de classification qui nécessite beaucoup de ressources ou un temps d'apprentissage trop long. D'un autre côté, en plus de la quantité des données, c'est aussi la qualité des données qui pose un problème aux algorithmes existant. Le bruit présent dans les bases de données, causé par une mauvaise labellisation, peut induire à une perte de performances. En plus, ils doivent faire face aussi aux données déséquilibrées de certaines bases, où le nombre de cas ciblés est largement inférieur

au nombre du cas non ciblés. De ce fait, l'apprentissage est orienté en faveur de la classe majoritaire.

Deux solutions possibles s'offrent pour résoudre ces problèmes. La première possibilité est l'élaboration de nouveaux algorithmes rapides et efficaces capables de faire face aux problèmes de données. La deuxième solution est l'implémentation d'approches de sélection d'instance pour l'adaptation des données aux algorithmes existants.

Le passage par une étape de prétraitement pour nettoyer et réduire la base d'apprentissage avant la construction du classifieur à démontrer de bons résultats dans divers domaines [3–5]. Cette étape a pour objectif de sélectionner un sous-ensemble de l'ensemble d'apprentissage formé par les instances les plus pertinentes, qui permet de créer un classifieur aussi performant que celui obtenu avec l'ensemble d'origine.

2 Objectif de la thèse

Durant les dernières années, plusieurs recherches ont était menées pour réduire les coûts sans pour autant compromettre les performances des modèles d'apprentissage. La sélection d'instances est l'une des étapes de prétraitement de la fouille de donnée durant laquelle la base de données sera plus « propre » et plus représentative. Cette étape permet d'éliminer les instances bruitées, redondantes ou superflues afin d'obtenir de meilleurs résultats de classification.

L'inconvénient majeur des méthodes de sélection existantes, comme les algorithmes évolutionnaires [6] ou l'algorithme K-nn [7], concerne la taille de la base des données où la complexité de l'algorithme augmente avec l'ordre de grandeur. Ce qui les rend inapplicables aux cas où leur utilisation est la plus nécessaire.

Le principal objectif de cette thèse est de proposer une approche capable d'effectuer une sélection qui permet un compromis entre le temps et la performance. En effet, il n'y a aucun intérêt à utiliser un algorithme de sélection qui nécessite un temps d'exécution et des ressources importantes. D'un autre côté, la perte d'information de l'ensemble d'origine ne doit pas être trop importante afin que le sous-ensemble obtenu par l'application de cet algorithme ne dégrade pas le taux de reconnaissance du classifieur. Donc, nous évaluerons les performances de notre approche en termes de temps d'exécution, de taux de classification et de taux de réduction.

3 Contribution de la thèse

Dans cette thèse nous nous intéressons à l'utilisation des algorithmes ensemblistes et notamment à l'algorithme Forêt aléatoire [8, 9] pour la sélection d'instance. Ce mécanisme, est actuellement considéré comme l'une des techniques de

classification les plus efficaces et rapide. Les méthodes ensemblistes se basent sur le principe « l'union fait la force » et plus exactement qu'un comité d'experts donnera une meilleure réponse qu'un seul expert.

Dans notre travail, nous proposons une approche pour l'utilisation de l'algorithme de forêt aléatoire pour la sélection d'instances. Nous partons du principe que plus les membres de l'ensemble sont en désaccord sur le label d'une instance, plus l'information qu'elle renferme est importante pour la discrimination entre les classes. Ce degré de désaccord est représenté par le principe de marge des méthodes ensemblistes.

4 Organisation de la thèse

Cette thèse est organisée comme suit : dans le chapitre I, nous établissons une base de connaissances sur les concepts de l'apprentissage automatique et l'analyse des données. Nous avons aussi introduit quelques algorithmes de classification.
Nous présentons dans le chapitre II une revue de l'état de l'art des algorithmes de sélection. Elle inclut les objectifs et la taxonomie des méthodes de sélection ainsi que la présentation de quelques algorithmes de sélection d'instances les plus représentatifs.
Le chapitre III est dédié aux méthodes ensemblistes. Nous expliquons les principes de fonctionnement des méthodes existantes et plus particulièrement à l'algorithme forêt aléatoire.
Notre contribution est présentée au chapitre IV avec les résultats des expérimentations sur 12 bases de données de grande taille. Dans la deuxième partie du chapitre IV, nous prouvons l'efficacité de notre approche sur un cas d'étude de segmentation d'images cytologique. Enfin, les conclusions et perspectives sont énoncées au chapitre V.

Chapitre I

Background : Apprentissage automatique et analyse des données

Introduction

L'apparition d'immenses bases de données, a fait apparaître une nouvelle discipline à l'interface de la statistique et de l'informatique, le « data mining » traduit en « fouille de données ». Les méthodes de cette discipline ont été initiées dans le cadre de la statistique puis appliquées et enrichies dans le cadre informatique suite au développement des ordinateurs et à l'accumulation de données.

La fouille de données est un domaine de recherche vaste regroupant plusieurs méthodes aux objectifs différents. Dans cette section, nous introduisons les concepts et les annotations de base et nous introduisons quelque unes des méthodes d'analyse de données et leurs objectifs.

1 Concepts et définitions

« La fouille de donnée est le processus d'extraction d'informations intéressantes, pertinentes et perspicaces ainsi que des modèles descriptifs, compréhensifs et prédictifs à partir d'un grand nombre de donnée [10]. »

L'apprentissage automatique vise à mettre en œuvre des systèmes intelligent en développant des méthodes permettant à un ordinateur d'apprendre à partir d'un ensemble de données. Plusieurs domaines bénéficient de ces algorithmes intelligents tels que la reconnaissance de forme, la détection d'intrusion, la bioinformatique, l'aide au diagnostic médical et beaucoup d'autres.

1.1 Matrice de données

Les données à explorer sont représentées sous forme d'une matrice nxm, avec n lignes et m colonnes. Chaque ligne représente une instance ou un individu et chaque colonne représente les attributs ou variables descriptifs de l'instance. Une matrice est représentée comme suit [10] :

$$\mathbf{D} = \left(\begin{array}{c|cccc} & X_1 & X_2 & \cdots & X_d \\ \hline \mathbf{x}_1 & x_{11} & x_{12} & \cdots & x_{1d} \\ \mathbf{x}_2 & x_{21} & x_{22} & \cdots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_n & x_{n1} & x_{n2} & \cdots & x_{nd} \end{array}\right)$$

FIGURE I.1 – Matrice de données

Le nombre n de lignes représente la taille de la base tandis que le nombre m de colonnes représente la dimension de la base.

Attribut1	Attribut2	Attribut3	Attribut4	Attribut5	Attribut6	Label
6	148	72	35	0.627	50	1

TABLE I.1 – Instance d'une base de donnée

La table I.1 représente un extrait de la base PIMA. Chaque instance de la base représente une patiente, les attributs $m-1$ représentent les différentes mesures utilisées pour le diagnostic et l'attribut m est la classe d'une instance (malade ou sain).

Il existe d'autre type de bases plus complexes formé par du texte, images, audio ou video. Dans la plupart des cas, ces bases peuvent être transformé en bases matricielles via une extraction de caractéristiques. Par exemple, pour le cas d'une base de données d'images, il est possible de la convertir en une base matricielle où chaque ligne représente une image ou un pixel et les colonnes représentent les caractéristiques telles que les espaces de couleurs, les textures, etc...

1.2 Type d'attribut

Chaque instance de la base est décrite par un ensemble d'attributs aussi appelé caractéristiques, variables, descripteurs ou champs. Il existe différents types d'attributs :

— Attribut quantitatif : ce type d'attribut exprime une quantité, les valeurs sont numérique et peuvent être exprimer sur un domaine continu sur $\mathbb{R}$ ou discret sur $\mathbb{N}$ ou $\mathbb{Z}$.
— Attribut qualitatif : les valeurs de cet attribut décrivent une qualité, exprimée de manière alphabétique ou codée par des nombres. Les valeurs possible sont appelé des modalités et sont dénombrables.

1.3 Méthode d'évaluation

Pour une évaluation réaliste et évité le phénomène de sur-apprentissage, il faut construire le modèle uniquement sur une fraction des données (base d'apprentissage) . Afin d'estimer les performances d'un classifieur, il est nécessaire que l'ensemble sur lequel le modèle effectue son apprentissage soit différent de celui sur lequel les mesures sont calculées. Le premier est nommé l'ensemble d'apprentissage X et le deuxième est l'ensemble de test T. Il existe dans la littérature plusieurs méthodes afin de construire les deux ensembles. Les méthodes de validation croisée (cross validation) et d'échantillonnage sont les plus utilisé :

k-fold cross-validation

Cette méthode [11] divise la base F en k sous-ensemble de même taille, nommés 'folds', $F_1, F_2, \ldots, F_k$. Pour k itération, un sous-ensemble est utilisé comme ensemble de test $T_i = F_i$ tandis que le reste des sous-ensembles forment l'ensemble d'apprentissage $X_i = F/F_i = \bigcup_{i \neq j} F_j$. Ainsi le modèle s'entraînera sur l'ensemble X_i et l'évaluation sur le sous-ensemble T_i donnera l'estimation des

performances θ_i. L'estimation globale des performances du classifieur est la moyenne des performances individuelles.

$$\theta = \frac{1}{k} \sum_{1=0}^{k} \theta_i$$

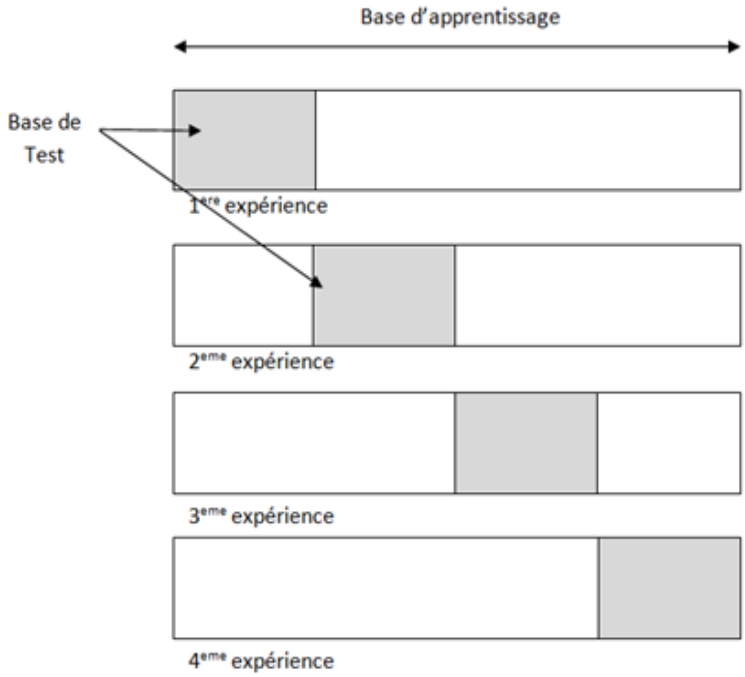


FIGURE I.2 – k-fold cross-validation

Leave-one-out cross-validation

Cette méthode est dérivée de la méthode de validation k-folds cross validation, en prenant $k = n$, n étant le nombre d'exemples. A chaque itération, l'apprentissage est effectué sur tous les exemples moins un, et testé sur un seul exemple, afin de vérifier s'il est prédit correctement. Cette approche est nettement plus coûteuse en calcul que la précédente sans être meilleure.

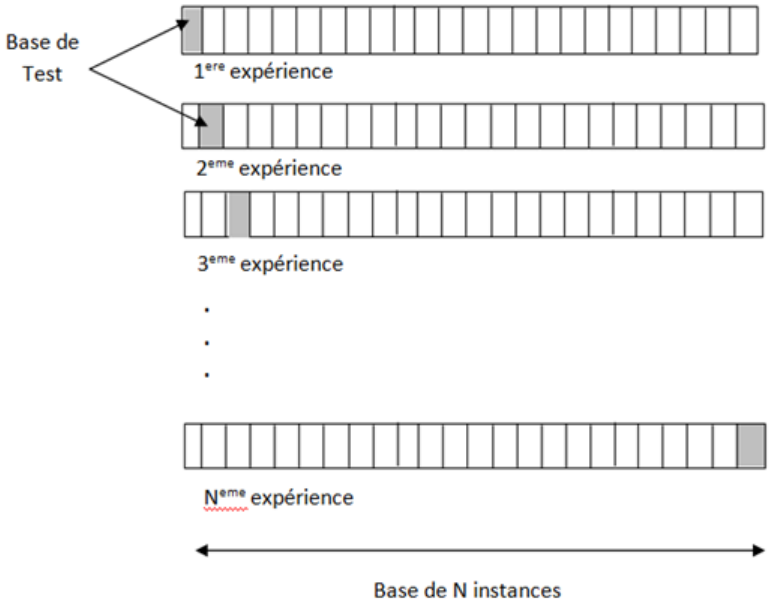


FIGURE I.3 – Leave-one-out cross-validation

Monte Carlo cross-validation

Cette approche [12] divise l'ensemble d'origine en deux sous-ensembles X pour l'apprentissage et T pour la validation en effectuant un tirage aléatoire sans remplacement. L'opération est répétée k fois. A la différence des deux méthodes précédentes, puisque la partition se fait aléatoirement, une instance peut apparaitre plusieurs fois comme appartenant à l'ensemble de test.

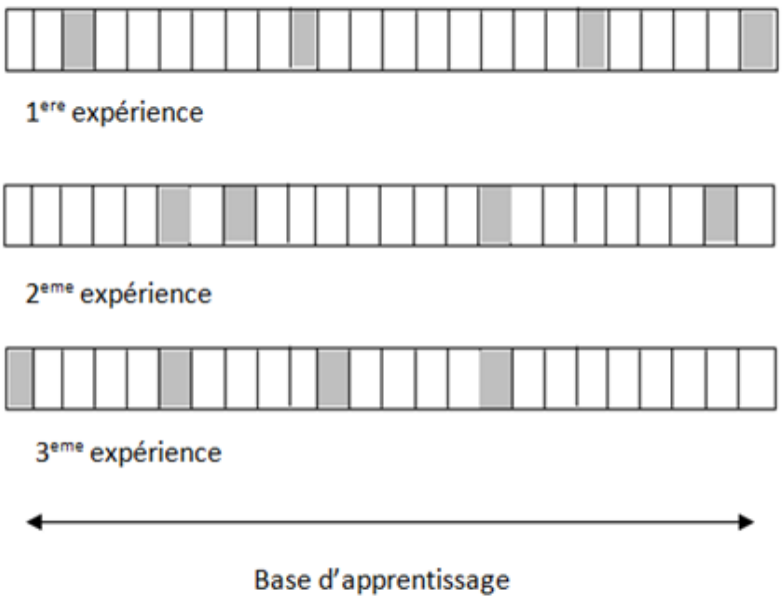


FIGURE I.4 – Monte Carlo cross-validation

Bootstrapping

Le bootstrapping se base sur le principe d'échantillonnage en tirant aléatoirement avec remplacement. Ainsi dans l'ensemble d'apprentissage X il y aura des instances se répétant et des instances absentes. Les instances ne faisant pas partie de l'ensemble X seront utilisées pour former l'ensemble de test T.

FIGURE I.5 – Bootstrapping

1.4 Mesures de performances

L'ensemble de test T est une matrice de n instances et d variables. L'ensemble $c_1, c_2, \ldots, c_k$ représente les k labels et M est le classifieur.

Pour l'instance $x_i \in T$, y_i représente la vrai classe de l'instance et $\hat{y}_i = M(x_i)$ présente la classe prédite par le classifieur M. Afin d'évaluer les performances du classifieur, nous avons recours à des mesures de performance telles que :

Matrice de confusion

Une matrice de confusion permet d'évaluer la qualité d'une classification. Elle est obtenue en comparant les labels $\hat{y}$ prédits par le modèle M avec les classes réelles y. Une matrice de confusion, pour un problème à deux classes, prend la forme suivante :

	Prédites	
Réelles	Positive	Négative
Positive	VP	FN
Négative	FP	VN

TABLE I.2 – Matrice de confusion

Erreur de classification

L'erreur de classification est la fraction d'instances incorrectement prédite, donc plus le taux d'erreur est faible, plus le classifieur est performant :

$e = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}) = \frac{FP+FN}{VP+VN+FP+FN}$

Avec

$$I = \begin{cases} 1 & si \quad y_i = \hat{y} \\ 0 & si \quad y_i \neq \hat{y} \end{cases} \tag{I.1}$$

Taux de classification

Le taux de classification est la fraction d'instances correctement prédite, donc plus le taux est élevé, plus le classifieur est performant :

$$acc = \frac{1}{n}\sum_{i=1}^{n} I(y_i = \hat{y}) = 1 - e = \frac{VP + VN}{VP + VN + FP + FN}$$

L'erreur et le taux de classification sont deux mesures générales qui ne tiennent pas compte des prédictions pour une classe spécifique.

Precision

La précision d'un classifieur M pour une classe c_i est le taux des prédictions correctes de la classe c_i parmi toutes les instances prédites de classe c_i :

$$prec_i = \frac{n_{ii}}{mi} = \frac{VP}{VP + FP}$$

Avec m_i, le nombre d'instances prédites comme appartenant à la classe c_i et $n_i i$, le nombre d'instances correctement classées comme classe c_i.

Rappel (sensibilité)

Le taux de rappel d'un classifieur M pour une classe c_i est la proportion d'éléments bien classés par rapport au nombre d'éléments de la classe c_i.

$$rapp_i = \frac{n_{ii}}{ni} = \frac{VP}{VP + FN}$$

Avec n_i, le nombre d'instances de classe c_i

F-score

La mesure F-score est utilisée quand la précision et le rappel sont importants. En effet, pour augmenter le taux de $rappel_i$ il suffit d'attribuer le label c_i à toute les instances mais ceci implique que $prec_i$ sera faible. Le F-score permet d'obtenir le meilleur compromis entre précision et rappel.

$$F_i = \frac{2}{\frac{1}{prec_i} + \frac{1}{rapp_i}} = \frac{2 \times prec_i \times rapp_i}{prec_i + rapp_i i} = \frac{2n_{ii}}{n_i + m_i}$$

Exemple : Considérons la table de confusion suivante représentant les prédictions d'un classifieur sur un problème de 3 classes et 30 instances (10 instances pour chaque classe) :

	prédites			
Réelles	c_1	c_2	c_3	
c_1	10	0	0	$n_1 = 10$
c_2	0	7	3	$n_2 = 10$
c_3	0	5	5	$n_3 = 10$
	$m_1 = 10$	$m_2 = 12$	$m_3 = 8$	

A partir de cette matrice de confusion, nous pouvons obtenir les mesures suivantes :

— $prec_1 = \frac{n_{11}}{m_1} = \frac{10}{10} = 1.0$

— $prec_2 = \frac{n_{22}}{m_2} = \frac{7}{12} = 0.58$

— $prec_3 = \frac{n_{33}}{m_3} = \frac{5}{8} = 0.62$

— $rapp_1 = \frac{n_{11}}{n_1} = \frac{10}{10} = 1.0$

— $rapp_2 = \frac{n_{22}}{n_2} = \frac{7}{10} = 0.7$

— $rapp_3 = \frac{n_{33}}{n_3} = \frac{5}{10} = 0.5$

— $F_1 = \frac{2 \times n_{11}}{n_1 + m_1} = \frac{20}{20} = 1.0$

— $F_2 = \frac{2 \times n_{22}}{n_2 + m_2} = \frac{14}{22} = 0.63$

— $F_3 = \frac{2 \times n_{33}}{n_3 + m_3} = \frac{10}{18} = 0.55$

— $F = \frac{1}{3}(1 + 0.63 + 0.55) = 0.73$

— $Acc = \frac{n_{11} + n_{22} + n_{33}}{n} = \frac{10+7+5}{30} = 0.73$

1.5 Type d'apprentissage

Le processus de construction du modèle est nommé 'l'apprentissage'. Cette étape s'effectue sur l'ensemble d'apprentissage tandis que la validation des performances du modèle se fait sur l'ensemble test. Il existe plusieurs types d'apprentissage :

Apprentissage Supervisé

L'apprentissage supervisé consiste à prédire la classe d'une instance non labellisée. Le modèle M prédit la classe $\hat{y}$ d'une instance x avec $\hat{y} = M(x)$ et

$\hat{y} \in c_1, c_2, \ldots, c_k$. En apprentissage supervisé, le modèle effectue sont entraînement sur un ensemble d'apprentissage X labellisé (la classe de chaque instance de l'ensemble est connue).
L'objectif de l'apprentissage supervisé est de développer un modèle (classifieur) capable de classer de nouvelles instances (ne faisant pas partie de X) avec un minimum d'erreur. Deux types de problèmes fréquents sont les problèmes de classification et de régression. Pour un problème de classification, y correspond à un ensemble fini de classes auxquelles peuvent appartenir x. Par contre pour celui de la régression, y correspond à un ensemble de valeurs continues.
Il existe plusieurs classifieurs dans la littérature tels que k-plus proche voisin, les arbres de décision ou les réseaux de neurones.

Apprentissage Non-supervisé

Le but de l'apprentissage non-supervisé est de trouver une structure et partitionner les instances de l'ensemble d'apprentissage suivant cette structure en groupes généralement nommé 'clusters' [13]. Ainsi les instances appartenant au même groupe sont similaire, tandis que ceux appartenant à des clusters distinct ont des caractéristiques différentes. Dans le cas de l'apprentissage non-supervisé l'ensemble d'entraînement est non labellisé.

Apprentissage Semi-supervisé

L'apprentissage semi-supervisé est à mi-chemin entre l'apprentissage supervisé et non-supervisé [13]. En effet, c'est un processus où l'apprenant (modèle) apprend des deux types d'instances ; labellisées et non labellisées. Ainsi l'ensemble d'apprentissage peut être divisé en deux sous-ensemble $X = X_l, X_u$ avec $X_l = (x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)$ et $X_u = x_l + 1, x_l + 2, \ldots, x_l + u$. L'approche la plus utilisées en apprentissage semi-supervisé est d'entraîner un apprenant sur les données labellisées X_l puis d'utiliser le modèle obtenu pour labelliser les instances de l'ensemble X_u.

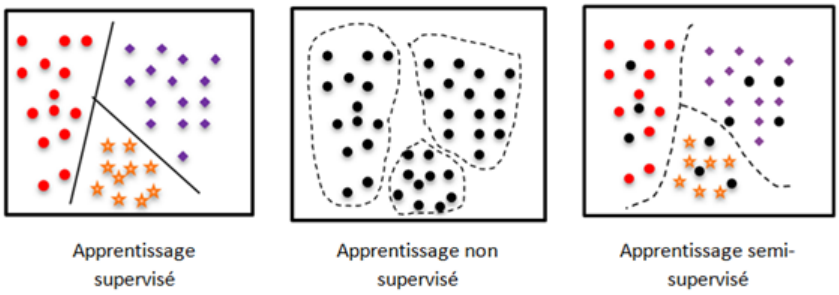


FIGURE I.6 – Type d'apprentissage

Apprentissage Actif Une variation de l'apprentissage semi-supervisé est l'apprentissage actif. Ce type d'apprentissage a à sa disposition des instances labellisées X_l et non labellisées X_u ou simplement des X_u. Son principe est de sélectionner les instances les plus informatives de l'ensemble X_u afin qu'elles soient

labellisées par un expert [14, 15]. Par exemple, dans le cas de détection des spam, l'approche serai de demander à l'utilisateur si un e-mail est un spam ou non. Après un certain nombre de classification, le processus construit un apprenant capable de classer automatiquement un e-mail comme étant un spam ou non.

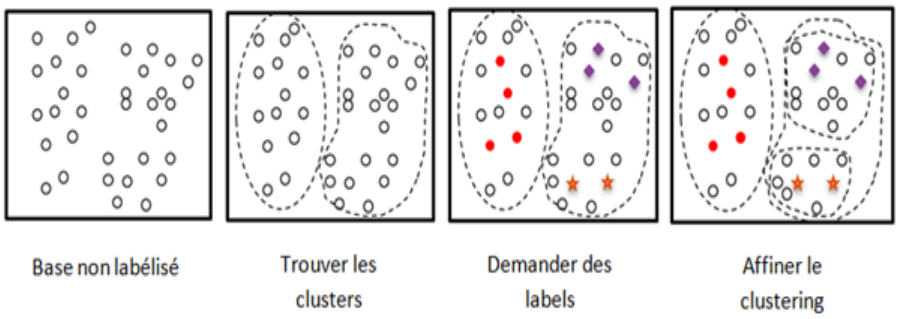


FIGURE I.7 – Apprentissage actif

1.6 Algorithme de classification

Dans le cadre de cette thèse, nous nous pencherons sur les problèmes de classification en apprentissage supervisé. Dans la suite de la section, nous présentons quelques uns des algorithmes de classification les plus populaires.

k-plus proches voisins (k-nearest neighbor)

La méthode des k-plus proches voisins (K-ppv, k-NN) [16] se base sur une comparaison directe entre le vecteur caractéristique de l'instance à classer et les vecteurs des instances de la base d'apprentissage. La comparaison consiste en un calcul de distances entre ces instances. Puis à l'instance a classer est assignée la classe majoritaire parmi les classes des k instances les plus proches. Il existe plusieurs distances employées par l'algorithme k-nn pour comparer deux instances, parmi ces métriques nous pouvons citer :

Notons par $X_i = (x_i1, , x_i2, , ..., x_in)$ le vecteur caractéristique de l'instance i, avec n le nombre de variables et par p et q deux instances à comparer.

Distance euclidienne :

$$D(X_p, X_q) = \sqrt{\sum_{i=1}^{n}(x_pi - x_qi)^2} \tag{I.2}$$

Distance de Manhattan : $D(X_p, X_q) = \sum_{i=1}^{n} |(x_pi - x_qi)|$

Distance de Minkowski : $D(X_p, X_q) = (\sum_{i=1}^{n}(x_pi - x_qi)^r)^{\frac{1}{r}}$

Distance de Tchebychev : $D(X_p, X_q) = max_{i=1}^{n}|x_p i - x_q i|$

Les principaux inconvénients de cette méthode sont le nombre d'opérations nécessaires pour classer une instance dans le cas d'une grande base d'apprentissage ainsi que sa sensibilité au bruit.

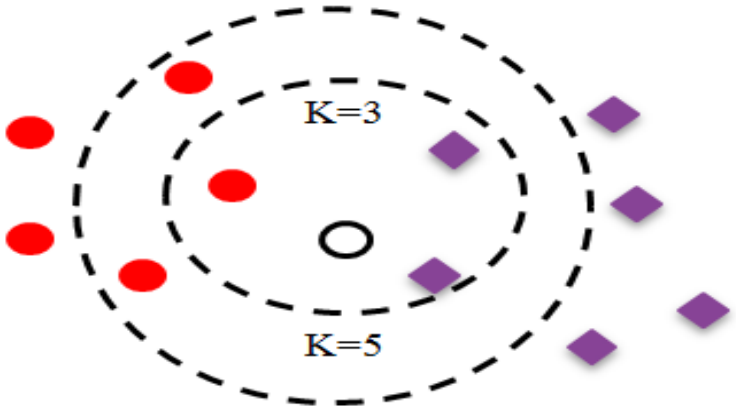


FIGURE I.8 – K-plus proche voisin

Machine à vecteurs de support (support vector machine, SVM)

Les machines à vecteurs de support ou séparateurs à vaste marge se basent sur les théories de Vapnik [17]. Un classifieur linéaire cherche un séparateur linéaire (frontière) entre les instances de classes différentes. Dans le cas d'un problème linéairement séparable, il existe plusieurs séparateurs possibles.

L'objectif du SVM est de trouver l'hyperplan (séparateur) dont la distance de deux instances les plus proches de classe différente (vecteur de support) est maximale. On appelle cette distance marge.

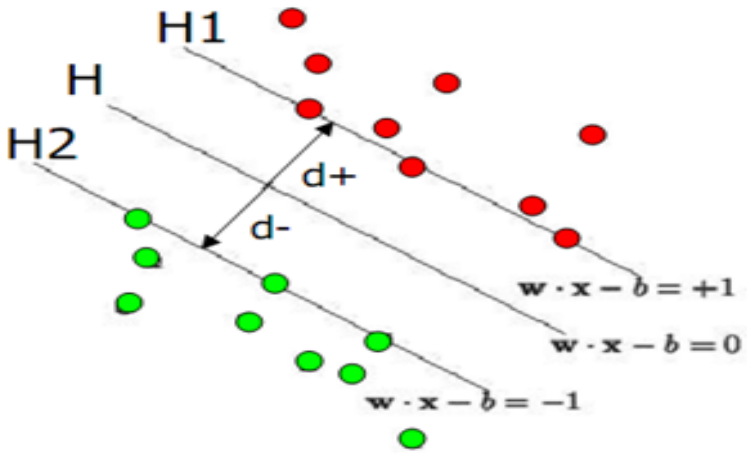


FIGURE I.9 – Machine à vecteurs de support

Les exemples les plus proches qui permettent de déterminer cet hyperplan

sont appels vecteurs de support, ou encore exemples critiques. L'hyperplan optimal est défini par le vecteur de poids w vérifiant l'équation :

$$h(x) = w^T x + b = w_1 x_1 + w_2 x_2 + \ + w_d x_d + b$$

Avec w un vecteur de poids et b un biais. Pour les points appartenant à l'hyperplan, nous avons :

$$h(x) = w^T x + b = 0$$

Cas linéairement séparable Si le problème est linéairement séparable, l'hyperplan recherché divise l'espace en deux sous-espaces contenant chacun des instances appartenant à une classe distincte. Dans ce cas, la fonction $h(x)$ de l'hyperplan est utilisé pour prédire la classe y d'une instance x, suivant la règle de décision [10] :

$$y = \begin{cases} +1 & if h(x) > 0 \\ -1 & if h(x) < 0 \end{cases}$$

D'un autre coté, la distance entre un point situé sur $H1$ et l'hyperplan est calculé par l'équation suivante :

$$\frac{|w.x + b|}{||w||} = \frac{1}{||w||}$$

Donc la distance entre les deux hyperplans représentant la marge est :

$$d = \frac{2}{||w||}$$

Le problème revient alors à trouver les valeurs de w et b telle que la marge d est maximale ou encore telle que la valeur inverse $d = \frac{||w||}{2}$ soit minimale.

Cas linéairement non séparable Généralement le problème à traiter n'est pas linéairement séparable. Dans ces cas, il faut faire une séparation plus souple en autorisant certaines instances à avoir une marge inférieure à 1, voire négative [18]. Pour cela il faut ajouter une variable de relâchement ξ_i des contraintes à minimiser avec $\xi_i > 0$:

$$y = \begin{cases} w.x_i + b = 1 - \xi_i & if h(x) > 0 \\ w.x_i + b = -1 + \xi_i & if h(x) < 0 \end{cases}$$

La variable de relâchement représente la distance séparant une instance placée au mauvais côté de l'hyperplan. Afin de limitée le nombre d'exemples avec une valeur positive de ξ_i, la violation est pénalisée par $C.\xi_i^r$. Avec C et r des paramètres définis par l'utilisateur pour réguler la souplesse du modèle [18].

Généralement, la valeur de r est 1. En parallèle, avec une valeur faible de C nous obtenons une marge souple tandis qu'une valeur élevée de C minimise les erreurs et nous obtenons une marge plus stricte. Ainsi, la fonction à minimiser est :

$$\frac{||w||}{2} + C \sum \xi_i$$

Dans le cas où la frontière entre les classes n'est pas linéaire, le problème est transposé à une dimension supérieure pour le rendre linéairement séparable.

Prenons l'exemple représente dans la figure I.10, chaque instance est décrite par deux variables $x = x_1, x_2$. Ce problème n'a pas de frontière linéaire entre les classes. La solution serait donc de projeter chaque instance dans un nouvel espace à trois dimension grâce à l'équation suivante :

$$z = x_1^2, \sqrt{2}x_1x_2, x_2^2$$

Ainsi chaque instance est décrite par trois variables dérivesé de l'ancienne représentation.

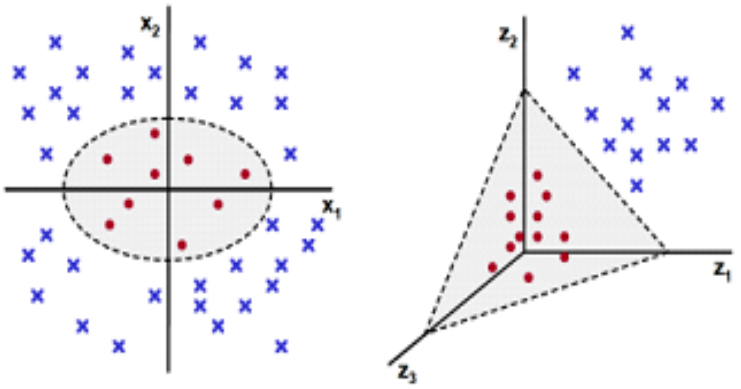


FIGURE I.10 – Cas non linéairement séparable

Arbre de décision

Dans certain domaine, comme le diagnostic médical, il est parfois nécessaire que la procédure de classification soit compréhensible et interprétable par l'utilisateur. Les arbres de décision, grâce à leur représentation graphique, répondent à cette contrainte.

Le formalisme des arbres de décision permet de classifier un nouvel objet en testant ses caractéristiques les unes à la suite des autres. Les exemples de l'ensemble d'apprentissage sont récursivement divisés par des tests définis sur les caractéristiques pour obtenir des sous-ensembles d'exemples ne contenant que des exemples appartenant tous à une même classe.

Les arbres de décision sont construits de haut en bas selon une structure d'arbre, où chaque nœud est soit :

— Une feuille : représente la classe des instances, soit
— Un nœud de décision : permettant de tester les instances sur la valeur d'un attribut, il possède deux ou plusieurs branches.

L'algorithme général d'un arbre de décision est le suivant :

Algorithm 1 Algorithme d'arbre de décision

Début
Entrées : base d'apprentissage S
Début
initialiser à l'arbre vide ; // la racine est le nœud courant
Répeter
décider si le nœud courant est terminal ;
Si le nœud est terminal alors affecter une classe ;
sinon sélectionner un test et créer le sous-arbre ;
passer au nœud suivant non exploré s'il existe ;
jusqu'à obtenir un arbre de décision ;
Fin

Les algorithmes existants (CART [19], ID3 [20], C4.5 [21]...) diffèrent essentiellement par leur façon de choisir, à une étape donnée, et parmi les caractéristiques disponibles, la caractéristique de segmentation et par le critère d'arrêt.

Algorithme ID3 Proposé par Quinlan [20] l'algorithme Iterative Dichotomiser 3 (ID3) est construit en sélectionnant à chaque nœud l'attribut qui maximisera le gain d'information en d'autre terme l'attribut permettant la meilleure discrimination entre les classes à ce niveau de l'arbre. La fonction utilisée pour calculer le gain d'information est :

$$Gain(n, T) = Entropie(p) - \sum_{j=1}^{n}(p_i * Entropie(p_j))$$

Avec Entropie, représentant l'entropie de Shannon :

$$Entropie(p) = -\sum_{i=1}^{n} p_i * log(p_i)$$

Algorithme C4.5 Proposé par Quinlan [21], cet algorithme apporta plusieurs amélioration à l'arbre de décision ID3 :

— Il peut traiter des données de types catégorielles et continues tandis qu'ID3 ne peut manipuler que des données catégorielles.
— C4.5, à la différence de ID3, utilise une phase d'élagage lui permettant d'éviter le sur-apprentissage en éliminant les branches superflues.

D'un autre coté, l'algorithme C4.5, utilise un test, nommé Gain Ratio, qui maximise le gain d'information :

$$GainRatio(p, T) = \frac{Gain(p,T)}{SplitInfo(p,T)}$$

Avec

$SplitInfo(p, test) = -\sum_{j=1}^{n} p'(\frac{j}{p}) * log(p'(\frac{j}{p}))$

Où et $P'(j/p)$ est la proportion des éléments présents à la position p prenant la j^eme valeur de test. La valeur de la fonction Splitinfo est grande lorsqu'un attribut comporte peu d'instances pour chaque valeur. Donc, en choisissant le test qui maximise le GainRatio, on choisi l'attribut générant le moins de sous-arbres.

Algorithme Classification And Regression Trees (CART) L'algorithme CART fut proposé par Breiman [22]. A la différence des arbres de décision cités précédemment, chaque nœud de l'arbre CART ne possède que deux fils et le critère utilisé pour la division est l'index de Gini.

2 Conclusion

Dans ce chapitre, nous avons présenté les concepts de base de l'apprentissage automatique, les différents types d'apprentissages, ainsi que plusieurs méthodes d'évaluation et mesures de performances.
Nous avons aussi introduit, quelques algorithmes de classification les plus répandus tels que les arbres de décision, les machines à vecteurs de supports et l'algorithme des k-plus proche voisins.

Dans le chapitre suivant, nous introduisons l'état de l'art des méthodes de sélection d'instance. Nous présentons la taxonomie de ces approches ainsi qu'un aperçu des méthodes les plus populaires.

Chapitre II

Les méthodes de sélection d'instances

Introduction

En apprentissage supervisé, nous supposons qu'un label $y \in Y$ dépend des valeurs d'un ensemble de variables $X = (X1, ..., Xd)$ tels que $x \in \mathbb{R}$. Le but est de trouver une fonction F tel que pour n'importe quel ensemble de variables $X = (X_1, ..., X_d)$, le label associé est correctement prédit.

Les principaux buts de l'apprentissage supervisé sont :

— obtenir un « bon » modèle dont la prévision est proche de la vraie valeur
— obtenir un modèle dont le temps de construction et de prévision soit réduit.

L'apprentissage a des applications dans des domaines très divers tels que la reconnaissance vocale, la vision par ordinateur, le traitement des langages naturels, les moteurs de recherche, les systèmes de recommandations.

Plusieurs algorithmes ont été proposés pour effectuer un apprentissage supervisé tels que : les arbres de décision, les réseaux de neurones artificiels, le K-plus proches voisins (k-NN), les algorithmes évolutionnaires ou l'algorithme SVM. Malheureusement, ayant été conçus quand les bases disponibles étaient plus réduites, ces algorithmes bien qu'efficace, souffrent d'un problème de scalabilité.

En effet, les méthodes nécessitant de conserver en mémoire toutes les instances pour la prise de décision, tels que le k-NN, voient leurs besoins en ressource et leur temps d'exécution augmenté en fonction de la taille de la base. D'un autre coté, les algorithmes tels que les arbres de décision perdent de leur interprétabilité. Sans oublieé que plusieurs de ces algorithmes sont sensible au bruit et voient leur performances se dégradée si la proportion d'instance bruitée est élevée dans la base d'apprentissage.

Une solution possible aux problèmes précédemment cité est d'effectuer une étape préliminaire de sélection d'instance visant à nettoyer et réduire la taille de la base d'apprentissage en éliminant les instances bruitées ou superflues.

1 Sélection d'instances

La sélection d'instances consiste à chercher un sous-ensemble S de l'ensemble original d'apprentissage A tel que S donne des résultats équivalents que A. La figure II.1, représente le processus de sélection d'instances.

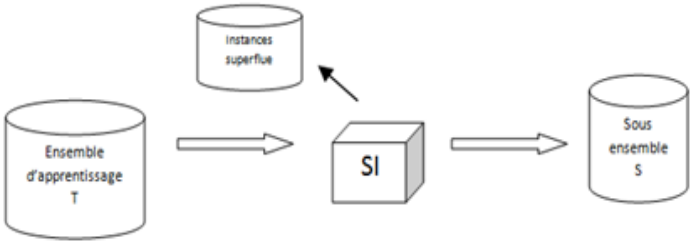


FIGURE II.1 – Processus de sélection d'instance

Les algorithmes de réduction d'instances sont classés en fonction de la représentation des instances. En effet, l'algorithme peut soit sélectionner un sous-ensemble de l'ensemble original soit de créer de nouvelles instances appelé « prototype » [23], [24] :

Sélection : des instances de l'ensemble original sont retenues pour l'étape d'apprentissage afin de réduire le temps d'apprentissage et le coût de stockage.

Remplacement : des prototypes sont créés pour représenter un cluster même si l'instance ne correspond à aucun point de l'ensemble original. Les données généralisées peuvent être sous forme de hyper-rectangles [25], de prototypes [26], ou de règle [27].

2 Objectif des algorithmes de sélection

La sélection d'instances permet d'optimiser les résultats des traitements de données. Cette étape a pour principaux objectifs de [1] :

Faciliter l'apprentissage : la plupart des algorithmes d'apprentissage sont sensibles à la taille de l'ensemble d'apprentissage. En effet, si la base est trop grande, l'algorithme ne pourra pas s'exécuter en un temps convenable. L'étape de sélection d'instances permet de réduire la taille de la base et permet ainsi à un algorithme d'apprentissage de s'exécuter correctement.

Focaliser sur l'information utile : une base de données contient tout type d'information et grâce à l'étape de sélection, il est possible de se focaliser sur la partie la plus informative des données et optimiser ainsi l'apprentissage.

Nettoyer le bruit : l'étape de sélection d'instance élimine les instances bruitées et redondantes.

3 Taxonomie des méthodes de sélection d'instances

Il existe plusieurs classifications possibles pour les algorithmes de sélection en fonction de la direction de recherche, de la méthode d'évaluation ou du type

de sélection.

3.1 Direction de recherche

Un algorithme de sélection peut être classé en fonction de la direction de recherche [23, 28] :

Incrémentale : un algorithme incrémental commence avec un sous-ensemble S vide puis ajoute progressivement les instances de l'ensemble A qui remplit le critère de sélection. Ce type d'algorithme est rapide car il démarre avec peu de données à traiter. Par contre, pour la même raison il a plus tendance à sélectionner des instances bruitées. Un autre inconvénient de ces approches est qu'elles sont sensibles à l'ordre de présentation des instances. De ce fait, la meilleure option est que la présentation soit aléatoire car ce type de recherche doit être capable de traiter de nouvelles informations. Une variation des approches incrémentales est d'ajouter les instances en examinant à chaque fois toutes les instances présentes dans l'ensemble A.

Le principal avantage d'une méthode incrémentale est qu'elle est capable de traiter des instances ajoutées après la première sélection. Cette capacité les rend très intéressant pour l'apprentissage en ligne. D'un autre coté, ils sont plus rapide et requièrent moins de ressources mémoire que les autres types de recherche.

Décrémentale : un algorithme décrémental commence avec un ensemble $S = A$ puis commence à éliminer des instances. Il est tout aussi sensible à l'ordre de présentation des instances que les algorithmes incrémentals mais il dispose de toute les données à n'importe quel moment de l'exécution. Un autre désavantage est le coût élevé du calcul et des ressources mémoire car ce type d'approche nécessite d'avoir accès a toutes les données. Par contre, les approches décrémentales obtiennent un plus grand taux de réduction que les approches incrémentales.

Batch : ce type d'algorithme décide auparavant quels instances répondent à un critère de suppression avant de les éliminer à la fois. Les performances de ce type d'algorithme dépendent fortement du critère d'élimination. Ainsi que les algorithmes à recherche décrémentale, les algorithmes batch souffrent d'une complexité supérieure à celle des approches incrémentales.

Mixte : ces méthodes commencent avec un sous-ensemble S pré-sélectionné aléatoirement ou par l'une des méthodes précédentes puis des instances peuvent être ajoutées ou supprimées en fonction d'un critère défini. Ce type de recherche permet de modifier la sélection afin d'optimiser la classification. Il présente les mêmes inconvénients que la recherche décrémentale. Ce genre d'algorithmes est similaire aux algorithmes incrémentals indépendant de l'ordre où la suppression d'instances est autorisée.

Fixe : pour ces méthodes la taille de S est fixée au début du programme. Cette recherche est un type de recherche mixte où la taille du sous-ensemble est prédéfinie.

3.2 Méthodes d'évaluation

Les méthodes de sélection peuvent être classées en deux catégories en fonction du type du critère de sélection :

Wrapper : le critère de sélection est basé sur les résultats obtenus par un classifieur.

Filtre : le critère de sélection utilise une fonction de sélection qui n'est pas basée sur la sortie d'un classifieur.

3.3 Type de sélection

Dans une base de données, nous pouvons distinguer trois types d'instances : Les instances internes sont localisées au centre des clusters, les instances frontalières se trouvent à la séparation entre les classes et les instances bruitées. Les méthodes de sélection diffèrent les unes des autres en fonction du type d'instances sélectionné.

Condensation : les algorithmes de condensation conservent les instances proches des frontières et éliminent les instances internes. Ces méthodes se basent sur l'idée que les instances internes n'interviennent pas dans la discrimination entre les classes et les considèrent superflu pour la prise de décision. Et par conséquence, les instances internes peuvent être éliminées sans dégrader gravement les performances du classifieur. Ce genre d'algorithmes obtient un taux de réduction assez élevé car dans une base de données la majorité d'instances sont des instances internes tandis que les instances frontalières sont moins nombreuses. Par contre, même si ces algorithmes obtiennent un bon taux de classification sur la base d'apprentissage, il n'est pas garanti d'obtenir de bonnes performances sur l'ensemble de test.

Edition : ces méthodes éliminent les instances se trouvant aux frontières entre les classes. Leur principal objectif est d'éliminer les instances bruitées ou celles qui ont une classe différente à celle de leur voisin. Étant donné que les instances internes sont conservées, le taux de réduction est assez faible. Par contre, le taux de classification sur la base de test est amélioré.

Hybride : ces algorithmes visent à construire un sous-ensemble S qui maintient ou améliore les performances du classifieur en éliminant ou conservant des instances internes ou frontalières.

Méta-heuristique : ces approches utilisent les algorithmes méta-heuristiques pour effectuer la réduction de la base.

3.4 Critères d'évaluation

Les algorithmes de sélection d'instance sont évalués sur un certain nombre de critères afin de comparer leurs performances. Ces critères sont : le taux de réduction, la tolérance au bruit, le taux de classification et le temps d'exécution [28].

Taux de réduction : l'un des principaux objectifs d'un algorithme de sélection d'instance est la réduction des ressources mémoire nécessaire à l'apprentissage. D'un autre coté, la réduction du nombre d'instances conduit invariablement à une réduction du temps d'apprentissage de l'algorithme de classification.

Tolérance au bruit : en présence de bruit dans la base d'apprentissage nous devons faire face à deux problèmes. En premier lieu, les performances de classification se dégradent surtout si les instances bruitées sont considérées comme appartenant aux frontières et sont conservées à la place des bonnes instances. Deuxièmement, le taux de réduction diminue car il faut conserver plus d'instances pour effectuer la discrimination entre les classes.

Taux de classification : un algorithme de sélection doit réduire la taille de la base sans pour autant réduire considérablement le taux de classification.

Temps d'exécution : il faut aussi que le temps d'exécution ne soit pas trop long car il n'y a aucun intérêt à utiliser un algorithme de réduction qui demande autant de temps que l'algorithme d'apprentissage. Faute de quoi, il devient impraticable pour des problèmes réels.

4 Algorithmes de sélection

Un grand nombre d'algorithmes de sélection d'instance ont été développés au cours des années. Dans cette section, nous présentions une série d'algorithme de sélection classé selon le type de sélection précédemment cité :

4.1 Algorithme de condensation

Cette catégorie englobe l'ensemble d'algorithme visant à sélectionner les instances se situant au bord des frontières entre les classes aussi connu sous le nom d'instances frontalières. Parmi ces approches, nous pouvons citer :

Condensed Nearest Neighbor (CNN) L'algorithme CNN [29] est une méthode incrémentale, elle démarre le processus avec l'ensemble $S = \emptyset$, puis une instance de chaque classe est aléatoirement ajoutée à S puis chaque instance de T est labélisée par les instances appartenant à S. Si une instance est mal classée, elle est ajoutée à S puis l'opération est refaite jusqu'à ce que toutes les instances de A§ soient bien classées. Cette méthode assure que S classe correctement toutes les instances de A mais son taux de réduction est faible.

L'algorithme CNN est sensible à l'ordre de présentation des instances de A. En effet, si l'ordre de A change, le sous ensemble S obtenu est différent.

Algorithm 2 CNN algorithm

Input Ensemble d'apprentissage A
$M \leftarrow |A|$
$P = x_1$
$flag = vrai$
while flag **do**
  $flag = faux$
  for $i = 1...M$ **do**
    $\overline{C}(x_i)$ =kNN(P; x_i)
    if $\overline{C}(x_i) \neq C(x_i)$ **then**
      $P = P \cup x_i$
      $A = A \setminus x_i$
      $flag = vrai$
    end if
  end for
end while
Sortie : P

Generalized Condensed Nearest Neighbor (GCNN) En 2006, Chou et al. proposent une variation du CNN nommé GCNN [30]. Les processus d'initialisation et de sélection sont modifiés. L'ensemble S est initialisé par une instance représentative de chaque classe choisi par vote. Chaque instance x de $A1 \subseteq A$, l'ensemble d'instances de class l émet un vote pour son plus proche voisin dans $A1$ x. Finalement l'instance avec le plus grand nombre de vote est ajoutée à S.

Pour l'étape de sélection, le critère d'absorption qui se calcule en fonction du plus proche voisin et plus proche ennemie (plus proche voisin de classe différente) d'une instance p dans S est utilisée. Une instance x est ajouté à S si elle n'est pas fortement absorbé. Une instance est fortement absorbée si pour les le plus proche voisin p et le plus proche ennemi q dans S :

$$||x - q|| - ||x - p|| > \rho d$$

Avec $\rho \in [0, 1]$ et d la distance minimum entre deux instances de classes différentes.

Le processus s'arrêtera quand les instances de A respecteront un seuil d'absorption proposé par l'utilisateur.

Reduced Nearest Neighbour (RNN) L'algorithme RNN est une modification de CNN proposé par [31]. A l'inverse de CNN, RNN est une approche décrémen-

tale. L'algorithme débute avec $S = A$ et supprime une instance si son élimination ne fait pas décroitre les performances de classification.

Algorithm 3 RNN algorithm

Input Ensemble d'apprentissage A
$M \leftarrow |A|$
$P = x_1$
for $i = 1...M$ **do**
  $\overline{C}(x_i)$ =kNN(P; x_i)
  if $\overline{C}(x_i) \neq C(x_i)$ **then**
    $P = P \cup x_i$
  end if
end for
$N \leftarrow |P|$
for $i = 1...N$ **do**
  $P = P \setminus x_i$
  for $j = 1...M$ **do**
    $\overline{C}(x_j)$ =kNN(P; x_j)
    if $\overline{C}(x_j) \neq C(x_j)$ **then**
      $S = S \cup x_i$
    end if
  end for
end for
Sortie : S

Pair Opposite Class-Nearest Neighbor (POC-NN) L'algorithme Pair Opposite Class-Nearest Neighbor (POC-NN) proposé par Raicharoen et Lursinsap en [32] sélectionne les instances qui se situent au bord des clusters de A. Le processus de sélection commence par déterminer l'instance central mi de chaque classe, une instance est frontalière si elle est la plus proche voisine d'un mi de classe différente.

Prototype Selection based on Clustering (PSC) Pour construire le sous-ensemble S cet algorithme [24] commence par extraire les clusters de l'ensemble d'apprentissage A en utilisant l'algorithme C-Means. Puis les centres de chaque cluster homogène (toutes les instances sont de la même classe) sont ajoutés à S. Si le cluster n'est pas homogène, toutes les instances de classe différente à la classe majoritaire sont ajoutées à l'ensemble S ainsi que leur plus proche voisin.

Patterns by Ordered Projections (POP) L'algorithme POP [33] partitionne l'espace représenté par l'ensemble d'apprentissage A en rectangle homogène et supprime les instances se trouvant au centre. Chaque instance est représentée dans un espace de $d - dimension$ (avec d le nombre de variables) et elles sont étudiées séparément. Une région est considérées comme homogène si tous ses éléments appartiennent à la même classe. Afin de déterminer si une instance est interne dans la i_{eme} dimension, toutes les valeurs de la variable i sont triées par ordre croissant et en cas d'égalité, les éléments sont triés par classe. Les séquences obtenues sont divisées en intervalle d'instance appartenant à la même classe.

Les instances se trouvant au bord des intervalles sont considérées comme des instances appartenant aux frontières, les autres sont internes. La valeur « $weakness$ » d'une instance est le nombre de fois qu'elle est interne. Si $weakness =$

d alors cet élément est interne dans toute les dimensions et il sera éliminé du sous-ensemble S.

Algorithm 4 POP algorithm

Input Ensemble d'apprentissage A
$M \leftarrow |A|$
$S = A$
for $i = 1...M$ **do**
$weakness(x_i) = 0$
end for
Pour chaque attribut non-nominal a^i
Trier les instances de A sur la valeur de a^i avec l'algorithme Quicksort
Trier cette séquence en fonction des classes
Diviser la séquence en intervalles
Pour chaque point interne incrémenter la valeur $weakness$ de 1
Fin Pour
Pour chaque attribut nominal a^i
Pour chaque valeur v_j^i
Déterminer l'ensemble $V_j = \{x \in A / x^i = v_j^i\}$
Déterminer $x* = argminweakness(x)$
Diviser la séquence en intervalles
Pour chaque $x \in V_j / \{x*\}$ incrémenter la valeur $weakness$ de 1
Fin Pour
Fin Pour
Eliminer x de S si $weakness(x) = d$
Sortie : S

Shell extraction algorithm Cet algorithme proposé par [34] exploite la géométrie de la base de donnée afin de sélectionner les instances proches des frontières et ainsi optimiser les performances de l'algorithme SVM.

Il existe bien d'autre algorithmes de condensation tels que : Reduced nearest neighbor RNN [31], Ullmann [35], Selective nearest neighbor SNN [36], Tomek condensed nearest neighbor TCNN [37], Modified condensed nearest neighbor MCNN [38], Mutual neighborhood value MNV [39], Shrink [40], Modified selective subset MSS [41], Generalized condensed nearest neighbor GCNN [26] ou Fast condensed nearest neighbor FCNN [42] parmi tant d'autre.

4.2 Algorithme d'édition

Cette catégorie englobe l'ensemble d'algorithme visant à sélectionner les instances se situant au centre des clusters aussi connu sous le nom d'instances internes. Parmi ces approches, nous pouvons citer :

Edited Nearest Neighbor (ENN) Edited Nearest Neighbor proposé par WILSON en 1972 élimine une instance p si elle est mal classée par ses k-ppv [43]. Cette méthode est souvent utilisée comme un filtre car elle supprime les instances bruitées i.e. des instances de classe différente à celle de leur voisinage.

Algorithm 5 Algorithme ENN

Input Ensemble d'apprentissage A
  $n \leftarrow |A|$;
  $P \leftarrow A$;
  for $j = 1...n$ **do**
    $\overline{C}(x_i) = k - NN(k, (A\ x_i), x_i)$
    if $\overline{C}(x_i) \neq C(x_i)$ **then**
      $P \leftarrow P\ x_i$
    end if
  end for
  Sortie P

Repeated ENN (RENN) Une variation de ENN nommé RENN est proposé par [43] , cet algorithme applique itérativement ENN jusqu'à ce que tous les prototypes de S aient la même classe que leurs k-plus proche voisins.

Modied Edited Nearest Neighbor (MENN) L'algorithme MENN [44] est similaire à ENN. L'algorithme commence avec $S = T$ puis chaque instance x_i est éliminée de S si elle a la même classe que ses $k + l$ proches voisins avec l qui représente toute les instances à la même distance que son dernier voisin.

All k-NN All k-NN est une extension de l'algorithme ENN [45]. L'algorithme marque chaque instance mal classée par ces i plus proches voisins. A la fin de la boucle, toutes les instances marquées seront éliminées.

Algorithm 6 Algorithme All-knn

Input Ensemble d'apprentissage A
  $M \leftarrow |A|$
  $P \leftarrow A$
  $flag \leftarrow vrai$
  for $j = 1...k$ **do**
    for $i = 1...M$ **do**
      $\overline{C}(x_i) \leftarrow 1 - NN(P; x_i)$
      if $\overline{C}(x_i) \neq C(x_i)$ **then**
        $flag(x_i) \leftarrow faux$
      end if
    end for
  end for
  for $i = 1...M$ **do**
    if $flag(x_i) = faux$ **then**
      $P \leftarrow P \setminus x_i$
    end if
  end for
  Sortie S

Nous pouvons aussi citer : MultiEdit [46], Relative Neighborhood Graph Editing RNGE [47], Nearest centroid neighbor edition NCNEdit [48], Edited normalized radial basis function ENRBF [49], Edited nearest neighbor estimating class probabilistic ENNProb [50], Edited nearest neighbor estimating ENNTh [50], Local support vector machines noise reduction LSVM [51] ou Fuzzy rough prototype selection FRPS [52].

4.3 Algorithme hybride

Cette catégorie d'algorithmes élimine des instances des deux types afin d'obtenir un sous-ensemble S capable de garder les mêmes performances que l'en-

semble A sur la base de test.

Decremental Reduction Optimization Procedure (DROP) Wilson et Martinez en [23] proposent une série de cinq méthodes nommées DROP1,..., DROP5 pour la sélection d'instances. À la différence des précédents algorithmes ils utilisent le principe d'associer à une instance p (les instances pour lesquels p est dans les k-plus proches voisins), donc une instance p est éliminée si ses associés sont bien classés sans p.

— DROP1 : supprime une instance si ses associés dans S sont correctement classés sans elle. Cet algorithme tend à éliminer les instances bruités et internes, puisque leur associés seront correctement classés même sans leur présences. En d'autre terme, DROP1 conserve les instances frontalières non bruitées.
— DROP2 : le principal inconvénient avec DROP1 est que si les voisins d'une instance bruitée sont éliminés, elle sera conservée. Afin de résoudre ce problème DROP2 supprime une instance si ses associés dans tout l'ensemble A sont correctement classés sans elle au lieu de considérer que les instances restantes dans S. Donc DROP2 considère même les instances précédemment supprimées de S.
— DROP3 : l'inconvénient avec DROP2 est que les instances bruitées sont considérées comme frontalière et sont conservées. D'un autre coté, si l'instance bruitée se situe au centre du cluster, elle conduit à la conservation des instances internes voisines. Afin de contourner ce problème DROP3 applique un filtre de bruit en éliminant toute les instances mal classé par leur k-plus proches voisins.
— DROP4 : cet algorithme est similaire à DROP3 mais à l'étape de filtrage n'élimine une instance mal classée par ces voisins que si son élimination ne dégrade pas le taux de classification.
— DROP5 : est basé sur DROP2 mais commence par supprimer les instances plus proche de leur plus proche ennemie. D'après les résultats des expérimentations dans [23], DROP1 obtient le taux de réduction le plus élevé suivi par DROP2 et DROP5, par contre son taux de classification est le plus faible. DROP3 est celui avec le meilleur taux de classification mais le taux de réduction le plus faible.

Iterative Case Filtering (ICF) Cet algorithme proposé par Brighton et Mellish dans leur article [3] est basé sur les ensembles de voisins et d'associés nommés respectivement « Coverage » et « Reachable ». Dans cette méthode, si $|Reachable(p)| > |Coverage(p)|$ alors l'instance p est marquée puis toutes les instances marquées seront éliminées.

Instance-Based Learning Algorithms (IB) Dans leur article [53], AHA & al. ont proposé une série l'algorithmes nommés Instance-Based Learning Algorithms (IB) où une instance p de T est sélectionnée si elle est mal classée par son plus proche voisin dans S.

— *IB2 :* l'algorithme commence avec le sous-ensemble S vide, puis chaque instance de l'ensemble A est sélectionnée si elle est mal classée par les instances présentes dans S. IB2 est similaire à CNN sauf qu'il n'initialise pas S avec une instance de chaque classe et n'effectue qu'un seul passage de sélection. A l'instar de CNN, IB2 est sensible au bruit, car les instances bruitées sont généralement mal classées par leur voisins.

Algorithm 7 Algorithme IB2

Input Ensemble d'apprentissage A
$M \leftarrow |A|$
$P \leftarrow \emptyset$
for $i = 1...M$ **do**
  $\overline{C}(x_i) \leftarrow 1 - NN(P, x_i)$
  if $\overline{C}(x_i) \neq C(x_i)$ **then**
    $P = P \cup x_i$
    $T = T \setminus x_i$
  end if
end for
Sortie P

— *IB3 :* se base sur l'algorithme IB2 avec en supplément une politique lui permettant de supprimer de S des instances considéré comme bruitées.

Instance Rank based on Borders (IRB) Cet algorithme proposé par Hernandez-Leal en 2013 [54] effectue un ranking en classant les instances en fonction de leur proximité d'instances de classe différente (les plus proche des frontières ayant le plus haut ranking) puis sélectionne un pourcentage d'instances de haut, moyen et faible ranking.

Dans cette catégorie, nous trouvons aussi : Polyline functions PF [55], Estimation of distribution algorithm EDA [56], Prototype selection using relative certainty gain PSRCG [57], CPruner [58], Backward sequential edition BSE [59], Noise removing based on minimal consistent set NRMCS [60], Hit miss network C HMNC [61], Hit miss network edition HMNE [61] ou Hit miss network edition iterative HMNEI [61].

4.4 Algorithme méta-heuristique

Algorithme évolutionnaire et coévolutionnaire Les algorithmes évolutionnaires ont été largement utilisées pour la sélection d'instances. L'idée de base est de créer une population de chromosomes représentant des instances puis de la faire évoluer afin d'améliorer une fonction de fitness basée sur le taux de classification et de réduction. Les algorithmes coévolutionnaires font évoluer plusieurs populations en collaboration pour résoudre le problème. Ces algorithmes donnent des résultats très prometteur comme nous pouvons voir dans les articles suivant [62], [6], [63], [4], [64], [65].

Dans l'article de Cano et al. [4], les auteurs ont comparé les performances de sélection d'algorithmes classique et quatre algorithmes évolutionnaires : Population-based incremental learning (PBIL), CHC genetic algorithm, genetic algorithms (GA) and steady state genetic algorithm (SSGA). Les résultats démontrent que

les algorithmes évolutionnaires surpassent les approches de sélection classique, notamment l'algorithme CHC.

Support vector machine (SVM) Il existe quelques travaux de sélection d'instances basés sur l'algorithme SVM tels que Support Vector Based Prototype Selection (SVBPS) proposé par [66]. Cet algorithme appliqué DROP2 sur le vecteur obtenu par SVM sur l'ensemble T. Dans l'article [67], Scholkopf et al. effectuent une sélection en gardant que les instances formant partie du vecteur obtenu par l'algorithme SVM. Dans leur article [68] les auteurs utilisent les Forêt Aléatoire pour sélectionner le sous-ensemble optimal pour la construction d'un SVM. L'algorithme la marge de chaque instance de la base et sélectionne les M instances de plus faible marge. L'inconvénient de cet algorithme est la perte d'information occasionné par le principe de sélection.

Nous pouvons aussi citer les travaux de [69, 70] sur l'utilisation des algorithmes mémétique pour la sélection d'instances. Ainsi que les articles [71, 72], utilisant l'algorithme d'optimisation par essaim de particule.

4.5 Hybridation avec les méthodes ensembliste

Dans leurs articles [73, 74], les auteurs ont effectué une hybridation entre les méthodes ensembliste et de sélection d'instance afin d'optimiser les performances de ces derniers.

Dans le premier travail, les auteurs utilisent un bagging d'algorithmes de sélection. Chaque algorithme de l'ensemble fournit un poids binaire pour voter sur la sélection ou l'élimination de chaque instance de l'ensemble d'apprentissage. Finalement, la moyenne des totaux des votes est calculée et utilisée avec un seuil d'acceptation afin de déterminer quelles instances conserver dans le sous-ensemble S.

Dans le second article, la méthode du boosting fût adaptée à l'approche de sélection d'instance. Plusieurs algorithmes de boosting tels que l'AdaBoost, FloatBoost, MadaBoost, MultiBoost et ReweightBoost ont été combiné avec des algorithmes de sélection d'instance. A l'instar d'un boosting de classifieur, à chaque itération l'algorithme de sélection se focalise sur les instances les plus difficiles à traiter par l'algorithme précédent puis la décision finale est prise par vote majoritaire.

4.6 Synthèse et Analyse

Nous remarquons que la plupart des méthodes existantes sont généralement basées sur le principe du K-plus proches voisins. L'inconvénient majeur des premières méthodes basées sur la mauvaise classification d'une instance par ses voisins (tels que CNN, GCNN, IB, ENN ou All-Knn) est la rétention d'instances bruitées puisqu'elles sont toujours mal classées par leurs voisins. Ce qui cause que la proportion de bruit dans S soit supérieure à celle de A. D'un autre côté,

ces méthodes sont sensibles à l'ordre de présentation, différentes permutations de l'ensemble d'apprentissage aboutissent a des sous-ensembles différents. Ce problème est notamment retrouvé au sein des algorithmes de condensation. Tandis que les algorithmes d'édition ont un taux de réduction assez faible.

Bien que les algorithmes d'édition soient considérés comme étant des filtres de bruit, leurs performances diminuent si le taux de bruit augmente. Si le nombre d'instance bruité est élevé, elles ne seront plus considérées comme des exceptions et seront correctement classées par d'autres instances bruitées [3].

Plusieurs travaux ont démontré que les approches hybrides sont plus performantes que celles appartenant aux deux groupes précédent [3, 7]. Mais l'inconvénient majeur lié à ces méthodes est la baisse des performances obtenues par l'ensemble réduit S s'il est appliqué avec un autre classifieur que K-NN [7].

Les algorithmes évolutionnaires peuvent être utilisés avec n'importe quel algorithme de classification pour le processus de sélection ce qui permet d'effectuer la sélection avec l'algorithme qui sera utilisé pour la classification et ainsi évitée une perte de performance. Dans l'article de Cano et al. [4]les auteurs ont mené une étude sur les algorithmes Generational Genetic Algorithm (GGA), CHC Adaptive Search Algorithm, Steady-State Genetic Algorithm (SGA) et Population-Based Incremental Learning (PBIL) pour la sélection d'instances. Selon les résultats obtenus, l'algorithme CHC présente les meilleures performances réduction/ classification. Par contre les algorithmes évolutionnaires souffrent d'un problème de scalabilité, en effet, le temps d'exécution augmente avec la taille des données, ce qui rend impossible leur application à des bases de données de grande taille.

Nous pouvons en conclure que la plupart des algorithmes de sélection d'instance souffre du même problème de scalabilité que les algorithmes d'apprentissage et ne peuvent être appliqué à des bases de grande taille où ils sont le plus necessaire.

5 Problème de scalabilité

Etant donné que le principal inconvénient des algorithmes de sélection d'instances est le problème de scalabilité, plusieurs travaux ont été proposé pour palier à ce problème. DeHaro-Garcia et al. [75] ont proposé une approche récursive basée sur le principe *"diviser pour régner"*. L'ensemble d'apprentissage T est divisé en sous-ensemble de même taille et un algorithme de sélection est appliqué sur chaque sous-ensemble, les instances sélectionnées sont regroupées en un nouvel ensemble puis le processus est répété jusqu'à ce qu'un critère d'arrêt soit atteint.

Dans leur article [76], les auteurs présentent une variation de leur précédente approche nommée Democratic instance selection. La base est divisée en plusieurs sous-ensembles disjoints puis un algorithme SI est appliqué sur chaque sous-ensemble, les instances sélectionnées reçoivent un vote, le processus est répété un certain nombre de fois et les instances qui reçoivent le plus de vote sont gardées.

DeHaro-Garcia et al. proposent une approche basée sur une implémentation en parallèle dans leur article [77].

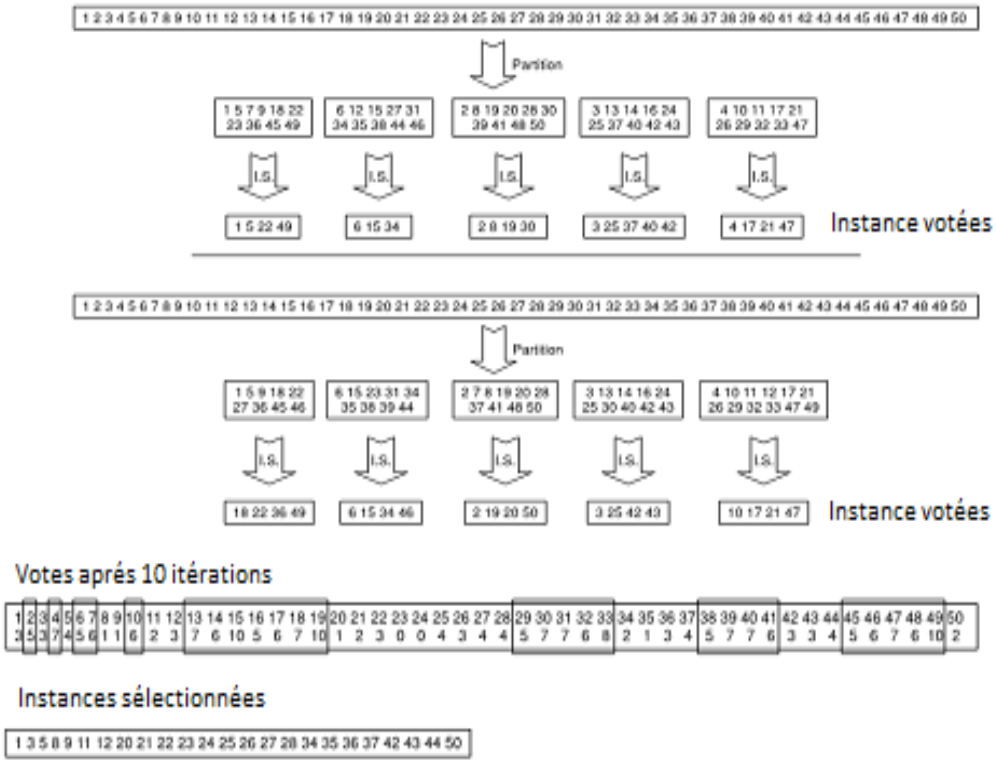


FIGURE II.2 – Democratic instance selection

En [78], une approche est proposée basé sur la stratification consistant à diviser l'ensemble en r sous-ensemble disjoints (strata) en conservant la distribution des classes et d'y appliquer l'algorithme de sélection puis les instances sélectionnées sont réunies en un seul ensemble. Les stratégies mentionnées peuvent être appliquées à n'importe quel algorithme de sélection d'instances même si elles ne constituent pas des algorithmes de sélection en soit.

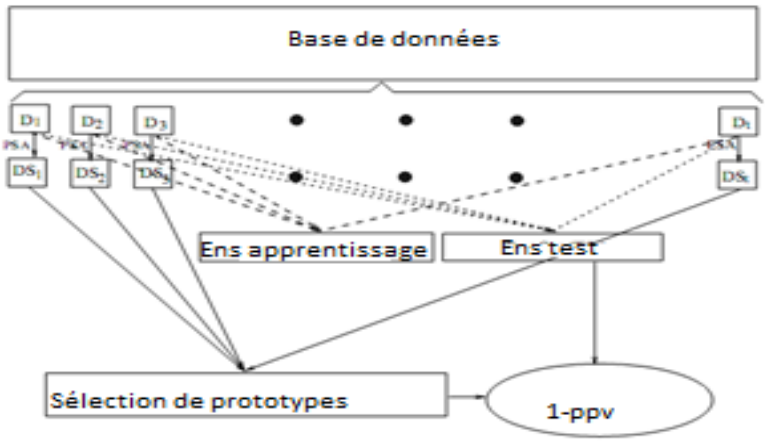


FIGURE II.3 – Approche de stratification

6 Conclusion

Dans ce chapitre, nous avons introduit les concepts de base des approches de sélection d'instance, leur taxonomie ainsi que quelque uns des algorithmes les plus populaires de la littérature. Nous nous sommes aussi penché sur une analyse de leurs avantages et limitations, notamment face aux bases de données de grande dimension où leurs utilisation est la plus nécessaire.

Nous avons aussi présenté, quelques approches qui ont été proposé pour faire face au problème de scalabilité des méthodes de sélection d'instance.

Dans le chapitre suivant, nous présentons les méthodes ensemblistes comme un puissant outil permettant l'optimisation d'algorithmes de classification.

Chapitre III

Les méthodes ensemblistes

Introduction

Bien que d'innombrables algorithmes de classification existent dans la littérature pour prédire la classe d'un prototype à partir d'un ensemble d'apprentissage, leurs performances varient d'un problème à un autre.

Les méthodes ensemblistes se basent sur le principe « l'union fait la force » en combinant plusieurs modèles de classification afin de générer un modèle plus performant que le meilleur d'entre eux [79]. L'heuristique de ces méthodes est qu'en générant beaucoup d'hypothèses, on explore plus largement l'espace des solutions, et l'agrégation des prédictions conduit à un prédicteur final qui représente la décision de toute cette exploration. Ce procédé permet d'utiliser des algorithmes simple et facile à implémenter tout en obtenant des performances de qualité. Les méthodes ensemblistes sont intrinsèquement parallélisable, ce qui les rend intéressant pour réduire le temps d'exécution si nous avons à notre disposition de multiples processeurs.

Par exemple dans le cas de classification binaire. Pour qu'une instance x soit mal classée, il faut qu'au moins la moitié des prédicteurs se trompent sur l'instance x. Ce qui est un cas très rare car même si les classifieurs individuels commettent des erreurs, il est peu probable, qu'ils se trompent tous sur la même instance. D'où la nécessité que les prédicteurs individuels soient relativement bon et différents les uns des autres. Ce qui est naturel car agréger de mauvais classifieurs ne construit pas un bon classifieur. Tandis que l'agrégation de prédicteurs semblables augmente la complexité sans améliorer les performances.

1 Construction de l'algorithme

Les algorithmes de classification composant l'ensemble peuvent être différents formant un ensemble hétérogène ou les mêmes formant un ensemble homogène. La construction d'un algorithme ensembliste s'effectue en deux étapes :

1.1 La construction des modèles

Pour obtenir un modèle ensembliste performant, il faut que les hypothèses de base soient différentes pour plus de diversité. En effet, la construction de modèles similaires augmente le coût des calculs sans apporter aucune amélioration. Il existe deux types d'ensembles :

Ensemble homogène : la diversification s'effectue sur les bases d'apprentissage par l'échantillonnage de la base d'origine en utilisant des techniques telles que le boostrapping pour le bagging et les forêts aléatoires ou la méthode random subspaces proposée par [80] où pour chaque hypothèse un sous-ensemble de variables est sélectionné pour l'apprentissage.

Ensemble hétérogène : Pour les modèles appartenant à cette catégorie, la diversification est au niveau des algorithmes d'apprentissage. Il est possible de procéder de deux manières distinctes, soit en utilisant le même algorithme mais en variant les paramètres, soit en effectuant un apprentissage avec différents algorithmes. L'avantage d'un ensemble hétérogène est qu'il y a très peu de probabilité que toutes les hypothèses fassent la même erreur.

1.2 Stratégie de diversification

Plusieurs méthodes ont été proposées afin de diversifier l'apprentissage des ensembles homogènes ce qui permettra aux hypothèses de commettre des erreurs différentes. Ces stratégies peuvent être divisées en quatre catégories [81] :

Manipulation de l'ensemble d'apprentissage Cette stratégie consiste à perturber l'ensemble d'apprentissage en ajoutant, supprimant ou modifiant les poids des instances. Ainsi, chaque classifieur de base est entrainé sur une différente version de l'ensemble d'origine. Afin que cette approche soit effective, il faut que l'apprenant de base soit instable c'est à dire que le plus faible changement dans l'ensemble d'apprentissage conduit à un classifieur différent. Les algorithmes les plus performants pour cette stratégie sont les arbres de décision ou les réseaux de neurone [82]. A la différence des algorithmes plus stable tels que Support Vector Machin (SVM) et le K-plus proche voisin (K-ppv ou K-nn) pour lesquels la méthode n'est effective que si les changements entre les bases sont importants.

L'une des méthodes pour appliquer cette stratégie est le boostrapping [83] qui se base sur un principe de tirage aléatoire avec remise pour générer les sous-ensembles d'apprentissage. Parmi les algorithmes ensemblistes utilisant les boostraps, nous pouvons citer le bagging ou les forêts aléatoires. Une autre méthode consiste à pondérer les instances et les modifier afin de contrôler l'impact d'une instance à l'étape d'apprentissage. Cette approche est utilisée pour l'algorithme boosting [84].

Le subagging et bragging [85], le wagging [86], le multiboosting [87] et le bagboosting [88] sont aussi des algorithmes ensemblistes se basant sur la stratégie de manipulation de la base d'apprentissage.

Manipulation de l'espace des variables Ces méthodes suppriment une partie des variables de l'espace d'apprentissage. Les variables à éliminer doivent être choisies soigneusement afin d'éviter une perte de l'information. L'algorithme random subspace [80] utilise cette stratégie d'ensemble en construisant chaque prédicateur sur un sous-ensemble d'attributs. Tumer et al. [89] proposent l'algorithme input decimation qui sélectionne un sous-ensemble de variables en fonction de leurs corrélation avec le label.

Une autre possibilité est de générer de nouvelles variables à partir des originales. Skurichina et al. [90] ainsi que Rodriguez et al. [91] utilisent la méthode d'analyse en composantes principales (principal component analysis (PCA)).

Manipulation des labels Les algorithmes dans cette catégorie perturbent les sorties des instances de l'ensemble d'apprentissage pour chaque apprenant de l'ensemble. L'algorithme ensembliste Error-correcting output coding (ECOC) [92] utilise cette technique afin de traiter des problèmes multi-classe avec des classifieurs ne traitant que des problèmes binaires. Breiman [93] propose d'injecter du bruit à la variable de sortie en respectant la proportion des classes au lieu de relabéliser les instances de l'ensemble d'apprentissage avec l'approche output randomization. L'algorithme class-switching [94] se base aussi sur l'injection de bruit au label mais sans conserver la distribution des classes ce qui la rend intéressante pour les problèmes de données déséquilibrées.

Manipulation de l'algorithme d'apprentissage Afin d'injecter de l'aléa aux algorithmes de base, il est possible d'en modifier les paramètres. Parmi ces méthodes, nous pouvons citer l'algorithme Forêts aléatoires à variables d'entrée aléatoires [95] basé sur la randomisation de la coupure des nœuds des arbres CART ainsi que l'approche basée sur l'algorithme C4.5 proposé par Dietterich [96].

1.3 La combinaison des estimations.

Une fois l'apprentissage terminé, nous obtenons un ensemble d'hypothèses différentes dont il faut combiner les prédictions. Il existe dans la littérature plusieurs méthodes à suivre :

Combinaison parallèle La méthode la plus utilisée est le vote majoritaire (bagging, subbagin et forêt aléatoire), chaque classifieur attribue un label à l'instance à classer et la décision finale est la classe qui a obtenu le plus de vote. En cas de problème de régression, la décision finale est la moyenne des décisions individuelles. Les algorithmes tels que le boosting ou adaboost [97] utilise un vote pondéré, en effet, les différents algorithmes contribues à la décision avec des poids différents. Il existe d'autres méthodes telle que le staking [98], la combinaison linéaire „ Naive Bayes Combination ou Wernecke .

Combinaison en cascade A la différence, des approches de combinaison parallèle où chaque algorithme de base est indépendant des autres, les méthodes basées sur la généralisation en cascade [99] utilisent les résultats du précédent apprenant pour l'étape d'apprentissage. La décision finale est prise par le dernier apprenant de la série. Cette méthode est utilisée avec le boosting avec de bons résultats.

Intégration dynamique des classifieurs Cette méthode présentée par Puuronen et al. [100]se base sur le principe que chaque classifieur obtient des performances différentes en fonction de l'espace de recherche. Pour chaque instance de la base de test, un classifieur est dynamiquement sélectionné en fonction des attributs de l'instance.

Tsymbal et Puuronen [101] appliquent le principe d'intégration dynamique au boosting et bagging avec de meilleurs résultats que le vote majoritaire. En effet, pour chaque nouvelle instance, les classifieurs ayant obtenu les meilleures performances sur la classification des plus proches voisins sont sélectionnés pour la prise de la décision finale.

2 Avantages

La construction d'un modèle ensembliste est plus coûteux que la construction d'un modèle unique mais les avantages surpassent les inconvénients [9] :

Statistique : même si le taux de classification obtenu par un algorithme sur la base d'apprentissage est bon, il n'est pas garanti que le résultat de classification sur une base de test soit bon et ainsi obtenir un modele peu performant. En combinant les résultats de plusieurs modèles, nous minimisons le risque de travailler avec un mauvais classifieur.

Base de données de grande taille : dans certains domaines les bases sont tellement volumineuses qu'il est impossible de construire un modèle. Les méthodes ensemblistes permettent de diviser l'ensemble d'apprentissage en sous-ensembles plus petits et de combiner leurs résultats.

Base de données de petite taille : si l'ensemble d'apprentissage est trop petite le modèle obtenu peut être instable. Une solution possible est de générer plusieurs sous-ensembles se chevauchant et de combiner les résultats des modèles obtenus.

Données hétérogène : pour certains problèmes, les données proviennent de sources différentes et sont représentées différemment. Par exemple, les patients atteints de cancer subissent plusieurs tests tels que es : analyses sanguin, génétiques, scanner, etc., dont chaque résultat est exprimé par différents attributs. Il est plus aisé de construire un modèle pour chaque teste plutôt qu'un seul qui traiterait les attributs de tous les tests.

Dans son article, Dietterich [96] identifie trois raisons expliquant la performance des méthodes ensemblistes [102] :

1. **Représentation :** considérons un algorithme d'apprentissage cherchant une solution linéaire dans un espace où la solution optimale est quadratique. Cette procédure a un problème de biais car elle cherche une solution optimale dans un espace de fonction qui n'en contient pas. Par contre, si un ensemble d'apprenants cherche chacun une solution linéaire, elles peuvent être combiné pour générer la solution optimale. Ceci conduit à une réduction du biais.

2. **Statistique :** considérons maintenant le cas inverse où nous avons un problème de variance. L'espace de recherche est riche en bonnes solutions. Sous ces conditions, il est difficile de choisir entre toutes les solutions, par contre, il serait intéressant d'en sélectionner plusieurs et de les combiner.

3. **Computationnelle :** les algorithmes d'apprentissage rencontrent souvent des problèmes d'optima locaux. En utilisant un ensemble d'hypothèses, nous évitons ce problème en explorant un espace plus large de solutions.

3 Algorithmes ensembliste

Dans cette section, nous allons citer quelque uns des algorithmes ensembliste les plus populaires.

3.1 Échantillonnage

Comme cité auparavant, les ensembles homogènes sont composés de modèles ayant effectué un apprentissage sur des bases différentes obtenues par un boostrapping, un échantillonnage aléatoire avec remise. Ceci produit des ensembles nommés boostrap de la même taille où les instances de la base originale peuvent soit être absent soit présent une ou plusieurs fois. Si le boostrap généré est de la même taille que l'ensemble original 36,8% seront des copies des 63.2% instances restantes [9].

Algorithm 8 Procedure du boostrapping

Entré : $A = (I_1, I_2, \ldots, I_m)$, T : Nombre de classifieurs.
for i :=1 **to** | T | **do**
  Sélectionner i
end for
return

3.2 Bootstrap Aggregation (Bagging)

Le bagging abréviation de Bootstrap Aggregation est un algorithme proposé par Breiman en 1996 [103] basé sur les concepts de boostrapping et l'agrégation. C'est l'un des premiers algorithmes ensembliste et l'un des plus simples à implémenter. Il appartient à la catégorie des algorithmes ensembliste homogène, les hypothèses effectuent leur apprentissage sur différentes variations de l'ensemble original $X = (X_1, X_2, ..., X_n)$ nommé boostrap ou bag $X_b = (X_b1, X_b2, ..., X_bn)$ généré par un échantillonnage avec remise (boostrapping). Au final, les prédictions individuelles sont regroupées par un vote majoritaire pour les problèmes de classification et par le calcul de la moyenne pour les problèmes de régression.

Le bagging trouve tout son intérêt utilisé avec des classifieurs instable, aussi connus comme de haute variance tels que les arbres de décision ou les réseaux

de neurone, où le plus petit changement dans l'ensemble d'apprentissage peut induire à une hypothèse totalement différente.

Comme précisé auparavant, 36,8% des instances de l'ensemble original ne sont pas présent dans un boostrap, ce que signifie que quelques bags ne contiennent pas d'instances bruitées ce qui conduit à la construction d'hypothèses plus performantes que celles obtenues avec l'ensemble A [104].

Algorithm 9 Algorithme Bagging

Entrées : $A = ((x_1, y_1), ..., (x_m, y_m)))$, T : Nombre de classifieurs.
for i :=1 **to** | T | **do**
  $S_t =$ Bootstrap(A), i.i.d. tirage avec remise de A.
  $h_t =$ Entrainer un classifieur(S_t).
  Ajouter h_t à l'ensemble.
end for
Sortie $h_S(x)$ =Vote Majoritaire $((h_1(x), ..., h_T(x)))$

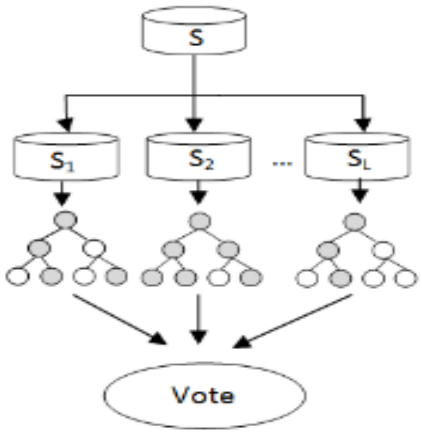


FIGURE III.1 – Illustration of the Bootstrap Aggregation method

3.3 Boosting

Le boosting proposé par Freund and Schapire combine des hypothèses faibles (i.e. juste meilleures qu'un tirage aléatoire) en une hypothèse forte. Cette méthode se base sur la théorie de l'apprentissage PAC (Probably Approximately Correct).

Au début des années 1990 Schapire [84] a introduit la première version de boosting. Cette version a été améliorée ensuite par Freund un an plus tard en se basant sur le paradigme suivant :

« tout algorithme capable, pour toute distribution, d'apprendre avec une confiance faible et une précision inférieur à ½ peut être transformé en un algorithme d'apprentissage avec une confiance aussi grande et une précision aussi bonne que désirée. »

L'idée principale sur laquelle se base le boosting est la suivante : Intuitivement, il est logique de penser que pour une décision nécessitant les connaissances d'un expert, plusieurs experts travaillant ensemble et combinant adéquatement leurs jugements pourraient être plusperformants que le jugement d'un seul expert pris séparément. Une méthode envisagée pour labéliser une instance automatiquement est d'interroger différents classifieurs et de combiner leurs décisions de classification en pondérant chaque classifieur selon sa performance testée sur des exemples de validation.
Théoriquement, plus les classifieurs sont indépendants par leur conception, leur entraînement et leur façon de labéliser, plus la performance du comité devrait être bonne.

L'algorithme boosting utilise un comité de classifieurs $H = h_1, \ldots, h_k$ formé à partir de l'apprentissage d'un ensemble d'algorithmes faibles en passant par plusieurs itérations. A chaque itération le poids des exemples incorrectement classifiés sont augmentés et seuls les classifieurs les plus performants du comité sont retenus pour les étapes suivantes.

A la différence de l'algorithme bagging, les hypothèses de l'ensemble d'un algorithme boosting sont entraînées séquentiellement, et non pas parallèlement, et ainsi un classifieur h_t prend en compte la performance des classifieurs précédent $h_1, \ldots, h_t - 1$ et essaie de classifier correctement les exemples mal-classifiés par eux.

Le poids de chaque instance représente le degré de difficulté rencontré par l'ensemble de classifieur $h_1, \ldots, h_t - 1$ pour une classification correcte. Ces poids seront exploités durant l'apprentissage de h_t et seront ajustés, respectivement, pour résoudre les cas ayant un poids élevé (les plus difficiles à classé), puis on diminue les poids des instances classées correctement tandis que les poids des instances mal-classées sont augmentés.

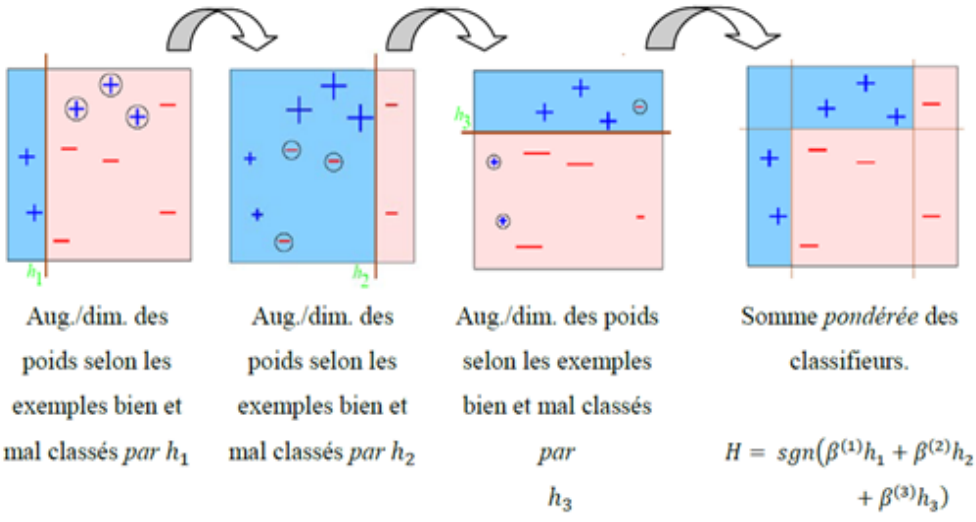


FIGURE III.2 – Classifieur H final

La figure III.1 présente un exemple de classification par boosting d'arbre de décision pour un problème binaire [105]. Dans cette figure, on dispose d'un ensemble de 10 exemples i.e. $N = 10$ dont 5 sont positifs et 5 sont négatifs, repartis aléatoirement dans un espace donné. Le but est de construire en 3 itérations, i.e. $T = 3$, un apprenti linéaire capable de séparer les exemples positifs des exemples négatifs pour pouvoir classer correctement un nouvel exemple i.e. il va falloir estimer s'il est positif ou négatif. Le poids de toutes les instances est initialisé à $1/N = 0.1$.

Dans la première itération, le premier arbre h_1 estime que le séparateur existe à gauche de l'espace (comme l'indique la première figure à gauche). Selon ce séparateur tous les exemples à gauche de la ligne sont considérés positifs et ceux qui sont à droite sont négatifs. Ainsi, h_1 a correctement classé deux exemples positifs mais a mal-classé trois autres et donc d'après l'algorithme on obtient :

$$\varepsilon 1 = 0.3$$
$$\beta(1) = 0.42$$

Dans ce cas là, le Boosting diminue le poids de tous les exemples classés correctement par une valeur $\cong 0.07$ selon la formule de l'algorithme et augmente les poids des trois autres d'une valeur $\cong 0.17$.

Le fait d'augmenter le poids d'un exemple incite le prochain arbre de décision à se concentrer plus sur sa classification. Ainsi h_2 estime que le séparateur existe où l'indique la figure au milieu. Ce séparateur a correctement classé tous les exemples positifs ainsi que les deux exemples négatifs à droite mais a raté les trois autres et donc :

$$\varepsilon 2 = 0.21$$
$$\beta(2) = 0.65$$

L'hypothèse h_3 estime que cette fois-ci le séparateur est horizontal et donc a raté 3 exemples (1 négatif et 2 positifs) et ainsi :

$$\varepsilon 3 = 0.14$$
$$\beta(3) = 0.92.$$

Et par conséquent le classifieur H final est :

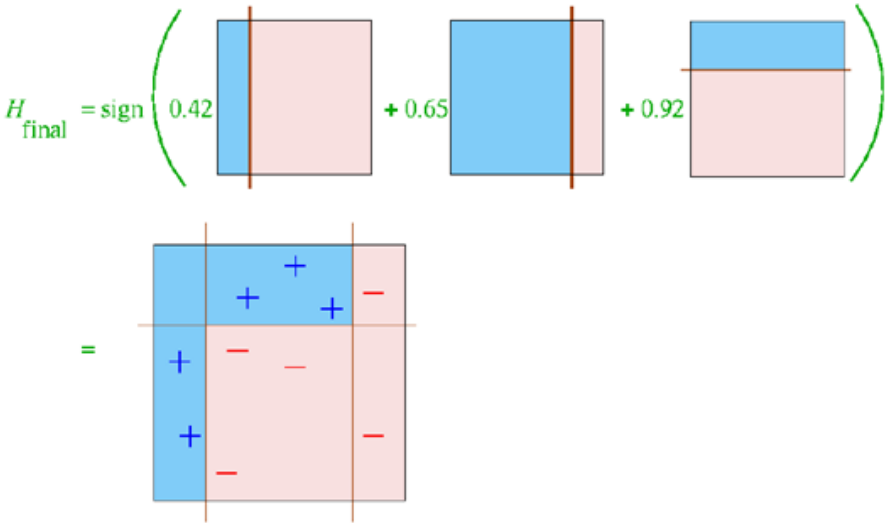


FIGURE III.3 – Classifieur H final

Pour classer un nouvel exemple, le classifieur final H interroge chacun des trois classifieurs en le pondérant avec le poids qui lui est associé et additionne les trois valeurs obtenus. Si le résultat de cette somme est une valeur positive c'est que le nouvel exemple est positif et si cette valeur est négative c'est qu'il est négatif.

3.4 Random Subspace Method (RMS)

Proposé par Ho [80] cette méthode ensembliste construit le classifieur de base non pas sur l'ensemble des variables mais sur un sous-ensemble de variable. En effet, à l'inverse du bagging, la modification des données d'entrée se fait sur l'espace des variables.

Le RMS suit la procédure suivante [106] : chaque instance X de l'ensemble d'apprentissage $L = X_1, X_2, \ldots, X_m$ est un vecteur décrit par p variables $X_i = x_i1, x_i2, \ldots, x_ip$. Au cour de la construction de l'ensemble RSM, un sous-ensemble de $r < p$ variables est aléatoirement sélectionné. Ainsi chaque instance de l'échantillon modifié d'apprentissage est formé de r attributs aléatoirement choisi parmi les p existantes $X_i = x_i1, x_i2, \ldots, x_ir$ et la décision final est la combinaison des différentes décisions individuelles.

La méthode RMS trouve tout son intérêt pour les problèmes où l'espace de variables est très supérieur au nombre d'instances de l'ensemble d'apprentissage ou quand il y'a des variables redondante. Dans ces cas, l'utilisation de classifieurs construit sur des sous-espaces aléatoires de variables produit de meilleurs résultats qu'un seul classifieur construit sur l'ensemble des variables.

En général, RMS utilise les arbres de décision comme classifieur individuel même s'il obtient aussi de bons résultats avec d'autre type d'algorithme.

3.5 Randomizing output

Cette méthode ensembliste fut proposée par Breiman en 2000 [93]. Les méthodes précédemment citées, baggin, boosting et RMS, construisent leur prédicteurs en ajoutant l'aléa à l'échantillon d'apprentissage ou aux variables. Par contre, Randomizing output se base sur un principe complètement diffèrent. En effet, pour cette méthode les labels des échantillons d'apprentissage de chaque classifieur individuel sont indépendamment modifiés en ajoutant une variable de bruit.

Dans son article, Breiman démontre qu'en effectuant l'étape d'apprentissage des hypothèses de base sur des échantillons à sortie randomisé, on obtient des prédicteurs différents dont l'agrégation donne de meilleurs résultats qu'un bagging.

3.6 Forêt aléatoire

Introduites par Breiman en 2001 [95], les forêts aléatoires sont une méthode ensembliste très performante dans de nombreux domaines d'étude. Depuis leur présentation, elles sont de plus en plus utilisées pour traiter de nombreux et divers problèmes réels. Voici la définition générale donnée par Breiman dans cet article :

Définition : Une forêt aléatoire est un classifieur formé par une collection d'arbres $h(x, theta_k), k = 1, \ldots$ *où* θ_k *sont un ensemble de vecteurs aléatoires identiquement distribués. Chaque arbre contribue à la décision finale faite par vote majoritaire.*

Cette méthode d'ensemble est une méthode où le classifieur de base est un arbre de décision d'où le nom de « forêt ».Chaque arbre est entrainé sur l'ensemble d'apprentissage et un vecteur θ_k généré aléatoirement et indépendamment des $k-1$ vecteurs précédemment générés mais avec la même distribution.

Si nous considérons les méthodes précédemment cité, seule le boosting ne peut être utilisé pour construire une forêt aléatoire. Ceci est dû au fait que les vecteurs du boosting ne sont pas indépendant les uns des autres. Par contre, les trois autres méthodes peuvent être considérées comme étant des cas particuliers.

Par exemple, pour le bagging l'aléa est au niveau de l'échantillon boostrap. Pour les méthodes RSM et Randomizing output, c'est la sélection aléatoire du sous-ensemble de variables et la modification des sorties qui représentent l'aléa supplémentaire.

Force et corrélation Breiman a définit deux propriétés principales qui sont la force permettant de mesurer la fiabilité de la forêt et la corrélation qui représente le degré de dépendance des arbres de la forêt.

Ces deux propriétés sont définies sur la fonction de marge de la forêt aléatoire. La marge ensembliste représente la différence entre la moyenne des votes pour

le label correct et celle pour le label erroné le plus voté. Cette marge peut être négative ou positive. Si elle est négative, alors la classe attribuée à l'instance n'est pas correcte. Dans le cas contraire, la prédiction est correcte et plus la valeur est élevée, plus de classifieurs classent correctement l'instance. La fonction marge est :

$$mg(X,Y) = av_k I(h_k(X) = Y) - \max_{j \neq Y} av_k I(h_k(X) = j)$$

Avec h_k est le k^eme classifieur de la forêt et $I()$ est une fonction indicatrice.

A partir de la fonction de marge, l'erreur de généralisation est définie comme suit :

$$PE* = P_{X,Y}(mg(X,Y) < 0) \tag{III.1}$$

Dans son article [95], Breiman montre que la fonction de marge peut s'écrire :

$$mr(X,Y) = P_\theta(h(X,\theta) = Y) - \max_{j \neq Y} P_\theta(h(X,\theta) = j) \tag{III.2}$$

Cette fonction mesure la différence entre la probabilité de la forêt de prédire la classe correcte et la probabilité de prédire la classe erronée la plus populaire. Et la force d'une forêt est mesurée par l'espérance de cette marge.

$$s = E_{(X,Y)} mr(X,Y)$$

Finalement, Breiman a montré que l'erreur de généralisation est un consensus entre la force des arbres et leur corrélation. D'après cette fonction, l'erreur est minimale si la force est élevée et la corrélation faible.

$PE* \leq \frac{\overline{\rho}(1-s^2)}{s^2}$

3.7 Arbre de décision : CART

Les arbres de décision se basent sur la stratégie de "diviser pour régner" qui consiste à subdiviser l'ensemble d'exemples en sous-groupes aussi pur que possible. Proposé par Breiman et al. en 1984 [19]. CART est une méthode d'apprentissage supervisée qui utilise les données labellisés de la base d'apprentissage afin de construire un arbre de décision capable de classer de nouvelles données. L'algorithme CART cherche parmi toutes les variables et valeurs possible afin de déterminer la meilleure division possible, celle qui divise les données en deux parties le plus homogènes possibles.

Le critère de division repose sur la définition d'une fonction d'hétérogénéité qui est nulle si tous les individus du nœud qui en découle sont de la même classe et maximale lorsque les valeurs de Y sont équiprobables. La division retenue sera celle qui minimise la somme des désordres des deux fils obtenus.

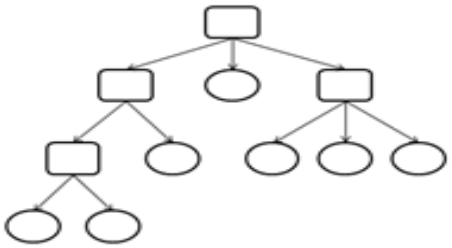


FIGURE III.4 – Arbre de décision

Principe L'arbre se construit en partitionnant récursivement l'ensemble d'apprentissage. En partant de la racine qui représente l'ensemble des données, une variable est sélectionnée à chaque étape pour découper l'espace en deux sous-ensembles représentés par deux nœuds fils.

Découpage Dans notre travail nous nous restreignons à des variables continues où l'espace d'entrée est R^p avec p est le nombre de variables. L'objectif de CART est de déterminer la meilleure découpe pour chaque nœud de l'arbre en suivant la règle suivante [8] :

$$\{Xj < d\} \cup \{Xj > d\}$$

où $j \in 1; \ldots; p$ et $d \in R$

Si la j-ième variable d'une instance est inferieure à d alors elle appartient au nœud fils gauche sinon elle appartient au nœud fils droit. La variable et la valeur de découpe sont automatiquement sélectionnées par l'algorithme en vu d'optimiser un critère en concret.

Pour les arbres CART la règle de découpe vise à minimiser la variance des nœuds fils en regression et l'indice de Gini en classification.

L'indice de Gini développé par le statisticien italien Corrado Gini est une mesure statistique de la dispersion d'une distribution dans une population donnée. La valeur de l'indice est un nombre variant de 0 à 1, où 0 signifie l'égalité parfaite et 1 signifie une inégalité parfaite.
$I = 1 - \sum_i^N f_i^2$.

Avec N le nombre de classe à prédire et f_i la fréquence de la classe i dans le nœud.

Critère d'arrêt et d'élagage La croissance d'un arbre CART ne respecte pas un critère d'arrêt. L'arbre est donc construit jusqu'à l'obtention d'un nœud où il n'existe plus de partition admissible ou, pour éviter un découpage inutilement fin, si le nombre d'observations qu'il contient est inférieur à une valeur seuil, ce nœud est nommé feuille ou nœud terminal.

Ce processus de construction conduit à des arbres instables fortement dépendant de l'ensemble d'apprentissage (sur-apprentissage). Le processus d'élagage de l'arbre proposé par Breiman permet l'obtention d'un sous-arbre non nécessai-

rement optimal mais indéniablement plus fiable.

Une fois la construction de l'arbre maximal terminé, les branches peu représentatives sont supprimées en fonction d'un critère d'élagage. Le processus s'effectue des extrémités vers la racine, en se basant sur une estimation du taux de classification : un arbre est élagué à un certain nœud si le taux d'erreur estimé à ce nœud (en y allouant la classe majoritaire) est inférieur au taux d'erreur obtenu en considérant les sous-arbres terminaux. Après élagage, les nouvelles feuilles sont labellisées sur base de la distribution des exemples d'apprentissage (classe majoritaire).

Il faut préciser que les arbres CART utilisés pour la construction d'une forêt aléatoire sont des arbres maximum. Un arbre maximal a une grande variance et un biais faible, cet inconvénient est résolu grâce à l'étape d'agrégation.

3.8 Random feature selection

Initialement proposé par Amit et Geman [107] pour la construction d'un ensemble d'arbres aléatoires, le principe fût repris par Breiman [95] pour ajouter un deuxième niveau d'aléa à la construction des forêts aléatoire. L'idée consiste à choisir une règle de partitionnement à partir d'un sous-ensemble de caractéristiques. Plus précisément, un nombre $K < M$ de variables est sélectionné par tirage aléatoire sans remise puis la meilleure des règles possible est choisie en utilisant ces K variables uniquement.

Des études ont été mener sur le choix du paramètre K [95]. Les expérimentations ont testé les valeurs de $K = 1, K = log(M) + 1$ et $K = sqrt(M)$. D'un autre côté, Bernard et al. ont proposé l'algorithme Forest-RK où le paramètre K n'est pas fixé mais choisi aléatoirement pour chaque nœud de la forêt.

3.9 Forêts aléatoires à variables d'entrée aléatoires (Random Forests - RI)

Proposées par Breiman [95], un RF-RI est un type de forêt aléatoire formée d'une variante d'arbre CART avec un aléa supplémentaire. Chaque arbre est construit sur un des échantillons bootstrap $L_1, \ldots, L_n$, suivant un processus différent du CART classique. En effet, au moment de découper un nœud, on ne cherche plus la meilleure coupure sur l'ensemble des variables mais plutôt parmi un sous-espace de variables m aléatoirement sélectionnées. Le tirage des m variables se fait sans remise et de manière uniforme. Chaque variable a une probabilité $1/p$ d'être choisie et le nombre $m <= p$ est identique pour tous les arbres et prédéfini au début de l'opération.

De plus, pour les forêts RF-RI, l'arbre est complètement développé sans étape d'élagage. La décision finale de l'ensemble est prise par vote majoritaire pour la classification et par la moyenne en régression.

Depuis leurs apparitions, plusieurs chercheurs ont proposésdes variations des RF-RI. Les forêt-RC introduite par Breiman [95], se distinguent des RF-RI à l'étape de découpe. Au lieu, d'effectuer une découpe sur une variable, elle est faite sur une combinaison linéaire de variables.

Algorithm 10 Forêts aléatoires à variables d'entrée aléatoires

Input : A ensemble d'apprentissage, L nombre d'arbres

Pour i=1 .. L **faire**
$T_i \leftarrow$ ensemble boostrap, dont les données sont tirées aléatoirement avec remise de A
$C_i \leftarrow$ Construir arbre CART (T_i) où à chaque noeud :
Sélectionner aléatoirement $K = \sqrt{(M)}$ variables parmis M variables.
sélectionner la variable la plus informative avec l'indice de Gini.
Créer un noeud avec cette variable .
$E \leftarrow$E UC_i
Fin Pour

Sortie : E la forêt.

3.10 Types de fôrets RF-RI

Plusieurs variations de l'algorithme de base Forest-RI ont eté proposé. Ces différentes propositions sont en général adapté à un domaine ou à un type de données en particulier. Parmi ces méthodes nous pouvons citer :

Forest-RC (Random Forests - Random Combinations) Dans le cas où le nombre de caractéristique est réduit, le principe de diversité est diminué et les forêts aléatoires traditionnelles ne sont plus aussi efficaces. Pour palier à ce problème, Breiman [95] propose l'algorithme « Forest-RC » qui au lieu de sélectionner la meilleure variable parmi K choisie au hasard, calcule K combinaisons des F caractéristiques et fait la sélection parmi ces nouvelles variables.

Extremely Randomized Trees (Extra-Trees) Geurts et al. propose un nouvel algorithme nommé Ensembles d'Arbres Extrêmement Aléatoires (Extremely Randomized Trees) [108]. L'algorithme Extra-Trees accentue le facteur aléatoire du processus de construction des arbres des RF-RI en sélectionnant le point de coupure de façon totalement aléatoire afin de réduire la variance. D'un autre coté, le biais est réduit en abandonnant le principe de Bagging et en utilisant la base d'apprentissage dans son intégralité.

Balanced Random Forests et Weighted Random Forests Dans leur article [109], Chen et al. présentent deux algorithmes Balanced Random Forests (BRF) et Weighted Random Forests (WRF) basés sur RF-RI, plus adapté au données déséquilibrées. En effet, les forêts aléatoires classiques négligent les classes minoritaires de la base d'apprentissage. Les algorithmes BRF et WRF ont pour objectif de remédier à cette sous-représentation en effectuant respectivement un sur-échantillonnage

et une pondération élevée des classes minoritaires.

L'algorithme BRF utilise des boostraps stratifiés. C'est-à-dire que chaque boostrap est constitué d'échantillons sélectionnés aléatoirement d'instances de chaque classe séparément. Puis les différents échantillons sont regroupés pour former la base d'apprentissage d'un classifieur.

L'algorithme WRF favorise les classes minoritaires en utilisant un schéma de pondération. L'algorithme attribue à chaque classe un poids inversement proportionnel à son taux de représentation dans la base. Ce poids est pris en considération à deux niveaux dans la construction de l'algorithme. Premièrement, pour le calcul du critère de Gini afin d'évaluer le partitionnement. Deuxièmement, par un vote majoritaire pondéré.

Rotation Forests L'algorithme Rotation Forest proposé par Rodriguez et al. [91] divise l'espace des variables en K sous-ensemble puis une analyse en composantes principales est appliquée à chaque sous-ensemble. Ainsi chaque arbre est construit sur les composantes principales obtenues.

4 Conclusion

Dans ce chapitre, nous nous somme intéressé à la famille des algorithmes ensemblistse. Nous avons présenté les différentes méthodes existantes et expliqué leurs principes de fonctionnement. Nous nous somme intéressé tout particulièrement aux forêts aléatoires et aux différents processus d'induction qui ont été proposé.
Ces méthodes sont efficaces et particulièrement compétitives avec les algorithmes de classification les plus performantes. Ils se prêtent particulièrement bien à une implémentation parallèle.
Dans le chapitre suivant, nous présentons notre contribution, une modification de l'algorithme de forêt aléatoire pour effectuer une sélection d'instance.

Chapitre IV

Approche proposée

Introduction

Le principal inconvénient des méthodes de sélection existantes est le coût de calcul élevé. Ceci les rend inapplicable sur les bases de grande taille. Dans ce chapitre, nous introduisons une nouvelle approche pour la sélection d'instances nommée EMIS (Ensemble Margin Instance Selection). Cette approche est basée sur les méthodes ensemblistes et notamment sur le principe de marge des forêts aléatoires. Nous présentons l'algorithme EMIS ainsi que les expérimentations effectuées sur plusieurs bases de données de grande taille.

1 Limitations des méthodes existantes

Dans le chapitre 3, nous avons présenté plusieurs algorithmes de sélection basés sur différents concepts et métriques. Ces algorithmes souffrent de deux inconvénients majeurs : sensibilité au bruit et grande complexité.

La sensibilité au bruit affecte notamment les algorithmes basés sur le principe du plus proche voisin car une instance bruité sera toujours mal classée par ces voisins. D'un autre coté, les besoins de ces algorithmes en temps d'exécutions et en ressources sont élevés à cause de leur grande complexité. Le tableau suivant présente les complexités de quelque uns des algorithmes de sélection.

TABLE IV.1 – Compléxité des algorithmes

Algorithme	**# Complexité**
CNN DROP 1-5 RNN GE	$O(N^3)$
ENN ELH Explore RMHC IRB	$O(N^2)$
IB3	$O(N^2 log2(N))$
All-knn RENN ICF LVQ	$O(iN^2)$

2 Le principe de sélection

Dans notre travail, nous nous sommes intéressé à l'utilisation des méthodes ensembliste pour la sélection d'instances. Il existe plusieurs études sur l'utilisation de ces algorithmes pour la sélection de variables [110] [111] [112], mais très peu sur leur application en sélection d'instances [68].

Nous proposons une approche de sélection d'instance nommée EMIS (Ensemble Margin Instance Selection) basée sur l'algorithme Forêt aléatoire à variables d'entrée aléatoires (RF-RI). Le principe le plus intéressant des algorithmes ensemblistes est la marge ensembliste, une valeur permettant de mesurer le degré de désaccord des arbres de la forêt pour la labellisation d'instances. Notre choix s'est porté sur une marge non-supervisée pour sa plus grande résistance au bruit.

2.1 Marge ensembliste non-supervisée

Présenté par Schapire et al. [113], la marge est un concept fondamental des méthodes ensemblistes, elle présente une bonne estimation des performances de l'ensemble des arbres de la forêt. Dans le chapitre 3, nous avons présenté une marge ensembliste supervisée, en d'autre terme la marge est calculée avec la classe réelle de l'instance. Dans notre algorithme, nous utilisons une marge non-supervisée plus résistante au bruit. La valeur de cette marge appartient à l'intervalle $[0, 1]$, elle est calculée par l'équation suivante :

$$Margin(x) = \frac{n_{c1} - n_{c2}}{T} \tag{IV.1}$$

Where
— n_{c1} représente le nombre de vote pour la classe la plus votée pour l' instance x.
— n_{c2} représente le nombre de vote pour la deuxième classe la plus votée pour l' instance x.
— T représente le nombre de classifieurs dans l'ensemble.

Logiquement, une large marge représente une correcte classification de l'instance par l'ensemble mais plus la marge est faible, plus les classifieurs sont en désaccord sur la classe à attribuer. En général, plus la marge est faible et plus elle est proche de la frontière. Par contre si la marge de l'instance est élevée, elle sera proche du centre du cluster de la classe. Les instances à faible marge sont moins nombreuses et l'information qu'elles fournissent est plus concluante pour la séparation des classes. Tandis que les instances centrales sont plus nombreuses et fournissent une information générale et redondante de la classe.

EMIS utilise la valeur de cette marge comme métrique pour ranger les instances de la base d'apprentissage. La figure IV.1 représente une illustration de la marge ensembliste des instances de la base Iris. Les instances appartenant au trois classes sont colorées en différentes nuances de bleu, vert et rouge en fonction de leur valeur de marge. Les instances de plus faible marge sont les plus foncé et plus la marge augmente plus l'instance est claire.

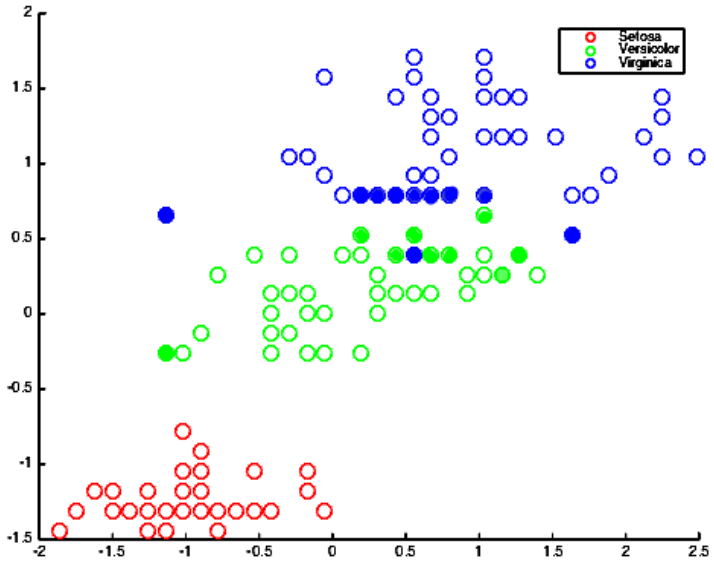


FIGURE IV.1 – Marge ensembliste

Cette figure confirme notre supposition que les instances à faible marge se situent aux frontières des clusters tandis que celles à forte marge sont au centre des clusters.

2.2 Etude des paramètres

L'algorithme EMIS est basé sur le concept de la marge ensembliste non-supervisée pour calculer l'importance de chaque instance de la base d'apprentissage. Plus la marge est faible, plus les arbres de l'ensemble sont en désaccord sur la classe de l'instance et il y'a plus de chance qu'elle se situe aux frontières d'un cluster.

L'objectif de notre algorithme est de construire un sous-ensemble de l'ensemble d'apprentissage en se basant sur le classement obtenu par la marge ensembliste non-supervisée de la forêt aléatoire à variables d'entrées aléatoires (RF-RI). Notre critère de sélection consiste à conserver un pourcentage α_1 d'instances frontalières et un pourcentage α_2 d'instances centrales.

Pour notre algorithme, nous considérons que chaque instance dont la marge est inferieuer à 0.5 est considérée comme une instance frontalière et sera sauvegardée dans l'ensemble $S1$ et le reste d'instances sera sauvegardée dans l'ensemble $S2$.

Afin d'étudier l'impact des paramètres α_1 et α_2 sur les performances d'EMIS, nous avons réalisé une série de tests avec différentes valeurs de α_1 et α_2 visant à déterminer les instances concernées par la sélection.

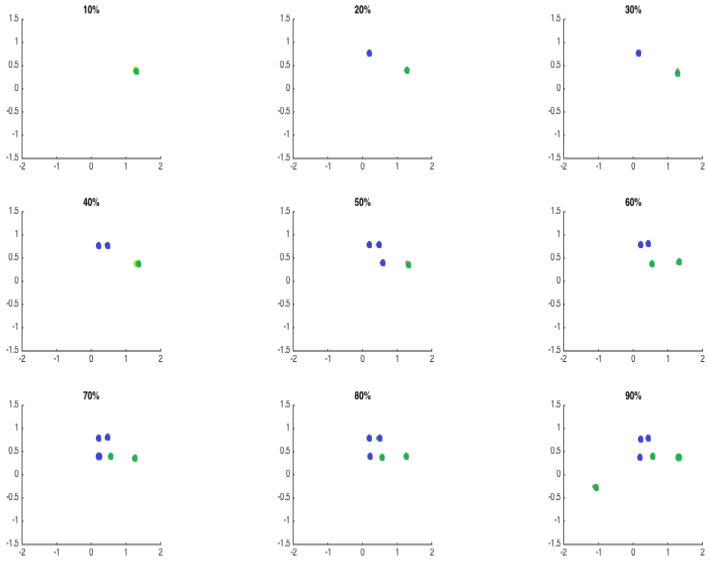


FIGURE IV.2 – Instances sélectionné pour différent valeurs de α_1

La figure IV.2 montre le sous-ensemble sélectionné de $S1$ en fonction de différente valeur de α_1. Il est possible de clairement apprécier que les points sélectionnés appartiennent aux frontières des clusters et que plus la valeur de α_1 est élevée plus la discrimination entre les classes est claire. Par contre à cause de la distribution des données dans l'espace, les instances appartenant à la classe « Setosa » ont toutes une marge élevée et sont considérées comme des instances centrales et sont éliminées, ce qui occasionne une grande perte d'information. La figure

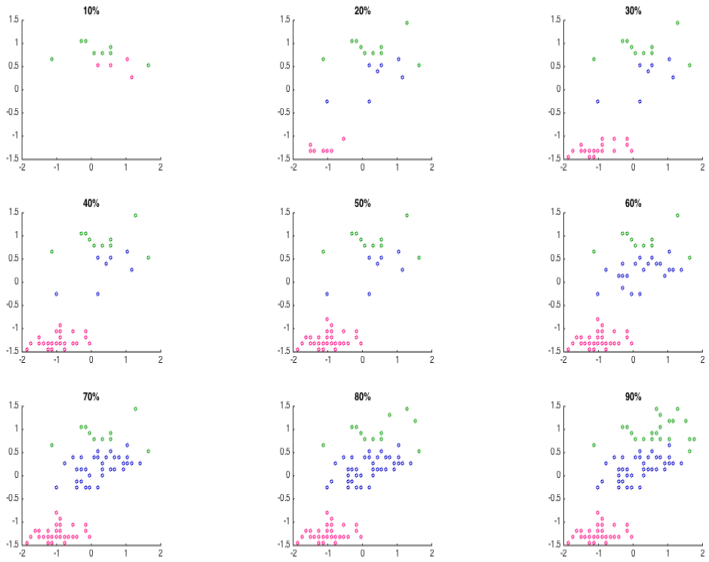


FIGURE IV.3 – Instances sélectionné pour différent valeurs de α_2

IV.3 représente les instances sélectionnées de $S2$ avec les différentes variations de

α_2. Nous pouvons constater que même un faible pourcentage d'instances à forte marge est suffisant pour une bonne représentation des données.

D'après ces expérimentations, nous en concluons que la meilleure approche est de garder le plus grand pourcentage possible d'instances à faible marge pour une bonne discrimination des classes et un faible ensemble d'instances à forte marge pour s'assurer de conserver une représentation correcte des données.

2.3 Algorithme modifié EMIS

Notre algorithme suit les étapes suivantes :
— Générer L boostraps de l'ensemble d'apprentissage A.
— Entrainer les L arbres de la forêt sur les boostraps.
— Calculer la marge ensembliste non-superviser de chaque instance de l'ensemble A.
— Trier les instances par ordre croissant de marge.
— Diviser l'ensemble en deux sous-ensemble $S1$ et $S2$. $S1$ contenant les instances dont la marge est inferieure à 0.5 et le reste d'instances est dans $S2$.
— Sélectionner un pourcentage α_1 et α_2 des premières instances des ensembles $S1$ et $S2$ respectivement.
— Entrainer un classifieur sur le sous-ensemble généré.

Algorithm 11 Algorithme EMIS

Entrés : Ensemble d'apprentissage $A = ((x_1, y_1), ..., (x_m, y_m)))$; L : Nombre d'arbres de la forêt ; α_1, α_2 : paramètres de sélection ; c : nombre de classes.
Sortie : Sous-ensemble S'.
Executer algorithme forêt aléatoire .
$M \leftarrow taille(A)$
for $i = 1 \rightarrow M$ **do**
  for $j = 1 \rightarrow L$ **do**
    if $x_i \in oob_j$ **then**
      $y_j \leftarrow h_j(x_i)$
      $cl[y_j] + +$ % moyenne de vote pour la classe y_j
    end if
  end for
end for
for $i = 1 \rightarrow L$ **do**
  $y_1 \leftarrow \max_{k=1}^{c} cl[y_j]$
  $y_2 \leftarrow \max_{\substack{k=1 \\ y_1 \neq y}}^{c} cl[y_j]$
  $Mg(x_i, y) \leftarrow (y_1 - y_2)/T$
end for
$Mg' \leftarrow$ trier Mg par ordre croissant.
for $i = 1 \rightarrow M$ **do**
  if $Mg'(x_i) < 0.5$ **then**
    $set1 \leftarrow x_i$
  else
    $set2 \leftarrow x_i$
  end if
end for
$S' \leftarrow set1(x) < \alpha_1$
$S' \leftarrow S' \cup set2(x) < \alpha_2$
Evaluer algorithme de sélection : entrainer un classifieur sur S'.

3 Expérimentation

Dans cette section nous présentons la méthodologie adoptée pour la phase d'expérimentation. L'objectif de cette phase est de vérifier si l'algorithme EMIS permet de réduire le coût d'apprentissage sans diminuer les performances du classifieur.

Nous commençons par faire différentes expérimentation afin de déterminer les meilleures valeurs des paramètres α_1 et α_2 puisqu'ils contrôlent le taux de classification et de réduction. Une fois les paramètres déterminés, nous testons les performances d'EMIS sur 13 bases de données de grande taille et nous comparons les résultats (taux de classification, taux de réduction et temps d'exécution) avec ceux de 11 approches de sélection en utilisant les classifieurs k-NN et CART.

3.1 Bases de données

Nous avons sélectionné 13 bases de données du répertoire UCI. Nous avons utilisé la procédure de 5-cross validation pour estimer les performances des sous-ensembles sélectionnés. La base est partitionnée en 5 sous-ensembles, un algorithme de sélection est appliqué à l'union de 4 d'entre eux afin d'obtenir un ensemble réduit. Nous utilisons l'ensemble obtenu pour l'apprentissage d'un classifieur qui sera testé sur le 5^{eme} sous-ensemble. Cette procédure est répétée 5 fois afin que chaque sous-ensemble soit utilisé comme ensemble de test au moins une fois.

TABLE IV.2 – Dèscription des bases de données

Bases	**# Instances**	**# Attributs**
Poker Hand	1025010	11
Covertype	581012	54
Skin Segmentation	245057	4
Shuttle	58000	9
Letter	20000	16
Magic Gamma Telescope (MGT)	19020	11
EEG Eye State	14980	15
Pendigits	10992	16
The Caravan Insurance Data (Tic)	9822	86
Hand Written Image Data (HWID)	9298	257
Thyroid Disease	7200	21
Optdigits	5620	64
Pages	5473	10

3.2 Paramétrage d'EMIS

Afin de choisir les meilleures valeurs pour les paramètres α_1 et α_2, nous avons évalué les taux de classification, de réduction et f-score en variant les pourcentages. Les tableaux IV.3, IV.4 et IV.5 présentent respectivement les taux de classification, le f-score et les taux de réduction obtenus par les valeurs suivantes de $(\alpha_1, \alpha_2) \in \{(90\%, 10\%), (80\%, 20\%), (70\%, 30\%), (60\%, 40\%)\}$.

TABLE IV.3 – Taux de classification des différentes valeurs α_1 et α_2

Bases	100	60/40	70/30	80/20	90/10
Poker hand	59,42± 0,04	51,81±0,20	53,40±0,18	50,09±0,21	48,02±0,23
Covertype	96,60±0,07	94,17±0,08	92,76±0,11	89,30±0,19	82,16±0,30
Skin Segmentation	99,97±0,01	99,93±0,03	99,83±0,04	94,29±1,27	94,28±1,30
Shuttle	99,83±0,02	99,77±0,03	99,70±0,03	99,69±0,06	99,64±0,05
Letter	95,58±0,41	90,63±1,51	88,20±0,42	83,75±1,36	78,60±0,91
MGT	82,18±0,57	81,76±0,63	83,76±0,78	80,05±1,31	77,81±1,34
EEG Eye State	94,93±0,50	90,39±0,92	87,25±0,90	87,85±0,80	77,30±1,06
Pendigits	99,47±0,13	97,26±1,03	97,26±1,01	96,33±1,17	95,33±1,43
Tic	89,87±0,42	89,26±0,35	89,91±0,53	87,94±1,02	88,30±1,30
HWID	97,47±0,53	96,51±0,56	95,70±0,73	95,05±0,46	92,53±0,85
Thyroid	93,33±0,64	92,22±0,45	91,81±0,52	91,81±0,57	90,97±0,53
Optdigits	97,78±0,35	97,06±0,67	96,35±0,70	95,64±0,68	93,77±0,73
Pages	96,26±0,50	98,08±0,69	97,72±0,91	83,45±0,78	84,11±0,63
Moyenne	92,51±0,31	90,68±0,52	90,28±0,50	87,33±0,78	84,83±0,80
Difference		0,40	2,95	2,49	

TABLE IV.4 – F-score des différentes valeurs α_1 et α_2

Bases	100	60/40	70/30	80/20	90/10
Poker hand	20,87	18,67	18,45	17,81	16,94
Covertype	93,79	90,96	88,58	84,00	76,20
Skin Segmentation	99,95	99,90	99,75	92,09	92,07
Shuttle	93,60	92,70	89,41	92,82	92,26
Letter	95,57	90,63	88,21	88,21	78,66
MGT	80,58	80,09	79,13	78,45	76,09
EEG Eye State	95,02	90,16	86,65	87,52	76,82
Pendigits	99,47	97,16	97,16	96,20	95,30
Tic	54,24	53,00	52,37	51,37	51,95
HWID	97,19	96,09	95,16	94,44	91,66
Thyroid	68,88	66,11	64,31	63,87	61,95
Optdigits	97,79	97,10	96,39	95,69	93,81
Pages	79,47	90,70	89,45	11,36	10,80
Moyenne	82,80	81,79	80,39	73,37	70,35
Difference		1,40	7,01	3,02	

Les résultats montrent qu'une faible valeur de α_2 et valeur élevée de α_1 incrémente le taux de réduction, par contre les taux de classification et le f-score décrémentent (voir Figure IV.5).

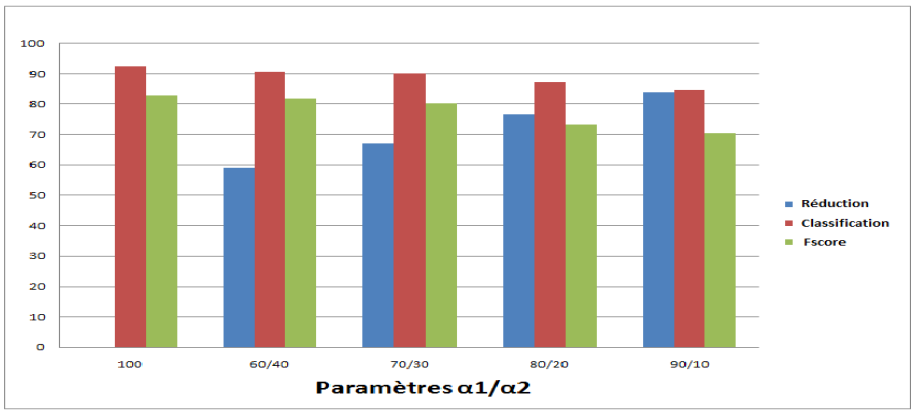


FIGURE IV.4 – Résultats des variations de α_1 et α_2

TABLE IV.5 – Taux de réduction des différentes valeurs α_1 et α_2

Bases	60/40	70/30	80/20	90/10
Poker hand	57,91±0,03	65,46±0,02	73,02±0,01	80,58±0,01
Covertype	58,19±0,01	66,37±0,01	74,56±0,02	82,74±0,03
Skin Segmentation	59,98±0,0	69,97±0,0	79,95±0,0	89,93±0,0
Shuttle	59,99±0,05	69,98±0,01	79,96±0,01	89,95±0,02
Letter	55,91±0,05	61,83±0,1	67,74±0,16	73,65±0,21
MGT	55,90±0,04	61,78±0,07	67,69±0,1	73,57±0,14
EEG Eye State	62,61±0,11	70,45±0,22	78,30±0,3	86,16±0,44
Pendigits	62,61±0,16	70,45±0,21	78,30±0,26	86,16±0,31
Tic	58,66±0,05	67,32±0,10	75,97±0,15	84,63±0,20
HWID	56,86±0,06	63,72±0,12	70,57±0,18	77,43±0,25
Thyroid	59,92±0,02	69,83±0,05	79,74±0,06	89,65±0,09
Optdigits	60,54±0,10	67,24±0,12	73,93±0,18	80,63±0,27
Pages	59,27±0,06	68,52±0,12	97,26±0,18	95,62±0,24
Moyenne	59,10±0,05	67,15±0,08	76,69±0,11	83,90±0,15
Difference	8,04	9,55	7,21	

Etant donné que la perte en taux de classification pour l'utilisation de $(\alpha_1, \alpha_2) = (60, 40)$ au lieu de $(70, 30)$ est seulement $0,4\%$ tandis que le taux de réduction s'incrémente de $8,04\%$, nous optons pour un compromis entre taux de classification et de réduction et nous sélectionnons les valeurs $(70, 30)$ pour (α_1, α_2).

3.3 Comparaison des performances

Dans cette section nous présentons les résultats obtenus par le classifieur 1-NN appliqué aux sous-ensembles générés par plusieurs algorithmes de sélection d'instances. Nous comparons l'algorithme EMIS avec IB2, IB3, CNN, RENN, All-Knn, ENN, POP, CHC, PBIL, CHC2 and PBIL2 (CHC et PBIL utilisent 1-NN l'évaluation de la fonction fitness tandis que CHC2 et PBIL2 utilisent CART).

Le tableau IV.6 présente le temps d'exécution de l'étape de sélection de chaque algorithme. A cause de leurs caractéristiques évolutionnaires, les algorithmes

CHC, PBIL, CHC2 et PBIL2 ont un temps d'exécution supérieur au reste d'algorithme. ENN, POP, PBIL et CHC sont inapplicable sur les plus grandes bases de données. D'un autre coté, IB2, IB3, CNN, RENN et All-knn sont efficace sur des bases de petite et moyenne taille mais à cause du critère de sélection basé sur la règle du plus proche voisin leur coût d'exécution s'élève avec l'augmentation la taille de la base. En définitive, les algorithmes basés sur le k-NN et les approches évolutionnaires ne sont pas adaptées aux bases de données de grande taille.

TABLE IV.6 – Temps d'exécution des algorithmes de sélection (seconde)

Bases	CNN	ENN	RENN	All-knn	IB2	IB3	POP	CHC	PBIL	PBIL2	CHC2	**EMIS**
Poker hand	44225	0	182632	82946	7568	132132	0	0	0	2960373	31245	30000
Covertype	29319	0	356891	106569	3560	40220	0	0	0	2739686	27686	28614
Skin Segmentation	70	0	9063	5508	70	4017	193	6719100	12094380	103931	4958	4781
Shuttle	90	0	835	1130	70	2160	67	13733	32143	31230	1686	1235
Letter	90	296	405	153	70	118	40	1282	3090	85728	1574	1119
MGT	90	90	112	100	70	317	20	990	1800	33176	832	761
EEG Eye State	99	160	103	90	90	300	15	827	1733	32124	783	695
Pendigits	65	40	70	90	50	78	119	426	784	11757	321	274
Tic	120	90	140	150	46	161	18	1332	3120	58583	1364	1384
HWID	120	80	227	140	49	161	68	2708	8135	167789	2802	3129
Thyroid	98	40	77	90	25	128	10	226	361	4497	221	142
Optdigits	50	40	70	60	10	61	28	238	468	26629	570	503
Pages	50	40	88	50	10	50	7	78	157	5686	245	151

Le tableau IV.7 montre les taux de réduction obtenus par les différents algorithmes de sélection tandis que le tableau IV.8 présente le taux de classification de l'application de l'algorithme k-NN sur le sous-ensemble généré. Comme nous pouvons le constatés, les meilleurs résultats de classification ont été obtenu par des algorithmes d'édition. Ces algorithmes visent à améliorer les performances des classifieurs en éliminant les instances bruitées uniquement. Par contre, le taux de réduction est faible et leurs performances diminuent si le taux de bruit est élevé. IB2 et IB3 obtiennent un taux de réduction supérieur à EMIS mais leur taux de classification est inferieur. Bien que les algorithmes évolutionnaires obtiennent des taux de classification et de réduction élevés, leur coût augmente face à des bases de grande taille.

TABLE IV.7 – Taux de réduction des algorithmes de sélection

Bases	CNN	ENN	RENN	All-knn	IB2	IB3	POP	CHC	PBIL	PBIL2	CHC2	**EMIS**
Poker hand	53,88	0	55,91	36,43	96,2	83,53	0	0	0	50,07	50,05	65,46
Covertype	90,18	0	34,56	20,85	93,3	94,74	0	0	0	47,65	50,07	66,37
Skin Segmentation	99,84	0	30,04	30,02	99,99	99,99	99,25	59,61	50,22	49,87	49,95	69,97
Shuttle	86,36	0	81,18	20,08	99,52	99,63	7,03	99,8	72,84	50,29	50,02	69,98
Letter	88,01	33,58	34,53	31,41	89,365	88,47	0	94,15	78,66	48,26	33,58	61,83
MGT	90,04	35,59	37,77	11,16	99,85	99,98	0,13	99,34	73,58	49,78	35,59	61,78
EEG Eye State	85,52	23,68	24,38	21,34	94,06	95,74	17,6	96,55	39,6	49,85	23,68	70,45
Pendigits	95,87	45,75	45,76	45,16	96,75	96,16	0	97,89	74,29	48,54	45,75	70,45
Tic	80,72	25,57	25,86	24,01	81,91	96,02	15,28	99,97	74	48,57	25,57	67,32
HWID	90,59	22,09	22,35	21,23	91,8	93,05	0	96,15	42,06	49,11	22,09	63,72
Thyroid	89,88	23,39	23,85	22,14	90,58	92,81	37,63	99,86	74,11	50,03	23,39	69,83
Optdigits	61,41	21,03	1,23	18,47	94,31	1,23	0	98,04	75,17	49,46	21,03	67,24
Pages	90,64	23,39	4,7	22,07	92,69	95,34	50,69	99,29	75,43	49,82	23,39	68,52
Moyenne	84,84	19,54	32,47	24,95	93,87	87,44	17,51	80,05	56,15	49,33	34,94	67,15
Ranking score	3	11	9	10	1	2	12	4	6	7	8	5

TABLE IV.8 – Taux de classification des algorithmes de sélection

Bases	CNN	ENN	RENN	All-knn	IB2	IB3	POP	CHC	PBIL	PBIL2	CHC2	**EMIS**
Poker hand	56,13	0	83,72	60,05	89,05	51,26	0	0	0	49,37	49,27	51,4
Covertype	95,51	0	91,76	96,35	89,16	83,28	0	0	0	92,2	91,62	92,76
Skin Segmentation	99,93	0	99,95	99,96	80,18	51,3	98,6	99,93	99,96	99,94	99,93	99,83
Shuttle	99,75	0	99,78	99,78	99,73	99,88	99,87	98	99,63	99,97	98	99,7
Letter	91,58	94,1	93,47	94,77	90,75	91	89,74	89,06	90,16	64	89,06	88,2
MGT	86,91	79,97	80,13	79,36	35,15	53,72	76,97	72,89	75,39	81,3	72,89	80,76
EEG Eye State	91,02	94,43	94,23	95,33	56	69,41	95,52	83,04	93,32	79,2	83,04	87,25
Pendigits	84,91	99,33	99,27	99,4	97,86	96,93	99,27	95,63	98,95	99,19	95,63	97,26
Tic	85,91	93,23	93,23	92,21	82,1	67,71	90,33	93,99	90,33	91,08	93,99	88,91
HWID	94,68	96,94	96,83	96,94	93,73	93,23	95,31	90,26	96,55	95	90,26	95,7
Thyroid	94,8	95,97	95,62	96,6	93,18	86,57	91,18	92,98	93,26	99,39	92,98	91,81
Optdigits	96,78	98,75	98,67	98,84	95,84	96,62	99,19	92,97	98,13	96,26	92,97	96,35
Pages	94,88	95,89	95,8	96,35	93,04	93,33	95,07	91,5	93,97	99,96	91,5	97,72
Moyenne	90,14	65,28	94,04	92,76	84,29	79,56	79,31	76,94	79,2	88,22	87,78	90,28
Friedman test	5.6923	5.3077	3.5769	6.8846	8.2308	8.3846	6.3462	6.7308	5.0769	6.6154	8.8462	6.3077
Ranking	2	/	1	5	6	7	/	/	4	8	3	

Le test non paramétrique de Freidmans indique qu'EMIS est parmi les meilleures approches de sélection. En effet, il faut noter que les algorithmes le surclassant, ENN, RENN et PBIL obtiennent un taux de réduction plus faible, tandis que CNN a un temps d'exécution trop élevé pour qu'il soit intéressant de l'appliquer sur de grandes bases, là où le besoin de réduction est le plus nécessaire. Par contre, EMIS a le coût d'exécution plus faible des algorithmes sur les bases de grandes dimension et ceci à cause de sa faible complexité : $O(LNlog(N))$, avec N le nombre d'instances et L le nombre d'arbres de la forêt. Ce qui fait de EMIS le meilleur compromis entre taux de classification, de réduction et de coût de calcul.

3.4 Comparaison avec le classifieur CART

Afin d'évaluer les performances en utilisant un autre classifieur, nous avons effectué le même protocole d'expérimentation mais en utilisant l'arbre de décision CART comme classifieur. Les résultats présentés dans le tableau IV.9 montre que EMIS conserve de bonnes performances à la différence des autres algorithmes. Nous remarquons que les algorithmes dont le critère de sélection est basé sur le k-NN ont des résultats inférieurs quand ils sont utilisés avec d'autres classifieurs. Le test de Freidman démontre qu'en utilisant l'algorithme CART, EMIS est uniquement surpassé par CHC et POP sur les moyennes bases. Par contre, ces deux approches ne sont pas applicables aux bases de données de plus grande taille. D'un autre coté, EMIS utilise les méthodes ensemblistes qui sont complètement parallélisable, ce qui peut réduire encore plus le temps d'exécution.

TABLE IV.9 – Taux de classification avec CART

Bases	CNN	ENN	RENN	All-knn	IB2	IB3	POP	CHC	PBIL	PBIL2	CHC2	**EMIS**
Poker hand	42,39	0	60,6	49,94	42,25	49,94	0	0	0	65,25	64,75	60,99
Covertype	57,92	0	57,78	70,69	58,44	58,24	0	0	0	90,34	90,68	88
Skin Segmentation	60,81	0	94,57	94,57	79,15	79,15	95,31	99,84	99,86	99,85	99,9	99,63
Shuttle	94	0	99,9	99,94	99,53	98,07	99,98	98,56	99,92	99,84	99,97	99,95
Letter	14,78	42,6	43,57	41,08	7,92	23,33	85,7	30,95	35,45	61,32	62,1	70,63
MGT	93,64	79,86	80,97	80,76	38,53	35,16	82,93	78,7	81,12	80,62	82,15	71,52
EEG Eye State	54,44	54,74	54,74	54,74	55,91	55,49	82,44	68,59	78,23	70,39	79,43	87,25
Pendigits	93,64	94,13	93	93,6	69,35	46,9	96,4	74,35	91,63	93,02	99,19	87,35
Tic	93,64	93,69	93,69	93,69	94,01	88,7	88,54	93,99	90,83	89,91	89,61	91,15
HWID	16,08	38,39	38,76	39,46	18,34	22,26	96,93	69,51	80,64	87,79	95,32	84,29
Thyroid	93,68	94,38	94,24	94,31	93,22	89,38	99,79	92,56	99,58	93,4	99,19	99,5
Optdigits	11,48	81,32	80,52	80,96	21,78	33,63	89,32	53,29	83,27	88,43	88,16	82,58
Pages	95,8	97,07	96,9	95,98	93,53	91	96,26	84,38	95,61	99,96	97,71	97,44
Moyenne	63,25	52,01	76,1	76,13	59,38	59,33	77,97	64,98	72,01	86,16	88,32	86,18
Friedman test	8.3077	8.5385	6.3462	6.0769	8.6538	9.6154	4.2308	2.7692	4.8462	6.0769	8.2308	4.3077
Ranking	6	/	4	3	7	8	/	/	2	5	1	

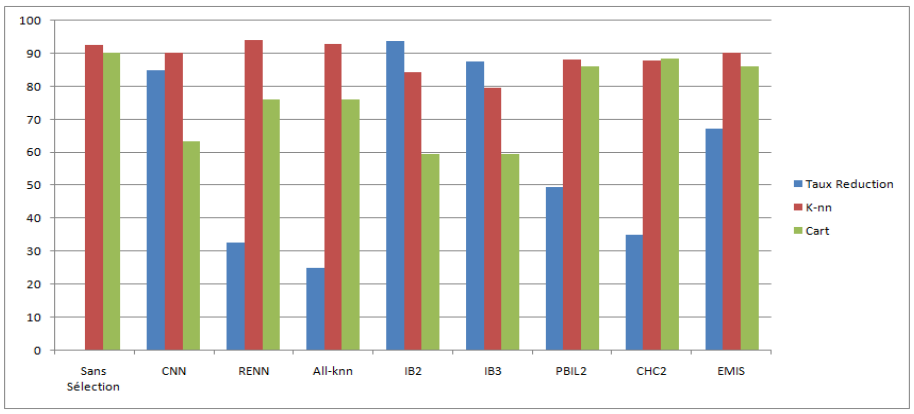


FIGURE IV.5 – Résultats des comparaisons

3.5 Résistance au bruit

Afin d'étudier l'effet du bruit sur les performances de notre algorithme, nous avons suivi la procédure présentée dans le papier [114]. Nous avons ajouté manuellement un niveau de bruit artificiel à chaque base de données. Pour un problème de classe binaire (X, Y) et un niveau de bruit x, une instance de classe X à $x.100\%$ de chance d'être corrompu et labélisé en Y. Pour notre travail, nous avons utilisé quatre niveau de bruit $(5\%, 10\%, 20\% et 30\%)$; le niveau de bruit exprime le pourcentage d'instances corrompues de la base initiale. Nous avons sélectionné la meilleure approche de chaque catégorie, RENN (édition), CNN (condensation), IB2 (hybride) et CHC (méta-heuristique), pour effectuer une comparaison.

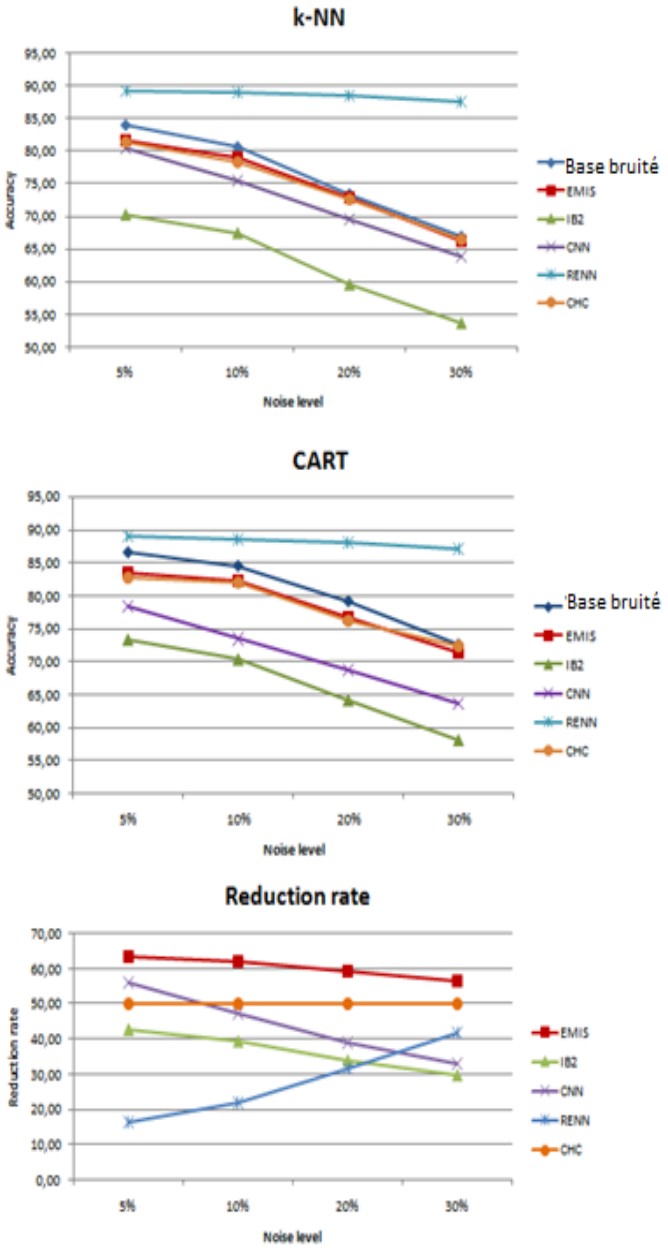


FIGURE IV.6 – Taux de classification et de réduction pour les bases bruitées

La figure IV.8 montre la moyenne des taux de réduction et de classification pour chaque algorithme de sélection pour l'ensemble des bases de données bruitées. Les meilleurs résultats de classification furent obtenus par un algorithme d'édition (RENN) suivi par EMIS. Par contre, le taux de réduction le plus élevé fut obtenu par EMIS. Ces résultats prouvent la robustesse de notre approche au bruit.

4 Application de la sélection d'instance pour la segmentation automatique des globules blancs

La reconnaissance automatique de globules blancs peut aider les hématologistes à diagnostiquer des maladies telles que la leucémie, le SIDA ou le cancer du sang, ce qui rend ce processus très important dans les procédures hématologiques. En général, ce processus nécessite des instruments de cytométrie en flux ou des compteurs cellulaires. L'inconvénient est que ces instruments ne possèdent pas de techniques d'analyse d'images et ne sont pas capable d'examiner les types de cellules. Donc l'utilisation de systèmes incluant des techniques de traitement d'images peut améliorer la segmentation d'image, réduire les erreurs de scan manuel et ainsi accélérer le processus de diagnostic.

En segmentation d'image par classification de pixels, nous considérons que chaque pixel de l'image transporte des informations concernant le label. Par contre, tous les pixels n'apportent pas le même type d'information.

Bien que la classification des pixels peut significativement améliorer le coût de calcul du traitement d'image, le processus en lui-même peut engendrer un surcroît de complexité, spécialement si les dimensions de l'image sont grandes et nous devons considérer tous les pixels.

En tenant compte de ces constatations, nous proposons d'appliquer une étape de sélection de pixels qui nous permet d'optimiser le processus de reconnaissance automatique du noyau et cytoplasme par classification des pixels.
Nous avons appliqué une étape de prétraitement, afin de réduire la taille de la base d'apprentissage. Nous comparons les performances de 5 algorithmes de sélection d'instances (édition, condensation, hybride et meta-heuristique) avec celle de notre approche.

La classification de pixels utilisés dans ce travail est basée sur la croissance de région. Cette approche consiste à classer les pixels voisins d'un point d'intérêt par apprentissage supervisé. Le point d'intérêt est détecté par un opérateur d'érosion morphologique. Finalement, l'algorithme de forêt aléatoire est utilisé pour classer les pixels pertinents. Cette approche donne une bonne segmentation d'image cytologique (noyau et cytoplasme) et un temps réduit et avec une intervention minimale de l'expert.

4.1 État de l'art de segmentation d'images cytologique

Plusieurs travaux ont été menés sur la segmentation des cellules. Nous pouvons citer Rezatofighi et al. [115] une approche pour la classification de cinq types de globules blancs : éosinophiles, basophiles, monocytes, lymphocytes et neutrophiles. Le processus est décomposé en deux phases, en premier lieu, les noyaux des cellules sont segmentés par la méthode orthogonalisation de Gram-Schmidt puis l'algorithme Snake est utilisé pour segmenter le cytoplasme.

Ben Chaabane et al. [116] ont fusionné le FCM (Fuzzy C-Means) et la théorie de Dempster-Shafer Evidence pour la segmentation de globules blancs. Dans un autre article Ben Chaabane et al. [117] proposent d'ajouter le principe de homogénéité floue à l'approche précédente afin d'améliorer la segmentation. Hiremath et al. [118] proposent une segmentation automatique basée sur l'égalisation d'histogramme, le seuillage et les algorithmes de détection de frontière pour l'identification et la classification de leucocytes.

D'autre travaux se sont portés sur l'analyse de contour, approche initialement proposée par Liao and Deng [119]. L'idée étant d'effectuer une segmentation basique avec un seuillage puis utiliser une analyse de contours pour détecter les globules blancs. Malheureusement, cette approche est uniquement applicable au globule de forme circulaire. En [120], Ko et al. proposent d'appliquer l'algorithme de clustering mean-shift pour une segmentation par région puis un schéma de fusion afin de fusionner les clusters en noyaux et une ouverture morphologique est utilisée pour la segmentation du cytoplasme.

Dans leur article [121], les auteurs ont segmenté les images avec l'algorithme SVM après avoir utilisé différent espaces de couleur pour la caractérisation des pixels de l'image puis ont appliqué des techniques de sélection de variables (*ReliefF*, *mRMR*, *SVM-RFE* and *LDA*) afin d'étudier l'importance des variables. Benomar et al. [122] ont appliqué la même approche avec les attributs Haralick des images en niveau de gris.

4.2 Approche proposée

L'idée est d'utiliser une étape de sélection d'instances afin de trouver le meilleur compromis entre performance et temps d'exécution pour la construction d'un modèle de classification de pixels. L'approche proposée est divisée en deux phases : l'apprentissage et la segmentation.

Phase d'apprentissage : nous effectuons une extraction de caractéristiques pour chaque image de la base, chaque pixel est caractérisé par un vecteur de paramètre de différents espaces de couleurs (IV.10). Puis nous appliquons une sélection d'instance afin de préparer une base d'apprentissage ne contenant que les pixels les plus informatifs. La base générée est utilisée pour construire un classifieur avec l'algorithme forêt aléatoire.

Phase de segmentation : effectuer la même caractérisation pour les images de tests. Nous appliquons l'opérateur d'érosion pour calculer les points d'intérêts. En utilisant le classifieur de la phase d'apprentissage, nous effectuons une approche de croissance de région en classifiant chaque pixel du voisinage du point d'intérêt.

Espaces Couleur	Equation
RGB	$R(i,j)$ $G(i,j)$ $B(i,j)$
LUV	$L = 116(\frac{Y}{Y_n})^{1/3} - 16$ If $\frac{Y}{Y_n} > 0.008856$ $= 903.3(\frac{Y}{Y_n})$ If $\frac{Y}{Y_n} \leq 0.008856$ $U = 13L(U' - U'_n)$ $V = 13L(V - V'_n)$
HSV	H = $\frac{G-B}{(Max-Min)}$ If R = Max = $\frac{B-R}{(Max-Min)}$+2 Si G = Max = $\frac{R-G}{(Max-Min)}$+4 Si B = Max S = $\frac{Max(R,G,B)-Min(R,G,B)}{Max(R,G,B)}$ $V = Max(R,G,B)$
YUV	$Y = 0.2989R + 0.5866G + 0.1145B$ $U = 0.5647(B - Y) = -0.1687R - 0.3312G + 0.5B$ $V = 0.7132(R-Y) = 0.5R-0.4183G-0.0817B$

TABLE IV.10 – Les formules des différents espaces de couleurs

Les étapes peuvent être décrites comme suit :

— Phase d'apprentissage

  — Extraction de caractéristique : extraction des quatre espaces de couleur les plus utilissé en traitement d'image [123] : RGB, LUV, HSL, YUV.
  — Application d'algorithme de selection d'instance : IB2, All-knn, CNN, PBIL, CHC et EMIS.
  — Effectuer un apprentissage par forêt aléatoire.

— Phase de segmentation

  — Extraction de caractéristique : RGB, LUV, HSL, YUV.
  — Calcule des points d'intérêts par érosion ultime (figure IV.7).
  — Application d'approche de croissance de région par classification des voisinages des pixels.

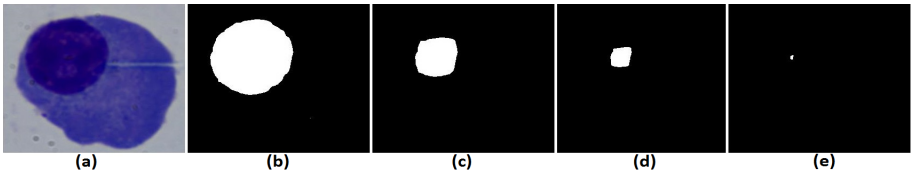


FIGURE IV.7 – Principe d'érosion ultime. (a) image originale, (b, c and d) séquence d'érosion du noyau, (e) résultat d'érosion du noyau.

More formally, the ultimate erosion UE of a set X is defined by :

$$UE(X) = \bigcup_n \frac{E(X, B_n)}{R[E(X, B_{(n+1)}); E(X, B_n)]} \tag{IV.2}$$

4.3 Base de données

La base d'images cytologiques utilisées fût obtenue à partir d'images réel du service de hémobiologie du CHU Tlemcen sur des slides de coloration MGG (May Grunwald Giemsa). L'environnement LEICA (caméra et microscope) a fournie des images couleur RGB de taille 768x1024. La base est constituée de 60 images cytologiques labélisées par l'expert, nous en avons utilisé 10 pour l'apprentissage et le reste pour le test (Fig.IV.8).

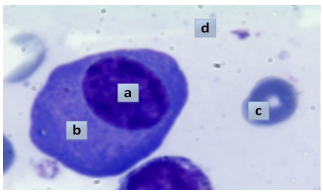


FIGURE IV.8 – (a) Noyau, (b) cytoplasme, (c) globule rouge et (d) plasma.

4.4 Résultats et discussions

Dans cette section, nous présentons les résultats des tests effectués. Afin d'évaluer les performances de notre algorithme de sélection, nous comparons les résultats sur la base du taux de classification et de réduction ainsi que le temps d'exécution des phases de sélection et d'apprentissage. Le tableau IV.11 présente les paramètres de chaque algorithme de sélection, le taux de réduction et le temps d'exécution de l'algorithme de sélection et celui de l'algorithme d'apprentissage sur la base réduite.

TABLE IV.11 – Paramètres des algorithmes de sélection et temps d'exécution

Méthodes	Paramèters	Réduction Rate (%)	Temps sélection (mn)	Temps apprentissage (mn)
AlKnn	k=16	17.30	3000	881
CHC	Pop=50, Gen=25	66.69	2083	300
CNN	k=1	82.85	4320	130
EMIS	N° tree=50	79,55	271	150
IB2	k=1	86.67	720	110
PBIL	Pop=50, Gen=25, learning rate=0.1,	50.50	1873	380

Le tableau IV.12 résume les performances de segmentation obtenues par apprentissage sur l'ensemble d'origine de 50 images test ou sur l'ensemble réduit. Cette évaluation est basée sur le taux de classification, le f-score et le temps d'exécution de l'étape de segmentation. La figure IV.9, affiche les performances de segmentation de 4 images de la base de test avec et sans étape de sélection.

TABLE IV.12 – Performance de classification avec et sans réduction

Régions	CYTOPLASME		NOYAU		Temps
Méthodes	**Classification**	**F-score**	**Classification**	**F-score**	d'exécution
Before selection	97.15	0.89	99.12	0.95	27204
AlKnn	97.91	0.84	99.55	0.92	3881
CHC	97.41	0.78	97.22	0.88	2383
CNN	92.52	0.55	96.97	0.78	4450
EMIS	95,05	0,61	99,05	0,88	421
IB2	94.32	0.65	96.63	0.800	830
PBIL	96.52	0.80	98.99	0.89	2253

Les résultats montrent que l'approche d'édition, All-knn, obtient le meilleur taux de classification, et améliore les performances de la base d'origine. Le sous-ensemble généré par All-knn réalise une bonne segmentation des image de test (figure IV.2). Par contre, le taux de réduction est faible (17,30%) donc le temps d'exécution reste très élevé.

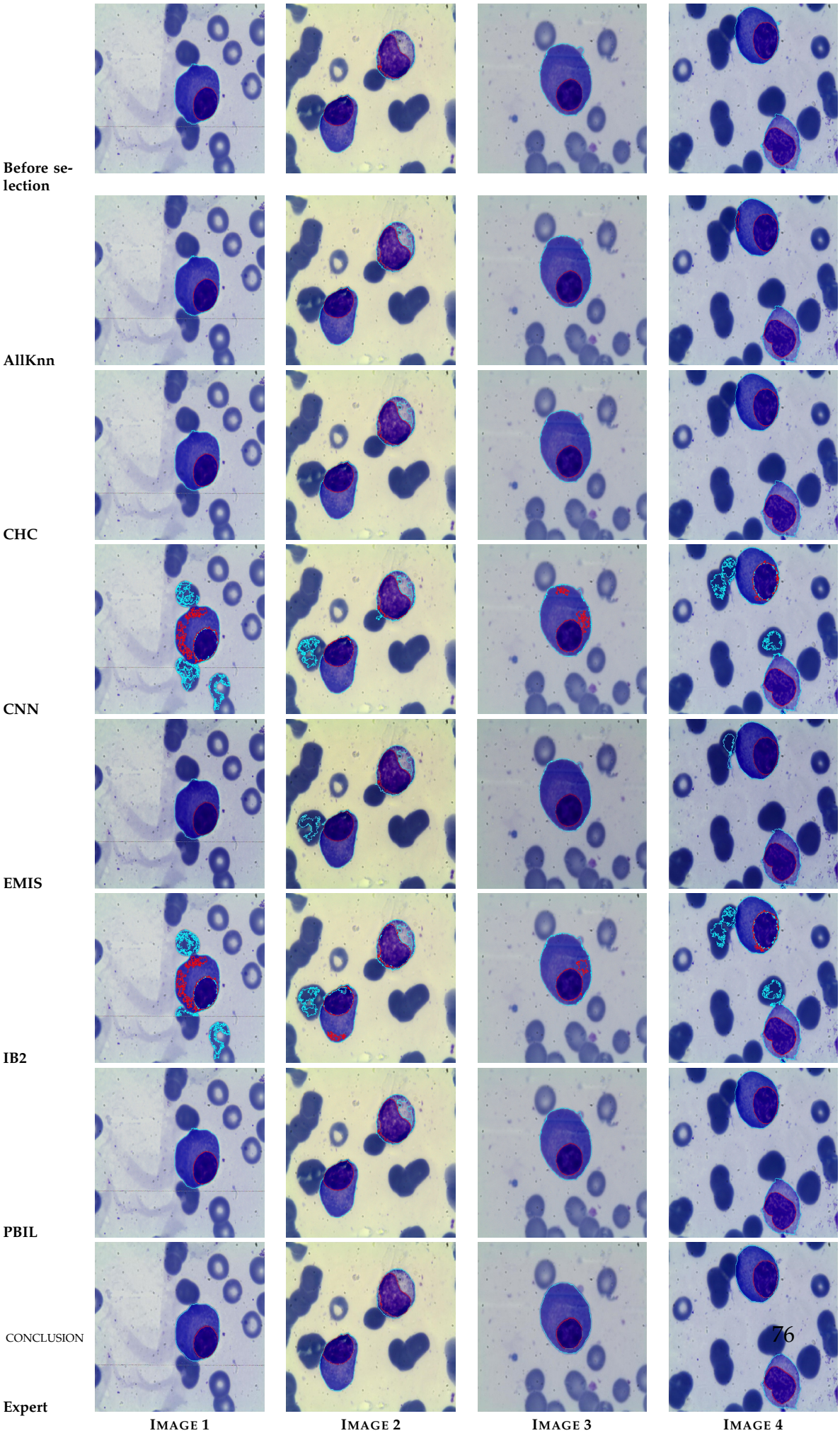
L'algorithme CNN obtient les pires performances. Il obtient un taux de réduction très élevé mais au détriment de la segmentation. Ceci peut être expliqué par le critère de sélection de CNN qui ne conserve que les instances aux frontières des clusters. Son principal inconvénient est sa sensibilité au bruit ce qui fait que la proportion de bruit dans la base réduite est supérieur à celui de la base d'origine. L'algorithme IB2 génère le plus petit sous-ensemble mais avec une meilleure performance que CNN. Par contre, il souffre de la même sensibilité au bruit et la segmentation reste insatisfaisante.

Les approches évolutionnaires CHC et PBIL obtiennent d'excellents résultats en taux de classification et réduction. Bien que les performances de CHC soit meilleures que celle de PBIL, il est plus lent. La principale limitation de ces algorithmes est le coût élevé de calcul mais le taux élevé de réduction et la bonne segmentation rendent ce compromis acceptable.
EMIS atteint des performances similaire avec un plus grand taux de réduction et le temps d'exécution le plus faible des algorithmes de sélection.

5 Conclusion

La plupart des bases de données existantes contiennent des informations bruitées et redondantes, ce qui peut dégrader les performances des classifieurs. Une étape de prétraitement visant à nettoyer la base permettra d'éliminer les instances superflues. Dans ce chapitre, nous avons proposé un algorithme modifié de sélection d'instances basées sur le principe de marge ensembliste nommé EMIS. Le concept sur lequel se base EMIS permet d'affronter le problème des ressources limitées des algorithmes d'apprentissage existant.

La phase d'expérimentation a démontré qu'EMIS obtient des résultats similaires aux meilleurs algorithmes de sélection d'instances de l'état de l'art. D'un autre coté, la faible complexité d'EMIS rend son utilisation idéal pour les bases de très grande dimension. En outre, étant donné qu'EMIS est basé sur les méthodes


Before selection
AllKnn
CHC
CNN
EMIS
IB2
PBIL
Expert
IMAGE 1
IMAGE 2
IMAGE 3
IMAGE 4

ensemblistes, il est complètement parallélisable et peut être adapté à n'importe quel classifieur.

Nous avons aussi évalué les performances de notre algorithme sur une base réel d'image cytologique. L'objectif de l'expérience était de réduire la taille de la matrice d'apprentissage afin d'effectuer une reconnaissance automatique des globules blancs. Les résultats ont démontré qu'EMIS obtient la même segmentation que celle des algorithmes les plus performants de l'état de l'art mais en un temps significativement plus réduit.

Conclusion et Perspectives

Les différents domaines de recherche tels que la biologie, l'économie ou l'astrophysique génèrent actuellement un flux constant de données. Cette masse de données a engendré des bases de données de grandes dimensions impossibles à interpréter manuellement. D'où la nécessité d'outils informatiques pour la fouille des données et l'extraction des connaissances telles que le clustering ou l'apprentissage artificiel, entre autres. Ce domaine regorge d'algorithmes variés pour traiter différents types de données (alphanumériques, images, son, ...), par exemple le k-NN, les algorithmes évolutionnaires ou les méthodes ensemblistes pour en cité que quelques-uns.

Ces méthodes ont permis pendant des années de traiter et d'extraire des connaissances des bases de données. Malheureusement, ils se retrouvent actuellement face à un problème du goulot d'étranglement d'acquisition de connaissance (bottleneck). A cause de l'énorme masse de données disponible. En effet, ces algorithmes furent initialement implémentés pour traiter des vases de quelques centaines voir quelque milliers d'instances. Donc, ils se retrouvent dans l'incapacité de gérer des bases composées de centaines de milliers ou de million d'individus en un temps acceptable.

L'objet de ce travail de recherche est d'améliorer le traitement des bases de données biologiques de grande taille. Il existe deux solutions pour ce problème : la première option est d'implémenter des algorithmes capables de gérer des grandes bases ou d'appliquer un prétraitement pour réduire en ne sélectionnant que les instances les plus pertinentes les bases afin qu'elles soient compatibles avec les algorithmes existants.

Cette thèse s'inscrit dans le cadre des méthodes de réduction. Son but majeur est d'implémenter une méthode de sélection d'instance permettant d'éliminer les instances bruitées et redondantes et de ne garder que les plus informatives. Nous nous sommes ainsi intéressé aux méthodes ensemblistes et plus particulièrement aux forêts aléatoires. Les forêts aléatoires ont la particularité d'utiliser que les arbres de décision CART comme classifieur particulièrement adapté aux méthodes d'ensemble à cause de leur instabilité.

Dans un premier temps, nous avons étudié quelques-uns des méthodes de sélection d'instances existantes et notamment aux différentes métriques, concepts et critères de sélection. Nous avons pu n'ainsi déterminer que la principale limitation de ces algorithmes est le temps d'exécution élevé ce qui les rend inapplicables sur les grandes bases alors que c'est là où elles sont le plus nécessaire.

Notre contribution consiste en l'implémentation d'une approche de sélection basée sur les méthodes ensemblistes nommées EMIS (Ensemble Margin instance Sélection). EMIS utilisée comme critère de sélection la marge ensembliste non supervisé, un concept des algorithmes d'ensemble permettant de mesurer le degré de désaccord de l'ensemble sur la classe d'une instance.

Notre approche est fondée sur la croyance que plus l'ensemble est en désac-

cord sur le label d'une instance plus probablement elle appartient aux frontières des clusters et plus elle apporte d'information sur la discrimination entre les classes. D'un autre côté les instances internes apportent aussi des informations générales sur la distribution des données dans l'espace. Dès lors, notre approche vise à conserver un pourcentage (α_1) élevé d'instances ayant une marge élevée qui représente les instances frontalières et un faible pourcentage (α_2) d'instances centrales.

Nous avons ainsi testé plusieurs pourcentages (α_1 et α_2) pour déterminer les valeurs optimales. Puis nous avons comparé les performances d'EMIS avec celle de plusieurs algorithmes de l'état de l'art sur différentes bases de grande taille.

Nous avons alors montré que cette approche remplit son objectif de réduction sans perte d'information avec un gain en temps d'exécution surtout en comparaison des algorithmes de sélection classiques.

La deuxième étape de notre travail était l'exploitation de notre approche sur un problème biologique réel. Pour cela, nous avons utilisé une base d'images cytologiques collectées du service d'hémobiologie du CHU Tlemcen. L'objectif de l'expérience était de réduire la taille de la matrice d'apprentissage afin d'effectuer une reconnaissance automatique des globules blancs. Les résultats ont démontré qu'EMIS obtient la même segmentation que celle des algorithmes les plus performants de l'état de l'art mais en un temps significativement plus réduit.

Ce travail, bien qu'il apporte des réponses au problème que rencontrent actuellement les algorithmes d'apprentissage, il ouvre la voie à de nouvelles perspectives. En premier, nous pensons qu'il serait intéressant d'automatiser le choix des paramètres α_1 et α_2 en fonction de la structure des données.
Deuxièmement, nous nous intéressons aussi à la sélection simultanée d'instance et de variables dans les bases à grande dimension comme celle d'images nécessitant différents types de caractérisation (couleur, spécial, texture) ou les bases de séquençage du génome.
Enfin, une dernière perspective est l'utilisation de notre algorithme pour la sélection d'instances à labéliser en apprentissage actif et le rééchantillonnage des bases déséquilibrées.

Bibliographie

[1] H. Liu and H. Motoda, "On issues of instance selection," *Data Mining and Knowledge Discovery*, vol. 6, pp. 115–130, 2002.

[2] Oleg Okun, *Feature Selection and Ensemble Methods for Bioinformatics : Algorithmic Classification and Implementations*, Medical Information Science Reference, 2011.

[3] H. Brighton and C. Mellish, "Advances in instance selection for instance-based learning algorithms," *Data Mining and Knowledge Discovery*, vol. 6(2), pp. 153–172, 2002.

[4] J.R Cano, F Herrera, and M Lozano, "Using evolutionary algorithms as instance selection for data reduction in kdd : An experimental study," *IEEE Transactions On Evolutionary Computation*, 2003.

[5] D. Ryu, J. I. Jang, and J. Baik, "A hybrid instance selection using nearest-neighbor for cross-project defect prediction," *Journal of Computer Science and Technology*, vol. 30, pp. 969–980, 2015.

[6] C-F Tsai, W. Eberle, and C-Y. Chu, "Genetic algorithms in feature and instance selection," *Knowledge-Based Systems*, vol. 39, pp. 240–247, 2013.

[7] J. A . Olvera-Lopez, J. A . Carrasco-Ochoa, J.F. Martinez-Trinidad, and J Kittler, "A review of instance selection methods," *Artificial Intelligence Review*, vol. 34, pp. 133–143, 2010.

[8] R. Genuer, *Forets aleatoires : aspects theoriques, selection de variables et applications*, Ph.D. thesis, Universite Paris-Sud XI, 2010.

[9] S. Dzeroski, P. Panov, and B. enko, *Ensemble Methods in Machine Learning*, Springer New York, 2012.

[10] M. J. Zaki and W. Meira, *Data Mining and Analysis : Fundamental Concepts and Algorithms*, Cambridge University Press, New York, NY, USA, 2014.

[11] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society*, pp. 111–147, 1974.

[12] Q. Xu and Y. Liang, "Monte carlo cross validation," *Chemometrics and Intelligent Laboratory Systems*, vol. 56, no. 1, pp. 1 – 11, 2001.

[13] Schoelkopf B. Chapelle, O. and A. Zien, Eds., *SemiSupervised Learning. Adaptive Computation and Machine Learning.*, The MIT Press, 2006.

[14] Y. Fu, X. Zhu, and B. Li, "A survey on instance selection for active learning," *Knowledge and Information Systems*, vol. 35, pp. 249–283, 2013.

[15] D. J. Hsu, "Algorithms for active learning," Tech. Rep., University of California - San Diego, 2010.

[16] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46 (3), pp. 175–185, 1992.

[17] V. Vapni k, *Statistical Learning Theory*, John Wiley and Sons, 1998.

[18] C. C. Aggarwal, *Data Mining : The Textbook*,, chapter Data Classi-cation, pp. 285–344, Springer International Publishing Switzerland, 2015.

[19] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification And Regression Trees*, Chapman and Hall, New York, 1984.

[20] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[21] J. Ross Quinlan, *C4.5 : Programs for Machine Learning*, Morgan Kaufmann, 1993.

[22] L. Breiman, J. H. Friedman, R. Olshen, and C. J. Stone, "Classification and regression trees," in *Wadsworth, Belmont, California*, 1984.

[23] D. Wilson and T. Martinez, "Reduction techniques for instance-based learning algorithms," *Machine Learning*, 2000.

[24] J.A. Olvera-Lopez, J.A. Carrasco-Ochoa, and J.F. Martinez-Trinidad, "A new fast prototype selection method based on clustering," *Pattern Analysis and Applications*, vol. 13(2), pp. 131–141, 2010.

[25] D. Wettschereck and T. G Dietterich, "An experimental comparison of nearest-neighbor and nearest hyperrectangle algorithms," *Machine Learning*, vol. 19, pp. 5–28, 1995.

[26] F. Chang, C.C. Lin, and C.J. Lu, "Adaptive prototype learning algorithms : Theoretical and experimental studies.," *Journal of Machine Learning Research*, vol. 7, pp. 2125–2148, 2006.

[27] P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning*, vol. 29, pp. 103–137, 1997.

[28] S. Garcia, S.a, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*, chapter Instance Selection, pp. 195–243, Springer International Publishing Switzerland, 2015.

[29] P.E. Hart, "The condensed nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 14, pp. 515 – 516, 1968.

[30] C-H. Chou, B-H. Kuo, and F. Chang, "The generalized condensed nearest neighbor rule as a data reduction method," *International Conference on Pattern Recognition*, 2006.

[31] G.W. Gates, "The reduced nearest neighbor rule," *IEEE Transactions on Information Theory*, vol. 22, pp. 431–433, 1972.

[32] T. Raicharoen and C. Lursinsap, "A divide-and-conquer approach to the pairwise opposite class-nearest neighbor (poc-nn) algorithm," *Pattern Recognition Letters*, 2005.

[33] J.C. Riquelme, Aguilar-Ruiz, and M. J.S., Toro, "Finding representative patterns with ordered projections," *Pattern Recognition*, vol. 36(4), pp. 1009–1018, 2003.

[34] C. Liu, W. Wang, M. Wang, F. Lv, and M. Konan, "An efficient instance selection algorithm to reconstruct training set for support vector machine," *Knowle dge-Base d Systems*, vol. 116, pp. 58–73, 2017.

[35] J.R. Ullmann, "Automatic selection of reference data for use in a nearest-neighbor method of pattern classication," *IEEE Transactions on Information Theory*, vol. 24, pp. 541–543, 1974.

[36] G.L. Ritter, H.B. Woodruff, S.R. Lowry, and T.L. Isenhour, "An algorithm for a selective nearest neighbor decision rule," *IEEE Transactions on Information Theory*, vol. 25, pp. 665–669, 1975.

[37] I. Tomek, "Two modications of cnn," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6(6), pp. 769–772, 1976.

[38] V.S. Devi and M.N. Murty, "An incremental prototype set building technique," *Pattern Recognition*, vol. 35(2), pp. 505–513, 2002.

[39] R. Gil-Pita and X. Yao, "Evolving edited k-nearest neighbor classiers," *International Journal of Neural Systems*, vol. 18(6), pp. 459–467, 2008.

[40] D. Kibler and D.W. Aha, "Learning representative exemplars of concepts : an initial case study," in *Proceedings of the Fourth International Workshop on Machine Learning*, 1987, pp. 24–30.

[41] R. Barandela, F.J. Ferri, and J.S. Snchez, "Decision boundary preserving prototype selection for nearest neighbor classication," *Int. J. Pattern Recognit and Artificial Intelligent*, vol. 19(6), pp. 787–806, 2005.

[42] F. Angiulli, "Fast nearest neighbor condensation for large data sets classication," *IEEE Trans. Knowl. Data Eng*, vol. 19(11), pp. 1450–1464, 2007.

[43] D. Wilson, "Asymptotic properties of nearest neighbor rules using edited data," *IEEE Transactions on Systems, Man and Cybernetics*, 1972.

[44] K. Hattori and M. Takahashi, "A new edited k-nearest neighbor rule in the pattern classication problem," *Pattern Recognit*, vol. 33(3), pp. 521–528, 2000.

[45] I. Tomek, "An experiment with the edited nearest-neighbor rule," *IEEE Transactions Systems Man Cybernetics*, vol. 6, pp. 448–452, 1976.

[46] P.A. Devijver, "On the editing rate of the multiedit algorithm," *Pattern Recogntion Letters*, vol. 4, pp. 9–12, 1986.

[47] J.S. Snchez, F. Pla, and F.J. Ferri, "Prototype selection for the nearest neighbor rule through proximity graphs," *Pattern Recognitin Letters*, vol. 18, pp. 507–513, 1997.

[48] J.S. Snchez, R. Barandela, A.I. Marqus, R. Alejo, and J. Badenas, "Analysis of newtechniques to obtain quality training sets," *Pattern Recognition Letters*, vol. 24(7), pp. 1015–1022, 2003.

[49] N. Jankowski and M. Grochowski, "Comparison of instances seletion algorithms i. algorithms survey," in *International Conference on Artificial Intelligence and Soft Computing - ICAISC 2004*, 2004, pp. 598–603.

[50] F. Vazquez, J.S. Snchez, and F. Pla, "A stochastic approach to wilsons editing algorithm," in *2nd Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2005, pp. 35–42.

[51] N. Segata, E. Blanzieri, S.J. Delany, and P. Cunningham, "Noise reduction for instance-based learning with a local maximal margin approach," *Journal of Intelligent Information Systems*, vol. 35(2), pp. 301–331, 2010.

[52] N. Verbiest, C. Cornelis, and F. Herrera, "Frps : a fuzzy rough prototype selection method," *Pattern Recognition*, vol. 46(10), pp. 2770–2782, 2013.

[53] D.H. Aha, D Kibler, and M.K Albert, "Instance-based learning algorithms," *Machine Learning*, vol. 6, pp. 37–66, 1991.

[54] P. Hernandez-Leal, J.A. Olvera-Lopez, J.A. Carrasco-Ochoa, and J.F. Martinez-Trinidad, "Instancerank based on borders for instance selection," *Pattern Recognition*, 2013.

[55] U. Lipowezky, "Selection of the optimal prototype subset for 1-nn classication," *Pattern Recogntion Letters*, vol. 19(10), pp. 907–918, 1998.

[56] B. Sierra, E. Lazkano, I. Inza, M. Merino, P. Larraaga, and J Quiroga, "Prototype selection and feature subset selection by estimation of distribution algorithms. a case study in the survival of cirrhotic patients treated with tips," in *Proceedings of the 8th Conference on AI in Medicine in Europe*, 2001, pp. 20–29.

[57] M. Sebban and R. Nock, "Instance pruning as an information preserving problem," in *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000, pp. 855–862.

[58] K.P. Zhao, S.G. Zhou, J.H. Guan, and A.Y. Zhou, "C-pruner : An improved instance pruning algorithm," in *Proceeding of the 2th International Conference on Machine Learning and Cybernetics*, 2003, pp. 94–99.

[59] J.A. Olvera-Lopez, J.F. Martnez-Trinidad, and J.A. Carrasco-Ochoa, "Edition schemes based on bse," in *10th Iberoamerican Congress on Pattern Recognition (CIARP)*, 2005, pp. 360–367.

[60] X.Z. Wang, B. Wu, Y.L. He, and X.H. Pei, "Nrmcs : Noise removing based on the mcs," in *Proceedings of the Seventh International Conference on Machine Learning and Cybernetics*, 2008, pp. 89–93.

[61] E. Marchiori, "Hit miss networks with applications to instance selection," *Journal of Machine Learning Research*, vol. 9, pp. 997–1017, 2008.

[62] Z-Y Chen, C-H Tsai, W Eberle, W-C Lin, and S-W Ke, "Instance selection by genetic-based biological algorithm," *Soft Computing*, vol. 19, pp. 1269–1282, 2015.

[63] M. A. Potter and K.A. De Jong, "Cooperative coevolution : An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, pp. 1–29, 2000.

[64] N Garcia-Pedrajas, J.A. Romero del Castillo, and D Ortiz-Boyer, "A cooperative coevolutionary algorithm for instance selection for instance-based learning," *Machine Learning*, vol. 78, pp. 381–420, 2010.

[65] J. Derrac, S. Garcia, and F. Herrera, "Ifs-coco : Instance and feature selection based on cooperative coevolution with nearest neighbor rule," *Pattern Recognition*, vol. 49, pp. 2082–2105, 2010.

[66] Y. Li, Z. Hu, Y. Cai, and W. Zhang, "Support vector based prototype selection method for nearest neighbor rules," in *First InternationalConference on Advances in NaturalComputation (ICNC), Lecture Notes in Computer Science*, 2005.

[67] B. Scholkopf, C. Burges, and V. Vapnik, "Extracting support data for a given task," in *In Proceedings of the First International Conference on Knowledge Discvoery and Data Mining*, 1995.

[68] L. Guo and S. Boukir, "Fast data selection for svm training using ensemble margin," *Pattern Recognition Letters*, vol. 51, pp. 112–119, 2015.

[69] S. Garcia, J. R. Cano, and F. Herrera, "A memetic algorithm for evolutionary prototype selection : A scaling up approach," *Pattern Recogn*, vol. 41, pp. 2693–2709, 2008.

[70] G. P. Figueredo, N. F. F. Ebecken, D. A. Augusto, and H. J. C. Barbosa, "An immune-inspired instance selection mechanism for supervised classification.," *Memetic Computing*, vol. 4, pp. 135–147, 2012.

[71] A. Cervantes, I. Galván, and P. Isasi, "Michigan particle swarm optimization for prototype reduction in classification problems," *New Generation Computing*, vol. 27, pp. 239–257, 2009.

[72] A. Lumini and L. Nanni, "A clustering method for automatic biometric template selection," *Pattern Recognition*, vol. 39, pp. 495–497, 2006.

[73] M. Blachnik and M. Kordos, *Bagging of Instance Selection Algorithms*, pp. 40–51, Springer International Publishing, Cham, 2014.

[74] N. Garcia-Pedrajas and A. Haro-Garcia, "Boosting instance selection algorithms," *Knowledge-Based Systems*, vol. 67, pp. 342 – 360, 2014.

[75] A. De Haro-Garcia and N. Garcia-Pedrajas, "A divide-and-conquer approach for scaling up instance selection algorithm.," *Data Mining Knowledge Discovery*, vol. 18, pp. 392–418, 2009.

[76] C. Garcia-Osorio, A. De Haro-Garcia, and N. Garcia-Pedrajas, "Democratic instance selection : A linear complexity instance selection algorithm based on classifier ensemble concepts," *Artificial Intelligence*, vol. 174, pp. 410–441, 2010.

[77] A. De Haro-Garcia, N. Garcia-Pedrajas, and J. A. Romero del Castillo, "Large scale instance selection by means of federal instance selection," *Data and Knowledge Engineering*, vol. 75, pp. 58–77, 2012.

[78] J.R. Cano, F. Herrera, and M. Lozano, "Stratification for scaling up evolutionary prototype selection," *Pattern Recognition Letters*, vol. 26, pp. 953–963, 2005.

[79] L. Rokach, *Data mining and knowledge discovery handbook*, chapter 45, pp. 957–980, 2005.

[80] T.K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans Pattern Analysis and Machine Intelligence*, vol. 20, pp. 832–844, 1998.

[81] D. Hernandez-Lobato, *Prediction Based on Averages over Automatically Induced Learners : Ensemble Methods and Bayesian Techniques*, Ph.D. thesis, Escuela Politecnica Superior Computer Science Department,, 2009.

[82] L. Breiman, "Arcing classifiers," *The Annals of Statistics*, vol. 26(3), pp. 801–824, 1998.

[83] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, Chapman and Hall/CRC., 1994.

[84] R. E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee, "Boosting the margin : a new explanation for the effectiveness of voting methods," in *Machine Learning : Proceedings of the Fourteenth International Conference*, 1997.

[85] P. Buhlmann, *Recent Advances and Trends in Nonparametric Statistics*, chapter Bagging, subagging and bragging for improving some prediction algorithms., pp. 19–34, 2003.

[86] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms : Bagging, boosting, and variants," *Machine Learning*, vol. 36(1-2), pp. 105–139, 1999.

[87] G. I. Webb, "Multiboosting : A technique for combining boosting and wagging," *Machine Learning*, vol. 40(2), pp. 159–196, 2000.

[88] M. Dettling, "Bagboosting for tumor classification with gene expression data," *Bioinformatics,*, vol. 20(18), pp. 3583–3593, 2004.

[89] K. Tumer and N. C. Oza, "Input decimated ensembles," *Pattern Analysis and Applications*, vol. 6(1), pp. 65–77, 2003.

[90] M. Skurichina and R. P. W. Duin, "Combining feature subsets in feature selection.," in *Proceedings of the 6th International Workshop on Multiple Classifier Systems*, 2005.

[91] 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*, vol. 28(10), pp. 1619–1630, 2006.

[92] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes.," *Journal of Artificial Intelligence Research*, vol. 2, pp. 263–286, 1995.

[93] L. Breiman, "Randomizing outputs to increase prediction accuracy," *Machine Learning*, vol. 40(3), pp. 229–242, 2000.

[94] G. Martinez-Munoz and A. Suarez, "Switching class labels to generate classification ensembles.," *Pattern Recognition*, vol. 38(10), pp. 1483–1494, 2005.

[95] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5–32, 2001.

[96] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, 2000.

[97] Yoav Freund and Robert E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)*, Lorenza Saitta, Ed. 1996, pp. 148–156, Morgan Kaufmann.

[98] D.H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[99] J. Gama and P. Brazdil, "Cascade generalization," *Machine Learning*, vol. 41(3), pp. 315–343, 2000.

[100] Terziyan V. Y. Puuronen, S. and A. Tsymbal, "A dynamic integration algorithm for an ensemble of classifiers.," in *Proceedings of the 11th International Symposium on Foundations of Intelligent Systems*, 1999.

[101] A. Tsymbal and S. Puuronen, *Principles of Data Mining and Knowledge Discovery*, chapter Bagging and boosting with dynamic integration of classifiers., pp. 116–125., Springer Berlin Heidelberg, 2000.

[102] V. PISETTA, *New Insights into Decision Trees Ensembles*, Ph.D. thesis, Laboratoire ERIC, Universite Lumiere Lyon 2, 2012.

[103] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996.

[104] M. Skurichina and R. P. W. Duin, "Bagging, boosting and the rsm for linear classifiers," *Pattern Analysis and Applications*, vol. 5, pp. 121–135, 2002.

[105] S. Raheel, *L'Apprentissage Artificiel pour la Fouille de Donnees Multilingues : Application a la Classification Automatique des Documents Arabes*, Ph.D. thesis, Universite Lumiere Lyon 2, 2010.

[106] M. Skurichina and R. P. W. Duin, "Bagging, boosting and the random subspace method for linear classi-ers," *Pattern Analysis and Applications*, vol. 5, pp. 121–135, 2002.

[107] Y. Amit and D. Geman, "Shape quantization and recognition with randomized trees," *Neural Computation*, vol. 9, no. 7, pp. 1545–1588, 1997.

[108] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 36(1), pp. 3–42, 2006.

[109] C. Chen, A. A. Liaw, and L. L. Breiman, "Using random forest to learn imbalanced data," Tech. Rep., Department of Statistics, University of California, Berkeley, 2004.

[110] E Tuv, A. Borisov, G. Runger, and K. Torkkola, "Feature selection with ensembles, artificial variables, and redundancy elimination," *Journal of Machine Learning Research*, vol. 10, pp. 1341–1366, 2009.

[111] O. DeMasi, J. Mezay, and D. H. Baileyz, "Dimension reduction using rule ensemble machine learning methods : A numerical study of three ensemble methods," Tech. Rep., Lawrence Berkeley National Laboratory, 2011.

[112] M. Embrechts, J. M. Santos, and J. D. Linton, "Random brains : An ensemble method for feature selection with neural networks," *ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium)*, 2013.

[113] R.E. Schapire and F. Freund, *Boosting : Foundations and Algorithms*, The MIT Press, 2012.

[114] X. Zhu and X. Wu, "Class noise vs. attribute noise : A quantitative study of their impacts," *Artificial Intelligence Review*, vol. 22, pp. 177–210, 2004.

[115] Seyed Hamid Rezatofighi, Kosar Khaksari, and Hamid Soltanian-Zadeh, "Automatic recognition of five types of white blood cells in peripheral blood," in *Proceedings of the 7th International Conference on Image Analysis and Recognition - Volume Part II*, Berlin, Heidelberg, 2010, ICIAR'10, pp. 161–172, Springer-Verlag.

[116] S. Ben Chaabane, M. Sayadi, F. Fnaiech, and E. Brassart, "Dempster-shafer evidence theory for image segmentation : Application in cells images," vol. 3, no. 11, pp. 590 – 596, 2009.

[117] Salim Ben Chaabane, Mounir Sayadi, Farhat Fnaiech, and Eric Brassart, "Colour image segmentation using homogeneity method and data fusion techniques," *EURASIP J. Adv. Signal Process*, vol. 2010, pp. 1 :1–1 :11, Jan. 2010.

[118] P.S Hiremath, Parashuram Bannigidad, and Sai Geeta, "Automated identification and classification of white blood cells (leukocytes) in digital microscopic images," *IJCA,Special Issue on RTIPPR*, , no. 2, pp. 59–63, 2010, Published By Foundation of Computer Science.

[119] Qingmin Liao and Yingying Deng, "An accurate segmentation method for white blood cell images.," in *ISBI*. 2002, pp. 245–248, IEEE.

[120] B.C. Ko, J-W Gim, and J-Y Nam, "Automatic white blood cell segmentation using stepwise merging rules and gradient vector flow snake," *Micron*, vol. 42, pp. 695–705, 2011.

[121] M. Benazzouz, I. Baghli, and A. Chikh, "Microscopic image segmentation based on pixel classification and dimensionality reduction.," *Int. J. Imaging Systems and Technology*, vol. 23, no. 1, pp. 22–28, 2013.

[122] Benomar Mohammed Lamine, Mourtada Benazzouz, and Med Amine Chikh, "Segmentation d-images microscopiques basée sur les attributs textures," in *ICA2IT International Conference on Artificial Intelligence and Information Technology. Ouargla, Algeria, March 10-12, 2014.*

[123] James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes, *Computer Graphics : Principles and Practice (2Nd Ed.)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

Annexe

Résumé

Le développement des modèles de classification est l'une des principales tâches dans le domaine de data mining. Toutefois, le volume élevé de données générées par différents domaines de recherche, allant du séquençage du génome humain, qui permet d'obtenir des niveaux d'expressions de plusieurs milliers de gènes, aux millions d'informations circulant sur internet rend l'utilisation des méthodes d'apprentissage automatique un vrai défi. D'où la nécessité d'une étape de prétraitement afin de préparer la base aux algorithmes d'apprentissage.

L'induction d'un modèle de classification pour le diagnostic avec autant d'instances et de variables est un défi majeur dans le domaine de l'apprentissage automatique. D'où la nécessité de réduire ce nombre. Parmi les processus de prétraitements applicables sur une base, nous trouvons les méthodes de réduction : les algorithmes de sélection d'instances et de variables.

Le sujet de cette thèse est orienté vers la recherche de méthodes efficaces de traitements des données médicales et biologiques. Nous nous sommes principalement intéressés à l'application d'une méthode de sélection d'instances pour nettoyer et réduire la base d'apprentissage avant la conception du classifieur.

Au cours de nos recherches, nous avons pu étudier les différentes approches existantes ainsi que leur avantages et limitations. Nous nous sommes intéressés aux méthodes ensemblistes afin de pallier les problèmes rencontrés par les méthodes de sélection classiques. Les méthodes ensemblistes sont un ensemble d'algorithmes qui s'inspire du principe « *l'union fait la force* », en effet ces méthodes combinent les décisions individuelles de plusieurs algorithmes de classification faibles afin d'améliorer leurs performances pour classer de nouveaux exemples.

Donc si on décide «*à la majorité*», alors on se trompe si et seulement si plus de la moitié du «*comité*» se trompe. En effet, la décision prise en groupe ne peut être fausse que si la majorité du groupe se trompe. Ceci rend les individus sur lesquels les classifieurs sont le plus en désaccord, les plus intéressants à traiter au cours de la sélection d'instance.

Un algorithme de sélection d'instances basé sur les algorithmes ensemblistes et notamment sur l'algorithme Forêt aléatoire a été implémenté. Nous avons testé notre proposition sur plusieurs problèmes de classification de UCI Machine Learning Repository ainsi que sur une base d'image cytologique afin d'optimiser la segmentation automatique de globules blancs. Les résultats obtenus démontrent que notre proposition est aussi performante que les méthodes existantes tout en étant moins coûteuse.

Mots clés
Sélection d'instances, méthodes ensembliste, forêt aléatoire, marge ensembliste, reconnaissance automatique des globules blancs, images cytologique

Abstract

Classification algorithm development is one of the main processes in data mining. However, the high volume of data generated by different areas of research, from the sequencing of the human genome, which provides levels of expression of several thousand genes, to millions of information circulating on the Internet makes the use of machine learning methods a real challenge. Hence there is a need for a pretreatment step to prepare the basis for learning algorithms.

The induction of a classification model for diagnosis with this huge amount of instances and variables is a major challenge in the field of statistical learning. So, it is necessary to reduce this number. Among the pretreatment processes applicable on a base, we find the methods of reduction : the algorithms of instance selection.

This thesis is devoted to the search for efficient methods of medical and biological data processing. We were mainly interested in applying an instance selection method to clean up and reduce the learning dataset before building the classifier. We were able to study the different existing approaches as well as their advantages and limitations. We are interested in the set-up methods in order to overcome the problems encountered by classical selection methods.
Ensemble methods are a set of algorithms inspired by the precept "union is strength", because these methods combine the individual decisions of several weak classification algorithms to improve their performance to classify new examples. So, if we decide "by majority", then we are wrong if and only if more than half of the "committee" is wrong.

Indeed, the fact that the group decision can only be false if the majority of the group is mistaken makes the individuals on which the classifiers are most at odds are those who are the most interesting to deal with in the selection process. A selection algorithm based on the ensemble algorithms and in particular the Random Forest algorithm has been implemented. We have tested our method on several UCI Machine Learning Repository classification issues as well as on a cytological image base in order to optimize the automatic segmentation of white blood cells. The results obtained demonstrate that our approach is as efficient as the existing methods while being less expensive.

Keywords

Instances selection, Ensemble methods, Random Forest, Ensemble margin, automatic recognition of white blood cells, cytological images.

ملخص

يعد تطوير نماذج التصنيف من المهام الرئيسية في مجال استخراج البيانات. ومع ذلك، فإن ارتفاع حجم البيانات التي تم إنشاؤها من قبل مجالات مختلفة من البحوث، بدءا من تسلسل الجينوم البشري، الذي يوفر مستويات التعبير عن عدة آلاف من الجينات، للملايين من المعلومات المتداولة على شبكة الإنترنت يجعل استخدام وطرق التعلم التلقائي تحديا حقيقيا. وبالتالي، هناك حاجة إلى خطوة المعالجة المسبقة لإعداد الأساس لخوارزميات التعلم. إن بناء نموذج تصنيف للتشخيص مع العديد من الحالات والمتغيرات يمثل تحديا كبيرا في مجال التعلم الآلي. وبالتالي الحاجة إلى تقليل هذا العدد. من بين عمليات المعالجة التي تنطبق على قاعدة، نجد أساليب الحد: خوارزميات اختيار الحالات والمتغيرات

موضوع هذه الدكتوراه موجه نحو البحث عن طرق فعالة لمعالجة البيانات الطبية والبيولوجية. كنا مهتمين أساسا في تطبيق طريقة اختيار سبيل المثال لتنظيف وتقليل قاعدة التعلم قبل تصميم المصنف

خلال بحثنا، كنا قادرين على دراسة مختلف النهج القائمة وكذلك مزاياها والقيود. كنا مهتمين في انشاء طرق من أجل التغلب على المشاكل التي تواجهها طرق الاختيار الكلاسيكية. الأساليب التعاونية هي مجموعة من الخوارزميات مستوحاة من مبدأ "الوحدة يجعل قوة" في الواقع هذه الطرق تجمع بين القرارات الفردية من عدة خوارزميات تصنيف ضعيفة لتحسين أدائها لتصنيف أمثلة جديدة. في الواقع، فإن قرار المجموعة لا يمكن أن يكون إلا خطأ إذا كانت أغلبية المجموعة مخطئة. وهذا يجعل الأفراد الذين يختلفون المصنفون أكثر، الأكثر إثارة للاهتمام للتعامل مع أثناء اختيار المثال

وقد تم تنفيذ خوارزمية اختيار الحالات استنادا إلى خوارزميات التعاونية وعلى وجه الخصوص على خوارزمية الغابات العشوائية. اختبرنا اقتراحنا على العديد من المشاكل تصنيف الاتحاد الدولي للدراجات آلة التعلم مستودع وعلى قاعدة صورة الخلوي لتحسين تجزئة التلقائي من خلايا الدم البيضاء. وتدل النتائج التي تم الحصول عليها على أن مقترحنا يتسم بالكفاءة التي تتسم بها الطرق القائمة في حين أنها أقل تكلفة

الكلمات الرئيسية

اختيار الحالات، تعيين الأساليب، والغابات عشوائية، تعيين الهامش، والاعتراف التلقائي من خلايا الدم البيضاء، والصور الخلوية

Résumé

Le développement des modèles de classification est l'une des principales tâches dans le domaine de data mining. Toutefois, le volume élevé de données générées par différents domaines de recherche, allant du séquençage du génome humain, qui permet d'obtenir des niveaux d'expressions de plusieurs milliers de gènes, aux millions d'informations circulant sur internet rend l'utilisation des méthodes d'apprentissage automatique un vrai défi. D'où la nécessité d'une étape de prétraitement afin de préparer la base aux algorithmes d'apprentissage. L'induction d'un modèle de classification pour le diagnostic avec autant d'instances et de variables est un défi majeur dans le domaine de l'apprentissage automatique. D'où la nécessité de réduire ce nombre. Parmi les processus de prétraitements applicables sur une base, nous trouvons les méthodes de réduction : les algorithmes de sélection d'instances et de variables.

Le sujet de cette thèse est orienté vers la recherche de méthodes efficaces de traitements des données médicales et biologiques. Nous nous sommes principalement intéressés à l'application d'une méthode de sélection d'instances pour nettoyer et réduire la base d'apprentissage avant la conception du classifieur. Au cours de nos recherches, nous avons pu étudier les différentes approches existantes ainsi que leur avantages et limitations. Nous nous sommes intéressés aux méthodes ensemblistes afin de pallier les problèmes rencontrés par les méthodes de sélection classiques. Les méthodes ensemblistes sont un ensemble d'algorithmes qui s'inspire du principe « l'union fait la force », en effet ces méthodes combinent les décisions individuelles de plusieurs algorithmes de classification faibles afin d'améliorer leurs performances pour classer de nouveaux exemples.

Donc si on décide «à la majorité», alors on se trompe si et seulement si plus de la moitié du «comité» se trompe. En effet, la décision prise en groupe ne peut être fausse que si la majorité du groupe se trompe. Ceci rend les individus sur lesquels les classifieurs sont le plus en désaccord, les plus intéressants à traiter au cours de la sélection d'instance. Un algorithme de sélection d'instances basé sur les algorithmes ensemblistes et notamment sur l'algorithme Forêt aléatoire a été implémenté. Nous avons testé notre proposition sur plusieurs problèmes de classification de UCI Machine Learning Repository ainsi que sur une base d'image cytologique afin d'optimiser la segmentation automatique de globules blancs. Les résultats obtenus démontrent que notre proposition est aussi performante que les méthodes existantes tout en étant moins coûteuse.

Mots clés : Sélection d'instances, méthodes ensembliste, forêt aléatoire, marge ensembliste, reconnaissance automatique des globules blancs, images cytologique

Abstract

Classification algorithm development is one of the main processes in data mining. However, the high volume of data generated by different areas of research, from the sequencing of the human genome, which provides levels of expression of several thousand genes, to millions of information circulating on the Internet makes the use of machine learning methods a real challenge. Hence there is a need for a pretreatment step to prepare the basis for learning algorithms.

The induction of a classification model for diagnosis with this huge amount of instances and variables is a major challenge in the field of statistical learning. So, it is necessary to reduce this number. Among the pretreatment processes applicable on a base, we find the methods of reduction : the algorithms of instance selection. This thesis is devoted to the search for efficient methods of medical and biological data processing. We were mainly interested in applying an instance selection method to clean up and reduce the learning dataset before building the classifier. We were able to study the different existing approaches as well as their advantages and limitations. We are interested in the set-up methods in order to overcome the problems encountered by classical selection methods. Ensemble methods are a set of algorithms inspired by the precept "union is strength", because these methods combine the individual decisions of several weak classification algorithms to improve their performance to classify new examples. So, if we decide "by majority", then we are wrong if and only if more than half of the "committee" is wrong. Indeed, the fact that the group decision can only be false if the majority of the group is mistaken makes the individuals on which the classifiers are most at odds are those who are the most interesting to deal with in the selection process. A selection algorithm based on the ensemble algorithms and in particular the Random Forest algorithm has been implemented. We have tested our method on several UCI Machine Learning Repository classification issues as well as on a cytological image base in order to optimize the automatic segmentation of white blood cells. The results obtained demonstrate that our approach is as efficient as the existing methods while being less expensive.

Keywords : Instances selection, Ensemble methods, Random Forest, Ensemble margin, automatic recognition of white blood cells, cytological images.

ملخص

يعد تطوير نماذج التصنيف من المهام الرئيسية في مجال استخراج البيانات. ومع ذلك، فإن ارتفاع حجم البيانات التي تم إنشاؤها من قبل مجالات مختلفة من البحوث، بدءا من تسلسل الجينوم البشري، الذي يوفر مستويات التعبير عن عدة آلاف من الجينات، للملايين من المعلومات المتداولة على شبكة الإنترنت يجعل استخدام وطرق التعلم التلقائي تحديا حقيقيا. وبالتالي، هناك حاجة إلى خطوة المعالجة المسبقة لإعداد الأساس لخوارزميات التعلم. إن بناء نموذج تصنيف للتشخيص مع العديد من الحالات والمتغيرات يمثل تحديا كبيرا في مجال التعلم الآلي. وبالتالي الحاجة موضوع هذه الدكتوراه موجه نحو البحث عن طرق فعالة إلى تقليل هذا العدد. من بين عمليات المعالجة التي تنطبق على قاعدة، نجد أساليب الحد: خوارزميات اختيار الحالات والمتغيرات خلال بحثنا، كنا قادرين على دراسة مختلف النهج لمعالجة البيانات الطبية والبيولوجية. كنا مهتمين أساسا في تطبيق طريقة اختيار سبيل المثال لتنظيف وتقليل قاعدة التعلم قبل تصميم المصنف القائمة وكذلك مزاياها والقيود. كنا مهتمين في انشاء طرق من أجل التغلب على المشاكل التي تواجهها طرق الاختيار الكلاسيكية. الأساليب التعاونية هي مجموعة من الخوارزميات مستوحاة من مبدأ "الوحدة يجعل قوة" في الواقع هذه الطرق تجمع بين القرارات الفردية من عدة خوارزميات تصنيف ضعيفة لتحسين أدائها لتصنيف أمثلة جديدة. في الواقع، فإن قرار المجموعة لا يمكن أن يكون إلا خطأ إذا كانت أغلبية المجموعة مخطئة. وهذا يجعل الأفراد الذين يختلفون المصنفون أكثر، الأكثر إثارة للاهتمام للتعامل مع أثناء اختيار المثال

وقد تم تنفيذ خوارزمية اختيار الحالات استنادا إلى خوارزميات التعاونية وعلى وجه الخصوص على خوارزمية الغابات العشوائية. اختبرنا اقتراحنا على العديد من المشاكل تصنيف الاتحاد الدولي للدراجات آلة التعلم مستودع وعلى قاعدة صورة الخلوي لتحسين تجزئة التلقائي من خلايا الدم البيضاء. وتدل النتائج التي تم الحصول عليها على أن مقترحنا يتسم بالكفاءة التي تتسم بها الطرق القائمة في حين أنها أقل تكلفة

الكلمات الرئيسية

اختيار الحالات، تعيين الأساليب، والغابات عشوائية، تعيين الهامش، والاعتراف التلقائي من خلايا الدم البيضاء، والصور الخلوية