Chapitre II

Classification associative adaptative et déséquilibre

Bahri E., Lallich S. (2010), Proposition d'une méthode de classification associative adaptative, *10e Conférence Extraction et Gestion des Connaissances, EGC 2010*, pp.501-512, Hammamet, Tunisie.

Bahri E., Lallich S. (2009), Improving prediction by weighting class association rules, *ICMLA'09, The Eighth International Conference on Machine Learning and Applications*, (eds. Wani M. A., Kantardzic M., Palade V., Kurgan L., Qi Y.), IEEE Press, pp. 765-770, Miami, USA.

Bahri E., Lallich S. (2009), Pruning for Extracting Class Association Rules without Candidate, *Proceedings of the 2009 International Conference on Data Mining, DMIN'09*, R. Stahlbock, S. F. Crone, and S. Lessmann (Eds.), CSREA Press, pp. 11-17, Las Vegas, Nevada, USA.

Bahri E., Lallich S. (2009), FCP-Growth to extract itemsets for classe association rules, *poster 22nd International Conference Florida Artificial Intelligence Research Society, FLAIRS 09*, Sanibel Island, Florida, USA, May 2009. E. Bahri, Lallich S (2009), FCP-Growth, une adaptation de FP-Growth pour générer des règles d'association de classe, *poster 9ème Conférence Extraction et Gestion des Connaissances, EGC 09*, Strasbourg, Janvier 2009

Bahri E., Lallich S. (2009), Introduction de l'élagage pour l'extraction de règles d'association de classe sans génération de candidats, *Actes atelier QDC, Qualité des Données et des Connaissances*, A6, pp. 29-36, en association avec la 9e conférence EGC, Strasbourg.

Bahri E., Lallich S. (2009), Pour une classification associative plus efficace, *poster 9ème Conférence francophone sur l'apprentissage artificiel, CAP 09*, Hammamet, Tunisie, Mai 2009

Sommaire

1. Introduction

Etant donné un ensemble d'exemples décrits suivant différentes caractéristiques et dont l'étiquette, ou classe d'appartenance, est connue (échantillon d'apprentissage noté LS), les méthodes d'apprentissage supervisé se proposent d'élaborer un procédé permettant de prédire l'étiquette d'un nouvel exemple d'étiquette inconnue à partir de ses caractéristiques ! La qualité de cette prédiction est évaluée à partir des exemples d'un échantillon test (TS) indépendant de l'échantillon d'apprentissage, éventuellement par validation croisée.

Dans la mesure où ces méthodes sont destinées à être appliquées à des données réelles et pas seulement à des *benchmarks*, leur enjeu est non seulement de classifier correctement de nouveaux exemples mais aussi d'assurer l'interprétabilité des résultats obtenus. Dans de nombreux domaines d'application (par exemple la banque ou la médecine), il faut pouvoir expliquer la décision prise, ce qui nous amènera à privilégier les méthodes procédant par induction de règles. En effet, parmi les méthodes d'apprentissage supervisé, les méthodes d'induction de règles, par exemple les arbres de décision (voir [Zighed and Rakotomalala, 2000] pour une présentation d'ensemble), ont une place particulière. Elles ne sont pas forcément les plus efficaces, mais elle ont l'immense avantage de fournir une justification intelligible de la décision qui est prise.

Parmi les problèmes que posent les données réelles, nous avons retenu dans cette thèse :

(1) **la qualité des données à classifier.** En effet, les bases de données peuvent être bruitées, en raison notamment du mode d'acquisition des données, souvent automatique et/ou passant par le web. Les bases de données réelles peuvent donc contenir des exemples erronés, contradictoires ou redondants. Cet aspect des données réelles a été traité dans le premier chapitre.

(2) **le déséquilibre des classes.** Dans les problèmes réels, le plus souvent les classes sont déséquilibrées et c'est la classe la moins nombreuse qui est la classe d'intérêt. La plupart des algorithmes pénalisent d'autant plus la classe minoritaire que sa fréquence est faible. Parmi les méthodes à base de règles, c'est en particulier le cas des arbres de décision, à chaque étape de la méthode, fonction d'éclatement, règle de prédiction à partir des feuilles de l'arbre, élagage. C'est sur cette caractéristique des données réelles que porte le présent chapitre.

Notre objectif est de proposer une nouvelle méthode de classification par induction de règles qui tout à la fois améliore les performances des méthodes usuelles d'apprentissage de règles, notamment en présence de données déséquilibrées, et conserve l'intelligibilité de ces méthodes.

Parmi les méthodes d'induction de règles, nous proposons de nous intéresser à la classification associative. Celle-ci utilise pour la prédiction des règles d'association particulières, appelées règles d'association de classe, ou règles d'association prédictives. Ce sont des règles dont le conséquent est l'une des modalités de classe. L'intérêt des règles de classe est de permettre la focalisation sur des groupes d'individus, éventuellement très petits, homogènes du point de vue des descripteurs et présentant la même classe. Par rapport aux arbres de décision, la classification associative a l'intérêt de ne pas pratiquer de stratégie gloutonne, ce qui permet d'explorer l'ensemble des règles construites. Ce type d'approche est bien adapté à notre étude puisqu'il permet de repérer des règles de classe correspondant aux classes de faible effectif sans être prisonnier d'une approche gloutonne, tout en respectant l'objectif d'interprétabilité des résultats.

Une proposition possible pour faire face aux exigences posées dans ce chapitre, notamment celle de déséquilibre des classes, est de nous inspirer des approches ensemblistes, du *Boosting* tout

particulièrement. En effet, selon [Weiss, 2004], le caractère adaptatif du *Boosting* prend en compte le déséquilibre des classes et permet d'améliorer le pouvoir prédictif d'un apprenant faible pour la classe minoritaire.

Ces deux propositions sont complémentaires. L'idée de ce chapitre est de proposer un apprenant à base de règles qui extrait de façon efficace, à la fois du point de vue du temps de réponse et de l'espace de stockage, des règles pertinentes aussi bien pour la classe majoritaire que la classe minoritaire. La performance de cet apprenant, qui doit être économe et rapide, sera ensuite améliorée grâce à une procédure adaptative inspirée du *Boosting* qui préserve l'intelligibilité du résultat final.

Dans ce chapitre, nous proposons tout d'abord, W-CARP, une méthode de classification associative qui a pour but de donner des résultats au moins équivalents à ceux des approches existantes, pour un temps d'exécution beaucoup plus réduit, dans la mesure où nous souhaitons par la suite appliquer cette méthode de façon itérative. W-CARP comporte deux phases, la première permet d'extraire les itemsets de classe (ceux dont l'un des items est une modalité de classe) fréquents et de construire les règles de classe, la seconde assure la prédiction à partir des règles de classe issues de la première phase.

Pour accomplir la première phase, nous proposons d'abord FCP-Growth, un algorithme d'extraction qui adapte FP-Growth aux exigences de la classification associative en se limitant à l'extraction des seuls itemsets fréquents qui comportent un item de classe, et qui utilise un support adapté à chaque classe afin de ne pas désavantager les classes minoritaires. Ensuite, nous proposons FCP-Growth-P, une amélioration de FCP-Growth, qui incorpore la condition d'élagage de Li [Li and Zhang, 2003] dans FCP-Growth, pour assurer une meilleure qualité des règles produites.

A partir de ces deux contributions, nous élaborons W-CARP, une méthode de classification associative qui utilise FCP-Growth-P pour accomplir la première phase d'extraction de règles. Pour réaliser la seconde phase, W-CARP construit une base de règles significatives rassemblant les règles significatives. Il pondère les prédictions des différentes règles qui couvrent un exemple par la valeur de leur mesure de Loevinger, qui a l'avantage de tenir compte de la fréquence du conséquent. Finalement, nous proposons CARBoost une procédure de classification associative qui utilise adaptativement la base de règles significatives produite par W-CARP pour se concentrer tout à la fois sur les cas difficiles à prédire et sur les règles qui prédisent correctement au moins un exemple difficile à prédire.

Ce chapitre sera structuré comme suit. La section 2 présente le contexte des classes déséquilibrées. Notre problématique et nos motivations sont exposées en section 3. La section 4 introduit le cadre général de l'agrégation de règles, de la classification associative et de ses méthodes. Nos différentes contributions sont ensuite détaillées, accompagnées des expériences correspondantes, FCP-Growth, FCP-Growth-P et W-CARP d'abord (section 5), puis CARBoost (section 6). La section 7 établit un cadre théorique de notre approche CARBoost. Finalement, nous concluons en section 8.

2. Contexte : le déséquilibre des classes

2.1. Présentation des données déséquilibrées

Les données sont dites déséquilibrées lorsqu'au moins une classe est sous-représentée par rapport aux autres. La classe d'intérêt est généralement la classe minoritaire, car dans la réalité les événements rares sont souvent ceux qui ont le plus d'intérêt. Le problème des données déséquilibrées est ainsi lié à celui des coûts de mauvaise classification asymétriques suivant la classe. Le déséquilibre des données

peut se manifester dans des problème du monde réel tels que la détection de fraudes ou d'intrusions, le diagnostic médical, le suivi médical, la bioinformatique, la catégorisation de textes etc... En outre, la distribution des données de test peut être différente de celle de l'échantillon d'apprentissage et les performances de la classification peuvent être inconnues lors d'apprentissage.

Si la question du déséquilibre des données a été soulevée dans de nombreux travaux parmi lesquels on citera ceux issus de deux ateliers consacrés à ce sujet [Japkowicz, 2000], [Chawla and Kolcz, 2003], ainsi que d'un numéro spécial de SIGKDD [Chawla and Kolcz, 2004], bon nombre de problèmes restent encore ouverts, le plus souvent rencontrés dans le traitement des données réelles, en particulier pour des volumes massifs de données. Afin de bien cerner le problème du déséquilibre des classes, nous avons cherché tout d'abord à comprendre quand et pourquoi les données déséquilibrées peuvent être considérées comme problématiques pour les méthodes d'apprentissage automatique.

Selon l'étude de [Provost, 2000], un classifieur "standard" ne peut pas être utilisé avec satisfaction sur des données déséquilibrées, puisqu'il va chercher avant tout à minimiser l'erreur de prédiction, et aura donc tendance à catégoriser les nouvelles instances comme appartenant à la classe majoritaire. Prenons l'exemple d'un ensemble de 100 individus, dont 90 appartiennent à la classe majoritaire et 10 à la classe minoritaire. Prédire systématiquement la classe majoritaire donne un taux d'erreur de 10% seulement. Du point de vue "taux de bonne prédiction", le résultat peut paraître satisfaisant alors que la performance du classifieur sur les exemples de la classe majoritaire compte neuf fois plus que sa performance sur les classes minoritaires. De ce fait, l'utilisation de classifieurs construits par les algorithmes d'apprentissage automatique standard sans ajuster le seuil de classification peut être considérée comme une erreur grave. Par exemple, l'étude empirique de [Weiss and Provost, 2001] montre que sur vingt-six jeux de données, le taux d'erreur des règles de classification pour la classe minoritaire est en moyenne 2 à 3 fois plus élevé que celui de la classe majoritaire. Si on utilise la précision et le rappel comme critères de mesure, on observe que sur des bases de données déséquilibrées, la précision et le rappel de la classe minoritaire sont très faibles et peuvent même être non définis (précision) ou nuls (rappel), comme nous le verrons dans le chapitre 3.

Différents auteurs ont proposé des méthodes d'apprentissage qui prennent en compte le déséquilibre des classes, pour une vision d'ensemble desquelles nous renvoyons à [Weiss, 2004], [Visa,] et [Japkowicz and Stephen, 2002]. Ces méthodes sont détaillées dans la section suivante.

2.2. Méthodes pour les données déséquilibrées

Parmi les méthodes proposées pour faire face au déséquilibre des classes, nous pouvons distinguer plusieurs grandes familles suivant le niveau auquel agissent ces méthodes :

- au niveau des données grâce au rééquilibrage du jeu de données initial avant l'apprentissage par ré-échantillonnage
- au niveau algorithmique, en modifiant les algorithmes pour les rendre moins sensibles au déséquilibre
- au niveau de l'orientation de la recherche qui se concentre sur l'apprentissage direct de la classe d'intérêt (*one class learning*)
- au niveau de l'évaluation en recourant à de nouvelles métriques adaptées aux données déséquilibrées

Rééquilibrage des données par ré-échantillonnage

Les techniques de ré-échantillonnage sont parmi les plus simples et les plus intuitives. Le principal objectif de ces méthodes consiste à rétablir l'équilibre entre les effectifs associés à chaque classe, soit en diminuant le nombre d'exemples de la classe majoritaire (sous-échantillonnage), soit en augmentant artificiellement le nombre d'exemples de la classe minoritaire (sur-échantillonnage). Dans le cas du sous-échantillonnage, la solution de base consiste à sélectionner aléatoirement parmi les exemples de la classe majoritaire un nombre d'individus égal au nombre d'exemples de la classe minoritaire, puis à employer un classifieur standard sur le nouveau jeu de données ainsi construit. Cette solution a le mérite d'être simple et facile à mettre en oeuvre.

On peut, cependant, s'interroger sur la pertinence d'une sélection aléatoire des exemples de la classe majoritaire. Plusieurs auteurs proposent de sélectionner les exemples de la classe majoritaire de façon plus intelligente. Par exemple, [Kubat and Matwin, 1997]) proposent d'éliminer les individus redondants de la classe majoritaire ainsi que les individus à la frontière de décision entre les classes à l'aide des liens de Tomek [Tomek, 1976]. [Zhang and Mani, 2003] proposent quatre techniques de sous-échantillonnage toutes fondées sur l'algorithme des K plus proches voisins. L'idée est de sélectionner les individus de la classe majoritaire dont la distance moyenne aux K plus proches voisins de la classe minoritaire est la plus faible.

Une alternative au sous-échantillonnage est le sur-échantillonnage où la solution de base est la version symétrique du sous-échantillonnage aléatoire. Ainsi, on réplique aléatoirement des individus de la classe minoritaire que l'on ajoute à l'ensemble d'apprentissage initial jusqu'à obtenir un nombre d'exemples identique pour les deux classes. Certains individus de la classe minoritaire se retrouvent donc plusieurs fois dans l'échantillon équilibré, d'autant plus que le déséquilibre est important. La pertinence de la réplication aléatoire peut être ici aussi remise en cause. En effet, cette solution court le risque de forcer le classifieur à apprendre sur des zones très spécifiques de l'espace de représentation, et introduit par conséquent un fort biais d'apprentissage. Pour diminuer ce biais, [Chawla et al., 2002] proposent de contourner le problème via la génération d'individus synthétiques ayant une représentation différente des individus de la classe minoritaire déjà présents. L'algorithme, nommé SMOTE, se déroule comme suit : on cherche les K plus proches voisins de chaque individu de la classe minoritaire. Les individus synthétiques sont construits en ayant une représentation correspondant à une localisation au hasard sur la ligne reliant le point de classe minoritaire concerné et un de ses K plus proches voisins. L'apprentissage utilise ensuite un classifieur standard sur le nouvel échantillon ainsi constitué. [Lebbah and Bennani, 2010] proposent une méthode de sous-échantillonnage adaptatif qui réalise un sous-échantillonnage des données majoritaires guidé par les données minoritaires lors d'un apprentissage semi-supervisé fondé sur les cartes auto-organisatrices.

Modification des algorithmes

Au niveau algorithmique, différents types d'action sont possibles. Les méthodes qui prennent en compte les coûts sont séduisantes, mais elles nécessitent en théorie de connaître la matrice de coûts, ce qui est assez rare. Par exemple, [Domingos, 1999] propose METACOST, une méthode générale qui permet d'introduire les coûts de mauvaise classification dans un algorithme d'apprentissage supervisé en ré-étiquetant chaque individu par la classe qui permet de minimiser le coût final grâce à une approche *bootstrap* pour ensuite construire le modèle final sur l'échantillon ainsi ré-étiqueté. Certains

auteurs suggèrent de prendre des coûts proportionnels aux effectifs de la matrice de confusion, par exemple [Ling and Zhang, 2004] qui proposent une méthode de construction d'arbre fondée sur un critère de coût minimal. On peut se reporter à [Zhou and Liu, 2006] pour une étude du bien-fondé des méthodes opérant sur les coûts et à [Weiss et al., 2007] pour une comparaison des méthodes de coûts et de rééquilibrage.

Une autre action possible pour répondre à la problématique du déséquilibre consiste à modifier les métriques utilisées pour guider l'algorithme dès l'induction d'un apprenant. Dans cette optique, [Marcellin et al., 2006] ont proposé une mesure d'entropie asymétrique tenant compte du déséquilibre des classes pour induire un arbre de classification adapté. Parallèlement, [Lallich et al., 2007] ont élaboré une méthode permettant de décentrer n'importe quelle entropie, notamment l'entropie de Shannon et l'entropie quadratique. L'idée directrice de ces différents auteurs est de modifier une des propriétés classiques des entropies qui est la symétrie. Cette dernière assure que le meilleur éclatement d'une population correspond à une répartition des classes la plus éloignée possible d'une répartition équilibrée, c'est à dire où les classes sont équiréparties. Cette propriété de symétrie doit être abandonnée dans le cas de données fortement déséquilibrées, puisqu'une répartition équilibrée peut être intéressante dans le cas où la répartition initiale des classes est a priori très déséquilibrée. Un autre type de solution a été proposé par [Chen and Breiman, 2004] qui remplacent le critère entropique usuel par un critère prenant en compte la notion de coût.

Il est aussi possible d'intervenir au niveau de la prédiction en adoptant des seuils de décision qui tiennent compte du déséquilibre des classes. Par exemple, dans le cas des arbres, plutôt que d'étiqueter les exemples d'une feuille de l'arbre avec la classe majoritaire dans la feuille, ce qui est très pénalisant pour la classe minoritaire, [Ritschard, 2005] fondent l'attribution de l'étiquette sur le calcul de l'indice d'implication, l'étiquette attribuée étant la classe qui maximise l'intensité d'implication. L'avantage est que l'on gagne en rappel de la classe positive, mais on perd en précision. Cette démarche d'étiquetage peut être appliquée à d'autres algorithmes que les arbres, par exemple les graphes de voisinage et les cartes de Kohonen dans leurs versions supervisées.

Enfin, en apprentissage par arbre, [Du et al., 2007] étudient des stratégies de pré-élagage efficaces pour éviter le sur-ajustement lorsque l'on utilise les méthodes fondées sur les coûts en induction par arbre. Dans le cas précis des arbres de décision avec C4.5, [Chawla, 2003] a étudié la qualité des estimations probabilistes, l'élagage et le prétraitement des données, trois problèmes habituellement considérés de façon séparée.

Apprentissage direct de la classe d'intérêt

Des études sur des données déséquilibrées ont montré que lorsqu'on apprend sur l'ensemble des données, la classe minoritaire est largement ignorée. Une solution qui apparaît judicieuse est d'apprendre seulement à partir des règles de classification de la classe minoritaire. Plusieurs méthodes utilisent cette solution, HIPPO [Japkowicz et al., 1995], BRUTE [Riddle et al., 1994], SHRINK [Kubat et al., 1998a] et RIPPER [Cohen, 1995] qui sont bien résumées par [Weiss, 2004].

HIPPO [Japkowicz et al., 1995] a la particularité de n'apprendre qu'à partir des exemples positifs, ceux de la classe minoritaire, à l'aide d'un réseau de neurones. Plutôt que de distinguer les exemples négatifs des exemples positifs, il reconnaît les associations entre attributs, parmi les exemples positifs. Les méthodes qui apprennent seulement la classe rare peuvent aussi apprendre à partir d'exemples

appartenant à toutes les classes, tel est le cas de BRUTE, SHRINK et RIPPER.

BRUTE a été utilisé pour rechercher des failles dans le processus de fabrication de l'entreprise BOEING. Afin de trouver les échecs, c'est-à-dire les cas rares (classe positive), BRUTE se concentre uniquement sur la performance des règles positives, celles qui permettent de prédire des échecs. Comme souligné par [Kubat et al., 1998b], en mesurant seulement les performances des règles positives, BRUTE n'est pas influencée par la précision toujours élevée due aux exemples négatifs qui ne sont pas couverts par les règles positives. SHRINK [Kubat et al., 1998a] utilise une approche similaire pour détecter les déversements d'huile rare à partir des images de radar par satellite. En se fondant sur l'hypothèse qu'il y aura beaucoup plus d'exemples négatifs que d'exemples positifs, SHRINK classifie les régions mixtes, c'est-à-dire les régions avec des exemples positifs et négatifs, avec la classe positive. La tâche consiste donc à rechercher la meilleure "région positive", celle possédant le ratio d'exemples positifs le plus élevé. RIPPER [Cohen, 1995] est une méthode d'induction de règles qui utilise une approche itérative pour construire des règles qui couvrent les exemples d'apprentissage qui ne sont pas couverts à l'itération d'avant. Chaque règle est spécialisée par l'ajout de conditions jusqu'à ce qu'aucun exemple négatif ne soit couvert. Un post-traitement permet de réduire la taille de l'ensemble de règles et d'améliorer sa qualité. RIPPER génère des règles pour chaque classe, de la classe la plus rare à la classe la plus nombreuse, ce qui lui permet de se limiter à l'apprentissage de la classe minoritaire.

Métriques d'évaluation adaptées aux données déséquilibrées

Les métriques "standard" qui ne prennent pas en compte le déséquilibre des classes, comme le taux de bonne prédiction (ou *Accuracy*), attribuent plus de poids aux classes ordinaires qu'aux classes rares, ce qui rend difficile pour un classifieur d'être performant sur ces dernières. Plusieurs auteurs ont proposé des métriques qui dépendent de la fréquence de la classe et donc du déséquilibre des données. Ces métriques peuvent améliorer la fouille des données en orientant mieux le processus de recherche et permettent de mieux évaluer le résultat final de la classification. Parmi ces métriques, on trouve principalement la précision et le rappel, la moyenne géométrique ou la F-mesure et finalement la courbe ROC.

La précision et le rappel sont des métriques issues du domaine de la recherche d'information (*information retrievial*) que nous avons décrites dans le chapitre 1. Des variantes dérivées de ces mesures peuvent être utilisées pour synthétiser la performance d'un algorithme sur des données déséquilibrées telles que la moyenne géométrique des rappels de chaque classe [Kubat et al., 1998b] ou la F-mesure qui dans sa version standard représente la moyenne harmonique de la précision et du rappel [van Rijsbergen, 1979]. Des études comparatives ([Joshi et al., 2001a], [Joshi et al., 2002] et [Estabrooks and Japkowicz, 2001]) ont utilisé la F-mesure pour comparer la performance des différents algorithmes de fouille de données.

La courbe ROC est un critère qui permet de comparer de façon globale et visuelle les performances des algorithmes d'apprentissage sans prendre en compte la fréquence des modalités des classes et les coûts de mauvaise classification. Elle s'applique aux problèmes à deux classes et correspond à un graphique dont l'abscisse indique le taux de faux positifs et l'ordonnée le taux de vrais positifs. La courbe ROC peut être appréciée de façon chiffrée par l'aire sous la courbe ou AUC (*Area Under Curve*), qui indique la probabilité qu'un individu positif soit classé devant un individu négatif. Si l'on classe les individus au hasard, l'AUC est égale à 0.5, ce qui fournit une valeur seuil. Ainsi l'AUC ne met

pas l'accent sur une classe ou sur l'autre, de telle sorte qu'elle n'est pas biaisée en défaveur de la classe minoritaire. Les courbes ROC peuvent également être utilisées pour évaluer différents scénarios : par exemple, le nombre d'exemples positifs bien classés peut être augmenté au détriment de l'introduction de faux positifs supplémentaires. Les analyses à base de la courbe ROC ont été utilisées par de nombreux systèmes conçus pour faire face au déséquilibre de données, comme le système de fouille de données de SHRINK [Kubat et al., 1998a].

2.3. Le *Boosting* face aux données déséquilibrées

Comme nous l'avons expliqué dans le chapitre précédent, le *Boosting*, et principalement son algorithme phare AdaBoost [Freund and Schapire, 1996], est une procédure adaptative de construction d'un ensemble de classifieurs. Le principe fondateur du *Boosting* est de créer à la première itération un classifieur faible (dont la performance est strictement meilleure que le hasard). Les exemples mal classés par ce premier classifieur seront ensuite surpondérés, alors que dans le même temps, les exemples ayant été bien classés seront sous-pondérés (on parle de mise à jour adaptative). Les diverses pondérations permettent de créer un nouvel échantillon d'apprentissage sur lequel on apprend de nouveau un classifieur faible. La procédure est répétée T fois, autrement dit, on se retrouve avec T classifieurs, tous construits sur un échantillon potentiellement différent. Finalement, un vote pondéré des T classifieurs est effectué pour classer un nouvel exemple. L'idée sous-jacente est qu'à chaque itération, le classifieur courant se concentre sur les exemples mal classés à l'itération précédente.

Il est raisonnable de penser que grâce à cette mise à jour adaptative, le *Boosting* tient compte du déséquilibre des classes et améliore la performance de la classification face à ce type de données. En effet, les premières itérations auront tendance à bien classer les exemples issus de la classe majoritaire dans le sens où un classifieur standard sera construit sur l'échantillon d'apprentissage initial. Toutefois, au fur et à mesure des itérations, les exemples des classes minoritaires seront de plus en plus surpondérés, jusqu'à devenir les exemples à bien classer en priorité par le classifieur courant.

Dans la littérature, plusieurs algorithmes de type *Boosting* spécifiquement dédiés au problème du déséquilibre des classes ont été proposés. On citera d'abord Adacost [Fan et al., 1999] qui utilise une matrice de coûts de mauvaise classification pour repondérer les exemples. RareBoost [Joshi et al., 2001b] repondère les exemples en fonction du rappel et de la précision. On trouve également SMOTEBoost qui consiste à utiliser l'algorithme SMOTE [Chawla et al., 2002] avant d'appliquer AdaBoost sur le nouvel échantillon construit par SMOTE.

Dans la section suivante, nous posons les bases de notre approche d'apprentissage dans un contexte de données réelles. Cette approche sera spécifiquement dédiée au problème du déséquilibre des classes. Nous mettons également l'accent sur la lisibilité des résultats, la rapidité d'apprentissage et la performance du système qui constituent des points primordiaux dans tout contexte industriel.

3. Apprentissage de règles

3.1. Présentation de l'apprentissage par induction de règles

Définition

En apprentissage automatique de données réelles, on cherche le plus souvent à obtenir des procédures de classification compréhensibles et interprétables par l'utilisateur humain. Pour faire face à cette exigence, on trouve les méthodes d'apprentissage par induction de règles qui recherchent des règles de classification à partir d'un ensemble d'exemples dont on connaît la classe d'appartenance, ou étiquetés. Elles consistent à découvrir des règles de type *Si Condition, alors Classe* où la condition porte sur les descripteurs ou attributs prédictifs. L'objectif est de rechercher les règles les plus pertinentes et les plus explicatives.

Principales méthodes

Ces méthodes peuvent être des approches de type *separate-and-conquer* telles que les arbres de décision, des inductions de règles, des règles floues ou encore les techniques du type *rough sets*.

Les arbres de décision Les arbres de décision reposent sur la technique consistant à «diviser pour régner». Cette technique se résume à identifier des sous-problèmes, à leur trouver une solution et à combiner ces solutions pour résoudre le problème général. Les arbres de décision apprennent à identifier les sous-espaces de l'espace d'entrée pour lesquels la solution est identique, dans le but de prédire (le plus précisément possible) la classe d'un nouvel objet en se fondant sur l'analyse des autres caractéristiques de cet objet.

La construction d'un arbre de décision repose sur l'analyse d'exemples dont on connaît les descripteurs et la classe. Au moyen d'un algorithme glouton et selon la stratégie "diviser pour régner" on partitionne l'espace de données selon des coupes orthogonales aux attributs prédictifs. Lorsque l'on recherche à chaque noeud l'attribut qui assure la meilleure coupure, ou éclatement, on choisit un critère qui évalue la qualité de la coupure et on retient l'attribut qui maximise la valeur de ce critère.

Les méthodes à base d'arbres de décision les plus importantes sont CART, ID3 et C4.5 :

- **CART**, qui a été développée par des statisticiens [Breiman et al., 1984], construit des arbres de décision binaires. Elle peut être étendue pour traiter le cas d'attributs continus. Le critère de coupure utilisé pour le partionnement est le critère de Gini fondé sur l'entropie quadratique. On calcule l'entropie quadratique avant la coupure puis la moyenne des entropies quadratiques après la coupure qui prend en compte la taille des noeud-fils formés par la coupure. On forme alors le gain d'entropie quadratique, que l'on peut rapporter à la valeur avant coupure, pour former le critère de Gini. L'élagage de l'arbre se fait par estimation de l'erreur réelle en utilisant un ensemble test.
- **ID3** est une méthode développée en 1983, améliorée en 1993 par une nouvelle version appelée C4.5. On ne se restreint pas à des attributs binaires [Quinlan, 1994]. ID3 cherche récursivement le meilleur partitionnement de l'échantillon en utilisant le gain d'entropie de Shannon et non pas le gain d'entropie quadratique. Détaillons le calcul du gain d'entropie de Shannon, dont le principe est identique à celui du calcul du gain d'entropie quadratique. L'entropie de Shannon avant coupure s'écrit de la manière suivante :

$$H(X) = -\sum_{j=1}^{k} P_j \log_2 P_j$$

où X représente l'échantillon au sein du noeud courant, k le nombre de modalités de la variable à prédire Y et P_j la fréquence relative de la modalité j dans le noeud considéré.
Par exemple, si la variable Y possède 2 modalités dont la répartition dans le noeud courant X est de $(0.3; 0.7)$, autrement dit 30 exemples de la classe 1 et 70 exemples de la classe 2. Nous obtenons : $H(X) = -(0.3 \log_2 0.3 + 0.7 \log_2 0.7) = 0.881$. Nous souhaitons évaluer une coupure créant deux nouveaux noeuds S_1 et S_2 tels que le premier d'entre eux contienne 10 exemples dont la répartition est de $(9/10; 1/10)$, et le second les 90 exemples restants dont la répartition est donc $(21/90; 69/90)$. L'entropie au sein du premier noeud-fils vaudra donc : $H(X1) = -(9/10 \log_2 9/10 + 1/10 \log_2 1/10) = 0.469$ et dans le second noeud-fils : $H(X2) = -(21/90 \log_2 21/90 + 69/90 \log_2 69/90) = 0.784$. L'entropie résultant de la coupure sera alors la moyenne pondérée par les effectifs des noeuds des deux entropies précédemment calculées, c'est à dire : $H_G = 10/100 \times 0.469 + 90/100 \times 0.784 = 0.752$. L'entropie du noeud initial valant $H(X) = 0.881$, le gain d'entropie de Shannon, ou gain d'information, est donc $H(X) - H_G = 0.881 - 0.752 = 0.129$.
ID3 va donc sélectionner la coupure qui maximise le gain d'information et continuer à partitionner l'échantillon tant qu'il peut maximiser le gain d'information. L'un des problèmes du gain d'information est qu'il est peu sensible à la taille des partitions générées. Ainsi, ID3 a tendance à sélectionner des coupures amenant à des noeuds purs de faible effectif.

- **C4.5** Il s'agit d'une amélioration d'ID3 qui contourne le problème précité en utilisant le critère du *gain-ratio* [Quinlan, 1993]. Le *gain-ratio* consiste à diviser le gain d'information par l'entropie du prédicteur, $H_{partition} = -\sum_{p=1}^{n_{part}} \frac{|S_p|}{|S|} \log_2 \frac{|S_p|}{|S|}$, où n_{part} représente le nombre de noeuds engendrés par la partition, ce qui pénalise les attributs prédictifs ayant le plus grand nombre de modalités. C4.5 va alors rechercher la coupure maximisant $H_G/H_{partition}$. En reprenant l'exemple précédant, nous avons $H_{partition} = -((10/100 \log_2 10/100) + (90/100 \log_2 90/100)) = 0.469$, et donc un gain ratio de $0.129/0.469 = 0.275$. L'utilisation d'un critère plus adapté à la situation où les prédicteurs n'ont pas le même nombre de modalités n'est pas la seule amélioration apportée par C4.5 sur ID3. La plus significative tient à la mise en place d'un processus d'élagage. C4.5 développe l'arbre jusqu'au maximum puis effectue une rétropropagation visant à couper les branches de l'arbres qui sont considérées comme non pertinentes car contenant trop peu d'exemples. Le critère d'élagage de C4.5 consiste à calculer l'"erreur pessimiste" d'un noeud. Si celle-ci est supérieure ou égale à 0.5, le noeud est retiré de l'arbre.

Induction de règles Il s'agit des méthodes qui induisent en général une règle sous FND (Forme Normale Direct) de type «Si Condition alors Conclusion». Chaque règle couvre un sous-ensemble des exemples positifs de la classe. L'idée de base est d'induire itérativement des règles pour les instances positives des exemples en se fondant sur le principe *separate-and-conquer*. Le principe est de construire une règle qui prédit au mieux sur un sous-ensemble d'exemples (conquête), de supprimer les exemples couverts par la règle de l'ensemble d'apprentissage (séparer), pour enfin réitérer jusqu'à épuisement des exemples de l'échantillon. Parmi ces méthodes, assez peu utilisées, on trouve principalement :

- AQ [Hong and Michalski, 1986] La méthode AQ est parmi les première approches du principe "diviser pour régner". Elle utilise pour le phase de construction de règles (conquête), le principe de généralisation (stratégie ascendante). Le principe est qu'au début on considère chaque observation comme une règle et qu'à chaque itération on réduit les conditions pour couvrir plus d'observations. Pour la phase séparation, AQ retire de la base toutes les observations couvertes par la règle.
- CN2 [Clark and Boswell, 1991] L'algorithme CN2 regroupe à la fois des idées de AQ et de ID3. CN2 induit une liste ordonnée de règles de classification à partir d'exemples en utilisant l'entropie comme heuristique (héritage de ID3) et un principe de *beam search* (héritage de AQ). En effet, pour la phase conquête, CN2 se base sur le principe de spécialisation (stratégie descendante) ayant l'entropie comme critère de partitionnement. Pour la phase séparation, CN2 utilise une méthode *beam-serach* plus sophistiquée que celle utilisée par AQ. Le principe de cette fonction est de chercher une règle couvrant un échantillon d'exemples qui sont semblables de point de vue de la variable à prédire en maintenant plusieurs solutions en parallèle lors de l'exploration des solutions. Ainsi, elle supprime la dépendance de la règle aux données spécifiques et élargit l'espace de recherche en incluant des règles qui sont très pertinentes aux données d'apprentissage.

Les règles floues Ce type d'approche est utilisé pour l'apprentissage à partir d'un ensemble d'exemples ayant comme variables descriptives ou à prédire des variables numériques. Le problème avec ce type de variables est la discrétisation, puisque certaines approches ne peuvent fonctionner qu'avec un partionnement prédéfini des attributs numériques [Wang and Mendel, 1992]. D'autres approches génèrent un partionnement semi-global de l'espace d'attributs comme les arbres de décision. En effet, pour traiter les variables numériques, les règles floues introduisent une gradation lors de la définition des régions d'appartenance aux groupes dans l'espace de représentation par comparaison à la méthode "*crisp*" utilisée dans les règles usuelles. En fait, en utilisant cette gradation, les règles floues réduisent la variance d'où une performance en prédiction accrue par rapport aux méthodes "*crisp*". Cependant, le problème avec ces approches de règles floues est que lors du classement d'un nouvel individu plusieurs règles, à des degrés divers, seront déclenchées, pouvant être contradictoires puisque les régions définies sont plus ou moins dans l'espace de représentation.

Rough Sets Pour générer des règles, ces approches telles que LERS [Grzymala-Busse, 1987] et RSES [Bazan et al., 2004] reposent sur la théorie fondamentales des *Rough sets* présentée par Zdzislaw Pawlak au début des années 1980. En fait, ces approches proposent un cadre formel pour la transformation automatisée de données en connaissances grâce à un résultat important de la théorie qui simplifie la recherche des attributs dominants conduisant à des propriétés spécifiques, ou tout simplement des règles. Ces approches sont intéressantes surtout face aux données bruitées puisque la théorie des *Rough Sets* manipule l'imprécision et l'incertitude inhérentes aux situations de décision. Pour la classification, ces approches utilisent la fonction d'appartenance brute (RMS) de la théorie des *Rough Sets*. Cette fonction exprime fortement la manière dont un élément x appartient à l'ensemble brut X en vue de l'information sur l'élément exprimée par l'ensemble des attributs B. La RMS peut être utilisée comme une mesure de l'exactitude ou de la validité de dominance de l'élément.

Avantages et limites

Les avantages de ces différentes approches et principalement des arbres de décision sont la rapidité de l'apprentissage, la possibilité de modéliser des phénomènes non linéaires et de capter d'éventuelles interactions entre les variables. Un des autres avantages très prisé des arbres vient de leur interprétation aisée. En effet, on peut transformer le résultat d'un arbre en un ensemble de règles de décision facilement compréhensibles par l'être humain. Ces propriétés attractives souffrent cependant de plusieurs défauts.

En effet, les arbres de décision apprennent en recherchant les meilleures coupures (partitionnements de l'échantillon) de manière gloutonne. Le problème découlant de ce type d'apprentissage est qu'une erreur (l'utilisation d'une partition non optimale) à un niveau k de l'arbre sera propagée à tous les niveaux descendants. Un autre problème bien connu de ces approches est l'exposition au problème du sur-apprentissage. En effet, il est difficile de déterminer a priori le moment où stopper l'induction, c'est-à-dire arrêter de partitionner l'échantillon. Plusieurs auteurs, tels [Quinlan, 1993] ou [Breiman et al., 1984] utilisent des approches visant à contrôler la complexité de l'arbre pour ne pas trop sur-apprendre. Plus généralement, les arbres ont pour principal problème d'avoir une forte variance, c'est-à-dire d'être instables, très dépendants de l'échantillon utilisé pour l'induction. Ainsi, de petits changements dans l'échantillon sont susceptibles de produire des arbres très différents [Hastie et al., 2001].

Dans la mesure où ils donnent des résultats relativement instables, les arbres de décision se prêtent bien aux approches ensemblistes. On citera d'abord les approches non adaptatives telles que le *bagging* d'arbres en relançant l'algorithme sur des échantillons *bootstrap* de l'échantillon de départ ou les Forêts Aléatoires lorsque l'on rajoute un aléa sur les prédicteurs à chaque noeud de chaque arbre. Les approches adaptatives rajoutent tout au long des itérations une surpondération des exemples mal classés de l'itération précédente. Elles utilisent en général C4.5 comme classifieur de base. Ces approches augmentent notablement les performances des arbres, mais elles perdent généralement l'intelligibilité des règles. On citera une exception, dans le cadre booléen, lorsque le classifieur de base est un arbre de décision à un seul niveau (*decision stumps* [Quinlan, 1994]). Les calculs sont très simplifiés et le résultat peut s'interpréter comme un score à base de règles.

Enfin, comme nous l'avons déjà noté, les arbres de décision sont particulièrement perturbés par le déséquilibre des classes. Ces perturbations apparaissent à chaque étape de la méthode, la fonction d'éclatement, l'élaboration d'une règle de prédiction à partir des feuilles de l'arbre, et enfin l'élagage.

3.2. Les règles d'association prédictives

Règles d'association non supervisées, support et confiance

En apprentissage non supervisé, étant donné un ensemble d'exemples décrits par des caractéristiques booléennes, la méthode des règles d'association permet de découvrir des associations intéressantes entre caractéristiques (ou items) à partir de la description des exemples (ou transactions). Ces associations prennent la forme de règles du type *Si A alors C* où A et C sont des conjonctions de caractéristiques n'ayant pas de caractéristique commune. A et C renvoient respectivement à l'antécédent et au conséquent de la règle.

Historiquement, la qualité des règles d'association est appréciée selon deux critères, le support et la confiance. Le support d'une règle est défini comme étant le pourcentage des exemples (transactions) qui

contiennent tous les items figurant dans l'antécédent et le conséquent de la règle, c'est-à-dire le support de l'itemset AC (au sens de la conjonction des indicatrices). Un seuil minimal de support, noté minSupp, est fixé, à partir duquel un ensemble d'items est dit fréquent. La confiance correspond au pourcentage des exemples (transactions) qui contiennent tous les items du conséquent, parmi ceux contenant tous les items de l'antécédent. Le support évalue la force de la règle, son utilité tient avant tout au fait que la condition de support supérieur au seuil est antimonotone, ce qui permet une exploration efficace du treillis des itemsets. La confiance évalue l'intérêt de la règle. Pour extraire les règles, on recherche habituellement les règles dont le support et la confiance dépassent des seuils préfixés. Dans un premier temps, on extrait les itemsets fréquents, puis on en déduit de chacun d'entre eux les règles dont la confiance dépasse minSupp.

Définition des règles d'association prédictives

La classification associative est fondée sur la prédiction de la classe à partir de règles d'association particulières, dites règles d'association de classe (*class association rules*) ou règles d'association prédictives. Une règle d'association de classe est une règle dont le conséquent doit être la variable indicatrice de l'une des modalités de la classe. Une telle règle s'écrit donc $A \rightarrow c_i$, où A est une conjonction de descripteurs booléens et c_i est la variable indicatrice de la i_e modalité de classe. L'intérêt des règles de classe est de permettre la focalisation sur des groupes d'individus, éventuellement très petits, homogènes du point de vue des descripteurs et présentant la même classe. Par rapport aux arbres de décision, la classification associative présente l'intérêt de ne pas pratiquer de stratégie gloutonne, ce qui permet d'explorer l'ensemble des règles construites.

Les méthodes de classification associative procèdent en deux phases, une phase de construction des règles d'association de classe suivie d'une phase de prédiction à partir des règles de classe obtenues. La première phase se décompose elle-même en deux étapes, l'une consacrée à l'extraction des itemsets fréquents, compte tenu du seuil de support choisi, l'autre portant sur la construction des règles d'association de classe satisfaisant le seuil de confiance préfixé.

Les méthodes de classification associative

Parmi ces méthodes, on citera d'abord CBA (*Classification Based on Association*, [Liu et al., 1998]), le premier algorithme proposé. Cet algorithme se déroule en deux phases :

- CBA-RG utilise Apriori pour générer toutes les règles d'association qui satisfont les seuils de support et de confiance choisis au départ ;
- CBA-CB est un algorithme heuristique qui assure la prédiction : chaque exemple d'apprentissage est couvert par la règle ayant la confiance la plus élevée et celle-ci est stockée. Les règles stockées sont classées par ordre de confiance décroissant et on applique à un nouvel exemple la règle qui le couvre de plus forte confiance.

[Yin and Han, 2003] ont proposé une version améliorée de CBA appelée CPAR (*Classification based on Predictive Association Rules*). Cette méthode s'inspire du principe de la méthode FOIL (*First Order Inductive Learner*, [Quinlan and Cameron-Jones, 1995]) pour générer les règles, évitant ainsi la répétition des calculs pour les règles redondantes. Lors de la phase de classification, CPAR ne sélectionne pas seulement la meilleure règle (celle ayant la confiance la plus élevée) mais il sélectionne

les K meilleures règles pour chaque classe, compare la variance de chacune et choisit celle ayant l'exactitude la plus élevée.

Une autre amélioration proposée est CMAR (*Classification based on Multiple Association Rules*, [Li et al., 2001b]). CMAR utilise FP-Growth au lieu de l'algorithme Apriori. Il sélectionne les règles avec une confiance élevée et analyse la corrélation entre ces règles. La mesure utilisée est le *weighted Chi-square* qui exprime la force d'une règle à partir de la condition de support et de la distribution de la classe. Dans un souci d'efficacité, CMAR emploie une nouvelle structuration de données, CR-Tree, pour enregistrer et sélectionner les règles et pour chercher rapidement l'antécédent de la règle.

Pour les données en grandes dimensions, on trouve la méthode CAEP (*Classification by Aggregating Emmerging Patterns*, [Dong et al., 1999], [Dong and Li, 1999]). Cette méthode utilise un nouveau type de connaissance, les motifs émergents (*emmerging patterns* (EPs)). Ce sont des itemsets dont le support augmente significativement d'une classe à une autre et peuvent constituer l'antécédent des règles permettant de prédire efficacement la classe de certaines instances. Plus précisément les EPs sont définis comme les itemsets dont le taux d'accroissement du support dépasse un seuil préfixé. Pour traiter les problèmes multi-étiquette, [Thabtah et al., 2004] ont proposé MMAC (*Multi-class and Multi-label Association Classification*). En fait, l'algorithme considère le multi-étiquette comme une liste de rangs d'étiquette de la classe associée à l'instance et utilise une phase intermédiaire, celle d'apprentissage récursif entre la phase de génération de règles et la phase de classification.

3.3. Cadre général de l'apprentissage à base de règles

Apprentissage à partir d'un ensemble de classifieurs

Selon [Friedman and Popescu, 2005], le cadre général de l'apprentissage à base de règles comporte de nombreuses similitudes avec celui de l'apprentissage ensembliste *classique* comme le *Bagging* [Breiman, 1996] le *Boosting* [Freund and Schapire, 1996] [Friedman, 2000] ou encore les Forêts aléatoires [Breiman, 2001a]. **Afin d'éviter d'alourdir les notations mathématiques qui suivent, nous considérerons que l'output y prend ses valeurs dans $\{-1, +1\}$.** Un classifieur ensembliste peut être vu comme un méta-classifieur F agrégeant un ensemble de T classifieurs individuels f_t par le biais d'une combinaison linéaire [Friedman and Popescu, 2005] :

$$F(x) = a_0 + \sum_{t=1}^{T} a_t f_t(x)$$

où chaque classifieur individuel $f_m(x) = \{-1, +1\}$ est pondéré par un scalaire $a_t \in \Re$ permettant ainsi d'attribuer des poids différents à chaque classifieur en fonction de sa pertinence. Par exemple, les Forêts Aléatoires [Breiman, 2001a] créent les classifieurs parallèlement puis les agrègent à partir d'un vote à la majorité simple. Cela correspond à choisir f_t comme étant un arbre de décision *randomisé* appliqué sur un échantillon boostrap, puis à choisir $a_0 = 0$ et les a_t tous égaux. Un *Boosting* d'arbres de décision consiste à choisir f_t comme étant un arbre de décision, puis de lui affecter un poids a_t correspondant à son exactitude lors de l'itération courante.

Les techniques ensemblistes classiques diffèrent donc principalement sur deux points : sur la génération des classifieurs individuels (construction des f_t), puis sur la combinaison de ces classifieurs (calcul des a_t). Le pouvoir prédictif d'un ensemble de classifieurs dépend donc de ces deux paramètres.

Les techniques comme les Forêts Aléatoires ou le *Bagging* utilisant un vote à la majorité sont optimales uniquement si tous les classifieurs ont la même performance, s'ils sont indépendants et s'ils ont tous une probabilité d'erreur de prédiction inférieure au hasard. En effet, dans ce cas, la meilleure combinaison consiste à choisir tous les a_i égaux. Ces deux algorithmes cherchent à respecter au maximum ces conditions dès l'induction des classifieurs simples par le biais d'une construction *randomisée* ayant pour but d'induire des classifieurs indépendants et de performance identique. Toutefois, ces deux pré-requis ne sont pas complètement respectés en pratique et un *post-processing* (i.e, une recherche plus adaptée des a_i) peut aboutir à de meilleurs résultats (cf. par exemple [Robnik-Sikonja, 2004], [Pisetta et al., 2010]. Il apparaît donc intéressant de post-traiter l'ensemble obtenu quasi-systématiquement. De façon générique, le *post-processing* des classifieurs générés peut être obtenu en cherchant une solution au problème d'optimisation $(P1)$ suivant :

$$(P1)\ \ \{\hat{a}\}_0^T = \arg\min_{\{a\}_0^T} \sum_{i=1}^{n} L\left(y_i, a_0 + \sum_{t=1}^{T} a_t f_t(x_i)\right) + \lambda \sum_{t=1}^{T} \|a_t\|^l$$

Le premier terme de la somme représente l'évaluation de l'erreur de prédiction (mesuré à partir d'une fonction de perte $L(.)$), alors que le second terme pénalise de trop grandes valeurs associées aux classifieurs individuels. L'influence (la force) de cette pénalité est régulée par le paramètre $\lambda \geq 0$. Appliquer une contrainte sur la longueur du vecteur a est une technique très fréquemment utilisée en apprentissage automatique connue sous le nom de **régularisation de Tikhonov**. Son objectif est d'éviter le sur-apprentissage. Divers paramétrages de l et $L(.)$ donnent plusieurs classifieurs très connus dans le domaine de l'apprentissage statistique. Par exemple, choisir $l = 2$, et $L(y_i, F(x_i)) = max(0, 1 - y_i F(x_i))$ (fonction de perte dite *hinge loss*) revient à résoudre le problème du Séparateur à Vaste Marge (SVM) [Vapnik, 1995]. Le tableau ci-joint donne les équivalences avec des méthodes connues selon la valeur de l et $L(.)$.

Nom	l	L(.)	Références
LPBoost	1	$L(y_i, F(x_i)) = max(0, 1 - y_i F(x_i))$	Demiriz et al. (2002)
SVM	2	$L(y_i, F(x_i)) = max(0, 1 - y_i F(x_i))$	Vapnik (1995)
Ridge Reg.	2	$L(y_i, F(x_i)) = (y_i - F(x_i))^2$	Hoerl et Kennard, 1970)
Lasso Reg.	1	$L(y_i, F(x_i)) = (y_i - F(x_i))^2$	Tibshirani, (1996) [Tibshirani, 1994]
SVR	2	$L(y_i, F(x_i)) = \begin{cases} 0 \ si \ \lvert y_i - F(x_i)\rvert \leq \epsilon \\ \lvert y_i - F(x_i)\rvert - \epsilon \ sinon \end{cases}$	Smola et Schölkopf (2004)

Le choix de la fonction de perte dépend principalement du type de problème d'apprentissage traité. En régression, on choisira préférentiellement la fonction des moindres carrés alors qu'en classification supervisée, on choisira plutôt la fonction *hinge loss*. Le choix de la régularisation est beaucoup plus subtil et son réel impact n'est pas encore complètement déterminé aujourd'hui. La principale différence mise en avant est que la régularisation par norme L_1 aboutit à un résultat beaucoup plus épars que la régularisation par la norme L_2 [Meir and Rätsch, 2003], [Rosset et al., 2007]. Supposons que nous ayons une infinité de classifieurs. Dans ce cas, résoudre $(P1)$ en utilisant $l = 1$ ne devrait faire intervenir qu'une très petite fraction de l'infinité de classifieurs. En clair, la majorité des classifieurs auront des poids (a_t) égaux à 0. Le fait que la solution optimale dans un espace infini ne fasse intervenir qu'un petit nombre de variables est extrêmement intéressant. En effet, dans la pratique, il est difficile de

construire une infinité de classifieurs. Cependant, en utilisant des contraintes basées sur la norme L_1, on sait qu'un petit nombre de classifieurs est suffisant pour résoudre $(P1)$, laissant ainsi l'espoir que le nombre fini T de classifieurs construits sera suffisant pour trouver cette solution.

Au contraire, la régularisation par la norme L_2 a tendance à donner des petits poids à toutes les variables (les classifieurs dans notre cas). Ainsi, l'utilisation de la norme L_1 semble préférable à celle de la norme L_2. Le principal problème est que le résultat des programmes d'optimisation utilisant la norme L_1 fait intervenir des heuristiques et qu'il est donc soumis aux problèmes inhérents à ces techniques (optima locaux, etc.). En revanche, les techniques utilisant la norme L_2 peuvent toutes être résolues de façon optimale (généralement de façon matricielle ou via la programmation semi-définie ou quadratique). De plus, certains de ces programmes, comme les SVM, sont capables de gérer des espaces de dimension infinie par le biais de l'astuce du noyau (*kernel trick*).

Adaptation à un ensemble de règles

Le cadre de classification ensembliste introduit précédemment peut facilement être adapté au cas où l'on dispose non pas d'un ensemble de classifieurs, mais d'un ensemble de règles de décision. Considérons S_j comme l'ensemble de toutes les valeurs possibles de la variable X_j et s_{jm} un sous-ensemble de S_j, $s_{jm} \subseteq S_j$. Une règle de décision r_m correspond à une conjonction de la forme [Friedman and Popescu, 2005] :

$$r_m(x) = \prod_{j=1}^{k} I(x_j \in s_{jm})$$

où $I(.)$ est l'indicatrice de la réalisation de la condition. Chaque règle $r_m(x)$ peut prendre deux valeurs : 0 ou 1. $r_m(x) = 1$ si la condition est vérifiée par l'exemple x, 0 sinon. La nature des s_{jm} dépend de la nature de la variable initiale X_j :

- X_j **nominale** : dans ce cas, s_{jm} correpsond à une modalité de X_j.
- X_j **ordonnable (continue ou ordinale)** : les sous-ensembles s_{jm} d'une variable continue ou ordinale sont les intervalles contigus :
  $s_{jm} = binf_{jm}, bsup_{jm}$

définis par une borne inférieure $binf_{jm}$ et une borne supérieure $bsup_{jm}$. Imaginons X_j comme étant une variable représentant l'âge d'une personne. On peut considérer alors par exemple $s_{jm} = (15, 25)$ pour désigner les personnes entre 15 et 25 ans.

Dans la suite, nous supposons que les données sont discrètes. On trouve dans la littérature plusieurs qui permettent d'obtenir de bonnes discrétisations. Dans nos expérimentations, nous avons discrétisé les variables continues selon la technique implémentée dans LUCS-KDD [Coenen, 2003b]. Le principe de cette discrétisation est de diviser l'attribut en N intervalles discrets. Pour chaque intervalle, on compte le nombre d'exemples qui appartiennent à cet intervalle et la distribution de ces exemples par rapport aux modalités de classe disponibles, puis on identifie la modalité de classe dominante. On ordonne ces intervalles, pour combiner les intervalles ayant les mêmes classes dominantes. Lors de l'absence d'une classe dominante, car aucun exemple ne tombe dans un intervalle, la classe dominante est attribuée à la classe du plus proche voisin.

Le cadre théorique de l'apprentissage à partir d'un ensemble de règles devient équivalent à celui de l'apprentissage à partir d'un ensemble de classifieurs dès lors que l'on considère chaque classifieur

$f_t(x)$ comme étant une règle $r_m(x)$. La figure II.1 schématise ce type d'apprentissage.

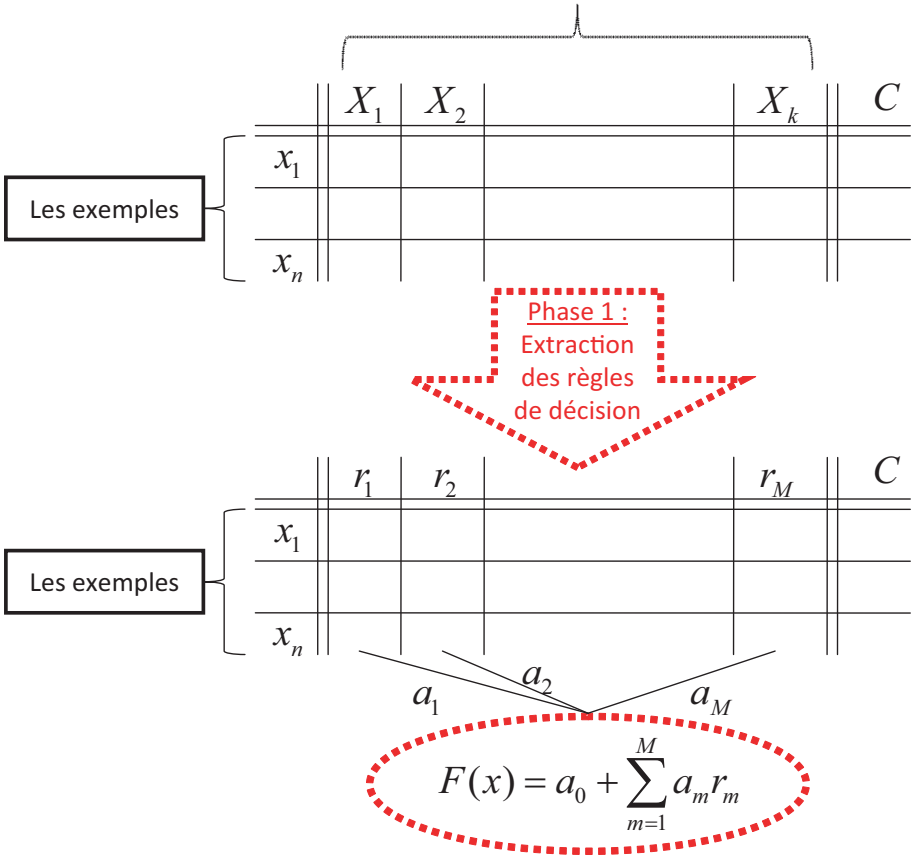

FIGURE II.1 – Apprentissage à partir d'un ensemble de règle

A ce niveau de l'analyse, le principal problème est de rechercher la méthodologie optimale opérant à deux niveaux. Le premier concerne l'extraction des règles de décision. Ces dernières vont constituer le nouvel espace de représentation à partir duquel nous allons apprendre. Il est par conséquent impératif que ce nouvel espace de représentation soit d'excellente qualité afin d'obtenir un résultat satisfaisant. Le second point concerne la procédure d'optimisation pour agréger les règles extraites. Plus précisément, quel paramétrage doit-on adopter (fonction de perte, l, p) afin d'obtenir le meilleur résultat ? Rappelons également que la notion de "meilleur résultat" est plus complexe que la simple minimisation du taux d'erreur. En plus d'être performant au niveau prédictif, le modèle extrait doit être facilement interprétable, et par conséquent ne doit surtout pas faire intervenir un nombre trop important de règles. Il doit de plus être d'exécution relativement rapide et ne doit pas nécessiter un trop grand espace de stockage.

Nous avons mis au point une procédure permettant de respecter tous ces impératifs. Notre point de vue est que l'extraction des règles doit être menée en fouillant tout l'espace des possibles en un temps rapide. Rechercher à travers tout l'espace garantit de ne pas passer à côté d'une règle potentiellement intéressante. La rapidité d'exécution ainsi que le faible espace de stockage nécessaire seront réalisés à partir d'une structure de fouille de treillis intelligente et particulièrement adaptée à la classification

supervisée. Les règles ainsi extraites seront ensuite agrégées via une procédure de *Boosting*. Cette dernière travaillant en norme L_1 nous paraît particulièrement adaptée pour ne faire intervenir qu'un nombre de règles peu important, tout en garantissant de bons résultats au niveau de la prédiction.

Dans la suite, afin de détailler notre procédure d'agrégation présenté par la figure II.2, nous commençons par présenter les principes d'**extraction de règles** et de prédiction à partir de ces règles, en nous référant à la classification associative. Par la suite nous nous intéressons au concept d'**agrégation des règles**.

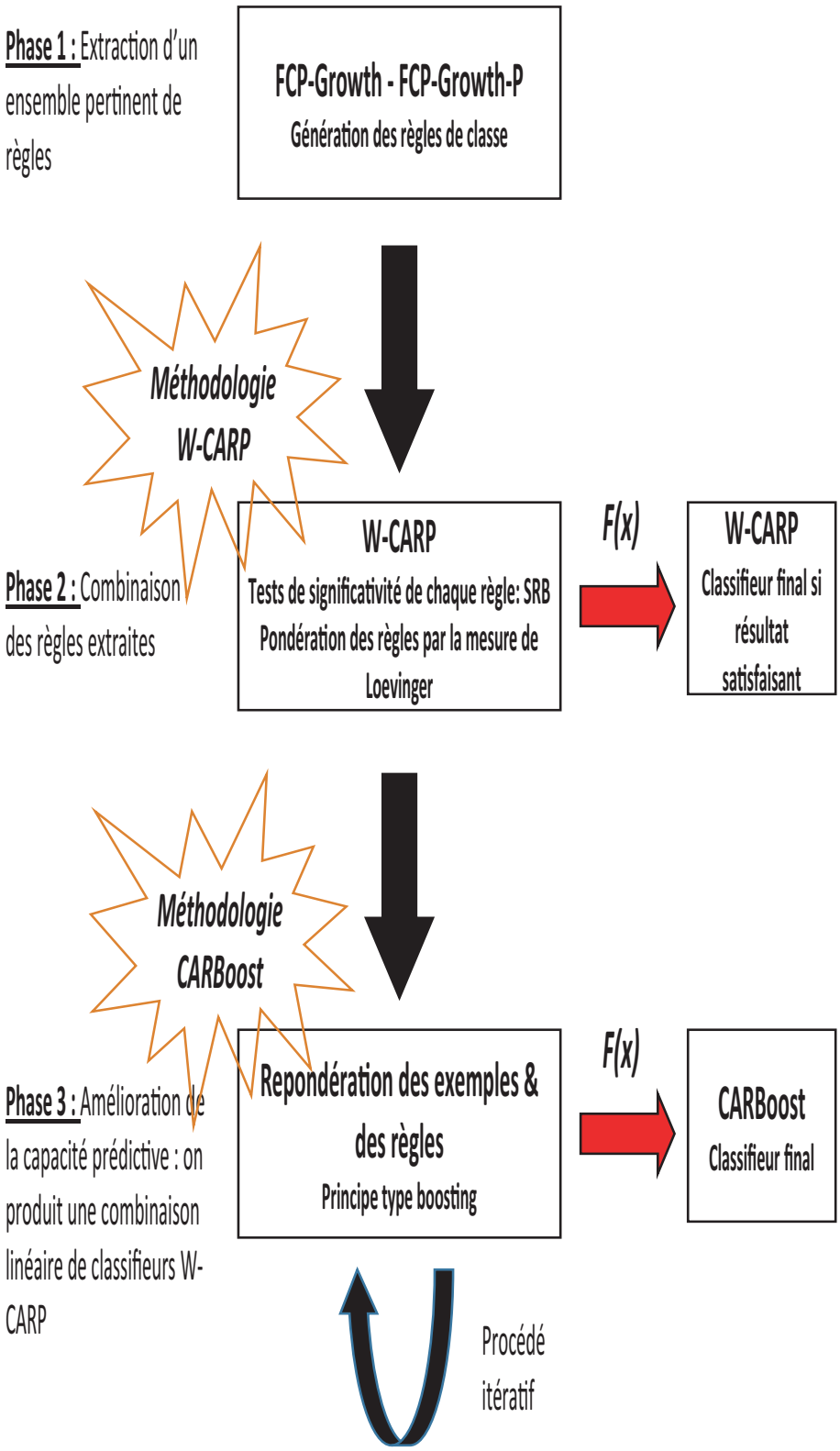

FIGURE II.2 – Schéma synthétique de nos contributions

4. Proposition d'un modèle à base de règles adapté au *Boosting* : phase d'extraction de règles

4.1. Les algorithmes d'extraction des itemsets fréquents

Comme nous l'avons constaté durant la section précédente, les méthodes de classification associative sont fondées sur une première phase de recherche des itemsets fréquents. Durant cette phase, ces méthodes ont recours aux algorithmes d'extraction de règles d'association utilisés en apprentissage non-supervisé. En effet, les auteurs ont eu recours à des algorithmes de recherche de règles d'association permettant de découvrir des informations utiles, cachées dans des grandes bases de données. C'est l'étape la plus coûteuse en terme de temps d'exécution car le nombre de ces items fréquents dépend exponentiellement du nombre d'items manipulés (pour p items, on a 2^p itemsets possibles). A partir des items fréquents on génère alors les règles d'association ayant une confiance supérieure à $minConf$, un seuil de confiance préfixé. Pour la prédiction, on choisit les règles ayant la meilleure qualité de bonne qualité, par exemple suivant une mesure telle que la confiance. Les principaux algorithmes d'extraction utilisés sont détaillés ci-dessous.

Apriori La stratégie utilisée par l'algorithme Apriori [Agrawal and Srikant, 1994] pour générer les itemsets fréquents est simple. Il détermine ceux-ci dans un ordre croissant de longueur, en se fondant sur la propriété d'antimonotonie de la condition de support. Selon cette propriété, tous les sur-itemsets d'un itemset non fréquent sont non fréquents et tous les sous-itemset d'un itemset fréquent sont fréquents. Au niveau k du treillis des itemsets, en fusionnant deux par deux tous les itemsets de longueur $k-1$, Apriori génère tous les itemsets candidats de longueur k et ne retient parmi ceux-ci que ceux dont la fréquence est supérieure ou égale au seuil de support $minSupp$. Ensuite, à partir de chaque itemset fréquent X, Apriori examine toutes les règles du type $X - Y \rightarrow Y$, où $Y \subset X$, $Y \neq \oslash$ et ne retient que celles dont la confiance est au moins égale à $minConf$, le seuil de confiance choisi. Ces règles sont alors ajoutées à ER, l'ensemble des règles d'associations qui satisfont aux seuils de support et de confiance préfixés. CBA utilise Apriori durant la première phase d'extraction de règles d'association de classe.

PRM L'algorithme PRM (*Predictive Rule Mining*) s'inspire du principe de FOIL (*First Order Inductive Learner*, [Quinlan and Cameron-Jones, 1995]). En effet, à chaque insertion d'items dans la règle, il calcule le FOIL-gain. Si le gain est maximisé, il garde cette règle et au lieu de supprimer les exemples qui sont couverts par cette règle, comme prévu dans FOIL, il diminue le poids de ces exemples pour générer plus de règles. Cet algorithme est utilisé dans l'approche CMAR

FP-Growth A l'opposé d'Apriori qui génère des itemsets candidats et qui les teste pour ne conserver que les itemsets fréquents, FP-Growth [Han et al., 2000] construit les itemsets fréquents sans génération de candidats. En fait, il utilise la stratégie "diviser et dominer" (*divide-and-conquer*). Tout d'abord, il compresse les itemsets fréquents représentés dans la base de données à l'aide des FP-Tree (*frequent-pattern tree*) dont les branches contiennent les associations possibles des items. Chaque association peut être divisée en fragments (*pattern fragment*) qui constituent les itemsets fréquents. FP-Growth transforme le problème de la recherche de l'itemset fréquent le plus long par la recherche du plus petit et sa concaténation avec le suffixe correspondant (le dernier itemset fréquent de la branche aboutissant à l'item considéré). Ceci permet de réduire le coût de la recherche. Cet algorithme est notamment celui utilisé par les auteurs de CMAR.

Eclat Contrairement aux deux méthodes déjà évoquées où l'on utilise des bases de données avec

des transactions horizontales, cet algorithme [Zaki, 2000]) est utilisé lorsque la base de données est stockée en verticale, l'appui d'un ensemble est obtenu par l'intersection simplement des couvertures de deux de ses sous-ensembles dont la réunion vaut l'ensemble lui-même. L'algorithme original d'Eclat a essentiellement employé cette technique à l'intérieur de l'algorithme Apriori. Pour la recherche des itemsets fréquents candidats, il est fondé sur le principe de la recherche en profondeur.

Dans la littérature, on trouve aussi des algorithmes qui dépendent de la nature des itemsets fréquents. Par exemple, pour les itemsets fréquents fermés, on trouve CLOSET [Pei et al., 2000]) et CHARM [Zaki and Hsiao, 1999]). [Garriga et al., 2006] ont étendu la notion d'itemset fermé au cas supervisé. Ils ont montré que les itemsets fermés caractérisent l'espace des attributs qui sont pertinents pour discriminer la classe en évitant la redondance.

Limites de ces algorithmes face à une approche itérative La plupart des auteurs accomplissent la première phase d'extraction de règles en ayant recours aux algorithmes utilisés pour extraire les règles d'association en apprentissage non-supervisé, en particulier Apriori, utilisé par [Liu et al., 1998], et FP-Growth, utilisé par [Li et al., 2001b]. Cette génération massive des règles ainsi que l'obligation de filtrer les règles obtenues pour ne garder que les règles de classe représente une perte de temps et d'espace et rend difficile l'obtention des pépites, à savoir les règles de très petit support mais de très forte confiance, particulièrement utiles pour la prédiction des classes minoritaires. De plus, dans les problèmes réels, le plus souvent les classes sont déséquilibrées et c'est la classe la moins nombreuse qui est la classe d'intérêt. L'extraction non supervisée des motifs dont la fréquence dépasse le seuil de support, à l'aide d'algorithmes comme Apriori ou FP-Growth pénalise d'autant plus la classe minoritaire que sa fréquence est faible. Si la fréquence de la classe minoritaire est de 1% et que l'on utilise un seuil de support de 1%, on ne trouvera que les règles logiques comme règles débouchant sur la classe minoritaire.

Cette étude critique a mis en évidence les faiblesse des différents algorithmes d'extraction existant dans la littérature, compte tenu de notre objectif qui est de proposer une méthode adaptative de classification associative. Nous devons choisir un algorithme d'extraction de règles et l'adapter à nos objectifs. Une première solution consisterait à adapter Apriori, en s'appuyant notamment sur l'article de [Srikant et al., 1997] qui propose trois variantes d'Apriori permettant de tenir compte de contraintes booléennes sur les items. Cependant, l'étude de [Li et al., 2001b]) a montré que l'utilisation de FP-Growth permet un gain d'espace de stockage par rapport à Apriori pouvant parfois atteindre 75% sur des bases de données de grande taille ainsi qu'une plus grande vitese d'exécution. La structure arborescente des FP-Tree permet une recherche plus efficace et plus économe des itemsets fréquents.

4.2. FP-Growth

Pour bien cerner le fonctionnement de FP-Growth, nous avons détaillé le déroulement de l'extraction d'itemsets fréquents sur un exemple. Nous avons choisi une base de données simplifiée qui contient des items notés I et des items de classe noté C (cf. tableau II.1).

Premièrement, FP-Growth parcourt la base de données et lit tout les items existant en calculant leur support ou fréquence d'apparition et les compare avec le support déjà fixé. Ainsi, il n'accepte que les items ayant un support supérieur ou égal au support minimum, appelés items fréquents. Pour cette exemple, on va choisir un support minimal de 1 (tableau II.2).

FP-Growth enregistre les items fréquents dans une liste L où les items sont classés par ordre

Id Transaction	Liste d'item-TID
T1	I1,I2,I5,C1
T2	I2,I4,C2
T3	I2,I1,I3,C1
T4	I1,I2,I4,C2

TABLE II.1 – Exemple

ITEMS	Support
I2	4
I1	3
I4	2
C1	2
C2	2
I3	1
I5	1

TABLE II.2 – items associés à leur support

descendant de support. Pour cet exemple, la liste correspondante est :

L= {I2 :4},{I3 :3},{I4 :2},{C1 :2},{C2 :2},{I3 :1},{I5 :1}

Deuxièmement un FP-Tree est construit par la création de Root assignée à null et l'on parcourt une deuxième fois la base de données mais cette fois-ci les items de chaque transaction sont énumérés dans l'ordre donné par la liste L. Par exemple, la première transaction T1 :I1,I2,I5,C1 sera sauvegardée sous la forme T1 :I2,I1,C1,I5. Cette première transaction, avec cet ordre, va constituer la première branche de FP-Tree avec 4 noeuds (I2 :1),(I1 :1),(C1 :1) et(I5 :1). La seconde transaction T2, contenant T2 dans l'ordre de L : I2,I4,C2, construit une branche où le noeud I2 est lié à root, le noeud I4 lié à I2 et le noeud C2 lié à I4. Cette branche partage un préfixe commun, I2, avec T1. Ainsi, lors de construction du noeud I4 :1, on incrémente le compteur du noeud I2 de 1(I2 :2) (voir figure II.3).

Troisièmement, le FP-Tree est fouillé par la création des (sub-)fragments conditionnels de base. En fait, pour trouver ces fragments, on extrait pour chaque fragment de longueur itemset-1, l'ensemble des itemsets existant dans le chemin du FP-Tree. Le fragment est considéré comme le *suffix pattern*, et l'ensemble des itemsets existant dans le chemin du FP-Tree comme le *conditional pattern base*. L'itemset fréquent est obtenu par la concaténation du *suffix* avec les fragments fréquents extraits des *conditional FP-Tree* (voir le tableau II.3)

Item	Cond P	cond FP-Tree	FP
I5	I1,I2,C1,I2,I1,I3	(I2 :2,I1 :1)	I2,I5 :2,I1,I5 :2,I2,I1,I5 :2
I4	I2 :1,I2,I1 :1	(I2 :2)	I2,I4 :2

TABLE II.3 – Fouille du FP-Tree

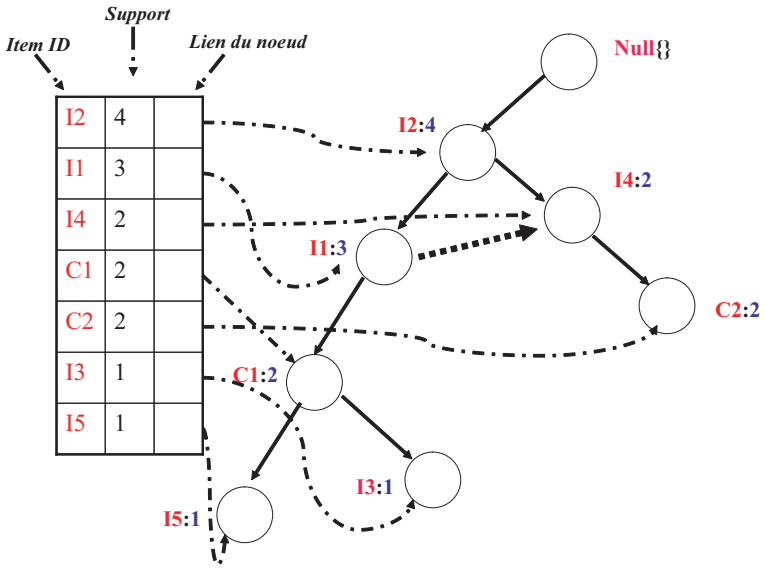

FIGURE II.3 – Construction de FP-Tree

4.3. Contribution 1 : adaptation de FP-Growth à la classification supervisée (FCP-Growth)

FCP-Growth Afin de limiter le coût de construction des règles d'association prédictives en termes de temps d'exécution et d'espace de stockage, on doit se limiter à la génération des seuls itemsets fréquents de classe, c'est-à-dire les itemsets fréquents qui contiennent l'une des modalités de classe. A cette fin, nous proposons FCP-Growth (*Frequent Class Pattern*) une variante de FP-Growth qui ne stocke un itemset fréquent dans le FCP-Tree (*Frequent Class Pattern Tree*) que si celui-ci est un itemset de classe, du type $c_i A$. Un tel itemset ne génère qu'une seule règle de classe, la règle $A \rightarrow c_i$.

Simultanément, pour résoudre le problème des classes déséquilibrées, nous proposons de fixer un seuil de support ajustable, qui tient compte de l'effectif de chaque modalité de la classe à prédire. Si n_i désigne l'effectif de c_i, la i^e modalité de classe, une fois choisi le seuil σ sur la base entière, le seuil de support adapté à la classe c_i s'écrit σ_i, où $n \times \sigma_i = n_i \times \sigma$. Le seuil de support adapté est donc proportionnel à l'effectif de classe, de telle sorte que si l'on exprime ce seuil en proportion de l'effectif de classe et non plus de l'effectif de la base, celui-ci soit le même pour chaque classe. De la sorte, on évite que les classes les moins nombreuses soient systématiquement désavantagées lors de la construction des itemsets de classe et des règles d'association prédictives associées.

Fonction Fonction 1 (Construction du FCP-Tree)(Entrées) : **Sorties**

Entrées : Une base de données de transactions DB, un seuil de support global, une classe à prédire.

Sorties : FCP-Tree, La frequent-class-pattern tree de DB.

[La FCP-Tree est construite comme suit.]

Scanner la base de transactions DB une fois

Chercher l'attribut à prédire (Classe)

Réorganiser la base de données de transactions de telle façon que la première colonne contient les items de classe

Collecter F, des items fréquents correspond à chaque modalité de classe, et le support de chaque item fréquents

Attribuer aux items de classe un poids supérieur

Enregistrer F dans une FList correpondant à la modalité de classe avec un ordre descendant du support.

Créer la racine de FP-Tree, T , et l'étiqueter avec "null".

Pour transaction Trans dans DB **faire**

Sélectionner les items fréquents dans Trans et les enregistrer selon l'ordre de FList. Notons la liste d'items fréquents enregistrés dans Trans par $[p|P]$, avec p est le premier élèment (l'item de classe) et P la liste correspondante

Appeler insert tree($[p|P], T$).

Fin Pour

Fonction insert tree($[p|P]$,T) : T)

Si (T possède un "fils N" telque $N.item-name = p.item-name$) **Alors**

incrémenter le compteur de N par 1

Sinon

créer un nouveau noeud N, avec un compteur initialisé à 1, le lien de son parent est attaché à T le lien de ce noeud est attaché aux noeuds ayant le même item-name à travers.

Fin Si

Si (P est non vide) **Alors**

appeler insert tree(P, N) récursivement

Fin Si

Fin

Fin

Algorithme 5 – Construction de FCP-Tree

Fonction Fonction2 (Fouille de FCP-Tree)(Entrées) : **Sorties**

Entrées : Une base de données DB, representée par FCP-Tree construit en utilisant Fonction1, un seuil du support global

Sorties : L'itemset de classe fréquent.

[Méthode : Appeler FCP-Growth(FCP-Tree, null)]

[Procédure de FCP-Growth(Tree, α)]

Si (Arbre "Tree" contient un unique chemin) **Alors**

Afficher erreur "arbre doit contenir au moins deux chemin cas binaire"

Sinon

Considérons Q un arbre "Tree"

Fin Si

[Fouille de multichemin de FCP-Tree]

Pour chaque item a_i dans Q **faire**

Générer motif "pattern" $\beta = a_i \bigcup \alpha$ avec un support = a_i .support

Construire le conditional pattern-base de β et aprés le conditional FCP-Tree de β noté C

Fin Pour

Si ($Tree_\beta \neq \phi$.) **Alors**

Appeler FCP-Growth($Tree_\beta$, β)

Fin Si

[Considérons un freq pattern set(Q) comme la liste des patterns générés]

Retourner ($freqpatternset(P) \bigcup freqpatternset(Q)$ $\bigcup (freqpatternset(P) X freqpatternset(Q))$)

Fin

Algorithme 6 – Fouille de FCP-Tree et extraction des items de classe fréquents

Explication de l'algorithme Notre adaptation a pour principe d'ajouter des modules à la version de base FP-Growth. En fait, on distingue 4 étapes :

1. Prétraitement de la base de données : au cours de cette étape, on donne la possibilité à l'utilisateur de choisir la classe à prédire à partir des différents attributs de la base de données. Une fois la classe à prédire choisie, on réorganise la base de données sous forme d'un tableau dont la première colonne contient les différentes modalités assignées à cette classe. Cette méthode nous assure de construire le FCP-Tree des seuls itemsets de classe fréquents. Pour expliquer notre méthode et la comparer avec la version d'origine FP-Growth, on va utiliser le même exemple. Après la phase de prétraitement, la base de données sera organisée comme suit (tableau II.4) :

TID	List of item-TID
T1	C1,I1,I2,I5
T2	C2,I2,I4
T3	C1,I2,I1,I3
T4	C2,I1,I2,I4

TABLE II.4 – Base de données après le prétraitement

2. Ajustement du seuil de support : Cette étape nous permet de définir, non pas un seuil de support fixe comme dans la version de base de FP-Growth, mais un seuil de support adapté à chaque modalité c_i, qui dépend de n_i, le nombre de transactions validant la modalité. Par exemple, pour la modalité c_i, on aura $\sigma_i = \sigma \times n_i/n$. Durant cette phase, au lieu d'une seule liste L qui contient les items fréquents par ordre décroissant du support comme FP-Growth, FCP-Growth construit autant de liste L que de modalités de classe et attribue aux items de classe le plus grand poids P qui sera considéré comme support. Ainsi, chaque liste commencera par l'item de classe correspondant à son support minimum de classe. Pour notre exemple, on supposera que le support de C1 est 200 et C2 de 100. Si on définit σ à 1% alors on aura un support minimum de 2 pour C1 et de 1 pour C2 (tableau II.5).

ITEMS	Support C1	Support C2
I2	4	4
I1	3	3
I4	2	2
I3	-	1
I5	-	1

TABLE II.5 – Items fréquents selon le support de classe

Pour cet exemple, on aura deux liste, L1 pour la classe C1 et L2 pour la classe C2 :
- L1={C1 :P1},{I2 :4},{I3 :3},{I4 :2}
- L2={C2 :P2},{I2 :4},{I3 :3},{I4 :2},{I3 :1},{I5 :1}

Donc d'après L1, la première transaction sera enregistrée comme suit : C1,I2,I1

3. Construction du FCP-Tree et des motifs fréquents : Cette étape se déroule de la même façon que FP-Growth. Cependant, seules les transaction qui commencent par une modalité de la classe à prédire seront traitées. La phase prétraitement nous a facilité cette tache. Ainsi aura-t-on autant de fils de la racine du FCP-Tree que de modalités de la classe. On commence, alors, par la construction de la racine NULL, on parcourt la base de données, à chaque fois qu'il y a une modalité de classe différente, on construit un noeud lié à la racine de FCP-Tree. Pour notre exemple, on construit deux noeud fils C1 et C2. La première branche de FCP-Tree est construite par la première transaction T1. Par exemple, la première branche correspondant à T1 sera liée au noeud C1 par deux noeuds I2 :1 et I1 :1. Ainsi, on est sûr que pour chaque item fréquent, on a un fragment conditionnel qui contient un item de classe (voir II.3).
4. Fouille de FCP-Tree et génération des items fréquents : On procède de la même façon que FP-Growth. On extrait les *suffix fragments* et on construit pour chacun ses fragments conditionnels.

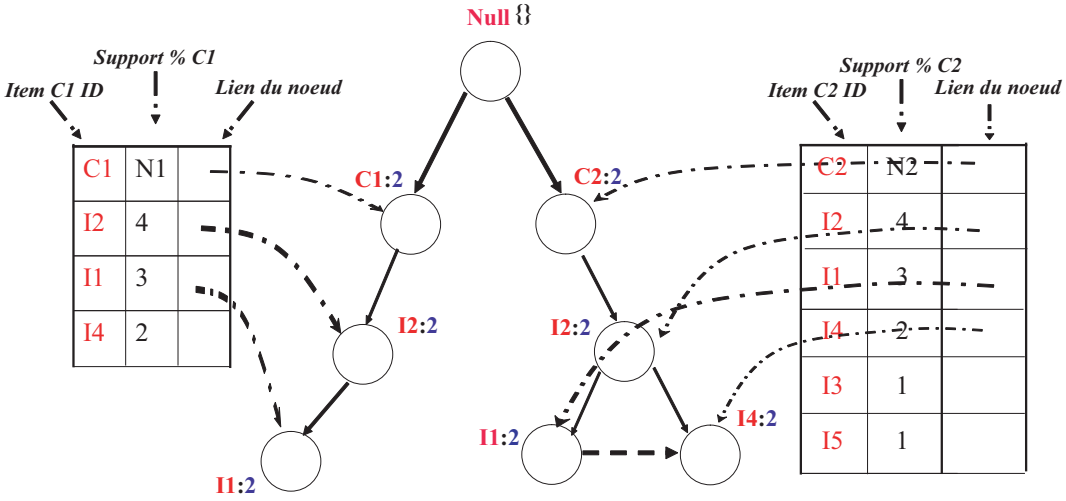

FIGURE II.4 – Construction de FP-Tree adapté

On extrait alors tous les itemsets de classe fréquents (i.e. les itemsets de classe ayant un support supérieur ou égal au seuil de support choisi). Les itemsets fréquents sont obtenus par l'association de chaque item avec ses fragment fréquents correspondants. Pour notre exemple, on est sûr de ne pas trouver l'itemset I4I1 trouvé dans la table 3 grâce à FCP-Growth et à la modification de construction du FCP-Tree où chaque item doit être attaché à une branche qui contient l'item de classe

4.4. Contribution 2 : amélioration de la qualité des règles extraites (FCP-Growth-P)

Les méthodes d'élagage usuelles en classification associative

L'objectif des méthodes d'apprentissage supervisé par induction de règles et en particulier de la classification associative est de fournir des règles performantes pour la prédiction. Le problème majeur de ces méthodes est l'extraction massive des règles non intéressantes. Ces règles peuvent être contradictoires, redondantes et surtout très évidentes. Examinant de prés ces méthodes, nous avons constaté que lors d'extraction des itemsets fréquents, elles se fondent le plus souvent sur un seul principe d'élagage, celui de la condition du support : si le nombre d'exemples qui valident l'itemset construit est supérieur ou égal au seuil de support préfixé, on le garde, sinon on le supprime. Donc si on veut réduire le nombre de règles extraites en maintenant la qualité, on doit utiliser un procédé plus efficace de sélection de règle ou d'élagage.

Dans la littérature, il existe plusieurs façon d'élaguer qui donnent de bons résultats. On trouve par exemple :

La règle la plus générale : soient R1 et R2 deux règles, R1 est plus générale que R2 si elle correspond à l'un ou l'autre des trois cas suivants :

- soit conf(R1)>conf(R2)
- soit conf(R1)=conf(R2) et supp(R1)>supp(R2)

– conf(R1)=conf(R2) et supp(R1)=supp(R2) et R1 possède moins d'items

De ce fait, l'idée est de supprimer la règle la moins générale. CMAR [Li et al., 2001b] utilise ce principe d'élagage dans son algorithme de génération de règles d'association de classe

La règle la plus performante : d'autre auteurs utilisent différentes mesures de la performance d'une règle. Ces mesures fondent une condition de suppression ou d'élagage. Citons l'exemple de FOIL [Quinlan and Cameron-Jones, 1995], utilisé par CPAR [Yin and Han, 2003], qui se base sur la mesure "$FOIL-Prune(R) = (pos-neg) \div (pos+neg)$" avec pos et neg qui sont les nombres de tuples positifs et négatifs couverts par la règle R. On trouve aussi d'autres mesures telles que la corrélation d'une règle avec la classe mesurée par le Khi^2. Cette méthode est utilisée par CMAR [Li et al., 2001b]

Comportement de la règle vis-à-vis des contre-exemples : les travaux de Li [Li and Jones, 2006], [Li, 2006], [Li and Zhang, 2003] et [Li et al., 2001a] proposent une condition d'élagage qui est utilisée avec Apriori pour les règles de classe. Si la spécialisation d'une règle ne diminue pas le nombre de contre-exemples, alors ni cette spécialisation, ni aucune sur-spécialisation n'est plus intéressante au sens de la confiance et de 12 autres mesures.

En fait, l'extraction de la règle optimale se base sur le principe de l'*informative rule set* pour élaguer les règles qui ne sont pas intéressantes. L'*informative rule set* est défini comme suit :

– Soit R_a une règle d'association
– Soit R_i un "informative rule set" alors R_i est le plus petit subset quand pour chaque itemset I, la prédiction de I à partir de R_i est la même qu'à partir de R_a.

C'est ce dernier critère d'élagage, utilisé par Li avec Apriori, que nous avons choisi d'introduire dans l'algorithme FCP-Growth pour élaborer FCP-Growth-P, un algorithme fondé FP-Growth qui se limite aux seuls itemsets de classe fréquents vérifiant la condition d'élagage de Li.

FCP-Growth-P : Elagage fondé sur le nombre de contre exemples

FCP-Growth-P (*Frequent Class Pattern-Growth-Prune*) est une variante de FCP-Growth. En fait, nous avons introduit dans FCP-Growth la méthode d'élagage précitée de Li [Li, 2006] qui repose sur le fait que toute spécialisation d'une règle de classe qui ne diminue pas le nombre de contre-exemples est moins intéressante que la règle de départ. Cette méthode a été proposée dans le cadre d'Apriori, un algorithme d'extraction d'itemsets fréquents avec génération de candidats. Pour l'introduire dans FCP-Growth qui est un algorithme d'extraction d'itemsets fréquents sans génération de candidats, nous avons dû procéder à quelques adaptations. A chaque transaction et la construction d'un noeud (item I), on met à jour et on calcule le nombre de transactions qui valident l'association des items depuis le début de la branche jusqu'au noeud de l'item, $supp(Ic)$, et le nombre de transactions qui contredit cette association, $Supp(I\overline{c})$.

Soit l'itemset de classe Ac, la règle de classe associée est $A \rightarrow c$. Les contre-exemples de cette règle correspondent à l'itemset $A\bar{c}$. Considérons l'item x auquel on associe la spécialisation $Ax \rightarrow c$. La condition d'élagage est la suivante : pour une mesure m optimale au sens de Li, si $supp(Ax\bar{c}) = Supp(A\bar{c})$, alors $m(Ax \rightarrow c) \leq m(A \rightarrow c)$. Pour intégrer cette contrainte dans un algorithme, il faut être capable de tenir à jour le nombre de contre-exemples de la règle de classe associée à l'itemset de classe courant.

Dans FCP-Growth, chaque fils de la racine vide correspond à une modalité de classe. Par exemple, si on a les itemsets de classe ca_1a_2, puis $ca_1a_2a_3$, la condition d'élagage conduit à comparer les supports des contre-exemples de chacune des deux règles de classe $a_1a_2 \rightarrow c$ et $a_1a_2a_3 \rightarrow c$, c'est-à-dire $p_{\bar{c}a_1a_2}$ et $p_{\bar{c}a_1a_2a_3}$.

FCP-Growth-P tient à jour les supports $p_{ca_1a_2}$, et $p_{ca_1a_2a_3}$. Pour calculer $p_{\bar{c}a_1a_2} = p_{a_1a_2} - p_{ca_1a_2}$ et $p_{\bar{c}a_1a_2a_3} = p_{a_1a_2a_3} - p_{ca_1a_2a_3}$, il faut connaître $p_{a_1a_2}$ et $p_{a_1a_2a_3}$. De façon générale, à chaque noeud cA de l'arbre il suffit de calculer non seulement p_{cA} mais aussi p_A, ce qui permet d'en déduire $p_{\bar{c}A} = p_A - p_{cA}$. Dans l'exemple proposé, si $p_{\bar{c}a_1a_2a_3} = p_{\bar{c}a_1a_2}$, on élague ce noeud, car ni la règle spécialisée $a_1a_2a_3 \rightarrow c$ ni aucune de ses sur-spécialisations ne sont plus intéressantes que la règle $a_1a_2 \rightarrow c$, au sens de la confiance ou de toute mesure optimale au sens de Li. Une condition nécessaire et suffisante d'optimalité au sens de Li donnée par [LeBras et al., 2009] permet de déceler facilement ces mesures.

4.5. Etude de la complexité de FP-Growth et de FCP-Growth-P

FCP-Growth et FCP-Growth-P ont une complexité en *O(nKR)*, sachant que n est le nombre d'exemples, K le nombre d'attributs et R le nombre de règles générées. Cette complexité est similaire à celle de FOIL et de FP-Growth.

Preuve. Pour le parcours de la bases de données et l'enregistrement des items fréquents dans l'ordre décroissant, FCP-Growth-P a une complexité en *O(nK)*, puisqu'il doit déterminer la fréquence de chaque item pour tous les exemples. Pendant la construction du FP-Tree, pour chaque exemple, il doit construire au pire une branche de K attributs donc la complexité est de l'ordre *O(nK)*. C'est ainsi que la construction d'une règle est de l'ordre de *O(nK)*. La complexité finale de FCP-Growth-P doit prendre en compte la construction de toutes les règles, ce qui fait qu'elle est de l'ordre de *O(nKR)*.

4.6. Performances de FCP-Growth et FCP-Growth-P

Pour évaluer l'efficacité de FCP-Growth et FCP-Growth-P, nous avons testé ces algorithmes sur 7 bases de données réelles volumineuses et nous avons calculé (tableau II.7) à partir d'un seuil de support $\sigma = 1\%$ les quantités suivantes :

- nombre total d'itemsets, noté # total Itemsets
- nombre d'itemsets de la classe Ci, noté # Itemsets Ci
- nombre d'itemsets non pertinents, noté # Itemsets non pertinents
- temps d'exécution
- taux de couverture

Bases de données	**#Exp**	**#Attr**	**#Classes**	**Poids de classe**
D100T20I6	100000	20	2	58%-42%
D100T20I2	100000	20	2	55%-45%
Retail data	63000	10	2	69%-31%
Adult	48842	14	2	75,22%-24,78%
Connect-4	67557	42	3	65,8%-24,6%-9,6%
Abalone	4177	8	3	53%-25%-22%
Census	48842	14	2	75.2%-24.8%

TABLE II.6 – Description des bases de données

La comparaison des résultats de FCP-Growth et de sa variante FCP-Growth-P, avec ceux de la version d'origine FP-Growth (tableaux II.7, II.8) est menée sur la base de 4 critères : la diminution du nombre total d'itemsets construits, l'augmentation du nombre d'itemsets de classe, la bonne représentation de la classe minoritaire et l'amélioration du taux de couverture (c'est-à-dire la proportion d'exemples couverts par un itemset de classe fréquent).

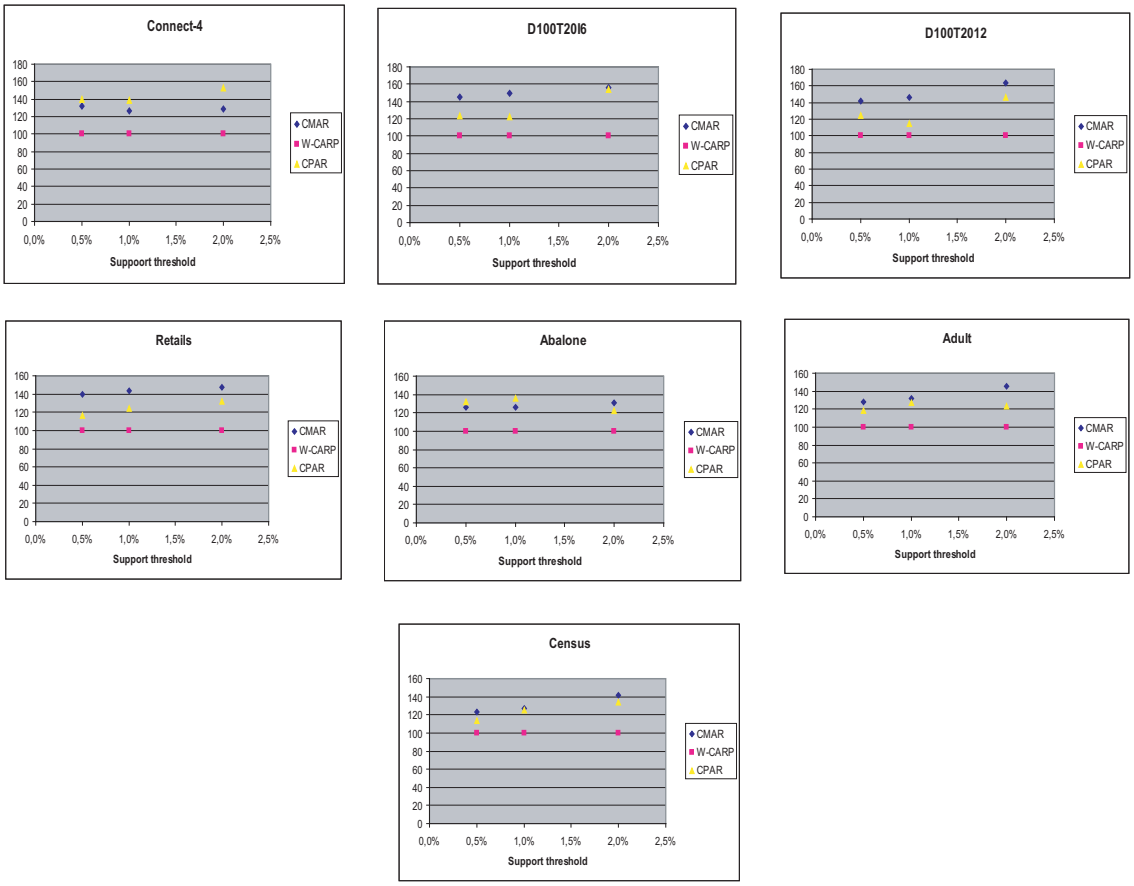

FIGURE II.5 – Comparaison du nombre de règles générées par CMAR, CPAR et W-CARP

Les résultats montrent, tout d'abord, l'importance du poids des itemsets non pertinents (les itemsets ne contenant pas la classe). Sur l'ensemble des sept bases traitées, il apparaît qu'entre le tiers et la moitié des itemsets générés par FP-Growth ne sont pas pertinents, ce qui alourdit inutilement la procédure. De plus, la mesure Variation1 (obtenue en divisant # Itemsets FCP-Growth par # Itemsets FP-Growth) montre bien que, grâce à sa procédure d'élimination des itemsets qui ne sont pas de classe et à son seuil adaptatif, FCP-Growth ainsi que sa variante FCP-Growth-P permettent d'extraire plus de règles de classes que FP-Growth, tout en évitant que la classe minoritaire soit trop défavorisée. Sur les sept bases de données, ce gain varie entre 17% et 23%.

Pour évaluer les performances de FCP-Growth et FCP-Growth-P, nous avons retenu le taux de couverture qui indique la proportion d'exemples qui sont couverts par au moins un itemset de classe, c'est-à-dire qui vérifient la règle de classe correspondant à cet itemset. Les résultats du tableau II.8 montrent très clairement la supériorité de FCP-Growth. En effet, pour chacune des bases, le taux de couverture est nettement augmenté pour arriver environ à plus de 96% de couverture. De plus, pour une meilleure qualité et grâce à sa procédure d'élagage, la variante FCP-Growth-P, héritant déjà des avantages de FCP-Growth en termes d'augmentation du nombre d'itemsets de classes, permet de réduire la quantité de règles de classe extraites en améliorant la qualité. En effet, la mesure variation2 du tableau II.7 (obtenue en divisant # Itemsets FCP-Growth-P par # Itemsets FCP-Growth) montre que FCP-Growth-P diminue le nombre de règles de classe extraites de 6,67% à 11,37%, tout en assurant le même taux de couverture (tableau II.8).

Alors que dans les expérimentations, FCP-Growth permet de traiter globalement moins d'itemsets que FP-Growth, tout en générant plus d'itemsets de classe et en améliorant le taux de couverture par les règles de classe, on constate que le temps d'exécution de FCP-Growth est toujours inférieur ou égal à celui de FP-Growth comme le montre le tableau II.8. Cependant, on note aussi bien que la procédure d'élagage, introduite dans la variante FCP-Growth-P, pénalise le temps d'exécution qui augmente nettement. Ceci était attendu puisque à chaque construction d'item, il doit calculer le nombre de contre-exemples de ce dernier.

Les résultats trouvés montrent que FCP-Growth et plus encore FCP-Growth-P permettent un gain de temps grâce à la génération des seuls itemsets fréquents de classe. Ce gain peut aussi être exprimé en termes de facilité de construction et de stockage. De plus, l'examen des taux de couverture montre que nous obtenons une qualité de règles supérieure à celle des règles générées par FP-Growth. Au vu des résultats obtenus, nous allons utiliser FCP-Growth-P comme algorithme de génération de règles pour la première phase d'une procédure de classification associative performante. La deuxième phase celle de la prédiction à partir des règles de classe extraites sera traitée dans la section suivante.

Bases de données	Variation1	FP-Growth	FCP-Growth	FCP-Growth-P	Variation2
Seuil du support absolu	*1,00%*	*1,00%*	*1,00%*	*1,00%*	*1,00%*
D100T20I6					
# Itemsets	-24,77%	1090	820	730	-10,97%
# Itemsets C1	12,22%	452	508	438	-13,72%
# Itemsets C2	28,46%	243	312	292	-6,48%
# Itemsets non pertinents	-	395	0	0	-
# Total itemsets de classe	17,98%	695	820	730	,-10,97%
D100T20I2					
# Itemsets	-24,62%	1415	1075	967	-10,04%
# Itemsets C1	23,78%	565	699	610	-12,73%
# Itemsets C2	23,06%	306	376	357	-6,48%
# Itemsets non pertinents	-	545	0	0	-
# Total itemsets de classe	23,56%	870	1075	967	-10,04%
Retail data					
# Itemsets	-25,77%	330	245	224	-8,57%
# Itemsets C1	5,13%	121	127	117	-8,51%
# Itemsets C2	92,16%	40	118	107	-8,59%
# Itemsets non pertinents	-	169	0	0	-
# Total itemsets de classe	52,17%	161	245	224	-8,57%
Abalone					
# Itemsets	-14,19%	665	562	518	-7,89%
# Itemsets C1	7,25%	209	224	206	-12,43%
# Itemsets C2	36,50%	144	196	180	-5,19%
# Itemsets C3	12,89%	126	142	132	-4,14%
# No relevant itemsets	-	176	0	0	-
# Tolal class itemsets	17,57%	478	562	518	-7,89%
Adult					
# Itemsets	-14,60%	890	760	675	-11,18%
# Itemsets C1	2,8%	392	403	398	-14,53%
# Itemsets C2	43,33%	249	357	277	-7,82%
# No relevant itemsets	-	249	0	0	-
# Total class itemsets	18,56%	641	760	675	-11,18%
Connect-4					
# Itemsets	-11,04%	6543	5820	5184	-11,37%
# Itemsets C1	11,12%	2094	2328	2160	-15,80%
# Itemsets C2	14,15%	1439	2037	1657	-8,84%
# Itemsets C3	17,03%	1243	1455	1341	-7,82%
# No relevant itemsets	-	1767	0	0	-
# Total class itemsets	21,85%	4776	5820	5184	-11,37%
Cencus					
# Itemsets	-15,51%	780	659	615	-6,67%
# Itemsets C1	10,63%	328	362	335	-11,67%
# Itemsets C2	52,07%	195	297	280	-4,55%
# No relevant itemsets	-	257	0	0	-
# Total class itemsets	26%	523	659	615	-6,67%

TABLE II.7 – Tableau récapitulatif des résultats des 7 bases de données

-	**Taux de couverture**		**Temps d'exécution**		
BD	**FP-Growth**	**FCP-Growth FCP-Growth-P**	**FP-Growth**	**FCP-Growth**	**FCP-Growth-P**
D100T20I6	63%	91%	14	12	23
D100T20I2	75%	93%	11	9	20
Retail Data	67%	89%	2	2	6
Adult	69%	92%	12	10	18
Connect-4	73%	90%	9	6	14
Abalone	79%	96%	8	6	12
cencus	76%	92%	6	4	10

TABLE II.8 – Taux de couverture et temps d'exécution

5. Proposition d'un modèle à base de règles adapté au *Boosting* : phase de prédiction

5.1. Contribution 3 : prédiction à partir d'une base de règles significatives pondérées (W-CARP)

Cette phase se base sur le choix des règles à utiliser pour classifier. On distingue deux procédés distincts, des approches qui utilisent pour la prédiction une seule règles et d'autres qui utilisent plusieurs règles.

Prédiction à partir d'une règle Plusieurs travaux se réfèrent aux mesures de support et de confiance pour le choix de la meilleure règle, ainsi CBA qui utilise des heuristiques fondées sur la confiance, le support et la taille de l'antécédent, dans cet ordre (CAS). Cependant, d'autres critères peuvent être utilisés pour le choix de la meilleure règle, ainsi le critère ACS , qui place en tête la taille de l'antécédent, tout en ne différant de CAS que par l'ordre.

Prédiction à partir de plusieurs règles Pour la phase de prédiction, des approches se sont fondées sur la pondération de règles avec des mesures diverses telles que la mesure de Laplace pour CPAR ou le *Khi*2 pour CMAR.

- En effet, au lieu de se limiter à la prise en compte de la meilleure règle (celle ayant la confiance la plus élevée), CPAR sélectionne les K meilleures règles qui couvrent l'exemple et compare leur variance en se basant sur le théorème de Laplace pour choisir celle ayant l'exactitude la plus élevée.
- Pour CMAR, les auteurs ne sélectionnent pas les K règles mais collectent toutes les règles qui satisfont le cas et évaluent chaque règle avec le *Weighted Khi2*, un score qui donne une idée de l'indépendance de l'antécédent et du conséquent de la règle. En fait, CMAR sélectionne les règles avec une confiance élevée et analyse la corrélation entre ces règles.
- [Wang et al., 2008] ont l'idée d'utiliser une approche hybride pour le choix des règles de classification, afin d'intégrer à la fois le critère confiance-support et le poids en utilisant les mesures précédentes, Laplace et *Khi2*. Les expériences montrent que l'hybridation améliore la performance de la classification.

Construction d'une base de règles significatives

Lors de la phase de prédiction, afin d'améliorer la rapidité tout en gardant une bonne précision, nous proposons de choisir plusieurs règles pour la prédiction d'un nouvel exemple au lieu de la seule meilleure règle comme c'est le cas dans l'algorithme CBA, pour ensuite pondérer ces règles par une mesure qui prend en considération la distribution des classes. Le principe de construction de la base de règles significatives est décrit ci-dessous.

Une règle issue de la première phase n'est sélectionnée que si sa confiance est significativement supérieure à la probabilité *a priori* de la classe. Pour une règle, le test se présente de la façon qui suit. Etant donnée une règle $A \rightarrow c$, on note $\pi(c)$ la proportion théorique de la classe c et $\pi(C/A)$ sa confiance théorique. Il s'agit de tester l'hypothèse $H_0 : \pi(C/A) = \pi(c)$ contre l'hypothèse alternative $H_1 : \pi(C/A) > \pi(c)$, au moyen des $n(A)$ exemples de la règle considérée. En négligeant l'aléa sur la marge on peut approximer $\pi(c)$ par $P(c)$ et construire ce test comme le test de conformité de la confiance de la règle au support de la classe, unilatéral à droite. La statistique de test est alors $S = \frac{\sqrt{n(A)}(P(C/A)-P(C))}{\sqrt{P(C)(1-P(C))}}$qui suit la loi normale sous H_0. On rejette H_0 lorsque la valeur observée de S, notée S_{obs}, dépasse la valeur critique unilatérale à droite de la loi normale centrée réduite au risque α_0, notée $u_{1-\alpha_0}$. Le plus couramment, ce risque est pris égal à 0,05, 0,01 ou 0,001. On peut aussi calculer la p-value de ce test qui est donnée par $P(N(0,1) > S_{obs})$ et refuser H_0 lorsque $p - value < \alpha_0$.

Le test étant répété autant de fois qu'il y a de règles, on s'expose à une inflation de fausses découvertes. Pour contrôler celles-ci, on peut en première instance abaisser le risque α_0. De façon plus satisfaisante, on peut aussi utiliser la procédure de [Benjamini and Liu, 1999] qui consiste à examiner les m règles dans l'ordre croissant de leur $p - value$ et à refuser H_0 pour r_i, la i^e règle examinée dans l'ordre, dès lors que $p - value(r_i) > i\alpha_0/m$. Cette procédure a l'intérêt de contrôler le taux de fausses découvertes, FDR, tout en étant robuste face à une dépendance positive des données. La base de règles significatives ainsi obtenue est notée SRB.

Prédiction à partir des règles pondérées

La procédure de prédiction de W-CARP est la suivante :

- pour chaque exemple à prédire, on détermine les règles de la SRB qui couvrent cet exemple, d'où l'intérêt d'avoir un bon taux de couverture.
- lorsqu'il y a plusieurs règles qui couvrent un même exemple, on pondère ces règles par la valeur de leur mesure de Loevinger, notée $\frac{P(c_j/A)-P(c_j)}{P(\overline{c}_j)}$ pour une règle $A \rightarrow c_j$. L'intérêt de cette mesure, qui est compatible avec la stratégie d'élagage de Li [Li, 2006], comme montré dans [LeBras et al., 2009], est qu'elle avantage la classe minoritaire.
- pour chaque exemple à prédire, on calcule le score de Loevinger de chaque classe, qui correspond à la moyenne des sommes des valeurs de Loevinger de toutes les règles couvrant l'exemple considéré qui débouchent sur la classe considérée
- la classe prédite est celle qui maximise le score de Loevinger

Etude de la complexité de W-CARP

La complexité de la phase de prédiction de W-CARP est de l'ordre de *0(R)* puisque pour chaque exemple à prédire on accède à la SRB et on parcourt au maximum toutes les règles. On note que la complexité de CPAR pour cette phase est plus importante, puisqu'il doit

réordonner les règles pour déterminer les K meilleures règles débouchant sur chaque classe.

5.2. Performances de W-CARP

Pour évaluer la précision et l'efficacité de W-CARP, nous avons comparé ses résultats avec ceux de C4.5 [Quinlan, 1993] ainsi qu'avec ceux des principales méthodes de classification associative, CBA [Liu et al., 1998], CMAR [Li et al., 2001b], CPAR [Yin and Han, 2003]. Nous avons utilisé les mêmes 26 bases de données de l'UCI Machine Learning Repository [Stolfo et al., 1999] que celles utilisées dans [Liu et al., 1998] et [Li et al., 2001b]. Toutes les expériences sont exécutées sur une machine de 2.8 GHz Pentium-4 avec 1GO de mémoire.

Bases de données	Nb exemples	NB attributs	NB classes
ANNEAL	898	38	6
AUSTRAL	690	14	2
AUTO	205	25	7
BREAST	699	10	2
CLEVE	303	13	2
CRX	690	15	2
DIABETES	768	8	2
GERMAN	1000	20	2
GLASS	214	9	7
HEART	270	13	2
HEPATITIS	155	19	2
HORSE	386	22	2
HYPO	3163	25	2
IONO	351	34	2
IRIS	150	4	3
LABO	57	16	2
LED7	3200	7	10
LYMPH	148	18	4
PIMA	768	8	2
SICK	2800	29	2
SONAR	208	60	2
TIC-TAC	958	9	2
VEHICLE	846	18	4
WAVEFORM	5000	21	3
WINE	178	13	3
ZOO	101	16	7

TABLE II.9 – Caractéristiques des bases de données

Nous avons tout d'abord comparé W-CARP et les approches traditionnelles en se fondant sur le taux de bonne prédiction ou *accuracy*. Le tableau II.10 indique l'*accuracy* en généralisation des cinq approches pour les 26 bases de données, estimée à l'aide d'une validation

croisée en 10 segments. En ce qui concerne W-CARP, les paramètres utilisés pour cette étude sont définis comme suit :

- support de référence, $\sigma = 1\%$
- confiance = 70%
- $u_{1-\alpha_0} = 3$, ce qui correspond à un seuil voisin de 1 pour 1000

Bases de données	**C4,5**	**CBA**	**CPAR**	**CMAR**	**W-CARP**
ANNEAL	94,8	97,9	98,4	97,3	98,6
AUSTRAL	84,7	84,9	86,2	86,1	87,1
AUTO	80,1	78,3	82	78,1	79,9
BREAST	95	96,3	96	96,4	96,8
CLEVE	78,2	82,8	81,5	82,2	83,4
CRX	84,9	84,7	85,7	84,9	85,8
DIABETES	74,2	74,5	75,1	75,8	76,3
GERMAN	72,3	73,4	73,4	74,9	77,5
GLASS	68,7	73,9	74,4	70,1	75,2
HEART	80,8	81,9	82,6	82,2	83,8
HEPATITIS	80,6	81,8	79,4	80,5	80,1
HORSE	82,6	82,1	84,2	82,6	84,6
HYPO	99,2	98,9	98,1	98,4	99,1
IONO	90	92,3	92,6	91,5	92,8
IRIS	95,3	94,7	94,7	94	95,8
LABO	79,3	86,3	84,7	89,7	87,6
LED7	73,5	71,9	73,6	72,5	74,9
LYMPH	73,5	77,8	82,3	83,1	82,7
PIMA	75,5	72,9	73,8	75,1	75,9
SICK	98,5	97	96,8	97,5	98,1
SONAR	70,2	77,5	79,3	79,4	80,7
TIC-TAC	99,4	99,6	98,6	96	98,9
VEHICLE	72,6	68,7	69,5	68,8	79,8
WAVEFORM	78,1	80	80,9	83,2	84,5
WINE	92,1	95	95,5	95	96,7
ZOO	92,2	96,8	95,1	97,1	96,9
Moyenne	83,3	84,7	85,2	85,1	86,7

TABLE II.10 – Accuracy : C4.5, CBA, CPAR, CMAR, W-CARP

Pour approfondir notre étude sur la performance de W-CARP, nous avons choisi de comparer aussi la précision et le rappel, étant donné que ces deux mesures sont des mesures d'évaluation bien adaptées au déséquilibre des classes. On a choisi pour la comparaison C4.5, ainsi que CMAR et CPAR qui sont déjà meilleures que CBA. Le tableau II.11 affiche la précision des cinq approches sur les mêmes bases de données.

Les résultats montrent que notre approche, malgré la simplicité de l'étape de prédiction,

Bases de données	**C4.5**	**CPAR**	**CMAR**	**W-CARP**
ANNEAL	92,8	95,2	96,1	98,5
AUSTRAL	81,5	85,6	85,1	86,3
AUTO	72,1	77,4	74,3	79,1
BREAST	89,4	94,8	95,1	96,5
CLEVE	74,7	80,1	80,9	82,2
CRX	80,4	85,1	83,9	85,2
DIABETES	69,2	74,9	75,3	75,4
GERMAN	67,3	72,4	72,7	74,6
GLASS	63,7	71,6	68,9	74
HEART	71,8	77,9	77,8	82,3
HEPATITIS	71,6	75,5	78,9	79,6
HORSE	73,2	80,1	79,1	83,5
HYPO	95,2	97,3	97,6	98,5
IONO	87,2	90,4	89,9	92,5
IRIS	90,3	94,7	92,3	94,6
LABO	79,5	80,2	85,2	85,8
LED7	68,5	71,2	70,2	72,6
LYMPH	73,5	69,8	80,2	81,5
PIMA	68,5	72,1	71,6	74,1
SICK	90,5	92,4	94,1	97,8
SONAR	73,2	75,8	76,1	79,3
TIC-TAC	92,4	95,4	96,3	98,4
VEHICLE	65,6	66,7	66,5	68,3
WAVEFORM	78,1	80,1	80,9	83,3
WINE	92,1	92,5	92,4	95,4
ZOO	92,2	95,1	96,1	95,5
Moyenne	79,0	82,5	83,0	85,2

TABLE II.11 – Précision : C4.5, CPAR, CMAR, W-CARP

est au moins aussi efficace que les approches CPAR et CMAR puisque la précision moyenne des 26 bases de données est de 85.25% par rapport à 83,0% pour CMAR et 82,5% pour CPAR. W-CARP l'emporte 15 fois sur 26 avec un ex-aequo face à CMAR et 13 fois sur 26 avec un ex-aequo face à CPAR. Ce trés léger avantage de W-CARP n'est pas significatif. On remarque que les bases de données pour lesquelles notre approche donne de meilleurs résultats (ANNEAL, AUSTRAL, BREAST, HYPO, SICK, WAVEFORM) sont caractérisées par un nombre d'attributs important.

Bases de données	**C4.5**	**CPAR**	**CMAR**	**W-CARP**
ANNEAL	0,82	0,82	0,84	0,85
AUSTRAL	0,76	0,77	0,78	0,78
AUTO	0,62	0,65	0,66	0,67
BREAST	0,73	0,79	0,78	0,78
CLEVE	0,75	0,77	0,8	0,79
CRX	0,81	0,81	0,83	0,83
DIABETES	0,62	0,67	0,78	0,68
GERMAN	0,65	0,68	0,67	0,67
GLASS	0,61	0,69	0,7	0,68
HEART	0,82	0,85	0,89	0,87
HEPATITIS	0,65	0,68	0,71	0,69
HORSE	0,82	0,86	0,88	0,86
HYPO	0,8	0,82	0,85	0,84
IONO	0,84	0,87	0,89	0,88
IRIS	0,91	0,93	0,96	0,95
LABO	0,67	0,74	0,77	0,75
LED7	0,72	0,72	0,75	0,71
LYMPH	0,54	0,59	0,62	0,63
PIMA	0,65	0,65	0,64	0,64
SICK	0,85	0,84	0,85	0,81
SONAR	0,59	0,65	0,69	0,67
TIC-TAC	0,89	0,9	0,92	0,91
VEHICLE	0,69	0,71	0,74	0,72
WAVEFORM	0,73	0,74	0,71	0,7
WINE	0,85	0,86	0,88	0,87
ZOO	0,8	0,85	0,87	0,87
Moyenne	0,738	0,766	0,787	0,773

TABLE II.12 – Rappel : C4.5, CPAR, CMAR, W-CARP

Le tableau II.12 présente les résultats trouvés pour le rappel. On remarque que notre approche améliore aussi le rappel par rapport à C4.5 (gain de 3,5 points de pourcentage en moyenne, *p-value* de l'ordre de 10^{-5} ; 22 victoires sur 26, $p - value = 0.0005$) et à CPAR (gain de 0.7 point, $p - value = 0.0491$; 18 victoires contre 7 défaites, $p - value = 0.0433$). En revanche, le rappel de W-CARP est devancé par celui de CMAR (perte de 1.4 points, $p - value = 0.0035$; 3 victoires et 17 défaites, $p - value = 0,0026$), ce qui nuance l'avantage pris par W-CARP en termes de précision.

Pour étudier la rapidité de W-CARP, nous avons comparé son temps d'exécution avec celui de CMAR et CPAR. La figure II.6 montre que notre approche est beaucoup plus rapide que CMAR. En effet, on note un gain de temps qui varie entre 50% et 70%. Le gain par rapport à CPAR est moins important. En effet, l'exécution de notre phase de prédiction est voisine de

celle de CPAR.

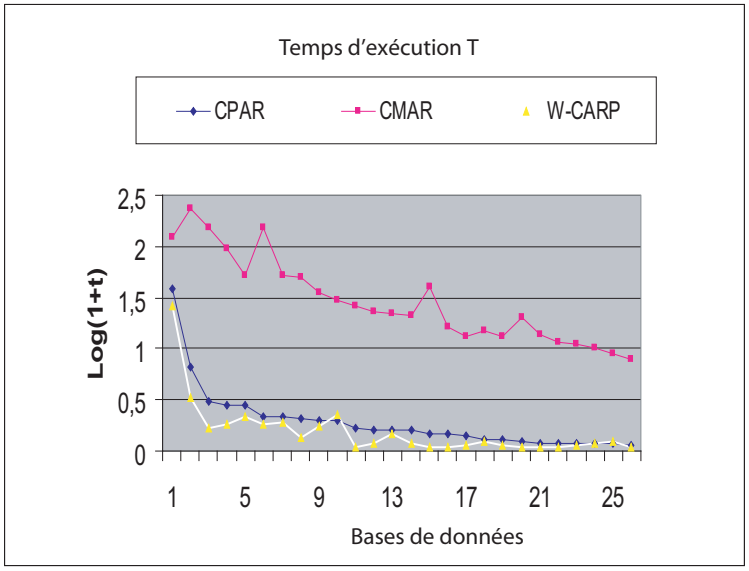

FIGURE II.6 – Comparaison de temps d'exécution de CMAR, CPAR et W-CARP

Afin de proposer un algorithme rapide et économe susceptible d'être le classfieur faible d'une procédure itérative, nous avons proposé W-CARP, un algorithme de classification associative. Pour la phase d'extraction de règles, cet algorithme utilise FCP-Growth-P, un algorithme supervisé de génération des itemsets de classe fréquents dérivé de FP-Growth dans lequel est introduit une condition d'élagage fondée sur les contre-exemples, pour contrôler la spécification des règles. Pour la deuxième phase celle de la prédiction, W-CARP utilise une base fixe de règles significatives, pondérées par la mesure de Loevinger. Les résultats obtenus montrent que notre approche est plus rapide que les approches traditionnelles, tout en assurant le meilleur taux de bonne prédiction et la meilleure précision et en étant légèrement devancé par CMAR pour le rappel. Ce travail est une première étape vers la conception d'une méthode de *Boosting* fondée sur la classification associative. La deuxième phase, celle de la construction d'une procédure adaptative inspirée du *Boosting*, fondée sur W-CARP, sera détaillée dans la section suivante.

6. Contribution 4 : CARBoost, un algorithme de classification associative adaptative

6.1. Intérêt des méthodes ensemblistes

En contrepoint de leur intelligibilité, les méthodes à base de règles présentent souvent des performances quelque peu inférieures à celles des autres algorithmes. Afin d'augmenter les performances de la classification associative, tout en veillant à conserver une bonne part de son intelligibilité, nous nous sommes tournés vers les méthodes ensemblistes.

Le théorème du Jury de Condorcet (1785) a montré que le vote à la majorité de plusieurs juges qui se prononcent indépendamment sur une alternative avec un même risque de se tromper inférieur à 0,5 permet de réduire considérablement le risque d'erreur du jury, jusqu'à être asymptotiquement nul.

Transposé en fouille des données, ce théorème signifie que l'agrégation de classifieurs (ou méthode ensembliste) permet de réduire considérablement le risque d'erreur à condition que les classifieurs soient suffisamment bons (risque d'erreur inférieur à 0.5) et suffisamment divers (au sens où leurs erreurs sont indépendantes). Toute la difficulté des approches ensemblistes est d'assurer cette diversité. Celle-ci peut provenir d'abord de l'utilisation de classifieurs hétérogènes. Sinon, en cas de relance d'un même algorithme, la diversité peut provenir notamment de la modification des paramètres de l'algorithme et/ou d'un aléa sur les individus par le biais d'échantillons bootstrap et/ou d'un aléa sur les attributs.

Plus formellement, les méthodes ensemblistes s'interprètent dans le cadre du compromis entre le biais des algorithmes d'apprentissage (erreur systématique qui ne dépend pas de l'échantillon d'apprentissage) et leur variance (qui provient de la variabilité des résultats issus de l'échantillon d'apprentissage). Le *Stacking* [Wolpert, 1992]) construit un méta-modèle de décision qui a pour but de minimiser le biais, alors que le *Bagging* [Breiman, 1996]) opère sur des échantillons *Bootstrap* de l'ensemble d'apprentissage pour réduire la variance sans trop augmenter le biais. [Freund and Schapire, 1996]) s'efforcent de réduire simultanément le biais et la variance en travaillant sur des échantillons bootstrap de l'ensemble d'apprentissage et en forçant le classifieur à se concentrer sur la prédiction des cas difficiles à prédire grâce à une repondération adaptative des cas, au risque de sur-apprendre en cas de données bruitées. Les Forêts Aléatoires [Breiman, 2001a]) combinent la construction d'arbres non élagués sur des échantillons bootstrap de l'ensemble d'apprentissage, qui diminue le biais, et la sélection au hasard des attributs qui participent à l'éclatement de chaque noeud de chaque arbre de la forêt, ce qui améliore la diversité des arbres de la forêt. Les Forêts Aléatoires sont ainsi une méthode de référence, rapide, très compétitive et robuste face au bruit.

Pour notre part, nous avons choisi de mettre en oeuvre une procédure adaptative inspirée du *Boosting* qui nous paraît bien correspondre à la classification associative, laquelle est capable de

fournir des règles couvrant peu d'exemples mais avec une très grande confiance. La procédure adaptive doit permettre à ces règles d'être valorisées.

A notre connaissance, peu de travaux ont appliqué les méthodes ensemblistes à la classification associative. On citera d'abord [Sun et al., 2006]), qui en appliquant Adaboost à des règles prédictives simples obtiennent de meilleurs résultats qu'en utilisant des règles complexes. [Yoon and Lee, 2008]) utilisent une approche équivalente au *Boosting* pour filtrer les règles d'association et s'adapter ainsi à la catégorisation de textes à grande échelle.

6.2. Pour une classification associative adaptative : CARBoost

L'idée initiale développée en apprentissage machine était d'améliorer les compétences d'un classifieur faible. La première proposition de Schapire [Shapire, 1990] a été affinée par Freund et Schapire [Freund and Schapire, 1996], qui ont décrit ADABOOST (Adaptative boosting), l'algorithme original du *Boosting* pour la prédiction d'une variable binaire. Notre approche CARBoost adopte le même principe général que l'agrégation adaptative de classifieurs : construction d'une famille de modèles qui sont ensuite agrégés par une moyenne pondérée des estimations ou un vote. Il diffère nettement sur le choix des apprenants ou hypothèses : à chaque itération, l'hypothèse est constituée par les règles de la base de règles significatives pondérées par la mesure de Loevinger, base qui sera considérée comme un modèle. Ce modèle est une version adaptative du précédent obtenue en donnant plus de poids aux observations mal prédites et aux règles qui classifient bien les exemples mal prédites lors de l'observation précédente. Intuitivement, cet algorithme concentre donc ses efforts sur les observations les plus difficiles à prédire tandis que l'agrégation de l'ensemble des modèles permet d'échapper au sur-ajustement.

Les poids des m exemples sont initialisés à $1/m$ pour l'estimation du premier modèle, puis évoluent à chaque itération, donc pour chaque nouvelle estimation. L'importance d'une observation w_i est inchangée si elle est bien classée, sinon elle croît proportionnellement au défaut d'ajustement du modèle. L'agrégation finale des prévisions, $\sum_{m=1}^{M} c_m \phi_m(x_0)$, est une combinaison pondérée par les qualités d'ajustement de chaque modèle. Sa valeur absolue appelée marge est proportionnelle à la confiance que l'on peut attribuer à son signe qui fournit le résultat de la prédiction.

Principes de CARBoost Le but poursuivi est la conception d'une méthode de classification associative qui bénéficie de l'apport des méthodes ensemblistes pour améliorer ses performances de classification tout en gardant au mieux la lisibilité de son principe de prédiction à base de règles. Cette méthode itérative, nommée CARBoost repose sur les principes suivants :

- le classifieur faible retenu est la base de règles significatives SRB produite par W-CARP. W-CARP produit une base de règles significatives qui permet de classer les exemples

avec des performances au moins aussi bonnes que CBA, CMAR et CPAR pour un temps d'exécution réduit.

- nous avons choisi une procédure adaptative inspirée du *Boosting* qui tient compte des erreurs de chaque itération pour forcer l'algorithme à se concentrer sur les exemples difficiles à prédire. A chaque itération, on travaille sur un échantillon bootstrap des exemples où le poids des exemples mal classés à l'itération précédente est accru.
- la SRB est construite une fois pour toutes, ce qui contribue à réduire la complexité de la méthode. Cependant, pour introduire de la diversité et favoriser la prédiction des exemples difficiles, à chaque itération, les règles sont repondérées de telle sorte que l'on favorise les règles qui ont prédit correctement au moins un exemple mal classé de l'itération précédente.
- la prédiction finale qui résulte du vote pondéré des classifieurs de base, permet d'échapper au sur-ajustement.

Les modifications d'une relance à l'autre sont donc dues à l'aléa sur les cas qui est issu de l'échantillon bootstrap, à la repondération des cas mal classés et à la repondération des règles efficaces sur au moins un exemple mal classé. Pour un nouvel exemple, d'une itération à l'autre, les règles qui le couvrent restent stables, de même que les classes qui sont les conséquents de ces règles, ce sont les caractéristiques et les pondérations de ces règles qui changent ainsi que le score de Loevinger associé à chaque classe pour l'exemple considéré. En effet, le bootstrap sur les individus, associé à l'évolution adaptative du poids de ceux-ci, modifie la mesure de Loevinger des règles. D'autre part, le poids des règles est modifié de façon adaptative pour favoriser les règles qui prédisent correctement au moins un exemple mal classé. La prédiction finale d'un nouvel exemple qui est obtenue par un vote pondéré des résultats des différentes itérations, se présente comme l'ensemble des listes de règles de la SRB couvrant l'exemple qui débouchent sur chacune des classes, chaque liste étant accompagnée d'un ensemble de poids optimisés qui résulte des repondérations précitées.

Pseudo code de CARBoost

Entrées :

-Base d'exemples

-Base de règles significatives obtenue à l'aide W-CARP. On génère les itemsets de classe fréquents à l'aide de FCP-Growth-P, en utilisant la condition d'élagage de Li, ce qui permet de ne générer que des itemsets de classe et de contrôler leur spécialisation. On filtre les règles obtenues au seuil de confiance préfixé pour ne garder que celles dont la confiance est significativement supérieure à la probabilité a priori de la classe qui figure en conséquent de la règle, constituant ainsi la SRB. Pour assurer la prédiction d'un nouveau cas, on pondère les règles qui couvrent ce cas avec la valeur de la mesure de Loevinger et on calcule le score de chaque classe. La classe prédite pour le nouveau cas est celle qui a le meilleur score de Loevinger.

[Procédure CARBoost]

Initialisation : on considère m cas qui ont tous le même poids $W_0 = 1/m$

[Soit T le nombre d'itérations]

Pour t **de** 1 **à** T **faire**

Construction d'un échantillon bootstrap des cas

Appel à SRB et application des règles pertinentes pour l'échantillon

Calcul du taux d'erreur ϵ_t en divisant le nombre d'erreurs de prédiction obtenues à l'aide du score de Loevinger par le nombre d'exemples

Calcul de l'exactitude $\alpha_t = 0.5 * Ln((1 - \epsilon_t)/\epsilon_t)$

Repondération des cas : on augmente le poids des cas mal prédits à l'itération t par le score de Loevinger

Si (le cas est mal prédit par le score de Loevinger) **Alors**

$W_{t+1} = W_t \exp^{\alpha_t}$

Fin Si

Si (le cas est bien prédit par le score de Loevinger) **Alors**

$W_{t+1} = W_t \exp^{-\alpha_t}$

Fin Si

Repondération des règles : on augmente le poids des règles qui classifient bien au moins un exemple mal classé à l'itération t. Si l'on désigne par $\epsilon(r, t)$ le taux d'erreur de la règle r à l'itération t, le poids de la règle r à l'itération t devient alors $\alpha_{r,t} = (1 - \epsilon(r, t))/\epsilon(r, t)$.

Fin Pour

Prédiction : La classe d'un nouveau cas est prédite en faisant voter les classifieurs faibles pondérés par les α_t

6.3. Les performances de CARBoost face aux méthodes usuelles

Pour évaluer l'efficacité de CARBoost, nous avons comparé expérimentalement son taux de bonne prédiction avec celui assuré par les principales méthodes de classification associative, à savoir CBA, CMAR et CPAR,plus W-CARP. En outre, pour servir de référence, nous avons inclus dans la comparaison deux autres méthodes à base de règles, à savoir C4.5, l'algorithme à base d'arbres de décision le plus populaire et RF, les Forêts Aléatoires [Breiman, 2001a]), qui sont considérées comme l'une des méthodes ensemblistes à base d'arbres de décision les plus performantes, tout en ayant perdu l'intelligibilité des arbres.

Cette comparaison porte sur les mêmes 26 bases de données de l'UCI Machine Learning Repository [Perner and Rosenfeld, 2003] que celles utilisées précédemment (2.6.1). Nous avons choisi pour ces expériences de pratiquer T= 10 itérations. En effet, c'est la valeur la plus couramment utilisée dans les procédures de type *Boosting-like*. C'est aussi celle choisie par [Sun et al., 2006]. Cette valeur sera discutée dans les prochaines sections. La comparaison est fondée sur le calcul des trois métriques d'évaluations (l'*accuracy*, la précision et rappel) de chaque méthode sur chaque base. Les tableaux II.13, II.14 et II.15 présentent le taux de bonne prédiction, la précision et le rappel des différentes approches sur les bases de données retenues.

Les résultats de point de vue taux de bonne prédiction montrent un avantage en faveur de CARBoost II.13.

Bases de données	**C4,5**	**CBA**	**CPAR**	**CMAR**	**W-CARP**	**RF**	**CARBoost**
ANNEAL	94,8	97,9	98,4	97,3	98,6	98,9	99,5
AUSTRAL	84,7	84,9	86,2	86,1	87,1	87,3	90,1
AUTO	80,1	78,3	82	78,1	79,9	82,3	81,9
BREAST	95	96,3	96	96,4	96,8	97,1	97,9
CLEVE	78,2	82,8	81,5	82,2	83,4	85,2	90,9
CRX	84,9	84,7	85,7	84,9	85,8	86,1	87,0
DIABETES	74,2	74,5	75,1	75,8	76,3	76,8	78,9
GERMAN	72,3	73,4	73,4	74,9	75,5	75,6	77,8
GLASS	68,7	73,9	74,4	70,1	75,2	79,4	78,5
HEART	80,8	81,9	82,6	82,2	83,8	84,7	86,8
HEPATITIS	80,6	81,8	79,4	80,5	80,1	86,4	87,2
HORSE	82,6	82,1	84,2	82,6	84,6	85,9	87,4
HYPO	99,2	98,9	98,1	98,4	99,1	98,7	97,8
IONO	90	92,3	92,6	91,5	92,8	92,9	95,3
IRIS	95,3	94,7	94,7	94	95,8	96,1	97,0
LABO	79,3	86,3	84,7	89,7	87,6	90,3	93,0
LED7	73,5	71,9	73,6	72,5	74,9	76,8	75,1
LYMPH	73,5	77,8	82,3	83,1	82,7	85,1	88,1
PIMA	75,5	72,9	73,8	75,1	75,9	77,5	78,1
SICK	98,5	97	96,8	97,5	98,1	98,8	97,7
SONAR	70,2	77,5	79,3	79,4	80,7	84,2	86,3
TIC-TAC	99,4	99,6	98,6	96	98,9	99,6	99,4
VEHICLE	72,6	68,7	69,5	68,8	79,8	74,2	76,8
WAVEFORM	78,1	80	80,9	83,2	84,5	82,8	82,9
WINE	92,1	95	95,5	95	96,7	95,4	96,4
ZOO	92,2	96,8	95,1	97,1	96,9	97,1	97,4
Moyenne	83,3	84,7	85,2	85,1	86,6	87,5	88,7

TABLE II.13 – Accuracy de C4.5, CBA, CPAR, CMAR, W-CARP, Forêts Aléatoires, CARBoost

Bases	**C4.5**	**CPAR**	**CMAR**	**RF**	**WCARP**	**CARBoost**
ANNEAL	92.8	95.2	96.1	98.8	98.5	99.5
AUSTRAL	81.5	85.6	85.1	86.7	86.3	88.2
AUTO	72.1	77.4	74.3	82.1	79.1	82.1
BREAST	89.4	94.8	95.1	97.3	96.5	98.2
CLEVE	74.7	80.1	80.9	83.5	82.2	86.7
CRX	80.4	85.1	83.9	85.9	85.2	86.8
DIABETES	69.2	74.9	75.3	75.7	75.4	81.9
GERMAN	67.3	72.4	72.7	73.8	74.6	83.9
GLASS	63.7	71.6	68.9	78.8	74.0	80.2
HEART	71.8	77.9	77.8	82.9	82.3	83.9
HEPATITIS	71.6	75.5	78.9	83.7	79.6	85.9
HORSE	73.2	80.1	79.1	84.5	83.5	85.7
HYPO	95.2	97.3	97.6	98.2	98.5	95.6
IONO	87.2	90.4	89.9	92.5	92.5	94.6
IRIS	90.3	94.7	92.3	95.5	94.6	95.4
LABO	79.5	80.2	85.2	87.5	85.8	88.9
LED7	68.5	71.2	70.2	74.5	72.6	73.9
LYMPH	73.5	69.8	80.2	83.8	81.5	85.4
PIMA	68.5	72.1	71.6	75.8	74.1	75.8
SICK	90.5	92.4	94.1	98.2	97.8	97.8
SONAR	73.2	75.8	76.1	82.0	79.3	83.6
TIC-TAC	92.4	95.4	96.3	99.3	98.4	99.8
VEHICLE	65.6	66.7	66.5	73.6	68.3	80.3
WAVEFORM	78.1	80.1	80.9	82.7	83.3	82.5
WINE	92.1	92.5	92.4	95.9	95.4	96.6
ZOO	92.2	95.1	96.1	96.8	96.5	98.2
Moyenne	79.0	82.5	83.0	86.5	85.2	88.1

TABLE II.14 – Precision de C4.5, CPAR, CMAR, Forêts Aléatoires, W-CARP, CARBoost

Pour s'assurer que les différences observées sont significatives, c'est-à-dire qu'elle ne sont pas le simple fruit du hasard (voir chapitre 1), nous avons d'abord utilisé le test apparié de Student pour comparer les précisions moyennes de deux méthodes sur les 26 mêmes bases. Nous avons redoublé ce test par un test non paramétrique pour échantillons appariés, le test du signe, qui teste la signification du résultat du match entre deux méthodes sur les 26 bases par comparaison avec la loi binomiale $B(26; 0.5)$ attendue sous l'hypothèse nulle (méthodes équivalentes). Les résultats de ces différents tests sont rapportés dans les tableaux II.16, II.17 où le niveau de signification des comparaisons est indiqué par *(significatif), ** (très significatif) ou *** (très hautement significatif) suivant que $0.01 < p - value < 0.05$, $0.001 < p - value < 0.01$ ou $p - value < 0.001$. On rappelle que la p-value d'un test est la probabilité d'obtenir une valeur de la statistique de test au moins aussi extrême, dans le sens de l'hypothèse alternative, que

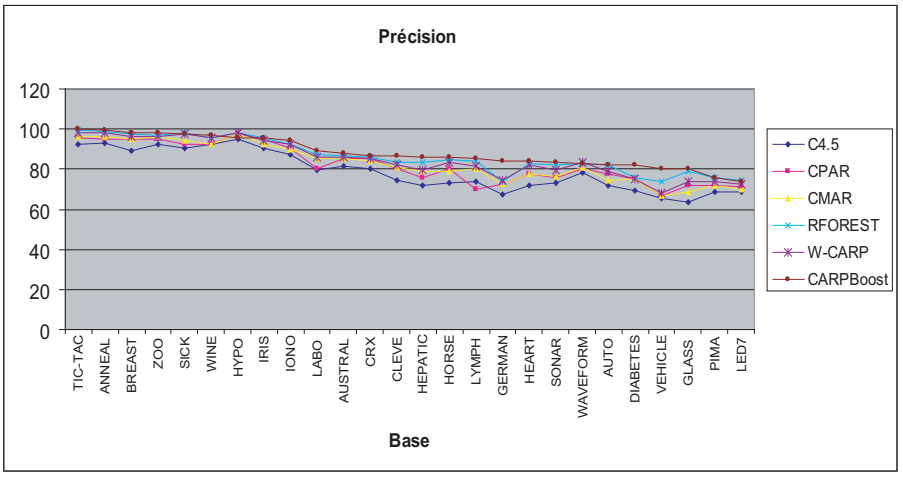

FIGURE II.7 – Précision

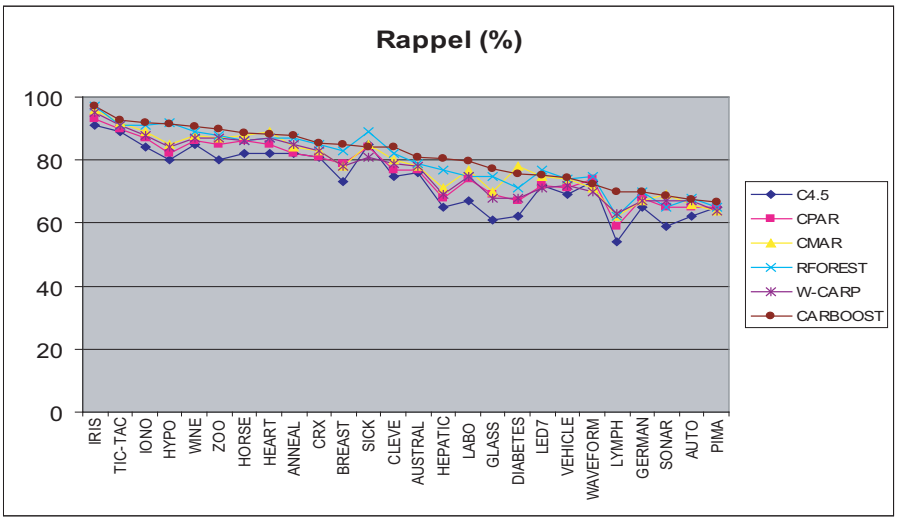

FIGURE II.8 – Rappel

celle qui a été effectivement observée, en supposant que l'hypothèse nulle est vraie.

Globalement, les résultats des expériences concernant la précision (tableaux II.14, II.16) conduisent à des conclusions très claires :

- par rapport à CPAR et CMAR, les méthodes usuelles de classification associative, CARBoost enregistre un gain supérieur à 5 points de précision qui est très hautement significatif (*p-values* du test apparié de Student de l'ordre de 10^{-7}). On notera que CARBoost l'emporte presque systématiquement sur les méthodes précitées (25 fois sur 26, ce qui est très hautement significatif selon la *p-value* du test du signe qui est de l'ordre 10^{-7}). Par rapport à W-CARP la méthode que nous avons élaborée pour être le classifieur faible de CARBoost,

Bases	**C4.5**	**CPAR**	**CMAR**	**RF**	**WCARP**	**CARBoost**
ANNEAL	82	82	84	87	85	89
AUSTRAL	76	77	78	79	78	81
AUTO	62	65	66	68	67	69
BREAST	73	79	78	83	78	85
CLEVE	75	77	80	82	79	83
CRX	81	81	83	85	83	86
DIABETES	62	67	78	71	68	75
GERMAN	65	68	67	70	67	71
GLASS	61	69	70	75	68	77
HEART	82	85	89	87	87	89
HEPATITIS	65	68	71	77	69	81
HORSE	82	86	88	86	86	89
HYPO	80	82	85	92	84	91
IONO	84	87	89	91	88	92
IRIS	91	93	96	97	95	97
LABO	67	74	77	75	75	79
LED7	72	72	75	77	71	74
LYMPH	54	59	62	62	63	69
PIMA	65	65	64	65	64	66
SICK	85	84	85	89	81	84
SONAR	59	65	69	65	67	69
TIC-TAC	89	90	92	91	91	93
VEHICLE	69	71	74	74	72	75
WAVEFORM	73	74	71	75	70	72
WINE	85	86	88	89	87	91
ZOO	80	85	87	88	87	90
Moyenne	73.4	76.6	78.7	80.0	77.3	81.4

TABLE II.15 – Rappel de C4.5, CPAR, CMAR, Forêts Aléatoires, W-CARP, CARBoost

le gain est à peine moins marqué (près de 3 points) tout en restant très hautement significatif (*p-value* de l'ordre de 10^{-5}) et en étant presque systématique (23 victoires et une égalité sur 26 bases, *p-value* de l'ordre de 10^{-5}).

- par rapport aux méthodes à base de règles prises comme référence, CARBoost améliore la précision de C4.5 de 9.1 points en moyenne (*p-value* très hautement significative, de l'ordre de 10^{-11}) et ne connaît aucune défaite sur les 26 bases testées (*p-value* de l'ordre de 10^{-11}). Comme attendu, le meilleur concurrent de CARBoost est l'algorithme des Forêts Aléatoires, mais CARBoost l'emporte quand même 19 fois sur 26 plus 2 égalités (*p-value* du test du signe égale à 0.0066), pour un gain moyen de 1.6 point, qui reste très significatif (*p-value* du test de Student égale à 0.0041).

En ce qui concerne le rappel, on trouve des résultats à peine moins marqués, très hautement

Précision	C4.5	CPAR	CMAR	RF	W-CARP
Avantage CARBoost	9,1	5,7	5,2	1,6	2,9
ratio de Student	11,46	7,29	7,91	3,16	4,73
p-value test Student	2E-11	1E-07	3E-08	0,0041	8E-05
signification	***	***	***	**	***
nb succès CARBoost	26	25	25	19	23
nb échecs CARBoost	0	1	1	5	2
nb égalités	0	0	0	2	1
p-value test signe	3E-08	8E-07	8E-07	0,0066	2E-05
signification	***	***	***	**	***

TABLE II.16 – Signification de la précision

Rappel	C4.5	CPAR	CMAR	RF	W-CARP
Avantage CARBoost	7,6	4,8	2,7	1,4	4,1
ratio de Student	8,29	7,89	4,72	2,94	8,44
p-value test de Student	1E-08	3E-08	0,0001	0,0070	9E-09
signification	***	***	***	**	***
nb succès CARBoost	24	24	21	21	26
nb échecs CARBoost	2	1	3	4	0
nb égalités	0	1	2	1	0
p-value Signe	1E-05	2E-06	0,0003	0,0009	3E-08
signification	***	***	***	***	***

TABLE II.17 – Signification du rappel

significatifs dans l'ensemble (Tableaux II.15, II.17) :

- CARBoost enregistre un gain de rappel de 4.8 points par rapport à CPAR avec 24 victoires sur 26 contre un gain de 2.7 points par rapport à CMAR avec 21 victoires sur 26 plus 2 égalités. Face à W-CARP, CARBoost améliore systématiquement le rappel, qui est augmenté de 4.1 points en moyenne.

- de la même façon, CARBoost gagne 7.6 points de rappel sur C4.5 en moyenne et l'emporte 24 fois sur 26. Face aux Forêts Aléatoires, le gain n'est que de 1.4 points mais il reste très significatif ($p-value = 0.0070$) et il se retrouve sur la plupart des bases, avec 21 victoires plus 1 égalité sur 26 ($p-value = 0.0009$)

6.4. Les performances de CARBoost face à Boost HPWR

Comme remarqué précédemment, peu de travaux ont appliqué les méthodes ensemblistes à la classification associative. Parmi ces travaux, on trouve l'article de [Sun et al., 2006] qui propose l'approche Boost HPWR. En effet, selon Sun, Adaboost donne des meilleurs résultats si on l'applique à des règles prédictives simples. Donc, le principe de Boost HPWWR est

d'appliquer à chaque itération la règle la plus simple parmi les règles extraites. [Sun et al., 2006] utilise comme apprenant faible pour le *Boosting* une seule règle et non pas une combinaison de règles comme dans notre cas. Nous avons comparé le taux de bonne prédiction de notre approche et celui de Boost HPWWR. Les tableaux II.18 et II.19 présentent les résultats obtenus et l'analyse statistique de ceux-ci. Les résultats donnent l'avantage à CARBoost (1 point en moyenne, 13 victoires contre 5 défaites), mais cet avantage n'est pas suffisant pour être significatif, les p-values étant de l'ordre de 0,10.

Bases de données	**Accuracy CARBoost**	**Accuracy Boost HPWWR**
ANNEAL	99,5	99,4
AUTO	81,9	80,3
BREAST	97,9	96,9
CLEVE	90,9	88,8
CRX	87,0	84,2
DIABETES	78,9	74,5
GERMAN	81,8	79,3
GLASS	78,5	74,6
HEPATITIS	87,2	92,8
HORSE	87,4	87,0
HYPO	97,8	99,2
IRIS	97,0	96,9
LABO	93,0	93,1
LYMPH	88,1	90,3
SICK	97,7	96,8
SONAR	86,3	86,0
VEHICLE	76,8	68,9
WAVEFORM	82,9	84,9
Moyenne	88,4	87,4

TABLE II.18 – Comparaison de CARBoost et Boost HPWWR

nb	18,0
écart-type	3,7
St ratio	1,63
nb succès CARBoost	13
nb échecs CARBoost	5
nb égalités	0
p-value Signe	0,0963

TABLE II.19 – Analyse des résultats

7. Analyse théorique et expérimentale de CARBoost

7.1. Motivations

L'objectif de cette section est de comprendre plus profondément le fonctionnement de CARBoost. Nous avons vu en section 3.3 qu'une façon optimale de combiner un ensemble de règles extraites consiste à résoudre le problème :

$$(P1')\ \{\hat{a}\}_0^M = \arg\min_{\{a\}_0^M} \sum_{i=1}^{n} L\left(y_i, a_0 + \sum_{m=1}^{M} a_m r_m(x_i)\right) + \lambda \sum_{m=1}^{M} \|a_m\|^l$$

Cela revient à chercher un hyperplan régularisé dans l'espace généré par les indicatrices des règles extraites. Différentes méthodes issues de la classification associative prédisent également à partir de combinaison linéaires dans l'espace des règles, mais elles sont basées sur des combinaisons plutôt intuitives et non rigoureusement définies.

Afin de résoudre $(P1)$, il est nécessaire de spécifier les valeurs de ses paramètres $L(.)$, l, et λ. La fonction de perte $L(.)$ doit être adaptée au problème de la classification supervisée. Toute fonction de perte en classification supervisée doit être fonction de la "marge" $yF(x)$. En effet, cette quantité est positive si la prédiction associée à un individu x_i et sa classe réelle y_i sont de même signe. Une fonction de perte très simple et intuitive peut être définie comme le taux d'erreur $L(y_i, F(x_i)) = I(y_i F(x_i) < 0)$, donnant une pénalité de 1 si l'exemple est mal classé et de 0 sinon.

[Hastie et al., 2001]) étudient les propriétés des fonctions de perte et leur robustesse et soulignent qu'une bonne fonction de perte doit pénaliser bien plus fortement les marges négatives ($y_i F(x_i < 0)$ représentant les individus mal classés que les marges positives puisque l'objectif de la classification supervisée est de produire l'apprenant le plus performant. Ils décrivent trois fonctions de perte adaptées à la classification supervisée :

- la fonction *hinge loss* : $L(y_i, F(x_i)) = max(0, 1 - y_i F(x_i))$
- la fonction de perte *exponentielle* : $L(y_i, F(x_i)) = exp(-y_i F(x_i))$
- la fonction de perte *binomial deviance* : $L(y_i, F(x_i)) = log(1 + exp(-2y_i F(x_i))$

Afin d'obtenir de bons résultats, notre stratégie se doit d'employer l'une des fonctions de perte précédentes. Concentrons-nous à présent sur le choix des paramètres p et l. Nous avons déjà évoqué les différences entre normes L_1 et norme L_2. Il nous semble bien plus pertinent d'utiliser la norme L_1 car le fait qu'il n'existe pas d'algorithme exact aboutissant à la résolution de ce type de problème nous paraît largement compensé par le fait que la solution réelle ne fasse intervenir qu'un petit nombre de variables, ce qui est notre cas. En effet, nous nous sommes concentrée sur l'extraction d'un ensemble de règles le plus petit possible. Nous pensons par conséquent que l'utilisation de la norme L_1 donnera de meilleurs résultats.

Nous avons vu en section 3.3 que le problème $(P1)$ défini avec $l = 1$ et $L(y_i, F(x_i)) = max(0, 1 - y_i F(x_i))$ est équivalent à LPBoost. Nous pourrions par conséquent utiliser cette

approche pour trouver une solution à $(P1')$. Toutefois, utiliser cette approche pose deux problèmes :

- LPBoost utilise l'algorithme de "génération de colonnes" pour trouver la solution. Cette approche itérative demande à chaque itération de résoudre un programme linéaire ce qui peut devenir long en temps de calcul ;
- il est nécessaire (comme pour toutes les autres approches) de fixer une valeur du paramètre λ. Cela nous contraint à lancer plusieurs fois l'algorithme LPBoost avec différentes valeurs pour ces paramètres, ce qui augmente encore le temps de calcul.

Afin de respecter les contraintes de performance et de temps de calcul, nous avons opté pour une modification du problème $(P1)$. Plutôt que de chercher directement une combinaison linéaire optimale des indicatrices de règles, nous allons générer plusieurs combinaisons linéaires possibles, puis agréger ces dernières de façon à obtenir une prédiction. Ainsi nous proposons de *booster* la combinaison linéaire des règles basée sur la mesure de Loevinger (W-CARP). Plusieurs points motivent cette approche :

- les règles extraites étant toutes significatives (au sens statistique), elle sont robustes et compte tenu du seuil de confiance retenu (0.70), il est fort probable que la plupart des combinaisons linéaires des règles constituent un classifieur faible ;
- la procédure de *Boosting* ne sera désormais effectuée qu'une seule fois ;
- l'interprétabilité est conservée ;
- nous allons voir que CARBoost est approximativement équivalent à $(P1)$ avec $l = 1$, $p = 1$ et $L(y_i, F(x_i)) = max(0, 1 - y_i F(x_i))$, mais remplace chaque règle r_m par une combinaison linéaire des règles r_m.

7.2. Comportement théorique de CARBoost

Afin d'obtenir une étude plus simple, nous considérerons le cas de la classification supervisée binaire.

Les méthodes d'agrégation que nous avons vues jusqu'à présent consistent toutes à associer un poids a_m à une règle r_m en fonction d'un indicateur statistique prédéfini. De plus, à chacun de ces poids a_t est associé une modalité de classe, et la modalité qui a la somme des poids maximale est donnée comme prédiction par le système. Dans le cas binaire, ce processus peut être défini plus rigoureusement de la façon suivante.

Considérons un ensemble de M règles $R = \{r_1, ..., r_M\}$ tel que $r_m(x) = 1$ si l'exemple x vérifie l'antécédent, 0 sinon. A chacune de ces règles est associé un poids $a_m = \varphi(r_m)$, où φ est une mesure d'intérêt. Par exemple, si φ est la confiance, elle vaudra $-1 \times conf(r_m)$ si $conf(r_m) < 0.5$, $1 \times conf(r_m)$ sinon. Bien qu'intuitive, cette façon de procéder n'est pas nécessairement optimale. En effet, cette technique d'agrégation ne serait optimale que si toutes les règles couvrant un même exemple étaient indépendantes (disjointes), ce qui n'arrive jamais

en pratique.

Comme nous l'avons vu, une meilleure alternative consisterait à chercher les coefficients a_t de sorte que l'on trouve une solution au problème $(P1')$. Considérons dans un premier temps que l'on cherche une solution plus globale en résolvant le problème suivant, pour l'une des fonctions de perte définies dans la section précédente :

$$(P2)\ \hat{F}(x) = \arg\min_{\{a\}_0^T} \sum_{i=1}^{n} L\left(y_i, F(x)\right)$$

Le problème $(P2)$ peut être résolu par des méthodes d'optimisation numérique type descente du gradient ou stratégie pas à pas en avant (*forward stagewise strategy* [Hastie et al., 2001]) dans la mesure où ces fonctions de perte sont toutes convexes. La stratégie pas à pas en avant consiste définir un ensemble de fonctions de base $b(x_i, \gamma) \in \Re$ paramétrées par γ dans lequel on ajuste un hyperplan de manière gloutonne. L'algorithme se décrit de la manière suivante :

1. $F_0(x) = 0$

2. Pour $t = 1$ à T

(a) Calculer

$(\beta_t, \gamma_t) = \arg\min_{\beta,\gamma} \sum_{i=1}^{n} L(y_i, F_{t-1}(x_i) + \beta b(x_i, \gamma))$

(b) $F_t(x) = F_{t-1}(x) + \beta_t b(x, \gamma_t)$

Comme souligné dans [Hastie et al., 2001], AdaBoost peut être vu comme une recherche gloutonne d'une pondération optimale des classifieurs $f_t(x)$ dans l'espace des "classifieurs faibles". La différence principale entre AdaBoost et la stratégie pas à pas en avant concerne le fait qu'à chaque itération d'AdaBoost, on ne cherche pas nécessairement le classifieur minimisant la fonction de perte. Considérons par exemple que l'on *booste* un arbre de décision à trois niveaux. Il serait nécessaire à chaque itération de trouver l'arbre minimisant la fonction de perte. Or un arbre de décision étant lui-même un algorithme glouton, il n'est pas certain que celui-ci soit le meilleur candidat. Comme souligné dans [Robnik-Sikonja, 2004]), ce point n'est toutefois pas d'une importance capitale et AdaBoost donne de bons résultats malgré tout.

[Hastie et al., 2001] ont également proposé de régulariser le classifieur employé à chaque itération d'AdaBoost (on parle de *shrinkage*) et ont obtenu de très bons résultats. Il est évident que la ligne de la stratégie pas à pas en avant n'est pas respectée dans le cas où on régularise le classifieur courant, ce qui renforce encore l'idée qu'AdaBoost n'est pas trop sensible à ce paramétrage.

Un point en revanche nécessaire concerne le fait que l'apprenant utilisé à chaque itération produise au moins un classifieur faible. Notre algorithme CARBoost entre pleinement dans le cadre introduit ci-dessus avec comme classifieur faible l'algorithme W-CARP. On peut penser que dans la mesure où chaque règle extraite par FP-Tree dépasse le seuil de confiance choisi (0.70 dans nos expériences de ce chapitre) et passe un test de signification sévère, toute combinaison linéaire basée sur W-CARP est au moins un classifieur faible. De plus,

sur toutes les bases testées, W-CARP atteint des résultats très encourageants. Dans le cas où les classes sont équilibrées, W-CARP est équivalent à une pondération des règles par la mesure de confiance et se révèle être à coup sûr un classifieur faible.

Une autre spécificité de CARBoost concerne la mise à jour adaptative des poids associés aux règles. Appliquer AdaBoost sur W-CARP aurait consisté à simplement calculer W-CARP sur chaque échantillon pondéré. Or, nous avons ajouté une procédure modifiant également itérativement le poids de chaque règle. Cette procédure a pour objectif de rapprocher W-CARP du classifieur linéaire optimal en augmentant le poids des règles qui ont bien classé au moins un exemple mal classé à l'itération précédente.

CARBoost peut ainsi être vu comme la recherche d'une combinaison linéaire optimale dans l'espace des combinaisons linéaires de règles. Plus formellement, CARBoost a pour objectif la recherche des coefficients β_t de sorte que :

$$(P3)\ \left\{\hat{\beta}\right\}_0^T = \arg\min_{\{\beta\}_0^T} \sum_{i=1}^{n} exp\left(-y_i \sum_{t=1}^{T} \beta_t f_t(x_i)\right)$$

où $f_t(x) = \sum_{m=1}^{M} a_m r_m$. Les coefficients a_m sont ici calculés à chaque itération, via la procédure W-CARP sur l'échantillon pondéré. Dans notre cas, chaque coefficient a_m équivaut au poids de la règle à l'itération courante divisé par le nombre de règles donnant la même conclusion que r_m.

Comportement asymptotique Le dernier paramètre ayant une importance capitale concerne le nombre d'itérations T choisies pour CARBoost. [Rätsch et al., 2001] remarquent qu'AdaBoost peut être vu comme une approximation du classifieur ensembliste infini ($T = \infty$) :

$$F(x) = \sum_{t=1}^{\infty} \beta_t f_t(x)$$

tel que le couple (β, f) est la solution optimale du problème $(P4)$ suivant :

$$(P4) \quad \min_{\beta_t \in \Re, f_t \in H} \sum_{t=1}^{\infty} \beta_t$$

$$s.c \quad y_i \left(\sum_{t=1}^{\infty} \beta_t f_t(x_i)\right) \geq 1 \qquad pour\ i = 1, ..., n$$

$$\beta_t \geq 0 \qquad pour\ t = 1, ..., \infty$$

H désignant ici l'ensemble des classifieurs faibles considérés. Dans le cas de CARBoost, H est l'espace des combinaisons linéaires des règles extraites par le FP-Tree.

Ainsi, AdaBoost peut être vu comme un algorithme glouton approximant la solution du problème $(P4)$. Les premières itérations du *Boosting* visent essentiellement à respecter la contrainte $y_i \left(\sum_{t=1}^{\infty} \beta_t f_t(x_i)\right) \geq 1$, autrement dit, à avoir une erreur de prédiction nulle

sur l'échantillon d'apprentissage. Les itérations suivantes se focalisent uniquement sur la minimisation de la norme L_1 des poids. Le fait qu'AdaBoost minimise implicitement la norme L_1 des poids est extrêmement intéressant dans la mesure où cela implique que peu de classifieurs seront nécessaires (propriété de la norme L_1). Ainsi, plus on ajoute d'itérations, plus le classifieur produit par AdaBoost maximise la marge en norme L_1. Maximiser la marge est généralement une bonne stratégie, excepté en présence de données bruitées. Dans ce dernier cas de figure, chercher le séparateur parfait à marge maximal est déjà du sur-apprentissage. Le nombre d'itérations contrôle donc l'importance de la maximisation de la marge relativement à la minimisation de la fonction de perte et agit donc comme la paramètre de régularisation.

7.3. Caractéristiques expérimentales

Impact du nombre d'itérations

Pour illustrer expérimentalement l'impact du nombre d'itérations de CARBoost sur la performance, nous avons analysé l'erreur en apprentissage et en généralisation sur 5 jeux de données de l'UCI.

Une première analyse de ces 5 bases de données montre que l'erreur en apprentissage est voisine de 0 dès $T = 10$ ou parfois $T = 20$, alors que l'erreur en généralisation décroît constamment jusqu'à $T = 50$, qu'elle commence à se stabiliser après cette valeur et qu'elle croît à nouveau à partir de $T = 200$. Toutefois, l'augmentation est peu marquée, et correspond à notre sens au fait que l'espace de représentation construit par les règles empêche l'algorithme de trop sur-apprendre (tableau II.9).

Complexité des règles utilisées par CARBoost

Selon [Sun et al., 2006], le *Boosting* ne donne ses meilleurs résultats que s'il utilise des règles simples de point de vue de la spécialisation (nombre d'items dans l'antécédent). Considérant ce point de vue, nous avons calculé le nombre moyen d'items existant dans l'antécédent des règles significatives générées par FCP-Growth-P. Le tableau II.20 indique pour chacune des 26 bases de données le nombre de règles de la SRB ainsi que le nombre moyen d'items de ces règles. A trois exceptions près, le nombre de règles est inférieur à 100, pour une moyenne de 63, alors que le nombre moyen d'items par antécédent de règle de classe est compris entre 1.7 et 2.2 suivant les jeux de données.

Les deux tableaux (II.21, II.22) détaillent les résultats trouvés sur BREAST et WAVEFORM, deux jeux de données choisis au hasard. Ces résultats montrent que les règles obtenues par CARBoost ne sont pas très complexes :

- BREAST, 699 exemples, 10 attributs ; 34 règles de classe significatives, 2,1 items en moyenne dans l'antécédent, 85% des règles ont au plus 2 items

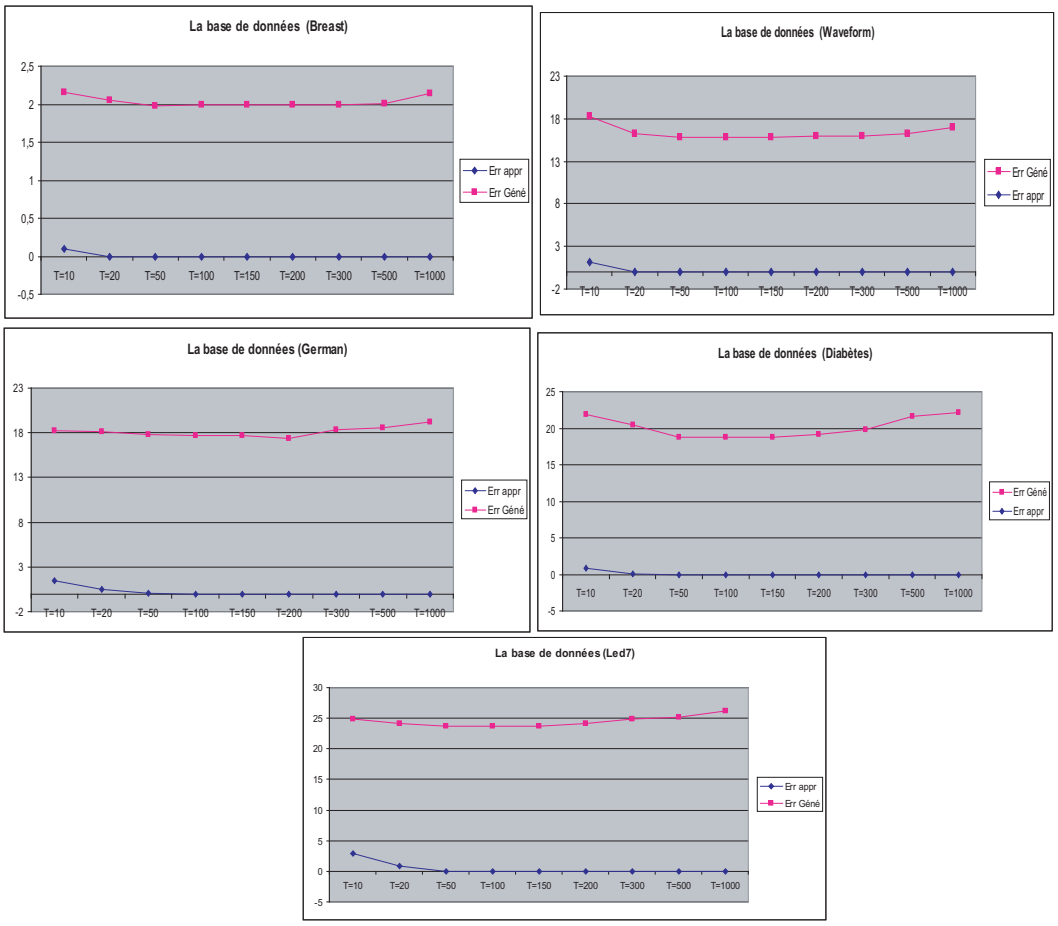

FIGURE II.9 – Evolution erreur en généralisation et en apprentissage selon T

– WAVEFORM, 5000 exemples, 21 attributs ; 342 règles de classe significatives, 2,2 items en moyenne, 70% des règles ont au plus 2 items

Interprétabilité des règles générées par CARBoost

L'une des principales qualités d'un ensemble de règles est de rendre un modèle intelligible. En effet, les règles de la forme "*Si (condition) alors (conclusion)*" sont directement interprétables et permettent à un être humain de déterminer les paramètres les plus influents du processus de classification. Dans notre cas, nous pouvons classer les règles selon une mesure de qualité de ces dernières. Par exemple, nous pouvons ordonner les règles extraites par ordre décroissant de leur mesure de Loevinger. Cette phase permet à un utilisateur d'avoir un aperçu immédiat sur les règles individuelles les plus discriminantes.

Toutefois, dans le cas de CARBoost, les règles sont combinées par un processus plus complexe. Parce que celui-ci permet généralement d'obtenir de meilleures performances que

Bases de données	Nombre de règles	nombre d'items moyens
ANNEAL	47	1,9
AUSTRAL	35	1,8
AUTO	54	2
BREAST	34	2,1
CLEVE	32	1,9
CRX	26	1,7
DIABETES	19	2,1
GERMAN	110	2,3
GLASS	59	2
HEART	75	1,9
HEPATITIS	38	1,9
HORSE	79	2,1
HYPO	159	2,2
IONO	35	1,9
IRIS	15	1,8
LABO	16	1,8
LED7	91	2,1
LYMPH	17	2
PIMA	80	2,2
SICK	75	2,1
SONAR	82	2,1
TIC-TAC	30	1,9
VEHICLE	43	2
WAVEFORM	342	2,2
WINE	26	2,1
ZOO	19	2
Moyenne	63	2,00

TABLE II.20 – Nb moyen d'items des règles sur les 26 bases de données

	xi	ni	nixi
1 Items	1	19	19
2 Items	2	10	20
3 Items	3	6	18
4 Items	4	4	16
	Nbre total de règles	34	73
	nb moy item		2,1

TABLE II.21 – Nb moyen d'items des règles sur Breast

le classifieur de base (non *boosté*), nous pensons qu'il peut être intéressant d'essayer de comprendre le processus de classification inhérent à CARBoost. Pour cela, rappelons qu'à l'issue des itérations de CARBoost, le classifieur global obtenu est une combinaison linéaire

	xi	ni	nixi
1 Items	1	149	149
2 Items	2	90	180
3 Items	3	44	132
4 Items	4	29	116
5Items	5	16	80
6Items	6	9	54
7items	7	5	35
	Nbre de régles	342	746
	nb moy item		2,2

TABLE II.22 – Nb moyen d'items des règles sur Waveform

de classifieurs qui sont eux-mêmes des combinaisons linéaires de règles, soit une combinaison linéaire de règles.

Considérons tout d'abord une combinaison linéaire de règles, autrement dit, un des T classifieurs faibles construits au fur et à mesure des itérations. L'importance d'une règle dans le processus de classification est directement proportionnelle à son poids, et la mesure de qualité d'une règle r_k peut donc simplement s'écrire comme étant le poids de cette règle, noté β_k.

Le classifieur global construit par CARBoost étant une combinaison linéaire de classifieurs, l'importance du classifieur faible issu de l'itération t est proportionnelle au poids de ce classifieur dans le classifieur global, c'est-à-dire à α_t. Dans ce contexte, on peut supposer qu'une règle est importante pour le processus de classification si (1) cette règle a généralement des poids élevés au sein des classifieurs faibles f_t et (2) cette règle a des poids forts lorsqu'elle est associée à un classifieur ayant lui-même un poids fort. En définitive, la mesure d'importance d'une règle r_k dans le classifieur global s'écrit :

$$Imp(r_k) = T^{-1} \sum_{t=1}^{T} \beta_{kt} \alpha_t \tag{II.1}$$

où β_{kt} représente le poids associé à la règle r_k à l'itération t. On voit aisément que cette mesure consiste simplement à calculer la somme des poids affectés à chaque règle au fur et à mesure des itérations. Chaque poids est pondéré par l'importance globale du classifieur $f_t(x)$. De cette façon, il est facile de retrouver les règles ayant la plus grande participation à la décision finale, et par conséquent, les plus importantes aux yeux de CARBoost.

Exemple illustratif : HEPATITIS Nous avons choisi le jeu de données HEPATITIS pour illustrer le comportement de cette mesure d'intérêt relativement à la mesure de Loevinger. Nous avons donc dans un premier temps extrait les règles pertinentes à l'aide de FCP-Growth-P. A l'issue de cette première phase, nous obtenons un ensemble de 33 règles de décision, pour chacune desquelles nous avons calculé la mesure de Loevinger. Nous avons

ensuite lancé le processus de classification CARBoost en utilisant 10 itérations et nous avons calculé l'importance de chaque règle dans le classifieur final. La figure II.10 compare les mesures d'importance associées à chacune des 33 règles au départ (mesure de Loevinger) et à la fin des itérations de CARBoost.

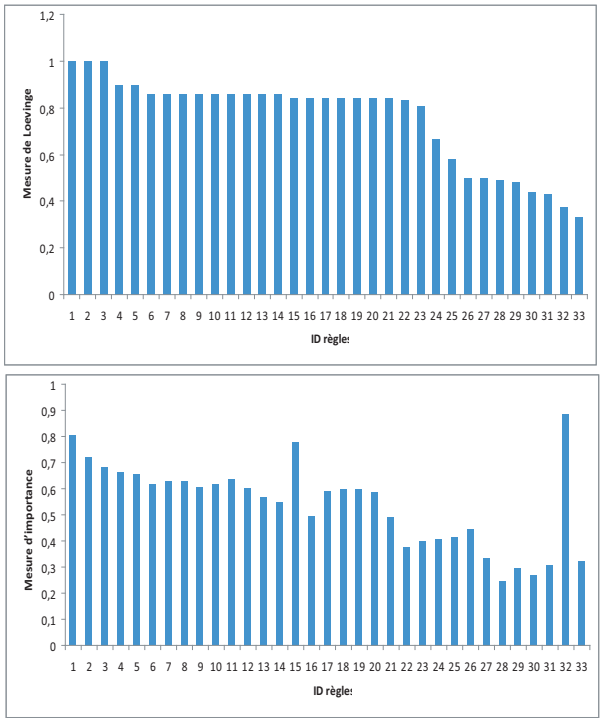

FIGURE II.10 – Mesure de Loevinger et mesure d'importance du *Boosting* associée à chaque règle.

Globalement, on observe que la mesure d'importance issue de CARBoost suit une tendance assez similaire à celle de la mesure de Loevinger. En effet, à l'exception de la règle 32, les règles 22 à 33 ont à la fois les mesures d'importance les plus faibles et les mesures de Loevinger les plus faibles. En revanche, on observe que trois règles ont des poids relativement importants selon CARBoost. Il s'agit des règles 1, 15 et 32. Un mesure d'importance forte associée à la règle 1 n'est pas forcément étonnante dans la mesure où cette règle est également celle qui a la mesure de Loevinger la plus forte. En revanche, des fortes importances associées aux règles 15 et 32 sont plus surprenantes et donc plus intéressantes.

Afin de comprendre cette observation surprenante, il est nécessaire de bien avoir en mémoire le processus de classification issu de la phase itérative de CARBoost. A chaque itération, les exemples mal classés à l'itération précédente voient leurs poids augmenter alors que ceux qui sont bien classés ont des poids qui diminuent. Ainsi, au fur et à mesure des itérations, les règles prenant des poids importants sont celles qui classent bien les exemples difficiles à classer,

Item ID	Frequence
1	2
4	2
6	8
7	1
9	2
10	1
11	5
12	6
13	7
25	2
29	2
30	5
34	1
36	2
52	1
53	2

TABLE II.23 – Frequence d'items

d'autant plus que CARBoost favorise les règles qui ont bien prédit au moins un exemple mal prédit de l'itération précédente.

Notre intuition est que les règles classant bien les exemples difficiles à classer sont les règles les plus rares et les plus spécifiques. En effet, ces règles couvrent des exemples sans doute proches de la "vraie" frontière de décision et se concentrent donc sur des zones où peu de règles peuvent être extraites. Afin de valider un peu plus notre intuition, nous avons calculé la fréquence d'apparition de chaque item extrait dans au moins une règle. Le tableau II.23 rassemble les résultats correspondants. Comme nous pouvons le voir, certains items comme 9, 36 et 53 n'apparaissent qu'une seule fois, i.e, ne sont dans l'antécédent que d'une seule règle. D'autres items sont eux aussi peu fréquents (apparaissant 2 fois en tout) : 7, 10, 34 et 52. Nous avons ensuite voulu déterminer si les items apparaissant rarement sont associés aux règles ayant des mesures d'importance forte selon CARBoost. Les règles 1, 15 et 32 ont les antécédents suivant :

$$R1 : \quad \{25, 34, 53\}$$

$$R15 : \quad \{9, 29, 52\}$$

$$R32 : \quad \{4, 6, 7\}$$

Ainsi, la règle 1 contient 2 items qui sont peu représentés. La règle 15 contient les items 9, 29 et 52 qui sont très peu représentés et enfin, la règle 32 contient la liste d'items 4, 6 et 7, c'est-à-dire deux items peu représentés. Ces règles couvrent donc des zones de l'espace

très peu couvertes par les autres règles. Ainsi, donnent-elles à CARBoost la possibilité d'être utilisées pour prédire correctement les points situés dans ces zones difficiles à catégoriser. Ceci peut être intéressant dans la mesure où de telles règles sont sans doute très importantes pour l'utilisateur. En effet, ce sont les règles permettant de trancher entre les classes pour les exemples difficiles à classer. Elles contiennent donc sans doute une information primordiale, utile pour la compréhension du phénomène étudié.

Application de CARBoost sur deux exemples de la base HEPATITIS

Pour bien comprendre le fonctionnement de CARBoost, nous avons analysé son fonctionnement sur deux exemples de la base HEPATITIS. Le premier exemple a été choisi de façon à montrer l'impact de CARBoost sur la précision, alors que le second illustre l'impact de CARBoost sur le rappel. Nous avons examiné l'évolution des résultats des différentes itérations de CARBoost pour ces deux exemples, concernant en particulier les poids respectifs de ces deux exemples, les décisions les concernant, ainsi que le poids des règles qui les couvrent. Nous rappelons brièvement les caractéristiques de la base de données HEPATITIS. Cette base de données contient 155 individus qui sont décrits par 20 variables descriptives (âge, sexe et variables médicales) ainsi que par le pronostic (variable à prédire) qui peut avoir deux modalités (DIE ou LIVE). La classe DIE qui contient 32 individus (20.6%) est la classe d'intérêt qui est minoritaire, alors que la classe LIVE contient 123 individus (79.4%).

Exemple pour la précision L'exemple que nous avons choisi est l'individu n° 104 de la base HEPATITIS qui a comme étiquette la classe LIVE. On trouvera la description de cet exemple dans le tableau II.24.

Attribut	Valeur
age :	51
sex :	1
steroid :	1
antviral :	2
fatigue :	1
malaise :	1
anorexia :	1
liver_big :	2
liver_firm :	1
spleen_palpable :	1
spiders :	1
ascites :	2
varices :	1
bilirubin :	4,6
alk_phosphate :	115
sgot :	269
albumin :	3,9
protime :	51
histology :	2

TABLE II.24 – Exemple 1

Attribut	Valeur
age :	34
sex :	1
steroid :	1
antviral :	2
fatigue :	1
malaise :	1
anorexia :	2
liver_big :	1
liver_firm :	1
spleen_palpable :	2
spiders :	1
ascites :	2
varices :	2
bilirubin :	2,8
alk_phosphate :	127
sgot :	182
albumin :	val manq
protime :	val manq
histology :	1

TABLE II.25 – Exemple 2

Parmi les règles de la SRB construite par CARBoost, il y en a 3 qui couvrent cet exemple, les règles n°9, 29 et 32 dont la signification est indiquée ci-dessous. Les caractéristiques de ces règles sont indiquées dans le tableauII.26

$$R9 : \{6, 12, 33\} \rightarrow \{55\}$$
$$R29 : \{4, 12, 36\} \rightarrow \{55\}$$
$$R32 : \{4, 6, 8\} \rightarrow \{56\}$$

En utilisant le schéma de normalisation de la base HEPATITIS, on trouve que :

$R9 : \{sex = Masculin, fatigue = non, 4 < BILIRUBIN \leq 6\} \rightarrow \{DIE\}$

$R29 : \{49 < Age \leq 63, fatigue = non, 79 < AlkPHosphate \leq 133\} \rightarrow \{DIE\}$

$R32 : \{49 < Age \leq 63, sex = Masculin, steroid = non\} \rightarrow \{LIVE\}$

Notre exemple est donc couvert par deux règles qui le classifient en tant que DIE et une règle qui le classifie en tant que LIVE. Lors de la première itération de CARBoost, il est prédit à tort comme étant de la classe DIE.

La figure II.11 retrace l'évolution des poids des règles R9 et R29 (aboutissant à la classe DIE) et le poids de la règle R32 (aboutissant à la classe LIVE). Cette figure montre qu'aux itérations 1, 2 et 6 l'exemple est classé DIE puisque la moyenne des poids des règles R9 et

Règles	Confiance	Support	Support Ant.	Support Cons.	Loevinger
R_9	87,5	7	8	20,65	0,842
R_29	88,9	8	9	20,65	0,860
R_32	89,2	83	93	79,35	0,479

TABLE II.26 – les caractéristiques des règles couvrant le 1er exemple

R29 qui votent pour cette classe est plus importante que le poids de la règle R32. Aux autres itérations, l'exemple est classé correctement LIVE car le poids de la règle R32 l'emporte sur le poids moyen des règles R9 et R29. Le classifieur final de CARBoost le prédit correctement comme étant de la classe LIVE.

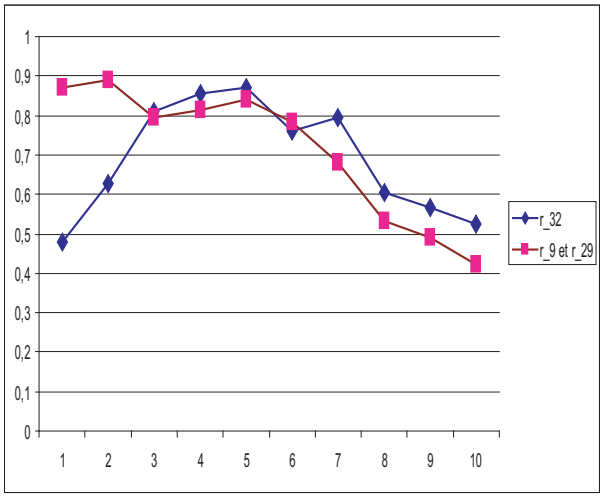

FIGURE II.11 – Poids des règles pour chaque classe couvrant l'exemple.

Le tableau II.27 affiche pour chaque itération les valeurs numériques des poids des règles et de l'exemple, ainsi que la décision et le poids du classifieur courant.

La figure II.12 montre l'évolution du poids de l'exemple tout au long des 10 itérations. On remarque qu'aux itérations 3 et 5 le poids de l'exemple n'a pas augmenté, ce qui montre que l'exemple est bien prédit par le classifieur (vote pondéré des règles qui le couvrent) et que son poids n'augmente pas à l'itération d'aprés. On remarque aussi qu'à partir de l'itération 7 le poids décroit donc l'exemple est bien classé.

Exemple pour le rappel Dans ce cas, nous avons choisi l'exemple n° 72 du jeu de données HEPATITIS. Cet exemple appartient à la classe DIE (classe minoritaire). On trouvera ses caractéristiques dans le tableau II.25.

Parmi les règles de la SRB construite par CARBoost, il y en a 4 qui couvrent cet exemple, les règles n°2 ,24, 28 et 33, dont la signification est indiquée ci-dessous. Les caractéristiques de

	préd min	**préd min**	**Sc DIE**	**préd major**	**Sc LIVE**	**Poids Ex**	**Pred**	**P Clfer**
Itér	poids R9	poids R29		poids R32		wi	1 ou 0	alpha i
1	0,842	0,897	0,870	0,479	0,479	0,012	1	65,50
2	0,969	0,807	0,888	0,627	0,627	0,025	1	71,80
3	0,859	0,726	0,793	0,811	0,811	0,025	0	68,67
4	0,911	0,713	0,812	0,855	0,855	0,052	1	71,80
5	0,838	0,845	0,842	0,871	0,871	0,052	0	81,50
6	0,805	0,760	0,783	0,759	0,759	0,123	1	85,57
7	0,724	0,634	0,679	0,794	0,794	0,118	0	80,00
8	0,645	0,421	0,533	0,604	0,604	0,109	0	81,50
9	0,612	0,366	0,489	0,565	0,565	0,100	0	73,26
10	0,496	0,348	0,422	0,525	0,525	0,090	0	81,65

TABLE II.27 – Poids de l'exemple, des règles, des classifieurs à chaque itération

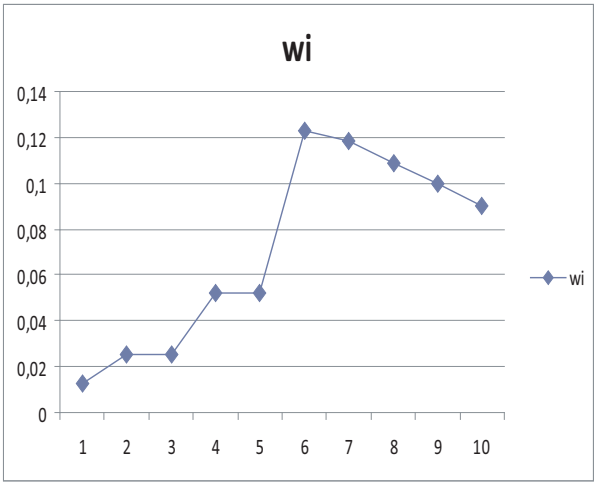

FIGURE II.12 – Poids de l'exemple dans chaque itération.

ces règles sont indiquées dans le tableauII.28

$$R2 : \{13, 53\} \rightarrow \{56\}$$
$$R33 : \{6, 13, 18\} \rightarrow \{56\}$$
$$R28 : \{2, 12, 36\} \rightarrow \{55\}$$
$$R24 : \{6, 11, 12, 31\} \rightarrow \{55\}$$

Règles	Confiance	Support	Support Antéce	Support Cons	Loevinger
R2	100	32	32	79,35	1
R33	97,9	46	47	79,35	0,897
R24	87,5	7	8	20,65	0,842
R28	88,9	8	9	20,65	0,860

TABLE II.28 – Les différentes caractéristiques des règles couvrant l'exemple.

	préd min	**préd min**	**Sc DIE**	**préd maj**	**préd maj**	**Sc LIVE**	**P Ex**	**Pred**	**P Clfer**
Itér	R_28	R_24		R_33	R_2		w_i	1 ou 0	α_i
1	0,860	0,842	0,851	0,897	1	0,948	0,008	1	65,50
2	0,896	0,877	0,886	0,877	0,976	0,927	0,015	1	71,80
3	0,921	0,894	0,908	0,866	0,959	0,913	0,025	1	68,67
4	0,934	0,935	0,934	0,823	0,913	0,868	0,043	0	71,80
5	0,859	0,879	0,869	0,715	0,885	0,800	0,033	0	81,50
6	0,816	0,967	0,891	0,610	0,762	0,686	0,030	0	85,57
7	0,763	0,947	0,855	0,584	0,693	0,638	0,029	0	80,00
8	0,687	0,900	0,793	0,542	0,513	0,528	0,028	0	81,50
9	0,584	0,990	0,787	0,466	0,436	0,451	0,027	0	73,26
10	0,560	0,891	0,726	0,448	0,4277	0,438	0,026	0	81,65

TABLE II.29 – Poids des règles de l'exemple à chaque itération.

$$R2 : \{FATIGUE = oui, HISTOLOGY = non\} \rightarrow \{LIVE\}$$

$$R33 : \{SEX = Masculin, FATIGUE = oui, LIVER\_BIG = non\} \rightarrow \{LIVE\}$$

$$R28 : \{21.2 \leq AGE < 35.4, FATIGUE = non, 79.8 \leq ALK_PHOSPHATE < 133.6\} \rightarrow \{DIE\}$$

$$R24 : \{SEX = Masculin, ANTIVIRALS = oui, FATIGUE = non, BILIRUBIN < 3.38\} \rightarrow \{DIE\}$$

L'évolution du poids de ces règles ainsi que le poids de l'exemple est décrit par le tableau II.29. Le tableau donne aussi le score pour chaque classe à chaque itération. La prédiction de l'appartenance de l'exemple dépend de ces scores calculés à partir des poids des règles. La prédiction finale est déterminée par le résultat donné à chaque itération pondéré par le poids de classifieur α_i (présenté dans la dernière colonne). La figure II.13 montre l'évolution des scores de chaque classe calculés à partir des poids des règles couvrant l'exemple.

A partir des résultats donnés par les scores, le poids de l'exemple varie. Si l'exemple est mal prédit, son poids augmente, sinon il diminue. La figure II.14 montre la diminution du poids de l'exemple à partir de l'itération 4 où il était bien prédit.

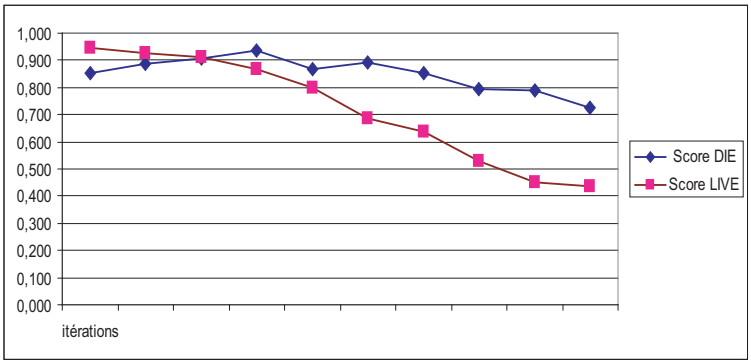

FIGURE II.13 – Evolution du score de chaque classe pendant les 10 itérations.

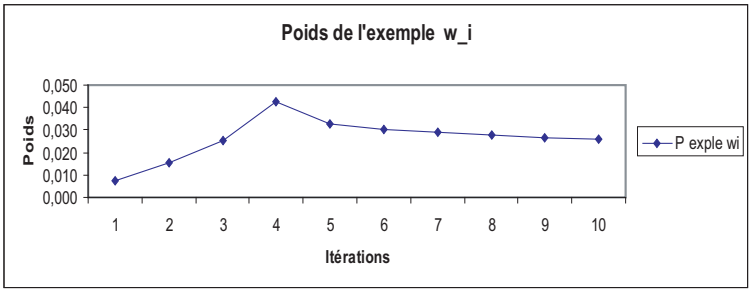

FIGURE II.14 – Poids de l'exemple selon la prédiction de chaque itération.

8. Conclusion et perspectives

Parmi les méthodes à base de règles, la classification associative a l'intérêt d'être une méthode de prédiction qui a déjà de bonnes performances, meilleures que celles des arbres. Les Forêts aléatoires et autres méthodes ensemblistes à base d'arbres comme le *Bagging* d'arbres ont permis d'améliorer les performances des arbres, mais le prix à payer est la perte d'intelligibilité des résultats obtenus.

Dans ce chapitre, nous avons proposé CARBoost, une version adaptative de la classification associative, inspirée du *Boosting*, qui tout à la fois améliore les performances de la classification associative et conserve l'essentiel de son intelligibilité. CARBoost permet de faire émerger les règles capables de prédire correctement les exemples difficiles. Les performances de CARBoost, tant en termes de précision moyenne que de rappel moyen sont très améliorées aussi bien par rapport à C4.5, que par rapport à CPAR et CMAR, les méthodes de classification associative les plus couramment utilisées. La supériorité de CARBoost sur les Forêts Aléatoires est un peu moins marquée, mais elle demeure très significative, de l'ordre de 1.5 point, aussi bien en rappel qu'en précision et les résultats de CARBoost restent intelligibles au contraire de ceux des Forêts Aléatoires. En outre, les améliorations indiquées ont un aspect systématique, car elles se retrouvent avec plus ou moins d'intensité sur la quasi totalité des bases testées.

Nous envisageons de poursuivre ce travail en testant diverses repondérations des règles lors des différentes itérations de la procédure et en intégrant des critères d'évaluation supplémentaires. Des expériences sur données réelles compléteront utilement celles déjà réalisées.