Conclusion et perspectives

1. Conclusion

Durant nos travaux de recherche, nous avons traité deux problèmes majeurs rencontrés par les méthodes d'apprentissage supervisé face aux données du monde réel. Ces problèmes sont d'une part le bruit, d'autre part le déséquilibre des classes. Nous avons choisi d'apporter des solutions à ces problèmes dans le cadre des méthodes adaptatives, connues pour leur bon pouvoir prédictif.

Face au bruit, nous avons proposé tout d'abord Glouton Boost, puis AdaBoost Hybride, une version améliorée d'AdaBoost M1, qui utilise un double effet de lissage afin d'améliorer la performance du classifieur à chaque itération. L'étude comparative effectuée sur plusieurs bases de données montre la bonne performance d'AdaBoost Hybride du point de vue pouvoir prédictif (*accuracy*, précision, rappel et surtout rapidité de la vitesse de convergence) tant sur des données bruitées que sur des données saines. Les résultats trouvés sont confirmés lors de l'utilisation d'AdaBoost Hybride sur les données réelles KDD-Cup 1999 dans l'étude de cas du chapitre III consacrée à la détection d'intrusions. Afin de valider notre nouvelle approche, nous avons étudié d'un point de vue théorique le comportement d'AdaBoost Hybride.

Face aux déséquilibre de données, nous avons élaboré une méthode adaptative de classification associative inspirée du Boosting. Tout d'abord, nous proposons comme algorithme de base (*weak learner*), W-CARP, un algorithme de classification associative économe de point de vue temps d'exécution et espace de stockage, qui tient compte du déséquilibre des classes. Cet algorithme repose sur FCP-Growth-P, une variante de FP-Growth qui permet de n'extraire que les itemsets de classe tout en intégrant la condition d'élagage de Li et un support adaptatif qui tient compte de la taille de la classe. Une fois constituée la base des règles de classe significatives, W-CARP opère la prédiction d'un nouvel exemple en pondérant les règles de classe qui couvrent l'exemple considéré, ce qui permet de fabriquer un score.

CARBoost, la méthode de classification associative que nous proposons utilise W-CARP comme classifieur faible. En gardant la base de règles initiale, elle met plus de poids sur les exemples mal classés de l'itération courante, qui sont souvent les exemples de la classe minoritaire, ainsi que sur les règles qui ont bien prédit au moins un exemple mal classé, ce qui permet d'optimiser le système de poids associé aux règles de classe, comme le montre l'étude théorique pratiquée. Les expériences pratiquées sur 26 bases de données de l'UCI montrent une supériorité très significative de CARBoost en termes de performances prédictives (*accuracy*, précision, rappel et F-mesure). Les exemples traités montrent bien que CARBoost est une méthode adaptative qui, tout en assurant d'excellentes performances, permet de conserver l'intelligibilité des règles propre à C4.5 ou à la classification associative, ce qui n'est ni le

cas du boosting, ni celui des forêts aléatoires. Appliqué aux données d'intrusions de KDD-Cup 1999, CARBoost fait la preuve de sa grande capacité à prédire correctement les classes les plus minoritaires sans détériorer la prédiction des classes les plus nombreuses.

2. Perspectives

Les perspectives que nous envisageons pour la suite de nos travaux de recherches peuvent être organisées selon leur caractère théorique ou applicatif.

2.1. Perspectives théoriques

Un premier projet concerne l'étude théorique d'AdaBoost Hybride, notre proposition face aux données bruitées. Il s'agit d'améliorer l'analyse théorique d'AdaBoost Hybride. Nous avons montré que celui-ci converge vers une solution stable étant donné un échantillon d'apprentissage de taille finie. Comme nous l'avons souligné, la propriété la plus désirable d'un système de décision automatique réside dans sa capacité à réduire au maximum l'erreur en généralisation. Les résultats expérimentaux suggèrent qu'AdaBoost Hybride est un procédé puissant pour obtenir un classifieur ayant une faible erreur en généralisation. Ainsi, il apparaît intéressant de mener une réflexion sur l'existence de bornes de l'erreur en généralisation pour AdaBoost Hybride. Ceci nous permettrait de nous comparer à d'autres techniques pourvues de bornes (par exemple AdaBoost, les Forêts aléatoires) et d'identifier les facteurs (taille de l'échantillon, dimensionalité, etc.) à même d'agir différemment sur la borne. Tout ceci nous permettrait d'attribuer des classes de compétence aux différents algorithmes et nous amènerait ainsi vers la sélection d'une méthode d'apprentissage appropriée au domaine étudié. Cela pourrait également nous guider vers d'autres procédés de lissage des poids susceptibles d'être plus performants que celui d'AdaBoost Hybride.

Une autre projet concerne l'étude théorique de CARBoost, notre contribution face aux données déséquilibrées. Nous envisageons d'approfondir notre étude théorique en traitant le cas où la variable de classe a plusieurs modalités (multi-classes). En effet, dans le chapitre II, nous avons étudié le comportement théorique de CARBoost dans le cas binaire. Il nous parait intéressant d'étudier le comportement de CARBoost et de le valider théoriquement dans le cas multi-classes. Le passage au cas multi-classe n'est pas évident en raison de la complexité des calculs.

Une importante perspective est la synthèse d'AdaBoost Hybride et CARBoost, en proposant une version de CARBoost qui intègre le principe de lissage qui est à la base d'AdaBoost Hybride. Cette synthèse est possible puisque AdaBoost Hybride introduit un effet de lissage dans la version standard AdaBoost, en construisant des classifieurs à mémoire à chaque itération et n'agit que sur les pondérations associées aux exemples et aux classifieurs de chaque itération, alors que CARBoost n'agit que sur l'apprenant faible en essayant de l'améliorer face au déséquilibre des données avec une meilleure combinaison possible des règles et donc n'intervient que sur les pondérations internes des règles constituant le classifieur à chaque itération. De ce fait, nous pensons que la synthèse des deux approches AdaBoost Hybride et CARBoost peut aboutir à une approche puissante de point de vue performance tant face au bruit que face au déséquilibre des classes, puisqu'elle hérite des avantages de chacune des approches testées.

2.2. Perspectives applicatives

Lors de l'implémentation des méthodes que nous proposons, nous avons préféré utiliser des implémentations déjà existantes dans des logiciels de Data mining gratuits, à savoir AdaBoost M1 sur Weka et FP-Growth sur LUCS KDD pour gagner du temps de programmation en utilisant des modules et des fonctions déjà codés. Cependant, dans le chapitre III, en essayant d'appliquer nos méthodes sur des données du monde réel de très grande taille ($LS-10\%$ de KDD-Cup 1999), nous avons rencontré des difficultés pour utiliser les deux logiciels.

- Avec LUCS KDD : une première difficulté a été rencontrée dès la première étape de numérisation du jeu de données $LS-10\%$ de KDD-Cup 1999. En effet, pour utiliser les algorithmes codés sur LUCS KDD, les fichiers doit être sous format .NUM [Coenen, 2003a]. Cette transformation de fichier est obligatoire. La numérisation n'a pas pu être effectuée à cause de la taille du fichier, qui était de 75M0. Sachant que LUCS KDD est implémenté sous Java, il fonctionne avec la machine virtuelle Java. Pour éviter que la mémoire soit saturée, nous avons augmenté la capacité maximale de la taille de la machine virtuelle jusqu'au maximum, 6GO. Mais l'exécution de la numérisation du fichier s'arrête. Donc pour pouvoir exécuter notre programme, nous avons réduit les données à un échantillon de 5% de $LS-10\%$
- Avec Weka, le chargement du format du fichier LS-10% de KDD-Cup 1999 est réussi. Le temps demandé par Weka pour charger le fichier KDD.arff est de 7 secondes. Cependant, on rencontre un problème de saturation de mémoire lors de l'exécution d'AdaBoost M1 (avec C4.5 et 10 itérations). Nous avons alors augmenté la taille de la machine virtuelle au maximum, 6GO. Mais le problème persiste encore. De ce fait, nous avons procédé comme pour LUCS KDD et nous avons restreint nos données à l'échantillon de 5% de $LS-10\%$ déjà évoqué.

Pour faire face à ces contraintes, nous envisageons de créer notre propre plateforme en nous fixant les exigences suivantes :

1. Un temps de traitement le plus faible possible. Cela concerne le temps d'importation des données et le temps d'exécution du programme.
2. Une occupation de mémoire réduite? Celle-ci peut être obtenue en agissant sur :
   - la mémoire occupée par le logiciel avant le lancement de l'éxécution. Par exemple, Weka utilise 52 MO lors son lancement
   - la mémoire occupée après l'importation des données. Par exemple, le logiciel KNIME utilise un excellent procédé permettant de "swapper" les données sur le disque, réduisant ainsi l'occupation mémoire)
   - la mémoire occupée lors du traitement et après la validation
3. Une rapidité du calcul accrue. Cette exigence dépend du choix du langage de programmation. Par exemple les programmes sous Java sont les plus lents, alors que TANAGRA augmente la rapidité du calcul en utilisant Delphi
4. le recours à un format de fichier facile à utiliser pour un utilisateur non informaticien, car ARFF de Weka et NUM de LUCS KDD ne sont pas très commodes, il faut disposer de programmes de conversion
5. Une meilleure adaptation des programmes aux différents problèmes qui peuvent être traités. En effet, la modification des programmes codés dans les logiciels libres de fouilles de données tels

que WEKA ou LUCS KDD est trés difficile et nécessite une maitrise des techniques dans le but d'adapter ces programmes pour traiter des problèmes bien spécifiques. Ce problème est abordé dans [Flouvat et al., 2006]. Afin de contourner les problèmes d'adaptation des programmes existants pour la recherche des motifs fréquents, les auteurs proposent une librairie implimentée sous C++ qui utilise d'algorithmes et de structures de données génériques transparents pour le programmeur, et seule l'implantation des propriétés spécifiques à son problème est laissée à la charge de l'utilisateur. On peut s'inspirer de cette étude pour proposer des algorithmes qui peuvent être adaptés à la résolution des problèmes bien spécifiques