Utilisation des méthodes d'apprentissage Ensembliste dans le Datamining distribué

Mokeddem Djamila, Belbachir Hafida *
** LSSD Département d'Informatique, USTO*
{Mokdjamila, H_belbach} @yahoo.com

1. Introduction

Le Datamining Distribué est un axe de recherche qui a suscité beaucoup d'intérêts dans la littérature ces dernières années [17]. C'est le processus du datamining classique qui consiste à extraire une information non triviale « nouvelle connaissance» à partir de sources de données distribuées, avec le minimum d'interaction entre les sites de données [2]. En plus de cet aspect distribué qu'il faut prendre en considération, cette démarche a hérité aussi du datamining classique deux défis majeurs: augmenter le taux de précision et diminuer le temps de calcul .

Ce processus a été utilisé en générale dans deux perspectives différentes. La première perspective est la fouille de données intrinsèquement distribuées où les données doivent être fouillées dans leur site, à cause de plusieurs contraintes comme le coût de stockage, de communication, de calcul et sécurité. La deuxième est le besoin de préserver l'efficacité des algorithmes en présence d'un très grand volume de données, dans ce cas, les données peuvent être partitionnées et distribuées sur différents sites, afin d'effectuer le processus du datamining simultanément, et sur des parties de données moins volumineuse.

Les méthodes du datamining (ou apprentissage) ensemblistes présentent actuellement des techniques très sollicitées dans la communié scientifique, grâce à leur apport significatif en terme de précision. L'idée est basée sur la combinaison des résultats de plusieurs algorithmes du datamining, appliqué chacun sur un ensemble diversifié de données. A première vue, cette stratégie de travail d'ensemble pourrait correspondre aux contraintes du datamining distribué, mais plusieurs pistes restent encore à explorer. En effet, dans les premiers travaux liés à ces approches, les algorithmes sont basés sur un même ensemble de données 'perturbées' par des méthodes ad hoc. Dans le datamining distribué, les données représentent en générale un partitionnement horizontal d'une base de données globale. Des adaptations sont alors nécessaires pour tirer profit des approches ensemblistes dans la fouille de données distribuées.

La classification, comme une tâche qui répond au besoin d'un large type d'applications, nécessite d'être au coeur du datamining distribué, en particulier les algorithmes d'arbres

de décision. Nous présentons dans cet article les différentes approches qui ont été utilisées, nous mettons le point particulièrement sur les approches des méthodes ensemblistes, et comment elles ont été appliquées dans ce domaine.
Le reste de l'article est organisé comme suit : La section 2 donne un aperçu sur le datamining distribué. La section 3 présente les concepts de base liés aux méthodes ensemblistes, ainsi que leur application dans la classification. La section 4 expose les approches les plus connues qui s'intéressent au datamining distribué par les techniques d'apprentissage ensemblistes. La conclusion est une discussion sur les pistes qui méritent toujours d'être explorées.

2. Le Datamining Distribué

A partir de la littérature liée au datamining distribué, on peut repérer deux techniques utilisées: la technique du parallélisme qui fait souvent appel à des machines dédiées et des outils de communication entre les processus parallèles, et une autre technique que nous appelons « technique d'agrégation », qui procède avec une vision purement distribuée, soit sur les données, soit sur les prédictions ou bien sur les modèles de base.

2.1 Techniques Parallèles

Ces techniques se basent sur l'extraction de l'aspect parallèle de l'algorithme utilisé, permettant ainsi d'exécuter plusieurs parties de l'algorithme en parallèle, sur différents processeurs. Ceci nécessite en générale des architectures dédiées comme les systèmes à mémoire partagée. Les grilles de calcul [37] représentent actuellement une alternative intéressante comme plate forme du calcul parallèle.
Le parallélisme des algorithmes du datamining peut être conçu selon deux approches principales, selon qu'on parallélise les données ou les tâches [2]. Le parallélisme par données correspond à diviser les données sur P processeurs, chaque processeur exécute un même traitement sur sa portion locale des données. Dans le parallélisme des tâches, les processeurs accomplissent des traitements différents indépendamment les uns des autres. En pratique, ces techniques nécessitent une intention particulière sur deux facteurs essentiels : l'équilibrage de charge entre les différents processeurs utilisés dans le calcul parallèle, et la charge de communication sur le réseau. Cette communication est souvent nécessaire pour l'échange de données ou de résultats partiels entre les processeurs.
Les techniques des arbres de décision sont très favorables au parallélisme, à cause de leur nature « diviser et régner». Elles ont fait l'objet de plusieurs travaux, comme *SLIQ parallèle* [24], *SPRINT Parallèle* [14] et *RainForest* [10]. Ces systèmes tentent d'optimiser le temps de calcul par l'utilisation de structures de données particulières, et le recours au parallélisme.

2.2 Techniques d'agrégation

D'autres techniques -que nous appelons « agrégation», sont utilisées dans le but de fouiller des données issues de plusieurs bases distribuées. Elles peuvent être classées selon que l'algorithme du datamining utilise une partie des données disponibles, dans ce cas on parle d'*Agrégation de données*-, ou la totalité des données, qui correspond soit à l'*agrégation de modèles* ou l'*agrégation de prédictions* [23]:

Agrégation de données : L'échantillonnage consiste en la création d'un échantillon représentatif d'une large base de données sous l'hypothèse qu'un algorithme de datamining entraîné sur cet échantillon n'aura pas de résultats significativement pires qu'un algorithme entraîné sur toute la base de données. Dans le contexte du datamining distribué, l'échantillonnage est appliqué sur chaque base répartie, générant des échantillons distincts dans chaque site. Ces derniers sont regroupés afin d'entraîner un seul algorithme.

L'algorithme d'apprentissage est appliqué sur une partie des données distribuées, qui peut être soit des échantillons issus de chaque base [40], ou bien quelques bases bien choisies [44]. L'inconvénient de l'agrégation de données est le temps de transfert ainsi que le temps de traitement pour choisir les bons échantillons.

Agrégation de modèles : Dans le cas de classification par exemple, des règles de classification sont construites, ensuite regroupées afin de produire un ensemble de règles unique. Bien qu'elles utilisent la totalité des données, ces techniques peuvent être appliquées sur des échantillons de données bien choisis [30]. On cite comme exemple de systèmes d'agrégation de modèles le travail de Hall et al. [28] ; celui ci produit un ensemble de règles de classification en parallèle à partir d'ensembles disjoints de données, ensuite les règles sont unies en un seul système. Ces auteurs proposent un ensemble de techniques de résolution de conflits entre les règles produites. Cette même approche a été utilisée dans [29], mais avec des algorithmes différents qui sont appliqués localement pour produire les règles.

Agrégation de prédiction : Les techniques d'agrégation de prédictions consistent à construire un ensemble de classificateurs de base –dans le cas de la tâche de classification- dont les prédictions seront combinées afin de classer de nouvelles données dont la classe est inconnue. Ces techniques appelées *Méthodes d'apprentissage ensembliste* (*ensemble learning*) ont été conçues au début pour être appliquées sur un seul ensemble de données, dans le but d'améliorer les performances de prédiction.

Malgré que ces techniques ne produisent que des modèles prédictifs qui ne peuvent expliquer le choix de leurs classifications à un analyste [23], elles présentent une alternative prometteuse pour le datamining distribué. Ceci est dû aux facteurs suivants :

− Leur apport en terme d'augmentation de la précision,

− L'aspect distribué qu'elles offrent,

− L'élimination de la phase de construction d'un classificateur global, comme dans le cas d'agrégation de modèles, pourrait accélérer le processus du datamining.

3. La classification par les méthodes ensemblistes

La classification est une des tâches du datamining qui permet de prédire si une instance de donnée est membre d'une classe prédéfinie. Elle utilise un ensemble *S* de données appelées ensemble d'apprentissage. Chaque donnée est typiquement représentée sous forme d'un vecteur d'attributs $x = <x_1, x_2, ..., x_m, y>$ avec *y* un attribut de classe. L'objectif de la classification est d'entraîner un algorithme de classification *A* sur l'ensemble *S*, pour trouver une bonne approximation d'une certaine fonction $f(x)= y$. La fonction approximative *Cl* calculée est appelée classificateur. L'évaluation de la précision de *Cl* est faite sur un ensemble de données *T* indépendant de *S*, appelé ensemble de test. Le classificateur sera par la suite capable de prédire la valeur de classe *y* pour de nouvelles données *d*, en calculant *Cl(d)*.

Dans le cas des méthodes ensemblistes, *N* classificateurs de base Cl_i sont construits, à partir de *N* ensembles de données S_i. Le classement d'une nouvelle donnée se fait par la combinaison des prédictions des *N* classificateurs de base, par un vote majoritaire par exemple. Malgré la simplicité de cette idée intuitive « l'union fait la force », elle repose sur une théorie statistique [19] renforcée par plusieurs études empiriques.
Ces études ont montré dans différents travaux de recherche [20, 22, 42, 43, 13] que la précision d'un algorithme d'apprentissage peut être améliorée d'une façon significative en appliquant le principe de perturbation et combinaison [19]. Les algorithmes les plus appropriés à l'application de cette approche sont ceux considérés comme non stable, c-a-d que des petites modifications dans les données d'apprentissage pourrait induire à un grand changement dans la fonction *Cl* estimée. Les arbres de décision par exemple sont considérés comme de bons candidats [19].
Cette perturbation permet de générer plusieurs ensembles d'apprentissage, à partir d'un ensemble de base, comme dans les techniques de boosting et bagging. Elle peut aussi être appliquée sur les algorithmes de construction des classificateurs, en utilisant plusieurs algorithmes différents, ou en modifiant certains paramètres [34]. Les résultats expérimentaux montrent que 50 répliques sont en générale suffisantes [19], mais le temps de calcul est encore un champ d'investigation.
On présente dans ce qui suit, des approches parmi les plus répandues dans la génération des ensembles d'apprentissage, ainsi que les techniques de combinaison. Des travaux récents relatifs aux arbres de décision seront aussi présentés.

3.1 Génération des ensembles d'apprentissage

Afin d'aboutir à divers ensembles d'apprentissage qui seront utilisés pour construire les classificateurs de base, plusieurs techniques peuvent être appliquées, parmi les plus utilisées, on présente les techniques: *bagging*, *boosting*, *comité de validation croisée*, et *les méthodes de sous espaces aléatoires*.

Bagging [20] (Boostrap AGGregatING) : Pour construire chacun des classificateurs de base, la diversité des ensembles d'apprentissage est obtenue par un choix aléatoire par remplacement, à partir d'un ensemble d'apprentissage d'origine. L'échantillon choisi par une technique statistique appelée «bootstrap» est de même taille que l'ensemble d'origine. Avec la réplication de certains exemples, l'échantillon contient en moyenne 2/3 de l'ensemble de données de départ. Les prédictions des modèles de base sont ensuite combinées par la méthode de vote à majorité (cf. 3.2).

Boosting [33] **:** L'idée de base est de construire un nouveau classificateur, selon la performance d'une série de classificateurs précédents, dans un processus séquentiel. L'ensemble d'apprentissage d'origine est renforcé par des poids qui seront ajustés à chaque étape, dans l'objectif 'd'amplifier' (boost) les exemples mal classés. Les poids des exemples bien classés -par le dernier modèle construit- seront alors décrémentés, et les poids des exemples mal classés seront incrémentés, en permettant ainsi au système de 'prêter plus d'attention' aux exemples mal classés. Les modèles sont combinés par vote à majorité pondéré (cf. 3.2) où la pondération est déterminée par la précision de prédiction de chaque classificateur. *Adaboost* [41] est un exemple d'algorithme utilisant cette technique.

Comité de validation croisée [38]**:** Cette technique consiste à diviser l'ensemble S en k parties disjointes $\{S_1, S_2, \ldots S_k\}$. Le processus suivant est répété k fois : construction d'un classificateur Cl_i avec l'ensemble S privé de S_i $(S\text{-}S_i)$, ensuite évaluer la précision de Cl_i testé sur S_i. La précision globale est obtenue par la moyenne. Les ensembles construits de cette manière sont appelés 'cross validated commitees'.

Méthode de sous espaces aléatoires (Random Subspace method) [39] : Cette technique consiste à choisir aléatoirement un certain nombre d'attributs -par des méthodes spécifiques-, les sous ensembles d'apprentissage obtenus seront alors utilisés pour construire les classificateurs de base. C'est une approche qui est très bénéfique pour les problèmes ayant un grand nombre d'attributs, avec de multiples redondances.

3.2 Méthodes de combinaison

Une fois les classificateurs de base construits, différentes techniques peuvent être utilisées pour combiner les résultats de chaque classificateur. On présente quelques unes parmi les plus citées dans la littérature : le vote majoritaire, le vote pondéré, stacking.

Le vote à majorité : C'est une technique simple et intuitive, qui consiste à classer la nouvelle instance selon la prédiction majoritaire des classificateurs de base. L'inconvénient de cette méthode est dans le cas ou plus de la moitié des classificateurs de base obtiennent de faux résultats.

Le vote à majorité pondéré : C'est un vote basé sur des poids associés aux classificateurs de base. Ces poids peuvent être diminués ou augmentés au fur et à mesure que les classificateurs s'entraînent, suivant qu'ils produisent respectivement une bonne ou une mauvaise prédiction.

Stacking : C'est une méthode qui permet de combiner plusieurs classificateurs de base. La première phase consiste à induire *N* classificateurs Cl_i, à partir de *N* ensembles de données $\{S_1, S_2, ..., S_N\}$. Le test est ensuite fait sur un ensemble d'évaluation $T=\{t_1, t_2, ..., t_L\}$, indépendant des ensembles d'apprentissage *Si*. Dans la deuxième phase, un nouvel ensemble de données *M* est formé par les valeurs calculées $Cl_i(t_j)$ et la vraie classe de l'instance *tj*, classe(t_j). Chaque instance de *M* sera de la forme $<Cl_1(t_j), Cl_2(t_j),\ldots, Cl_N(t_j), classe(t_j)>$. Dans la dernière étape, un classificateur global est construit à partir de *M*. Les classificateurs de base peuvent être construits avec des algorithmes différents (arbres de décision, réseaux de neurone..) selon les contraintes du problème [8].

3.3 Cas des arbres de décision comme classificateur de base

Les arbres de décision sont très utilisés en classification à cause de leur simplicité d'interprétation et leur qualité de précision relative. Leur nature 'instable' les rend aussi de bons candidats pour l'application des méthodes ensemblistes. En effet, beaucoup de travaux dans le *Bagging* et *Boosting* sont réalisés à base de cette technique, en utilisant comme algorithme de base *C4.5* [12]. Des améliorations significatives de C4.5 -en terme de précision- ont été obtenues dans beaucoup de travaux, en appliquant l'approche d'apprentissage ensembliste [9]. Parmi les utilisations les plus récentes, on présente un aperçu sur les algorithmes *RandomForest* (2001), *CaScading Tees* (2003) et *RotationForest* (2006)

RandomForest (2001) [22] : Les ensembles de données utilisés par chaque classificateur sont obtenus par la technique du *Bagging.* La diversité est aussi renforcée par l'utilisation de la technique *du choix de sous espaces aléatoires*. Le calcul de gain d'informations à chaque noeud est optimisé par un choix aléatoire de *M* attributs (*M* est un paramètre de l'algorithme). Des résultats empiriques ont démontré une amélioration significative de la précision, par rapport à *C4.5* standard, et même les approches *Bagging* et *Boosting*.

CaScading Trees (CS4) (2003) [16]: Toujours dans une approche ensembliste, CS4 (*CaScading-and-combination FOR* constructing *decision tree ensembles*) a été conçu dans l'objectif d'améliorer la précision de la famille des algorithmes C4.5. Contrairement au *boosting* et *bagging* qui créent la diversité par des changements sur les données de base, CS4 utilise le même ensemble de données pour construire un ensemble d'arbres de décision. Les changements sont effectués sur la phase d'apprentissage des classificateurs de base. Les *k* arbres de décision sont construits à partir des *k* premiers attributs, ordonnés selon le gain d'information ; le ième attribut sera le noeud racine du ième arbre. Cette méthode a été testée sur des données bio-médicales de haute dimension (plus de 10.000 attributs), et a donné des résultats intéressants. Dans un domaine où l'interprétation des prédictions est importante comme le bio-médical, CS4 donne aussi la possibilité d'avoir un modèle global significatif, ce qui est impossible dans les méthodes classiques.

RotationForest (2006) [11]: Les classificateurs de base sont des arbres de décision construits indépendamment, mais chaque arbre est entraîné sur la totalité des données, dans un espace d'attributs 'en rotation'. Afin de créer l'ensemble de données d'apprentissage pour un classificateur de base, l'espace d'attributs est aléatoirement divisé en *k* (un paramètre de l'algorithme) sous ensembles et une analyse en composantes principales est appliquées aux sous ensembles. L'objectif de cette méthode est d'améliorer la précision individuelle de chaque classificateur de base et d'obtenir une diversité dans l'ensemble. Les résultats expérimentaux reporté dans ce travail affirment que *RotationForest* peut fournir une meilleure précision que *RandomForest*, *Bagging* et *Boosting*.

4. Les méthodes ensemblistes dans le Datamining Distribué

Nous étudions dans ce qui suit le datamining distribué effectué par les méthodes ensemblistes selon les deux perspectives visées : soit le traitement de données intrinsèquement distribuées, soit la haute performance des méthodes ensemblistes elles même.

4.1 La prédiction à partir de données distribuées

Dans des applications où les données sont distribuées géographiquement et ne peuvent être traitées en totalité dans un site centrale, il est légitime de se demander si on peut se contenter de la diversité naturelle, présente à priori dans les sous ensembles de données, sans provoquer des perturbations comme dans les méthodes ensemblistes classiques. Dans de telles applications, les données représentent soit *un partitionnement par données* (horizontal), ou *un partitionnement par attributs* (vertical), d'un ensemble global de données.

Le travail présenté dans [36] étudie l'effet du partitionnement vertical de l'ensemble de données, sur la qualité de la classification dans un environnement distribué ; il note que ce type de partitionnement n'est pas beaucoup traité dans la littérature du datamining distribué, par rapport au partitionnement horizontal. Pour chaque partition un arbre de décision est construit, et les prédictions sont ensuite combinés par deux schémas : le vote majoritaire et le vote pondéré.

Un travail plus récent [7] (Janvier 2008) propose une technique appelée *ensemble attribute* qui permet une prédiction à partir de données verticalement partitionnées. Cette technique est évaluée en fonction de plusieurs facteurs : la précision par rapport à une prédiction centralisée, la taille totale des arbres de décision et le temps d'exécution. Un problème considéré 'encore ouvert' d'après Skillicorn et al. [7] est de savoir si les attributs qui sont corrélés entre eux ont un effet sur la performance s'ils sont placés sur le même site ou s'ils sont séparés à travers les différents sites

Malgré que les perspectives sont différentes, l'étude de la prédiction à partir de données distribuées par un partitionnement horizontal, peut s'inspirer des travaux effectués dans l'objectif de passage à l'échelle, notamment l'approche de sous ensembles disjoints, qui sera présentée dans le paragraphe suivant.

4.2 Le passage à l'échelle (scaling up) des méthodes ensemblistes

Typiquement, un algorithme du datamining classique est destiné à traiter entre 200 et 100.000 exemples d'apprentissage, en secondes ou minutes, sur une plate forme bureau [6]. Pour la communauté du datamining, on parle de bases de données très grande (very large data bases) à partir de 100.000 exemples avec une douzaine d'attributs [31]. En effet, Avec une taille de données de l'ordre de tera-octet, et des centaines d'attributs à manipuler, la question du passage à l'échelle « scalability » consiste à voir si l'algorithme peut traiter efficacement une grande masse de données à partir de laquelle on veut construire les meilleurs modèles possibles. Ceci se complique encore plus dans le cas des méthodes ensemblistes, où le traitement de la totalité de l'ensemble de données est répété des dizaines de fois ou plus.

Le passage à l'échelle des méthodes ensemblistes se fait généralement via deux techniques : la technique de réduction des données traitées en manipulant des sous ensembles disjoints, et la technique classique du parallélisme

4.2.1 Approche de sous ensembles disjoints

Les algorithmes classiques des approches ensemblistes permettent de construire chaque classificateur de base à partir d'un ensemble de données de même taille que l'ensemble d'origine. Les premières expériences du bagging, par exemple, étaient dans le contexte de petits ensembles de données, allant jusqu'à 20.000 exemples [26]. Or, dans une

perspective de haute performance dans le datamining distribué, la taille de l'ensemble de données devrait être beaucoup plus importante. Une première approche consiste à choisir des sous ensembles de taille plus petite, ce qui pourrait diminuer le temps d'exécution, mais il est difficile, à priori, de connaître si la précision ne sera pas affectée.

Breiman dans [18] présente un algorithme qui permet de choisir à partir de l'ensemble de base, des sous ensembles aléatoires. C'est une approche qui peut traiter des ensembles de données de très grande taille, mais le nombre d'exemples utilisés par chaque classificateur est d'environ 800 exemples dans les expériences discutées, ce qui nécessite l'utilisation d'un très grand nombre de classificateurs.

Chawla et al. [25] explorent différentes stratégies de partitionnement de l'ensemble d'apprentissage. Les résultats expérimentaux démontrent qu'un simple partitionnement aléatoire des données, en plusieurs sous ensembles disjoints, engendre une meilleure précision par rapport à la création de plusieurs bags de même taille, ainsi qu'un gain considérable en temps de calcul. L'application d'une méthode 'plus intelligente' pour partitionner les données 'clustering' a été encore plus bénéfique qu'un simple partitionnement aléatoire.

La diversité des classificateurs étant primordial dans l'approche ensembliste, elle peut être aussi obtenue par un échantillonnage au niveau des noeuds dans le cas des arbres de décision comme dans [3], où les calculs au niveau des noeuds de l'arbre utilisent seulement une partie aléatoire de l'ensemble d'apprentissage. En général, des valeurs de précision compétitives ont été trouvées par rapport au Boosting et Bagging, à l'exception d'ensemble de données de très grande taille.

Dans un travail plus récent [27], les classificateurs de base sont aussi construits à partir de différentes partitions de données disjointes. Ils exploitent deux algorithmes *IVotes* et *Rvotes* conçus par Breiman [21]. Les résultats démontrent la possibilité de construire des centaines de classificateurs de base, avec des ensembles de données de taille très limitée. Ce travail a permis d'obtenir des précisions similaires ou meilleures que le boosting ou boosting distribué (cf. 4.2.2), avec un gain considérable dans le temps d'exécution.

4.2.2 Le parallélisme

L'aspect parallèle est très visible à travers la possibilité de construire les classificateurs de base simultanément. Le parallélisme du Bagging est relativement immédiat. Un travail présenté dans [5] propose de partitionner les données aléatoirement et équitablement, à travers plusieurs processeurs. Chaque processeur exécute l'algorithme séquentiel sur ses données locales, jusqu'à l'obtention des prédictions adéquates.

La technique de boosting a été beaucoup utilisée dans le but d'améliorer la précision d'un classificateur, sur un même et seul ensemble de données centralisé, d'une taille assez limitée pour tenir en mémoire centrale d'un ordinateur typique. Contrairement à l'approche du bagging où les classificateurs de base sont construits d'une façon indépendante, dans l'approche standard du boosting, les classificateurs de base sont construits en série. L'entraînement se fait sur l'ensemble de données entier, en affectant à chaque cycle des poids aux données.
Un travail de Lazarevic et al. [1] propose une version parallèle de l'algorithme boosting, qui peut être appliquée dans le cas où des ensembles de données disjoints ne peuvent pas être traités comme un seul ensemble. Les classificateurs de base sont construits en parallèle à partir d'ensembles de données disjoints. Les vecteurs des poids locaux sont mis à jour dans chaque site, et leurs sommes sont communiquées à tous les sites. Les résultats expérimentaux montrent que l'algorithme proposé a une précision comparable ou parfois meilleure que le boosting appliqué sur l'union des ensembles de données distribués. L'algorithme RandomForest souffre également du coût élevé de calcul, malgré sont apport important en terme de précision ; une version parallèle de RandomForest est étudiée dans [15].

5. Conclusion et Discussion

Le datamining distribué est né du besoin de traiter des données qui peuvent être d'une part très volumineuses, et/ou éventuellement distribuées géographiquement à travers plusieurs sites. Les méthodes d'apprentissage ensemblistes présentent des techniques prometteuses dans le monde du datamining, particulièrement en terme de précision. En classification, elles consistent à créer plusieurs classificateurs de base, pour ensuite combiner les prédictions.
Cet article a visé particulièrement le couplage méthodes ensembliste- datamining distribué. Ceci a été fait à base des deux perspectives: le passage à l'échelle des méthodes ensemblistes elle-même, et la prédiction à partir de données distribuées.
La problématique du passage à l'échelle concerne les algorithmes classiques du datamining, et d'autant plus les méthodes ensemblistes qui appliquent l'algorithme sur des dizaines d'ensembles de données, avec 100% de la taille d'origine. Une première solution immédiate est de faire recours au parallélisme ; une autre solution consiste à réduire la taille des ensembles d'apprentissage. Nous nous intéressons à cette deuxième approche qui consiste à répondre à la question : est-ce que la totalité de l'ensemble de données est vraiment utile pour construire les meilleurs modèles possibles ? En survolant la littérature qui traite ce sujet, on constate que ceci n'a pas encore obtenu un consensus général.

En effet, certains notent que le datamining rapporte de meilleurs résultats si plus de données sont analysées [21], d'autres déclarent que l'échantillonnage d'un très grand volume de données peut simplifier la tâche de l'apprentissage, il peut aussi dégrader la précision [4]. Beaucoup d'autres travaux constatent par contre que la construction d'un ensemble à partir d'une opération d'échantillonnage peut améliorer la précision [35].

Si effectivement on n'a pas besoin de la totalité des données, plusieurs pistes restent encore à explorer : la méthodologie d'échantillonnage, la taille adéquate des échantillons, le maintient d'une précision acceptable, …etc. D'autre part, une autre question est posée : C'est de voir si la réduction de l'ensemble d'apprentissage se fait par données ou par attributs. Les auteurs dans [32] par exemple soutiennent la proposition que le partitionnement par attribut est mieux que par données dans les méthodes d'apprentissage ensemblistes.

Dans la perspective de faire du datamining à partir de données distribuées, il est légitime de se demander si on peut se contenter de la diversité naturelle, présente à priori dans les sous ensembles de données, sans provoquer des perturbations comme dans les méthodes ensemblistes classiques. Dans de telles applications, les données représentent soit un partitionnement par données (horizontal), ou un partitionnement par attributs (vertical), d'un ensemble global de données.

Enfin, nous constatons que les méthodes ensemblistes présentent encore un potentiel de recherche, particulièrement dans une démarche de datamining distribué. La notion de diversité utilisée dans l'apprentissage ensembliste inspire un grand nombre de combinaisons d'approches, qui mériteraient d'être étudiées. D'autre part, les travaux réalisés dans l'une ou l'autre perspective (passage à l'échelle ou fouille de données distribuées), pourraient être complémentaires.

References:

1. A. Lazarevic and Z. Obradovic: Boosting Algorithms for Parallel and Distributed Learning. Distributed and Parallel Databases: An International Journal, Special Issue on Parallel and Distributed Data Mining, 2:203-229, (2002).

2. B. Park and H. Kargupta: Distributed data mining: algorithms, systems, and applications. In: Nong Ye editor Data Mining Handbook, pp. 341—358 (2002).

3. C. Kamath and E. Cantú-Paz: Creating ensembles of decision trees through sampling. Proceedings, 33-rd Symposium on the Interface of Computing Science and Statistics, Costa Mesa, CA, June (2001).

4. C. Perlich, F. Provost, and J. Simonoff. Tree induction vs. logistic regression: A learning-curve analysis. Journal of Machine Learning Research, 4:211–255, (2003).

5. C. Yu and D.B. Skillicorn. Parallelizing Boosting and Bagging. Technical report, Queen's University, Kingston, Ontario, Canada K7L 3N6, February (2001).

6. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, (1998).

7. D. B. Skillicorn, S. M. McConnell: Distributed prediction from vertically partitioned data. Journal of Parallel and Distributed Computing.Volume 68, isuue 1, pp:16-38 (January 2008).

8. David H. Wolpert: Stacked generalization. Neural Networks, 5(2) :241-259 (1992).

9. Dietterich, Thomas G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning, 40(2), 139–157 (2000).

10. J. Gehrke, R. Ramakrishnan, and Venkatesh Ganti: Rainforest – a framework for fast decision tree construction of large datasets. In Proceedings of the 24rd International Conference on Very Large Data Bases, pp 416-427 (1998).

11. J. J. Rodr´ıguez, L. I. Kuncheva, and C. J. Alonso: Rotation forest: A new classifier ensemble method. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28(10):1619–1630 (Oct 2006).

12. J. R. Quinlan: C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers, Inc (1993)

13. J. Ross Quinlan: Bagging, boosting, and C4.5. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI 96, AAAI Press, pp 725–730, Portland, Oregon (August 1996).

14. J. Shafer, R. Agarwal, and M. Mehta.: SPRINT: A scalable parallel classifier for data mining. In Proc. of 22nd International Conference on Very Large Databases, Mumbai, India (1996).

15. Ianyong Dai, Joohan Lee, Morgan C. Wang.: Efficient Parallel Data Mining for Massive Datasets: Parallel Random Forest. The International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA) (June 2005).

16. Jinyan Li, Huiqing Liu: Ensembles of Cascading Trees. In the Proceedings of the Third IEEE International Conference on Data Mining (ICDM'03), Melbourne, Florida, USA, November 19 – 22 (2003).

17. K. Liu, H. Kargupta, and J. Ryan: Distributed data mining bibliography. release 1.7 (December 2006).

18. Leo Breiman: Pasting small votes for classification in large database and on-line. Machine Learning, 36:85-103 (1999).

19. Leo Breiman: Arcing Classifiers. technical report. Dept. of Statistics, University of California, Berkeley (1996).

20. Leo Breiman: Bagging predictors. Machine Learning, 24:123–140 (1996).

21. Leo Breiman: Pasting bites together for prediction in large data sets. Machine Learning, 36(2):85–103 (1999).

22. Leo Breiman : Random forest. Machine Learning, 45:5–32 (2001).

23. M. Aoun-Allah : Le forage distribué des données : une approche basée sur l'agrégation et le raffinement de modèles. Thèse Ph.D., Université Laval (2006).

24. M. Mehta, R. Agrawal, and J. Rissanen: SLIQ: A fast scalable classifier for data mining. In Proceedin. Of EDBT (1996). 25. N. Chawla, S. Eschrich, and L. O. Hall: Creating Ensembles of Classifiers. IEEE International Conference on Data Mining, pp 580-581 (2001).

26. N. V. Chawla, T. E. Moore, L. O. Hall, K. W. Bowyer, W. P. Kegelmeyer, and C. Springer: Distributed Learning With Bagging-like Performance. Pattern Recognition Letters, 24:455--471 (2003).

27. Nitesh V. Chawla, Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer: Learning Ensembles from Bites: A Scalable and Accurate Approach. Journal of MachineLearning Research, 5:421—45, (April 2004).

28. O. Lawrence Hall, Nitesh Chawla, and W. Kevin Bowyer: Combining Decision Tress Learned in Parallel. In Working notes of KDD (1998).

29. O. Lawrence Hall, W. Kevin Bowyer.W. Philip Kegelmeyer, E. Thomas Moore, and Chi-ming Chao: Distributed learning on very large data sets. In Workshop on Distributed and Parallel Knowledge Discovery, (KDD00), pp 79-84 (aug 2000).

30. Paul Bradley, Johannes Geheke, Raghu Ramakrishnan, and Ramakrishnan Srikant: Scaling mining algorithms to large databases. Communications of the ACM, 45(8): 37-43 (August 2002)

31. R.A Pearson: A coarse grained parallel induction heuristic. In H. Kitano, V. Kumar, and
C.B. Suttner, editors, Parallel Processing for Artificial Intelligence 2, Elsevier Science, pp 207-226 (1994).

32. Robert Bryll, Ricardo Gutierrez-Osuna, and Francis Quek: Attribute Bagging: Improving Accuracy of Classifier Ensembles by Using Random Feature Subsets. Pattern Recognition, Elsevier Science, Vol. 36, No. 6, pp. 1291-1302 (June 2003).

33. Robert E. Shapire: The strength of weak learnability. Machine Learning, 5(2) : 197-227. (1990).

34. Robert P.W. Duin: The Combining Classifier: to Train or Not to Train?. Proceedings 16th International Conference on Pattern Recognition, 2: 765- 770 (2002).

35. S. Eschrich, N. V. Chawla, and L. O. Hall: Learning to predict in complex biological domains. Journal of System Simulation, 2:1464 – 1471 (2002).

36. Sabine McConnell David B. Skillicorn: Building Predictors from Vertically Distributed Data. Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research, pp 150-162 (2004).

37. Shu -Tzu Tsai, Chao-Tung Yang: Decision Tree Construction for Data Mining on Grid Computing. Proceedings IEEE International Conference on e-Technology, e-Commerce and e-Service (EEE'04) pp. 441-447 (2004).

38. T. G. Dietterich: Ensemble methods in machine learning. In J. Kittler and F. Roli, editors, First International Workshop on Multiple Classifier Systems, pp 1-15 (2000).

39. Tin Kam Ho: The Random Subspace Method for Constructing Decision Forests. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8):832—844 (1998).

40. Winton Davies and Pete Edwards: Dagger: A new approach to combining multiple models leraned from disjoint subsets. In machine Learning (2000).

41. Yoav Freud: Boosting a weak learning algorithm by majority. Information and Computation, 121(2) : 256-258 (Septembre 1995).

42. Yoav Freund and Robert E. Schapire: Experiments with a new boosting algorithm. In Lorenza Saitta, editor, Machine Learning: Proceedings of the Thirteenth International Conference, pp 148–156, Morgan Kaufmann, Bari, Italy (July, 1996).

43. Yongdai Kim: Convex hull ensemble machine. In Proceedings of 2002 IEEE International Conference on Data Mining (ICDM 2002), IEEE Computer Society, pp 243 – 249, Maebashi City, Japan (2002).

44. Zhang, S.C., Wu, X.D., and Zhang, C.Q.:Multi-database mining. IEEE Computational Intelligence Bulletin, Vol. 2 No. 1, IEEE Computer Society, pp. 5-13 (2003).