Arbres en ligne basés sur des statistiques d'ordre

Christophe Salperwyck[*,**], Vincent Lemaire[*]

[*]Orange Labs
2, Avenue Pierre Marzin 22300 Lannion
prenom.nom@orange.com
[**] LIFL (UMR CNRS 8022) - Université de Lille 3
Domaine Universitaire du Pont de Bois
59653 Villeneuve d'Ascq Cedex

Résumé. De nouveaux domaines d'application émergent où les données ne sont plus persistantes mais "passagères" : gestion de réseaux, modélisation de profils web... Ces données arrivent rapidement, massivement, et ne sont visibles qu'une fois. Il est donc nécessaire de les apprendre au fur et à mesure de leurs arrivées. Pour les problèmes de classification, les modèles à base d'arbres de décision en ligne sont performants et très largement utilisés sur ces problématiques. Dans cet article, nous proposons une nouvelle approche de construction d'un arbre uniquement basée sur des statistiques d'ordre. Le résumé habituellement utilisé dans la construction d'un arbre en ligne est basé sur une estimation de quantiles. Une méthode performante et robuste de discrétisation supervisée utilise ce résumé pour fournir à la fois un critère de coupure et un estimateur de densité par intervalle. Cette estimation permet la construction d'un classifieur Bayésien naïf dans les feuilles qui améliore la prédiction en début d'apprentissage.

1 Introduction

Les techniques d'apprentissage ont montré leurs capacités à traiter des volumétries importantes de données et ce sur des problèmes réels (Guyon et al., 2009; Féraud et al., 2010). Néanmoins le plus gros effort a été réalisé pour des analyses de données homogènes et stationnaires. La plupart des approches d'apprentissage statistique (machine learning) utilisent des bases d'apprentissage de taille finie et produisent des modèles statiques.

De nouveaux domaines d'application de la fouille de données émergent où les données ne sont plus sous la forme de tableaux de données persistants mais plutôt sous la forme de données "passagères", sous la forme de flux. Parmi ces domaines on citera : la gestion de réseaux de télécommunications, la modélisation des utilisateurs au sein d'un réseau social, le web mining... L'un des défis techniques est de concevoir des algorithmes permettant de traiter ces nouvelles contraintes applicatives. Les données arrivant rapidement et n'étant visibles qu'une fois, il est nécessaire de les apprendre au fur et à mesure de leur arrivée. L'apprentissage incrémental apparaît dès lors comme une solution naturelle à ces problèmes de flux.

Parmi les méthodes d'apprentissage incrémental, les modèles à base d'arbres de décision inspirés de l'algorithme « Very Fast Decision Tree » (VFDT) (Domingos et Hulten, 2000)

sont très largement utilisés. La construction de l'arbre est incrémentale et les feuilles se transforment en nœud au fur et à mesure de l'arrivée des exemples. Les nouveaux exemples descendent l'arbre et sont agrégés par variable dans un résumé. Un critère (indice de Gini ou entropie) est ensuite appliqué sur ce résumé afin de trouver les points de coupure pour transformer une feuille en nœud. La qualité de prédiction de l'arbre peut encore être améliorée par l'ajout d'un modèle local dans chacune des feuilles comme dans VFDTc (Gama et al., 2003).

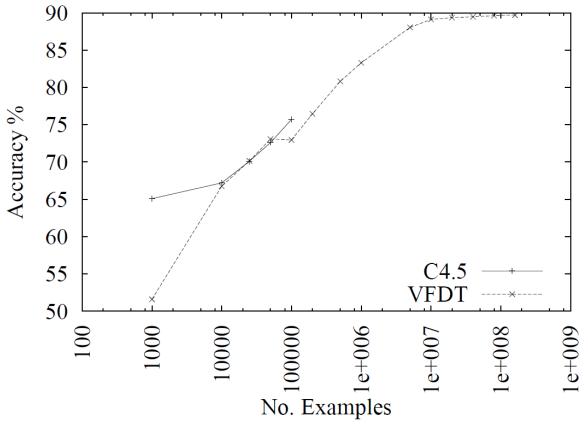

FIG. 1 – *Comparaison de C4.5 avec VFDT (extraite de (Domingos et Hulten, 2000))*

Le taux d'erreurs de l'algorithme est plus important en début d'apprentissage qu'un algorithme « batch » comme C4.5. Mais après avoir appris plusieurs centaines de milliers d'exemples, ce taux d'erreur devient plus faible car C4.5 n'est pas capable de travailler avec des millions d'exemples et doit donc n'en utiliser qu'une partie. La figure 1 extraite de l'article de VFDT illustre ce comportement et l'intérêt de VFDT par rapport à C4.5.

Dans cet article nous nous intéressons à la construction d'arbres de décisions en ligne. La section 2 présente notre positionnement par rapport aux approches existantes en termes de critère de coupure, résumé dans les feuilles, et de modèles locaux. En section 3 notre approche exclusivement basée sur des statistiques d'ordre est détaillée. La partie expérimentale comparant notre approche à celles existantes se trouve en section 4. La section 5 discute des travaux futurs et des possibilités de gestion de la dérive de concept issue de notre approche. La dernière partie conclut cet article.

2 Travaux antérieurs

La construction d'arbres de décision en ligne repose sur trois choix principaux. Premièrement, il est impossible dans le contexte des flux de données potentiellement de taille infinie de conserver tous les exemples. L'utilisation de résumé de données de taille finie est nécessaire afin de pouvoir contrôler la consommation mémoire de l'arbre. Le fait que les décisions soient locales à chaque feuille justifie le stockage des résumés dans chacune des feuilles. Deuxièmement le choix du point de coupure se fait par l'évaluation dans chaque feuille d'un critère

(généralement l'indice de Gini ou l'entropie). Ce choix étant une action définitive il faut s'assurer de sa robustesse et qu'il soit fait avec une certaine confiance. Finalement avant qu'une coupure ne se produise, l'information disponible dans les feuilles doit pouvoir être exploitée. L'utilisation d'un modèle local dans chacune des feuilles de l'arbre permet d'exploiter cette information afin d'améliorer la prédiction globale de l'arbre. La figure 2 illustre par un exemple simple l'approche des arbres en ligne et leurs différents composants.

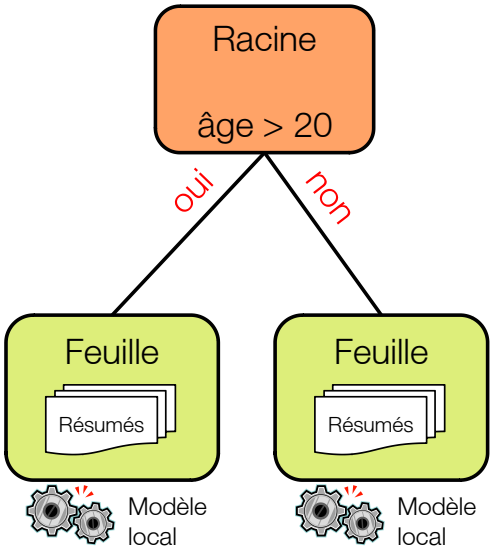

FIG. 2 – *Différents composants d'un arbre en ligne*

Cette section présente les différentes approches utilisées dans la littérature pour répondre aux trois points clés énoncés ci-dessus. La qualité d'un arbre en ligne dépend (i) des résumés dans les feuilles, (ii) du critère de coupure utilisé, (iii) du modèle local choisi.

2.1 Résumés dans les feuilles

Le but des résumés de données dans les feuilles d'un arbre de décision en ligne est de mémoriser les caractéristiques du flux et ainsi de pouvoir par la suite trouver le meilleur point de coupure pour transformer la feuille en nœud. Ce résumé peut aussi être utilisé pour construire un modèle local dans la feuille. Dans certains domaines applicatifs, comme par exemple la gestion d'un réseau de télécommunication ou d'énergie, le flux est potentiellement de taille infinie. L'arbre généré possèdera alors de fait un grand nombre de nœuds de décision. Cependant la mémoire disponible est elle souvent de taille finie. Il en découle que ces résumés doivent être de faibles tailles et gérer le compromis précision/erreur.

Les paragraphes ci-dessous illustrent certaines de ces techniques de résumés, où les attributs numériques et catégoriels sont en général traités à l'aide de techniques de résumés différentes.

2.1.1 Attributs numériques

Gama et Pinto proposent une méthode de résumé des attributs numériques appelée Partition Incremental Discretization - PiD (Gama et Pinto, 2006) qui est partiellement incrémentale. Cette solution est basée sur deux niveaux. Le niveau 1 réalise une première discrétisation où les comptes sont stockés par intervalle. Ce premier niveau implémente un algorithme incrémental qui combine les méthodes « Equal Frequency » et « Equal Width » et il doit contenir plus d'intervalles que le niveau 2. Le deuxième niveau utilise la discrétisation réalisée au niveau 1 pour effectuer la discrétisation finale qui peut être de plusieurs types : « Equal Frequency », « Equal Width », K-means, Recursive entropy discretization, Proportional discretization. La consommation mémoire de cette méthode dépend principalement du nombre d'intervalles du premier niveau. Le deuxième niveau n'est lui pas incrémental.

Dans le cadre des arbres en ligne (Pfahringer et al., 2008) ont réalisé une étude sur les résumés pour les attributs numériques ; étude dédiée aux arbres utilisant la borne d'Hoeffding. Ils ont testé différentes méthodes pour la réalisation de ces résumés :

- VFML (Very Fast Machine Learning) : cette méthode, très simple, prend les k premières valeurs du flux et s'en sert pour construire $k + 1$ intervalles. Les valeurs suivantes sont agrégées dans l'intervalle le plus proche défini par les premières valeurs. La consommation mémoire dépend du k choisi. Cette méthode provient du code source (Hulten et Domingos, 2003) fourni par Domingos et Hulten créateur de VFDT.
- Approximation par une Gaussienne : la distribution des données est supposée être une loi normale que l'on va chercher à approximer. Pour cela il suffit de ne conserver que les trois éléments qui définissent cette Gaussienne : la moyenne, l'écart type (ou la variance) et le nombre d'éléments. Le maintien de ces trois valeurs pouvant se faire de manière incrémentale cette méthode est parfaitement adaptée à une utilisation en ligne. Elle est choisie par Kirkby (Kirkby, 2008) dans toutes ses expérimentations. La consommation mémoire est constante quel que soit la nature du flux observé.
- Arbres binaires exhaustifs : Gama et al. utilisent cette méthode pour leur arbre VFDTc (Gama et al., 2003). Un arbre binaire de recherche, par variable numérique, est construit de manière incrémentale dans lequel toutes les valeurs observées sont stockées. La structure de l'arbre et la mémorisation des comptes des valeurs observées permet un accès immédiat aux comptes de part et d'autre d'un point de coupure. La consommation mémoire de l'arbre dépend du nombre de valeurs différentes arrivant dans la feuille.
- GK : cette méthode est un résumé basé sur des quantiles. Elle maintient un tableau trié d'intervalles et contrôle l'erreur sur la position du quantile. Elle sera détaillée dans la section 3.1.1. La consommation mémoire dépend soit de l'erreur maximale tolérée sur les quantiles soit du nombre fixé de quantiles à conserver.

2.1.2 Attributs catégoriels

A notre connaissance toutes les publications précédemment citées dans le cadre des arbres en ligne utilisent une seule et unique méthode de résumé pour les attributs catégoriels qui est basée sur un comptage exhaustif. Cette méthode consiste à compter le nombre d'occurrences d'une valeur par variable. Elle peut être suffisante si le nombre de valeurs est limité et donc que ce comptage exhaustif peut tenir dans l'espace mémoire accordé pour le comptage. Sa consommation mémoire dépend donc du nombre de valeurs différentes du flux de données.

2.2 Critère de coupure

Lors de la construction de l'arbre de décision un critère de pureté est utilisé pour transformer une feuille en nœud. L'objectif est de produire des groupes d'individus les plus homogènes possibles du point de vue de la variable à prédire. Pour transformer une feuille en un nœud il faut déterminer à la fois l'attribut sur lequel couper ainsi qu'un point de coupure.

La littérature des arbres de décision hors ligne et en ligne utilise principalement deux critères de coupure : l'indice de Gini que l'on retrouve dans l'algorithme CART et le « gain ratio » basé sur l'entropie utilisé dans C4.5. Ces deux critères permettent de trouver le meilleur point de coupure pour un attribut et fournissent un score. Il suffit ensuite de réaliser la coupure sur l'attribut ayant le meilleur score. Ce processus de transformation d'une feuille en nœud se répète afin d'induire l'arbre de décision final.

La différence entre un arbre en ligne et hors-ligne réside dans le fait que l'arrivée des données se fait en continu. Le choix de l'attribut de coupure se fait d'après le résumé construit et non à l'aide de toute la distribution des exemples. Le choix permettant la transformation d'une feuille en nœud est une action définitive. Afin de s'assurer que ce choix soit réalisé avec une certaine confiance, Domingos et Hulten propose dans VFDT d'utiliser la borne d'Hoeffding (Hoeffding, 1963). Cette borne apporte une garantie sur le choix du bon attribut. La borne d'Hoeffding a par la suite été très souvent utilisée pour la réalisation d'arbre de décision en ligne : VFDTc (Gama et al., 2003), CVFDT (Hulten et al., 2001), IADEM (Ramos-Jimenez et al., 2006), « ensemble d'arbres d'Hoeffding » (Kirkby, 2008)... Nous utiliserons aussi dans la suite de cet article cette borne dans le cadre de la construction de l'arbre en ligne proposé. Nous la détaillons ci-dessous.

Borne d'Hoeffding La borne d'Hoeffding permet de s'assurer que la vraie moyenne d'une variable aléatoire comprise dans un intervalle R ne sera pas différente, à ϵ près, de sa moyenne estimée après le tirage de n observations indépendantes, tout cela avec une probabilité de $1-\delta : \epsilon = \sqrt{\frac{R^2}{2n} ln(\frac{1}{\delta})}$. L'intérêt de cette borne est qu'elle ne dépend que : (i) de la plage de valeurs R, (ii) du nombre d'observations n, (iii) de la confiance désirée δ. Cette borne est plus conservatrice qu'une borne faisant une hypothèse sur la distribution des valeurs.

Cette borne permet de s'assurer (avec une probabilité de $1-\delta$) que le choix de l'attribut de coupure dans un nœud est le bon. Pour cela la borne est utilisée sur la moyenne d'un critère G d'évaluation de la qualité d'une coupure. Le meilleur attribut a est considéré définitivement meilleur que le deuxième meilleur attribut b si $\bar{G}_a - \bar{G}_b > \epsilon$.

2.3 Modèles locaux

Gama et al. (Gama et al., 2003) observent empiriquement qu'il faut de 100 à 1000 exemples avant de transformer une feuille en nœud. Ces exemples dans les feuilles ne sont pas utilisés pour améliorer le modèle global tant que la feuille n'est pas transformée en nœud. Ils proposent alors d'utiliser ces exemples en ajoutant dans chaque feuille un modèle local. Cet algorithme, dénommé VFDTc, supporte par ailleurs les attributs numériques. On peut noter que cette technique d'utilisation de modèles locaux positionnés dans les feuilles d'un arbre existe depuis longtemps. Par exemple l'algorithme NBTree (Kohavi, 1996) utilise un arbre de décision avec un classifieur Bayésien naïf dans les feuilles.

Un bon modèle local pour les arbres en ligne doit consommer peu d'espace mémoire, être rapide à construire et à retourner une prédiction. Il doit de plus être en adéquation avec les résumés construits dans les feuilles. Une bonne capacité de prédiction du modèle local avec un faible nombre d'exemples est nécessaire car les résumés dans les feuilles peuvent être basés sur peu d'exemples. Une étude (Salperwyck et Lemaire, 2010) menée sur la vitesse (en nombre d'exemples) d'apprentissage de différents classifieurs montre que les forêts d'arbre et le classifieur Bayésien naïf sont des classifieurs qui requièrent peu d'exemples pour bien apprendre. Parmi ces deux classifieurs seul le classifieur Bayésien naïf respecte aux mieux les conditions imposées par la construction en ligne. En effet, il ne nécessite aucun espace mémoire supplémentaire si le résumé permet de retourner une estimation de densité par intervalle (ce qui est le cas de tous les résumés présentés précédemment). De plus il est rapide à construire et possède une faible complexité algorithmique en prédiction. Ce classifieur est d'ailleurs celui choisi dans VFDTc et il améliore la prédiction de l'arbre sur les jeux de données WaveForm et LED de l'UCI (Gama et al., 2003).

3 Approche proposée

Cet article introduit des travaux en cours sur la construction d'arbres de décisions en ligne basés sur les statistiques d'ordre. Pour les résumés nous avons choisi des méthodes consommant peu de mémoire et contrôlant leurs erreurs en plus d'être basées sur les statistiques d'ordre. Pour le critère et les modèles locaux, le choix s'est porté sur la méthode MODL qui permet à la fois d'évaluer un point de coupure et de fournir une discrétisation ensuite utilisable par un classifieur Bayésien naïf.

3.1 Résumés dans les feuilles

Les résumés utilisés par notre approche ont à la fois une consommation mémoire fixe et des garanties fortes en termes de contrôle de l'erreur sur les comptes. Ces résumés sont présentés ci-dessous.

3.1.1 Attributs numériques : résumé de quantiles (GK)

Les quantiles fournissent une statistique d'ordre sur les données. Le ϕ-quantile, avec $\phi \in [0, 1]$ est défini comme étant l'élément à la position $\lceil \phi N \rceil$ dans la suite triée des N valeurs vues jusqu'à présent. Soit ϵ l'erreur que l'on s'autorise sur la position de l'élément. Un élément est une ϵ approximation d'un ϕ-quantile si son rang est entre $\lceil (\phi - \epsilon)\, N \rceil$ et $\lceil (\phi + \epsilon)\, N \rceil$. Ceci revient au même que de réaliser une discrétisation en « Equal Frequency » ; le nombre de quantiles étant le nombre d'intervalles dans ce cas là.

Le résumé de quantiles GK (Greenwald et Khanna, 2001) est un algorithme de calcul de quantile dont la consommation de mémoire est en $O(\frac{1}{\epsilon} log(\epsilon N))$ dans le pire des cas. Cette méthode ne nécessite pas de connaître la taille des données à l'avance et est insensible à l'ordre d'arrivée des données. Selon le besoin, et pour un nombre de quantiles fixé, on peut soit définir une borne d'erreur maximale, soit définir une borne de mémoire maximale à utiliser. L'algorithme est basé sur un tableau de tuples $< v_i, g_i, \Delta_i >$ où :

– v_i est une valeur du flux de données

– g_i correspond au nombre de valeurs vues entre v_{i-1} et v_i
– Δ_i est l'erreur maximale possible sur g_i

Cette méthode est évaluée dans (Pfahringer et al., 2008) en conservant un résumé par classe et par attribut. Elle est évaluée comme étant moins performante que la méthode par Gaussienne ou la méthode VFML. Cependant nous avons adapté le résumé GK afin de n'avoir qu'un résumé par attribut mais dans lequel chaque tuple contient le nombre d'individus par classe. Expérimentalement cette modification améliore la qualité de prédiction (pour des raisons de place les expérimentations sur ce point ne sont pas détaillées dans cet article).

3.1.2 Attributs catégoriels : Count-min Sketch (CMS)

Le but du Count-min Sketch (Cormode et Muthukrishnan, 2005) est de trouver les top-k plus importantes occurrences d'une valeur avec une erreur maximale ϵ dans un flux de données. Une matrice de comptage de taille $t \times b$ est utilisée pour son stockage. Il utilise t fonctions de hachage h_i dans $\{1, \ldots, b\}$ qui sélectionnent la cellule de la matrice à incrémenter :

$$\forall i = 1, \ldots, t \quad h_i(e) = h_i(e) + 1$$

Le choix de t et b se fait à l'aide de deux paramètres δ et ϵ. Si l'on veut que l'estimation de la fréquence d'apparition $\hat{f}$ d'un item n'ait pas une erreur supérieure à ϵn avec une probabilité d'au moins $1 - \delta$ alors il faut prendre $t = log\frac{1}{\delta}$ et $b = O(\frac{1}{\epsilon})$. La fréquence d'un élément e est estimée par le minimum de $h_i(e)$.

$$\hat{f} = \underset{i}{argmin}(h_i(e))$$

L'utilisation du Count-min Sketch est pertinente quand le nombre de valeurs différentes est important. Dans le cas contraire l'utilisation d'un simple comptage est un meilleur choix qui n'introduit pas d'erreurs et consomme peu de mémoire.

3.2 Critère

Afin d'être cohérent avec nos résumés (GK et CMS), le critère MODL qui est basé sur les statistiques d'ordre et les comptes a été choisi pour réaliser les coupures et les groupements. L'approche MODL, initialement appliquée à la discrétisation et au groupage de valeurs, permet d'évaluer la qualité d'une coupure ou d'un groupement de valeurs respectivement pour une variable numérique et catégorielle. Son indice d'évaluation $l \in [0..1]$ correspond à un taux de compression qui indique l'informativité d'une variable numérique ou catégorielle.

Les méthodes MODL de discrétisation (Boullé, 2006) et de groupage de valeurs (Boullé, 2005) sont supervisées et sans paramètre. Elles se basent sur les comptes des classes par rapport à tous les découpages possibles en intervalles pour les variables numériques et en groupes pour les variables catégorielles. L'évaluation de la qualité du modèle est basée sur une approche Bayesienne. Son but est de trouver les meilleurs paramètres de la discrétisation/groupage : nombre d'intervalles, bornes des intervalles et répartition des classes dans les intervalles au sens Bayésien.

L'arbre, proposé dans cet article, est construit en ligne à la manière de VFDT en utilisant la borne d'Hoeffding mais le critère MODL remplace le critère du gain en Entropie. L'utilisation

du critère MODL a un intérêt supplémentaire car il retourne une valeur de critère $l > 0$ si et seulement si le modèle de discrétisation/groupage est meilleur que le modèle qui retourne la classe majoritaire. Cette propriété permet un pré élagage automatique de l'arbre alors que dans VFDT ce pré élagage a dû être implémenté séparément. De plus ce critère n'évalue pas seulement une coupure binaire mais peut aussi évaluer les coupures n-aires, ce qui permet de construire des arbres ayant des nœuds avec plus de deux fils.

3.3 Modèles locaux : approche à deux niveaux basée sur les statistiques d'ordre

Nous avons choisi le classifieur Bayésien naïf comme modèle local dans les feuilles pour ses performances lorsqu'il est construit avec peu de données ainsi que pour sa faible complexité algorithmique (voir section 2.3). Ce classifieur nécessite une estimation de la densité conditionnelle aux classes. Cette densité est estimée au sein de chaque intervalle ou groupe de valeurs calculés par la méthode MODL appliquée respectivement aux résumés GK ou CMS contenus dans les feuilles. Cette discrétisation sur les résumées correspond à la méthode de discrétisation à deux niveaux proposés par Gama avec sa méthode PiD (voir 2.1.1). Toutefois dans notre cas, notre approche à deux niveaux est basée entièrement sur des statistiques d'ordre et peut traiter indifféremment des attributs numériques ou catégoriels. Pour les variables numériques, le premier niveau correspond au résumé de quantiles GK et le second à la méthode de discrétisation MODL. Pour les variables catégorielles, le premier niveau correspond au Count-min Sketch (ou a un comptage) suivi du groupage MODL.

L'un des intérêts de la méthode MODL réside dans sa robustesse et son absence de paramètres. L'approche MODL discrétise/groupe une variable numérique/catégorielle en plusieurs intervalles/groupes si et seulement si cette discrétisation/groupage est meilleure (au sens Bayésien) que le modèle à un seul intervalle/groupe. Dans le cas ou le modèle le plus probable est le modèle à un intervalle (ou à un seul groupe de valeurs) cela signifie que l'attribut n'est pas informatif étant donné les données observées. Dans ce cas c'est la classe majoritaire qui est prédite. Kirkby dans (Kirkby, 2008) étudie justement ce comportement en comparant des arbres d'Hoeffding prédisant la classe majoritaire et ceux ayant un classifieur Bayésien naïf dans les feuilles. Il observe que parfois l'un est meilleur parfois l'autre. De ce fait il propose une méthode choisissant automatiquement soit l'un soit l'autre en évaluant leur taux d'erreur sur les exemples qui passent dans les feuilles. La méthode MODL devrait permettre de retrouver automatiquement ce comportement.

4 Expérimentations

Cette section constitue une évaluation préliminaire de la pertinence de notre approche par rapport aux méthodes décrites en section 2. Nous présentons tout d'abord les flux utilisés, puis les différentes méthodes évaluées et enfin les résultats obtenus.

4.1 Flux utilisés

Afin d'avoir assez de données pour tester les algorithmes d'apprentissage sur les flux, des générateurs artificiels existent pour permettre de générer des millions d'exemples. Un état de

l'art de ces générateurs est présenté dans (Kirkby, 2008) ou (Gama, 2010). Pour nos expérimentations nous allons utiliser les générateurs suivants :

- Random RBF (Radial Basis Function) : plusieurs centroïdes définies par leur centre et leur diamètre sont tout d'abord générées à différentes positions dans l'espace. Puis des exemples appartenant à ces bulles sont générés en s'éloignant du centre d'une distance générée aléatoirement à partir d'une loi gaussienne.
- Random Tree : génération d'un arbre avec certains paramètres (profondeurs minimale et maximale, nombres de variables numériques et catégorielles) puis génération de données à partir de cet arbre. Ce générateur favorise les algorithmes d'apprentissage basés sur les arbres.
- Waveform : problème à trois classes généré à partir de fonctions en forme d'ondes différentes et combinées. Ce générateur est basé sur une loi Gaussienne et favorise donc les classifieurs faisant une hypothèse Gaussienne.
- Function (F6) : génération d'exemples à partir d'une fonction basée sur 6 attributs numériques et 3 attributs catégoriels. Dans nos expérimentations nous avons choisi d'utiliser la fonction N°6 proposé par Agrawal avec 5% de bruit.

4.2 Algorithmes testés

Pour nos expérimentations nous avons utilisé la boîte à outils MOA : Massive Online Analysis (Bifet et al., 2009) développée par l'université de Waikato qui reprend la librairie VFML fourni par Hulten et Domingos (Hulten et Domingos, 2003). Cette boîte à outils permet de générer les flux et fournit les algorithmes de l'état de l'art auxquels nous souhaitons nous comparer. Nous y avons ajouté nos arbres basés sur les statistiques d'ordre.

Tous les arbres évalués partent du même algorithme basé sur les arbres d'Hoeffding. Cet algorithme contient trois paramètres : (i) le nombre d'exemples à considérer avant d'essayer de trouver un point de coupure ; (ii) la précision que l'on souhaite avoir sur la moyenne dans la borne d'Hoeffding et (iii) le paramètre τ qui est utilisé pour imposer le choix entre deux attributs dont la différence de critère est trop proche pour que la borne d'Hoeffding puisse les départager. Le paramétrage de toutes les versions des arbres testés ici est le même à savoir : $n = 200$, $\delta = 10^{-6}$, $\tau = 0.05$.

Les résumés dans les feuilles pour les attributs catégoriels n'utilisent pas, dans ces expériences, le count-min sketch et le groupage MODL car les jeux de données utilisés comportent peu de valeurs différentes. Le résumé est donc basé sur un simple comptage. Les attributs numériques sont eux résumés soit par une estimation basée sur une Gaussienne soit par des quantiles GK.

L'attribut et le point de coupure sélectionnés pour transformer une feuille en nœud sont ceux possédant la valeur de critère maximale non nulle. Ce critère évalue, pour les variables catégorielles le modèle créant une feuille par valeur de la variable et pour les variables numériques uniquement les coupures binaires.

Les différentes approches ont été testées soit sans **modèles locaux** soit avec un classifieur Bayésien naïf dans chaque feuille. Le tableau 1 présente une vue synthétique des algorithmes présentés ci-dessous :

- **HT : Arbre d'Hoeffding.** Cet algorithme correspond à l'arbre d'Hoeffding de VFDT (Domingos et Hulten, 2000). Les résumés numériques sont basés sur l'estimation de densité par une Gaussienne par classe (évalué comme étant le plus performant dans

(Pfahringer et al., 2008)). Pour les attributs numériques 10 coupures binaires par classe, prises entre le minimum et le maximum observés, sont évaluées. Cet arbre ne possède pas de classifieur dans les feuilles.

- **HTNB : Arbre d'Hoeffding avec résumé Gaussien et classifieur Bayésien naïf.** Cette version est identique à la précédente mais possède un classifieur Bayésien naïf dans les feuilles. Les densités conditionnelles sont estimées pour (i) les variables numériques à l'aide de la loi Gaussienne paramétrée par les 3 éléments du résumé (μ, σ, n) ; et pour (ii) les variables catégorielles à l'aide des comptes par classe et par valeur.
- **HTmGKc : Arbre d'Hoeffding avec critère de coupure MODL.** Cette version correspond à la version que nous avons décrite dans la section 3. Le résumé pour les variables numériques est basé sur le résumé de quantiles GK avec des tuples contenant directement les comptes par classe. Chaque variable est résumée par 10 tuples. Le critère de coupure utilisé est basé sur le critère MODL décrit section 3.2. Les 10 coupures binaires évaluées proviennent des quantiles du résumé GK. Cette version ne possède pas de modèle local dans les feuilles.
- **HTmGKcNBm : Arbre d'Hoeffding avec critère de coupure MODL et classifieur Bayésien naïf.** Ce classifieur est identique à la version « HTmGKc » mais les feuilles contiennent un classifieur Bayésien naïf utilisant les probabilités conditionnelles aux classes. Celles-ci sont estimées pour les variables numériques par intervalles de valeurs issues du modèle de discrétisation MODL ((Boullé, 2006)) et pour les variables catégorielles à l'aide des comptes par classe et par valeur.

Méthode	Résumé numérique	Critère	Discrétisation numérique	Modèle local
HT	Gaussien	Gain en Entropie	Aucune	Aucun
HTNB	Gaussien	Gain en Entropie	Aucune	NB
HTmGKc	GK 10 tuples	Level MODL	Aucune	Aucun
HTmGKcNBm	GK 10 tuples	Level MODL	MODL sur GK	NB

TAB. 1 – *Spécifications des différents algorithmes testés (NB : Bayésien naïf)*

4.3 Résultats

Les résultats de nos expérimentations sont présentés dans la figure 3 sous la forme d'une courbe par jeu de données. Chaque générateur crée un flux de 11 millions d'exemples. Parmi les méthodes d'évaluation (Gama et al., 2009) nous avons choisi la méthode basée sur un ensemble de test. Le premier million d'exemples générés est utilisé comme ensemble de test et les 10 millions suivants comme ensemble d'apprentissage. Le critère d'évaluation utilisé est la précision et les classifieurs sont évalués tous les 300.000 exemples. De manière générale notre méthode se comporte bien sur les différents flux testés et est compétitive comparée aux autres méthodes. Le critère de coupure MODL basé sur les résumés numériques GK et les comptes par classe pour les variables catégorielles est globalement meilleur que celui basé sur le gain d'entropie sur un résumé numérique Gaussien et les comptes par classe pour les variables catégorielles

Modèle local L'apport du classifieur Bayésien naïf dans les feuilles est discutable dans le cas de l'utilisation du résumé avec une Gaussienne car parfois il peut très significativement soit améliorer (WaveForm) soit dégrader (F6) les performances du classifieur global. Dans le cas de l'utilisation de nos résumés à deux niveaux, le classifieur Bayésien naïf améliore toujours le résultat surtout en début d'apprentissage. Cette non dégradation est dûe à la robustesse de l'approche MODL. Celle-ci crée des intervalles seulement s'ils contiennent de l'information. Si la variable n'est pas informative aucun modèle de discrétisation n'est proposé. L'estimation basée sur ces intervalles est ensuite fournie au classifieur Bayésien naïf.

L'estimation de densité à base de Gaussienne donne de meilleurs résultats sur les flux de données « WaveForm » et « Random RBF ». Ce phénomène est particulièrement marqué en début d'apprentissage. Ce résultat n'est pas surprenant car il paraît normal que le classifieur Bayésien naïf ayant comme estimateur de densité une loi Gaussienne fonctionne bien sur des données issues des générateurs « WaveForm » et « Random RBF » qui utilisent des Gaussiennes pour générer les données.

5 Discussion

5.1 Mémoire limitée et critère global

Le comportement des algorithmes pour une quantité de mémoire limitée n'a pas été étudié dans cet article. Cependant il existe plusieurs techniques pour améliorer un classifieur en ligne utilisant des arbres qui seraient contraints de travailler avec une mémoire limitée.

L'une des plus simples consiste à supprimer le résumé dans les feuilles des variables les moins intéressantes. Le level MODL calculé pour chaque variable fournit une indication précise de l'information contenue dans une variable. Il serait donc intéressant de regarder si son utilisation donne de meilleurs résultats que celle du gain en entropie utilisée dans la littérature.

Une deuxième technique consiste à désactiver des feuilles et donc à limiter l'accroissement d'une partie de l'arbre. Ceci est réalisé en conservant les feuilles contenant le plus d'erreurs et dont on espère le plus d'amélioration. Une autre possibilité serait d'utiliser un critère global qui permettrait de savoir quelle partie de l'arbre peut potentiellement améliorer le plus sa qualité. Cette approche a déjà été utilisée pour une construction d'arbre « hors-ligne » (Voisine et al., 2009). Il s'agirait d'adapter le critère pour une utilisation « en-ligne ».

La troisième technique est de contraindre dynamiquement la quantité de mémoire utilisée par les résumés. Dans notre cas cette approche serait assez simple à réaliser en fusionnant des quantiles (GK) ou en diminuant la taille des nouveaux CMS.

5.2 Gestion de la dérive de concept

Cette section présente brièvement l'intérêt d'utiliser les statistiques d'ordre et le critère MODL dans les arbres en ligne pour détecter le changement de concept. En effet si le processus sous-jacent qui « génère » les données n'est pas stationnaire, le concept cible à apprendre et/ou la distribution des exemples peuvent varier au cours du temps. L'algorithme d'apprentissage doit donc être capable de détecter ces changements et d'adapter le modèle en conséquence.

On considère que le concept à apprendre peut varier dans le temps mais qu'il est persistant et consistant (voir (Lazarescu et al., 2004)) entre deux changements. La période de temps

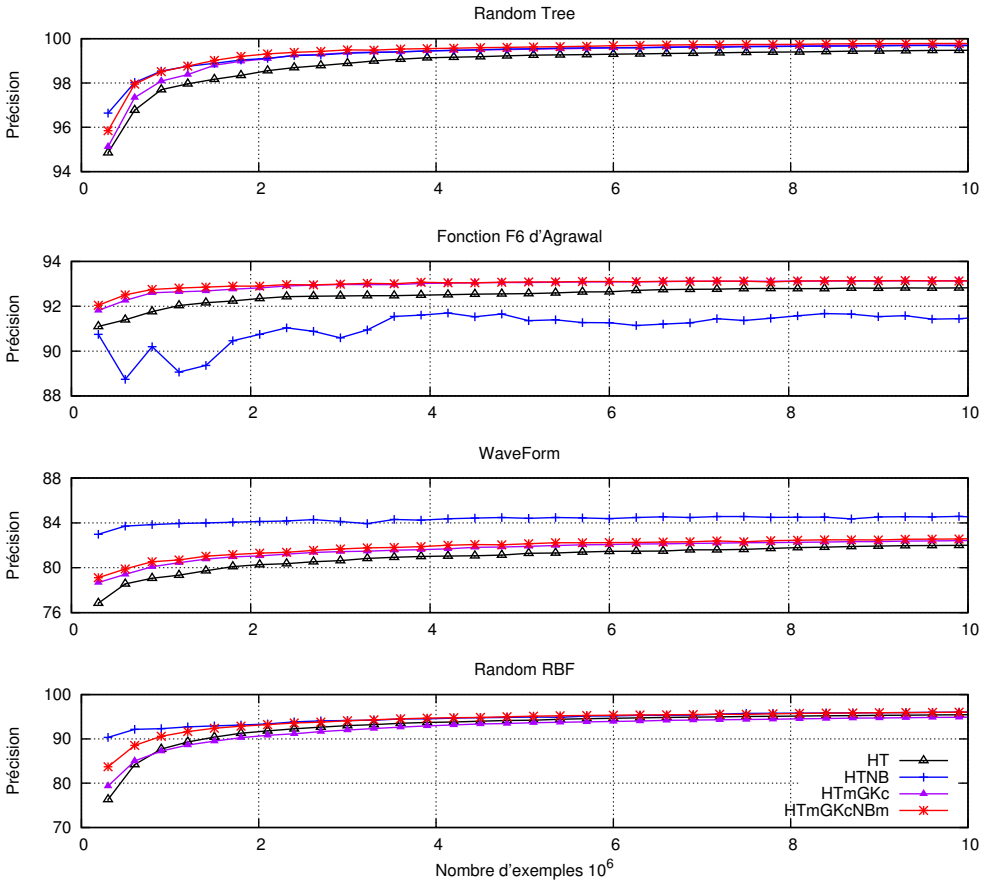

FIG. 3 – *Précision en fonction du nombre de données apprises*

entre deux changements de concept est appelée contexte (Gama, 2010). La dérive de concept apparaît à travers les exemples qui sont générés : les anciennes observations du processus deviennent caduques vis-à-vis des observations actuelles du processus. Les anciennes observations n'appartiennent pas au même contexte que les observations actuelles.

Si on suppose qu'entre deux changements de contexte il existe un concept suffisamment persistant et consistant pour lequel on peut collecter assez d'exemples d'apprentissage alors gérer la dérive de concept se ramène souvent à la détection de ce changement.

La détection du changement est étudiée dans l'état de l'art à l'aide de deux voies prédominantes : (i) la surveillance des performances du modèle courant de classification et (ii) la surveillance de la distribution des exemples. Le lecteur intéressé trouvera dans le tutorial présenté à PAKDD 2011 une excellente introduction sur ce sujet (voir `http://www.cs.waikato.ac.nz/~abifet/PAKDD2011/`).

Dans les sections précédentes de cet article on a présenté les résumés utilisés, les critères

de coupures (ou de groupage pour les attributs catégoriels) ainsi que les modèles locaux utilisés dans les feuilles de l'arbre. Ces trois éléments clefs peuvent aider à gérer le drift. Nous présentons ci-dessous des éléments de discussion de futurs travaux de recherche qui pourront être réalisés.

- **Utilisation des résumés dans les feuilles** : Lors de la décision de couper une feuille pour en faire un nœud de décision il est possible de conserver le résumé utilisé à ce moment précis. Un nouveau résumé est alors construit (mise à jour) en partant éventuellement du clone du résumé initial. Le résumé initial n'étant plus lui mis à jour. Muni de ces deux résumés on peut alors utiliser une méthode similaire à celle proposée dans (Bondu et Boullé, 2011) et qui permet de détecter des changements dans la distribution des exemples.
- **Utilisation du critère de coupure / groupage** : Le critère de coupure utilisé pour les variables numériques ainsi que le critère de groupage pour les variables catégorielles est un critère Bayésien. Ce critère évalue la probabilité d'un modèle, M_d de discrétisation (ou de groupage), $P(M_d|D)$, étant données les données observées par la feuille. Ce critère Bayésien pourrait être utilisé pour détecter le concept drift.
  A l'aide de la formule de Bayes il serait possible de calculer la probabilité que chaque exemple (d) ait été généré par le modèle de discrétisation (ou de groupage) : $P(d_j|M_{d_j})$; j étant l'indice de la variable explicative ayant été utilisée pour réaliser la coupure et M_{d_j} le modèle de discrétisation ou de groupage de la variable j. La comparaison, par nœud de l'arbre, de la distribution des $P(d_j|M_{d_j})$ au moment de la coupure et de la distribution des $P(d_j|M_{d_j})$ sur une fenêtre glissante (les z dernières valeurs de $P(d_j|M_{d_j})$ peuvent alors être stockées dans les feuilles) donnera une indication de la présence d'un concept drift sur j.
- **Utilisation des modèles locaux** : Le classifieur naïf de Bayes présent dans les feuilles utilise comme prétraitement la discrétisation à deux niveaux présentés lors de la section 3.3. Les valeurs des variables numériques et catégorielles sont respectivement discrétisées ou groupées pour produire des probabilités conditionnelles aux classes ($P(d_j|C)$). Ces probabilités conditionnelles sont utilisés pour élaborer le modèle de classification, M_c, le classifieur naïf de Bayes. Sous la condition d'indépendance des variables on pourrait calculer la probabilité que chaque exemple (d) ait été généré par le classifieur naïf de Bayes : $P(d|M_c) \approx \prod_{j=1}^{J} p(d_j|M_c) \approx \prod_{j=1}^{J} p(d_j|M_{d_j})$ (J étant le nombre de variables explicatives utilisées par le classifieur). Là encore la comparaison, par feuille, de la distribution des $P(d|M_c)$ sur une fenêtre de référence et sur une fenêtre glissante donnera une indication de la présence d'un concept drift sur M_c.

Il est ici intéressant de noter que les détections de drift présentées ci-dessus sont réalisées à l'aide des constituants de l'arbre et non à l'aide des performances de l'arbre. Les critères de construction de l'arbre peuvent être utilisés pour replier tout ou partie de l'arbre. Cette approche paraît plus consistante que celles basées sur les techniques de détection basées sur la performance du classifieur ; performance du classifieur rarement utilisée comme critère de construction de ce dernier.

6 Conclusion

Cet article constitue un travail en cours sur une méthode d'arbre de décision incrémentale construit en ligne sur des statistiques d'ordre. Notre approche a été présentée et située par rapport à l'état de l'art sur les différents points le constituant. Les résumés de quantiles GK et le résumé CMS sont utilisés dans les feuilles. Ces résumés sont par la suite utilisés par la méthode MODL pour fournir à la fois un critère de coupure (ou de groupage) et un estimateur de densité par intervalle (ou groupe). Cette estimation permet par la suite de construire un classifieur Bayésien naïf dans les feuilles. Les différentes expérimentations ont montré que notre arbre était performant comparé à quelques méthodes de l'état de l'art. Tout particulièrement le classifieur local améliore toujours la prédiction en début d'apprentissage.

Des expérimentations plus approfondies sont nécessaires afin de pouvoir tester notre approche sur plus de jeux de données et dans un contexte de mémoire limitée. Par ailleurs il serait intéressant de pouvoir comparer notre méthode sur des flux de données réels plutôt que seulement sur des jeux de données générés artificiellement.

Références

Bifet, A., G. Holmes, B. Pfahringer, R. Kirkby, et R. Gavaldà (2009). New ensemble methods for evolving data streams. *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '09*, 139.

Bondu, A. et M. Boullé (2011). Détection de changements de distribution dans un flux de données : une approche supervisée. In *Extraction et gestion des connaissances (EGC'2011)*, pp. 191–196.

Boullé, M. (2005). A grouping method for categorical attributes having very large number of values. *Machine Learning and Data Mining in Pattern Recognition*, 228–242.

Boullé, M. (2006). MODL : A Bayes optimal discretization method for continuous attributes. *Machine Learning 65*(1), 131–165.

Cormode, G. et S. Muthukrishnan (2005). An improved data stream summary : the count-min sketch and its applications. *Journal of Algorithms 55*(1), 58–75.

Domingos, P. et G. Hulten (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 71–80. ACM New York, NY, USA.

Féraud, R., M. Boullé, F. Clérot, F. Fessant, et V. Lemaire (2010). The orange customer analysis platform. In *Industrial Conference on Data Mining (ICDM)*, pp. 584–594.

Gama, J. (2010). *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC Press.

Gama, J. et C. Pinto (2006). Discretization from data streams : applications to histograms and data mining. In *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 662–667.

Gama, J., R. Rocha, et P. Medas (2003). Accurate decision trees for mining high-speed data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 523–528. ACM New York, NY, USA.

Gama, J., P. Rodrigues, et R. Sebastião (2009). Evaluating algorithms that learn from data streams. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 1496–1500. ACM New York, NY, USA.

Greenwald, M. et S. Khanna (2001). Space-efficient online computation of quantile summaries. *ACM SIGMOD Record 30*(2), 58–66.

Guyon, I., V. Lemaire, G. Dror, et D. Vogel (2009). Analysis of the kdd cup 2009 : Fast scoring on a large orange customer database. *JMLR : Workshop and Conference Proceedings 7*, 1–22.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*.

Hulten, G. et P. Domingos (2003). VFML – a toolkit for mining high-speed time-changing data streams.

Hulten, G., L. Spencer, et P. Domingos (2001). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 97–106. ACM New York, NY, USA.

Kirkby, R. (2008). *Improving Hoeffding Trees*. Ph. D. thesis, University of Waikato.

Kohavi, R. (1996). Scaling up the accuracy of naive-Bayes classifiers : A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Volume 7. Menlo Park, USA : AAAI Press.

Lazarescu, M. M., S. Venkatesh, et H. H. Bui (2004). Using multiple windows to track concept drift. *Intelligent Data Analysis 8*(1), 29–59.

Pfahringer, B., G. Holmes, et R. Kirkby (2008). Handling numeric attributes in hoeffding trees. *Advances in Knowledge Discovery and Data Mining*, 296–307.

Ramos-Jimenez, G., J. del Campo-Avila, et R. Morales-Bueno (2006). Incremental algorithm driven by error margins. *Lecture Notes in Computer Science 4265*, 358.

Salperwyck, C. et V. Lemaire (2010). Impact de la taille de l'ensemble d'apprentissage : une étude empirique. In *Atelier CIDN de la conférence ECG 2011*, Brest.

Voisine, N., M. Boullé, et C. Hue (2009). Un critère d'évaluation Bayésienne pour la construction d'arbres de décision. *Extraction et gestion des connaissances (EGC'2009)*, 259–264.

Summary

New application domains generate data which are not any more persistent but volatile: network management, web profile modeling... These data arrive quickly, massively and are visible just once. It is thus necessary to learn them according to their arrivals. For classification problems on-line decision trees are known to perform well and are widely used on streaming data. In this paper, we propose a new decision tree method based on order statistics. The summary, usually used in the construction of an on-line tree, is here based on bounded error quantiles. A robust and performing discretization or grouping method uses this summary to provide at the same time a criterion to perform a split and interval density estimations. This estimation is then used to build a naive Bayes classifier in the leaves to improve the prediction in the early learning stage.