Arbres de décision flous adaptatifs

Wenlu Yang, Christophe Marsala, Andrea Pinna, Maria Rifqi, Patrick Garda

Arbres de décision flous adaptatifs

Adaptive Fuzzy Decision Trees

Wenlu Yang[1] Christophe Marsala[1] Andrea Pinna[1] Maria Rifqi[2] Patrick Garda[1]

[1] Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6, UMR7606, 4 place Jussieu, 75005 Paris, France
[2] Université Panthéon Assas, Univ. Paris 02, LEMMA, Paris, France
Email :{Prénom.Nom}@lip6.fr

Résumé :
Dans cet article, une approche d'adaptation dynamique d'un arbre de décision flou est présentée et des résultats d'expérimentations sont donnés montrant l'intérêt et l'efficacité de cette approche. De plus, la capacité d'adaptabilité des arbres ainsi que les limites de cette approche sont étudiées.

Mots-clés :
Arbres de décision flous adaptatifs, dérive de concept.

Abstract:
In this paper, an adaptive fuzzy decision tree approach is presented and a set of experiments are shown that highlight the interest and the efficiency of this approach. Adaptability of these trees and their limits are shown.

Keywords:
Adaptive fuzzy decision trees, concept drift.

1 Introduction

Depuis ces dernières années, de grands volumes de données sont générés par diverses applications dans différents domaines : finance, cybersécurité, réseaux de capteurs, etc. En particulier, la reconnaissance de concepts (ou classes) et la classification de données issues de flux est devenue un sujet de recherche important et très actif. Dans les applications réelles, de tels flux de données sont bruités et des changements relatifs aux concepts peuvent se produire au fil du temps, un phénomène de *dérive de classes* se produit alors [4]. En apprentissage automatique, de telles données engendrent de nouvelles problématiques pour la construction de modèles capables de prédire ces concepts. Il existe 2 types principaux de dérive [16] : les *changements brusques* et les *changements progressifs*. Lors d'un changement brusque, les concepts ou leurs frontières se trouvent modifiées radicalement et de façon rapide. Le modèle de prédiction appris doit alors oublier l'ancien concept pour pouvoir gérer les données récentes. La reconstruction complète du modèle à partir de ces données récentes est alors une solution efficace mais très coûteuse. Lors d'un changement progressif, les concepts sont modifiés graduellement et/ou localement. Le modèle appris devient alors inefficace sur les données impactées par ce changement, mais reste valide pour les zones stables des données. Il n'est alors pas nécessaire de reconstruire complètement le modèle pour en créer un nouveau car il suffit de l'adapter à partir des seules données impactées par les évolutions des concepts.

Nous nous intéressons à ces méthodes adaptatives et, en particulier, à l'apprentissage par arbre de décision qui est un modèle performant et interprétable. Ce modèle est pourtant sujet à une forte variance qui peut être atténuée par l'utilisation de la théorie des sous-ensembles flous et l'introduction des arbres de décision flous (ADF). Ainsi, des approches incrémentales ou adaptatives ont été proposées pour construire des arbres. L'approche la plus semble est basée sur un fenêtrage des exemples. Une fenêtre temporelle est fixée est les exemples arrivant dans cette fenêtre temporelle sont utilisés pour construire une arbre [13]. Avec cette méthode, le modèle est très représentatif des données récentes mais il peut perdre la connaissance "historique" fournie par les exemples hors de la fenêtre. De plus, la

reconstruction de l'arbre est coûteuse. Des alternatives permettant de conserver un historique et d'être plus rapide au changement sont proposées avec VFDT [5] et ID5 [17] Ces méthodes incrémentales introduisent une flexibilité de l'arbre de décision, mais entraînent une complexification de la structure de l'arbre, ce qui peut nuire à son interprétabilité. Il est alors intéressant de proposer une approche permettant de modifier légèrement l'arbre sans modifier grandement sa structure [5, 9, 10, 20].

Dans cet article, nous présentons une nouvelle approche et des expérimentations pour en étudier ses propriétés et ses limites. Dans la Section 2, un rappel est fait sur les ADF ainsi que sur l'approche de mise au point adaptative d'un arbre. Dans la Section 3, des expérimentations sur cette approche sont présentées. La Section 4 conclut et des perspectives sont proposées.

2 Arbres de décisions flous

Cette section présente un rappel sur les ADF, pour plus de détails, voir [7] ou [11].

2.1 Construction d'un ADF

Il existe plusieurs méthodes pour construire des ADF, la plupart sont de type "Top Down Induction of Decision Tree" [2] et [14], et introduisent, pour sélectionner les attributs de l'arbre, l'utilisation soit d'une entropie d'événements flous [15, 19, 6], soit d'une autre mesure (mesure de Gini, mesure d'ambiguïté, ou autre) [1, 3, 12, 18, 21].

Les éléments importants dans la construction d'un ADF sont les sous-ensembles flous qui apparaîtront dans l'arbre. Ils sont soit fournis par un expert [6, 21], soit déterminés dynamiquement durant la construction de l'arbre. Par exemple, dans le système *Salammbo* une approche utilisant un filtrage morphologique permet de déterminer une partition floue de l'ensemble des exemples d'apprentissage au niveau de chaque nœud [8, 11].

2.2 Classification avec un ADF

La méthode de classification avec un ADF est détaillée dans [7], nous la rappelons ici car elle est essentielle pour l'algorithme d'adaptation présenté ensuite. Lors de la classification d'un nouvel exemple, un chemin de la racine de l'arbre vers une feuille est équivalent à une règle de décision. La prémisse de cette règle est constituée des tests sur les attributs rencontrés sur le chemin, et sa conclusion est la valeur de la décision associée à la feuille. Une feuille d'un ADF peut être associée à un ensemble de classes $\{c_1, \ldots, c_K\}$, chaque valeur c_j étant associée à un poids déterminé lors de l'apprentissage. Ainsi, un chemin est équivalent à une règle : *si* $A_{l_1} = v_{l_1}$ *et ... et* $A_{l_p} = v_{l_p}$ *alors* $C = c_1$ *avec le degré* $P^*(c_1|(v_{l_1}, \ldots v_{l_p}))$ *et ... et* $C = c_K$ *avec le degré* $P^*(c_K|(v_{l_1}, \ldots v_{l_p}))$, où P^* est la mesure de probabilité d'un événement flou [22]. Chaque valeur v_i de cette règle peut être soit précise, soit floue, et elle est décrite par une fonction d'appartenance μ_{v_i} de l'ensemble des exemples vers $[0, 1]$.

Lors de la classification d'un nouvel exemple, sa description $\{A_1 = w_1; ...; A_n = w_n\}$ est comparée à la prémisse de chaque règle r et on mesure le *degré de similarité* $\texttt{Deg}(w_{l_i}, v_{l_i})$ entre w_{l_i} et la valeur v_{l_i} correspondante. On associe alors à l'exemple un sous-ensemble flou sur l'univers des classes. L'appartenance à la classe c_j est mesurée par le *degré final* $\texttt{Fdeg}_r(c_j)$ obtenu par agrégation des degrés $\texttt{Deg}(w_{l_i}, v_{l_i})$ à l'aide de la t-norme *minimum* [1]

$$\texttt{Fdeg}_r(c_j) =$$

$$\min_{i=1\ldots p} \texttt{Deg}(w_{l_i}, v_{l_i}) \cdot P^*(c_j|(v_{l_1}, \ldots v_{l_p}))$$

La classe finale de l'exemple est obtenue en combinant tous les chemins activés de l'arbre, chaque classe c_j est associée à un degré d'appartenance $\texttt{Fdeg}(c_j)$, de $[0, 1]$ déterminé par :

$$\texttt{Fdeg}(c_j) = \max_{r=1\ldots n_\rho} \texttt{Fdeg}_r(c_j) \qquad (1)$$

1. Toute autre t-norme peut être utilisée [7].

avec n_ρ le nombre de chemins de l'arbre.

Au final, afin d'obtenir une décision tranchée, on associe généralement à l'exemple la classe c_k qui possède le plus haut degré $\texttt{Fdeg}(c_k)$.

2.3 Adaptation d'un ADF

L'adaptation d'un ADF s'effectue quand celui-ci n'est plus suffisamment précis pour classer de nouveaux exemples [9]. Soit e un nouvel exemple à classer (dont la vraie classe est fournie pour contrôle). Lors de sa classification par un arbre $\mathcal{T}$, deux cas peuvent se produire : soit e est bien classé et dans ce cas $\mathcal{T}$ est correct ; soit e est mal-classé et dans ce cas $\mathcal{T}$ doit être adapté afin de mieux prendre en compte de futurs exemples similaires à e. Il existe 2 types de mauvaises classifications : soit e est mal-classé dans c_k avec un degré $\texttt{Fdeg}(c_k) = 1$; soit e est mal-classé dans c_k avec un degré $\texttt{Fdeg}(c_k) < 1$. Dans ces deux cas, selon la méthode d'agrégation vue dans la section précédente, le degré de classification final $\texttt{Fdeg}(c_k)$ est fourni par une seule feuille (celle qui fixe la valeur de $\texttt{Fdeg}(c_k)$).

Dans le premier cas, la décision finale erronée vient d'un problème structurel de $\mathcal{T}$: e est un point dans l'espace de représentation des données qui se trouve loin de toute frontière de séparation des classes induites par l'arbre. Adapter $\mathcal{T}$ à cet exemple demande de modifier sa structure globale. Dans le second cas, e se trouve dans une région de l'espace de description proche d'une frontière de séparation des classes, dans la zone floue de cette frontière. Corriger l'erreur de classification pour e est alors possible en modifiant légèrement les partitions floues concernées. cela constitue le type d'adaptation de l'arbre considérée dans cet article : nous proposons de modifier la partition floue d'un nœud de l'arbre après une telle erreur de classification. La règle qui a donné la valeur au plus haut degré $\texttt{Fdeg}(c_k)$ est sélectionnée. Dans cette règle r, on peut mettre en évidence le nœud qui a fourni la valeur $\texttt{Fdeg}_r(c_k)$ du plus petit degré. Ce nœud est celui sur lequel la mise

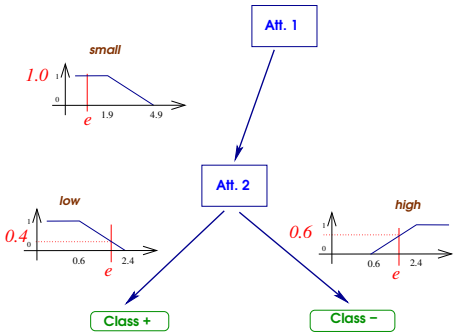

Figure 1 – Exemple de classification

à jour est réalisée : la partition floue de ce nœud est modifiée pour pallier l'erreur de classification.

Par exemple, dans la figure Fig. 1, e est associé à un degré d'appartenance de 1 au premier arc issu de la racine de l'arbre et à un degré 0.4 à l'arc gauche sortant du nœud suivant ainsi qu'au degré 0.6 correspondant à l'arc droit de ce noœud. Pour les 2 chemins de la racine aux feuilles représentés dans la Fig. 1, on a pour la branche gauche, $\texttt{Fdeg}_l(c_+) = \min(1, 0.4) = 0.4$ et pour la branche droite, $\texttt{Fdeg}_h(c_-) = \min(1, 0.6) = 0.6$. Au final, la classe associée à e est c_- avec le degré $\max(0.4, 0.6) = 0.6$. Si e est mal classé, la partition floue associée au nœud de l'attribut $Att.2$ doit être mise à jour car c'est elle qui a amené la valeur finale du degré d'appartenance à c_-, c'est-à-dire 0.6.

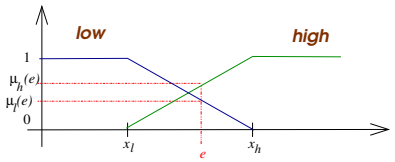

Figure 2 – Partition floue originale

Dans la partition [2] de la Fig. 2, le sous-ensemble flou "low" est défini par sa plus grande valeur de noyau x_l et sa plus grande valeur de support x_h.

Soit e l'exemple à classer, et notons e aussi sa valeur pour l'attribut concerné. On note $\mu_l(e)$

2. Par souci de simplicité, nous illustrons ce point avec une partition binaire mais tous types de partitions sont possibles.

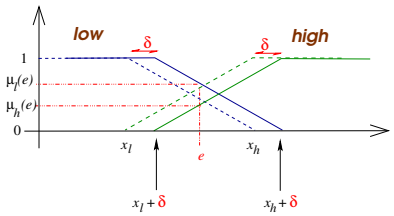

Figure 3 – Partition adaptée par translation

le degré d'appartenance de e au sous-ensemble flou "low", et $\mu_h(e)$ son degré d'appartenance à "high". Dans la Fig. 2, on considère $\mu_l(e) < \mu_h(e)$. Comme e est mal classé, on modifie la partition floue afin de pouvoir avoir $\mu_l(e) > \mu_h(e)$, ce qui est susceptible d'offrir une classification correcte. Pour cela, les fonctions d'appartenance sont translatées comme suit (voir Fig. 3).

Soit $d = sgn(\mu_h(e) - \mu_l(e))$ la direction de la translation. Quand $d = 1$, la partition est translatée vers la droite, sinon elle est translatée vers la gauche. L'amplitude de la translation vaut $\delta = \dfrac{\max(e - x_l, x_h - e)}{1 + f}$ avec $f \geq 0$. Ce paramètre définit l'amplitude de la translation : plus f est petit, plus la translation est importante. La valeur de δ conditionne l'augmentation du degré d'appartenance de e au sous-ensemble flou qui lui donnait le plus petit degré d'appartenance avant l'adaptation. Quand $f = 0$, l'amplitude est maximale ce qui permet à e d'être mis entièrement dans le sous-ensemble flou. Quand $f > 0$, la translation est plus graduelle et permet de positionner la frontière de séparation des classes plus près de e et d'augmenter donc son degré d'appartenance.

Dans l'approche proposée, l'adaptation est faite pour les exemples mal classés avec un degré $\texttt{Fdeg}(c_k) < 1$. Comme dit précédemment, ces exemples sont proches de la frontière de décision, une zone souvent plus complexe à caractériser et dans laquelle les exemples sont plus difficiles à classer. Dans cette zone, avec le procédé d'adaptation, chaque exemple mal classé contribue au déplacement de la frontière de décision. En absence de bruit, suite à ces changements mineurs, la frontière de décision peut converger vers une frontière idéale.

D'autre part, un exemple mal classé avec un degré $\texttt{Fdeg}(c_k) = 1$ est loin de la frontière de décision. Cela doit donc être soit un exemple aberrant, soit un exemple produit par la dérive de classes. Dans notre approche, aucune adaptation n'est faite pour de tels exemples, ceci afin d'éviter d'avoir un modèle trop sensible aux exemples aberrants ou un modèle qui possède une variance trop importante. Pour limiter un sur-apprentissage, sans utiliser une approche de type "batch", notre approche profite des caractéristiques de robustesse et de gradualité des ensembles flous pour ajuster plus finement la frontière de décision. L'intérêt majeur de cette façon de faire est qu'elle ne nécessite alors aucune ressource supplémentaire pour le stockage des données ou pour maintenir des statistiques conséquentes sur l'historique des exemples.

3 Étude expérimentale

Dans cette section, un ensemble d'expérimentations sont présentées pour tester les performances de l'approche proposée et mettre en évidence sa capacité à s'adapter à une évolution dans les données.

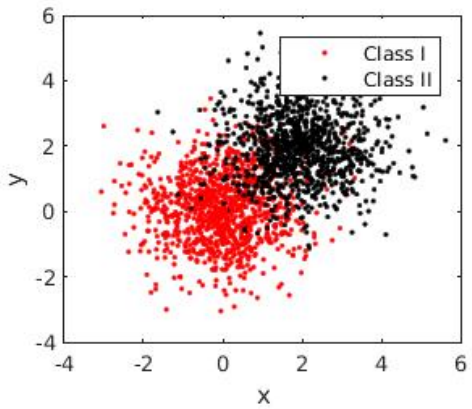

Figure 4 – Jeu de données synthétiques

3.1 Jeux de données utilisés

Un jeu de données a été généré à partir de 2 classes évoluant graduellement. Les données de chaque classe sont générées à partir d'une distribution normale multivariée [3] (Fig. 4). Pour simuler une dérive de classes dans les données, plusieurs changements peuvent être appliqués aux données (rapprochement, éloignement, translation). Cela permet de mettre en évidence d'une part la chute de performance provoquée par une dérive de classes, ainsi que le rétablissement de cette performance grâce à l'adaptabilité du modèle. Des changements tels un rapprochement ou un éloignement des classes modifient la difficulté du problème : si les classes sont proches, il est plus difficile de les distinguer que si elles sont éloignées. Par conséquent, on considère ici des translations conjointes des classes.

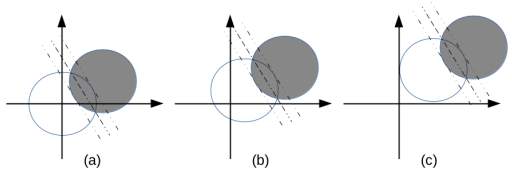

Figure 5 – Génération de données

Le processus de génération des données est le suivant. Au départ, les 2 classes étudiées sont centrées respectivement en $(0, 0)$ et en $(2, 2)$ (Fig. 5 (a)). Les données sont générées pour ces 2 classes à partir de ces centres. Ensuite, les positions des centres sont translatées conjointement en $(0.3, 0.3)$ et $(2.3, 2.3)$ respectivement (Fig. 5 (b)). Une dernière translation est effectuée pour placer les centres en $(0.6, 0.6)$, $(3.6, 3.6)$ (Fig. 5 (c)). Une telle translation des classes permet d'analyser l'adaptation aux changements et la préservation des performances en présence de dérive de classes. On étudie d'abord la performance de l'ADF sans dérive de classes. On choisit les données issues du premier tirage. Un ADF est créé avec ces données en utilisant le logiciel Salammbo [11]. La base d'apprentissage est divisée en 10 paquets, un paquet est choisi comme base de test, et les autres comme base d'apprentissage (en utilisant d'abord 1 paquet, puis 2,... pour construire l'arbre). La variation du taux de bonne classification selon le nombre d'exemples est représentés dans la Fig. 6. On remarque qu'en augmentant le nombre d'exemples, la performance de l'arbre atteint une limite (mais la taille de arbre et le temps de construction augmentent, eux, toujours). Le même résultat est observé sur les données des deuxième et troisième tirage.

3. Nous illustrons ici sur un problème de classification pour 2 classes en 2 dimensions.

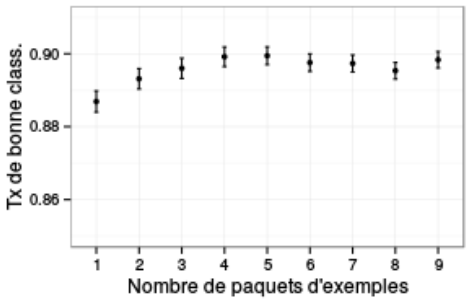

Figure 6 – Variation de la performance

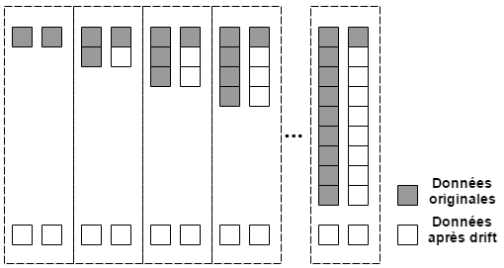

Figure 7 – Protocole : effets d'une dérive de classes

Impacts d'une dérive des classes. On distingue deux états dans l'expérimentation, l'état original, et l'état après la dérive de classes. Dans l'état original, un ADF est créé. Sur ce modèle, la méthode d'adaptation est utilisée pour voir si le modèle est capable de suivre le changement dans les données. Trois types d'expérimentations sont réalisées : une valida-

tion de l'approche, une validation de l'adaptabilité du modèle au fil du temps et une étude des limites de cette méthode.

Pour montrer la validité de l'approche, la performance du modèle original sans adaptation est comparée à celle du modèle avec adaptation après dérive de classes. Une validation croisée est réalisée pour évaluer cela. La Fig. 7 présente le protocole de cette expérimentation. Chaque carré présent un paquet de 200 exemples. Les données générées dans l'état original sont en gris, et celles tirées après la dérive de classes sont en blanc. Chaque bloc (ligne pointillée) présente un groupe d'expérimentations. Un modèle sans adaptation est créé à partir des données présentées à gauche d'un bloc, et le modèle comparé, mis au point par adaptation, est présenté à droite. Le modèle adaptatif est créé avec des données dans l'état original et adapté avec des données dans l'état après dérive de classes. Les deux modèles sont testés sur des données après la dérive de classes (carrés en bas, présenté en blanc). Pour montrer l'effet de l'adaptation, on incrémente le nombre de exemples d'apprentissage par paquet à chaque expérimentation. Plusieurs tests sont réalisés pour chaque cas en échangeant l'ordre des paquets.

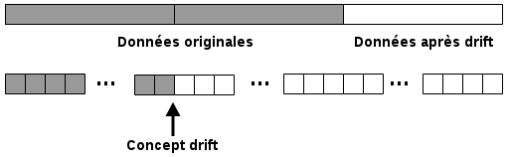

Figure 8 – Protocole : adaptabilité de la méthode

Adaptabilité de la méthode. Pour montrer l'adaptabilité de la méthode, on teste sa performance sur des données avec une dérive de classes. Le modèle de départ est créé à partir des données tirées avec le concept original (400 exemples générés). La Fig. 8 illustre le protocole de cette expérimentation. Le modèle reçoit une séquence de données (présentée par la barre en haut de la figure). Chaque fois qu'un nouveau exemple arrive et est mal classé, une adaptation est faite. L'arbre est ensuite testé avec un paquet d'exemples (la séquence des paquets test est présentée en bas de la figure). Afin de mettre en évidence l'adaptabilité, 3 scénarios sont intégrés dans ce protocole. D'abord le modèle reçoit des données du concept original et est testé sur un paquet de données de ce concept original. Ensuite, le modèle continue de recevoir des données du concept original mais il est testé sur des données du nouveau concept. A la fin, le modèle reçoit des données du nouveau concept et il est testé sur des paquets de données du nouveau concept. Chaque scénario prend 1000 exemples, et la performance est testée sur 1000 exemples. Cela permet de montrer la capacité du modèle à s'adapter et à préserver ses performances en présence d'une dérive de classes.

Limites de la méthode. L'intérêt de cette méthode est de pouvoir adapter un ADF en cas de dérive de classes graduel. Dans cette expérimentation nous étudions les limites d'une telle approche avec des tests dans des conditions de changements différentes. Nous reprenons le protocole utilisé dans le test de l'adaptabilité et faisons une comparaison des performance entre des dérives de classes d'intensité graduelle, moyenne et brusque.

3.2 Résultats

Validation de la méthode. La Fig. 9 présente le résultat obtenu par le protocole de validation. On peut remarquer que la performance du modèle avec adaptation est toujours meilleure que celle du modèle sans adaptation. La méthode de mise au point de l'arbre est donc efficace. De plus, la méthode avec adaptation a une variance plus faible quand le nombre d'exemples augmente ce qui s'explique par la simplicité du modèle construit au départ qui permet d'éviter ainsi un sur-apprentissage.

Adaptabilité de la méthode. La Fig. 10 présente le résultat que l'on obtient avec le protocole d'adaptabilité. La ligne noire présente la performance du modèle sans adaptation. La

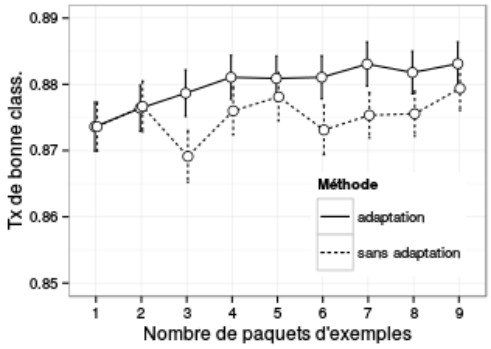

Figure 9 – Validation de la méthode

ligne rouge présente la performance du modèle avec adaptation ($f = 3$). Malgré une fluctuation du taux de bonne classification qui se produit en raison de l'adaptation, la variation est de l'ordre de 0.1% ce qui reste acceptable. Pour faciliter la visualisation, Une autre courbe, en rouge, lisse la courbe précédente et présente la performance moyenne.

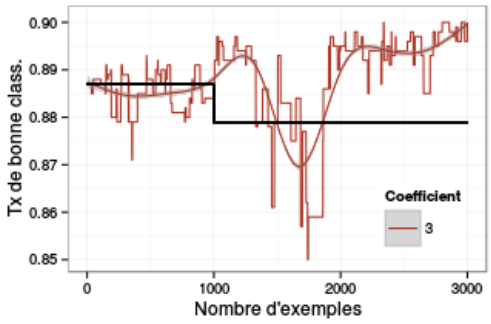

Figure 10 – Performances en présence de dérive de classes

Au millième exemple, un drift est introduit. Il y a alors une chute de performances pour le modèle sans adaptation, alors que le modèle avec adaptation commence à s'adapter aux exemples reçus. Entre 1000 à 2000, les exemples sont générés avec le concept original. La variance de la performance est très grande car le modèle essaye de s'adapter au changement. Mais comme l'adaptation est basée sur des données du concept original et que le test est réalisé sur des données du nouveau concept, le modèle n'arrive pas à trouver la nouvelle frontière de décision. Par conséquent, la performance de l'arbre chute brusquement. Ensuite, à partir de 2000 exemples, les données arrivant sont issues du nouveau concept, le modèle s'adapte avec ces nouvelles données au fur et au mesure et finit par converger vers une performance assez stable.

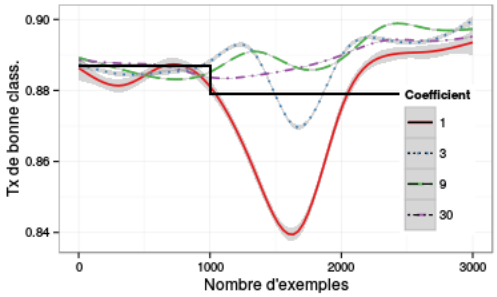

Figure 11 – Evolution de la performance en présence de dérive de classes

Limites de la méthode. Un paramètre important est le coefficient f. La Fig. 11 présente l'effet de différentes valeurs de f sur l'adaptabilité du modèle. On peut remarquer que, quelle que soit la valeur de f, les modèles restent capables de s'adapter au drift et de maintenir leurs performances. Avec une petite valeur de f, le changement du modèle est important ainsi que la variance de sa performance, alors qu'avec une valeur plus grande, le modèle est plus stable, mais prend plus de temps pour converger vers de bonnes performances.

La Fig. 12 présente la performance pour des dérives de classes d'intensités différentes. L'état initial correspond aux centres $(0, 0)$ et $(2, 2)$, puis différentes intensités de drift sont introduites : drift graduel (centres décalés en $(0.3, 0.3)$ et $(2.3, 2.3)$), drift moyen (centres décalés en $(0.6, 0.6)$ et $(2.6, 2.6)$), drift brusque (centres décalés en $(1, 1)$ et $(3, 3)$). On peut constater que, avec l'adaptation, les performances des modèles s'améliorent. Par contre, pour des drift trop importants, le modèle ne peut

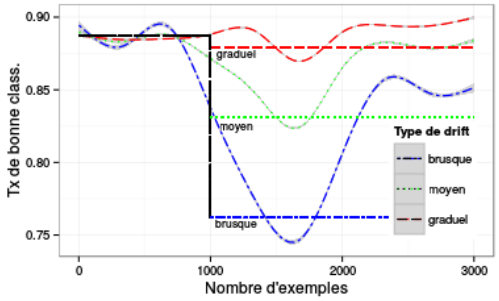

Figure 12 – Impact de l'intensité du drift

plus retrouver sa performance initiale. Dans cette situation, la performance du modèle est dégradée car un tel cas nécessite une adaptation plus globale.

4 Conclusion

Dans cet article une nouvelle approche pour l'adaptation d'un ADF est étudiée. Des expérimentations sont présentées pour valider cette approche. L'adaptabilité ainsi que les limites de cette approche sont étudiées et montre l'efficacité de cette nouvelle approche.

En perspective, des études sont encore à réaliser sur le meilleur moyen de modifier les fonctions d'appartenance lors de l'adaptation, ainsi que sur le choix des nœuds à modifier dans l'arbre. Il sera aussi intéressant de coupler cette adaptation locale à une approche plus globale afin de pouvoir gérer des dérives de classes importants.

Remerciements :

Ce travail a été réalisé dans le cadre du projet SENSE du labex SMART (ANR-11-LABX-65), financé par un Investissements d'Avenir (référence ANR-11-IDEX-0004-02).

Références

[1] X. Boyen and L. Wehenkel. Automatic induction of fuzzy decision tree and its application to power system security assessmen. *Fuzzy Sets and Systems*, 102(1) :3–19, 1999.

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

[3] K. Cios and L. Sztandera. Continuous ID3 algorithm with fuzzy entropy measures. In *Proc. of the 1st IEEE Int. Conf. on Fuzzy Systems*, San Diego, 1992.

[4] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4) :44 :1–44 :37, Mar. 2014.

[5] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc. of the 7th ACM SIGKDD int. conf. on Knowledge discovery and data mining*, pages 97–106. ACM, 2001.

[6] C. Z. Janikow. Fuzzy decision trees : Issues and methods. *IEEE Trans. on Systems, Man and Cybernetics*, 28(1) :1–14, 1998.

[7] C. Marsala. *Apprentissage inductif en présence de données imprécises : construction et utilisation d'arbres de décision flous.* Thèse de doctorat, Univ. P. et M. Curie, Paris, France, Janvier 1998.

[8] C. Marsala. Fuzzy Partitioning Methods. In W. Pedrycz, editor, *Granular Computing : an Emerging Paradigm*, pages 163–186. Springer-Verlag, 2001.

[9] C. Marsala. Incremental tuning of fuzzy decision trees. In *Proc. of the 6th Int. Conf. on Soft Computing and Intell. Systems (SCIS-ISIS)*, pages 2061–2064, Kobe, Japan, Nov. 2012.

[10] C. Marsala. Fuzzy decision trees for dynamic data. In *Evolving and Adaptive Int. Systems (EAIS), 2013 IEEE Conf. on*, pages 17–24. IEEE, 2013.

[11] C. Marsala and B. Bouchon-Meunier. An adaptable system to construct fuzzy decision trees. In *Proc. of the NAFIPS'99*, pages 223–227, New York, UA, 1999.

[12] C. Olaru and L. Wehenkel. A complete fuzzy decision tree technique. *Fuzzy sets and systems*, 138(2) :221–254, 2003.

[13] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1) :81–106, 1986.

[14] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1) :86–106, 1986.

[15] M. Ramdani. Une approche floue pour traiter les valeurs numériques en apprentissage. In *Journ. Franc. d'apprentissage et d'explication des connaissances*, 1992.

[16] A. Tsymbal. The problem of concept drift : definitions and related work. *Computer Science Dpt., Trinity College Dublin*, 106, 2004.

[17] P. E. Utgoff. ID5 : an incremental ID3. *Machine Learning Proceedings 1988*, page 107, 2014.

[18] X. Wang, B. Chen, G. Qian, and F. Ye. On the optimization of fuzzy decision trees. *Fuzzy Sets and Systems*, 112(1) :117–125, May 2000.

[19] R. Weber. Fuzzy-ID3 : A class of methods for automatic knowledge acquisition. In *IIZUKA'92 Proc. of the 2nd Int. Conf. on Fuzzy Logic*, pages 265–268, 1992.

[20] H. Yang and S. Fong. Incremental optimization mechanism for constructing a decision tree in data stream mining. *Math. Problems in Engineering*, 2013, 2013.

[21] Y. Yuan and M. Shaw. Induction of fuzzy decision trees. *Fuzzy Sets and systems*, 69 :125–139, 1995.

[22] L. Zadeh. Probability measures of fuzzy events. *Journal Math. Anal. Applic.*, 23, 1968.