Exploration des mémoires à court et long terme pour la classification multi-labels en flux

Xihui Wang, Pascale Kuntz, Frank Meyer

Exploration des mémoires à court et long terme pour la classification multi-labels en flux

Xihui WANG*,**, Pascale KUNTZ**, Frank MEYER*

*Orange Labs, 2 Avenue Pierre Marzin, 22300 Lannion
prenom.nom@orange.com,
**Laboratoire des Sciences du Numérique de Nantes - Site Polytech - 44300 Nantes
prenom.nom@univ-nantes.fr

Résumé. La classification multi-labels, dans laquelle un texte, une image ou une cyber-attaque, par exemple, peuvent être associés à plusieurs labels simultanément devient de plus en plus nécessaire dans les applications récentes. Lorsque les besoins de réactivité sont eux-mêmes cruciaux, la classification de flux de données devient un enjeu important. Nous proposons dans cet article un nouvel algorithme, Online Memory k-means (OMk), pour traiter la problématique de la classification en flux. OMk est un modèle de type k-plus-proches-voisins qui utilise deux types de mémoire, l'une court-terme basée sur une fenêtre glissante FIFO, et l'autre long-terme, basée sur un échantillonnage en réservoir. Ces deux mémoires permettent de gérer les flux de données avec des dérives de concepts et de pouvoir résister au phénomène d'oubli catastrophique. En utilisant ces structures de données simples avec des tailles de mémoire relativement limitées et en considérant l'information portée par les corrélations entre labels, notre algorithme est compétitif avec les algorithmes actuels de l'état de l'art, EaHTps et MLSAMPKNN, à la fois en qualité de prédiction et en temps de réponse. La faible complexité d'OMk et les performances obtenues nous permettent d'envisager son extension à la classification multi-labels extrême de données en flux, qui est un problème encore peu exploré.

1 Introduction

Le paradigme de la classification multi-labels est une extension du problème mono-label à plusieurs labels qui sont extraits d'un ensemble prédéfini de possibilités : il s'agit d'associer un objet décrit par un vecteur de variables à un sous-ensemble restreint de concepts d'intérêt, appelés "labels". Pour illustration, en annotation de textes, on peut qualifier un texte à la fois de "drôle", "en français" et "consacré au cinéma". La classification multi-labels a connu un fort essor cette dernière décennie stimulée par de nombreuses applications : par exemple, en biologie pour la classification des fonctions des gènes et des protéines (Clare et King, 2001), en science de l'information pour la catégorisation de textes (Kong et Philip, 2011a) ou en multimédia pour la catégorisation et l'annotation d'images, de vidéos et de musiques (Madjarov et al., 2012). Depuis les travaux pionniers de Boutell et al. (2004), Tsoumakas et al. (2009)

et Zhang et Zhou (2007), de nombreux algorithmes ont été proposés. Ces recherches ont permis d'améliorer significativement la qualité des résultats sur des données de taille restreinte. Plus récemment, la croissance du volume des données recueillies a conduit à un changement d'échelle : on parle désormais de classification multi-labels extrême (XMLC - eXtreme Multi-Labels Classification) pour un ordre de grandeur des variables et des labels entre 10^4 et 10^7. Par exemple, dans un challenge de la communauté scientifique, la classification portait sur des articles de Wikipédia avec un corpus de 2.4 millions d'articles encodés par des sacs n-grams de 1.6 millions de variables où la cible des labels comportait $350\,000$ possibilités (Partalas et al., 2015). Ce problème s'étend aujourd'hui aux données en flux qui doivent être classifiées au fil de l'eau. Leur essor est porté à la fois par leur présence qui ne cesse de croître, par des contraintes de réactivité (ex : la détection de cyber-attaques décrites par des modalités multiples) et par les nouvelles volumétries de données qu'il est parfois plus efficace de traiter selon un modèle de flux à travers une seule passe d'apprentissage.

Au problème générique de classification multi-labels en flux qui consiste à apprendre à chaque instant t un nouveau modèle à partir du précédent, peuvent s'ajouter quatre propriétés (Bifet et Gavaldà, 2009) : (i) chaque instance arrive séquentiellement, (ii) il existe potentiellement une infinité d'exemples, (iii) la distribution statistique des exemples n'est pas nécessairement stationnaire et les ensembles de labels et d'attributs peuvent évoluer dans le temps, et (iv) après son traitement chaque exemple est supprimé ou archivé dans une mémoire de taille très limitée. Vue l'importance des enjeux, différents algorithmes de classification multi-labels de données en flux ont été proposés ces dernières années mais les résultats expérimentaux portent généralement sur des données de taille restreinte et, à notre connaissance, aucun d'eux ne satisfait les propriétés précédentes. Pour soulever ce défi, nous procédons en deux étapes successives : (1) développer un algorithme compétitif de faible complexité satisfaisant les objectifs sur les jeux de données de taille standard, puis (2) prochainement intégrer cet algorithme dans un processus récursif basé sur des forêts aléatoires pour changer d'échelle. Cet article se focalise sur la première étape : nous proposons un composant algorithmique (OMk - Online Memory kmeans) qui sera intégré à l'avenir récursivement via une structure arborescente dans un modèle visant à traiter des volumétries très grandes. Le composant OMk a été conçu pour tenter de surmonter deux problèmes difficiles de l'apprentissange en ligne : l'oubli catastrophique (French, 1999) et la dérive de concept (Gama et al., 2014). Notre stratégie consiste à combiner deux mémoires de temporalités différentes et de faible complexité : un échantillonnage en réservoir (Vitter, 1985) couplé à un apprentissage basé sur un K-means online et une fenêtre glissante. De plus, en s'inspirant d'un des meilleurs algorithmes de l'état de l'art en classification XMLC (Siblini et al., 2018), notre modèle considère conjointement les corrélations entre les labels et les relations entre les labels et les attributs.

Nous comparons OMk avec l'un des algorithmes actuels les plus performants pour les données multi-labels en flux MLSAMPKNN (Roseberry et al., 2019) et deux algorithmes basés sur un arbre de Hoeffding, ISOUPTree (Osojnik et al., 2017) et HTps (Read et al., 2012). Les résultats expérimentaux sur 11 jeux de données classiques comprenant entre $O(10)$ et $O(1000)$ attributs et entre $O(10)$ et $O(100)$ labels confirment que OMk permet d'obtenir des performances compétitives avec une mémoire restreinte et qu'il mérite donc d'être exploité dans des travaux futurs en XMLC que nous discutons.

2 Etat de l'art

Dans la suite, nous considérons un flux de données $D = \{(x_0, y_0), (x_1, y_1)..., (x_t, y_t), ...\}$ où (x_t, y_t) est un exemple arrivant au temps t. Plus précisément, chaque instance est notée $x_t = [x_{t,j}]_{j\in[1,M]} \in R^M$ où M est le nombre d'attributs, et chaque ensemble de labels associé à x_t est noté $y_t \subset \{\lambda_j\}_{j\in[1,L]}$ où L est le nombre de labels possibles. En pratique, les labels sont décrits par un vecteur binaire $y_t = [y_{t,j}]_{j\in[1,L]} \in \{1, 0\}^L$: si le label λ_j est lié à x_t, alors $y_{t,j}$ = 1, sinon, $y_{t,j}$ = 0. Le problème générique de classification multi-labels en flux consiste à calculer à chaque instant t un nouveau modèle $h_t(x)$, basé sur le modèle précédent $h_{t-1}(x)$ et le nouvel exemple (x_t, y_t) observé entre les temps t et $t + 1$, qui optimise une mesure de qualité de la prédiction.

De façon générale, les approches de la littérature pour ce problème peuvent se structurer en trois grandes familles issues des états de l'art initiaux de la classification multi-labels : (i) les approches d'apprentissage par transformation qui transforment le problème multi-labels en plusieurs problèmes mono-label, (ii) les approches d'apprentissage par adaptation qui adaptent les algorithmes en-ligne mono-label au traitement des données multi-labels, et (iii) les approches "ensemble" qui combinent par des techniques de "bagging" ou de "boosting" des résultats à partir d'un ensemble de modèles plus simples. Nous reprenons ici cette organisation pour présenter les algorithmes multi-labels en flux (tableau 1).

Par transformation	**Par adaptation**	**Ensemble**
DWA (Kong et Philip, 2011b)	HTps (Read et al., 2012)	DWM (Kolter et Maloof, 2007)
MINAS-BR (Júnior et al., 2019a)	ISOUPTree (Osojnik et al., 2017)	EaHTps (Read et al., 2012)
MINAS-PS (Júnior et al., 2019b)	ML-AMRules (Sousa et Gama, 2018)	SWMEC (Wang et al., 2017)
MuENL (Zhu et al., 2018)	OSML-ELM (Venkatesan et al., 2017)	EaISOUPTree (Osojnik et al., 2017)
ML-Bayesian (Nguyen et al., 2019)	MLSAMPKNN (Roseberry et al., 2019)	GOOWE-ML (Büyükçakir et al., 2018)

TAB. 1 – *Classification des algorithmes multi-labels en flux.*

Les approches par transformation s'appuient sur des stratégies variées : construire un classifieur binaire pour chaque label (Woloszynski et Kurzynski, 2011) et (Júnior et al., 2019a), construire une fonction de score pour chaque label avec un seuil commun puis minimiser une fonction de perte de classement (Zhu et al., 2018), construire k classifieurs pour prédire k-labels en chaîne de sorte que la prédiction du label courant dépende de l'entrée et des prédictions des précédents classifieurs (Nguyen et al., 2019), considérer chaque combinaison de labels comme une nouvelle classe puis résoudre le problème multi-labels comme un problème multi-classes (Júnior et al., 2019b). Certaines de ces méthodes de transformation sont simples à mettre en oeuvre mais leur faiblesse majeure est leur absence de prise en compte des corrélations entre les labels. A contrario ***les approches par adaptation*** construisent un modèle pour tous les labels en adaptant les algorithmes mono-label en-ligne existants. Ces méthodes com-

binent souvent différentes stratégies de transformation ; par exemple, une méthode basée sur un classifieur binaire peut être appliquée sur les feuilles des arbres de Hoeffding (Read et al., 2012). ***Les approches ensembles*** sont composées de plusieurs modèles "simples" pour tenter de former par agrégation un modèle avec des performances de prédiction plus élevées. Elles s'appuient notamment sur l'attribution de poids optimisés aux classifieurs pour améliorer la qualité des performances (Büyükçakir et al., 2018). Une adaptation rapide par réinitialisation du plus mauvais classifieur (Read et al., 2011) peut également être introduite pour surmonter le problème de dérive de concept. Ces méthodes peuvent être coûteuses en temps de calcul. Dans leur grande majorité, les approches que nous avons listées se focalisent généralement sur une contrainte particulière de la classification en flux multi labels. Dans cet article, au contraire, nous nous focalisons simultanément sur trois impératifs : (i) la prise en compte des corrélations entre labels, (ii) l'équilibre entre l'oubli (dérive de concept) et la mémoire (oubli catastrophique), et (iii) la rapidité de l'apprentissage sans détriment de sa qualité.

3 Module algorithmique "Online Memory k-means" (OMk)

Pour la gestion des données en flux sous la contrainte de mémoire limitée, le module OMk se compose de deux mémoires complémentaires : une mémoire "à long terme" associée à un échantillonnage en réservoir et une mémoire "à court terme" basée sur une fenêtre temporelle glissante de taille fixe. Ces deux composantes visent à affronter en synergie les problèmes d'oubli catastrophique et de dérive de concept bien connus en apprentissage de flux. Le réservoir permet de conserver à tout instant un échantillonnage uniforme sur le flux quelle que soit sa taille. Cependant, lorsque la taille T du flux augmente la fréquence de l'échantillonnage s'atténue, et l'apprentissage de nouvelles données devient moins efficace. Pour éviter l'écueil de la dérive de concept, nous introduisons une mémoire à court terme définie par une fenêtre temporelle glissante de type FIFO qui mémorise les N dernières instances au fil de l'eau.

Notation	Description
k	Nombre de réservoirs / Le nombre de séparateurs
S	Taille du réservoir d'échantillonnage
T_i	Nombre de données traitées par l'ième réservoir au temps t, $i \in [1, k]$
N	Taille de la fenêtre temporelle glissante FIFO
n	Nombre de plus proches voisins pour la prédiction
$N\_initial$	Taille de l'initialisation d'OMk ($N\_initial \geq \max(k, N)$)

TAB. 2 – *Les paramètres d'OMk*

Mémoire à long terme. Il s'agit d'une structure en arbre de profondeur 1 où la racine de l'arbre joue le rôle d'un k-séparateur lié à k feuilles. Chaque feuille représente un réservoir $R_i = \{(x_{i,j}, y_{i,j})\}_{j\in[1,S]}$ de taille S qui mémorise la T-ième instance considérée dans le flux selon une règle d'échantillonnage : avec une probabilité de 1 si $T \leq S$, et de S/T sinon. Une nouvelle instance mémorisée est stockée dans le réservoir à un index r tiré au hasard de manière uniforme dans $[1, ..., S]$; ce qui conduit à oublier l'instance précédemment stockée à l'index r (Vitter, 1985). Chaque séparateur $\{((c_i^{(x)}, c_i^{(y)})\}_{i\in[1,k]}$ est représenté par un couple

de centroïdes des attributs et des labels des mémoires stockées dans le réservoir. Lorsqu'un changement est effectué dans un réservoir, les centroïdes du séparateur sont mis à jour.

Mémoire à court terme. Une fenêtre temporelle glissante FIFO de taille N restreinte $F_t = \{(x_{t-i}, y_{t-i})\}_{i\in[1,N]}$ qui supprime la donnée la plus ancienne, puis stocke la dernière donnée à chaque instant t.

Initialisation. Le module OMk collecte les $N\_initial$ premières données du flux (noté $Initial\_Stream$) pour initialiser ses deux mémoires. Pour la mémoire à long terme et ses centroïdes associés, l'initialisation procède comme suit : un premier clustering en k clusters est effectué, sur l'espace des labels, sur toutes les données (x_t, y_t) de $Initial\_Stream$. Chaque donnée (x_t, y_t) est ensuite affectée à son cluster $c_i^{(y)}$ et est simultanément ajoutée à son réservoir d'échantillonnage R_i. Chaque cluster $c_i^{(x)}$ est ensuite calculé comme le barycentre sur l'espace des attributs de son réservoir associé R_i. Pour la mémoire à court terme, la fenêtre FIFO est simplement remplie avec les N dernières données de $Initial\_Stream$.

Algorithme 1 : Apprentissage

Input : (x_t, y_t)
Result : $Modele\_updated$
$p \leftarrow getNearestCentroidY(y_t, {c_i^{(y)}}_{i\in[1,k]})$; `// l'index du centroide le plus proche - selon l'espace Y des labels`
if $T_p < S$ **then**
  $R_p.add((x_t, y_t))$;
  $(c_p^{(x)}, c_p^{(y)})_{new} = \frac{(c_p^{(x)}, c_p^{(y)})_{old} * (T_p - 1) + (x_t, y_t)}{T_p}$;
else
  $r = randomInteger(1, T_p)$;
  if $r < S$ **then**
    $(x_r, y_r) = (x_{p,r}, y_{p,r})$;
    $(x_{p,r}, y_{p,r}) = (x_t, y_t)$; `// l'ancienne donnée à l'index` r `dans` R_p `est remplacée par la nouvelle donnée.`
    $(c_p^{(x)}, c_p^{(y)})_{new} = \frac{(c_p^{(x)}, c_p^{(y)})_{old} * S + (x_t, y_t) - (x_r, y_r)}{S}$
  end
end
$F_t.removeOldest()$;
$F_t.add((x_t, y_t))$;

Après l'initialisation, OMk traite chaque exemple (x_t, y_t) arrivant selon les deux étapes suivantes :

Règles d'affectation des instances dans la phase d'apprentissage. Lorsqu'une nouvelle donnée (x_t, y_t) est considérée, elle est comparée aux k centroïdes des k réservoirs et affectée au réservoir R_p dont le centroïde est le plus proche selon le processus d'échantillonnage décrit ci-dessus. La comparaison de l'instance au centroïde se fait uniquement sur l'espace des labels et la métrique utilisée est la distance euclidienne usuelle. Puis R_p est mis à jour avec (x_t, y_t) selon une règle d'échantillonnage. Les centroïdes $(c_p^{(x)}), c_p^{(y)})$ sont re-calculés selon les nouvelles mémoires dans le réservoir et l'instance (x_t, y_t) est également mémorisée dans la fenêtre glissante.

Règles de prédiction. La donnée à prédire x_t est comparée aux k centroïdes des k réservoirs, sur l'espace des attributs. Après que le réservoir R_p le plus proche ait été identifié, les n plus proches voisins de la donnée x_t sont calculés sur l'espace des attributs -parmi l'union du réservoir R_p et de la fenêtre FIFO- et la prédiction renvoyée en résultat est définie par le vote majoritaire de leurs labels.

Algorithme 2 : Prediction

Input : x_t
Result : $prediction_t$
$p \leftarrow getNearestCentroid(x_t, c_i^{(x)}{}_{i\in[1,k]})$; `// l'index du centroide le plus proche -selon l'espace X`
$\{I_i\}_{i\in[1,n]} \leftarrow getNearestIns(x_t, R_p.Attributs \cup F_t.Attributs)$; `// Retourne les` n `index d'instance les plus proches du vecteur d'attributs` x_t
$votes[] = \emptyset$;
for $Index \in \{I_i\}_{i\in[1,n]}$ **do**
  $labels = (R_p \cup F_t).Labels[Index]$;
  $votes.add(labels)$
end
$prdiction_t \leftarrow vote.majority(votes)$

La complexité du processus d'OMk sur un exemple. Considérons M le nombre d'attributs, L le nombre de labels et les notations dans le tableau 2. La complexité du temps d'apprentissage d'un exemple est en $O(k * L)$ et la complexité du temps de prédiction est en $O((S + N) * M)$.

4 Protocole expérimental

Nous comparons OMk à trois algorithmes de la littérature (Roseberry et al., 2019), (Osojnik et al., 2017) et (Read et al., 2012) en nous plaçant dans un scénario d'apprentissage supervisé dans lequel le véritable ensemble de labels devient disponible immédiatement après la prédiction.

La procédure d'évaluation expérimentale est ici basée sur une stratégie *Interleaved Test-Then-Train* qui est adaptée au flux de données : chaque exemple est utilisé pour tester le modèle avant de l'utiliser pour l'entraînement. Plus précisément, à chaque instant t l'exemple (x_t, y_t) arrivant est traité selon trois étapes dans l'ordre suivant :

1. **Prédiction :** le modèle en vigueur $h_{t-1}(x)$ effectue la prédiction $p_t = [p_{t,j}]_{j\in[1,L]} \in \{1, 0\}^L$ pour x_t.
2. **Évaluation :** la performance du modèle est calculée en se basant sur une somme cumulée d'une fonction de perte entre la prédiction et le véritable ensemble de données $S_t = \frac{1}{t}\sum_{i=1}^{t} f(p_i, y_i)$
3. **Apprentissage :** le nouveau modèle $h_t(x)$ est construit à partir du modèle $h_{t-1}(x)$ et du nouvel exemple (x_t, y_t).

Afin de réduire l'impact des erreurs précoces et d'évaluer plus pertinemment la performance d'un modèle sur une période de temps pouvant être arbitrairement longue, nous avons utilisé un mécanisme de pondération temporelle (Gama et al., 2013).

Pour l'évaluation des performances, nous avons sélectionné trois mesures complémentaires de la littérature (Pereira et al., 2018) : $F1_{Exemple}$ basée sur les exemples, $F1_{Micro}$ basée sur les labels qui donne un poids uniforme à la performance sur chaque label, et $F1_{Macro}$ qui donne un poids égal à la performance sur chaque instance. Nous les complétons par les temps - mesurés en secondes - de calcul de la prédiction et de la mise à jour du modèle sur un jeu de données complet. Tous les algorithmes ont été implémentés sur la même plate-forme logicielle Massive Online Analysis (MOA[1]). Nos codes sources (algorithmes et expérimentations) sont disponibles au public[2] pour faciliter la reproductibilité des résultats.

Pour vérifier la capacité d'OMk à traiter efficacement les problèmes de dérive et d'oubli, nous l'avons comparé avec deux algorithmes de référence basés sur des arbres (ISOUPTree et HTps) et un algorithme basé sur les k plus proches voisins (MLSAMPKNN). Comme ISOUPTree et HTps ne peuvent gérer que les concepts stationnaires, nous les avons combinés avec la méthode adaptative ADWIN Bagging pour les adapter à la dérive de concept. L'algorithme MLSAMPKNN avec une fenêtre auto-ajustable qui mémorise des données passées vise, comme OMk, à s'adapter à la fois à un flux de données stationnaires et à un flux de données présentant des dérives de concepts.

Nos expérimentations portent sur 7 jeux de données issus d'applications réelles et variées (texte, image, vidéo) qui sont classiquement utilisées dans la littérature dans le contexte stationnaire et 4 jeux de données synthétiques. Nous avons utilisé deux générateurs disponibles sur la plateforme MOA (Read et al., 2011) pour générer des données artificielles présentant des propriétés spécifiques : le générateur d'arbres aléatoires (RTG) et la fonction de base radiale (RBF) qui permet de simuler des flux de données avec une dérive de concept. Ces deux générateurs nous ont permis de simuler deux cas différents de données avec (i) des changements de cardinalités des labels et (ii) des changements de dépendances de labels. Les détails des jeux de données expérimentales sont donnés dans le tableau 3.

5 Résultats expérimentaux

Cette section présente les résultats expérimentaux et compare les performances d'OMK avec celles obtenues par EaHTps, EaISOUPT et MLSAMPKNN pour des flux de données stationnaires et des flux de données avec une dérive de concepts. Tous les algorithmes ont été comparés sur la plateforme logicielle MOA version 2019.05, avec une JVM version 11.0.4, sur un PC Linux avec un processeur de Intel Core i7-670 3.40GHz, 8 coeurs et 15,5 Go Ram.

Les comparaisons des performances du module OMk et des trois algorithmes de référence sont présentées dans le tableau 4. Les paramètres choisis pour les expérimentations sont issus de l'analyse préalable présentée ci-dessus : la taille de la fenêtre FIFO, le nombre de centroïdes, la taille du réservoir et le nombre de plus proches voisins sont fixés respectivement à

1. MOA : https ://moa.cms.waikato.ac.nz/
2. Les codes sources OMk sont disponibles à l'adresse : https ://github.com/Cici-xihui/OMK

Jeu	Domaine	Nombre d'exemples	Nombre d'attributs	Nombre de labels	Cardinalité de labels
20NG	Text	19 300	1006	20	1.02
Bookmarks	Text	87 856	2150	208	2.03
Corel16k	Image	13 770	500	153	2.859
Enron	Text	1 702	1001	53	3.38
Mediamill	Video	43 907	120	101	4.38
Ohsumed	Text	13 930	1002	23	1.663
TMC2007	Text	28 596	500	22	2.22
RBF-drift1	Synthétique	100 000	10	10	$1.5 \rightarrow 3.5$
RTG-drift1	Synthétique	100 000	10	10	$1.8 \rightarrow 3.0$
RBF-drift2	Synthétique	100 000	10	10	2.8 (LD :$0.25 \rightarrow 1$)
RTG-drift2	Synthétique	100 000	10	10	1.6 (LD :$0.25 \rightarrow 1$)

TAB. 3 – *Jeux de données et leurs caractéristiques.*

10, 30, 100 et 3 par analyse préalable sur des données de tests internes. Le tableau 4 présente les valeurs des mesures $F1_{Exemple}$, $F1_{Micro}$ et $F1_{Macro}$ de chaque modèle sur les 11 jeux de données du tableau 3. Les meilleurs résultats sont indiqués en gras et les meilleurs seconds sont soulignés.

Ces résultats mettent en évidence trois grandes tendances. Notre nouvelle approche OMk obtient de très bonnes performances dans les deux types de flux de données, à dérive ou stationnaire, grâce à ses deux types de mémoires. Il produit le comportement le plus stable indépendamment de la nature des flux de données. L'algorithme EaHTps est performant pour les données avec une dérive de concept, alors que MLSAMPKNN se démarque davantage sur les données stationnaires. La stratégie avancée adaptative ADWIN Bagging permet à EaHTps de mieux s'adapter aux données de dérive, au prix d'un modèle complexe et conçu pour faire face à la dérive de concept. En contrepartie, les performances d'EaHTps s'effondrent sur plusieurs flux de données stationnaires non reconnus comme tels, et ses temps d'exécution sont en moyenne d'un ordre de grandeur plus lents que ceux de MLSAMPKNN et OMk.

Plus précisément, le tableau 4 montre qu'OMk domine largement pour la mesure $F1_{Macro}$ en étant en première position dans 6 cas sur 11 et en deuxième position dans les autres cas. Pour la mesure $F1_{Exemple}$, OMk est à égalité avec MLSAMPKNN, qui est l'un des meilleurs de l'état de l'art actuel. Pour la mesure $F1_{Micro}$, les meilleures performances sont obtenues par MLSAMPKNN et EaHTps pour des raisons qui, à ce stade, ne sont pas encore explicitées.

Ces résultats sont très prometteurs car nous nous sommes volontairement restreints dans la conception et dans le paramétrage (tailles des mémoires notamment) à un algorithme très simple. Le tableau 5 confirme que cette simplicité est associée à une rapidité d'exécution : OMk obtient dans 9 cas sur 11 les meilleurs temps d'exécution.

Jeu	F-Measure	EaHTps	EaISOUPT	MLSAMPKNN	OMk
20NG	$F1_{Exemple}$	0.411	0.318	0.597	**0.627**
	$F1_{Micro}$	0.409	0.315	0.59	**0.618**
	$F1_{Macro}$	0.185	0.136	0.214	**0.278**
Bookmarks	$F1_{Exemple}$	0.001	0.166	**0.374**	0.222
	$F1_{Micro}$	0.004	0.088	**0.277**	0.184
	$F1_{Macro}$	0	0.008	**0.163**	0.093
Corel16k	$F1_{Exemple}$	0	0.107	**0.199**	0.178
	$F1_{Micro}$	0	0.134	**0.217**	0.205
	$F1_{Macro}$	0	0.023	**0.092**	0.087
Enron	$F1_{Exemple}$	0.332	0.292	**0.401**	0.388
	$F1_{Micro}$	0.329	0.295	**0.416**	0.4
	$F1_{Macro}$	0.03	0.045	0.142	**0.154**
Mediamill	$F1_{Exemple}$	0.069	0.512	**0.547**	0.434
	$F1_{Micro}$	0.071	0.482	**0.539**	0.417
	$F1_{Macro}$	0.033	0.048	0.158	**0.173**
Ohsumed	$F1_{Exemple}$	0.35	0.298	0.094	**0.392**
	$F1_{Micro}$	0.322	0.275	0.097	**0.388**
	$F1_{Macro}$	0.147	0.143	0.051	**0.33**
TMC2007	$F1_{Exemple}$	0.535	**0.56**	0.368	0.335
	$F1_{Micro}$	0.53	**0.533**	0.333	0.364
	$F1_{Macro}$	0.032	0.03	0.038	**0.27**
RBF-drift1	$F1_{Exemple}$	**0.511**	0.248	0.332	0.365
	$F1_{Micro}$	**0.554**	0.256	0.341	0.372
	$F1_{Macro}$	**0.448**	0.111	0.231	0.296
RTG-drift1	$F1_{Exemple}$	**0.422**	0.327	0.383	0.376
	$F1_{Micro}$	**0.426**	0.348	0.389	0.386
	$F1_{Macro}$	**0.361**	0.216	0.275	0.305
RBF-drift2	$F1_{Exemple}$	**0.385**	0.088	0.256	0.297
	$F1_{Micro}$	**0.389**	0.089	0.258	0.303
	$F1_{Macro}$	0.266	0.067	0.219	**0.277**
RTG-drift2	$F1_{Exemple}$	**0.379**	0.175	0.266	0.312
	$F1_{Micro}$	**0.383**	0.174	0.268	0.316
	$F1_{Macro}$	**0.33**	0.126	0.237	0.287

TAB. 4 – *Comparaison des algorithmes avec les mesures $F1_{Exemple}$, $F1_{Micro}$ et $F1_{Macro}$ sur les 11 jeux de données.*

Jeu	EaHTps	EaISOUPT	MLSAMPKNN	OMk
20NG	519.85	40.64	24.08	**18.89**
Bookmarks	597.55	2788.38	419.99	**171.28**
Corel16k	137.98	67.81	14.42	**5.76**
Enron	20.26	8.74	3.93	**3.51**
Mediamill	206.37	112.26	18.11	**5.85**
Ohsumed	439.82	32.73	**4.88**	12.91
TMC2007	56.96	28.51	**6.46**	12.41
RBF-drift1	18.94	21.81	2.33	**2.06**
RTG-drift1	26.26	19,75	2.13	**2.09**
RBF-drift2	17,38	27.61	2.64	**1.97**
RTG-drift2	33.78	26.52	2.01	**1.57**

TAB. 5 – *Comparaison des temps de calculs des algorithmes sur les 11 jeux de données*

6 Conclusion et futurs travaux

Dans les travaux consacrés à la classification multi-labels en flux présentés dans cet article, nous avons proposé un nouveau modèle pour tenter d'améliorer conjointement deux problèmes bien connus : la dérive de concept et l'oubli catastrophique. Notre modèle combine deux mémoires spécifiques fonctionnant sur des temporalités différentes. Les premières expérimentations confirment la qualité des résultats obtenus eu égard aux algorithmes de la littérature pour des jeux de tailles restreintes. Différentes expérimentations complémentaires seront à faire à l'avenir pour argumenter les choix des centroïdes dans les réservoirs comme élément prototypique, du vote majoritaire pour la prédiction et de la sélection des paramètres.

Cependant, avec la qualité des résultats obtenus, la simplicité et l'efficacité en temps d'exécution d'OMk en font déjà un candidat très prometteur pour une intégration dans une structure arborescente profonde. L'objectif à terme serait de pouvoir traiter des problèmes de taille XMLC. Plus précisément, OMk pourrait devenir, en tant que k-séparateur, la brique de base d'un arbre de décision qui permettrait d'adapter les principes de l'algorithme CRAFTML (Siblini et al., 2018) basé sur des arbres pour les très grands flux de données. Pour illustrer cette idée, les instances pourraient être routées à chaque noeud d'un arbre dans l'un des k sous-noeuds fils. Lors de l'apprentissage, les informations portées par les instances dans un noeud seraient exploitées pour la création d'un k-séparateur qui permettrait ensuite de router récursivement les instances dans k noeuds fils. Enfin, pour la prédiction, une instance serait routée, sur la base de ses attributs explicatifs de noeud en noeud via leur k-séparateur jusqu'à une feuille finale qui contiendrait un modèle prédictif local. Cette approche permettrait de gérer une mémoire à long terme hiérarchisée de très grande capacité, en conservant une complexité de calcul super-linéaire, compatible avec le traitement online de très grand flux de données multi-labels.

Références

Bifet, A. et R. Gavaldà (2009). Adaptive learning from evolving data streams. In *International Symposium on Intelligent Data Analysis*, pp. 249–260. Springer.

Boutell, M. R., J. Luo, X. Shen, et C. M. Brown (2004). Learning multi-label scene classification. *Pattern recognition 37*(9), 1757–1771.

Büyükçakir, A., H. Bonab, et F. Can (2018). A novel online stacked ensemble for multi-label stream classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pp. 1063–1072.

Clare, A. et R. D. King (2001). Knowledge discovery in multi-label phenotype data. In *european conference on principles of data mining and knowledge discovery*, pp. 42–53. Springer.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences 3*(4), 128–135.

Gama, J., R. Sebastião, et P. P. Rodrigues (2013). On evaluating stream learning algorithms. *Machine learning 90*(3), 317–346.

Gama, J., I. Žliobaitė, A. Bifet, M. Pechenizkiy, et A. Bouchachia (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR) 46*(4), 1–37.

Júnior, J. C., E. Faria, J. Silva, J. Gama, et R. Cerri (2019a). Novelty detection for multi-label stream classification. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 144–149. IEEE.

Júnior, J. D. C., E. R. Faria, J. A. Silva, J. Gama, et R. Cerri (2019b). Pruned sets for multi-label stream classification without true labels. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE.

Kolter, J. Z. et M. A. Maloof (2007). Dynamic weighted majority : An ensemble method for drifting concepts. *Journal of Machine Learning Research 8*(Dec), 2755–2790.

Kong, X. et S. Y. Philip (2011a). An ensemble-based approach to fast classification of multi-label data streams. In *7th International Conference on Collaborative Computing : Networking, Applications and Worksharing (CollaborateCom)*, pp. 95–104. IEEE.

Kong, X. et S. Y. Philip (2011b). An ensemble-based approach to fast classification of multi-label data streams. In *7th International Conference on Collaborative Computing : Networking, Applications and Worksharing (CollaborateCom)*, pp. 95–104. IEEE.

Madjarov, G., D. Kocev, D. Gjorgjevikj, et S. Džeroski (2012). An extensive experimental comparison of methods for multi-label learning. *Pattern recognition 45*(9), 3084–3104.

Nguyen, T. T., T. T. T. Nguyen, A. V. Luong, Q. V. H. Nguyen, A. W.-C. Liew, et B. Stantic (2019). Multi-label classification via label correlation and first order feature dependance in a data stream. *Pattern recognition 90*, 35–51.

Osojnik, A., P. Panov, et S. Džeroski (2017). Multi-label classification via multi-target regression on data streams. *Machine Learning 106*(6), 745–770.

Partalas, I., A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-R. Amini, et P. Galinari (2015). Lshtc : A benchmark for large-scale text classification. *arXiv preprint arXiv :1503.08581*.

Pereira, R. B., A. Plastino, B. Zadrozny, et L. H. Merschmann (2018). Correlation analysis

of performance measures for multi-label classification. *Information Processing & Management 54*(3), 359–369.

Read, J., A. Bifet, G. Holmes, et B. Pfahringer (2011). Streaming multi-label classification. In *Proceedings of the Second Workshop on Applications of Pattern Analysis*, pp. 19–25.

Read, J., A. Bifet, G. Holmes, et B. Pfahringer (2012). Scalable and efficient multi-label classification for evolving data streams. *Machine Learning 88*(1-2), 243–272.

Roseberry, M., B. Krawczyk, et A. Cano (2019). Multi-label punitive knn with self-adjusting memory for drifting data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD) 13*(6), 1–31.

Siblini, W., P. Kuntz, et F. Meyer (2018). Craftml, an efficient clustering-based random forest for extreme multi-label learning. In *ICML*, pp. 4671–4680.

Sousa, R. et J. Gama (2018). Multi-label classification from high-speed data streams with adaptive model rules and random rules. *Progress in Artificial Intelligence 7*(3), 177–187.

Tsoumakas, G., I. Katakis, et I. Vlahavas (2009). Mining multi-label data. In *Data mining and knowledge discovery handbook*, pp. 667–685. Springer.

Venkatesan, R., M. J. Er, M. Dave, M. Pratama, et S. Wu (2017). A novel online multi-label classifier for high-speed streaming data applications. *Evolving Systems 8*(4), 303–315.

Vitter, J. S. (1985). Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS) 11*(1), 37–57.

Wang, L., H. Shen, et H. Tian (2017). Weighted ensemble classification of multi-label data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 551–562. Springer.

Woloszynski, T. et M. Kurzynski (2011). A probabilistic model of classifier competence for dynamic ensemble selection. *Pattern Recognition 44*(10-11), 2656–2668.

Zhang, M.-L. et Z.-H. Zhou (2007). Multi-label learning by instance differentiation. In *AAAI*, Volume 7, pp. 669–674.

Zhu, Y., K. M. Ting, et Z.-H. Zhou (2018). Multi-label learning with emerging new labels. *IEEE Transactions on Knowledge and Data Engineering 30*(10), 1901–1914.

Summary

While Multi-label classification has been widely studied, few works considered the Multi-label streaming problem. To address this issue, we propose a new algorithm: Online Memory k-Means (OMk). OMk is based on a K-nearest neighbors' strategy which integrates two types of memories with limited size: a short-term and a long-term memory. The short-term memory is a FIFO window containing only the recent concept while the long-term memory, storing previous concepts, relies on a tree-structure using a reservoir sampling strategy. The designed memories allow us to manage streams with concept drifts and to resist to catastrophic forgetting. The experimental study compares the proposal to the current state-of-the-art algorithms using seven real-world and four artificial multi-label data streams on three multi-label metrics and evaluation time. OMk shows competitiveness and low time complexity, in both stationary and concept drift scenarios.