Classification Multi-Labels en flux : comparaisons d'approches et nouvelles propositions

Xihui Wang

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Nantes Université

THÈSE DE DOCTORAT DE

NANTES UNIVERSITE

ECOLE DOCTORALE N° 641
Mathématiques et Sciences et Technologies du numérique, de l'Information et de la Communication
Spécialité : *Informatique*

Par
« Xihui WANG »

« Classification Multi-Labels en flux »

« Comparaisons d'approches et nouvelles propositions »

Thèse présentée et soutenue à « l'amphithéâtre du LS2N », le « 28-02-2023 »
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes (LS2N – équipe Duke)

Rapporteurs avant soutenance :

Julien Velcin	Professeur des universités, Université de Lyon 2
Jean-Charles Lamirel	Maître de conférences HDR, Université de Strasbourg

Composition du Jury :

Président :	Mustapha Lebbah	Professeur des universités, Université Paris Saclay – Versailles
Examinateurs :	Armelle Brun	Professeure des universités, Université de Lorraine–Nancy
	Mustapha Lebbah	Professeur des universités, Université Paris Saclay – Versailles
Dir. de thèse :	Pascale Kuntz	Professeure des universités, Université de Nantes

Invité(s)

Frank Meyer	Docteur Ingénieur R&D, Orange Labs Lannion

Remerciements

Ce manuscrit conclut trois ans de travail et je tiens à exprimer ma reconnaissance envers tous ceux qui de près ou de loin y ont contribué. Sans leur présence, je n'aurais pas réussi à mener à bien ce projet toute seule dans le contexte si particulier que nous avons connu. J'ai passé presque un an de ma thèse dans une situation de confinement ou de restriction de circulation !

Tout d'abord, je remercie tous les membres du jury pour l'intérêt qu'ils portent à travers leur présence à mes travaux. Je remercie Julien VELCIN et Jean-Charles LAMIREL d'avoir accepté d'être rapporteurs de mes travaux. Je remercie également Armelle BRUN et Mustapha LEBBAH d'avoir accepté de participer au jury.

Je tiens spécialement à remercier ma directrice de thèse Pascale KUNTZ et mon encadrant industriel Frank MEYER. Pascale KUNTZ m'a accompagnée sans relâche dans la rédaction de mes articles. En particulier, sa rigueur scientifique m'a guidée dans le temple de la recherche et chacune de ces questions m'a permis de pousser plus loin mes réflexions. Je suis également reconnaissante qu'elle m'ait fait entièrement confiance et qu'elle m'ait toujours encouragée et soutenue, ce qui m'a donné beaucoup de confiance et de courage pour aller jusqu'au bout de cette thèse.

Frank MEYER m'a accompagnée sans relâche dans la conception de mes algorithmes et à la mise en œuvre de mes expérimentations. Il m'a été d'une aide précieuse et ses conseils toujours pertinents. Je lui suis vraiment reconnaissante pour sa compagnie au cours de ces trois années tant sur le plan professionnel que personnel. Je tiens donc à le remercier sincèrement non seulement pour sa bienveillance et ses précieux conseils, mais également pour m'avoir donné la liberté de choisir ma direction de recherche.

Je voudrais aussi remercier mon équipe au sein d'Orange, en particulier Vincent LEMAIRE, Marc BOULLÉ et Fabrice CLÉROT. Je suis reconnaissante à Vincent LEMAIRE de m'avoir aidée scientifiquement et son érudition m'a aidée à comprendre les problèmes auxquels je faisais face. Je suis également reconnaissante de sa gentillesse et de ses encouragements. J'ai apprécié énormément chacun de nos échanges scientifiques.

Je tiens à remercier Marc BOULLÉ pour sa disponibilité et ses nombreux échanges qui m'ont également beaucoup aidée à saisir mes problèmes. Sa passion pour la recherche et sa rigueur scientifique exceptionnelle m'ont été d'une grande inspiration. Je remercie Fabrice CLÉROT pour m'avoir guidée dans mes recherches bibliographiques et dans le positionnement de mes travaux. Nos nombreuses discussions et ses remarques toujours pertinentes m'ont aidée à mieux positionner mes problèmes.

Je tiens également à remercier tous les autres collègues de mon équipe qui m'ont accompagnée. Je suis vraiment très heureuse d'avoir pu travailler dans cette équipe pendant ces trois années. La passion de la recherche m'a profondément touchée et j'ai attrapé le virus, un sympathique celui-là ! Dans le même temps, l'humour et la joie de tous m'ont rendue très heureuse lors de ces trois premières années de ma vie professionnelle. Je chérirai toujours le temps passé ensemble.

Enfin, je souhaiterais remercier affectueusement mon compagnon Thibault CORDIER, mes parents et les siens. Je leur suis profondément reconnaissante du fond de mon cœur pour leur dévouement et le soutien immense, sans lesquels je n'aurais pas pu accomplir cette tâche difficile. C'est aussi la passion pour la recherche de Thibault qui m'a inspirée et qui m'a également fait aimer la recherche. Ces trois années seront des souvenirs des plus mémorables. Je n'ai pas pu revoir mes parents, mais leur soutien à distance m'a donné également beaucoup de confiance et de force pour atteindre mon objectif, celui de finaliser cette thèse.

Table des matières

Table des figures

Liste des tableaux

Notations mathématiques

Notation	Définition mathématique
$\mathcal{X}$	L'espace de d attributs, $\mathbb{R}^d$
$\mathcal{L}$	L'ensemble de l labels, $\{\lambda_1, ..., \lambda_l\}$
$\mathcal{Y}$	L'espace de l labels, $\{0, 1\}^l$
$\mathbf{x}$	Le vecteur d'attributs, $\mathbf{x} = [x^1, ..., x^d] \in \mathcal{X}$
$\mathbf{y}$	Le vecteur de labels, $\mathbf{y} = [y^1, ..., y^l] \in \mathcal{Y}$
$\mathbf{p}$	Le vecteur de prédictions, $\mathbf{p} = [p^1, ..., p^l] \in \mathcal{Y}$
$\mathcal{T}$	L'ensemble d'apprentissage, $\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i) \| 0 \leq i \leq \|\mathcal{T}\|\}$
$\mathcal{S}$	L'ensemble de test, $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i) \| 0 \leq i \leq \|\mathcal{S}\|\}$
y	Le variable cible associée aux labels, $y \in [1, l]$
p	La prédiction associée aux labels, $p \in [1, l]$
$\mathcal{D}_s$	Le flux de données mono-label, $\mathcal{D}_s = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_t, y_t), ...\}$
$\mathcal{D}_m$	Le flux de données multi-labels, $\mathcal{D}_m = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_t, \mathbf{y}_t), ...\}$

Tableau 1 – Résumé des notations mathématiques majeures.

Chapitre 1

Introduction

Sommaire

L'évolution considérable des capacités de collecte et de stockage des données associée à l'explosion des capacités de calcul disponibles a profondément transformé le champ de l'analyse des données et de l'apprentissage automatique, regroupés aujourd'hui sous les champs de la science des données et de l'intelligence artificielle. Si les données d'observation ont joué de tout temps un rôle central dans le développement des sciences, leurs nouvelles échelles d'acquisition renouvellent en profondeur les méthodologies et la temporalité des découvertes. A titre illustratif, les observatoires astronomiques, les satellites de détection de la Terre et les réseaux d'observation du climat génèrent chaque jour des téraoctets de données. Le projet Australian Square Kilometre Array Pathfinder (ASKAP) a obtenu 7,5 téraoctets/seconde de d'images, qui devraient passer à 750 téraoctets/seconde d'ici 2025 [3]. En biologie, les intégrations des différentes échelles des données omics représentent un nouveau défi pour une compréhension de plus en plus précise de la complexité du vivant [4]. L'attractivité des capacités à traiter des données stimule en rétroaction le rythme de production des données : 90% des données mondiales ont été générées au cours des deux dernières années [5].

Cette évolution de la recherche s'est accompagnée de l'émergence puis de la transformation de tout un pan de l'industrie. A titre illustratif, Google reçoit chaque jour 3,5 milliards de requêtes de recherche ; YouTube met en ligne 300 heures de vidéo par minute et devrait atteindre 1,700 heures par minute d'ici 2025 ; près de 2 milliards d'utilisateurs actifs de Facebook partagent 4,5 milliards de contenus ; Amazon vend environ 13 millions

d'articles dans le monde. Ces entreprises, qui occupent aujourd'hui une position économique majeure, ont démontré que les données pouvaient être la matière première centrale d'un processus de création de valeur.

Actives à la fois dans la recherche académique et industrielle, la science des données et l'intelligence artificielle sont ainsi devenues des champs très attractifs qui articulent plusieurs spécialités, notamment l'analyse exploratoire de données, la visualisation de données et l'apprentissage. En particulier, l'apprentissage automatique a connu des progrès considérables cette dernière décennie associés notamment au développement de l'apprentissage profond, qui a su revisiter les modèles des réseaux de neurones des années 90 en tirant partie des nouvelles capacités de calcul et de mémorisation des calculateurs actuels. Les applications sont aujourd'hui très variées : prédire des maladies [6], traduire automatiquement des textes [7], évaluer des risques de fraudes [8], etc. Et la progression de la qualité des résultats obtenus continue de stimuler des développements et de nouvelles applications dans de très nombreux domaines. Cependant, l'absence d'intelligibilité des calculs effectués pour les décisions proposées laisse la voie ouverte pour d'autres approches motivées par des enjeux sociétaux. Citons par exemple l'explicabilité des résultats qui est une nécessité éthique dans la prise de certaines décisions, en médecine notamment.

En sus de l'évolution des volumes de données à traiter et des approches de traitement, la nature même des données a évolué. Les données d'entrée des processus d'apprentissage ont longtemps été essentiellement des données « statiques » qui étaient disponibles à l'entrée de l'algorithme. Aujourd'hui, un nombre croissant d'applications génèrent des données massives et en continu sur des durées non bornées a priori. Or dans certains cas avec l'explosion de la volumétrie des données, les ressources de calcul et de stockage peuvent être insuffisantes pour une mémorisation permanente de tous les exemples. S'ajoutent à ces contraintes associées à des évolutions technologiques dans l'accès aux données, de nouvelles préoccupations : une volonté d'accessibilité de l'IA au plus grand nombre en offrant des algorithmes qui fonctionnent sur des moyens de calcul « standards » [9] et une maîtrise des coûts énergétiques qui s'appuie sur une limitation des capacités de calcul et de stockage [10]. Dans ces différents contextes, les données ne peuvent plus être considérées comme disponibles en permanence, et le problème de classification se transforme en un problème de classification de données en flux.

Ce changement de paradigme confronte les modèles d'apprentissage à deux changements majeurs. L'hypothèse i.i.d (identiquement et indépendamment distribuée) considérée dans l'apprentissage traditionnel n'est plus toujours valide dans le contexte des flux de données. Un exemple typique est celui des préférences d'achat des clients qui peuvent changer avec le temps, en fonction du jour de la semaine, de la disponibilité d'alternatives, du taux d'inflation, etc. Le changement de la distribution sur le flux de données est un problème bien connu sous le terme de dérive de concept (« concept drift ») [1]. Un autre défi pour les algorithmes « on-line » est de gérer efficacement le dilemme entre

stabilité et plasticité sous la contrainte d'une mémoire limitée. Une grande stabilité est nécessaire pour préserver l'information déjà apprise, et une grande plasticité est cruciale pour l'apprentissage rapide de nouvelles informations [11].

Cette thèse est centrée sur un problème d'apprentissage concernant ces données en flux : la classification supervisée multi-labels. Il s'agit de pouvoir classer « en continu » des instances décrites par un grand nombre d'attributs et auxquelles peuvent être associées dans le processus de classification un ensemble de labels.

1 Motivations applicatives

La classification multi-labels a connu un fort développement cette dernière décennie stimulé par l'essor des besoins applicatifs qui nécessitent de prendre en compte des relations de dépendance entre des labels, et qui ne peuvent donc plus s'appuyer sur les algorithmes éprouvés de la classification mono-label sous l'hypothèse d'indépendance.

Par exemple, dans le domaine de la santé, de nombreuses maladies peuvent coexister et être liées entre elles - le diabète peut être accompagné des maladies cardio ou cérébro-vasculaires, de la neuropathie et de la rétinopathie [12] [13]. En exploitant la corrélation entre les comorbidités, des travaux montrent que la classification multi-labels améliore grandement la précision de la prédiction de quatre comorbidités diabétiques [14]. Dans le domaine de la psychiatrie, pour l'exploration d'une approche diagnostic qui classifie des troubles mentaux, un modèle multi-label est considéré comme très prometteur eu égard à des approches antérieures [15]. La classification multi-labels est également largement utilisée dans l'annotation automatique de textes. Par exemple, on peut concevoir un système qui permet d'associer les actualités à différents sujets, e.g, un article d'actualité concernant une conférence sur le changement climatique peut être étiquetée à la fois par les labels « politique » et « environnement » [16]. Dans une bibliothèque musicale également, un modèle de la classification multi-labels peut annoter automatiquement une musique comme étant à la fois « heureuse » et « classique ». Au cours des dernières décennies, de nombreuses applications (analyse de sons/musique [17] [18], vision par ordinateur [19][20], analyse de textes [21][22], biologie et santé [23][24], systèmes de recommandation [25][26], etc) ont donc stimulé le développement de la classification multi-labels.

Dans l'entreprise Orange dans laquelle a été effectuée cette thèse, un domaine d'application priviligié a été initialement l'analyse d'opinions. Pour faire face à la volumétrie des données à traiter et à la combinatoire potentiellement grande des conjonctions de labels, une thèse précédente [27] a été consacrée au passage à l'échelle des algorithmes de classification multi-labels. Les travaux de recherche cette thèse ont contribué au développement de l'outil logiciel VIPE [28] qui permet de classer des textes courts (tweets, mails, enquêtes, ...) à partir d'un ensemble de labels d'intérêts définis par un expert (« client

globalement satisfait », « évoque la rapidité du débit »,...). VIPE est une application Web qui permet d'importer des textes courts en partageant les textes sources entre les utilisateurs et en conservant les labels dans le domaine privé de chacun. VIPE apprend un modèle unique sur les différentes sources et labels, mais chaque utilisateur peut annoter ou consulter les résultats pour ses propres labels. Les résultats peuvent être de trois types : (i) les textes correspondants à des labels donnés - sélectionnés par un ranking - ; (ii) les labels prédits pour un texte donné ; et (iii) la matrice des scores des labels prédite pour l'ensemble des textes. L'outil a été utilisé dans le service marketing d'Orange avec des retours d'usage positifs. Mais il se heurte à deux limites : il ne s'applique pas à des flux de données et il ne prend pas en compte explicitement les changements d'intérêts des utilisateurs.

D'autres domaines d'intérêt pour la classification multi-labels ont été identifiés depuis chez Orange :

— la gestion des messages clients : dans le domaine de la relation client (CRM, Customer Relationship Management), Orange reçoit chaque jour des milliers de messages en provenance de ses clients. Les messages sont pré-analysés pour être routés vers les bons destinaires et peuvent être pré-labellisés pour accélérer leur lecture, leur compréhension et leur traitement. L'utilisation de modèles mono-labels dans ce cadre est sous-optimale puisque les labels sont très souvent corrélés, et un modèle multi-labels est donc mieux adapté.

— les systèmes de veille collaborative : dans le système Better Together déployé dans l'intranet d'Orange, de nombreux articles issus de sites internet sont collectés puis proposés à différents types d'utilisateurs (financiers, juristes, ingénieurs réseaux, ingénieurs logiciels, Responsables RH,...). Afin d'orienter au mieux les articles selon leur centre d'intérêt, les utilisateurs peuvent filtrer les articles selon des tags, et peuvent en parallèle tagger les articles de manière collaborative, en leur ajoutant des labels qui font sens pour eux et pour le reste de leur communauté. Un système de recommandation de tags multi-labels (au sens du filtrage collaboratif [29]) permettrait de faciliter fortement la pertinence du ciblage des articles et l'usage du système.

— la qualification de certaines menaces dans le domaine de la cybersécurité (voir la figure 1.1) : certaines menaces issues de logiciels, ou d'appels à des URL dangereuses, doivent être qualifiées selon différents aspects.

Dans ces différents domaines, les données arrivent en flux et la réactivité des modèles prédicatifs est souhaitable voire même indispensable pour les applications de cybersécurité. Elle est en effet importante dans le cas de la gestion de messages clients ou d'un système de veille car un événement soudain (une panne réseau, des annonces importantes sur un domaine de l'internet) doit pouvoir être pris en compte rapidement pour associer les bons labels aux messages et aux articles. Elle peut être cruciale dans le cas de la

cybersécurité, où une nouvelle alerte associant un nouvel objet (logiciel, ou URL,) à des nouvelles menaces doit être prise en compte, apprise et généralisée dans les plus brefs délais.

(a) Exemple 1 (b) Exemple 2

FIGURE 1.1 – Deux exemples de menaces pour les URLs dans le domaine de la cybersécurité.

Ces exemples illustrent l'intérêt croissant des organisations comme Orange pour l'apprentissage des données muti-labels en flux. Il vise à augmenter la capacité de systèmes automatiques à fournir une description prédictive plus riche, composée de plusieurs propriétés (les labels), et il doit permettre une meilleure réactivité à de nouveaux événements que les solutions fournies par des « processus batch » mis à jour cycliquement.

2 Des données multi-labels complètes aux données multi-labels en flux

Pour préciser le positionnement académique de la thèse dans le domaine de la classification multi-labels, nous rappelons synthétiquement ci-dessous les évolutions des modèles d'apprentissage rendus nécessaires par l'évolution des caractéristiques des données et des usages.

2.1 De la classification mono-label à la classification multi-labels

Le paradigme historiquement le plus étudié est l'apprentissage supervisé mono-label [30]. Son objectif est d'associer un objet décrit par un vecteur d'attributs à un unique concept d'intérêt que l'on qualifie ici de label. Par exemple, on peut souhaiter concevoir un système qui permet de déterminer si un mail décrit par les mots qu'il contient (attributs) est un spam ou non [31]. Dans ce cas, chaque objet est décrit par deux catégories exclusives étiquetées, dont une seule est correcte, on parle donc de classification binaire.

Pour couvrir une famille plus large de problèmes, le paradigme de la classification multi-classes a été développé : il permet de considérer une variable de sortie avec plus de deux modalités. Dans ce cas, chaque objet est associé à une classe parmi un ensemble de classes possibles. Par exemple, il permet de déterminer, pour des images de textes manuscrits décrites par des pixels (attributs), les chiffres qui y sont écrits et la variable de sortie prend donc une unique valeur parmi "1","2",...,"10" [32]. Cependant, pour certaines applications, les objets doivent intrinsèquement être décrits par plusieurs labels parmi un ensemble de possibilités prédéfinies. Par exemple, en annotation de textes, on peut souhaiter construire un système qui qualifie un texte à la fois de drôle, traitant de sport et en anglais [33] ; on peut également construire un système pour annoter les images qui identifier simultanément tous les objets différents dans chaque image [34]. Nous illustrons différents types de classifications dans la Figure 1.2.

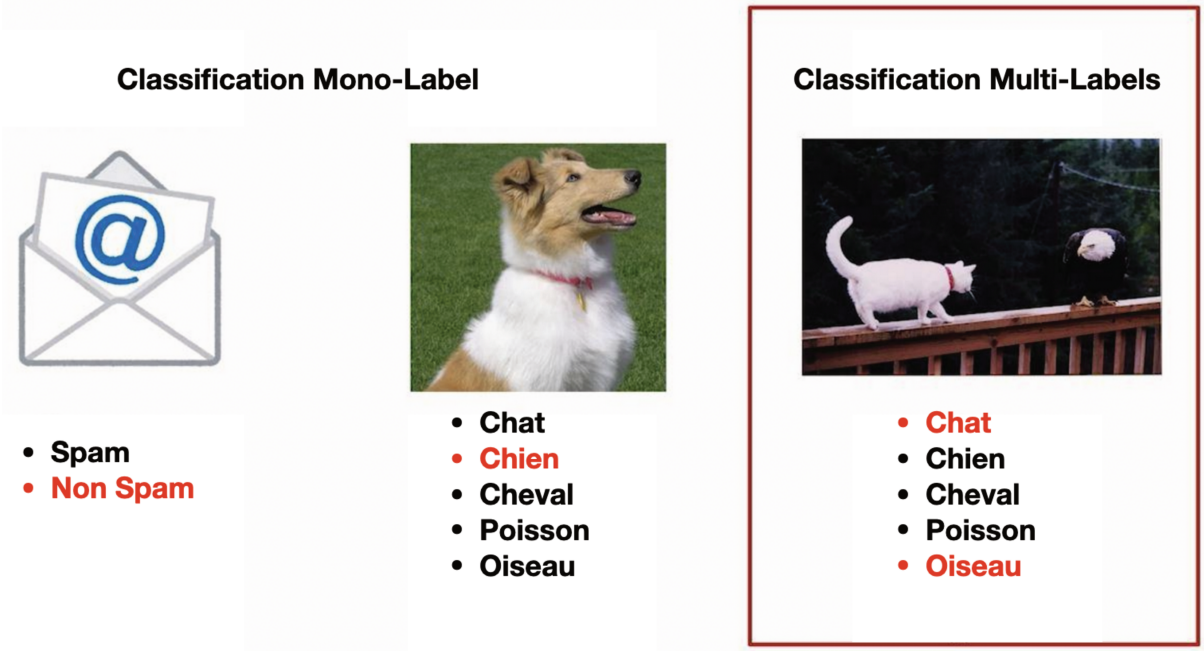
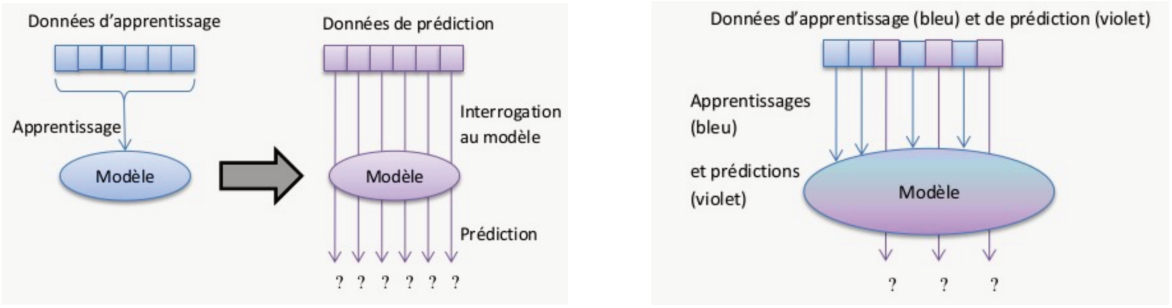


FIGURE 1.2 – Présentation des différents types de classification.

2.2 L'apprentissage « off-line » et « on-line »

L'apprentissage traditionnel (« off-line ») se base en entrée sur une disponibilité de l'ensemble des données pour leur traitement. Dès que la distribution des données change, un nouveau modèle doit être ré-entraîné sans mémorisation des connaissances acquises précédemment. Cette démarche ne répond pas aux contraintes actuelles posées par les très gros volumes de données qui doivent être traités en un temps limité. Elle ne permet pas non plus de gérer les évolutions des tendances dans le temps. Pour répondre aux défis

posés par les flux, l'apprentissage incrémental aussi qualifié de « on-line » a suscité un intérêt croissant ces dernières années. Il vise à apprendre et à prédire les données une à une sans interruption, à intégrer continuellement des informations dans ses modèles, et à rendre optionnel le stockage intégral des données. Les principes de l'apprentissage « off-line » et « on-line » sont illustrés dans la Figure 1.3.



(a) L'apprentissage off-line (b) L'apprentissage incrémental ou on-line

FIGURE 1.3 – Principes de l'apprentissage off-line et on-line

Si dans cette petite présentation nous avons employé les deux termes « on-line » et « incrémental » sans distinction, notons que dans la littérature les définitions associées à ces concepts sont souvent bien floues [35]. Certains auteurs les utilisent indifféremment, tandis que d'autres introduisent différentes distinctions. Dans la suite du manuscrit, nous avons adopté les définitions proposées par Losing et al [2]. Les algorithmes d'apprentissage incrémentaux sont des algorithmes qui génèrent une chaîne de modèles sur un flux de données donné, où chaque modèle est construit sur le modèle précédent et l'exemple le plus récent. Les algorithmes d'apprentissage « on-line» sont basés sur des algorithmes d'apprentissage incrémentaux et limités par la complexité d'espace du modèle et la complexité du temps d'exécution, c'est-à-dire la capacité d'apprendre sur le long temps en continu sur des dispositifs aux ressources limitées. Dans cette thèse, nous considérons l'existence de contraintes de ressources de stockage et de calcul limitées pour les applications visées, et nous utilisons donc le terme d'apprentissage « on-line ».

3 Contenu du manuscrit

Les principales contributions des travaux de recherche présentés dans ce manuscrit sont de trois ordres : bibliographique, méthodologique et algorithmique.

Comme nous l'avons évoqué ci-dessus l'intérêt récent pour la classification multi-labels en flux peut se positionner comme une extension du problème plus ancien de la classification multi-labels. Les rares états de l'art pour les données en flux [36] se structurent donc autour de la typologie désormais classique en classification multi-labels de Tsoumakas et Katakis [30]. Elle distingue trois familles d'approches : (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 mono-label au traitement des données multi-labels, et (iii) les approches ensembles qui utilisent un ensemble de classifieurs issus de la première ou de la deuxième famille d'approches. Si cette typologie reste valide pour les approches proposées pour les données en flux, elle masque cependant une information majeure spécifique à ces données : les différents types de dérives de concepts qui peuvent apparaître dans des flux non stationnaires. Dans notre état de l'art, nous avons donc croisé la typologie classique avec les travaux portant plus généralement sur la dérive de concepts dans les données temporelles [1]. En nous appuyons sur la distinction entre les approches passives et actives, nous avons ainsi distingué plus précisément les différentes stratégies déployées en classification multi-label en flux pour faire face aux dérives. Nous avons complété cet état de l'art par une analyse des protocoles expérimentaux mis en œuvre dans la littérature pour la validation des approches.

L'analyse des protocoles nous a confirmé l'absence de cadre consensuel dans la communauté. La grande majorité des algorithmes sont testés sur des jeux de données stationnaires et les quelques jeux non stationnaires utilisés n'explicitent pas les spécificités des dérives qui peuvent apparaître et limitent ainsi la compréhension approfondie des résultats obtenus. Pour tenter de pallier cette difficulté, nous avons donc proposé un nouveau protocole expérimental pour les données non stationnaires. Ce protocole permet d'introduire des dérives de concepts de différentes natures dans le flux et de suivre ainsi leurs impacts respectifs sur le modèle. Nos comparaisons expérimentales s'appuient sur ce protocole.

Ces apports bibliographiques et méthodologiques n'étaient guère prévisibles au démarrage de la thèse, l'objectif initial de notre travail étant de développer un nouvel algorithme de classification multi-labels en flux qui satisfasse un « bon » compromis entre la qualité des performances prédictives du modèle, les ressources de calcul nécessaires en mémoire et la vitesse d'exécution. Pour des raisons initiales d'accessibilité, qui croisent aujourd'hui avec des contraintes écologiques, nous avions comme contrainte de développer un modèle d'apprentissage qui puisse se déployer sur un ordinateur standard. Pour faire face à la contrainte de la complexité, nous avons développé une première approche dont les principes sont fortement inspirés de l'algorithme CraftML (Clustering-based RAndom Forest of predictive Trees for extreme Multi-label Learning) [37] qui fait partie de l'état de l'art de la classification multi-labels en grandes dimensions (« extreme multi-label learning »). Il s'agit de segmenter l'espace de recherche en sous-espaces de tailles réduites qui permettent de limiter la complexité des calculs tout en exploitant les corrélations entre les labels pour améliorer la qualité de l'apprentissage. Les résultats expérimentaux étant encourageants sur des flux stationnaires, nous avons poursuivi cette direction en proposant *in fine* un nouvel algorithme A2ML (Adaptive Memories for Multi-Label stream classification) qui permet l'adaptabilité du modèle aux variations des flux non stationnaires en

s'appuyant sur une stratégie prometteuse que nous avions identifiée dans l'état de l'art : la combinaison de mémoires avec des temporalités différentes. Un résultat théorique nous permet de garantir la complexité réduite du temps de calcul. Et les comparaisons expérimentales avec des algorithmes concurrents à la fois sur des flux stationnaires et sur des flux non stationnaires confirment la qualité de notre approche.

Le reste du manuscrit est structuré en quatre chapitres. **Le chapitre 2** dresse un état de l'art des algorithmes pour la classification multi-labels traditionnel et pour la classification mono-label en flux. Une mise en perspective de ces différents problèmes de classification permet de comprendre plus finement les défis de la classification multi-labels en flux. Pour la classification multi-labels traditionnelle, les états de l'art sont présentés selon les trois grandes familles citées ci-dessus. Pour la classification mono-label en flux, l'état de l'art est structuré selon la nature du flux de données, stationnaire et non-stationnaire. Dans le cas stationnaire, nous présentons les algorithmes les plus connus qui adaptent les algorithmes de la classification mono-label traditionnelle au problème d'apprentissage on-line. Dans le cas non stationnaire, nous détaillons d'abord les modalités de changements possibles de la distribution de données en flux, puis nous présentons les approches qui peuvent s'adapter au changement selon deux catégories : les approches actives et les approches passives. La fin du chapitre porte sur l'évaluation des modèles qui est une question essentielle dans la mise en œuvre des comparaisons expérimentales.

Le chapitre 3 complète le panorama présenté dans le chapitre 2 en approfondissant l'état de l'art sur la classification multi-labels en flux. La présentation des différents problèmes de classification nous a permis d'identifier les spécificités liées au passage des données complètes aux données en flux. Il s'agit de combiner à la fois un temps limité dans l'adaptation du modèle d'apprentissage aux nouvelles informations et des changements de distributions, sans oublier les caractéristiques intrinsèques aux données multi-labels portant sur l'existence de corrélation entre les labels et l'espace de sortie en grande dimension. La nature des flux ayant un impact majeur dans les performances des algorithmes, nous avons structuré l'état de l'art de ce chapitre autour de la caractéristique principale des flux : classification pour des flux stationnaires et classification pour des flux non stationnaires. L'analyse détaillée des algorithmes nous permet de préciser leurs avantages et leurs inconvénients respectifs et nous guide sur la direction de nos recherches à mener. Nous complétons le chapitre 3 par une présentation des flux de données non stationnaires utilisés dans les expérimentations de la littérature en discutant leurs limites.

Les chapitres 4 et 5 décrivent ensuite nos contributions méthodologiques et algorithmiques. **Le chapitre 4** a pour objectif d'étudier le problème de la classification multi-labels sur les flux stationnaires, en considérant notamment la limite de la complexité

temporelle qui est un enjeu important pour l'apprentissage « on-line ». En s'inspirant de l'algorithme CraftML [37], nous proposons une nouvelle approche pour la classification multi-labels en flux (MLT-ML). Cette approche est basée en particulier sur une adaptation d'une stratégie appelée Learning Vector Quantization (LVQ) [38] qui permet de regrouper les combinaisons de labels selon leur similarité, et qui combinée à une règle d'échantillonnage permet l'acquisition et la conservation continues de résumés de connaissances à partir du flux de données dans une mémoire limitée. Les expérimentations menées sur 15 flux de données multi-labels confirment que notre approche présente un très bon compromis entre la qualité des performances prédictives, les ressources de calcul nécessaires et la vitesse d'exécution. Ces expérimentations montrent qu'elle est compétitive face aux approches les plus performantes de l'état de l'art actuel tout en présentant une complexité temporelle plus faible.

Le chapitre 5 est centré sur les flux non stationnaires. En s'appuyant sur la stratégie de partitionnement de l'espace de recherche éprouvée dans le chapitre précédent, nous avons cherché à intégrer dans une nouvelle approche la condition d'adaptabilité de l'apprentissage aux variations de distributions des données au cours du temps. L'état de l'art a montré que les algorithmes basés sur une combinaison de mémoires sont particulièrement prometteurs pour faire face à des changements de différentes natures dans le flux. Nous avons ainsi développé successivement deux nouvelles approches basées sur une combinaison de mémoires – à court terme et à long terme-. La première ODM (Online Dual Memory) [39] a permis de valider la stratégie et de confirmer la capacité d'apprentissage sous maîtrise des coûts de calcul. La seconde A2ML (Adaptive Memories for Multi-Label stream classification) [40] intègre des mécanismes d'adaptation aux dérives significativement améliorés. Les comparaisons expérimentales de A2ML avec les quatre meilleurs algorithmes identifiés dans les comparaisons du chapitre précédent sur deux flux de plus de 80, 000 données montrent que notre nouvelle approche est non seulement la plus performante quelle que soit la nature de la dérive, mais qu'elle maintient une complexité de calcul faible.

Le chapitre 6 clôt ce manuscrit en présentant d'abord une synthèse des résultats obtenus, puis en ouvrant sur des perspectives à la fois algorithmiques et applicatives. Sur le plan théorique et expérimental, les travaux à venir devront intégrer deux aspects peu traités dans cette thèse et qui posent encore des questions largement ouvertes dans la communauté scientifique : le passage aux très grandes échelles (de l'ordre de 1 million d'attributs et de labels comme en « extreme multi-labels learning ») de la classification multi-labels en flux, et la prise en compte de l'évolution (création/suppression) d'attributs et de labels. Sur le plan applicatif, la classification multi-label en flux « on-line » attire un intérêt croissant dans l'entreprise Orange avec notamment un projet en cours de

discussion dans le domaine de la cybersécurité.

Chapitre 2

Classification multi-labels : panorama des approches classiques

Sommaire

La classification multi-labels se distingue des cas binaires et multi-classes, mais il est possible de comparer ces différentes classifications en constatant que toutes sont des cas particuliers de la classification multi-classes : la classification binaire est un cas particulier de la classification multi-classes à 2 classes et la classification multi-labels est un cas particulier de la classification multi-classes à 2^l classes possibles. La complexité du problème augmente donc de façon exponentielle lorsque le nombre de labels augmente, ce qui complexifie grandement l'apprentissage.

Ces liens entre les différents problèmes de classification nous paraissent utiles pour tenter de comprendre ensuite en profondeur le problème de la classification multi-labels en flux, qui est au cœur de cette thèse. Ainsi, dans ce chapitre, nous commençons par dresser un panorama synthétique de l'état de l'art sur la classification multi-labels « classique » puis nous présentons plus en détail la classification mono-label en flux. Une attention particulière est portée à la fin du chapitre à l'évaluation des modèles qui reste, comme nous le verrons dans la suite, une question encore délicate dans le cas des flux.

1 Classification Multi-Labels Traditionnelle

Dans la suite, on considère $\mathcal{X} = \mathbb{R}^d$ l'espace de d attributs et $\mathcal{Y} = \{0, 1\}^l$ l'espace de l labels. Pour l'apprentissage classique, chaque jeu de données est divisé en deux

ensembles, un pour l'apprentissage $\mathcal{T} = \{(\mathbf{x}_i, \mathbf{y}_i)|1 \leq i \leq |\mathcal{T}|\}$ et l'autre pour le test $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)|1 \leq i \leq |\mathcal{S}|\}$. Tous les exemples de $\mathcal{T}$ et $\mathcal{S}$ sont indépendants et identiquement distribués. Chaque exemple $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y}$ est composé d'un vecteur de d attributs $\mathbf{x}_i = [x_i^1, ..., x_i^d]$ et d'un vecteur de l labels $\mathbf{y}_i = [y_i^1, ..., y_i^l]$ où $y_i^j = 1$ (resp. 0) si le label λ_j, $j = 1$ à l, est présent. La tâche de l'algorithme de la classification multi-labels est d'apprendre une fonction $h : \mathcal{X} \to \mathcal{Y}$ à partir d'un ensemble d'apprentissage $\mathcal{T}$. Cette fonction h doit retourner un vecteur de prédictions $\mathbf{p}_i \in \mathcal{Y}$, $1 \leq i \leq |\mathcal{S}|$, pour chaque exemple non vu dans l'ensemble de test $\mathcal{S}$.

Les hypothèses couramment admises pour le problème multi-labels sont les suivantes [41] :

— L'ensemble des labels $\mathcal{L} = \{\lambda_1, ..., \lambda_l\}$ est prédéfini et interprétable.

— Chaque vecteur d'attributs est associé à au moins un label.

— Le nombre d'attributs d peut être élevé.

— Le nombre d'exemples $|\mathcal{T}| + |\mathcal{S}|$ peut être élevé.

La croissance des volumes de données accessibles dans certaines applications aujourd'hui a conduit à introduire une autre hypothèse : le nombre de labels l peut être également élevé. Par exemple, le nombre de labels dans une application de marquage de photos peut atteindre des dizaines de milliers [42], et le nombre de labels pour une tâche de classification de texte peut atteindre des millions [43]. De plus, les données réelles peuvent avoir deux autres caractéristiques :

— 1) Les labels ne sont généralement pas indépendants les uns des autres ; ils sont corrélés et peuvent apparaître conjointement à des fréquences différentes. Par exemple, les articles de journaux sont plus susceptibles d'être associés à la fois aux catégories "science" et "environnement" qu'aux catégories "environnement" et "sport". Dans une base de données de films, les labels "famille" et "guerre" peuvent n'apparaître que très rarement ensemble.

— 2) Les distributions des données peuvent être déséquilibrées, et les déséquilibres peuvent apparaître à deux niveaux : dans la distribution de chacun des labels, dans la distribution des combinaisons de labels. Par exemple pour le premier cas, sur certains jeux des données multi-labels classiquement utilisés dans les expérimentations, e.g. Mediamille, IMDB, certains labels apparaissent beaucoup plus fréquemment que d'autres.

1.1 Principales approches

Comme nous l'avons indiqué dans l'introduction, les approches de la littérature pour la classification multi-labels peuvent se structurer en trois grandes familles [30] : les approches d'apprentissage par transformation, les approches d'apprentissage par adap-

tation, et les approches ensembles. Nous reprenons ici cette organisation pour présenter les algorithmes multi-labels (Figure 2.1). Dans la suite, nous utilisons à plusieurs reprises les terminologies en anglais qui sont massivement employées même dans la communauté francophone.

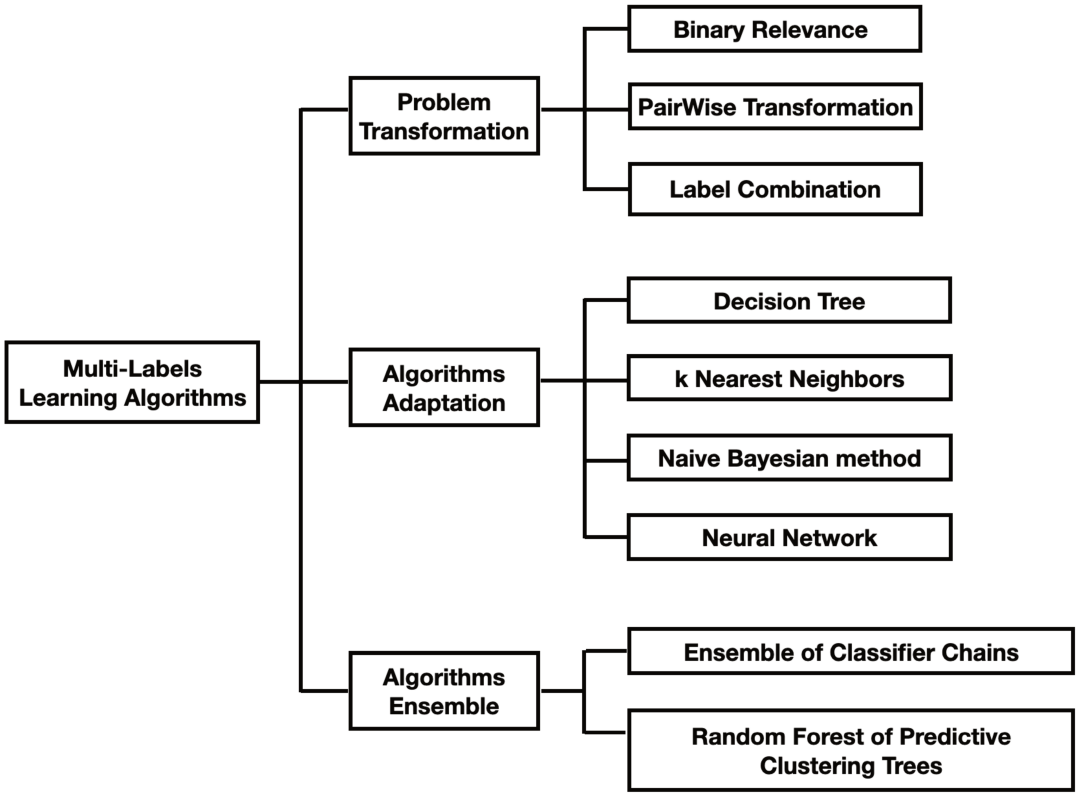


FIGURE 2.1 – Catégorisation des algorithmes d'apprentissage multi-labels représentatifs

1.1.1 Approches par transformations

Les approches par transformation s'appuient principalement sur trois stratégies qui dépendent notamment de leur capacité à prendre en compte ou non les corrélations entre les labels [44] : 1) *Binary Relevance*, 2) *PairWise transformation*, 3) *Label Combination*.

Binary Relevance (BR) est la stratégie par transformation la plus connue [45]. BR décompose le problème multi-labels en l sous-problèmes de classification binaires, chacun étant associé à un seul label. Pour chaque sous-problème du label λ_i, $i = 1$ à l, les exemples de l'ensemble d'entraînement sont considérés comme positifs s'ils sont associés au label λ_i, sinon ils sont négatifs. BR présente deux avantages : 1) la complexité de calcul est linéaire avec le nombre de labels ; 2) chaque label peut être traité en parallèle. En raison de ces deux avantages, de nombreux algorithmes adoptent cette stratégie [46]. Bien que la stratégie BR soit simple, intuitive et largement utilisée, elle souffre souvent de mauvaises performances prédictives car elle ignore la corrélation entre les labels [47].

De plus, nous avons mentionné que la distribution de chaque label n'est pas toujours équilibrée, et la stratégie BR est mal adaptée aux situations où les exemples positifs sont insuffisants pour apprendre la classe positive [48]. Pour exploiter les corrélations entre les labels, J. Read et al [49] ont proposé le modèle des chaînes de classifieur (CC) qui est une amélioration de la méthode BR. Les classifieur binaires sont entraînés dans un ordre aléatoire défini avant la phase d'apprentissage dans le modèle CC. Et chaque classifieur binaire prend en compte toutes les sorties des classifieur précédents comme nouvelles caractéristiques pour l'apprentissage. Cependant, la performance de CC est fortement influencée par l'ordre de la séquence et différentes séquences apportent des performances de classification significativement différentes.

PairWise (PW) transforme le problème multi-labels en $\frac{l(l-1)}{2}$ sous-problèmes de classification binaire, chaque sous-problème correspondant à une paire de labels [50]. Le sous-problème de chaque paire de labels (λ_i, λ_j), $\lambda_i, \lambda_j \in \mathcal{L}$ et $\lambda_i \neq \lambda_j$, contient tous les exemples liés au label λ_i ou λ_j dans le problème original, mais les exemples qui sont liés à ces deux labels simultanément sont exclus. De cette façon, les exemples liés au label λ_i sont des exemples positifs, tandis que les autres sont considérés comme des exemples négatifs. Par conséquent, ce sous-problème peut être résolu par un classifieur binaire qui construit une frontière de décision pour chaque paire de labels. Il convient de noter que, contrairement aux algorithmes de BR, les classifieur binaires de PW ne produisent pas immédiatement le vecteur de prédictions, mais plutôt un ensemble de préférences par paires d'abord. PW effectue la prédiction finale avec une fonction *ranking* supplémentaire qui utilise l'ensemble des préférences comme entrée et obtient un classement en comptant les votes reçus pour chaque label. Les labels dont le nombre de votes est supérieur à un seuil prédéfini sont les labels de la prédiction finale [51]. La complexité temporelle est un problème pour PW : elle est quadratique par rapport au nombre de labels. De plus, dans certaines applications réelles, les corrélations des labels ne sont pas conformes à l'hypothèse d'une corrélation par paire [47].

Label Combination (LC) considère chaque combinaison de labels distincte comme une classe unique [30]. LC consiste donc à entraîner un classifieur multi-classe qui a 2^l classes possibles. Bien que LC puisse prendre en compte directement toutes les corrélations de labels, sa complexité de calcul est très élevée. De plus, le déséquilibre des données peut entraîner des résultats médiocres : certaines combinaisons peuvent apparaître rarement ou pas du tout. L'algorithme *Pruned Sets* (PS) [52] étend LC en essayant d'éviter ces problèmes liés à la complexité et aux données déséquilibrées en élaguant les combinaisons moins fréquentes avec un seuil prédéfini. Mais le nombre de combinaisons de labels fréquents peut également être élevé. L'algorithme Random k labelsets (RAkEL) [53] utilise une stratégie de partitionnement qui divise l'ensemble des combinaisons de labels en m

sous-ensembles. Chaque sous-ensemble contient k combinaisons de labels, qui sont choisies aléatoirement parmi toutes les combinaisons possibles. La valeur de k est prédéfinie par l'utilisateur et la valeur de m est fonction de k et du nombre total de combinaisons de labels. Basé sur ces m sous-ensembles, RAkEL construit m classifieur de LC. En utilisant cette stratégie de partitionnement, la complexité de calcul de RAkEL est beaucoup plus faible. Comme RAkEL, l'algorithme *Clustering-based RAndom Forest of predictive Trees for extreme multi-labels Learning* (CRAFTML) [37] utilise également la stratégie de partitionnement. Cependant, CRAFTML partitionne l'ensemble de combinaisons de labels avec Clustering [54] ce qui permet de placer les combinaisons de labels similaires dans le même sous-ensemble et séparer les combinaisons de labels non-similaires. Ainsi, CRAFTML prend en compte la corrélation de tous les labels en réduisant la complexité du calcul. Actuellement, CRAFTML est un des états de l'art le plus performant.

1.1.2 Approches par adaptation

Les approches par adaptation construisent un modèle pour tous les labels en adaptant des algorithmes mono-label. Nous listons dans la suite les approches par adaptation les plus utilisées.

L'arbre de décision est été adapté dans *Predictive Clustering Trees* (PCT) [55] pour le traitement de données multi-labels. Chaque arbre PCT est construit comme un arbre d'instances/de décision standard. La condition de séparation à chaque nœud est monovariée (un attribut, un seuil) et vise à minimiser la variance des labels dans les deux sous-ensembles obtenus. Le critère de coupure correspond à la maximisation du critère de gain d'information basé sur l'entropie multi-labels. En supposant l'indépendance entre les labels, l'entropie multi-labels est calculée d'une manière décomposable :

$$Entropy(\mathcal{T}) = -\sum_{i=1}^{l}(\mathbb{P}(\lambda_i)log_2(\mathbb{P}(\lambda_i)) + (1-\mathbb{P}(\lambda_i))log_2(1-\mathbb{P}(\lambda_i))) \tag{2.1}$$

où $\mathbb{P}(\lambda_i)$ présente la fréquence relative du label λ_i dans l'ensemble d'apprentissage $\mathcal{T}$. La stratégie du vote majoritaire est intégrée dans les feuilles pour donner la prédiction finale. L'un des principaux avantages de PCT est sa grande efficacité et elle peut être facilement utilisée avec les méthodes ensembles [47].

Les k plus proches voisins (k-NN) combinent la règle du maximum a posteriori (MAP) pour faire des prédictions sur les données multi-labels [56]. Pour prédire chaque exemple de test, ML-kNN trouve k exemples dont les vecteurs d'attributs sont les plus proches de $\mathbf{x}$ dans l'ensemble d'apprentissage et compte le nombre d'occurrences de chaque label dans ces k voisinages, puis combine ce compte avec les probabilités a

posteriori de chaque label à partir des exemples d'apprentissage pour calculer les labels les plus probables comme prédiction. ML-kNN est simple et peut atténuer le problème du déséquilibre des classes dû aux probabilités antérieures estimées pour chaque label. Cependant, il doit effectuer un grand nombre de calculs et de comparaisons de distances pour trouver les k plus proches voisins dans les données d'apprentissage. De plus, il ignore complément l'exploitation des corrélations entre les labels.

L'approche Naïve Bayes a été adaptée dans [57] pour les données multi-labels en utilisant la stratégie BR qui prédit chaque label avec un classifieur binaire : la valeur de la prédiction p^i, $i = 1$ à l est déterminée en utilisant la méthode du maximum a posteriori (MAP) :

$$p^i = \arg \max_{b \in \{0,1\}} \mathbb{P}(H_b^i | \mathbf{x}) \tag{2.2}$$

où $\mathbb{P}(H_0^i|\mathbf{x})$ (resp. $\mathbb{P}(H_1^i|\mathbf{x})$) représente la probabilité de l'évènement H_0^i (resp. H_1^i) "le label λ_i est absent (resp. présent) sachant $\mathbf{x}$". En adoptant l'hypothèse d'indépendance conditionnelle entre les attributs comme le font les classifieurs classiques de Naïve Bayes, l'équation 2.2 peut être réécrite comme suit :

$$p^i = \arg \max_{b \in \{0,1\}} \frac{\mathbb{P}(H_b^i)\mathbb{P}(\mathbf{x}|H_b^i)}{\mathbb{P}(\mathbf{x})} \sim \arg \max_{b \in \{0,1\}} \mathbb{P}(H_b^i) \prod_{j=1}^{d} \mathbb{P}(x^j | H_b^i) \tag{2.3}$$

La probabilité conditionnelle $\mathbb{P}(x^j|H_b^i)$ est calculée à l'aide d'une approximation gaussienne. En pratique, le calcul des probabilités peut être très complexe lorsque le nombre d'attributs est élevé.

Le réseau neuronal peut être appliqué simplement à la classification multi-labels en prédéfinissant un seuil. Supposons que les l unités de sortie d'un réseau neuronal correspondent à l labels. Si la valeur d'unité de sortie o^i, $i = 1$ à l, est supérieure au seuil préfixé, alors la valeur de la prédiction p^i est égal à 1, sinon à 0. Et l'algorithme classique *Backpropagation* [58] peut être utilisé pour minimiser l'erreur. Pour chaque vecteur d'attributs $\mathbf{x}$, l'erreur est calculée par la distance classique :

$$erreur = \sum_{i=1}^{l} (o^i - y^i)^2 \tag{2.4}$$

où y^i représente la vraie valeur de chaque label. La somme des erreurs de tous les exemples d'entraînement est utilisée pour optimiser les paramètres du réseau neuronal. Cependant, cette fonction d'erreur classique ne considère pas les corrélations entre les différents labels. Pour pallier ce manque, [59] a modifié la fonction (1.4) en s'inspirant de la perte de classement par labels - la valeur de sortie du label associé à $\mathbf{x}$ doit être supérieure à la

valeur de sortie du label non associée- :

$$erreur = \frac{1}{|\mathcal{A}||\bar{\mathcal{A}}|} \sum_{(i,j)\in\mathcal{A}\times\bar{\mathcal{A}}} \exp(-(o^i - o^j)) \tag{2.5}$$

où $\mathcal{A}$ est l'ensemble des index du label associé à $\mathbf{x}$, $\mathcal{A} \subseteq \{1, .., l\}$, et $\bar{\mathcal{A}}$ l'ensemble des index du label non-associé. Le nombre de paramètres dans le réseau neuronal est proportionnel au nombre d'attributs et de labels.

1.1.3 Approches ensembles

Les approches ensembles combinent les prédictions de plusieurs classifieurs pour obtenir de meilleures performances, ce qui s'est avéré très efficace dans la classification mono-label. Les approches les plus connues sont les méthodes Ensembles de Classifieur en Chaînes(ECC) [49] et les forêts aléatoires (Random Forest of Predictive Clustering Trees RF-PCT) [55].

ECC entraîne plusieurs classifieurs CC simultanément. Pour diversifier les classifieurs, elles sous-échantillonnent de façon répétitive (avec remise) l'ensemble d'apprentissage (i.e. bagging [60]). Par ailleurs, chaque classifieur CC est entraîné suivant un ordre de label différent défini aléatoirement. Pour une nouvelle instance, les prédictions de tous les classifieurs sont combinées à l'aide d'un vote majoritaire.

RF-PCT entraîne un ensemble de arbres de décisions multi-labels PCT. Pour introduire de la diversité, les arbres sont appris sur des sous-échantillons (crées de manière répétitive avec remise) de l'ensemble d'apprentissage (i.e. bagging [60]) et en sélectionnant des sous-ensembles d'attributs aléatoires pour entraîner chaque arbre. La prédiction globale de la forêt est alors la moyenne des prédictions de chaque arbre.

L'avantage de ces approches ensembles est qu'elles tentent d'améliorer les performances de leurs classifieurs de base en multipliant les modèles. Néanmoins, leur complexité d'apprentissage croît linéairement avec le nombre de classifieurs appliqués.

1.2 Évaluation des classifieurs multi-labels

Les critères les plus utilisés pour évaluer la performance prédictive d'un classifieur multi-labels peuvent être regroupés dans deux familles principales : ceux qui dépendent des exemples, et ceux qui dépendent des labels. Les critères basés sur les exemples dépendent d'une évaluation préalable de la performance du modèle d'apprentissage sur chaque exemple pris séparément, puis sur un calcul d'une valeur moyenne sur tous les

exemples. Les critères basés sur les labels évaluent les performances du modèle pour chaque label séparément, puis renvoient une valeur moyenne sur tous les labels.

1.2.1 Critères basés sur les exemples

Pour chaque exemple, les mesures basées sur les exemples évaluent un écart entre le vrai vecteur de labels $\mathbf{y}$ et le vecteur de prédiction $\mathbf{p} = h(\mathbf{x})$. Elles sont alors définies par une agrégation des écarts obtenus sur tous les exemples de l'ensemble de test $\mathcal{S}$. Les mesures différent selon les définitions choisies pour l'écart et l'agrégation.

Exact Match vérifie si le vecteur de prédictions $\mathbf{p}$ est identique au vrai vecteur de labels $\mathbf{y}$. Il est défini par :

$$ExactMatch = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} I[\mathbf{p}_i = \mathbf{y}_i]$$

où $I[vrai] = 1$ et $I[faux] = 0$.

Hamming Loss calcule la proportion des labels mal prédits. Il est défini par :

$$HammingLoss = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \frac{1}{l}|\mathbf{y}_i \Delta \mathbf{p}_i|$$

où Δ est la différence symétrique entre le vecteur de labels $\mathbf{y}$ et le vecteur de prédictions $\mathbf{p}$. Notons que la *Hamming Loss* est généralement utilisée dans le *Hamming Score*, c-à-d, $HammingScore = 1 - HammingLoss$.

Accuracy évalue la similarité de Jaccard entre le vrai vecteur de labels $\mathbf{y}$ et le vecteur de prédictions $\mathbf{p}$. Il est défini par :

$$Accuracy = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \frac{|\mathbf{y}_i \cap \mathbf{p}_i|}{|\mathbf{y}_i \cup \mathbf{p}_i|}$$

Precision calcule la proportion des labels correctement prédits sur tous les labels prédits positifs, i.e. le taux de bonnes prédictions. Il est défini par :

$$Precision = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \frac{|\mathbf{y}_i \cap \mathbf{p}_i|}{|\mathbf{p}_i|}$$

Rappel calcule la proportion des labels correctement prédits sur tous les vrais labels. Il est défini par :

$$Rappel = \frac{1}{|\mathcal{S}|}\sum_{i=1}^{|\mathcal{S}|} \frac{|\mathbf{y}_i \cap \mathbf{p}_i|}{|\mathbf{y}_i|}$$

Fmesure est la moyenne harmonique entre les critères de rappel et de Precision. Elle

est définie par :

$$Fmesure = \frac{2Precision \cdot Rappel}{Precision + Rappel} = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \frac{2 \times |\mathbf{y}_i \cap \mathbf{p}_i|}{|\mathbf{y}_i| + |\mathbf{p}_i|}$$

Tous les critères sont définis dans l'intervalle $[0, 1]$ et leurs plus grandes valeurs indiquent les meilleures performances hormis pour le Hamming Loss pour lequel la plus petite valeur indique la meilleure performance.

1.2.2 Critères basés sur les labels

Les mesures basées sur les labels sont des fonctions de quatre indicateurs bien classiques en classification, à savoir les vrais positifs (VP), faux positifs (FP), vrais négatifs (VN) et faux négatifs (FN). Pour chaque label λ_j, $j = 1$ à l, leurs définitions sont comme suit :

$$VP^j = \sum_{i=1}^{|\mathcal{S}|} \mathbb{1}|y_i^j = 1 \wedge p_i^j = 1 \qquad FP^j = \sum_{i=1}^{|\mathcal{S}|} \mathbb{1}|y_i^j = 0 \wedge p_i^j = 1$$

$$VN^j = \sum_{i=1}^{|\mathcal{S}|} \mathbb{1}|y_i^j = 0 \wedge p_i^j = 0 \qquad FN^j = \sum_{i=1}^{|\mathcal{S}|} \mathbb{1}|y_i^j = 1 \wedge p_i^j = 0$$

Ces quatre quantités nous permettent d'étendre les définitions de précision, de rappel et de *Fmesure* d'une manière basée sur les labels. Cependant, nous avons deux choix pour combiner les contributions de chaque label, la macro- et la micro-moyenne. Dans le cas de la macro-moyenne, chaque mesure est calculée par label puis ensuite une moyenne des mesures est effectuée. Pour la micro-moyenne les valeurs VP, FP, VN et FN sont initialement calculées sur tous les labels.

Mesures macro-moyennes. La macro-moyenne suppose des "poids égaux" pour chaque label. Les définitions de la précision et du rappel sont comme suit :

$$Precision_{macro} = \frac{1}{l} \sum_{j=1}^{l} \frac{VP^j}{VP^j + FP^j} \qquad Rappel_{macro} = \frac{1}{l} \sum_{j=1}^{l} \frac{VP^j}{VP^j + FN^j}$$

La définition de la *Fmesure* macro-moyenne peut être étendue en remplaçant les formules de précision et de rappel, ce qui donne la formule suivante :

$$Fmesure_{macro} = \frac{1}{l} \sum_{j=1}^{l} \frac{2Precision_{macro}^j \cdot Rappel_{macro}^j}{Precision_{macro}^j + Rappel_{macro}^j}$$

Mesures micro-moyennes. La micro-moyenne agrège les contributions de tous les labels pour calculer la métrique moyenne. Les définitions de la précision et du rappel pour la micro-moyenne sont les suivantes :

$$Precision_{micro} = \frac{\sum_{j=1}^{l} VP^j}{\sum_{j=1}^{l} VP^j + \sum_{j=1}^{l} FP^j} \qquad Rappel_{micro} = \frac{\sum_{j=1}^{l} VP^j}{\sum_{j=1}^{l} VP^j + \sum_{j=1}^{l} FN^j}$$

La formule de la *Fmesure* micro-moyenne est donc donnée dans ce qui suit :

$$Fmesure_{micro} = \frac{2Precision_{micro} \cdot Rappel_{micro}}{Precision_{micro} + Rappel_{micro}}$$

1.2.3 Discussion

Nowak et al. [61] ont comparé les performances des critères basés sur les exemples à celles des critères basés sur les labels et ont conclu que les critères basés sur les exemples sont plus adaptés aux problèmes de classification multi-labels car ils considèrent tous les labels simultanément, et peuvent ainsi prendre en compte des corrélations entre les labels. Nous considérons donc uniquement dans la suite des critères d'évaluation basés sur les exemples.

Certaines études ont été mené spécifiquement une analyse critique sur les différentes métriques de la famille basé sur les exemples. Plus précisément, la mesure *Exact Match* est souvent considérée comme un critère d'évaluation très strict parce qu'il punit sévèrement les prédictions du classifieur [47] et le *Hamming Loss* peut favoriser le modèle qui donne la prédiction nulle parce que la fréquence de la présence de chaque label est très faible. Autrement dit, son intérêt principal réside dans l'identification des vrais négatifs [62]. Nous ne retenons donc dans nos évaluations que les deux critères {*Accuracy* et *Fmesure*} basés sur les exemples qui sont les plus utilisés et les plus recommandés dans la littérature. Dans la section suivante, nous présentons le problème de classification mono-label en flux afin de mieux comprendre les enjeux qui dépendent des différentes caractéristiques du flux.

1.3 Conclusion

Dans cette section, nous avons présenté synthétiquement l'état de l'art pour le problème classique de de la classification multi-labels. Les comparaisons expérimentales montrent que les méthodes ensembles sont les plus performantes. Mais leur complexités en calcul et en mémoire peuvent augmenter de façon importante avec le nombre de modèles. On retrouve ici aussi le problème classique de la recherche de l'équilibre entre la complexité des calculs et les performances du modèle. Ce problème est encore plus complexe pour la classification multi-labels sur les flux de données. Dans ce chapitre, nous

présentons le problème de classification mono-label en flux afin de mieux comprendre les enjeux qui dépendent des différentes caractéristiques du flux.

2 Classification Mono-Label en flux

Dans ce qui suit, nous précisons les caractéristiques de la classification des données en flux en nous restreignant au cadre mono-label. D'une part, ce cadre est celui qui a été le plus traité dans la littérature, et d'autre part il permet de guider l'intuition sur les défis spécifiques du passage de l'apprentissage « off-line » à l'apprentissage « on-line ».

2.1 Caractéristiques du problème

Rappelons qu'en classification mono-label, on cherche à prédire une variable cible $y \in [1, l]$ à partir d'un vecteur d'attributs/entrées $\mathbf{x} \in \mathcal{X}$. La variable cible y est l'indice du label associé à $\mathbf{x}$. Dans la suite, on considère un flux de données mono-label $\mathcal{D}_s = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_t, y_t), ...\}$, où chaque exemple $(\mathbf{x}_t, y_t) \in \mathcal{X} \times [1, l]$ est généré selon une distribution de probabilité jointe $\mathbb{P}_t(\mathbf{x}, y)$ inconnue à l'instant t. Dans le contexte du problème en flux, la distribution de probabilité jointe $\mathbb{P}_t(\mathbf{x}, y)$ est appelée aussi "concept". Le flux $\mathcal{D}_s$ est stationnaire lorsque $\mathbb{P}_t(\mathbf{x}, y)$ est stable au cours du temps, et non stationnaire sinon [63].

Pour préciser le problème, quatre hypothèses sous-jacentes à l'apprentissage on-line doivent être explicitées : 1) la mémoire et la capacité de calcul des algorithmes sont limitées ; 2) le nombre d'exemples du flux est potentiellement infini ou beaucoup plus grand que la taille de la mémoire ; 3) la vitesse du flux de données est rapide ; 4) la distribution de données $\mathbb{P}_t(\mathbf{x}, y)$ peut évoluer au fil du temps. Il est important de noter que la majorité des travaux existants supposent implicitement ou explicitement que les données d'un flux non-stationnaire ne sont pas distribuées de manière identique mais néanmoins indépendante [64], [65], et nous avons donc adopté cette hypothèse.

Ainsi, afin de satisfaire ces hypothèses, les algorithmes de la classification en flux doivent répondre aux cinq exigences suivantes [66] :

— être capables de fournir une prédiction à tout moment étant donnée une entrée $\mathbf{x}$.
— apprendre chaque exemple une seule fois (au maximum) et dans son ordre d'arrivée.
— utiliser une quantité de mémoire limitée.
— utiliser un temps de calcul limité pour prédire et apprendre.
— être capable de s'adapter aux changements de la distribution de données.

La tâche de l'apprentissage on-line consiste donc à générer un nouveau modèle h_t à chaque instant t à partir du modèle précédent h_{t-1} et du nouvel exemple $(\mathbf{x}_t, y_t)$ avant

l'arrivée du prochain exemple. Le nouveau modèle h_t doit pouvoir donner une prédiction associée aux labels $p_{t+1} \in [1, l]$ pour le prochain exemple. De plus, les modèles doivent non seulement apprendre de manière incrémentale en cours du temps lorsque $\mathbb{P}_t(\mathbf{x}, y)$ est stable (cas stationnaire), mais aussi s'adapter rapidement lorsque $\mathbb{P}_t(\mathbf{x}, y)$ change. Le processus général d'apprentissage on-line est décrit dans la figure 1.2, et les différentes situations de changements de la distribution sont détaillées dans la section 2.3.

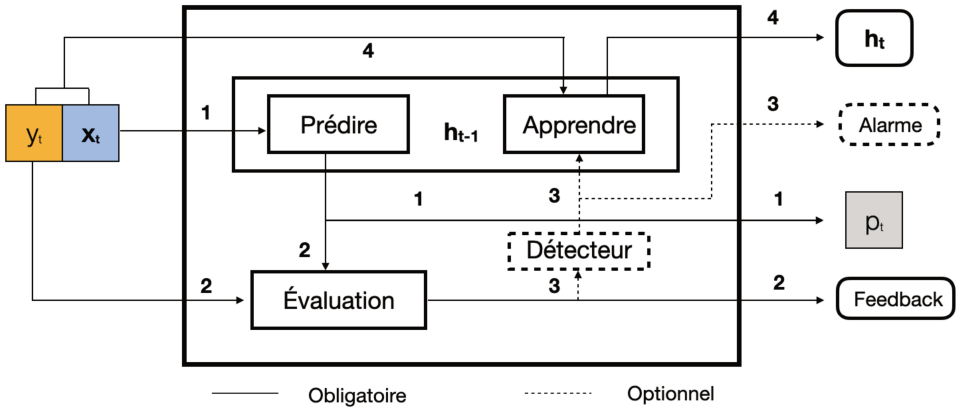


FIGURE 2.2 – Procédure générale d'apprentissage on-line à chaque instant : 1) le modèle h_{t-1} donne la prédiction p_t ; 2) le modèle h_{t-1} est évalué en comparant p_t et y_t ; 3) détection de la dérive ; 4) le nouveau modèle h_t est construit. (extrait de[1])

Dans la suite, nous présentons brièvement les principaux algorithmes qui peuvent apprendre de manière incrémentale en nous concentrant tout d'abord sur le cas des flux stationnaires, puis en étendant aux flux non stationnaires. La dernière partie du paragraphe est consacrée à la présentation des différentes modalités de la dérive de concepts (« concept drift ») et aux stratégies développées pour s'y adapter.

2.2 Classification mono-label sur des flux stationnaires

De nombreux algorithmes d'apprentissage off-line ont été adaptés au problème d'apprentissage on-line. Nous présentons ici les plus connus : l'arbre de décision on-line, les mémoires combinées avec les k plus proches voisins, l'approche naïve bayes on-line, les systèmes à base de règles et les réseaux neuronaux.

L'arbre de décision on-line est induit à partir du flux de données de manière incrémentale en utilisant le principe de la borne de Hoeffding [67], d'où son nom d'arbre d'Hoeffding dans le cas de l'apprentissage on-line. La borne de hoeffing est définie de la manière suivante. Soit une variable aléatoire x dont les valeurs sont comprises dans un intervalle de longueur R et $\overline{x_n}$ sa moyenne empirique après n observations indépendantes. La borne d'Hoeffding assure, avec une probabilité de $1 - \delta$, que la différence entre la moyenne empirique $\overline{x_n}$ et l'espérance de x ne soit pas supérieure à ε avec $\varepsilon = \sqrt{\frac{R^2}{2n} \ln \frac{1}{\delta}}$.

L'intérêt de cette borne est qu'elle ne dépend pas de la distribution des valeurs mais seulement de trois paramètres : la longueur de l'intervalle R, le nombre d'observations n et la confiance désirée δ. Cependant, cette borne est plus conservatrice que des bornes prenant en compte de la distribution des valeurs.

Dans le cadre de l'apprentissage off-line, chaque noeud dans un arbre de décision est déterminé par l'attribut présentant le gain d'information le plus élevé sur l'ensemble des données disponibles. Afin de conserver cette stratégie dans un environnement de flux de données, Domingos et Hulten ont adapté la borne d'Hoeffding pour assurer que le choix de l'attribut de coupure dans les seuls n exemples observés sur le flux est le bon. Ainsi, dans chaque feuille, les gains d'information (IG) pour chaque attribut sont calculés avec n exemples mémorisés. Pour les deux attributs (x^i et x^j) ayant le gain d'information le plus élevé, si $IG(x^i) - IG(x^j) > \sqrt{\frac{R^2}{2n} \ln \frac{1}{\delta}}$, la feuille est divisée sur l'attribut x^i.

Les mémoires combinées avec un clustering constituent une stratégie de base de structures de données pouvant stocker des connaissances sur le flux avec une complexité limitée. Les différentes mémoires combinent généralement une structuration des informations obtenues avec les k plus proches voisins et un vote majoritaire pour la prédiction. L'une des techniques les plus représentatives est la règle d'échantillonnage - *Reservoir Sampling* [68] qui permet d'obtenir un échantillon non biaisé du flux d'une taille prédéfinie.

Plus récemment, Losing [69] a proposé une autre technique pour stocker les informations représentatives, appelée mémoire à long-terme. Sa mémoire à long terme est un ensemble $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$ de m exemples. Une fois que la mémoire est remplie par ces m exemples, ils sont regroupés en l clusters $\{C_i, i \in [1, l]\}$ selon leurs labels : les exemples ayant le même label sont regroupés dans le même cluster. Ensuite, pour limiter la taille de la mémoire, l'algorithme de clustering k-Means [70] est utilisé dans chaque cluster pour obtenir $\frac{|C_i|}{2}$ sous-clusters. Chaque sous-cluster est représenté par son centroïde $(\mathbf{c}^{\mathbf{x}}{}_j, c_j^y) \in \mathcal{X} \times \mathcal{L}$, $j \in [1, \frac{|C_i|}{2}]$, et la mémoire à long-terme est mise à jour par l'ensemble des $\frac{|C_i|}{2}$ centroïdes de chaque sous-cluster.

L'approche naïve bayes on-line calcule une probabilité pour chacune des labels en fonction des valeurs d'attributs, et sélectionne le label ayant la probabilité la plus élevée dans le temps [71]. En supposant l'indépendance des attributs, la probabilité de chaque label est calculée comme suit :

$$\mathbb{P}(y|\mathbf{x}) \cong \mathbb{P}(y) \cdot \prod_{i=1}^{d} \mathbb{P}(x^i|y) \tag{2.6}$$

où $\mathbb{P}(y)$ est la probabilité priori et $\mathbb{P}(x^i|y)$ est la vraisemblance de l'attribut x^i étant donnée la présence du label y. Sur le flux, les valeurs de $\mathbb{P}(y)$ et $\mathbb{P}(x^i|y)$ sont estimées en

enregistrant les fréquences d'occurrences en temps réel.

Le système à base de règles repose sur un ensemble de règles de décision qui apportent un facteur d'explicabilité supplémentaire. Rivest [72] a proposé d'utiliser un système à base de règles de décision pour généraliser les arbres de décision. L'avantage de cette représentation est l'interprétabilité : chaque règle peut être interprétée indépendamment des autres et peut être supprimée sans affecter les autres . Dans le cas du flux, le plus populaire des systèmes à base de règle est l'algorithme AMRules [73]. Dans AMRules, chaque règle est une conjonction de littéraux et chaque littéral est une condition de deux formes possibles : $x^i = v$ si x^i est un attribut discret, et v une de ses valeurs ou $v \leq x^i$ ou $x^i \geq v$ si x^i est un continu et v une valeur réelle. Dans ce schéma, une règle couvre une entrée $\mathbf{x}$ si elle satisfait à toutes les conditions de la règle et le label le plus fréquent des exemples couverts par cette règle sert de la prédiction pour la nouvelle entrée $\mathbf{x}$. La création de chaque littéral dans la règle est similaire à la création du noeud d'un arbre de Hoeffding. Lorsqu'aucune des règles ne satisfait le nouvel exemple, une nouvelle règle est créée.

Les réseaux neuronaux on-line qui sont mis à jour avec une descente de gradient stochastique [74] sont souvent considérés comme l'une des premières méthodes d'apprentissage on-line. Pour avoir un processus d'apprentissage rapide, l'*Online Sequential-Extreme Learning Machine* (OS-ELM) basé sur un réseau de neurones n'ayant qu'une seule couche cachée a attiré beaucoup d'attention récemment [75]. Les poids entre les entrées et les nœuds cachés sont répartis au hasard et jamais mis à jour ; seuls les poids entre les nœuds cachés et les sorties sont appris en une seule étape à chaque instant. Supposons que les l unités de sortie d'OS-ELM, o^i, $i = 1$ à l, correspondent respectivement à l labels et que $\mathbf{o}$ est le vecteur de sortie. Un OS-ELM peut être représenté comme suit :

$$\mathbf{o} = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}) \tag{2.7}$$

où $\mathbf{W}_1$ est la matrice des pondérations d'entrée-à-couche cachée et $\mathbf{W}_2$ la matrice des pondérations de couche cachée-à-sortie. La vraie variable cible y du vecteur d'entrée $\mathbf{x}$ peut être représentée par un vecteur binaire $\mathbf{y}$, où uniquement la valeur de l'indice y est 1 et les valeurs des autres indices sont 0. L'objectif général est de minimiser $|\mathbf{o} - \mathbf{y}|$. Le processus d'apprentissage pour OS-ELM est d'optimiser les poids de sortie $\mathbf{W}_2$ dans l'équation $\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}) = \mathbf{y}$. Puisque $\mathbf{W}_1$ est fixe, le processus de calcul de $\mathbf{W}_2$ est essentiellement le même que celui de l'apprentissage d'un modèle linéaire.

2.3 Classification mono-label sur des flux non-stationnaires

La différence majeure avec le cas précédent est la nécessité de prendre en compte les dérives de concept. Dans la littérature sur la classification des données en flux, un concept est généralement défini par la probabilité jointe $\mathbb{P}_t(\mathbf{x}, y)$. Et classiquement, une dérive de concepts est associée à une évolution temporelle de la distribution de probabilité jointe $\mathbb{P}_t(\mathbf{x}, y)$ [63]. Plus formellement, entre deux pas de temps $t_1 < t_2$:

$$\mathbb{P}_{t_1}(\mathbf{x}, y) \neq \mathbb{P}_{t_2}(\mathbf{x}, y) \tag{2.8}$$

Les modèles sur des flux doivent donc être capables de s'adapter à ces changements de distribution pour maintenir une haute performance prédicative dans le temps. Plus précisément, la distribution conjointe $\mathbb{P}_t(\mathbf{x}, y)$ peut être décomposée comme suit :

$$\mathbb{P}_t(\mathbf{x}, y) = \mathbb{P}_t(y|\mathbf{x})\mathbb{P}_t(\mathbf{x}) = \mathbb{P}_t(\mathbf{x}|y)\mathbb{P}_t(y) \tag{2.9}$$

et donc un changement dans l'une de ces quatre composantes de la distribution conjointe peut entraîner une dérive de concepts [76]. Or pour la tâche de classification, $\mathbb{P}_t(y|\mathbf{x})$ est la probabilité postérieure des labels qui dépend de la probabilité à priori $\mathbb{P}_t(y)$ et de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|y)$ [77]. Selon le théorème de Bayes, $\mathbb{P}_t(y|\mathbf{x})$ peut être représentée comme suit :

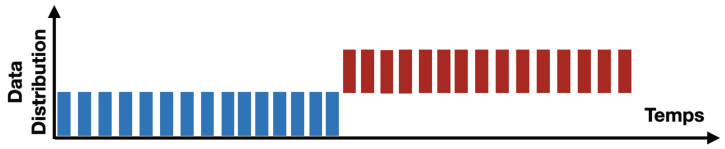
$$\mathbb{P}_t(y|\mathbf{x}) = \frac{\mathbb{P}_t(\mathbf{x}|y)\mathbb{P}_t(y)}{\mathbb{P}_t(\mathbf{x})} \tag{2.10}$$

où $\mathbb{P}_t(\mathbf{x}) = \int_{d_y} \mathbb{P}_t(\mathbf{x}|y)\mathbb{P}_t(y)$. Les changements de $\mathbb{P}_t(\mathbf{x}|y)$ ou/et $\mathbb{P}_t(y)$ peuvent entraîner un changement de la probabilité postérieure des labels qui peut donc conduire à une dégradation de la performance de prédiction du modèle. Les algorithmes d'apprentissage doivent donc être capables d'adapter leur modèle en conséquence.

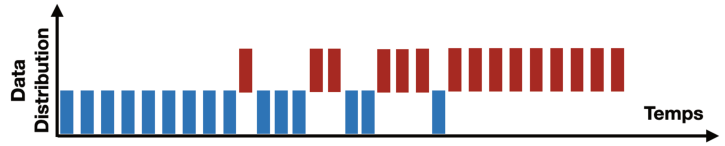
Les raisons pour lesquelles ces changements peuvent dégrader les performances du modèle ont été, à notre connaissance, peu explicitées dans la littérature. Or, leur connaissance est nécessaire pour comprendre et évaluer plus précisément les comportements des algorithmes. Nous les précisons donc ci-dessous.

Nous pouvons distinguer trois causes de dérives de concepts que nous illustrons à titre pédagogique dans un cas simple de classification binaire d'images de chats et de chiens.

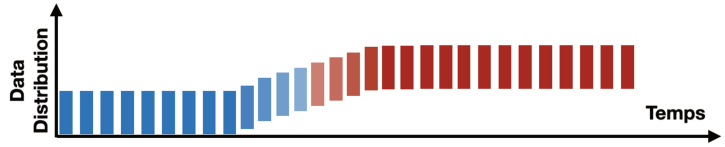
— **Cause I** : $\mathbb{P}_{t1}(y) \neq \mathbb{P}_{t2}(y)$ mais $\mathbb{P}_{t1}(\mathbf{x}|y) = \mathbb{P}_{t2}(\mathbf{x}|y)$. Lorsque $\mathbb{P}_t(\mathbf{x}|y)$ est constante, cela signifie que les caractéristiques des images associées aux chiens et celles associées aux chats n'ont pas changé. Cependant, la proportion du nombre d'images de chiens et de chats peut changer au cours du temps $\mathbb{P}_{t1}(y) \neq \mathbb{P}_{t2}(y)$. Par exemple, nous pouvons imaginer qu'avant l'instant t_2, il y a seulement des images de chiens qui ont été présentes dans le flux et, qu'après l'instant t_2, seules des images de



(a) Abrupte : le nouveau concept remplace l'ancien en peu de temps.



(b) Graduelle : le nouveau concept remplace progressivement un ancien concept sur une période de temps.



(c) Incrémentale : l'ancien concept se transforme progressivement en un nouveau concept sur une période de temps.

FIGURE 2.3 – Trois types de dérive de concept.

chats apparaissent. Puisque le modèle ne peut être entraîné initialement que sur des images de chiens, les images de chats ne vont pas être bien prédites pendant une certaine période ; ce qui entraîne une dégradation des performances.

— **Cause II** $\mathbb{P}_{t1}(\mathbf{x}|y) \neq \mathbb{P}_{t2}(\mathbf{x}|y)$ mais $\mathbb{P}_{t1}(y) = \mathbb{P}_{t2}(y)$. Par exemple, avant l'instant t_2, supposons une situation où il y a seulement des images de chiens et de chats blancs dans le flux et, qu'après l'instant t_2, seulement des images de chiens et chats noirs apparaissent. Puisque le modèle n'a jamais rencontré d'images de chiens et chats noirs, sa performance peut être aussi dégradée.

— **Cause III** $\mathbb{P}_{t1}(\mathbf{x}|y) \neq \mathbb{P}_{t2}(\mathbf{x}|y)$ et $\mathbb{P}_{t1}(y) \neq \mathbb{P}_{t2}(y)$. La probabilité à priori $\mathbb{P}_t(y)$ et la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|y)$ peuvent changer simultanément. Par conséquent, la performance du modèle peut être dégradée plus gravement.

2.3.1 Types de dérives

Afin de fournir une description qualitative plus claire des différents types de dérives, il convient de préciser trois termes : la sévérité de la dérive, la durée de la dérive et la période de stabilité [78] :

— La sévérité évalue une différence entre deux concepts différents. Plus formellement, elle peut se définir par $d(\mathbb{P}_{t_1}(\mathbf{x}, y), \mathbb{P}_{t_2}(\mathbf{x}, y))$, où d est une fonction permettant de mesurer la divergence de deux distributions de données. Plus la valeur de la distance est grande, plus la dérive est considérée comme sévère.

— La durée de la dérive τ, aussi appelée la période d'instabilité, est le temps qui s'écoule entre l'instant t_s où la distribution de données commence à changer et l'instant t_e où la distribution de données redevient stable, $\tau = t_e - t_s$. Plus formellement, $\forall t \in [t_s, t_e], d(\mathbb{P}_{t_s}(\mathbf{x}, y), \mathbb{P}_t(\mathbf{x}, y)) > 0$.

— La période de stabilité est une période pendant laquelle un concept reste stable. Plus formellement, nous pouvons définir un intervalle de temps $[t_v, t_u]$ et $\forall t \in [t_v, t_u], d(\mathbb{P}_{t_v}(\mathbf{x}, y), \mathbb{P}_t(\mathbf{x}, y)) = 0$.

Ainsi, un flux peut être considéré comme étant composé d'un certain nombre de périodes de stabilité, entrecoupées de périodes d'instabilité, et chaque dérive sur le flux peut être classée en trois types (Figure 2.3) :

— **abrupte**, lorsque la durée de la dérive τ est très faible et la sévérité de la dérive est grande.

— **graduelle**, lorsque pendant une grande durée de la dérive τ, les deux concepts successifs avec une grande sévérité coexistent. En particulier, avec le temps, les données proviennent moins fréquemment de l'ancien concept et plus du nouveau.

— **incrémentale**, lorsqu'un concept évolue lentement vers un autre pendant la durée de la dérive avec la co-présence de plusieurs concepts intermédiaires entre les deux. Supposons l'instant t_s où le premier concept $\mathbb{P}_{t_s}(\mathbf{x}, y)$ commence à changer et l'instant t_e où le deuxième concept $\mathbb{P}_{t_e}(\mathbf{x}, y)$ devient stable. La sévérité de ces deux concepts, $\Delta = d(\mathbb{P}_{t_s}(\mathbf{x}, y), \mathbb{P}_{t_e}(\mathbf{x}, y))$, peut être grande. Pendant la durée de la dérive $[t_s, t_e]$, $\exists t \in [t_s, t_e]$, des concepts intermédiaires entre le premier et le deuxième concept existent. La différence Δ_t entre ces concepts intermédiaires et le premier concept est toujours inférieure à Δ, $\Delta_t = d(\mathbb{P}_{t_s}(\mathbf{x}, y), \mathbb{P}_t(\mathbf{x}, y)) \leq \Delta$. Au fur et à mesure que t augmente, Δ_t augmente jusqu'à ce qu'elle soit égale à Δ.

Lorsqu'une dérive apparaît dans le flux, le ou les concepts concernés peuvent être soient des nouveaux concepts jamais rencontrés précédemment, soient des concepts déjà rencontrés, puis ayant disparu, qui réapparaissent ; on parle alors de **concepts récurrents**.

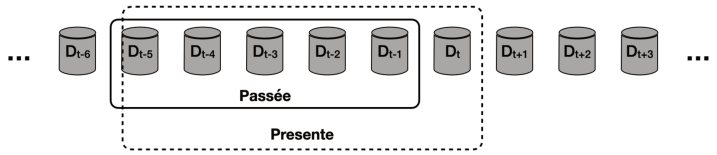
Ces différents comportements peuvent être précisés en établissant une analogie avec une séquence de tirages aléatoires avec remise de boules dans une urne. Lorsque la distribution de probabilité des tirages est uniforme (resp. varie au cours du temps), la modélisation s'apparente à un flux de données stationnaire (resp. non-stationnaire). La dérive de concept abrupte s'apparente au cas où, à un instant arbitraire, l'urne est échangée avec une autre. La dérive graduelle est analogue à la co-existence pendant une période de deux urnes dans lesquelles les tirages se font alternativement. La dérive incrémentale correspond à des changements progressifs -par ajout ou retrait de boules- de la composition de l'urne pendant la période de transition.

2.4 Stratégie pour l'adaptation aux dérives de concept

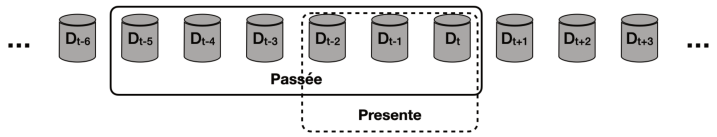
De façon générale, les stratégies qui prennent en compte la dérive de concept se structurent en deux grandes familles [1] : les approches actives, également nommées approches d'adaptation informée, qui mettent à jour le modèle lorsqu'un changement est détecté, et les approches passives, aussi nommées approches aveugles, qui mettent à jour le modèle avec une temporalité préfixée indépendante de l'apparition d'une dérive de concept.

2.4.1 Approches actives

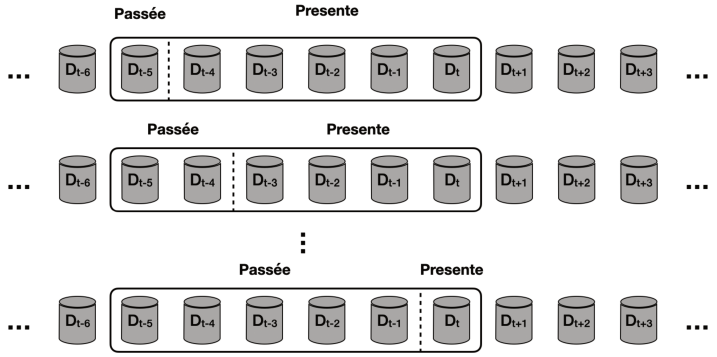
Les approches actives comportent deux composantes principales : un détecteur qui identifie l'instant du changement, et un modèle qui réagit aux changements détectés en activant un mécanisme d'adaptation.



(a) DDM : le point de départ de la fenêtre est fixe, tandis que le point final de la fenêtre est étendu après la réception d'un nouvel exemple.



(b) STEPD : la fenêtre temporelle globale est comparée avec la fenêtre de temps la plus récente.



(c) ADWIN : toutes les coupes possibles de la fenêtre sont examinées et les tailles optimales des sous-fenêtres sont calculées en fonction de la différence maximale.

FIGURE 2.4 – Différentes stratégies sur les fenêtres passée et présente.

Les détecteurs sont généralement basés sur une analyse des statistiques sur deux périodes du flux, une passée et une présente, chaque période étant définie par une fenêtre d'une certaine taille [63]. La détection de dérive est réalisée principalement par deux

stratégies, l'une basée sur la performance du modèle et l'autre sur la distribution des exemples. Les approches basées sur la performance considèrent qu'une dérive se produit lorsque la performance du modèle diminue significativement. Les méthodes les plus réputées, comme DDM [79], STEPD [80], ADWIN [81], se distinguent les unes des autres essentiellement par les mesures choisies et la gestion des fenêtres de mesure (Figure 2.4). Les approches basées sur la distribution des exemples déterminent la présence d'une dérive en quantifiant directement la sévérité de la dérive. Elles utilisent couramment deux fenêtres glissantes avec une taille préfixée, une passée et l'autre présente. Ces approches se distinguent essentiellement par les différentes mesures utilisées, les principales étant la divergence de Kullback-Leibler [82], et l'entropie [83]. Quelle que soit la méthode de détection utilisée, dès qu'il y a une différence significative dans l'information statistique entre les deux fenêtres, une alerte est déclenchée.

Les modèles peuvent être classés en deux catégories : ceux qui sont basés sur une mise à jour globale lorsqu'un changement est détecté, et ceux qui suppriment/modifient/ajoutent une composante du modèle. La stratégie dépend du type de modèle utilisé. Les arbres de Hoeffding et l'ELM doivent être entièrement reconstruits en cas de changement de concepts alors que les systèmes à base de règles peuvent s'adapter par remplacement d'une règle obsolète par une nouvelle.

La qualité des résultats des approches actives dépend étroitement de la précision du détecteur utilisé, qui dépend elle-même de la taille de sa fenêtre d'observation des changements. En pratique, ce choix reste un compromis entre une adaptation rapide (petite fenêtre) et une bonne robustesse pour éviter les fausses détections (grande fenêtre). La taille de la fenêtre est un problème majeur pour l'efficacité du détecteur. Les expériences montrent que les approches actives sont capables de détecter rapidement les dérives abruptes, mais qu'elles ont du mal à gérer les dérives incrémentales, qui peuvent ne pas être suffisamment statistiquement significatifves pour être identifiées sur un temps court.

2.4.2 Approches passives

Les approches passives adaptent continuellement leurs modèles sans explicitation du changement ; cela évite les effets de seuils des détecteurs. Elles peuvent se classer en quatre catégories : la fenêtre glissante, la stratégie d'atténuation, l'apprentissage continu de nouvelles distributions, et la combinaison de différents mémoires.

La fenêtre glissante est l'une des techniques les plus simples pour gérer les changements de concept. Elle joue le rôle d'une mémoire qui ne garde que les exemples les plus

récents et oublie tous les précédents. Mais elle présente le même inconvénient évident de dépendre, comme le détecteur, de la taille préfixée de la fenêtre. Il faudrait connaître les caractéristiques du concept pour pouvoir fixer celle de la fenêtre. Une fenêtre trop grande peut induire une réaction lente à la dérive abrupte, et une fenêtre trop petite peut impacter la qualité du modèle.

La stratégie d'atténuation pondère les informations mémorisées en fonction de leur ancienneté ; elle permet d'oublier les informations passées à une certaine vitesse constante. Différentes fonctions de pondération peuvent être utilisées - notamment linéaire, exponentielle et logarithmique. Cependant, cette stratégie souffre de retards coûteux en cas de dérive abrupte.

L'apprentissage continu de nouvelles distributions est basé sur la combinaison d'un ajout de nouveaux modèles et d'un retrait des modèles anciens les moins performants. Cette stratégie diffère de l'approche active car elle ajoute constamment de nouvelles informations, qu'il y ait ou non une dérive. Elle permet une adaptation de l'apprentissage dans des flux de données qui comportent plusieurs dérives de concept. Mais sa complexité élevée en temps de calcul est un inconvénient majeur.

La combinaison d'une mémoire à court terme et d'une mémoire à long terme vise à pallier les limites des approches précédentes : l'adaptabilité rapide de la mémoire à court terme permet au modèle de résister à des changements soudains, et la stabilité de la mémoire à long terme permet au modèle de résister aussi bien aux changements graduels qu'aux concepts récurrents. Dans *Self Adjusting Memory* (SAM) [69], la mémoire à court terme est une fenêtre adaptative et la mémoire à long terme basée sur un ensemble de clusters qui est déjà présenté ci-dessus 2.2. Comme ADWIN, la mémoire à court terme est évaluée en comparant les moyennes des erreurs de différentes sous-fenêtres, mais SAM utilise une autre stratégie pour faire évoluer la fenêtre. SAM prédétermine deux paramètres : la taille maximale de la fenêtre m_{max}, et sa taille minimum m_{min}. Supposons que la taille de la fenêtre à l'instant t soit $m \leq m_{max}$, SAM compare alors respectivement les sous-fenêtres de la taille, m, $m/2$, $m/4$ ..., m_{min} avec un nombre logarithmique et il ne garde que la sous-fenêtre avec la moyenne des erreurs la plus faible. En particulier, les exemples supprimés de la mémoire à court terme ne sont pas complètement perdus ; ils sont insérés dans la mémoire à long terme pour éviter les concepts récurrents. Les approches basées sur la combinaison de mémoires ont des performances très prometteuses qui confirment leur capacité à s'adapter à une variété de types de dérives différentes.

2.5 Protocoles expérimentaux d'évaluation

Contrairement aux algorithmes d'apprentissage traditionnel où des jeux de données sont partagés par toute la communauté avec des mesures communes, la prise en compte des variations des distributions dans le temps rend l'évaluation des modèles beaucoup plus complexe. Nous présentons ici deux protocoles d'évaluation les plus souvent utilisés : le protocole *Hold-Out* et le protocole *Interleaved Test-Then-Train*.

2.5.1 Protocole *Hold-Out*

Le protocole *Hold-Out* s'apparente à un protocole expérimental hors-ligne avec un jeu de données séparé en deux sous-ensemble de données, un d'apprentissage et un de test. Mais au lieu d'accéder, comme dans le cas classique, à toutes les données d'entraînement à la fois, les modèles sont entraînés séquentiellement en leur fournissant une donnée après l'autre (Figure 2.5). Pour suivre les performances du modèle en flux, le modèle peut être évalué périodiquement, par exemple après chaque million d'exemples d'entraînement. La précision moyenne sur l'ensemble de test permet d'analyser la capacité de généralisation des algorithmes à des exemples non vus.

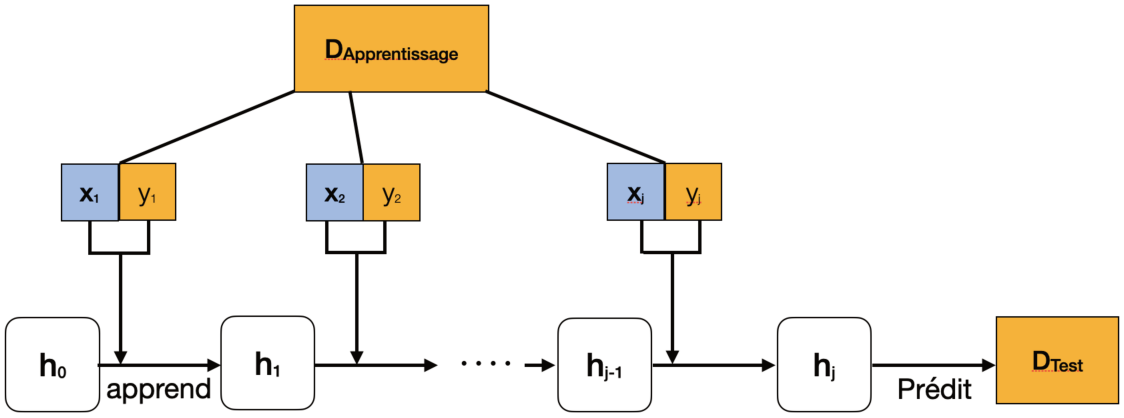


FIGURE 2.5 – Protocole *Hold Out* (extrait de [2])

Le protocole *Hold-Out* peut fournir des mesures précises et non biaisées lorsque les ensembles d'entraînement et de test sont indépendants les uns des autres, ont des distributions identiques et comportent un nombre suffisamment grand d'exemples [2]. Cependant, dans un flux de données non-stationnaire, l'ensemble de test doit toujours représenter exactement la même distribution $\mathbb{P}_t(\mathbf{x}, y)$ à l'instant t lors de l'évaluation d'un algorithme d'apprentissage. En d'autres termes, si la distribution $\mathbb{P}_t(\mathbf{x}, y)$ change, l'ensemble de test doit être mis à jour pour représenter la nouvelle distribution. Cependant, cela est très complexe à mettre en place pour des applications réelles.

2.5.2 Protocole *Interleaved Test-Then-Train*

Dans le protocole Interleaved Test-Then-Train, chaque exemple est utilisé pour tester le modèle avant d'être utilisé pour l'entraînement (Figure 2.6). Plus précisément, à chaque instant t, l'exemple $(\mathbf{x}_t, y_t)$ est traité selon trois étapes dans l'ordre suivant :

1. **Prédiction :** le modèle en vigueur h_{t-1} effectue la prédiction $p_t = h_{t-1}(\mathbf{x}_t)$.
2. **Évaluation :** la performance du modèle est calculée en se basant sur une somme cumulée d'une fonction de perte entre la prédiction et la vrais classe $Acc_t = \frac{1}{t}\sum_{i=1}^{t} 1 - loss(p_t, y_t)$
3. **Apprentissage :** le nouveau modèle h_t est construit à partir du modèle h_{t-1} et du nouvel exemple $(\mathbf{x}_t, y_t)$.

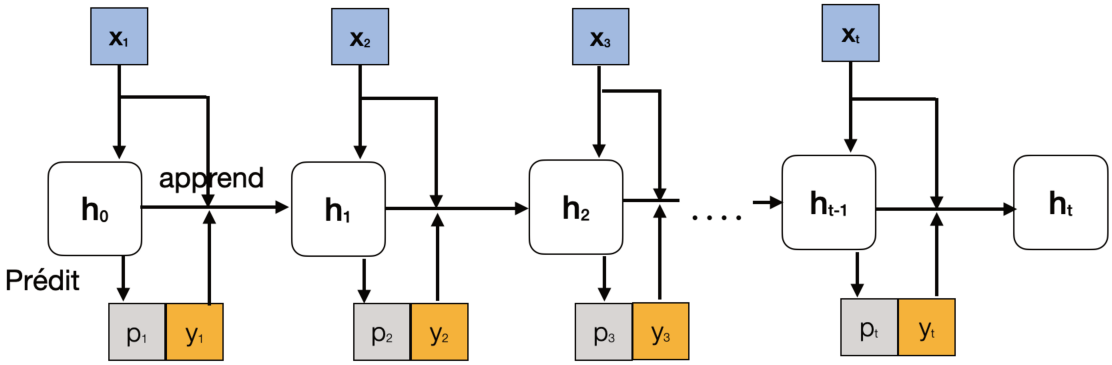
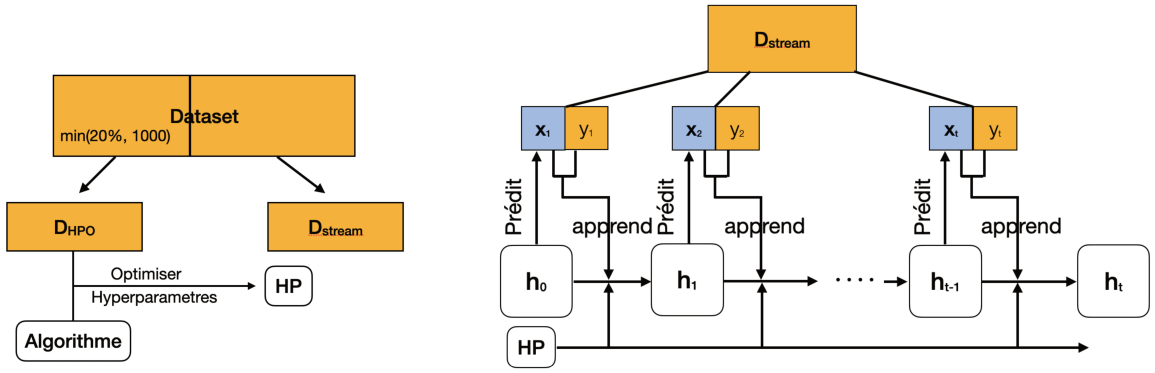


FIGURE 2.6 – Protocole *Interleaved Test-Then-Train* (extrait de [2])

Ce protocole ne nécessite pas la conservation d'un ensemble de tests ; ce qui permet d'utiliser pleinement les données disponibles. Il offre ainsi une grande commodité pour l'expérimentation avec des flux de données non stationnaires en permettant d'évaluer la capacité des algorithmes à s'adapter rapidement aux changements de la distribution des données. En outre, ce protocole permet les prédictions immédiates contrairement au protocole *hold-out* qui nécessite une collecte de nombreuses données et un entraînement avant de pouvoir faire des prédictions.

Cependant, le problème du protocole *Interleaved Test-Then-Train* est qu'il peut être mis en défaut par certains flux non-stationnaires. Par exemple, lorsque des données ayant la même classe apparaissent dans la même période, un classifieur naïf qui prédit en utilisant la classe de l'exemple précédent peut également être très performant, alors que sa capacité de généralisation est très faible. Losing [2] souligne que le protocole Interleaved Test-Then-Train peut fournir une mesure de la capacité de généralisation du modèle uniquement sur le flux stationnaire où toutes les données sont identiquement et indépendamment distribuées.



(a) 20 % de données ou 1,000 premiers exemples pour optimiser les hyperparamètres

(b) L'évaluation Interleaved Test-Then-Train sur le reste des données

FIGURE 2.7 – Processus d'optimisation des hyperparamètres (extrait de [2])

2.6 Optimisation des hyperparamètres

Tous les algorithmes de classification multi-labels, qu'ils soient adaptés aux données off-line classiques ou aux données en flux, nécessitent la mise au point d'hyperparamètres. Pour l'apprentissage off-line, les meilleures valeurs des hyperparamètres d'un modèle sont généralement obtenues dans l'ensemble d'apprentissage, puis la généralisation du modèle avec ces valeurs d'hyperparamètres est évaluée sur un ensemble de test [84]. Cependant, il est difficile d'émuler cette approche dans un scénario de flux. Une approche naïve consiste à séparer un ensemble initial d'instances des premières instances vues et à effectuer un réglage off-line des hyperparamètres du modèle sur ces instances. Néanmoins, cela suppose que les valeurs des hyperparamètres sélectionnés restent optimales même en cas de dérives de concepts. Le défi consiste alors à concevoir une approche qui incorpore le réglage des hyperparamètres dans le cadre du processus d'apprentissage on-line ; ce qui peut impliquer la généralisation du modèle, la détection des dérives, l'adaptation aux dérives, etc. En pratique, il y a, à notre connaissance, encore très peu de recherches sur l'ajustement des hyperparamètres dans le cas des flux.

Losing et al [2] ont présenté pour la première fois une procédure expérimentale complète pour l'optimisation des hyperparamètres pour l'apprentissage incrémental sur un flux stationnaire. Les auteurs considèrent des jeux de données classiquement utilisés pour l'apprentissage off-line et mélangent l'ordre des données aléatoirement pour simuler un flux stationnaire. Et uniquement 20 % des données ou 1,000 exemples sont utilisés pour optimiser les hyperparamètres ; ce qui est plus proche des volumétries des cas pratiques. Une fois que les valeurs de hyperparamètres ont été déterminées, elles sont fixées pour tous les modèles au fil du temps. Le reste des données est utilisé pour comparer les performances des différents algorithmes avec le protocole Interleaved Test-Then-Train (comme illustré dans la figure 2.7).

Cette méthode fonctionne bien sur les flux stationnaires. Cependant, à notre connaissance, il manque encore une procédure expérimentale complète pour optimiser les hyperparamètres sur les flux non-stationnaires.

3 Conclusion

Dans ce chapitre, nous avons rappelé les principes de la classification multi-labels classique sur des données complètes et la classification mono-label en flux.

Pour la classification multi-labels, nous avons d'abord précisé les caractéristiques des données multi-labels, puis nous avons passé en revue les algorithmes principaux et les critères d'évaluation de mesure. La tendance générale est que les algorithmes qui tiennent compte de la corrélation entre les labels ont souvent une meilleure performance en prédiction. Cependant, l'espace de labels en grande dimension pose un grand défi à la complexité de calcul du modèle.

Pour la classification mono-label en flux, nous avons complété l'état de l'art en précisant les caractéristiques importantes des flux non stationnaires qui sont souvent peu identifiées dans la littérature. Cette analyse permet de mieux saisir les différences entre la classification « off–line » et la classification « on-line » et montre que la nécessité d'adaptation des modèles aux variabilités des distributions change le problème en profondeur. Nos lectures nous ont également montré que la grande majorité des modèles ne prenait pas ou peu en compte les contraintes de complexité en mémoire et en temps. Les performances eu égard aux mesures d'évaluation classique prévalent. Enfin, nous avons complété le chapitre par une présentation des protocoles expérimentaux utilisés pour l'apprentissage on-line et pour l'optimisation des hyperparamètres du modèle. Le chapitre qui suit prolonge l'analyse de l'état de l'art à la classification multi-labels en flux.

Chapitre 3

Classification Multi-labels en Flux

Sommaire

1 Introduction

L'extension des problèmes précédents aux données multi-labels en flux combine deux tâches difficiles : la classification multi-labels et l'apprentissage en flux. L'algorithme doit prendre en compte non seulement les caractéristiques des données multi-labels (espace de recherche en grande dimension intégrant des corrélations « cachées » entre labels), mais aussi des caractéristiques du flux (contraintes temporelles de l'apprentissage, non-stationnarité des distributions des données). De plus, l'algorithme doit avoir une faible complexité en raison des limitations des ressources de calcul et de stockage.

Ces nouveaux besoins stimulent les recherches en classification multi-labels en flux (MLSC - Multi-Labels Stream Classification) comme en témoigne un état de l'art récent [36] que nous complétons dans ce chapitre. Dans cette revue de la littérature, nous avons constaté que la majorité des algorithmes se basent sur une adaptation des approches de classification mono-label en flux en négligeant la prise en compte des spécificités des données multi-labels. Cette stratégie limite l'amélioration des performances. Pour y pallier et tenter de mieux prendre en compte les problèmes liés à la dérive de concepts, plusieurs approches ensembles ont été proposées ces dernières années. Mais elles se heurtent à la

complexité des calculs.

L'amélioration des performances des algorithmes qui souvent combinent plusieurs mécanismes (ex : clustering, projections, etc) nécessite une compréhension en profondeur de leurs comportements. Or, la comparaison des algorithmes d'apprentissage en flux reste une tâche difficile et pose des questions encore largement ouvertes. En effet, les expérimentations de l'état de l'art ne s'appuient pas toutes sur les mêmes jeux de données ni sur les mêmes hypothèses concernant leurs distributions statistiques [76]. La grande majorité des travaux référencés considère des flux stationnaires ou des flux non-stationnaires variés générés à partir de données multi-labels réelles selon leur ordre d'apparition, et les caractéristiques des dérives ne sont généralement pas identifiées.

Ce chapitre ne se limite donc pas à une revue classique de la littérature qui liste, en les organisant, les grandes familles les approches développées : comme nous l'avons fait dans le cas mono-label, nous tentons de saisir plus précisément les comportements des algorithmes vis-à-vis des propriétés des flux. Après une description du problème de classification multi-labels en flux, nous distinguons les approches centrées sur l'apprentissage de flux stationnaires de celles qui tentent d'intégrer la non stationnarité des distributions et les dérives de concepts associées. Nous complétons le chapitre par une présentation des générateurs des flux de données non stationnaires utilisés dans la littérature en discutant leurs limites.

2 Définition et les caractéristiques du problème

Dans la suite, nous considérons un flux de données multi-labels $\mathcal{D}_m = \{(\mathbf{x}_1, \mathbf{y}_1), ..., (\mathbf{x}_t, \mathbf{y}_t), ...\}$, où chaque exemple $(\mathbf{x}_t, \mathbf{y}_t) \in \mathcal{X} \times \mathcal{Y}$ à l'instant t est généré selon une distribution de probabilité jointe $\mathbb{P}_t(\mathbf{x}, \mathbf{y})$ inconnue. Dans le problème classique de la classification multi-labels, l'algorithme apprend une fonction statistique h sur l'ensemble d'apprentissage et peut prédire les labels d'exemples non vus dans l'ensemble de test. Dans son extension aux données en flux, les caractéristiques de l'algorithme recherché dépendent du protocole d'apprentissage préalablement défini.

Comme dans le cas mono-label en flux présenté précédemment, le protocole le plus utilisé reste aussi ici le protocole Interleaved Test-Train où trois étapes sont appliquées séquentiellement à l'instant t : (i) la prédiction $\mathbf{p}_t \in \mathcal{Y}$ du vecteur des labels associé au vecteur d'attributs est déterminée par le modèle précédent : $\mathbf{p}_t = h_{t-1}(\mathbf{x}_t)$; (ii) le modèle est évalué après la découverte du vrai vecteur de labels en comparant $\mathbf{y}_t$ et $\mathbf{p}_t$ et (iii) l'algorithme appliqué A calcule un nouveau modèle h_t à partir du modèle précédent h_{t-1} et du nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$: $h_t = A(h_{t-1}, (\mathbf{x}_t, \mathbf{y}_t))$. L'algorithme A doit construire un modèle à chaque instant qui est prêt à prédire pour le prochain exemple ; chaque modèle doit donc être capable de généraliser les informations dans les données actuelles lorsque $\mathbb{P}_t(\mathbf{x}, \mathbf{y})$ est stable, et doit s'adapter «rapidement » pour apprendre de nouvelles

informations si $\mathbb{P}_t(\mathbf{x}, \mathbf{y})$ change.

2.1 Les caractéristiques du problème

Deux caractéristiques majeures du problème de classification multi-labels restent valides pour les données en flux : la volumétrie de l'espace des labels et la présence de corrélations entre labels. Nous les rappelons ci-dessous en les instanciant dans le cas des données en flux.

Espace de sortie à grande dimension. Un flux de données multi-labels peut avoir 2^l combinaisons de labels possibles. Cette dimensionnalité induit deux problèmes majeurs :

- la *malédiction de la dimension* [85]. Plus le nombre de labels augmente, plus le volume de l'espace croît et des données peuvent se retrouver « isolées » dans l'espace de recherche. Cela est problématique pour les méthodes qui nécessitent un nombre significatif de données pour l'apprentissage. Et, dans le cadre des données en flux, il peut être plus difficile de généraliser les informations, de détecter les dérives et de s'y s'adapter.
- la *limitation en temps de calcul*. Le flux de données est supposé infini et les temps de réponse des algorithmes sont limités. La complexité de calcul d'un modèle d'apprentissage étant souvent dépendante des tailles des espaces d'attributs et de labels et du nombre d'exemples considérés, il est nécessaire de trouver un compromis sur les ressources nécessaires en calcul et en stockage.

Corrélations entre labels. Dans le cas stationnaire, il est bien connu que la prise en compte des corrélations entre labels améliore les performances d'un modèles [47]. Dans le cas non stationnaire, la prise en compte de cette corrélation est également nécessaire. A titre d'exemple, considérons une situation très simplifiée avec quatre labels λ_1, λ_2, λ_3 et λ_4. Supposons que la fréquence des labels individuels reste constante tout au long du flux et que les combinaisons λ_1, λ_2 et λ_3, λ_4 soient très fréquentes alors que toutes les autres combinaisons soient rares. Si le modèle prédit la présence du label λ_1, il est très probable que le label λ_2 soit également prédit. Considérons alors un scénario dans lequel les paires de labels λ_1, λ_2 et λ_3, λ_4 n'apparaissent plus ensemble et où seules les paires λ_1, λ_3 et λ_2, λ_4 apparaissent désormais. Si le modèle ne s'adapte pas, ses performances se dégradent alors rapidement.

3 Classification multi-labels sur flux stationnaires

L'intérêt croissant ces dernières années pour la MLSC a conduit à l'exploration de paradigmes différents qui s'appuient sur les trois familles issues des états de l'art initiaux

de la classification multi-labels présentées précédemment. Le tableau 3.1 synthétise les propositions de l'état de l'art.

Transformation	**Adaptation**	**Ensemble**
AMW [86]	HTps [87]	EaHTps [87]
MINAS-PS [88]	ISOUPTree [89]	EaISOUPTree [89]
HTcl [90]	ML-AMRules [91]	ML-Random Rules [91]
-	OSML-ELM [92]	GOOWE-ML [93]
-	MLSAMPKNN [94]	MLSAMkNN [95]
-	MLSAkNN [96]	AESAKNNS [97]
-	ML-Bayésien [98]	-

TABLEAU 3.1 – Classification des algorithmes MLSC.

3.1 Les approches par transformation

Les stratégies de transformation les plus couramment utilisées pour la MLSC sont *Binary Relevance* et *Pruned Sets* qui présentent différents avantages et inconvénients : la première présente une faible complexité de calcul mais ignore complètement la corrélation entre les labels, tandis que la seconde tient compte de toutes les corrélations entre les labels mais présente une complexité de calcul élevée.

3.1.1 *Binary Relevance*

Basés sur la stratégie classique BR, les algorithmes transforment le problème multi-label en l problèmes binaires et traitent chaque problème en flux indépendamment. Dans la section 1.2.1, nous avons déjà identifié l'impact des déséquilibres où des exemples négatifs peuvent dominer. Afin de tenter de résoudre ce problème, l'algorithme AMW (A Multiple Windows Approach) [86] introduit deux fenêtres glissantes de taille fixe par label, une pour les exemples positifs et une pour les négatifs. Lorsqu'un nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$ arrive, pour chaque label λ_i, $i \in [1, l]$, le vecteur d'attributs $\mathbf{x}_t$ est ajouté dans la fenêtre des exemples positifs si $y_t^i = 1$, ou dans la fenêtre des exemples négatifs sinon. Lorsqu'une fenêtre est remplie, le vecteur d'attributs le plus ancien est oublié afin de libérer une place pour un nouveau. Le classifieur kNN est utilisé pour la prédiction : si les voisins les plus proches de $\mathbf{x}_t$ sont plus fréquents dans la fenêtre des exemples positifs alors la valeur prédite de ce label est 1, ou 0 sinon. L'algorithme Multi-Target Régression consiste à convertir la tâche de classification binaire en une tâche de régression pour chaque label [89]. Et la régression est mise à jour à chaque instant par une descente de gradient stochastique avec un taux d'apprentissage prédéterminé.

3.1.2 *Pruned Sets*

La stratégie PS considère chaque vecteur de labels comme une classe distincte et seuls les vecteurs de labels les plus fréquents sont pris en compte. PS a été utilisée pour la première fois pour résoudre le problème MLSC dans HTps [87], mais dans cette publication l'hypothèse considérée est très simple : $1,000$ premiers exemples du flux peuvent représenter la distribution complète des données dans un flux stationnaire. Ainsi, les vecteurs de labels qui apparaissent plus d'une fois dans les $1,000$ premiers exemples du flux sont considérés comme les vecteurs de labels les plus fréquents et ne sont plus modifiés au cours du temps. Cependant, cette hypothèse n'est pas confirmée par des preuves. Les auteurs de HTcl [90] soutiennent que les vecteurs de labels fréquents devraient être pris en compte en temps réel et proposent donc d'identifier dynamiquement des nouveaux vecteurs de labels fréquents en enregistrant leurs nombre d'occurrences ; lorsque le nombre d'occurrences est plus grand qu'un seuil prédéterminé, le vecteur de labels est considéré comme fréquent. Cependant, la complexité de calcul avec PS reste généralement élevée.

3.2 Les approches par adaptation

Différentes stratégies bien connues en apprentissage mono-label en flux ont été adaptées aux données multi-labels.

- **L'arbre d'Hoeffding** est adapté en modifiant les gains d'informations avec l'entropie multi-labels. Il peut utiliser tout classifieur multi-labels dans les feuilles, tels que ceux basés sur le vote majoritaire intégrant la stratégie PS (HTps [87] et HTcl [90]) ou ceux basés sur la Multi-Target régression (ISOUPTree [89]).
- **L'OSML-ELM**, qui est un réseau de neurones avec une seule couche cachée, a été simplement adapté en prédéfinissant un seuil μ [92]. Toutes les valeurs de sorties o^i, $i = 1$ à l, sont comparées avec le seuil μ, et la prédiction p^i est égale à 1 si $o^i \geq \mu$, sinon à 0.
- **Le ML-AMRules**, qui est un système à base de règles de décision, s'appuie sur le vote majoritaire multi-labels pour la prédiction [91]. Pour chaque règle, la prédiction est calculée à partir des labels qui sont présents dans la majorité des derniers exemples couverts.
- **L'approche Naive-Bayes** est adaptée en calculant la probabilité conditionnelle à la fois entre les labels et les attributs et entre les labels eux-mêmes en enregistrant leurs fréquences d'occurrences (ML-Bayésien [98]). Le premier label prédit $\lambda^{(1)} \in \mathcal{L}$ est celui qui a la plus grand probabilité conditionnelle $\mathbb{P}(\lambda_i|\mathbf{x})$ avec $i \in [1, l]$. Soit $\hat{\mathcal{L}} = \{\lambda^{(1)}, .., \lambda^{(k)}\} \subseteq \mathcal{L}$ l'ensemble des labels prédits pour le nouveau vecteur d'attributs $\mathbf{x}_t$ après k étapes ; le $k+1$ ième label est sélectionné en calculant la probabilité maximale $\mathbb{P}(\lambda_i|\mathbf{x}) \cdot \prod_{j=1}^{k} \mathbb{P}(\lambda^{(j)}|\lambda_i)$ avec $i \in [1, l]$ et $\lambda_i \notin \hat{\mathcal{L}}$. Le nombre de labels prédits est déterminé par la moyenne du nombre de labels associés aux

exemples rencontrés par le modèle en cours.

— **Différentes stratégies basées sur des mémoires** ont été adaptées pour la MLSC en intégrant un kNN multi-label et un vote majoritaire pour la prédiction. Le récent algorithme MLSAMPkNN (Multi-label Punitive kNN with Self-Adjusting Memory) combine une fenêtre adaptive qui peut ajuster automatiquement sa taille pour mémoriser les exemples [94]. Une variation de cette approche, MLSAkNN (Self-Adjusting Multi-Label kNN), auto-ajuste la valeur du paramètre k de kNN pour chaque label [96].

Parmi ces cinq stratégies, seul l'algorithme Naive-Bayes pour la MLSC tient compte explicitement des corrélations entre les paires de labels. Cependant, l'algorithme multi-labels Naive-Bayes est basé sur une hypothèse d'indépendance conditionnelle des attributs qui n'est pas toujours vérifiée et sa complexité de calcul est élevée pour être applicable dans un espace de grande dimension. En supposant l'indépendance mutuelle entre les labels, les quatre autres stratégies n'ont pas besoin de prendre en compte la haute dimensionnalité de l'espace des labels et la corrélation entre les labels ; ce qui leur permet de maintenir leur faible complexité de calcul dans un flux. Cependant, leurs performances en prédiction peuvent être dégradées car elles ignorent une partie précieuse des informations sur les données multi-labels.

3.3 Les approches ensembles

Deux techniques d'ensemble principales sont utilisées dans la classification multi-label en flux : 1) la stratégie de "*Online bagging*" [99] où chaque modèle de base est mis à jour indépendamment avec le nouvel exemple k fois à chaque instant, où k est une valeur tiré selon la loi de $poisson(1)$, et un vote majoritaire est utilisé pour la prédiction ; 2) la stratégie de "*weighted majority*" [100] qui pondère à chaque instant les prédictions des classifieurs selon certains critères préfixés.

La stratégie de "*Online bagging*" est la plus utilisée car elle peut s'appliquer à n'importe quel classifieur. Elle a été mise en œuvre dans les approches précitées suivantes : EaHTps avec un ensemble d'arbres de Hoeffding avec Prune Sets [87], EaISOUPTree avec un ensemble d'arbre de Hoeffding avec le Multi-Target Régression [89] et AESAKNNS avec un ensemble de fenêtres adaptatives [97].

La stratégie de "*weighted majority*" a été explorée plus récemment pour tenter d'améliorer les performances prédictives de l'agrégation des modèles. GOOWE-ML (Geometrically Optimum Online Weighted Ensemble for multi-label) [93] combine plusieurs ISOUPTrees et optimise les poids des arbres en minimisant la distance euclidienne entre

les vecteurs prédits de chaque classifieur et le vrai vecteur de labels. MLSAMkNN [95] adapte SAM pour MLSC en considérant chaque vecteur de labels comme une classe distincte et en optimisant les poids de chaque mémoire en fonction de leurs erreurs de prédictions. ML-Random Rules [91] est un ensemble de systèmes à base de règles qui combine la stratégie de "*Online bagging*" pour entraîner chaque modèle de base et une stratégie de "*weighted majority*" pour optimiser les poids des modèles en fonction de leurs erreurs de prédictions afin d'obtenir la meilleur performance.

Dans le contexte de l'apprentissage off-line, un ensemble de modèles peut être classiquement composé d'une centaine de modèles de base ou plus. Toutefois, dans le contexte de l'apprentissage en flux, les exigences supplémentaires liées à la combinaison des modèles doivent être soigneusement prises en compte. Par exemple, pour un ensemble de cent modèles le temps de calcul sera démultiplié et les contraintes mémoires imposent des modèles plus petits. L'équilibre entre les performances du modèle et la complexité de calcul est donc le principal défi de la combinatoire des modèles dans le contexte des flux de données.

4 Classification multi-labels sur flux non stationnaires

Bien que le problème de la dérive des concepts ait été largement traité en classification mono-label, sa prise en compte pour la MLSC est beaucoup plus récente. Selon notre étude de la littérature, les stratégies actuellement proposées pour la MLSC sont des modifications assez directes des stratégies pour le flux mono-label. Elles peuvent donc également être présentées en deux familles : les approches actives et les approches passives (voir Tableau 3.2).

4.1 Les approches actives

La façon courante de prendre en compte la dérive de concept consiste à la détecter puis à déclencher une réaction. Les approches actives utilisent donc un détecteur pour déterminer explicitement l'instant du changement t, et puis modifient ou reconstruisent le modèle selon le signal du détecteur.

Les détecteurs utilisés pour la MLSC proviennent de stratégies développées précédemment en classification mono-label. Le détecteur le plus couramment utilisé est ADWIN qui combine différentes mesures de tests statistiques basées notamment sur la performance du modèle ("Exact Match") et sur la distribution des données avec l'entropie multi-labels [90] et la divergence de Kullback-Leibler [101].

Algorithme	Approches	Types de dérives				CL	Complexité de calcule
		A	B	C	D		
AMW [86]	Passive	√					faible
ML-AMRules [91]	Active	√					faible
OSML-ELM [92]	Passive		√	√			faible
ML-Bayésien [98]	Passive		√	√		√	grande
MLSAMPKNN [94]	Passive	√	√	√			faible
MLSAkNN [96]	Passive	√	√	√			faible
EaHTps [87]	Active	√					grande
EaISOUPTree [89]	Active	√					grande
ML-Random Rules [91]	Active	√					grande
MLSAMkNN [95]	Passive	√	√	√	√	√	grande
GOOWE-ML [93]	Passive		√	√			grande
AESAKNNS [97]	Active	√	√	√			grande

TABLEAU 3.2 – Taxonomie des algorithmes de l'état de l'art en fonction des trois critères suivants : 1) la capacité à s'adapter à la dérive, 2) la prise en compte des corrélations entre les labels et 3) la complexité de calcul.
* **A** : Abrupte ; **B** : Graduelle ; **C** : Incrémentale ; **D** : Récurrents ; **CL** : Corrélations entre les labels

Les modèles peuvent être classés en deux catégories : ceux basés sur la mise à jour d'une composante du modèle lorsqu'un changement est détecté, et ceux qui suppriment / modifient / ajoutent un modèle parmi un ensemble de modèles. Dans la première catégorie, ML-AMRules [91] surveille la performance de chaque règle de décision en temps réel : si la performance d'une règle est significativement dégradée, cette règle est supprimée et une nouvelle est ajoutée. La deuxième catégorie regroupe les approches ensembles EaHTps [87], EaISOUPTree [89], ML-Random Rules [91] et AESAKNNS [97]. Les opérations de mise à jour du modèle sont déclenchées en fonction de sa performance ou avec une alarme émise par un détecteur.

La qualité des résultats des approches actives dépend étroitement de la précision du détecteur utilisé. Un « bon » détecteur doit éviter de détecter la dérive de manière incorrecte et de supprimer des informations utiles. Il ne doit pas non plus ignorer une dérive et entraîner des retards dans l'adaptation du modèle. La qualité de détecteur dépend notamment de deux facteurs :

— **La taille de la fenêtre.** La fenêtre devrait être grande sur le flux de données multi-labels pour éviter la détection de faux changements car l'espace de sortie en MLSC peut être très grand. Or une grande fenêtre limite la vitesse de réaction aux changements. Le défi concernant l'amélioration de la précision et de la rapidité pour les détecteurs reste donc très difficile en MLSC. Les détecteurs actuels utilisent simplement la même taille de fenêtre que ceux implémentés pour la classification mono-label en flux, ce qui n'est pas nécessairement bien adapté aux combinaisons

plus complexes du MLSC.

— **La mesure choisie.** Parmi celles basées sur la performance du modèle, nous avons déjà expliqué dans la section 1.2.1 que "Exact Match" est un critère d'évaluation qui ignore les labels qui ont été partiellement prédits correctement. Le détecteur basé sur l'évolution de l'erreur de classification suppose que le modèle ait déjà convergé. En sus, les changements dans la distribution de l'erreur peuvent simplement être causés par l'adaptation continue du modèle, plutôt que par un changement de la distribution des données. En pratique, cette hypothèse est souvent violée. Pour les mesures qui sont basées sur la distribution de données, l'entropie multi-labels et la divergence de Kullback-Leibler ne tiennent pas compte des corrélations entre les labels, et elles ne peuvent pas détecter efficacement une dérive de concepts provoquée par les changements de corrélation entre les labels. En outre, ces mesures ne sont applicables qu'aux données à faible dimension, car leur temps de calcul augmente souvent de manière exponentielle avec le nombre de dimensions.

De façon générale, les approches actives manquent d'une définition spécifique de la sévérité de la dérive de concept. Or, dans le cas non stationnaire, il est nécessaire de qualifier la sévérité pour identifier une présence d'une dérive, et adapter la réaction à différentes graduations de la sévérité. Si cette dernière est faible et que l'algorithme réagit sans discernement il y a un risque de perdre des informations utiles lors de la mise à jour du modèle entraînant une suppression de données.

4.2 Les approches passives

Dans la littérature, deux combinaisons de mécanismes sont utilisées selon les approches : la stratégie de la fenêtre et la stratégie d'atténuation pour les méthodes de transformation et d'adaptation d'une part, et l'ajout continu de nouveaux modèles et la combinaison de différentes mémoires pour les approches ensembles d'autre part.

— **La stratégie de la fenêtre :** AMW utilise une petite fenêtre glissante pour représenter la distribution actuelle. Cela permet une adaptation rapide à la dérive, mais les performances du modèle peuvent être détériorées pendant les périodes de stabilité. MLSAMPkNN [94] et MLSAkNN [96] utilisent une fenêtre de taille adaptative qui est grande pendant les périodes de stabilisation et devient petite pendant les périodes de dérive.

— **La stratégie d'atténuation :** L'algorithme ML-Bayesien [98] basé sur un Naive-Bayes et OSML-ELM [92] basé sur un réseau de neurones réduisent l'impact de l'ancienne distribution sur les informations qu'ils enregistrent avec un taux d'apprentissage constant qui doit être préfixé.

— **L'apprentissage continu de nouvelles distributions :** GOOWE-ML [93] ajoute

un nouveau classifieur après la collecte d'un nombre fixé d'exemples, puis met à jour les poids de chaque classifieur en fonction de ses prédictions et supprime le classifieur ayant le poids le plus faible afin de maintenir sa pertinence par rapport à la nouvelle distribution.

— **La combinaison de différentes mémoires :** MLSAMkNN [95] utilise une mémoire à court terme pour s'adapter à un changement abrupt, et une mémoire à long terme pour s'adapter aux autres types de changements.

Les méthodes passives ont fait l'objet d'une attention accrue ces dernières années car elles sont souvent simples, faciles à utiliser et permettent de limiter les fausses détections ou les omissions. Mais être capable de s'adapter à différents types de dérives et de maintenir une faible complexité reste un défi ouvert. A notre connaissance, seul MLSAMkNN peut traiter les différents types de dérive de concepts que nous avons présentés, mais la complexité de sa mémoire à long-terme est très élevée. En effet, rappelons que MLSAMkNN considère chaque vecteur de labels comme une classe distincte ; ce qui entraîne un regroupement des exemples en 2^l clusters à chaque mise à jour de la mémoire à long terme.

5 Cadre expérimental

En classification multi-labels classique, les comparaisons expérimentales s'appuient sur des jeux de données partagés par la communauté scientifique. Ces jeux comprennent des données aux caractéristiques volumétriques variables : de quelques milliers d'attributs et de labels [102] à des centaines de milliers en classification multi-label. Plus récemment, en classification multi-labels extrême les données ont dépassé le million [43]. Le protocole d'évaluation est bien connu en apprentissage : comparer les performances d'un sous-ensemble d'algorithmes sur un échantillon de jeux de données pour un ensemble de mesures d'évaluation sélectionnées [57].

Pour la classification multi-label en flux, le cadre expérimental est aujourd'hui encore loin d'être aussi balisé. Il n'existe pas à notre connaissance de jeux de données consensuels partagés par la communauté. La difficulté majeure consiste à générer des flux qui contiennent différents types de dérives de concepts identifiés pour analyser avec précision les comportements d'adaptation des différents algorithmes comparés. Nous détaillons dans ce paragraphe les principes des jeux de données les plus utilisés dans la littérature.

5.1 Jeux de données

MOADans les expérimentations de la littérature, la grande majorité (76.5%) des articles référencés dans le tableau 3.1 considère des flux non stationnaires issus de données

réelles. Le tableau 3.3 synthétise les caractéristiques de 15 jeux classiquement utilisés qui sont disponibles sur le dépôt KDIS. Les autres articles utilisent des données synthétiques générées par deux générateurs multi-labels de la plate-forme MOA [103].

5.1.1 Données réelles

Jeux	Domaines	Exemples	Attributs	Labels	Cardinalité	Diversité
20NG	Text	19,300	1,006	20	1.020	0.003
Bibtex	Text	7,395	1,836	159	2.402	0.386
Bookmarks	Text	87,856	2,150	208	2.028	0.213
Cooking	Text	10,490	577	400	2.225	0.436
Corel16k	Image	13,770	500	153	2.859	0.349
Enron	Text	1,702	1,001	53	3.378	0.442
Eukaryote	Biology	7,766	440	22	1.146	0.014
Human	Biology	3,106	9,844	14	1.185	0.027
IMDB	Text	120,900	1,001	28	2.000	0.037
Mediamill	Video	43,907	120	101	4.376	0.149
Reuters-K500	Text	6,000	500	103	1.462	0.135
Scene	Image	2,407	294	6	1.074	0.234
Slashdot	Text	3,782	1,079	22	1.181	0.041
TMC2007	Text	28,600	500	22	2.22	0.041
Yeast	Biology	2,417	103	14	4.237	0.082

TABLEAU 3.3 – Jeux de données réels et leurs caractéristiques. La cardinalité est le nombre moyen de labels de chaque jeu et la diversité est le taux de vecteurs de labels distincts.

Bifet et al. [104] ont noté la difficulté de trouver de grands jeux de données du monde réel pour l'évaluation comparative, en particulier avec des dérives des concepts. A notre connaissance, il n'y a actuellement que le jeu de données Enron, utilisé pour la classification d'e-mails, qui ait été initialement construit à partir d'un ordre chronologique explicite défini par l'ordre de réception des e-mails par les utilisateurs [105]. Il peut être considéré comme un flux de données. Les autres jeux de données sont basés sur des jeux initialement proposés pour la classification traditionnelle qui ont été adaptés pour simuler un flux de données non stationnaire à partir de l'ordre de collecte des données.

Dans tous les cas, les natures des dérives sur ces jeux de données réels ont été très peu étudiées. Ce manque d'informations du flux limite l'interprétation des performances prédictives d'un modèle. Par exemple, certains jeux de données réels peuvent intégrer une dépendance temporelle des labels où des données associées à des vecteurs de labels identiques apparaissent au cours de la même période [76]. Dans ce cas, un classifieur naïf prédisant que le vecteur de labels en $t + 1$ est le même que celui en t peut obtenir de très bonnes performances avec le protocole d'évaluation *Interleaved Test-Then-Train* alors que sa capacité de généralisation est très mauvaise.

5.1.2 Données synthétiques

Pour tenter de mieux maîtriser les propriétés des données en flux, deux générateurs de données multi-labels ont été développés sur la plateforme de tests MOA : le MultiLabel générateur basé sur du clustering [88] et le Meta Multi-label générateur [106].

Le générateur Clustering Multi-label génère k clusters dans un espace d'attributs réduit à deux dimensions en associant chaque cluster à un label. Le caractère multi-label est alors basé sur la présence d'exemples à l'intersection de plusieurs clusters : l'exemple est décrit par l'union des labels associés aux différents clusters auxquels il appartient. Cependant, le processus de génération de données et les codes n'ont pas été publiés ; ce qui entrave son usage pour des flux de données non stationnaires dans un espace de grande taille.

Le générateur Meta Multi-label est à notre connaissance le seul générateur de données multi-labels complet et disponible actuellement ; il est donc le plus souvent utilisé pour générer des flux de données multi-labels synthétiques. La génération d'un flux de données est effectuée en deux étapes successives : la génération d'un flux de données multi-labels stationnaires, puis l'introduction des dérives conceptuelles.

1) Générer des flux de données multi-labels stationnaires : le générateur Meta Multi-label génère des données dont les attributs sont conditionnés à la présence ou non de certaines combinaisons de labels : un attribut est positif si la combinaison de labels qui lui est associée est incluse dans la donnée.

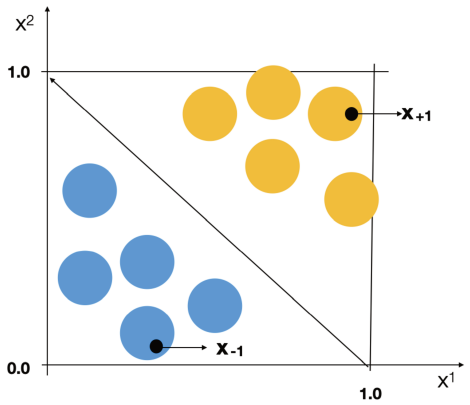


FIGURE 3.1 – Procédure de génération de vecteurs d'attributs binaires en deux dimensions : la distribution des clusters jaunes représente les exemples positifs et celle des clusters bleus représente les exemples négatifs.

Plus précisément, à chaque instant t, le générateur procède en deux étapes pour créer une nouvelle donnée : il génère d'abord le vecteur de labels $\mathbf{y}_t$ basé sur une probabilité a priori prédéterminée par $\mathbb{P}_t(\mathbf{y})$, puis il génère le vecteur d'attributs $\mathbf{x}_t$ correspondant à $\mathbf{y}_t$ selon la probabilité conditionnelle prédéterminée $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$.

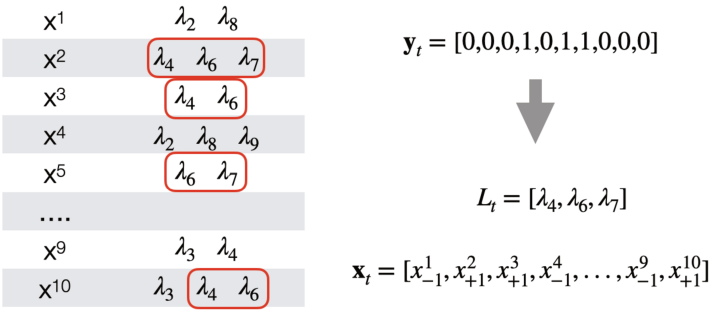


FIGURE 3.2 – Exemple de génération de vecteurs d'attributs multi-labels.

Pour la première étape, le processus de génération des combinaisons de labels $\mathbf{y}_t$ peut être décomposé selon la formule suivante : $\mathbb{P}(\lambda_1) \cdot \prod_{i=2}^{l} \mathbb{P}(\lambda_i|\lambda_{i-1})$. Ainsi, un label λ_i est choisi conditionnellement à la présence des labels précédemment examinés.

L'idée principale du processus de génération des vecteurs d'attributs $\mathbf{x}_t$ -étant donné le vecteur de labels $\mathbf{y}_t$- est de générer de nombreux points aléatoires dans l'espace des attributs et de les diviser en deux groupes, l'un positif et l'autre négatif de sorte que les points générés puissent être séparables dans l'espace des attributs, et que chaque dimension associée à un attribut puisse être un marqueur de la présence d'une certaine combinaison de labels.

Nous désignons par $L_t \subseteq \mathcal{L}$ sa représentation ensembliste où $\forall i \in [1, l], \lambda_i \in L_t$, si $y_t^i = 1$,. La probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$ peut être alors décomposée selon la formule suivante (en faisant l'hypothèse d'indépendance entre les attributs) :

$$P(\mathbf{x}_t|y_t) = \prod_{i=1}^{d} P(x_t^i|\mathbf{y}_t) = \prod_{i=1}^{d} P(x_t^i|L_t) \tag{3.1}$$

Ces probabilités sont prédéterminées par la présence d'une sous combinaison de labels dans la combinaison de référence L_t : chaque attribut x_t^i est associé à une combinaison de labels L_i dont la répartition suit la distribution de probabilité conditionnelle :

$$P(x_t^i|L_t) = P(x_t^i|L_i \subseteq L_t)P(L_i \subseteq L_t) + P(x_t^i|L_i \nsubseteq L_t)P(L_i \nsubseteq L_t) \tag{3.2}$$

$$= P(x_{+1}^i)P(L_i \subseteq L_t) + P(x_{-1}^i)P(L_i \nsubseteq L_t) \tag{3.3}$$

Ainsi, à chaque instant t, le générateur applique d'abord un générateur binaire de MOA pour générer deux vecteurs d'attributs, un pour $\mathbf{x}_{+1}$ et un autre pour $\mathbf{x}_{-1}$ (comme illustré sur la figure 3.1). Puis, il génère la valeur des attributs x_t^i, $i = 1$ à l, selon L_t comme suit : $x_t^i = x_{+1}^i$ si L_t contient la combinaison de labels L_i associés à l'attribut x^i, $L_i \subseteq L_t$; sinon $x_t^i = x_{-1}^i$ (comme illustré sur la figure 3.2).

Cependant eu égard à l'objectif initial du générateur, la propriété de séparabilité dans l'espace des attributs n'implique pas nécessairement la propriété de séparabilité

dans chacun des sous-espaces (comme illustré dans la figure 3.3) :

$$P(x_{+1}) \neq P(x_{-1}) \nRightarrow \forall i \in [1, d], P(x^i_{+1}) \neq P(x^i_{-1})$$

Par conséquent, le générateur peut rendre impossible la détection d'une certaine combinaison de labels si celle-ci est associée à un attribut dont les lois marginales associées $P(x^i_{+1})$ et $P(x^i_{-1})$ ne sont pas suffisamment distinctes.

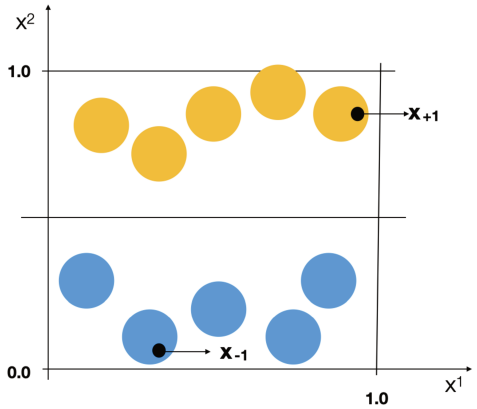


FIGURE 3.3 – Les points sont **séparables** dans l'espace à deux dimensions, mais ne le sont plus dans le sous-espace restreint à la dimension x_1.

2) Introduire des dérives de concepts : le générateur Meta Multi-labels modélise les dérives de concepts selon le modèle proposé par Bifet et al [104]. Plus précisément, à chaque instant t, pour introduire une dérive, le générateur détermine tout d'abord les causes de dérives, puis les types de dérives.

Concernant le premier point, le générateur peut introduire les trois différentes causes de la dérive identifiées précédemment en modifiant simplement la probabilité a priori $\mathbb{P}_t(\mathbf{y})$ et/ou la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$.

Concernant le deuxième point, le générateur simule des flux de données non-stationnaires avec des dérives en combinant plusieurs flux de données stationnaires ayant des concepts différents. Par exemple, considérons deux deux flux stationnaires $\mathcal{D}_{m1}$ et $\mathcal{D}_{m2}$. Il est possible de simuler un flux non-stationnaire $\mathcal{D}_{m3}$ en combinant les flux stationnaires $\mathcal{D}_{m1}$ et $\mathcal{D}_{m2}$ de telle sorte que $\mathcal{D}_{m3} = \mathcal{D}_{m1} \oplus^w_{t_0} \mathcal{D}_{m2}$ où w est la durée de la dérive et t_0 est le centre de changement - suppose que la dérive se produit à l'instant t_e, le centre de changement est donc égale à $t_0 = t_e + w/2$. En particulier, le générateur aléatoire est simulé selon une loi de Bernoulli dont la probabilité évolue avec le temps en suivant une fonction logistique $f(t) = \frac{1}{1-e^{-4(t-t_0)/W}}$. Cette modélisation permet à chaque instant de la dérive de définir la probabilité de tirage d'une instance du flux issus de l'ancien concept $\mathcal{D}_{m1}$ ou du nouveau concept $\mathcal{D}_{m2}$. Nous pouvons observer que la probabilité que les données proviennent du flux $\mathcal{D}_{m2}$ est petite lorsque le temps est inférieur à t_0, et devient plus en plus grande au fil du temps. Ce générateur modélise différents types de dérive en définissant la taille de

w : la dérive est abrupte lorsque la durée de la dérive W est faible ; sinon, "graduelle" selon l'auteur.

Cette modélisation simule un flux avec une dérive basée sur la disparition progressive de l'ancien concept et sur l'apparition progressive du nouveau. Mais lorsque la durée de la dérive w est grande, cette dérive peut être vue comme incrémentale avec un grand nombre de concepts intermédiaires plutôt que graduelle selon les définitions de la littérature. En effet, pendant la période de dérive, le concept à chaque instant peut être considéré comme un concept intermédiaire dont les proportions du premier concept et du dernier sont respectivement $1-f(t)$ et $f(t)$ selon cette modélisation. De plus, ces concepts intermédiaires se rapprochent progressivement du concept suivant à mesure que le temps passe. Ainsi une autre modélisation est nécessaire pour générer la dérive graduelle. En outre, les valeurs des paramètres (i.e, la durée de la dérive w) utilisées dans les différents articles sont soit différentes, soit non spécifiées, ce qui entrave la reproduction des simulations.

5.2 Environnement d'évaluation

Pour l'apprentissage en flux, la plateforme MOA [1] [103], proposée par l'Université de Waikato, est la plus utilisée dans les expérimentations. La plupart des algorithmes en flux et le générateur Meta Multi-labels sont en effet accessibles sur MOA. Cependant, de nombreux algorithmes présentés dans les articles ne sont pas accompagnés d'un code accessible au public ou/et leurs procédures expérimentales, et leurs paramètres d'optimisation ne sont pas toujours précisés. Ces lacunes rendent la reproduction des expériences particulièrement difficile.

6 Conclusion

Dans ce chapitre, nous avons présenté le problème de la classification multi-labels en flux en complétant l'état de l'art par une analyse plus précise des caractéristiques spécifiques de la prise en compte d'un flux. Cette analyse permet de distinguer plus précisément les différences entre les flux stationnaires qui restent les plus utilisés dans les expérimentations et les flux non stationnaires.

Dans le contexte du flux stationnaire, l'équilibre entre les performances du modèle et la complexité de calcul reste le principal défi pour les algorithmes publiés dans la littérature. Dans le contexte du flux non stationnaire, plusieurs questions restent largement ouvertes à la fois pour l'amélioration des stratégies d'apprentissage et pour leur validation. Pour les stratégies actives, les détecteurs de dérive sont très sensibles à la taille de la fenêtre et aux mesures utilisées ; pour les stratégies passives, l'adaptation aux différents

1. MOA : https ://moa.cms.waikato.ac.nz/

types et causes de dérives des concepts méritent également d'être approfondies. Pour les expérimentations, le manque récurrent d'informations sur les natures des dérives du flux limite non seulement l'interprétation de la performance prédictive des modèles, mais rend également très difficile la comparaison des modèles. Nous tentons dans la suite d'aborder ces défis en développant des stratégies d'apprentissage pour les flux stationnaires et non stationnaires et en proposant un cadre expérimental qui prenne en compte explicitement différentes caractéristiques des dérives de concept.

Chapitre 4

Exploration de la mémoire à long terme pour la classification multi-labels en flux stationnaire

Sommaire

1 Introduction

Le flux de données le plus simple est le flux stationnaire, où les exemples sont distribués de manière identique et indépendante. Notre première proposition d'algorithme est validée sur des flux stationnaires comme un grand nombre d'algorithmes de classification multi-labels en flux de la littérature. Rappelons que les deux défis principaux sont ici : 1) l'espace de recherche en grande dimension qui intègre des corrélations « cachées » entre labels et 2) les contraintes temporelles de l'apprentissage qui impose que chaque exemple ne peut être appris qu'une seule fois puisqu'il disparaît après l'arrivée de l'exemple suivant, de sorte que le modèle doit finir les processus de prédiction et d'apprentissage pour chaque exemple avant qu'il disparaisse.

Pour faire face à ce problème, nous nous sommes inspirés de l'algorithme CraftML [37] qui s'est avéré très efficace dans la classification multi-labels traditionnelle en grandes dimensions. Il partitionne hiérarchiquement le problème initial en sous-problèmes à petite

échelle en utilisant une structure arborescente. Cette décomposition arborescente peut réduire la complexité de l'entraînement et des prédictions en découpant l'apprentissage en sous-tâches. Contrairement aux arbres de décision, la stratégie de partitionnement utilisée dans CraftML est basée sur une clusterisation. Plus précisément, CraftML est une structure de clusterisation hiérarchique qui utilise une stratégie descendante, et qui divise les clusters en sous-clusters de cardinaux plus petits à chaque étape. Cependant, la construction d'une structure de clusterisation hiérarchique robuste dans le contexte des données en flux est plus complexe que dans le cadre statique : pour maintenir l'efficacité de l'arbre en conservant une mémoire limitée, différents mécanismes doivent être conçus, tels que la suppression des branches inutiles ou incompatibles avec le nœud actuel et l'équilibrage de l'arbre. Ces mécanismes peuvent augmenter considérablement la complexité de calcul [107]. Ici nous n'avons donc pas repris la structure hiérarchique mais nous avons conservé la stratégie de clusterisation inspirée de CraftML pour partitionner l'espace de données. Nous présentons ci-dessous une nouvelle approche appelée Mémoire à Long-Terme pour la classification Multi-Label en flux (MLT-ML). Elle utilise une structure de clusterisation qui partitionne les données en un nombre prédéterminé de clusters basés sur la similarité entre les centroïdes des clusters [108].

Cette mémoire à long terme présente un très bon compromis entre la qualité des performances prédictives, les ressources de calcul nécessaires et la vitesse d'exécution. Des expériences sur 15 flux de données multi-labels montrent qu'elle est compétitive face aux approches les plus performantes de l'état de l'art actuel tout en représentant une complexité temporelle plus faible.

La suite de ce chapitre détaille les mécanismes de gestion de la mémoire à long terme, puis présente les comparaisons expérimentales avec des algorithmes de l'état de l'art.

2 Mémoire à Long-Terme pour la classification Multi-Labels en flux

Pour tenter d'aborder la grande complexité de l'espace des labels, MLT-ML partitionne l'espace des labels en un nombre restreint de sous-espaces. Ainsi le problème initial est réduit en un ensemble restreint de problèmes à une échelle inférieure et de complexité plus faible. Le partitionnement s'effectue via un clustering qui regroupe dans un même cluster les exemples avec des vecteurs de labels similaires. La similarité entre les vecteurs de labels dans MLT-ML est mesurée avec la distance du cosinus : pour deux vecteurs de labels $\mathbf{y}_i, \mathbf{y}_j \in \mathcal{Y}$, leur distance du cosinus est calculé par :

$$distance(\mathbf{y}_i, \mathbf{y}_j) = 1 - \frac{\mathbf{y}_i \cdot \mathbf{y}_j}{\|\mathbf{y}_i\| \|\mathbf{y}_j\|} \tag{4.1}$$

Plus la distance du cosinus entre deux vecteurs de labels est petite, plus la similarité est grande, et vice versa. La valeur maximale de la distance du cosinus 1 représente donc la non-similarité des deux vecteurs et la valeur minimale 0 représente l'identité des deux vecteurs. Puisque chaque vecteur de labels est représenté par un vecteur binaire, la similarité entre les vecteurs de labels peut s'expliquer par le fait que les combinaisons de labels partagent des mêmes labels entre elles. Lorsque deux labels apparaissent fréquemment dans le même cluster, cela signifie qu'ils sont fortement corrélés à l'intérieur de ce cluster. Avec cette stratégie, nous regroupons les exemples ayant les labels corrélés et séparons ceux ayant des labels non-corrélés.

Nous illustrons ici un exemple de la classification multi-labels avec quatre labels (λ_1, λ_2, λ_3 et λ_4). Nous supposons que les labels λ_1 et λ_2 sont fortement corrélés, alors que les corrélations entre les labels λ_1 et λ_3 sont très faibles. En conséquence, les combinaisons de labels $\{\lambda_1, \lambda_2\}$ sont fréquentes et les combinaisons $\{\lambda_1, \lambda_3\}$ sont très rares ; le vecteur de labels $[1, 1, 0, 0]$ est fréquent là où le vecteur $[1, 0, 1, 0]$ est très rare. Selon la similarité choisie entre les vecteurs de labels, $[1, 1, 0, 0]$, $[1, 0, 0, 0]$ et $[0, 1, 0, 0]$ seront regroupés dans le même cluster -les labels λ_1 et λ_2 sont regroupés dans le même cluster-. Dualement, comme les vecteurs $[1, 0, 0, 0]$ et $[0, 0, 1, 0]$ sont dissimilaires, ils seront donc affectés à différents clusters -les labels λ_1 et λ_3 sont regroupés dans différents clusters-. Avec cette stratégie, MLT-ML divise le problème initial qui met en jeu un grand nombre de labels en des sous-problèmes avec peu de labels en utilisant les corrélations entre les labels.

Plus précisément, dans MLT-ML nous avons adapté la stratégie appelée *Learning Vector Quantization* (LVQ) [38] pour la clusterisation et utilisé la stratégie *Reservoir Sampling* (RS) [68] combinée avec un kNN classificateur dans chaque cluster. Nous décrivons ci-dessous notre modèle en détail.

2.1 La structure du modèle

Pour simplifier les notations, les différentes composantes de l'algorithme sont décrites dans la suite pour un instant quelconque t du flux de données.

La mémoire à long terme est constituée d'un ensemble de s clusters la $M_{LT} = \{C_v | v = 1, ..., s\}$ décrits chacun par un triplet $C_v = (\mathbf{c}_v^x, \mathbf{c}_v^y, S_v)$ composé de deux prototypes et d'un échantillon qui fournit un résumé statistique des distributions des exemples du cluster dans les espaces $\mathcal{X}$ et $\mathcal{Y}$. Le X-prototype $\mathbf{c}_v^x$ (resp. Y-prototype $\mathbf{c}_v^y$) d'un cluster C_v est le centroïde des vecteurs d'attributs (resp. vecteurs de labels) de ses exemples. A chaque instant, les X et Y-prototypes sont mis à jour en fonction de leur similarité avec le nouvel exemple selon la stratégie adaptée de LVQ. L'échantillon $S_v = \{(\mathbf{x}_j, \mathbf{y}_j) \in \mathcal{X} \times \mathcal{Y} | j = 1, \ldots, r\}$ des r exemples représentatifs du cluster C_v est construit par la stratégie *Reservoir Sampling* qui permet de conserver d'une manière équilibrée un résumé des connaissances

acquises au cours du temps dans une mémoire limitée.

Lorsqu'un nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$ arrive à l'instant t, l'algorithme fait une prédiction avant l'apprentissage selon le protocole *Interleaved Test-Train*.

2.2 Prédiction

La prédiction des labels $\mathbf{p}_t$ d'un nouvel exemple $\mathbf{x}_t$ est basée sur une identification du cluster ayant le X-prototype le plus similaire à $\mathbf{x}_t$, suivie d'un calcul de la prédiction $\mathbf{p}_t$ dans l'échantillon de ce cluster avec un classificateur kNN adapté au cadre multi-label (voir figure 4.1). Cette recherche heuristique rapide est basée sur l'hypothèse que les vecteurs d'attributs associés à des vecteurs de labels similaires sont plus susceptibles d'être eux-mêmes similaires et les vecteurs d'attributs appartenant à des clusters différents sont plus susceptibles d'être non-similaires.

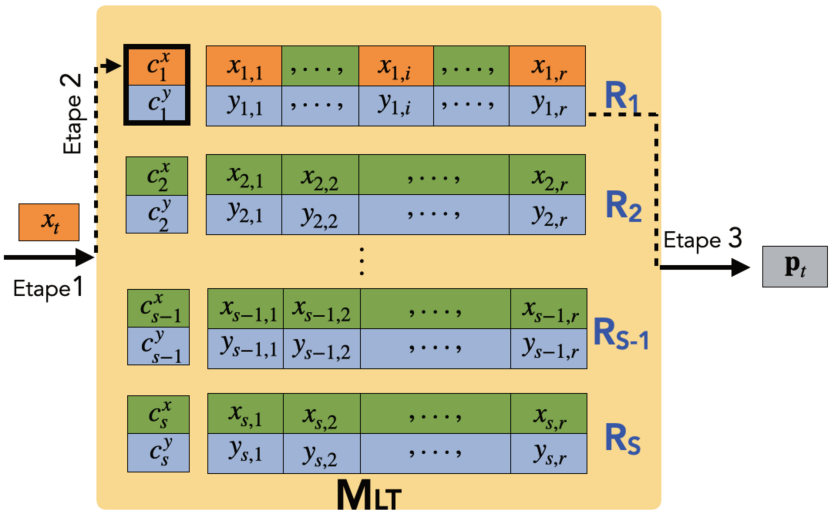


FIGURE 4.1 – Prédiction de MLT-ML à un instant t. Etape 1 : calcul des s distances entre $\mathbf{x}_t$ et chaque X-prototype $\mathbf{c}_i^x$, $i = 1$ à s. Etape 2 : identification du cluster prédit C_p. Etape 3 : calcul de la prédiction $\mathbf{p}_t$ par vote majoritaire avec kNN.

2.2.1 Identification du cluster ayant le X-prototype le plus similaire à $\mathbf{x}_t$

La similarité entre les vecteurs d'attributs dans MLT-ML est également mesurée avec la distance du cosinus qui est souvent une mesure de distance bien adaptée aux espaces en grande dimension, notamment dans le cas des données « sparse » [109]. Ainsi, les s distances du cosinus entre $\mathbf{x}_t$ et les s X-prototypes $\mathbf{c}^x_{v\,v\in[1,s]}$ sont calculées selon la règle suivante : $d_v^x = distance(\mathbf{x}_t, \mathbf{c}_v^x) = 1 - \frac{\mathbf{x}_t \cdot \mathbf{c}_v^x}{\|\mathbf{x}_t\| \|\mathbf{c}_v^x\|}$. L'index du cluster prédit est donc identifié par : $p = argmin(d_v^x, i \in [1, s])$.

Pour éviter que les attributs ayant des valeurs élevées n'écrasent les contributions de celles dont les valeurs sont plus faibles, tous les attributs sont transformés en utilisant

une normalisation par redimensionnement. La valeur de chaque attribut x^i , $i \in [1, d]$, est mise à l'échelle de telle sorte que :

$$x'^i = \frac{x^i - x^i_{min}}{x^i_{max} - x^i_{min}} \tag{4.2}$$

où x^i_{min} est la valeur minimum et x^i_{max} est la valeur maximum de l'attribut x_i. Cependant, avec des données qui arrivent au fil du temps, la plage possible de chaque attribut n'est pas connue à l'avance. Ainsi, pour l'exemple $(\mathbf{x}_t, \mathbf{y}_t)$ arrivant à l'instant t, la normalisation est effectuée sur la base du domaine connu jusqu'à cet instant : x^i_{min} et x^i_{max} qui sont définies comme les valeurs minimum et maximum observées des instants 0 à $t-1$.

2.2.2 Prédiction avec un classificateur kNN sur un échantillon du cluster

Lorsque le cluster C_p est identifié, l'algorithme recherche les k plus proches voisins du vecteur d'attribut $\mathbf{x}_t$ dans l'échantillon S_p de C_p : $N_p = \{(\mathbf{x}_j, \mathbf{y}_j) | j = 1, ..., k\} \subseteq S_p$.

La prédiction finale est ensuite déterminée par un vote majoritaire de ces k plus proches voisins. Pour chaque label $\lambda_i \in \mathcal{L}$, la fréquence f^i de ce label parmi les plus proches voisins est définie par :

$$f^i = \frac{1}{k} \sum_{j=0}^{k} \mathbb{1} | y^i_j = 1 \tag{4.3}$$

où y^i_j est la valeur du i^e label du j^e plus proche voisin, $(\mathbf{x}_j, \mathbf{y}_j) \in N_p$. La valeur de la prédiction finale pour chaque label p^i_t est égale à 1 si $f^i \geq 0.5$, sinon elle est égale à 0.

En raison de l'initialisation de la MLT-ML dans la phase d'apprentissage qui nécessite $1,000$ exemples, la prédiction initiale $\mathbf{p}_t$ est obtenue par un vote majoritaire sur les $min\{k, t-1\}$ plus proches voisins dans les $t-1$ exemples collectés. La première prédiction $\mathbf{p}_0$ est définie par le vecteur nul.

2.3 Apprentissage

Lorsque le vecteur de label $\mathbf{y}_t$ est révélé, MLT-ML affecte ce nouvel exemple au cluster dont le Y-prototype a la plus petite distance du cosinus par rapport à $\mathbf{y}_t$ dans l'espace des labels. Puis, l'exemple est ajouté à l'échantillon du meilleur cluster selon la règle d'échantillonnage du *Reservoir Sampling*, et les prototypes du meilleur cluster et du cluster prédit sont mis à jour avec la stratégie LVQ étendue que nous détaillons dans la suite. La figure 4.2 illustre le processus.

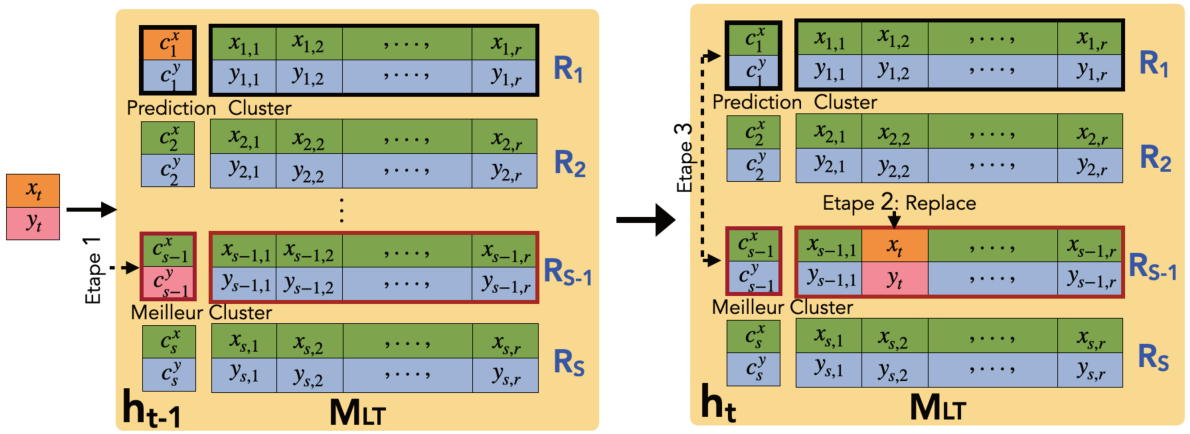


FIGURE 4.2 – L'apprentissage de la MLT avec un nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$. Etape 1 : identification du meilleur cluster C_b. Etape 2 : ajout de $(\mathbf{x}_t, \mathbf{y}_t)$ au réservoir R_b avec une règle d'échantillonnage. Etape 3 : mise à jour des prototypes du meilleur cluster et du cluster prédit.

2.3.1 Mise à jour de l'échantillon

L'insertion du nouvel exemple dans l'échantillon S_b est effectuée par une règle d'échantillonnage de type *Reservoir Sampling* : si la taille de l'échantillon est inférieure à un seuil préfixé r, le nouvel exemple est ajouté sans autre modification, sinon un échantillonnage uniforme est appliqué. Supposons que le nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$ soit le n_b-ième exemple attribué à S_b, $n_b > r$. Un nombre e est tiré au hasard de manière uniforme dans $[1, n_b]$. Si $e \leq r$ alors le nouvel exemple remplace l'exemple à la e-ième position dans le réservoir. Sinon, il est écarté.

En fait, ce mécanisme permet à chaque exemple du flux affecté au cluster C_b d'avoir la même probabilité de rester dans l'échantillon S_b : lorsque $n_b \leq r$, tous les exemples sont stockés, la probabilité d'insertion vaut 1 ; sinon chaque exemple a une probabilité r/n_b d'être stocké ou de rester dans S_b. Par conséquent, avec cette stratégie d'échantillonnage, chaque échantillon peut contenir r exemples pour représenter les connaissances acquises au cours du temps dans une mémoire limitée.

2.3.2 Mises à jour des prototypes

Les mises à jour des prototypes dans le cluster sont basées sur une extension de la stratégie LVQ [38] qui permet de partitionner l'espace des attributs en fonction de chaque groupe de labels similaires. Nous rappelons ci-dessous le principe de la stratégie LVQ.

Learning Vector Quantization : L'algorithme LVQ a été proposé à l'origine pour résoudre des problèmes en classification multi-classes. Il utilise un nombre prédéfini de prototypes, chaque prototype étant composé d'un vecteur d'attributs et d'un label exclusif. L'objectif de LVQ est de déterminer les vecteurs d'attributs de sorte que chaque

prototype représente correctement les « régions » de leur label. Les régions associées aux labels sont définies par des hyperplans entre les prototypes qui forment une partition de Voronoï (voir Figure 4.3) dans l'espace des attributs [110]. Il est important de souligner qu'il est possible que plusieurs prototypes soient associés au même label ; ce qui entraîne que la région d'un label peut être représentée par plusieurs vecteurs d'attributs différents. Puisque chaque label doit avoir au moins un prototype pour être représenté, le nombre de prototypes doit être supérieur ou égal au nombre de labels.

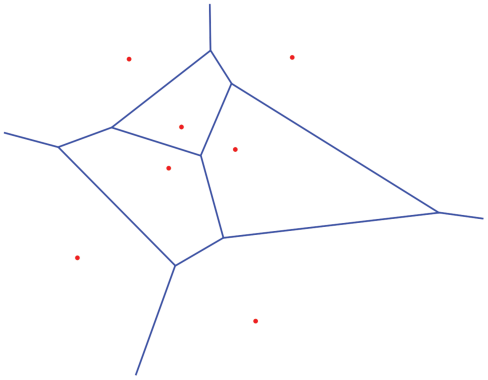


FIGURE 4.3 – La partition de Voronoï associée aux centroïdes en rouge

La version la plus populaire de LVQ est LVQ 2.1 qui met à jour deux prototypes à chaque étape, le meilleur et le second. Le meilleur (resp. second) prototype est celui dont le vecteur d'attribut est le plus proche du vecteur d'attribut du nouvel exemple parmi les prototypes qui ont le même label (resp. qui n'ont pas le même label) que le nouvel exemple. Pour mieux représenter la région du label, le meilleur prototype se rapproche du nouvel exemple, tandis que le second prototype s'éloigne du nouvel exemple avec un taux d'apprentissage. En raison de sa simplicité et de ses performances compétitives, nous avons adapté LVQ 2.1 à la classification multi-labels en flux.

L'extension LVQ 2.1 en multi-label (ML-LVQ) : Le « meilleur cluster » $C_b = \{\mathbf{c}_b^x, \mathbf{c}_b^y, S_b\}$ est ici défini par celui qui contient le Y-prototype le plus proche du vecteur de labels $\mathbf{y}_t$, $d_b^y = min(distance(\mathbf{y}_t, \mathbf{c}_v^y), v \in [1, s])$, et le cluster prédit $C_p = \{\mathbf{c}_p^x, \mathbf{c}_p^y, S_p\}$ par celui qui contient le X-prototype le plus proche du vecteur d'attributs $\mathbf{x}_t$, $d_p^x = min(distance(\mathbf{x}_t, \mathbf{c}_v^x), v \in [1, s])$. Selon le principe du partitionnement de MLT-ML, le nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$ appartient au meilleur cluster. Lorsque le nouvel exemple est inséré dans le meilleur cluster C_b, ses deux prototypes $\mathbf{c}_b^x$ et $\mathbf{c}_b^y$ sont « rapprochés » de cet exemple pour conserver l'homogénéité du cluster respectivement dans l'espace des attributs et des labels :

$$(\mathbf{c}_b^x, \mathbf{c}_b^y) = (\mathbf{c}_b^x, \mathbf{c}_b^y) + ((\mathbf{x}_t, \mathbf{y}_t) - (\mathbf{c}_b^x, \mathbf{c}_b^y)) \times lr_b \tag{4.4}$$

où la valeur du taux d'apprentissage lr_b est fonction du nombre d'exemples alloués

au cluster C_b. Lorsque le meilleur cluster d'affectation C_b est différent du cluster prédit C_p, ce dernier est « éloigné » du nouvel exemple pour renforcer l'hétérogénéité entre les clusters, et donc guider plus efficacement la prédiction :

$$(\mathbf{c}_p^x, \mathbf{c}_p^y) = (\mathbf{c}_p^x, \mathbf{c}_p^y) - ((\mathbf{x}_t, \mathbf{y}_t) - (\mathbf{c}_p^x, \mathbf{c}_p^y)) \times lr_p \tag{4.5}$$

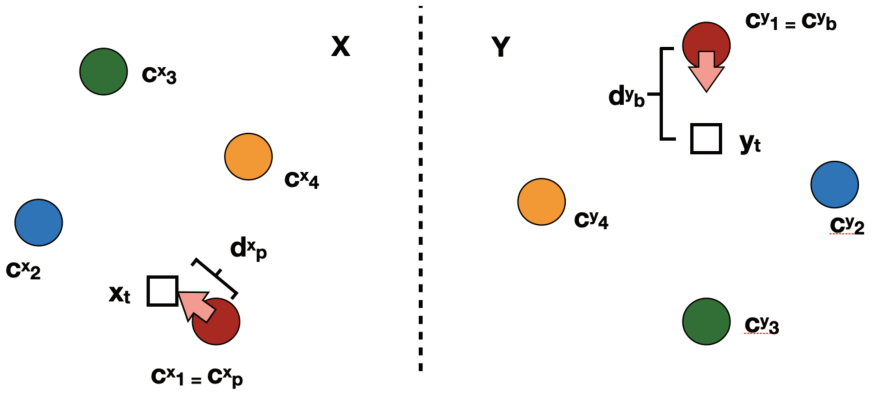
où lr_p est fonction du nombre d'exemples alloués au cluster C_p. Pour que les prototypes restent cohérents avec la distribution de données dans l'échantillon, le taux d'apprentissage lr_v, $v = 1$ à s, est maintenu en adéquation avec celui de l'échantillon : à l'initialisation, quand l'échantillon est vide, les prototypes sont initialisés avec le premier exemple et le taux d'apprentissage lr_v vaut 1 ; lorsque le nombre d'exemples n_v attribué au cluster C_v augmente, la probabilité r/n_v d'ajouter un nouvel exemple à échantillon diminue et le taux d'apprentissage diminue également, $lr_v = 1/n_v$.

Il est important de rappeler ici que chaque Y-prototype est le centroïde du vecteur de labels dans le cluster ; en d'autres termes, la valeur du label sur chaque Y-prototype représente en fait la fréquence d'apparition de chaque label dans son cluster, et a donc une valeur comprise entre 0 et 1.

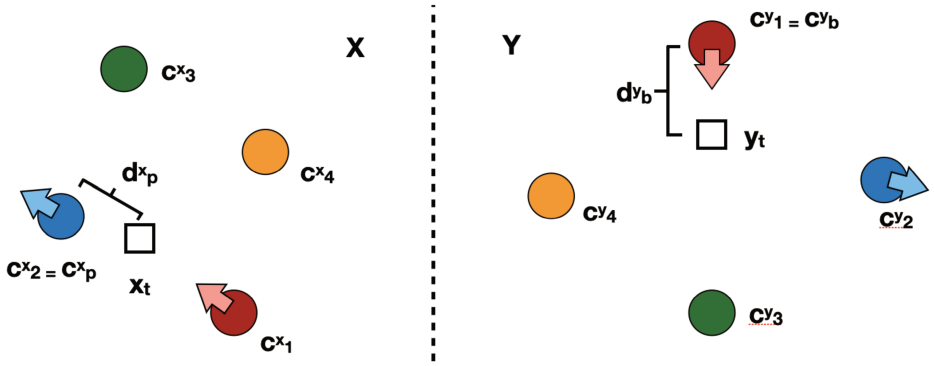
Dans notre extension de LVQ 2.1 à la classification multi-labels, chaque cluster peut donc regrouper des exemples avec des vecteurs de labels similaires, de sorte que son Y-prototype résume les distributions de labels du cluster, et que son X-prototype représente la région des labels du cluster dans l'espace d'attributs. Pour tout nouvel exemple, l'algorithme identifie le X-prototype c_p^x le plus proche de $\mathbf{x}_t$ et le Y-prototype c_b^y le plus proche de $\mathbf{y}_t$, qui peut appartenir théoriquement à deux clusters différents. Contrairement à LVQ 2.1 qui modifie toujours deux prototypes, le meilleur et le second, le processus d'apprentissage de ML-LVQ dépend de la présence simultanée du meilleur prototype et du prototype prédit dans le même cluster. Dans le cas où ils appartiennent au même cluster, le X-prototype $c_p^x = c_b^x$ et le Y-prototype $c_p^y = c_b^y$ se rapprochent tous les deux des vecteurs xt et yt (voir Figure 4.4a). Dans le cas contraire, les prototypes cbx et cbY du meilleur cluster se rapprochent respectivement de $\mathbf{x}_t$ et $\mathbf{y}_t$ alors que les prototypes c_p^x et c_p^y du cluster prédit s'en éloignent (voir Figure 4.4b).

2.3.3 Initialisation

MLT-ML est initialisée avec s clusters qui sont chacun construits à partir d'un premier exemple, tiré aléatoirement parmi les $1,000$ premiers exemples. Pour chaque cluster, le X-prototype (resp. Y-prototype) est initialisé avec le vecteur d'attributs (resp. le vecteur de labels) du premier exemple qui est inséré dans l'échantillon. Les autres exemples non sélectionnés sont tous utilisés ensuite pour mettre à jour MLT-ML selon les étapes d'apprentissage décrites ci-dessus : chaque exemple est inséré dans le cluster ayant le Y-prototype le plus proche de son vecteur de labels.



(a) Cas 1) : les prototypes meilleur et prédit sont identiques, $p = b = 1$. Seuls les meilleurs prototypes c_b^x et c_b^y vont s'approcher de $\mathbf{x}_t$ et $\mathbf{y}_t$, les autres prototypes ne changent pas.



(b) Cas 2) : les prototypes meilleur et prédit sont différents, $p = 2$ et $b = 1$. Les prototypes c_b^x et c_b^y vont s'approcher de $\mathbf{x}_t$ et $\mathbf{y}_t$ alors que les prototypes c_p^x et c_p^y vont s'en éloigner.

FIGURE 4.4 – Schéma de la règle de mise à jour de ML-LVQ.

2.4 Complexité en temps de calcul

Pour chaque exemple donné, la complexité de MLT-ML est $O((s + r) \times d) + (s \times l)$. Plus précisément, pour la phase de prédiction, elle consiste à identifier le X-prototype le plus proche dans un ensemble de s clusters et à calculer ses k plus proches voisins dans un échantillon de r exemples. La complexité de la première étape est $O(s \times d)$ et celle de la seconde est $O(r \times d)$ où d est le nombre de dimensions de l'espace d'attribut. Par conséquence, la complexité temporelle de la prédiction est $O((s + r) \times d)$. Et pour la phase d'apprentissage, elle consiste à identifier le Y-prototype le plus proche dans un ensemble de s clusters $O(s \times l)$, à mettre à jour les meilleurs prototypes et les prototypes prédits qui sont des opérations linéaires $O(d + l)$, et à insérer le nouvel exemple dans l'échantillon $O(1)$. La complexité temporelle de l'apprentissage est donc dominée par $O(s \times l)$. En conséquence, la complexité en temps de calcul pour chaque exemple est $O((s + r) \times d) + (s \times l)$.

3 Protocole expérimental

Dans cette section, nous décrivons en détail la configuration des expérimentations en précisant les jeux de données utilisés, le protocole de test et les choix effectués pour la mise en œuvre de MLT-ML.

3.1 Jeux de données

Nos expérimentations sont effectuées sur 15 jeux de données réels (voir Tableau 3.3) qui sont classiquement utilisés dans la littérature dans le contexte stationnaire. Pour simuler une distribution indépendante des données dans un flux stationnaire, l'ordre des données dans chaque jeu de données réel a été déterminé aléatoirement. Nous évaluons tous les classificateurs selon le même ordre des données fixé afin de rendre compte des performances dans les mêmes conditions expérimentales. Nous supposons un scénario d'apprentissage entièrement supervisé, dans lequel le vrai vecteur de labels est disponible immédiatement après que la prédiction ait été faite.

3.2 Protocole de test

Pour évaluer les performances des algorithmes, nous avons utilisé le protocole *Interleaved Test-Then-Train* pour les raisons rappelées dans les chapitres 2 et 3.

De plus, nous avons utilisé la procédure proposée par Losing et al [2] pour optimiser des hyperparamètres : les valeurs des hyperparamètres ont été déterminées sur la base de 20% des données et le reste de données a été utilisé pour comparer les performances des différents algorithmes. Les calculs ont été effectués avec un processeur Intel Xeon E5-1630v3@3.70GHZ avec 128 Go de mémoire sur un système Windows 10. Nos sources (code et résultats expérimentaux) sont disponibles sur Github [1].

Dans la suite nous décrivons les analyses préalables effectuées pour calibrer MLT-ML pour les expérimentations. En particulier, nous tentons de répondre aux questions suivantes :

- Quelle est la sensibilité de la sélection des paramètres de MLT-ML ?
- Quel est l'effet du choix de la distance vectorielle sur la performance du modèle ?
- Est-il nécessaire de prélever des échantillons pour chaque cluster ?
- La stratégie MLT-ML apporte-t-elle une amélioration significative par rapport à la simple utilisation de la moyenne de l'échantillon comme prototype ?

Les expérimentations mises en œuvre pour répondre à ces questions utilisent comme mesure d'évaluation le critère d'*Accuracy* basé sur les exemples selon l'analyse dans la

1. MLT-ML source code : https ://github.com/Cici-xihui/MLT-ML

section 1.2.

3.3 La sensibilité des hyperparamètres

Le comportement de MLT-ML est régi par trois hyper-paramètres : le nombre s des clusters, la taille r des échantillons et le nombre k des plus proches voisins considérés lors de la phase de prédiction. La valeur de k a été fixée à 3 selon toutes les recommandations d'algorithmes basées sur le classificateur kNN multi-label [94].

Nous nous sommes donc concentrés sur l'analyse de la sensibilité de MLT-ML à ses deux autres paramètres principaux : s et r. Les résultats sont présentés dans le tableau 4.1. Ils montrent que les performances augmentent lorsque s augmente puis se stabilisent à partir de $s = 40$; ce qui suggère qu'il suffit de partitionner l'espace des labels de manière limitée. En outre, pour la taille r des échantillons, les performances augmentent lorsque r augmente puis se stabilisent à partir de $r = 100$; ce qui signifie que la performance du modèle n'augmente pas indéfiniment selon la taille de la mémoire. De plus, il n'y a pas de différence significative dans les résultats obtenus ; ce qui tend à indiquer que MLT-ML est un classificateur stable.

Accuracy	$r = 25$	$r = 50$	$r = 100$	$r = 200$	$r = 400$	$r = 800$
$s = 20$	0.273	0.277	0.280	0.281	0.283	0.283
$s = 30$	0.292	0.294	0.296	0.296	0.297	0.297
$s = 40$	0.297	0.298	0.299	0.300	0.300	0.300
$s = 50$	0.299	0.299	0.299	0.299	0.300	0.300

TABLEAU 4.1 – Évolution des performances en fonction du nombre r des clusters et de la taille s des échantillons.

Nous choisissons les valeurs de paramètres $s = 40$ et $r = 100$ établies dans les expériences précédentes : elles représentent un bon compromis entre la qualité des résultats et la complexité en temps de calcul.

3.4 L'effet du choix de la distance vectorielle

Le choix de la mesure de distance entre les vecteurs de labels associés aux exemples a un impact à la fois sur la topologie des clusters et sur la complexité des calculs. Dans la très grande bibliothèque des mesures de distances (e.g. [111]), des références indiquent l'intérêt de la distance du cosinus pour des tâches de classification en grande dimension où les données sont décrites par des matrices binaires creuses [109] [112] [113]. Nous avons donc retenu cette distance mais nous l'avons tout de même préalablement comparée à la très populaire distance euclidienne pour valider le gain obtenu dans notre contexte précis.

Dans le tableau 4.2 ci-dessous nous désignons par X^E (resp. Y^E) la distance euclidienne dans l'espace des attributs (resp. labels) et X^C (resp. Y^C) la distance du cosinus

dans l'espace des attributs (resp. labels). Et nous comparons les quatre combinaisons (X^EY^E, X^EY^C, X^CY^E, X^CY^C).

Datasets	X^EY^E	X^EY^C	X^CY^E	X^CY^C
20NG	0.528	0.586	0.598	**0.647**
Bibtex	0.157	0.231	0.287	**0.303**
Bookmarks	0.187	0.216	0.220	**0.261**
Cooking	0.109	0.124	0.129	**0.162**
Corel16k001	0.118	0.110	0.116	**0.120**
Enron	0.373	0.368	0.399	**0.400**
Eukaryote	0.268	0.218	**0.273**	0.220
Human	0.312	0.294	**0.315**	0.296
Imdb	0.148	0.152	**0.156**	0.146
Mediamill	**0.288**	0.272	0.286	0.269
Reuters-K500	0.385	0.385	0.392	**0.398**
Scene	**0.679**	0.640	0.659	0.599
SLASHDOT	0.455	0.460	**0.464**	0.427
TMC2007	0.502	0.502	0.517	**0.520**
Yeast	0.444	0.411	**0.445**	0.416

TABLEAU 4.2 – Comparaisons des performances prédictives du modèle de 4 combinaisons de mesures de distance sur 15 jeux de données selon le critère d'*Accuracy*.

Nous constatons que la quatrième combinaison (X^CY^C) donne les meilleurs résultats pour les jeux de données avec une taille d'espace d'attributs supérieure à 500 et une taille d'espace de labels supérieure à 100. Cependant, pour les jeux de données avec un espace d'attributs limité ou un petit espace de labels, la quatrième combinaison ne présente aucun avantage. Cela confirme que la distance cosinus est plus performante dans un espace en grande dimension, mais pas nécessairement meilleure qu'une distance euclidienne dans les dimensions inférieures.

Dans la perspective future d'étendre nos travaux à « l'extreme multi-labels learning », nous nous focalisons sur les résultats obtenus pour la combinaison (X^CY^C) qui indiquent la pertinence du choix de la distance cosinus en grande dimension.

3.5 L'effet de l'échantillonnage

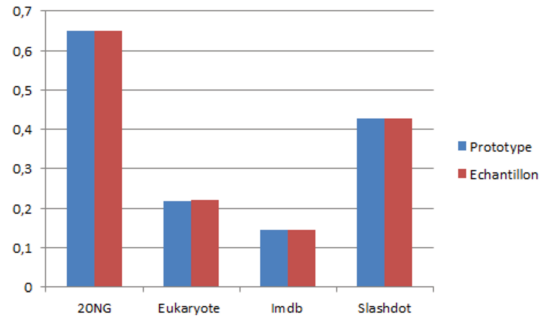
Dans MLT-ML, chaque paire de prototypes résume l'information sur les données de chaque cluster. Plus précisément, chaque Y-prototype résume la distribution des labels de son cluster associé. La valeur de chaque label représente la fréquence d'occurrence de ce label dans le cluster : plus elle est proche de 1, plus la fréquence d'occurrence est élevée et plus elle est proche de 0, plus la fréquence d'occurrence est faible. De même, chaque X-prototype résume la distribution des attributs associés de son cluster associé. Bien que ces prototypes résument une partie de l'information et peuvent être utilisés pour

prédire les labels d'un nouvel exemple, MLT utilise des échantillons supplémentaires pour apporter des informations plus granulaires afin d'améliorer les performances prédictives du modèle.

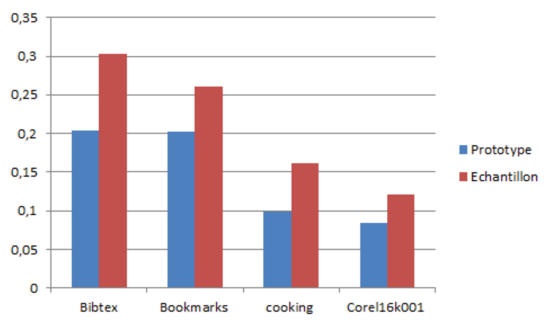
Les échantillons sont principalement destinés aux jeux de données présentant une cardinalité et une diversité relativement grandes, tels que Bibtex, Bookmark, Cooking et Corel16k. En effet, des valeurs élevées de ces deux caractéristiques représentent non seulement une plus grande variété de vecteurs de labels, mais aussi un plus grand nombre de labels associés à chaque exemple. Certains labels peuvent donc être omis si la prédiction est faite en utilisant uniquement les labels qui apparaissent fréquemment, $c_v^y \geq 0.5$, dans chaque cluster $C_v, v \in [0.s]$. Cependant, lorsque les deux caractéristiques sont petites, la variété des vecteurs de labels est petite et le nombre de labels associés à chaque exemple est généralement faible. Par exemple, la cardinalité de 20NG vaut 1.02, ce qui indique que la plupart des exemples n'ont qu'un seul label. Sachant que les exemples d'un même cluster partagent les mêmes labels, la prédiction avec le Y-prototype et la prédiction avec l'échantillon devraient donc être les mêmes dans ce cas.

Pour mesurer ces différents effets, nous avons mené des expériences sur deux ensembles de jeux de données : le premier (Figure 4.5a) considère des jeux de données à cardinalité et à diversité faibles (20NG, Eukaryote, IMDB et Slashdot) et le second (Figure 4.5b) considère des jeux de données à cardinalité et à diversité plus élevées (Bibtex, Bookmark, Cooking et Corel16k). Pour vérifier l'effet de l'échantillonnage, nous avons comparé les performances de classification du modèle avec la stratégie de prédiction utilisant le Y-prototype et la stratégie de prédiction utilisant l'échantillon.

Pour le premier ensemble de données (a), les résultats expérimentaux sont similaires, tandis que pour le deuxième (b), l'utilisation d'un échantillon a donné de meilleurs résultats. Afin de rester compétitif pour des jeux de données variés, nous avons donc conservé la stratégie d'échantillonnage. Cependant, dans des applications réelles, si une vérification



(a) Jeux de données à cardinalité et à diversité faibles



(b) Jeux de données à cardinalité et à diversité plus élevées

FIGURE 4.5 – Comparaison des performances du modèle avec prédiction basée sur les Y-prototypes et prédiction basée sur l'échantillon selon le critère d'*Accuracy*.

préalable des caractéristiques des jeux de données peut être menée et que celles-ci sont conformes au cas (a), l'ajout d'un échantillon pourrait être supprimé ; ce qui permettrait un gain en mémoire et en temps de calcul.

3.6 Contribution de la stratégie ML-LVQ

Les mises à jour des prototypes dans MLT-ML sont basées sur une nouvelle stratégie ML-LVQ. Pour vérifier son impact, nous avons comparée MLT-ML à une autre approche plus simple, qui consiste à calculer la moyenne de tous les exemples de l'échantillon du cluster pour la mise à jour des prototypes à chaque instant (stratégie Means).

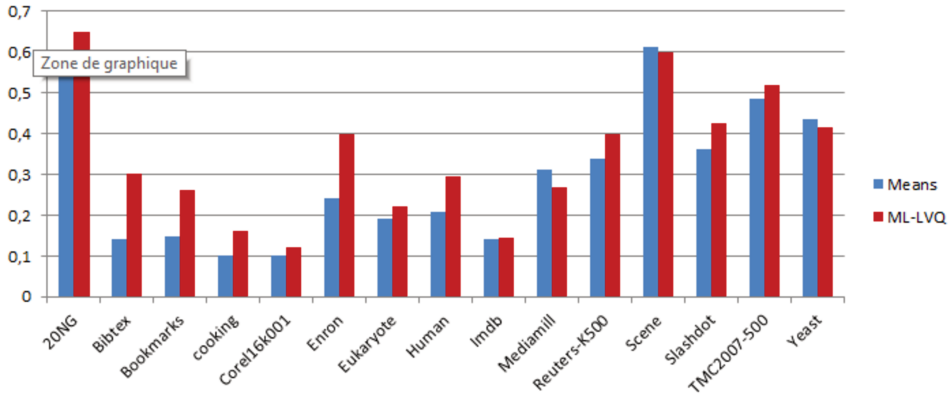


FIGURE 4.6 – Comparaisons des performances du modèle en mettant à jour les prototypes avec deux approches, ML-LVQ et *Means*, selon le critère d'*Accuracy*.

Les résultats (figure 4.6) montrent que ML-LVQ surpasse Means dans 11 des 15 jeux de données et qu'il n'y a pas de différence significative de performances entre les deux dans les 4 autres jeux. La performance prédictive du modèle est significativement améliorée par l'introduction de ML-LVQ eu égard à la stratégie basée sur un calcul de moyenne.

4 Comparaisons expérimentales

Dans cette section, nous avons comparé MLT-ML à neuf algorithmes de l'état de l'art représentatifs des meilleures stratégies exposées dans le Chapitre 3 : cinq algorithmes basés sur une stratégie d'adaptation (HTps, ISOUPTree, ML-AMRules, MLSAMPkNN et MLSAKNN) et quatre méthodes Ensemble (EaHTps, GOOWEML, RandomAMRules et AESAKNNS). Il convient de noter que nous n'avons pas supprimé les mécanismes de traitement des dérives de concept introduits dans les algorithmes dans nos expériences dans le contexte stationnaires car nous pensons que la gestion de la dérive ne devrait pas être déclenchée dans ce cas et ne devrait donc pas affecter les performances des algorithmes sur des flux de données stationnaires.

Les valeurs des paramètres de ces neuf algorithmes de l'état de l'art sont sélectionnées en fonction des valeurs recommandées par les auteurs. Nous avons ajouté un simple vote majoritaire qui joue le rôle de référence : il enregistre les fréquences de tous les vecteurs de labels distincts en temps réel et utilise le vecteur de labels ayant la fréquence le plus élevée comme la prédiction pour le nouvel exemple.

Critères d'évaluation de la classification. Les comparaisons expérimentales ont été mesurées avec deux critères basés sur les exemples présentés dans la section 1.2.3 : l'*Accuracy* et la *Fmesure*. Une analyse plus poussée avec des tests statistiques sur ces deux critères a été également effectuée pour évaluer les différences significatives et les similitudes entre les algorithmes. En utilisant le package R *scmamp* [114], nous avons appliqué le test de Friedman avec $\alpha = 0.05$ (confiance 95%) et l'avons complété avec le test post-hoc de Nemenyi.

De plus, la complexité temporelle est importante dans les scénarios de flux de données puisque les classificateurs doivent viser à maximiser la précision sous contrainte de temps. Nous avons donc également comparé les temps d'évaluation utilisés par chaque algorithme dans différents flux de données. Le temps d'évaluation représente le temps (en secondes) nécessaire pour faire une prédiction et mettre à jour le modèle de classification.

4.1 Résultats

Dans les comparaisons, nous distinguons les algorithmes d'adaptation et les approches ensemble afin de mieux identifier les similarités et différences entre leurs comportements.

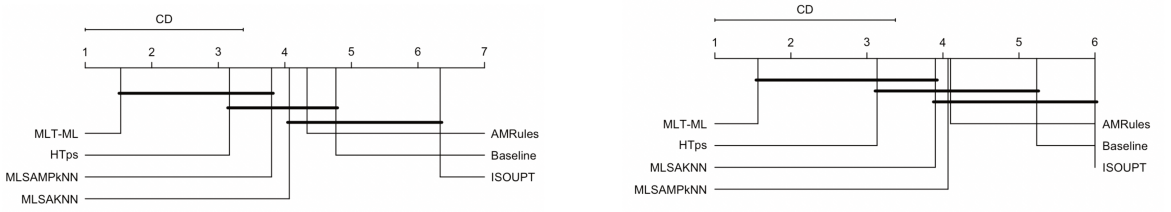
4.1.1 Comparaisons avec les algorithmes d'adaptation

Le Tableau 4.3 montre les résultats des comparaisons de MLT-ML avec les d'algorithmes d'adaptation et une baseline (les meilleurs résultats sont indiqués en gras). Pour évaluer la signification statistique des résultats, le test de Friedman et le test post hoc de Nemenyi ont été appliqués avec un même niveau de confiance égal à 0.05. Les valeurs p utilisées avec le test de Friedman étaient de 1.9e-07 et 6.07e-07 respectivement. Ce test appliqué à la fois avec l'*Accuracy* et la *Fmesure* confirme le rejet de l'hypothèse nulle selon laquelle les performances de toutes les méthodes sont similaires.

L'analyse post-hoc de Nemenyi (figure 4.7) montre que le classement moyen de LTM-ML est le meilleur pour ces deux mesures sur l'ensemble des algorithmes d'adaptation considérés. Mais des variations existent dans les comparaisons par paires : LTM-ML est significativement meilleur que tous les algorithmes d'adaptation à l'exception de HTps et MLSAMPKNN pour les deux mesures d'évaluation *Accuracy* et F-*Fmesure*. Les résultats détaillés montrent que LTM-ML est en première position pour 9 jeux sur 15 et en deuxième position pour 4 autres jeux.

	Baseline	HTps	ISOUPT	AMRules	MLSAMPKNN	MLSAKNN	MLT-ML
Accuracy							
20NG	0.049 ± 0.003	0.323 ± 0.019	0.150 ± 0.018	0.244 ± 0.006	0.079 ± 0.004	0.110 ± 0.007	$\mathbf{0.621 \pm 0.008}$
Bibtex	0.066 ± 0.006	0.124 ± 0.010	0.020 ± 0.006	0.022 ± 0.003	0.024 ± 0.003	0.083 ± 0.007	$\mathbf{0.261 \pm 0.009}$
Bookmarks	0.072 ± 0.002	0.182 ± 0.004	0.111 ± 0.007	0.025 ± 0.001	0.163 ± 0.003	0.172 ± 0.003	$\mathbf{0.249 \pm 0.003}$
Cooking	0.002 ± 0.001	0.049 ± 0.005	0.000 ± 0.000	0.065 ± 0.003	0.007 ± 0.001	0.008 ± 0.002	$\mathbf{0.157 \pm 0.006}$
Corel16k	0.042 ± 0.003	0.081 ± 0.005	0.001 ± 0.000	$\mathbf{0.135 \pm 0.004}$	0.032 ± 0.002	0.029 ± 0.003	0.119 ± 0.004
Enron	0.177 ± 0.017	0.174 ± 0.015	0.168 ± 0.012	0.250 ± 0.012	0.242 ± 0.017	0.260 ± 0.018	$\mathbf{0.365 \pm 0.016}$
Eukaryote	0.248 ± 0.010	$\mathbf{0.284 \pm 0.011}$	0.025 ± 0.006	0.217 ± 0.007	0.167 ± 0.009	0.150 ± 0.013	0.270 ± 0.010
Human	0.275 ± 0.017	$\mathbf{0.285 \pm 0.020}$	0.008 ± 0.006	0.223 ± 0.011	0.178 ± 0.014	0.104 ± 0.016	0.272 ± 0.017
IMDB	$\mathbf{0.210 \pm 0.002}$	0.210 ± 0.002	0.004 ± 0.001	0.049 ± 0.001	0.111 ± 0.002	0.084 ± 0.002	0.161 ± 0.002
Mediamill	0.326 ± 0.003	0.326 ± 0.003	0.331 ± 0.003	0.092 ± 0.001	0.338 ± 0.003	0.366 ± 0.003	$\mathbf{0.385 \pm 0.003}$
Reuters-K500	0.065 ± 0.007	0.217 ± 0.021	0.002 ± 0.001	0.017 ± 0.003	0.295 ± 0.013	0.286 ± 0.016	$\mathbf{0.409 \pm 0.013}$
Scene	0.174 ± 0.017	0.387 ± 0.054	0.009 ± 0.004	0.454 ± 0.015	$\mathbf{0.601 \pm 0.022}$	0.525 ± 0.033	0.597 ± 0.021
SLASHDOT	0.142 ± 0.012	0.132 ± 0.013	0.003 ± 0.002	0.026 ± 0.006	0.232 ± 0.015	0.173 ± 0.021	$\mathbf{0.377 \pm 0.017}$
TMC2007-500	0.187 ± 0.004	0.415 ± 0.010	0.331 ± 0.007	0.465 ± 0.004	0.402 ± 0.004	0.428 ± 0.006	$\mathbf{0.545 \pm 0.005}$
Yeast	0.372 ± 0.013	0.417 ± 0.025	0.391 ± 0.014	0.413 ± 0.012	$\mathbf{0.447 \pm 0.014}$	0.355 ± 0.018	0.443 ± 0.014
Fmesure							
20NG	0.049 ± 0.003	0.326 ± 0.019	0.151 ± 0.018	0.259 ± 0.007	0.080 ± 0.004	0.112 ± 0.007	$\mathbf{0.625 \pm 0.008}$
Bibtex	0.067 ± 0.006	0.138 ± 0.011	0.030 ± 0.008	0.027 ± 0.003	0.033 ± 0.004	0.116 ± 0.010	$\mathbf{0.318 \pm 0.009}$
Bookmarks	0.073 ± 0.002	0.191 ± 0.004	0.112 ± 0.007	0.038 ± 0.001	0.166 ± 0.003	0.182 ± 0.003	$\mathbf{0.271 \pm 0.003}$
Cooking	0.003 ± 0.001	0.066 ± 0.006	0.000 ± 0.000	0.094 ± 0.004	0.009 ± 0.002	0.012 ± 0.002	$\mathbf{0.204 \pm 0.007}$
Corel16k	0.053 ± 0.004	0.119 ± 0.007	0.001 ± 0.001	$\mathbf{0.199 \pm 0.005}$	0.047 ± 0.003	0.044 ± 0.004	0.159 ± 0.005
Enron	0.215 ± 0.018	0.224 ± 0.018	0.267 ± 0.019	0.354 ± 0.015	0.303 ± 0.019	0.359 ± 0.022	$\mathbf{0.470 \pm 0.017}$
Eukaryote	0.264 ± 0.011	$\mathbf{0.308 \pm 0.012}$	0.025 ± 0.006	0.286 ± 0.008	0.181 ± 0.009	0.166 ± 0.014	0.291 ± 0.011
Human	0.290 ± 0.017	$\mathbf{0.305 \pm 0.020}$	0.008 ± 0.006	0.293 ± 0.013	0.191 ± 0.015	0.115 ± 0.017	0.290 ± 0.017
IMDB	0.251 ± 0.002	$\mathbf{0.265 \pm 0.003}$	0.005 ± 0.001	0.076 ± 0.001	0.140 ± 0.002	0.113 ± 0.003	0.208 ± 0.002
Mediamill	0.434 ± 0.003	0.455 ± 0.003	0.478 ± 0.004	0.163 ± 0.001	0.453 ± 0.003	$\mathbf{0.522 \pm 0.003}$	0.497 ± 0.003
Reuters-K500	0.067 ± 0.007	0.226 ± 0.022	0.002 ± 0.001	0.018 ± 0.004	0.308 ± 0.013	0.309 ± 0.017	$\mathbf{0.435 \pm 0.013}$
Scene	0.179 ± 0.017	0.408 ± 0.057	0.009 ± 0.004	0.557 ± 0.014	0.612 ± 0.022	0.543 ± 0.033	$\mathbf{0.613 \pm 0.021}$
SLASHDOT	0.144 ± 0.013	0.137 ± 0.013	0.003 ± 0.002	0.027 ± 0.006	0.239 ± 0.015	0.181 ± 0.021	$\mathbf{0.394 \pm 0.017}$
TMC2007-500	0.228 ± 0.005	0.540 ± 0.012	0.447 ± 0.009	0.558 ± 0.005	0.490 ± 0.005	0.558 ± 0.007	$\mathbf{0.635 \pm 0.004}$
Yeast	0.481 ± 0.014	0.532 ± 0.028	0.530 ± 0.016	0.533 ± 0.013	$\mathbf{0.551 \pm 0.014}$	0.485 ± 0.022	0.548 ± 0.014

TABLEAU 4.3 – Comparaison des algorithmes d'adaptation avec les mesures *Accuracy* et *F-mesure* sur les 15 jeux de données.



(a) *Accuracy* (b) *Fmesure*

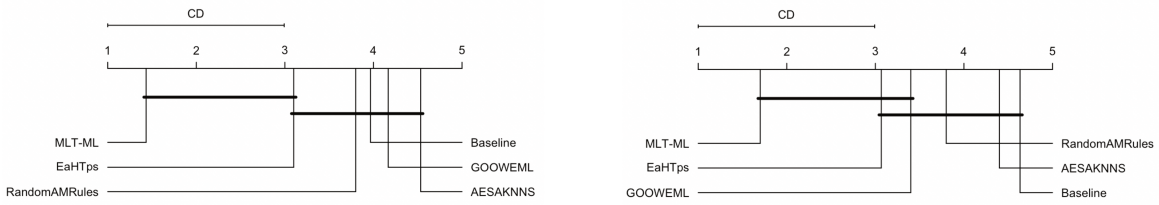
FIGURE 4.7 – Test post hoc de Nemenyi sur les résultats des algorithmes d'adaptation.

4.1.2 Comparaisons avec les approches ensembles

Le tableau 4.4 montre les résultats des comparaisons de LTM-ML avec les approches ensembles et une baseline. Les conclusions générales sont similaires au cas précédent : l'hypothèse nulle de similarité entre les performances est rejetée et le classement moyen de LTM-ML est le meilleur pour les deux mesures considérées (voir dans la figure 4.8). Des distinctions apparaissent également dans la comparaison paire à paire : LTM-ML est significativement meilleur que RandomAMRules, GOOWE-ML, AESAKNN et baseline pour les deux mesures *Accuracy* et *Fmesure*, mais la différence entre LTM-ML et EaHTps n'est pas statistiquement significative. Les résultats détaillés montrent que LTM-ML est en première position pour 11 jeux sur 15 et en deuxième position dans 2 autres jeux.

	Baseline	EaHTps	GOOWEML	RandomAMRules	AESAKNNS	MLT-ML
Accuracy						
20NG	0.049 ± 0.003	0.271 ± 0.025	0.148 ± 0.007	0.242 ± 0.001	0.001 ± 0.001	$\mathbf{0.621 \pm 0.008}$
Bibtex	0.066 ± 0.006	0.111 ± 0.013	0.025 ± 0.001	0.022 ± 0.003	0.063 ± 0.009	$\mathbf{0.261 \pm 0.009}$
Bookmarks	0.072 ± 0.002	0.176 ± 0.005	0.020 ± 0.000	0.021 ± 0.001	0.133 ± 0.004	$\mathbf{0.249 \pm 0.003}$
Cooking	0.002 ± 0.001	0.003 ± 0.002	0.013 ± 0.000	0.065 ± 0.003	0.005 ± 0.001	$\mathbf{0.157 \pm 0.006}$
Corel16k	0.042 ± 0.003	0.048 ± 0.004	0.052 ± 0.001	$\mathbf{0.136 \pm 0.004}$	0.026 ± 0.003	0.119 ± 0.004
Enron	0.177 ± 0.017	0.173 ± 0.016	0.199 ± 0.013	0.250 ± 0.012	0.268 ± 0.018	$\mathbf{0.365 \pm 0.016}$
Eukaryote	0.248 ± 0.010	$\mathbf{0.284 \pm 0.012}$	0.159 ± 0.005	0.217 ± 0.007	0.142 ± 0.011	0.270 ± 0.010
Human	0.275 ± 0.017	$\mathbf{0.282 \pm 0.017}$	0.185 ± 0.009	0.223 ± 0.011	0.034 ± 0.016	0.272 ± 0.017
IMDB	$\mathbf{0.210 \pm 0.002}$	0.210 ± 0.002	0.161 ± 0.001	0.111 ± 0.002	0.013 ± 0.002	0.161 ± 0.002
Mediamill	0.326 ± 0.003	0.337 ± 0.003	0.180 ± 0.002	0.092 ± 0.001	0.368 ± 0.003	$\mathbf{0.385 \pm 0.003}$
Reuters-K500	0.065 ± 0.007	0.196 ± 0.024	0.038 ± 0.001	0.016 ± 0.003	0.258 ± 0.018	$\mathbf{0.409 \pm 0.013}$
Scene	0.174 ± 0.017	0.079 ± 0.018	0.242 ± 0.017	0.454 ± 0.015	0.092 ± 0.017	$\mathbf{0.597 \pm 0.021}$
SLASHDOT	0.142 ± 0.012	0.122 ± 0.015	0.102 ± 0.005	0.026 ± 0.006	0.007 ± 0.003	$\mathbf{0.377 \pm 0.017}$
TMC2007-500	0.187 ± 0.004	0.430 ± 0.012	0.299 ± 0.006	0.466 ± 0.004	0.352 ± 0.013	$\mathbf{0.545 \pm 0.005}$
Yeast	0.372 ± 0.013	0.398 ± 0.016	0.423 ± 0.024	0.413 ± 0.012	0.294 ± 0.016	$\mathbf{0.443 \pm 0.014}$
Fmesure						
20NG	0.049 ± 0.003	0.274 ± 0.026	0.244 ± 0.011	0.266 ± 0.003	0.001 ± 0.001	$\mathbf{0.625 \pm 0.008}$
Bibtex	0.067 ± 0.006	0.122 ± 0.015	0.049 ± 0.002	0.027 ± 0.003	0.085 ± 0.011	$\mathbf{0.318 \pm 0.009}$
Bookmarks	0.073 ± 0.002	0.184 ± 0.005	0.038 ± 0.001	0.033 ± 0.001	0.138 ± 0.004	$\mathbf{0.271 \pm 0.003}$
Cooking	0.003 ± 0.001	0.004 ± 0.002	0.025 ± 0.001	0.094 ± 0.004	0.007 ± 0.001	0.204 ± 0.007
Corel16k	0.053 ± 0.004	0.067 ± 0.006	0.098 ± 0.002	$\mathbf{0.199 \pm 0.005}$	0.039 ± 0.005	0.159 ± 0.005
Enron	0.215 ± 0.018	0.224 ± 0.019	0.328 ± 0.021	0.354 ± 0.015	0.372 ± 0.021	$\mathbf{0.470 \pm 0.017}$
Eukaryote	0.264 ± 0.011	$\mathbf{0.311 \pm 0.012}$	0.267 ± 0.008	0.286 ± 0.008	0.156 ± 0.012	0.291 ± 0.011
Human	0.290 ± 0.017	$\mathbf{0.303 \pm 0.018}$	0.302 ± 0.015	0.293 ± 0.013	0.037 ± 0.017	0.290 ± 0.017
IMDB	0.251 ± 0.002	0.265 ± 0.003	$\mathbf{0.277 \pm 0.002}$	0.076 ± 0.001	0.016 ± 0.003	0.208 ± 0.002
Mediamill	0.434 ± 0.003	0.469 ± 0.004	0.305 ± 0.003	0.163 ± 0.001	$\mathbf{0.524 \pm 0.004}$	0.497 ± 0.003
Reuters-K500	0.067 ± 0.007	0.205 ± 0.025	0.073 ± 0.003	0.018 ± 0.004	0.273 ± 0.019	$\mathbf{0.435 \pm 0.013}$
Scene	0.179 ± 0.017	0.083 ± 0.019	0.352 ± 0.024	0.557 ± 0.014	0.092 ± 0.017	$\mathbf{0.613 \pm 0.021}$
SLASHDOT	0.144 ± 0.013	0.126 ± 0.015	0.181 ± 0.008	0.027 ± 0.006	0.007 ± 0.003	$\mathbf{0.394 \pm 0.017}$
TMC2007-500	0.228 ± 0.005	0.564 ± 0.014	0.458 ± 0.007	0.559 ± 0.005	0.463 ± 0.016	$\mathbf{0.635 \pm 0.004}$
Yeast	0.481 ± 0.014	0.525 ± 0.018	$\mathbf{0.569 \pm 0.031}$	0.533 ± 0.013	0.406 ± 0.021	0.548 ± 0.014

TABLEAU 4.4 – Comparaison des algorithmes ensembles avec les mesures *Accuracy* et *Fmesure* sur les 15 jeux de données.



(a) *Accuracy* (b) *Fmesure*

FIGURE 4.8 – Test post hoc de Nemenyi sur les résultats des algorithmes ensembles.

4.1.3 Temps de calcul

Les temps de calcul moyens sur tous les jeux de données sont fournis dans le tableau 4.5. LTM-ML maintient un de temps de calcul le plus faible par rapport aux autres algorithmes de l'état de l'art.

Les approches par adaptation						Les approches ensembles			
HTps	ISOUPTree	AMRules	MLSAMPKNN	MLSAKNN	MLT-ML	EaHTps	GOOWE	RandomAMRules	AESAKNNS
146.85	231.3	568.04	39.83	93.24	**23.02**	1817.14	2281.59	13254.46	2856.93

TABLEAU 4.5 – Temps de calcul moyen des 10 algorithmes sur 15 jeux de données en secondes

4.2 Conclusion

En exploitant la corrélation entre les labels, notre nouvelle méthode d'apprentissage multi-label en flux stationnaire MLT-ML partitionne l'espace des labels en un nombre restreint de sous-espaces pour réduire le grand problème initial en des problèmes à une échelle réduite et de complexité plus faible. Les résultats expérimentaux montrent que dans le contexte des flux stationnaires MLT-ML est très compétitive avec les algorithmes de l'état de l'art. De plus, il a la complexité en temps de calcul la plus faible. Ainsi, la stratégie mise en œuvre dans MLT-ML s'avère prometteuse pour une extension au cadre des flux non stationnaires. Le chapitre suivant est consacré à cette extension : nous conservons la stratégie de partitionnement de l'espace de recherche par une clusterisation pour réduire la complexité des calculs en introduisant des mécanismes de gestion des dérives de concepts qui peuvent apparaître dans les flux. Un premier algorithme (ODM – *Online Dual Memory*) permet de valider l'intérêt de la combinaison de deux mémoires à temporalités différentes (à court et à long terme) pour gérer les dérives de concepts de différentes natures. Un second algorithme (A2ML - *Adaptive Memories for Multi-Label stream classification* permet d'améliorer significativement la capacité d'adaptation aux différentes distributions des données dans le flux de ODM.

Chapitre 5

Explorations de la combinaison de deux mémoires pour la classification multi-labels en flux non-stationnaires

Sommaire

1 Introduction

Dans les applications, les changements sur les distributions de données peuvent intervenir sous des modalités variées. Cependant, comme nous l'avons constaté dans la bibliographie la majeure partie des algorithmes de l'état de l'art ne sont capables de gérer que des types spécifiques de dérive de concept (voir le Tableau 3.2). Lorsque des « formes » inattendues par rapport aux modèles se produisent, les systèmes qui ne prévoient qu'un nombre limité de formes de dérive ont, au mieux, des performances médiocres ou, au pire, échouent complètement [69]. Dans l'éventail des algorithmes présentés dans notre état de l'art, ceux basés sur une combinaison de mémoires ont retenu notre attention car ils semblent particulièrement prometteurs pour faire face à des dérives conceptuelles de différentes natures. Cependant, leur grande complexité en temps de calcul reste souvent

une limite majeure lorsqu'ils sont déployés dans des applications réelles. Un exemple typique concerne le domaine de la cybersécurité, dans lequel plusieurs milliers d'événements peuvent se produire par seconde. Un algorithme qui ne parvient pas à identifier les attaques à temps peut causer par rétroaction des dommages importants.

Pour soulever les défis posés par les dérives de concept en MLSC, tout en conservant l'intérêt de la combinaison des deux mémoires, nous avons développé une nouvelle approche appelée *Online Dual Memory* (ODM) [39]. En termes simples, ODM est un modèle combinant une version de la mémoire à long-terme présentée au chapitre 4 avec une mémoire à court-terme basée sur une fenêtre glissante. Nous avons introduit une mémoire à court-terme pour adapter l'apprentissage à des dérives abruptes et nous avons conservé le principe d'une mémoire à long-terme pour prendre en compte d'autres types de dérives. Les résultats expérimentaux sur 15 jeux de données aux propriétés variées montrent que ODM obtient des résultats compétitifs tout en maintenant une faible complexité. Cependant, les mécanismes de mise à jour qui régissent la mémoire à long-terme entraînent une dégradation de l'efficacité d'adaptation à de nouveaux concepts au fil du temps sur les flux non stationnaires. Une analyse détaillée du comportement de l'algorithme nous a permis de comprendre que sa dégradation sur le long terme était causée par la déficience de la mémoire à long terme d'ODM à maintenir un apprentissage efficace sur un flux potentiellement très long. Pour surmonter cette limitation, tout en conservant les bonnes propriétés d'ODM, nous avons donc proposé un deuxième nouvel algorithme, appelé Adaptive *Adaptive Memories for Multi-Label stream classification* (A2ML) [40], qui intègre plus efficacement les changements de distributions au cours du temps.

En outre, pour une compréhension plus approfondie du comportement de l'algorithme sur les flux non-stationnaires, nous avons développé un nouveau protocole d'évaluation pour générer des flux non-stationnaires. Ce nouveau protocole permet d'introduire explicitement différents types de dérives et différents changements de distribution. Nous avons comparé ODM et A2ML avec deux algorithmes de l'état de l'art représentatifs des meilleures stratégies sur huit différents flux de données non stationnaires de plus de $80,000$ exemples générés par notre nouveau protocole. Les résultats expérimentaux montrent que notre nouvel algorithme A2ML est le plus performant quelle que soit la nature de la dérive tout en conservant comme ODM une faible complexité algorithmique.

La suite de ce chapitre est organisée comme suit. La Section 2 décrit l'algorithme ODM, et analyse avec précision ses avantages et ses limites. Le nouvel algorithme A2ML développé pour améliorer la stratégie implémentée dans ODM est détaillé dans la Section 3 avec une analyse de ses propriétés. La Section 4 décrit le nouveau protocole adapté aux expérimentations avec des flux de données non stationnaires. Les comparaisons expéri-

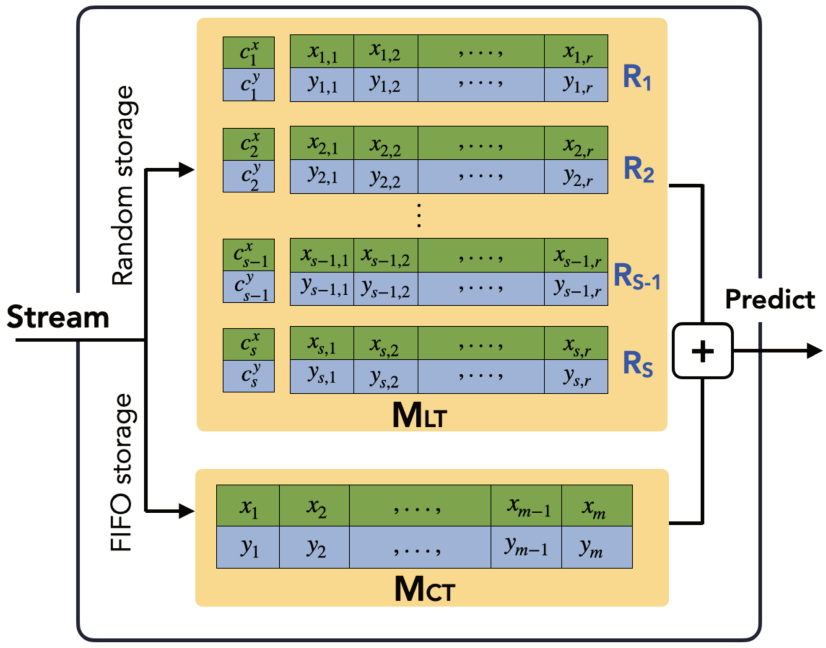


FIGURE 5.1 – Structure d'ODM : une combinaison d'une mémoire à court terme M_{CT} et d'une mémoire à long terme M_{LT}.

mentales de A2ML avec des algorithmes de l'état de l'art sur des flux non stationnaires sont présentées dans la Section 5.

2 L'algorithme ODM (Online Dual Memory)

Pour simplifier les notations, les différents composants de l'algorithme sont décrits pour un instant quelconque du flux de données.

La mémoire à court-terme et la mémoire à long-terme dans ODM sont deux ensembles distincts (voir la Figure 5.1), dénotés par M_{CT} and M_{LT} respectivement, qui évoluent à chaque pas de temps t. Plus précisément, la mémoire à court-terme M_{CT} est une fenêtre glissante de taille fixe qui stocke les m exemples les plus récents du flux dans une structure de données FIFO (*First-In-First-Out*) afin de s'adapter à la nouvelle distribution de données rapidement et de résister aux dérives abruptes, $M_{CT} = \{(\mathbf{x}_u, \mathbf{y}_u) \in \mathcal{X} \times \mathcal{Y} | u = 1, ..., m\}$.

La mémoire à long-terme M_{LT} est un ensemble de s clusters $M_{LT} = \{C_v | v = 1, ..., s\}$ décrits chacun par un triplet $C_v = (\mathbf{c}_v^x, \mathbf{c}_v^y, S_v)$ composé de deux prototypes et d'un échantillon. Nous reprenons ici la structure de la mémoire à long terme que nous avions introduite au Chapitre 4 car nos résultats expérimentaux ont confirmé son efficacité. Les clusters peuvent ici varier dans le temps afin de s'adapter aux changements de distribution. À chaque instant, le nouvel exemple est affecté au cluster en fonction de la similarité entre son vecteur de labels et les Y-prototypes. Une fois que le cluster le plus similaire est identifié, ses X et Y-prototypes sont modifiés via la stratégie ML-LVQ afin de représenter

la nouvelle distribution de données apportés par un nouvel exemple et son échantillon stocke le nouvel exemple par une stratégie de « reservoir sampling » (RS) qui sert à conserver un résumé des connaissances acquises au fil du temps dans une mémoire limitée. Avec ces mécanismes d'apprentissage, la mémoire à long-terme peut à la fois s'adapter aux dérives progressivement et se rappeler des anciens concepts dans une mémoire limitée.

Dans la suite, nous décrivons plus précisément les processus de prédiction et d'apprentissage d'ODM selon le protocole *Interleaved Test-Train*.

2.1 Prédiction

A chaque instant t, la prédiction $\mathbf{p}_t$ d'un nouvel exemple $\mathbf{x}_t$ est calculée par une stratégie d'agrégation des vecteurs prédits indépendamment par les deux mémoires. Pour la mémoire à court terme, la prédiction $\mathbf{p}_{CT}$ est calculé par un classifieur kNN multi-label appliqué sur l'ensemble M_{CT}. Pour la mémoire à long terme, la prédiction $\mathbf{p}_{LT}$ est basée sur une identification du cluster prédit de la mémoire dont le X-prototype est le plus similaire à $\mathbf{x}_t$, suivie d'un calcul avec un classifieur kNN dans ce cluster. Enfin, la prédiction $\mathbf{p}_t$ est déduite de l'agrégation des prédictions $\mathbf{p}_{CT}$ et $\mathbf{p}_{LT}$ calculées à partir des mémoires à court et à long terme :

$$\mathbf{p}_t = \frac{w_{CT} \times \mathbf{p}_{CT} + w_{LT} \times \mathbf{p}_{LT}}{w_{CT} + w_{LT}} \tag{5.1}$$

où les contributions respectives w_{CT} et w_{LT} de chaque prédiction sont défini par la moyenne des *Accuracy* entre les prédictions et les vrais vecteurs de labels sur les derniers m exemples du flux.

En fait, lorsque la dérive se produit, la mémoire à court terme M_{CT} s'adapte plus rapidement au nouveau concept, tandis que la mémoire à long terme M_{LT} prend plus de temps pour s'adapter progressivement à la dérive afin de résumer plus d'informations à partir de la nouvelle distribution. Ainsi, la stratégie consistant à calculer le poids en fonction de la précision de la prédiction permet à la mémoire à long terme et à la mémoire à court terme de coopérer afin que le modèle maintienne toujours de meilleures prédictions.

2.2 Apprentissage

A chaque instant t, les deux mémoires M_{CT} et M_{LT}, ainsi que leurs poids associés w_{CT} et w_{LT} sont mis à jour indépendamment avant la prochaine prédiction.

Pour la mémoire à court-terme M_{CT} : chaque nouvel exemple est simplement inséré à la première position de la fenêtre, et tous les autres exemples dans la fenêtre sont décalés d'une position. Lorsque le nombre d'exemples dépasse la taille de la fenêtre, l'exemple le plus ancien de la fenêtre est oublié afin de maintenir la taille de la mémoire à court-

terme constante. Une valeur restreinte de m permet d'adapter rapidement le modèle à la distribution récente du flux de données afin de résister aux dérives abruptes.

Pour la mémoire à long terme M_{LT} : le nouvel exemple est inséré dans le meilleur cluster dont le Y-prototype est le plus similaire à $\mathbf{y}_t$. La processus reprend celui présenté au Chapitre 4. A chaque instant, le meilleur cluster et le cluster prédit s'adaptent au nouveau concept en se mettant à jour avec le nouvel exemple. En fait, dans M_{LT}, la capacité des prototypes à s'adapter à la dérive détermine la capacité de la mémoire à long terme à s'adapter à la dérive, car toutes les prédictions et les apprentissages sont guidés par les prototypes.

Pour illustrer le mécanisme d'adaptation des prototypes à l'arrivée d'un nouveau concept, considérons un exemple jouet de classification multi-labels avec quatre labels (λ_1, λ_2, λ_3 et λ_4). Supposons que les combinaisons de labels $\{\lambda_1, \lambda_2\}$ et $\{\lambda_3, \lambda_4\}$ se produisent fréquemment, mais que λ_1 n'apparaît jamais simultanément avec λ_3 et λ_4 avant l'instant t_1. Selon la stratégie d'apprentissage de M_{LT}, les exemples avec les labels λ_1 et λ_2 sont regroupés ensemble dans le même cluster C_1, tandis que les exemples avec les labels λ_3 et λ_4 sont regroupés ensemble dans le même cluster C_2. Le X-prototype de cluster C_1 (resp. C_2) représente donc la distribution d'attributs associée aux labels λ_1 et λ_2 (resp. λ_3 et λ_4) et le Y-prototype représente la distribution de labels dont les valeurs y^1 et y^2 (resp. y^3 et y^4) sont proches de 1 et les valeurs y^3 et y^4 (resp. y^1 et y^2) sont proches de 0.

Comme la dégradation de la performance du modèle est dûe principalement au changement de $\mathbb{P}_t(\mathbf{y})$ ou/et du changement de $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$, l'adaptation des prototypes aux changements dépend des différentes causes :

- **Pour la cause I** : $\mathbb{P}_{t1}(\mathbf{y}) \neq \mathbb{P}_{t2}(\mathbf{y})$ mais $\mathbb{P}_{t1}(\mathbf{x}|\mathbf{y}) = \mathbb{P}_{t2}(\mathbf{x}|\mathbf{y})$. Supposons qu'après l'instant t_2, la combinaison de labels $\{\lambda_1, \lambda_3, \lambda_4\}$ se produise fréquemment et que la combinaison de labels $\{\lambda_1, \lambda_2\}$ disparaisse. Le vecteur de labels $\mathbf{y}_t = [1, 0, 1, 1]$ du nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$ est le plus proche du Y-prototype de C_2 : C_2 est donc le meilleur cluster. Mais le vecteur d'attributs $\mathbf{x}_t$ est le plus proche du X-prototype de C_1 parce que $\mathbf{c}_1^x$ représente la distribution d'attributs associée à λ_1 ; C_1 est donc le cluster prédit. Dans le meilleur cluster C_2, mis à jour avec le nouvel exemple, le Y-prototype $\mathbf{c}_2^y$ augmente les valeurs de y^1 en s'approchant de $\mathbf{y}_t$, tandis que le X-prototype $\mathbf{c}_2^x$ apprend la distribution d'attributs associée au label λ_1 en s'approchant de $\mathbf{x}_t$. En revanche, le cluster prédit C_1 réduit l'influence de λ_1 respectivement dans l'espace d'attributs et de labels en s'éloignant du nouvel exemple. Au fil du temps, les deux clusters s'adaptent à la nouvelle distribution de données et le meilleur cluster devient le cluster prédit comme attendu.
- **Pour la cause II** : $\mathbb{P}_{t1}(\mathbf{y}) = \mathbb{P}_{t2}(\mathbf{y})$ mais $\mathbb{P}_{t1}(\mathbf{x}|\mathbf{y}) \neq \mathbb{P}_{t2}(\mathbf{x}|\mathbf{y})$. Supposons qu'après l'instant t_2, les valeurs d'attributs associées au label λ_1 changent. Comme la proba-

bilité $\mathbb{P}_t(y)$ ne change pas, C_1 reste le meilleur cluster pour les nouveaux exemples associés au label λ_1. En se mettant à jour avec ces nouveaux exemples, le X-prototype $\mathbf{c}_1^x$ du meilleur cluster change progressivement les nouvelles valeurs des attributs associées au label λ_1 en approchant les vecteurs d'attributs des nouveaux exemples dans le temps pour s'adapter à la nouvelle distribution de données. Si la valeur de y^1 dans $\mathbf{c}_1^y$ valait déjà à 1, après rapprochement, la valeur reste à 1.

— **Pour la cause III** : lorsque $\mathbb{P}_t(y)$ et $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$ changent simultanément, le meilleur cluster et le cluster prédit s'adaptent de la même manière au nouveau concept en se mettant à jour avec le nouvel exemple.

À chaque instant, les prototypes permettent donc d'orienter les nouveaux exemples vers un échantillon du meilleur cluster, de sorte que chaque échantillon s'adapte à la nouvelle distribution de données en même temps. Et, comme les exemples appartenant à l'ancienne distribution ne sont pas oubliés dans l'échantillon avec la stratégie RS au fil du temps, cela permet à M_{LT} de réagir rapidement lorsque d'anciens concepts réapparaissent.

Pour la mise à jour des poids : le poids w_{CT} est recalculé selon les dernières m prédictions de M_{CT}, selon la formule suivante :

$$w_{CT} = \frac{1}{m} \sum_{i=t-m+1}^{t} Acurracy_i = \frac{1}{m} \sum_{i=t-m+1}^{t} \frac{|\mathbf{y}_i \cap \mathbf{p}_{CT_i}|}{|\mathbf{y}_i \cup \mathbf{p}_{CT_i}|} \tag{5.2}$$

où $\mathbf{p}_{CT_i}$ est la prédiction candidate basée sur la mémoire à court-terme et $\mathbf{y}_i$ est le vrai vecteur de labels à l'instant $i, i \in [t-m+1, t]$. Le calcul de w_{LT} est le même selon la formule 5.2 avec les dernières m prédictions de M_{LT} et les derniers m vrais vecteurs de labels.

2.3 Complexité en temps de calcul pour un nouvel exemple du flux

Dans le Chapitre 4, nous avons déjà analysé la complexité de MLT-ML qui est $O((s+r) \times d + (s \times l))$. Pour la mémoire à court-terme M_{CT}, la complexité du processus est déterminée par la recherche des k plus proches voisins dans une fenêtre de taille m, qui est en $O(m \times d)$ où d est le nombre de dimensions de l'espace d'attributs ; et la complexité en temps de calcul de l'apprentissage est déterminée par la stratégie FIFO qui est une opération constante $O(1)$.

En conséquence, pour chaque exemple donné, la complexité globale d'ODM est de l'ordre de $O((s+r+m) \times d + (s \times l))$.

2.4 L'analyse d'ODM

En combinant les mémoires à court-terme et à long-terme, ODM est non seulement capable de s'adapter à tous les types de dérive, mais est aussi capable de maintenir la stabilité dans un flux de données stationnaires. De plus, par construction, ODM a une complexité en temps faible.

Cependant, en raison de sa mémoire à long terme, ODM reste un algorithme qui connaît des limites pour les flux non-stationnaires. Rappelons que le taux d'apprentissage pour les prototypes dans chaque cluster est maintenu cohérent avec la probabilité d'insertion dans l'échantillon. Cependant, cette probabilité d'insertion avec le « reservoir sampling » est $\frac{r}{n}$ qui tend vers 0 au fur et à mesure que le nombre d'exemples n alloués à l'échantillon augmente [115]. Autrement dit, la stratégie RS effectuée dans l'échantillon est victime du phénomène d'« apprentissage gelé » : la vitesse à laquelle le cluster apprend le nouveau concept diminue à mesure que la probabilité d'insertion diminue. Ainsi, l'efficacité d'ODM à s'adapter au nouveau concept diminue gravement avec le temps.

De plus, les vecteurs de labels des exemples nouvellement arrivés peuvent être non-similaires aux Y-prototypes des clusters existants en raison des variations de la distribution des labels $\mathbb{P}_t(y)$ sur le flux non-stationnaire. Rappelons que les clusters sont initialisés en sélectionnant aléatoirement des exemples parmi les $1,000$ premiers du flux de données du flux de données. La distribution de fréquence des labels peut être très déséquilibrée et certains labels peuvent ne jamais apparaître dans les $1,000$ premières données. Cependant, comme la distribution des labels change avec le temps, ces labels qui n'apparaissent pas dans le flux de données initial peuvent apparaître fréquemment dans d'autres périodes. Le problème est que ces apparitions ultérieures et les apparitions initiales des labels peuvent être indépendantes les unes des autres - ces labels ne sont jamais associés aux mêmes exemples. Par conséquent, le vecteur de labels représenté par les labels apparus ultérieurement peut être non-similaires aux Y-prototypes des clusters existants : leur similarité du cosinus vaut 0.

Cependant, ODM ne dispose pas d'un mécanisme permettant de créer de nouveaux clusters pour gérer ces nouveaux labels. En effet, ODM les affecte systématiquement au premier cluster C_1 ; les nouveaux vecteurs de labels non-similaires ont une distance cosinus de 1 par rapport à tous les Y-prototypes, et le meilleur cluster est systématiquement choisi comme étant C_1. Cela rend C_1 très hétérogène. La mémoire à long terme d'ODM est donc victime d'un phénomène que l'on peut qualifier de « réticence aux nouvelles combinaisons » qui entraîne une dégradation des performances prédictives du modèle au fil du temps. Pour tenter de surmonter les limites de la mémoire à long-terme d'ODM décrites précédemment, nous avons proposé un nouvel algorithme A2ML qui reprend la stratégie de base d'ODM mais qui s'en distingue par deux nouveaux mécanismes dans

l'apprentissage de la mémoire à long terme.

3 L'algorithme A2ML (Adaptive Memories for Multi-Label stream classification)

L'algorithme d'A2ML a la même structure de données que ODM : la mémoire à court terme $M_{CT} = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} | i = 1, ..., m\}$ est composée d'une fenêtre glissante de taille préfixé m pour résister aux dérives abruptes et la mémoire à long terme $M_{LT} = \{C_v\}_{1 \leq v \leq s}$ est structurée par un ensemble limité de s clusters pour accumuler progressivement des informations sur le flux au-delà des m derniers exemples afin de s'adapter à la dérive graduellement et de bien représenter les nouveaux concepts.

La mémoire à court terme M_{CT} est alimentée par un processus identique à celui utilisé dans ODM. En revanche, deux points majeurs distinguent les stratégies utilisées pour la mémoire à long terme : l'usage dans A2ML d'un échantillonnage de type stratégie Biased Random Sampling [116] pour la mise à jour des échantillons et l'insertion dynamique de nouveaux clusters pertinents dans M_{LT}. Ces différences ont des conséquences directes dans les mécanismes d'apprentissage. Nous décrivons dans la suite les deux nouveautés concernant la mise à jour de la mémoire à long terme.

Biased Random Sampling : Afin de réguler le choix de l'échantillonnage du flux, Tabassum et Gama [116] proposent une extension du RS, appelée Biased Random Sampling (BRS), qui permet d'améliorer la qualité de l'apprentissage de la distribution récente. Comme pour la stratégie RS, elle commence par remplir l'échantillon avec les premiers exemples du flux de données. Mais ensuite, BRS ne calcule pas la probabilité d'insertion car chaque nouvel exemple est inséré systématiquement dans l'échantillon. L'insertion est effectuée en remplaçant l'instance à une position aléatoire (générée aléatoirement dans l'intervalle $[0, r]$ où r est la taille de l'échantillon) par la nouvelle instance du flux. Ce caractère systématique permet de mettre à jour de façon continue la distribution actuelle et le caractère aléatoire de l'insertion permet de conserver la trace des anciennes données. Une fois que l'échantillon est rempli, on peut considérer que la vitesse de mise à jour de l'échantillon est de $1/r$ parce qu'un exemple dans l'échantillon est remplacé par un nouvel systématiquement avec la probabilité $1/r$. Comme la valeur de r est prédéterminée et fixe dans A2ML, le taux d'apprentissage ne diminue pas avec le temps. Ainsi, A2ML n'est pas confronté au problème d'apprentissage gelé et sa capacité d'adaptation persiste à travers le temps.

Insertion dynamique de nouveaux clusters pertinents : L'insertion dynamique de nouveaux clusters « pertinents » dans la mémoire à long terme vise à mieux traiter

le problème de « réticence aux nouvelles combinaisons». Dans un flux non-stationnaire, de nouveaux vecteurs de labels pouvant être fortement dissimilaires aux clusters précédemment créés, il est alors nécessaire de créer de nouveaux clusters pour ces nouveaux vecteurs de labels. La stratégie de création est étroitement inspirée de l'algorithme CluStream [117] qui est bien adapté à l'objectif visé. La première étape de CluStream consiste à remplir une mémoire de taille fixe par un ensemble de clusters caractérisés chacun par un diamètre maximal qui délimite sa zone de couverture. Plus précisément, le diamètre est défini par l'écart type de la distance moyenne des exemples du cluster à son centroïde. Ensuite, l'arrivée d'un nouvel exemple conduit à vérifier s'il est dans la zone de couverture d'un cluster existant. Si c'est le cas, il est ajouté au cluster identifié ; sinon un nouveau cluster est créé et le nouvel exemple est ajouté à ce cluster.

Le calcul de la zone de couverture de chaque cluster a été adapté dans A2ML. Nous faisons ici l'hypothèse qu'un nouveau cluster est créé si aucun des clusters déjà existants ne possède les labels du nouvel exemple. Ainsi, chaque combinaison de labels dans un même cluster doit avoir au moins un label en commun avec les autres. En pratique, cela peut se vérifier avec la distance du cosinus entre le vecteur de labels du nouvel exemple et les Y-prototypes des clusters existants. La distance maximale est alors fixée à 1.0 dans A2ML, ce qui correspond bien à la propriété requise selon laquelle chaque combinaison de labels dans un même cluster a au moins un label en commun avec les autres.

Le processus de prédiction est identique à celle d'ODM. Nous détaillons donc uniquement le processus d'apprentissage d'A2ML dans la suite.

3.1 Le processus d'apprentissage

Après avoir découvert le vrai vecteur de labels $\mathbf{y}_t$ du nouvel exemple, les deux mémoires, ainsi que leurs poids associés, sont mis à jour avec le nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$.

La mémoire à court terme M_{CT} est alimentée par un processus d'apprentissage identique à celui utilisé dans ODM. La mise à jour de la mémoire à long terme M_{LT} lors de l'arrivée d'un nouvel exemple est réalisée par la combinaison de deux mécanismes qui sont déclenchés en fonction de l'évaluation du changement apporté par $\mathbf{y}_t$ dans la distribution des labels. Lorsque $\mathbf{y}_t$ est "proche" des vecteurs de labels des exemples représentés par un des clusters C_v, $v = 1$ à s, le nouvel exemple est ajouté au cluster le plus proche, qui est alors mis à jour ; sinon, un nouveau cluster est créé et initialisé avec le nouvel exemple (voir l'algorithme 1). Ces étapes sont décrites plus en détail ci-dessous.

Identification du cluster où le nouvel exemple est ajouté. Pour conserver l'homogénéité des clusters de M_{LTM} au cours du temps, un nouvel exemple est ajouté au cluster avec lequel il a le plus de labels en commun. Cela se vérifie aisément avec la distance du cosinus $distance(\mathbf{y}_t, \mathbf{c}_v^y)$, $v = 1$ à s : si aucune des distances du cosinus dans

Algorithme 1 : Mise à jour de la mémoire à long terme M_{LT} à chaque instant t

Input : $M_{LT} = \{\{\mathbf{c}_v^x, \mathbf{c}_v^y, S_v\} | v = 1, \ldots, s\}$, le nombre maximum d'exemples stockés dans chaque échantillon r, le nouvel exemple $(\mathbf{x}_t, \mathbf{y}_t)$
Output : Mise à jour M_{LTM}
for $(\mathbf{c}_v^x, \mathbf{c}_v^y, S_v) \in M_{LT}$ **do**
  $d_v^y = distance(\mathbf{c}_v^y, \mathbf{y}_t)$
  $d_v^x = distance(\mathbf{c}_v^x, \mathbf{x}_t)$
end for
$b = argmin(d_v^y, v \in 1, ..., s)$ # L'indice du meilleur cluster
$p = argmin(d_v^x, v \in 1, ..., s)$ # L'indice du cluster prédit
if $d_b^y < 1.0$ **then**
  # Mettre à jour le meilleur cluster et le cluster prédit
  $(\mathbf{c}_b^x, \mathbf{c}_b^y) = (\mathbf{c}_b^x, \mathbf{c}_b^y) + \frac{1}{|S_b|} \times ((\mathbf{x}_t, \mathbf{y}_t) - (\mathbf{c}_b^x, \mathbf{c}_b^y))$
  if $b \neq p$ **then**
    $(\mathbf{c}_p^x, \mathbf{c}_p^y) = (\mathbf{c}_p^x, \mathbf{c}_p^y) - \frac{1}{|r|} \times ((\mathbf{x}_t, \mathbf{y}_t) - (\mathbf{c}_p^x, \mathbf{c}_p^y))$
  end if
  # utiliser la stratégie BRS
  $e = random(0, r)$ # génère une valeur aléatoirement dans l'intervalle $[0, r]$
  $S_b[e] = (\mathbf{x}_t, \mathbf{y}_t)$
else
  # Créer un nouveau cluster
  $(\mathbf{c}_{s+1}^x, \mathbf{c}_{s+1}^y) = (\mathbf{x}_t, \mathbf{y}_t)$
  $S_{s+1} = \{(\mathbf{x}_t, \mathbf{y}_t)\}$
  $M_{LT}.ajoute(\{\mathbf{c}_{s+1}^x, \mathbf{c}_{s+1}^y, S_{s+1}\})$
end if
return M_{LT}

l'espace des labels n'est inférieure à 1.0, alors le nouvel exemple est considéré comme trop éloigné des clusters existants, un nouveau cluster est donc créé et ajouté à M_{LTM} : le X-prototype $\mathbf{c}_{s+1}^x$ (resp . Y-prototype $\mathbf{c}_{s+1}^y$) est égale à $\mathbf{x}_t$ (resp. $\mathbf{y}_t$) et l'échantillon S_{s+1} est initialisé avec le nouvel exemple, $S_{s+1} = (\mathbf{x}_t, \mathbf{y}_t)$. Lors de l'initialisation de la phase d'apprentissage, le premier cluster de M_{LT} est créé à partir du premier exemple du flux.

Mise à jour des clusters. Le meilleur cluster C_b est mis à jour en insérant le nouvel exemple dans son échantillon avec la stratégie BRS et en rapprochant ses prototypes du nouvel exemple selon la formule 4.4. Il est important de noter que le taux d'apprentissage lr_b pour mettre à jour le prototype est fonction du nombre d'exemples stockés dans l'échantillon afin que les prototypes restent cohérents avec l'échantillon dans le même cluster, $lr_b = \frac{1}{|S_b|}$. Lors de l'initialisation, l'échantillon est vide, les prototypes sont initialisés avec le premier exemple et lr_b est égal à 1.0. Au fur et à mesure que le nombre d'exemples dans l'échantillon augmente, le taux d'apprentissage diminue jusqu'à atteindre sa valeur minimale de $1/r$. Lorsque le cluster prédit C_p est différent du meilleur cluster, les prototypes du cluster prédit sont éloignés du nouvel exemple selon la formule 4.5. Notons que lr_p est fixé à $1/r$ pour éviter que le processus d'éloignement ne perturbe le processus de rapprochement lors de l'initialisation.

Les poids w_{ST} et w_{LT} associés à chacune des mémoires pour la phase de prédiction sont actualisés selon la formule 5.2.

3.2 Complexité en temps de calcul

La complexité en temps de calcul sur un exemple pour A2ML est identique à celle d'ODM ($O((s+r+m) \times d + (s \times l).$), puisque le processus d'insertion d'A2ML garantit un nombre fini l de clusters dans M_{LT} (proposition 1).

Proposition 1 : A chaque instant, le nombre s de clusters dans la mémoire à long terme M_{LT} est inférieur ou égal au cardinal l des labels, $s \leq l$.

Preuve. Si un nouvel exemple à chaque instant t a au moins un label rencontré précédemment, alors la distance cosinus du vecteur de label d'au moins un cluster dans la M_{LT} est inférieure à 1.0, et le nouvel exemple est ajouté au meilleur cluster ; sinon, un nouveau cluster est créé. De cette façon, par conception, un nouveau cluster est créé uniquement lorsqu'au moins un nouveau label est rencontré. Le nombre de clusters s est donc limité par le nombre total de labels. Par conséquent, il n'est pas nécessaire d'introduire, comme c'est le cas dans d'autres approches de la littérature, un mécanisme de suppression de cluster qui ajoute de la complexité. La complexité de calcul de A2ML est donc de l'ordre de $O((l+r+m) \times d + l^2$.

4 Simulation des flux non stationnaires

Dans cette section, nous décrivons en détail notre nouveau protocole pour simuler les flux non-stationnaires avec différentes natures de dérive. Partant d'un jeu de données stationnaire $\mathcal{D}$, nous simulons des flux non-stationnaires en deux étapes :

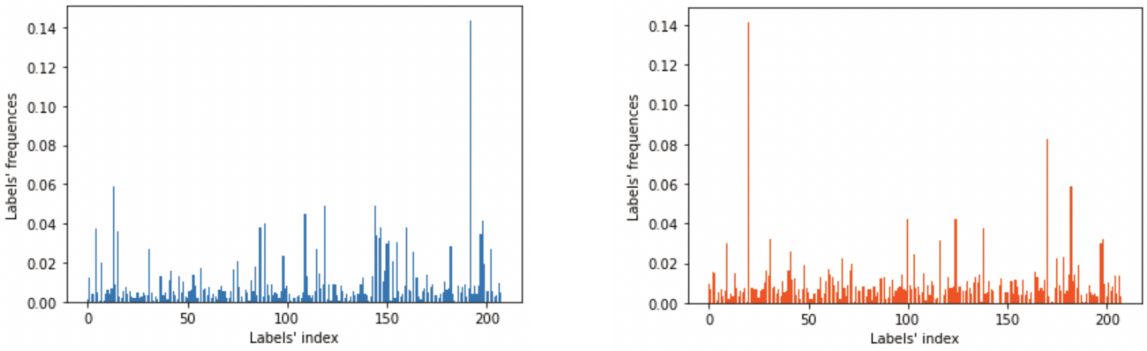
- 1) introduction de la dérive des concepts : division de $\mathcal{D}$ en deux sous-ensembles de même taille n mais avec des distributions différentes en changeant une composante de la distribution conjointe ;
- 2) génération des flux de données non-stationnaires : obtention de différents types de dérive en recombinant ensuite ces deux sous-ensembles.

4.1 Introduction de la dérive des concepts

Nous considérons ici les changements de la probabilité à priori $\mathbb{P}_t(\mathbf{y})$ et de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$ qui sont les principales causes de la dégradation de la performance de prédiction du modèle au flux de données :

- **La simulation du changement de la probabilité a priori** $\mathbb{P}_t(\mathbf{y})$ est basée sur le clustering dans l'ensemble des vecteurs de labels. Nous utilisons l'algorithme K-Means [118] pour partitionner tous les vecteurs de labels du jeux de données

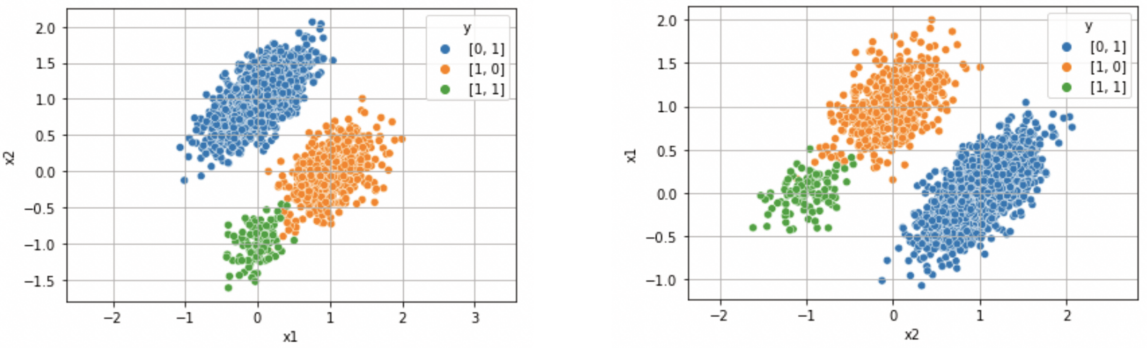
$\mathcal{D}$ en deux clusters différents. Les exemples dont les vecteurs de labels sont classés dans le même cluster sont ensuite regroupés dans le même sous-ensemble $\mathcal{D}_i$, $i = 1, 2$. L'ordre des exemples dans chaque sous-ensemble est ensuite randomisé pour garantir que chaque sous-ensemble est distribué de manière identique et indépendante. Les flux résultants $\mathcal{D}_1$ et $\mathcal{D}_2$ ont différentes probabilités a priori $\mathbb{P}_1(\mathbf{y})$ et $\mathbb{P}_2(\mathbf{y})$ tandis que $\mathbb{P}(\mathbf{x}|\mathbf{y})$ reste constant. Un exemple du clustering sur les vecteurs de labels du jeu de données Bookmarks est donnée dans la Figure 5.2.



(a) $\mathbb{P}_1(\mathbf{y})$ *du sous-ensemble* $\mathcal{D}_1$ (b) $\mathbb{P}_2(\mathbf{y})$ *du sous-ensemble* $\mathcal{D}_2$

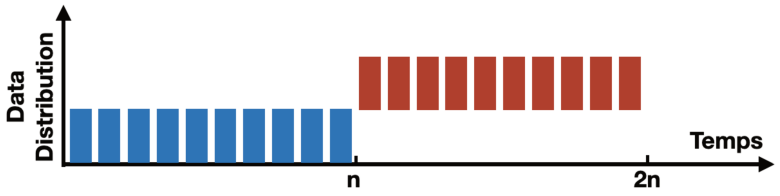
FIGURE 5.2 – Simulation du changement de la probabilité a priori $\mathbb{P}_t(\mathbf{y})$ avec le jeu de données Bookmarks.

— **La simulation du changement de la probabilité conditionnelle** $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$ est basée sur l'échange des valeurs d'attributs [119] mais en gardant les mêmes vecteurs de labels. Par exemple, pour chaque exemple $(\mathbf{x}_t, \mathbf{y}_t)$, la valeur originale de x_t^1 remplace x_t^2, la valeur originale de x_t^2 remplace x_t^3, et ainsi de suite, tandis que la valeur du dernier attribut x_t^d remplace x_t^1 ; les vecteurs de labels restent les même. Dans un jeu de données avec un espace d'attributs à deux dimensions et deux labels, la distribution des données change comme le montre la Figure

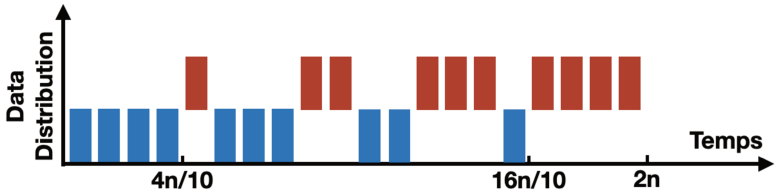


(a) $\mathbb{P}_1(\mathbf{x}, \mathbf{y})$ *originale* (b) $\mathbb{P}_2(\mathbf{x}, \mathbf{y})$ *après l'échange des valeurs d'attributs*

FIGURE 5.3 – Simulation du changement de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}, \mathbf{y})$ pour un jeu de données avec un espace d'attributs à deux dimensions et deux labels.



(a) Le flux non-stationnaire avec une dérive abrupte.



(b) Le flux non-stationnaire avec une dérive graduelle.

FIGURE 5.4 – Simulation d'un flux non-stationnaire avec une dérive, passant du concept de $\mathcal{D}_1$ au concept de $\mathcal{D}_2$.

5.3. Dans la pratique, nous partitionnons d'abord $\mathcal{D}$ en deux sous-ensembles $\mathcal{D}'_1$ et $\mathcal{D}'_2$ ayant la même distribution, puis nous permutons aléatoirement les valeurs d'attributs des exemples de $\mathcal{D}'_1$ en utilisant l'algorithme de Fisher-Yates [120], tout en préservant leurs labels. Les flux résultants $\mathcal{D}'_1$ et $\mathcal{D}'_2$ ont ainsi des probabilités conditionnelles de classes différentes $\mathbb{P}_1(\mathbf{x}|\mathbf{y})$ et $\mathbb{P}_2(\mathbf{x}|\mathbf{y})$, tandis que $\mathbb{P}(\mathbf{y})$ reste constant.

4.2 Génération de différents types de dérives

Nous utilisons trois méthodes différentes pour simuler les flux non stationnaires avec les trois différents types de dérives identifiés dans le chapitre 2. Les simulations d'une dérive (i) abrupte et (ii) graduelle sont en fait les plus simples et ne nécessitent qu'une simple recombinaison de deux sous-ensembles de données différents comme le montre la Figure 5.4. Les simulations de dérive (iii) incrémentale sont un peu plus complexes et nécessitent la création de concepts intermédiaires entre le premier et deuxième concept (voir la Figure 5.6).

— (i) **Dérive abrupte**. Lors d'une simulation de dérive abrupte, sur la période $[0, n]$, toutes les données proviennent d'un même sous-ensemble et sont distribués de manière identique et indépendante. Sur la période $[n, 2n]$, toutes les données proviennent d'un autre sous-ensemble. Donc dans les flux de données non stationnaires avec une dérive abrupte, la durée de la dérive est nulle.

— (ii) **Dérive graduelle**. Lors d'une simulation de dérive graduelle, sur les périodes $\{[0, \frac{4n}{10}], [\frac{5n}{10}, \frac{8n}{10}], [n, \frac{12n}{10}], [\frac{15n}{10}, \frac{16n}{10}]\}$, toutes les données proviennent d'un même sous-ensemble. Sur les périodes $\{[\frac{4n}{10}, \frac{5n}{10}], [\frac{8n}{10}, n], [\frac{12n}{10}, \frac{15n}{10}], [\frac{16n}{10}, 2n]\}$, toutes les

données proviennent d'un autre sous-ensemble. En conséquence, dans les flux de données non stationnaires avec une dérive graduelle, la durée de la dérive est de $6n/5$.

— (iii) **Dérive incrémentale**. Pour simuler la dérive incrémentale, des concepts intermédiaires entre le premier et le deuxième concept doivent d'abord être créés. Le concept intermédiaire peut être considéré comme un mélange de différentes proportions du premier et du deuxième concepts. Le flux non-stationnaire avec la dérive incrémentale $\mathcal{D}_{ns}$ peut être une combinaisons de tous les ensembles de données avec des concepts distincts disposés séquentiellement. En effet, au cours du temps, la différence entre les concepts intermédiaires et le premier concept augmente alors que la différence entre les concepts intermédiaires et le deuxième concept diminue. En pratique, nous créons cinq sous-ensembles intermédiaires $\mathcal{D}^i_{inter}, i \in [1, 5]$ avec différentes proportions de $\mathcal{D}_1$ et $\mathcal{D}_2$, dont les ratios sont $\{\frac{5}{6}, \frac{1}{6}\}$, $\{\frac{4}{6}, \frac{2}{6}\}$, $\{\frac{3}{6}, \frac{3}{6}\}$, $\{\frac{2}{6}, \frac{4}{6}\}$ et $\{\frac{1}{6}, \frac{5}{6}\}$, respectivement. En supposant que les premier et deuxième concepts suivent une distribution gaussienne, alors les 5 concepts intermédiaires - les distributions de chacun des 5 sous-ensembles intermédiaires - évoluent dans le temps comme le montre la Figure 5.5. Sur la Figure 5.6, lors de la dérive incrémentale sur notre flux, le premier et le deuxième concept durent $\frac{5}{8}n$ chacun et chaque concept intermédiaire dure $\frac{3}{20}n$ de sorte que la dérive se produit à l'instant $t = \frac{5}{8}n$ et la durée de la dérive est égale à $\frac{3}{4}n$.

Nous pouvons également générer des flux non-stationnaires avec deux dérives basées sur la ré-émergence du premier concept après le deuxième qui dure une période donnée pour simuler des concepts récurrents. Par exemple, nous pouvons simuler un flux non-stationnaire avec deux dérives abruptes, passant du concept de $\mathcal{D}_1$ au concept de $\mathcal{D}_2$ à l'instant n, puis remplaçant le concept de $\mathcal{D}_2$ par le concept de $\mathcal{D}_1$ à l'instant $2n$, comme le montre la Figure 5.7.

4.3 Génération de flux de données non-stationnaires à partir des jeux de données réels

Le jeu de données initial $\mathcal{D}$ choisi est Bookmarks [121] car il est de taille suffisamment grande ($80,000$ exemples sont utilisés) pour simuler un flux et qu'il existe une partition de sa distribution en deux sous-ensembles équilibrés ($n = 40,000$) et suffisamment différents. En modélisant la changement de la probabilité a priori, nous avons obtenu la bipartition $\{\mathcal{D}_1, \mathcal{D}_2\}$ sur $\mathcal{D}$. Avec cette bipartition $\{\mathcal{D}_1, \mathcal{D}_2\}$, nous avons généré trois flux non-stationnaires où une dérive est produite par le remplacement du premier concept $\mathcal{D}_1$ par le deuxième $\mathcal{D}_2$ selon les trois modalités (abrupte, graduelle, et incrémentale). Nous avons également généré un quatrième flux non-stationnaire de $120,000$ exemples avec deux dérives abruptes : la première dérive se produit à l'instant $t = n$ avec le remplacement du

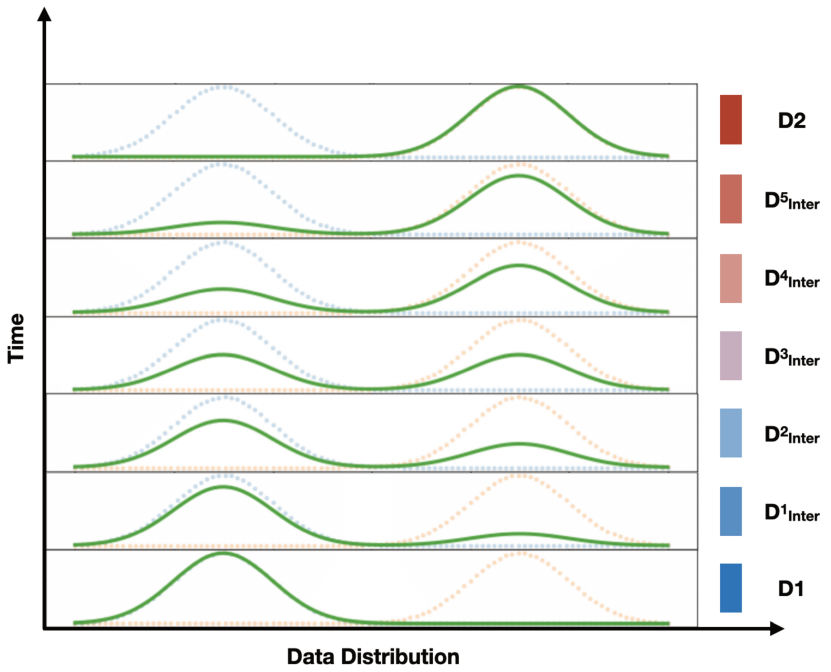


FIGURE 5.5 – Simulation des concepts intermédiaires pour la dérive incrémentale, passant du concept de $\mathcal{D}_1$ au concept de $\mathcal{D}_2$. Chaque couleur représente un concept d'un sous-ensemble de données.

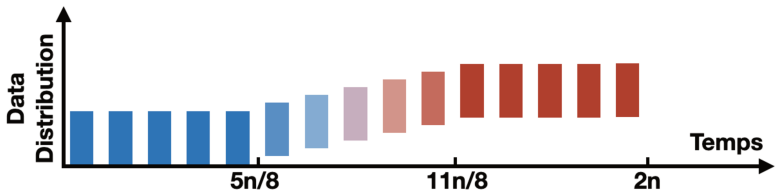


FIGURE 5.6 – Simulation d'un flux non-stationnaire avec une dérive incrémentale, passant du concept de $\mathcal{D}_1$ au concept de $\mathcal{D}_2$, qui contient 5 concepts intermédiaires pendant la dérive.

premier concept par le second et la deuxième dérive se produit à l'instant $t = 2 \times n$ avec la réapparition du premier concept qui remplace alors le second. Le quatrième flux de données non-stationnaire est conçu pour tester l'adaptabilité de l'algorithme aux concepts récurrents.

En modélisant la changement de la probabilité conditionnelle, nous avons obtenu ainsi la bipartition $\{\mathcal{D}'_1, \mathcal{D}'_2\}$ sur $\mathcal{D}$ et nous avons généré également quatre flux non-stationnaires avec $\{\mathcal{D}'_1, \mathcal{D}'_2\}$.

5 Comparaisons expérimentales

Dans cette section, nous comparons ODM et A2ML avec deux algorithmes de l'état de l'art représentatifs des meilleures performances obtenues dans les comparaisons menées en Section 4.1 : un basé sur une stratégie d'adaptation (HTps) et un algorithme Ensemble

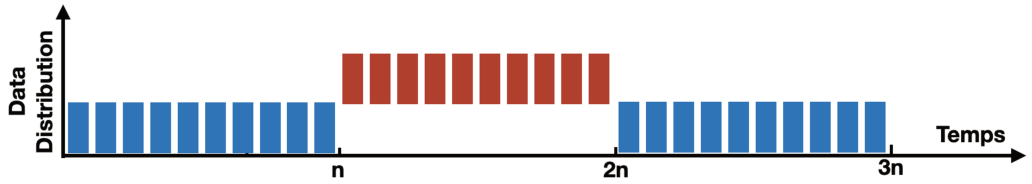


FIGURE 5.7 – Simulation d'un flux non-stationnaire avec deux dérive abruptes, passant du concept de $\mathcal{D}_1$ au concept de $\mathcal{D}_2$ à l'instant n, puis remplaçant le concept de $\mathcal{D}_2$ par le concept de $\mathcal{D}_1$ à l'instant $2n$.

(EaHTps). Les codes d'ODM [1] et d'A2ML [2] sont tous disponibles sur Github.

Les expérimentations ont été également effectuées selon le protocole d'évaluation *Interleaved Test-Then-Train*. La qualité des résultats expérimentaux a été évaluée avec le critère *Accuracy*.

Nous avons d'abord vérifié les performances d'ODM et d'A2ML pour les 15 flux stationnaire présentés dans la Section 3. L'objectif est de s'assurer que les mécanismes introduits pour gérer les dérives n'entraînent pas de dégradation des performances dans les scenarii stationnaires. Puis nous avons évalué les capacités des algorithmes à s'adapter à la dérive sur huit flux non-stationnaires (quatre avec un changement de la probabilité a priori et quatre avec un changement de la probabilité conditionnelle) que nous avons générés dans le section 4.3.

5.1 Choix des hyperparamètres pour ODM et A2ML

Le contrôle des hyperparamètres dans les scenarii non stationnaires reste une question délicate qui a encore été peu étudiée dans la littérature. Nous avons adopté ici l'hypothèse simpliste selon laquelle la valeur optimale des hyperparamètres pour un modèle sélectionné off-line sur un petit nombre d'exemples reste optimale même en cas de dérives de concepts. En fixant les valeurs des hyperparamètres, nous pouvons nous assurer que les complexités de la mémoire et du calcul du modèle restent bornées.

ODM et A2ML partagent trois hyper-paramètres en commun : la taille m de la mémoire à court terme M_{CT}, le nombre k des plus proches voisins considérés lors de la phase de prédiction, et la taille r des échantillons associés aux clusters stockés dans la mémoire à long terme M_{LT}. Pour le nombre de clusters s dans M_{LT}, ODM et A2ML sont considérés différemment. Pour ODM, s est également un hyperparamètre qui doit être optimisé. En revanche, dans A2ML, s s'auto-ajuste au fur et à mesure que la distribution des labels change et nous avons montré dans la proposition 1 qu'il restait inférieur ou

1. ODM source code : https ://github.com/Cici-xihui/ODM
2. A2ML source code : https ://github.com/Cici-xihui/A2ML

égal au cardinal l des labels.

Pour ODM : La valeur de m est identique à celle de l'algorithme MLSAMPKNN [94] où la mémoire à court terme joue un rôle similaire. Cette valeur doit être suffisamment petite (ici $m = 50$) pour faciliter un ajustement rapide lorsqu'un nouveau concept arrive, tout en étant assez grande pour estimer certaines statistiques du flux. Pour les trois hyperparamètres dans la mémoire à long terme M_{LT}, nous avons pris les mêmes valeurs optimales que celles issues des expérimentations du chapitre 4 : $k = 3$; $s = 40$ et $r = 100$.

Pour A2ML : La valeur de m est la même que dans ODM parce que la mémoire à court terme d'A2ML joue le même rôle. La valeur de k a été fixée à 3 selon les préconisations des méthodes kNN multi-labels [94] [95]. La valeur de r a été déterminée par la même procédure d'optimisation que celle présentée dans la section 2.6 et a été testée sur les 15 flux de données du Tableau 2. La performance d'A2ML a été évaluée avec 6 valeurs de r : 25, 50, 100, 200, 400 et 800.

Taille d'échantillon	$r = 25$	$r = 50$	$r = 100$	$r = 200$	$r = 400$	$r = 800$
Accuracy	0.323	0.336	0.345	0.348	0.350	0.352

TABLEAU 5.1 – Évolution des performances d'A2ML en fonction de la taille des échantillons.

Les résultats du Tableau 5.1 indiquent que les performances augmentent lorsque r augmente puis se stabilisent à partir de $r = 100$ qui est la valeur retenue pour un bon compris entre la qualité des résultats et la complexité en temps de calcul.

Pour HTps et EaHTps : Les valeurs des paramètres des autres algorithmes sont celles recommandées par leurs auteurs dans les références du Tableau 3.1.

5.2 Résultats dans le cas stationnaire

Les résultats des comparaisons d'ODM et A2ML avec les deux algorithmes de l'état de l'art sont détaillés dans le tableau 5.2. Le test de Friedman et le test post hoc de Nemenyi ont également été appliqués avec un même niveau de confiance égal à 0.05. Le test de Friedman avec une p-value égale à 0.002 confirme le rejet de l'hypothèse nulle selon laquelle les performances de toutes les méthodes sont égales. L'analyse post-hoc de Nemenyi (Figure 5.8) confirme que le classement moyen d'A2ML est le meilleur sur l'ensemble des algorithmes. Des variations existent cependant dans les comparaisons par paires : la différence entre A2ML et ODM n'est pas statistiquement significative mais leurs performances sont significativement meilleures que celles de HTps et EaHTps. Les résultats détaillés montrent qu'ODM est en première position pour 5 flux sur 15 et en

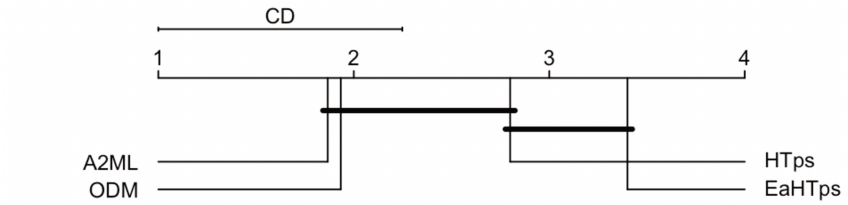


FIGURE 5.8 – Test post hoc de Nemenyi sur les *Accuracy* moyennes des 4 algorithmes.

deuxième position pour 8 autres flux, tandis que A2ML est en première position pour 8 flux sur 15 et en deuxième position pour 4 autres flux.

Datasets	HTps	EaHTps	ODM	A2ML
20NG	32.3	27.1	60.9	**64.6**
Bibtex	12.4	11.1	22.3	**29.6**
Bookmarks	18.2	17.6	23.2	**26.6**
Cooking	4.9	0.3	**15.6**	15.1
Corel16k001	8.1	4.8	**11.2**	9.0
Enron	17.4	17.3	**33.2**	31.2
Eukaryote	**28.4**	**28.4**	25.5	23.1
Human	28.5	28.2	26.9	**30.9**
Imdb	**21.0**	**21.0**	15.4	13.0
Mediamill	32.6	33.7	**38.8**	30.0
Reuters-K500	21.7	19.6	37.3	**39.8**
Scene	38.7	7.9	52.3	**67.3**
SLASHDOT	13.2	12.2	33.6	**41.6**
TMC2007	41.5	43.0	**54.1**	44.1
Yeast	41.7	39.8	42.5	**44.6**
avg. value	24.04	20.80	32.85	**34.03**
avg. rank	2.80	3.40	**1.93**	**1.87**

TABLEAU 5.2 – *Accuracy* moyenne (%) des 4 algorithmes sur 15 flux de données stationnaires.

De plus, la comparaison des temps de calculs moyens confirme que ODM et A2ML maintiennent une complexité de temps de calcul plus faible que les autres algorithmes (voir tableau 5.3).

	HTps	EaHTps	ODM	A2ML
Temps en calcul	194.32	2230.14	32.78	26.71

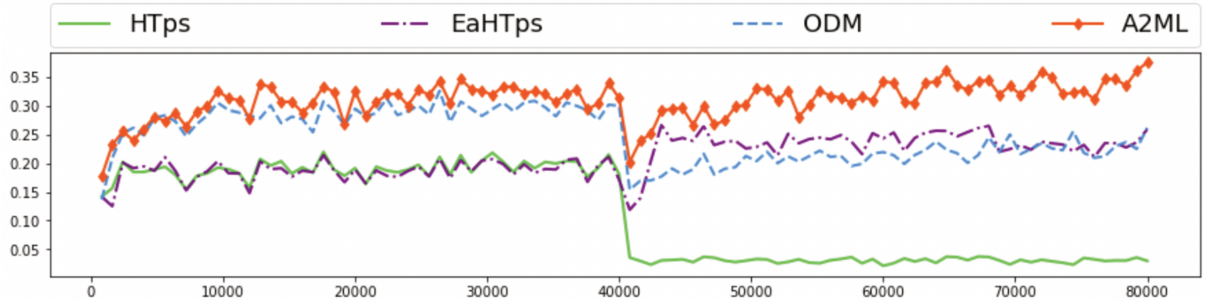
TABLEAU 5.3 – Temps de calcul moyen des 4 algorithmes sur 15 flux stationnaires en secondes

Dans l'expérimentation stationnaire, nous constatons donc que ODM et A2ML présentent un très haut niveau de performance tout en maintenant une faible complexité. Ces résultats rassurants nous confirment l'intérêt de poursuivre leur évaluation sur des flux de données non stationnaires.

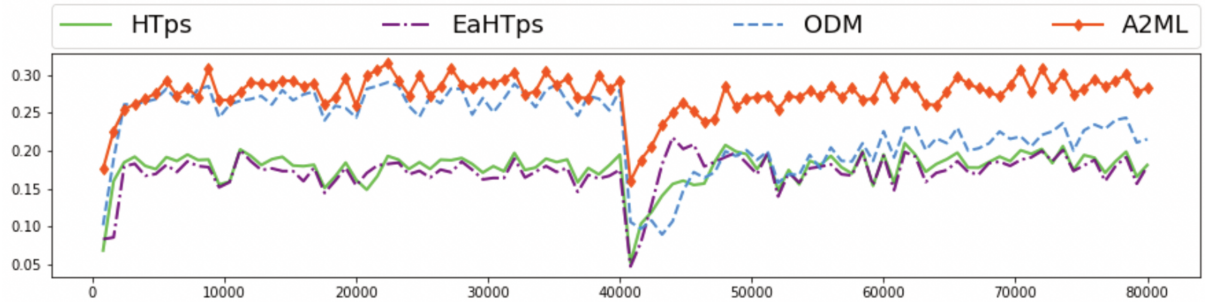
5.3 Résultats dans le cas non stationnaire

Dans cette section, nous présentons les résultats des 4 algorithmes précédents pour les flux du scénario non-stationnaire.

5.3.1 Flux non stationnaires avec une dérive abrupte



(a) Avec changement de la probabilité a priori $\mathbb{P}_t(\mathbf{y})$



(b) Avec changement de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$

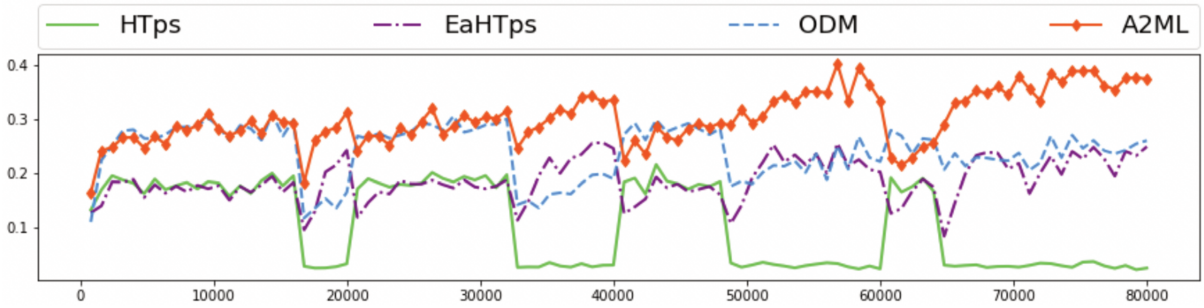
FIGURE 5.9 – Comparaison des performances (*Accuracy*) pour les flux de 80, 000 exemples avec une dérive abrupte.

La figure 5.9 présente les évaluations des performances des quatre algorithmes sur deux flux non-stationnaires avec une dérive abrupte, où la dérive est causée par le changement de $\mathbb{P}_t(\mathbf{y})$ (Figure 5.9a) et le changement de $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$ (Figure 5.9b), respectivement.

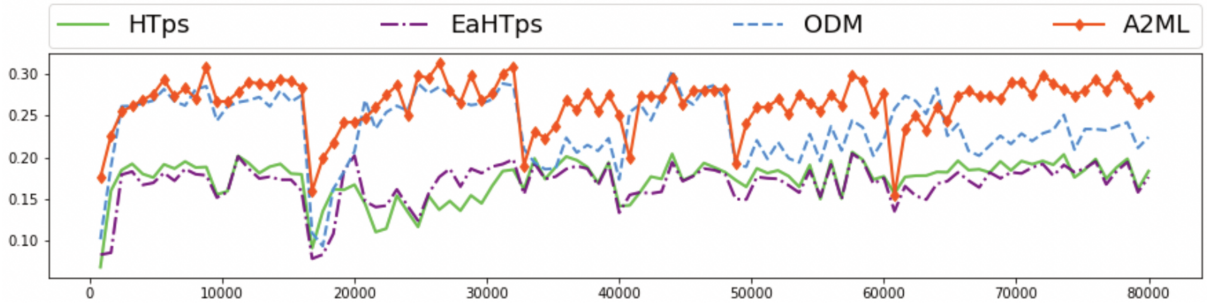
Les Figures montrent que les performances de HTps se détériorent tandis que celles d'EaHTps, d'ODM et d'A2ML s'adaptent à l'apparition d'un nouveau concept, lorsque $\mathbb{P}_t(\mathbf{y})$ change, alors que tous les algorithmes subissent une perte de performance et s'adaptent tous rapidement au changement de $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$. Après la dérive abrupte, sur la base de la pente ascendante de la courbe, ces deux Figures confirment que l'algorithme EaHTps s'adapte le plus rapidement à la dérive abrupte, suivi par A2ML. A2ML conserve de meilleures performances prédictives avant et après la dérive. Les valeurs moyennes des performances pour HTps, EaHTps, ODM et A2ML pour ces deux flux sont respectivement de 14.35%, 19.02%, 23.78% et 29.23%.

5.3.2 Flux non stationnaires avec une dérive graduelle

La figure 5.10 présente les évaluations des performances des quatre algorithmes sur deux flux non-stationnaires avec une dérive graduelle.



(a) Avec changement de la probabilité a priori $\mathbb{P}_t(\mathbf{y})$



(b) Avec changement de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$

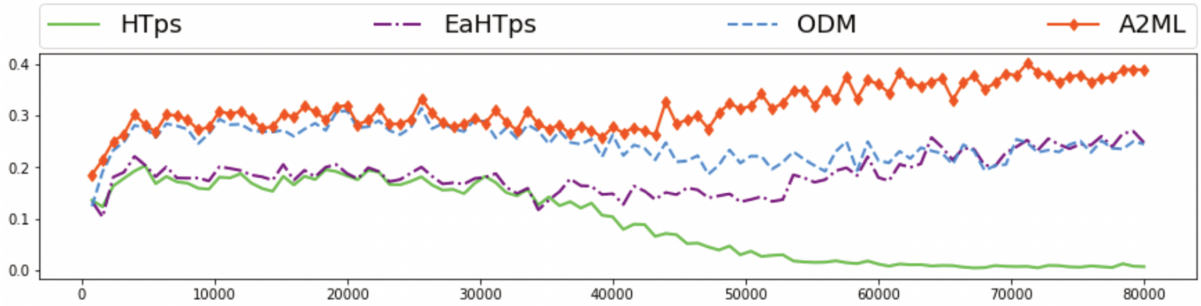
FIGURE 5.10 – Comparaison des performances (*Accuracy*) pour les flux de 80,000 exemples avec une dérive graduelle.

La Figure 5.10a confirme que HTps ne peut pas s'adapter à une dérive due au changement de $\mathbb{P}_t(\mathbf{y})$, mais qu'il peut s'adapter à une dérive due au changement de $\mathbb{P}_t(\mathbf{x}, \mathbf{y})$. En revanche, les autres algorithmes peuvent s'adapter aux deux causes de dérive conceptuelle. De plus, nous notons que tous les algorithmes ne subissent pas de grosse perte de performance sur les flux de données avec dérive graduelle. Les valeurs moyennes des performances pour HTps, EaHTps, ODM et A2ML pour ces deux flux sont respectivement de 13.33%, 17.79%, 23.74% et 29.81%. Nous pouvons également observer que la performance prédictive globale d'A2ML est plus élevée que dans les cas abrupts : A2ML s'adapte mieux à un nouveau concept dans le cas de la dérive graduelle.

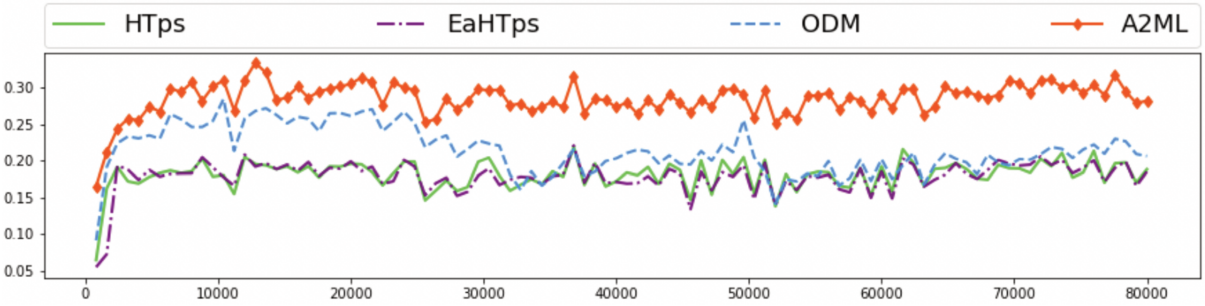
5.3.3 Flux non stationnaires avec une dérive incrémentale

La Figure 5.11 présente les évaluations des performances des quatre algorithmes sur deux flux non-stationnaires avec une dérive incrémentale.

Les courbes sur la Figure 5.11 sont beaucoup plus « lisses » que les courbes des Figures 5.9 et 5.10. Avec le changement de la probabilité à priori $\mathbb{P}_t(\mathbf{y})$, nous pouvons observer que la tendance globale des courbes de performances d'EaHTps et d'A2ML est à la hausse et que la tendance globale de HTps et d'ODM est à la baisse. Mais sur le flux avec le



(a) Avec changement de la probabilité a priori $\mathbb{P}_t(\mathbf{y})$



(b) Avec changement de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$

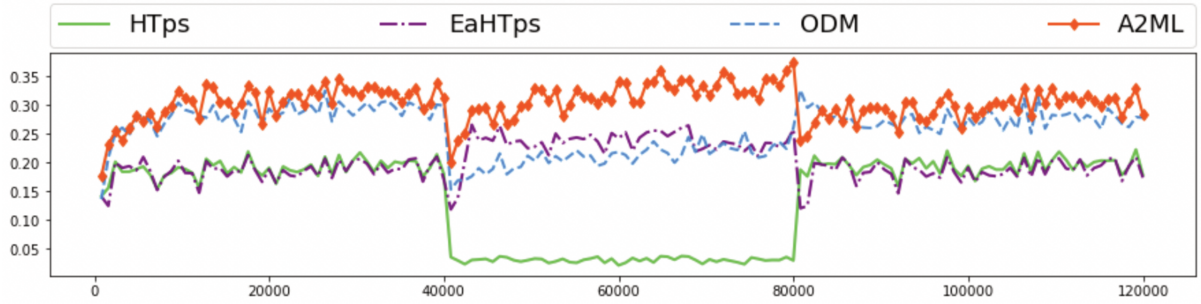
FIGURE 5.11 – Comparaison des performances (*Accuracy*) pour les flux de 80,000 exemples avec une dérive incrémentale.

changement de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$, ces quatre algorithmes s'adaptent à la dérive et ne perdent pas en performance prédictive. Les valeurs moyennes des performances pour HTps, EaHTps, ODM et A2ML pour ces deux flux sont respectivement de 14.33%, 18.23%, 23.31% et 30.23%.

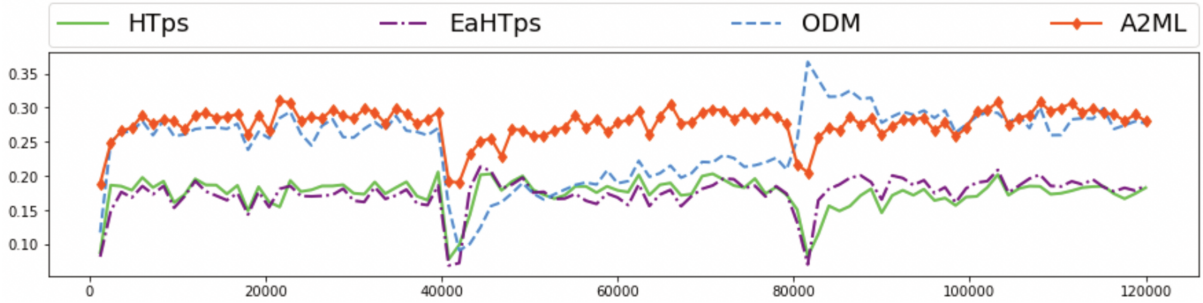
5.3.4 Flux non stationnaires avec deux dérives abruptes

La figure 5.12 présente les évaluations des performances des quatre algorithmes sur deux flux non-stationnaires avec deux dérives abruptes où la deuxième dérive est due à la récurrence du premier concept.

Nous constatons que tous les algorithmes s'adaptent rapidement et stabilisent leurs performances prédictives lorsqu'ils sont confrontés à des concepts récurrents dans la deuxième dérive. Plus précisement, EaHTps a les performances les plus dégradées lors que l'ancien concept réapparaît. En revanche, ODM ne subit aucune perte et obtient récupère un niveau de prédiction élevé. Ces comportements sont directement liés aux stratégies mises en œuvre. EaHTps est un ensemble de modèles qui adapte la dérive en supprimant l'ancien modèle et en ajoutant un nouveau modèle : il a donc une faible mémoire pour les informations sur les distributions passées. En revanche, l'adaptation d'ODM à de nouveaux concepts diminue progressivement à cause de ses mécanismes de l'apprentissage et sa mémoire est davantage conservée dans la distribution du passé. En comparaison, A2ML s'adapte non seulement rapidement aux nouveaux concepts, mais



(a) Avec changement de la probabilité a priori $\mathbb{P}_t(\mathbf{y})$



(b) Avec changement de la probabilité conditionnelle $\mathbb{P}_t(\mathbf{x}|\mathbf{y})$

FIGURE 5.12 – Comparaison des performances (*Accuracy*) pour les flux de 120,000 exemples avec deux dérives abruptes : pour la première, le second concept remplace le premier, et pour la deuxième, le premier concept réapparaît et remplace le premier.

aussi plus rapidement que ses concurrents aux anciens. Les valeurs moyennes des performances pour HTps, EaHTps, ODM et A2ML pour ces deux flux sont respectivement de 15.83%, 18.85%, 25.28% et 29.07%.

6 Conclusion

Dans ce chapitre nous avons présenté deux nouvels algorithmes qui ont été développés séquentiellement pour pouvoir maintenir un apprentissage de qualité sur des flux de données variés –stationnaires et non stationnaires avec des types de dérives différents- tout en maintenant une complexité réduite en mémoire et en temps de calcul. Les deux algorithmes sont basés sur la combinaison de deux mémoires : une mémoire à court terme capable d'adapter le modèle à des changements brutaux, et une mémoire à long terme qui vise à une adaptation aux variations des propriétés du flux sur des temporalités plus grande. Les expérimentations menées avec le premier algorithme ODM (Online Dual Memory) ont montré l'intérêt d'un agencement de deux mémoires. Cependant, l'analyse du comportement de l'algorithme a permis d'identifier une faiblesse qui est la dégradation de ses capacités d'adaptation dans le temps. Pour surmonter cette limite, nous avons développé un deuxième algorithme A2ML (Adaptive Memories for Multi-Label stream classification) qui conserve le principe général de combinaison des mémoires qui a été expérimentalement éprouvé tout en différant significativement d'ODM par la gestion de

la mémoire à long terme. A2ML met à jour le modèle avec un échantillonnage biaisé et la gestion de la mémoire prend en compte la variabilité des distributions en créant des nouveaux clusters associés à des nouveaux exemples dont les vecteurs de labels sont différents de ceux qui ont été préalablement mémorisés. Nous avons montré que la gestion spécifique de ces nouveaux clusters dans la mémoire était compatible avec des ressources limitées en mémoire.

En complément, ce chapitre a également présenté une avancée méthodologique pour les comparaisons expérimentales sur les flux de données non stationnaires. Nous avons introduit ici un nouveau protocole expérimental qui permet de mieux identifier les comportements des algorithmes testés. Les résultats expérimentaux avec ce nouveau protocole qui complètent ceux obtenus dans le cas stationnaires sur des données classiques de la littérature confirment l'intérêt de notre nouvel algorithme A2ML. Il permet un apprentissage adaptatif efficace quelles que soient les variations introduites dans le flux tout en maintenant une complexité en temps significativement plus faible que les algorithmes concurrents.

Chapitre 6

Conclusion et perspectives

Sommaire

Cette thèse a contribué à l'analyse du problème de la classification multi-label en flux. Les informations considérées pour l'apprentissage dans ce cadre ont des spécificités : chaque exemple arrive en séquence avec une vitesse élevée, le nombre total d'exemples dans le flux peut être très grand voire théoriquement illimité, et la distribution des données peut évoluer au cours du temps. De plus, des contraintes de ressources en mémoire peuvent être ajoutées au problème ; ce qui empêche le stockage permanent de toutes les données rencontrées. Contrairement à l'abondante littérature consacrée à la classification traditionnelle multi-labels et la classification mono-label en flux, les travaux consacrés à la classification multi-labels en flux sont plus récents et plusieurs questions restent largement ouvertes. En premier lieu, la modélisation du problème intégrant les caractéristiques de la dynamique des flux, et notamment les différentes modalités des dérives de concept, reste souvent assez succinte. De plus, des contraintes de ressources nécessitent de traiter un exemple avant sa disparition avec une faible complexité de calcul et les variations de distributions nécessitent une adaptation « continue » sans reconstruction permanente du modèle. Enfin, plus globalement, on retrouve aussi l'importance de la prise en compte des corrélations entre l'espace des labels et l'espace de sortie en grande dimension, mais la diversité plus importante des données au cours de l'évolution temporelle peut rendre l'apprentissage plus complexe que dans le cas statique.

1 Contributions principales

En sus d'un état de l'art approfondi, nos contributions à ce problème se sont donc organisées selon deux directions complémentaires : une modélisation précise des problèmes considérés associée à un nouveau protocole expérimental pour l'évaluation des algorithmes de classification multi-labels en flux, et la proposition d'un nouvel algorithme qui s'appuie sur l'exploration de deux précédentes propositions basées sur des combinaisons de mémoires.

Plus précisément, nous avons dissocié les cas des flux stationnaires et non stationnaires intégrant des variations dans la distribution des données. Cela nous a conduit à compléter l'état de l'art où les algorithmes sont généralement regroupés en trois familles (méthodes par transformation, adaptation et méthodes ensemble), en distinguant les deux stratégies principales d'adaptation aux dérives de concept : les approches actives et passives. Cet état des lieux nous a permis de préciser deux besoins majeurs découlant des limites des approches existantes : la nécessité d'équilibrer la complexité en temps de calcul et la performance du modèle, et la nécessité d'équilibrer la plasticité et la stabilité de l'algorithme sur le flux.

Dans un premier temps nous nous sommes concentrés sur les performances du modèle sous la contrainte d'une faible complexité temporelle, et nous nous sommes restreints aux flux stationnaires. Nous avons ainsi proposé une nouvelle approche, MLT-ML (*Mémoire à Long Terme pour la classification Multi-Labels en flux*) basée sur une stratégie de regroupement des exemples multi-labels basée sur une similarité des combinaisons de labels inspirée de l'algorithme CraftML, qui est l'un des meilleurs de l'état de l'art dans la classification multi-label traditionnelle. La version adaptée aux flux de données a introduit une extension de l'algorithme LVQ (*Learning Vector Quantization*) avec un échantillonnage dans chaque sous-espace permettant un stockage aléatoire des exemples avec une mémoire limitée. Les résultats expérimentaux ont montré que MLT-ML est pertinent pour trouver un équilibre entre une performance prédictive du modèle élevé et une complexité en calcul faible.

Ensuite, notre recherche s'est étendue aux flux non stationnaires. Deux nouvelles méthodes ont été proposées dans cette thèse, chacune basée sur une combinaison de deux mémoires, une à court-terme et l'autre à long-terme. Notre architecture basée sur l'interaction de ces deux mémoires a permis une adaptation à différents types de dérives. Plus précisément, la première approche ODM (*Online Dual Memory*) combine MLT-ML avec une mémoire à court-terme basée sur une fenêtre glissante. Bien qu'elle puisse s'adapter à la dérive pendant un certain temps et qu'elle ait une faible complexité temporelle, sa capacité d'adaptation diminue graduellement avec le temps. Un autre inconvénient est que sa performance est fortement affectée par l'initialisation, ce qui peut l'empêcher de s'adapter à certaines dérives causées par la distribution des labels. Nous avons donc proposé une

deuxième approche, A2ML (*Adaptative Memories for Multi-Label stream classification*) qui contourne les limitations d'ODM, tout en conservant ses avantages. Les comparaisons expérimentales avec les états de l'art ont confirmé la haute performance prédictive d'A2ML sur les flux stationnaires et les flux non-stationnaires. Nos résultats ont montré qu'A2ML réussit à trouver un compromis entre la plasticité et la stabilité sur le flux.

Le travail expérimental pour l'évaluation des algorithmes s'est appuyé sur un état de l'art des protocoles utilisés dans la littérature dans le cas de la classification multi-labels en flux. Ce dernier a montré que le cadre non stationnaire n'avait pas encore suffisamment gagné en maturité pour avoir abouti à des protocoles partagés par toute la communauté de chercheurs. De plus, les cadres expérimentaux existants ne permettent pas d'identifier explicitement les caractéristiques des différents types de dérives. Nous avons donc contribué à cette réflexion expérimentale en proposant un nouveau modèle pour la simulation de flux non stationnaires. Ce protocole permet, en réutilisant des jeux de données existants, de simuler et de contrôler différents types de dérives de concept : abrupte, graduelle et incrémentale. Il ouvre ainsi la voie à des analyses plus précises sur le comportement des algorithmes faces aux variations de distributions que l'on peut rencontrer dans les flux de données réels.

2 Perspectives

Nos travaux ouvrent des perspectives dans deux directions : algorithmique et applicative.

2.1 Perspectives algorithmiques

Une première direction de recherche concerne l'extension de l'algorithme A2ML pour traiter des problèmes de classification extrême multi-labels en flux qui considère des données avec un nombre de variables (attributs et labels) extrêmement grand. En effet, dans certaines applications, la dimension des données continue de fortement augmenter. Par exemple, dans le récent challenge Kaggle [1] portant sur la catégorisation automatique d'articles de Wikipédia, le corpus étant composé d'environ 2,4 millions d'articles. A partir du contenu des articles, encodés comme des sacs de n-grams avec 1,6 millions d'attributs, le but est de déterminer automatiquement la ou les catégories de Wikipédia associées parmi environ 350, 000 possibilités. Lorsque l'on dispose de ressources en mémoire et en calcul limitées - posées par des considérations d'accessibilité de l'IA ou des considérations écologiques-, ces dimensions qualifiées d'extrêmes se heurtent aux limites de l'usage des algorithmes « en batch » classiques.

1. https ://www.kaggle.com/c/extreme-classification-dataset1

Pour contourner ces limites, les données peuvent être considérées comme des flux de données : l'apprentissage est alors effectué « on-line » donnée par donnée et en une passe de calcul sans stockage de toutes les données en RAM. La stratégie déployée dans A2ML est prometteuse mais elle nécessite cependant d'être complétée par une technique de réduction de dimension. Or les approches les plus efficaces visent essentiellement aujourd'hui les données statiques off-line. Les scenarii en flux peuvent comporter des changements statistiques qui ont un impact dans la réduction de dimension et cette prise en compte nécessite des recherches plus approfondies.

Une deuxième direction de recherche concerne l'extension de A2ML pour une adaptation efficace à l'évolution des attributs ou des labels [35]. Dans les flux de données réels, il est courant que des attributs ou des labels apparaissent ou disparaissent au fil du temps. Si un nouvel attribut (resp. label) devient disponible en cours du temps et s'il est considéré comme pertinent, on peut supposer qu'une évolution des attributs (resp. labels) s'est produite ; cet attribut (resp. label) peut alors être intégré au processus d'apprentissage. De même, si un attribut (resp. label) devient indisponible, toutes ses valeurs peuvent être considérées comme manquantes et le modèle d'apprentissage doit alors ignorer son existence. Mais actuellement, il est encore très difficile de s'adapter à des espaces d'attributs et de labels dynamiques dans des tâches liées à des flux de données.

Une troisième direction de recherche qui s'inscrit dans la suite directe de cette thèse concerne l'apprentissage en continu (« continual learning »). Il s'agit alors de développer des modèles capables d'apprendre de nouvelles tâches ou de nouvelles connaissances sans oublier ce qu'ils ont appris auparavant. Les algorithmes ODM et A2ML que nous avons développés sont par construction compatibles avec les contraintes de l'apprentissage continu. En effet, ils peuvent apprendre sur des flux potentiellement infinis, et leur capacité à traiter des données multi-labels ouvre la voie vers le traitement de tâches différentes via l'établissement d'une relation entre les tâches et des sous-ensembles de labels. En introduisant des mécanismes adaptés de gestion des attributs et des labels, ces algorithmes pourraient donc se positionner comme prometteurs dans le champ de l'apprentissage continu. Cette voie commence à être explorée chez Orange avec le démarrage d'une nouvelle thèse.

2.2 Perspectives applicatives

En pratique, nos travaux ouvrent de nouvelles applications industrielles telles que l'intégration de A2ML pour la détection d'URL malveillantes.

A titre illustratif, la figure 6.1 montre un extrait de base de données d'URLs malveillantes. Ces bases de données comportent un début d'URL, nommé FQDN pour Fully

Qualified Domain Name, qui donne le nom de domaine d'un serveur. Une URL commune a en général un chemin et des paramètres complémentaires. Une URL malveillante est souvent générée dynamiquement avec un chemin et des paramètres complémentaires aléatoires et changeants qui rendent son identification plus compliquée.

L'exposition à des URLs malveillantes entraîne différents risques, comme par exemple une redirection automatique sur un site de hameçonnage ou un téléchargement d'un logiciel piégé (« malware »). D'autres URL peuvent être moins dangereuses tout en restant nuisibles car elles perturbent le fonctionnement des navigateurs, en générant du spam, ou en proposant des plugins ou des librairies non sécurisées. Des organismes internationaux fournissent des abonnements à des listes d'URLs malveillantes et les mises à jour, sous forme d'alertes, peuvent être nécessaires à tout moment. Les opérateurs internet se servent de ces listes noires pour bloquer les URLs dangereuses et ainsi protéger leurs clients.

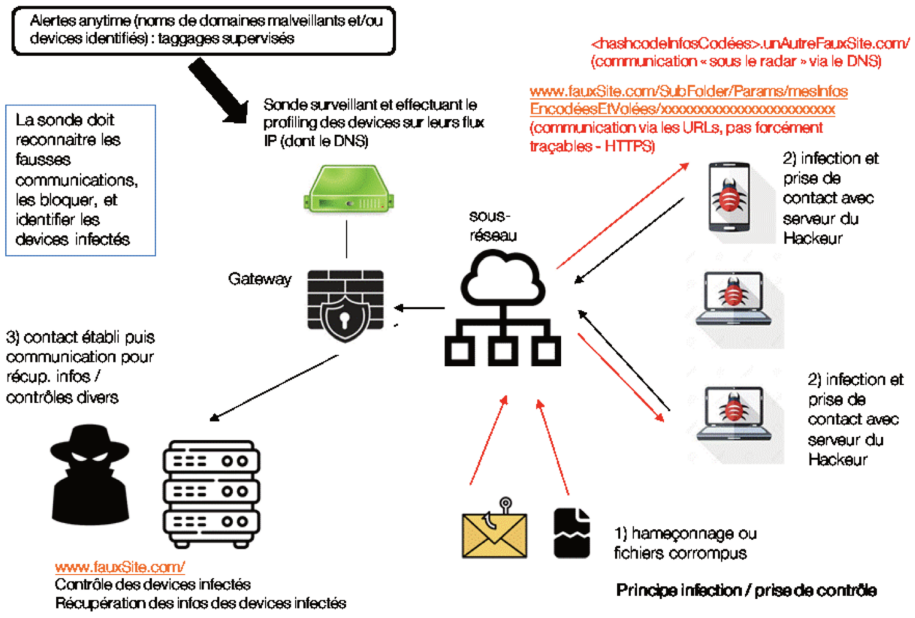


FIGURE 6.1 – Sonde de surveillance et de détection des machines infectées par des malwares en mode Command and Control.

Comme les URLs peuvent prendre des formes variées (adresses et/ou paramètres dynamiquement générés par les fraudeurs), il peut être utile d'apprendre les menaces en fonction de la forme des URLs (en profitant des biais des générateurs dynamiques) ou en utilisant les informations disponibles sur les pages pointées par les URLs. Les menaces correspondant aux URLs malveillantes peuvent être catégorisées en labels, tandis que les paramètres et informations pointées par les URLs correspondent aux attributs. Un classifieur multi-labels peut donc être alors utilisé pour prédire les différentes menaces associées à chaque URL.

Les attaques informatiques sont parfois de grande ampleur et se propagent très rapi-

dement. Si un type d'attaque se prépare, plusieurs URLs et plusieurs sites peuvent être concernés. Des alertes peuvent être générées par des organismes de lutte contre les cyberattaques, mais ces alertes ne concernent qu'une portion des URLs malveillantes. L'enjeu est donc alors d'apprendre très rapidement les motifs discriminants associés à ces URLs, pour être capable de reconnaître et d'alerter dans les instants qui suivent toute URL potentiellement liée à une attaque en cours.

Pour ce type d'application, un mécanisme d'apprentissage online, incrémental et multi-labels peut s'avérer être pertinent. En effet la capacité de réactivité et de généralisation en temps réel, à partir de quelques exemples d'URLs dangereuses, aux autres URLs dynamiquement générées correspondant à la même attaque est cruciale. Un système qui apprend « en batch », chaque semaine ou même chaque nuit n'est pas assez réactif et peut passer à côté d'URLs malveillantes qui ont plusieurs heures ou plusieurs jours pour se propager puis être consultées par les clients. Par ailleurs la qualification des types de menaces, non exclusive, est importante ; certaines menaces sont corrélées, et pourraient être mieux identifiées via une analyse multi-labels ; d'autres menaces peuvent être moins importantes, ou correspondre à une « zone grise » où le blocage systématique doit être remplacé par un message indiquant aux utilisateurs de prendre de précautions, etc. Au moment de la finalisation du manuscrit, cette perspective applicative est considérée avec une grande attention chez Orange et un projet expérimental de détection des menaces par apprentissage on line est en cours de montage.

Bibliographie

[1] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4) :1–37, 2014.

[2] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning : A review and comparison of state of the art algorithms. *Neurocomputing*, 275 :1261–1274, 2018.

[3] Ralph Spencer. The square kilometre array : The ultimate challenge for processing big data. In *IET Seminar on Data Analytics 2013 : Deriving Intelligence and Value from Big Data*, pages 1–26. IET, 2013.

[4] Zachary D Stephens, Skylar Y Lee, Faraz Faghri, Roy H Campbell, Chengxiang Zhai, Miles J Efron, Ravishankar Iyer, Michael C Schatz, Saurabh Sinha, and Gene E Robinson. Big data : astronomical or genomical ? *PLoS biology*, 13(7) :e1002195, 2015.

[5] Ase Dragland. Big data–for better or worse. *SINTEF. no. 22 May 2013. Web. 27 Oct*, 2013.

[6] Kun Lan, Dan-tong Wang, Simon Fong, Lian-sheng Liu, Kelvin KL Wong, and Nilanjan Dey. A survey of data mining and deep learning in bioinformatics. *Journal of medical systems*, 42(8) :1–20, 2018.

[7] Martin Kay, Martin Röscheisen, et al. Text-translation alignment. *Computational linguistics*, 19(1) :121–142, 1994.

[8] Yufeng Kou, Chang-Tien Lu, Sirirat Sirwongwattana, and Yo-Ping Huang. Survey of fraud detection techniques. In *IEEE international conference on Networking, sensing and control*, volume 2, pages 749–754. IEEE, 2004.

[9] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings International Conference on Knowledge Discovery and Data mining*, pages 71–80, 2000.

[10] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134 :75–88, 2019.

[11] Stephen Grossberg. Nonlinear neural networks : Principles, mechanisms, and architectures. *Neural networks*, 1(1) :17–61, 1988.

[12] Huimei Hu, Monika Sawhney, Lizheng Shi, Shengnan Duan, Yunxian Yu, Zhihong Wu, Guixing Qiu, and Hengjin Dong. A systematic review of the direct economic burden of type 2 diabetes in china. *Diabetes Therapy*, 6(1) :7–16, 2015.

[13] Zhaolan Liu, Chaowei Fu, Weibing Wang, and Biao Xu. Prevalence of chronic complications of type 2 diabetes mellitus in outpatients-a cross-sectional hospital based survey in urban china. *Health and quality of life outcomes*, 8(1) :1–9, 2010.

[14] Liang Zhou, Xiaoyuan Zheng, Di Yang, Ying Wang, Xuesong Bai, and Xinhua Ye. Application of multi-label classification models for the diagnosis of diabetic complications. *BMC medical informatics and decision making*, 21(1) :1–10, 2021.

[15] SO Folorunso, SG Fashoto, J Olaomi, and OY Fashoto. A multi-label learning model for psychotic diseases in nigeria. *Informatics in Medicine Unlocked*, 19 :100326, 2020.

[16] Ken Lang. Newsweeder : Learning to filter netnews. In *Machine Learning Proceedings 1995*, pages 331–339. Elsevier, 1995.

[17] Hung-Yi Lo, Ju-Chiang Wang, Hsin-Min Wang, and Shou-De Lin. Cost-sensitive multi-label learning for audio tag annotation and retrieval. *IEEE Transactions on Multimedia*, 13(3) :518–529, 2011.

[18] Forrest Briggs, Yonghong Huang, Raviv Raich, Konstantinos Eftaxias, Zhong Lei, William Cukierski, Sarah Frey Hadley, Adam Hadley, Matthew Betts, Xiaoli Z Fern, et al. The 9th annual mlsp competition : New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *2013 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–8. IEEE, 2013.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet : A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pages 248–255. IEEE, 2009.

[20] Ricardo Cabral, Fernando De la Torre, Joao Paulo Costeira, and Alexandre Bernardino. Matrix completion for weakly-supervised multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1) :121–135, 2015.

[21] Jung-Yi Jiang, Shian-Chi Tsai, and Shie-Jue Lee. Fsknn : multi-label text categorization based on fuzzy similarity and k nearest neighbors. *Expert Systems with Applications*, 39(3) :2813–2821, 2012.

[22] Haytham Elghazel, Alex Aussem, Ouadie Gharroudi, and Wafa Saadaoui. Ensemble multi-label text categorization based on rotation forest and latent semantic indexing. *Expert Systems with Applications*, 57 :1–11, 2016.

[23] Xiao Wang, Weiwei Zhang, Qiuwen Zhang, and Guo-Zheng Li. Multip-schlo : multi-label protein subchloroplast localization prediction with chou's pseudo amino acid composition and a novel multi-label classifier. *Bioinformatics*, 31(16) :2639–2645, 2015.

[24] Jian-Sheng Wu, Sheng-Jun Huang, and Zhi-Hua Zhou. Genome-wide protein function prediction through multi-instance multi-label learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 11(5) :891–902, 2014.

[25] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1) :5–53, 2004.

[26] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. Social collaborative filtering by trust. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(8) :1633–1647, 2017.

[27] Wissam Siblini. *Apprentissage multi label extrême : comparaisons d'approches et nouvelles propositions*. PhD thesis, Nantes, 2018.

[28] Frank Meyer, Sylvie Tricot, Pascale Kuntz, and Wissam Siblini. Vipe : un outil interactif de classification multilabel de messages courts. In *Extraction et Gestion des Connaissances. EGC 2017*, 2017.

[29] Dolly Carrillo, Vivian F López, and María N Moreno. Multi-label classification for recommender systems. *Trends in Practical Applications of Agents and Multiagent Systems*, pages 181–188, 2013.

[30] Grigorios Tsoumakas and Ioannis Katakis. Multi-label classification : An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3) :1–13, 2007.

[31] Patrick Pantel, Dekang Lin, et al. Spamcop : A spam classification & organization program. In *Proceedings of AAAI-98 Workshop on Learning for Text Categorization*, pages 95–98, 1998.

[32] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[33] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 115–124, 2017.

[34] Gulisong Nasierding, Grigorios Tsoumakas, and Abbas Z Kouzani. Clustering based multi-label classification for image annotation and retrieval. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 4514–4519. IEEE, 2009.

[35] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data : state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 21(2) :6–22, 2019.

[36] Xiulin Zheng, Peipei Li, Zhe Chu, and Xuegang Hu. A survey on multi-label data stream classification. *IEEE Access*, 8 :1249–1275, 2019.

[37] Wissam Siblini, Pascale Kuntz, and Frank Meyer. Craftml, an efficient clustering-based random forest for extreme multi-label learning. In *International Conference on Machine Learning*, pages 4664–4673. PMLR, 2018.

[38] David Nova and Pablo A Estévez. A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25(3) :511–524, 2014.

[39] Xihui Wang, Pascale Kuntz, Frank Meyer, and Vincent Lemaire. Multi-label knn classifier with online dual memory on data stream. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 405–413. IEEE, 2021.

[40] Xihui Wang, Pascale Kuntz, Frank Meyer, and Vincent Lemaire. A2ml : Adaptive memories for multi-label stream classification. 2022.

[41] Jesse Read. *Scalable multi-label classification.* PhD thesis, University of Waikato, 2010.

[42] Jason Weston, Samy Bengio, and Nicolas Usunier. Wsabie : Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[43] Kush Bhatia, Himanshu Jain, Purushottam Kar, Manik Varma, and Prateek Jain. Sparse local embeddings for extreme multi-label classification. In *Advances in neural information processing systems*, volume 28, 2015.

[44] Wei Weng, Yu-Wen Li, Jing-Hua Liu, Shun-Xiang Wu, and Chin-Ling Chen. Multi-label classification review and opportunities. *J Netw Intell*, 6(2) :255–275, 2021.

[45] Matthew R Boutell, Jiebo Luo, Xipeng Shen, and Christopher M Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9) :1757–1771, 2004.

[46] Min-Ling Zhang, Yu-Kun Li, Xu-Ying Liu, and Xin Geng. Binary relevance for multi-label learning : an overview. *Frontiers of Computer Science*, 12(2) :191–202, 2018.

[47] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8) :1819–1837, 2013.

[48] Adane Nega Tarekegn, Mario Giacobini, and Krzysztof Michalak. A review of methods for imbalanced multi-label classification. *Pattern Recognition*, 118 :107965, 2021.

[49] Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3) :333–359, 2011.

[50] Eyke Hüllermeier, Johannes Fürnkranz, Weiwei Cheng, and Klaus Brinker. Label ranking by learning pairwise preferences. *Artificial Intelligence*, 172(16-17) :1897–1916, 2008.

[51] Yuncheng Li, Yale Song, and Jiebo Luo. Improving pairwise ranking for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3617–3625, 2017.

[52] Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Multi-label classification using ensembles of pruned sets. In *2008 eighth IEEE international conference on data mining*, pages 995–1000. IEEE, 2008.

[53] Grigorios Tsoumakas and Ioannis Vlahavas. Random k-labelsets : An ensemble method for multilabel classification. In *European conference on machine learning*, pages 406–417. Springer, 2007.

[54] Rui Xu and Donald Wunsch. Survey of clustering algorithms. *IEEE Transactions on neural networks*, 16(3) :645–678, 2005.

[55] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski. Tree ensembles for predicting structured outputs. *Pattern Recognition*, 46(3) :817–833, 2013.

[56] Min-Ling Zhang and Zhi-Hua Zhou. Ml-knn : A lazy learning approach to multi-label learning. *Pattern recognition*, 40(7) :2038–2048, 2007.

[57] Min-Ling Zhang and Kun Zhang. Multi-label learning by exploiting label dependency. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 999–1008, 2010.

[58] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[59] Min-Ling Zhang and Zhi-Hua Zhou. Multilabel neural networks with applications to functional genomics and text categorization. *IEEE transactions on Knowledge and Data Engineering*, 18(10) :1338–1351, 2006.

[60] Leo Breiman. Bagging predictors. *Machine learning*, 24(43) :123–141, 1996.

[61] Stefanie Nowak, Hanna Lukashevich, Peter Dunker, and Stefan Rüger. Performance measures for multilabel evaluation : a case study in the area of image classification. In *Proceedings of the international conference on Multimedia information retrieval*, pages 35–44, 2010.

[62] Andrés Felipe Giraldo-Forero, Jorge Alberto Jaramillo-Garzón, and César Germán Castellanos-Domínguez. Evaluation of example-based measures for multi-label classification performance. In *International Conference on Bioinformatics and Biomedical Engineering*, pages 557–564. Springer, 2015.

[63] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift : A review. *IEEE Transactions on Knowledge and Data Engineering*, 31(12) :2346–2363, 2018.

[64] Indre Zliobaite. Learning under concept drift : an overview. *arXiv preprint arXiv :1010.4784*, 41, 2010.

[65] Indrė Žliobaitė, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, 98(3) :455–482, 2015.

[66] Albert Bifet and Richard Kirkby. Data stream mining a practical approach. 2009.

[67] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. In *The collected works of Wassily Hoeffding*, pages 409–426. Springer, 1994.

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

[69] Viktor Losing, Barbara Hammer, and Heiko Wersing. Knn classifier with self adjusting memory for heterogeneous concept drift. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 291–300. IEEE, 2016.

[70] David Arthur and Sergei Vassilvitskii. k-means++ : The advantages of careful seeding. Technical report, Stanford, 2006.

[71] Pat Langley, Wayne Iba, Kevin Thompson, et al. An analysis of bayesian classifiers. In *Aaai*, volume 90, pages 223–228. Citeseer, 1992.

[72] Ronald L Rivest. Learning decision lists. *Machine learning*, 2(3) :229–246, 1987.

[73] Ezilda Almeida, Carlos Ferreira, and Joao Gama. Adaptive model rules from data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 480–492. Springer, 2013.

[74] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5) :185–196, 1993.

[75] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine : theory and applications. *Neurocomputing*, 70(1-3) :489–501, 2006.

[76] Vinicius Souza, Denis M dos Reis, Andre G Maletzke, and Gustavo EAPA Batista. Challenges in benchmarking stream learning algorithms with real-world data. *Data Mining and Knowledge Discovery*, 34(6) :1805–1858, 2020.

[77] Indrė Žliobaitė. Learning under concept drift : an overview. *arXiv preprint arXiv :1010.4784*, 2010.

[78] Geoffrey I Webb, Roy Hyde, Hong Cao, Hai Long Nguyen, and Francois Petitjean. Characterizing concept drift. *Data Mining and Knowledge Discovery*, 30(4) :964–994, 2016.

[79] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Brazilian symposium on artificial intelligence*, pages 286–295. Springer, 2004.

[80] Kyosuke Nishida and Koichiro Yamauchi. Detecting concept drift using statistical testing. In *International conference on discovery science*, pages 264–269. Springer, 2007.

[81] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[82] Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

[83] Peter Vorburger and Abraham Bernstein. Entropy-based concept shift detection. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 1113–1118. IEEE, 2006.

[84] Rémi Bardenet, Mátyás Brendel, Balázs Kégl, and Michele Sebag. Collaborative hyperparameter tuning. In *International conference on machine learning*, pages 199–207. PMLR, 2013.

[85] Christophe Giraud. *Introduction to high-dimensional statistics*. Chapman and Hall/CRC, 2021.

[86] Eleftherios Spyromitros-Xioufis, Myra Spiliopoulou, Grigorios Tsoumakas, and Ioannis Vlahavas. Dealing with concept drift and class imbalance in multi-label stream classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[87] Jesse Read, Albert Bifet, Geoffrey Holmes, and Bernhard Pfahringer. Efficient multi-label classification for evolving data streams. 2010.

[88] Joel D Costa Júnior, Elaine R Faria, Jonathan A Silva, João Gama, and Ricardo Cerri. Pruned sets for multi-label stream classification without true labels. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[89] Aljaž Osojnik, Panče Panov, and Sašo Džeroski. Multi-label classification via multi-target regression on data streams. *Machine Learning*, 106(6) :745–770, 2017.

[90] Zhongwei Shi, Yun Xue, Yimin Wen, and Guoyong Cai. Efficient class incremental learning for multi-label classification of evolving data streams. In *2014 international joint conference on neural networks (IJCNN)*, pages 2093–2099. IEEE, 2014.

[91] Ricardo Sousa and João Gama. Multi-label classification from high-speed data streams with adaptive model rules and random rules. *Progress in Artificial Intelligence*, 7(3) :177–187, 2018.

[92] Rajasekar Venkatesan, Meng Joo Er, Shiqian Wu, and Mahardhika Pratama. A novel online real-time classifier for multi-label data streams. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1833–1840. IEEE, 2016.

[93] Alican Büyükçakir, Hamed Bonab, and Fazli Can. A novel online stacked ensemble for multi-label stream classification. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1063–1072, 2018.

[94] Martha Roseberry, Bartosz Krawczyk, and Alberto Cano. Multi-label punitive knn with self-adjusting memory for drifting data streams. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(6) :1–31, 2019.

[95] Martha Roseberry and Alberto Cano. Multi-label knn classifier with self adjusting memory for drifting data streams. In *Second International Workshop on Learning with Imbalanced Domains : Theory and Applications*, pages 23–37. PMLR, 2018.

[96] Martha Roseberry, Bartosz Krawczyk, Youcef Djenouri, and Alberto Cano. Self-adjusting k nearest neighbors for continual learning from multi-label drifting data streams. *Neurocomputing*, 442 :10–25, 2021.

[97] Gavin Alberghini, Sylvio Barbon Junior, and Alberto Cano. Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams. *Neurocomputing*, 481 :228–248, 2022.

[98] TT Nguyen, TTT Nguyen, AV Luong, QVH Nguyen, WC Liew, and B Stantic. Multi-label classification via labels correlation and one-dependence features on data stream. *Pattern Recognition*, 90, 2019.

[99] Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*, pages 229–236. PMLR, 2001.

[100] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9) :1469–1495, 2017.

[101] Yange Sun, Han Shao, and Shasha Wang. Efficient ensemble classification for multi-label data streams with concept drift. *Information*, 10(5) :158, 2019.

[102] Huimin Xu, Wenting Wang, Xinnian Mao, Xinyu Jiang, and Man Lan. Scaling up open tagging from tens to thousands : Comprehension empowered attribute value extraction from product title. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5214–5223, 2019.

[103] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Philipp Kranen, Hardy Kremer, Timm Jansen, and Thomas Seidl. Moa : Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the first workshop on applications of pattern analysis*, pages 44–50. PMLR, 2010.

[104] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavalda. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 139–148, 2009.

[105] Milosz R Kmieciak and Jerzy Stefanowski. Handling sudden concept drift in enron messages data stream. 2011.

[106] Jesse Read, Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Scalable and efficient multi-label classification for evolving data streams. *Machine Learning*, 88(1-2) :243–272, 2012.

[107] Anand Rajagopalan, Fabio Vitale, Danny Vainstein, Gui Citovsky, Cecilia M Procopiuc, and Claudio Gentile. Hierarchical clustering of data streams : scalable algorithms and approximation guarantees. In *International conference on machine learning*, pages 8799–8809. PMLR, 2021.

[108] Alaettin Zubaroğlu and Volkan Atalay. Data stream clustering : a review. *Artificial Intelligence Review*, 54(2) :1201–1236, 2021.

[109] Stephen L France, J Douglas Carroll, and Hui Xiong. Distance metrics for high dimensional nearest neighborhood recovery : Compression and normalization. *Information Sciences*, 184(1) :92–110, 2012.

[110] Georges Voronoi. Nouvelles applications des paramètres continus à la théorie des formes quadratiques. deuxième mémoire. recherches sur les parallélloèdres primitifs. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1908(134) :198–287, 1908.

[111] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38 :293–306, 1985.

[112] Sen Wu, Xiaodong Feng, and Wenjun Zhou. Spectral clustering of high-dimensional data exploiting sparse representation vectors. *Neurocomputing*, 135 :229–239, 2014.

[113] Guangliang Chen. Scalable spectral clustering with cosine similarity. In *2018 24th International conference on pattern recognition (ICPR)*, pages 314–319. IEEE, 2018.

[114] Borja Calvo and Guzman Santafe. scmamp : Statistical comparison of multiple algorithms in multiple problems. *The R Journal*, Accepted for publication, 2015.

[115] Charu C Aggarwal. On biased reservoir sampling in the presence of stream evolution. In *Proceedings of the 32nd international conference on Very large data bases*, pages 607–618. Citeseer, 2006.

[116] Shazia Tabassum and João Gama. Sampling massive streaming call graphs. In *Proceedings of the 31st annual ACM symposium on applied computing*, pages 923–928, 2016.

[117] Charu C Aggarwal, S Yu Philip, Jiawei Han, and Jianyong Wang. A framework for clustering evolving data streams. In *Proceedings 2003 VLDB conference*, pages 81–92. Elsevier, 2003.

[118] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2) :129–137, 1982.

[119] Sasthakumar Ramamurthy and Raj Bhatnagar. Tracking recurrent concept drift in streaming data using ensemble classifiers. In *Sixth International Conference on Machine Learning and Applications (ICMLA 2007)*, pages 404–409. IEEE, 2007.

[120] Ronald Aylmer Fisher and Frank Yates. *Statistical tables for biological, agricultural and medical research*. Hafner Publishing Company, 1953.

[121] Ioannis Katakis, Grigorios Tsoumakas, and Ioannis Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD*, volume 18, page 5. Citeseer, 2008.

Titre : Classification Multi-Labels en flux : Comparaisons d'approches et nouvelles propositions.

Mots clés : classification multi-labels, flux de données, dérive conceptuelle

Résumé : Avec l'évolution conjointe des volumes de données à traiter et de la nature même de ces données, les algorithmes de classification multi-labels sont confrontés à un défi majeur : leur capacité à apprendre des modèles à partir de données en flux et à s'adapter aux changements de leurs distributions statistiques au fil du temps en prenant en compte des ressources matérielles limitées en stockage et en calcul. Dans cette thèse, nous abordons ce défi pour deux types de données : des flux stationnaires et non stationnaires. Pour la classification multi-labels de flux stationnaires nous avons développé un nouvel algorithme (MLT-ML) qui, avec une faible complexité temporelle, permet d'obtenir des performances en prédiction compétitives en exploitant les corrélations entre labels pour partitionner l'espace de recherche à chaque instant et réduire ainsi la complexité de l'apprentissage. Pour la classification de flux non-stationnaires nous avons développé successivement deux nouveaux algorithmes (ODM et A2ML) qui combinent une mémoire à court terme et une mémoire à long terme. Cette combinaison permet une adaptation efficace des modèles d'apprentissage aux dérives de concepts. En particulier, nous avons montré expérimentalement l'apport dans A2ML de l'introduction d'une règle d'échantillonnage biaisée pour la gestion de la mémoire à long terme ainsi que l'efficacité de la création de nouveaux clusters associés à l'apparition de nouveaux labels dans le flux. Pour combler l'absence de protocoles d'évaluation consensuels pour la classification multi-labels sur des données en flux, nous avons développé un nouveau cadre de simulation qui permet d'introduire explicitement des dérives de différents types et donc de mieux comprendre les changements de comportements des différentes stratégies de classification. Les comparaisons avec les meilleurs algorithmes de l'état de l'art menées sur des flux non stationnaires de plus de 50 000 exemples confirment le niveau élevé de performances de notre nouvel algorithme A2ML qui a une complexité temporelle significativement plus réduite que tous les autres.

Title : Multi-Labels Stream Classification : Comparisons of approaches and new proposals.

Keywords : multi-Labels classification, data stream, concept drift

Abstract : Due to the ever-increasing number of current applications, multi-label classification algorithms are facing a major challenge: their capacity for learning models from streaming data that include changes in distribution over time, while constantly coming up against limited computational and storage resources. In this thesis, we first study the multi-label classification problem on stationary streams and propose a new algorithm MLT-ML. This algorithm not only has a very low time complexity, but also has a high prediction performance by using the labels' correlation to partition the label space at each time. Then, we provide two new algorithms, ODM and A2ML, for non-stationary streams, which both combine a short-term memory with a long-term one. This combination ensures an efficient adaptation to the various types of concept drift. In particular, by using the biased reservoir sampling strategy and creating new clusters for new labels, A2ML can adapt to drift more effectively than ODM and its efficiency will not decrease over time. In addition, in order to further understand the behavior of the algorithm on the non-stationary stream, we also propose a new evaluation protocol to generate various types of concept drift. The experimentation confirmes A2ML's high levels of performance, and reveal computation times that are lower than those of the state of the art.

COLLEGE ED MATHEMATIQUES
DOCTORAL STIC
PAYS DE LA LOIRE

Nantes Université