Grille bivariée pour la détection de changement dans un flux étiqueté

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

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

Résumé. Nous présentons une méthode en-ligne de détection de changement de concept dans un flux étiqueté. Notre méthode de détection est basée sur un critère supervisé bivarié qui permet d'identifier si les données de deux fenêtres proviennent ou non de la même distribution. Notre méthode a l'intérêt de n'avoir aucun *a priori* sur la distribution des données, ni sur le type de changement et est capable de détecter des changements de différentes natures (changement dans la moyenne, dans la variance...). Les expérimentations montrent que notre méthode est plus performante et robuste que les méthodes de l'état de l'art testées. De plus, à part la taille des fenêtres, elle ne requiert aucun paramètre utilisateur.

1 Introduction

De nombreux acteurs de l'informatique doivent faire face à l'arrivée massive de données. Les plus connus sont Google et Yahoo avec le traitement des logs pour la publicité en-ligne, Facebook et Twitter qui modélisent les données provenant de leurs centaines de millions d'utilisateurs, les opérateurs téléphoniques pour la gestion de réseaux de télécommunications. La volumétrie de ces données continue de croître rapidement et les quantités ne sont plus compatibles avec l'utilisation de la plupart des méthodes hors-ligne qui supposent de pouvoir accéder à toutes les données. Dans ces conditions, il est préférable de traiter les données à leur passage ce qui impose d'y accéder une seule fois et dans leur ordre d'arrivée. On parle alors d'un accès sous la forme d'un flux de données.

En classification supervisée, on appelle concept $P(C|X)$ la probabilité conditionnelle de la classe C connaissant les données X. Les flux de données peuvent ne pas être stationnaires et comporter des changements de concept si le processus qui génère les données varie au cours du temps. Dans ce cas le modèle de classification supervisée doit être adapté au fur et à mesure que le concept change.

Cet article propose une nouvelle méthode de détection de changement basée sur l'observation des données du flux. Notre méthode utilise deux fenêtres et permet d'identifier si les données de ces deux fenêtres proviennent ou non de la même distribution. Elle est capable de détecter les changements de diverses natures (moyenne, variance...) sur la distribution des

données conditionnellement ou non aux classes. Notre méthode a l'intérêt de n'avoir aucun *a priori* sur la distribution des données, ni sur le type de changement. De plus, à part la taille de la fenêtre, elle ne requiert aucun paramètre utilisateur.

Cet article présente tout d'abord, dans la section 2, les approches existantes de l'état de l'art auxquelles l'approche proposée sera comparée. La section 3 présente notre méthode de détection ainsi qu'une validation expérimentale sur des données artificielles. La section 4 montre comment notre méthode de détection peut être utilisée au sein d'une méthode de gestion de la dérive de concept. La dernière partie conclut cet article.

2 Etat de l'art des méthodes de détection

L'état de l'art sur la gestion de la dérive de concept est abondant (Bifet et al., 2009; Žliobaite, 2010). Les méthodes de gestion de la dérive peuvent se diviser en plusieurs familles : détection de changement, ensemble de classifieurs, pondération des données selon leur ancienneté... Le but de cet article étant de présenter une nouvelle méthode de détection, l'état de l'art présenté ici se focalise donc sur la famille des méthodes de détection de changement.

Le but de la classification supervisée sur flux de données est d'optimiser les performances du classifieur. Dans ce cas, l'état de l'art peut encore être divisé en deux familles :

- méthodes sans classifieur : les auteurs s'intéressent directement aux distributions d'intérêt du flux de données : $P(X)$, $P(C)$, $P(X|C)$
- méthodes avec classifieur : les auteurs s'intéressent à la performance, liée à $P(C|X)$, d'un classifieur.

D'un point de vue bayésien cela revient à détecter les variations d'une des quantités de la formule suivante : $P(C|X) = (P(C)P(X|C))\,/P(X)$ avec :

- $P(C)$ la proportion des classes dans les données
- $P(X)$ la probabilité des données
- $P(X|C)$ la probabilité conditionnelle des X connaissant la classe C.

A notre connaissance il n'existe pas de méthodes permettant de détecter directement les variations dans la distribution jointe $P(X, C)$.

La méthode proposée dans cet article ayant vocation à détecter les changements sur les trois termes mentionnés ci-dessus, on présente dans les sous sections suivantes une brève description des méthodes de l'état de l'art testées de façon comparative dans cet article.

2.1 Méthodes sans classifieur

Les méthodes sans classifieur sont essentiellement des méthodes basées sur des tests statistiques qui sont utilisés pour détecter un changement entre différentes fenêtres d'observations.

Test t de Welch : Le test t de Welch est une adaptation du test t de Student qui s'applique à deux échantillons de tailles N_1 et N_2. Ce test permet de tester statistiquement l'hypothèse d'égalité de deux moyennes ($\overline{X}_1$ et $\overline{X}_2$) avec deux échantillons de variances inégales (s_1^2 et s_2^2). Ce test a pour formule : $t = (\overline{X}_1 - \overline{X}_2)/\left(\sqrt{(s_1^2/N_1) + (s_2^2/N_2)}\right)$. On utilise le test t de Welch pour tester l'hypothèse nulle suivante : « les moyennes de deux populations sont égales ». Ce test retourne une $p\text{-}value$ qui permet de rejeter ou non l'hypothèse nulle.

Test de Kolmogorov-Smirnov : Le test d'hypothèse de Kolmogorov-Smirnov est utilisé pour déterminer si un échantillon suit bien une loi donnée ou bien si deux échantillons suivent la même loi. Ce test est basé sur les propriétés des fonctions de répartition empirique. Nous utiliserons ce test pour vérifier si deux échantillons suivent la même loi. Soient deux échantillons de tailles N_1 et N_2 possédant respectivement les fonctions de répartition empirique $F_1(x)$ et $F_2(x)$. La distance de Kolmogorov-Smirnov est définie de la manière suivante : $D = \max_x |F_1(x) - F_2(x)|$. L'hypothèse nulle, stipulant que les deux échantillons proviennent de la même distribution, est rejetée avec une confiance α si : $\left(\sqrt{(N_1 N_2)/(N_1 + N_2)}\right) D > K_\alpha$. K_α se retrouve à l'aide des tables de Kolmogorov-Smirnov.

MODL $P(W|X_i)$: Cette méthode a été proposée par (Bondu et Boullé, 2011) pour la détection de changement dans le cadre de l'observation d'une variable numérique X_i. Elle traite le problème de la détection de changement comme un problème d'apprentissage à deux classes. Cette méthode utilise deux fenêtres pour détecter un changement : une fenêtre de référence W_{ref} qui contient les données du concept de départ et une fenêtre courante W_{cur} glissante/sautante sur le flux. Les exemples de la fenêtre de référence ont comme étiquette la classe « ref », et ceux de la fenêtre courante la classe « cur ». Ces deux étiquettes constituent la variable cible $W \in \{W_{ref}, W_{cur}\}$ du problème d'apprentissage. Les exemples de ces deux fenêtres sont fusionnées et la discrétisation supervisée MODL (Boullé, 2006) appliquée à ces exemples. Si la variable X_i est discrétisée en au moins 2 intervalles cela signifie qu'il y a au moins 2 zones où la distribution des exemples conditionnellement à la fenêtre W est significativement différente. Dans ce cas la méthode conclut qu'un changement s'est produit.

2.2 Méthodes avec classifieur

Les méthodes avec classifieur observent les performances du classifieur et détectent un changement quand les performances varient de manière significative. Ces méthodes ont comme hypothèses de départ que le classifieur est un processus stationnaire et que les données sont indépendantes et identiquement distribuées (iid). Bien que ces hypothèses ne soient pas toujours validées, ces méthodes ont prouvé leur intérêt sur diverses expérimentations (Gama et al., 2004; Baena-García et al., 2006; Bifet et al., 2009). Les deux principales méthodes avec classifieur référencées dans l'état de l'art sont décrites ci-dessous.

DDM : La méthode DDM, proposée par Gama et al. (Gama et al., 2004), détecte les changements en observant l'évolution du taux d'erreur du classifieur. L'algorithme prend en entrée une distribution binomiale provenant de la variable binaire qui indique si l'exemple est bien classé (0) ou mal classé (1) par le classifieur. Cette loi binomiale est approximée par une loi normale après avoir vu 30 exemples. La méthode estime pour chaque exemple la probabilité qu'il soit mal classé p_i (p_i correspond aussi au taux d'erreur) et son écart type $s_i = \sqrt{p_i(1-p_i)/i}$. Cette méthode considère qu'une augmentation significative du taux d'erreur correspond à un changement de concept et donc qu'il faut réapprendre. Les valeurs p_i et s_i et les minima atteints p_{min} et s_{min} sont conservés. La méthode fonctionne en deux temps, tout d'abord une alerte est levée (« warning ») puis une détection. Ces deux niveaux sont définis de la manière suivante : (1) alerte : $p_i + s_i \geqslant p_{min} + 2 \cdot s_{min}$; (2) détection : $p_i + s_i \geqslant p_{min} + 3 \cdot s_{min}$. Entre la phase d'alerte et de détection un nouveau classifieur est construit. De cette manière,

après la détection, il est possible de ne pas repartir d'un classifieur sans apprentissage mais d'un classifieur ayant appris sur les données du flux comprises entre l'alerte et la détection.

EDDM : La méthode EDDM (Baena-García et al., 2006) reprend le mode de fonctionnement de DDM mais propose un autre critère pour évaluer les seuils d'alerte et de détection. Cette méthode utilise la distance entre les erreurs de classification plutôt que le taux d'erreur. Cette distance correspond aux nombres de bonnes prédictions entres deux mauvaises prédictions. EDDM calcule la distance moyenne entre les erreurs p'_i et l'écart type s'_i et conserve les maxima de la moyenne p'_{max} et l'écart s'_{max}. De la même manière que pour DDM un seuil d'alerte et de détection sont définis : (1) alerte : $(p'_i + 2 \cdot s'_i)/(p'_{max} + 2 \cdot s'_{max}) < \alpha$; (2) détection : $(p'_i + 2 \cdot s'_i)/(p'_{max} + 2 \cdot s'_{max}) < \beta$. Pour les expérimentations les paramètres sont fixés à $\alpha = 90\%$ et $\beta = 95\%$. Sur des jeux de données synthétiques et réels EDDM détecte plus rapidement que DDM les changements graduels.

3 Une nouvelle méthode de détection supervisée

3.1 Présentation

La méthode proposée dans cet article pour détecter les changements de concept s'inspire de la méthode MODL $P(W|X_i)$ (Bondu et Boullé, 2011) présentée dans la section 2.1. Le flux de données est constitué de d variables explicatives ($X_i, i \in \{1, ..., d\}$). Notre méthode fait l'hypothèse d'indépendance des variables conditionnellement aux classes. Le test de changement est donc réalisé par variable X_i sur les données provenant de deux fenêtres. La première fenêtre W_{ref} contient les données du concept de départ. La deuxième W_{cur} est une fenêtre glissante/sautante sur le flux qui permet de capturer les données d'un éventuel nouveau concept. Le choix de la taille des fenêtres est un compromis entre la réactivité aux changements et le nombre de mauvaises détections. Une grande taille de fenêtre permet de détecter avec plus de confiance des motifs potentiellement plus complexes. Une plus petite taille permet d'être plus réactif. Fixer cette taille dépend du flux observé et du type de changements que l'on veut détecter. De manière générale on peut traiter ce problème en utilisant plusieurs tailles de fenêtre en parallèle.

Les données sont étiquetées par fenêtre : $W \in \{W_{ref}, W_{cur}\}$. Notre intérêt porte sur la détection de la dérive du concept dans le cadre de la classification supervisée. Par conséquent on s'intéresse à la probabilité de la fenêtre connaissant à la fois la classe C et la variable X_i : $P(W|C, X_i)$. Il sera donc nécessaire d'utiliser une méthode permettant de mesurer la différence entre $P(W_{ref}|C, X_i)$ et $P(W_{cur}|C, X_i)$.

Parmi les méthode de l'état l'art de discrétisation/groupage bivarié capables de prendre en compte la variable X_i et la classe C nous avons choisi d'utiliser l'approche MODL (Boullé, 2009) pour ses caractéristiques : pas d'*a priori* sur la distribution des données, faible sensibilité aux valeurs atypiques, pas de paramètres utilisateur, régularisation pour éviter le sur-apprentissage.

Un changement est détecté quand la variable à expliquer (dans notre cas W) peut être discriminée à partir de l'observation de X_i et C. Ceci se traduit dans l'approche MODL bivariée par l'observation d'une grille de discrétisation/groupage de plus d'une cellule. Si la grille n'a qu'une seule cellule, alors la distribution des données n'a pas changé entre les deux fenêtres.

La complexité algorithmique de cette méthode est en $O(T\sqrt{T}log(T))$ où T est la somme de la taille des deux fenêtres (référence et courante).

3.2 Validation expérimentale : détection sans classifieur

Dans la suite de cette section on s'intéresse au comportement de la méthode proposée sur des données artificielles provenant de la simulation de différents types de changement. Le comportement dans le cas stationnaire est étudié dans la section 3.2.1, la capacité à détecter un changement dans la section 3.2.2 et la capacité à détecter différents types de changement dans la section 3.2.3. Les expérimentations comparent les méthodes de l'état de l'art présentées précédemment ainsi que notre approche :

- test t de Welch avec signification statistique de 1%, 5% et 10%
- test de Kolmogorov-Smirnov (KS) avec signification statistique de 1%, 5% et 10%
- méthode supervisée MODL $P(W|X_i)$
- méthode supervisée bivariée MODL $P(W|X_i, C)$

3.2.1 Comportement dans le cas stationnaire

Le but de cette première expérimentation est d'étudier le comportement des méthodes dans le cadre d'un flux stationnaire où l'on ne doit détecter aucun changement. Les fenêtres de référence et courante ont la même taille. Etant donné que la méthode fait l'hypothèse d'indépendance des variables, une seule variable numérique X_a est utilisée pour la validation expérimentale. La distribution des classes se fait selon une gaussienne de paramètres $(\mu_1 = -1, \sigma_1 = 1)$ pour la classe 1 et une gaussienne de paramètres $(\mu_2 = 1, \sigma_2 = 1)$ pour la classe 2. Différentes tailles de fenêtre ont été choisies entre 10 et 5 000 exemples. Les expérimentations sont réalisées 1000 fois.

Les résultats sont présentés dans le tableau 1. Ceux-ci confirment la robustesse de la méthode bivariée MODL car aucune détection n'est observée sur $P(W|X_a, C)$. Par contre quelques fausses alarmes ont lieu sur $P(W|X_a)$ (Bondu et Boullé, 2011) dans moins de 1% des cas et pour de petites tailles de fenêtre. Les autres tests se comportent bien pour des petites valeurs de signification statistique (1%). Pour de plus grandes valeurs (10%), de 1% à 3% de fausses détections sont réalisées par le test t de Welch et le test de Kolmogorov-Smirnov.

méthode→ ↓ taille	Welch (1%)	Welch (5%)	Welch (10%)	KS (1%)	KS (5%)	KS (10%)	MODL $P(W\|X_a)$	MODL $P(W\|X_a, C)$
10	0	9	20	0	0	8	3	0
20	0	7	13	0	5	22	4	0
30	1	6	19	0	3	12	3	0
50	0	5	21	0	2	17	6	0
100	0	3	19	0	4	20	1	0
200	0	5	28	0	5	13	0	0
300	0	6	16	1	5	21	0	0
500	1	6	22	2	7	18	1	0
1000	1	8	25	0	5	24	0	0
2000	0	4	13	0	7	20	0	0
5000	0	6	26	0	7	19	0	0

TAB. 1 – *Nombre de mauvaises détections selon la méthode détection et la taille de la fenêtre pour 1000 expérimentations.*

3.2.2 Capacité à détecter un changement

Le but de cette expérimentation est d'observer le temps nécessaire (en nombre d'exemples) pour détecter un changement en fonction de la taille de la fenêtre et des méthodes. La plupart des méthodes sont capable de détecter ce type de changement. Le point important est ici d'analyser la vitesse de détection.

Les fenêtres de référence et courante ont la même taille. Une seule variable X_a est utilisée. Le concept 1 est défini ainsi :

- la classe 0 suit une distribution $\mathcal{N}_0(\mu = -1, \sigma = 1)$;
- la classe 1 suit une distribution $\mathcal{N}_1(\mu = 1, \sigma = 1)$.

On simule ce changement en changeant la moyenne de la classe 0, qui passe de -1 à 2 et sa variance qui passe de 1 à 0,5. Pour la classe 1, seule la moyenne change en passant de 1 à 0. On obtient donc le concept 2 suivant :

- la classe 0 suit une distribution $\mathcal{N}_0(\mu = 2, \sigma = 0,5)$;
- la classe 1 suit une distribution $\mathcal{N}_1(\mu = 0, \sigma = 1)$.

Différentes tailles de fenêtres ont été testées entre 10 et 5 000 exemples et les expérimentations sont réalisées 1000 fois. La position du changement dans la fenêtre est tirée de manière aléatoire car dans un cas réel la position du changement n'est pas connue. Les résultats obtenus correspondent aux délais moyens de détection du changement en fonction de la taille de la fenêtre et des méthodes étudiées.

Les résultats sont présentés dans le tableau 2. On observe que jusqu'à une taille de fenêtre de 100 exemples la détection est difficile dans la fenêtre où a lieu le changement. A partir d'une taille de 200 exemples, le délai moyen est inférieur à la taille de la fenêtre et ce pour toutes les méthodes. L'augmentation du seuil de signification à 5% et 10% entraine, pour les tests de Welch et de Kolmogorov-Smirnov, une baisse du délai de détection. La méthode basée sur MODL $P(W|X_a)$ est légèrement plus longue à détecter que les méthodes paramétriques à 1%. Cependant, si on prend notre méthode MODL $P(W|X_a, C)$ alors celle-ci est meilleure que les deux autres méthodes paramétrées à 1%.

méthode→ ↓ taille	Welch (1%)	Welch (5%)	Welch (10%)	KS (1%)	KS (5%)	KS (10%)	MODL $P(W\|X_a)$	MODL $P(W\|X_a, C)$
10	15	14	14	15	15	15	15	15
20	29	26	24	29	28	26	28	29
30	41	36	33	43	40	36	41	40
50	62	53	49	67	57	54	65	58
100	103	90	86	110	96	90	109	96
200	178	160	150	185	166	160	195	163
300	241	218	211	252	227	214	269	218
500	367	337	321	375	344	330	404	325
1000	665	620	598	678	636	613	719	600
2000	1224	1188	1154	1260	1188	1174	1334	1120
5000	2886	2766	2741	2911	2781	2756	3081	2671

TAB. 2 – *Délai moyen de détection d'un changement selon la méthode de détection et la taille de la fenêtre pour 1000 expérimentations.*

3.2.3 Différents types de changements détectés

Les expériences de cette section ont pour but d'observer quels types de changement (moyenne, variance, inversion des classes) les différentes méthodes sont capables de détecter. Pour tous

les types de changement testés, le concept de départ est le concept 1 défini ainsi : la classe 0 suit une distribution $\mathcal{N}_0(\mu = 0, \sigma = 0,5)$ et la classe 1 suit une distribution $\mathcal{N}_1(\mu = 2, \sigma = 1)$. Différents changements sont appliqués au concept 1 pour expérimenter le comportement des différentes méthodes.

- **Changement de moyenne.** On simule ce changement en changeant la moyenne de la classe 0, qui passe de 0 à 1. On obtient le concept 2 : la classe 0 suit une distribution $\mathcal{N}_0(\mu = 1, \sigma = 0,5)$ et la classe 1 suit une distribution $\mathcal{N}_1(\mu = 2, \sigma = 1)$.
- **Changement de variance.** On simule ce changement en changeant la variance de la classe 0, qui passe de 1 à 0,5. On obtient le concept 2 : la classe 0 suit une distribution $\mathcal{N}_0(\mu = 0, \sigma = 1)$ et la classe 1 suit une distribution $\mathcal{N}_1(\mu = 2, \sigma = 1)$.
- **Inversion des classes.** On simule ce changement en inversant les étiquettes des classes du concept 1.

Pour ces expérimentations et pour les différents types de changement, les tailles des fenêtres ont été fixées à 1000. La fenêtre de référence contient le concept 1 et la fenêtre courante le concept 2. Le tableau 3 présente les résultats en termes de nombre de détections pour 1000 expérimentations. Les meilleures méthodes sont celles qui sont capables de réaliser le plus de détections.

Méthode	Moyenne	Variance	Inversion des classes
Welch (1%)	1000	0	0
Welch (2%)	1000	0	0
Welch (10%)	1000	19	10
KS (1%)	1000	998	0
KS (2%)	1000	1000	0
KS (10%)	1000	1000	15
MODL $P(W\|X_a)$	1000	1000	1
MODL $P(W\|X_a, C)$	1000	1000	1000

TAB. 3 – *Nombre de détections par changement pour les différentes méthodes de détection.*

Toutes les méthodes sont capables de détecter de manière fiable un changement dans la moyenne. Le test de Welch n'est pas capable de détecter les changements de variance et ces expériences le confirment. Quelques détections (19) pour le test à 10% se produisent, mais celles-ci correspondent à des fausses détections et non pas à une détection de changement de variance. Le test de Kolmogorov-Smirnov et la détection MODL $P(W|X_a)$ se comportent très bien pour les détections dans les changements de moyenne et variance. Tous les tests précédents détectent seulement des changements dans la répartition des données mais sans s'intéresser à la classe. Par conséquent aucun d'entre eux n'est capable de détecter un changement qui n'intervient que par rapport aux classes. Contrairement à ces tests statistiques, la méthode de détection bivariée MODL $P(W|X_a, C)$ est capable de détecter ce genre de changements. Les expérimentations confirment cette capacité.

4 Application à la gestion de la dérive de concept

Détecter les changements requiert ensuite de réagir à ces derniers. Si l'on sait détecter les changements de concepts au cours du temps on pourra alors (Žliobaite, 2010) : (i) soit réapprendre le modèle de classification à partir de zéro ; (ii) soit adapter le modèle courant ; (iii) soit adapter un résumé des données sur lequel se fonde le modèle courant ; (iv) soit travailler

avec la séquence des modèles de classification appris au cours du temps. Cette section propose d'intégrer la méthode de détection précédente dans un algorithme de gestion de la dérive de concept afin de remplacer le classifieur après une détection de changement.

4.1 Algorithme MDD

Nous proposons d'intégrer notre méthode de détection à un nouvel algorithme que nous appelons MDD : MODL Drift Detection Method (Algorithme 1). Notre algorithme n'utilise pas les performances du classifieur pour détecter les changements contrairement aux algorithmes DDM et EDDM décrits dans la section 2.2. Il ne dépend donc pas du type de classifieur.

Le remplacement de l'ancien classifieur par un nouveau se fait suite à une détection sur au moins l'une des variables par la méthode bivariée MODL $P(W|X_i, C)$. Le remplacement n'est effectif que lorsque le taux d'erreur du nouveau classifieur est plus faible que l'ancien. Ce taux d'erreur est calculé, pour nos expérimentations, à l'aide de la méthode `tauxErreur` qui correspond à une moyenne mobile exponentielle de paramètre $\alpha = 1/tailleW_{cur}$. Le paramètre n_{min} est le nombre minimum d'exemples utilisé pour comparer les performances des deux classifieurs, sa valeur est fixée à 30 ($n_{min} = 30$) pour toutes les expérimentations. Cette même valeur est utilisée par les méthodes DDM et EDDM avant qu'elles ne commencent à chercher un changement.

4.2 Protocole expérimental

Notre algorithme est configuré avec une même taille de fenêtre de 1000 pour la fenêtre de référence et la fenêtre sautante. La problématique du réglage de la taille de la fenêtre n'est pas abordée en détail dans cet article. Une grande taille de fenêtre permet de détecter avec plus de confiance ainsi que des motifs plus complexes. Une plus petite taille permet d'être plus réactif. Fixer cette taille dépend du flux observé et des changements que l'on veut détecter. De manière générale on peut traiter ce problème en utilisant plusieurs tailles de fenêtre en parallèle comme le propose (Lazarescu et al., 2004).

On peut Deux types de classifieur sont utilisés :

- un classifieur bayésien naïf utilisant une estimation de densité conditionnelle des classes basée sur des résumés à deux niveaux (Salperwyck et Lemaire, 2012). Le premier niveau est un résumé de quantiles à 100 tuples et le second niveau la discrétisation MODL.
- un arbre de Hoeffding (Domingos et Hulten, 2000) avec un résumé de quantiles à 10 tuples et un classifieur bayésien naïf utilisant la discrétisation MODL.

On présente les résultats pour le générateur basé sur un hyperplan en mouvement proposée dans (Hulten et al., 2001). Ce générateur est configuré avec 10 attributs, une vitesse de changement de 10^{-3} et 10% de bruit de classe.

Une méthode de l'état de l'art pour l'évaluation en-ligne des classifieurs (Gama et al., 2009) est utilisée. La méthode choisie mesure la performance en utilisant les exemples du flux comme données de test avant qu'ils ne soient appris. La mesure utilisée est la précision moyenne du classifieur entre le début du flux et l'instant t.

Notations :
x : un exemple du flux
W_{ref} : fenêtre de référence ($tailleW_{ref}$: sa taille)
W_{cur} : fenêtre sautante ($tailleW_{cur}$: sa taille)
$classif$: classifieur en cours d'utilisation ($tauxErreurClassif$: son taux d'erreur)
$nouvClassif$: classifieur qui apprend depuis la détection ($tauxErreurNouvClassif$: son taux d'erreur)
n : nombre d'exemples utilisés pour l'apprentissage du nouveau classifieur
n_{min} : nombre minimum d'exemples avant de comparer la performance des deux classifieurs

```
n ← 0
tant que x ← flux () faire
    // s'il y a un nouveau classifieur, est-il meilleur que l'ancien ?
    si estDémarré (nouvClassif) alors
        n ← n + 1
        tauxErrClassif ← tauxErreur (tauxErreurClassif, classif, x)
        tauxErrNouvClassif ← tauxErreur (tauxErreurNouvClassif, nouvClassif, x)
        apprendre (nouvClassif, x)
        si n > n_min et tauxErreurNouvClassif < tauxErreurClassif alors
            W_ref ← ∅
            W_cur ← ∅
            classif ← nouvClassif
            tauxErrClassif ← 0
            tauxErrNouvClassif ← 0
            n ← 0
    // remplissage des fenêtres
    si |W_ref| < tailleW_ref alors
        W_ref ← W_ref ∪ x
    sinon si |W_cur| < tailleW_cur alors
        W_cur ← W_cur ∪ x
    sinon
        pour i ← 1 à d faire
            // calcul du critère bivarié (formule (4) de (Boullé, 2009)) pour
               toutes les variables
            l ← l + CalculerMODLBivarié (W_ref, W_cur, i)
        // détection ?
        si l > 0 alors
            // changement détecté : apprentissage d'un nouveau classifieur
            créer (nouvClassif)
        sinon
            // flux stationnaire : plus besoin d'avoir un nouveau classifieur
            détruire (nouvClassif)
    apprendre (classif, x)
```

Algorithme 1: *Notre algorithme MDD (MODL Drift Detection) de remplacement du nouveau classifieur après changement de concept.*

4.3 Résultats

La figure 1 présente les résultats pour le jeu de données basé sur un « hyperplan en mouvement ». Sur cette figure toutes les méthodes permettent au classifieur d'avoir les mêmes performances respectivement pour le naïf Bayes et l'arbre de décision jusqu'à 100 000 et 200 000 exemples appris. Pour le classifieur bayésien naïf, entre 100 000 et 200 000 exemples DDM devient meilleur, notre méthode MDD reste relativement stable et EDDM bien moins

bon. Après 300 000 exemples alors que MDD reste stable, DDM devient bien moins bon et EDDM meilleur, mais moins bon que notre méthode. On observe que notre méthode de détection est meilleure et beaucoup plus stable que les deux autres méthodes. Pour l'arbre de décision, entre 200 000 et 600 000 exemples DDM devient meilleur, la méthode de détection MDD reste relativement stable et EDDM bien moins bon. Après 600 000 exemples alors que MDD continue à s'améliorer, DDM devient bien moins bon et EDDM un peu meilleur. Comme pour le classifieur bayésien naïf, on observe que notre méthode de détection est bien meilleure et beaucoup plus stable que les deux autres méthodes sur ce jeu de données.

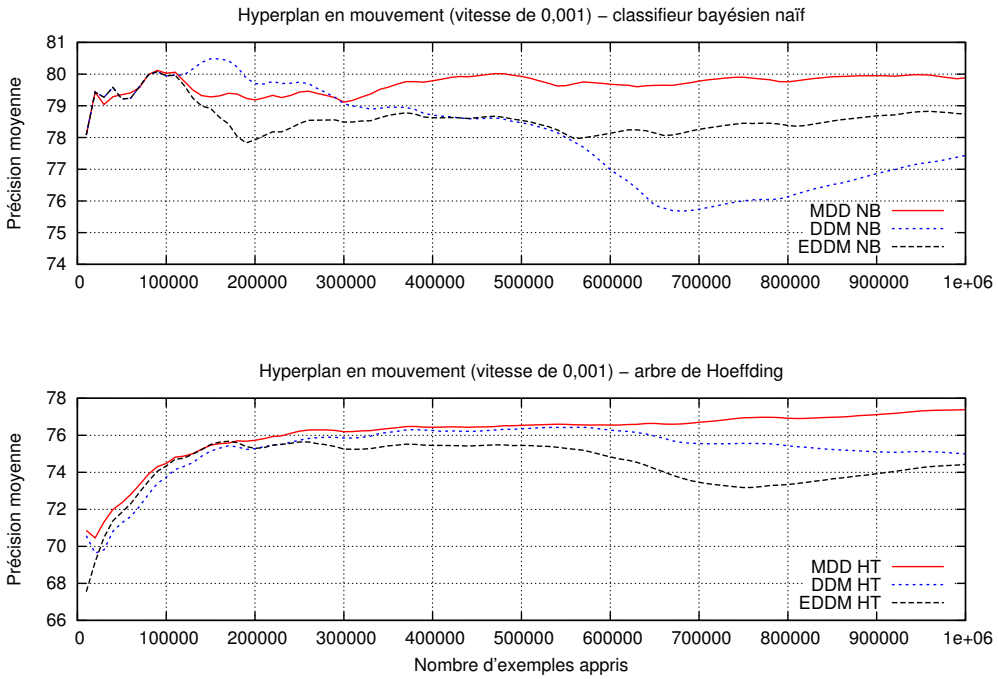

FIG. 1 – *Précision moyenne des différentes méthodes de détection sur le jeu de données basé sur un hyperplan en mouvement pour le classifieur bayésien naïf et l'arbre de Hoeffding.*

La figure 2 présente le nombre de détections et d'alertes des différents méthodes. On observe que le comportement de DDM et EDDM n'est pas stable bien que la vitesse de changement (la vitesse de rotation de l'hyperplan) soit constante. Ces méthodes ont tendance à détecter beaucoup de changements sur certaines périodes et aucun sur d'autres. Leurs hypothèses de départ (processus stationnaire et données iid), ne sont sans doute pas valides, ce qui aboutit à un manque de robustesse. Notre méthode de détection n'utilise pas le classifieur pour la détection mais seulement les données du flux. On observe que son nombre de détections est relativement régulier pour un changement ayant une vitesse constante.

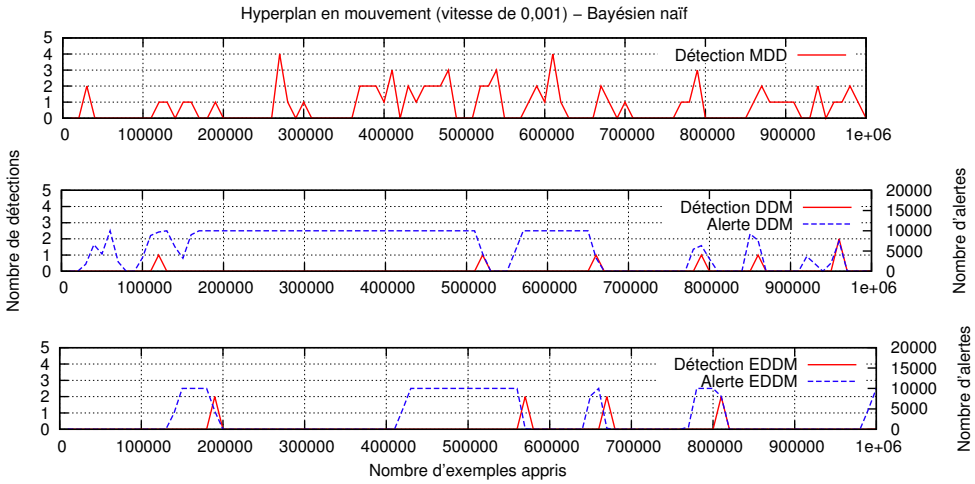

FIG. 2 – *Nombre de détections et alertes en fonctions du nombre d'exemples du flux pour différentes méthodes de détection pour le jeu de données basé sur un hyperplan en mouvement.*

5 Conclusion

Cet article présente une nouvelle méthode de détection de changement dans les flux de données. Celle-ci est basée sur l'observation du changement des distributions des exemples dans deux fenêtres. Notre méthode n'a pas d'*a priori* sur la distribution des données ni sur le type de changement à détecter. Elle est capable de détecter des changements rapides ou lents, que cela soit sur la moyenne, la variance ou tout autre changement dans la distribution. Notre méthode fait l'hypothèse d'indépendance des variables conditionnellement aux classes et est capable de détecter des changements sur les probabilités $P(W|X_i, C)$ en utilisant l'approche MODL. Chaque fenêtre est étiquetée et un changement est détecté s'il existe un modèle de discrétisation/groupage permettant de distinguer les deux fenêtres.

La nouvelle méthode que nous proposons utilise le critère bivarié MODL $P(W|X_i, C)$ qui, dans le cadre de notre méthode, est à la fois : (i) robuste dans le cas d'un flux stationnaire, (ii) rapide à détecter tous types de changements dans la distribution des données, (iii) capable d'utiliser l'information de classe.

Notre méthode a été comparée à deux méthodes de l'état de l'art détectant les variations de performance d'un classifieur. Nous avons proposé un nouvel algorithme, appelé MDD, basé sur notre méthode de détection permettant au classifieur de se mettre à jour. Celle-ci n'utilise pas le classifieur pour la détection mais seulement les données du flux. Ses performances, en termes de précision, sont meilleures et plus constantes que les deux méthodes de l'état de l'art testées.

Références

Baena-García, M., J. Del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, et R. Morales-Bueno (2006). Early Drift Detection Method. *Fourth International Workshop on Knowledge Discovery from Data Streams 6*, 77–86.

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

Bondu, A. et M. Boullé (2011). A supervised approach for change detection in data streams. *International Joint Conference on Neural Networks (IJCNN)*.

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

Boullé, M. (2009). Optimum simultaneous discretization with data grid models in supervised classification : a Bayesian model selection approach. *Advances in Data Analysis and Classification*.

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

Gama, J., P. Medas, G. Castillo, et P. Rodrigues (2004). Learning with drift detection. *Advances in Artificial Intelligence - SBIA 2004*, 286–295.

Gama, J., P. P. Rodrigues, R. Sebastiao, et P. Rodrigues (2009). Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 329–338. ACM New York, NY, USA.

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

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

Salperwyck, C. et V. Lemaire (2012). A two layers incremental discretization based on order statistics. *Advances in Data Analysis and Classification*.

Žliobaite, I. (2010). Learning under Concept Drift : an Overview. Technical report, https ://sites.google.com/site/zliobaite/Zliobaite_CDoverview.pdf.

Summary

We present an on-line method for concept change detection on labeled data streams. Our detection method uses a bivariate supervised criterion to determine if the data in two windows come from the same distribution. Our method has no hypothesis neither on data distribution nor on change type. It has the ability to detect changes of different kinds (mean, variance...). Experiments show that our method has better results than well-known methods from the literature. Moreover, except from the window sizes, no user parameter is required in our method.