Nouvelle méthode de détection de dérive basée sur la distance entre les erreurs de classification

Imen Khamassi, Moamar Sayed-Mouchaweh, Moez Hammami, Khaled Ghédira

Nouvelle méthode de détection de dérive basée sur la distance entre les erreurs de classification

Imen KHAMASSI[1], Moamar SAYED-MOUCHAWEH[2], Moez HAMMAMI[1], Khaled GHÉDIRA[1]

[1] Laboratoire de Stratégies d'Optimisation et Informatique intelligentE (SOIE),
Université de Tunis,
Institut Supérieur de Gestion de Tunis, La Tunisie.
imenkhamassi@yahoo.fr
moezhammami@gmail.com
khaled.ghedira@isg.rnu.tn

[2] Univ Lille Nord de France, F-59000 Lille, France
Mines-Douai, IA, F-59500 Douai, France.
moamar.sayed-mouchaweh@mines-douai.fr

Résumé— **La classification dynamique s'intéresse au traitement des données non-stationnaires issues des environnements évolutifs dans le temps. Ces données peuvent présenter des dérives, qui affectent la performance du modèle d'apprentissage initialement construit. Aujourd'hui, beaucoup d'intérêts sont portés sur la surveillance, la mise à jour et le diagnostic de ces dérives afin d'améliorer la performance du modèle d'apprentissage. Dans ce contexte, une nouvelle méthode de détection de dérive basée sur la distance entre les erreurs de classification est présentée. Cette méthode, nommée EDIST, surveille la distribution des distances des erreurs de classification entre deux fenêtres de données afin de détecter une différence à travers un test d'hypothèse statistique. EDIST a été testée à travers des bases de données artificielles et réelles. Des résultats encourageants ont été trouvés par rapport à des méthodes similaires. EDIST a pu trouver les meilleurs taux d'erreur de classification dans la plupart des cas et a montré une robustesse envers le bruit et les fausses alarmes.**

Mots-clés— **Apprentissage dans un environnement dynamique, détection de dérive, données non-stationnaires, apprentissage supervisé.**

I. INTRODUCTION

L'Apprentissage Automatique est le domaine de recherche scientifique qui tente de comprendre et de reproduire des facultés d'apprentissage à un système artificiel. Il s'agit plus précisément, de concevoir des systèmes d'apprentissage capables d'extraire à partir des informations disponibles, les connaissances utiles servant à la prise de décision.

Ces systèmes d'apprentissage ont la faculté d'apprendre d'une façon autonome de nouvelles connaissances (non connues *a priori*). Divers modèle d'apprentissage ont ainsi été développés pour la mise au point de Système d'Aide à la Décision de nos jours très utilisés dans de nombreuses applications: supervision des processus, reconnaissance vocale, reconnaissance de visages, …etc.

Lorsque les connaissances (déjà apprises) évoluent dans le temps, il est important que le système d'apprentissage soit capable de s'adapter en tenant compte de ces évolutions. Ces changements peuvent être dus soit à une variation normale des paramètres et/ou de la structure du système, soit à des dégradations affectant ses caractéristiques intrinsèques et son comportement.

Ces systèmes, appelés évolutifs, dynamiques ou non-stationnaires, couvrent un large champ d'application: électromécanique, chimique, thermique, médical, énergétique, … etc., et manipulent des données non-stationnaires qui peuvent présenter trois difficultés majeures. La première réside dans l'énorme quantité de données, constamment générées par des diverses applications au cours du temps. Ces flux de données sont potentiellement infinis, et il est presque impossible de les stocker. Seul un bref résumé de ces données peut être extrait et stocké pour des futures utilisations. La deuxième difficulté est due à la grande vitesse d'arrivée de ces flux de données qui empêche leurs traitements en temps réel. Enfin, la troisième difficulté émerge lorsque la distribution qui génère ces données évolue au cours du temps, c'est ce qu'on appelle la "dérive du concept". Formellement, le terme "concept" fait référence à la distribution de la probabilité jointe $P(X, w)$, où X représente les attributs d'entrée et w représente les classes. La dérive d'un concept peut être présentée par un changement *réel* ou *virtuel* dans cette distribution. Une dérive réelle affecte la probabilité à posteriori $P(w = w_i \backslash X)$ ce qui signifie que la classe obtenue pour les mêmes valeurs d'attributs peut changer; nous notons $w_i \in W_c$ l'ensemble des classes c avec $1 \leq i \leq c$. Une dérive virtuelle affecte la probabilité conditionnelle des classes $P(X \backslash w = w_i)$, ce qui signifie que la distribution qui génère les données d'une même classe change au cours du temps. Il convient de souligner que la dérive peut affecter également la probabilité a priori $P(w = w_i)$ d'une classe particulière, ce qui est connu par "l'évolution du concept". Il s'agit, entre autres, de mécanisme d'apparition de nouvelles classes ou de fusion des classes existantes [1].

Trois étapes sont nécessaires pour le traitement d'une dérive:

- *Étape de la surveillance* : lors de cette étape, les méthodes averties « informed methods » [2] intègrent des mécanismes de détection de dérive, pour fournir des descriptions concernant l'emplacement, la vitesse et la gravité de dérive. Ces mécanismes peuvent être basés sur des indicateurs de performance du modèle d'apprentissage, des paramètres de distributions de données ou de la structure du modèle.
- *Étape de la mise à jour*: lors de cette étape, les stratégies de mise à jour diffèrent selon les méthodes utilisées pour traiter les dérives. Les méthodes aveugles « blind methods » [3] procèdent par une adaptation implicite et régulière du modèle, sans aucun mécanisme de détection de dérive. Tandis que, les méthodes averties peuvent réinitialiser le modèle de nouveau, ou le mettre à jour en utilisant des données récentes lorsqu'une dérive est détectée. Cependant dans ce cas, deux difficultés peuvent se présenter : combien de données faut-il retenir ou oublier? Et quelle est la taille optimale de la fenêtre

de données afin de ne préserver que les données les plus significatives? Dans des études antérieures [4], la taille de la fenêtre de données était fixe. Toutefois, si la taille est trop petite, les données disponibles ne sont pas suffisantes pour construire le modèle; et si la taille est trop grande, le modèle peut conserver des données obsolètes pouvant réduire sa précision. La fenêtre de taille fixe peut bien fonctionner si la vitesse et la sévérité de dérive sont connues d'avance ou si on dispose des instructions rigoureuses provenant d'un expert, mais c'est rarement le cas. Des études récentes ont opté pour des fenêtres dont la taille est dynamique qui s'adapte à la dérive détectée ([5] et [6]). D'autres études [7] ont combiné plusieurs fenêtres de tailles différentes afin d'adapter le modèle aux changements progressivement.

- *Étape du diagnostic*: cette étape vise à interpréter les changements détectés dans les concepts ou dans les paramètres du modèle. Cette interprétation peut être utilisée comme un pronostic sur la tendance future du système. Il est à noter que lors de cette étape, il est important de faire la différence entre le bruit dans les données et les changements réels [8] et [9]. Idéalement, un compromis entre la robustesse du modèle contre le bruit et la flexibilité dans le suivi des dérives doit être atteint.

Face à ces défis, une nouvelle approche pour la détection de dérive, nommée EDIST, est proposée. Cette approche est basée sur la distance entre les erreurs de classification. Cette distance représente le nombre d'instances, de données, qui se trouvent entre deux erreurs de classification consécutives. EDIST surveille cette distance à travers deux fenêtres de données: W_0 qui est constituée des instances obtenues lorsqu'un nombre fixe d'erreurs se produisent, et la fenêtre globale W_G qui est ajustée de façon adaptative grâce à un test d'hypothèse statistique. Nous avons testé l'approche à travers des bases de données synthétiques en faisant varier la vitesse, la gravité et le temps d'apparition des dérives, et des bases réelles utilisées dans des études similaires. Des résultats encourageants ont été trouvés par rapport à des méthodes similaires. EDIST a trouvé les meilleurs taux d'erreur de classification dans la plupart des cas et a montré une robustesse envers le bruit et les fausses alarmes.

Le reste de l'article est organisé comme suit. Dans la *Section II*, une synthèse des travaux similaires dans la littérature est présentée. Dans la *Section III*, l'approche proposée est détaillée. Dans les *Section IV* et *V*, les expérimentations et les résultats obtenus sont présentés et analysés. Enfin, une conclusion et des travaux futurs sont discutés dans la *Section VI*.

II. SYNTHESE DES TRAVAUX SIMILAIRES DANS LA LITTERATURE

A. DDM: méthode de détection de dérive

La méthode de détection de dérive (Drift Detection Method (DDM)) [5] surveille le nombre d'erreurs produites par le modèle d'apprentissage et suppose que le taux d'erreur suit la loi binomiale. Dans un échantillon de *n* exemples, cette distribution donne la probabilité d'erreur de classification p_i avec un écart type $s_i = \sqrt{p_i(1-p_i)/i}$ pour chaque donnée ou instance i.

La méthode suppose que si la distribution des exemples est stationnaire, le taux d'erreur diminue avec l'augmentation de nombre de données. Une augmentation significative du taux d'erreur lors de l'apprentissage implique un changement dans la distribution générant les données. DDM enregistre p_{min} et s_{min} qui correspondent à la probabilité minimale et à l'écart-type minimal, puis définit deux niveaux comme suit:

- *Le niveau d'avertissement* lorsque $p_i + s_i \geq p_{min} + 2.\, s_{min}$; les données seront stockées pour un éventuel changement.
- *Le niveau de dérive* lorsque $p_i + s_i \geq p_{min} + 3.\, s_{min}$; la dérive sera confirmée et le modèle d'apprentissage est réinitialisé à partir des données enregistrées depuis le niveau d'avertissement. Notez que p_{min} et s_{min} sont également remis à zéro.

Cette méthode détecte facilement les dérives rapides. Cependant, elle a des difficultés dans la détection des dérives lentes.

B. EDDM : méthode rapide de détection de dérive

L'idée derrière (Eatly Drift Detection Method (EDDM)) [6] est de considérer la distance entre les erreurs de classification. Cette distance représente le nombre d'instances, qui se trouvent entre deux erreurs de classification consécutives. La méthode suppose que si la distribution de données est stationnaire, le modèle d'apprentissage permet d'améliorer la prédiction. Cela implique que la distance d'erreur augmente avec l'augmentation du nombre de données. Ainsi qu'une diminution significative de la distance entre les erreurs implique une dérive. Soient $d_1, d_2, d_3, \ldots d_i, \ldots$ les distances entre les erreurs de classification consécutives $e_1, e_2, e_3, \ldots e_i, \ldots$

EDDM calcule la moyenne de ces distances p'_i et l'écart type s'_i pour chaque erreur e_i ; les compare à la moyenne maximale p'_{max} et au écart type maximal s'_{max}, puis définit deux niveaux comme suit:

- *Le niveau d'avertissement* lorsque $(p'_i + 2.\, s'_i)/(p'_{max} + 2.\, s'_{max}) < \alpha$; les données seront stockées pour un éventuel changement.
- *Le niveau de dérive* lorsque $(p'_i + 2.\, s'_i)/(p'_{max} + 2.\, s'_{max}) < \beta$; la dérive sera confirmée et le modèle d'apprentissage est réinitialisé à partir des données enregistrées depuis le niveau d'avertissement. α et β sont respectivement fixés à 0,95 et 0,9.

EDDM est plus adaptée pour détecter les dérives lentes, mais elle présente une sensibilité aux valeurs de α et β, dans le sens où les grandes valeurs sont plus adaptées à la détection des dérives lentes alors que les petites valeurs sont plus adaptées à la détection des dérives rapides. Par conséquent, un compromis entre ces valeurs est nécessaire pour atteindre de bons résultats pour différents types de dérives.

III. EDIST : METHODE DE DETECTION DE DERIVE BASEE SUR LA DISTANCE ENTRE LES ERREURS DE CLASSIFICATION

Nous considérons le cadre de l'apprentissage en ligne où les instances, arrivent une par une et nous supposons que le modèle d'apprentissage est en mesure de faire une prédiction dès qu'une instance est disponible. Une fois que la prédiction est faite, le système peut apprendre à partir de ces instances en les intégrant au modèle d'apprentissage. Chaque instance se présente sous la forme de paires $(\vec{x}_i, w_i)$ où $\vec{x}_i$ est un vecteur d'attribut et w_i est l'étiquette, ou le label, de classe. La prédiction du modèle w'_i est dite correcte lorsque $w'_i = w_i$, incorrecte sinon.

L'idée d'EDIST est inspirée de la méthode de détection de dérive EDDM qui étudie la distance entre deux erreurs de classification consécutives. Cette distance représente le nombre d'instances qui se trouvent entre deux erreurs de classification consécutives.

Dans EDIST, la dérive est surveillée à travers deux fenêtres de données. La première représente la fenêtre globale W_G qui

est ajustée de manière adaptative. En effet, lorsqu'EDIST ne détecte pas de dérive, W_G est élargie, tandis que dans le cas contraire, elle est rétrécie pour ne contenir que les instances les plus significatives. La deuxième fenêtre W_0 représente un lot d'instances récentes et il faut noter que sa taille est mesurée à partir du nombre d'erreurs commises. Dans EDIST, la distribution des distances entre les erreurs de classification durant W_G et W_0 est estimée, puis la moyenne de ces distances est surveillée afin de détecter une dérive.

Dans EDIST, nous employons la même hypothèse utilisée dans [6], ce qui suppose que, si la distribution de données est stationnaire, le modèle d'apprentissage améliore sa prédiction et la distance entre les erreurs augmente. Ainsi que lorsque cette distance diminue au cours de l'apprentissage, cela implique qu'il y a un changement dans la distribution des exemples et que le modèle n'est plus approprié.

Contrairement à EDDM qui compare la moyenne des distances entre les erreurs de classification et l'écart-type à la moyenne maximale et son écart-type enregistrés à partir des données antérieures ; EDIST utilise un test d'hypothèse statistique afin de comparer les distributions des distances d'erreur de classification durant W_G et W_0 et vérifie si la différences entre les moyennes dépasse un seuil ε. L'originalité de notre méthode est que le seuil ε n'est pas défini a priori, dans le sens où il ne nécessite aucun réglage a priori en fonction de la vitesse ou de la gravité du changement. ε est ajusté d'une manière adaptative en fonction d'un test d'hypothèse statistique.

A. Test d'hypothèse statistique

Soit $d_1, d_2, d_3, \ldots d_t, \ldots$ la séquence des distances d'erreur lorsque chaque valeur d_t est disponible à l'instant t et générée à partir d'une distribution de distance d'erreur D_t.

Soient X_G et X_0 deux variables aléatoires suivant respectivement les deux distributions de distance d'erreur D_G et D_0 de W_G et W_0.

Nous supposons que $X_G \sim N\left(\mu_G, \frac{\sigma_G}{\sqrt{N}}\right)$ et $X_0 \sim N\left(\mu_0, \frac{\sigma_0}{\sqrt{n}}\right)$ suivent la loi Normale avec N et n les nombres d'erreurs de classification survenus durant respectivement W_G et W_0. Posons $\mu_d = \mu_G - \mu_0$ et définissons :

- *L'hypothèse nulle* (H_0) avec $\mu_d = 0$ où on suppose qu'il n'y a pas de changement entre les deux distances moyennes des distributions D_G et D_0.
- *L'hypothèse alternative* (H_1) avec $\mu_d > 0$ où on suppose que la distance moyenne de D_0 a diminué, ce qui implique qu'il y a un changement entre les deux distributions D_G et D_0.

B. Région d'acceptation

On suppose que H_0 est vraie, ce qui implique que la différence entre les moyenne est une variable aléatoire $X_d \sim N(\mu_d, \sigma_d)$ normalement distribuée avec $\mu_d = 0$ et $\sigma_d = \sqrt{\frac{\sigma_G}{\sqrt{N}} + \frac{\sigma_0}{\sqrt{n}}}$, et on pose α = 0,05 le risque de rejeter H_0 alors qu'elle est vraie.

On calcule ε tel que la probabilité pour que X_d soit inférieure ou égale à $\mu_d + \varepsilon$ est égale à 95% :

$$P(X_d \leq \mu_d + \varepsilon) = 0.95 \qquad (1)$$

Soit $T = \frac{X_d - \mu_d}{\sigma_d}$ une variable aléatoire qui suit la loi Normale centrée réduite $N(0,1)$, nous aurons donc:

$$P\left(T \leq \frac{\varepsilon}{\sigma_d}\right) = 0.95 \qquad (2)$$

ainsi que la fonction de répartition de la loi normale est définie comme suit :

$$\varphi\left(\frac{\varepsilon}{\sigma_d}\right) = 0.95 \qquad (3)$$

et d'après la table de la loi Normale, on peut écrire :

$$\frac{\varepsilon}{\sigma_d} = t_{1-\alpha} \qquad (4)$$

(4) peut s'écrire donc,

$$\varepsilon = t_{1-\alpha} * \sigma_d \qquad (5)$$

avec $\sigma_d = \sqrt{\frac{\sigma_G}{\sqrt{N}} + \frac{\sigma_0}{\sqrt{n}}}$ et $t_{1-\alpha} = t_{0.95} = 1.65$

C. Règle de décision

Si $\mu_d \leq \varepsilon$ alors on accepte l'hypothèse H_0 avec un risque d'erreur de 5% sinon on accepte l'hypothèse H_1.

Comme dans [5] et [6], EDSIT définit trois niveaux :

- *Le niveau du contrôle* : lorsque $\mu_d \leq \varepsilon$; on affirme qu'il n'y a pas de changement entre les deux distributions; la fenêtre W_G sera donc agrandie en ajoutant les instances de la fenêtre W_0, et le modèle d'apprentissage sera mis à jour à partir de la nouvelle fenêtre W_G . W_0 est ensuite réinitialisée afin de recueillir de nouvelles instances.
- *Le niveau d'avertissement* : lorsque $\mu_d > \varepsilon + r * \sigma_d$; les instances seront stockées pour une éventuelle détection de dérive. Toutefois, si la similitude entre les deux distributions pendant W_G et W_0 augmente de nouveau au cours de cette phase, on néglige ces instances et on considère cette alerte comme étant une fausse alarme.
- *Le niveau de dérive* : lorsque $\mu_d > \varepsilon + s * \sigma_d$; la dérive sera confirmée et W_G sera réinitialisée à partir des instances enregistrées depuis le niveau d'avertissement. Le modèle d'apprentissage sera ensuite reconstruit en utilisant les instances de cette nouvelle fenêtre W_G.

Il est à noter que r et s sont des valeurs entières qui représentent la quantité de changement pour définir respectivement les niveaux d'avertissement et de dérive; sachant que $s > r$.

Dans un contexte statique, lorsqu'on considère que $r = 2$, cela signifie que μ_d a une variance inférieure à $2 * \sigma_d$ avec 95% de confiance, tandis que si $s = 3$, cela signifie que μ_d a une variance inférieure à $3 * \sigma_d$ avec 99,7% de confiance. Dans la pratique, nous avons varié les valeurs de r et s de 0 à 3, afin d'étudier la relation entre ces valeurs et les différents types de dérive dans un contexte non-stationnaire.

IV. EVALUATION EXPERIMENTALE

EDIST a été exécuté sur PC Intel ® Core ™ 2 Duo CPU 2,26 GHz et 2,27 GHz avec 3Go de RAM, et programmé en Java en utilisant la plateforme (MOA : Massive Online Analysis). MOA [10] est une plateforme d'apprentissage en ligne utilisant des flux de données non-stationnaires. Il dérive de la plateforme WEKA, et contient une collection des méthodes d'apprentissage en ligne et hors ligne pour la classification et le clustering.

Pour évaluer EDIST, nous avons utilisé l'algorithme d'apprentissage Arbre de Hoeffding (Hoeffding Tree (HT)) [11]. HT est un arbre de décision par induction, qui est

capable d'apprendre à partir d'un flux de données massives. Il effectue la prédiction soit en choisissant la classe majoritaire issue de chaque feuille; soit en ajoutant un autre algorithme de prédiction comme le classifieur naïf de Bayes « Naive Bayes » aux feuilles. Nous avons utilisé HT programmé sur MOA en utilisant comme critère de répartition le gain d'information et le classifieur adaptative et naïf de Bayes « Adaptive Naive Bayes » comme algorithme d'apprentissage au niveau des feuilles.

A. Réglage des paramètres

Pour évaluer la performance de l'approche EDIST, nous avons utilisé la méthode d'évaluation présentée dans [12]. Cette méthode « predictive sequential method » a été développée pour évaluer la performance (erreur moyenne de classification ou prédiction) d'un modèle d'apprentissage ou un classifieur dans un environnement non-stationnaire. Elle évalue un classifieur à partir d'un flux de donnée en testant puis en apprenant à partir de chaque instance du flux. Cette méthode peut utiliser une fenêtre glissante ou un facteur de non-apprentissage (pour plus de détails voir [12]). Nous avons utilisé cette méthode d'évaluation avec une fenêtre glissante de taille 5000.

Les paramètres utilisés pour l'évaluation de l'approche EDIST sont les suivants:

- *NB* représente le nombre minimum de données pour initialiser le modèle d'apprentissage et il est fixé à 30
- *n* représente le nombre d'erreurs de classification survenu durant W_0 et il est fixé à 90
- *r* représente la quantité de changement du niveau d'alerte, et prend une des deux valeurs 0 ou 1
- *s* représente la quantité de changement du niveau de dérive et varie de 1 à 3; avec $s > r$.

Nous comparons notre méthode à DDM [5], EDDM [6] et à des multi-classifieurs bien connus comme: ADWIN Bagging[13], ADWIN Boosting[13], Leveraging Bagging[14], Accuracy Weighted Ensemble[3] et Accuracy Updated Ensemble[15]. Au cours des expérimentations, nous évaluons les indicateurs de performance (le pourcentage de classification correcte (PCC) et le pourcentage de fausses alarmes (PFA)) des différentes méthodes en utilisant des bases de données synthétiques et réelles.

B. Bases de données synthétiques

Les bases de données synthétiques sont primordiales pour l'étude du comportement de la méthode proposée ; puisque la vitesse, la sévérité et le temps de changement des dérives sont connus et paramétrables.

- *Hyperplane tournante* [16] a été largement utilisée pour simuler l'évolution d'un concept qui se repose sur le mouvement d'une hyperplane.
- *STAGGER* [17] dont les concepts sont des fonctions booléennes de trois attributs qui codent des objets de différentes tailles, formes et couleurs.
- *Waveform* [18] a pour but de faire la différence entre trois classes de forme ondulaire, dont chacune est produite à partir d'une combinaison de deux ou trois ondes de base.
- *Agrawal* [19] qui génère une fonction de prêt bancaire basée sur dix fonctions générant des classes binaires afin de déterminer si le prêt va être approuvé ou non.

C. Bases de données réelles

Dans les bases de données réelles, la vitesse, la sévérité et le temps de début des dérives ne sont pas connus, mais puisque ces données sont issues des environnements réels, on peut supposer qu'elles sont soumises implicitement à des changements qui les rendent non-stationnaires.

- *Electricity* [20] est une base de données réelle collectée à partir du marché d'électricité australien de New South Wales. Cette base contient 45312 instances dont les prix ne sont pas fixes et peuvent être affectés par l'offre et la demande.
- *Forest Covertype* [18] contient les types de couverture forestière pour des zones de 30 x 30 mètre obtenues à partir du service forestier des Etats Unis. Cette base contient 581012 instances et 54 attributs, et la tâche est de prédire les types de couverture forestière.
- *Poker-Hand* [18] se compose de 1000000 instances et 11 attributs. Chaque instance représente un tour à jouer qui est composé de 5 cartes à tirer parmi 52. Chaque instance contient 10 attributs, et un attribut de classe qui décrit le "Poker Hand".
- *Airlines* [18] est une base réelle collectée à partir d'une compagnie aérienne américaine. Elle se compose de 120 millions d'instances décrivant le départ et l'arrivée pour tous les vols commerciaux au sein des États-Unis, d'octobre 1987 à avril 2008. Elle contient 13 attributs et la tâche est de prédire si le vol sera retardé ou non.

V. RESULTATS

A. Détection de dérive

Dans ce paragraphe, nous avons comparé le pourcentage de classification correcte (PCC), le pourcentage de fausses alarmes (PFA) et le nombre de dérives non détectées pour DDM, EDDM et EDIST. Nous avons varié les vitesses et le nombre de dérives pour chacune des bases STRAGGER, Waveform et Agrawal. Les résultats sont exposés dans la Table I où EDIST a obtenu les meilleures performances en utilisant les bases STRAGGER et Agrawal avec le plus bas taux de fausses alarmes. Tandis qu'EDDM a obtenu le meilleur pourcentage de classification correct en utilisant la base Waveform mais avec un taux de fausses alarmes élevé.

B. Robustesse au bruit

Dans ce paragraphe, nous avons évalué la robustesse de chaque méthode contre le bruit. La Fig. 1 montre les résultats expérimentaux d'EDIST, DDM et EDDM en variant le bruit de 5% à 40% dans en utilisant la base Hyperplane contenant 100000 instances et une dérive de largeur 10000. EDIST a maintenu le meilleur pourcentage de classification correcte (PCC) pour les différentes intensités du bruit. Ces résultats confirment qu'EDIST présente un comportement stable et robuste envers le bruit surtout lorsque la dérive est lente.

Base de données	STAGGER			Waveform			Agrawal		
Nombre de dérive	2			3			4		
Largeur de dérive	50000			20000			10000		
Méthode de détection	DDM	EDDM	EDIST	DDM	EDDM	EDIST	DDM	EDDM	EDIST
PCC (%)	78,8	79	**80**	33,2	82	**81,1**	92,7	96	**96,4**
PFA (%)	25	61	14	0	82	18	23	64	14
Nombre de dérive non détectée	0	0	0	1	0	0	0	0	0

Table I: Le Pourcentage de Classification Correcte (PCC), le Pourcentage de Fausses Alarmes (PFA) et nombre de dérives non détectées pour DDM, EDDM et EDIST.

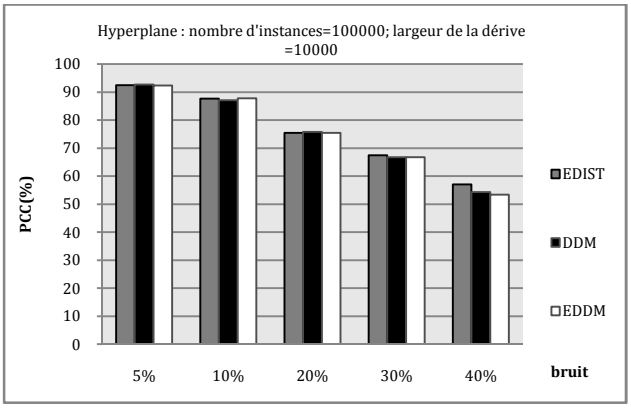

Fig.1: Etude de stabilité de DDM, EDDM et EDIST en présence de bruit.

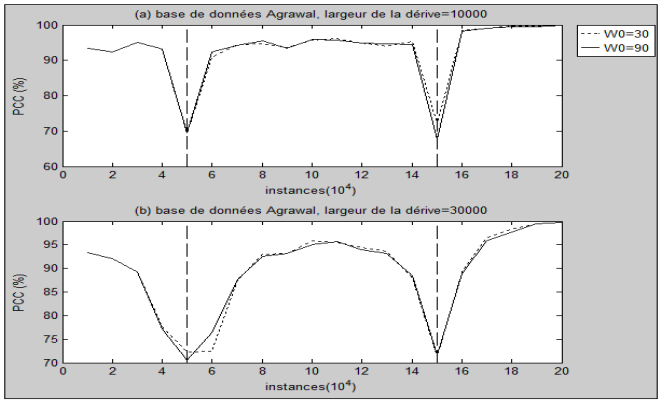

Fig.2: L'impact de la taille de la fenêtre W_0 dans le suivi de dérive dans la base Agrawal contenant 200000 instances et présentant deux dérives à t_0=50000 et 15000.

C. Sensibilité des paramètres r et s par rapport à la sévérité et à la largeur de dérive

Comme expliqué à la *Section III*, nous avons utilisé deux paramètres *r* et *s*, qui définissent la quantité du changement dans les niveaux d'avertissement et de dérive. A travers les expérimentations représentées dans les Tables II et III, nous avons étudié l'influence de ces deux paramètres sur la précision et la détection de dérive d'EDIST lorsqu'on fait varier la sévérité (sev) et la largeur de dérive.
Les expérimentations montrent que ces deux paramètres permettent un équilibre entre la précision de la prédiction et la robustesse aux fausses alarmes en présence de différents types de dérives. Dans le sens où les grandes valeurs de ces paramètres sont adéquates pour détecter les dérives lentes alors que les petites valeurs sont plus adaptées pour détecter les dérives rapides. Cela est dû au fait que lorsque la sévérité et la vitesse d'une dérive sont importantes, les changements au niveau des distributions sont plus perceptibles. Toutefois, lorsque la dérive est lente, la détection doit se faire avec un retard acceptable afin de collecter suffisamment de données pour réinitialiser le modèle d'apprentissage.

D. Impact de la taille de la fenêtre W_0 dans le suivi de dérive

Dans ce paragraphe, nous étudions l'impact de la taille de W_0 dans le suivi des dérives. Les courbes de suivi de dérive sont représentées dans la Fig.2, où la ligne continue correspond à W_0 = 90, la ligne en pointillé correspond à W_0= 30 et les barres verticales en pointillé correspondent aux positions des dérives. A partir de cette expérience, on constate que (i) lors d'une dérive, le pourcentage de classification correcte diminue au plus faible niveau puis converge rapidement. (ii) La convergence de la courbe correspondant à W_0 = 30 est plus rapide que celle de W_0= 90 lorsque la dérive est relativement rapide, comme le montre la Fig.2.a. Tandis que lorsque la dérive est plus lente la courbe correspondant à W_0= 90 converge plus rapidement que celle de W_0= 30 (Fig.2.b). Ces expérimentations montrent que le modèle d'apprentissage s'adapte progressivement aux changements avec un retard acceptable.

E. Comparaison des performances d'EDIST par rapport à des multi-classifieurs

Nous avons comparé EDIST avec les multi-classifieurs : ADWIN Bagging, ADWIN Boosting, Leveraging Bagging, Accuracy Weighted Ensemble (AWE) et Accuracy Updated Ensemble (AUE) à travers des bases réelles. Malgré les différentes caractéristiques de chaque base, des résultats encourageants ont été trouvés sauf pour le cas de Poker-Hand ; où EDIST n'a pas réussi à trouver le meilleure PCC. Cependant, EDIST a réalisé ces résultats en un temps d'exécution considérablement inférieur par rapport aux autres multi-classifieurs, ce qui explique la simplicité et l'efficacité de notre méthode (voir Table IV).

V. CONCLUSION

Dans cet article, nous avons présenté une nouvelle méthode de détection de dérive basée sur la distance entre les erreurs de classification, nommée EDIST. Cette méthode surveille la distribution des distances d'erreur de classification entre deux fenêtres de données. La première fenêtre W_G est une fenêtre globale ajustée d'une façon adaptative, et la deuxième fenêtre W_0 est constituée des instances obtenues lorsqu'un nombre fixe d'erreurs se produisent. EDIST utilise un test d'hypothèse statistique afin de comparer les distributions des distances de W_G et W_0 et vérifier si la différence des moyennes des distances durant ces deux fenêtres dépasse un seuil ε. L'originalité de notre méthode est que le seuil ε n'est pas défini a priori, dans le sens où il ne nécessite aucun réglage a priori en fonction de la vitesse ou de la sévérité de dérive. Nous avons testé l'approche à travers des bases de données synthétiques et réelles. Des résultats encourageants ont été trouvés par rapport à des méthodes similaires. EDIST a trouvé les meilleurs taux de classification correcte dans la plupart des cas et a montré une robustesse envers le bruit et les fausses alarmes.

Au cours des expérimentations, nous avons constaté l'importance des deux paramètres *r* et *s*, qui définissent respectivement la quantité du changement dans les niveaux d'avertissement et de dérive. Ces paramètres ont montré un effet considérable sur la robustesse de notre approche au bruit et aux fausses alarmes. Ainsi, dans les travaux futurs nous avons l'intention de développer ce point dans le but de donner une définition adaptative de ces paramètres en fonction de la sévérité et la vitesse de dérive.

Table II: Le pourcentage de classification correcte (PCC) and le nombre de dérives détectées (Nbre_dérives) dans la base Hyperplane contenant 100000 instances et présentant *une dérive* à t_0=50000 avec 30% de sévérité.

	largueur=500		largueur =1000		largueur =10000		largueur =40000		largueur =50000	
	PCC	Nbre dérives	PCC	Nbre dérives	PCC	Nbre dérives	PCC	Nbre dérives	PCC	Nbre dérives
EDIST r=0,s=1	**67,6**	14	**67,5**	14	**67,8**	8	67,6	9	66	10
EDIST r=1,s=2	67,3	4	67,2	4	67,6	5	**66,9**	3	66,7	2
EDIST r=1,s=3	67,3	4	67,3	3	67,4	3	66,1	1	**66,5**	1

Table III: Le pourcentage de classification correcte (PCC) and le nombre de dérives détectées (Nbre_dérives) dans la base Hyperplane contenant 100000 instances et présentant *une dérive* à t_0=50000 avec une largeur = 10000.

	sev 20%		sev 30%		sev 40%		sev 50%		sev 60%	
	PCC	Nbre dérives	PCC	Nbre dérives	PCC	Nbre dérives	PCC	Nbre dérives	PCC	Nbre dérives
EDIST r=0,s=1	**67,6**	10	**67,8**	8	66,4	10	66,2	9	67,1	11
EDIST r=1,s=2	66,9	5	67,6	5	**67,3**	5	67,2	5	67,3	5
EDIST r=1,s=3	66,9	3	67,4	3	67,2	3	**67,4**	3	**67,3**	3

Table IV: Le pourcentage de classification correcte (PCC) et le temps d'exécution (CPU) en seconde dans des bases de données réelles.

	ELEC2		Covtyp		Poker		Airline	
	PCC	CPU	PCC	CPU	PCC	CPU	PCC	CPU
EDIST	**89,7**	0,7	**97,4**	49,1	82,8	27,11	**65,9**	37,7
AccuracyUpdatedEnsemble	69,2	10,7	94,9	1089,5	68,1	314,4	54,7	740,2
AccuracyWeightedEnsemble	78,2	9,7	95,5	993,1	56,2	197,2	57,4	768,2
LeveragingBag	89,5	14,4	97	503,1	**99,4**	224,8	61,1	959,5
OzaBagAdwin	75,3	5,4	97,1	368,3	65,5	111,4	61,6	330,8
OzaBoostAdwin	88	22,2	97,2	1704,3	85,8	1123,2	58,2	739,6

RÉFÉRENCES

[1] Masud M., Gao J., Khan L., Han J., et Thuraisingham B. Classification and novel class detection in concept-drifting data streams under time constraints. *IEEE Transactions on Knowledge and Data Engineering*, vol 23, n°6, pp 859–874, 2011.

[2] Ikonomovska E, Gama J., Sebastio R., et Gjorgjevik D. Regression trees from data streams with drift detection. 12th International Conference on Discovery Science, Berlin, Germany, 2009.

[3] Kolter J. et Maloof M. Dynamic weighted majority: a new ensemble method for tracking concept drift. *The Journal of Machine Learning Research*, vol. 8,n°1, pp 2755-2790, 2007.

[4] Sobhani P. et Beigy H. New drift detection method for data streams. 2nd international conference on Adaptive and intelligent systems, Berlin, Germany, 2011.

[5] Gama J.,. Medas P, Castillo G., et Rodrigues P. Learning with local drift detection. *2nd International Conference on Advanced Data Mining and Applications*, Xi'an, China, 2006.

[6] Baena-García M., Campo-Avila J. D, Fidalgo R., Bifet A.,Gavaldà R., et Morales-Bueno R. Early drift detection method. 4th International Workshop on Knowledge Discovery from Data Streams, Berlin, Germany, 2006.

[7] Lazarescu M., Venkatesh S. et Bui H. Using multiple windows to track concept drift. *Intelligent data analysis*, vol. 8, n°1, pp. 29-59, 2004.

[8] Sayed-Mouchaweh M. Semi-supervised classification method for dynamic applications. *Fuzzy Sets and Systems*, vol.161, n°4, pp 544–563, 2010.

[9] Lughofer E. et Angelov P. Handling Drifts and Shifts in On-Line Data Streams with Evolving Fuzzy Systems. *Applied Soft Computing*, vol.11, n°2, pp 2057-2068, 2011.

[10] Bifet A., Holmes G., Kirkby R., et Pfahringer B. MOA: Massive Online Analysis. *Journal of Machine Learning Research*, vol.11, n°4, pp 1601-1604, 2010.

[11] Hulten G., Spencer L., et Domingos P. Mining time-changing data streams". 7th international conference on Knowledge Discovery and Data Mining, KDD'01, San Francisco, CA, 2001.

[12] Gama J., Sebastião R. et Rodrigues P. Issues in evaluation of stream learning algorithms. 15th international conference on Knowledge discovery and data mining, KDD'09, Paris, France, 2009.

[13] Bifet A., Holmes G., Pfahringer B., Kirkby R., et Gavalda R. New ensemble methods for evolving data streams. 15th international conference on Knowledge discovery and data mining, KDD'09, Paris, France, 2009.

[14] Bifet A., Holmes G.,et Pfahringer B. Leveraging Bagging for Evolving Data Streams Machine Learning and KnowledgeDiscovery in Databases. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD, Barcelona, Spain, 2010.

[15] Brzezinski D., et Stefanowski J. Accuracy Updated Ensemble for Data Streams with Concept Drift. 6th international conference on hybrid artificial intelligent systems,Wroclaw, Poland, 2011.

[16] Hulten G., Spencer L., et Domingos P. Mining time-changing data streams. 7th international conference on Knowledge Discovery and Data Mining, KDD'01, California, USA, 2001.

[17] Schlimmer J. C. et Granger R. H. Incremental learning from noisy data. *Machine Learning*, vol.1, n°3, pp 317-354, 1986.

[18] Asuncion A. et Newman D. UCI machine learning repository, 2007.

[19] Agrawal R., Imielinski T., et Swami A. Database mining: A performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, vol.5, n°6, pp 914–925, 1993.

[20] Harries M. Splice-2 comparative evaluation: Electricity pricing. Rapport technique, The University of South Wales, Autralia, 1999.