# Classification et apprentissage actif à partir d'un flux de données évolutif en présence d'étiquetage incertain

Mohamed-Rafik Bouguelia

UNIVERSITÉ DE LORRAINE

École doctorale IAEM Lorraine

# Classification et apprentissage actif à partir d'un flux de données évolutif en présence d'étiquetage incertain

## THÈSE

présentée et soutenue publiquement le 25 mars 2015

pour l'obtention du

**Doctorat de l'Université de Lorraine**

**(mention informatique)**

par

Mohamed-Rafik Bouguelia

**Composition du jury**

| | | |
|---|---|---|
| *Président :* | Rolf Ingold | Professeur, Université de Fribourg |
| *Rapporteurs :* | Robert Sabourin | Professeur, École de Technologie Supérieure, Montréal |
| | Laurence Likforman-Sulem | Maître de Conférence HDR, Télécom ParisTech |
| *Examinateurs :* | Thierry Paquet | Professeur, Université de Rouen |
| | Yolande Belaïd | Maître de conférence, Université de Lorraine |
| *Invité :* | Vincent Poulain d'Andecy | Directeur de projets de recherche, ITESOFT |
| *Directeur de thèse :* | Abdel Belaïd | Professeur, Université de Lorraine |

**Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503**

Mis en page avec la classe thesul.

# Remerciements

Je remercie tout d'abord Dieu tout puissant de m'avoir donné le courage, la force et la patience d'achever ce modeste travail.

Je remercie mon directeur de thèse Abdelwaheb Belaïd pour avoir dirigé ma thèse avec beaucoup d'efforts et de patience. Je tiens aussi à remercier Mme Yolande Belaïd pour son encadrement et ses précieux conseils, elle fut tout le temps attentive et disponible malgré ses nombreuses charges. J'exprime tous mes remerciements à l'ensemble des membres de mon jury : Madame Laurence Likforman-Sulem, Monsieur Rolf Ingold, Monsieur Robert Sabourin, Monsieur Thierry Paquet, Monsieur Vincent Poulain d'Andecy. Je remercie également l'ensemble des membres de l'équipe READ du LORIA, ainsi que tous mes amis qui m'ont apporté leur soutien moral pendant cet année d'études, je les en remercie sincèrement.

Je tiens à exprimer mes plus vifs remerciements à la personne la plus importante de ma vie. Il s'agit de ma merveilleuse, splendide et vertueuse épouse qui m'a gratifié de son amour. Je lui adresse toute ma gratitude du fond du cœur.

Enfin, les mots les plus simples étant les plus forts, j'adresse toute mon affection à ma famille, et en particulier à mes parents. Malgré mon éloignement depuis de (trop) nombreuses années, leur confiance, leur tendresse, leur amour me portent et me guident tous les jours. Merci pour avoir fait de moi ce que je suis aujourd'hui. Je vous aime.

Pour terminer, un grand merci aux personnes qui ont cru en moi et qui m'ont permis d'arriver au bout de cette thèse.

*à ma très chère femme,*
*et à mes très chers parents.*

# Sommaire

# Résumé

Cette thèse traite de l'apprentissage automatique pour la classification de données. Afin de réduire le coût de l'étiquetage, l'apprentissage actif permet de formuler des requêtes pour demander à un opérateur d'étiqueter seulement quelques données choisies selon un critère d'importance. Nous proposons une nouvelle mesure d'incertitude qui permet de caractériser l'importance des données et qui améliore les performances de l'apprentissage actif par rapport aux mesures existantes. Cette mesure détermine le plus petit poids nécessaire à associer à une nouvelle donnée pour que le classifieur change sa prédiction concernant cette donnée. Nous intégrons ensuite le fait que les données à traiter arrivent en continu dans un flux de longueur infinie. Nous proposons alors un seuil d'incertitude adaptatif qui convient pour un apprentissage actif à partir d'un flux de données et qui réalise un compromis entre le nombre d'erreurs de classification et le nombre d'étiquettes de classes demandées. Les méthodes existantes d'apprentissage actif à partir de flux de données, sont initialisées avec quelques données étiquetées qui couvrent toutes les classes possibles. Cependant, dans de nombreuses applications, la nature évolutive du flux fait que de nouvelles classes peuvent apparaître à tout moment. Nous proposons une méthode efficace de détection active de nouvelles classes dans un flux de données multi-classes. Cette méthode détermine de façon incrémentale une zone couverte par les classes connues, et détecte les données qui sont extérieures à cette zone et proches entre elles, comme étant de nouvelles classes. Enfin, il est souvent difficile d'obtenir un étiquetage totalement fiable car l'opérateur humain est sujet à des erreurs d'étiquetage qui réduisent les performances du classifieur appris. Cette problématique a été résolue par l'introduction d'une mesure qui reflète le degré de désaccord entre la classe donnée manuellement et la classe prédite et une nouvelle mesure d'"informativité" permettant d'exprimer la nécessité pour une donnée mal étiquetée d'être ré-étiquetée par un opérateur alternatif.



**Mots-clés:** Classification, Apprentissage actif, Flux de données, Détection de nouveautés, Erreurs d'étiquetage



# Abstract

This thesis focuses on machine learning for data classification. To reduce the labelling cost, active learning allows to query the class label of only some important instances from a human labeller. We propose a new uncertainty measure that characterizes the importance of data and improves the performance of active learning compared to the existing uncertainty measures. This measure determines the smallest instance weight to associate with new data, so that the classifier changes its prediction concerning this data. We then consider a setting where the data arrives continuously from an infinite length stream. We propose an adaptive uncertainty threshold that is suitable for active learning in the streaming setting and achieves a compromise between the number of classification errors and the number of required labels. The existing stream-based active learning methods are initialized with some labelled instances that cover all possible classes. However, in many applications, the evolving nature of the stream implies that new classes can appear at any time. We propose an effective method of active detection of novel classes in a multi-class data stream. This method incrementally maintains a feature space area which is covered by the known classes, and detects those instances that are self-similar and external to that area as novel classes. Finally, it is often difficult to get a completely reliable labelling because the human labeller is subject to labelling errors that reduce the performance of the learned classifier. This problem was solved

by introducing a measure that reflects the degree of disagreement between the manually given class and the predicted class, and a new informativeness measure that expresses the necessity for a mislabelled instance to be re-labeled by an alternative labeller.

**Keywords:** Classification, Active learning, Data stream, Novelty detection, Label noise.

# Notations

1. $\mathbb{R}$ est l'espace des nombres réels
2. $\mathbb{N}$ est l'espace des nombres entiers
3. $d$ est la dimension de l'espace
4. $X \subseteq \mathbb{R}^d$ est l'espace des instances (aussi appelé espace des entrées ou des caractéristiques), où $d$ est le nombre de dimensions (ou de caractéristiques)
5. $x \in X$ est un vecteur de caractéristiques représentant une instance (une donnée)
6. $x_i$ est une $i^{\text{ème}}$ instance
7. $x_i^{(j)}$ la $j^{\text{ème}}$ composante de l'instance $x_i$ (c.à.d. la valeur de la $j^{eme}$ caractéristique pour l'instance $x_i$)
8. $Y$ est l'ensemble des classes de données
9. $y \in Y$ est une étiquette de classe (un nom unique d'une classe)
10. $(x, y)$ est un couple représentant une instance $x$ étiquetée avec $y$ ($x$ est dit membre de la classe $y$)
11. $h$ est un modèle de classification (classifieur) présenté sous forme d'une fonction $h : X \to Y$, tel que $h(x) = y$
12. $\underset{i}{\operatorname{argmax}} F(i)$ permet la sélection de la valeur $i$ qui donne la valeur maximale pour la fonction $F(i)$
13. $\underset{i}{\operatorname{argmin}} F(i)$ permet la sélection de la valeur $i$ qui donne la valeur minimale pour la fonction $F(i)$
14. $\mathcal{O}$ est l'ordre de grandeur d'une complexité algorithmique
15. $dist(x_1, x_2)$ est la distance euclidienne entre deux instances $x_1$ et $x_2$
16. $L$ est un ensemble de données étiquetées
17. $U$ est un ensemble de données non étiquetées
18. $H(x, r)$ est l'hypersphère de centre $x$ et de rayon $r$
19. $p$ est la mesure de probabilité
20. $w$ est un poids associé à une instance
21. $\hat{w}$ est le plus petit poids (poids suffisant) à associer à une instance étiquetée pour faire basculer la prédiction du classifieur.
22. $\epsilon$ est une constante représentant le taux d'apprentissage $0 < \epsilon \ll 1$
23. $\theta$ est le seuil d'incertitude
24. $y_o$ l'étiquette observée pour une instance $x$ (donnée par un opérateur).
25. SVM est l'abréviation de Séparateur à Vaste Marge ("**S**upport **V**ector **M**achine")
26. $k$NN est l'abréviation de la méthode des $k$ plus proches voisins ("$k$ **N**earest **N**eighbours")

# Chapitre 1

# Introduction générale

**Sommaire**



## 1.1 Contexte

Le cadre général de cette thèse concerne les systèmes capables d'améliorer leur performance avec l'expérience. Ce vaste domaine, connu sous le nom d'apprentissage automatique, a pour objectif de développer des algorithmes permettant aux machines d'apprendre à partir de données existantes. Cet apprentissage est effectué dans le but d'accomplir des tâches précises impliquant la prise de décisions concernant de nouvelles données. Dans cette thèse, nous nous intéressons plus particulièrement à la tâche de classification automatique de données, qui consiste à attribuer une classe ou catégorie à chaque donnée.

Cette thèse a été réalisée dans le contexte du projet industriel DoD ("Documents on Demand") proposé par la société ITESOFT. Cette société est un éditeur de logiciels novateurs spécialisé dans différents aspects de l'analyse et la gestion de documents. Elle traite des milliers de documents administratifs hétérogènes qui sont quotidiennement numérisés. Leur traitement doit être rapide et efficace afin de permettre un gain de productivité pour les administrations et un gain de réactivité et de temps de réponse pour les utilisateurs. Dans le contexte du projet DoD, les documents arrivent sous forme d'un flux continu, et la tâche de classification automatique intervient à différents niveaux.

Premièrement, la classification des documents dans différentes catégories telles que des chèques bancaires, des reçus de frais médicaux, des factures, des ordonnances, etc. permet de les diriger automatiquement vers des dispositifs de traitements spécifiques (selon leur catégorie) ou vers des services spécialisés dans leur gestion. Leur classification par thème permet également de donner du contexte au texte du document afin de lever des ambiguïtés concernant la reconnaissance de certains mots clés. Deuxièmement, la classification des zones contenues dans un document administratif en différentes classes telles que "logos", "signatures", "tables", "annotations manuscrites", "tampons", etc. permet d'identifier le type de ces zones afin d'utiliser des techniques de reconnaissance spécifiques à leur contenu. Par exemple, si nous savons qu'une zone à l'intérieur d'un document représente un tableau, alors des techniques spécialisées dans l'extraction des informations à partir de tableaux peuvent être utilisées. Enfin, les notions d'apprentissage et de classification abordées dans cette thèse sont génériques et peuvent concerner d'autres

données traitées par la société (e.g. courriers électroniques), mais aussi diverses autres applications en dehors du contexte d'ITESOFT.

## 1.2 Problématique

Il existe plusieurs types d'apprentissage automatique pour la classification : supervisé, non-supervisé, semi supervisé et actif. L'apprentissage supervisé permet aux utilisateurs de préciser les classes voulues selon les besoins spécifiques de l'application. Il utilise une base de données d'apprentissage dans laquelle chaque donnée est étiquetée manuellement avec sa classe respective. L'obtention d'un classifieur performant nécessite l'étiquetage manuel d'un nombre suffisamment grand de données pour chacune des classes, ce qui s'avère être coûteux. L'apprentissage non supervisé ne nécessite pas d'étiquetage et essaie plutôt de découvrir des clusters (groupes de données proches entre elles) dans un ensemble de données non étiquetées. Mais les clusters découverts ne correspondent pas toujours aux classes spécifiques voulues par l'utilisateur. L'apprentissage semi-supervisé permet aux utilisateurs de spécifier des classes sans nécessiter beaucoup d'étiquetage. En effet, les techniques d'apprentissage semi-supervisé peuvent apprendre en utilisant à la fois des données étiquetées et non étiquetées, ce qui permet d'alléger le coût d'étiquetage. Enfin, l'apprentissage actif permet de demander à un opérateur l'étiquetage de quelques données "importantes" sélectionnées de manière pertinente, ce qui réduit considérablement le coût d'étiquetage. Le principal problème à résoudre pour un apprentissage actif concerne la façon de sélectionner les données à étiqueter. Ceci n'est pas un problème trivial car une solution idéale doit permettre à la fois (i) d'étiqueter autant de données que nécessaires pour avoir un classifieur performant, et (ii) de réduire au maximum le coût d'étiquetage, qui peut être estimé par le nombre de données à étiqueter.

Par ailleurs, quel que soit le type d'apprentissage utilisé pour entraîner un classifieur, la plupart des méthodes existantes ont besoin de connaître à l'avance l'ensemble des données utilisées pour l'apprentissage. Ceci n'est pas possible dans le cas d'un flux où les données arrivent en continu et deviennent disponibles au fil du temps (flux infini). Dans ce cas, l'apprentissage se fait au fur et à mesure de l'arrivée des données : chaque nouvelle donnée est utilisée immédiatement et une seule fois, pour mettre à jour le modèle de façon incrémentale. La plupart des méthodes d'apprentissage à partir de flux de données supposent que le nombre de classes de données dans le flux est connu. Cependant, dans les applications réelles, cette supposition est souvent incorrecte. En effet, il est difficile d'obtenir des échantillons de données étiquetées à partir de toutes les classes de données possibles qui apparaîtront dans le futur. De plus, la nature évolutive du flux fait que de nouvelles classes peuvent apparaître à tout moment. Si une nouvelle classe n'est pas automatiquement détectée, toutes les données de cette classe seront inévitablement classées par erreur dans des classes existantes (connues). L'identification de nouvelles classes est donc importante pour éviter de telles erreurs de classification.

Les méthodes d'apprentissage actif à partir de flux de données supposent implicitement que les données utilisées pour leur initialisation, couvrent toutes les classes possibles, et demandent à un opérateur les étiquettes des données "importantes" dont l'importance est déterminée selon ces classes connues. La détection de nouvelles classes n'est pas triviale pour ces méthodes car elles doivent demander des étiquettes pour des données se trouvant dans différentes régions inexplorées de l'espace des entrées, ce qui rend la réduction du coût d'étiquetage plus difficile.

En outre, les méthodes d'apprentissage existantes, qu'elles soient supervisées, semi-supervisées ou actives, supposent que les étiquettes observées (i.e. données par un opérateur) sont parfaitement correctes. Cependant, dans des applications du monde réel, cette hypothèse n'est souvent pas satisfaite. En effet, il est difficile d'obtenir des étiquettes totalement fiables car l'opérateur humain peut commettre des

erreurs d'étiquetage qui réduisent les performances de classification du modèle appris.

L'apprentissage actif permet de demander l'étiquetage des instances qui sont susceptibles d'améliorer le modèle de classification si l'on suppose que les étiquettes de classes obtenues sont correctes. La présence d'erreurs d'étiquetage fait que l'algorithme a besoin de plus d'étiquetage afin de converger vers un bon modèle, ce qui rend la réduction du coût d'étiquetage encore plus difficile.

À travers cette discussion, apparaissent trois aspects principaux qui doivent être pris en compte par l'apprentissage et qui sont traités dans cette thèse :

1. Minimisation des interventions humaines pour :
   - L'étiquetage manuel des données utilisées pour l'apprentissage.
   - Le paramétrage du système.
2. Adaptation au caractère évolutif du flux de données qui implique l'émergence de nouvelles classes non apprises et qui n'ont jamais été vues auparavant.
3. Apprentissage en présence d'erreurs d'étiquetage dues à la non fiabilité de l'opérateur qui supervise le système.

## 1.3 Organisation du mémoire

Ce mémoire est organisé comme suit. Dans le **chapitre** 2, nous introduisons la terminologie relative à la tâche de classification puis les différents types d'apprentissage, avant de justifier notre choix d'apprentissage actif semi-supervisé. Nous présentons également les bases des données utilisées lors des différents expérimentations.

Dans le **chapitre** 3, nous exposons les stratégies utilisées en apprentissage actif pour choisir les données qui améliorent l'apprentissage (données informatives) et qui sont donc manuellement étiquetées de sorte à minimiser l'effort d'étiquetage fourni par l'humain. Nous proposons ensuite une nouvelle stratégie d'incertitude qui permet de caractériser l'"informativité" des données et qui améliore les performances de l'apprentissage actif par rapport aux stratégies d'incertitudes existantes. Des expérimentations sont ensuite présentées pour prouver l'efficacité de la stratégie d'apprentissage actif proposée.

Dans le **chapitre** 4, nous étudions des méthodes basées sur l'algorithme NG ("Neural Gas"), permettant d'apprendre de façon non supervisée la topologie des données en entrée. Nous proposons ensuite une extension semi-supervisée et active, de l'un de ces algorithmes. La méthode proposée dans ce chapitre constitue principalement un point de départ pour la méthode de détection de nouvelles classes proposée dans le chapitre suivant.

Dans le **chapitre** 5, nous traitons le problème de détection active de nouvelles classes dans un flux de données. Tout d'abord, nous étudions les méthodes qui sont habituellement utilisées dans la littérature pour la détection de nouveautés. Comme, elles ne sont pas adaptées à un apprentissage actif à partir d'un flux de données multi-classes, nous étendons l'algorithme proposé dans le chapitre 4 pour détecter automatiquement les nouvelles classes dans le flux. Le chapitre se termine par des expérimentations qui montrent l'efficacité de la méthode de détection de nouvelles classes proposée.

Dans le **chapitre** 6, nous abordons le problème de gestion des erreurs d'étiquetage. Tout d'abord, nous présentons des méthodes existantes qui permettent de filtrer des données mal étiquetées ou de réduire leur impact sur les résultats de classification. Ensuite nous proposons une méthode pour traiter ce problème dans le cadre d'un apprentissage actif à partir d'un flux de données.

Ce mémoire se termine par une discussion sur ce qui est accompli ainsi que les perspectives à court et long terme dans le **chapitre** 7.

## 1.4 Résumé des contributions

### Stratégie d'apprentissage actif à partir de flux de données

Nous proposons une stratégie qui détermine le plus petit poids nécessaire à associer à une nouvelle donnée pour que le classifieur change sa prédiction concernant cette donnée. Si un petit poids est suffisant pour modifier la prédiction, le classifieur est considéré incertain, et la vraie classe de cette donnée est demandée à l'opérateur. Afin de déterminer si le poids suffisant est "assez petit", nous proposons un seuil d'incertitude adaptatif qui convient pour un apprentissage actif à partir d'un flux de données, tout en faisant un compromis entre le nombre d'erreurs et le nombre d'étiquettes de classe demandées. La stratégie proposée permet également de s'abstenir de classifier des données incertaines afin de réduire le taux d'erreurs. Ainsi, la contribution de cette partie se résume dans les points suivants :

- Une nouvelle mesure d'incertitude basée sur la pondération des instances.
- Un seuil d'incertitude adaptatif pour l'apprentissage actif à partir d'un flux de données.
- Un algorithme d'apprentissage incrémental semi-supervisé actif pour la classification de flux de données.

Ces contributions ont donné lieu à 5 publications [Bouguelia 15(3)], [Bouguelia 15(2)], [Bouguelia 13(3)], [Bouguelia 13(2)], [Daher 14].

### Détection de nouvelles classes dans un flux de données

Nous proposons une méthode de détection de classes inconnues, qui maintient de façon incrémentale une zone couverte par les classes connues (les régions de l'espace des entrées où les données de classes connues ont été observées), et détecte les données qui sont extérieures à cette zone et proches entre elles, comme étant de nouvelles classes. L'algorithme permet de demander les vraies étiquettes de classes seulement pour les instances qui permettent de mieux discriminer entre les classes existantes et les nouvelles classes identifiées. Ainsi, la contribution de cette partie se résume dans les points suivants :

- Méthode adaptative pour la détection, en temps réel, de données qui sont étrangères aux classes. connues
- Distinction efficace entre les classes nouvelles et existantes.

La contribution de cette partie a donné lieu à trois articles [Bouguelia 14(3)], [Bouguelia 14(2)], [Bouguelia 14(1)].

### Gestion des erreurs d'étiquetage

Concernant la gestion des erreurs d'étiquetage, nous développons les questions suivantes : parmi les instances dont les étiquettes ont été demandées, lesquelles sont les plus susceptibles d'être mal étiquetées ? Faut il toujours s'abstenir d'apprendre lorsqu'on suspecte qu'une donnée est mal étiquetée ? Parmi les instances mal étiquetées lesquelles nécessitent un ré-étiquetage ? Nous proposons une mesure qui reflète la vraisemblance de mauvais étiquetage (le degré de désaccord entre la classe donnée manuellement et la classe prédite) pour filtrer dès réception les instances potentiellement mal étiquetées. Un opérateur expert alternatif peut ainsi être sollicité pour corriger l'instance filtrée. La méthode est capable de sélectionner (pour ré-étiquetage) seulement les instances qui méritent d'être corrigées selon une nouvelle mesure d'informativité. Ainsi, la contribution de cette partie se résume dans les points suivants :

- Mesure du degré de désaccord entre la classe donnée manuellement et la classe prédite.
- Mesure d'informativité permettant d'exprimer combien une donnée mal étiquetée mérite d'être ré-étiquetée.

Cette contribution a été publiée dans [Bouguelia 15(1)].

# Chapitre 2

# Préliminaires

**Sommaire**



## 2.1 Classification

La classification est la tâche consistant à attribuer une classe à une donnée qu'on veut classer, ou autrement dit, à assigner une donnée à une classe de données.

Plus formellement, soit $X \subseteq \mathbb{R}^d$ un ensemble représentant un espace à $d$ dimensions, appelé l'*espace des instances*. La donnée $x \in X$ est appelée une *instance* et représente un point dans l'espace $X$. L'instance $x$ est présentée sous forme d'un vecteur de taille $d$, $x = (x^{(1)}, ..., x^{(d)})$, où chaque composante $x^{(i)} \in \mathbb{R}$ est une valeur discrète ou continue. Soit $Y$ un ensemble *fini* de classes où chaque classe $y \in Y$ est présentée sous forme d'une valeur discrète appelée *étiquette de classe* (un nom ou identifiant unique pour la classe). Le classifieur se présente alors sous forme d'une fonction de classification $h$ (appelée aussi modèle de classification) permettant d'associer une donnée $x \in X$ à une étiquette de classe $y \in Y$ (équation 2.1).

$$h : \left| \begin{array}{ccc} \mathbf{X} & \longrightarrow & \mathbf{Y} \\ x & \longmapsto & y = h(x) \end{array} \right. \tag{2.1}$$

Notez que pour le problème de *classification*, l'espace des réponses $Y \subset \mathbb{N}$ est discret et fini, vu que chaque $y \in Y$ représente une classe. Lorsque $Y$ est continu (c.à.d. $Y \subset \mathbb{R}$), on parle alors du problème de *régression* qui sert à estimer la relation entre une ou plusieurs variables $x^{(i)} \in \mathbb{R}$ et une autre variable $y \in \mathbb{R}$. Dans le cadre de cette thèse, nous nous intéressons au problème de classification plutôt qu'au problème de régression.

La probabilité conditionnelle $p(y|x)$ est appelée probabilité a posteriori de la classe $y$ pour l'instance $x$. Une discussion sur les propriétés de classifieurs spécifiques dépasserait le cadre de cette introduction. Nous mentionnons simplement que la plupart des classifieurs, non seulement retournent la classe $y$ prédite pour une instance $x$ telle que définie dans l'équation 2.1, mais donne également un score ou une estimation $\hat{p}(y|x)$ de la probabilité a posteriori.

Afin d'évaluer la qualité d'une classification pour un classifieur $h$, une fonction $l_h$ dite *fonction de coût* (en anglais "loss function" ou "cost function"), est généralement utilisée :

$$l_h : \left| \begin{array}{rcl} \mathbf{X} \times \mathbf{Y} & \longrightarrow & \mathbb{R}^+ \\ (x, y) & \longmapsto & l(h(x), y) = \begin{cases} 0 \text{ si } h(x) = y \\ 1 \text{ si } h(x) \neq y \end{cases} \end{array} \right. \tag{2.2}$$

où $x$ est une instance dont la classe est $y$, et $l(h(x), y)$ donne un coût nul lorsque $h(x) = y$ (c.à.d. lorsque la classe de $x$ est correctement prédite).

Ainsi, le taux d'erreur d'un classifieur $h$ peut être estimé en pratique sur un ensemble de $n$ données étiquetées suivant l'équation 2.3. Lorsque $n$ tend vers l'infini, ce taux d'erreur correspond alors à la probabilité de mauvaise prédiction $P(h(x) \neq y)$, et la probabilité complémentaire $P(h(x) = y)$ correspond au taux de bonne classification et est appelée "exactitude" ("accuracy" en anglais) ou taux de reconnaissance.

$$E_h = \frac{1}{n} \sum_{i=1}^{n} l_h(x_i, y_i) \tag{2.3}$$

La question principale qui se pose, est de savoir comment obtenir un bon classifieur $h$, autrement dit, comment apprendre $h$. Ce point est abordé ci-après.

## 2.2 Apprentissage

La notion d'apprentissage inclut toute méthode permettant de produire une fonction, des règles générales, ou plus généralement un modèle qui représente les informations présentes dans les données d'apprentissage. Un algorithme d'apprentissage essaie de modéliser la relation qui existe entre les entrées (instances) et les sorties (classes) en construisant la fonction $h : X \longrightarrow Y$ de sorte que la probabilité de mauvaise prédiction $P(h(x) \neq y)$ soit minimale.

A titre d'exemple, dans le scénario standard d'apprentissage supervisé, on dispose d'un ensemble $L = \{(x_1, y_1), ..., (x_n, y_n)\}$ d'instances étiquetées, appelé *ensemble de données d'apprentissage* ou tout simplement *base d'apprentissage*. Les données d'apprentissage sont utilisées avec un algorithme d'apprentissage pour produire un classifieur $h$, qui est ensuite utilisé pour prédire la classe d'une nouvelle instance. Le rôle des étiquettes de classes associées aux données d'apprentissage, est de guider ou de superviser un algorithme d'apprentissage vers la règle de classification désirée qui minimise l'erreur de classification des nouvelles instances.

### 2.2.1 Types d'apprentissage

Il existe plusieurs types d'apprentissage permettant de produire des classifieurs.

#### 2.2.1.1 Supervisé

L'apprentissage supervisé nécessite une base d'apprentissage où chaque instance est préalablement étiquetée avec sa classe respective. Supposons que nous disposions d'une grande quantité de données non étiquetées. Un opérateur étiquette manuellement autant de données que possible pour les utiliser dans un algorithme d'apprentissage supervisé, dans le but d'apprendre un classifieur. Notez ici que les données d'apprentissage à étiqueter manuellement sont choisies préalablement et au hasard parmi les données disponibles. On dit alors que l'apprentissage se fait de façon passive.

Il existe principalement deux familles de méthodes d'apprentissage supervisé pour la classification : les méthodes discriminatives et les méthodes génératives (voir Figure 2.1). Les méthodes discriminatives sont celles qui sont les plus utilisées et incluent, entre autres : SVM, KNN, régression logistique et réseaux de neurones. Elles permettent de définir une séparation entre les classes en créant des frontières de décision entre elles. La classification d'une nouvelle instance se fait selon la position de cette instance par rapport aux frontières de décision. Les méthodes génératives visent, quant à elle, à apprendre des modèles qui décrivent la façon dont les données de chaque classe sont générées en considérant les classes indépendamment les unes des autres. Des exemples de cette famille de méthodes incluent les HMM, la classification naïve bayésienne, etc.

De nombreuses méthodes d'apprentissage supervisé pour la classification, existent dans la littérature. Plus de détails sur le fonctionnement de ces méthodes sont donnés dans [Sebastiani 02, Kotsiantis 06, Bishop 06]. Plusieurs de ces méthodes ont été appliquées à la classification de documents images et sont revues dans [Chen 07].

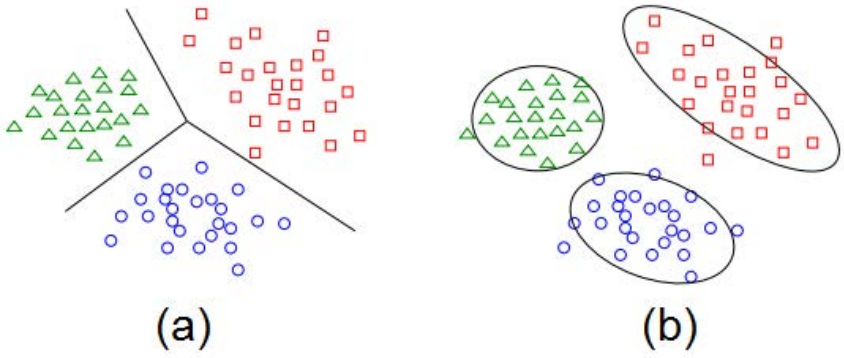



FIGURE 2.1 – (a) Approches discriminatives où des frontières sont construites pour la séparation, (b) Approches génératives ou un modèle est défini par classe.

#### 2.2.1.2 Non supervisé

L'apprentissage non supervisé ne nécessite pas l'étiquetage des données de la base d'apprentissage. Ce type d'apprentissage peut être utilisé pour découvrir des clusters formés par l'ensemble des données. Il s'agit de partitionner les instances en différents ensembles homogènes (clusters) tel que (i) les instances d'un même cluster partagent des caractéristiques communes, qui correspondent à des critères de proximité que l'on définit le plus souvent grâce à des mesures de distance entre les paires d'instances, et (ii) les instances appartenant à des clusters différents soient différenciées (éloignées). Pour exemple, la mesure de distance la plus connue est probablement la distance euclidienne définie par l'équation 2.4 pour deux instances $x_1$ et $x_2$.

$$dist(x_1, x_2) = \sqrt{\sum_{i=1}^{d} (x_1^{(i)} - x_2^{(i)})^2} \tag{2.4}$$

Cependant, les clusters découverts naturellement dans les données, ne correspondent pas nécessairement aux classes que nous voulons apprendre. Ceci provient du fait que les caractéristiques avec lesquelles les instances sont représentées dans l'espace $X$, ne sont jamais parfaitement corrélées aux classes qu'on désire obtenir. La Figure. 2.2 montre le cas d'un espace à $d = 2$ dimensions (les instances sont représentées par deux caractéristiques). Un apprentissage non supervisé n'est pas forcément adapté pour

la tâche de classification. Pour utiliser un apprentissage non supervisé pour une tâche de classification, les clusters doivent coïncider avec les frontières de décision des classes désirées. Si cette hypothèse est fausse (ce qui est souvent le cas), les résultats obtenus seront erronés.

De nombreuses méthodes pour la classification non supervisée de documents text, sont revues dans [Aggarwal 12].

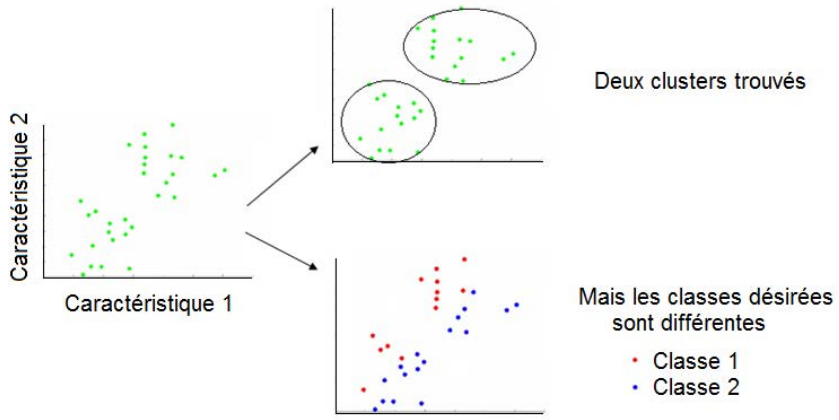



FIGURE 2.2 – Cas d'un apprentissage non supervisé : les clusters ne coïncident pas avec les classes désirées.

#### 2.2.1.3 Semi-supervisé

Dans l'apprentissage semi-supervisé, l'idée de base est de réduire le coût d'étiquetage en utilisant peu de données étiquetées et en exploitant les données non étiquetées. À titre d'exemple, une méthode d'apprentissage semi-supervisée très basique consiste à apprendre un modèle de classification à partir des données étiquetées et prédire les classes des données non étiquetées. Les données non étiquetées les plus sûres (pour lesquelles le classifieur est le moins incertain), avec leurs étiquettes prédites, sont alors ajoutées à la base d'apprentissage pour apprendre un nouveau modèle de classification, et ainsi de suite.

L'hypothèse faite par l'apprentissage semi-supervisé est que la distribution des données non étiquetées $p(x)$ partage des caractéristiques en commun avec la distribution des données dans leurs classes respectives $p(x|y)$. Ainsi, l'apprentissage semi-supervisé peut aider à améliorer les résultats d'apprentissage si cette hypothèse est vraie, mais peut conduire à une dégradation des performances du classifieur appris dans le cas contraire.

Les méthodes d'apprentissage semi-supervisé ont été appliquées à la classification de documents text dans [Krithara 08, Carmagnac 04, Nigam 06]. Une revue de littérature des principales méthodes d'apprentissage semi-supervisé est présentée dans [Zhu 08].

#### 2.2.1.4 Actif

Dans le cas d'un apprentissage supervisé, le but est d'apprendre un modèle qui lie les instances à leurs classes respectives, en utilisant un échantillon aléatoire de données d'apprentissage étiquetées. Dans ce cas, l'algorithme d'apprentissage apprend un modèle de classification en fonction des données

dont il dispose déjà, autrement dit, de données imposées. Afin d'enrichir la qualité des données utilisées pour l'apprentissage, il est possible d'autoriser des interactions entre l'algorithme d'apprentissage et l'opérateur qui effectue l'étiquetage des données. Il s'agit alors d'un apprentissage actif (à l'opposé d'un apprentissage passif).

L'apprentissage actif peut être considéré comme un protocole d'étiquetage agissant comme un apprentissage semi-supervisé afin de réduire le coût d'étiquetage, en permettant de choisir les données à étiqueter. Le choix des instances à étiqueter pour l'apprentissage, peut influencer considérablement la qualité du classifieur appris. En effet, les étiquettes de classes associées aux instances, dirigent le système vers la règle de classification désirée. Il est donc important d'étiqueter des instances *informatives* qui permettent de converger plus rapidement vers cette règle de classification. Par exemple, le gain d'étiqueter manuellement une instance qui est très bien classée par le classifieur (avec une grande confiance) est faible, comparé à d'autres instances plus informatives (pour lesquelles le classifieur est incertain). Contrairement à un apprentissage passif, l'apprentissage actif consiste ainsi à se demander quelles seraient les instances les plus informatives à présenter à l'opérateur, afin de les utiliser comme données étiquetées pour l'apprentissage.

Nous donnons un état de l'art des méthodes d'apprentissage actif existantes dans la section 3.3.

### 2.2.2 Apprentissage à partir de flux de données

Un flux de données peut être vu comme un ensemble possiblement de longueur infinie où les données arrivent les unes après les autres de façon continue. Leur ordre n'est pas maîtrisé, et l'accès à ces données est strictement séquentiel.

Généralement, les méthodes d'apprentissage, quelque soit leur type, ont besoin d'une base de données d'apprentissage préparée à l'avance et effectuent plusieurs itérations sur ces données. Ce mode d'apprentissage est dit "hors-ligne". Dans ce dernier, les phases d'apprentissage et de classification (reconnaissance) sont séparées. Le classifieur est d'abord entraîné statiquement sur une base d'apprentissage, puis il est utilisé pour classer de nouvelles données. Dans le cas d'un flux de données, il existe plusieurs méthodes d'apprentissage qui sont capables d'apprendre "en-ligne". Dans cette configuration, l'apprentissage et la classification sont effectuées conjointement, ainsi, le système apprend continuellement pour améliorer ses performances tout au long de son utilisation. L'apprentissage en-ligne est souvent effectué de façon incrémentale, même s'il est aussi possible de refaire l'apprentissage lorsque de nouvelles données sont disponibles (en stockant toutes les données). Un apprentissage incrémental est capable d'apprendre à partir de chaque nouvelle donnée en mettant à jour le modèle de classification et en ajustant ses paramètres, sans remettre en cause ce qui a déjà été appris et sans refaire un apprentissage sur toutes les données précédentes.

Les méthodes d'apprentissage à partir d'un flux de données sont souvent confrontées à de nouvelles contraintes. Par exemple, ces méthodes sont particulièrement sensibles à l'ordre d'arrivée des données vu que chaque donnée n'est visitée qu'une seule fois et est utilisée dès sa réception pour mettre à jour le modèle appris. De plus, vu que la base d'apprentissage n'est pas disponible à l'avance, l'ensemble des données d'apprentissage initial n'est souvent pas diversifié et conséquent. Ainsi, le système doit s'adapter pendant son utilisation à différentes situations, telles que l'apparition de nouvelles classes de données, des classes de données qui ne sont que partiellement représentées dans l'espace des entrées, l'adaptation automatique des paramètres d'apprentissage, etc.

## 2.3 Description des bases de données utilisées dans l'évaluation

Afin de diversifier nos données et montrer l'efficacité de nos algorithmes, nous avons utilisé plusieurs types de bases : bases de documents, bases de chiffres manuscrits, bases de lettres et bases d'images.

### Bases de documents

Nous considérons dans nos évaluations expérimentales différents ensembles de documents administratifs numérisés fournis par des clients de la société ITESOFT. Chaque document est traité par OCR [1] et représenté par un sac de mots textuels, un vecteur qui contient les nombres d'occurrences des mots dans le document. Les vecteurs de chaque ensemble sont représentés dans un espace à $d$ dimensions où $d$ est la taille du vocabulaire des mots utilisés. Les ensembles de documents utilisés contiennent des nombres différents de classes (de 13 à 141 classes) :

- DocSet : 779 documents, d = 413, 13 classes.
- CAF : 1158 documents, d = 271, 141 classes.
- LIRMM : 1951 documents, d = 277, 24 classes.
- MMA : 2591 documents, d = 292, 25 classes.
- APP : 19742 documents, d = 328, 139 classes.

Les documents considérés peuvent être mono-pages ou multi-pages et sont de natures hétérogènes. D'une part, les documents d'une même classe peuvent présenter une large variabilité. À titre d'exemple, la figure 2.3 montre deux documents différents de la base MMA, qui appartiennent à une même classe. D'autre part, les documents de deux classes différentes peuvent être similaires, difficiles à distinguer, et sont donc susceptibles d'être confondus. À titre d'exemple, la figure 2.4 montre deux documents de la base APP qui présentent beaucoup de similarités (visuelles et textuelles), mais qui appartiennent à deux classes différentes. De plus, certains documents (par exemple, le document de la figure 2.5) peuvent contenir une faible quantité d'informations.

### Bases de chiffres manuscrits

Deux bases de données représentant des chiffres manuscrits sont utilisées ; elles sont disponibles dans le répertoire UCI [Bache 13].

1) *Optdigits* est dédiée à la reconnaissance optique de chiffres manuscrits à partir de formulaires préimprimés. 43 personnes ont contribué à la création de ces données. Afin de réduire la dimensionnalité et donner de l'invariance aux petites déformations, les images de taille $32 \times 32$ sont divisées en blocs de $4 \times 4$ qui ne se chevauchent pas, et le nombre de pixels noirs est compté dans chaque bloc. Ceci donne lieu à une matrice de taille $8 \times 8$ (donc $d = 8 \times 8 = 64$) où chaque élément est un nombre entier entre $0$ et $16$. Des exemples d'images de la base *optdigits* sont donnés dans la figure 2.6a. Cette base contient plusieurs cas de chiffres ambigus et difficiles à distinguer. Par exemple, la figure 2.6b montre des chiffres manuscrits appartenant à la classe du chiffre 2, mais certains échantillons (entourés en rouge sur la figure) peuvent facilement être confondus avec la classe du chiffre $1$.

2) *Pendigits* est utilisée pour la reconnaissance de chiffres manuscrits en-ligne, écrits à l'aide de stylos numériques. $44$ personnes ont été invitées à écrire sur une tablette $250$ chiffres chacune, dans un ordre aléatoire, à l'intérieur de zones de $500 \times 500$ pixels. Les dix premiers chiffres sont ignorés parce que la plupart des sujets ne sont pas familiers avec ce type de périphériques d'entrée. Les chiffres sont représentés en utilisant leurs coordonnées dans la zone d'écriture, sous forme de vecteurs de caractéristiques de longueur fixe, en utilisant des techniques de ré-échantillonnage temporel et spatial. La Figure

1. reconnaissance optique de caractères (en anglais, Optical Character Recognition)

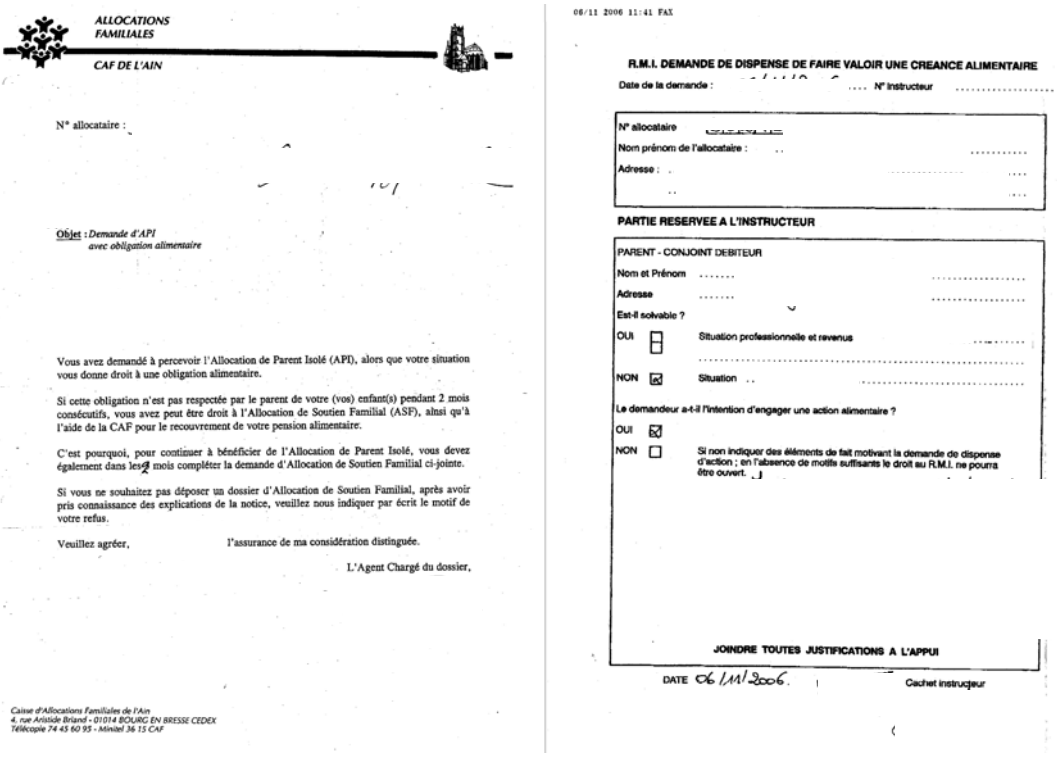



FIGURE 2.3 – Deux documents différents appartenant à une même classe.


FIGURE 2.4 – Deux documents similaires mais appartenant à deux classes différentes.

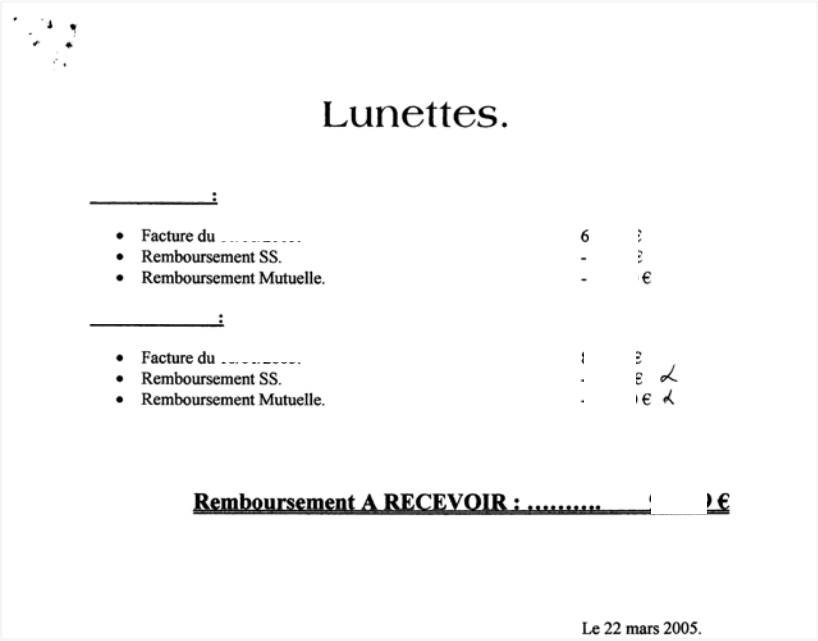



FIGURE 2.5 – Document contenant une faible quantité de text.


FIGURE 2.6 – Exemples d'images de chiffres manuscrits de la base optdigits.

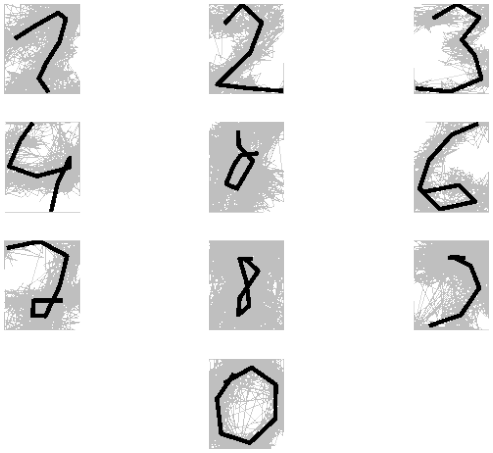


FIGURE 2.7 – Exemples d'images de chiffres manuscrits de la base pendigits. Tous les échantillons des chiffres manuscrits sont tracés en gris et superposés par classes. Les moyennes sont sont visualisées en noir.

2.7 montre des images de la base *pendigits*. Tous les échantillons des chiffres de la base sont tracés en gris et superposés, par classes. Les moyennes sont visualisées en noir.

En résumé, voici les informations sur ces deux bases :

– Optdigits : 5620 instances, d = 64, 10 classes.
– Pendigits : 10992 instances, d = 16, 10 classes.

### Base de lettres

La base *Letter-recognition* est disponible dans le répertoire UCI [Bache 13]. L'objectif est de reconnaître des lettres de l'alphabet anglais. Les images de caractères appartiennent à 20 polices différentes. Des exemples de lettres sont donnés dans la Figure 2.8.

– Letter-recognition : 20000 instances, d = 16, 26 classes.

### Base d'images

Dans la base *Image-segmentation* du répertoire UCI [Bache 13], les données ont été choisies au hasard à partir d'une base d'images de paysages. Les images ont été segmentées manuellement en régions de $3 \times 3$ pixels classées dans les 7 classes suivantes : brique, ciel, feuillage, ciment, fenêtre, chemin et herbe. Chaque région est représentée par 19 descripteurs morphologiques se basant sur les couleurs (R,G,B) et la densité des pixels. Ainsi, la base est présentée comme suit :

– Image-segmentation : 2310 instances, d = 19, 7 classes.

Les images originales de cette base ne sont pas disponibles publiquement.

## 2.4 Conclusion

Dans ce chapitre, nous avons introduit les principaux types d'apprentissage automatique utilisés pour la tâche de classification de données. La classification a été décrite comme la tâche consistant à assigner une donnée à une classe de données, en utilisant un modèle de classification (i.e. classifieur) qui lie les instances à leurs classes respectives. Afin d'apprendre ce modèle, les différents type d'apprentissage, leurs avantages et inconvénients, ont été présentés.

FIGURE 2.8 – Exemples d'images de lettres de la base Letter-recognition.

Dans le chapitre suivant, nous nous focaliserons sur l'apprentissage actif qui permet des interactions entre l'algorithme d'apprentissage et l'opérateur humain afin d'améliorer la qualité de l'apprentissage et réduire l'effort humain qui est nécessaire pour la supervision du système.

# Chapitre 3

# Apprentissage actif pour la classification de données

**Sommaire**



## 3.1 Introduction

Dans les méthodes d'apprentissage entièrement supervisées, un modèle de classification est appris en effectuant plusieurs passes sur un ensemble de données contenant un nombre suffisamment grand de données étiquetées. Ces approches ne sont pas applicables dans notre contexte pour deux raisons : (i) il n'est pas possible de réaliser plusieurs passes car les données arrivent continuellement à partir d'un flux de longueur infinie ; (ii) l'étiquetage manuel d'un grand nombre de données est un procédé très coûteux. Les techniques d'apprentissage semi-supervisées peuvent apprendre en utilisant des données étiquetées et non étiquetées, et peuvent donc être utilisées pour alléger le coût d'étiquetage. Dans le cas d'un apprentissage actif, c'est l'algorithme qui décide de demander les étiquettes de classes de certaines données "importantes" à un opérateur. Ce type d'apprentissage est adapté pour la classification de flux de données, où les données non étiquetées sont disponibles en grande quantité alors que leurs étiquettes sont coûteuses à obtenir.

L'apprentissage actif peut être mis en œuvre avec n'importe quel classifieur. Le principal problème à résoudre concerne la façon de sélectionner les instances pour lesquelles un étiquetage sera demandé. Pour ce faire, plusieurs stratégies ont été étudiées dans la littérature. Nous en verrons les principales dans la Section 3.3.

Notre contribution dans ce chapitre est faite selon deux actions. Tout d'abord, nous proposons dans la Section 3.4 une nouvelle mesure d'incertitude qui améliore les performances de l'apprentissage actif par rapport aux mesures d'incertitude de la littérature. La mesure proposée détermine le plus petit poids qu'il faut associer à une instance pour que la prédiction du classifieur change d'une étiquette à une autre. Si un petit poids est suffisant pour que la prédiction change, le classifieur est incertain, et la vraie étiquette de classe est demandée à un opérateur. Ensuite, nous proposons dans la sous-section 3.4.5 un seuil d'incertitude adaptatif qui convient pour un apprentissage actif à partir d'un flux de données, tout en faisant un compromis entre le nombre d'erreurs et le nombre d'étiquettes demandées.

## 3.2 Apprentissage actif vs passif

L'apprentissage actif peut être mieux expliqué si on l'oppose à l'apprentissage passif. Dans l'apprentissage supervisé usuel, qui représente un cas d'apprentissage passif, l'objectif est d'apprendre un classifieur à partir de données d'apprentissage étiquetées. Ces données sont présentées sous forme de paires d'entrées-sorties $(x, y)$ : la sortie (ou l'étiquette) $y$ représente la bonne réponse à une question associée à l'entrée (ou l'instance) $x$. Par exemple, dans le problème de la classification du courrier électronique, l'étiquette $y$ peut être la réponse "oui" ou "non" indiquant si un e-mail particulier $x$ est un spam ou non. Ces instances étiquetées sont collectées avant le processus d'apprentissage, et sont utilisées pour apprendre un classifieur qui sera déployé afin de prédire les étiquettes des instances à rencontrer dans l'avenir.

L'apprentissage actif modélise un processus légèrement différent dans lequel les données disponibles au départ ne sont pas associées à des étiquettes de classe. Autrement dit, chaque donnée d'apprentissage est tout simplement une entrée $x$ sans étiquette $y$ associée. L'objectif de l'apprenant actif est le même que celui d'un apprenant passif : apprendre un classifieur performant à partir des entrées. Cependant, l'apprenant actif est autorisé à demander l'étiquette $y$ de n'importe quelle entrée $x$. Dans l'exemple de classification du courrier électronique, cette fonction de l'apprenant actif peut être considérée comme demandant à l'utilisateur si un e-mail particulier dans la boîte aux lettres est un spam ou non.

Les motivations pratiques de l'apprentissage actif sont fondées sur le fait que les données non étiquetées sont aujourd'hui souvent disponibles en grande quantité. Mais, l'étiquetage nécessite une intervention manuelle pour évaluer ou analyser les données en entrée, et cela peut s'avérer être une tâche coûteuse. Par conséquent, dans de nombreuses applications de l'apprentissage automatique, seules les instances non étiquetées sont disponibles en grande quantité, tandis que les étiquettes sont coûteuses à obtenir. L'apprentissage actif relève le défi rencontré dans ces applications en modélisant explicitement le processus d'obtention des étiquettes pour les données non étiquetées. Le but est que l'apprenant actif demande juste les étiquettes de quelques instances soigneusement choisies, afin de produire un classifieur performant.

## 3.3 État de l'art

Il existe dans la littérature principalement deux approches pour sélectionner les données à étiqueter par un opérateur. La première décrite dans [Angluin 88], consiste à apprendre en formulant des requêtes avec des données synthétisées (générés de façon automatique). Cette méthode d'apprentissage peut demander l'étiquette de n'importe quelle instance de l'espace d'entrée $x \in X$, y compris des instances

que la méthode génère elle même. Cependant, comme indiqué dans [Angluin 04], cette approche n'est efficace que pour des problèmes à domaine $X$ fini. De plus, de telles instances peuvent être difficiles à étiqueter pour un opérateur humain. Par exemple, dans [Lang 92], les auteurs ont utilisé cette approche pour entraîner un classifieur à reconnaître des caractères manuscrits, mais ils ont rencontré un problème : la plupart des images générées contenaient des caractères qui n'avaient pas de signification sémantique naturelle et ne contenaient donc pas de symboles facilement reconnaissables par un humain. Pour remédier à ces problèmes, une deuxième approche plus naturelle a été introduite. Au lieu de demander l'étiquetage d'une instance qu'on génère dans une région arbitraire de l'espace des caractéristiques, on sélectionne une instance existante issue de la distribution réelle des données, selon un critère d'importance. Nous nous intéressons dans ce chapitre exclusivement aux méthodes issues de cette dernière approche. Ce choix est essentiellement dû : (i) aux domaines d'applications propres à ITESOFT, dans lesquelles les données non étiquetées arrivent abondamment à partir d'un flux, alors que l'étiquetage de ces données est coûteux ; et (ii) au fait que l'on veut étiqueter des données réelles, ce que l'opérateur humain sait faire sans ambiguïté.

### 3.3.1 Apprentissage actif statique et séquentiel

Soit $X$ l'espace d'entrée (des instances) et $Y$ l'espace de sortie. Le but d'un apprentissage actif est d'apprendre un modèle de classification efficace $h : X \rightarrow Y$ en utilisant un nombre minimal d'étiquettes demandées. En vérité, il existe principalement deux cas d'apprentissage actif :

**Cas statique.** Pour de nombreux problèmes d'apprentissage réels, une grande collection de données non étiquetées peut être réunie en une seule fois. Ainsi, on dispose d'un ensemble statique de données non étiquetées $U \subset X$ et d'un petit ensemble initial de données étiquetées $L$. La méthode d'apprentissage doit sélectionner de manière itérative une instance $x \in U$ afin de demander son étiquette de classe $y \in Y$ à un opérateur. La sélection de telle ou telle instance pour étiquetage peut être faite selon plusieurs stratégies de demandes. En général, les instances sont triées selon une mesure d'"informativité" utilisée pour évaluer toutes les instances de l'ensemble $U$. L'instance la plus informative est alors présentée à un opérateur pour l'étiquetage.

**Cas séquentiel.** Dans le cas d'un apprentissage actif séquentiel, à chaque instant $t$, on reçoit une nouvelle instance non étiquetée $x \in X$ à partir d'un flux de données de longueur infinie. La méthode d'apprentissage doit décider, au temps $t$, si oui ou non l'étiquette correspondante $y \in Y$ doit être demandée. Si l'étiquette $y$ de $x$ est demandée, alors le modèle de classification est entraîné en utilisant l'instance étiquetée $(x, y)$. Sinon, l'étiquette prédite $y = h(x)$ est retournée. Cette approche est dite séquentielle, car contrairement au cas statique, les instances non étiquetées arrivent une à une à partir d'une source de données (sous forme d'un flux), et la méthode d'apprentissage doit décider de demander son étiquette ou de l'ignorer. Pour une instance donnée $x$, la décision de demander ou non son étiquette peut être exprimée de plusieurs façons. On peut, par exemple, définir explicitement une région d'incertitude [Cohn 94], qui représente la région de l'espace des entrées qui est encore ambigüe au modèle ; l'ambiguïté est quantifiée à l'aide d'une mesure d'informativité et d'un seuil défini sur cette mesure. Les étiquettes des instances ayant une informativité au dessus de ce seuil, sont demandées.

La principale différence entre l'apprentissage actif statique et l'apprentissage actif séquentiel est que le premier évalue et trie tout l'ensemble de données avant de choisir la meilleure instance (la plus informative), tandis que le second parcourt les données séquentiellement et prend des décisions de demandes individuelles. Nous exposons dans la Section 3.3.2 les différentes stratégies de demandes utilisées dans la littérature.

### 3.3.2 Stratégies pour mesurer l'informativité des instances

Il existe différentes stratégies pour mesurer l'informativité des instances et donc pour demander ou non leurs étiquettes de classes à un opérateur. Ces stratégies sont basées sur divers critères que nous présentons dans cette section.

Comme expliqué précédemment, un apprentissage actif statique consiste à choisir pour étiquetage l'instance (parmi toutes les instances) qui maximise une mesure d'informativité, tandis que celui séquentiel consiste à demander l'étiquetage d'une instance dont l'informativité est supérieure à un seuil. Les stratégies présentées dans cette section sont utilisables telles quelles dans le cas d'un apprentissage actif statique mais nécessitent d'introduire un seuil dans le cas séquentiel.

#### 3.3.2.1 Stratégies basées sur l'incertitude

Les stratégies les plus simples et les plus courantes sont celles basées sur l'incertitude. Avec ces stratégies, l'apprentissage actif demande l'étiquetage des instances pour lesquelles il est le moins certain (ou le plus incertain) de leur classe. Il existe principalement 3 types de mesures d'incertitude :

**Faible probabilité de prédiction.** Les instances à étiqueter sont sélectionnées en fonction de la probabilité (ou score) de prédiction retournée par le classifieur [Lewis 94]. L'instance la plus informative $\hat{x}$ parmi un ensemble de données non étiquetées est donc celle qui est la plus incertaine selon l'équation suivante :

$$\hat{x} = \underset{x}{\operatorname{argmin}} P_h(y_1|x) \tag{3.1}$$

où $P_h(y_1|x)$ indique la probabilité que $x$ appartient à la classe $y_1$, et $y_1$ est l'étiquette de classe la plus probable pour $x$ sous le modèle $h$ :

$$y_1 = \underset{y \in Y}{\operatorname{argmax}} P_h(y|x)$$

Toutefois, ce critère ne prend en compte que la classe la plus probable et ignore les probabilités de prédiction des autres étiquettes de classes.

**Marge entre les deux classes les plus probables.** Une autre mesure d'incertitude appelée "marge" [Scheffer 01] est aussi utilisée dans le cadre de l'apprentissage actif. Cette mesure vise à corriger une lacune de la stratégie précédente, en intégrant la probabilité de la deuxième classe la plus probable. Essentiellement, la différence (marge) entre les probabilités des deux classes les plus probables est calculée :

$$\hat{x} = \underset{x}{\operatorname{argmin}} P_h(y_1|x) - P_h(y_2|x) \tag{3.2}$$

où $y_1$ et $y_2$ sont respectivement la $1^{\text{ère}}$ et la $2^{\text{ème}}$ classe les plus probables pour $x$ sous le modèle $h$ :

$$y_1 = \underset{y}{\operatorname{argmax}} P_h(y|x) \qquad y_2 = \underset{y \neq y_1}{\operatorname{argmax}} P_h(y|x)$$

Intuitivement, les classes des instances avec de grandes marges sont faciles à prédire correctement, vu que dans ce cas, le classifieur a peu de doute dans la différenciation entre les deux classes les plus probables. Une instance avec une petite marge est plus ambigüe, vu que le classifieur n'arrive pas à bien différencier entre les deux classes les plus probables. Dans ce cas, connaître la vraie étiquette de classe aiderait le modèle à discriminer plus efficacement entre ces deux classes.

**Entropie.** Principalement utilisée en théorie de l'information, l'entropie [Shannon 48] est souvent considérée comme une mesure d'incertitude ou d'impureté dans le domaine d'apprentissage automatique. Contrairement aux deux mesures précédentes, l'entropie prend en considération la distribution de probabilités pour toutes les classes :

$$\hat{x} = \underset{x}{\operatorname{argmin}} \sum_{i=1}^{|Y|} P_h(y_i|x) \log P_h(y_i|x) \tag{3.3}$$

où $y_i$ prend toutes les étiquettes de classes possibles.

Un des avantages des stratégies basées sur l'incertitude est qu'elles sont faciles à mettre en œuvre et peuvent être utilisées avec n'importe quel classifieur, qu'il soit probabiliste ou non. En effet, des stratégies d'incertitude similaires ont été considérées avec des classifieurs non probabilistes en considérant le score de classification retourné par le classifieur, par exemple, pour les arbres de décision dans [Lewis 94 (2)], les plus proches voisins (KNN) dans [Lindenbaum 04], ou encore les SVM dans [Tong 00].

Les stratégies d'apprentissage actif basées sur l'incertitude qui sont présentées ici, permettent toutes de réduire le coût d'étiquetage par rapport au cas passif. Par ailleurs, la mesure de "marge" semble appropriée si l'objectif est de réduire les erreurs de classification, car elle privilégie les instances qui aideraient le modèle à mieux discriminer entre des classes spécifiques. Cependant, des comparaisons empiriques de ces mesures d'incertitudes dans [Korner 06, Schein 07, Settles 08] ont donné des résultats différents, ce qui laisse entendre que la meilleure stratégie peut dépendre de l'application.

Dans le cas d'un apprentissage actif à partir d'un flux de données, si une instance incertaine $x$ est étiquetée manuellement et correctement, deux objectifs sont atteints : d'une part, le classifieur évite de retourner une probable erreur de classification (vu qu'il était incertain de la classe de $x$), et d'autre part, connaître la vraie classe de $x$ est utile pour améliorer $h$ et réduire son incertitude globale ($x$ est dite informative).

#### 3.3.2.2 Stratégies basées sur un ensemble de modèles

D'autres types de stratégies pour sélectionner des instances informatives sont basées sur l'utilisation d'un ensemble de modèles [Seung 92]. L'approche consiste à maintenir un ensemble de $C$ modèles du même type (issus d'un même classifieur) qui sont tous entraînés sur un ensemble d'instances étiquetées $L$, mais représentent des modèles concurrents. Chaque modèle de l'ensemble est alors autorisé à voter sur l'étiquetage d'instances candidates. L'instance la plus informative est considérée comme l'instance pour laquelle les modèles sont le plus en désaccord. L'idée fondamentale de ces stratégies est de minimiser l'*espace de versions*, qui est l'ensemble des modèles qui sont cohérents avec les données étiquetées $L$. L'apprentissage actif peut alors être vu comme la recherche du "meilleur" modèle dans l'espace de versions. Dans ce cas, le but de l'apprentissage actif est de réduire la taille de cet espace autant que possible, de sorte que la recherche devienne plus précise, en utilisant le moins possible d'instances étiquetées.

Afin de mettre en œuvre un algorithme basé sur ce type de stratégie, il faut être en mesure de construire un ensemble de modèles qui représentent différentes régions de l'espace de versions. Pour les types de modèles génératifs tels que l'analyse discriminante linéaire (LDA) ou la classification naïve bayésienne, cela peut être fait par la génération aléatoire d'un nombre arbitraire de modèles à partir d'une distribution a posteriori $P(h|L)$, comme par exemple dans [McCallum 98] pour la classification naïve bayésienne. Pour les types de modèles discriminants ou non probabilistes, l'auteur dans [Abe 98] construit l'ensemble des modèles en employant des méthodes d'apprentissage d'ensemble bien connues

(boosting et bagging) afin de construire l'ensemble des modèles. D'autre méthodes telles que celles mentionnées dans [Muslea 00, Melville 04] construisent un ensemble de modèles en partitionnant l'espace des caractéristiques afin d'encourager explicitement la diversité parmi les modèles de l'ensemble. Il n'y a pas d'accord général dans la littérature sur la taille appropriée de l'ensemble à utiliser, qui peut en effet varier selon les types de modèle ou d'application.

Afin de mesurer le niveau de désaccord entre les modèles de l'ensemble pour une instance donnée $x$, une approche courante utilisée dans [Dagan 95], est le vote par entropie qui peut être vue comme une généralisation de l'incertitude basée sur l'entropie :

$$\hat{x} = \underset{x}{\operatorname{argmin}} \sum_{i} \frac{V(y_i)}{C} \log \frac{V(y_i)}{C} \tag{3.4}$$

où $C$ est la taille de l'ensemble, $y_i$ parcourt toutes les étiquettes de classes possibles, et $V(y_i)$ est le nombre de votes pour l'étiquetage de $x$ avec $y_i$.

D'autre approches issues de la théorie de l'information telles que la divergence de Kullback-Leibler ou encore la divergence de Jensen–Shannon ont également été utilisées pour mesurer le niveau de désaccord dans [McCallum 98] et [Melville 05] respectivement.

Cependant, les méthodes utilisées pour construire un ensemble de modèles ne sont que des approximations de l'espace de versions. En effet, l'auteur dans [Haussler 94] montre que la taille de l'espace de versions peut croître de façon exponentielle avec la taille de $L$. Cela signifie qu'en général, l'espace de versions d'un type de modèle arbitraire ne peut pas être représenté de manière explicite dans la pratique. Aussi, contrairement aux stratégies basées sur l'incertitude, cette stratégie n'est pas indépendante du type de modèle utilisé.

#### 3.3.2.3 Minimisation de l'espérance d'erreur

Une autre stratégie, basée sur la théorie de la décision, vise à mesurer la réduction de l'espérance d'erreur future (aussi appelée risque). L'objectif ici est de réduire le nombre total espéré de prédictions incorrectes. Cette stratégie a été proposée dans [Roy 01].

Pour une instance non étiquetée $x \in U$ qui est candidate pour l'étiquetage, et pour chaque étiquette possible $y_i$, l'idée est d'estimer l'espérance d'erreur d'un modèle entraîné en utilisant $L \cup \{(x, y_i)\}$ sur tout l'ensemble des instances non étiquetées $U$, qui est supposée être disponible à l'avance et utilisée comme une sorte de jeu de validation. L'espérance est calculée en moyennant sur toutes les étiquettes de classes $y_i$ possibles. L'instance candidate avec l'espérance d'erreur minimale est choisie pour l'étiquetage. Plus formellement :

$$\hat{x} = \underset{x}{\operatorname{argmin}} \sum_{u \in U} E_y[1 - P_{h+(x,y)}(\hat{y}|u)] \tag{3.5}$$

où $E_y$ est l'espérance sur tous les étiquetages possibles de $x$, et $(1 - P_{h+(x,y)}(\hat{y}|u))$ est l'incertitude du modèle sur l'instance $u$ après avoir été entraîné en utilisant $x$.

Notez que nous ne connaissons pas la véritable étiquette de chaque instance $x$, donc nous approximons en utilisant l'espérance sur toutes les étiquettes possibles sous le modèle actuel $h$. Ainsi, l'instance $x$ choisie, peut être vue comme celle qui donne lieu a un modèle plus "certain" sur ses prédictions, s'il est entraîné avec cette dernière.

D'autres variantes de cette stratégie ont aussi été utilisées dans la littérature. Par exemple, dans [Guo 01], la même stratégie est utilisée en calculant l'espérance avec différentes mesures d'incertitude (voir Section 3.3.2.1). Dans [Zhu 03], cette stratégie est combinée avec une approche d'apprentissage semi-supervisé, conduisant à une amélioration des résultats de classification.

L'avantage de cette stratégie est qu'elle est quasi-optimale et ne dépend pas du type du modèle utilisé. Elle nécessite bien sûr un classifieur qui permet d'estimer les probabilités de prédiction et surtout, que l'ensemble des instances non étiquetées $U$ soient disponibles à l'avance. Elle ne peut donc pas être adaptée à un flux de données. De plus, cette stratégie est la plus coûteuse en calcul, car non seulement une estimation de l'espérance d'erreur est nécessaire sur tout l'ensemble $U$ pour chaque requête $x$, mais un nouveau modèle doit aussi être re-entraîné pour chaque estimation.

#### 3.3.2.4 Autres stratégies

**Exploitation de la structure des clusters.** D'autres stratégies qui sont différentes des précédentes, consistent à partitionner l'ensemble des données non étiquetées $U$ (qui est supposé être disponible à l'avance) en clusters et à exploiter la structure de ces clusters pour un apprentissage actif. L'idée est de demander l'étiquetage de quelques instances de chaque cluster, d'attribuer à chaque cluster l'étiquette majoritaire, puis d'utiliser l'ensemble de ces données étiquetées pour l'apprentissage (voir Figure. 3.1). Des exemples de cette stratégie sont proposés dans [Nguyen 04, Dasgupta 08, Dasgupta 11]. Néanmoins, en général, la structure des clusters n'est pas aussi clairement définie, et on peut obtenir des résultats médiocres lorsque les clusters ne coïncident pas du tout avec les limites de décision des vraies classes.

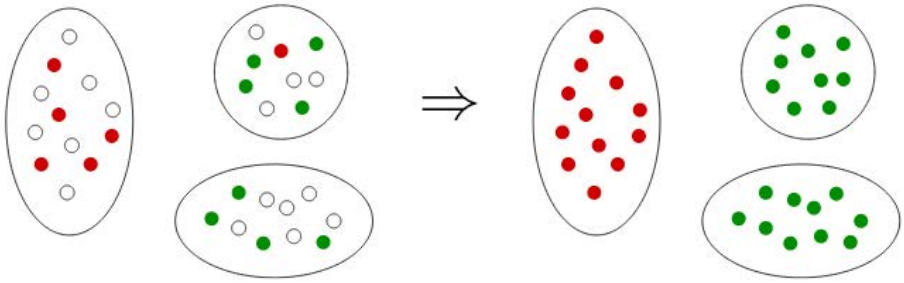


FIGURE 3.1 – Exploitation de la structure des clusters

**Maximisation de l'espérance de changement du modèle.** Une autre stratégie moins courante consiste à sélectionner les instances qui auraient une grande influence sur le modèle courant si elles sont étiquetées. Cette influence est estimée par le degré de changement du modèle avant et après l'introduction de l'instance qui est candidate pour l'étiquetage. Par exemple, l'auteur dans [Settles 08 (2)], utilise cette stratégie dans le cadre d'un apprentissage multi-instances [Sorower 10] avec les méthodes basées sur l'algorithme du gradient. Ainsi, on demande la classe d'une instance $x$ si le changement du gradient qui aurait lieu, si cette instance était apprise, est grand. Cependant, cette stratégie reste aussi coûteuse que celle présentée dans le paragraphe 3.3.2.3, et dépend fortement du type de modèle utilisé.

## 3.4 Stratégie d'incertitude basée sur la pondération des instances

Dans cette section, nous proposons une nouvelle stratégie d'apprentissage actif. La méthode proposée mesure l'incertitude du classifieur en se basant sur la notion de pondération des instances. Elle détermine le plus petit poids qu'il faut associer à une instance, pour que la prédiction du classifieur bascule d'une étiquette à une autre. Si un petit poids est suffisant pour que la prédiction change, le classifieur est incertain, et la vraie étiquette de classe est demandée à un opérateur.

### 3.4.1 Pondération des instances

La pondération des instances est une pratique courante dans le domaine de l'apprentissage automatique pour la classification [Yang 07, Jiang 07, Dudani 76, Frank 02, Xia 13]. Un poids associé à une instance permet de quantifier l'importance relative de cette instance par rapport aux autres. Toute méthode d'apprentissage usuelle peut être modifiée de sorte à donner plus d'importance aux instances auxquelles on associe un poids élevé. De cette façon, une instance avec un petit poids n'affectera pas beaucoup le modèle de classification $h$, contrairement à une instance avec un poids plus élevé. La pondération des instances est généralement utilisée en classification pour les problèmes d'adaptation de domaine [Jiang 07] ou des problèmes de classification dans le cas de données non équilibrées (en donnant plus de poids aux instances des classes minoritaires). Dans ce chapitre, nous utilisons la notion de pondération des instances pour proposer une nouvelle stratégie d'apprentissage actif performante.

D'une façon générale, la pondération des instances est prise en compte en multipliant la fonction que le classifieur minimise, par le poids associé à l'instance. Par exemple certains classifieurs tels que SVM ou la régression logistique, essaient de minimiser une fonction de coût ("loss function" en anglais) lors de l'apprentissage. Dans ce cas, chaque instance peut être pondérée en multipliant tout simplement ce coût par un poids associé à cette instance, tel que décrit dans [Jiang 08]. Dans le cas de la méthode des plus proches voisins (KNN), la classe prédite pour une instance $x$ est celle qui est majoritaire dans son voisinage. Ainsi, chaque instance parmi les $K$ plus proches de $x$, vote sur une étiquette pour classer $x$. Si les instances sont pondérées, alors leurs contribution au vote est pondérée en conséquence.

Dans ce qui suit, nous considérons un modèle de classification noté $h$, indépendamment du type de classifieur utilisé. D'une façon générale, entraîner le modèle $h$ en utilisant l'instance $(x, y, w)$ (c.à.d. $x$ étiqueté avec $y$ et pondérée par $w$), peut aussi être mise en œuvre en entraînant $h$ sur $(x, y)$, un nombre de fois proportionnel à $w$. Ainsi, associer un poids à une instance lors de l'apprentissage revient simplement à dupliquer cette instance, c.à.d., apprendre plusieurs fois sur cette instance, selon le poids.

### 3.4.2 Notion de poids suffisant

Supposons que nous disposons de deux classes $y_1$ et $y_2$, et que le modèle $h$ prédise la classe $y_1$ pour l'instance $x$, c.à.d., $h(x) = y_1$. Supposons que nous forçons $h$ à apprendre $x$, mais que $x$ est étiquetée avec l'étiquette de classe opposée $y_2$ (i.e. différente de celle qui a été prédite : $y_1$) et pondérée par un poids $w \in ]0, 1]$. Notons $\bar{h}_w$ le modèle obtenu. Si un petit poids $w$ est suffisant pour que $\bar{h}_w$ prédise $y_2$ (c.à.d. pour que $\bar{h}_w(x) = y_2$), alors $h$ était incertain dans sa première prédiction $y_1$. En d'autres termes, le modèle $h$ est incertain dans sa prédiction sur $x$ si un petit poids $w$ est suffisant pour que la prédiction change d'étiquette. De même, $h$ est assez confiant dans sa prédiction si un poids élevé $w$ est nécessaire pour modifier sa prédiction concernant $x$. L'idée de base de la mesure d'incertitude proposée est de rechercher le poids suffisant $\hat{w}$ (le plus petit poids) pour que le classifieur $h$ change son avis sur la classe prédite pour une donnée $x$ en entrée.

Plus formellement, étant donné un modèle de classification $h$ et une nouvelle instance $x$ tel que $h(x) = y_1$, si $\bar{h}_w$ fait référence au modèle $h$ après avoir été entraîné avec $(x, y_2, w)$, alors le "poids suffisant" $\hat{w}$ est défini selon l'équation 3.6 comme étant le poids minimal pour lequel la prédiction $\bar{h}_w(x)$ donne $y_2$ au lieu de $y_1$.

$$\hat{w} = \min \{w \in ]0, 1] \mid \bar{h}_w(x) \neq h(x)\} \tag{3.6}$$

Plus la valeur de $\hat{w}$ est petite, plus l'instance $x$ est incertaine. En effet, une petite valeur de $\hat{w}$ signifie que le modèle de classification $h$ n'a pas été beaucoup affecté, mais malgré cela, sa prédiction a basculé de $y_1$ à $y_2$. Ainsi, $h$ est incertain dans sa prédiction.

Dans un cadre multi-classes, $y_1$ et $y_2$ peuvent être définies respectivement comme la première et la deuxième classe les plus probables pour $x$.

$$y_1 = h(x) = \underset{y \in Y}{\operatorname{argmax}} P_h(y|x) \qquad y_2 = \underset{y \neq y_1}{\operatorname{argmax}} P_h(y|x)$$

### 3.4.3 Mesure de l'incertitude par l'approximation du poids suffisant

Pratiquement, au lieu de rechercher la valeur exacte du "poids suffisant" $\hat{w}$, il est possible d'approximer cette valeur à l'aide d'une recherche dichotomique sur l'intervalle $]0, 1]$. En effet, puisque $\hat{w}$ est le poids minimal pour que $\bar{h}_{\hat{w}}$ prédise une étiquette différente de $y_1$ (voir l'équation 3.6), alors pour tout poids $w$ inférieur à $\hat{w}$, $\bar{h}_w(x)$ donne toujours $y_1$, et pour tout poids $w$ supérieur ou égal à $\hat{w}$, $\bar{h}_w(x)$ change :

$$\forall w < \hat{w}, \bar{h}_w(x) = y_1 \tag{3.7}$$

$$\forall w \geq \hat{w}, \bar{h}_w(x) \neq y_1 \tag{3.8}$$

À partir des équations 3.7 et 3.8, nous savons que si $w$ fait que le modèle change sa prédiction sur $x$, alors le poids suffisant $\hat{w}$ est inférieur ou égal à $w$, sinon, il est supérieur à $w$. Par conséquent, la valeur de $\hat{w}$ peut être estimée à l'aide d'une recherche dichotomique qui est illustrée par l'algorithme 1 et dont la complexité est de $\mathcal{O}(\log_2 \frac{1}{\mathcal{E}})$, où $\mathcal{E}$ est la taille de l'intervalle de tolérance pour approximer $\hat{w}$, tel que $0 < \mathcal{E} \ll 1$. Dans la suite, nous ferons référence à $\hat{w}$ comme étant la valeur retournée par l'algorithme 1.

**Algorithm 1** Approximation du poids suffisant

**Entrées :** $x$, $h$ (copie du modèle), $\mathcal{E}$
Soit $y_1 = \underset{y}{\operatorname{argmax}} P_h(y|x)$, $y_2 = \underset{y \neq y_1}{\operatorname{argmax}} P_h(y|x)$
Soit $\text{low} = 0$, $\text{up} = 1$
**répéter**
    $w = \frac{\text{up} + \text{low}}{2}$
    Définir $\bar{h}_w$ en entraînant $h$ avec $(x, y_2, w)$
    **si** $\bar{h}_w(x) = y_1$ **alors**
        $\text{low} = w$
    **sinon**
        $\text{up} = w$
    **fin si**
**jusqu'à** $\text{up} - \text{low} \leq \mathcal{E}$
**retourner** $w$

### 3.4.4 Apprentissage actif statique et séquentiel avec le poids suffisant

Dans le cas d'un apprentissage actif statique, le poids suffisant $\hat{w}$ doit être calculé en utilisant l'algorithme 1 pour chaque instance non étiquetée $x \in U$. L'instance ayant le plus petit poids suffisant $\hat{w}$ (c.à.d. l'instance la plus incertaine) est sélectionnée pour être étiquetée (voir Algorithm 2).

Dans le cas d'un apprentissage actif à partir d'un flux de données (séquentiel), nous demandons l'étiquette d'une nouvelle instance $x$ si son poids suffisant est inférieur à un seuil $\theta$, autrement dit, l'étiquette est demandée si $\hat{w} < \theta$. Cela nécessite de calculer $\hat{w}$ pour chaque nouvelle instance $x$ et de le comparer

**Algorithm 2** Apprentissage actif statique

**Entrées :** modèle de classification $h$, ensemble de données non étiquetées $U$
**tant que** critère d'arrêt non vérifié **faire** // e.g. atteindre le nombre souhaité d'étiquettes demandées
    $min\_\hat{w} = 1$ // initialisée à la plus grande valeur du poids
    **pour chaque** $x \in U$ **faire**
        Calculer le poids suffisant $\hat{w}$ pour l'instance $x$ via l'algorithme 1
        **si** $\hat{w} < min\_\hat{w}$ **alors**
            $min\_\hat{w} = \hat{w}$
            $\hat{x} = x$
        **fin si**
    **fin pour**
    demander $y$ la classe de $\hat{x}$ à un opérateur
    entraîner $h$ en utilisant $(\hat{x}, y)$
    $U = U - \{\hat{x}\}$
**fin tant que**

à $\theta$. En fait, il est possible d'éviter de calculer $\hat{w}$ pour toutes les instances. L'équation 3.8 implique que, pour un poids donné $w$, si $\hat{w} < w$ alors $\bar{h}_w(x) \neq y_1$. Il s'ensuit que pour un seuil donné $\theta$, si $\hat{w} < \theta$, alors $\bar{h}_\theta(x) \neq y_1$. Par conséquent, demander l'étiquette de $x$ quand $\hat{w} < \theta$ est équivalent à demander l'étiquette de $x$ quand $\bar{h}_\theta(x) \neq y_1$ (voir Figure 3.2). En d'autres termes, il suffit de demander l'étiquette de $x$ si le seuil $\theta$, utilisé comme un poids, fait que le modèle change sa prédiction concernant $x$.

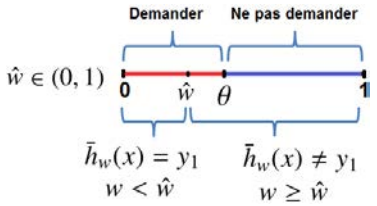



FIGURE 3.2 – Seuil $\theta$ pour demander l'étiquetage des instances

L'algorithme 3 montre l'apprentissage actif à partir d'un flux de données avec la stratégie proposée, en utilisant un seuil fixe $\theta$. Ce seuil peut être adapté automatiquement comme présenté dans la section suivante.

### 3.4.5 Seuil adaptatif pour l'apprentissage actif à partir d'un flux de données

Comme nous l'avons mentionné précédemment, dans le cas d'un flux de données, les méthodes habituelles d'apprentissage actif qui sont basées sur l'incertitude, définissent une région d'incertitude en utilisant un seuil de confiance fixe $\theta$. L'étiquette d'une instance est demandée à un opérateur si la confiance est inférieure à ce seuil. Il s'agit d'une instance incertaine. Cependant, il est souvent difficile de fixer manuellement le seuil $\theta$ car il dépend fortement des données en entrée et de la mesure de confiance utilisée. Lorsque $\theta$ est trop élevé, beaucoup d'instances sont étiquetées, bien que la plupart d'entre elles ne soient pas nécessairement informatives. De même, quand $\theta$ est trop petit, peu d'étiquettes sont demandées, ce qui peut donner lieu à un classifieur sous entraîné. Nous proposons ici un seuil adaptatif qui peut être utilisé avec n'importe quelle stratégie d'incertitude. Cependant, nous considérons comme exemple, la stratégie proposée basée sur le "poids suffisant" que nous avons défini précédemment.

Soient $x$ une nouvelle instance du flux, $\hat{w}$ le poids suffisant correspondant à cette instance, obtenu en

**Algorithm 3** Apprentissage actif à partir d'un flux

**Entrées :** modèle de classification sous-jacent $h$, seuil de décision $\theta$
**pour chaque** nouvelle instance $x$ à partir du flux **faire**
    Soit $y_1 = \underset{y}{\operatorname{argmax}}\, P_h(y|x)$, $y_2 = \underset{y \neq y_1}{\operatorname{argmax}}\, P_h(y|x)$
    Définir $\bar{h}_\theta$ en entraînant $h$ avec $(x, y_2, \theta)$
    **si** $\bar{h}_\theta(x) \neq y_1$ **alors** // (équivalent à $\hat{w} < \theta$)
        demander $y$ la classe de $x$ à un opérateur
        entraîner $h$ en utilisant $(x, y)$
    **sinon**
        sortir $y_1$ la classe prédite pour $x$
    **fin si**
**fin pour**

utilisant l'algorithme 1, $y_1 = h(x)$ l'étiquette de la classe prédite pour $x$, et $y$ l'étiquette de la vraie classe de $x$ qui est demandée à un opérateur si $\hat{w} < \theta$. Soit $\epsilon$ une constante représentant un taux d'apprentissage tel que $0 < \epsilon \ll 1$. Un bon seuil $\theta$ doit prendre en considération deux objectifs : (i) minimiser les demandes d'étiquetage non nécessaires, (ii) minimiser les erreurs de prédiction.

(i) *Minimiser les demandes d'étiquetage non nécessaires*. Supposons que l'étiquette $y$ de $x$ a été demandée parce que $x$ est considérée comme incertaine, étant donné la valeur actuelle de $\theta$ (c.à.d. $\hat{w} < \theta$), mais que l'étiquette prédite $y_1$ était correcte (c.à.d. $y_1 = y$). Dans ce cas, nous devrions demander moins d'étiquettes, c.à.d. être plus confiant. Cela se fait en *décrémentant* la valeur de $\theta$ proportionnellement à une quantité $\Delta_{dec}$

$$\theta_{new} = \theta - \epsilon \times \Delta_{dec}$$

(ii) *Minimiser les erreurs de prédiction*. Si $y_1 \neq y$, alors une erreur de prédiction a eu lieu et le classifieur doit être entraîné sur plus de données étiquetées. Dans ce cas, nous devrions être moins confiants et demander plus d'étiquettes. Cela se fait en *incrémentant* la valeur de $\theta$ proportionnellement à une quantité $\Delta_{inc}$

$$\theta_{new} = \theta + \epsilon \times \Delta_{inc}$$

Afin de définir les quantités $\Delta_{dec}$ et $\Delta_{inc}$, nous considérons chaque cas séparément.

**Minimiser les demandes d'étiquetage non nécessaires** Pour le premier cas, la décision de demander la vraie classe $y$ a été prise parce que $\hat{w} < \theta$, bien que le classifieur ait prédit correctement l'étiquette de classe de $x$. Cela correspond à une demande non nécessaire. Cependant, si la valeur de seuil $\theta$ était au moins égale à $\hat{w}$, la décision inverse (c.à.d. de ne pas demander $y$) aurait été prise. Ainsi, la valeur à soustraire de $\theta$ afin de prendre la décision inverse est : $\theta - \hat{w}$, parce que dans ce cas, la valeur du seuil aurait été $\theta - (\theta - \hat{w}) = \hat{w}$. Par conséquent, si $y$ est demandée et $y_1 = y$, alors $\theta$ est décrémentée proportionnellement à $\Delta_{dec} = \theta - \hat{w}$, comme suit :

$$\theta_{new} = \theta - \epsilon \times (\theta - \hat{w}) \tag{3.9}$$

La valeur $\theta$ devant être définie sur l'intervalle $]0, 1]$, il est important de noter qu'en diminuant $\theta$ de cette façon, sa valeur ne devient jamais inférieure à 0, même si elle est décrémentée une infinité de fois. En effet, à tout moment, nous soustrayons à $\theta$ une valeur plus petite qu'elle-même ($\epsilon \times (\theta - \hat{w})$), ainsi, le seuil qui en résulte est toujours positif $\theta - \epsilon \times (\theta - \hat{w}) > 0$.

**Minimiser les erreurs de prédiction** Pour le second cas, une erreur de prédiction s'est produite ($y_1 \neq y$) et $\theta$ devrait être augmenté. Si l'erreur de prédiction survient alors que le classifieur est confiant sur sa prédiction, alors $\theta$ devrait encore être augmenté. Une valeur élevée de $\hat{w}$ signifie que le classifieur est assez confiant sur sa prédiction. Ainsi, dans le cas d'une erreur de prédiction, $\theta$ devrait être augmenté proportionnellement à $\hat{w}$. Cependant, nous voulons aussi que $\theta$ soit toujours inférieur à 1, même s'il est augmenté une infinité de fois. Par conséquent, si $y$ est demandée et $y_1 \neq y$, nous augmentons $\theta$ proportionnellement à $\Delta_{inc} = \frac{\hat{w}\ (1-\theta)}{\theta}$, comme suit :

$$\theta_{new} = \theta + \epsilon \times \frac{\hat{w}\ (1-\theta)}{\theta} \tag{3.10}$$

En incrémentant $\theta$ de cette façon, il est augmenté proportionnellement à $\hat{w}$ et on peut facilement prouver que les valeurs résultantes de $\theta$, même s'il est incrémenté une infinité de fois, sont toujours inférieures à 1. En effet, vu que l'étiquette $y$ est demandée lorsque $\hat{w} < \theta$, nous avons :

$$\hat{w} < \theta \Leftrightarrow \hat{w}\frac{1-\theta}{\theta} < 1-\theta \Leftrightarrow \theta + \hat{w}\frac{1-\theta}{\theta} < 1 \Leftrightarrow \theta + \epsilon \times \frac{\hat{w}(1-\theta)}{\theta} < 1$$

En adaptant $\theta$ selon les équations 3.9 et 3.10, ses valeurs constituent une séquence qui est positive et bornée supérieurement par 1, donc à tout moment $\theta \in ]0, 1[$. L'algorithme général d'apprentissage actif à partir d'un flux en utilisant un seuil adaptatif est décrit par l'algorithme 4. N'importe quel classifieur peut être utilisé pour apprendre le modèle $h$.

**Algorithm 4** Apprentissage actif séquentiel avec un seuil adaptatif

**Entrées :** modèle de classification $h$, valeur initiale du seuil $\theta$ (e.g. 0.5)
**pour chaque** nouvelle instance $x$ à partir du flux **faire**
    Soit $y_1 = h(x) = \underset{y}{\operatorname{argmax}}\, P_h(y|x)$, $y_2 = \underset{y \neq y_1}{\operatorname{argmax}}\, P_h(y|x)$
    Obtenir $\bar{h}_\theta$ en entraînant $h$ avec $(x, y_2, \theta)$
    **si** $\bar{h}_\theta(x) \neq y_1$ **alors** // (équivalent à $\hat{w} < \theta$)
        demander $y$ la classe de $x$ à un opérateur
        calculer $\hat{w}$ en utilisant Algorithm 1
        **si** $y_1 = y$ **alors**
            $\theta \leftarrow \theta - \epsilon \times (\theta - \hat{w})$
        **sinon**
            $\theta \leftarrow \theta + \epsilon \times \frac{\hat{w}(1-\theta)}{\theta}$
        **fin si**
        entraîner $h$ en utilisant $(x, y)$
    **sinon**
        sortir $y_1$ la classe prédite pour $x$
    **fin si**
**fin pour**

## 3.5 Expérimentations

Afin d'évaluer la stratégie d'apprentissage actif proposée dans ce chapitre, un SVM est utilisé comme classifieur. Nous utilisons l'implémentation python disponible sur scikit-learn [Pedregosa 11].

Plus la constante $\mathcal{E}$, utilisée dans Algorithm 1, est petite, plus la valeur approximée du poids suffisant est proche de la vraie valeur $\hat{w}$. Ainsi, $\mathcal{E}$ est fixée à une petite valeur 0.0001 qui représente un bon

compromis entre l'approximation du poids suffisant $\hat{w}$ et la complexité de l'algorithme. En effet, avec cette valeur de $\mathcal{E}$, le poids suffisant pour une instance, est approximé par $\pm 0.0001$ après seulement 13 itérations : $\log_2 \frac{1}{\mathcal{E}} = \log_2 \frac{1}{0.0001} \simeq 13$.

Les expérimentations sont faites en deux temps. Dans la section 3.5.1, la mesure d'incertitude basée sur la notion de poids suffisant, est évaluée dans le cas statique afin de la comparer à d'autres mesures d'incertitude. Dans la section 3.5.2, le seuil adaptatif proposé est évalué avec la stratégie de poids suffisant dans le cas d'un flux de données.

Pour ces deux expérimentations, nous avons initialisé le modèle de classification en utilisant environ $10\%$ des données, choisies aléatoirement à partir de l'ensemble des données. Cependant, dans le cas de la base *pendigits*, la différence de performance entre les différentes stratégies comparées n'est perceptible qu'à la fin de l'apprentissage ; c'est pourquoi, pour cette base, le modèle de classification est initialisé en utilisant un nombre plus élevé d'échantillons. Les données utilisées pour l'initialisation du modèle de classification, couvrent toutes les classes possibles, afin de pouvoir demander à un opérateur les étiquettes des données dont l'incertitude est déterminée selon ces classes connues. Ceci n'est plus le cas dans le chapitre 5 où le système peut démarrer de zero connaissance et détecter automatiquement les données appartenant à de nouvelles classes inconnues.

### 3.5.1 Stratégie du poids suffisant

Dans cette première expérimentation, nous évaluons la stratégie d'apprentissage actif proposée. La stratégie du "poids suffisant" est comparée à une stratégie aléatoire et deux stratégies basées sur des mesures d'incertitude couramment utilisées : la "marge" et l'"entropie" (voir paragraphe 3.3.2.1). Pour chaque stratégie, nous sélectionnons de manière itérative, pour l'étiquetage, l'instance la plus incertaine parmi un lot d'instances non étiquetées.

Les graphiques de la figure 3.3 montrent le taux de reconnaissance que l'on obtient sur une base de données de test ($D_t$) en utilisant le modèle appris $h$, en fonction du nombre d'étiquettes qui ont été demandées pour apprendre $h$. Chaque graphique correspond à un ensemble de données différent. Le nombre de données de chaque base de test $|D_t|$ représente environ $30\%$ du nombre total des données. Le taux de reconnaissance $R_h$ sur la base de test est calculé selon l'équation 3.11

$$R_h = \frac{1}{|D_t|} \sum_{(x_i,y_i)\in D_t} \mathbf{1}(h(x_i) = y_i) \tag{3.11}$$

où $\mathbf{1}(C)$ vaut 1 si la condition $C$ est remplie et 0 sinon.

Ainsi, chaque fois qu'une $n^{\text{ème}}$ donnée sélectionnée est étiquetée et utilisée pour entraîner $h$, le taux $R_h$ est calculé. Ceci permet d'évaluer la capacité de $h$ à classer correctement des données non vues de la base de test, en étant entraîné avec seulement $n$ données étiquetées choisies selon une stratégie d'apprentissage actif.

Dans la figure 3.3, les courbes en rouge montrent le taux de reconnaissance $R_h$ obtenu, en choisissant les instances à étiqueter, via la mesure du "poids suffisant" proposée. Les courbes en vert et bleu, représentent l'utilisation des mesures de "marge" et d'"entropie", respectivement. Les courbes en violet représentent une stratégie aléatoire qui n'est basée sur aucune mesure, et qui sélectionne au hasard les instances à étiqueter, indépendamment de leur caractère informatif.

Tout d'abord, nous pouvons remarquer sur la figure 3.3 que, pour toutes les bases de données, la stratégie aléatoire (courbes en violet) donne un taux de reconnaissance inférieur à celui des autres stratégies. Ceci confirme que l'utilisation d'une stratégie d'apprentissage actif, pour choisir les données à étiqueter, est toujours préférable au choix aléatoire. Par exemple, pour la base "MMA", en étiquetant $400$ instances choisies aléatoirement, le taux de reconnaissance obtenu est de $\sim 75,7\%$ ; il est plus grand

si les $400$ instances sont choisies via une mesure d'incertitude. Deuxièmement, pour les bases "MMA", "LIRMM", "DocSet", "Image-Segmentation" et "Pendigits" (i.e. toutes les bases à l'exception de la base "Optdigits"), nous pouvons constater que la stratégie du "poids suffisant" donne les meilleurs résultats. Par exemple, pour atteindre un taux de reconnaissance de $94\%$ dans le cas de la base "LIRMM", il a fallu étiqueter $\sim 60$ instances choisies via la mesure du "poids suffisant" ; or, pour atteindre ce même taux de reconnaissance, il faut étiqueter respectivement $\sim 105$ et $\sim 130$ instances, si elles sont choisies via la mesure de "marge" ou d'"entropie". Ainsi, la stratégie proposée en utilisant la mesure du "poids suffisant" permet toujours d'obtenir un meilleur taux de reconnaissance que les stratégies basées sur les mesures de "marge" ou d'"entropie", à l'exception de la base "optdigits" où les trois mesures atteignent une performance relativement égale. Cela prouve que les instances sélectionnées, en utilisant la stratégie proposée, sont généralement plus informatives.

### 3.5.2 Seuil d'incertitude adaptatif

Dans cette deuxième expérimentation, nous évaluons l'apprentissage actif à partir d'un flux de données en utilisant le seuil adaptatif proposé. Nous considérons tout d'abord un cas idéal où le modèle est entraîné en utilisant chaque nouvelle instance du flux étiquetée avec sa vraie étiquette de classe. Ce cas entièrement supervisé conduit à un modèle qui permet d'obtenir les meilleures performances en terme de taux de reconnaissance sur la base de test, vu que toutes les instances sont étiquetées manuellement. Ainsi, une bonne stratégie devrait atteindre un taux de reconnaissance qui est très proche du cas idéal, mais en utilisant beaucoup moins d'instances étiquetées. Le cas "entièrement supervisé" est comparé à la stratégie utilisant la "marge" et la stratégie proposée "poids suffisant". Ces deux dernières stratégies utilisent le seuil adaptatif proposé dans la Section 3.4.5.

Les figures 3.4, 3.5 et 3.6 illustrent les résultats obtenus sur les différents ensembles. Pour chaque ensemble de données, nous montrons (i) le taux de reconnaissance obtenu par le modèle appris sur un ensemble de tests, (ii) le nombre d'instances étiquetées parmi les instances reçues à partir du flux, et (iii) le nombre d'erreurs parmi les classes prédites qui sont retournées par l'algorithme (voir ligne $15$ de l'algorithme 4). La courbe en violet montre le cas entièrement supervisé, où chaque instance du flux est étiquetée manuellement. Les courbes en vert et en rouge, montrent respectivement l'utilisation de la mesure de "marge" et du "poids suffisant" pour choisir les instances à étiqueter.

Tout d'abord, nous observons, pour tous les ensembles de données, qu'un taux de reconnaissance proche du cas "entièrement supervisé" (i.e. cas idéal) est obtenu dans les deux cas : "poids suffisant" et "marge". Nous observons aussi, pour tous les ensembles de données, que le nombre d'étiquettes demandées (i.e. nombre d'instances manuellement étiquetées) est beaucoup plus faible que celui du cas entièrement supervisé. Par exemple, pour la base "LIRMM" (voir la figure 3.4(b)), le nombre d'étiquettes demandées dans le cas "entièrement supervisé" est supérieur à $1200$. Ce même nombre est proche de $200$ seulement dans les deux cas : "poids suffisant" et "marge". Ainsi, l'utilisation d'un seuil adaptatif dans les deux cas "marge" et "poids suffisant", permet d'obtenir un taux de reconnaissance proche du cas parfait, tout en minimisant considérablement le nombre d'étiquettes demandées. Ceci prouve l'efficacité du seuil adaptatif proposé. De plus, on peut voir pour la plupart des bases, que la stratégie de "poids suffisant" atteint un taux de reconnaissance qui est plus proche du cas "entièrement supervisé" et parfois même égal. Ceci est particulièrement vrai pour les bases "MMA" (figure 3.4(a)), "Pendigits" (figure 3.5(c)), "DocSet" (figure 3.5(d)) et "Optdigits" (figure 3.6(f)), où les courbes de taux de reconnaissance en rouge et en violet, sont quasi superposées. Par ailleurs, on peut voir pour toutes les bases de données, que le nombre d'erreurs de prédiction est plus petit dans le cas de la stratégie du "poids suffisant", sauf pour la base "Image-segmentation" (figure 3.6(e)) où la stratégie de "marge" réalise un meilleur résultat.

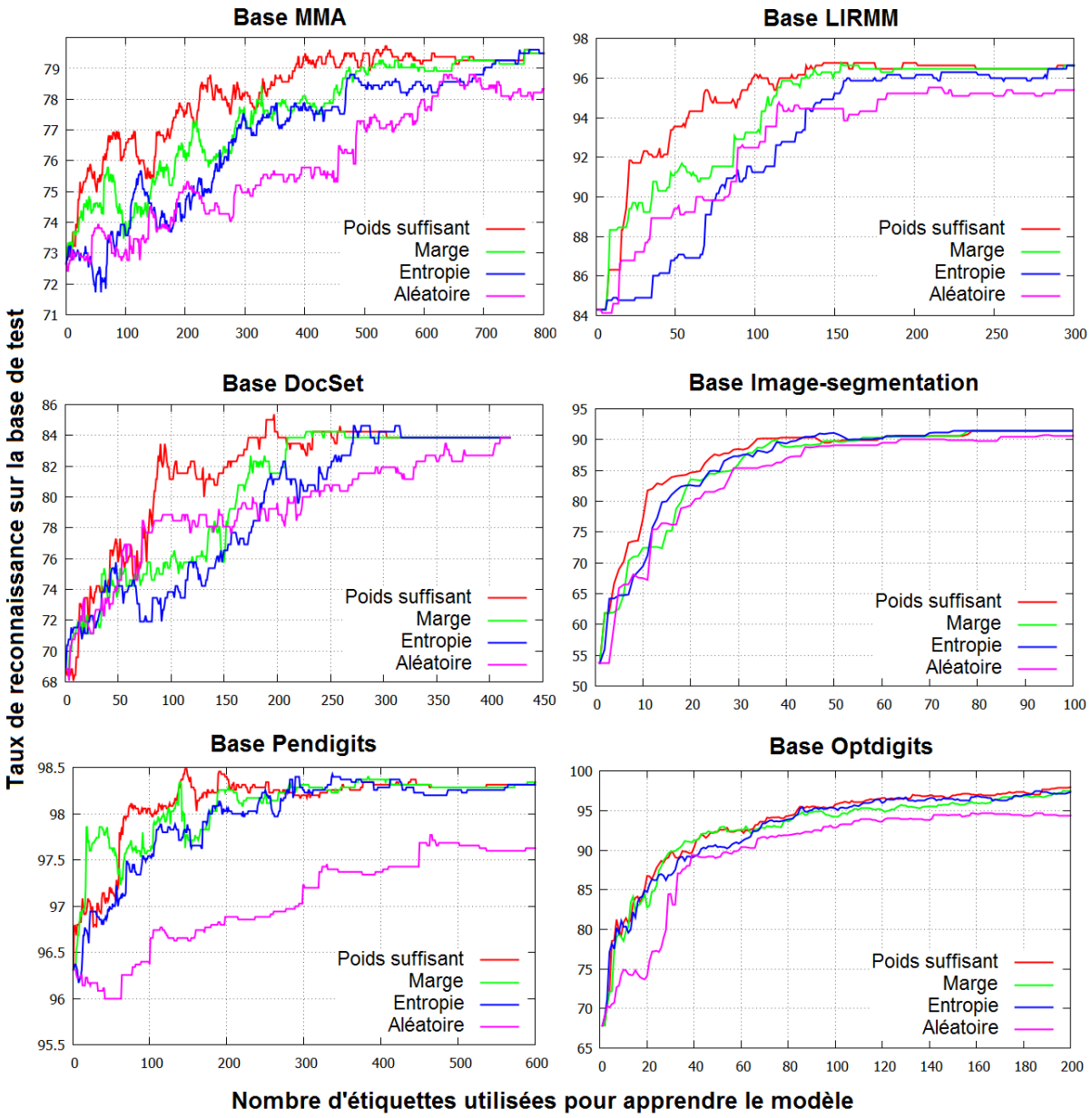



FIGURE 3.3 – Apprentissage actif dans le cas statique

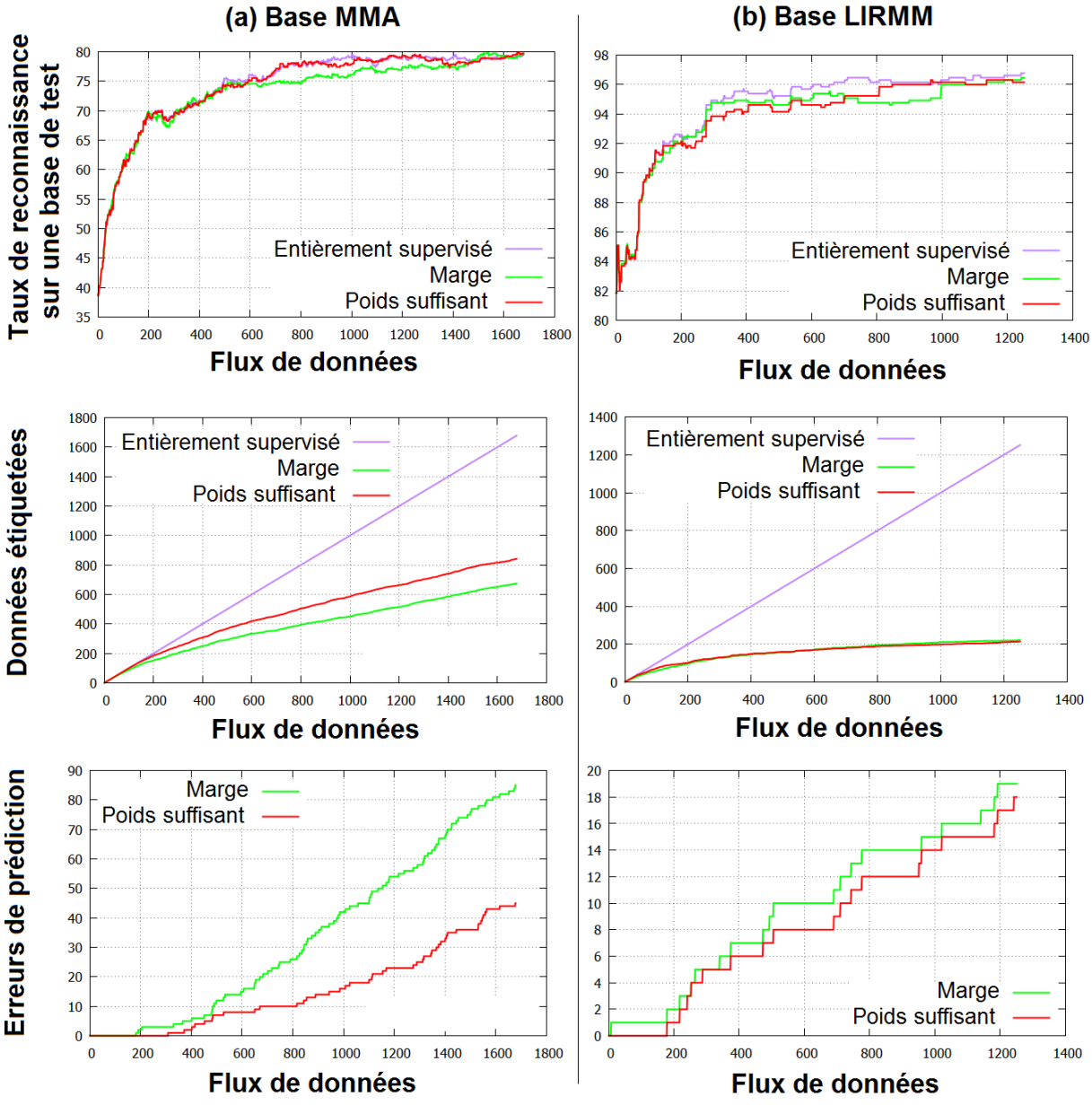



FIGURE 3.4 – Apprentissage actif à partir d'un flux de données, bases MMA et LIRMM

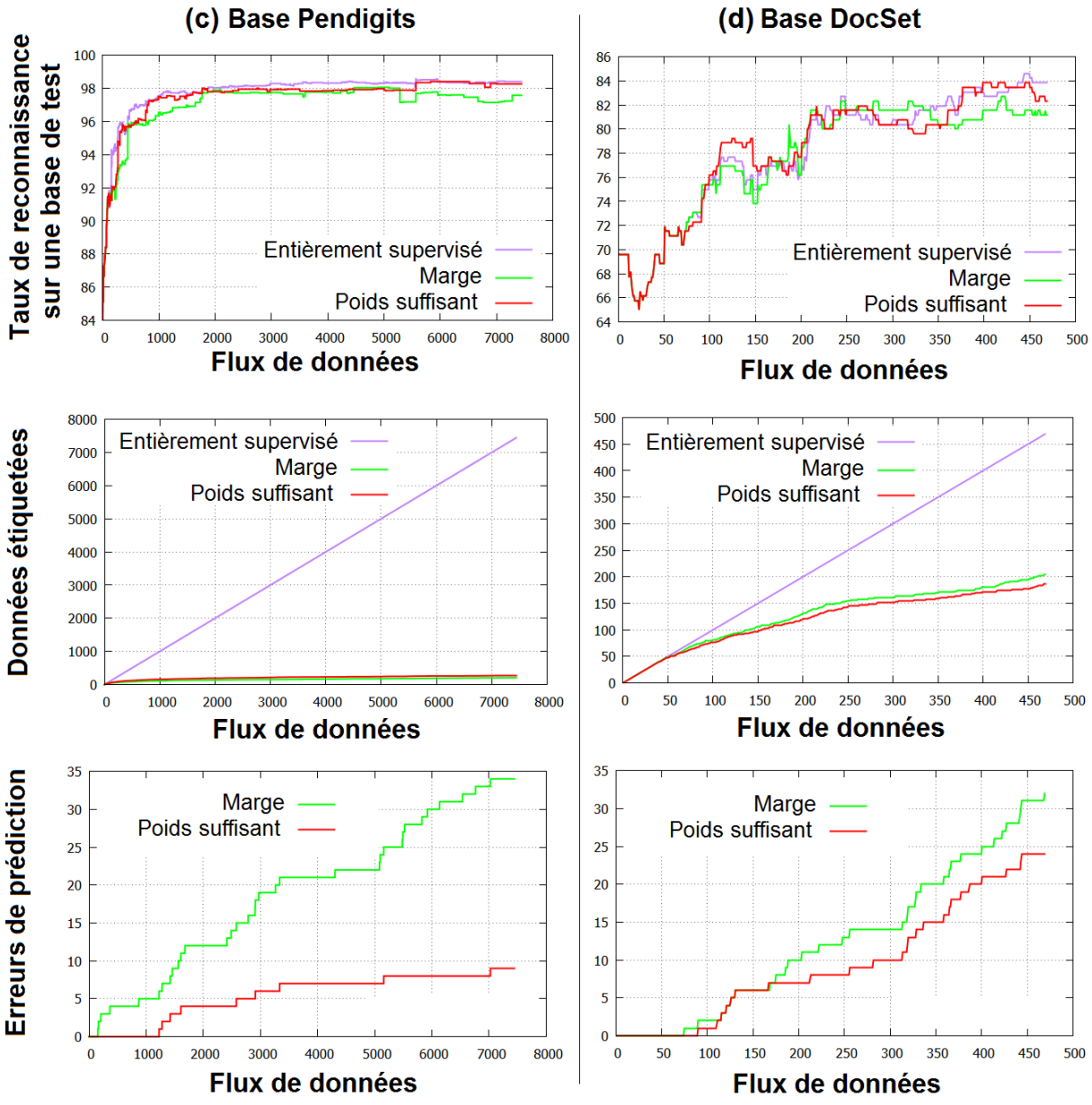



FIGURE 3.5 – Apprentissage actif à partir d'un flux de données, bases Pendigits et DocSet

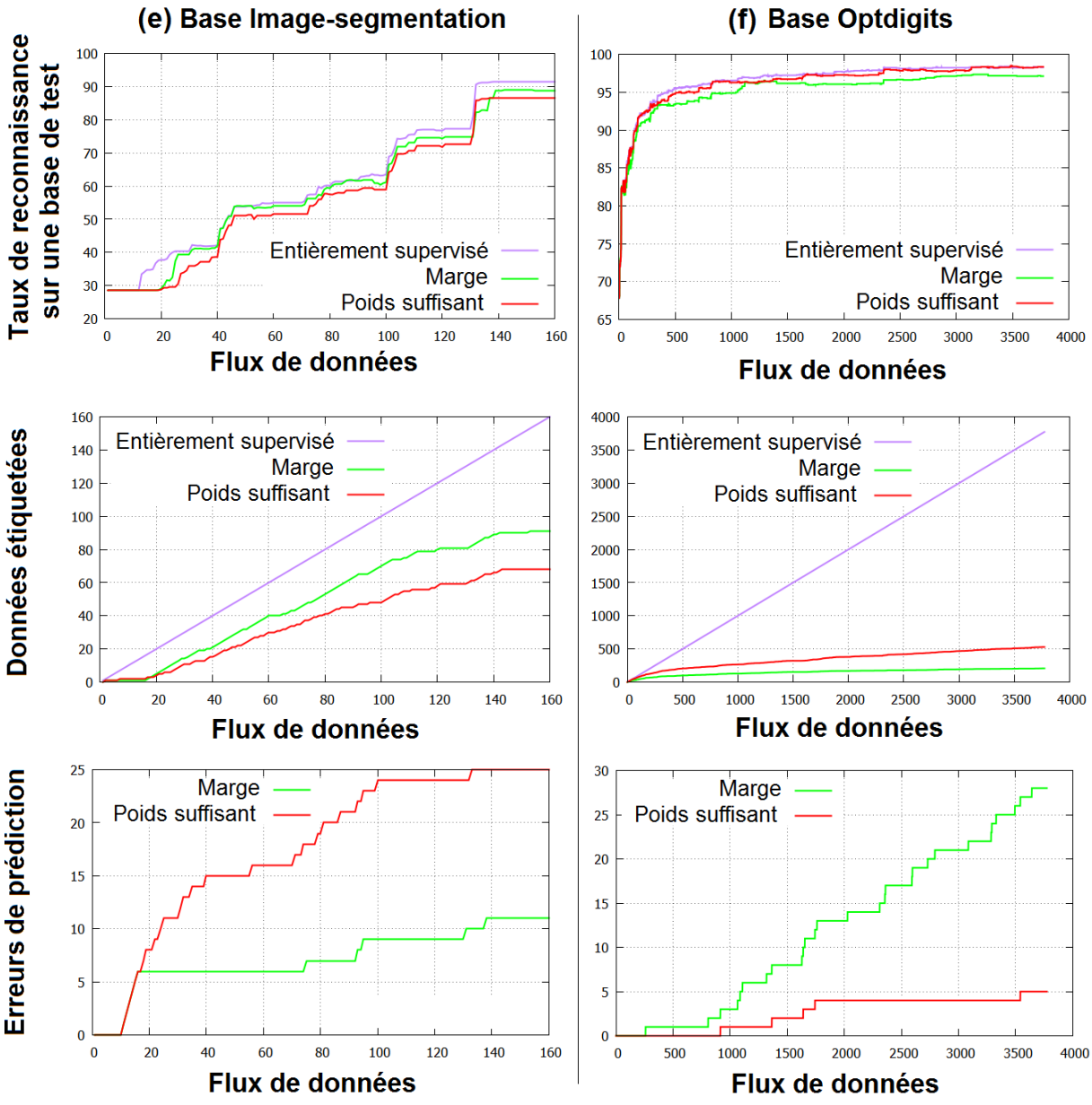



FIGURE 3.6 – Apprentissage actif à partir d'un flux de données, bases Segmentation et Optdigits

## 3.6 Conclusion

Dans ce chapitre, nous avons abordé le problème de l'apprentissage actif à partir d'un flux de données. Nous avons d'abord présenté les différentes stratégies existant dans la littérature. Puis, nous avons proposé une nouvelle stratégie d'apprentissage actif pour sélectionner les instances qui doivent être étiquetées, en se basant sur la pondération d'instances et l'incertitude définie par le principe du "poids suffisant". La stratégie proposée améliore significativement les performances de l'apprentissage actif par rapport aux stratégies d'incertitude couramment utilisées. Ensuite, nous avons proposé un seuil adaptatif qui peut être utilisé avec n'importe quelle mesure d'incertitude. Le seuil adaptatif proposé permet à l'apprentissage actif d'atteindre un taux de reconnaissance qui est très proche du cas idéal (c.à.d. entièrement supervisé) tout en nécessitant beaucoup moins d'étiquetage.

Dans le chapitre suivant, nous proposons une méthode d'apprentissage incrémentale semi-supervisée active pour la classification de flux de données, qui instancie la stratégie d'apprentissage actif proposée dans ce chapitre. Cette méthode sera ensuite étendue dans le chapitre 5 afin de traiter le problème de détection de nouvelles classes dans un flux de données.

# Chapitre 4

# Apprentissage de la topologie des données

**Sommaire**



## 4.1 Introduction

Ce chapitre présente une méthode d'apprentissage incrémental pour la classification de flux de données qui utilise la stratégie d'apprentissage actif définie dans le chapitre précédent. En se basant sur un algorithme d'apprentissage non supervisé appelé IGNG (en anglais "Incremental Growing Neural Gas"), nous proposons un algorithme d'apprentissage semi-supervisé actif qui permet de maintenir un graphe de nœuds étiquetés (représentants de données) et qui apprend de façon incrémentale la topologie des données entrant à partir du flux. Cette méthode sera utilisée et étendue dans le chapitre 5 pour traiter le problème de détection de nouvelles classes.

Nous décrivons dans la section 4.2 une famille de méthodes neuronales d'apprentissage non supervisé, qui sont à l'origine de la méthode proposée dans ce chapitre. Dans la section 4.3 nous détaillons notre extension de l'algorithme "IGNG" en un algorithme d'apprentissage semi-supervisé actif que nous appelons "A2ING" (en anglais "Active Adaptive Incremental Neural Gas"). Nous terminons ce chapitre par quelques expérimentations.

## 4.2 Méthodes neuronales d'apprentissage non supervisé

Il existe dans la littérature différentes méthodes permettant de faire un apprentissage non supervisé (clustering) de façon incrémentale, à partir d'un flux de données de longueur infinie, notamment, les

méthodes neuronales non supervisées qui prennent en compte les relations de voisinage entre les représentants des données et font preuve de bonnes performances. Parmi ces méthodes neuronales, l'algorithme GNG [Fritzke 95] ainsi que ses dérivées ont attiré une attention toute particulière. Ils permettent d'apprendre la topologie des données en maintenant un graphe de façon dynamique, à l'aide d'une stratégie d'apprentissage Hebbien compétitif [Martinetz 93], dont les nœuds, aussi appelés neurones, sont des représentants de données.

### 4.2.1 Algorithme NG

La méthode Neural Gas (NG) [Martinetz 91] est une méthode d'apprentissage non-supervisée inspirée de l'approche des cartes auto-organisatrices (en anglais, SOM : Self Organizing Maps) de Kohonen [Kohonen 90]. Cette approche cherche une représentation optimale, par des vecteurs de caractéristiques, d'un ensemble de données en entrée. L'appellation "gaz neuronal" est due à la dynamique des vecteurs de caractéristiques des neurones au cours du processus d'apprentissage, qui se répartissent comme un gaz dans l'espace d'entrée.

Soit un ensemble fini de $N$ neurones $n_i \in \mathbb{R}^d, i = 1, ..., N$ présentés sous forme de vecteurs de caractéristiques. À chaque instant, une nouvelle donnée $x$ (appelée instance dans la suite) est présentée à l'algorithme et la distance de cette donnée vers les neurones $n_i$ est calculée. Notons $i_0$ l'indice du neurone le plus proche de $x$ (c.à.d., $n_{i_0}$), $i_1$ l'indice du second neurone le plus proche de $x$, etc. jusqu'à $i_{N-1}$ qui est l'indice du neurone le plus éloigné de $x$. Chaque neurone ($k = 0, ..., N-1$) est adapté en fonction de cette nouvelle donnée $x$ en corrigeant sa position selon l'équation 4.1.

$$n_{i_k} \leftarrow n_{i_k} + \epsilon \times e^{\frac{-k}{\lambda}} \times (x - n_{i_k}) \tag{4.1}$$

où $\epsilon \in ]0, 1[$ est un taux d'apprentissage et $\lambda$ est un paramètre appelé "rang de voisinage".

À chaque adaptation, non seulement le neurone le plus proche de l'entrée $x$ mais aussi tous les autres neurones, sont repositionnés, c.à.d. adaptés par un pas d'adaptation $\epsilon \times e^{\frac{-k}{\lambda}}$ qui décroit avec l'accroissement de l'ordre de distance $k$. Après un nombre suffisant d'étapes d'adaptation (c.à.d. un nombre suffisant de données en entrée), les neurones couvrent l'espace de données avec une erreur de représentation minimale. Néanmoins, un inconvénient majeur de cet algorithme est que le nombre de neurones est fini et fixé au départ.

### 4.2.2 Algorithme GNG

L'algorithme Growing Neural Gas (GNG) [Fritzke 95] résout ce problème en permettant au nombre de neurones d'évoluer. Les neurones dans GNG représentent des nœuds dans un graphe. Ils seront "adaptés" au cours de l'apprentissage, en fonction des caractéristiques de la distribution des données. Ainsi, un nombre minimal de nœuds est initialement créé, puis, de nouveaux nœuds et de nouvelles connexions de voisinage (arêtes) entre eux sont ajoutés au fil de l'apprentissage. L'algorithme permet également de supprimer des nœuds ainsi que les connexions entre eux en se basant sur une notion d'âge limite des arêtes $a_{max}$. Les arêtes qui ont atteint cet âge, sans avoir été "rafraîchies", sont supprimées ainsi que les nœuds devenus isolés. Par ailleurs, la création d'un nouveau nœud se fait seulement périodiquement (après chaque nombre fixe d'itérations ou périodes de temps $\lambda$) entre les deux nœuds voisins qui ont cumulé l'erreur la plus importante dans la représentation des données. L'algorithme 5 décrit le fonctionnement de GNG.

Un inconvénient majeur de l'algorithme GNG concerne le choix de la période $\lambda$ (pour la création d'un nouveau nœud) et rien n'indique s'il faut ou non continuer à créer de nouveaux nœuds à une période donnée. Certaines adaptations ont été apportées à GNG pour faire face à cette contrainte d'évolution

**Algorithm 5** GNG

Initialiser un graphe $G$ avec deux nœuds choisis au hasard parmi les instances disponibles
$nbr \leftarrow 0$ *// nombre d'itérations*
**pour** chaque nouvelle instance $x$ **faire**
    Soient $n_1, n_2$ les deux nœuds les plus proches de $x$ dans $G$
    Incrémenter l'âge des arêtes liées à $n_1$
    Ajouter la distance au carrée, entre $x$ et $n_1$, à une variable d'erreur locale

$$err_{n_1} \leftarrow err_{n_1} + dist(x, n_1)^2$$

    Modifier $n_1$ et ses voisins $n_v$ (liés à $n_1$) pour se rapprocher de $x$ par les taux $\epsilon_1$ et $\epsilon_2$.

$$n_1 \leftarrow n_1 + \epsilon_1 \times (x - n_1)$$

$$n_v \leftarrow n_v + \epsilon_2 \times (x - n_v), \forall n_v \in \text{Voisins}(n_1)$$

    **si** $n_1$ est lié par une arête à $n_2$, **alors** remettre l'âge de l'arête à 0 (rafraîchir la connexion)
    **sinon** lier $n_1$ et $n_2$ par une nouvelle arête d'âge 0
    **fin si**
    Supprimer les arêtes avec un âge supérieur à $a_{max}$. Si ceci donne lieu à des nœuds isolés, alors les supprimer aussi.
    $nbr \leftarrow nbr + 1$
    **si** $nbr$ est multiple de $\lambda$ **alors** *// on a atteint une période de temps*
        Trouver le nœud $n_q$ ayant la plus grande erreur cumulée $err_{n_q}$
        Trouver le nœud $n_f$ ayant la plus grande erreur cumulée $err_{n_f}$ parmi les voisins de $n_q$
        Créer un nouveau nœud $n_{new}$ situé à mis distance entre $n_q$ et $n_f$

$$n_{new} \leftarrow 0.5 \times (n_q + n_f)$$

    **fin si**
**fin pour**

périodique. La principale adaptation est la version incrémentale de GNG, "IGNG" (Incremental Growing Neural Gas) [Prudent 05] qui relâche la contrainte relative à l'évolution périodique.

### 4.2.3 Algorithme IGNG

Le principe de base de IGNG [Prudent 05] est que la création d'un nouveau nœud se fait selon un seuil de distance global $r$. Chaque fois que la distance d'une instance d'entrée $x$ au nœud le plus proche est supérieure à $r$, un nouveau nœud basé sur $x$ est créé. Une version simplifiée de IGNG est présentée dans l'algorithme 6.

L'inconvénient principal de IGNG est que le seuil $r$ doit être fourni en tant que paramètre avant l'apprentissage, dépendant des données en présence. Or, ceci n'est pas facile à déterminer dans le cas où les données arrivent en continu, ce qui met en défaut le caractère incrémental annoncé par ses auteurs. Certaines extensions de IGNG ont été proposées pour essayer de remédier à ce problème en définissant un seuil propre à chaque nœud. Dans l'algorithme SOINN [Furao 07], le seuil associé à un nœud $n_i$ est défini heuristiquement comme étant la distance de $n_i$ à son voisin le plus éloigné si ce dernier existe, sinon la distance de $n_i$ au nœud le plus proche dans $G$. L'algorithme I2GNG (Improved IGNG) [Hamza 08] définit un seuil local $r_i$ à chaque nœud $n_i$, seuil qui est continuellement adapté durant l'apprentissage. Si

**Algorithm 6** IGNG

Initialiser un graphe $G$ avec deux nœuds choisis au hasard parmi les instances disponibles
**pour** chaque nouvelle instance $x$ **faire**
    Soient $n_1, n_2$ les deux nœuds les plus proches de $x$ dans $G$
    **si** $dist(x, n_1) > r$ **alors**
        Ajouter un nouveau nœud $n_{new}$ au graphe $G$ à la position de $x$
    **sinon**
        **si** $dist(x, n_2) > r$ **alors**
            Ajouter un nouveau nœud $n_{new}$ au graphe $G$ à la position de $x$
            Lier $n_{new}$ avec $n_1$ par une arête d'âge 0
        **sinon**
            Incrémenter l'âge des arêtes liées à $n_1$
            Modifier $n_1$ et ses voisins $n_v$ (liés par une arête à $n_1$) pour les rapprocher de $x$

$$n_1 \leftarrow n_1 + \epsilon_1 \times (x - n_1)$$

$$n_v \leftarrow n_v + \epsilon_1 \times (x - n_v), \forall n_v \in Voisins(n_1)$$

            **si** $n_1$ est lié par une arête à $n_2$, **alors** remettre l'âge de l'arête à 0 (rafraîchir la connexion)
            **sinon** lier $n_1$ et $n_2$ par une nouvelle arête d'âge 0
            **fin si**
            Supprimer les arêtes avec un âge supérieur à $a_{max}$. Si ceci donne lieu à des nœuds isolés, alors les supprimer aussi.
        **fin si**
    **fin si**
**fin pour**

aucune donnée n'est associée à un nœud $n_i$ alors un seuil par défaut $r$ (paramètre donné manuellement) est utilisé. Dans les autres cas, le seuil $r_i$, associé au nœud $n_i$, est défini selon l'équation 4.2

$$r_i = \bar{d}_{n_i} + \alpha \times \sigma_{n_i} \tag{4.2}$$

où $\bar{d_{n_i}}$ est la distance moyenne de $n_i$ aux instances qui lui sont associées, $\sigma_{n_i}$ est l'écart type correspondant, et $\alpha$, un paramètre.

L'inconvénient principal de I2GNG est lié à son initialisation. En effet, le choix des valeurs pour les paramètres $r$ et $\alpha$ est important car l'adaptation du seuil dépend fortement de ces derniers. Par exemple, si ces paramètres sont fixés à des valeurs trop petites, alors un nombre excessivement grand de nœuds peut être créé inutilement (et inversement si leur valeur est trop grande). Cela rend les systèmes utilisant un tel seuil dépendant d'un utilisateur expert et ne joue pas en faveur du caractère incrémental de l'algorithme.

## 4.3 Extension de "IGNG" en apprentissage semi-supervisé actif "A2ING"

Les algorithmes précédents sont non supervisés et ne sont pas directement applicables à la tâche de classification souhaitée. Nous proposons une extension de la méthode IGNG pour apprendre de façon semi-supervisée (à partir de données étiquetées et non étiquetées) un graphe $G$ dont les nœuds sont étiquetés. Nous appelons la méthode présentée ici "A2ING" pour "**A**ctive **A**daptive **I**ncremental **N**eural **G**as".

A2ING maintient un graphe $G$ (voir Figure.4.1) de nœuds étiquetés qui est modifié à chaque nouvelle arrivée d'une instance. Chaque nœud $n \in G$ est un représentant de données présenté sous forme d'un vecteur de caractéristiques qui est continuellement mis à jour.

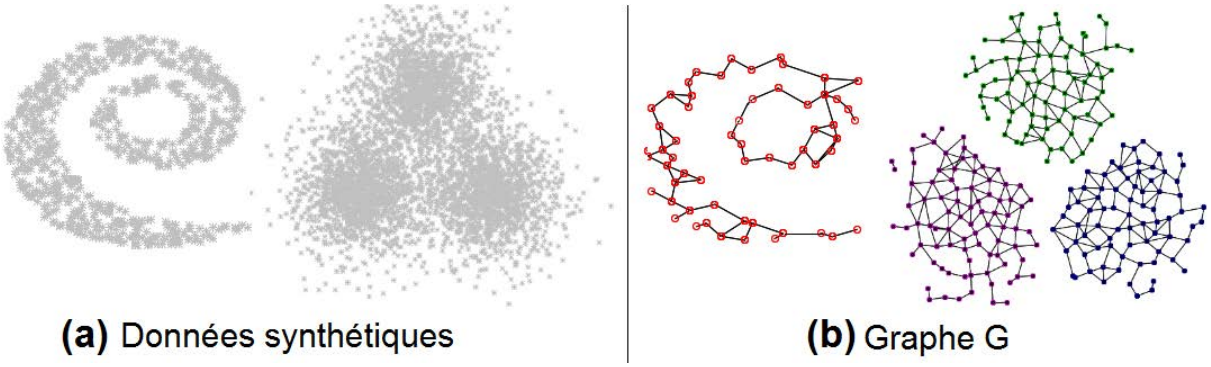



FIGURE 4.1 – A2ING apprend la topologie des données de façon incrémentale. (a) : données synthétiques à 2 dimensions. (b) : graphe $G$ de nœuds étiquetés

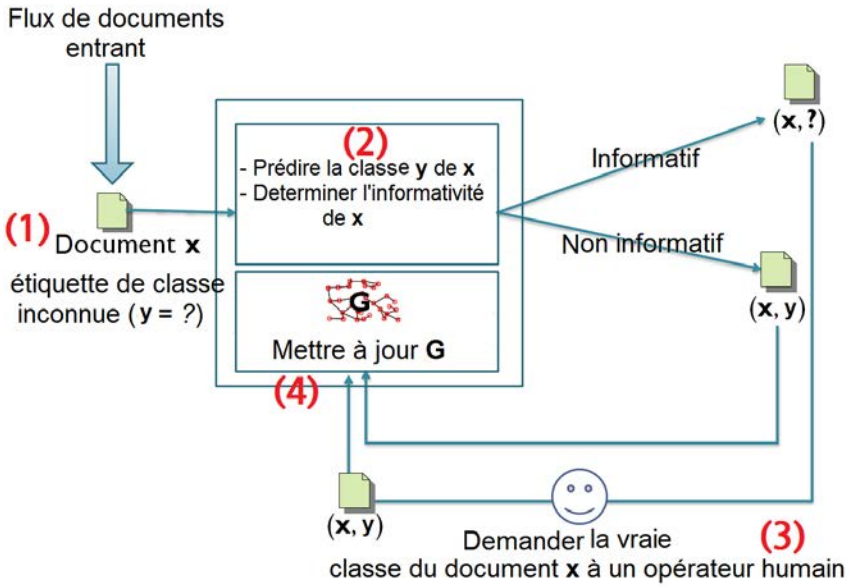



FIGURE 4.2 – Le schéma général de la méthode

Le graphe $G$ est initialisé avec quelques nœuds étiquetés qui représentent les premières instances du flux. Ensuite, pour chaque nouvelle instance $x$ (voir Figure 4.2 (1)), un indice d'incertitude est calculé pour déterminer si l'instance $x$ est informative (voir Figure 4.2 (2)). La méthode utilisée pour ce calcul, est celle qui est décrite dans le chapitre 3. Si l'instance est considérée comme informative, sa vraie classe (étiquette) $y$ est demandée à un opérateur (voir Figure 4.2 (3)) ; sinon, sa classe est prédite $y = h(x)$. L'instance classée $(x, y)$ est ensuite utilisée (voir Figure 4.2 (4)) pour mettre à jour le graphe $G$ tel que décrit dans la sous-section 4.3.2.

### 4.3.1 Classification et informativité d'une instance

Pour chaque nouvelle instance $x$, nous utilisons la méthode des $k$ plus proches voisins (KNN) pour calculer une probabilité d'appartenance à ses deux classes les plus probables.

Soit N($x$) = $\{n_i | i = 1, ..., k\}$ les $k$ nœuds étiquetés les plus proches de $x$ sélectionnés à partir de $G$, triés dans l'ordre croissant en fonction de leur distance à $x$. $P(y|x)$, la probabilité que l'instance $x$ appartienne à la classe $y$, est déterminée comme suit :

$$P(y|x) = \frac{\sum\limits_{n_i \in \mathrm{N}(x)} f(n_i, y)}{k} \tag{4.3}$$

où $f(n_i, y) = \begin{cases} 1 & \text{if } label(n_i) = y \\ 0 & \text{sinon} \end{cases}$

Pour une instance $x$ donnée, la fonction de classification $h(x)$ retourne l'étiquette de classe $y_1$ qui maximise la probabilité $P(y|x)$. Notons $y_1 = \underset{y}{\operatorname{argmax}}\, P(y|x)$ et $y_2 = \underset{y \neq y_1}{\operatorname{argmax}}\, P(y|x)$ respectivement les première et seconde classes les plus probables, telle que $P(y_1|x) \geq P(y_2|x)$.

Afin de calculer le poids suffisant tel que décrit dans la section 3.4, il suffit de recalculer la probabilité $P(y_2|x)$ en intégrant parmi les $k$ plus proches nœuds de $x$, l'instance $(x, y_2, w)$, c.à.d., l'instance $x$ étiquetée avec $y_2$ (plutôt que la classe prédite $y_1$) et associée à un poids $w$. Le poids suffisant $\hat{w}$ est alors celui qui fait que $P(y_2|x)$ devienne supérieure à $P(y_1|x)$.

Pour illustrer l'emplacement des instances incertaines (avec un faible poids suffisant) qui sont informatives pour notre modèle, la Figure 4.3 (a) montre trois classes de données synthétiques qui se chevauchent, et la Figure 4.3 (b), les données pour lesquelles on demande la classe, c'est-à-dire celles qui sont situées dans une région d'incertitude et donc considérées comme informatives. Connaître leur véritable étiquette de classe, permettra de mieux séparer les classes qui se chevauchent.

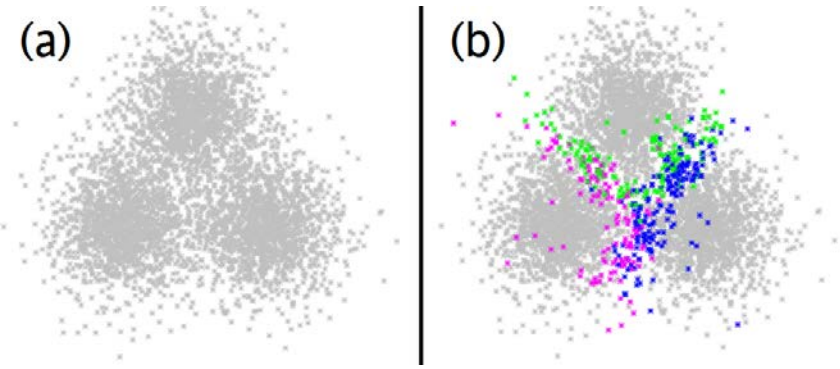



FIGURE 4.3 – (a) 3 classes de données synthétiques (suivant une distribution gaussienne) qui se chevauchent. (b) Les données incertaines (informatives).

### 4.3.2 Mise à jour du modèle

Soit $(x, y)$ une instance classée, où $x$ est une nouvelle instance reçue à partir du flux, et $y$ est soit l'étiquette de la classe prédite soit celle de la classe observée de $x$ (demandée à un opérateur). Soient $n_1 \in G$ et $n_2 \in G$ les deux plus proches nœuds de $x$, telle que $\mathrm{dist}(x, n_1) < \mathrm{dist}(x, n_2)$. Étant donné un seuil de distance $r$, l'algorithme met à jour de façon dynamique le graphe $G$ comme décrit ci-après, selon les 3 cas traités par l'algorithme IGNG et illustrés sur la Figure 4.4.

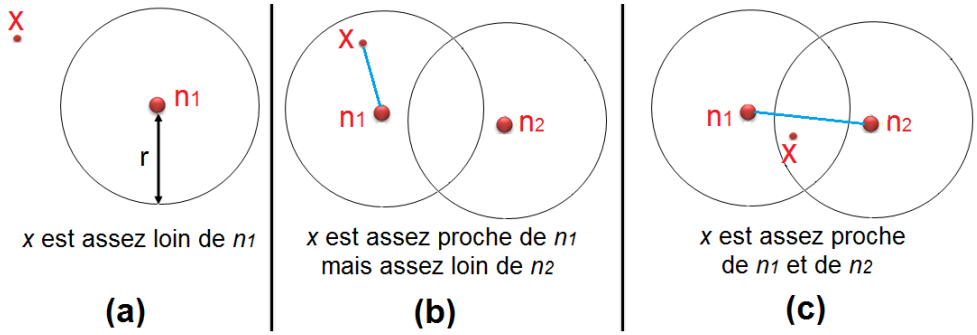



FIGURE 4.4 – les 3 cas de A2ING pour la génération et l'adaptation des nœuds

1. **si** $r < \text{dist}(x, n_1)$ alors (Figure 4.4(a))
   - Ajouter un nouveau nœud $n_{new} = x$ étiqueté par $y$ au graphe $G$
2. **si** $\text{dist}(x, n_1) \leq r < \text{dist}(x, n_2)$ alors (Figure 4.4(b))
   - Ajouter un nouveau nœud $n_{new} = x$ étiqueté par $y$ au graphe $G$
   - Relier $n_{new}$ à $n_1$ par une arête.
3. **si** $\text{dist}(x, n_1) < \text{dist}(x, n_2) \leq r$ alors (Figure 4.4(c))
   - Relier $n_1$ à $n_2$ par une arête (s'ils ne sont pas déjà reliés)

   - si $y = y_{n_1}$ ($x$ est étiqueté de la même façon que $n_1$)
     - Modifier $n_1$ pour le rapprocher de $x$, $n_1 \leftarrow n_1 + \epsilon_1 \times (x - n_1)$
     - Modifier les voisins de $n_1$ étiquetés différemment de $x$ pour les éloigner de $x$ $\forall n_v \in Voisins(n_1)$ et $y \neq y_{n_v} : n_v \leftarrow n_v - \epsilon_2 \times (x - n_v)$

   - si $y \neq y_{n_1}$ ($x$ est étiqueté différemment de $n_1$)
     - Modifier $n_1$ pour l'éloigner de $x$, $n_1 \leftarrow n_1 - \epsilon_1 \times (x - n_1)$
     - Modifier les voisins de $n_1$ ayant la même étiquette que $x$ pour les rapprocher de $x$ $\forall n_v \in Voisins(n_1)$ et $y = y_{n_v} : n_v \leftarrow n_v + \epsilon_2 \times (x - n_v)$

Lorsque $x$ est assez loin de $n_1$ ($1^{er}$ cas), un nouveau nœud basé sur $x$ et étiqueté avec $y$ est ajouté au graphe $G$. Lorsque $x$ est assez proche de $n_1$ mais assez loin de $n_2$ ($2^{eme}$ cas), un nouveau nœud basé sur $x$ et étiqueté avec $y$ est aussi ajouté au graphe $G$, et il est relié à $n_1$ par une nouvelle arête. Lorsque $x$ est assez proche de $n_1$ et de $n_2$, le nœud $n_2$ devient un voisin de $n_1$ (c.à.d. $n_1$ et $n_2$ sont liés par une arête). Si ces deux nœuds sont étiquetés différemment, l'arête les reliant sera appelée arête inter-classes. L'algorithme peut déterminer une meilleure séparation entre les classes, en éloignant deux nœuds voisins étiquetés différemment. En effet, si $n_1$ a la même étiquette que $x$ (i.e. $y = y_{n_1}$), alors $n_1$ est mis à jour pour être moins éloigné de $x$, et les nœuds voisins de $n_1$ qui sont étiquetés différemment de $x$, sont mis à jour pour être plus éloignés de $x$. D'autre part, si $n_1$ est étiqueté différemment de $x$ (i.e. $y \neq y_{n_1}$), alors $n_1$ est mis à jour pour être plus éloigné de $x$, et les nœuds voisins de $n_1$ qui sont étiquetés comme $x$ sont mis à jour pour être moins éloignés de $x$.

Pour rapprocher ou éloigner le nœud $n_1$ et ses voisins de $x$, on leur applique une mise à jour en utilisant un taux d'apprentissage : $\epsilon_1$ pour $n_1$ et $\epsilon_2$ pour ses nœuds voisins. Généralement, un taux d'apprentissage trop grand favorise une instabilité des nœuds, tandis qu'un taux d'apprentissage trop petit conduit à ce que les nœuds n'apprennent pas assez à partir des données qui leur sont attribuées. Les valeurs typiques sont $0 < \epsilon_1 \ll 1$ et $0 < \epsilon_2 \ll \epsilon_1$. Soit $N_{n_1}$ le nombre de documents attribués à $n_1$. Dans A2ING, $\epsilon_1 = \frac{1}{N_{n_1}}$, il diminue lentement, proportionnellement au nombre de données attribuées

à $n_1$ (plus $n_1$ apprend, plus il devient stable), et $\epsilon_2$ est tout simplement défini comme $\epsilon_2 = \frac{1}{N_{n_1} \times 100}$, c'est-à-dire une valeur 100 fois plus petite que la valeur de $\epsilon_1$ ($\epsilon_2 \ll \epsilon_1$).

Notez que cet apprentissage de la topologie des données est incrémental et n'a pas besoin de sauvegarder les données déjà vues. Le graphe étiqueté $G$ évolue dynamiquement en fonction des nouvelles données. L'apprentissage du graphe est décrit en détail dans l'algorithme 7

**Algorithm 7** A2ING

**Input :** graphe $G$ initialisé avec au moins deux nœuds étiquetés
**pour chaque** nouvelle instance $x$ à partir du flux **faire**
  Soit $(x, y)$ l'instance classée, où $y$ est la classe prédite ou demandée via l'algorithme 4
  Soient $(n_1, y_{n_1})$ et $(n_2, y_{n_2})$ les deux nœuds étiquetés les plus proches de $x$ dans $G$
  **si** $r < \text{dist}(x, n_1)$ **alors**
    $G \leftarrow G \cup \{(n_{new}, y) | n_{new} = x\}$
  **sinon si** $r < \text{dist}(x, n_2)$ **alors** // c.à.d. $\text{dist}(x, n_1) \leq r < \text{dist}(x, n_2)$
    $G \leftarrow G \cup \{(n_{new}, y) | n_{new} = x\}$
    Relier $n_{new}$ à $n_1$ par une arête.
  **sinon** // c.à.d. $\text{dist}(x, n_1) < \text{dist}(x, n_2) \leq r$
    Relier $n_1$ à $n_2$ par une arête (s'ils ne sont pas déjà reliés)
    **si** $y = y_{n_1}$ **alors**
      $n_1 \leftarrow n_1 + \epsilon_1 \times (x - n_1)$
      $\forall n_v \in Voisins(n_1) \text{ et } y \neq y_{n_v} : n_v \leftarrow n_v - \epsilon_2 \times (x - n_v)$
    **sinon**
      $n_1 \leftarrow n_1 - \epsilon_1 \times (x - n_1)$
      $\forall n_v \in Voisins(n_1) \text{ et } y = y_{n_v} : n_v \leftarrow n_v + \epsilon_2 \times (x - n_v)$
    **fin si**
  **fin si**
**fin pour**

## 4.4 Expérimentations

Nous considérons dans notre évaluation expérimentale différents ensembles de documents administratifs réels fournis par des clients de la société ITESOFT. Chaque document est d'abord traité par un OCR (type FineReader) et ensuite représenté par un sac de mots textuels, un vecteur où chaque composante représente le nombre de ses occurrences dans le document. Les vecteurs de chaque ensemble sont représentés dans un espace à $p$ dimensions, où $p$ est la taille du vocabulaire [2].

Nous testons, en plus de la méthode proposée, une variante active de SVM [Bordes 05]. Les données les plus proches de la limite de décision de SVM sont celles qui sont les plus susceptibles d'être étiquetées manuellement. Nous utilisons aussi, pour effectuer des comparaisons, un certain nombre de classifieurs classiques en les entraînant via un apprentissage entièrement supervisé tel que : KNN [Jiang 07 (2)], LogitBoost [Friedman 98], NaiveBayes [Lowd 05] et RandomForest [Breiman 01]. Pour rappel, la méthode que nous proposons offre deux qualités supplémentaires, comparée aux classifieurs considérés (sauf la variante active de SVM) : (i) seulement quelques documents considérés informatifs sont manuellement labellisés durant le processus de classification, (ii) la classification et l'apprentissage sont faits en parallèle en traitant les documents un par un à la volée (à la différence des méthodes classiques où chaque document peut être revisité de nombreuses fois au cours de l'apprentissage).

2. Ceci est le nombre de mots significatifs ou fréquents pour chaque ensemble de documents

Les mesures d'évaluation considérées sont :
– Le taux des instances qui ont été manuellement étiquetées.
– Le taux d'erreurs calculé sur une base de test.
– La précision moyenne sur une base de test.
– Le rappel moyen sur une base de test.

Pour une classe $y$ donnée, soient $vp_y$ le nombre d'instances correctement classées dans la classe $y$ (vrais positifs), $fp_y$ le nombre d'instances incorrectement classées dans la classe $y$ (faux positifs), et $fn_y$ le nombre d'instances qui ont pour vraie classe $y$, mais n'ont pas été classées dans la classe $y$ (faux négatifs).

La précision de la classe $y$ est $precision_y = \frac{vp_y}{vp_y+fp_y}$, elle exprime le ratio d'instances correctement classées dans la classe $y$ par rapport au nombre total d'instances classées dans la classe $y$.

La rappel de la classe $y$ est $rappel_y = \frac{vp_y}{vp_y+fn_y}$, il exprime le ratio de données correctement classées dans la classe $y$ par rapport au nombre total de données de la classe $y$.

Nous utilisons le rappel et précision moyens par rapport à toutes les classes. Le taux d'erreurs est calculé sur une base de test ($D_t$) différente. Le nombre de données de la base de test $|D_t|$ représente $\sim 30\%$ du nombre total des données. Le taux d'erreurs calculé correspond au nombre total d'instances incorrectement classées sur le nombre d'instances de la base de test. Ce taux est calculé selon l'équation 4.4

$$Error = \frac{1}{|D_t|} \sum_{(x_i,y_i)\in D_t} \mathbf{1}(h(x_i) \neq y_i) \tag{4.4}$$

où $\mathbf{1}(C)$ vaut $1$ si la condition $C$ est remplie et $0$ sinon.

Les résultats obtenus sont présentés dans le Tableau 6.1. Nous faisons quelques remarques à propos de ces résultats. Tout d'abord, dans la méthode développée, le nombre d'instances qui ont été étiquetées manuellement (en interrogeant un opérateur) représentent en moyenne $41.05\%$ du nombre total d'instances utilisées, ce qui est meilleur que le taux obtenu par le SVM actif. Ensuite, les autres méthodes sont entièrement supervisées et ont donc besoin que l'ensemble des données utilisées pour l'apprentissage soient étiquetées. Le Tableau 6.1 montre que pour la base *DocSet*, notre méthode réalise les meilleures performances. Pour la base *CAF*, notre méthode a nécessité l'étiquetage de moins de données que les autres méthodes, tout en atteignant des performances similaires. En ce qui concerne la base *LIRMM*, bien que la méthode proposée nécessite seulement l'étiquetage de $23.67\%$ des instances, elle réalise une performance comparable aux autres méthodes. Cependant, RandomForest et LASVM réalisent une meilleure performance. Pour la base *MMA*, SVM est légèrement supérieur à notre méthode, tandis que pour l'ensemble *APP*, notre méthode réalise les meilleures performances en termes de taux d'erreurs et de rappel. Enfin, les résultats moyens sur toutes les bases de données sont indiqués en bas du Tableau 6.1. Nous pouvons constater que la méthode proposée atteint en moyenne de meilleures performances en ce qui concerne la précision, et des performances comparables aux autres méthodes en ce qui concerne le rappel et le taux d'erreurs, tout en utilisant nettement moins de données étiquetées.

Ceci confirme l'efficacité de la stratégie d'apprentissage actif proposée, qui permet d'obtenir des résultats meilleurs ou comparables à ceux obtenus par un apprentissage supervisé classique, mais en étiquetant moins de données, et en réduisant donc le coût d'étiquetage.

## 4.5 Conclusion

Les algorithmes neuronaux d'apprentissage non supervisé sont intéressants pour l'apprentissage incrémental de la topologie des données en entrée. Cependant, ces algorithmes ne sont pas directement applicables pour la tâche de classification souhaitée. Nous avons donc présenté dans ce chapitre une

TABLE 4.1 – Résultats expérimentaux sur différentes bases de documents

| **Méthode** | **Étiquettes %** | **Erreur %** | **Précision %** | **Rappel %** |
|---|---|---|---|---|
| Base DocSet | | | | |
| **A2ING** | **49.13** | **18.8** | **83.2** | **81.1** |
| KNN | 100 | 25.7 | 76.0 | 74.2 |
| LogitBoost | 100 | 19.2 | 80.9 | 80.8 |
| NaiveBayes | 100 | 25.3 | 76.6 | 74.6 |
| RandomForest | 100 | 21.1 | 77.9 | 78.8 |
| SVM actif | 51.05 | 20.7 | 81.3 | 79.2 |
| Base CAF | | | | |
| A2ING | **61.78** | 29.8 | 74.4 | 70.2 |
| KNN | 100 | 31.6 | 74.7 | 68.4 |
| LogitBoost | 100 | 38.0 | 69.9 | 61.9 |
| **NaiveBayes** | 100 | **28.7** | **75.8** | **71.2** |
| RandomForest | 100 | 32.6 | 72.5 | 67.4 |
| SVM actif | 66.9 | 29.2 | 73.9 | 70.7 |
| Base LIRMM | | | | |
| A2ING | **23.67** | 4.7 | 95.6 | 95.2 |
| KNN | 100 | 5.6 | 94.6 | 94.3 |
| LogitBoost | 100 | 4.4 | 95.4 | 95.5 |
| NaiveBayes | 100 | 4.4 | 96.1 | 95.5 |
| **RandomForest** | 100 | **3.07** | **96.9** | **96.8** |
| SVM actif | 42.2 | 3.53 | 96.2 | 96.4 |
| Base MMA | | | | |
| A2ING | **43.11** | 24.9 | 78.4 | 75.0 |
| KNN | 100 | 27.2 | 76 | 72.8 |
| LogitBoost | 100 | 27.4 | 73.1 | 72.5 |
| NaiveBayes | 100 | 24.4 | 76.5 | 75.6 |
| RandomForest | 100 | 25.7 | 76 | 74.3 |
| **SVM actif** | 43.5 | **22.5** | **79.1** | **77.4** |
| Base APP | | | | |
| **A2ING** | **27.6** | **14.8** | 85.3 | **85.1** |
| KNN | 100 | 15.8 | 84.4 | 84.2 |
| LogitBoost | 100 | 18.9 | 81.6 | 81.0 |
| NaiveBayes | 100 | 22.8 | 81.1 | 77.1 |
| RandomForest | 100 | 16.0 | 84.3 | 83.9 |
| SVM actif | 30.2 | 15.22 | **85.7** | 84.8 |
| Résultats moyens sur tous les Base de documents | | | | |
| **A2ING** | **41.05** | 18.6 | **83.4** | 81.3 |
| KNN | 100 | 22.26 | 81.14 | 78.78 |
| LogitBoost | 100 | 21.58 | 80.18 | 78.34 |
| NaiveBayes | 100 | 21.12 | 81.22 | 78.8 |
| RandomForest | 100 | 19.69 | 81.25 | 80.24 |
| **SVM actif** | 46.77 | **18.23** | 83.24 | **81.7** |

extension semi-supervisée active de l'algorithme IGNG pour la classification de flux de données. La méthode A2ING permet d'apprendre à partir de données étiquetées et non étiquetées un graphe $G$ de nœuds étiquetés qui représentent la topologie des données en entrée.

Cependant, A2ING, tout comme les autres méthodes d'apprentissage actif, demande l'étiquetage des instances informatives dont l'informativité est déterminée en fonction des classes connues (déjà vues). Or, le flux étant de nature évolutive, de nouvelles classes inconnues peuvent apparaître à tout moment. Le graphe $G$ appris par A2ING définit un "espace couvert par les classes connues". Le volume de cet espace dépend du seuil de distance $r$ qui est utilisé dans IGNG et repris par A2ING. Ce seuil est fixé manuellement et n'est pas adapté au problème de détection de nouvelles classes. De même, les versions adaptatives de ce seuil, tel que défini par I2GNG par exemple, ne sont pas initialement pensées pour la détection de nouvelles classes. Nous étendons donc A2ING dans le chapitre suivant pour traiter de façon adaptative et efficace, le problème de détection de nouvelles classes.

# Chapitre 5

# Détection de nouvelles classes

**Sommaire**



## 5.1 Introduction

La plupart des méthodes existantes pour la classification de flux de données [Gaber 07, Aggarwal 07, Zliobaite 11] supposent un nombre fixe de classes dans le flux. Cependant, dans les scénarios réels, cette hypothèse n'est souvent pas satisfaite. En effet, il est difficile d'obtenir des instances étiquetées à partir de toutes les classes possibles. De plus, vu la nature évolutive du flux, de nouvelles classes (inconnues) peuvent apparaître à tout moment. Si ces nouvelles classes ne sont pas détectées, toutes leurs instances seront inévitablement classées par erreur dans des classes existantes (connues) avec lesquelles le classifieur aura déjà été entraîné. Ce problème majeur de classification de flux de données est généralement appelé "évolution du concept" (concept evolution) [3]

Un simple classifieur, par exemple, un simple SVM linéaire comme dans la Figure. 5.1, est capable de discriminer entre les classes avec lesquelles il a été entraîné, mais échoue lorsqu'il s'agit de classer les instances d'une nouvelle classe non détectée. L'identification de nouvelles classes est très importante car elle permet de réduire considérablement le travail à fournir par un humain pour corriger les erreurs de classification dues à une nouvelle classe non détectée.

3. Ceci est différent de ce qui est appelé dérive de concept (concept drift) [Zliobaite 11] où les changements concernent la relation entre les données existantes et leur classes associées.

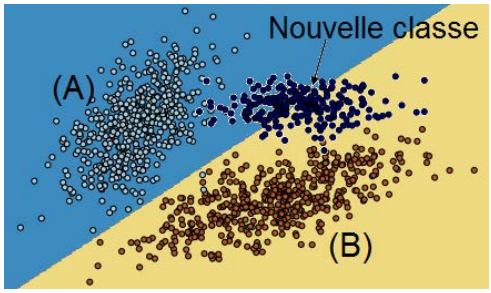



FIGURE 5.1 – Un SVM avec un noyau linéaire entraîné sur deux classes de données $A$ et $B$, classifie par erreur dans $A$ et $B$ les instances d'une nouvelle classe non détectée.

Comme expliqué dans le Chapitre 3, les méthodes d'apprentissage actif sont intéressantes pour la classification des flux de données, car elles réduisent le coût d'étiquetage manuel, en ne demandant à un opérateur humain que les étiquettes de classes des données qui sont informatives (souvent, les données incertaines). Cependant, ces méthodes supposent implicitement que les données utilisées pour leur initialisation, couvrent toutes les classes possibles, et elles demandent les étiquettes des instances informatives dont l'informativité est déterminée en fonction de ces classes connues. La détection de nouvelles classes n'est pas triviale pour ces méthodes parce qu'elles doivent demander des étiquettes pour des instances se trouvant dans différentes régions inexplorées de l'espace. Ceci représente une difficulté supplémentaire et un obstacle pour l'apprentissage actif afin de réduire la complexité de l'étiquetage [Dasgupta 05].

Nous verrons dans la Section 5.2 que les méthodes habituellement utilisées dans la littérature pour la détection de nouveautés ne sont pas adaptées pour un apprentissage actif à partir d'un flux de données multi-classes. Nous étendons ensuite l'algorithme A2ING, vu dans le chapitre 4, pour détecter automatiquement les nouvelles classes. La méthode proposée maintient progressivement une topologie de nœuds (représentants de données) qui évoluent de façon dynamique. Les nœuds sont des centres d'hypersphères couvrant les régions de l'espace des caractéristiques où les données de classes connues ont été observées. Une donnée se trouvant en dehors de la zone couverte, est appelée "étrangère" et peut être un membre d'une possible nouvelle classe. En collectant plusieurs données étrangères, notre algorithme est capable de distinguer avec précision les données étrangères qui sont similaires entre elles et qui sont donc plus susceptibles d'être membres d'une nouvelle classe, de celles qui sont plus susceptibles d'appartenir à des classes existantes qui sont en expansion. L'algorithme demande les vraies étiquettes de classes seulement pour les instances qui permettent de mieux discriminer entre les classes existantes et les nouvelles classes identifiées.

## 5.2 État de l'art

La détection de nouveautés dans les données est étroitement liée à la détection d'anomalies ou de données aberrantes ("outliers" en anglais). De nombreuses applications nécessitent d'être capables de décider si une nouvelle instance appartient à la même distribution que les instances existantes (c'est une donnée normale ou régulière), ou doit être considérée comme différente (c'est une donnée aberrante ou irrégulière). Cette capacité peut être utilisée pour nettoyer les ensembles de données, ou pour détecter des nouveaux patterns dans les données :

– Détection de nouveautés : la base d'apprentissage n'est pas polluée par des données aberrantes mais contient des données appartenant à de nouvelles classes.
– Détection des données aberrantes : la base d'apprentissage contient des données aberrantes qui

sont éliminées afin d'améliorer la qualité de l'apprentissage.

En général, les méthodes utilisées pour la détection de données aberrantes peuvent aussi être utilisées pour la détection de nouvelles classes. Cette section présente les méthodes qui sont habituellement utilisées dans la littérature. Tout d'abord, nous étudions dans la section 5.2.1 les méthodes basées sur la notion de distance. Ces méthodes déterminent la nouveauté des instances principalement par rapport à leur distance aux données voisines ou aux représentants de données. Nous verrons ensuite dans la section 5.2.2 des méthodes qui permettent de construire une frontière regroupant les données dites "normales" dans une classe et détectent toute donnée qui dévie de cette classe comme étant une donnée aberrante ou appartenant à une nouvelle classe. Enfin, nous étudions dans la section 5.2.3 les méthodes d'apprentissage actif à partir de flux de données qui prennent en considération la détection de nouvelles classes et d'autres méthodes statiques qui proposent des stratégies actives pour découvrir des classes inconnues qui n'étaient pas disponibles lors de la phase d'apprentissage.

### 5.2.1 Méthodes basées sur la distance

Des approches utilisant une mesure de distance (ou similarité) entre deux instances peuvent être utilisées pour la détection de nouvelles classes. Ces approches se basent principalement sur le clustering ou sur les méthodes des plus proches voisins.

Pour le clustering, les auteurs dans [Srivastava 06] utilisent des méthodes type k-means. Dans ce type de méthodes générales, une classe dite "normale" est caractérisée par un petit nombre de représentants dans l'espace d'entrée. Pour quantifier une nouveauté, la distance d'une instance de test au plus proche représentant est utilisée [Zhou 11]. Plus la distance est grande, plus il est probable qu'il s'agisse d'une instance d'une nouvelle classe. La mesure de distance utilisée est souvent une métrique classique tel que la distance Euclidienne. Dans [Ertoz 04], l'auteur explore un algorithme de clustering qui remplace la notion de distance classique par la notion des "plus proches voisins en commun". Cette technique cherche d'abord les plus proches voisins de chaque instance et redéfinit la similarité entre deux instances en termes de nombre de voisins communs entre elles. Une approche basée sur le clustering adaptée au flux de données est proposée dans [Spinosa 08]. Cette méthode suppose qu'il existe une seule classe connue et que toutes les autres classes sont nouvelles et n'est donc pas pas directement applicable à un flux de données multi-classes.

Les approches basées sur les plus proches voisins sont les méthodes les plus couramment utilisées pour la détection de données aberrantes et peuvent directement être utilisées pour la détection des données appartenant à de nouvelles classes, en considérant qu'une donnée "normale" est un membre d'une classe existante et qu'une donnée "aberrante" est un membre d'une nouvelle classe. La méthode des $k$ plus proches voisins (KNN) est basée sur l'hypothèse que les instances normales ont des voisins proches "normaux", alors que les instances aberrantes sont situées loin des instances "normales" [Hautamaki 04]. Une instance est donc déclarée comme aberrante ou appartenant à une probable nouvelle classe si elle se trouve loin de ses voisins. Par exemple, l'approche proposée dans [Angiulli 02] considère la somme des distances de chaque instance vers ses $k$ plus proches voisins et classifie les instances qui ont les plus grandes sommes comme des données aberrantes.

Certaines techniques, comme dans [Knorr 98], partitionnent l'espace d'entrée en une grille d'hypercubes de tailles fixes. Si un hypercube contient de nombreuses instances, ces instances sont susceptibles d'être normales. Inversement, si une instance de test se trouve dans un hypercube qui contient très peu d'instances, cette dernière est susceptible d'être une donnée aberrante.

Une autre méthode pour la détection de données aberrantes se basant sur la densité, a été proposée dans [Knorr 00]. Une mesure pour caractériser les données aberrantes est calculée pour chaque instance. Cette mesure permet de quantifier l'isolement d'une instance en calculant sa densité locale et les densités locales de ses voisins. La zone du voisinage pour une instance est déterminée par un nombre minimum

d'instances voisines, qui est un paramètre fourni par l'utilisateur. En comparant la densité locale d'une instance aux densités locales de ses voisins, il est possible d'identifier des régions de densités similaires, et des instances ayant une densité sensiblement inférieure à celle de leurs voisins. Celles-ci sont considérées comme des données aberrantes. Notez que cette mesure trie les instances en considérant seulement la densité des instances dans le voisinage et peut donc rater d'éventuelles instances aberrantes dont les densités sont proches de celles de leurs voisins. Dans [Tang 02], l'auteur présente une variante de la méthode précédente (proposée dans [Knorr 00]) qui considère à la fois la densité d'une instance de test dans son voisinage et une nouvelle mesure qui reflète le degré de connectivité de l'instance avec d'autres instances. Cette mesure est calculée en utilisant le rapport entre la distance moyenne de l'instance vers ses $k$ plus proches voisins et la distance moyenne de ses $k$ plus proches voisins avec leurs propres $k$ plus proches voisins. Les instances qui ont une valeur grande pour cette mesure sont plus susceptibles d'être des données aberrantes ou appartenant à de nouvelles classes. Cette approche s'est révélée être plus efficace pour la détection des données aberrantes, en particulier pour les données éparses, où les instances "normales" peuvent se trouver dans des régions de faible densité.

Une autre méthode est proposée dans [Yu 06] pour détecter des données aberrantes qui peuvent se trouver dans le voisinage de données "normales", plutôt que d'être loin d'elles. La similarité entre les instances est mesurée par un graphe de similarités pondéré. Un poids pour une paire d'instances indique la similarité entre elles. Une instance peut être considérée comme aberrante si sa similarité dans son voisinage (sa similarité avec ses voisins) est inférieure à la similarité de ses voisins dans leurs voisinages. Cette utilisation des graphes de similarité surmonte l'inconvénient des autres méthodes basées sur la similarité (qui supposent que les instances aberrantes sont très loin des instances "normales").

Les approches basées sur la distance ne nécessitent pas une connaissance a priori de la distribution des données. Cependant, les techniques basées sur les plus proches voisins, reposent sur l'existence de métriques de distance appropriées afin d'établir la similarité entre deux instances. En outre, la plupart de ces techniques identifient les nouvelles instances seulement de façon globale et ne sont pas assez flexibles pour détecter des nouveautés locales dans des données qui ont différentes densités et des formes arbitraires.

### 5.2.2 Méthodes basées sur la construction de frontières

Les méthodes de détection de nouveautés basées sur la construction de frontières essaient de répondre à la question suivante : est ce qu'une nouvelle instance est différente ou semblable à l'ensemble des données existantes ? Autrement dit, pouvons-nous la distinguer des autres données ou non ? Pour apporter une réponse, ces méthodes apprennent de façon non supervisée une frontière représentée dans un espace de dimension $d$, délimitant le contour des données initiales. Si de nouvelles données se situent à l'intérieur du sous-espace délimité par cette frontière, elles sont considérées comme provenant de la même population que les données initiales. Sinon, nous pouvons dire qu'elles sont anormales ou nouvelles avec une confiance proportionnelle à leur distance de la frontière.

Les SVMs représentent une technique connue pour former des frontières de décision qui séparent les données en différentes classes. La méthode SVM d'origine utilise un hyperplan qui maximise la marge de séparation entre deux classes. Les instances de la base d'apprentissage qui se trouvent près de la frontière de décision qui définit cette marge de séparation sont appelées vecteurs de support. Depuis l'introduction de cette méthode, plusieurs modifications et améliorations lui ont été apportées [Song 02, Hu 03]. Ces méthodes ne traitent pas le problème de détection de nouveautés, mais il existe cependant des SVMs, communément appelés "SVMs à une classe", utilisés pour la détection de nouveautés. Par exemple, dans [Scholkopf 01], l'auteur introduit une extension de l'algorithme SVM pour le cas de données non étiquetées, permettant de créer une frontière de décision englobant les données de la base d'apprentissage (une classe dite "normale", de données régulières). Comme pour la méthode SVM d'origine, cette méthode

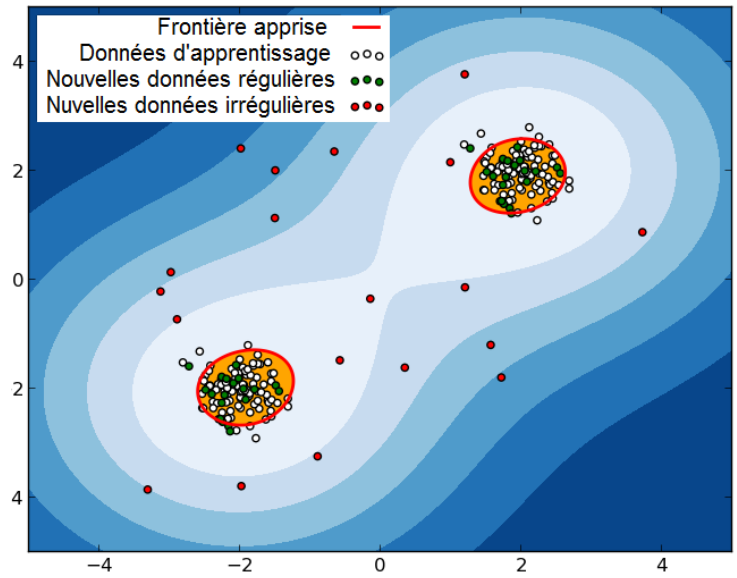



FIGURE 5.2 – Visualisation d'une frontière apprise par un SVM non supervisé avec un noyau non-linéaire. Source : scikit-learn.org

utilise seulement les instances qui se trouvent proches de la frontière et toutes les autres instances de la base d'apprentissage ne sont pas prises en compte dans la construction de la frontière de décision. L'appartenance de nouvelles données à la classe "normale" est alors déterminée selon leur position par rapport à la frontière. Dans [Li 03], l'auteur propose une extension de cet algorithme qui diffère du SVM d'origine, dans le sens où les données d'apprentissage sont traitées de manière séquentielle. Une autre méthode à une classe basée sur SVM est proposée dans [Scholkopf 99]. Cette dernière nécessite de fixer a priori le pourcentage d'instances "normales" qui sont autorisées à être en dehors de la description de la classe "normale" définie par la frontière de décision. Cependant, comme discuté dans [Manevitz 02], ce paramètre influe fortement sur la performance de cette approche.

Une autre approche appelée SVDD (Support Vector Data Description), proposée dans [Tax 99], définit la frontière de décision comme étant l'hypersphère de volume minimum qui renferme toutes les (ou la plupart des) instances d'apprentissage existantes. Cette méthode optimise automatiquement les paramètres du modèle en utilisant des données non étiquetées générées artificiellement, uniformément réparties dans l'hypersphère englobant la classe "normale". Une instance de test qui se trouve à l'intérieur de l'hypersphère est considérée comme étant une instance normale, sinon elle est aberrante ou appartient à une nouvelle classe.

La figure 5.2 montre la frontière apprise autour de quelques données synthétiques en utilisant un SVM à une classe [Scholkopf 01]. Les données d'apprentissage (régulières) sont visualisées par des cercles blancs, et les nouvelles données (régulières et irrégulières) par des cercles verts et rouges respectivement.

En résumé, les méthodes basées sur la construction de frontières présentées dans cette section apprennent une frontière de décision (une surface) sous forme d'une seule classe qui englobe les instances régulières, puis détectent tout simplement si une nouvelle instance s'écarte de cette classe. Ces méthodes sont généralement insensibles à la densité de la classe cible (définissant les données normales) et à la distribution des données, car elles décrivent seulement la frontière de cette classe et non pas sa densité. Cependant, il n'est pas facile de choisir des valeurs pour les paramètres qui contrôlent la taille de la frontière de décision. De plus, ces méthodes supposent qu'il n'y a qu'une seule classe de données "régu-

lières" et sont incapables de déterminer si une donnée "anormale" est une instance d'une nouvelle classe ou tout simplement une donnée aberrante.

La méthode que nous proposons est différente de ces méthodes traditionnelles à "une classe". Tout d'abord, notre méthode est capable de différencier différentes classes et détecter l'émergence d'une nouvelle classe dans le flux en déterminant s'il y a de fortes similitudes entre des instances non régulières, ce qui confirme la présence d'une nouvelle classe. De plus, le modèle de détection de nouvelles classes ainsi que ses paramètres évoluent de façon dynamique.

### 5.2.3 Méthodes basées sur l'apprentissage actif

Il existe des méthodes comme celles présentées dans [He 07, Escudeiro 12], qui fournissent des stratégies d'apprentissage actif pour découvrir, lors de la phase de test, des classes inconnues qui n'étaient pas disponibles lors de l'apprentissage. Par exemple, dans [Escudeiro 12], l'algorithme proposé utilise une mesure d'incertitude pondérée par la distance afin de demander l'étiquetage des instances qui sont à la fois incertaines par rapport aux classes connues (via une mesure d'incertitude comme celles vues dans le chapitre 3) et aussi loin d'elles. Dans [He 07], l'algorithme proposé détecte les instances qui sont potentiellement membres de nouvelles classes, en mesurant la densité locale autour de chaque instance dans la base d'apprentissage, puis sélectionne de façon itérative l'instance avec la variation maximale de densité, pour demander son étiquette. La variation de densité pour une instance est définie comme étant la différence entre la densité locale de cette instance et la densité locale d'une autre instance de son voisinage. Ainsi, une instance est considérée comme membre d'une classe inconnue si elle se trouve dans une région à forte densité et contient dans son voisinage une instance avec une faible densité. Étant donné l'ensemble des données $S$, la densité locale $D(x)$ d'une instance $x$ est mesurée selon l'équation 5.1 comme étant le nombre d'instances proches de $x$ à l'intérieur d'un rayon $r$ donné.

$$D(x) = |\{x_i | x_i \in S, dist(x, x_i) \leq r\}| \tag{5.1}$$

Ces méthodes nécessitent que l'ensemble des données soient disponibles à l'avance et ont besoin de parcourir ces données plusieurs fois, ce qui n'est pas envisageable pour un flux de données. De plus, la méthode proposée dans [He 07] a besoin de connaître à l'avance le nombre de classes et d'avoir une estimation de la taille de chaque classe (nombre d'instances dans chaque classe).

Une méthode de classification de flux de données avec détection de nouvelles classes est proposée dans [Masud 11]. Cette méthode est la plus proche de celle que nous proposons dans ce chapitre. Elle divise le flux de données en plusieurs grands blocs de taille identique et maintient un ensemble de classifieurs, chaque classifieur étant entraîné sur un bloc. Afin de détecter les nouvelles classes, une frontière de décision est créée pour les classes connues en appliquant la méthode de clustering K-means sur chaque bloc de données, ce qui réduit l'importance de cette méthode et la rend partiellement adaptée au flux de données. D'autre part, pour les données détectées comme membres d'une nouvelle classe, la méthode ne permet pas de choisir lesquelles sont les plus informatives pour demander leur étiquetage. En outre, bien que la méthode fournisse un moyen de détection de nouvelles classes, il est supposé qu'après la classification des données du bloc courant, toutes les vraies étiquettes des données de ce bloc seront obtenues ultérieurement. Ceci n'est pas concevable lorsque l'étiquetage est coûteux. Cette méthode a également été considérée avec un apprentissage actif dans [Masud 10]. Cependant la stratégie d'incertitude utilisée reste définie en fonction des classes connues. Par conséquent, les instances incertaines des classes connues sont présentées avec toutes les instances qui sont identifiées comme nouvelles, pour un étiquetage manuel. La méthode que nous proposons dans ce chapitre est différente dans le sens où elle maintient de façon incrémentale un modèle multi-classe qui est capable d'identifier l'émergence d'une

nouvelle classe dont les instances sont similaires entre elles. De plus, au lieu de demander l'étiquetage de toutes les instances nouvelles, la méthode peut sélectionner les instances pour lesquelles nous sommes le plus incertain de leur nouveauté, afin de demander leurs vraies étiquettes de classe.

## 5.3 Détection des nouvelles classes

L'algorithme A2ING, tel qu'il a été présenté dans le chapitre 4, se concentre sur la sélection d'instances incertaines dont l'incertitude est déterminée en fonction des classes existantes. En effet, cette incertitude est définie par la notion de "poids suffisant" qui fait basculer la prédiction du classifieur d'une classe existante $y_1$ à une autre classe existante $y_2$ (voir chapitre 3). Comme les méthodes habituelles d'apprentissage actif, A2ING n'explore pas explicitement les régions de l'espace où de nouvelles classes peuvent apparaître. Par conséquent, il ne parvient pas à détecter de nouvelles classes, sauf si par hasard quelques instances de ces nouvelles classes se trouvent dans sa région d'incertitude.

Dans cette section, nous étendons l'algorithme A2ING, en introduisant une nouvelle méthode de détection de nouvelles classes. Nous présentons d'abord dans la Section 5.3.1 une stratégie adaptative pour la détection de données dites "étrangères". Nous expliquons ensuite dans la Section 5.3.2 comment atteindre une meilleure distinction entre les classes existantes et nouvelles. Enfin, nous montrons dans la Section 5.3.3 comment demander les étiquettes des instances incertaines parmi celles d'une nouvelle classe détectée.

### 5.3.1 Espace couvert et détection adaptative de données étrangères

Chaque nœud $n \in G$ constitue le centre d'une hypersphère définie par le rayon $r$. Nous appelons zone ou espace couvert l'union de toutes les hypersphères. Une donnée $x$ est à l'extérieur (resp. à l'intérieur) de la zone couverte si la distance à son nœud le plus proche est supérieure (resp. inférieure) à $r$ (vu que toutes les hypersphères ont le même rayon $r$).

$$x \stackrel{\text{def}}{=} \text{étrangère} \iff \min_{n \in G} \text{dist}(x, n) > r$$

$$x \stackrel{\text{def}}{=} \text{régulière} \iff \min_{n \in G} \text{dist}(x, n) \leq r$$

Étant donné une valeur fixe de $r$, nous voulons observer combien de données étrangères (resp. régulières) sont effectivement membres de nouvelles classes (resp. de classes existantes). Nous effectuons alors un test sur la base "Optdigits" qui contient 10 classes : nous utilisons 7 classes dans le flux comme des classes connues (existantes), et nous conservons les 3 autres classes inconnues afin de les introduire à l'instant $t$ avec les autres instances des classes existantes. En utilisant différentes valeurs fixes de $r$ nous déterminons les taux suivants :

1. Instances étrangères qui sont vraiment des membres de nouvelles classes (précision de détection de nouvelles classes).
2. Instances régulières qui sont vraiment des membres des classes existantes (précision de détection de classes existantes).
3. Instances appartenant à de nouvelles classes qui sont détectées comme des instances étrangères (rappel de détection des nouvelles classes).
4. Instances appartenant à des classes existantes qui sont détectées comme des instances régulières (rappel de détection des classes existantes)

5. F-score pour les nouvelles classes et les classes existantes (moyenne harmonique de la précision et du rappel)
6. Bonnes distinctions (l'exactitude) entre les classes nouvelles et existantes.

La figure 5.3 montre la distinction correcte entre les instances appartenant aux nouvelles classes et celles appartenant aux classes existantes, en fonction de différentes valeurs de $r$. Lorsque $r$ est trop petit, de nombreuses instances sont détectées comme étrangères et la plupart d'entre elles sont donc considérées à tort comme des instances de nouvelles classes. De même, lorsque $r$ est trop grand, de nombreuses instances sont détectées comme régulières et la plupart d'entre elles sont à tort considérées comme des instances de classes existantes. Dans la figure 5.3, une valeur optimale de $r$ pour la base considérée, semble se situer autour de la valeur 23. En fait, il est difficile d'initialiser manuellement $r$ à une valeur convenable, car elle dépend fortement des données. Par conséquent, vu que notre méthode est active et semi-supervisée, nous utilisons les étiquettes associées aux instances pour ajuster automatiquement la valeur du rayon $r$.

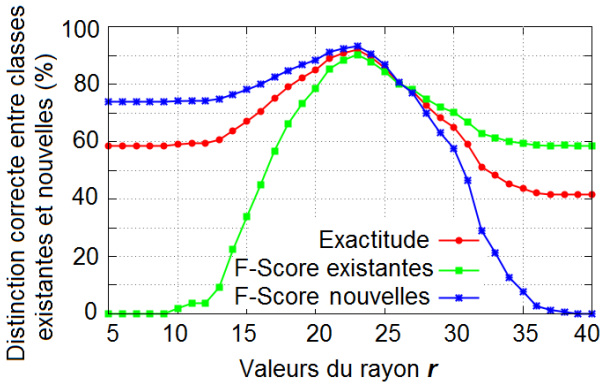



FIGURE 5.3 – Distinction des classes existantes et nouvelles en fonction de valeurs du rayon $r$, sur la base Optdigits. Une valeur optimale pour cette base est d'environ 23

Soit $x$ une donnée dont la classe est prédite en utilisant A2ING ou demandée (si $x$ est incertaine ou étrangère). Soit $n_1$ le nœud le plus proche de $x$ et $\epsilon$ (tel que $0 < \epsilon \ll 1$) une petite valeur constante représentant un taux d'apprentissage. $r$ est adapté de façon incrémentale comme suit :

1) Si $x$ est une instance étrangère et $label(x) = label(n_1)$ alors $x$ a été considéré comme donnée étrangère à tort. Dans ce cas, nous incrémentons $r$ comme suit :

$$r \leftarrow r + \epsilon \times |\operatorname{dist}(x, n_1) - r|$$

2) Si $x$ est une instance régulière et $label(x) \neq label(n_1)$ alors $x$ a été considéré comme donnée régulière à tort. Dans ce cas, nous décrémentons $r$ comme suit :

$$r \leftarrow r - \epsilon \times |\operatorname{dist}(x, n_1) - r|$$

Nous montrons dans les expérimentations que quelle que soit sa valeur initiale, $r$ converge toujours vers une valeur qui est proche de la valeur optimale. Autrement dit, la valeur initiale de $r$ n'affecte pas les résultats. Par conséquent, $r$ peut toujours être initialisé à la plus petite valeur initiale possible $r = 0$, car il est adaptatif.

### 5.3.2 Distinction entre les classes existantes et nouvelles

Selon la méthode décrite précédemment, nous décidons si une nouvelle instance appartient à une nouvelle classe ou à une classe existante dès que nous recevons cette instance. Néanmoins, en recueillant plus de données, nous sommes en mesure de distinguer plus précisément les instances des nouvelles classes de celles des classes existantes, en déterminant les données étrangères qui sont similaires entre elles. Cela est possible lorsque un temps d'attente est toléré jusqu'à ce que quelques données soient collectées à partir du flux, ou quand un mini-lot de données est traité globalement.

Soit $B$ un mini-lot (buffer) de nouvelles instances non étiquetées collectées à partir du flux. Ces instances sont d'abord séparées en deux ensembles : $N$ (ensemble des instances étrangères) et $E$ (ensemble des instances régulières) comme suit :

$$N = \{x \in B : \min_{n \in G} \mathrm{dist}(x, n) > r\}$$

$$E = \{x \in B : \min_{n \in G} \mathrm{dist}(x, n) \leq r\}$$

Nous rappelons que A2ING maintient le graphe $G$ de nœuds étiquetés représentant des classes existantes (avant d'intégrer les données de $B$). Soit $\tilde{G}$ le graphe de nœuds non étiquetés obtenus à partir de $N$ en utilisant l'algorithme 8.

**Algorithm 8** getUnlabelledGraph $(N, r)$

```
Initialiser G̃ avec deux nœuds choisis aléatoirement à partir de N
pour chaque x ∈ N faire
    Soit n1, n2 les deux nœuds les plus proches de x dans G̃
    si dist(x, n1) > r alors
        Ajouter un nouveau nœud n_new à G̃
    fin si
    si dist(x, n1) ≤ r < dist(x, n2) alors
        Ajouter un nouveau nœud n_new à G̃
        Lier n_new à n1 par une nouvelle arête
    fin si
    si dist(x, n1) < dist(x, n2) ≤ r alors
        Lier n1 à n2 par une nouvelle arête
        Mettre à jour n1 et ses nœuds voisins pour qu'ils soient plus proche de x
    fin si
fin pour
retourner G̃
```

Étant donné une instance $x \in B$, notons $d_x^E$ resp. $d_x^N$ la distance moyenne de $x$ à ses $k$ plus proches nœuds de $G$ resp. $\tilde{G}$ :

$$d_x^E = \frac{\sum_{n \in S_x} \mathrm{dist}(x, n)}{k} \qquad d_x^N = \frac{\sum_{n \in \tilde{S}_x} \mathrm{dist}(x, n)}{k}$$

où $S_x$ (resp. $\tilde{S}_x$) est l'ensemble des $k$ nœuds les plus proches de $x$ dans $G$ (resp. dans $\tilde{G}$).

La probabilité $P(N|x)$ que $x$ appartient à une nouvelle classe est proportionnelle à la distance $d_x^E$, et la probabilité $P(E|x)$ que $x$ appartient à une classe existante est proportionnelle à la distance $d_x^N$. Elles sont donc déterminées comme suit :

$$P(N|x) = \frac{d_x^E}{d_x^E + d_x^N} \qquad P(E|x) = \frac{d_x^N}{d_x^E + d_x^N}$$

**Algorithm 9** distinguishExistingNovel $(G, B, r)$

1: $E \coloneqq \{x \in B \mid \min_{n \in G} \text{dist}(x, n) \leq r\}$ // régulières
2: $N \coloneqq \{x \in B \mid \min_{n \in G} \text{dist}(x, n) > r\}$ // étrangères
3: **répéter**
4: $\tilde{G} = \text{getUnlabelledGraph}(N, r)$ // Appel à l'algorithme 8
5: $N' \coloneqq \{x \in N \mid P(N|x) < P(E|x)\}$ *// fausses nouvelles*
6: $E' \coloneqq \{x \in E \mid P(N|x) > P(E|x)\}$ *// fausses existantes*
7: $N \coloneqq \{x \in N \mid x \notin N'\} \cup E'$ *// déplacement des fausses existantes de $E$ vers $N$*
8: $E \coloneqq \{x \in E \mid x \notin E'\} \cup N'$ *// déplacement des fausses nouvelles de $N$ vers $E$*
9: **jusqu'à** $N' = E' = \emptyset$ // il n y a plus d'instance qui se déplace
10: **retourner** $N, E$

L'algorithme 9 montre comment atteindre une meilleure distinction entre les instances des nouvelles classes et celles des classes existantes. L'algorithme 8 est appelé afin d'obtenir un graphe de nœuds non étiquetés à partir de l'ensemble $N$, qui est utilisé pour déterminer les probabilités d'appartenance aux classes nouvelles ou existantes. L'ensemble $N'$ reçoit les instances dites "fausses-nouvelles" qui sont identifiées comme étant les instances de $N$ qui sont plus susceptibles d'appartenir à des classes existantes (ayant $P(N|x) < P(E|x)$). De même, l'ensemble $E'$ reçoit les instances dites "fausses-existantes". Ensuite, les deux ensembles $E$ et $N$ sont mis à jour en déplaçant les "fausses-nouvelles" instances sélectionnées à partir de $N$ vers $E$, et vice versa. Cette tâche peut être répétée un certain nombre de fois jusqu'à ce qu'il n'y ait plus d'instance qui se déplace entre les deux ensembles $N$ et $E$. Enfin, l'algorithme retourne les ensembles $N$ et $E$ des instances qui sont identifiées comme membres de nouvelles classes et de classes existantes respectivement.

La Figure. 5.4 montre le taux de bonne distinction entre les nouvelles classes et les classes existantes pour les instances qui sont dans $B$, durant chaque itération. L'algorithme se termine après seulement $5$ itérations. Dans le cas où aucune itération n'est effectuée, les instances de $B$ sont juste séparées en instances étrangères et instances régulières, et le taux obtenu de bonne distinction entre les classes nouvelles et existantes est de $88.15\%$. Après une seule itération, ce taux s'élève à $96.66\%$.

Selon l'application, il est possible d'équilibrer entre la rapidité de la détection de nouveautés et son efficacité en contrôlant la taille du buffer $|B|$. Par exemple, si une application donnée fonctionne en temps réel et ne tolère pas de temps d'attente, la taille du buffer peut être mise à $1$, ce qui s'avère être le même scénario que celui de la Section 5.3.1 (c.à.d. une instance étrangère va être immédiatement considérée comme membre d'une nouvelle classe). L'influence de la taille de $B$ sur le résultat final est discutée dans les expérimentations (voir Figure. 5.7).

### 5.3.3 Étiquetage des nouvelles instances incertaines

Après avoir distingué les instances qui sont probablement membres d'une nouvelle classe, le modèle doit être mis à jour avec certaines instances étiquetées de cette classe. Demander l'étiquetage de toutes les instances qui sont détectées comme nouvelles serait coûteux. Il est possible de choisir au hasard certaines instances, à partir de $N$ pour l'étiquetage manuel. Il est également possible de demander les étiquettes des nœuds obtenus à partir de $N$ en appliquant l'algorithme 8. Néanmoins, nous proposons une stratégie plus intéressante qui consiste à demander l'étiquetage des instances de $N$ pour lesquelles la méthode est la plus incertaine concernant leur nouveauté.

Les nouvelles instances qui sont incertaines se trouvent généralement dans les zones de chevauchement entre les nouvelles classes et les classes existantes. Soit la quantité $Q_x$ définie par l'équation

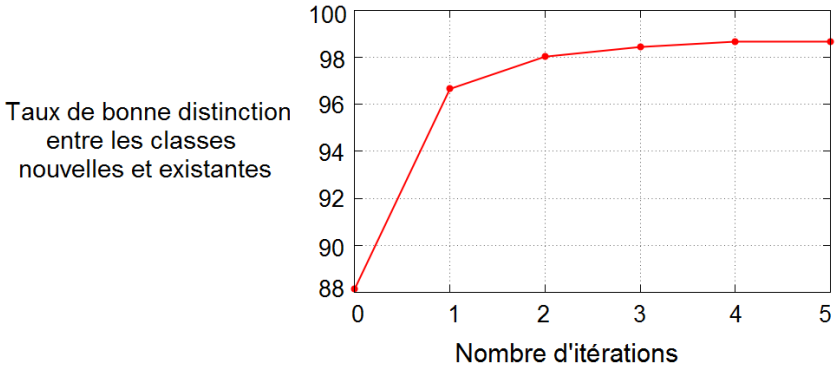



FIGURE 5.4 – Le taux de bonne distinction entre les classes nouvelles et existantes selon différentes itérations, sur la base Optdigits, avec un buffer de taille $|B| = 200$

suivante :

$$Q_x = |P(N|x) - P(E|x)|$$

Une instance incertaine $x$ aurait alors une faible valeur de $Q_x$ parce que la probabilité $P(N|x)$ que $x$ appartienne à une nouvelle classe serait proche de la probabilité $P(E|x)$ que $x$ appartienne à une classe existante. Par conséquent, au lieu de sélectionner au hasard $m$ instances de $N$ pour l'étiquetage manuel, nous sélectionnons les $m$ instances ayant les plus faibles valeurs $Q_x$. Ces instances sont informatives car connaître leurs vraies étiquettes de classe serait utile pour mieux discriminer entre les classes nouvelles et existantes.

## 5.4 Expérimentations

Dans cette section nous expérimentons, tout d'abord, la capacité de la méthode proposée à détecter correctement les instances étrangères de façon dynamique. Ensuite, nous testons la capacité de la méthode à mieux distinguer les classes nouvelles et existantes. Enfin, nous comparons la méthode proposée avec une méthode existante de détection de nouveautés, basée sur un SVM à une classe.

### 5.4.1 Capacité de détection des instances étrangères

La première série d'expérimentations permet de constater la capacité de la méthode proposée à détecter correctement les instances étrangères et les instances régulières à l'aide du seuil adaptatif $r$ représentant le rayon d'hypersphère décrit dans la section 5.3.1. Nous considérons qu'une donnée est correctement détectée comme étrangère (resp. régulière) si elle appartient à une classe nouvelle (resp. existante). Nous montrons notamment que, étant donné un ensemble de données et une valeur initiale de $r$ : (i) à terme, $r$ converge vers la même valeur, indépendamment de la valeur initiale, (ii) cette valeur est proche de la valeur optimale, et (iii) cette convergence nécessite peu de données.

La Figure. 5.5 (A) montre les valeurs de $r$ obtenues après adaptation, en fonction des valeurs initiales de $r$. Nous pouvons voir que, quelque soit sa valeur initiale, $r$ converge toujours vers des valeurs qui sont proches les unes des autres. La Figure. 5.5 (B) montre le taux de bonne distinction entre les données étrangères et régulières selon différentes valeurs de $r$. Nous constatons que lorsqu'on utilise une valeur fixe de $r$ (c.à.d. fixée manuellement et non adaptative), la valeur optimale du taux est obtenue avec $r = 23$ pour cette base de données. Tandis que lors de l'utilisation d'un $r$ adaptatif, le taux obtenu est toujours proche de l'optimal quel que soit la valeur de $r$ initiale. La Figure. 5.5 (C) montre la valeur courante de $r$

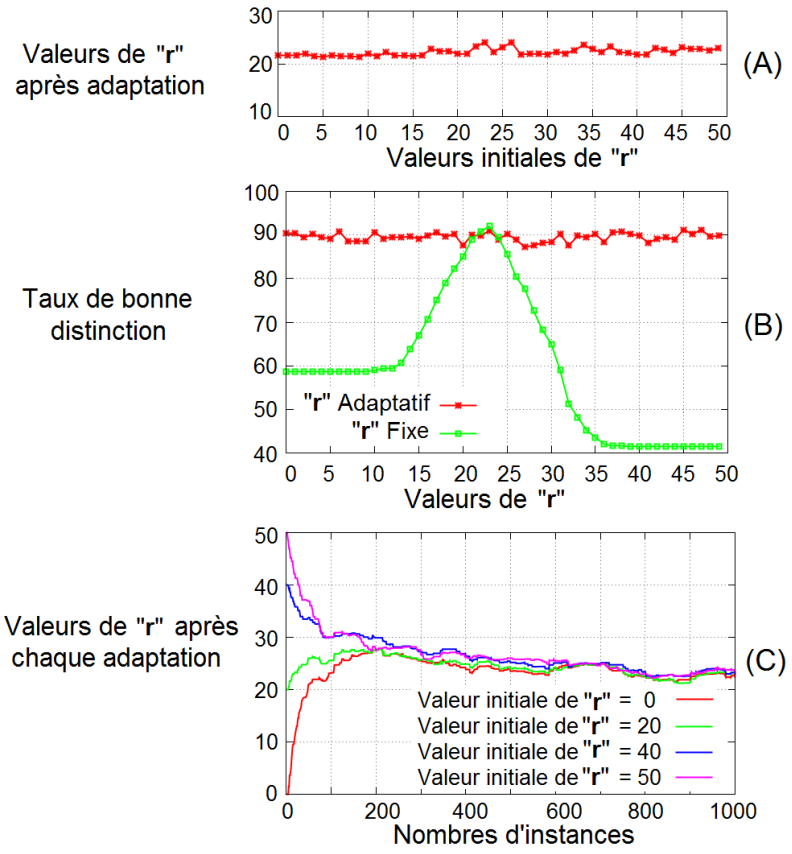



FIGURE 5.5 – Capacité à bien détecter les données étrangères/régulières en utilisant le seuil (rayon) adaptatif $r$, sur la base optdigits

selon le nombre de données observées à partir du flux et en choisissant différentes valeurs initiales de $r$. Nous pouvons en conclure que quelque soit sa valeur initiale, $r$ converge rapidement sans nécessiter de nombreuses données pour son adaptation. Cela prouve que la méthode est insensible à la valeur initiale de $r$ qui peut donc toujours être initialisée à $0$.

Un paramètre intuitif impliqué dans l'adaptation de $r$, est le taux d'apprentissage $\epsilon$. Afin de montrer l'effet de $\epsilon$ sur $r$, la Figure. 5.6 indique la valeur de $r$ selon le nombre de données observées à partir du flux, en utilisant différentes valeurs de $\epsilon$. Le taux d'apprentissage est généralement beaucoup plus petit que $1$ et légèrement supérieur à $0$ (c.à.d. $0 < \epsilon \ll 1$). Pour toutes les valeurs de $\epsilon$, $r$ se stabilise à terme, mais ceci arrive avec des vitesses différentes selon la valeur de $\epsilon$. En effet, quand $\epsilon$ est petit (par exemple $0.01$), $r$ change peu à chaque étape de l'adaptation. D'autre part, une grande valeur de $\epsilon$ (par exemple $0.3$), fait que $r$ change de façon plus importante à chaque étape de l'adaptation. Pour la suite des expérimentations, nous fixons $\epsilon$ à une valeur constante $\epsilon = 0.02$, intéressante pour quasiment toutes les bases de données.

### 5.4.2 Capacité à mieux distinguer les classes existantes et nouvelles

La deuxième série d'expérimentations permet de comprendre la capacité de la méthode proposée à détecter avec précision les instances des nouvelles classes et de les distinguer des instances de classes existantes, comme décrit dans la section 5.3.2.

La Figure. 5.7 (A) montre le temps moyen nécessaire pour la détection de nouvelles classes selon

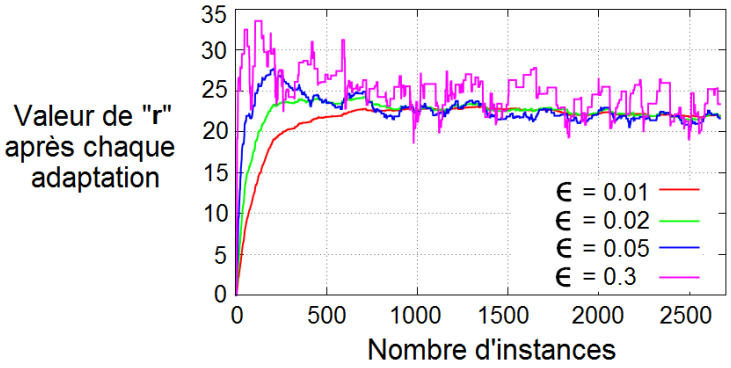



FIGURE 5.6 – L'effet de $\epsilon$ sur l'adaptation de $r$, sur la base optdigits

différentes valeurs de la taille du buffer (mini-lot) $|B|$. Ce temps croit linéairement avec la taille du buffer. La Figure. 5.7 (B) montre le taux de bonne distinction entre les instances de nouvelles classes et celles de classes existantes en fonction de la taille du buffer. Ce taux commence par croître rapidement puis se stabilise. Cela est dû au fait que la collecte d'un nombre suffisant de données permet de détecter plus précisément les instances étrangères qui sont similaires entre elles et confirme la présence d'une nouvelle classe formée par ces instances.

### 5.4.3 Comparaison

Nous comparons notre méthode de détection de nouvelles classes à une méthode de détection bien connue basée sur un SVM à une classe (nous utilisons l'implémentation en python disponible sur la plateforme scikit-learn [Pedregosa 11]).

Les deux méthodes sont utilisées avec A2ING comme classifieur de base afin de comparer les résultats définitifs de classification. Chaque jeu de données est organisé sous forme d'un flux dans lequel de nouvelles classes sont introduites progressivement.

Le Tableau 5.1 montre les résultats obtenus en termes de F-scores pour les classes nouvelles et existantes ($F_N$% and $F_E$%), le taux de bonne distinction entre les classes nouvelles et existantes ($Acc_1$ %), et le taux de reconnaissance (classification) final ($Acc_2$%). Le score $F_N$ est la moyenne harmonique de la précision $P_N$ et du rappel $R_N$. De même, $F_E$ est la moyenne harmonique de la précision $P_E$ et le rappel $R_E$. Ils sont exprimés comme suit :

$$F_N = \frac{2\times P_N \times R_N}{P_N + R_N} \quad \text{avec} \begin{cases} P_N = \frac{\text{true\_novel}}{\text{true\_novel} + \text{false\_novel}} \\ R_N = \frac{\text{true\_novel}}{\text{true\_novel} + \text{false\_exist}} \end{cases}$$

$$F_E = \frac{2\times P_E \times R_E}{P_E + R_E} \quad \text{avec} \begin{cases} P_E = \frac{\text{true\_exist}}{\text{true\_exist} + \text{false\_exist}} \\ R_E = \frac{\text{true\_exist}}{\text{true\_exist} + \text{false\_novel}} \end{cases}$$

où $\text{true\_novel}$ est le nombre d'instances qui sont correctement identifiées comme membres de nouvelles classes ; $\text{false\_novel}$ est le nombre d'instances de classes existantes qui sont identifiées à tort comme membres de nouvelles classes ; $\text{true\_exist}$ est le nombre d'instances qui sont correctement identifiées comme membres des classes existantes ; et $\text{false\_exist}$ est le nombre d'instances de nouvelles classes qui

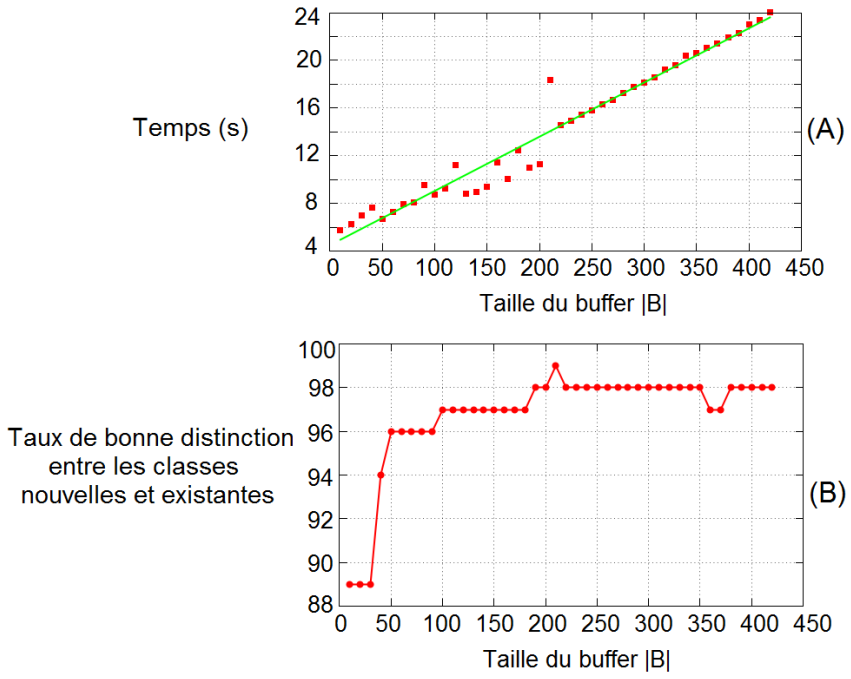



FIGURE 5.7 – Temps et taux de bonnes distinctions selon la taille du buffer $B$, sur la base optdigits

sont à tort identifiées comme membres des classes existantes. $Acc_1$ est exprimé comme

$$Acc_1 = \frac{\text{true\_novel} + \text{true\_exist}}{\text{true\_novel} + \text{false\_novel} + \text{true\_exist} + \text{false\_exist}}$$

Le taux de reconnaissance final $Acc_2$ représente le nombre d'instances qui sont correctement classées dans leurs classes correspondantes, sur le nombre total d'instances.

La méthode basée sur le SVM à une classe est considérée dans deux cas : dans le $1^{er}$ cas ("SVM Novelty $V_1$"), toutes les classes connues sont représentées par un modèle SVM. Dans le $2^{\text{ème}}$ cas ("SVM Novelty $V_2$"), les classes connues sont représentées par l'union de plusieurs modèles SVM, chacun d'entre eux étant entraîné sur une classe connue. Notre méthode est également considérée dans deux cas : avec un buffer (taille $|B| = 200$), et sans buffer ($|B| = 1$).

En ce qui concerne la distinction entre les classes nouvelles et existantes, nous pouvons voir à partir du tableau 5.1 ($F_N$, $F_E$, et $Acc_1$) que : (i) la méthode basée sur le SVM à une classe donne de meilleurs résultats dans le cas où plusieurs modèles SVM sont utilisés (SVM Novelty $V_2$), et (ii) notre méthode donne toujours de meilleurs résultats que celle basée sur SVM. De plus, les résultats sont meilleurs dans le cas des mini-lots (avec buffer). Ceci est dû au fait que la présence d'instances étrangères qui sont similaires entre elles dans le buffer permet de mieux confirmer la présence d'une nouvelle classe formée par ces dernières. En ce qui concerne le taux de reconnaissance final ($Acc_2$), le tableau 5.1 montre qu'il est généralement proportionnel au taux de bonne distinction entre les classes nouvelles et existantes ($Acc_1$). Cependant, dans certains cas, comme pour les bases "optdigit" et "documents", le taux de reconnaissance final est légèrement plus élevé quand "SVM Novelty $V_2$" est utilisé, par rapport à notre méthode sans buffer. Cela est dû au fait que le taux de reconnaissance dépend également de l'informativité des nouvelles instances manquées (non détectées).

Par ailleurs, comme indiqué dans le paragraphe précédent, $Acc_1$ est meilleur dans le cas où les instances arrivent par lots, plutôt qu'isolément. Si une instance est correctement détectée comme appartenant à une nouvelle classe, alors le modèle de classification est entrainé avec cette dernière et apprendra

à mieux classer des instances similaires à celle-ci. De ce fait, détecter correctement les instances appartenant à de nouvelles classes, permet d'améliorer les performances du classifieur. C'est pourquoi le taux de bonne distinction entre les classes nouvelles et existantes ($Acc_1$ %) est en général proportionnel au taux de reconnaissance ($Acc_2$%). Il est alors naturel que le taux de reconnaissance ($Acc_2$%) soit meilleur dans le cas où les instances arrivent par lots, plutôt qu'isolément.

TABLE 5.1 – Résultats comparatifs

| **Méthode** | $F_N$ **%** | $F_E$ **%** | $Acc_1$ **%** | $Acc_2$ **%** |
|---|---|---|---|---|
| *Base Optdigits* | | | | |
| SVM Novelty $V_1$ | 71.70 | 46.53 | 63.0 | 95.13 |
| SVM Novelty $V_2$ | 86.19 | 79.80 | 83.6 | *98.25* |
| Proposée (sans buffer) | *91.32* | *86.04* | *89.3* | 97.33 |
| Proposée (avec buffer) | **98.26** | **97.63** | **98.0** | **98.46** |
| *Base Pendigits* | | | | |
| SVM Novelty $V_1$ | 83.38 | 62.66 | 77.0 | 95.01 |
| SVM Novelty $V_2$ | 88.98 | 85.28 | 87.4 | 97.44 |
| Proposée (sans buffer) | *89.97* | *86.51* | *88.5* | *97.61* |
| Proposée (avec buffer) | **94.73** | **93.54** | **94.2** | **98.30** |
| *Base Letters-recognition* | | | | |
| SVM Novelty $V_1$ | 18.69 | 77.86 | 65.2 | 82.33 |
| SVM Novelty $V_2$ | 55.98 | *80.31* | 72.8 | 82.42 |
| Proposée (sans buffer) | *74.60* | 79.96 | *77.6* | *84.67* |
| Proposée (avec buffer) | **87.21** | **90.53** | **86.8** | **85.40** |
| *Base de Documents* | | | | |
| SVM Novelty $V_1$ | 29.52 | 89.38 | 81.0 | 88.47 |
| SVM Novelty $V_2$ | 24.69 | *88.24* | 79.66 | *88.91* |
| Proposée (sans buffer) | *59.25* | 88.17 | *81.66* | 88.69 |
| Proposée (avec buffer) | **90.84** | **91.14** | **91.0** | **90.08** |
| *Résultats moyens sur toutes les bases* | | | | |
| SVM Novelty $V_1$ | 50.82 | 69.10 | 71.55 | 90.23 |
| SVM Novelty $V_2$ | 63.96 | 83.40 | 80.86 | 91.75 |
| Proposée (sans buffer) | *78.78* | *85.17* | *84.26* | *92.07* |
| Proposée (avec buffer) | **92.76** | **93.21** | **92.5** | **93.06** |

## 5.5 Conclusion

Dans ce chapitre, nous avons abordé le problème de la détection de nouvelles classes pour la classification des flux de données. Notre méthode diffère des méthodes existantes selon plusieurs aspects. Tout d'abord, elle maintient de façon incrémentale un modèle multi-classe des classes connues, tout en étant capable de détecter l'émergence de nouvelles classes dans un flux de données de longueur infinie. Deuxièmement, elle demande seulement l'étiquetage des instances informatives appartenant aux classes nouvelles et existantes. Troisièmement, elle ne fait aucune supposition sur la distribution des données. Enfin, elle est adaptative.

Par ailleurs, un problème qui est souvent omis dans la littérature sur l'apprentissage actif concerne les erreurs d'étiquetage. En vérité, nous n'avons pas toujours la garantie d'obtenir une étiquette parfaite-

ment fiable lorsqu'on la demande à un opérateur humain. Des erreurs d'étiquetage peuvent se produire pour plusieurs raisons : erreurs accidentelles ou d'inattention, la subjectivité de classes, l'expertise de l'opérateur, etc. Ces erreurs d'étiquetage ont non seulement un impact sur les performances de la classification mais aussi sur la sélection des instances informatives. Nous abordons ce problème dans le chapitre suivant.

# Chapitre 6

# Gestion des erreurs d'étiquetage

**Sommaire**



## 6.1 Introduction

Comme nous l'avons montré dans le Chapitre 3, l'apprentissage actif réduit le coût d'étiquetage manuel, en demandant à un opérateur humain seulement les étiquettes de classes pour les données qui sont informatives (généralement les données incertaines). Nous avons vu que les méthodes d'apprentissage actif donnent de très bons résultats pour la classification des flux de données. La plupart de ces méthodes supposent que les étiquettes demandées (données par un opérateur) sont parfaitement correctes. Cependant, dans la pratique, cette hypothèse n'est souvent pas satisfaite. En effet, il est difficile d'obtenir des étiquettes totalement fiables. L'opérateur est sujet à des erreurs d'étiquetage qui peuvent se produire et cela pour plusieurs raisons : l'inattention ou les erreurs accidentelles, les connaissances incertaines du domaine, la subjectivité des classes, etc.

Comme il a été expliqué dans le chapitre 3, l'apprentissage actif demande l'étiquetage des instances incertaines. Ces instances sont susceptibles d'améliorer le modèle de classification si l'on suppose que

les étiquettes de classe qui sont demandées sont correctes. Ainsi, l'apprentissage actif vise à chercher des instances qui réduisent son incertitude. Toutefois, les erreurs d'étiquetage conduisent l'algorithme à concentrer de manière incorrecte la recherche sur les régions pauvres (ne contenant pas des instances informatives) de l'espace des entrées. Cela représente une difficulté supplémentaire pour l'apprentissage actif pour réduire la complexité de l'étiquetage [Dasgupta 05]. Si les erreurs d'étiquetage ne sont pas détectées et atténuées, l'apprenant actif peut facilement échouer à converger vers un bon modèle. Par conséquent, les erreurs d'étiquetage sont nocives pour l'apprentissage actif et leur gestion est donc une tâche importante.

La détection des erreurs d'étiquetage n'est pas triviale dans le cadre d'un apprentissage actif à partir d'un flux, principalement pour deux raisons. D'une part, dans le contexte d'un flux de données, la décision de filtrer ou non une instance, potentiellement mal étiquetée, doit être prise immédiatement. D'autre part, vu que l'apprentissage est actif, les instances mal étiquetées sont celles dont le classifieur est incertain de leur classe.

Dans la méthode que nous proposons, nous effectuons un apprentissage actif à partir d'un flux de données en présence d'erreurs d'étiquetage. La principale question que nous abordons est la suivante : parmi les instances dont les étiquettes ont été demandées, lesquelles sont mal étiquetées et lesquelles nécessitent un ré-étiquetage ? Nous proposons une mesure qui reflète la vraisemblance de mauvais étiquetage pour filtrer dès réception toutes les instances potentiellement mal étiquetées. Un opérateur expert alternatif peut ainsi être sollicité pour corriger l'instance filtrée. La méthode est capable de sélectionner (pour le ré-étiquetage) seulement les instances qui méritent d'être corrigées selon une nouvelle mesure d'informativité.

## 6.2 État de l'art

Les erreurs d'étiquetage représentent un problème important pour la classification, avec des conséquences négatives. De nombreux travaux dans la littérature ont été consacrés à l'étude des erreurs d'étiquetage et au développement de techniques pour y remédier. Dans cette section, nous donnons la définition des erreurs d'étiquetage, les potentielles sources de ces erreurs, leurs types et leurs conséquences sur les classifieurs. Ensuite, nous présentons des méthodes conçues pour l'apprentissage en présence de données mal étiquetées.

Les erreurs d'étiquetage que nous prenons en considération représentent des étiquettes observées jugées incorrectes. Nous supposons qu'aucune autre information n'est disponible, contrairement à d'autres contextes où l'opérateur peut par exemple fournir un indice de confiance ou d'incertitude sur son étiquetage.

### 6.2.1 Apprendre à partir de données mal étiquetées

Quelle que soit la méthode d'apprentissage utilisée, l'approche usuelle consiste à produire un classifieur à partir d'un ensemble de données étiquetées où les étiquettes de classes guident l'algorithme d'apprentissage vers la règle de classification désirée. L'étiquette associée à une donnée d'apprentissage correspond généralement à la vraie classe de cette donnée. Cependant, l'étiquetage, tel que présenté dans [Angluin 88], peut être soumis à un processus de "bruit" avant d'être présenté à l'algorithme d'apprentissage. Les données du monde réel peuvent être exposées à des erreurs d'étiquetage qui sont définies dans [Hickey 96] comme étant tout ce qui obscurcit la relation entre les caractéristiques d'une donnée et sa classe. Il est donc important de distinguer la véritable étiquette de classe d'une instance de l'étiquette observée.

Les erreurs d'étiquetage deviennent de plus en plus courantes dans les tâches de classification. Ceci

est dû au fait que les données étiquetées de manière fiable, sont souvent coûteuses et difficiles à obtenir. D'autant plus que les tâches impliquant un étiquetage deviennent de plus en plus complexe. Par exemple, dans des applications d'analyse de documents administratifs hétérogènes, les classes de documents ne sont pas toujours faciles à distinguer. Dans des applications médicales, les tests de diagnostic médical ne sont pas parfaits et atteignent rarement une précision de 100%, comme c'est indiqué dans [Bross 54, Joseph 95, Hadgu 96]. De nombreux travaux tels que [Zhu 04, Krishnan 90, Krishnan 01, Yasui 04, Malossini 06] ont montré que les erreurs d'étiquetage affectent négativement les performances de classification des classifieurs existants ; ce qui implique la nécessité de mettre en œuvre des techniques qui éliminent l'effet des erreurs d'étiquetage ou réduisent leurs conséquences. Dans certains travaux, les erreurs d'étiquetage sont aussi appelées "bruit d'étiquettes" ("label noise" en anglais). Dans [Quinlan 86] et [Zhu 04], deux types de bruits sont distingués :

- **Bruit de caractéristiques** : affecte les valeurs d'une caractéristique pendant leur extraction.
- **Bruit d'étiquettes** : affecte les étiquettes de classes qui sont associées aux instances.

Dans [Zhu 04] et [Saez 14], il est montré que le bruit d'étiquettes est potentiellement plus nocif que le bruit de caractéristiques, ce qui souligne l'importance de prendre en compte ce type de bruit. Ceci est expliqué principalement par deux raisons :

1. Pour une instance donnée, il existe de nombreuses caractéristiques alors qu'il n'y a qu'une seule étiquette de classe associée à cette instance.
2. L'importance des caractéristiques est variable alors que celles des étiquettes de classes est toujours grande car les étiquettes guident l'apprentissage vers la règle de classification désirée.

Des constats similaires sont faits dans [Quinlan 86] où le bruit de caractéristiques semble être moins nocif que le bruit d'étiquettes (erreurs d'étiquetage) pour les arbres de décision, sauf dans le cas où un grand nombre de caractéristiques sont affectées par le bruit de caractéristiques.

### 6.2.2 Causes des erreurs d'étiquetage

Selon l'auteur de [Hickey 96], les causes des erreurs d'étiquetage ne sont pas nécessairement aussi importantes que les conséquences de ces erreurs. Cependant, elles peuvent aider à mieux comprendre les différents types d'erreurs et peuvent être utiles dans certains cas pour modéliser le processus qui génère les erreurs d'étiquetages. Par exemple dans [Fang 13], les erreurs sont dues aux connaissances incertaines de l'opérateur concernant les instances de certaines classes, ce qui a permis aux auteurs de modéliser explicitement les connaissances de l'opérateur afin de ne lui demander que l'étiquetage de ce qu'il connait. Néanmoins, les causes des erreurs d'étiquetage sont diverses et il est souvent impossible de connaître avec précision la source d'une erreur. Les erreurs d'étiquetage sont humaines et peuvent avoir plusieurs causes potentielles :

1. La nature subjective des classes impliquées dans la tâche d'étiquetage. Par exemple, comme il est indiqué dans [Smyth 94] pour l'analyse de données images, il peut exister une variabilité importante dans l'étiquetage par plusieurs experts.
2. D'après l'auteur dans [Hickey 96], l'information qui est fournie à l'opérateur peut être de mauvaise qualité ou insuffisante pour déterminer la véritable étiquette d'une instance donnée. L'utilisateur peut donc juste supposer ou deviner la classe qu'il pense être convenable pour cette instance.
3. L'opérateur a des connaissances incertaines du domaine. En effet, vu que la collecte des étiquettes fiables est une tâche fastidieuse et coûteuse, il y a un intérêt croissant pour l'utilisation d'opérateurs non experts pour un étiquetage facile et pas cher, tel que dans [Snow 08, Ipeirotis 10, Raykar 10, Yuen 11], en utilisant des frameworks tels que "Amazon Mechanical Turk". Les étiquettes fournies par des opérateurs non experts sont moins fiables.

4. Des erreurs intentionnelles telles que celles considérées dans [Servedio 03, Biggio 11, Xiao 12]. Par exemple, dans les tâches de filtrage de mails, un utilisateur mal intentionné peut introduire des erreurs d'étiquetage volontairement pour tromper le classifieur de spam.
5. L'incompréhension du mécanisme utilisé pour effectuer l'étiquetage, ou tout simplement le manque d'attention (e.g. des clics accidentels), ou même des erreurs de perception dues à la fatigue.

### 6.2.3 Types des erreurs d'étiquetage

Selon les auteurs dans [Sloan 95, Angluin 88(2)], les erreurs d'étiquetage peuvent être classées en deux grandes catégories : les erreurs aléatoires et les erreurs non aléatoires, tel que décrit ci-après. Considérons les notations suivantes : $x$ est une instance, $y$ est la classe de $x$, $y_o$ est l'étiquette observée. $y \neq y_o$ indique donc une erreur d'étiquetage.

#### 6.2.3.1 Erreurs aléatoires

Les erreurs d'étiquetage aléatoires se produisent indépendamment de la donnée en entrée. Ainsi, l'étiquette erronée est susceptible de concerner n'importe quelle instance. Autrement dit, la survenance d'une erreur est indépendante de $x$. Dans ce cas, l'étiquette observée $y_o$ est différente de $y$ la vraie classe avec une probabilité $p_e = P(y \neq y_o)$, qui représente le taux d'erreurs. Afin de simuler des erreurs de ce type, l'auteur dans [Aslam 96] décide aléatoirement si l'étiquette observée $y_o$ est correcte ou non, puis, l'étiquette erronée est choisie au hasard dans l'ensemble $Y - \{y\}$, où $Y$ est l'ensemble de toutes les étiquettes de classes possibles.

La plupart des travaux considèrent que les erreurs d'étiquetage peuvent affecter toutes les instances sans distinction. Ces erreurs aléatoires sont celles qu'on rencontre le plus souvent et peuvent survenir par exemple à cause de l'inattention, la fatigue ou à cause d'un utilisateur qui souhaite augmenter sa productivité en donnant simplement l'étiquette sans analyser attentivement les données.

#### 6.2.3.2 Erreurs non aléatoires

Contrairement aux erreurs aléatoires, les erreurs non-aléatoires dépendent de la donnée en entrée, et cette dernière a une influence sur l'attribution de l'étiquette par l'opérateur. Autrement dit, vu que la survenance d'une erreur dépend de $x$, les erreurs d'étiquetage sont plus probables dans certaines régions de l'espace d'entrée $X$. Ainsi, l'étiquette observée $y_o$ est différente de la vraie étiquette $y$ de $x$ avec une probabilité $p_e(x) = P(y \neq y_o|x)$, qui dépend de la donnée $x$.

Selon les auteurs dans [Kolcz 09, Takenouchi 08], les instances peuvent être plus susceptibles d'être mal étiquetées quand elles sont similaires aux instances d'autres classes. Ainsi, les erreurs non aléatoires ont tendance à apparaître plus près de la limite de décision, là où les instances sont plus difficiles à classer. Les étiquettes sont aussi moins fiables dans les régions de faible densité, où l'étiquetage des opérateurs est basé sur un très petit nombre d'instances similaires rencontrées précédemment. Dans le contexte de la classification de données textuelles, une preuve empirique est donnée dans [Klebanov 10] montrant que plusieurs instances difficiles à classer (proches de la limite de décision ou dans les régions à faible densité) sont mal étiquetées. Ainsi, afin de simuler des erreurs de ce type, l'instance à laquelle on assigne une étiquette erronée est choisie selon sa proximité aux régions d'incertitude.

Du point de vue applicatif, les erreurs aléatoires peuvent être considérées comme étant plus générique car dans ce cas, on fait moins d'hypothèses sur la nature des erreurs. Une erreur non aléatoire, dépend plus de l'application et de la connaissance a priori sur la nature de l'erreur peut être utile pour la traiter plus efficacement.

Les erreurs non aléatoires traitées dans [Servedio 03, Biggio 11, Xiao 12] sont des erreurs intentionnelles et malicieuses dont le but est de tromper le fonctionnement du classifieur. Nous ne traitons pas ces cas d'erreurs intentionnelles dans le contexte de cette thèse.

#### 6.2.3.3 Erreurs symétriques et non symétriques

Selon certains auteurs tels que dans [Lugosi 92], que les erreurs soient aléatoires ou non aléatoires, il est possible de distinguer encore deux sous-catégories : les erreurs symétriques et asymétriques. Les erreurs de type symétrique se produisent de manière uniforme pour toutes les classes. Autrement dit, la probabilité qu'une classe soit perturbée (contienne des instances mal étiquetées) est la même pour toutes les classes. Par contre, les erreurs de type asymétrique peuvent se produire dans une classe plus qu'une autre. On peut distinguer les quatre cas suivants :

1. *Erreurs d'étiquetage aléatoires symétriques* : les erreurs d'étiquetage sont équiprobables pour toutes les instances de toutes les classes. Dans ce cas, l'erreur d'étiquetage est indépendante de $x$ (l'instance en entrée) et de $y$ (la vraie classe de $x$).
2. *Erreurs d'étiquetage aléatoires asymétriques* : les erreurs d'étiquetage peuvent concerner n'importe quelle instance mais sont plus probables pour certaines classes. Dans ce cas, l'erreur d'étiquetage dépend de $y$.
3. *Erreurs d'étiquetage non aléatoires symétriques* : les erreurs d'étiquetage sont plus probables dans certaines régions de l'espace d'entrée $X$ et peuvent concerner n'importe quelle classe. Dans ce cas, l'erreur d'étiquetage dépend de $x$.
4. *Erreurs d'étiquetage non aléatoires asymétriques* : les erreurs d'étiquetage sont plus probables dans certaines régions de l'espace d'entrée $X$ et pour certaines classes. Dans ce cas, l'erreur d'étiquetage dépend de $x$ et de $y$.

Dans le cas d'absence d'informations et d'hypothèses a priori sur la nature des erreurs, le cas le plus général est souvent considéré, à savoir, que les erreurs peuvent affecter n'importe quelle classe sans distinction (erreurs symétriques). Toutefois, il est illustré dans [Lugosi 92] que, si on suppose que les classes sont distribuées symétriquement dans l'espace des données, alors les erreurs de type symétrique affectent moins les résultats, mais cette supposition est souvent incorrecte dans les données du monde réel.

Le type exact des erreurs d'étiquetage est souvent inconnu, car la cause des erreurs d'étiquetage est difficilement identifiable.

### 6.2.4 Types des erreurs dans le cas d'un apprentissage actif

Dans le cas d'un apprentissage actif, les erreurs d'étiquetage peuvent affecter n'importe quelle instance parmi celles qui sont manuellement étiquetées. Cependant, l'apprenant actif demande l'étiquetage des instances incertaines seulement. Par conséquent, parmi toutes les instances disponibles (étiquetées ou non), celles qui sont les plus susceptibles d'être mal étiquetées sont les plus incertaines. On peut donc en déduire que dans le cas d'un apprentissage actif, une erreur d'étiquetage est naturellement de type *non aléatoire*, vu qu'elle dépend forcément de l'instance en entrée $x$ (cf. paragraphe 6.2.3.2)

### 6.2.5 Solutions existantes au problème

Il existe un nombre croissant de travaux de recherche qui visent à résoudre les problèmes liés à l'apprentissage à partir de données associées à des étiquettes de classes qui peuvent être erronées.

#### 6.2.5.1 Méthodes de filtrage des erreurs

La plupart des approches existantes effectuent un pré-traitement pour éliminer ou ré-étiqueter toutes les données suspectées d'être mal étiquetées [Brodley 99, Sanchez 03, Barandela 00, Muhlenbach 04, Jiang 04]. Elles essaient de détecter les instances mal étiquetées en utilisant diverses heuristiques qui ont montré leur efficacité contre les erreurs d'étiquetage. Les approches de filtrage sont faciles à mettre en œuvre, mais présentent parfois le risque d'éliminer une quantité assez importante de données utiles, ce qui peut influencer négativement la performance de classification, surtout lorsque le nombre de données d'apprentissage est limité.

La détection des instances mal étiquetées n'est pas triviale. De nombreuses méthodes ont été proposées pour "nettoyer" des bases de données d'apprentissage, avec différents degrés de succès. Nous décrivons ci-après plusieurs méthodes qui permettent de détecter, supprimer, ou ré-étiqueter des instances mal étiquetées. Ces méthodes se basent sur des mesures de confiance associées à des seuils, la prédiction du classifieur, les voisins les plus proches, ou la combinaison de plusieurs classifieurs.

**Filtrage basé sur des mesures de confiance et des seuils :** Plusieurs méthodes de filtrage des erreurs d'étiquetage se basent sur des mesures dites d'anomalie. Dans ce cas, les instances peuvent être supprimées lorsque la mesure dépasse un seuil prédéfini. Par exemple, dans [Sun 07], l'anomalie est vue au travers de la mesure d'entropie de la distribution de probabilité $P(Y|X)$ estimée en utilisant un classifieur probabiliste. Les instances, avec une faible entropie, correspondent à une classification confiante. Par conséquent, ces instances (confiantes) pour lesquelles le classifieur est en désaccord avec l'étiquette observée sont ré-étiquetées en utilisant l'étiquette prédite. Cependant, de telles instances confiantes sont typiquement celles qui sont loin de la limite de décision. Or, les erreurs d'étiquetage peuvent apparaître n'importe où dans l'espace d'entrée ou, lorsqu'il s'agit d'erreurs non aléatoires (ex. dans le cas d'un apprentissage actif) forcément près de la limite de décision. De plus, ces instances sont celles dont le classifieur est incertain, donc leur étiquette prédite peut être erronée et ne doit pas être utilisée pour le ré-étiquetage de ces instances.

**Filtrage basé sur la prédiction du classifieur :** Les prédictions des classifieurs peuvent être utilisées pour détecter les instances mal étiquetées. Par exemple, dans [Thongkam 08], la méthode entraîne un SVM en utilisant des données d'apprentissage puis supprime toutes les instances qui sont étiquetées différemment de l'étiquette prédite par le SVM. Cette méthode peut être appliquée à tout classifieur de base. Par exemple, dans [Jeatrakul 10], la même méthode est utilisée avec un réseau de neurones. Dans [Miranda 09], un ensemble de classifieurs est utilisé en faisant un vote sur les prédictions des différents classifieurs. Ces méthodes éliminent toutes les instances qui se trouvent du mauvais côté de la frontière de classification, ce qui peut altérer les performances [Guyon 96]. Par ailleurs, les instances qui sont incorrectement classées par le classifieur, ne sont pas forcément des instances mal étiquetées. Elles peuvent constituer des erreurs de prédiction tout à fait normales ou provenir des faibles performances du classifieur s'il a été entraîné sur un ensemble de données contenant des erreurs d'étiquetage.

**Filtrage basé sur un ensemble de classifieurs :** Les méthodes précédentes étant confrontées au risque d'écarter beaucoup d'instances, des ensembles de classificateurs sont utilisés dans [Brodley 99, Brodley 96, Brodley 96(2)] pour éviter ce problème. L'objectif est d'apprendre $m$ différents classifieurs en utilisant un sous-ensemble aléatoire de données et de les utiliser pour classer les données restantes. Puis, pour chaque donnée classée, on décide si elle est mal étiquetée ou non. Deux possibilités sont étudiées dans [Brodley 99, Brodley 96, Brodley 96(2)] : un "vote majoritaire" et un "vote par consensus". Le vote majoritaire classe une instance comme étant mal étiquetée si une majorité des $m$ classifieurs la classent de façon erronée. Le vote par consensus, quant à lui, exige que tous les classifieurs classent l'instance de

façon erronée, pour qu'elle soit considérée comme mal étiquetée. Ce processus est répété plusieurs fois jusqu'à ce que aucune instance ne soit considérée comme mal étiquetée. Les méthodes de ce type ne sont pas adaptées à un flux de données car elles ont besoin d'entraîner un ensemble de classifieurs plusieurs fois sur un ensemble de données disponibles à l'avance.

Une autre méthode proposée dans [Zhu 03 (2)], partitionne l'ensemble de données en petits sous-ensembles et apprend un classifieur sur chaque sous-ensemble. L'ensemble des classifieurs peut être utilisé pour filtrer l'ensemble des données en utilisant un vote majoritaire ou par consensus. La méthode décrite dans [Venkataraman 04] utilise également un ensemble de classifieurs. Chaque classifieur de l'ensemble est obtenu en entraînant un SVM sur un sous-ensemble différent de caractéristiques. Les auteurs affirment que la méthode est efficace contre les erreurs d'étiquetage. Cependant, le problème avec cette approche, est que les classifieurs de l'ensemble sont construits à partir des données d'apprentissage qui contiennent toujours des erreurs d'étiquetage.

**Filtrage basé sur les plus proches voisins :** Plusieurs méthodes de filtrage des erreurs d'étiquetage basées sur les plus proches voisins, telles que celles présentées dans [Wilson 00, Wilson 97], existent dans la littérature. La méthode proposée dans [Koplowitz 81] classe une instance $x$ en utilisant $kNN$, puis vérifie s'il y au moins $k'$ instances appartenant à la classe prédite parmi les $k$ plus proches voisins de $x$. Si tel est le cas, l'instance est ré-étiquetée avec la classe prédite, sinon elle est simplement écartée de la base d'apprentissage. Cette heuristique vise à garder seulement les instances dont l'étiquette apparaît fortement dans leur voisinage. Dans [Barandela 00], une méthode similaire à la précédente est introduite pour écarter de façon itérative les instances dont l'étiquette de classe est en désaccord avec la classe de la plupart de leurs voisins les plus proches, si le désaccord dépasse un seuil prédéfini. Il est principalement montré que la répétition de la méthode proposée dans [Koplowitz 81], un certain nombre de fois, améliore les résultats en présence d'erreurs d'étiquetage. Ces techniques supposent que l'étiquette d'une instance mal étiquetée a tendance à être en désaccord avec l'étiquette d'autres instances dans son voisinage immédiat. L'avantage de cette approche est sa robustesse à la forme de la classe car elle ne fait aucune hypothèse sur la distribution des données. Cependant, la détection des erreurs d'étiquetage est effectuée localement en supposant que la plupart des voisins d'une instance mal étiquetée sont étiquetés correctement.

Il existe encore d'autres méthodes pour le filtrage des données mal étiquetées. Par exemple, dans [Hughes 04], étant donnée une base de données étiquetées, les étiquettes des instances (et non les instances elles-mêmes) qui sont proches des frontières de classification sont supprimées, puisque l'opérateur (qui effectue l'étiquetage) peut être moins fiable dans cette région (erreurs non aléatoires). Puis, un apprentissage semi-supervisé est effectué en utilisant à la fois les instances étiquetées et les instances non étiquetées (auxquelles on a retiré les étiquettes). Cependant, les instances dépourvues de leurs étiquettes sont justement celles pour lesquelles un étiquetage manuel est demandé dans le cas d'un apprentissage actif, et ce sont les seules instances étiquetées dont on dispose.

Dans plusieurs travaux telles que [Miranda 09, Cuendet 08], il a été observé que la simple élimination des instances mal étiquetées est aussi efficace que leur ré-étiquetage. Cependant, tel que souligné par Teng dans [Teng 00, Teng 01, Teng 05], les méthodes de filtrage peuvent éliminer beaucoup d'instances (sur-filtrage), surtout lorsqu'il s'agit de classes minoritaires. En effet, les instances des classes minoritaires sont plus susceptibles d'être éliminées (car elles sont aussi plus susceptibles d'être incorrectement classées), ce qui rend l'apprentissage, selon [Seiffert 07], encore plus difficile. D'une part, il est montré dans [Matic 92] que le sur-filtrage peut réduire les performances des classifieurs ; et d'autre part, il est suggéré dans [Brodley 99] que garder des instances mal étiquetées peut nuire au classifieur

plus que d'enlever trop d'instances correctement étiquetées. Par conséquent, un compromis doit être trouvé entre l'élimination de "beaucoup d'instances mal étiquetées" et de "peu d'instances correctement étiquetées". Il n'est pas toujours trivial de réaliser ce compromis dans l'absence d'informations sur les erreurs d'étiquetage (par exemple, le taux des erreurs et leur type). Au lieu d'écarter les instances qui sont susceptibles d'être mal étiquetées, une autre solution serait de les faire ré-étiqueter par un expert, comme proposé dans [Matic 92, Rebbapragada 12]. Toutefois, selon les applications, ceci peut se révéler trop coûteux.

#### 6.2.5.2 Méthodes basées sur un apprentissage semi-supervisé

Un algorithme d'apprentissage semi-supervisé en présence d'erreurs d'étiquetage est proposé dans [Breve 10]. L'ensemble de données est converti en un graphe, où les instances sont des nœuds (étiquetés et non étiquetés) avec des arêtes entre les instances similaires. Chaque nœud étiqueté diffuse sont étiquette à travers le graphe et coopère avec les nœuds étiquetés de la même façon dans son voisinage, afin d'étiqueter les instances non étiquetées. Le comportement intéressant dans cette méthode est que les étiquettes des instances mal étiquetées sont dominées par les instances proches qui sont étiquetées différemment, ce qui empêche une instance mal étiquetée d'influencer l'étiquetage des instances non étiquetées proches. Dans [Bruzzone 11] et [Guan 11], les instances étiquetées sont d'abord utilisées pour étiqueter les instances non étiquetées à l'aide d'un algorithme d'apprentissage semi-supervisé, puis les nouvelles étiquettes sont utilisées pour filtrer les instances mal étiquetées.

#### 6.2.5.3 Méthodes basées sur la détection de données aberrantes

Traiter les erreurs d'étiquetage est étroitement lié à la détection de données aberrantes et à la détection d'anomalies. En effet, une instance mal étiquetée peut apparaître comme une instance aberrante par rapport à la classe qui correspond à son étiquette incorrecte, ou si son étiquette a une faible probabilité d'occurrence dans son voisinage. Par conséquent, de nombreuses méthodes comme celles décrites dans [Xiong 06] et [Lukashevich 09] sont développées pour gérer les données aberrantes sont aussi utilisées pour gérer les erreurs d'étiquetage. Cependant, comme indiqué dans [Collett 76], il est important de souligner que les données mal étiquetées ne sont pas nécessairement des données aberrantes ou des anomalies. De même, il est montré dans [Liu 02] qu'une donnée aberrante n'est pas nécessairement une instance mal étiquetée, car elle peut provenir du bruit des caractéristiques ou tout simplement d'un événement à faible probabilité.

#### 6.2.5.4 Cas de flux de données et d'apprentissage actif

La plupart des méthodes présentes dans la littérature pour gérer les erreurs d'étiquetage supposent que l'ensemble de données est disponible préalablement et essaient de le nettoyer avant que l'apprentissage n'ait lieu en enlevant, de façon répétitive, les instances les plus susceptibles d'être mal étiquetées parmi toutes les instances de l'ensemble de données. Toutefois, il existe quelques méthodes qui permettent un apprentissage à partir de flux de données en présence d'erreurs d'étiquetage. Une méthode proposée dans [Zhu 08] est conçue pour le nettoyage des flux de données, en enlevant les instances potentiellement mal étiquetées à partir d'un flux de données étiquetées. Cependant, la méthode ne considère pas un cadre d'apprentissage actif où les erreurs d'étiquetage concernent des instances incertaines. De plus, la méthode divise le flux de données en gros morceaux et essaie de détecter les instances mal étiquetées dans chaque morceau, ce qui rend la méthode partiellement adaptée au flux de données.

En ce qui concerne l'apprentissage actif, des méthodes comme dans [Fang 13, Tuia 13] prennent en compte uniquement les erreurs d'étiquetage liées aux connaissances incertaines de l'opérateur. D'une

manière générale, ces méthodes reposent sur le principe qu'il vaut mieux prévenir que guérir. En effet, elles essaient de modéliser la connaissance de l'opérateur et évitent de demander l'étiquette d'une instance si elle appartient à l'ensemble des connaissances incertaines de l'opérateur. Toutefois, ceci peut entraîner la perte de nombreuses données informatives. De plus, ces méthodes supposent implicitement que l'opérateur est toujours le même, vu que sa connaissance est modélisée.

Une autre méthode proposée dans [Rebbapragada 12] considère une correction active des étiquettes, mais l'apprentissage lui-même n'est pas actif. La méthode sélectionne de façon itérative les $k$ instances susceptibles d'être les plus mal étiquetées à partir d'un ensemble de données étiquetées et les présentent à un expert pour la correction plutôt que de les écarter.

Des méthodes telles que celles présentées dans [Sheng 08, Ipeirotis 14, Yan 11] peuvent être appliquées à l'apprentissage actif, mais elles essaient de réduire l'effet des erreurs différemment : plutôt que d'essayer de détecter les instances qui sont éventuellement mal étiquetées, elles demandent l'étiquette d'une instance à plusieurs opérateurs en utilisant des techniques de "crowd-sourcing". Ces méthodes nécessitent de multiples opérateurs qui peuvent fournir des étiquettes redondantes pour chaque instance incertaine et ne sont pas destinées à être utilisées avec un seul opérateur alternatif.

## 6.3 Impact des erreurs d'étiquetage sur l'apprentissage actif

Dans cette section, nous montrons comment les erreurs d'étiquetage impactent l'apprentissage actif.

Comme mentionné précédemment, nous n'avons pas la certitude d'obtenir une étiquette parfaitement fiable lorsqu'on la demande à un opérateur humain. Nous considérons un processus d'erreurs d'étiquetage dans lequel l'opérateur non fiable a une probabilité $\sigma$ de donner une mauvaise réponse et $1 - \sigma$ de donner la bonne réponse, chaque fois qu'une étiquette de classe lui est demandée pour une instance incertaine.

La Figure. 6.1 montre les résultats obtenus en utilisant un algorithme d'apprentissage actif séquentiel en présence d'erreurs d'étiquetage avec différentes intensités de $\sigma$ et par rapport au cas sans erreurs $\sigma = 0$. La Figure. 6.1 (A) montre le taux de reconnaissance du modèle $h$ sur une base de test, selon le nombre d'instances reçues à partir du flux. Comme pour les méthodes d'apprentissage supervisé usuelles, il n'est pas surprenant de voir que pour l'apprentissage actif, les erreurs d'étiquetage réduisent le taux de reconnaissance. La Figure. 6.1 (B) représente le taux de reconnaissance en fonction du nombre d'étiquettes demandées (instances étiquetées manuellement). Nous pouvons constater que, en plus de dégrader le taux de reconnaissance, les erreurs d'étiquetage entraînent un accroissement du nombre de demandes d'étiquetage. Ceci est confirmé par la Figure. 6.1 (C) qui montre le nombre d'instances dont l'étiquette est demandée en fonction du nombre d'instances observées dans le flux. Ceci s'explique par le fait que l'instance la plus incertaine peut être informative si nous obtenons sa vraie étiquette de classe, mais peut facilement devenir celle qui induit le plus en erreur si elle est mal étiquetée. Par conséquent, les instances mal étiquetées font que l'apprentissage actif se concentre à tort sur les régions pauvres de l'espace des entrées, et s'écarte des instances qui sont vraiment informatives.

En résumé, l'apprentissage actif à partir d'un flux est très sensible aux erreurs d'étiquetage. Ces erreurs ont non seulement une incidence sur les capacités prédictives du modèle appris, mais conduisent aussi à demander les étiquettes des instances qui ne sont pas nécessairement informatives. Ceci provoque une demande plus élevée d'instances étiquetées et représente un obstacle pour la minimisation de la complexité de l'apprentissage actif en terme de nombre d'étiquettes demandées [Dasgupta 05].

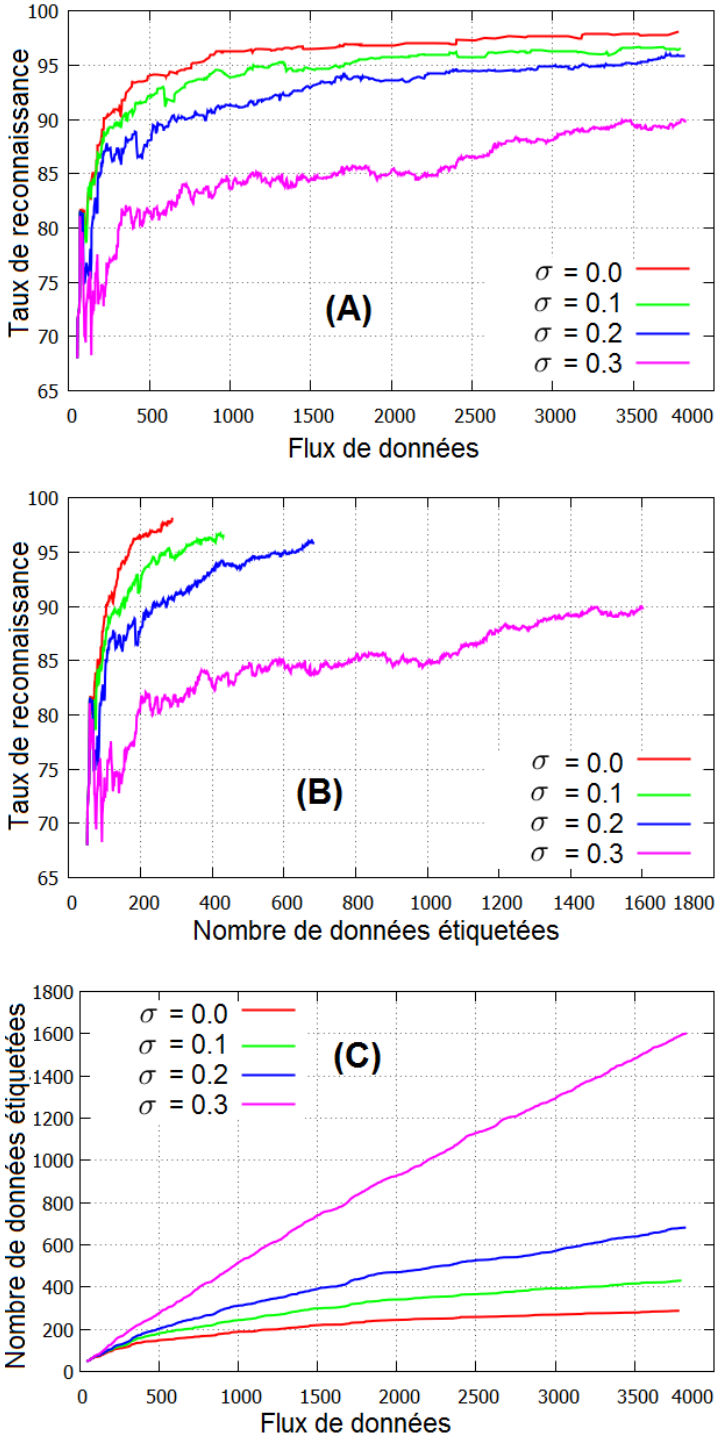



FIGURE 6.1 – Un apprentissage actif à partir d'un flux de données (base optdigits) avec différents niveaux d'intensité d'erreurs ($\sigma$)

## 6.4 Caractérisation des instances mal étiquetées

Dans cette section, nous proposons des mesures pour caractériser les instances mal étiquetées et leur importance. Tout d'abord, nous présentons un coefficient de désaccord qui reflète la probabilité qu'une instance soit mal étiquetée. Puis, nous dérivons une mesure d'informativité qui reflète l'importance de l'instance, qui est ensuite utilisée pour déterminer si l'instance mérite d'être ré-étiquetée.

Soit $x$ une donnée dont l'étiquette est demandée. L'étiquette de classe donnée par l'opérateur est notée $y_g$. Soit $y_p = \underset{y \in Y}{\operatorname{argmax}} P_h(y|x)$ l'étiquette de classe de $x$ qui est prédite par le classifieur. Si $y_p = y_g$, alors l'étiquette donnée par l'opérateur est approuvée et nous considérons que ce n'est pas une erreur ; sinon, une erreur peut avoir eu lieu.

### 6.4.1 Vraisemblance d'erreur

Supposons que $y_p \neq y_g$. Nous exprimons la probabilité que $x$ soit mal étiquetée en estimant le degré de désaccord entre la classe prédite $y_p$ et la classe observée $y_g$, qui est proportionnel à la différence de probabilités de $y_p$ et $y_g$. Soient $p_p = P(y_p|x)$ et $p_g = P(y_g|x)$. En utilisant le coefficient de silhouette [Rousseeuw 87] et étant donné que $p_p \geq p_g$, nous avons :

$$\frac{p_p - p_g}{\max(p_p, p_g)} = \frac{p_p - p_g}{p_p} = 1 - \frac{p_g}{p_p}$$

Par conséquent, le degré de désaccord $D_1(x) \in [0, 1]$ est exprimé par :

$$D_1(x) = 1 - \frac{P(y_g|x)}{P(y_p|x)}$$

La Figure. 6.2 montre la répartition d'un ensemble d'instances dont l'étiquette $y_g$ a été demandée à un opérateur et ayant $y_g \neq y_p$, selon le degré de désaccord $D_1$. Plus la valeur de $D_1$ est grande, plus il est probable que $x$ soit mal étiquetée avec $y_g$, parce que la probabilité que $x$ appartienne à $y_g$ serait faible par rapport à celle de $y_p$.

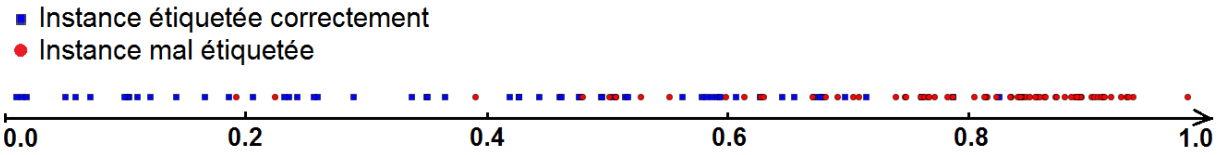



FIGURE 6.2 – Instances mal étiquetées et correctement étiquetées, distribuées selon la mesure $D_1$

Nous présentons une deuxième mesure inspirée par l'apprentissage multi-vues [Sun 87] pour estimer le degré de désaccord. Dans ces approches, les classifieurs sont entraînés sur différentes vues des données à l'aide de différents sous-ensembles de caractéristiques. Les instances dans une vue (décrite par une caractéristique $f_1$) sont dispersées de façon différente que les mêmes instances dans une autre vue (décrite par une caractéristique $f_2$). Par conséquent, il est possible pour une instance $x$ d'être incertain sur sa classe dans une vue et moins incertain dans une autre vue.

Considérons les caractéristiques séparément. La valeur de chaque caractéristique $f_i$ d'une instance $x$ a une contribution $q_y^{f_i}$ pour la classification de $x$ dans une classe $y$. Par exemple, dans un document textuel dont les caractéristiques sont des mots, certains mots peuvent avoir un grand poids dans le choix de la classe. $q_y^{f_i}$ peut être considéré comme un score qui montre avec quel poids la valeur de $f_i$ attire

$x$ vers la classe $y$. Par exemple, soit $x_{f_i}$ l'instance $x$ limitée à la caractéristique $f_i$. Soit $d_y^{f_i}$ la distance moyenne de $x_{f_i}$ à ses $k$ voisins les plus proches appartenant à la classe $y$, limités à la caractéristique $f_i$. Alors, $q_y^{f_i}$ peut être définie comme inversement proportionnelle à la distance $d_y^{f_i}$, par exemple, $q_y^{f_i} = \frac{1}{d_y^{f_i}}$.

Compte tenu de la classe prédite $y_p$ et la classe donnée $y_g$, $q_{y_p}^{f_i}$ et $q_{y_g}^{f_i}$ représentent la contribution de la caractéristique $f_i$ à la classification de $x$ dans $y_p$ et $y_g$ respectivement. Soit $F_p$ l'ensemble des caractéristiques qui contribuent à la classification de $x$ dans la classe prédite plus que dans la classe donnée, et inversement pour $F_g$ :

$$F_p = \{f_i | q_{y_p}^{f_i} > q_{y_g}^{f_i}\} \qquad F_g = \{f_i | q_{y_p}^{f_i} \leq q_{y_g}^{f_i}\}$$

La quantité d'information qui reflète l'appartenance de $x$ à $y_p$ (resp. $y_g$) est :

$$q_p = \sum_{f_i \in F_p} (q_{y_p}^{f_i} - q_{y_g}^{f_i}) \qquad q_g = \sum_{f_i \in F_g} (q_{y_g}^{f_i} - q_{y_p}^{f_i})$$

Notons que $q_p \in [0, +\infty)$ et $q_g \in [0, +\infty)$. Encore une fois, en appliquant le coefficient de silhouette, un degré de désaccord $D_2'(x) \in [-1, 1]$ entre $y_p$ et $y_g$ peut être exprimé par :

$$D_2'(x) = \frac{q_p - q_g}{\max(q_p, q_g)}$$

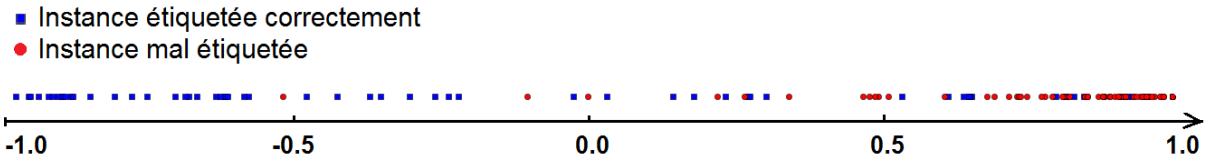



FIGURE 6.3 – Instances mal étiquetées et correctement étiquetées réparties selon $D_2'$

La Figure. 6.3 montre quelques données réparties selon le degré de désaccord $D_2'$. Ce dernier peut être normalisé pour être dans $[0, 1]$ plutôt que dans $[-1, 1]$ simplement par $D_2 = \frac{D_2'+1}{2}$.

La répartition des instances selon $D_1$ et $D_2$ est présentée dans la Figure. 6.4. Nous pouvons définir un score de mauvais étiquetage $D$ en fonction de $D_1$ ou $D_2$ ou d'une combinaison des deux : la moyenne $\frac{D_1+D_2}{2}$, $\max(D_1, D_2)$, ou $\min(D_1, D_2)$. Afin de décider si une instance $x$ est potentiellement mal étiquetée, une façon habituelle consiste à définir un seuil (que nous noterons $t_D$) sur $D$.

En résumé, la mesure de désaccord présentée exprime la probabilité que $x$ soit mal étiquetée. Une grande quantité d'information reflétant la classe prédite et une faible quantité d'information reflétant la classe observée, indique une erreur d'étiquetage. C'est le cas par exemple lorsque la plupart des termes dans un document textuel attirent fortement vers une classe, alors que les autres termes attirent faiblement vers la classe donnée par l'opérateur. Néanmoins, le score de mauvais étiquetage ne donne pas d'informations sur l'importance d'une instance, ni sur la nécessité d'un éventuel ré-étiquetage. Ce point est abordé dans la section suivante.

### 6.4.2 L'informativité des instances mal étiquetées

Dans l'apprentissage actif, les instances pour lesquelles le modèle est incertain, sont désignées comme informatives et leur classe est demandée. Dans cette section, nous ne discutons pas de l'informativité des instances en termes d'incertitude (les instances considérées sont déjà incertaines). Nous

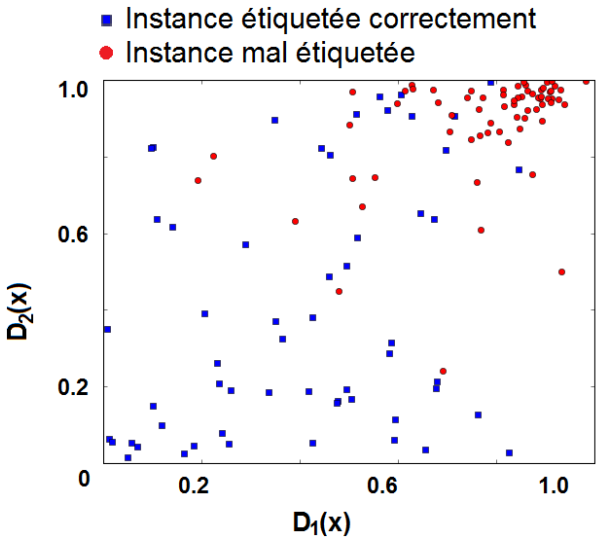



FIGURE 6.4 – Répartition des instances mal étiquetées et correctement étiquetées selon $D_1$ et $D_2$

essayons de déterminer dans quelle mesure le ré-étiquetage d'une instance éventuellement mal étiquetée serait utile.

Il est possible que le degré de désaccord, utilisant $D_1$ et/ou $D_2$, soit incertain concernant une instance $x$, si elle se trouve dans la région de chevauchement entre les instances mal étiquetées et les instances correctement étiquetées (Figure. 6.4). Cela se produit essentiellement lors de la présence d'informations contradictoires dans $x$, soit lorsque $P(y_p|x) \simeq P(y_g|x)$ et $q_p \simeq q_g$.

Prenons l'exemple d'un document textuel :

- *Informations fortement contradictoires* : certains termes attirent fortement le document vers $y_p$ et d'autres termes l'attirent avec la même force vers $y_g$. Dans ce cas $q_p$ et $q_g$ sont à la fois élevés et proches l'un de l'autre.
- *Informations faiblement contradictoires* : les termes attirent faiblement mais de façon égale le document vers $y_p$ et $y_g$, c.à.d. qu'il n'existe pas d'information pertinente ni pour $y_p$ ni pour $y_g$. Dans ce cas $q_p$ et $q_g$ sont à la fois faibles et proches l'un de l'autre.

Dans ces deux cas, $q_p - q_g$ est faible. Cependant, les instances présentant des informations fortement contradictoires sont plus informatives si leurs vraies classes sont connues, et méritent d'être corrigées plus que les autres instances. Par conséquent, en plus de la mesure d'erreur d'étiquetage, nous définissons la mesure d'informativité $I \in [0, +\infty)$ telle que $I = \sqrt{q_p \times q_g}$

Plus $I$ est grand, plus l'étiquette demandée mérite d'être examinée et éventuellement corrigée. La Figure. 6.5 justifie le choix de $I = \sqrt{q_p \times q_g}$, car on obtient une valeur élevée lorsque les valeurs $q_p$ et $q_g$ sont toutes les deux élevées et proches l'une de l'autre. La mesure $I$ n'a pas de borne supérieure mais peut être normalisée, par exemple en divisant par une valeur maximale de $I$ qui peut être calculée sur un ensemble de données de validation. De cette façon, un seuil (que nous noterons $t_I$) peut être défini sur la mesure d'informativité $I$.

## 6.5 Atténuation de l'effet des erreurs d'étiquetage

Lorsqu'une instance $x$ est considérée comme potentiellement mal étiquetée, l'étape suivante consiste à décider quel traitement lui adapter. Dans les méthodes habituelles de filtrage des erreurs d'étiquetage, une base de données étiquetées est disponible à l'avance et les méthodes éliminent les instances qui sont

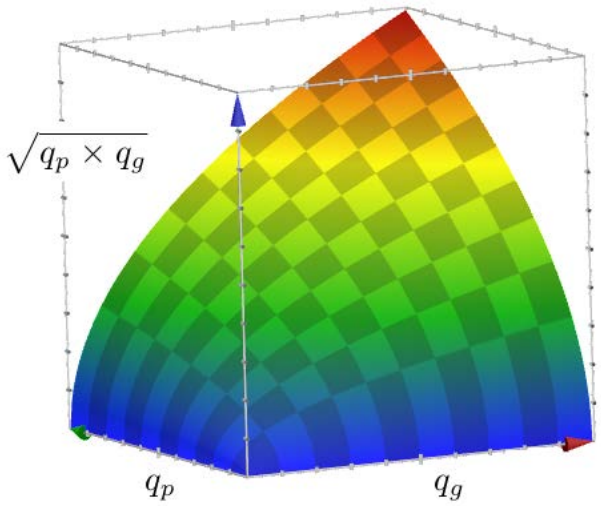



FIGURE 6.5 – $\sqrt{q_p \times q_g}$ est élevée quand $q_p$ et $q_g$ sont élevées et proches l'une de l'autre

susceptibles d'être mal étiquetées ou prédisent leurs étiquettes. Afin d'atténuer l'impact des erreurs sur l'apprentissage actif, nous étudions plusieurs stratégies : le rejet, la pondération, le ré-étiquetage par un opérateur, et nous proposons une approche hybride.

### 6.5.1 Rejet, pondération et ré-étiquetage

**Ce qu'il ne faut pas faire (prédiction) :** Dans un apprentissage actif à partir d'un flux de données, la correction d'une instance mal étiquetée en prédisant son étiquette n'est pas la bonne façon de faire. En effet, la mise à jour du modèle $h$ en utilisant l'étiquette prédite de $x$ ($y = h(x)$) plutôt que l'étiquette demandée, est généralement plus nuisible à l'apprentissage actif que l'étiquette erronée elle-même. Cela est dû au fait que les instances mal étiquetées sont les instances pour lesquelles le modèle $h$ était incertain de leur classe. Par conséquent, prédire leurs étiquettes va probablement entraîner une erreur que le modèle serait incapable de détecter (sinon il l'aurait évitée).

**Rejet :** Lorsque $D(x) > t_D$, $x$ est considéré comme mal étiqueté, sinon il est considéré comme correctement étiqueté. De cette façon, si $x$ est identifié comme étant mal étiqueté, il peut être simplement écarté, c'est-à-dire qu'on ne met pas à jour le modèle de classification $h$ avec $x$.

**Pondération :** En fonction du classifieur de base utilisé, les instances peuvent être pondérées afin que le classifieur apprenne plus à partir des instances avec un poids plus élevé. Une alternative possible pour atténuer l'effet des erreurs d'étiquetage sans définir un seuil sur $D$, est de mettre à jour le modèle $h$ en utilisant toute instance $x$ avec son étiquette observée $y_g$ pondérée par $w = 1 - D(x)$ qui est petit lorsque la probabilité de mauvais étiquetage $D(x)$ est grande. Ainsi, une instance qui a une forte probabilité d'erreur d'étiquetage aura un poids plus proche de 0 et n'affectera pas beaucoup le modèle de classification $h$, contrairement à une instance avec une faible probabilité d'erreur d'étiquetage.

**Ré-étiquetage :** Si un opérateur alternatif fiable est disponible, l'étiquette d'une instance potentiellement mal étiquetée (ayant $D > t_D$) peut être vérifiée et éventuellement corrigée par cet opérateur. Ensuite, le modèle $h$ est mis à jour en utilisant l'instance et son étiquette corrigée.

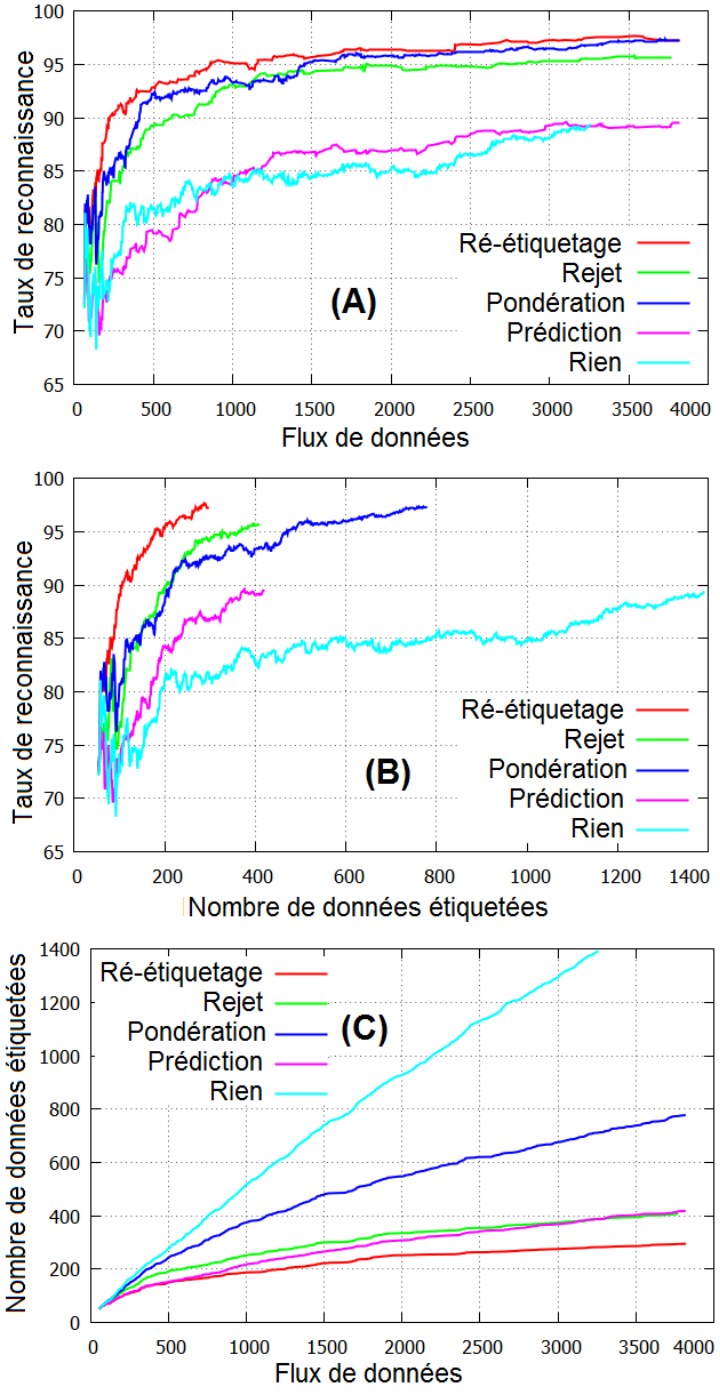



FIGURE 6.6 – Apprentissage actif à partir d'un flux de données avec une intensité d'erreurs $\sigma = 0.3$ et $t_D = 0.6$. La base de données utilisée est optdigits. Le cas "Rien" indique qu'aucune stratégie n'est utilisée : les erreurs d'étiquetage ne sont pas atténuées.

La Figure. 6.6 montre les résultats obtenus en utilisant les différentes stratégies. La meilleure solution consiste à ré-étiqueter correctement les instances qui sont identifiées comme mal étiquetées. Toutefois, cela a un coût et nécessite l'intervention d'un opérateur expert qui est supposé être fiable. Rejeter les instances potentiellement mal étiquetées peut améliorer les résultats de la classification. Toutefois, des instances informatives qui ont été correctement étiquetées peuvent être perdues, surtout si de nombreux cas sont rejetés à tort (dépendamment du seuil $t_D$). Plutôt que de rejeter des instances potentiellement mal étiquetées, la pondération de toutes les instances, avec une importance qui dépend de la probabilité d'erreur peut améliorer les performances sans perdre des instances informatives (mais avec le risque de sous-pondération de certaines instances informatives). Enfin, la Figure. 6.6 confirme qu'il est dangereux de prédire l'étiquette d'une instance filtrée $x$ et de mettre à jour le modèle en utilisant l'instance $x$ avec son étiquette prédite. En effet, certaines méthodes de nettoyage de bases de données utilisées pour l'apprentissage supervisé proposent de prédire l'étiquette des instances potentiellement mal étiquetées. Cependant, dans le cas d'un apprentissage actif, cela devient dangereux, parce que les étiquettes demandées sont précisément celles des instances incertaines (pour lesquels le classifieur est incertain sur sa prédiction).

**Stratégie hybride :** La correction des instances mal étiquetées en utilisant leur véritable étiquette de classe donne les meilleurs résultats. Cependant, cela nécessite un opérateur expert et implique un coût élevé car cet opérateur est supposé ne pas faire d'erreurs. Nous présentons une approche hybride qui minimise le coût de l'utilisation d'un opérateur expert. Vu que le ré-étiquetage est coûteux, nous supposons que nous avons un budget limité $B$ pour le ré-étiquetage, c'est-à-dire que l'expert peut examiner et corriger seulement $B$ instances. Étant donné un budget $B$, le problème consiste à choisir les $B$ instances à corriger. En fait, le ré-étiquetage des instances qui sont informatives selon la mesure $I$ (voir la section 6.4.2) est plus susceptible d'améliorer l'exactitude de la classification. Par conséquent, si une instance $x$ est identifiée comme étant mal étiquetée et a une grande informativité $I(x) > t_I$, alors elle sera ré-étiquetée. Autrement, la stratégie de rejet ou de pondération peut être utilisée.

## 6.6 Expérimentations

Nous utilisons pour nos expérimentations les bases de données vues précédemment. Un SVM est utilisé comme classifieur de base (nous utilisons l'implémentation python disponible sur scikit-learn [Pedregosa 11]). Le seuil $t_D$ est fixé à $0.6$ pour tous les ensembles de données, même si un meilleur seuil peut être trouvé pour chaque ensemble de données.

### 6.6.1 Évaluation du degré de désaccord

Cette expérimentation montre la capacité de la mesure de désaccord $D$ proposée à caractériser correctement les instances mal étiquetées. Les étiquettes des instances sont corrompues de telle sorte que les instances avec une faible probabilité de prédiction sont plus susceptibles d'être mal étiquetées. Ensuite, les instances de chaque ensemble de données sont triées selon leur degré de désaccord défini en termes de $D_1$ et $D_2$. Les résultats sont comparés à une mesure d'entropie $E$ qui est couramment utilisée dans l'apprentissage actif. Une instance avec une faible entropie implique une classification confiante. Les instances qui ont une entropie $E$ faible sont plus susceptibles d'être mal étiquetées si leur étiquette prédite $y_p$ (qui est confiante) est différente de l'étiquette observée $y_g$.

$$E = -\sum_{y \in Y} P_h(y|x) \times \log P_h(y|x)$$

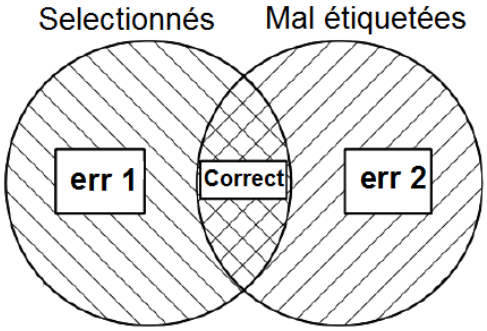



FIGURE 6.7 – Types d'erreurs comme présentées dans [Brodley 99] : $err_1$ représente les instances correctement étiquetées qui sont détectées à tort comme étant mal étiquetées, et $err_2$ les instances mal étiquetées qui ne sont pas détectées.

Nous sélectionnons les $n$ instances ayant le plus grand degré de désaccord de chaque ensemble de données et nous calculons deux types d'erreurs : $err_1$ qui représente les instances correctement étiquetées qui sont sélectionnées à tort, et $err_2$ les instances mal étiquetées qui ne sont pas sélectionnées (Figure 6.7) :

$$err_1 = \frac{\text{correct\_selected}}{\text{correct}} \qquad err_2 = \frac{\text{mislabelled\_unselected}}{\text{mislabelled}}$$

où "correct_selected" est le nombre d'instances correctement étiquetées qui sont sélectionnées comme étant potentiellement mal étiquetées. "mislabelled_unselected" est le nombre d'instances mal étiquetées qui ne sont pas sélectionnées. "correct" et "mislabelled" sont respectivement le nombre total d'instances correctement étiquetées et mal étiquetées.

Nous calculons également le pourcentage d'instances sélectionnées qui sont mal étiquetées $prec$, ce qui représente la précision de détection des erreurs :

$$prec = \frac{\text{mislabelled\_selected}}{\text{selected}}$$

où "mislabelled_selected" est le nombre d'instances mal étiquetées qui sont sélectionnées et "selected" est le nombre d'instances sélectionnées.

Le Tableau 6.1 montre les résultats obtenus. Nous pouvons voir que pour les bases *documents* et *pendigits*, $D_2$ atteint de meilleurs résultats que $D_1$ et inversement pour les bases *optdigits* and *letter-recognition*. Cependant, combiner $D_1$ et $D_2$ en utilisant $D = \max(D_1, D_2)$ peut donner de meilleurs résultats que $D_1$ et $D_2$ séparés, et atteint toujours de meilleurs résultats que la mesure d'entropie $E$. Ceci est confirmé par les résultats "en moyenne" obtenus sur l'ensemble des bases. Dans la suite, nous utilisons la mesure $D$, comme mesure de désaccord pour tous les ensembles de données.

### 6.6.2 Atténuation de l'effet des erreurs d'étiquetage

Dans cette expérimentation, nous considérons un apprentissage actif à partir d'un flux où les erreurs d'étiquetage sont atténuées suivant différentes stratégies : ré-étiquetage, rejet et pondération. Une stratégie hybride est également considérée où seul un petit nombre d'instances est étiqueté manuellement selon un budget de ré-étiquetage $B$. Pour la stratégie hybride, sans perte de généralité, nous avons utilisé dans nos expériences, un budget de $B = 20$ instances autorisées à être ré-étiquetées. Les résultats avec d'autres valeurs de $B$ conduisent à des conclusions similaires. Les stratégies envisagées sont énumérées ci-dessous :

TABLE 6.1 – Mesures de vraisemblance d'erreurs

| | $D_1$ | $D_2$ | $D$ | $E$ |
|---|---|---|---|---|
| **Base Optdigits** | | | | |
| $err_1$ | **0.67** | 1.49 | 1.12 | 1.27 |
| $err_2$ | **1.22** | 3.14 | 2.26 | 2.61 |
| $prec$ | **98.44** | 96.53 | 97.4 | 97.05 |
| **Base Pendigits** | | | | |
| $err_1$ | 2.34 | 2.19 | **2.07** | 2.61 |
| $err_2$ | 1.37 | 1.02 | **0.75** | 2.00 |
| $prec$ | 94.75 | 95.10 | **95.35** | 94.15 |
| **Base des Documents** | | | | |
| $err_1$ | 3.51 | 3.29 | **2.96** | 4.50 |
| $err_2$ | 3.07 | 2.56 | **1.79** | 5.38 |
| $prec$ | 92.2 | 92.69 | **93.42** | 90.0 |
| **Base Letter-recognition** | | | | |
| $err_1$ | **1.82** | 3.2 | 1.93 | 2.29 |
| $err_2$ | **21.1** | 24.3 | 21.35 | 22.18 |
| $prec$ | **94.87** | 91.03 | 94.6 | 93.57 |

| | $D_1$ | $D_2$ | $D$ | $E$ |
|---|---|---|---|---|
| **Résultats moyens sur toutes les bases** | | | | |
| $err_1$ | 2.08 | 2.54 | **2.02** | 2.66 |
| $err_2$ | 6.69 | 7.75 | **6.53** | 8.04 |
| $prec$ | 95.07 | 93.84 | **95.20** | 93.70 |

- Ré-étiquetage complet : ré-étiqueter toute instance $x$ qui est identifiée comme mal étiquetée (c.à.d. si $D(x) > t_D$, alors $x$ est ré-étiquetée).
- Rejet complet : rejeter toutes les instances $x$ qui sont identifiées comme mal étiquetées.
- Pondération complète : utiliser toutes les instances avec leurs étiquettes demandées $(x, y_g)$ pondérées par $w = 1 - D(x)$ pour mettre à jour le modèle de classification.
- Rejet et ré-étiquetage hybride : ré-étiqueter une instance qui est identifiée comme mal étiquetée seulement si elle a une grande informativité ($I(x) > t_I$) et le budget $B$ n'est pas encore épuisé. Dans le cas contraire, l'instance est rejetée.
- Pondération et ré-étiquetage hybride : identique à la stratégie hybride ci-dessus mais en utilisant la pondération au lieu du rejet.

Les résultats obtenus pour chacune des stratégies sont illustrés à la Figure. 6.8 et le tableau 6.2. Le taux de reconnaissance obtenu sur une base de test est représenté sur la Fig. 6.8 en fonction du nombre d'instances étiquetées. Le Tableau 6.2 montre l'exactitude finale de la classification, le nombre final d'instances $N_1$ dont l'étiquette a été demandée au premier opérateur (non fiable), et le nombre d'instances $N_2$ qui sont ré-étiquetées par l'opérateur expert (fixé à $N_2 = B = 20$ pour les stratégies hybrides). Soient $c_1$ et $c_2$ respectivement le coût requis par le premier opérateur et l'opérateur expert pour étiqueter une seule instance. On suppose que $c_2 > c_1$ vu que l'opérateur expert est supposé être fiable (ou beaucoup plus fiable que le premier opérateur). Le coût d'étiquetage est alors $c_1 \times N_1$, le coût du ré-étiquetage est $c_2 \times N_2$, et le coût global est $C = c_1 \times N_1 + c_2 \times N_2$.

Tout d'abord, les résultats sur la Figure 6.8 confirment que la stratégie de "ré-étiquetage complet" est évidement la stratégie la plus efficace en termes d'exactitude de la classification puisque toutes les instances qui sont identifiées comme étant mal étiquetées sont ré-étiquetées par l'opérateur expert. Ensuite, les résultats obtenus sur l'ensemble des jeux de données montrent que rejeter les instances qui sont potentiellement mal étiquetées n'est pas meilleure que la stratégie de pondération. En fait, il a été observé dans de nombreux travaux qui traitent les erreurs d'étiquetage [Gamberger 96, Brodley 99] que l'apprentissage avec les instances mal étiquetées nuit plus que d'enlever trop de cas correctement étiquetés, mais ce n'est pas vrai dans le cadre d'un apprentissage actif, comme il est confirmé par les résultats obtenus. Ceci est dû au fait que, dans le cadre d'un apprentissage actif, les instances écartées sont plus susceptibles d'améliorer le modèle de classification si elles sont correctement étiquetées, ainsi, les rejeter peut avoir un impact négatif sur les performances de l'apprentissage actif. Pour la même raison, nous pouvons voir que la stratégie de "pondération et ré-étiquetage hybride" fonctionne mieux que la stratégie de "rejet et ré-étiquetage hybride". En outre, la Figure. 6.8 montre pour la stratégie de "pondération et ré-étiquetage hybride" que le ré-étiquetage de seulement $B = 20$ instances ayant une valeur élevée de $I$,

TABLE 6.2 – Taux de reconnaissance et coût final selon différentes stratégies

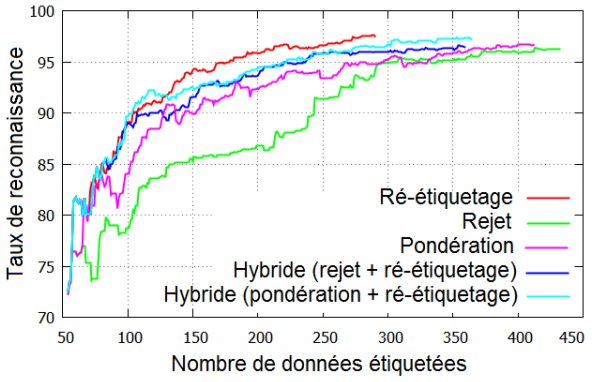
| | Ré-étiquetage | Rejet | Rejet et ré-étiquetage | Pondération | Pondération et ré-étiquetage |
|---|---|---|---|---|---|
| **Base Optdigits** | | | | | |
| Reconnaissance | 97.49% | 96.21% | 96.43% | 96.66% | 97.21% |
| $N_1$ | 290 | 432 | 359 | 412 | 364 |
| $N_2$ | 92 | 0 | 20 | 0 | 20 |
| **Base Pendigits** | | | | | |
| Reconnaissance | 98.11% | 94.02% | 97.31% | 96.36% | 97.88% |
| $N_1$ | 307 | 464 | 420 | 420 | 363 |
| $N_2$ | 105 | 0 | 20 | 0 | 20 |
| **Base Letters-recognition** | | | | | |
| Reconnaissance | 85.98% | 53.95% | 66.79% | 80.26% | 85.52% |
| $N_1$ | 914 | 1242 | 956 | 1537 | 1344 |
| $N_2$ | 211 | 0 | 20 | 0 | 20 |
| **Base Documents** | | | | | |
| Reconnaissance | 96.15% | 81.84% | 90.61% | 94.46% | 96.0% |
| $N_1$ | 406 | 413 | 424 | 701 | 646 |
| $N_2$ | 104 | 0 | 20 | 0 | 20 |
| **Moyenne sur l'ensemble des bases** | | | | | |
| Reconnaissance | 94.42% | 81.50% | 87.78% | 91.93% | 94.15% |
| $N_1$ | 479.25 | 637.75 | 539.75 | 767.5 | 689.25 |
| $N_2$ | 128 | 0 | 20 | 0 | 20 |

améliore grandement l'exactitude par rapport à la stratégie de "rejet complet" ou de "pondération complète". Nous pouvons voir dans le Tableau 6.2 que la stratégie de "pondération et ré-étiquetage hybride" atteint une exactitude de classification finale qui est assez proche de celle de la stratégie de "pondération complète". Par exemple, l'exactitude finale obtenue par la stratégie "pondération et ré-étiquetage hybride" pour l'ensemble de données "optdigits" est $97.21\%$, tandis que celle obtenue par la stratégie du "pondération complète" est $97.49\%$. Comme expliqué précédemment, les erreurs d'étiquetages font que l'apprentissage actif demande plus d'instances étiquetées. Cela explique pourquoi $N_1$ dans le tableau 6.2 est plus petit dans la stratégie de "pondération complète". Toutefois, en prenant en compte le coût induit par le ré-étiquetage des instances mal étiquetées, la stratégie de "pondération complète" aura un coût global plus élevé que les autres stratégies, car toutes les instances qui sont identifiées comme étant mal étiquetées sont ré-étiquetées.

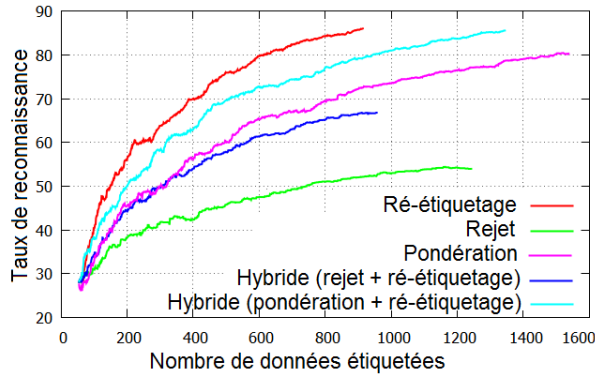
Enfin, nous pouvons conclure que, bien que la stratégie de "pondération et ré-étiquetage hybride" ait un faible coût de ré-étiquetage, elle permet d'obtenir un taux de reconnaissance finale qui est proche de celui obtenu par la stratégie de "ré-étiquetage complet". Par conséquent, si un budget de ré-étiquetage limité est disponible, ce budget devrait être consacré au ré-étiquetage des instances avec une grande informativité $I$.

## 6.7 Conclusion

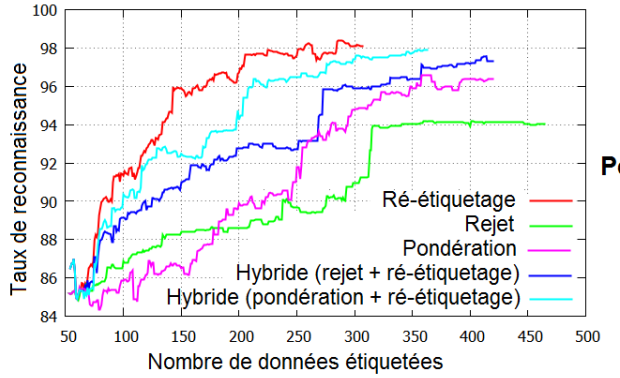
Dans ce chapitre, nous avons abordé le problème de détection et d'atténuation des erreurs d'étiquetage dans le cadre d'un apprentissage actif à partir d'un flux de données. Afin d'identifier les instances potentiellement mal étiquetées, nous avons proposé une mesure qui reflète la vraisemblance qu'une instance soit mal étiquetée en se basant sur le degré de désaccord entre les probabilités et la quantité d'informations que l'instance porte pour la classe prédite et la classe observée. Ensuite, nous avons calculé une



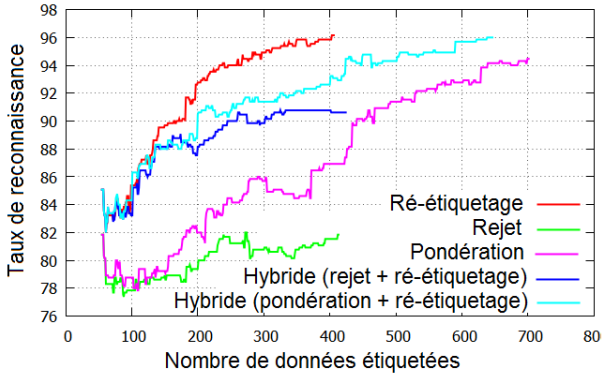

**Base Optdigits**




**Base Letters-recognition**




**Base Pendigits dataset**




**Base Documents**

FIGURE 6.8 – Taux de reconnaissance obtenu sur une base de test en fonction du nombre d'étiquettes demandées en utilisant les différentes stratégies

mesure d'informativité qui reflète à quel point une étiquette demandée serait utile si elle est corrigée. Nos expériences sur des données réelles montrent que le coefficient de désaccord proposé est plus efficace dans la caractérisation des erreurs d'étiquetage par rapport à la mesure d'entropie couramment utilisée. L'évaluation expérimentale montre également que les instances potentiellement mal étiquetées avec des informations hautement contradictoires méritent d'être ré-étiquetées.

# Chapitre 7

# Conclusion et travaux futurs

**Sommaire**



## 7.1 Conclusion générale

Nous avons abordé dans cette thèse, le problème de l'apprentissage et de la classification de flux de données. L'une des principales questions que nous avons considérée concerne la supervision du système d'apprentissage pour savoir comment réduire le coût de l'étiquetage manuel tout en atteignant des performances de classification similaires à celles d'un apprentissage entièrement supervisé. Pour répondre à cette question, nous avons proposé une nouvelle stratégie d'apprentissage actif sélectionnant les instances qui doivent être étiquetées manuellement. Cette stratégie est accompagnée d'un seuil adaptatif lui permettant d'être utilisable dans le cadre d'un flux de données et d'être indépendant du paramétrage d'un utilisateur expert. Les expérimentations réalisées confirment que la stratégie proposée améliore significativement les performances de l'apprentissage actif par rapport aux stratégies d'incertitude couramment utilisées. Elle permet d'atteindre des performances similaires à celles d'un apprentissage entièrement supervisé, tout en nécessitant beaucoup moins d'étiquetage. De plus, elle joue le rôle d'une fonction de rejet afin de réduire le taux d'erreurs en permettant au classifieur de s'abstenir de classer les données dont il est incertain.

Par ailleurs, dans de nombreuses applications, le flux de données est de nature évolutive et de nouvelles classes inconnues peuvent donc apparaître à tout moment. Ceci pose un problème pour l'apprentissage actif car il détermine l'importance des instances à étiqueter, selon des classes connues. Pour répondre à cette contrainte, nous avons proposé une méthode adaptative pour la détection active de nouvelles classes. L'évaluation expérimentale montre la capacité de la méthode à s'adapter efficacement à l'évolution du flux, et montre des résultats supérieurs en comparaison avec la méthode "SVM à une classe", couramment utilisée pour la détection de nouveautés.

En outre, les erreurs d'étiquetage ont un impact négatif sur l'apprentissage actif. Afin de répondre à ce problème, nous nous sommes posés deux questions principales : (i) quelles instances sont susceptibles d'être mal étiquetées ? et (ii) quelles instances méritent d'être ré-étiquetées ? Nous avons répondu à la première question par l'introduction d'une mesure qui calcule le degré de désaccord entre la classe prédite et celle donnée par l'opérateur. La seconde question a été abordée par l'introduction d'une nouvelle mesure d'informativité caractérisant des informations hautement contradictoires. Les expérimentations effectuées concluent qu'une approche hybride consistant à filtrer les instances mal étiquetées via la pre-

mière mesure et ré-étiqueter les instances informatives via la deuxième mesure, améliore significativement les performances de l'apprentissage.

Les contributions apportées trouvent toutes leurs intérêts dans les domaines d'application où les données peuvent être disponibles en grandes quantités, mais où l'étiquetage manuel de ces données est fastidieux, non fiable et coûteux. Par exemple, dans le domaine de la recherche d'information, les systèmes doivent être entraînés avec des documents étiquetés avec des annotations détaillées. L'opérateur doit identifier des entités, tels que les noms de personnes et d'organisations, ou des relations d'intérêt dans le texte, tel que le fait qu'une personne travaille pour une organisation particulière. La localisation de ces entités et relations est un processus coûteux en temps et énergie. De même, apprendre à classer des documents très divers issus du Web, nécessite qu'un utilisateur donne une étiquette particulière pour chaque document. L'étiquetage de milliers de ces documents peut être fastidieux et même redondant. Dans le domaine de la reconnaissance vocale, un étiquetage précis de séquences vocales est extrêmement coûteux en temps ; en effet, il est rapporté dans [Zhu 05] que l'annotation d'un mot peut prendre dix fois plus de temps que sa prononciation (c.à.d. une seule minute de parole prend dix minutes pour être étiquetée).

## 7.2 Travaux futurs

Dans le futur, nous envisageons d'apporter les améliorations suivantes.

Jusqu'à maintenant, le coût d'étiquetage a été présenté en termes de nombre de données à étiqueter manuellement. Cependant, il est aussi possible d'introduire une vraie notion de "coût d'étiquetage" : si par exemple les données sont des documents, alors il peut être plus coûteux, pour un opérateur humain, d'étiqueter certains documents plutôt que d'autres. Ainsi, en fonction du type et de la qualité des documents, on peut prendre en compte un coût d'étiquetage variable selon les documents. Ce coût peut être, par exemple, fonction de la durée moyenne nécessaire pour étiqueter chaque type de document.

Par ailleurs, l'une des limites de la méthode de détection de nouvelles classes proposée est que sa performance est affectée négativement par les rares nouvelles classes de petites tailles qui se chevauchent fortement avec les classes existantes. Les quelques méthodes existantes pour détecter ce genre de classes [He 07], font des hypothèses qui ne sont pas valables pour un flux de données. Ceci reste un défi de recherche ouvert qui n'a pas encore été résolu pour les flux de données et constitue l'un de nos travaux futurs. En outre, la détection de nouvelles classes dépend de la fonction de distance utilisée ; elle peut souffrir de ce qui est appelé "fléau de la dimension" dans le cas de données hautement multidimensionnelles. Cela peut être résolu en utilisant la "distance partagée des plus proches voisins" (en anglais, "shared-nearest-neighbour distance") proposée dans [Houle 10] pour laquelle il a été montré que les résultats sont améliorés dans le cas d'un grand nombre de dimensions et de caractéristiques non pertinentes. Cependant, le calcul d'une telle distance est très coûteux en temps.

Concernant la gestion des erreurs d'étiquetage, une limitation de la méthode proposée est qu'elle nécessite un seuil sur la mesure d'informativité $I$ qui est difficile à adapter automatiquement, vu qu'on ne dispose pas d'une vérité terrain indiquant si une instance est correctement étiquetée ou pas. Ainsi, une amélioration possible consiste à minimiser le coût de ré-étiquetage, par la définition et l'optimisation d'une fonction multi-objectifs qui combine à la fois : (i) la probabilité d'erreur d'étiquetage, (ii) le caractère informatif d'une instance, et (iii) le coût de ré-étiquetage.

Un autre aspect à approfondir concerne le comportement de la méthode proposée dans les environnements dynamiques. Actuellement, la méthode est capable de s'adapter à l'apparition de nouvelles classes dans le flux. Un travail futur consiste à caractériser le comportement de la méthode en présence d'une dérive des concepts.

# Annexe A

# Liste de publications

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *An adaptive streaming active learning strategy based on instance weighting*. Submitted to Pattern Recognition Letters - PRL, 2015.

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *Online unsupervised neural-gas learning method for infnite data streams*. Advances in Intelligent and Soft Computing, Pattern Recognition Applications and Methods, Springer International Publishing, vol. 318, pp. 57-70, 2015.

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *Stream-based active learning in the presence of label noise*. International Conference on Pattern Recognition Applications and Methods - ICPRAM, Lisbon (Portugal), pp. 1-10, 2015. (**Best paper award**).

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *Classification de flux de documents évolutifs avec apprentissage de classes inconnues*. Document Numérique (Hermès/Lavoisier), vol. 17, no. 3, pp. 25-46, 2014.

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *Efficient active novel class detection for data stream classification*. IEEE International Conference on Pattern Recognition - ICPR, Stockholm (Sweden), pp. 2826-2831, 2014.

– Hani Daher, Mohamed-Rafik Bouguelia, Abdel Belaïd and Vincent Poulain d'Andecy. *Multipage administrative document stream segmentation*. IEEE International Conference on Pattern Recognition - ICPR, Stockholm (Sweden), pp. 966-971, 2014.

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *Classification active de flux de documents avec identification des nouvelles classes*. In Colloque International Francophone sur l'Ecrit et le Document - CIFED, pp. 75-89, Nancy (France), 2014. (**Best paper award**).

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *A stream-based semi-supervised active learning approach for document classification*. IEEE International Conference on Document Analysis and Recognition - ICDAR, Washington DC (USA), pp. 611-615, 2013.

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *Document image and zone classification through incremental learning*. IEEE International Conference on Image Processing - ICIP,

Melbourne (Australia), pp. 4230-4234, 2013.

– Mohamed-Rafik Bouguelia, Yoland Belaïd and Abdel Belaïd. *An adaptive incremental clustering method based on the growing neural gas algorithm*. International Conference on Pattern Recognition Applications and Methods - ICPRAM, Barcelona (Spain), pp. 42-49, February 2013.

# Bibliographie


[Bouguelia 15(3)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *An adaptive streaming active learning strategy based on instance weighting*. Submitted to Pattern Recognition Letters - PRL, 2015.

[Bouguelia 14(1)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *Classification de flux de documents évolutifs avec apprentissage de classes inconnues*. Document Numérique (Hermès/Lavoisier), vol. 17, no. 3, pp. 25-46, 2014.

[Bouguelia 15(2)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *Online unsupervised neural-gas learning method for infnite data streams*. Advances in Intelligent and Soft Computing, Pattern Recognition Applications and Methods, Springer International Publishing, volume 318, pp. 57-70, 2015.

[Bouguelia 15(1)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *Stream-based active learning in the presence of label noise*. International Conference on Pattern Recognition Applications and Methods - ICPRAM, Lisbon (Portugal), pp. 1-10, 2015.

[Bouguelia 14(3)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *Efficient active novel class detection for data stream classification*. IEEE International Conference on Pattern Recognition - ICPR, Stockholm (Sweden), pp. 2826-2831, 2014.

[Daher 14] H. Daher, M.R. Bouguelia, A. Belaïd, V.P. D'Andecy *Multipage administrative document stream segmentation*. IEEE International Conference on Pattern Recognition - ICPR, Stockholm (Sweden), pp. 966-971, 2014.

[Bouguelia 14(2)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *Classification active de flux de documents avec identification des nouvelles classes*. In Colloque International Francophone sur l'Ecrit et le Document - CIFED, pp. 75-89, Nancy (France), 2014.

[Bouguelia 13(3)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *A stream-based semi-supervised active learning approach for document classification*. IEEE International Conference on Document Analysis and Recognition - ICDAR, Washington DC (USA), pp. 611-615, 2013.

[Bouguelia 13(2)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *Document image and zone classification through incremental learning*. IEEE International Conference on Image Processing - ICIP, Melbourne (Australia), pp. 4230-4234, 2013.

[Bouguelia 13(1)] M.R. Bouguelia, Y. Belaïd, A. Belaïd. *An adaptive incremental clustering method based on the growing neural gas algorithm*. International Conference on Pattern Recognition Applications and Methods - ICPRAM, Barcelona (Spain), pp. 42-49, February 2013.

[Sebastiani 02] F. Sebastiani. *Machine learning in automated text categorization*. ACM computing surveys (CSUR), vol. 34, no 1, pp. 1-47, 2002.

[Kotsiantis 06] S. B. Kotsiantis, I. Zaharakis, , P. Pintelas. *Machine learning : a review of classification and combining techniques*. Artificial Intelligence Review, vol. 26, no 3, pp. 159-190, 2006.

[Bishop 06] C.M. Bishop *Pattern recognition and machine learning*. Springer, vol. 4, no 4, 2006.

[Chen 07] N. Chen, D. Blostein *A survey of document image classification : problem statement, classifier architecture and performance evaluation*. International Journal of Document Analysis and Recognition, vol. 10, no 1, pp. 1-16, 2007.

[Aggarwal 12] C.C. Aggarwal C. Zhai *A survey of text clustering algorithms*. Mining Text Data. Springer US, pp. 77-128, 2012.

[Zhu 08] X. Zhu. *Semi-supervised learning literature survey*. Computer Sciences Technical Report 1530, University of Wisconsin-Madison, 2008.

[Nigam 06] K. Nigam, A. McCallum, T. Mitchell. *Semi-supervised text classification using EM*. Semi-Supervised Learning, pp. 33-56, 2006.

[Carmagnac 04] F. Carmagnac, P. Héroux, É. Trupin. *Multi-view HAC for semi-supervised document image classification*. Document Analysis Systems VI, Springer Berlin Heidelberg, pp. 191-200, 2004.

[Krithara 08] A. Krithara, M. R. Amini, J. M. Renders, C. Goutte. *Semi-supervised document classification with a mislabeling error model*. European Conference on Information Retrieval, pp. 191-200, 2008.

[Bache 13] K. Bache, M. Lichman. *UCI Machine Learning Repository*. [http ://archive.ics.uci.edu/ml]. Irvine, CA : University of California, School of Information and Computer Science, 2013.

[Pedregosa 11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay. *Scikit-learn : Machine Learning in Python*. Journal of Machine Learning Research, pp. 2825-2830, 2011.

[Angluin 88] D. Angluin. *Queries and concept learning*. Machine Learning, vol. 2, no 4, pp. 319-342, 1988.

[Angluin 04] D. Angluin. *Queries revisited*. Theoretical Computer Science, vol. 313, no 2, pp. 175-194, Springer-Verlag, 2004.

[Lang 92] K. Lang and E. Baum. *Query learning can work poorly when a human oracle is used*. IEEE International Joint Conference on Neural Networks, pp. 335-340, 1992.

[Cohn 94] D. Cohn, L. Atlas, and R. Ladner. *Improving generalization with active learning*. Machine Learning, pp. 201-221, 1994.

[Dagan 95] I. Dagan and S. Engelson. *Committee-based sampling for training probabilistic classifiers*. International Conference on Machine Learning, pp. 150-157, 1995.

[Lewis 94] D. Lewis and W. Gale. *A sequential algorithm for training text classifiers*. SIGIR Conference on Research and Development in Information Retrieval, pp. 3-12, 1994.

[Scheffer 01] T. Scheffer, C. Decomain, and S.Wrobel. *Active hidden Markov models for information extraction*. International Conference on Advances in Intelligent Data Analysis, pp. 309-318, 2001.

[Shannon 48] C.E. Shannon. *A mathematical theory of communication*. Bell System Technical Journal, pp. 623-656, 1948.

[Lewis 94 (2)] D. Lewis and J. Catlett. *Heterogeneous uncertainty sampling for supervised learning*. International Conference on Machine Learning, pp. 148-156, 1994.

[Lindenbaum 04] M. Lindenbaum, S. Markovitch, and D. Rusakov. *Selective sampling for nearest neighbor classifiers*. Machine Learning, pp. 125-152, 2004.

[Tong 00] S. Tong and D. Koller. *Support vector machine active learning with applications to text classification*. International Conference on Machine Learning, pp. 999-1006, 2000.

[Korner 06] C. Korner and S. Wrobel. *Multi-class ensemble-based active learning*. European Conference on Machine Learning, pp. 687-694, 2006.

[Schein 07] A.I. Schein and L.H. Ungar. *Active learning for logistic regression : An evaluation*. Machine Learning, vol. 68, no 3, pp. 235-265, 2007.

[Settles 08] B. Settles and M. Craven. *An analysis of active learning strategies for sequence labeling tasks*. Conference on Empirical Methods in Natural Language Processing, pp. 1069-1078, 2008.

[Seung 92] H.S. Seung, M. Opper, and H. Sompolinsky. *Query by committee*. Workshop on Computational Learning Theory, pp. 287-294, 1992.

[McCallum 98] A. McCallum and K. Nigam. *Employing EM in pool-based active learning for text classification*. International Conference on Machine Learning, pp. 287-294, 1998.

[Abe 98] N. Abe and H. Mamitsuka. *Query learning strategies using boosting and bagging*. International Conference on Machine Learning, pp. 1-9, 1998.

[Melville 05] P. Melville, S.M. Yang, M. Saar-Tsechansky, and R. Mooney. *Active learning for probability estimation using Jensen-Shannon divergence*. European Conference on Machine Learning, pp. 268-279, 2005.

[Haussler 94] D. Haussler. *Learning conjunctive concepts in structural domains*. Machine Learning, pp. 7-40, 1994.

[Roy 01] N. Roy and A. McCallum. *Toward optimal active learning through sampling estimation of error reduction*. International Conference on Machine Learning, pp. 441-448, 2001.

[Guo 01] Y. Guo and R. Greiner. *Optimistic active learning using mutual information*. International Joint Conference on Artificial Intelligence, pp. 823-829, 2007.

[Zhu 03] X. Zhu, J. Lafferty, and Z. Ghahramani. *Combining active learning and semisupervised learning using Gaussian fields and harmonic functions*. ICML Workshop on the Continuum from Labeled to Unlabeled Data, pp. 58-65, 2003.

[Nguyen 04] H.T. Nguyen and A. *Active learning using pre-clustering*. International Conference on Machine Learning, pp. 79-86, 2004.

[Dasgupta 08] S. Dasgupta, D. Hsu. *Hierarchical sampling for active learning*. International Conference on Machine Learning, pp. 208-215, 2008.

[Dasgupta 11] S. Dasgupta. *Two faces of active learning*. Theoretical computer science, pp. 1767-1781, 2011.

[Settles 08 (2)] B. Settles, M. Craven, and S. Ray. *Multiple-instance active learning*. Advances in Neural Information Processing Systems, pp. 1289-1296, 2008.

[Sorower 10] M. S. Sorower *A literature survey on algorithms for multi-label learning*. Oregon State University, Corvallis, 2010.

[Yang 07] X. Yang, Q. Song, and Y. Wang. *A weighted support vector machine for data classification*. International Journal of Pattern Recognition and Artificial Intelligence, pp. 961-976, 2007.

[Jiang 07] J. Jiang and C. Zhai. *Instance weighting for domain adaptation in NLP*. Annual Meeting of the Association for Computational Linguistics, pp. 254-271, 2007.

[Dudani 76] S. A. Dudani. *The distance-weighted k-nearest-neighbor rule*. IEEE Transactions on Systems, Man and Cybernetics, pp. 325-327, 1976.

[Frank 02] E. Frank, M. Hall, and B. Pfahringer *Locally weighted naive bayes*. Conference on Uncertainty in Artificial Intelligence, pp. 249-256, 2002.

[Xia 13] R. Xia, X. Hu, , J. Lu, J. Yang, C. Zong. *Instance selection and instance weighting for cross-domain sentiment classification via PU learning*. International joint conference on Artificial Intelligence, pp. 2176-2182, 2013.

[Jiang 08] J. Jiang *A literature survey on domain adaptation of statistical classifiers*, 2008 URL : http ://sifaka.cs.uiuc.edu/jiang4/domain_adaptation/survey

[Karampatziakis 10] N. Karampatziakis, J. Langford *Importance Weight Aware Gradient Updates*. CoRR Computing Research Repository, 2010.

[Muslea 00] I. Muslea, S. Minton, and C.A. Knoblock. *Selective sampling with redundant views*. Proceedings of the National Conference on Artificial Intelligence, pp. 621-626, 2000.

[Melville 04] P. Melville and R. Mooney. *Diverse ensembles for active learning*. International Conference on Machine Learning, pp. 584-591, 2004.

[Fritzke 95] B. Fritzke *A growing neural gas network learns topologies*. Advances in neural information processing systems, vol. 7, pp. 625-632, 1995.

[Martinetz 93] T. Martinetz *Competitive hebbian learning rule forms perfectly topology preserving maps*. International Conference on Artificial Neural Networks, pp. 427-434, 1993.

[Martinetz 91] T. Martinetz, K. Schulten *A "neural-gas" network learns topologies*. University of Illinois at Urbana-Champaign, pp. 397-402, 1991.

[Kohonen 90] T. Kohonen *The self-organizing map*. Proceedings of the IEEE, vol. 78, no 9, pp. 1464-1480, 1990.

[Prudent 05] Y. Prudent, A. Ennaji *An incremental growing neural gas learns topologies*. IEEE International Joint Conference Neural Networks, vol. 2, pp. 1211-1216, 2005.

[Furao 07] S. Furao, T. Ogura, O. Hasegawa *An enhanced self-organizing incremental neural network for online unsupervised learning*. IEEE International Joint Conference Neural Networks, vol. 20, no 8, pp. 893-903, 2007.

[Hamza 08] H. Hamza, Y. Belaïd, A. Belaïd, B.B. Chaudhuri *Incremental classification of invoice documents*. IEEE International Conference on Pattern Recognition, pp. 1-4, 2008.

[Bordes 05] A. Bordes, S. Ertekin, J. Weston, L. Bottou. *Fast Kernel Classifiers with Online and Active Learning*. Journal of Machine Learning Research, pp. 1579-1619, 2005.

[Jiang 07 (2)] L. Jiang, Z. Cai, D. Wang, S. Jiang *Survey of improving k-nearest-neighbor for classification*. Fuzzy Systems and Knowledge Discovery, pp. 679-683, 2007.

[Friedman 98] J. Friedman, T. Hastie, R. Tibshirani *Additive Logistic Regression : a Statistical View of Boosting*. Annals of Statistics, pp. 337-407, 1998.

[Lowd 05] D. Lowd, P. Domingos *Naive Bayes models for probability estimation*. International Conference on Machine Learning, pp. 529-536, 2005.

[Breiman 01] L. Breiman *Random Forests*. Machine learning, vol. 45, no 1, pp. 5-32, 2001.

[Gaber 07] M.M. Gaber, A. Zaslavsky, and S. Krishnaswamy. *Survey of classification methods in data streams*. Data Streams, Springer US, pp. 39-59, 2007.

[Aggarwal 07] M.M. Gaber, A. Zaslavsky, and S. Krishnaswamy. *Data streams : models and algorithms*. Springer, vol. 31, 2007.

[Zliobaite 11] I. Zliobaite, A. Bifet, B. Pfahringer, G. Holmes. *Active learning with evolving streaming data*. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, pp. 597-612, 2011.

[Dasgupta 05] S. Dasgupta *Coarse sample complexity bounds for active learning*. Neural Information Processing Systems, pp. 235-242, 2005.

[Scholkopf 01] B. Scholkopf, J. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. *Estimating the support of a high-dimensional distribution*. Neural Computation, pp. 1443-1471, 2001.

[Agarwal 05] D. Agarwal. *An Empirical Bayes Approach to Detect Anomalies in Dynamic Multidimensional Arrays*. International Conference on Data Mining, pp. 1-8, 2005.

[Spinosa 08] E.J. Spinosa, A.P. de Leon, F. de Carvalho, and J. Gama. *Cluster-Based Novel Concept Detection in Data Streams Applied to Intrusion Detection in Computer Networks*. Symposium On Applied Computing, pp. 976-980, 2008.

[He 07] J. He, and J. Carbonell. *Nearest-neighbor-based active learning for rare category detection*. Neural Information Processing Systems, pp. 633-640, 2007.

[Escudeiro 12] N.F. Escudeiro, and A.M. Jorge. *D-Confidence : an active learning strategy to reduce label disclosure complexity in the presence of imbalanced class distributions*. Journal of the Brazilian Computer Society, vol. 18, no 4, pp. 311-330, 2012.

[Masud 11] M.M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham. *Classification and novel class detection in concept-drifting data streams under time constraints*. IEEE Transactions on Knowledge and Data Engineering, vol. 23, no 6, pp. 859-874, 2011.

[Masud 10] M.M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham *Classification and novel class detection in data streams with active mining*. Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, pp. 311-324, 2010.

[Houle 10] M.E. Houle, H.P. Kriegel, P. Krger, E. Schubert, A. Zimek. *Can shared-neighbor distances defeat the curse of dimensionality ?*. Scientific and Statistical Database Management, Springer Berlin Heidelberg, pp. 482-500, 2010.

[Srivastava 06] A.N. Srivastava *Enabling the discovery of recurring anomalies in aerospace problem reports using high-dimensional clustering techniques*. IEEE Aerospace Conference, pp. 17-pp, 2006.

[Zhou 11] J. Zhou, Y. Fu, C. Sun, Y. Fang *Unsupervised distributed novelty detection on scientific simulation data*. Journal of Computational Information Systems, pp. 1533-1540, 2011.

[Ertoz 04] L. Ertöz, M. Steinbach, V. Kumar *Finding topics in collections of documents : A shared nearest neighbor approach*. Springer US, pp. 83-103, 2004.

[Hautamaki 04] V. Hautamaki, I. Karkkainen, P. Franti *Outlier Detection Using k-Nearest Neighbour Graph*. IEEE International Conference on Pattern Recognition, pp. 430-433, 2004.

[Angiulli 02] F. Angiulli, C. Pizzuti *Fast outlier detection in high dimensional spaces*. Principles of Data Mining and Knowledge Discovery, Springer Berlin Heidelberg, pp. 15-27, 2002.

[Knorr 98] E. Knorr and R. Ng *Algorithms for mining distance-based outliers in large datasets*. International Conference on Very Large Data Bases, pp. 392-403, 1998.

[Knorr 00] M. Breunig, H.P. Kriegel, R. Ng, J. Sander *LOF : identifying density-based local outliers*. ACM Special Interest Group on Management of Data, vol. 29, no 2, pp. 93-104, 2000.

[Tang 02] J. Tang, Z. Chen, A. Fu, D. Cheung *Enhancing effectiveness of outlier detections for low density patterns*. Advances in Knowledge Discovery and Data Mining, Springer Berlin Heidelberg, pp. 535-548, 2002.

[Yu 06] J. Yu, W. Qian, H. Lu, A. Zhou *Finding centric local outliers in categorical/numerical spaces*. Knowledge and Information Systems, pp. 309-338, 2006.

[Song 02] Q. Song, W. Hu, W. Xie *Robust support vector machine with bullet hole image classification*. IEEE Transactions on Systems, Man, and Cybernetics, pp. 440-448, 2002.

[Hu 03] W. Hu, Y. Liao, V. Vemuri *Robust anomaly detection using support vector machines*. International Conference on Machine Learning, pp. 282-289, 2003.

[Li 03] G. Li, C. Wen, Z. Li *A New Online Learning With Kernels Methods In novelty detection*. Proceedings of the 37th Annual Conference on IEEE Industrial Electronics Society (IECON), pp. 2311-2316, 2011.

[Scholkopf 99] B. Scholkopf, R. Williamson, A. Smola, J. Shawe-Taylor, J. Platt *Support Vector Method for Novelty Detection*. Neural Information Processing Systems, pp. 582-588, 1999.

[Manevitz 02] L. Manevitz, M. Yousef *One-class SVMs for document classification* Journal of Machine Learning Research, pp. 139-154, 2002.

[Tax 99] D. Tax, R. Duin *Support vector domain description*. Pattern recognition letters, pp. 1191-1199, 1999.

[Zhu 08] X. Zhu, et al. *Cleansing noisy data streams*. International Conference on Data Mining, pp. 584-591, 2008.

[Rebbapragada 12] U. Rebbapragada, C.E. Brodley, D. Sulla-Menashe, M. A. Friedl. *Active Label Correction*. International Conference on Data Mining, pp. 1080-1085, 2012.

[Fang 13] M. Fang, X. Zhu. *Active learning with uncertain labeling knowledge*. Pattern Recognition Letters, pp. 1080-1085, 2013.

[Tuia 13] D. Tuia, J. Munoz-Mari. *Learning User's Confidence for Active Learning*. IEEE Transactions on Geoscience and Remote Sensing, pp. 872-880, 2013.

[Sheng 08] V.S. Sheng, F. Provost, P.G. Ipeirotis. *Get another label ? improving data quality and data mining using multiple noisy labelers*. International conference on Knowledge Discovery and Data Mining, pp. 614-622, 2008.

[Ipeirotis 14] P.G. Ipeirotis, F. Provost, V.S. Sheng, J. Wang. *Repeated labeling using multiple noisy labelers*. International conference on Knowledge Discovery and Data Mining, pp. 402-441, 2014.

[Yan 11] Y. Yan, G.M. Fung, R. Rosales, J.G. Dy. *Active learning from crowds*. International Conference on Machine Learning, pp. 1161-1168, 2011.

[Rousseeuw 87] P.J. Rousseeuw. *Silhouettes : a Graphical Aid to the Interpretation and Validation of Cluster Analysis*. Computational and Applied Mathematics, pp. 53-65, 1987.

[Sun 87] S. Sun. *A survey of multi-view machine learning*. Neural Computing and Applications, pp. 2031-2038, 2013.

[Gamberger 96] D. Gamberger, N. Lavrac, S. Dzeroski. *Noise elimination in inductive concept learning : A case study in medical diagnosis*. Algorithmic Learning Theory, pp. 199-212, 1996.

[Hickey 96] R.J. Hickey *Noise modelling and evaluating learning from examples*. Artificial Intelligence, vol. 82, no 1, pp. 157-179, 1996.

[Bross 54] I. Bross *Misclassification in 2 $\times$ 2 tables*. Biometrics, pp. 478-486, 1954.

[Joseph 95] L. Joseph, T.W. Gyorkos, L. Coupal *Bayesian estimation of disease prevalence and the parameters of diagnostic tests in the absence of a gold standard*. American Journal of Epidemiology, vol. 141, no 3, pp. 263-272, 1995.

[Hadgu 96] A. Hadgu *The discrepancy in discrepant analysis*. The Lancet, vol. 348, no 9027, pp. 592-593, 1996.

[Zhu 04] X. Zhu, X. Wu, *Class noise vs. attribute noise : A quantitative study*. Artificial Intelligence Review, vol. 22, no 3, pp. 177-210, 2004.

[Krishnan 90] T. Krishnan, S.C. Nandy *Efficiency of discriminant analysis when initial samples are classified stochastically*. Pattern Recognition, vol. 23, no 5, pp. 529-537, 1990.

[Krishnan 01] N.D. Lawrence, B. Scholkopf *Estimating a kernel Fisher discriminant in the presence of label noise*. International Conference on Machine Learning, pp. 306-313, 2001.

[Yasui 04] Y. Yasui, M. Pepe, L. Hsu, B. Adam, Z. Feng *Partially supervised learning using an EM-boosting algorithm*. Biometrics, vol. 60, no 1, pp. 199-206, 2004.

[Malossini 06] A. Malossini, E. Blanzieri, R.T. Ng. *Detecting potential labeling errors in microarrays by data perturbation*. Bioinformatics, vol. 22, no 17, pp. 2114-2121, 2006.

[Quinlan 86] J.R. Quinlan *Induction of decision trees*. Machine Learning, vol. 1, no 1, pp. 81-106, 1986.

[Saez 14] J. Sàez, M. Galar, J. Luengo, and F. Herrera *Analyzing the presence of noise in multi-class problems : alleviating its influence with the One-vs-One decomposition*. Knowledge and information systems, vol. 38, no 1, pp. 179-206, 2014.

[Angluin 88(2)] D. Angluin, P. Laird *Learning from noisy examples*. Machine Learning, vol. 2, no 4, pp. 343-370, 1988.

[Smyth 94] P. Smyth, U. M. Fayyad, M. C. Burl, P. Perona, P. Baldi *Inferring ground truth from subjective labelling of venus images*. Advances in Neural Information Processing Systems, vol. 7, pp. 1085-1092, 1994.

[Snow 08] R. Snow, B. O'Connor, D. Jurafsky, A.Y. Ng *Cheap and fast But is it good ? Evaluating non-expert annotations for natural language tasks*. Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, pp. 254-263, 2008.

[Ipeirotis 10] P.G. Ipeirotis, F. Provost, J. Wang *Quality management on Amazon mechanical turk*. ACM Conference on Knowledge Discovery and Data Mining, workshop on human computation, pp. 64-67, 2010.

[Raykar 10] V.C. Raykar, S. Yu, L. H. Zhao, G. H. Valadez, C. Florin, L. Bogoni, et al. *Learning from crowds*. Journal of Machine Learning Research, vol. 11, pp. 1297-1322, 2010.

[Yuen 11] M.C. Yuen, I. King, K.S. Leung *A survey of crowdsourcing systems*. IEEE International Conference on Social Computing, pp. 766-773, 2011.

[Sloan 95] R.H. Sloan *Four types of noise in data for PAC learning*. Information Processing Letters, vol. 54, no 3, pp. 157-162, 1995.

[Aslam 96] J. A. Aslam *On the sample complexity of noise-tolerant learning*. Information Processing Letters, vol. 57, no 4, pp. 189-195, 1996.

[Kolcz 09] A. Kolcz G.V. Cormack *Genre-based decomposition of email class noise*. ACM Conference on Knowledge Discovery and Data Mining, pp. 427-436, 2009.

[Takenouchi 08] T. Takenouchi, S. Eguchi, N. Murata, T. Kanamori *Robust boosting algorithm against mislabeling in multiclass problems*. Neural Computation, vol. 20, no 6, pp. 1596-1630, 2008.

[Servedio 03] R. A. Servedio *Smooth boosting and learning with malicious noise*. Journal of Machine Learning Research, vol. 4, pp. 633-648, 2003.

[Biggio 11] B. Biggio, B. Nelson, P. Laskov *Support Vector Machines Under Adversarial Label Noise*. Journal of Machine Learning Research, pp. 97-112, 2011.

[Xiao 12] H. Xiao, H. Xiao, C. Eckert *Adversarial label flips attack on support vector machines*. European Conference on Artificial Intelligence, pp. 870-875, 2012.

[Klebanov 10] B. B. Klebanov, E. Beigman *Some empirical evidence for annotation noise in a benchmarked dataset*. Annual Conference of the North American Chapter of the Association for Computational Linguistics, pp. 438-446, 2010.

[Lugosi 92] G. Lugosi *Learning with an unreliable teacher*. Pattern Recognition, vol. 25, no 1, pp. 79-87, 1992.

[Brodley 99] C.E. Brodley, M.A. Friedl *Identifying mislabeled training data*. Journal of Artificial Intelligence Research, pp. 131-167, 1999.

[Sanchez 03] J. S. Sanchez, R. Barandela, A.I. Marques, R. Alejo, J. Badenas *Analysis of new techniques to obtain quality training sets*. Pattern Recognition Letters, vol. 24, no 7, pp. 1015-1022, 2003.

[Barandela 00] R. Barandela, E. Gasca *Decontamination of training samples for supervised pattern recognition methods*. Advances in Pattern Recognition, Springer Berlin Heidelberg, pp. 621-630, 2000.

[Muhlenbach 04] F. Muhlenbach, S. Lallich, D.A. Zighed *Identifying and handling mislabelled instances*. Journal of Intelligent Information Systems, vol. 22, no 1, pp. 89-109, 2004.

[Jiang 04] Y. Jiang, Z.H Zhou *Editing training data for k-nn classifiers with neural network ensemble*. Advances in Neural Networks, Springer Berlin Heidelberg, pp. 356-361, 2004.

[Sun 07] J.W. Sun, F.Y. Zhao, C.J. Wang, and S.F. Chen *Identifying and correcting mislabeled training instances*. Future Generation Communication and Networking, vol. 1, pp. 244-250, 2007.

[Thongkam 08] J. Thongkam, G. Xu, Y. Zhang, F. Huang *Support vector machine for outlier detection in breast cancer survivability prediction*. Advanced Web and NetworkTechnologies, Springer Berlin Heidelberg, pp. 99-109, 2008.

[Jeatrakul 10] P. Jeatrakul, K. W. Wong, C. C. Fung *Data cleaning for classification using misclassification analysis*. Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 14, no 3, pp. 297-302, 2010.

[Miranda 09] A. L. Miranda, L. P. Garcia, A. C. Carvalho, A. C. Lorena *Use of classification algorithms in noise detection and elimination*. Hybrid Artificial Intelligence Systems, pp. 417-424, 2009.

[Guyon 96] I. Guyon, N. Matic, V. Vapnik *Discovering informative patterns and data cleaning*. Advances in Knowledge Discovery and Data Mining, pp. 181-203, 1996.

[Brodley 96] C. E. Brodley, M. A. Friedl *Identifying and eliminating mislabeled training instances*. National Conference on Artificial Intelligence, vol. 1, pp. 799-805, 1996.

[Brodley 96(2)] C. E. Brodley, M. A. Friedl *Improving automated land cover mapping by identifying and eliminating mislabeled observations from training data*. Geoscience and Remote Sensing Symposium, pp. 27-31, 1996.

[Zhu 03 (2)] X. Zhu, X. Wu, Q. Chen. *Eliminating class noise in large datasets*. International Conference on Machine Learning, pp. 920-927, 2003.

[Venkataraman 04] S. Venkataraman, D. Metaxas, D. Fradkin, C. Kulikowski, I. Muchnik. *Distinguishing mislabeled data from correctly labeled data in classifier design*. IEEE International Conference on Tools with Artificial Intelligence, pp. 668-672, 2004.

[Wilson 97] D.R. Wilson, T.R. Martinez. *Instance pruning techniques*. International Conference on Machine Learning, pp. 403-411, 1997.

[Wilson 00] D.R. Wilson, T.R. Martinez *Reduction techniques for instance-based learning algorithms*. Machine Learning, vol. 38, no 3, pp. 257-286, 2000.

[Koplowitz 81] J. Koplowitz *On the relation of performance to editing in nearest neighbor rules*. Pattern Recognition, vol. 13, no 3, pp. 251-255, 1981.

[Hughes 04] N. P. Hughes, S. J. Roberts, L. Tarassenko *Semi-supervised learning of probabilistic models for ECG segmentation*. Annual International Conference of Engineering in Medicine and Biology Society, pp. 434-437, 2004.

[Cuendet 08] S. Cuendet, D. Hakkani-Tür, E. Shriberg *Automatic labeling inconsistencies detection and correction for sentence unit segmentation in conversational speech*. Machine Learning for Multimodal Interaction, Springer Berlin Heidelberg, pp. 144-155, 2008.

[Teng 00] C.M. Teng *Evaluating noise correction*. Topics in Artificial Intelligence, Springer Berlin Heidelberg, pp. 188-198, 2000.

[Teng 01] C.M. Teng *A comparison of noise handling techniques*. International Conference of the Florida Artificial Intelligence Research Society, pp. 269-273, 2001.

[Teng 05] C.M. Teng *Dealing with data corruption in remote sensing*. Advances in Intelligent Data Analysis, Springer Berlin Heidelberg, pp. 452-463, 2005.

[Seiffert 07] C. Seiffert, T. M. Khoshgoftaar, J. V. Hulse, A. Folleco *An empirical study of the classification performance of learners on imbalanced and noisy software quality data*. IEEE International Conference on Information Reuse and Integration, pp. 651-658, 2007.

[Matic 92] N. Matic, I. Guyon, L. Bottou, J. Denker, V. Vapnik *Computer aided cleaning of large databases for character recognition*. International Conference on Pattern Recognition Methodology and Systems, pp. 330-333, 1992.

[Breve 10] F. A. Breve, L. Zhao, M. G. Quiles *Semi-supervised learning from imperfect data through particle cooperation and competition*. International Joint Conference on Neural Networks, pp. 1-8, 2010.

[Bruzzone 11] L. Bruzzone, C. Persello *A novel context-sensitive semisupervised SVM classifier robust to mislabeled training samples*. IEEE Transactions on Geoscience and Remote Sensing, vol. 47, no 7, pp. 2142-2154, 2009.

[Guan 11] D. Guan, W. Yuan, Y.K. Lee, S. Lee *Identifying mislabeled training data with the aid of unlabeled data*. Applied Intelligence, vol. 35, no 3, pp. 345-358, 2011.

[Xiong 06] H. Xiong, G. Pandey, M. Steinbach, V. Kumar *Enhancing data analysis with noise removal*. IEEE Transactions on Knowledge and Data Engineering, vol. 18, no 3, pp. 304-319, 2006.

[Lukashevich 09] H. Lukashevich, S. Nowak, P. Dunker *Using one-class SVM outliers detection for verification of collaboratively tagged image training sets*. IEEE Transactions on Multimedia and Expo, vol. 18, no 3, pp. 682-685, 2009.

[Collett 76] D. Collett, T. Lewis *The subjective nature of outlier rejection procedures*. Applied Statistics, pp. 228-237, 1976.

[Liu 02] X. Liu, G. Cheng, J. X. Wu *Analyzing outliers cautiously*. IEEE Transactions on Knowledge and Data Engineering, vol. 14, no 2, pp. 432-437, 2002.

[Zhu 05] X. Zhu. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, 2005.