Fonctions d'influences pour la détection d'exemples mal étiquetés : une étude comparative

Mohamed Boukrani[*,**], Pierre Nodet[*], Vincent Lemaire[*]

[*]Orange Research, France
prenom.nom@orange.com
[**]ENSAI, Rennes

Résumé. En apprentissage automatique, la qualité d'un modèle d'apprentissage automatique dépend en grande partie de la qualité des données qui ont servi à son entraînement. Cependant, il est fréquent dans des cas pratiques de collecter des données de mauvaise qualité notamment des données mal étiquetées, ce qui dégrade la performance des modèles appris sur ces données. Parmi l'état de l'art permettant de mesurer la "qualité des exemples" (en termes d'étiquette) les fonctions d'influence sont des méthodes qui permettent de mesurer l'impact des observations d'entraînement sur les prédictions faites par le modèle, on parle "d'attribution des données d'entraînement". Dans cet article, pour des problèmes de classification, une étude comparative est menée pour différentes méthodes de calcul de fonctions d'influence. Elles sont comparées pour (i) identifier les exemples mal-étiquetés, (ii) filtrer ces exemples pour obtenir de meilleures performances d'apprentissage sur le modèle.

1 Introduction

La qualité des données constitue un enjeu fondamental dans le développement de modèles de machine learning performants. Parmi les défis majeurs rencontrés, la présence d'exemples mal étiquetés, ou bruités, peut considérablement dégrader la précision et la robustesse des modèles. La détection et la gestion de ces exemples "erronés" sont donc essentielles pour améliorer la performance des systèmes d'apprentissage automatique. Au fil des années, diverses approches ont été proposées pour identifier ces exemples problématiques et notamment Frenay et al. (Frénay et Verleysen, 2014) qui proposent une taxonomie des différents paramètres du bruit d'étiquetage. Devant des données à étiquettes potentiellement bruitée ou erronée trois solutions existent :

(i) la révision manuelle qui est une façon pragmatique de résoudre ce problème. Elle consiste à examiner manuellement les étiquettes afin d'identifier les étiquettes corrompues. Toutefois, cette tâche est très similaire à l'étiquetage manuel et elle est très coûteuse en temps et en ressources humaines ;

(ii) utiliser des algorithmes robustes, ce qui consiste à rechercher des algorithmes d'apprentissage qui intègrent une forte résistance aux étiquettes biaisées dans les ensembles de données et produisent des modèles de classification dont les performances peuvent être

relativement robustes à l'ajout de bruit. On peut citer, par exemple, des algorithmes (ou classifieurs) utilisant des fonctions de perte résistantes à certains types de bruits (Van Rooyen et al., 2015; Charoenphakdee et al., 2019; Le Baher et al., 2021) ;

(iii) enfin le filtrage et nettoyage automatique - au lieu de s'efforcer de rendre l'algorithme d'apprentissage robuste au bruit, on se concentre sur le nettoyage du jeu d'apprentissage pour utiliser sans crainte son algorithme d'apprentissage de prédilection. Le lecteur intéressé pourra parcourir la littérature des méthodes "centrées sur les données" (Pleiss et al., 2020; Northcutt et al., 2021).

Cet article se concentre sur la première étape de cette troisième approche, à savoir le filtrage. Parmi les méthodes de filtrage, l'utilisation de fonctions d'influence est particulièrement intéressant, car ces dernières permettent d'évaluer l'impact individuel de chaque exemple sur le modèle, notamment pour identifier d'éventuelles étiquettes erronées. Cependant, la comparaison de ces fonctions d'influence dans le contexte de la détection d'étiquettes incorrectes reste encore peu approfondie. Ainsi, cet article propose une analyse comparative de plusieurs fonctions d'influence employées pour la détection d'exemples mal étiquetés. En évaluant leurs performances sur différents jeux de données, cette étude vise à mieux comprendre leur comportement en fonction des caractéristiques du type de bruit rencontré.

2 Concepts utilisés

2.1 Fonction d'influence : Les fonctions d'influence font partie de la famille des méthodes de "Training Data Attribution (TDA)" (Cheng et al., 2025). Une fonction d'influence, en machine learning (Koh et Liang, 2017), est une mesure qui évalue l'impact qu'un exemple individuel dans un jeu de données a sur un modèle d'apprentissage. En d'autres termes, elle quantifie dans quelle mesure la présence ou la suppression d'un exemple spécifique modifie la performance ou les paramètres du modèle. Elle permet (i) d'identifier les exemples influents : déterminer quels exemples ont une influence significative sur le modèle, qu'ils soient bénéfiques ou nuisibles ; (ii) de détecter les données bruitées : repérer les exemples qui dégradent la performance du modèle, souvent en raison d'étiquettes incorrectes ou de bruit.

Les fonctions d'influence sont souvent calculées en analysant la sensibilité du modèle à la modification ou à la suppression d'un exemple. Selon Hammoudeh et Lowd (2024), les méthodes d'influence se répartissent en deux grandes catégories : "Retraining-Based " et "Gradient-Based". Dans le premier cas la méthode d'attribution idéale et la plus précise est la méthode Leave-One-Out (LOO). Elle permet de mesurer le vrai impact que peut avoir un point d'entraînement sur le modèle en se basant sur un réentraînement. Pour ce faire, elle consiste à entraîner le modèle deux fois. La première avec toutes les données d'entraînement et la deuxième fois avec toutes les données sauf celle pour laquelle on souhaite mesurer l'effet. Le lecteur comprendra aisément que cette méthode est beaucoup trop coûteuse et que donc il faudra passer par une autre voie. Dans le deuxième cas les méthodes estiment l'influence en utilisant des calculs et des comparaisons basés sur les gradients des exemples d'entraînement et de test, que ce soit pendant ou à la fin de l'entraînement. Celles-ci, estiment l'effet d'une modification infinitésimale des données d'entraînement sur les prédictions du modèle, sans nécessiter le réentraînement Bae et al. (2024). Par exemple, *"TracIn"* (Pruthi et al., 2020), *"Representer Point"* (Yeh et al., 2018), et enfin *"Fonctions d'influence"* (Koh et Liang, 2017).

Dans cet article on s'intéressera uniquement à la deuxième catégorie et à des fonctions d'influence inspirées de Koh et Liang (2017).

2.2 Bruit d'étiquetage : Le bruit d'étiquetage est un problème invasif lors de l'entraînement supervisé de modèles d'apprentissage automatique, en particulier sur les modèles les plus complexes (Algan et Ulusoy, 2020). Ce bruit reflète les imperfections inhérentes aux processus d'annotation, qu'ils soient humains (ex. fatigue, subjectivité) ou automatisés (ex. erreurs de règles heuristiques). Pour autant ces imperfections n'ont pas toutes les mêmes natures et les mêmes impacts et il est possible de catégoriser le bruit d'étiquetage en trois grandes familles (Frénay et Verleysen, 2014) :

— Noisy Completly At Random (NCAR) : appelé aussi *uniform noise*, ce type de bruit suppose que les étiquettes des individus sont inversées de manière totalement aléatoire, sans dépendance avec leurs caractéristiques ni avec leur vraie classe. Ce modèle est parfois aussi appelé *bruit symétrique*, car la probabilité qu'une classe soit inversée vers une autre est identique pour toutes les classes. Par exemple, une fleur pourrait être annotée à tort comme une voiture, sans lien entre les deux.
— Noisy At Random (NAR) : appelé également *class-dependent noise*, désigne un type de bruit où la probabilité que l'étiquette d'une instance soit corrompue dépend uniquement de sa vraie classe, sans tenir compte de ses caractéristiques ou de sa similarité avec d'autres instances. Par exemple, en considérant uniquement les classes "voiture" et "camion", en se basant uniquement sur les classes, la probabilité d'inverser les deux classes est de 30%. Cette estimation repose sur une connaissance générale des erreurs possibles, même si, dans la pratique, deux images de voitures ou de camions peuvent être très différentes visuellement.
— Noisy Not At Random (NNAR) : ou *feature-dependent noise*, ce type de bruit désigne le fait que les étiquettes sont inversées en se basant sur les informations contenues dans les variables explicatives de chaque individu. C'est le modèle de bruit le plus réaliste, notamment dans des cas où certains objets se ressemblent fortement. Par exemple, plusieurs "voitures sportives" ressemblent à des "voitures classiques", ce qui augmente la probabilité qu'elles soient mal étiquetées par des humains.

3 Fonctions d'influences étudiées

3.1 Introduction - Principe du calcul d'une fonction d'influence

Le but de cette section n'est pas de re-détailler les calculs clairs proposés dans (Koh et Liang, 2017) mais plutôt de donner les intuitions sur le calcul opéré et les différences entre les variantes testées plus bas dans cet article. Le but principal des fonctions d'influence est de mesurer l'influence des observations d'entraînement sur la performance des modèles. Lors de l'apprentissage du modèle f, un ensemble de données d'entraînement $\mathcal{D}_{\text{train}} := \{z_i = (x_i, y_i)\}_{i=1}^{n_{\text{train}}}$ est utilisé. On suppose que $\theta \in \mathbb{R}^d$ représente les paramètres du modèle paramètrique f_θ que l'on cherche à ajuster en minimisant le risque empirique :

$$\hat{\theta} := \arg\min_{\theta} \mathcal{R}(\theta) = \arg\min_{\theta} \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \mathcal{L}(z_i, f_\theta) \tag{1}$$

où $\mathcal{L}$ est la fonction de loss (ℓ_2 loss, cross-entropy,...).

On se pose alors la question suivante : Comment changera le comportement du modèle f_θ, pour un exemple de test (z_{test}), si un point d'apprentissage ($z_j = (x_j, y_j) \in \mathcal{D}_{\text{train}}$) est éliminé du jeu d'entraînement ? Pour cela on va regarder comment la fonction de loss décroit (ou augmente) à l'aide de la formule suivante :

$$\mathcal{I}(z_{\text{test}}, z_j) = \nabla_\theta \mathcal{L}(z_{\text{test}}, f_{\hat{\theta}})^T H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}}) \tag{2}$$

où (i) $H_{\hat{\theta}} = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \nabla_\theta^2 \mathcal{L}(z_i, f_{\hat{\theta}})$ est la matrice Hessienne moyenne des données d'apprentissage ; (ii) $\nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}})$ est le gradient de la fonction loss évaluée au point d'entraînement z_j ; (iii) $\nabla_\theta \mathcal{L}(z_{\text{test}}, f_{\hat{\theta}})$ est le gradient de la fonction loss évaluée au point z_{test} et (iv) $\mathcal{L}$ la fonction de loss.

En se basant sur l'équation 2 : si le point z_j est **positivement influent** sur la loss du point de test z_{test}, c'est-à-dire si $\mathcal{I}(z_{\text{test}}, z_j) > 0$, cela signifie que le fait de **retirer** le point z_j tend à **augmenter** la loss du modèle sur z_{test}. Inversement, si le point est **négativement influent** ($\mathcal{I}(z_{\text{test}}, z_j) < 0$), alors le retirer tend à **diminuer** la loss. À présent si on repart de l'équation 2 la littérature a proposé différentes versions de l'équation 2 que l'on propose d'étudier de manière expérimentale.

3.2 Les fonctions d'influences testées

3.2.1 Self-influence - La self-influence permet de mesurer l'influence qu'un exemple possède sur lui-même, en remplaçant z_{test} dans l'équation 2 par z_j :

$$\mathcal{I}(z_j, z_j) = \nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}})^T H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}}) \tag{3}$$

D'après Bejan et al. (2023), un exemple d'entraînement ayant un score de self-influence élevé, montre qu'il s'agit d'un point anormal et a un comportement différent des autres observations d'entraînement sur le modèle et donc doit être "étudié". Cette approche a déjà été utilisée pour la détection des exemples mal-étiquetés (Koh et Liang, 2017; Pruthi et al., 2020).

3.2.2 Influence sur la loss moyenne - Il s'agit ici de calculer l'influence de chaque exemple d'entraînement sur la loss moyenne d'un jeu de validation ou d'apprentissage. Au contraire de la self-influence qui cherche à mesurer à quel point un exemple a été appris par cœur par un modèle d'apprentissage, on cherche à mesurer ici si l'exemple a participé positivement ou négativement à l'apprentissage du modèle, ou même si l'exemple a permis la bonne généralisation du modèle sur un jeu de validation :

$$\mathcal{I}(\mathcal{D}_{\text{val}}, z_j) = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \left(\nabla_\theta \mathcal{L}(z_i, f_{\hat{\theta}})^T H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}}) \right) \tag{4}$$

$$\mathcal{I}(\mathcal{D}_{\text{train}}, z_j) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(\nabla_\theta \mathcal{L}(z_i, f_{\hat{\theta}})^T H_{\hat{\theta}}^{-1} \nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}}) \right) \tag{5}$$

3.2.3 Influence Relative - Barshan et al. (2020) ont proposé une version alternative des fonctions d'influence, appelée Relative Influence (RelatIF). Cette approche a été motivée par le constat que, dans leur forme classique, les fonctions d'influence mesurent l'impact des observations d'entraînement de manière globale, ce qui tend à faire ressortir systématiquement les

mêmes exemples (outliers, données mal étiquetées, points anormaux ; souvent ceux avec une loss élevée) comme étant les plus influents (e.g. sur la loss de l'ensemble $\mathcal{D}_{\text{val}}$). Pour remédier à ce problème, les auteurs suggèrent de mesurer l'influence des points d'entraînement relativement à leur influence globale sur le modèle, il s'agit en fait d'une **normalisation**. Pour cela, ils proposent d'imposer deux types de contraintes sur l'influence de ces points d'entraînement, ce qui conduit à deux variantes différentes des fonctions d'influence :

$\theta-$**RelatIF** - Cette première variante introduit une contrainte directe sur la variation des paramètres du modèle, en normalisant l'influence par la norme de cette variation :

$$\mathcal{I}_\theta(\mathcal{D}_{\text{val}}, z_j) = \frac{\mathcal{I}(\mathcal{D}_{\text{val}}, z_j)}{\|H_{\hat{\theta}}^{-1}\nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}})\|} = \frac{\nabla_\theta \mathcal{L}(\mathcal{D}_{\text{val}}, f_{\hat{\theta}})^T H_{\hat{\theta}}^{-1}\nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}})}{\|H_{\hat{\theta}}^{-1}\nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}})\|} \tag{6}$$

$\ell-$**RelatIF** - La seconde variante impose une contrainte sur la variation de la loss globale du modèle, en normalisant l'influence de l'exemple z_j sur $\mathcal{D}_{\text{val}}$ par l'influence globale de cet exemple sur lui-même (self-influence) :

$$\mathcal{I}_l(\mathcal{D}_{\text{val}}, z_j) = \frac{\mathcal{I}(\mathcal{D}_{\text{val}}, z_j)}{\sqrt{\mathcal{I}(z_j, z_j)}} \tag{7}$$

3.2.4 Influence sans Hessienne - Les fonctions d'influence nécessitent le calcul de la matrice Hessienne qui est coûteuse à calculer, il existe des approches (Pruthi et al., 2020) qui se basent uniquement sur le calcul des gradients et donc de la matrice Identité à la place de la matrice Hessienne. Dans ce cas, et à titre d'exemple, l'équation 5 devient :

$$\mathcal{I}(\mathcal{D}_{\text{val}}, z_j) = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \left(\nabla_\theta \mathcal{L}(z_i, f_{\hat{\theta}})^T \nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}})\right) \tag{8}$$

3.2.5 Influence de ré-étiqueter un point - Koh et Liang (2017) ont introduit une autre notion d'influence, qui consiste à évaluer l'impact de la perturbation des données. Plutôt que de se demander comment le modèle évolue suite à l'ajout ou la suppression d'un point, cette approche s'intéresse à la modification d'un point d'entraînement spécifique : Comment changera le modèle si on perturbe un point d'entraînement ? Aussi, et c'est une proposition de cet article, on s'intéressera à mesurer l'influence sur le modèle, résultante du changement de la classe (ré-étiquetage) d'une donnée $z_j = (x_j, y_j)$ vers toutes les autres, K, classes possibles $z_{\text{flip}} = (x_j, y_{\text{flip}})$, avec $y_{\text{flip}} \in [\![1, K]\!]$.

En suivant la même démarche de calcul que Koh et Liang (2017) l'influence de ré-étiqueter z_j vers z_{flip} sur la loss $\mathcal{L}$ moyenne de l'ensemble de validation est exprimée comme suit :

$$\mathcal{I}_{\text{flip}}(\mathcal{D}_{\text{val}}, z_j \to z_{\text{flip}}) = -\nabla_\theta \mathcal{L}(\mathcal{D}_{\text{val}}, f_{\hat{\theta}})^T H_{\hat{\theta}}^{-1} \left(\nabla_\theta \mathcal{L}(z_{\text{flip}}, f_{\hat{\theta}}) - \nabla_\theta \mathcal{L}(z_j, f_{\hat{\theta}})\right) \tag{9}$$

Cela donne jusqu'à K valeurs d'influence pour chaque observation d'entraînement, qu'il faut ensuite agréger. Nous avons choisi de prendre la valeur maximale parmi ces influences sauf celle de la classe initiale : $\mathcal{I}_{\text{flip}}(\mathcal{D}_{\text{val}}, z_j) = \max_{k \neq z_j} \ \mathcal{I}_{\text{flip}}(\mathcal{D}_{\text{val}}, z_j \to k)$.

3.2.6 Récapitulatif - Au final la liste des fonctions d'influence testées est donnée dans la Table 1. De plus, ces différentes définitions peuvent se conjuguer. Par exemple, on peut calculer une influence sur la loss moyenne en validation, ensuite normalisée avec l'influence θ-relative, le tout avec ou sans la Hessienne.

Influence	Formule
Self-influence	Équation 3
Influence sur la loss moyenne en validation	Équation 4
Influence sur la loss moyenne en apprentissage	Équation 5
Influence θ-relative	Équation 6
Influence l-relative	Équation 7
Influence sans la Hessienne	Équation 8
Influence de ré-étiqueter un point	Équation 9

TAB. 1 – Liste des fonctions influences testées

4 Protocole expérimental

4.1 Chaîne de traitement utilisée

La chaîne de traitement utilisée pour comparer les résultats qui seront décrits plus bas est présentée dans la figure 1. Le modèle entraîné à la première étape est appelé "modèle de base", il sera utilisé pour calculer les scores d'influence de tous les exemples d'entraînement, tandis que le modèle de la deuxième étape sera appelé "modèle final" et sera entraîné à l'aide des données filtrées jugées comme bien-étiquetées. Les deux modèles de chacune des étapes peuvent être différents, mais dans cet article ils sont identiques (voir Section 4.3).

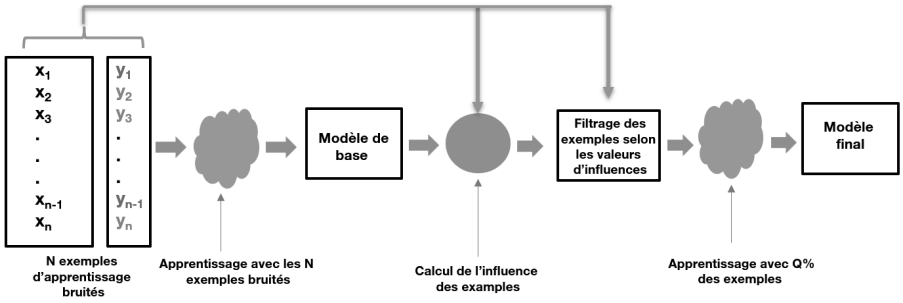

FIG. 1 – Pipeline de traitement proposé pour évaluer les différentes fonctions d'influence

La première étape consiste à entraîner le modèle de base puis à calculer la valeur s_i (score) d'une fonction d'influence pour chacun des exemples d'apprentissage (i). Ensuite, ces valeurs sont utilisées dans une phase de filtrage. Le modèle final est ensuite entraîné sur les exemples d'apprentissage filtrés et ses performances sont mesurées.

Les valeurs des hyperparamètres du modèle de base sont déterminées une seule fois par cross-validation, tandis que les valeurs des hyperparamètres du modèle final ainsi que le seuil de filtrage sont déterminées par cross-validation pour toutes les fonctions d'influence testées. Les détecteurs seront évalués à deux étapes différentes de ce processus. Ils seront premièrement évalués après le calcul des scores s_i en évaluant la qualité du tri induit en classant les exemples par valeur de score pour séparer les exemples bien étiquetés des mal-étiquetés. Puis ils seront évalués à la fin de ce pipeline en évaluant la performance du meilleur modèle final appris sur le jeu de test. Pour évaluer la qualité du modèle final, deux baselines seront utilisées à titre de

comparaison : *none* qui correspond au modèle appris sans filtrage, et *silver* qui correspond au modèle appris avec un filtrage parfait (les vraies étiquettes étant connues dans notre protocole).

4.2 Jeux de données et Bruits d'étiquetages

Les jeux de données utilisés pour l'étude comparative comprennent des jeux de données textuels et tabulaires issus de l'état de l'art en apprentissage faiblement supervisé (Zhang et al., 2021; Rühling Cachay et al., 2021; Boecking et al., 2021). Ils sont essentiellement axés sur des tâches de classification et couvrent des domaines diversifiés tels que les actualités, les vidéos, ainsi que des secteurs spécialisés comme les publications biomédicales et chimiques. Ils sont présentés dans la Table 2. Les jeux de données ainsi que leurs étiquettes propres et bruitées sont téléchargés via le package open source *mislabeled* George et al. (2024).

Dataset	n	d	K	$p(y)$	ρ
hausa	2.92K	750	5		50%
yoruba	1.91K	539	7		40%
professor	24.6K	4.06K	2		18%
bank-marketing	45.2K	78	2		26%
basketball	20.3K	2.05K	2		25%
bioresponse	3.75K	12.4K	2		46%
census	31.9K	125	2		19%
mushroom	8.12K	128	2		13%
phishing	11.1K	46	2		21%
spambase	4.6K	57	2		25%
tennis	8.8K	2.05K	2		13%
sms	5.57K	1.37K	2		3%
trec	6.00K	946	6		48%
youtube	2.06K	423	2		15%

TAB. 2 – Description globale des jeux de données utilisés. n est le nombre total de données, d nombre de variables, K nombre de classes, $p(y)$ la distribution des différentes classes, et ρ le taux de bruit.

Ces jeux de données sont particulièrement intéressants pour l'évaluation de détecteurs d'exemples mal-étiquetés, car en plus d'être des jeux de données publiques, qui étaient originellement relativement bien étiquetés, et dont les vraies étiquettes sont disponibles, la communauté a conçu différentes règles d'étiquetage pour générer des exemples mal-étiquetés particulièrement réalistes sur ceux-ci. Ces règles d'étiquetage sont des fonctions qui pour un exemple donné lui assignent soit une des classes soit s'abstiennent (Ratner et al., 2017). Par exemple, pour le jeu de données *sms*, si le SMS contient le mot *"free"*, une des règles va assigner la classe spam à ce SMS, si le mot n'est pas présent, la règle ne renvoie rien. On génère alors une étiquette, potentiellement bruitée en agrégeant par vote majoritaire ces règles d'étiquetage. Ainsi les étiquettes seront plus bruitées pour des exemples mal couverts par ces règles (des SMS avec des mots de vocabulaires atypiques), ou bien pour des exemples difficiles (avec une sémantique ambiguë), générant un bruit NNAR difficile à détecter.

En ce qui concerne le bruit de type NCAR, les étiquettes sont inversées de manière aléatoire, tout en conservant les mêmes pourcentages de bruit que dans le cas NNAR.

La séparation en jeu d'apprentissage, de validation et de test est pour la plupart du temps fournie avec les jeux de données, sinon, la séparation est faite de manière stratifiée pour respecter l'équilibre des classes dans chacun des trois jeux, avec une répartition de 80%, 10% et 10%. Les jeux de validation et de test ne contiennent pas de bruit d'étiquetage. Pour pouvoir calculer les fonctions d'influence sur un jeu de validation, celui-ci est encore une fois séparé en deux, une partie étant réservée au calcul des influences, l'autre pour la sélection d'hyperparamètres.

4.3 Prétraitements et classifieur

Les données vidéo sont fournies déjà prétraitées, avec une représentation prête à être exploitée par un modèle d'apprentissage. Les données textuelles ont été vectorisées à l'aide de la méthode TF-IDF (Spärck Jones, 1972) afin de convertir le texte en une représentation numérique exploitable par le modèle. Enfin les variables catégorielles dans les données tabulaires ont été one-hot encodées et les variables numériques standardisées. Pour les deux modèles (de base ou final), un classifieur linéaire est utilisé, optimisé itérativement en utilisant la **descente de gradient stochastique**. La régularisation ℓ_2 et le pas d'apprentissage sont réglés à l'aide de l'ensemble de validation (voir Section 4.1) par recherche aléatoire. La fonction de loss $\mathcal{L}$ choisie est l'*entropie croisée*. Comme les données ne sont pas toujours linéairement séparables, il a été décidé de les transformer d'abord à l'aide d'une approximation de noyau RBF pour générer 1000 composantes de Fourier basées sur (Rahimi et Recht, 2007). Ce processus projette les données dans un espace où elles deviennent plus faciles à séparer à l'aide d'un modèle linéaire. Le prétraitement des données et l'apprentissage des classifieurs sont réalisés à l'aide du package *scikit-learn* (Pedregosa et al., 2011).

5 Résultats

Nous chercherons au cours de la présentation des résultats à répondre à quatre questions pour chaque type de bruit (NNAR et NCAR). De manière à contrôler le nombre d'expériences à mener (et la taille de la présentation des résultats) les réponses aux quatre questions seront présentées à la manière de la résolution d'un problème a tiroir : la première réponse sert à répondre à la deuxième question et ainsi de suite. Les quatre questions sont : (i) Quel est le bon choix de l'échantillon sur lequel l'influence est mesurée ? (ii) Quelle est l'influence d'utiliser la matrice Hessienne versus la matrice Identité ? (iii) Quelle est la bonne technique de normalisation ? L'influence calculée sur le réétiquetage a-t-elle de bonnes performances ? Pour ces quatre questions nous évaluerons seulement la qualité de détection via l'aire sous la courbe roc (AUC) du score d'influence s_i pour discriminer les exemples bien et mal-étiquetés. Enfin, après avoir trouvé la meilleure combinaison de fonctions d'influence, nous présenterons les performances d'un modèle appris après le processus de filtrage des données utilisant cette même combinaison.

Choix de l'échantillon d'exemples Nous commençons par répondre à la question liée au choix de l'échantillon sur lequel on mesure l'influence des exemples d'entraînement : *self-influence* (Équation 3), influence sur la loss moyenne de l'ensemble d'*entraînement* et influence sur la loss moyenne de l'ensemble de *validation* (Équations 5 et 4).

Le tableau 3 présente une comparaison des méthodes d'influence (basées sur la matrice Hessienne) par jeu de données. On remarque que la détection est plus précise lorsque l'influence est calculée sur l'ensemble de validation dans le cas du bruit NNAR. En revanche, pour le bruit NCAR, la détection est plus efficace lorsqu'on calcule la self-influence.

Pour apprendre un exemple mal-étiqueté dans le cas NCAR, le modèle est obligé de l'apprendre par cœur et donc la self-influence qui mesure la mémorisation de l'exemple par le modèle est particulièrement efficace pour détecter de tels exemples. Ce qui est moins vrai pour le cas NNAR où il existe un signal propre aux exemples mal-étiquetés.

Dataset	NNAR			NCAR		
	self	*train*	*val*	*self*	*train*	*val*
bank-marketing	0,83	**0,90**	0,86	**0,92**	0,87	0,88
basketball	**0,84**	0,63	0,61	**0,98**	0,87	0,62
bioresponse	0,48	0,40	**0,62**	0,46	0,45	**0,63**
census	0,71	0,36	**0,80**	**0,84**	0,50	0,80
hausa	0,59	0,27	**0,70**	**0,84**	0,71	0,67
mushroom	0,83	0,76	**0,98**	**1,00**	0,50	0,97
phishing	0,77	0,70	**0,81**	**0,93**	0,56	0,88
professor	**0,85**	0,77	0,72	**0,95**	0,74	0,77
sms	0,62	0,29	**0,73**	**0,98**	0,88	0,76
spambase	0,70	0,64	**0,84**	0,83	0,55	**0,84**
tennis	**0,61**	0,53	0,52	**0,91**	0,59	0,63
trec	0,45	0,35	**0,61**	**0,73**	0,68	0,66
yoruba	0,44	0,30	**0,66**	**0,88**	0,81	0,64
youtube	**0,85**	0,64	0,63	**0,91**	0,49	0,67

TAB. 3 – Performance de détection (AUC) en fonction du type de bruit et de l'échantillon sur lequel l'influence est calculée.

Dataset	NNAR		NCAR	
	H	*I*	*H*	*I*
bank-marketing	0,86	**0,91**	0,92	**0,97**
basketball	0,61	**0,79**	0,98	**0,99**
bioresponse	0,62	**0,63**	0,46	**0,47**
census	**0,80**	0,76	0,84	**0,92**
hausa	0,70	**0,73**	**0,84**	0,74
mushroom	**0,98**	0,91	1,00	1,00
phishing	**0,81**	0,73	0,93	**0,98**
professor	0,72	**0,80**	**0,95**	0,93
sms	**0,73**	0,28	0,98	**1,00**
spambase	**0,84**	0,80	0,83	**0,94**
tennis	0,52	0,52	**0,91**	0,88
trec	0,61	**0,76**	**0,73**	0,69
yoruba	0,66	0,66	**0,88**	0,82
youtube	0,63	**0,83**	0,91	**0,95**

TAB. 4 – Performance de détection (AUC) selon le type de bruit et le choix entre la matrice Hessienne et l'Identité.

On remarquera que parfois l'AUC de détection est inférieure à 0,5 lorsqu'on regarde l'influence d'un exemple sur le jeu d'apprentissage. C'est particulièrement le cas pour les jeux de données les plus bruités dans le cas NNAR (*sms*, *hausa*, ...). Dans ce cas le signal prédominant dans le jeu de données n'est plus celui porté par les exemples bien étiquetés mais par ceux mal étiquetés et la fonction d'influence considère les exemples bien étiquetés comme mal étiquetés.

Choix d'utilisation de la matrice Hessienne La deuxième question est le choix entre l'utilisation de la matrice Hessienne ou de la matrice Identité pendant le calcul des influences. En nous basant sur la conclusion précédente (self-influence pour NCAR, validation pour NNAR), nous dressons le constat suivant. D'après le tableau 4, il apparaît que, généralement, l'utilisation de la matrice Hessienne permet d'obtenir des scores d'AUC plus élevés dans le cas NNAR. En revanche, pour le cas NCAR, la matrice Identité semble offrir de meilleures performances.

Choix de la méthode de normalisation La troisième question est le choix du type de normalisation à utiliser. En se basant sur la conclusion précédente, concernant le choix de l'échantillon pour chaque type de bruit, les résultats sont présentés dans les tableaux 5. On constate que, pour le cas de bruit NNAR, une bonne précision de détection est généralement atteinte lorsque aucune normalisation n'est appliquée. En revanche, pour le bruit NCAR, les normalisations $\ell-$RelatIF et $\theta-$RelatIF sont équivalentes à l'approche sans normalisation, et ces trois méthodes offrent des performances précises (le tableau de résultats est donc omis, les résultats étant similaires à ceux du tableau 4, colonne **NCAR**, *I*).

Choix de ré-étiqueter l'exemple Les résultats du protocole expérimental ont montré que pour les différents types de bruit, l'**influence classique** tend à surpasser celle du réétiquetage dans la détection des données mal étiquetées, avec des différences significatives notamment dans le cas du bruit de type NCAR. Cela ne remet pas en cause la performance de l'influence du réétiquetage dans la détection, mais pourrait être lié à la méthode utilisée pour l'agrégation des scores d'influence. Pour un gain de place les résultats détaillés sont donc omis.

Dataset	NNAR		
	Non-normalisé	**$\theta-$RelatIF**	**$\ell-$RelatIF**
bank-marketing	**0,86**	0,80	0,80
basketball	0,61	**0,63**	0,62
bioresponse	**0,62**	0,53	0,55
census	**0,80**	0,69	0,68
hausa	**0,70**	0,65	0,64
mushroom	**0,98**	**0,98**	**0,98**
phishing	**0,81**	0,75	0,74
professor	**0,72**	**0,72**	0,71
sms	0,73	0,80	**0,81**
spambase	**0,84**	0,79	0,80
tennis	0,52	**0,53**	0,50
trec	0,61	0,61	**0,63**
yoruba	0,66	0,66	**0,74**
youtube	0,63	**0,67**	**0,67**

TAB. 5 – Performance en AUC en fonction du type de normalisation utilisée via *l'ensemble de validation* et en utilisant la matrice *H*essienne – NNAR

Dataset	NNAR			NCAR		
	silver	*val, H*	*none*	*silver*	*self, I*	*none*
bank-marketing	0,69	0,64	0,69	0,70	**0,66**	0,59
basketball	0,75	**0,74**	0,50	0,86	**0,83**	0,64
bioresponse	0,50	**0,58**	0,50	0,71	**0,51**	0,42
census	0,54	**0,70**	0,57	0,75	**0,73**	0,69
hausa	0,44	**0,45**	0,24	0,75	**0,59**	0,57
mushroom	0,92	**0,93**	0,87	1,00	1,00	1,00
phishing	0,90	**0,90**	0,81	0,94	**0,93**	0,92
professor	0,90	**0,89**	0,86	0,91	**0,91**	0,89
sms	0,86	**0,86**	0,84	0,95	0,95	0,95
spambase	0,86	**0,85**	0,69	0,92	0,90	0,90
tennis	0,89	0,87	0,88	0,88	0,88	0,89
trec	0,46	**0,40**	0,23	0,76	0,64	0,65
yoruba	0,48	0,39	0,44	0,45	0,40	0,44
youtube	0,91	**0,90**	0,85	0,93	**0,91**	0,87

TAB. 6 – Balanced accuracy du modèle après le filtrage des exemples d'entraînement jugés mal-étiquetés

Performances après filtrage Pour le filtrage le processus utilisé est décrit dans la figure 1. Pour évaluer les classifieurs nous utilisons la précision équilibrée (PE, qui est la moyenne arithmétique du rappel pour chaque classe) sur l'ensemble de test, dont les valeurs sont données dans le tableau 6. Les tableaux présentent aussi les résultats des baselines *none* et *silver* telles que décrites en Section 4.1. Les valeurs en gras dans ces tableaux correspondent au cas où le filtrage a produit des classifieurs plus performants que la baseline *none*.

On constate que le filtrage réalisé pour le cas NNAR est plutôt efficace, les modèles appris après filtrage étant la plupart du temps significativement meilleurs que la baseline *none*. Alors que le bruit NCAR était supposé être plus simple, le filtrage avec les scores de self-influence se détache plus difficilement de la baseline *none*, peut-être dû au fait même qu'apprendre avec du bruit NCAR ne détériore pas beaucoup les modèles. Aussi dans George et al. (2024), il était souligné que les performances de détection étaient parfois peu corrélées à une bonne performance après filtrage.

6 Conclusion

Dans cet article nous nous sommes intéressés à la question de l'utilisation des fonctions d'influence pour la détection d'exemples mal étiquetés. Dans la littérature c'est la plupart du temps la self-influence qui est utilisée car elle mesure à quel point un modèle a appris par cœur un exemple d'apprentissage. Nous avons montré que cette métrique était certes pertinente pour les bruits d'étiquetage les plus simples (NCAR), mais qu'elle l'était moins pour du bruit plus réaliste (NNAR). Nous avons donc dans notre étude exploré les différentes possibilités de calculer les différentes fonctions d'influence proposées dans l'état de l'art et montré que l'utilisation d'un jeu de validation permettait de correctement détecter les exemples mal-étiquetés même dans les cas les plus difficiles. Bien que filtrer les exemples mal-étiquetés reste pertinent pour les traiter, dans des travaux futurs il serait intéressant de réutiliser l'idée des fonctions d'influence de réétiquetage pour corriger automatiquement (en leur assignant la

bonne étiquette) les exemples mal-étiquetés pour obtenir des modèles encore plus performants (Nodet et al., 2021).

Références

Algan, G. et I. Ulusoy (2020). Label noise types and their effects on deep learning. *arXiv preprint arXiv :2003.10471*.

Bae, J., W. Lin, J. Lorraine, et R. Grosse (2024). Training data attribution via approximate unrolling. In *Conference on Neural Information Processing Systems (NeurIPS)*, Red Hook, NY, USA. Curran Associates Inc.

Barshan, E., M.-E. Brunet, et G. K. Dziugaite (2020). Relatif : Identifying explanatory training samples via relative influence. In *International Conference on Artificial Intelligence and Statistics (ICML)*, pp. 1899–1909.

Bejan, I., A. Sokolov, et K. Filippova (2023). Make every example count : On the stability and utility of self-influence for learning from noisy nlp datasets. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Boecking, B., W. Neiswanger, E. P. Xing, et A. Dubrawski (2021). Interactive weak supervision : Learning useful heuristics for data labeling. In *International Conference on Learning Representations (ICLR)*.

Charoenphakdee, N., J. Lee, et M. Sugiyama (2019). On symmetric losses for learning from corrupted labels. In *International Conference on Machine Learning (ICML)*, Volume 97, pp. 961–970.

Cheng, D., J. Bae, J. Bullock, et D. Kristofferson (2025). Training data attribution (TDA) : Examining its adoption & use cases. *Convergence Analyis*.

Frénay, B. et M. Verleysen (2014). Classification in the presence of label noise : a survey. *IEEE Transactions on Neural Networks and Learning Systems (TNNLS) 25*(5).

George, T., P. Nodet, A. Bondu, et V. Lemaire (2024). Mislabeled examples detection viewed as probing machine learning models : concepts, survey and extensive benchmark. *Transactions on Machine Learning Research (TMLR)*.

Hammoudeh, Z. et D. Lowd (2024). Training data influence analysis and estimation : A survey. *Machine Learning (ML)*, 1–53.

Koh, P. W. et P. Liang (2017). Understanding black-box predictions via influence functions. In *International Conference on Machine Learning (ICML)*, pp. 1885–1894.

Le Baher, H., V. Lemaire, et R. Trinquart (2021). On the intrinsic robustness to noise of some leading classifiers and symmetric loss function – an empirical evaluation. In *Workshop "Data Quality Assessment for Machine Learning (DQAML)" SIGKDD 2021*. `https://arxiv.org/abs/2010.13570`.

Nodet, P., V. Lemaire, A. Bondu, A. Cornuéjols, et A. Ouorou (2021). From weakly supervised learning to biquality learning : an introduction. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. IEEE.

Northcutt, C., L. Jiang, et I. Chuang (2021). Confident learning : Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research (JAIR) 70*, 1373–1411.

Pedregosa, F., 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, et E. Duchesnay (2011). Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research (JMLR) 12*, 2825–2830.

Pleiss, G., T. Zhang, E. Elenberg, et K. Q. Weinberger (2020). Identifying mislabeled data using the area under the margin ranking. *Advances in Neural Information Processing Systems (NeurIPS) 33*, 17044–17056.

Pruthi, G., L. Frederick, K. Satyen, et S. Mukund (2020). Estimating training data influence by tracing gradient descent. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada.

Rahimi, A. et B. Recht (2007). Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems (NeurIPS)*.

Ratner, A., S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, et C. Ré (2017). Snorkel : Rapid training data creation with weak supervision. In *International conference on very large data bases (VLDB)*, Volume 11, pp. 269.

Rühling Cachay, S., B. Boecking, et A. Dubrawski (2021). End-to-end weak supervision. In *Advances in Neural Information Processing Systems (NeurIPS)*, Volume 34, pp. 3754–3766. Curran Associates, Inc.

Spärck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation 60*(5), 493–502.

Van Rooyen, B., A. Menon, et R. C. Williamson (2015). Learning with symmetric label noise : The importance of being unhinged. *Advances in neural information processing systems 28*.

Yeh, C.-K., J. Kim, I. E.-H. Yen, et P. K. Ravikumar (2018). Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, Volume 31.

Zhang, J., Y. Yu, Y. Li, Y. Wang, Y. Yang, et M. Yang (2021). Wrench : A comprehensive benchmark for weak supervision. In *Neural Information Processing Systems (NeurIPS), Track on Datasets and Benchmarks*.

Summary

In machine learning, the quality of a machine learning model depends largely on the quality of the data used to train it. However, in practical cases, it is often the case that poor-quality data is collected, particularly badly labelled data, which degrades the performance of the models trained on this data. Among the state-of-the-art methods for measuring the 'quality of examples' (in terms of labels) are influence functions, which measure the impact of training observations on the model's predictions, known as 'training data attribution'. In this article, a comparative study is conducted for classification problems using different methods of calculating influence functions. They are compared for (i) identifying mislabelled examples, (ii) filtering these examples to obtain better performance.