Biquality learning : from weakly supervised learning to distribution shifts

Pierre Nodet

université
PARIS-SACLAY

Biquality Learning: from Weakly Supervised Learning to Distribution Shifts

Apprentissage biqualité: de l'apprentissage faiblement supervisé aux décalages de distribution

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°580 : sciences et technologies de l'information et de la communication (STIC)
Spécialité de doctorat: Informatique
Graduate School : Informatique et sciences du numérique
Référent : Faculté des sciences d'Orsay

Thèse préparée dans l'unité de recherche UMR MIA Paris-Saclay (Université Paris-Saclay, AgroParisTech, INRAE), sous la direction de Antoine CORNUÉJOLS, Professeur en Informatique, la co-encadrement de Vincent LEMAIRE, Research Scientist, le co-encadrement de Alexis Bondu, Research Scientist

Thèse soutenue à Paris-Saclay, le 12 avril 2023, par

Pierre NODET

Composition du jury

Membres du jury avec voix délibérative

Alexandre Gramfort Research Scientist, PhD & HDR Meta AI	Président
Albert Bifet Professeur en Informatique Université de Waikato et Télécom Paris	Rapporteur & Examinateur
Pascale Kuntz Professeure en Informatique Ecole Polytechnique de l'Université de Nantes	Rapportrice & Examinatrice
Mathilde Mougeot Professeure en Science des données Université Paris-Saclay, ENSIIE	Examinatrice
Charlotte Pelletier Maitre de Conférence en Informatique Université Bretagne Sud	Examinatrice

THESE DE DOCTORAT
NNT : 2023UPASG030

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre: Apprentissage biqualité: de l'apprentissage faiblement supervisé aux décalages de distribution
Mots clés: apprentissage faiblement supervisé, décalage de jeu de données, apprentissage biqualité

Résumé: Le domaine de l'apprentissage avec des faiblesses en supervision est appelé apprentissage faiblement supervisé et regroupe une variété de situations où la vérité terrain collectée est imparfaite. Les étiquettes collectées peuvent souffrir de mauvaise qualité, de non-adaptabilité ou de quantité insuffisante. Dans ce mémoire nous proposons une nouvelle taxonomie de l'apprentissage faiblement supervisé sous la forme d'un cube continu appelé le cube de la supervision faible qui englobe toutes les faiblesses en supervision. Pour concevoir des algorithmes capables de gérer toutes supervisions faibles, nous supposons la disponibilité d'un petit ensemble de données de confiance, sans biais ni corruption, en plus de l'ensemble de données potentiellement corrompu. L'ensemble de données de confiance permet de définir un cadre de travail formel appelé apprentissage biqualité. Nous avons examiné l'état de l'art de ces algorithmes qui supposent la disponibilité d'un petit jeu de données de confiance. Dans ce cadre, nous proposons un algorithme basé sur la repondération préférentielle pour l'apprentissage biqualité (IRBL). Cette approche agnostique du classificateur est basée sur l'estimation empirique de la dérivée de Radon-Nikodym (RND), pour apprendre un estimateur conforme au risque sur des données non fiables repesées. Nous étendrons ensuite le cadre proposé aux décalages de jeu de données. Les décalages de jeu de données se produisent lorsque la distribution des données observée au moment de l'apprentissage est différente de celle attendue au moment de la prédiction. Nous proposons alors une version améliorée d'IRBL, appelée IRBL2, capable de gérer de tels décalages de jeux de données. Nous proposons aussi KPDR basé sur le même fondement théorique mais axé sur le décalage de covariable plutôt que le bruit des étiquettes. Pour diffuser et démocratiser le cadre de l'apprentissage biqualité, nous rendons ouvert le code source d'une bibliothèque Python à la Scikit-Learn pour l'apprentissage biqualité : biquality-learn.

Title: Biquality Learning: from Weakly Supervised Learning to Distribution Shifts
Keywords: Weakly Supervised Learning, Distribution Shift, Biquality Learning

Abstract: The field of Learning with weak supervision is called Weakly Supervised Learning and aggregates a variety of situations where the collected ground truth is imperfect. The collected labels may suffer from bad quality, non-adaptability, or insufficient quantity. In this report, we propose a novel taxonomy of Weakly Supervised Learning as a continuous cube called the Weak Supervision Cube that encompasses all of the weaknesses of supervision. To design algorithms capable of handling any weak supervisions, we suppose the availability of a small trusted dataset, without bias and corruption, in addition to the potentially corrupted dataset. The trusted dataset allows the definition of a generic learning framework named Biquality Learning. We review the state-of-the-art of these algorithms that assumed the availability of a small trusted dataset. Under this framework, we propose an algorithm based on Importance Reweighting for Biquality Learning (IRBL). This classifier-agnostic approach is based on the empirical estimation of the Radon-Nikodym derivative (RND), to build a risk-consistent estimator on reweighted untrusted data. Then we extend the proposed framework to dataset shifts. Dataset shifts happen when the data distribution observed at training time is different from what is expected from the data distribution at testing time. So we propose an improved version of IRBL named IRBL2, capable of handling such dataset shifts. Additionally, we propose another algorithm named KPDR based on the same theory but focused on covariate shift instead of the label noise formulation. To diffuse and democratize the Biquality Learning Framework, we release an open-source Python library à la Scikit-Learn for Biquality Learning named biquality-learn.

Synthèse

L'apprentissage supervisé est le domaine d'étude dédié à rendre les machines capables d'apprendre des tâches à partir d'exemples étiquetés. Ce paradigme repose sur l'apprentissage statistique d'une fonction de correspondance entre l'espace des exemples et l'espace des étiquettes, via des paires d'exemples et d'étiquettes. Dans ce paradigme, la disponibilité d'une quantité de paires suffisante, et la véracité des étiquettes présentes, sont nécessaires pour que les machines puissent correctement généraliser sur des exemples différents de ceux vu à l'apprentissage. Si par exemple les étiquettes à disposition ne représentent pas la vérité terrain, alors les machines n'arriveront pas à généraliser et surapprendront les motifs bruités.

Le domaine de l'apprentissage avec des faiblesses en supervision est appelé apprentissage faiblement supervisé et regroupe une variété de situations où la vérité terrain collectée est imparfaite. Les étiquettes collectées peuvent souffrir de mauvaise qualité, de non-adaptabilité ou de quantité insuffisante. Hors, les taxonomies existentes de l'apprentissage faiblement supervisées sont segmentées, limitant les interactions entre ses sous-domaines, correspondants à chacune des faiblesses en supervision. Dans ce mémoire nous proposons une nouvelle taxonomie de l'apprentissage faiblement supervisé sous la forme d'un cube continu appelé le cube de la supervision faible qui englobe toutes les faiblesses en supervision. Néanmoins, aucune solution existante ne permet de résoudre cette catégorie continue de tâches d'apprentissage automatique.

Pour concevoir des algorithmes capables de gérer toutes supervisions faibles, nous supposons la disponibilité d'un petit ensemble de données de confiance, sans biais ni corruption, en plus de l'ensemble de données potentiellement corrompu. L'ensemble de données de confiance permet de définir un cadre de travail formel appelé apprentissage biqualité. Nous avons examiné l'état de l'art de ces algorithmes qui supposent la disponibilité d'un petit jeu de données de confiance et proposons de l'organiser par leur manière de corriger les données non-fiables, notamment par repondération, correction des étiquettes, ou adaptation des attributs.

Dans ce cadre, nous proposons un algorithme basé sur la repondération préférentielle pour l'apprentissage biqualité (IRBL). Cette approche agnostique du classificateur est basée sur l'estimation empirique de la dérivée de Radon-Nikodym (RND), pour apprendre un estimateur conforme au risque sur des données non fiables repesées. Nous avons comparé empiriquement l'approche proposée sur un grand nombre de jeux de données variés montrant de manière significative la performance de celle-ci. Nous avons aussi réalisé une étude sur l'impact de la calibration et de l'expressivité du classifieur sur IRBL.

Nous étendons ensuite le cadre proposé aux décalages de jeu de données. Les décalages de jeu de données se produisent lorsque la distribution des données observée au moment de l'apprentissage est différente de celle attendue au moment de la prédiction. Ce changement de distribution peut prendre plusieurs formes, comme un changement dans la distribution d'un seul attribut, d'une combinaison d'attributs, ou bien dans le concept à apprendre. Ainsi, l'hypothèse selon laquelle les données d'entraînement et de test suivent les mêmes distributions est souvent rejetée dans les applications réelles. Nous croyons que le cadre de l'apprentissage biqualité est un cadre approprié pour concevoir des algorithmes capables de gérer à la fois les décalages de jeu de données et les faiblesses en supervision. Nous avons donc proposé une version

améliorée d'IRBL, appelée IRBL2, capable de gérer de tels décalages de jeux de données. Nous proposons aussi KPDR basé sur le même fondement théorique mais axé sur le décalage de covariable plutôt que sur le bruit des étiquettes. Grâce à une comparaison étendues aux décalages de distributions, nous avons montrés que KPDR superforme de manière significative par rapport aux compétiteurs de l'état de l'art.

Pour diffuser et démocratiser le cadre de l'apprentissage biqualité, nous rendons ouvert le code source d'une bibliothèque Python à la scikit-learn pour l'apprentissage biqualité. biquality-learn vise à rendre accessibles les algorithmes biqualité les plus connus et éprouvés à tous, et à aider les chercheurs à expérimenter de manière reproductible sur des données biqualitées. Elle a été spécialement conçue pour s'intégrer parfaitement à scikit-learn et à d'autres packages basés sur la même API.

Remerciements

Tout d'abord, je tiens à exprimer ma profonde gratitude envers mes encadrants de thèse, le docteur Vincent Lemaire et le docteur Alexis Bondu, ainsi que mon directeur de thèse, le Professeur Antoine Cornuéjols. Leurs enseignements, conseils et discussions ont été d'une valeur inestimable tout au long de ce parcours de recherche. Leur soutien constant et leurs précieuses orientations ont été les piliers qui ont permis la réalisation de ces travaux. Leurs contributions ont éclairé mon chemin académique, stimulé ma réflexion et m'ont poussé à repousser les limites de mes connaissances. Leurs retours constructifs m'ont permis de perfectionner mon travail. Sans leur expertise, leur disponibilité et leur bienveillance, cette thèse n'aurait pu aboutir de la sorte. Je suis honoré d'avoir eu l'opportunité de travailler à leurs côtés, et je leur suis infiniment reconnaissant pour leur contribution exceptionnelle à mon développement académique et professionnel.

J'adresse tous mes remerciements au Professeur Albert Bifet, ainsi qu'à la Professeure Pascale Kuntz, d'avoir accepté d'être rapporteurs de cette thèse et d'avoir pris le temps de relire ce manuscrit. J'exprime aussi ma gratitude au docteur Alexandre Gramfort, à la Professeure Mathilde Mougeot et à la Maitre de Conférence Charlotte Pelletier qui ont bien voulu être mes examinateurs. Je remercie tous les membres du jury pour leurs remarques pertinentes et leur enthousiasme quant à mes travaux.

Je tiens aussi à remercier l'équipe BRAIN et PROF d'Orange Labs et l'équipe Ekinocs de l'INRAe pour leur bienveillance et leur soutien qui m'ont permis de pleinement travailler sur ma thèse et de grandir en tant que personne, mais aussi pour tous les échanges enrichissants qui m'ont toujours poussé à aller plus loin dans ma démarche d'apprenti chercheur.

Je souhaite remercier chaleureseument les trois autres fantastiques Antoine, Dinesh et Romain pour les superbes soirées d'été mais aussi Jérémy, Sylvain et Matthias qui m'ont encouragé de tout coeur à commencer cette aventure depuis la Chine. Merci aussi aux joyeux lurons, Samir, Frédéric, Jules et Henri pour ces rigolades jusqu'au bout de la nuit. Par tous ces moments passé ensemble, vous avez tous contribué à la réussite de cette thèse.

Je remercie du fond du coeur Laure la méritante, qui aura vécu cette thèse à mes côtés avec ses hauts et ses bas, mais qui, part son soutien indéfectible, m'a permis de garder le cap pendant ces trois années de thèse.

Enfin il m'est impossible pour moi de terminer ces remerciements sans remercier mes parents et ma soeur qui sont à l'origine de tout ce qui je suis et et de tout ce que j'accomplis. Je remercie leur amour et leurs encouragements sans lesquels je n'aurais jamais pu réaliser cette thèse.

Merci à vous tous.

Contents

List of Figures

List of Tables

Introduction

Context

Supervised Learning is the field of study dedicated to making machine learn tasks from labeled examples. This paradigm is based on the statistical learning of a mapping function from the input space to the label space, thanks to pairs of samples and labels. It seeks to learn a mapping function that can generalize to new data thanks to a massive amount of training data and the correctness of the label associated with each sample. For example, when training a machine to classify images of dogs and cats, we have to provide the machine with pictures of dogs and cats, and for every image, the label tells the machine if the picture is one of a dog or one of a cat. By learning by heart on a massive amount of perfectly labeled images, the machine can correctly guess the nature of the animal on completely new images with better accuracy than humans.

In real-world scenarios, the labels' quality could be imperfect for multiple reasons, such as human errors, cyber-attacks on databases, or even a lack of resources to annotate datasets. With our previous example of machine learning to classify images of dogs and cats, achieving better accuracy than humans will require a vast amount of annotated images which is many hours of repetitive human labor. That is why practitioners resorts to using weaker way to annotate samples, such as using Amazon Mechanical Turk.

Training classic supervised learning algorithms with poorly labeled data will considerably reduce their accuracy as they tend to focus more and more on more challenging examples during their training process. These curriculums lead to better generalization with perfectly labeled data, but with poorly labeled data, they lead to machines that overfit on noisy patterns.

This situation could be even direr when only annotated pictures of wolfs and tigers are the only source of training data available to learn an efficient classifier of dogs and cats, or when only pictures of prehistorical dogs and cats are available.

These real-world scenarios are pretty diversified and could happen simultaneously, making learning from weak supervision quite a vast field of study. This thesis aims to determine if the design of a training algorithm that could automatically adapt to all these constraints is possible.

Motivation

The field of Learning with weak supervision is called Weakly Supervised Learning and aggregates a variety of situations where the collected ground truth is imperfect. The collected labels may suffer from bad quality, non-adaptability, or insufficient quantity. We propose a novel taxonomy of Weakly Supervised Learning

as a continuous cube called the Weak Supervision Cube that encompasses all of the weaknesses of supervision. However, an existing solution is needed to solve this continuous category of machine learning tasks.

To design algorithms capable of handling weak supervisions, with robustness to varying label quality, we suppose the availability of a small trusted dataset, without bias and corruption, in addition to the potentially corrupted and untrusted dataset. The trusted dataset allows the definition of a generic learning framework named Biquality Learning that we define in Chapter 2. We reviewed the state-of-the-art of these algorithms that supposed the availability of a small trusted dataset, and we proposed to organize them by leveraging this dataset to correct and refine the corrupted samples.

Under this framework, we proposed an algorithm based on Importance Reweighting for Biquality Learning (IRBL). This classifier-agnostic approach is based on the empirical estimation of the Radon-Nikodym derivative (RND), which is the sample reweighting scheme to build a risk-consistent estimator on untrusted data. We estimate this RND using two probabilistic classifiers that model both the trusted and untrusted concepts. Finally, we build a final classifier using trusted and reweighted untrusted data. We extensively benchmarked the proposed approach against state-of-the-art algorithms on various datasets and statistically showed that it beats competitors on Biquality Tasks. We conducted ablation studies to empirically evaluate the impact of the calibration and expressiveness of the classifier used on the final performance of IRBL.

Then we extended the proposed framework to dataset shifts. Dataset shifts happen when the data distribution observed at training time is different from what is expected from the data distribution at testing time. This distribution change can take multiple forms, such as a change in the distribution of a single feature, a combination of features, or the concept to be learned. Thus, the common assumption that the training and testing data follow the same distributions is often violated in real-world applications. We believe that the biquality data setup is a suitable framework to design algorithms capable of handling both dataset shifts and weaknesses of supervision simultaneously. So we proposed an improved version of IRBL named IRBL2, capable of handling such dataset shifts. Additionally, we proposed another algorithm named KPDR based on the same RND but focused on covariate shift instead of the label noise formulation. We conducted the same exhaustive benchmarks and found that KPDR was significantly outperforming the state-of-the-art on various datasets shifts and weaknesses of supervision.

Finally, we explored the impact of training Deep Neural Networks (DNN) on untrusted data, especially on the learned representation of the features. Indeed training DNN is fundamentally different from training shallow classifiers, as in addition to learning the classification task, the representation learning task is added. The representation learning part is at high risk when learning on noisy data, and we review the state-of-the-art of method to preserve the representation while learning

on noisy labels. Finally, we studied the impact of these robust algorithms on the representation learned thanks to samples only named Self Supervised Learning.

To diffuse and democratize the Biquality Learning Framework, we proposed a Python library à la Scikit-Learn for Biquality Learning. **biquality-learn** aims to make accessible well-known and proven biquality learning algorithms for everyone and help researchers experiment in a reproducible way on biquality data. It was specially designed to integrate perfectly with Scikit-Learn and other packages based on the same API.

Contributions

Chapter 1 explains in detail the motivations of the thesis. This Chapter presents the state-of-the-art of Weakly Supervised Learning in its current organization. We highlight some practical and philosophical limits to this organization. We then propose a new continuous organization of this state of the art through the cube of weakly supervised Learning, allowing us to go beyond the dimensions studied until now. We introduce the Biquality Data setup (training machine learning from a trusted and untrusted dataset) and Biquality Learning as one of the faces of this cube. Finally we explain why we think the Biquality Data setup is a suitable framework for designing algorithms able to handle any weaknesses of supervision. This introduction has been published at an international conference: Pierre Nodet, Vincent Lemaire, Alexis Bondu, Antoine Cornuéjols, and Adam Ouorou. From Weakly Supervised Learning to Biquality Learning: an Introduction. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

Chapter 2 is a position chapter of Biquality Learning. We propose a generic framework for Biquality Learning, allowing us to define formally the learning task to be accomplished. Then we present a state of the art of Biquality Learning, organized by their respective underlying intuitions. Finally, an experimental protocol for evaluating Biquality learning algorithms is proposed, using public Biquality datasets, or proposing several synthetic corruptions models for usual supervised classification datasets.

Chapter 3 proposes a Biquality Learning algorithm called Importance Reweighting for Biquality Learning (IRBL). It is based on the estimation of the Radon-Nykodym derivative (RND) [131] of the trusted concept with respect to the untrusted concept. This scheme is the theoretical reweighting scheme to correct the untrusted dataset for a trusted dataset. We propose an estimation of the RND thanks to two probabilistic classifiers learned on both datasets respectively. Extensive experiments, as described in Chapter 2, demonstrate that the proposed approach outperforms baselines and state-of-the-art approaches. This Chapter was published at an international conference: Pierre Nodet, Vincent Lemaire, Alexis Bondu, Antoine Cornuejols, and Adam Ouorou. Importance reweighting for biquality learning. In *International Joint Conference on Neural Networks (IJCNN)*.

IEEE, 2021.

Chapter 4 extends our previous work to a broader definition of Biquality Learning including dataset shifts. Previously, Biquality Learning only considered supervisory weaknesses by denoting a difference in concept between the two untrusted and trusted distributions. This Chapter extends the previous work to a difference in joint distribution between the two distributions, named distributions shifts. In this framework, we propose a new version of IRBL, IRBL2 inspired from the concept drift litterature, and an alternative instantiation of IRBL2 based on the work of [49] named K-ProbabilisticDensityRatio (KPDR), inspired from the covariate shift litterature. This Chapter has been submitted to an international machine learning journal.

Chapter 5 explores Biquality Learning of deep neural networks, particularly the difficulty of learning deep data representations from weak supervisions. In particular, this Chapter discusses the use of pre-trained unsupervised representation and raises the question of the efficiency of Biuality learning algorithms (and learning from noisy labels algorithms) to improve this representation. This Chapter also describes how these algorithms attempt to preserve this deep representation from weaknesses of supervisions. This Chapter has been published at an international conference workshop: Pierre Nodet, Vincent Lemaire, Alexis Bondu, and Antoine Cornuéjols. Contrastive representations for label noise require fine-tuning. In Georg Krempl, Vincent Lemaire, Daniel Kottke, Andreas Holzinger, and Barbara Hammer, editors, *Proceedings of the ECML Workshop on Interactive Adaptive Learning (IAL@ECML PKDD 2021)*, number 3079 in CEUR Workshop Proceedings, pages 89–104, Aachen, 2021.

Finally, Chapter 6 presents a Python code library for Biquality Learning following the scikit-learn APIs [140], implementing the most recognized algorithms of the domain for better reproducibility of the experiments for the researchers and immediate handling for the engineers. It also provides tools to generate corruptions in datasets to simulate Biquality datasets synthetically.

1 - Weakly Supervised Learning

Contents

This Chapter has been published at an international conference by the author: Pierre Nodet, Vincent Lemaire, Alexis Bondu, Antoine Cornuéjols, and Adam Ouorou. From Weakly Supervised Learning to Biquality Learning: an Introduction. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

1.1 . Introduction

In the field of machine learning, the task of classification can be performed by different approaches depending on the level of supervision of training data. As shown in Figure 1.1, unsupervised, weakly supervised, and supervised approaches form a continuum of possible situations, starting from the absence of ground truth and ending with complete and perfect ground truth. For the most part, the accuracy of the models learned increases as the level of supervision of data increases. Additionally, the level of supervision of a dataset can be increased in return for a labeling cost. In [166], the authors indicate that an interesting goal could be to obtain a high accuracy while spending a low labeling cost.

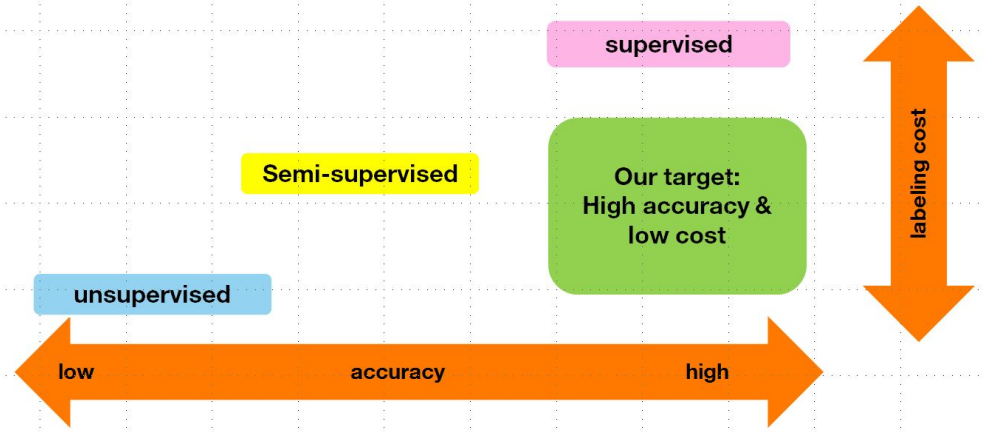


Figure 1.1: Classification of Classification from [166]

In *Weakly Supervised Learning* (WSL) use cases (e.g., fraud detection, cybersecurity, crowd-source labelling), a variety of situations exists where the collected ground truth is imperfect. In this context, the collected labels may suffer from bad quality, non-adaptability (defined in Section 1.6), or insufficient quantity. For instance, an automatic labeling system could be used without guaranteeing that the data is complete, exhaustive, and trustworthy. Alternatively, manual labeling is also problematic in practice as obtaining labels from an expert is costly, and the availability of experts is often limited. Consequently, there are many real-life situations where imperfect ground truth must be used because of some practical considerations, such as cost optimization, expert availability, and difficulty to certainly choose each label.

This general problem of supervision deficiency has attracted a recent focus in the literature. The paradigm of *Weakly Supervised Learning* attempts to list and cover these problems with associated solutions. The work of Zhou in [224] is one of the first successful efforts to synthesize this domain. However, the proposed organization is segmented, limiting interactions between different subdomains of WSL, and requiring knowledge on the weakness of supervision that might be unavailable

for the practitioner. Thus we propose a new organization of *Weakly Supervised Learning* as a continuous cube that allows for a more encompassing interpretation of the literature.

Section 1.2 proposes to review the usual *Weakly-Supervised Learning* state-of-the-art organized hierarchically in a Tree. Then Section 1.3 proposed a new organization of the *Weakly-Supervised Learning* literature named the Weak Supervision Cube. Section 1.4 will go through the third axis of the Weak Supervision Cube and review the most notable subdomains of which it is composed. Section 1.5 gives additional elements that must be considered at the crossroad of these three axes or when dealing with Weak learning problems. Section 1.6 suggests three measurable quantities which help summarize WSL: Quantity, Quality, and Adaptability. In Section 1.7, these quantities are used to raise links between some learning frameworks jointly used in WSL as in Biquality Learning.

1.2 . The Weak Supervision Tree

Supervision deficiencies in weakly supervised learning refer to the lack of sufficient or accurate labeling in the training data. In weakly supervised learning, a model is trained on a dataset that has only been partially labeled, or has been labeled in a vague or imprecise way. This can make it difficult for the model to learn effectively and lead to poor performance on the task it is being trained for. Various factors, such as limited resources for labeling data or the inherent ambiguity of the learned task, can cause supervision deficiencies in weakly supervised learning. To overcome these deficiencies, weakly supervised learning techniques often rely on additional assumptions or constraints to help the model make more accurate predictions.

When supervision deficiencies arise, we leave the field of Strong Supervised Learning and enter the Weakly Supervised Learning one. The Weakly Supervised Learning field is characterized by a high degree of fragmentation, which is largely attributed to the diverse nature of supervision weaknesses. Consequently, in order to effectively utilize these methods, practitioners must have a clear understanding of the specific weaknesses of their problem. This requirement renders the practical application of the literature in real-world scenarios challenging, as it frequently necessitates modifying the assumptions of the techniques employed, or leads to sub-optimal performances.

Thus, we introduce a hierarchical organization of the taxonomy introduced in [224] of the Weakly Supervised Learning literature that follows a pattern of yes-no questions in the form of a Tree to deduce which sub-literature is adapted to solve a particular weakly supervised machine learning task.

The first split of the Weakly Supervision Tree from Figure 1.2 is the strong-weakly supervised split, denoting whether supervision deficiencies are present. Then when labeled data are available, the first question to answer is about the noisiness

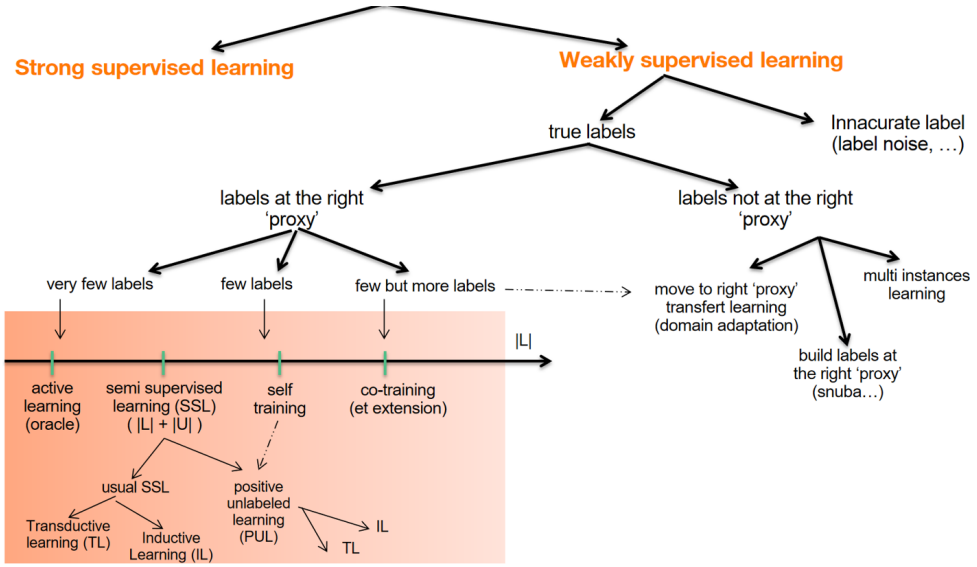


Figure 1.2: The Weak Supervision Tree

of the annotation process. If the labels are noisy, it is wise to refer to the literature on Robust Learning under Label noise (RLL). Otherwise, we can follow the other branch. Next, another split represents whether the labels are at the suitable proxy or not, indicating if the labels are annotated for the machine-learning task of interest. In this case, Domain Adapatation, Transfer Learning, or even Multi-Instance Learning are among the literature of interest. If the labels are at the right proxy (proxy definition will be given in the following Section), there is the last split representing the level of annotated data that organize the Semi-Supervised learning literature, from Active Learning to Multi-View Training going through Novel Class Discovery [220, 174].

Finally, Weakly Supervised Learning is about answering these questions from the data or prior knowledge and using the most adapted algorithms and methodologies to solve practical machine-learning tasks.

However, there is a crucial limit in the taxonomy proposed in Figure 1.2: we need to be able to find our position in the Tree to go through it, meaning answering questions that relate to the compliance of the assumptions of a sub-domain. Nonetheless, hypotheses used in different weakly supervised settings are pretty hard to verify in practice. For example, when learning under label noise, identifying whether the noise's corruption model is independent of the features is impossible in practice, leading to using approaches that lose their correctness.

Another limit is the divisive nature of a Decision Tree that cannot make continuous connections between different sub-domains. Figure 1.2 makes adding a combination of different weaknesses of supervision, such as Semi-Supervised Learning with Noisy Labels, quite challenging.

Finally, if the literature does not yet cover a certain weakness of supervision, Figure 1.2 does not provide any solutions whatsoever.

In the following Section, we propose a novel taxonomy of Weakly Supervised Learning as a continuous cube that encompasses all of the weaknesses above supervision. For a more graphic comparison: we need more than a map to navigate in the 3D land of weak supervision, we need a GPS device.

1.3 . The Weak Supervision Cube

The taxonomy proposed in this Chapter is organized as a "cube" and presented in Figure 1.3. This section progressively presents the differences between weakly supervised approaches by going through the axes of this cube.

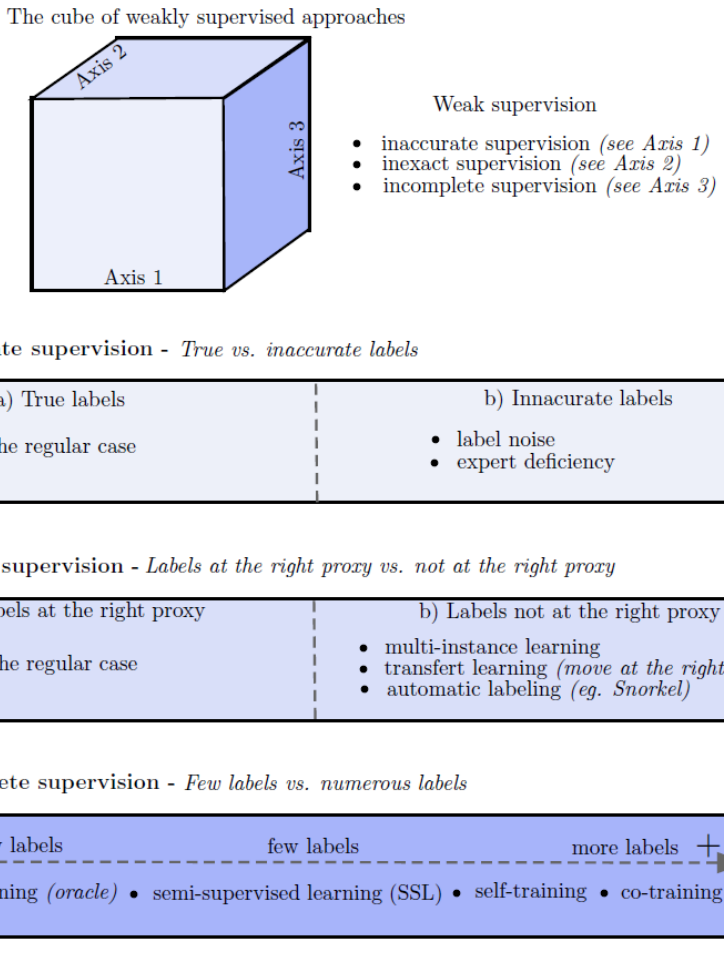


Figure 1.3: Taxonomy: an attempt - The big picture

First, a distinction must be made between *strong* and *weak* supervision. On the one hand, *strong* supervision corresponds to the typical case in *machine learning* where the training examples are expected to be exhaustively labeled with true

labels, i.e., without any corruption or deficiency. On the other hand, *weak* supervision means that the available ground truth is imperfect or even corrupted. The WSL field aims to address a variety of supervision deficiencies which can be categorized in a "cube" along the following three axes as illustrated in Figure 1.3: inaccurate labels (Axis 1), inexact labels (Axis 2), incomplete labels (Axis 3).

These three dimensions are detailed in the rest of this section and constitute the proposed taxonomy.

1.3.1 . Axis 1: Inaccurate Supervision - True Labels *vs.* Inaccurate Labels

Lack of confidence in data sources is a frequent issue regarding real-life use cases. The values used as learning targets, called labels or classes, can be incorrect due to many factors.

In practice, a variety of situations can lead to inaccurate labels: (i) a label can be assigned to a "bag of examples", such as a bunch of keys. In this case, at least one of the examples in the keychain belongs to the class indicated by the label. Multi-instance learning [200, 223, 53, 18] is an appropriate technique to deal with this learning task. (ii) a label may not be "guaranteed" and may be noisy. In theory, the learning set should be labeled in an unbiased way with respect to the concept to be learned. However, the data used in real-world applications provide an imperfect ground truth that does not match the concept to be learned. As defined in [80], noise is "anything that obscures the relationship between the features of an instance and its class". According to this definition, every *error* or *imprecision* in an attribute or label is considered as noise, including human deficiency. Noise is not a trivial issue because its origin is never clearly evident. In practical cases, this leads to trouble evaluating the existence and the strength level of noise in a dataset. Frenay et al. in [55] provide a good overview of noise sources, the impact of labeling noise, types of noise, and dependency on noise. Below is a non-exhaustive list of common ways to learn a model in the presence of labeling noise[1]:

- in case of marginal noise level, a standard learning algorithm that is natively robust to label noise, is used [99, 126, 52, 227];
- uses a loss function which solves theoretically (or empirically) the problem in case of (i) noise completely at random[2] [24]; or (ii) class dependent noise [79, 196]. In most cases, this type of approach is known in the literature as "Robust Learning to Label noise (RLL)";
- model noise to assess the quality of each label (requires assumptions on noise) [170];

[1] Note: the number of articles published on this topic has exploded in recent years.
[2] defined in Section 1.6.2.

- enforce consistency between the model's predictions and the proxy labels [148];
- clean the training set by filtering noisy examples [172, 117, 121, 118, 176];
- trust a subset of data provided by the user to learn a model at once on trusted examples (without label noise) and untrusted ones [79, 78, 134].

Another kind of "noise" appears when each training example is not equipped with a single true label but a set of candidate labels containing the true label. To deal with this kind of training examples, Partial Label Learning (PLL) has been proposed [84] (also called ambiguously labeled learning). It has attracted attention as, for example, in the algorithms IPAL [212], PL-KNN [84], CLPL [33], and PL-SVM [128] or when suggesting semi-supervised partial label learning as in [185]. This setting is motivated, for example, by a common scenario in many images and video collections, where only partial access to labels is available. The goal is to learn a classifier that can disambiguate the partially-labeled training instances and generalize to unseen data [34].

1.3.2 . Axis 2: Inexact Supervision - Labels at the Right Proxy *vs.* not at the Right Proxy

The second axis describes *inexact labeling*, which is orthogonal to the first type of supervision deficiency - i.e., inexact labeling and noisy labeling may coexist. Here, the labels are provided not at the right proxy, which corresponds to one (or possibly a mixture) of the following situations:

- *Proxy domain*: the target domain differs between the training set and the test set. For instance, it could be learning to discriminate "panthers" from other savanna animals based on "cats" and "dogs" labels. Two cases can be distinguished: (i) training labels are available in another target domain than test labels (ii) or training labels are available in a sub-domain that belongs to the original target domain. Domain transfer [43] or domain adaptation [8] are techniques aimed at addressing these learning tasks.
- *Proxy labels*: some unlabeled examples are automatically labeled, either by a rule-based system or by a learned model, to increase the training set's size. These kinds of labels are called proxy labels and can be considered as coming from a proxy concept close to the one to be learned. Only the true labels stand for the ground truth. The way proxy labels are used depends on their origin. In the case where the classifier itself provides proxy labels without any additional supervision, the self-training (ST) [48], the co-training (CT), and their variants attempt to improve the learned model by including proxy-labels in the training set as regular labels. Other approaches exploit the confidence level of the classifier to produce soft-proxy-labels and then exploit it as weighted training examples [173]. When a rule-based

system generates proxy labels, the quality of labels depends on the expert's knowledge which is manually encoded into the rules. Ultimately, a classifier learned from such labels can be considered a means of smoothing the set of rules, allowing the end-user to score any new example. Some recent automatic labeling systems offer an intermediate way that mixes rule-based systems and machine learning approaches (MIX) [147, 181].

- *Proxy individuals*: the statistical individuals are not equally defined between the training set and the test set. For instance, it could be learning to classify images based on labels that only describe parts of the images. Multi-instance learning (MIL) is another example that consists in learning from labeled groups of individuals. In the literature, many algorithms have been adapted to work within this paradigm [200, 223, 53, 18].

1.3.3 . Axis 3: Incomplete Supervision - Few labels *vs.* Numerous

The third axis describes incomplete supervision, which consists of processing a *partially labeled* training set. In this situation, labeled and unlabeled examples coexist within the training set, and it is assumed that there need to be more labeled examples to train a performing classifier. The objective is to use the entire training set, including the unlabeled examples, to achieve better classification performance than learning a classifier only from labeled examples.

In the literature, many techniques exist capable of processing partially labeled training data, i.e., active learning (AL), semi-supervised learning (SSL), positive unlabeled learning (PUL), self-training (ST), and Co-Training (CT). At the bottom of Figure 1.3, we suggest sorting these methods according to the quantity of labeled examples they require. We decide to review all these subdomains in their own separate Section for the sake of clarity.

1.4 . A Lookup on Axis 3

1.4.1 . Active Learning (AL)

Modern supervised learning approaches are known to require large amounts of training examples in order to achieve their best performance. These examples are mainly obtained by human experts who label them manually, making the labeling process costly in practice. Active learning (AL) [157] is a field that includes all the selection strategies that allow one to iteratively build the training set of a model in interaction with a human expert (also called an oracle). The aim is to select the most informative examples to minimize the labeling cost.

Active learning is an iterative process that continues until a labeling budget is exhausted or a predefined performance threshold is reached. Each iteration begins with the selection of the most informative example. This selection is generally based on information collected during previous iterations (predictions of a classifier, density measures). The selected example is then submitted to the oracle, which

returns the associated class, and the example is added to the training set (L). The new learning set is then used to improve the model, and the new predictions are used to perform the next iteration.

In conventional heuristics, the utility measures used by active learning strategies [157] differ in their positioning with respect to the trade-off between exploiting the current classifier and exploring training data. Selecting an unlabelled example in an unknown region of the observation space helps to explore the data to limit the risk of learning a hypothesis that is too specific to the set L of currently labeled examples. Conversely, selecting an example in an already sampled region allows for refining the predictive model locally. We do not intend to provide an exhaustive overview of existing AL strategies and refer to [3, 157] for a detailed overview, [141, 202, 142] for some recent benchmarks and a new way to treat uncertainty in [85].

Another *meta active learning* paradigm exists, which combines conventional strategies using bandit algorithms [6, 44, 82, 28, 30, 136]. These meta-learning algorithms intend to select online the best AL strategy according to the observed improvements of the classifier. These algorithms can adapt their choice over time as the classifier improves. However, learning must be done using fewer examples to be useful, and these algorithms suffer from the cold start problem. Even thought these approaches are able to select online the best AL strategy, they are not capable to learn new strategy from the data.

Other meta-active-learning algorithms have been developed to learn an AL strategy starting from scratch, using multiple sources of datasets. These algorithms transfer the learned AL strategy to new target datasets [94, 95, 137]. Most of them are based on modern reinforcement learning methods. The major challenge is to learn an AL strategy general enough to automatically control the exploitation/exploration trade-off when used on new unlabeled datasets (which is impossible using heuristic strategies). A recent evaluation of learning active learning can be found in [41].

1.4.2 . Semi-Supervised Learning (SSL)

Early work on semi-supervised learning dates back to the 2000s; an overview of these pioneering papers can be found in [156, 22, 21, 226, 225]. In the literature, the SSL approaches can be categorized into two groups:

- Algorithms that use unlabeled examples unchanged. In this case, the unlabeled examples are treated as unsupervised information added to the labeled examples. Four main categories exist: generative methods, graph-based methods, low-density separation methods, and disagreement-based methods [224].

- Semi-supervised learning algorithms that produce proxy labels on unlabeled examples, which are used as targets in addition to the labeled examples.

These proxy labels are produced by the model itself or by its variants. In the end, these inaccurate labels (see Section 1.3.1) can be considered noisy. The rest of this section deals with particular cases of SSL and presents the Positive Unlabeled Learning, Self-Training, and Co-training approaches.

1.4.3 . Postive Unlabeled Learning (PUL)

Learning from Positive and Unlabeled examples (PUL) is a special case of binary classification, and SSL [7]. In this particular setting, the unlabeled examples may contain positive and negative examples with hidden labels. These approaches differ from a one-class classification [92] since they explicitly use unlabeled examples in the learning process. In the literature, the PUL approaches can be divided into three groups:

1. Two-step techniques [109] for PUL build on the assumptions of separability and smoothness. The idea is that all positive examples are similar to the labeled examples and that the negative examples are very different from them. The two-step technique consists of identifying reliable negative examples in the first step [60] and then using semi-supervised learning techniques with the labeled positive examples, the reliable negatives, and optionally the remaining unlabeled examples in the second step.

2. Biased PUL methods treat unlabeled examples as negatives examples with class label noise, the noise is taken into account by, for example, placing higher penalties on misclassified positive examples or tuning hyperparameters based on an evaluation metric that is suitable for PU data [109].

3. Finally, under the selected completlty at random assumption, the class prior can be incorporated directly into the training algorithms, or can be added in a preprocessing or postprocessing step of the training procedure. Postprocessing trains a non-traditional probabilistic classifier by considering the unlabeled data as negative and modifies the output probabilities [47], preprocessing changes the dataset by using the class prior [45]. The class prior can be determined using dedicated algorithms or it can be tuned using evaluation metrics for PU data.

1.4.4 . Self Training (ST)

Self-training lacks a clear definition in the literature. It can be viewed as a "single-view weakly supervised algorithm". First, a classifier is trained from the available labeled examples, and then this classifier is used to make predictions and build new proxy labels. Only those examples where confidence in proxy labeling exceeds a certain threshold are added to the training set. Then, the classifier is retrained from the training set enriched with proxy-labels. This process is repeated in an iterative way [48].

1.4.5 . Co-Training (CT)

Starting from a set of partially labeled examples, co-training algorithms [12, 39, 215, 120] aim to increase the number of labeled examples by generating proxy-labels.

Co-training algorithms work by training several classifiers from the initially labeled examples. Then, these classifiers are used to make predictions and generate proxy-labels on the unlabeled examples. The most confident predictions on these proxy-labels are then added to the set of labeled data for the next iteration.

One crucial aspect of co-training is the relationship between the views (the sets of explicative variables) used in learning the different models.

The original co-training algorithm [12] states that the independence of the views is required to properly perform automatic labeling. More recent works [130, 2, 29] show that this assumption can be relaxed. Another requirement is to obtain at the iteration step a "reasonable" classifier in terms of performances, explaining why we place co-training on the left of AL and SSL in Figure 1.3 and Figure 1.2. In [127], a study is given on the optimal selection of the co-training parameters.

Co-training can also be considered as a member of "multi-view training" family to which some other members belong to, such as: Democratic Co-learning [221], Tri-training [219], Asymmetric tri-training [154], Multi-task tri-training [152], which are not described here.

1.5 . Beyond the Axes

1.5.1 . Learning at the crossroad of the three axes

Using a cube to describe the literature on Weakly Supervised Learning allows us to use the axes but also the volume of the cube to position existing approaches. It is now easy to position the approaches related to several axes. For example, Partial Label Learning may be related to two supervision deficiencies: i) inexact supervision because multiple labels are provided for each training example; ii) inaccurate supervision because only one of the labels provided is correct. Positioning the PLL on the plane defined by these two axes seems more relevant.

Also, this representation highlights some exciting intersections between two axes or between an axis and a plane. One of these points of interest is the origin of the three axes, which corresponds to the case where supervision is absolutely inaccurate, imprecise, and incomplete, which ultimately amounts to unsupervised learning. Similarly, the point at the opposite end of the cube corresponds to perfectly precise, accurate, and complete supervision, which equates to supervised learning.

Finally, this representation could provide insights into the reasons of using proven techniques from a particular subfield of Weakly Supervised Learning can be efficient in another. For instance, DivideMix [102] chooses to reuse the efficient MixUp [210] approach from Semi-Supervised Learning to tackle the problem of

Learning with Label Noise. This approach uses Data Augmentation [160] and Model Agreement [145] to estimate labels probabilities and then discard or keep the provided labels.

This section is not exhaustive, and interested readers will be able to position the literature approaches in the cube themselves.

1.5.2 . Deficiency Model

The deficiency model describes the nature of the supervision deficiency. It is usually described as a probability measure called $\rho : (x, y) \mapsto \rho(x, y)$, representing the probability of a sample being corrupted. ρ can depend on the value the explanatory variables $x \in \mathcal{X}$, the label value $y \in \mathcal{Y}$ or both of them (x, y).

Different types of supervision deficiencies are described in this section, namely: Completely At Random (CAR), At Random (AR), and Not At Random (NAR).

If the probability of being corrupted is the same for all training examples, $\rho : (x, y) \mapsto \rho_c$, $\rho_c \in [0, 1]$, then the supervision deficiency model is Completely At Random (CAR). It implies that the cause of the supervision deficiency data is unrelated to the data. If the probability of being corrupted is the same within classes, $\rho : (x, y) \mapsto \rho_y$, $\forall y \in \mathcal{Y}$, $\rho_y \in [0, 1]$, then the supervision deficiency model is At Random (AR). If neither CAR nor AR holds, then we speak of Not at Random (NAR) model. Here, the probability of being corrupted depends on the sample and the label value, $\rho : (x, y) \mapsto \rho(x, y)$. These three deficiency models can be ranked in a descendant manner, having the NAR model being the most complex one as it depends on both the instance and label value, which requires a function to model, to CAR model where only one constant is enough to describe it. These models may help the practitioner to find links between supervision deficiencies. For example, PUL is SSL with only one class labeled, which means that the missingness of the label is linked to the label value, so PUL is an extreme case of SSL AR with $\rho_0 = 1 - e$ and $\rho_1 = e$ (where e is called the propensity score).

AL is another form of SSL where examples are labeled thanks to a strategy, previously labeled instances, and the ordered iterative process leading to non-iid labeled data. As such, AL is part of the SSL NAR family. We want to reiterate that the deficiency model can be applied to any supervision deficiency, even if it has been primarily featured in RLL and SSL.

1.5.3 . Transductive learning vs. Inductive Learning

As we consider the WSL framework, one may be tempted to use the test set to guide the choice of the model. But in this case, we need to carefully decide if in the future the need for a model to predict on another test (deployment) dataset is required or not. Two points of view could be considered: transductive learning vs. inductive learning, and this is why we now add a note on them.

Training a machine could take many forms: supervised, unsupervised, active, online, etc. The number of members in the family is significant, and new members appear regularly like "federative learning". However, one may establish a separation

between two constant classes based on how the user would like to use the "learning machine" at the deployment stage. A user does not necessarily want a predictive model for subsequent use on new data and would rather get insights on its already gathered data. It is, therefore, necessary to distinguish between inductive learning and transductive learning.

On one side, the goal of inductive learning is, essentially, to learn a function (a model) that will later be used on new data to predict classes (classification) or numerical values (regression). The predictions may be seen as "by-products" of the model. Induction is reasoning from observed training cases to general rules, which are then applied to the test cases. On the other side, the goal of transductive learning is not to obtain a function or a model but only to make predictions on a given test database and only on this set of instances. Transduction was introduced by Vladimir Vapnik in the 1990s, motivated by the intuition that transduction is preferable to induction since, induction requires solving a more general problem (inferring a function) before solving a more specific problem (computing outputs for new cases). However, the distinction between inductive and transductive learning could be hazy, for example, in the case of semi-supervised learning. Knowing this, the view of Zhou in [224] about "pure semi-supervised learning" and transductive learning is engaging. The distinction between Transductive and Inductive Learning concerned most of the learning forms in Figure 1.3.

1.6 . Measurable quantities of WSL

Until now, we have seen many forms of learning and weaknesses intertwine. A way to resume their aspect was given in Figure 1.3. From this point of view, one may identify three common concepts that are described now.

1.6.1 . Quantity $|L|$

An insufficient quantity of labels or training examples occurs when many training examples are available, but only a tiny portion is labeled, e.g., due to the cost of labeling. For instance, this occurs in cyber security, where human forensics is needed to tag attacks. Usually, this issue is addressed by Few Shot Learning (FSL), active learning (AL) [157] semi-supervised learning (SSL) [21], Self Training, or Co-Training or active learning (AL), which have been described briefly above in this Chapter. Another way to see the "quantity" could be the ratio between the number of examples labeled and unlabeled (p).

1.6.2 . Quality q

In this case, all training examples are labeled, but the labels may be corrupted. This usually happens when outsourcing labeling to crowds [175]. The Robust Learning to Label Noise (RLL) approaches tackle this problem [56], with three types of label noise identified: i) the *completely at random* noise corresponds to a uniform probability of label change; ii) the *class dependent* label noise when the

probability of label change depends upon each class, with uniform label changes within each class; iii) the *instance dependent* label noise is when the probability of label change varies over the input space of the classifier. This last type of label noise is the most difficult to deal with and typically requires making sometimes strong assumptions about the data.

1.6.3 . Adaptability a [144]

This is the case, for instance, in Multi Instance Learning (MIL) [200, 223, 53, 18], in which there is one label for each bag of training examples, and each example has an uncertain label.

Some scenarios in Transfer Learning (TL) [190] imply that only the labels in the source domain are provided while the target domain labels are not. Often, these non-adapted labels are associated with the existence of slightly different learning tasks *(e.g., more precise and numerous classes are dividing the original categories)*.

Alternatively, non-adapted labels may characterize a differing statistical individual [31] *(e.g., a subpart of an image instead of the entire image)*.

1.7 . From Weakly Supervised Learning to Biquality Learning ($a = 1$)

All the types of supervision deficiencies presented above are addressed separately in the literature, leading to highly specialized approaches. In practice, it is tough to identify the type(s) of deficiencies with which a real dataset is associated.

For this reason, it would be beneficial to suggest another point of view as a tentative of a unified framework for (a part of the) Weakly Supervised Learning to design generic approaches capable of dealing with not a single type of supervision deficiency. This section's purpose is mainly given for cases where data are adapted to the task to learn ($a = 1$).

Learning using biquality data has recently been put forward in [23, 79, 76] and consists in learning a classifier from two distinct training sets, one trusted and the other not. The initial motivation was to unify semi-supervised and robust learning through a combination of the two. This chapter shows that this scenario is not limited to this unification and can cover a more extensive range of supervision deficiencies, as demonstrated by the algorithms we suggest and their results.

The two datasets contain the same set of features $\mathcal{X}$ and the same set of labels $\mathcal{Y}$. The trusted dataset D_T consists of pairs of labeled examples (x_i, y_i) where all labels $y_i \in \mathcal{Y}$ are supposed to be correct according to the true underlying conditional distribution $\mathbb{P}_T(Y|X)$. In the untrusted dataset D_U, examples x_i may be associated with incorrect labels. We note $\mathbb{P}_U(Y|X)$ the corresponding conditional distribution.

At this stage, no assumption is made about the nature of the supervision deficiencies, which could be of any type, including label noise, missing labels,

concept drift, non-adapted labels, ..., more generally, a mixture of these supervision deficiencies.

The difficulty of a learning task performed on biquality data can be characterized by two quantities. First, the ratio of trusted data over the whole data set, denoted by p:

$$p = \frac{|D_T|}{|D_T| + |D_U|} \tag{1.1}$$

Second, a measure of the quality, denoted by q, which evaluates the usefulness of the untrusted data D_U to learn the trusted concept. For example in [76] q is defined using a ratio of Kullback-Leibler divergence between $\mathbb{P}_T(Y|X)$ and $\mathbb{P}_U(Y|X)$.

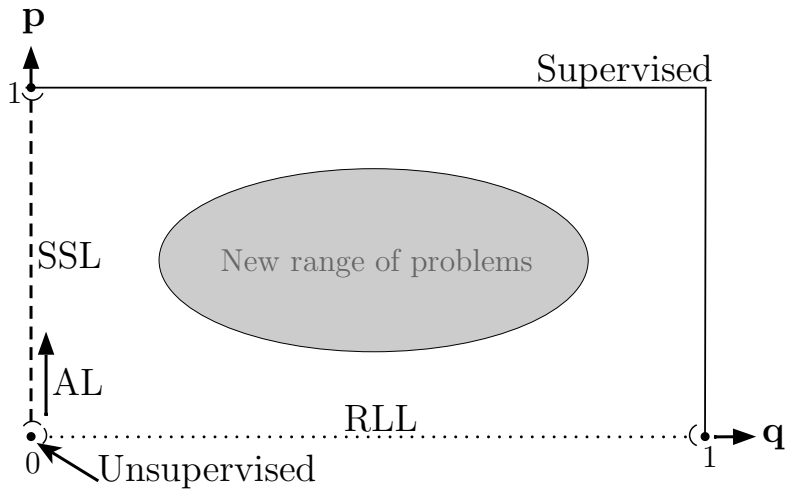


Figure 1.4: The different learning tasks covered by the biquality setting, represented on a 2D representation.

The biquality setting covers a wide range of learning tasks by varying the quantities q and p, as represented in Figure 1.4.

- When ($p = 1$ OR $q = 1$)[3] all examples can be trusted. This setting corresponds to a standard **supervised learning** (SL) task.
- When ($p = 0$ AND $q = 0$), there is no trusted examples and the untrusted labels are not informative. We are left with only the inputs $\{x_i\}_{1 \leq i \leq m}$ as in **unsupervised learning (UL)**.
- On the vertical axis defined by $q = 0$, except for the two points $(p, q) = (0, 0)$ and $(p, q) = (1, 0)$, the untrusted labels are not informative, and trusted examples are available. The learning task becomes **semi-supervised learning (SSL)** with the untrusted examples as unlabeled, and the trusted as labeled.

[3] $p = 1 \implies D_U = \emptyset \implies q = 1$

- An upward move on this vertical axis, from a point $(p, q) = (\epsilon, 0)$ characterized by a low proportion of labeled examples $p = \epsilon$, to a point $(p', 0)$, with $p' > p$, corresponds to **Active Learning** if an oracle can be called on unlabeled examples. The same upward move can also be realized in **Self-training** and **Co-training**, where unlabeled training examples are labeled using the predictions of the current classifier(s).

- On the horizontal axis defined by $p = 0$, except for the points $(p, q) = (0, 0)$ and $(p, q) = (0, 1)$, only untrusted examples are provided, which corresponds to the range of learning tasks typically addressed by **Robust Learning to Label noise (RLL)** approaches.

Only the edges of Figure 1.4 have been envisioned in previous works – i.e. the points mentioned above – and a whole new range of problems corresponding to the entire plan of the figure remains to be explored.

Biquality Learning may also be used to tackle particular tasks belonging to WSL, for instance:

- Positive Unlabeled Learning (PUL) [7] where the trusted examples are only positive and untrusted examples from the unknown class.

- Self Training and Co-training [12, 39, 215] could be addressed at the end of their self-labeled process: the initial training set is the trusted dataset, all examples labeled after (during the self-labeling process) are the untrusted examples.

- Concept drift [61]: when concept drift occurs, all the examples used before a drift detection may be considered untrusted examples, while the examples available after it is viewed as the trusted ones, assuming a perfect labeling process.

- Self Supervised Learning system as Snorkel [147]: the small initial training set is the trusted dataset, and all examples automatically labeled using the labeling functions correspond to the untrusted examples.

As seen from the above list, the Biquality framework is quite general, and its investigation seems a promising avenue to unify different aspects of Weakly Supervised Learning.

1.8 . Conclusion

In this Chapter, we propose a unified view of Weak Supervised Learning to cope with the shortcomings of supervision in the field of Machine Learning. We discussed these shortcomings through a cube along with three axes corresponding to the characteristics of training labels (inaccurate, inexact, and incomplete). The detailed presentation of these axes gives an insight into the different existing learning approaches, which can be more subtly positioned on the cube. In this way, the links between some subfields of WSL with Biquality Learning are highlighted, showing how the algorithms of the latter field can be used within the framework of WSL.

In Chapter 2 of this thesis, we will propose a theoretical Biquality Learning framework to ground and regroup existing works in this field. Then, an exhaustive State-of-the-Art will be compiled and organized around the originating story of the field. Lastly, we will propose an experimental protocol to fairly evaluate Biquality Learning algorithm with either synthetic or real-world Biquality datasets, statistical tests, and visualizations.

2 - Biquality Learning

Contents

2.1 . Introduction

Weakly Supervised Learning (WSL) is the field of Machine Learning related to Learning with imperfect labels. The imperfections happen in the real world when labeled samples are hard to get or when manually crafted. When labels are complex to get, Machine Learning practitioners have been quite ingenious in how to reuse previously gathered labels, use knowledge from a different task to solve a target supervision task in Transfer Learning (TL), or design strategies to optimize the labeling process in Active Learning (AL). When labels are manually crafted, errors can occur due to experts who need to satisfy the constraints of the labeling process, as in Learning with Label Noise (RLL) or Covariate Shift (CS). These edge cases, among others, have been poured into the WSL field, leading to many different algorithm families and recent WSL taxonomy having difficulty reconciling and organizing them in a fluid and continuous manner.

In Chapter 1, we revisited the usual WSL taxonomy by defining three axes. These axes represent supervision deficiencies, namely inaccurate supervision, inexact supervision, and incomplete supervision. Inaccurate supervision represents cases where samples are not labeled correctly, corresponding to a targeted supervision task. Inexact supervision corresponds to cases where labels are not at the right proxy, meaning that not every sample has an associated label, for example. These axes define a continuous cube where sub-fields fluidly lie in a unified framework. We highlighted the benefits of such a vision and proceeded to detail one particular plan of this cube named Biquality Learning.

Biquality Learning is a hot topic at the Orange company because it fits the main Machine Learning use cases such as fraud detection, scammer detection, or cyber-security attacks detection. Usually, a small, trusted dataset can be crafted by domain experts but requires lengthy forensic procedures. In contrast, a substantial untrusted dataset can be crafted by reusing simple rule systems designed by these experts. Effectively using both datasets and their different natures is essential for Orange and pushing the Biquality Learning literature is one way to do so.

In Chapter 2, we will provide a theoretical framework for Biquality Learning algorithms to describe our subject study precisely. Then we will talk about related domains of Biquality Learning to put into perspective this new domain within the existing literature. Then, an exhaustive State-of-the-Art will be compiled and organized around the intuitions used by the author to design their proposed algorithm. Lastly, we will propose an experimental protocol to fairly evaluate Biquality Learning algorithm with either synthetic or real-world Biquality datasets, statistical tests, and visualizations.

2.2 . Biquality Learning Framework

Biquality Learning algorithms aim to learn a decision function $f : \mathcal{X} \to \mathcal{Y}$ that classifies samples $X \in \mathcal{X}$ by assigning them to a label $Y \in \mathcal{Y}$. This decision function f is learned using two datasets, a trusted one D_T and an untrusted one D_U. The two datasets contain the same set of features $\mathcal{X}$ and the same set of labels $\mathcal{Y}$. The decision function f corresponds to the trusted concept $\mathbb{P}_T(Y|X)$ from which samples from the trusted dataset are labeled. With biquality learning, no assumption is made about the difference between the trusted distribution $\mathbb{P}_T(Y|X)$ and the untrusted one $\mathbb{P}_U(Y|X)$.

Learning the true concept[1] $\mathbb{P}_T(Y|X)$ on $D = D_T \cup D_U$ means minimizing the risk R on D with a loss L for a probabilistic classifier f:

$$\begin{aligned} R_{D,L}(f) &= \mathbb{E}_{D,(X,Y)\sim T}[L(f(X), Y)] \\ &= \mathbb{P}(X \in D_T)\mathbb{E}_{D_T,(X,Y)\sim T}[L(f(X), Y)] \\ &+ \mathbb{P}(X \in D_U)\mathbb{E}_{D_U,(X,Y)\sim T}[L(f(X), Y)] \end{aligned} \tag{2.1}$$

where $L(\cdot, \cdot)$ is a loss function, from $\mathbb{R}^{|\mathcal{Y}|} \times \mathcal{Y}$ to $\mathbb{R}$ since $f(X)$ is a vector of probability over the classes. Since the true concept $\mathbb{P}_T(Y|X)$ cannot be learned from D_U, the last line of Equation 2.1 is not tractable as it stands. That is why we propose a **generic formalization** based on a mapping function g that enables us to learn the true concept from the modified untrusted examples of D_U. Equation 2.1 becomes:

$$\begin{aligned} R_{D,L}(f) &= \mathbb{P}(X \in D_T)\mathbb{E}_{D_T,(X,Y)\sim T}[L(f(X), Y)] \\ &+ \lambda\, \mathbb{P}(X \in D_U)\mathbb{E}_{D_U,(X,Y)\sim U}[g(L(f(X), Y))] \end{aligned} \tag{2.2}$$

In Equation 2.2, the parameter $\lambda \in [0, 1]$ reflects the quality of the untrusted examples of D_U modified by the function g. This time, the last line is tractable since it consists of a risk expectancy estimated over the training examples of D_U, which follows the untrusted concept $\mathbb{P}_U(Y|X)$, modified by the function g.

To estimate the expected risk, this formalization requires learning three items: g, λ, and then f. Both D_T and D_U are used to learn a mapping function, g, between the two datasets. Then, either λ is considered a hyperparameter to be learned using D_T or λ is provided by an appropriate quality measure and is considered an input of the learning algorithm. Finally, f is learned by minimizing the risk R on D using the mapping g.

In this formalization, the mapping function g plays a central role. Not exhaustively, we identify three different ways of designing the mapping function. For each of these, a different function g' enters the definition of function g:

- The first option consists in **correcting the label** for each untrusted examples of D_U. The mapping function thus takes the form $g(L(f(X), Y)) =$

[1] For reasons of readiness, we denote $\mathbb{P}_T(Y|X)$ by T and $\mathbb{P}_U(Y|X)$ by U.

$L(f(X), g'(Y, X))$, with $g'(Y, X)$ denote the new corrected labels and $f(X)$ the predictions of the classifier.

- In the second option, the untrusted labels are used unchanged. The untrusted examples X are **moved** in the input space where the untrusted labels become correct with respect to the true underlying concept. The mapping function becomes $g(L(f(X), Y)) = L(f(g'(X)), Y)$, where $g'(X)$ is the "moved" input vector of the modified untrusted examples.

- In the last option, g' **weights** the contribution of the untrusted examples in the risk estimate. Accordingly, we have $g(L(f(X), Y)) = g'(Y, X)L(f(X), Y)$. In this case, the parameter λ may disappear from Equation 2.2 since it can be considered as included in the function g'.

Biquality Learning algorithms, designed with such mapping functions g, helps leverage the knowledge from untrusted data to improve the classifier's efficiency on the trusted task. However, using additional information from other sources or correcting past data to improve the performance of a classifier is not new to Biquality Learning and has been explored in other machine learning domains.

2.3 . Related Domains

Various areas within the field of Machine Learning focus on utilizing additional information in addition to the primary training dataset to enhance the training of models. However, the motivations behind these efforts can vary greatly and have resulted in different fields with their own specific goals. The upcoming Section will examine the relationship between these Machine Learning domains and Biquality Learning, explore how they can be utilized to develop more effective Biquality Learning algorithms, and investigate the limitations of their approaches when applied to Biquality Data.

2.3.1 . Inductive Transfer Learning

Transfer Learning focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. Two datasets are at our disposal, a source dataset D_S and a target dataset D_T. They are related to a source domain $\mathcal{X}_S$ with source labels $\mathcal{Y}_S$ and a target domain $\mathcal{X}_T$ with target labels $\mathcal{Y}_T$ to solve the target task $\mathbb{P}(Y_T|X_T)$ with the help of the source task $\mathbb{P}(Y_S|X_S)$. Transfer learning aims to leverage the knowledge gained from solving the source problem to solve the target problem more efficiently.

Briefly, we can draw a parallel between Biquality Learning notations and Transfer Learning notations primarily by substituting (source, S) by (untrusted, U) and (target, T) by (trusted, T).

Formally, this description of Transfer Learning is the most general setup that would incorporate Biquality Learning and other well-known setups such as Domain

Adaptation, Transductive Transfer Learning, or Covariate Shift. Most specifically for Biquality Learning, the Transfer Learning setup that best fit is the Inductive Transfer Learning [135] where:

- $\mathcal{X}_T = \mathcal{X}_U$
- $\mathcal{Y}_T = \mathcal{Y}_U$
- $\mathbb{P}(X_T) = \mathbb{P}(X_U)$
- $\mathbb{P}(Y_T|X_T) \neq \mathbb{P}(Y_U|X_U)$

From these analogies, any Transfer Learning algorithm that requires, at most, the Inductive Transfer Learning assumptions could be applied to the Biquality Data setup with success.

One main assumption of Transfer Learning that has yet to be highlighted in this Section is the usefulness of the source task to train or improve machine learning models on the target task. However, in the Biquality Data setup, we can have untrusted datasets that bring no information to the trusted task when labels are assigned randomly or even adversarial information when data poisoning have corrupted the dataset. The impact of this assumption can be directly observed when using TrAdaBoost [36] on Biquality Data which is a well-known Inductive Transfer Learning algorithm.

In TrAdaBoost, the first step implies learning a classifier on all trusted and untrusted data combined. When the untrusted dataset is so corrupted that it makes learning a classifier impossible in the first iterations when samples are not corrected, the condition of beating random guessing will not be met, and the algorithm will stop. Even without this condition, the learned classifier will be close to random guessing, and no correction provided by TrAdaBoost will be useful to train a new base learner. The only alternative is to use a highly robust classifier, but no classifier is robust to data poisoning.

These problems come in all Transfer Learning algorithms such as Multi-Task Learning [19] or Fine-Tuning [206]. Thus having Inductive Transfer Learning algorithms leads to bad predictive performances when used on Biquality Data with strongly corrupted data.

2.3.2 . Supervised Domain Adaptation

Another closely related domain to Transfer Learning is Domain Adaptation. Domain adaptation refers to the process of adapting a model trained on one distribution of data to work well on a different distribution of data. This is often necessary because it is impractical or impossible to collect a large enough dataset to train a model that can generalize well for all possible inputs it may encounter in practice. The setup is the same as in Transfer Learning with source and target datasets with their respective domain and labels.

The parallel between Domain Adaptation and Biquality Learning is more or less the same as the one between Transfer Learning and Biquality Learning. Nevertheless, more specifically, if Inductive Transfer Learning was the closest setup of Transfer Learning to Biquality Learning, for Domain Adaptation it is Supervised Domain Adaptation, which follows these assumptions:

- $\mathcal{X}_T = \mathcal{X}_U$
- $\mathcal{Y}_T = \mathcal{Y}_U$
- $\mathbb{P}(X_T) \neq \mathbb{P}(X_U)$
- $\mathbb{P}(Y_T|X_T) = \mathbb{P}(Y_U|X_U)$

When looking at these assumptions, especially with $\mathbb{P}(Y_T|X_T) = \mathbb{P}(Y_U|X_U)$ there is no clear evidence on how Domain Adaptation could be helpful to Biquality Learning. Indeed, adapting features of untrusted samples would have no correction effect on these data as one of the assumptions of Biquality Learning made in this Chapter was the absence of covariate shift between the trusted and untrusted dataset: $\mathbb{P}(X_T) = \mathbb{P}(X_U)$.

Remark. The impact of Covariate Shift on Biquality Learning will be studied in Chapter 4.

However, when looking at the distribution of features in Biquality Data given the same class $\mathbb{P}(X|Y)$, we can observe a change in distribution between the two datasets. Indeed, given the Bayes Formula $\mathbb{P}(X|Y) = \frac{\mathbb{P}(Y|X)\mathbb{P}(X)}{\mathbb{P}(Y)}$, and at $\mathbb{P}(X)$ and $\mathbb{P}(Y)$ constant, corrupting $\mathbb{P}(Y|X)$ is equivalent to corrupting $\mathbb{P}(X|Y)$. We illustrate this behavior with the following toy dataset where untrusted samples have been corrupted with Completly at Random label noise:

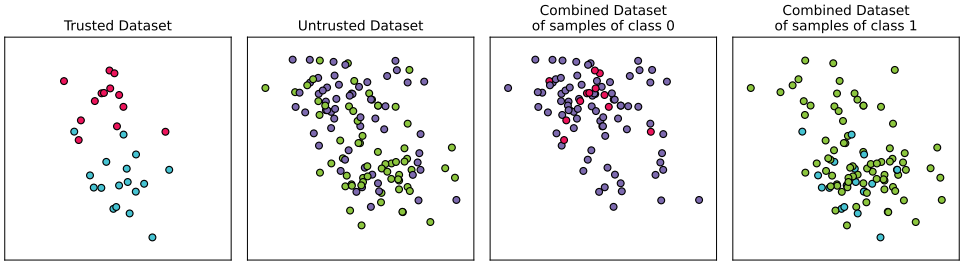


Figure 2.1: Illustration of the equivalence of Conditional Covariate Shift and Concept Drift on a toy dataset.

In Figure 2.1, we observe that some untrusted samples of class 0 (in purple) seem to be out of the normal distribution for trusted samples of class 0 (in red). The same behavior can be observed for samples of class 1 (respectively in green and blue).

Then, one approach that could be taken to use Domain Adaptation methods to Biquality Learning would be to train a classifier with K-Domain Adaptations (with K being the number of classes). This approach can be summarized in an algorithm we chose to name KDA and will be explored in Chapter 4:

Algorithm 1: K-Domain Adaptation (KDA)

Input: Trusted Dataset D_T, Untrusted Dataset D_U, Probabilistic Classifier Family $\mathcal{F}$, Domain Adaptation Algorithm Family $\mathcal{A}$

for $k \in [\![1, K]\!]$ **do**
  Let $D_T^k = \{\forall (x, y) \in D_T \mid y = k\}$
  Let $D_U^k = \{\forall (x, y) \in D_U \mid y = k\}$
  Correct D_U^k feature distribution in respect to D_T^k with $a \in \mathcal{A}$ to a corrected dataset D_C^k.
Let $D_C = \cup_{k=1}^{K} D_C^k$
Learn $f \in \mathcal{F}$ on $D_T \cup D_C$

Output: f

2.3.3 . Multi-Source Learning

In the context of machine learning, multi-source learning refers to the process of training a model using data from multiple different sources [35]. This can be useful when trying to improve the model's performance on a specific task, as it can help introduce a greater diversity of data and potentially reduce over-fitting. It can even be used to mitigate the unreliability of multiple sources to aggregate knowledge and recover the target distribution.

In Multi-Source Learning, all sources are treated equally as being reliable to the target task, without one specific source considered trusted. Nonetheless, Biquality Learning can be generalized to a Multi-Quality Learning setup with multiple datasets of different untrustedness and one trusted dataset. Then multiple mapping functions can be designed to correct all these untrusted datasets to fit the trusted data distribution. Multi-Quality Learning could be an open research question as no literature has been found on this subject.

2.3.4 . Concept Drift

In machine learning, concept drift refers to the phenomenon where the statistical properties of the target variable change between training and prediction time, leading to a degradation in the performance of a model that was trained on past data. This can occur when the data distribution changes or the relationships between the features and the target variable change. Concept drift can be a challenge in machine learning because models must be continuously retrained or updated to maintain their accuracy and effectiveness.

The most significant difference between Biquality Learning and Concept Drift in practice relates to data availability in the training and prediction phases. In Biquality Learning, both concepts and data are available at training time, and only the trusted concept will be needed at prediction time. For Concept Drift, however, only the untrusted concept will be available at training, and the trusted concept will be available at prediction time in an online fashion as the untrusted concept fades away.

This difference in data availability makes Concept Drift literature unusable for Biquality Learning. Nonetheless, practical or theoretical ideas could be reused with some significant reworking, especially in evaluating machine learning models under concept drift. It could inspire ideas of new ways to synthetically corrupt datasets or formally verify the expected accuracy of Biquality Learning algorithms under various corruptions.

2.3.5 . Table of Domains

We propose to summarize the previous Section in the form of a table stating why each related domain are different from the Biquality Learning setup:

Related Domain	**Differences with Biquality Learning**
Transfer Learning	Assumes the usefulness of the untrusted task
Domain Adaptation	Different assumptions on the distribution shift
Multi-source Learning	Lack of knowledge of which source is trusted
Concept Drift	Trusted Data not available at training time

Table 2.1: Table of Related Domains of state-of-the-art

2.4 . State of the Art

Machine learning algorithms on biquality data have been developed in many weakly supervised learning domains. Some of these sub-domains are robust learning to label noise, learning under covariate shift, or transfer learning. Because of this diversity and sometimes different assumptions about the data available, there is no pre-existing state-of-the-art of Biquality Learning. Thus we propose a state-of-the-art of *Biquality Learning*, which is a novel word for a fragmented field which had no previously agreed conventions. It is organized by the intuitions used by the author to design their proposed algorithm. At the end of this state-of-the-art, we will propose another organization based on the formalization introduced at the beginning of this Chapter (see Section 2.2) and the corruption model that the algorithm can handle in the form of a table.

2.4.1 . Transition Matrices

In the Robust Learning to Label Noise literature, label noise models can be represented thanks to their transition matrices $\mathbf{T}$, which represents the probability of seeing a noisy label $\tilde{Y}$ given the true label Y and the features X:

$$\forall (i,j) \in [\![1,K]\!]^2, \forall x \in \mathcal{X}, \mathbf{T}_{i,j}(x) = \mathbb{P}(\tilde{Y} = j | Y = i, X = x)$$

Thus, an essential part of the literature has sought to estimate these transition matrices to correct the classifier's training procedure from noisy data.

One way to use these transition matrices is to correct the prediction of classifiers learned on noisy data. Given the law of total probability:

$$\mathbb{P}(\tilde{Y}|X) = \sum_{k=1}^{K} \mathbb{P}(\tilde{Y}|X, Y = k)\mathbb{P}(Y = k|X)$$

We can recognize a dot product between one row of the transition matrix and the output of a classifier learned on clean data: $\forall x \in \mathcal{X}, \tilde{f}(x) = \mathbf{T}^t(x) \cdot f(x)$. If $\mathbf{T}$ is invertible, we can recover the clean distribution thanks to a noisy classifier with the following formula: $\forall x \in \mathcal{X}, f(x) = \mathbf{T}^{-1}(x) \cdot \tilde{f}(x)$, which has been proposed in [213].

This formula provides many other ways to train clean classifiers with noisy data and a transition matrix. Noisy data can be reweighted [111, 186], noisy labels transformed to pseudo-clean labels [139], or the minimzed objective can be corrected when training against noisy labels [124, 178, 139].

Unfortunately, a considerable part of the estimators proposed to estimate these transition matrices have restrained themselves to estimate transition matrices for instance independent noise, with $\forall x \in \mathcal{X}, \mathbf{T}(x) = \mathbf{T}$.

To our knowledge, only one estimator using trusted data [79] improves the quality of the estimation over the numerous estimators based on untrusted data (noisy data) only [139, 203, 213, 228]. This estimator is based on a soft version of the usual confusion matrix used to estimate classifiers' accuracy:

$$\forall i \in [\![1,K]\!], \hat{T}_{(i,*)} = \frac{1}{|D_T^i|} \sum_{x \in D_T^i} f_U(x)$$

where $D_T^i = \{(x,y) \in D_T | y = i\}$ and f_U is a classifier learned on untrusted data.

This estimator compares the true label on the trusted data against the predicted probabilities by f_U for each class for a given sample. Then, these predictions are averaged per class. It is a soft version of the confusion matrix because the predicted probabilities are used instead of the predicted class.

Thanks to transition matrix estimators based on Biquality Data, it makes all this sub-literature of Robust Learning to Noisy Labels relevant for the Biquality Learning framework.

This class of algorithms is particularly efficient in correcting untrusted datasets with complex class-dependent noise. However, there is room for more efficient transition matrix estimators with instance-dependent label noise [201, 27] to make Biquality Learning algorithms that are truly uninformed of the corruption model.

2.4.2 . Radon-Nikodym Derivative

In the Empirical Risk Minimization (ERM) framework, when learning a classifier $f : \mathcal{X} \mapsto \mathcal{Y}$, we seek to minimize its risk $R(f)$ given a loss function $L : \mathcal{Y}^2 \mapsto \mathbb{R}$:

$$R(f) = \mathbb{E}[L(f(X), Y)] = \int L(f(X), Y) \, \mathrm{d}\mathbb{P}(X, Y)$$

However, when the true labels Y cannot be observed because of weaknesses of supervision, we can rewrite given the observed labels $\tilde{Y}$:

$$R(f) = \int \frac{\mathrm{d}\mathbb{P}(X, Y)}{\mathrm{d}\mathbb{P}(X, \tilde{Y})} L(f(X), \tilde{Y}) \, \mathrm{d}\mathbb{P}(X, \tilde{Y})$$

Thus, in order to minimize the true risk of the classifier on true but unobserved labels, we need to reweight the loss function on the observed labels by $\mathbb{P}(X, Y)/\mathbb{P}(X, \tilde{Y})$. This measure is called the Radon-Nikodym Derivative (RND) [131] of $\mathbb{P}(X, Y)$ with respect to $\mathbb{P}(X, \tilde{Y})$.

This reweighting scheme has particularly inspired the literature of covariate shift [65, 167], and many algorithms have been implemented to estimate the RND, regrouped in an umbrella named Density Ratio Estimators [169].

Nevertheless, the assumptions about the distribution difference between the observed and the unobserved data are different in the two fields, as cited in the previous Section. Indeed, in covariate shift, the ratio we seek to estimate is $\mathbb{P}(X)/\mathbb{P}(\tilde{X})$ meanwhile, in Biquality Learning it is $\mathbb{P}(Y|X)/\mathbb{P}(\tilde{Y}|X)$.

One solution is to rewrite the ratio using the Bayes Formula to highlight a ratio between the distribution of features :

$$\frac{\mathbb{P}(Y|X)}{\mathbb{P}(\tilde{Y}|X)} = \frac{\mathbb{P}(X|Y)\mathbb{P}(Y)}{\mathbb{P}(X|\tilde{Y})\mathbb{P}(\tilde{Y})}$$

This approach has been followed in [49], where they proposed an algorithm we called K-DensityRatio (KDR), which is a particular instance of the KDA algorithm we proposed in subsection 2.3.2. Especially, they correct the feature distribution by reweighting samples by β thanks to a density ratio estimator $e \in \mathcal{E}$.

In [49] Kernel Mean Matching (KMM) [83, 65] has been used as the Density Ratio algorithm, e, to handle covariate shift.

Another solution to estimate the RND is to use Transition Matrices $\mathbf{T}$ [111, 186]:

$$\frac{\mathbb{P}(Y|X)}{\mathbb{P}(\tilde{Y}|X)} = \frac{\mathbb{P}(Y|X)}{\mathbf{T}^t(X) \cdot \mathbb{P}(Y|X)} \approx \frac{\mathbf{T}^{-1}(X) \cdot \mathbb{P}(\tilde{Y}|X)}{\mathbb{P}(\tilde{Y}|X)}$$

Algorithm 2: K-DensityRatio (KDR)

Input: Trusted Dataset D_T, Untrusted Dataset D_U, Density Ratio Estimator Family $\mathcal{E}$, Probabilistic Classifier Family $\mathcal{F}$

for $k \in [\![1, K]\!]$ **do**
  Let $D_T^k = \{\forall (x, y) \in D_T \mid y = k\}$
  Let $D_U^k = \{\forall (x, y) \in D_U \mid y = k\}$
  Learn $e^k \in \mathcal{E}$ on D_T^k and D_U^k
for $(x_i, y_i) \in D_U$ **do**
  $\hat{\beta}(x_i, y_i) = e^{y_i}(x_i) \frac{|D_T^{y_i}|}{|D_U^{y_i}|}$
for $(x_i, y_i) \in D_T$ **do**
  $\hat{\beta}(x_i, y_i) = 1$
Learn $f \in \mathcal{F}$ on $D_T \cup D_U$ with weights $\hat{\beta}$

Output: f

However, this solution requires inverting the estimated Transition Matrix, which has yet to be investigated to the scope of instance-dependent corruptions and has shown to have subpar performances compared to an algorithm that does not require to invert it [139].

As with other reweighting algorithms, this class of approaches can be pretty efficient on instance-dependent corruptions. Nonetheless, reweighting samples is the less flexible approach to deal with corrupted samples, as the only two outcomes are keeping or amplifying the sample (affecting a weight of 1, or amplifying it with a weight > 1) and discarding (affecting a weight of 0). It is fundamentally unable to correct the sample and can struggle with label noise that flips class label.

In Chapter 4, we will expand the scope of this class of algorithms to distribution shifts when the ratio of interest is $\mathbb{P}(X, Y)/\mathbb{P}(X, \tilde{Y})$.

2.4.3 . Auxiliary Data Sources

Learning with Auxiliary Data Sources has been introduced in [194] to train machine learning models with a second auxiliary source of data drawn from a different distribution than the primary data source. It combines the training on the two sources by minimizing an objective function J, which combines the expected loss on the two sources D_1 and D_2 with a parameter γ:

$$J(f) = \mathbb{E}_{D_1}[L(f(X), Y)] + \gamma \mathbb{E}_{D_2}[L(f(X), Y)]$$

Typically, the γ parameter is optimized with cross-validation to maximize the risk of f, $R(f)$, on the primary source.

In the case of Biquality Learning, the untrusted dataset or auxiliary source, could be detrimental to the optimization procedure of the objective function. It has been proposed to change the loss used on the untrusted dataset to a combination

of a robust loss to label noise and a semi-supervised loss [76]. The drawback of the proposed approach is that it introduces another hyperparameter to balance the Robust Learning to Label noise loss and the Semi-Supervised loss.

2.4.4 . Small Loss Samples

When training loss-based models, such as neural networks, on label noise, it can be helpful to use the loss value of a training example to determine whether its label is noisy. Research has shown that deep neural networks can learn general and high-level patterns from the data before being affected by overfitting, mainly when label noise is present [5, 103]. However, when noisy examples are encountered later on in the training process, they are often associated with a high loss value and can have a significant impact on the training procedure [209]. To address this, one approach is to prioritize small loss and easy examples in the early stages of training and save high loss and difficult examples for later. This approach is known as Curriculum Learning and is achieved by using heuristic-based schemes [9]. Theses schemes are based on sample reweighting with a factor β which usually depends on the loss value of the sample and the fraction of the training procedure.

Self Paced Learning (SPL) [98] is an example of such algorithm, where all samples with a model f loss L below a given threshold λ are defined as easy. This threshold grows linearly with the number of iterations t of the learning procedure by a factor μ.

$$\begin{aligned} \beta_{\text{SPL}}(x_i, y_i, t) &= \mathbb{1}(L(y_i, f(x_i)) < \lambda_t) \\ \lambda_{t+1} &= \lambda_t * \mu \end{aligned} \tag{2.3}$$

Other approaches such as Hard Negative Mining (HNM) [50] have taken the complete opposite direction and have been found to be empirically efficient.

$$\begin{aligned} \beta_{\text{HNM}}(x_i, y_i, t) &= \mathbb{1}(L(y_i, f(x_i)) > \lambda_t) \\ \lambda_{t+1} &= \frac{\lambda_t}{\mu} \end{aligned} \tag{2.4}$$

Many more algorithms exist with hand-designed curricula, such as Focal Loss [106] or Linear Weighting [87]. The main issue with these methods is that the curriculum is hand designed. Sometimes the prior and bias of designers do not apply well to a given dataset, and practitioners need to try many schemes before finding the right fitting one.

Designing an algorithm that learns a curriculum from the data would be much more versatile and efficient. MentorNet [88] proposes a modern take on curriculum learning and design such an algorithm that learns a curriculum from Biquality Data instead of hand designing it. MentorNet is composed of a student and a teacher model. The student learns from the examples chosen by the teacher. The teacher learns to pick an example from the data, the label, and the loss of the student. Both learn iteratively, one after the other. The teacher network is trained

by learning to predict which samples are trusted or untrusted from the features described previously. Then the student learns from the untrusted data thanks to the curriculum provided by the teacher.

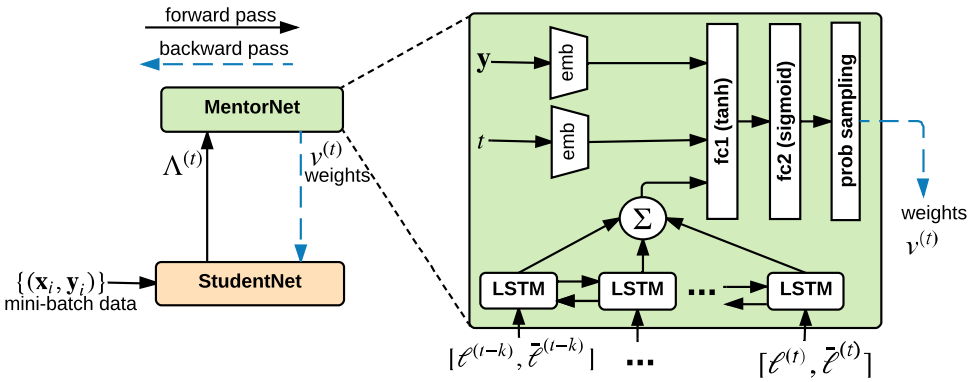


Figure 2.2: Mentor Net Architecture from [88]

2.4.5 . Meta Learning

Meta-Learning is the *learning to learn* field of Machine Learning [81]. It consists of an inner or base learning algorithm that seeks to solve a given task and an outer or meta-learning algorithm that updates the inner learning algorithm such that the model it learns improves an outer objective. Meta-Learning can be viewed as a Bi-Level Optimization [165] where one optimization contains another optimization as a constraint [54, 163]:

$$\omega^* = \arg\min_{\omega} L^{\text{outer}}(\theta^*(\omega), \omega, D_{\text{outer}}) \tag{2.5}$$

$$\text{s.t. } \theta^*(\omega) = \arg\min_{\theta} L^{\text{inner}}(\theta, \omega, D_{\text{inner}}) \tag{2.6}$$

where θ represents the parameters of the inner model and ω represents the *meta-knowledge*.

Meta-Learning approaches, especially from the view of Bi-Level Optimization, particularly fit the needs of Biquality Learning, especially through the framework of Section 2.2. In the Biquality Learning framework, we seek to learn a mapping function g which optimizes the outer performance of our model f learned on the inner mapped untrusted dataset.

Thanks to this observation, a myriad of Meta-Learning approaches for Biquality Learning has been designed. Most notably, approaches that seek to learn what we described in the previous Sections of this state-of-the-art instead of hand-designing estimators.

For example, approaches have been designed to learn Transition Matrices as if they were model parameters thanks to a Bi-Level optimization [162], sample

weights have been learned as embeddings of size 1 with Meta-Learning [149, 107], or curriculums have been learned from the data instead of hand designing them [161]. More specifically, people have learned such mapping function g that are able to correct labels of untrusted samples, such that the performance of f on the trusted task is optimized. For example, Meta Label Correction (MLC) [218] proposed to model g as a neural network that takes the input features or embeddings of samples and their untrusted labels and output a corrected label. Other approaches have tried to reuse efficient ideas from the Robust Learning to Label noise literature and modified the hand-crafted solutions to learned algorithms thanks to Meta-Learning with great success [195, 222, 171]

A considerable drawback of such approaches is the complexity of training them. Usually, these algorithms are trained with the Model Agnostic Meta Learning (MAML) framework [51], which is known to be computationally and memory intensive as it requires storing multiple iterations of the base models and going through it multiple times for each training iteration. Moreover, the MAML framework only works on base model and meta knowledge that are differentiable, which does not allow the use of the most famous shallow classifiers.

Though they are not trained precisely in the same manner as the previously cited algorithms, MentorNet [89] can also be considered a Meta-Learning algorithm, as it uses alternative optimizations.

2.4.6 . Table of methods

The Table 2.2 summarizes this State-of-the-Art Section that lists all known Biquality Learning algorithms with some additional details. The column *Family* corresponds to what kind of correction the algorithm makes to the untrusted dataset by referring to the three proposed methods earlier in the Chapter (Section 2.2). The *CAR* (Completely at Random), *AR* (At Random), and *NAR* (Not at Random) tell which corruption model the algorithm is capable of handling. Finally the Table is split into three parts, the first part list classifier-agnostic approaches, the second part is dedicated to Neural Networks only algorithms and the third part is for Meta-Learning approaches using the Bi-Level Optimization Framework.

From this Table 2.2, we can observe that, lately, the community has been focused on designing more clever and complex algorithms based on Meta-Learning and the Bi-Optimization Framework. The resulting algorithms are quite efficient on a vast number of tasks but are restrained to a class of machine learning algorithms that is capable of being trained with Meta-Learning (based on TensorFlow [1] or Pytorch [138]).

Moreover, we can observe that we did not find approaches that tried to model the mapping function g to move samples to the correct feature point. Making more connections between the Domain Adaptation and Biquality Learning literature could be an interesting new research path.

	Algorithms	Family	CAR	AR	NAR
Agnostic	Backward [124, 139] (with GLC [79])	Reweighting (3)	✓	✓	×
	Plugin [213] (with GLC [79])	Correction at Prediction (∅)	✓	✓	×
	IR [111, 186] (with GLC [79])	Reweighting (3)	✓	✓	×
	DIW [49]	Reweighting (3)	✓	✓	✓
DNN	Forward [139] (with GLC [79])	Loss Correction (∅)	✓	✓	×
	Mixmixup [76]	Robust Loss (∅)	✓	×	×
	MentorNet [88]	Reweighting (3)	✓	✓	✓
Meta-Learning	MW-Net [161]	Reweighting (3)	✓	✓	×
	L2RW [149] (SOSELETO [107])	Reweighting (3)	✓	×	✓
	MLC [218]	Correcting (1)	✓	✓	×
	MSLC [195]	Correcting (1)	✓	✓	×
	L2B [222]	Correcting (1)	✓	✓	✓
	WarPI [171]	Correcting (1)	✓	✓	✓

Table 2.2: Table of Biquality Learning State-of-the-Art.

2.5 . Biquality Datasets

A Natural Biquality Dataset consists of two datasets, one with correctly labeled samples for a supervised classification task that can be the trusted dataset and another with corrupted labels that can be the untrusted dataset. There are three well-known Biquality Datasets up to this date that are publicly available to benchmark Biquality Learning algorithms. They are all datasets for Image Classification with web-scrapped images. As these datasets are huge, containing up to a million images, it was impossible to label all of them correctly. The labels assigned to each image correspond to the query made to the search engine. Yet, for all three datasets, a small part of the dataset has been cleanly annotated, constituting a trusted dataset.

The most widely known of them is called Clothing1M [199]. This dataset of web-crawled clothing images from several online shopping websites comprises 1M untrusted samples, and 14K trusted samples with an approximate noise rate of 38.5%.

The WebVision [104] dataset is another dataset that contains 2.4M untrusted images crawled from Flickr and Google with 50K trusted images classified in 1000 classes and an approximate noise rate of 18%.

Food-101N [101] is an image dataset containing about 310K untrusted images of food recipes classified in 101 classes along 5K trusted labels with a noise rate of 20%. It is a bigger version of the Food-101 [13] dataset with way more noisy images and a trusted dataset.

In addition to these three datasets, two datasets have been recently published, CIFAR-10N and CIFAR-100N [189] which are noisy versions of the same usual CIFAR-10 and CIFAR-100 datasets. For these two datasets, the usual clean version of the label is available in addition to a label annotated by a human using Amazon

Dataset	$\lvert D_T \rvert$	$\lvert D_U \rvert$	$\hat{q}$	$\lvert \mathcal{Y} \rvert$
Clothing1M [199]	14K	1M	0.6	14
Food-101N [101]	5K	310K	0.8	101
WebVision [104]	50K	2.4M	0.8	1000

Table 2.3: Natural Biquality Datasets used for the evaluation. Columns: number of trusted examples ($|D_T|$), number of untrusted examples ($|D_U|$), estimation of untrusted quality ($\hat{q}$), and number of classes ($|\mathcal{Y}|$).

Mechanical Turk. Thus these datasets can be randomly split into a trusted and untrusted dataset, with the trusted dataset getting labels from the clean CIFAR-10 and CIFAR-100 datasets and the untrusted dataset getting the label from CIFAR-10N and CIFAR-100N.

2.6 . Simulated Deficiencies

If natural Biquality datasets are a good way to benchmark Biquality Learning algorithms' efficiency in the real world, they are limited for two reasons. First, there is yet to be a sufficient amount of Biquality datasets to significantly compare two algorithms to decide which one performs statically better thanks to statistical tests. Second, the quality and ratio of trusted data are set in advance because of their nature, and it is harder to understand *when* a Biquality Learning algorithm performs better than another. That is why simulating artificial deficiencies in usual classification datasets is useful when evaluating Biquality Learning algorithms.

To create a Biquality dataset from a public classification dataset supposedly clean, the first part consist in splitting the dataset into two parts using a stratified random draw, where p is the percentage for the trusted part. Then, the trusted dataset is left untouched, whereas corrupted labels are simulated in the untrusted dataset using different techniques. Usually, these various corruptions will be parametrized by a corruption strength.

2.6.1 . Label Noise

Label noise is a great way to corrupt samples in a practical, easy-to-understand, and reproducible manner.

As stated earlier in this Section, label noise models can be represented thanks to their transition matrices $\mathbf{T}$, which represents the probability of seeing a noisy label $\tilde{Y}$ given the true label Y and the features X:

$$\forall (i,j) \in [\![1,K]\!]^2, \forall x \in \mathcal{X}, \mathbf{T}_{i,j}(x) = \mathbb{P}(\tilde{Y} = j | Y = i, X = x)$$

In order to more easily define label noise patterns with a measurable strength, the transition matrix is usually split into a convex combination of two matrices: the identity matrix $\mathbf{I}$ and a corruption matrix $\mathbf{C}$ which is row stochastic, with a factor $\rho : \mathcal{X} \mapsto [0,1]$:

$$\forall x \in \mathcal{X}, \mathbf{T}(x) = (1 - \rho(x))\, \mathbf{I}_K + \rho(x)\mathbf{C}$$

The factor ρ defines the overall corruption strength of the synthetic label noise, which locally can depend on the feature values. In particular if $\rho(x) = 0$, then $\mathbf{T}(x) = \mathbf{I}_K$ and no noise is generated, whereas if $\rho(x) = 1$, then $\mathbf{T}(x) = \mathbf{C}$ and generated labels are completely noisy. The overall corruption strength is defined by $q = 1 - \int_{\mathcal{X}} \rho$.

One of the commonly used corruption matrices is the uniform matrix:

$$\mathbf{C} = \frac{1}{K}\mathbf{J}_K$$

The uniform matrix is the matrix of ones normalized by the number of classes K [111, 139]. When a sample is corrupted, it gets assigned a value sampled randomly from the set of all possible labels. At ρ constant, the uniform matrix leads to a **Completely At Random** corruption as the noisy label is independent of the class value ($\mathbf{C}$ is column stochastic).

Another widely used corruption matrix is the background matrix:

$$\mathbf{C} = \mathbf{E}_{*J}$$

The background matrix is a matrix of zeros with a column $J \in [\![1, K]\!]$ filled with ones [149]. When a sample is corrupted, the label J is automatically assigned. At ρ constant, the background matrix leads to a **At Random** corruption as samples already labeled J cannot be corrupted ($\mathbf{C}$ is not column stochastic).

Finally, another used corruption matrix is the flip matrix:

$$\mathbf{C} = \mathbf{P}_K$$

The flip matrix is a permutation matrix with exactly one entry of 1 in each row and each column and 0s elsewhere [161]. When a sample is corrupted, it gets assigned automatically to the associated noisy label of its clean class. The flip matrix obviously leads to a class-dependent or **At Random** corruption at ρ constant.

The factor $\rho : \mathcal{X} \mapsto [0, 1]$ will tell if the generated label noise is instance-dependant. If not constant, it could be designed using classifier uncertainty or even the density of the feature distribution [197].

2.6.2 . Weak Labels

Using weak labels to corrupt a dataset can be useful for simulating the scenario of label noise that may arise from using a less accurate classification system. In real-world applications, it is common for labels to be provided by sources that are not 100% accurate, for example, scraping image labels from surrounding text on web pages. These labels may still provide valuable information but may also introduce noise into the dataset if not corrected.

In practice, weak labels can be generated by training a classifier on the untrusted dataset and using its predictions to relabel untrusted samples. To vary the corruption strength, the classifier can be trained on a restricted train dataset

instead of the entire untrusted dataset. Finally, instead of assigning to untrusted samples the predicted label by the classifier, one can sample a label from the distribution of the predicted probabilities of the classifier [79].

2.6.3 . Data Poisoning

Data Poisoning are inputs to a machine learning model that have been specifically crafted to cause the model to make a mistake. These inputs are typically very similar to valid inputs but have small, carefully chosen perturbations that cause the model to misclassify them. Data Poisoning would be the most potent way to corrupt the untrusted dataset synthetically [216, 198].

2.7 . Evaluation of Biquality Learning Algorithms

In order to assess the efficiency of Biquality Learning algorithms, we have to be careful on two points. First, we need to check that the proposed methods verify some required behaviors that are listed under the form of baselines in the following Subsection. Then we will need to define what makes a Biquality Learning better than another.

2.7.1 . Baselines

We propose the three following baselines that Biquality Learning algorithms need to beat:

- *Trusted Only*: The final classifier f obtained with our algorithm should be better than a classifier f_T that learned only from the trusted dataset, insofar as untrusted data bring useful information about the trusted concept. At least, f should not be worse than using only trusted data.

- *No Correction*: The final classifier f should be better than a classifier f_{NC} learned from both trusted and untrusted datasets without correction. A biquality learning algorithm should leverage the information provided by having two distinct datasets. Meaning the ability to efficiently correct low-quality untrusted datasets and not get degraded by high-quality untrusted datasets (negative transfer) [105].

- *Semi-Supervised*: The final classifier f should be better than a classifier f_{ST} learned from both trusted and untrusted datasets, using a semi-supervised approach by discarding untrusted labels. A biquality learning algorithm should leverage the information provided by the untrusted labels.

2.7.2 . What Makes an Efficient Biquality Learning Algorithm ?

Telling if a Biquality Learning algorithm is efficient is not a straightforward task. Indeed, in usual supervised learning, we can compare the test accuracy of two models coming from different training algorithms and test which accuracy is the

highest. However, if we do that we could get only a partial response for Biquality Learning algorithms. For example, when willing to compare the *Untrusted Only* and *Trusted Only* on a biquality dataset, if the quality of the untrusted dataset is high, the *Untrusted Only* baseline will win because of the number of untrusted samples, and if the quality is low, the *Trusted Only* baseline will win. We thus need to look at two metrics:

- The average accuracy over all tested qualities to assess the bias of the algorithm, meaning the intrinsic efficiency to leverage biquality data.
- The standard deviation of the accuracy over all tested qualities to asses the variance of the algorithm, meaning its robustness to different qualities.

The same reasoning can be applied to the ratio of trusted data.

That is why using synthetic corruptions is important for assessing the efficiency of Biquality Learning algorithms, as they allow the complete control of both the ratio of trusted data and the quality of untrusted labels. It is thus important to report each metrics for every tested p and q when comparing two competitors. Yet, it is still important to report the global metric for the sake of ranking multiple approaches.

To do so, the literature usually uses the error curve where the accuracy of the classifier is plotted against different corruption strength.

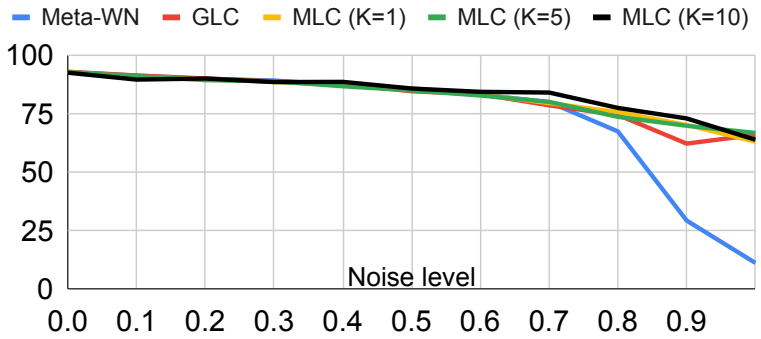


Figure 2.3: Example of on Error Curve plotting the accuracy w.r.t to noise level on uniform label noise on CIFAR10 from [218]

This plot allows us to see both the mean performance of the algorithm and it's variation when the corruption strength grow. Then, the curve is summarized thanks to the area under the curve to have an overall performance of the Biquality Learning algorithm. Finally, these experiments are conducted multiple times with different ratio of trusted data and are aggregated in a table. Still, this way to compare two competitors seems to lack statistical significance and does not tell on which configurations of p and q one competitor is beating another. It seems that the literature is lacking a true statistical testing approach to compare two

Biquality Learning Algorithms. We will propose in Chapter 3 and 4 a rigorous approach based on the Wilcoxon signed-rank test [191] for this problem.

2.8 . SotA Limits for Orange

From what has been presented in this Chapter, we can see some of the limits of the actual state-of-the-art of Biquality Learning for Orange.

Most notably, the community has been focused on designing more clever and complex algorithms based on Deep Neural Networks. Yet, classic shallow machine learning models (such as those implemented in Scikit-Learn [140]) are still the better approach for tabular classification tasks [67], which are of interest to Orange, a company disposing of many of such datasets. The field needs algorithms that are genuinely classifier agnostic.

Moreover, all existing Biquality Datasets are restricted to image classification. Yet again, one of the problems of datasets for image classification is to encourage the use of Deep Neural Networks as shallow classifiers under-performs on this task. Additionally, the low number of existing Biquality Datasets could produce noisy outcomes when using statistical tests to compare to competitors. Thus, it would be preferable to resort to synthetic corruptions on tabular classification datasets to improve the number of comparisons between algorithms and to feed the need for Orange better.

2.9 . Conclusion

In this Chapter, we proposed a generic formalization for the problem of Biquality Learning based on a mapping function that enables us to learn the true concept from the modified untrusted examples. We identified three ways of designing the mapping function: correcting the label of untrusted examples, moving untrusted samples in the input space, or reweighting untrusted samples. We examined the relationship between other fields of Machine Learning that focus on utilizing additional information and Biquality Learning. We proposed a state-of-the-art of Biquality Learning approaches organized by the intuitions used by their authors. We then reviewed the existing Biquality Datasets in the wild and ways to craft them synthetically. Finally, we exposed how to properly benchmark them and understand their strength and weakness.

In Chapter 3, we will propose an algorithm for Importance Reweighting for Biquality Learning (IRBL) and do an extensive benchmark on a vast number of datasets and synthetic corruptions to prove the efficiency of the proposed approach.

3 - Importance Reweighting for Biquality Learning

Contents

This Chapter is an extension of a publication of the authors: Pierre Nodet, Vincent Lemaire, Alexis Bondu, Antoine Cornuejols, and Adam Ouorou. Importance reweighting for biquality learning. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

3.1 . Introduction

When training supervised machine learning models in real world, trusted labeled samples might not be available in sufficient quantity. One solution is to obtain labeled samples from worse quality and untrusted sources. Nevertheless, these samples could have been badly corrupted and must be corrected before being used to train a machine learning model.

In Chapter 2, we proposed the Biquality Learning framework to correct untrusted samples via a mapping function g from the untrusted dataset D_U to the trusted dataset D_T. g should modifies the empirical risk R of a given loss function L such that minimizing R on D_U with the new objective $g \circ L$ should be equivalent to minimize R with the objective L on D_T. This function can take several particular forms, notably the following: $g(L(f(X), Y)) = g'(Y, X)L(f(X), Y)$ defined in Equation 2.2.

In this case, the mapping function g' defines a reweighting scheme for the untrusted samples such that the untrusted distribution they are sampled from is empirically corrected to the trusted distribution. From measure theory, we know that such a reweighting scheme always exists and is unique and is called the Radon-Nikodym derivative (RND) [131] of $\mathbb{P}_T(X, Y)$ with respect to $\mathbb{P}_U(X, Y)$:

Theorem 3.1.1 (Radon-Nikdoym-Lebesgue theorem [153])**.** *Let μ and ν two positive σ-finite measures defined on a measurable space $(X, \mathcal{A})$ with ν being absolutely continuous with respect to μ. Then there exists a unique positive measurable function f defined on X such that:*

$$\forall A \in \mathcal{A}, \nu(A) = \int_A f d\mu \tag{3.1}$$

Typically, machine learning datasets form measurable spaces, and probability densities are positive finite measures on these measurable spaces. Thus the mapping function g' exists, is unique, and is the RND of $\mathbb{P}_T(X, Y)$ with respect to $\mathbb{P}_U(X, Y)$.

As such, in this Chapter, we propose a new algorithm, IRBL, to estimate the Radon-Nikodym derivative between the two datasets in the context of Biquality Learning.

Section 3.2 introduces our proposed algorithm for Importance Reweighting for Biquality Learning (IRBL). Section 3.3 describes our experimental protocol to evaluate Biquality Learning algorithms. Section 3.4 is dedicated to an extensive benchmark of the proposed algorithm against state-of-the-art biquality learning algorithms. Experiments are conducted on a vast number of datasets under different and complex corruptions models, such as instance dependent label noise. Then, Sections 3.5 and 3.6.1 will open two discussions about the methodology and the efficiency of the proposed algorithm IRBL. Section 3.5 will study the impact of the calibration of the classifiers used for IRBL (readers will understand why in Section

3.2). Next, Section 3.6 studies the impact of the expressiveness (i.e VC dimension [180]) of classifiers and its capacity to improve with a growing training set on IRBL. Finally, Section 3.7 will extend experiments of Section 3.4 to multi-class classification datasets with class-dependent label noise.

3.2 . A new Importance Reweighting approach for Biquality Learning

Inspired by the importance reweighting trick from the covariate shift literature [112], we use the Radon-Nikodym Derivative (RND) [131] of $\mathbb{P}_T(X,Y)$ with respect to $\mathbb{P}_U(X,Y)$ which is $\frac{\mathrm{d}\mathbb{P}_T(X,Y)}{\mathrm{d}\mathbb{P}_U(X,Y)}$ as a reweighting scheme for the untrusted dataset. Equation 3.2 shows that minimizing the reweighted empirical risk by the RND on the untrusted data is equivalent to minimizing the empirical risk on trusted data.

$$\begin{aligned} R_{(X,Y)\sim T,L}(f) &= \mathbb{E}_{(X,Y)\sim T}[L(f(X),Y)] \\ &= \int L(f(X),Y)\,\mathrm{d}\mathbb{P}_T(X,Y) \\ &= \int \frac{\mathrm{d}\mathbb{P}_T(X,Y)}{\mathrm{d}\mathbb{P}_U(X,Y)}\,L(f(X),Y)\,\mathrm{d}\mathbb{P}_U(X,Y) \\ &= \mathbb{E}_{(X,Y)\sim U}[\frac{\mathbb{P}_T(X,Y)}{\mathbb{P}_U(X,Y)}L(f(X),Y)] \\ &= \mathbb{E}_{(X,Y)\sim U}[\beta\,L(f(X),Y)] \\ &= R_{(X,Y)\sim U,\beta L}(f) \end{aligned} \tag{3.2}$$

By using Biquality Learning hypothesis, we can simplify β using the Bayes Formula. Indeed, it assumes that the distribution of the features $\mathbb{P}(X)$ between the two datasets is the same (the impact of such a difference will be studied in Chapter 4). However, the two underlying concepts $\mathbb{P}_T(Y|X)$ and $\mathbb{P}_U(Y|X)$ are possibly different due to a supervision deficiency.

$$\beta(X,Y) = \frac{\mathbb{P}_T(X,Y)}{\mathbb{P}_U(X,Y)} = \frac{\mathbb{P}_T(Y|X)\mathbb{P}(X)}{\mathbb{P}_U(Y|X)\mathbb{P}(X)} = \frac{\mathbb{P}_T(Y|X)}{\mathbb{P}_U(Y|X)} \tag{3.3}$$

Finally, we propose Algorithm 3 to estimate β:

The proposed algorithm, Importance Reweighting for Biquality Learning (IRBL), aims to estimate β from D_T and D_U, whatever the unknown supervision deficiency. It consists of two successive steps. First, the vector of ratios between $\mathbb{P}_T(Y|X)$ and $\mathbb{P}_U(Y|X)$ is estimated by the term $\left\langle \frac{f_T(x_i)}{f_U(x_i)} \right\rangle$, using the models f_T and f_U, whose size K corresponds to the number of classes. For each untrusted example, the weight $\hat{\beta}$ is the y_i-th element of this vector[1] *(see line 4)*; while $\hat{\beta}$ is fixed to 1 for the trusted examples *(see line 6)*. Then, the final classifier is learned from $D_T \cup D_U$ with examples reweighted by $\hat{\beta}$.

[1] If $f_U(.) = 0$, then $\hat{\beta} = 0$ as in [112].

Algorithm 3: Importance Reweighting for Biquality Learning (IRBL)

Input: Trusted Dataset D_T, Untrusted Dataset D_U, Probabilistic Classifier Familiy $\mathcal{F}$

Learn $f_U \in \mathcal{F}$ on D_U

Learn $f_T \in \mathcal{F}$ on D_T

for $(x_i, y_i) \in D_U$ **do**

  $\hat{\beta}(x_i, y_i) = \left\langle \frac{f_T(x_i)}{f_U(x_i)} \right\rangle_{y_i}$

for $(x_i, y_i) \in D_T$ **do**

  $\hat{\beta}(x_i, y_i) = 1$

Learn $f \in \mathcal{F}$ on $D_T \cup D_U$ with weights $\hat{\beta}$

Output: f

It is noted that the proposed algorithm IRBL does not require assumptions on the nature of the untrusted dataset's corruption nor on its level which makes the approach truly **uninformed**.

Our algorithm is theoretically grounded since it is asymptotically equivalent to minimizing the risk on the true concept using the entire data set (see proof in Equation 3.4).

$$
\begin{aligned}
\hat{R}_{D,\hat{\beta}L}(f) &= \frac{1}{|D|} \sum_{(x_i,y_i)\in D} \left(\mathbb{1}_{(x_i,y_i)\in D_T} L(f(x_i), y_i) + \mathbb{1}_{(x_i,y_i)\in D_U} \hat{\beta}(x_i, y_i) L(f(x_i), y_i) \right) \\
&= \frac{1}{|D|} \sum_{(x_i,y_i)\in D_T} L(f(x_i), y_i) + \frac{1}{|D|} \sum_{(x_i,y_i)\in D_U} \hat{\beta}(x_i, y_i) L(f(x_i), y_i) \\
&= \frac{p}{|D_T|} \sum_{(x_i,y_i)\in D_T} L(f(x_i), y_i) + \frac{1-p}{|D_U|} \sum_{(x_i,y_i)\in D_U} \hat{\beta}(x_i, y_i) L(f(x_i), y_i) \\
&= p\hat{R}_{D_T,L}(f) + (1-p)\hat{R}_{D_U,\hat{\beta}L}(f) \\
&\approx p\hat{R}_{D_T,L}(f) + (1-p)\hat{R}_{D_T,L}(f) \\
&\approx \hat{R}_{D_T,L}(f)
\end{aligned}
\tag{3.4}
$$

Proof in Equation 3.4 is an asymptotic result: in practice, our algorithm relies on the quality of the estimation of $\mathbb{P}_T(Y|X)$ and $\mathbb{P}_U(Y|X)$ in order to be efficient. In the biquality setting, they both could be hard to estimate because of the small size of D_T and the poor quality of D_U. Furthermore choosing the right family of classifier in terms of calibration and specification matters and is studied in Section 3.5 and Section 3.6.

3.3 . Simulating Supervision Deficiencies

In order to evaluate and benchmark the proposed IRBL algorithm against state-of-the-art competitors, we need to have fine-grained control over the size ratio between the trusted and untrusted dataset and the strength of the corruption of the untrusted dataset.

3.3.1 . Datasets

To do so, we selected a list of public datasets for classification. In Section 3.4, these datasets will be restricted to binary classification and will be extended to multi-class classification in Section 3.7.

Moreover, in this chapter, we decided to restrict our experiments to tabular datasets, in opposition to images or text datasets. Indeed, in industrial applications familiar to us, such as fraud detection, Customer Relationship Management (CRM), and churn prediction, we are mainly faced with binary classification problems. The available data is of average size regarding the number of explanatory variables and involves mixed variables (numerical and categorical).

The tabular datasets for binary classification are listed in Table 3.1.

Table 3.1: Binary classification datasets used for the evaluation. Columns: number of examples ($|\boldsymbol{D}|$), number of features ($|\boldsymbol{\mathcal{X}}|$), number of classes ($|\boldsymbol{\mathcal{Y}}|$), and ratio of examples from the minority class (**min**).

Datasets	$\lvert\boldsymbol{D}\rvert$	$\lvert\boldsymbol{\mathcal{X}}\rvert$	$\lvert\boldsymbol{\mathcal{Y}}\rvert$	**min**	**Datasets**	$\lvert\boldsymbol{D}\rvert$	$\lvert\boldsymbol{\mathcal{X}}\rvert$	$\lvert\boldsymbol{\mathcal{Y}}\rvert$	**min**
ad	3K	1558	2	0.14	svmguide3	1K	22	2	0.24
eeg	15K	14	2	0.45	web	37K	123	2	0.24
ibn_sina	21K	92	2	0.38	mushroom	8K	22	2	0.48
zebra	61K	154	2	0.05	skin-segmentation	245K	3	2	0.21
musk	6K	167	2	0.15	mozilla4	16K	5	2	0.33
phishing	11K	30	2	0.44	electricity	45K	8	2	0.42
spam	5K	57	2	0.39	bank-marketing	45K	16	2	0.12
ijcnn1	191K	22	2	0.10	magic-telescope	19K	10	2	0.35
diabetes	768	8	2	0.35	phoeneme	5K	5	2	0.29
credit-g	1K	20	2	0.30	poker	1M	10	2	0.50
hiva	43K	1617	2	0.04					

The datasets in Table 3.1 come from different sources: UCI [42], libsvm [20], and active learning challenge [70]. A part of these datasets comes from past challenges on active learning, where high performances with a low number of labeled examples have proved challenging to obtain. With this choice of datasets, a large range of the class ratio is covered with different levels of imbalance. Also, the number of rows and columns varies significantly, with a corresponding impact on the difficulty of the learning tasks.

3.3.2 . Simulated Supervision Deficiencies

To obtain a trusted dataset D_T and an untrusted one D_U, each dataset is split into two parts using a stratified random draw, where p is the percentage for the trusted part. The trusted datasets are left untouched, whereas corrupted labels are simulated in the untrusted datasets using different techniques.

This chapter will focus on label noise, although other corruptions could have been tested such as concept drift [115] or data poisoning [155]. Label noise models can be represented thanks to their transition matrices T, with $\forall(i,j) \in [\![1,K]\!], \forall x \in \mathcal{X}, \mathbf{T}_{i,j}(x) = \mathbb{P}(\tilde{Y} = j | Y = i, X = x)$. The two label noise models we are going to focus on are:

- Noisy Completely At Random (NCAR): Corrupted untrusted examples are uniformly drawn from D_U with a probability $\rho \in [0,1]$ and are assigned a random label that is also uniformly drawn from $\mathcal{Y}$. The corresponding transition matrix is : $\mathbf{T} = (1-\rho)\mathbf{I}_K + \frac{\rho}{K}\mathbf{J}_K$, where $\mathbf{I}_K$ is the identity matrix, $\mathbf{J}_K$ is the unit matrix, and ρ is the noise ratio. Label noise completely at random is both instance independent, as $\mathbf{T}$ does not depend on x, and class independent, as $\mathbf{T}$ is column stochastic.

- Noisy Not At Random (NNAR): Corrupted untrusted examples are drawn from D_U with a probability measure $\rho : \mathcal{X} \mapsto [0,1]$ that depends on the instance value and are assigned a random label that is also uniformly drawn from $\mathcal{Y}$. The corresponding transition matrix is : $\forall x \in \mathcal{X}, \mathbf{T}(x) = (1 - \rho(x))\mathbf{I}_K + \frac{\rho(x)}{K}\mathbf{J}_K$, where $\mathbf{I}_K$ is the identity matrix, $\mathbf{J}_K$ is the unit matrix, and ρ is the noise probability function. This model of label noise not at random is still class independent but not instance independent anymore.

As an instance of the ρ function for NNAR, we propose to use the uncertainty function from the Active Learning literature [158]. Uncertainty functions are used in sampling processes to get annotations from a human for samples in which the trained model has the least confidence. Several examples of such functions are the *uncertainty*, *margin*, or the *entropy* function.

We propose to use such functions as our noise probability function because we want to model label noise from human error, where annotators would be more likely to make an error when annotating a sample for which they are uncertain of their true label. As such, we will use the uncertainty function in the following experiments: $\rho(x) = 1 - \max_{y \in \mathcal{Y}} \mathbb{P}(Y = y | X = x)$. To model the human annotator, or the distribution $\mathbb{P}(Y|X)$, we will preemptively learn a model f on the whole dataset D unmodified.

However, by using directly the noise probability function $\rho(x)$, we cannot control the average noise probability or noise ratio of the untrusted dataset. Thus in these experiments, we are going to use $\rho(x)^\theta$ as the probability function with θ being optimized such that $\mathbb{E}[\rho(X)^\theta] = 1 - q$. Similarly in the NCAR case with $\rho(x) = \rho$, we have $\rho = 1 - q$.

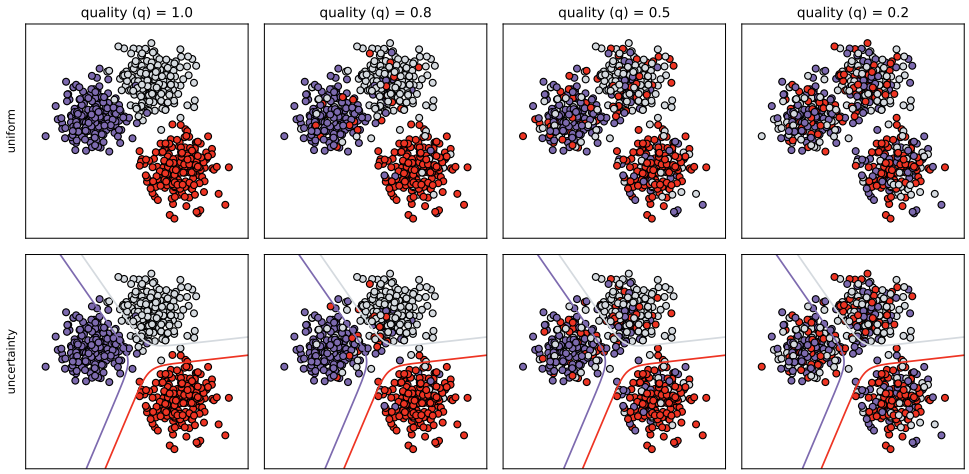


Figure 3.1: Artificial dataset perturbed by different label noises, NCAR (*uniform*), and NNAR (*uncertainty*) with decreasing quality (1 to 0.2).

Figure 3.1 illustrates the proposed noise model. This figure represents an artificial dataset composed of three blobs, each corresponding to one of three classes illustrated with different colors. Each column corresponds to a different quality q, and each row corresponds to a different noise model: the first line is NCAR, and the second line is NNAR. On the second line were added decision boundaries of One Versus Rest linear classifiers denoted by colored lines (corresponding to the "one" class).

From Figure 3.1, we can see the effect of using $\rho(x)$ instead of a constant ρ by looking at the differences between the first and the second line. In the second line, samples closest to the decision boundaries have a way higher chance of getting corrupted, making the center of each Figure noisier.

3.4 . Experiments

The experiments aim to answer the two following questions:

1. Is our algorithm properly designed?
2. Does it perform well?

We will answer these two questions by empirically evaluating the quality of our reweighting scheme and then comparing the proposed algorithm IRBL against the state-of-the-art.

3.4.1 . Quality of the Reweighting Scheme

At first, an example dataset is used as cherry-picking to graphically showcase the proposed algorithm's performances and properties. The *ad* dataset from Table 3.1 is used with a ratio of trusted data $p = 0.1$ corresponding to 10% of trusted data.

To see if the proposed reweighing scheme works properly, Figure 3.2 shows the histogram of the weights assigned to each untrusted example, either corrupted or not, when quality $q = 0.5$ with NCAR.

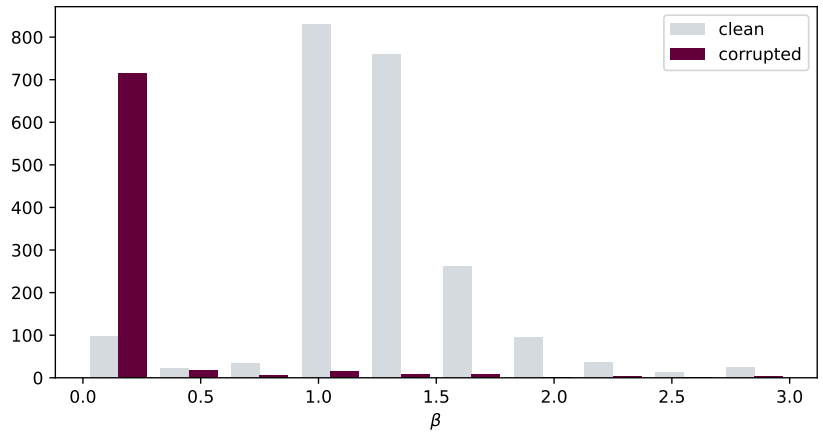


Figure 3.2: Histogram of the β values on AD for $p = 0.1$ and $q = 0.5$ for NCAR for the clean and corrupted untrusted examples.

Two densities are displayed in Figure 3.2 in the form of histograms. The first one is the distribution of the estimated $\hat{\beta}$ of clean untrusted samples (samples with label unmodified by NCAR), and the second one is the distribution of the estimated $\hat{\beta}$ of corrupted untrusted samples (samples with label modified by NCAR).

The density of $\hat{\beta}$ for the corrupted samples is nearly unimodal and close to 0. On the other hand, the $\hat{\beta}$ distribution stays around 1 as the assigned weights for the trusted samples.

It is clear from Figure 3.2 that our method can detect corrupted and non-corrupted labels from the untrusted dataset.

To get a broader picture, we can extend the previous Figure 3.2 by plotting the distribution of weights thanks to boxplots when the quality q decreases.

Figure 3.3 shows that the weight β estimated for corrupted samples follows the proper behavior, having all weights close to 0 for these samples for all qualities. If the used classifier can adequately use these weights, the corrupted samples will not perturb the training procedure.

On the other hand, the weight of clean samples tends to have a higher variance when the quality decrease. This behavior could lead to clean samples being wrongly

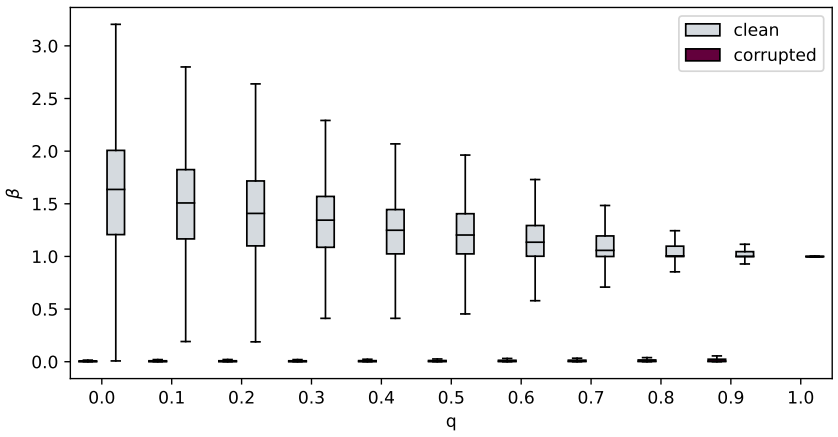


Figure 3.3: Boxplot the β values on AD for $p = 0.1$ versus the quality, from $q = 0$ to $q = 1$ for NCAR for the clean and corrupted untrusted examples..

removed from the training procedure, or could make clean untrusted samples have more importance than trusted samples during classifier learning.

3.4.2 . Benchmark against State-of-the-art-competitors

An extensive Biquality Learning state-of-the-art have been presented already in Chapter 2. From these extensive state-of-the art we decided to pick one uniformed algorithm that works on Biquality Data with label noise like corruptions named the Noise Corrected Plugin approach [213]. The specificity of this method is that it does not imply any property of the base learner as IRBL, which makes it suited to learn tabular classifiers on the datasets from Table 3.1. Moreover, as the following experiments are particularly conducted on label noise corruptions, we decided to pick an additional uniformed method from the Robust Learning to Label noise, the Unhinged Classifier[177, 24].

- *Unhinged*: In recent literature, a new emphasis is put on the research of new loss functions that are conducive to better risk minimization in the presence of noisy labels. For example, [177, 24] show theoretically and experimentally that when the loss function satisfies a symmetry condition, described below, this contributes to the robustness of the classifier on label NCAR. A loss function L is said *symmetrical* if $\sum_{y\in\{-1;1\}} L(f(x), y) = c$, where c is a constant and $f(x)$ is the score on the class y. This loss function is used on $D_T \cup D_U$.

- *Noise Corrected Plugin Classifier*: To the best of our knowledge, Noise Corrected Plugin classifier [213] is among the best performing algorithm on biquality data. It learns a classifier f_U from the untrusted data and corrects the predictions made by the classifier with an estimated transition matrix of

the noise $f(x) = f_U(x)\mathbf{T}^{-1}$.

In the original version, the authors used an algorithm to estimate a transition matrix from untrusted samples only. In this benchmark, we propose to use an algorithm from the Biquality Learning literature to find a better estimate of the transition matrix: GLC [79]. GLC compares the predictions of the untrusted classifier f_U on the trusted dataset against the trusted labels thanks to a soft confusion matrix to estimate the noise transition matrix.

Additionally, we added three baselines in the benchmark as a sanity check:

- *Trusted Only*: The final classifier f obtained with our algorithm should be better than a classifier f_T that learned only from the trusted dataset, insofar as untrusted data bring useful information about the trusted concept. At least, f should not be worse than using only trusted data.
- *No Correction*: The final classifier f should be better than a classifier f_{NC} learned from both trusted and untrusted datasets without correction. A biquality learning algorithm should leverage the information provided by having two distinct datasets.
- *Self Training*: The final classifier f should be better than a classifier f_{ST} learned from both trusted and untrusted datasets, using a semi-supervised approach by discarding untrusted labels. A biquality learning algorithm should leverage the information provided by the untrusted labels.

  The Self-Training semi-supervised classifier works by iteratively learning a classifier on available labeled data and labeling unlabeled data with the predictions of the learned classifier. In practice, only the high-confidence predictions are kept, and a new iteration is made on the newly labeled samples until it reaches convergence.

To evaluate the competitors' performance, we use the same probabilistic classifier family, Logistic Regression from Scikit-Learn [140] optimized with the L-BFGS optimizer [110] and L2 Regularization.

First of all, the choice of classifiers was guided by the idea of comparing algorithms for biquality learning rather than searching for the best classifiers. This choice was also guided by the nature of the datasets used in the experiments (see section 3.1).

LR is known to be limited, in the sense of the Vapnik-Chervonenkis (VC) dimension [180] since it can only learn linear separations of the input space $\mathcal{X}$, which could underfit the conditional probabilities $\mathbb{P}(Y|X)$ on D_T and D_U and lead to bad β estimations. Nevertheless, if met, this impediment will equally affect all the compared algorithms. Section 3.6.1 will test classifiers with varying VC dimensions.

To obtain reliable estimations of conditional probabilities $\mathbb{P}(Y|X)$, the outputs of all classifiers have been calibrated thanks to Isotonic Regression provided by scikit-learn [140].

Finally, accuracy is the metric used to quantify their efficiency on the classification tasks.

3.4.3 . Results

For a first global comparison, two critical diagrams are presented in Figures 3.4 and 3.5, which rank the different methods for the NCAR and NNAR label noise. The Nemenyi test [125] is used to rank the different approaches in terms of mean accuracy, calculated for all values of p and q and over all the 20 datasets described in Section 3.1 and is used at 95 % level of confidence. The Nemenyi test consists of two successive steps. First, the Friedman test is applied to the mean accuracy of competing approaches to determining whether their overall performance is similar. Second, if not, the post-hoc test is applied to determine groups of approaches whose overall performance is significantly different from that of the other groups.

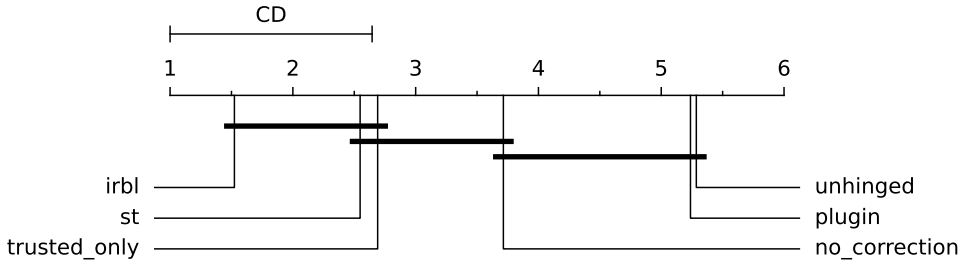


Figure 3.4: Nemenyi test for the 20 binary classification datasets $\forall p, q$ for NCAR.

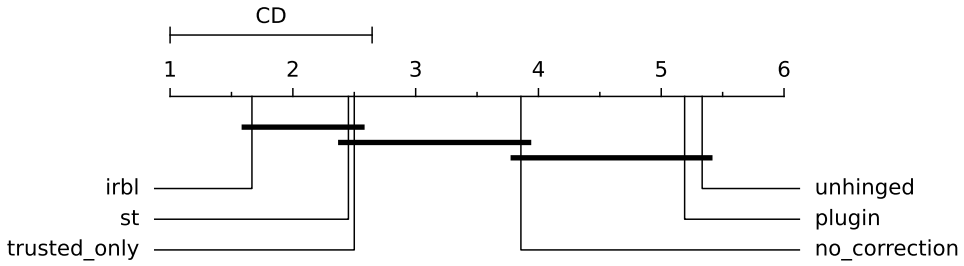


Figure 3.5: Nemenyi test for the 20 binary classification datasets $\forall p, q$ for NNAR.

The figures 3.4 and 3.5 show that the IRBL method is ranked first for the two kinds of label noise and performs better than the other competitors. Table 3.2

gives a more detailed perspective by reporting the mean accuracy for each trusted ratio p and corruptions.

The standard deviation reported in the table is the standard deviation of the accuracy of the competitor over all the quality values q. A lower standard deviation means that the competitor will always have the same performance, whatever the quality of the untrusted dataset.

	p	IRBL	No Correction	Plugin	Self Training	Trusted Only	Unhinged
(2)	0.02	**82.96 ± 0.54**	80.36 ± 7.75	77.51 ± 8.53	82.58 ± 0.0	82.71 ± 0.0	70.12 ± 3.72
	0.05	**83.24 ± 0.44**	81.59 ± 4.96	77.36 ± 8.64	83.00 ± 0.0	83.16 ± 0.0	70.99 ± 2.56
	0.10	**83.67 ± 0.35**	82.28 ± 3.26	76.89 ± 9.38	83.32 ± 0.0	83.41 ± 0.0	70.80 ± 1.94
	0.25	**84.08 ± 0.21**	83.46 ± 1.44	77.07 ± 8.42	83.74 ± 0.0	83.95 ± 0.0	71.09 ± 0.80
(1)	0.02	**82.93 ± 0.52**	80.01 ± 7.77	77.36 ± 8.80	82.58 ± 0.0	82.71 ± 0.0	70.58 ± 3.46
	0.05	**83.39 ± 0.48**	81.28 ± 4.85	77.16 ± 8.80	83.00 ± 0.0	83.16 ± 0.0	71.18 ± 2.12
	0.10	**83.71 ± 0.39**	81.95 ± 3.32	76.77 ± 9.80	83.32 ± 0.0	83.41 ± 0.0	71.12 ± 1.59
	0.25	**84.16 ± 0.23**	83.23 ± 1.45	76.97 ± 8.59	83.74 ± 0.0	83.95 ± 0.0	71.43 ± 0.47
Mean		**83.56 ± 0.46**	78.85 ± 6.27	78.47 ± 6.85	83.16 ± 0.0	83.31 ± 0.0	69.55 ± 2.89

Table 3.2: Mean Accuracy (rescaled score to be from 0 to 100) and standard deviation computed on the 20 binary classification datasets $\forall q$ for (1) NCAR and (2) NNAR.

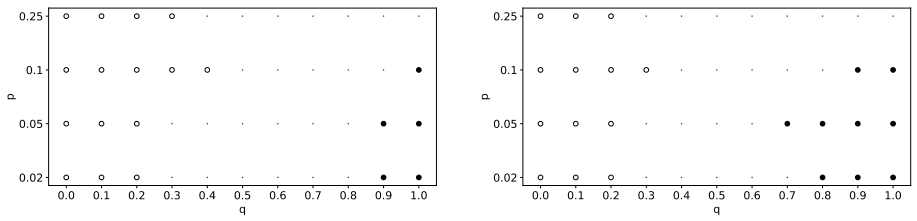
These values are computed for different values of p over all qualities q and all datasets. This table also helps to see how far the methods are compared to perfect total training data. Overall, IRBL obtains the best results with lower variability.

To have more refined results, the Wilcoxon signed-rank test [191] is used with a confidence level at 95 % to find out under which conditions – i.e., by varying the values of p and q – IRBL performs better or worse than the competitors.

Figure 3.6 presents eight graphics where each one represents the Wilcoxon test evaluating our approach against a competitor, based on the mean accuracy over the 20 datasets. The two types of label noise (see Section 2.6) correspond to the columns in Figure 3.6, and a wide range of q and p values are considered.

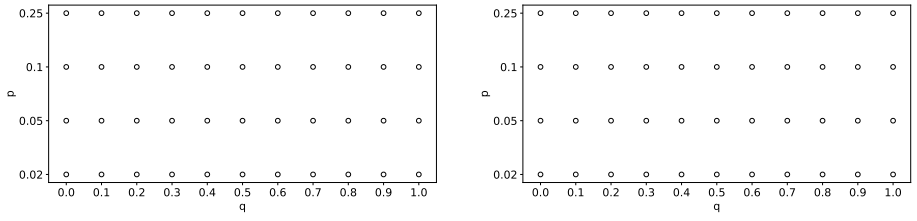
Thanks to these graphs, we can compare our method (IRBL) in more detail with the competitors. Concerning the No Correction method, Figures 3.6a and 3.6b indicate progressive results versus the couple value of (p, q). For low-quality values, whatever the value of p, IRBL is significantly better. For intermediate values of quality, there is no winner. For high-quality values and low values of p, the No Correction method is significantly better (this result seems to be observed in [79] as well). This result is not surprising since the No Correction baseline is equivalent to learning with perfect labels at high-quality value.

Regarding the competitors Unhinged and Plugin, Figures 3.6c, 3.6e, 3.6d and 3.6f show that IRBL is always better or indistinguishable to them. IRBL performs well regardless of the type of noise. This significant result allows us to deal not only with NCAR noise but also with instance dependent label noise (NNAR). Plugin has ties with IRBL when the quality is high, whatever the label noise, which is also an interesting result. The proposed method has been tested on a large variety and strength of label corruption. In each case, IRBL exhibits competitive results, and thus IRBL can be safely used in applications where biquality learning is needed.



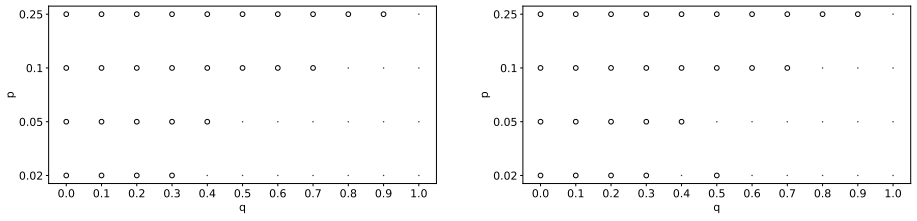
(a) IRBL vs NoCorrection for NCAR

(b) IRBL vs NoCorrection for NNAR



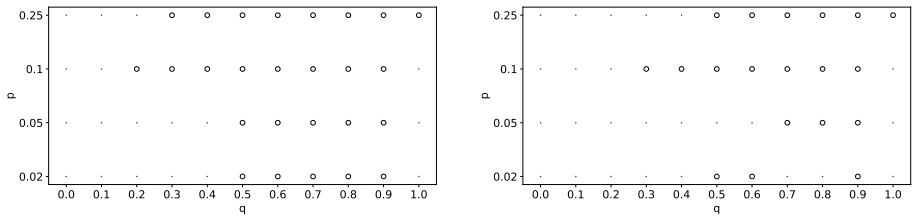
(c) IRBL vs Unhinged for NCAR

(d) IRBL vs Unhinged for NNAR



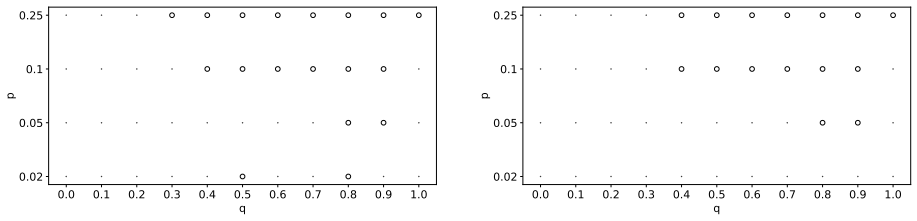
(e) IRBL vs Plugin for NCAR

(f) IRBL vs Plugin for NNAR



(g) IRBL vs Trusted Only for NCAR

(h) IRBL vs Trusted Only for NNAR



(i) IRBL vs Self Training for NCAR

(j) IRBL vs Self Training for NNAR

Figure 3.6: Results of the Wilcoxon signed rank test computed on the 20 datasets. Each Figure compares IRBL versus one of the competitors. In each figure "○", "·" and "●" indicate respectively a win, a tie or a loss of the first competitor compared to the second competitor. The vertical axis is p and the horizontal axis is q.

3.5 . On the Calibration of Classifiers

The proposed Algorithm 3, IRBL, estimates the Radon-Nikodym derivative (RND) between the trusted and the untrusted concepts by relying on classifiers *(see line 4 of Algorithm 3)*. The capacity of a classifier to estimate the posterior probability of a sample belonging to a given class refers to the calibration of the classifier. Intuitively, the classifier is said to be well-calibrated if the number of samples attributed to the positive class with a predicted probability p to be is equal to a fraction p of the members of the positive class [40].

In particular, most classifiers have the option to have as output a probability estimate of a given target class y conditional on the feature vector x denoted $\tilde{\mathbb{P}}(Y|X)$. Then, a more precise definition of being well-calibrated is given by the estimated probability $\tilde{\mathbb{P}}$ respecting the following equation:

$$\mathbb{P}\left(Y = y|\tilde{\mathbb{P}}(Y = y|X) = p\right) = p \tag{3.5}$$

In other words, in a calibrated classifier, the estimated conditional probability $\tilde{\mathbb{P}}(Y|X)$ is close to the one obtained with the underlying probability measure $\mathbb{P}(Y|X)$. Note that a classifier can be efficient without necessarily being calibrated.

In the case of IRBL, the calibration of the two classifiers f_T and f_U may impact the efficiency of the estimation of the RND, $\hat{\beta}$. In this Section, we will look at the impact of the calibration of f_T and f_U on $\hat{\beta}$.

3.5.1 . Under-Confidence and Over-Confidence

If a classifier is not well-calibrated, two situations may arise:

- The classifier can be under-confident in its classification, meaning that a predicted probability p corresponds to a smaller fraction $f < p$ of the positive class.
- The classifier can produce over-confident predicted probability, meaning that a predicted probability p corresponds to a more considerable fraction $f > p$ of the positive class.

Having under-confident predictions means that the predicted probabilities of the classifier tend to be centered around 0.5. This situation arises most notably with Support Vector Machines [143]. On the other hand, over-confident predictions mean that the classifier's predicted probabilities will be pushed to 0 and 1. This situation arises in many modern machine learning classifiers, especially Neural Networks and Gradient Boosted Trees learned with cross-entropy loss [69]. Indeed the cross-entropy loss naturally pushes the log odds to infinity, squishing the predicted probability distribution.

For IRBL, using under-confident or over-confident classifiers would lead to different errors in the estimated weights:

- If f_U is under-confident, the sole variability of $\hat{\beta}$ would come from f_T. Indeed, all f_U predictions would be close to each other around the decision boundary.

Thus, only f_T would be decisive on $\hat{\beta}$ being able to separate noisy and clean untrusted samples. So, the final classifier would only gain knowledge from D_T through f_T missing on a lot of additional knowledge.

- If f_U is over-confident, the variance of $\hat{\beta}$ would be very high, leading to numerical instability, especially for samples where f_U predictions would be close to 0. This numerical instability could worsen the efficiency of the final classifier.

3.5.2 . Calibrating non-calibrated Classifiers

Calibrating classifiers is a well-studied field of machine learning [129] with many different algorithms and metrics to correct and measure the calibration of classifiers. It usually involves learning a monotonic function that does not modify the outputted probability of the classifier $\tilde{\mathbb{P}}$ but rectify it to better fit the underlying probability measure $\mathbb{P}$.

In this Section, we chose to use the Isotonic Regression (IR) algorithm to calibrate classifiers [208], which is a non-parametric approach using regression with monotonic constraints.

Naturally, practitioners would use well-calibrated classifiers when using IRBL, the following study is akin to an ablation study of IRBL when used classifiers are over-confident or under-confident.

3.5.3 . Simulating Poorly Calibrated Classifiers

To evaluate the impact of under-confident, over-confident, and well-calibrated classifiers, we chose to reiterate the experiments from Section 3.4 with the same Logistic Regression (LR) from Scikit-Learn, which is known to be well-calibrated [46], and with two variants of this Logistic Regression.

The first variant will be called UnderConfident (UC) Logistic Regression, and the second one OverConfident (OC) Logistic Regression. To modify the confidence of the base Logistic Regression, we scale the predicted log odds $\boldsymbol{\theta X}$ by a factor α, with $\alpha < 1$ for the UC Logistic Regression and $\alpha > 1$ for the OC Logistic Regression. The training procedure is unchanged.

The following calibration plots show the impact of the factor α on the calibration quality of the Logistic Regression and show the efficiency of the Isotonic Regression to re-calibrate these perturbed Logistic regressions:

Figure 3.7 shows that UC Logistic Regression and OC Logistic Regression are not well-calibrated, whereas Logistic Regression is, as the calibrated UC Logistic Regression with Isotonic Regression.

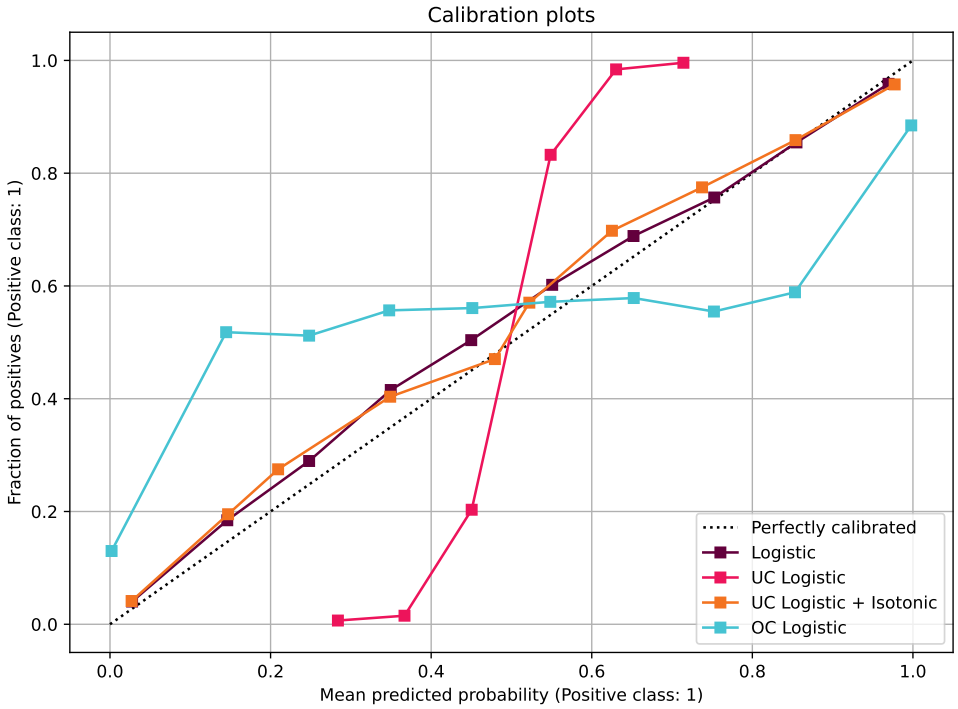


Figure 3.7: Calibration plots of poorly calibrated Logistic Regressions on an artificial dataset.

3.5.4 . Results

First, we can see the impact of weight estimation on a cherry-picked example:

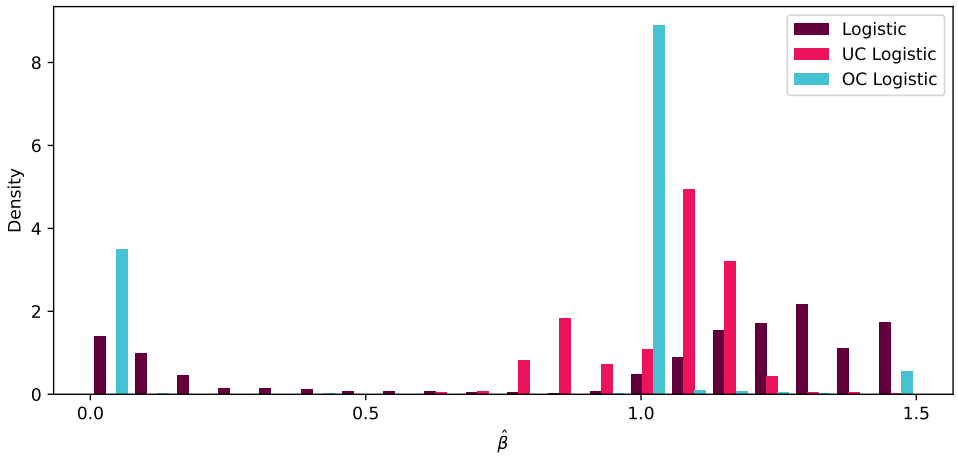


Figure 3.8: Normalized histograms of β estimated by IRBL with differently calibrated classifiers on *ad* dataset with NCAR label noise and $q = 0.5$

Figure 3.8 shows the density distribution of β estimated by IRBL with differently

calibrated classifiers on *ad* dataset with NCAR label noise and $q = 0.5$. We can observe what was described in Subsection 3.5.1 where the under-confident classifier fails to assign low weights to corrupted samples. Even though Figure 3.8 is clipped, the over-confident classifier suffers from a heavy-tailed distribution with estimated weights going as high as 10^8.

Remark. Estimating a ratio of densities is a well-known and studied problem [167] that notably solves the issue of dividing by a probability distribution with values close to 0. The impact of these techniques is studied in Chapter 4.

We conducted experiments as described in Section 3.4 with three variations of IRBL:

1. IRBL with LR (with and without calibration)
2. IRBL with UCLR (with and without calibration)
3. IRBL with OCLR (with and without calibration)

All these variations only affect the classifiers used to estimate f_T and f_U, the final classifier is not modified.

We added the Trusted Only baseline with LR as the difference in calibration of LR, UCLR and OCLR makes no impact on the prediction itself.

The results are synthesized in the following Nemenyi plot:

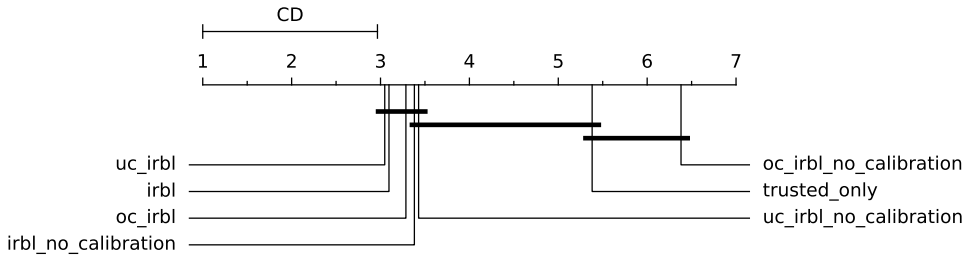


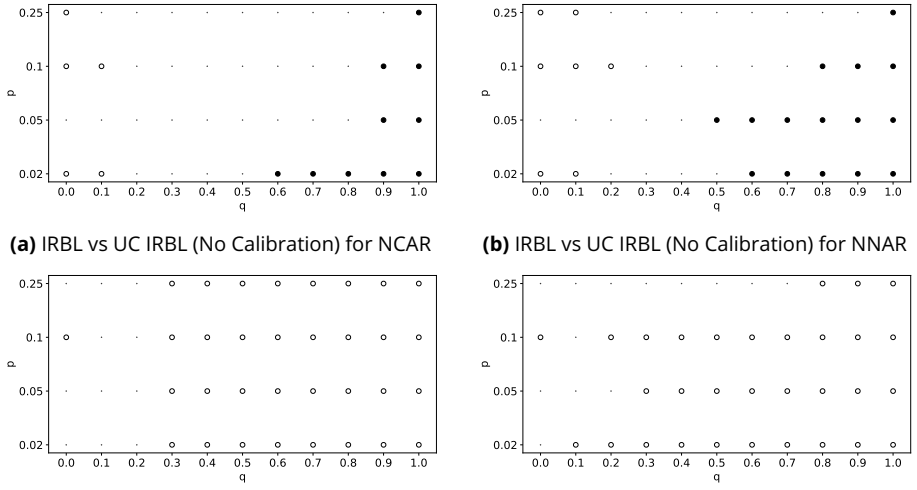
Figure 3.9: Nemenyi test with variations of poorly calibrated Logistic Regressions for the 20 datasets, $\forall p, q$, and for NCAR and NNAR combined.

The first result that stands out in Figure 3.9 is the difference between the performance of IRBL with LR and with OCLR, and UCLR being as good as LR. The over-confident classifier rank last, far from the baselines; meanwhile, the unmodified Logistic Regression ranks first. The quality of the calibration of the classifiers significantly impacts the performance of IRBL. Especially if the classifier is over-confident, the estimated weights' spread significantly deteriorates the final classifier's learning procedure. Nonetheless, the under-confidence of the classifier is less impactful on the overall performance of IRBL.

Secondly, we can notice that when calibrated with Isotonic Regression, UCLR and OCLR are close to the performance of LR for IRBL, ranking very close to

each other. Thus IRBL can be used with poorly calibrated classifiers after proper calibration.

In order to better understand where wrongly calibrated classifiers fail, we generated the following Wilcoxon plots:



(a) IRBL vs UC IRBL (No Calibration) for NCAR

(b) IRBL vs UC IRBL (No Calibration) for NNAR

(c) IRBL vs OC IRBL (No Calibration) for NCAR

(d) IRBL vs OC IRBL (No Calibration) for NNAR

Figure 3.10: Results of the Wilcoxon signed rank test computed on the 20 datasets. Each Figure compares IRBL versus one of the poorly calibrated IRBLs. In each figure "○", "·" and "●" indicate respectively a win, a tie or a loss of the first competitor compared to the second competitor. The vertical axis is p and the horizontal axis is q.

Thanks to the Wilcoxon plots from Figure 3.10, we can better understand how UC IRBL (No Correction) performs against IRBL. The under-confidence of the classifier leads to an IRBL performing better on high-quality datasets and worse on low-quality datasets. UC IRBL weights cannot correctly distinguish corrupted and clean samples from the untrusted dataset.

Several key messages can be resumed thanks to these experimental results. First, when using non-calibrated classifiers, regular calibration algorithms lead to on-par performances with well-calibrated classifiers with IRBL. Then, over-confident classifiers generate weights with high numerical values, destabilizing the final classifier's training procedure. This hypothesis has been confirmed in additional experiments where clipping such weights would stabilize the training procedure. Finally, using under-confident weights leads to a final classifier being close to the *No Correction* baseline, getting good results at high quality but crushingly bad at low quality.

3.6 . On the Specification of Classifiers

The reader may have noticed that the benchmark results have been aggregated by averaging results over multiple datasets to draw significant conclusions between multiple Biquality Learning algorithms, especially between Biquality Learning algorithms and some baselines.

Indeed Biquality Learning algorithm should have some good and even necessary properties such as :

- Being better than the Trusted Only classifier, meaning the ability to leverage knowledge from the untrusted dataset
- Being better than the No Correction classifier, meaning the ability to efficiently correct low-quality untrusted datasets and not get degraded by high-quality untrusted datasets (negative transfer).
- Being better than Semi-Supervised classifiers, meaning the ability to leverage untrusted labels.

If IRBL has verified these properties in our benchmarks on average on all tested datasets, it does not mean that it would work for all datasets.

Indeed these properties are all originating from the same underlying assumptions about the dataset and the classifier: adding more samples to the training dataset will benefit the classifier in better estimating the decision boundary of the classification task. However, this assumption might not hold in practice for some datasets.

For example, if the classification task is too simple to the point where a classifier can learn the decision boundary with very few samples perfectly, or if the classification task is too hard, and the classifier does not have the expressiveness to learn the decision boundary, adding more samples (corrected untrusted samples) will not benefit the classifier.

Thus, we will introduce two critical notions for defining such capacities. The first notion is the suitability of the model family with the classification task (bound to the dataset). Such notion can be empirically measure by the learning curve of a classifier on a given dataset reflecting the capacity to improve its performance with a growing training set. The second notion is the expressiveness of the classifier, meaning its ability to learn complex decision boundaries and patterns. Theoretically, multiple complexity measures exists to quantify the expressiveness such as the VC dimension [180]. Empirically the number of learnable parameters is used as a proxy, such as the number of connections in a Neural Network or the number of iterations of a Gradient Boosting Machine.

Then we will experimentally observe how Biquality Learning algorithms, particularly IRBL, perform when the learning curve and the number of parameters change through different datasets.

3.6.1 . Suitability and Expressiveness of Classifiers

The learning curve is a graphical representation of how well a classifier performs for a given classification task when increasing the number of samples available during training. It is one measure of the suitability of the classifier to the classification task. A flat or constant learning curve (no matter the level) is a synonym for independence between the classifier performance and the size of the training dataset. In practice, this is a synonym for the classification task being too easy, too hard, or ill-defined.

Figure 3.11 represents learning curves of a Gradient Boosting Machine (GBM) [58] learned with an increasing number of Decision Stumps [146] on the same dataset of $i.i.d.$ samples from a 10-dimensional Gaussian distribution, with different labeling processes:

1. The original labeling process described in [74], where:

$$y = \begin{cases} 1 & \text{if}; x \cdot x \geq 9.34 \\ 0 & \text{else} \end{cases}$$

2. A labeling process where labels are assigned at random among the set of classes,

3. An over-complex labeling process, where the feature space is subdivided in multiple n-dimensional cubes, and labels are assigned at random in each cubes with different class ratios.

The learning curves from Figure 3.11 reveal different behaviours for classifiers belonging from the same classifier family on increasingly hard labelling processes.

The first row correspond to tasks where the classifier was able to learn the decision boundary. Indeed the two learning curves are curved enough to see that adding more samples to train on is beneficial to the classifier. However, they were not beneficial on the same level in the three cases, as the difference in accuracy between a low number and a high number of Decision Stumps is significant.

The second and third rows correspond to tasks where the classifier is unable to learn the decision boundary, but it is for two distinct reasons. The first case corresponds to a classification task too hard to be learned (random labels) even by an expressive classifier (GBM with hundreds of decision stumps). However, the second case is a learnable classification task, but the used classifier family needed to be more expressive (single decision stump) to learn the classification task.

Hence more than the curvature of the learning curve is needed to assess if Biquality Learning algorithms will improve the performance of a given classifier against the previously cited baselines. We need a measurable quantity that defines the expressiveness of the classifier.

Multiple measures exist to quantify the expressiveness of a classifier such as the Vapnik-Chevonenkis (VC) dimension. However, such measures are often not

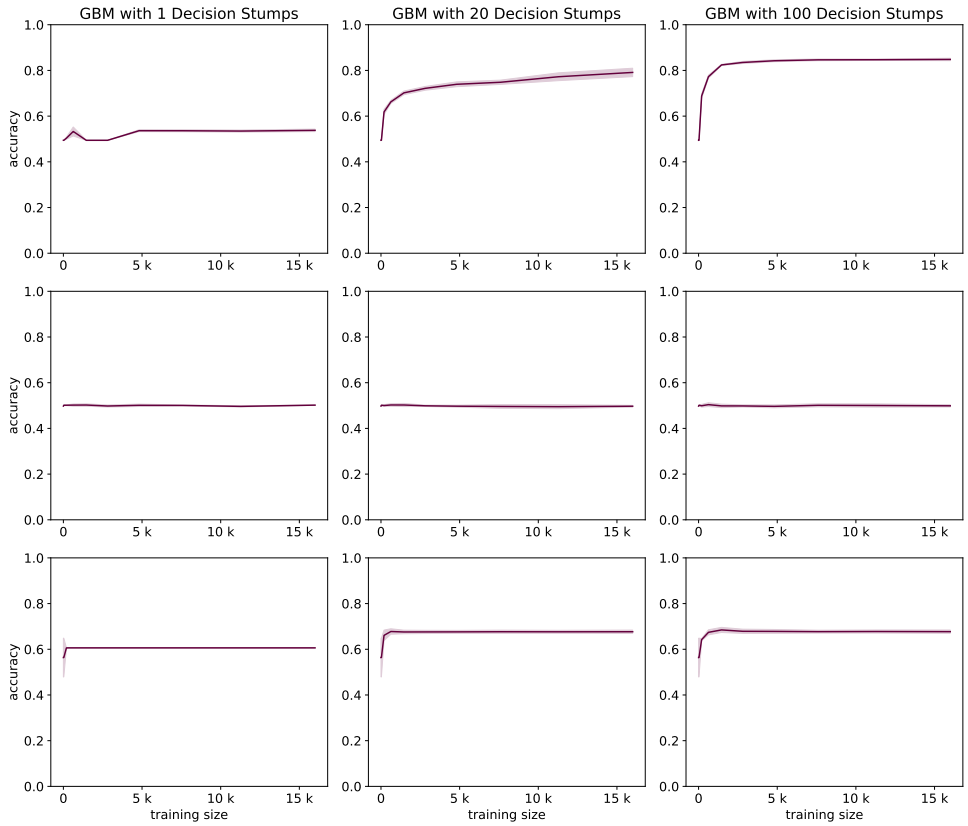


Figure 3.11: Learning curves of a Gradient Boosting Machine with Decision Stumps on an artificial dataset for binary classification [74].

computable in practice and the number of learnable parameters can be a good proxy for these measures such as the number of iterations of Gradient Boosted Trees.

Reusing the same toy examples, we train a GBM Classifier with Decision Stumps with a varying training size and number of iterations.

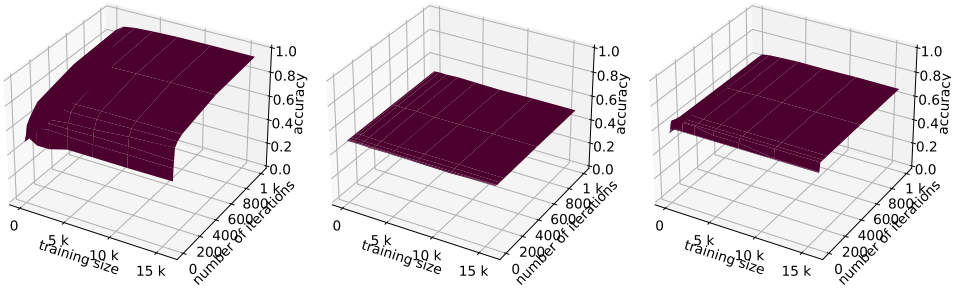


Figure 3.12: Learning curves of GBM Classifiers with Decision Stumps on an artificial dataset for binary classification [74].

Figure 3.12 shows two surface plots that behave fundamentally differently. The first one is curved with higher values of accuracy when the VC dimension and the training size increase. It represents the behavior of a well-suited classifier family which benefits from more training samples. However, the second plot is flat, meaning that the classification task is too hard for the classifier family. Combining the Learning curve and the VC curve is mandatory to fully interpret how the classifier family behaves on a given classification task.

3.6.2 . Wrongly Specified Classifiers

Specification, or the process of selecting the appropriate model and parameters for a machine learning problem, is a crucial step in the machine learning process. Incorrect specification can result in the model being under-specified or over-specified, leading to sub-optimal performance.

Under-specification, also known as bias, occurs when the model is not expressive enough and unable to accurately capture the underlying relationships in the data. This can lead to poor performance on both the training and testing data. To avoid under-specification, it is important to select a model with enough complexity to capture the patterns in the data, such as adding more layers to a neural network or increasing the number of decision trees in a random forest model.

On the other hand, over-specification, also known as variance, occurs when the model is too expressive and fits the noise in the data rather than the underlying patterns. This can lead to good performance on the training data but poor performance on the testing data. To avoid over-specification, it is important to use regularization techniques, such as adding a penalty term to the loss function or using dropout in a neural network, to prevent the model from becoming over-fitted.

3.6.3 . Effects on IRBL

Picking a wrongly specified classifier for IRBL can be detrimental to the overall training process, with different impacts when the model is under-specified or over-specified: on the weight estimation, and on the final classifier.

The effect of using a sub-specified classifier family for IRBL could be two-fold. The first big impact would be on the weight estimation. If the classifier is not capable of learning the trusted and the untrusted decision boundary, the weights of untrusted samples would be imprecise and the untrusted dataset will not be corrected well enough to learn the true concept. Moreover an under-specified classifier will not be able to leverage the additional information contained in the reweighted untrusted samples.

To evaluate such impact, we are going to conduct the same experiments from Section 3.4 with four alternatives tested, using all combinations of an under-specified and over-specified classifier. The under-specified classifier will be the Logistic Regression (linear model) described in Section 3.4, whereas the over-specified classifier will be a GBM classifier with a thousand Decision Stumps from Scikit-Learn [140]. It is worth mentioning calibrating the classifiers did not modify

the results significantly and has been left out of the reported results.

3.6.4 . Results

The results of this new set of experiments is resumed in the following Figure thanks to a critical diagram computed with the four alternatives described in the previous Subsection.

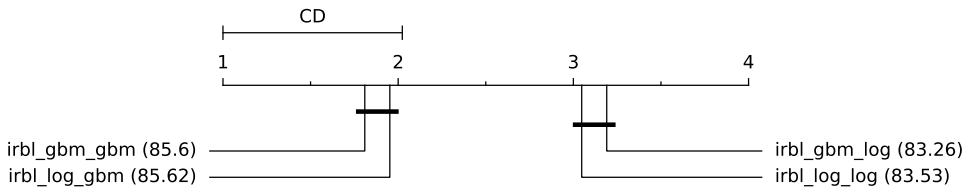


Figure 3.13: Nemenyi test for the 20 binary classification datasets $\forall p, q$ for NCAR and NNAR combined. The average accuracy over all datasets, p, and q is reported between parenthesis next to each methods. The methods' name is composed of the name of the classifier used for weight estimation and then the name of the final classifier.

Figure 3.13 highlights two groups of methods where a first group of method used the GBM classifier for the final classifier and a second group used the Logistic Regression. This Figure reveals that the expressiveness of the classifier for the weight estimation does not significantly change the performance of the overall algorithm. It may showcase the robustness of IRBL to an imprecise estimation of the weight, or a incapacity of GBM to leverage more precise weights to improve its accuracy. However GBM is way more capable than the Logistic Regression to leverage all the re-weighted untrusted samples to improve its performance.

Then we decided to plot the average accuracy of IRBL when the number of GBM iterations increases. The IRBL weight estimations has been done with the maximum number of iterations (even if it does not matter for the final accuracy).

Figure 3.14 shows that IRBL highly benefits of an increase of expressiveness for the final classifier. Even with a thousand iterations, we were not able to consistently make IRBL over-fit on training data as we did not observe a decrease in test accuracy.

Finally, a more expressive classifier can leverage the additional information contained in the newly re-weighted untrusted samples to improve the performance of the final classifier.

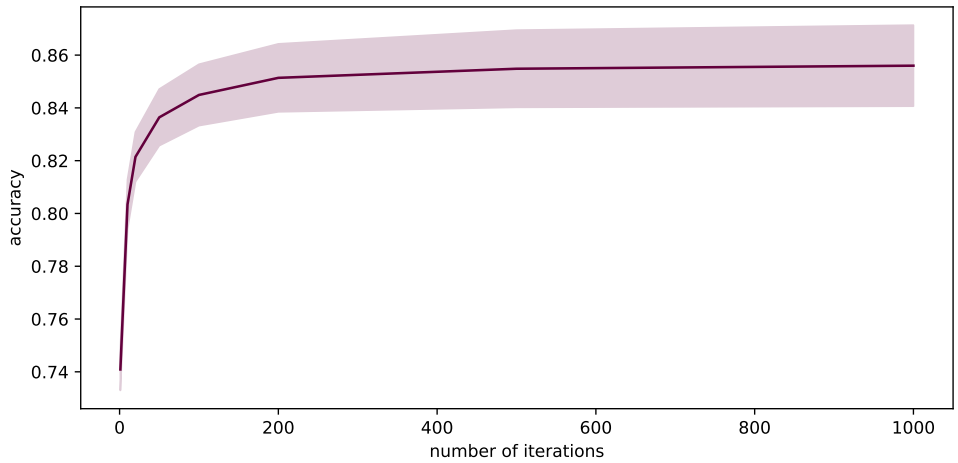


Figure 3.14: Accuracy of IRBL in function of the expressiveness of the base classifier (number of iterations of a GBM) for the 20 binary classification datasets $\forall p, q$ for NCAR and NNAR combined.

3.7 . IRBL and Multiclass Classification

In this section, we propose to extend the experiments conducted in Section 3.4 to multi-class classification datasets.

Actually, IRBL is already capable of learning multi-class classification tasks on Biquality Data, as the Trusted Only and No Correction baseline, and the Corrected Plugin classifier. However, the Unhinged classifier is only capable of learning binary classification tasks. As such, in the following section, we will refer to the Unhinged classifier, the One versus Rest classifier [11] using Unhinged classifiers for each binary classification task.

We decided to use the multi-class classification datasets from the same sources as the binary ones. These datasets are listed in Table 3.3.

Table 3.3: Multi-class classification datasets used for the evaluation. Columns: number of examples ($|\boldsymbol{D}|$), number of features ($|\boldsymbol{\mathcal{X}}|$), number of classes ($|\boldsymbol{\mathcal{Y}}|$), and ratio of examples from the minority class (**min**).

Datasets	$\lvert\boldsymbol{D}\rvert$	$\lvert\boldsymbol{\mathcal{X}}\rvert$	$\lvert\boldsymbol{\mathcal{Y}}\rvert$	**min**	**Datasets**	$\lvert\boldsymbol{D}\rvert$	$\lvert\boldsymbol{\mathcal{X}}\rvert$	$\lvert\boldsymbol{\mathcal{Y}}\rvert$	**min**
ldpa	165K	7	11	0.01	first-ord-theorem	6K1	51	6	0.08
letter	20K	16	26	0.04	artificial-chars	10K	7	10	0.06
pendigits	11K	16	10	0.10	spoken-arabic-digit	263K	14	10	0.10
har	10K	561	6	0.14	isolet	7K8	617	26	0.04
japanese-vowels	10K	14	9	0.08	covertype	581K	54	7	0.00
gas-drift	14K	128	6	0.12	connect-4	68K	42	3	0.10
walking-activity	150K	4	22	0.01	dna	3K2	180	3	0.24
satimage	6K4	36	6	0.10	splice	3K2	60	3	0.24
shuttle	58K	9	7	0.00	mnist	70K	784	10	0.09
usps	9K2	256	10	0.08					

We then ran the same experiments described in Section 3.4 to these multi-class classification datasets and constructed the exact respective figures.

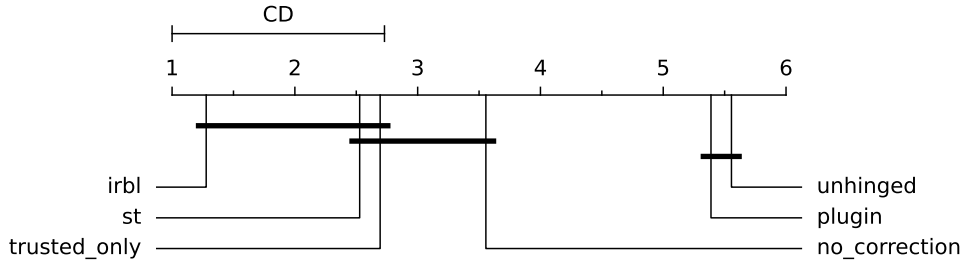


Figure 3.15: Nemenyi test for the 20 multi-class classification datasets $\forall p, q$ for NCAR.

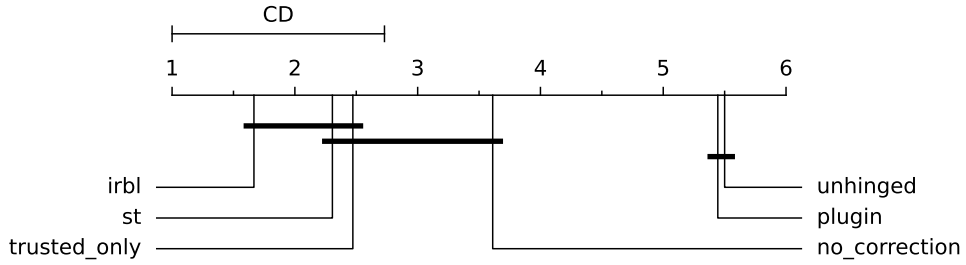


Figure 3.16: Nemenyi test for the 20 multi-class classification datasets $\forall p, q$ for NNAR.

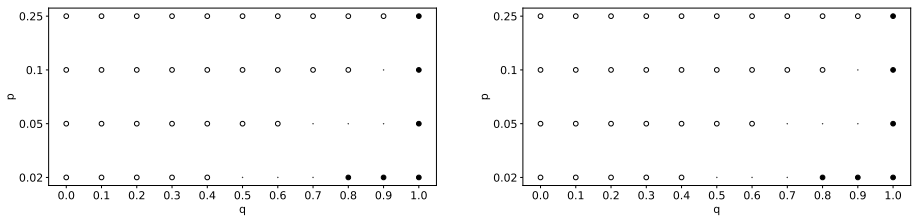
From Figure 3.15 and Figure 3.16, IRBL is still the best performing approach on multi-class classification tasks.

	p	IRBL	No Correction	Plugin	Self Training	Trusted Only	Unhinged
(1)	0.02	**67.31 ± 3.09**	61.58 ± 14.49	52.46 ± 17.25	65.44 ± 0.0	66.28 ± 0.0	44.78 ± 6.15
	0.05	**70.45 ± 1.80**	63.30 ± 10.89	53.41 ± 17.15	69.91 ± 0.0	69.99 ± 0.0	46.18 ± 3.45
	0.10	71.60 ± 1.29	65.03 ± 7.84	53.73 ± 17.01	**71.82 ± 0.0**	71.20 ± 0.0	46.82 ± 2.20
	0.25	72.99 ± 0.71	67.72 ± 4.32	52.25 ± 18.20	**73.00 ± 0.0**	72.65 ± 0.0	47.52 ± 1.40
(2)	0.02	**67.76 ± 2.99**	61.98 ± 14.57	53.10 ± 16.89	65.44 ± 0.0	66.28 ± 0.0	45.76 ± 6.21
	0.05	**70.81 ± 1.86**	63.66 ± 10.90	53.28 ± 16.99	69.91 ± 0.0	69.99 ± 0.0	47.23 ± 3.60
	0.10	**71.91 ± 1.31**	65.18 ± 7.80	53.63 ± 17.01	71.82 ± 0.0	71.20 ± 0.0	48.04 ± 2.09
	0.25	**73.17 ± 0.71**	67.84 ± 4.33	52.96 ± 17.27	73.00 ± 0.0	72.65 ± 0.0	48.59 ± 0.99
Mean		**70.87 ± 1.74**	57.90 ± 13.65	54.42 ± 14.50	70.04 ± 0.0	70.03 ± 0.0	46.26 ± 3.34

Table 3.4: Mean Accuracy (rescaled score to be from 0 to 100) and standard deviation computed on the 20 multi-class classification datasets $\forall q$ for (1) NCAR and (2) NNAR.

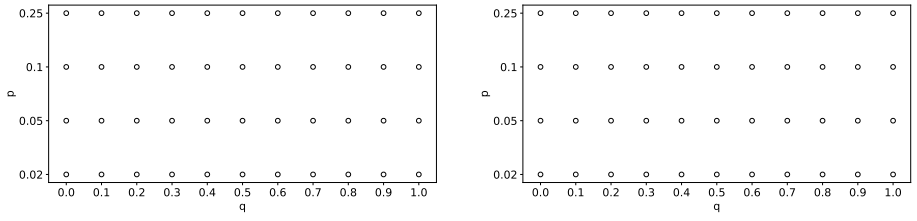
Table 3.4 confirms the results from the two previous critical diagrams.

The Wilcoxon tests from Figure 3.17 exhibit the same behavior as in the binary case.



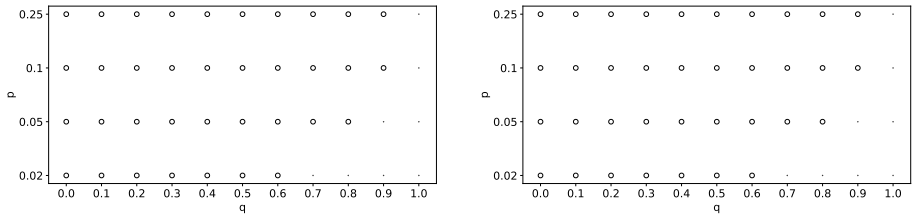
(a) IRBL vs NoCorrection for NCAR

(b) IRBL vs NoCorrection for NNAR



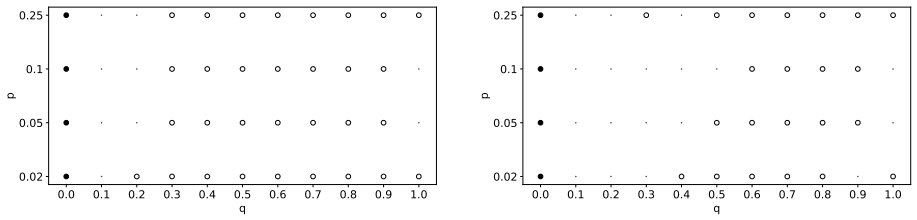
(c) IRBL vs Unhinged for NCAR

(d) IRBL vs Unhinged for NNAR



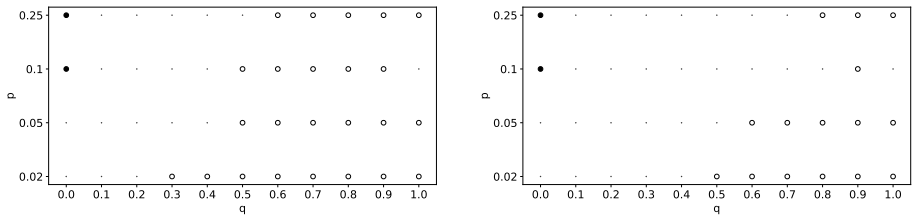
(e) IRBL vs Plugin for NCAR

(f) IRBL vs Plugin for NNAR



(g) IRBL vs Trusted Only for NCAR

(h) IRBL vs Trusted Only for NNAR



(i) IRBL vs Self Training for NCAR

(j) IRBL vs Self Training for NNAR

Figure 3.17: Results of the Wilcoxon signed rank test computed on the 20 datasets. Each Figure compares IRBL versus one of the competitors. In each figure "○", "·" and "●" indicate respectively a win, a tie or a loss of the first competitor compared to the second competitor. The vertical axis is p and the horizontal axis is q.

3.8 . Conclusion

In this chapter, we implemented a new Importance Reweighting approach for Biquality Learning (IRBL). Extensive experiments have shown that IRBL significantly outperforms state-of-the-art approaches by simulating completely-at-random and not-at-random label noise over a wide range of quality and quantity values of untrusted data and base learners. Moreover, we studied the impact on the calibration of the classifiers and showed that using well-calibrated classifiers was key to the efficiency of IRBL. Furthermore, we showed that the more the final classifier is expressive and capable of learning complex decision boundaries, the more it can benefit from the biquality learning framework and IRBL in particular. Finally, we showed that IRBL generalizes naturally to multi-class classification tasks.

In Chapter 4, we will extend the Biquality Learning framework to more complexes corruptions of the untrusted dataset introduction Distributions Shifts. Instead of having in difference in concept only $\mathbb{P}(Y|X)$, the joint distribution $\mathbb{P}(X, Y)$ will differ between the two datasets.

4 - Reboot Biquality Learning with Distribution Shifts

Contents

This chapter has been submitted to an international machine learning journal and is under review.

4.1 . Introduction

Supervised machine learning has been studied and explored extensively during the last decades, and both theoretical and experimental solutions exist to accomplish this task [75]. Weakly supervised machine learning (WSL) has not reached this state yet. WSL is the machine learning field where algorithms learn a model from data with weak supervision instead of strong supervision. Multiple weak supervisions have been identified in [224] such as *inaccurate supervision* when samples are mislabeled, *inexact supervision* when labels are not adapted to the classification task, or *incomplete supervision* when labels are missing which reflects the inadequacy of the available labels in the real world. For every kind of weak supervision, assumptions are needed to design sound algorithms, especially on the corruption model, the generative process behind the weakness of supervision.

Weaknesses in supervision are one facet of weakly supervised learning, and dataset shifts are another. Dataset shifts happen when the data distribution observed at training time is different from what is expected from the data distribution at testing time [122]. This distribution change can take multiple forms, such as a change in the distribution of a single feature, a combination of features, or the concept to be learned. Thus, the common assumption that the training and testing data follow the same distributions is often violated in real-world applications. Again, designing algorithms to handle dataset shifts usually requires assumptions on the nature of the shift [38].

We believe that the biquality data setup proposed in [134] is a suitable framework to design algorithms capable of handling both dataset shifts and weaknesses of supervision simultaneously.

Biquality Learning assumes that two datasets are available at training time: a *trusted dataset* D_T and an *untrusted dataset* D_U both composed of labeled samples $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$. These datasets share the same features $\mathcal{X}_T = \mathcal{X}_U$ and the same set of labels $\mathcal{Y}_T = \mathcal{Y}_U$, having a closed-set of features $\mathcal{X}$ and labels $\mathcal{Y}$. However, the two datasets differ in terms of the joint distribution $\mathbb{P}_T(X, Y) \neq \mathbb{P}_U(X, Y)$ where the trusted distribution is the distribution of interest.

In practice, it is often the case that the trusted dataset is not large enough to learn an efficient model to estimate $\mathbb{P}_T(X, Y)$. By contrast, it is generally easy to get a large enough untrusted dataset that enables to correctly estimate $\mathbb{P}_U(X, Y)$. However, this data distribution could be completely different from the one of interest, $\mathbb{P}_T(X, Y)$. In the biquality data setup, there is no assumption on the difference in joint distribution between the two datasets, and it can cover a wide range of known problems. From the Bayes Formula:

$$\mathbb{P}(X, Y) = \mathbb{P}(X \mid Y)\mathbb{P}(Y) = \mathbb{P}(Y \mid X)\mathbb{P}(X) \tag{4.1}$$

distribution shift covers covariate shift $\mathbb{P}_T(X) \neq \mathbb{P}_U(X)$, concept drift $\mathbb{P}_T(Y \mid X) \neq \mathbb{P}_U(Y \mid X)$, class-conditional shift $\mathbb{P}_T(X \mid Y) \neq \mathbb{P}_U(X \mid Y)$ and prior shift

$\mathbb{P}_T(Y) \neq \mathbb{P}_U(Y)$.

The Biquality Data setup typically occurs in three scenarios in practice.

1. The first scenario corresponds to the case where annotating samples is expansive to the point of being prohibitive to label an entire dataset, but labeling a small part of the dataset is doable. It's typically the case in Fraud Detection and in Cyber Security where labeling samples require complex forensics from domain experts. There, the rest of the dataset is usually labeled by hand-engineered rules which might not perfectly fit the classification task and the labels can't properly be trusted [147].

2. The second scenario happens when there are data shifts during the labeling process over the course of time. For example in MLOps [97], when a model is first learned on clean data and then deployed in production, predictions can be used to learn an updated model [182]; these predictions may be faulty and need to be dealt with. A second example in MLOps occurs when newly clean data is acquired to retrain a deployed model; the most recent clean data is considered trusted and the old clean data is considered untrusted when dataset shifts occurred [62].

3. The third scenario occurs when multiple annotators are responsible for dataset labeling. In natural language processing (NLP) products, for example, multiple annotators follow labeling guidelines to annotate verbatims. The efficiency of annotators to follow these guidelines may vary and might not be trusted. However, if one annotator can be trusted, all the other annotators can be set as untrusted and the biquality data setup may apply. Especially, considering each untrusted annotator against the trusted annotator can be viewed as a biquality learning task [207].

Having the trusted and untrusted datasets available at training time makes it possible to design algorithms dealing with closed-set distribution shifts. If algorithms designed for biquality data with concept drift only have been explored recently [133], algorithms that deal with distribution shift are still behind. In this paper, we propose two biquality approaches that adapt methods from either the covariate shift literature or the concept drift literature in order to deal with both corruptions simultaneously.

Multiple biquality learning algorithm designs have been identified in [134]. They have been divided into three main families based on how they modify instances to correct the global learning procedure. Untrusted instances can be (i) relabeled correctly, (ii) modified in the feature space, or (iii) reweighted such that the untrusted dataset seems sampled from the trusted distribution $\mathbb{P}_T(X, Y)$. We propose here two corresponding algorithms for the third case of importance reweighting.

In Section 4.2, a brief state-of-the-art relates what has been already achieved on biquality learning. Then, Section 4.3 further focuses on the state-of-the-art of

importance reweighting on biquality learning. Section 4.4 and Section 4.5 introduce our proposals to use classifiers to reweight untrusted instances for biquality learning with distribution shifts. Then, Section 4.6 describes the experiments that evaluate the efficiency of our proposed approaches on real datasets and corruptions. Finally, Section 4.7 presents the results of the proposed experiments before concluding.

4.2 . Related work

Machine learning algorithms on biquality data have been developed in many different sub-domains of weakly supervised learning. Some of these sub-domains are robust learning to label noise, learning under covariate shift, or transfer learning. Because these subdomains expect different corruptions, not all algorithms designed for some subcases of biquality learning will work in a more general setting with distribution shifts.

For example, Gold Loss Correction (GLC) [79] and Importance Reweighting for Biquality Learning (IRBL) [134] are algorithms designed to specifically deal with a concept drift between the trusted and the untrusted datasets. GLC, on the one hand, corrects the learning procedure on the untrusted dataset using a noise transition matrix between the trusted and untrusted concept. IRBL, on the other hand, reweight untrusted instances using an estimation of the ratio of both concepts. These algorithms are not theoretically designed to handle covariate shift; nevertheless, they could be empirically efficient on this task and serve as a reference.

Another group of algorithms, such as Kernel Mean Matching (KMM) [65], and Probabilistic Density Ratio Estimation (PDR) [10], only deals with covariate shift between the two datasets. These algorithms seek to reweight untrusted instances such that the distribution of features between the two datasets is equivalent. KMM minimizes the difference of the features mean in a reproducing kernel Hilbert space. PDR learns the classification task of predicting if an instance is untrusted or not and uses the predicted probability of being an untrusted instance as the weight. These algorithms are not designed to handle concept drift and will serve as references too.

However, a recent proposal aims to adapt these algorithms to the biquality framework with distribution shift [49]. They proposed to use one density ratio estimation algorithm per class, which, when combined, corrects the distribution shift. They also proposed to transform the joint density ratio estimation problem by combining the features and labels of the data into a new feature space that allows for a single density ratio estimation algorithm. These adapted algorithms will also serve as competitors.

Finally, recent approaches such as Learning to Reweight (L2RW) [149], or Meta-Weight-Net (MWNet) [161] based on deep learning and meta-learning have not been tested in this paper. Indeed, they require a class of algorithms with a

differentiable and incremental learning procedure that does not fit most popular families of classifiers, such as gradient boosting trees. They are left to be tested in future works.

Biquality Data is not a new setup per see, as previous work exists on this setup going back to [89] to the best of our knowledge. Each previous work was carried out in different sub-domains of weakly supervised learning and thus achieved different goals based on different setups [79, 161, 149, 218, 89]. These setups used different terms, definitions, hypotheses, and requirements but still sought to solve the same fundamental problem of biquality learning. Only recently, some efforts have been done to provide clear and concise definitions of the biquality learning framework [134]. We propose in this paper to extend it to include dataset shifts. This extension is, to the best of our knowledge, a new problem to tackle that few tried already [49], limiting existing prior literature.

4.3 . Reweighting for distribution shift

The previous Section introduced the most common algorithms used in machine learning for biquality data. Most of them were based on instance reweighting, and specifically, on estimating the Radon-Nikodym Derivative (RND) [131] of $\mathbb{P}_T(X,Y)$ with respect to $\mathbb{P}_U(X,Y)$ which is $\frac{\mathrm{d}\mathbb{P}_T(X,Y)}{\mathrm{d}\mathbb{P}_U(X,Y)}$. This comes from the fact that minimizing the reweighted empirical risk by the RND on the untrusted data is equivalent to minimizing the empirical risk on trusted data:

$$\begin{aligned} R_{(X,Y)\sim T,L}(f) &= \mathbb{E}_{(X,Y)\sim T}[L(f(X),Y)] \\ &= \int L(f(X),Y)\,\mathrm{d}\mathbb{P}_T(X,Y) \\ &= \int \frac{\mathrm{d}\mathbb{P}_T(X,Y)}{\mathrm{d}\mathbb{P}_U(X,Y)} L(f(X),Y)\,\mathrm{d}\mathbb{P}_U(X,Y) \\ &= \mathbb{E}_{(X,Y)\sim U}[\frac{\mathbb{P}_T(X,Y)}{\mathbb{P}_U(X,Y)} L(f(X),Y)] \\ &= \mathbb{E}_{(X,Y)\sim U}[\beta L(f(X),Y)] \\ &= R_{(X,Y)\sim U,\beta L}(f) \end{aligned} \tag{4.2}$$

However, estimating the RND can be a difficult task, especially in the case of distribution shift where the joint distribution ratio β needs to be estimated. Proposals have been made to ease this estimation.

A *first proposal* has been made in IRBL [134] which focused first on the concept drift between datasets using the Bayes Formula:

$$\beta(X,Y) = \frac{\mathbb{P}_T(X,Y)}{\mathbb{P}_U(X,Y)} = \frac{\mathbb{P}_T(Y \mid X)\mathbb{P}_T(X)}{\mathbb{P}_U(Y \mid X)\mathbb{P}_U(X)} \tag{4.3}$$

Their proposed algorithm is based on the decomposition of the joint density ratio estimation task into three sub-tasks. The first one is to estimate the trusted

concept $\mathbb{P}_T(Y \mid X)$, and is done by learning a classifier on the trusted dataset. The second task is to estimate the untrusted concept $\mathbb{P}_U(Y \mid X)$, which is done by learning a classifier on the untrusted dataset. And the third task about density ratio estimation $\frac{\mathbb{P}_T(X)}{\mathbb{P}_U(X)}$ was skipped as no covariate shift was introduced in their benchmark, but it is a well known and solved machine learning task [169].

A *second proposal* has been made in [49] which focused on the covariate shift between datasets using the Bayes Formula differently:

$$\beta(X,Y) = \frac{\mathbb{P}_T(X,Y)}{\mathbb{P}_U(X,Y)} = \frac{\mathbb{P}_T(X \mid Y)\mathbb{P}_T(Y)}{\mathbb{P}_U(X \mid Y)\mathbb{P}_U(Y)} \tag{4.4}$$

In their proposed algorithm, the joint density ratio estimation task has been decomposed into K-tasks where K is the number of classes to predict. For each class, only examples of the given class are selected on both datasets, such that the samples are drawn from the $\mathbb{P}(X \mid Y)$ distribution. Then, a density ratio estimation procedure usually employed to estimate $\frac{\mathbb{P}_T(X)}{\mathbb{P}_U(X)}$ is learned on these sub-datasets to estimate $\frac{\mathbb{P}_T(X|Y)}{\mathbb{P}_U(X|Y)}$, effectively handling distribution shift from Equation 4.4. As it uses K density ratio algorithms, this generic approach will be named K-DensityRatio (KDR) in the rest of the paper.

Finally, a *last approach* is to focus on the density ratio estimation task by finding a deterministic and invertible transformation f as proposed in [49]:

$$\beta(X,Y) = \frac{\mathbb{P}_T(X,Y)}{\mathbb{P}_U(X,Y)} = \frac{\mathbb{P}_T(Z)}{\mathbb{P}_U(Z)}, \quad Z = f(X,Y) \tag{4.5}$$

An example of such transformation [49] is the classification loss of a model learned on the biquality data. One density ratio estimation procedure is done on these new features $\mathcal{Z}$ to directly estimate $\frac{\mathbb{P}_T(Z)}{\mathbb{P}_U(Z)}$.

IRBL has experimentally proved to efficiently solve the biquality learning task on tabular data [134]. However, the experiments were conducted on corruptions only affecting the untrusted concept $\mathbb{P}(Y \mid X)$ and not the joint distribution $\mathbb{P}(X,Y)$. We propose here to adapt IRBL to handle distribution shifts by solving the third task of density ratio estimation with a probabilistic classifier. Moreover, we propose a new version of KDR using probabilistic classifiers to solve the K density ratio estimation tasks. This proposition is driven by the desire to reuse efficient tricks from IRBL and to rely on a non parametric approach by contrast to the original proposal [49].

4.4 . First proposed approach: IRBLV2

Importance Reweighting for Biquality Learning (IRBL) [134] is a biquality learning algorithm designed to handle closed-set concept-drift. The algorithm is based on using two probabilistic classifiers: first, to estimate both concepts $\mathbb{P}_T(Y \mid X)$ and $\mathbb{P}_U(Y \mid X)$ and, second, using these classifiers' outputs to estimate the RND between both data distributions. In the particular case of label noise, especially instance dependent label noise, it has been shown to be the best approach experimentally on a wide variety of datasets.

We propose to adapt it to handle covariate shift by estimating the ratio $\frac{\mathbb{P}_T(X)}{\mathbb{P}_U(X)}$ by using a third probabilistic classifier as in [10]. This algorithm works by defining a new supervised classification task by learning to predict if a sample is trusted or untrusted by only using its features. If there exists covariate shift between the datasets, the classifier should be able to discriminate between the two datasets.

Let's introduce S as the new target:

$$s_i(x_i) = \begin{cases} 0, & \text{if } x_i \in D_U \\ 1, & \text{if } x_i \in D_T \end{cases} \tag{4.6}$$

Estimating $\mathbb{P}(S \mid X)$ allows us to estimate $\frac{\mathbb{P}_T(X)}{\mathbb{P}_U(X)}$ directly without estimating both distributions:

$$\frac{\mathbb{P}_T(X)}{\mathbb{P}_U(X)} = \frac{\mathbb{P}(X \mid S=1)}{\mathbb{P}(X \mid S=0)} = \frac{\mathbb{P}(S=1 \mid X)\mathbb{P}(X)}{\mathbb{P}(S=1)} \times \frac{\mathbb{P}(S=0)}{\mathbb{P}(S=0 \mid X)\mathbb{P}(X)} \tag{4.7}$$

Combining equation 4.3 and 4.7 :

$$\frac{\mathbb{P}_T(Y \mid X)\mathbb{P}_T(X)}{\mathbb{P}_U(Y \mid X)\mathbb{P}_U(X)} = \frac{\mathbb{P}_T(Y \mid X)}{\mathbb{P}_U(Y \mid X)} \times \frac{\mathbb{P}(S=1 \mid X)}{\mathbb{P}(S=1)} \times \frac{\mathbb{P}(S=0)}{\mathbb{P}(S=0 \mid X)} \tag{4.8}$$

We propose to estimate equation 4.8 by learning probabilistic classifiers $f \in \mathcal{F}$ to estimate each of its terms. A probabilistic classifier f_T is learned on D_T to estimate $\mathbb{P}_T(Y \mid X)$, f_U is learned on D_U to estimate $\mathbb{P}_U(Y \mid X)$, and f_S is learned on $\{(x, s(x)) \mid \forall x \in D_T \cup D_U\}$ to estimate $\mathbb{P}_U(S \mid X)$, leading to the following Algorithm 4.

Algorithm 4: Importance Reweighting for Biquality Learning V2 (IRBLV2)

Input: Trusted Dataset D_T, Untrusted Dataset D_U, Probabilistic Classifier Family $\mathcal{F}$

Learn $f_U \in \mathcal{F}$ on D_U

Learn $f_T \in \mathcal{F}$ on D_T

Learn $f_S \in \mathcal{F}$ on $\{(x, s(x)) \mid \forall x \in D_T \cup D_U\}$

for $(x_i, y_i) \in D_U$ **do**

  $\hat{\beta}(x_i, y_i) = \frac{f_T(x_i)_{y_i}}{f_U(x_i)_{y_i}} \frac{f_S(x_i)_1}{|D_T|} \frac{|D_U|}{f_S(x_i)_0}$

for $(x_i, y_i) \in D_T$ **do**

  $\hat{\beta}(x_i, y_i) = 1$

Learn $f \in \mathcal{F}$ on $D_T \cup D_U$ with weights $\hat{\beta}$

Output: f

4.5 . Second proposed approach: K-PDR

K-DensityRatio (KDR) [49] is an alternative approach to design a biquality learning algorithm able to handle distribution shift. The focus is made on the covariate shift between the two datasets. It handles the covariate shift in a class conditional fashion to deal with distribution shifts by using covariate shift correction once per class.

From Equation 4.4, KDR evaluates the ratio $\frac{\mathbb{P}_T(X|Y)}{\mathbb{P}_U(X|Y)}$ with density ratio estimation algorithms. To do so, it first samples data from the $X \mid Y$ distribution by selecting only samples from a given class $k \in [\![1, K]\!]$ in both datasets D_T and D_U. Then, it uses density ratio estimation algorithms $e \in \mathcal{E}$ on these sub-datasets to estimate $\frac{\mathbb{P}_T(X|Y=k)}{\mathbb{P}_U(X|Y=k)}$ independently k times. The class priors $\mathbb{P}_T(Y)$ and $\mathbb{P}_U(Y)$ are estimated empirically from both training sets. See Algorithm 5.

In [49], Kernel Mean Matching (KMM) [83, 65] has been used as the Density Ratio algorithm e to handle covariate shift. Empirically, KMM is an algorithm that matches with quadratic programming [193] the mean of both datasets in a feature space induced by a kernel k on the domain $\mathcal{X} \times \mathcal{X}$:

$$\begin{aligned} \min_{\beta_i} \quad & \left\| \frac{1}{|D_U|} \sum_{i=0}^{|D_U|} \beta_i \Phi(x_i) - \frac{1}{|D_T|} \sum_{i=0}^{|D_T|} \Phi(x_i) \right\|_{\mathcal{H}} \\ \text{s.t.} \quad & 0 \leq \beta_i \leq B \\ & \left| \frac{1}{|D_U|} \sum_{i=0}^{|D_U|} \beta_i - 1 \right| < \epsilon \end{aligned} \tag{4.9}$$

where $\phi : \mathcal{X} \to \mathcal{H}$ denotes the canonical feature map, $\mathcal{H}$ is the reproducing kernel Hilbert space induced by the kernel k, $|\cdot|_{\mathcal{H}}$ is the norm on $\mathcal{H}$ and B and ϵ are regularization and normalization constraints.

Algorithm 5: K-DensityRatio (KDR)

Input: Trusted Dataset D_T, Untrusted Dataset D_U, Density Ratio Estimator Family $\mathcal{E}$, Probabilistic Classifier Familiy $\mathcal{F}$

for $k \in [\![1, K]\!]$ **do**
  Let $D_T^k = \{\forall (x, y) \in D_T \mid y = k\}$
  Let $D_U^k = \{\forall (x, y) \in D_U \mid y = k\}$
  Learn $e^k \in \mathcal{E}$ on D_T^k and D_U^k
for $(x_i, y_i) \in D_U$ **do**
  $\hat{\beta}(x_i, y_i) = e^{y_i}(x_i) \frac{|D_T^{y_i}|}{|D_T|} \frac{|D_U|}{|D_U^{y_i}|}$
for $(x_i, y_i) \in D_T$ **do**
  $\hat{\beta}(x_i, y_i) = 1$
Learn $f \in \mathcal{F}$ on $D_T \cup D_U$ with weights $\hat{\beta}$

Output: f

As such, KMM is a parametric algorithm based on kernels. We propose to use instead a probabilistic classifier to handle covariate shift, in the same fashion as in Equations 4.6 and 4.7 to make a non-parametric version of KDR as shown in Equation 4.8.b.

$$\begin{aligned}
\frac{\mathbb{P}_T(X \mid Y)\mathbb{P}_T(Y)}{\mathbb{P}_U(X \mid Y)\mathbb{P}_U(Y)} &= \frac{\mathbb{P}(X \mid Y, S=1)\mathbb{P}(Y \mid S=1)}{\mathbb{P}(X \mid Y, S=0)\mathbb{P}(Y \mid S=0)} \\
&= \frac{\mathbb{P}(S=1 \mid X, Y)\mathbb{P}(X, Y)}{\mathbb{P}(Y \mid S=1)\mathbb{P}(S=1)} \times \frac{\mathbb{P}(Y \mid S=0)\mathbb{P}(S=0)}{\mathbb{P}(S=0 \mid X, Y)\mathbb{P}(X, Y)} \times \frac{\mathbb{P}(Y \mid S=1)}{\mathbb{P}(Y \mid S=0)} \\
&= \frac{\mathbb{P}(S=1 \mid X, Y)}{\mathbb{P}(S=0 \mid X, Y)} \times \frac{\mathbb{P}(S=0)}{\mathbb{P}(S=1)}
\end{aligned} \tag{4.8.b}$$

The main advantage of the non-parametric approach is that it does not require assumptions about the data distribution, which may not be satisfied in many real-world datasets and could lead to poor performances. Moreover, the scalability of K-PDR is better than the scalability of K-KMM both in space and time complexity. K-PDR has K times the same complexity as the complexity of learning the chosen probabilistic classifier, which is $\mathcal{O}(K \times |\mathcal{X}| \times |D_u^k| \log(|D_u^k|))$ for Decisions Trees [32]. Meanwhile, K-KMM memory complexity is $\mathcal{O}(|D_u^k|^2 + |D_u^k| \times |D_t^k|)$ to sequentially build matrices necessary for the quadratic program, and a worst-case time complexity of $\mathcal{O}(K \times |D_u^k|^3)$ to solve the quadratic program [205]. Finally, the proposed approach is even more flexible than the previous one, as any family of machine learning classifiers could be used instead of kernels. This leads to Algorithm 6.

Algorithm 6: K-Probabilistic Density Ratio (K-PDR)

Input: Trusted Dataset D_T, Untrusted Dataset D_U, Probabilistic Classifier Familiy $\mathcal{F}$

for $k \in [\![1, K]\!]$ **do**
  Let $D_T^k = \{\forall (x, y) \in D_T \mid y = k\}$
  Let $D_U^k = \{\forall (x, y) \in D_U \mid y = k\}$
  Learn $f_S^k \in \mathcal{F}$ on $\{(x, s(x)) \mid \forall x \in D_T^k \cup D_U^k\}$
for $(x_i, y_i) \in D_U$ **do**
  $\hat{\beta}(x_i, y_i) = \frac{f_S^{y_i}(x_i)_1}{f_S^{y_i}(x_i)_0} \frac{|D_U|}{|D_T|}$
for $(x_i, y_i) \in D_T$ **do**
  $\hat{\beta}(x_i, y_i) = 1$
Learn $f \in \mathcal{F}$ on $D_T \cup D_U$ with weights $\hat{\beta}$

Output: f

4.6 . Experiments

Benchmarking biquality learning algorithms means evaluating their efficiency and resilience on both dataset shifts and weaknesses of supervision in a joint manner. Introducing these corruptions synthetically in usual public multi-class classification datasets allows a fined grained and controlled evaluation of these algorithms.

From Equation 4.1, introducing distribution shift can be done in four ways: by introducing covariate shift, concept drift, class-conditional shift, or prior shift. Especially modifying both concept drift and covariate shift or class-conditional shift and prior shift at the same time leads to particularly complex distribution shifts. Table 4.1 sums up the hierarchy of distribution shift sources.

Table 4.1: Hierarchy of Distribution Shift sources

Distribution Shift $\mathbb{P}(X, Y)$			
$\mathbb{P}(Y \mid X)\mathbb{P}(X)$		$\mathbb{P}(X \mid Y)\mathbb{P}(Y)$	
Concept Drift $\mathbb{P}(Y \mid X)$	Covariate Shift $\mathbb{P}(X)$	Class-Conditional Shift $\mathbb{P}(X \mid Y)$	Prior Shift $\mathbb{P}(Y)$

We chose one method to synthetically generate each distribution shift source in our experiments, all of them described in the following subsections. We reused proven methods from the state-of-the-art [177, 49, 17]. Note that generating class-conditional shift has not been done yet to our knowledge. This paper, therefore, proposes a new experimental protocol to synthetically create a class-conditional shift in real-world datasets (see Subsection 4.6.3).

4.6.1 . Concept Drift

We introduce concept drift by corrupting the untrusted dataset with label noise to modify $\mathbb{P}(Y \mid X)$. We pick a symmetric noise [177] where noisy samples are given a random class with equal probability r from the whole set of labels $\mathcal{Y}$.

4.6.2 . Covariate Shift

We introduce covariate shift by adding class imbalance. Modifying class prior $\mathbb{P}(Y)$ to introduce class imbalance is going to affect $\mathbb{P}(X)$ as different features sub-spaces are associated with different classes if a classification model can be learned. Indeed by rewriting Equation 4.1:

$$\mathbb{P}(X) = \frac{\mathbb{P}(X \mid Y)\mathbb{P}(Y)}{\mathbb{P}(Y \mid X)}$$

We follow the same experimental protocol as in [49, 17]. First, we sort all classes in descending order by their number of samples, then we select at least a fraction $\mu = 0.5$ of all classes as the "majority" class group. The rest are considered the "minority" class group. Finally, we sub-sample all the classes in the "majority" group evenly to obtain a ratio of samples between the "majority" group and "minority" group of ρ.

4.6.3 . Class-Conditional Shift

We propose a new experimental protocol to synthetically modify the class-conditional distribution $\mathbb{P}(X \mid Y)$. To do so, we first split the original dataset by selecting samples from the same class k out of the K classes to create D^k. Then we train a K-Means [113] algorithm on these sub-datasets in order to learn a division of the feature space $\mathcal{X}$, the number of clusters is selected to maximize the average silhouette score [151]. Finally, we sub-sample some of these clusters to modify the class-conditional distribution $\mathbb{P}(X \mid Y)$ with the same methodology as the class imbalance from Subsection 4.6.2 on the labels cluster.

4.6.4 . Prior Shift

We introduced prior shift $\mathbb{P}_U(Y)$ by introducing class imbalance with the same methodology from Subsection 4.6.2. Thus the prior shift and covariate shift experiments will be equivalent.

4.6.5 . Datasets

We randomly picked supervised classification datasets, see Table 4.2, from different sources: UCI [42], libsvm[1], active learning challenge [70] and openML [179]. A part of these datasets comes from past challenges in active learning, where high performances with a low number of labeled examples have proved challenging to obtain, which makes leveraging the untrusted dataset necessary. For each dataset, 80% of samples were used for training, and 20% were used for the test. With this choice of datasets, an extensive range of class ratios, number of classes, number of features, and dataset sizes are covered.

[1] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Table 4.2: Multi-class classification datasets used for the evaluation. Columns: number of examples ($|\boldsymbol{D}|$), number of features ($|\boldsymbol{\mathcal{X}}|$), number of classes ($|\boldsymbol{\mathcal{Y}}|$), and ratio of examples from the minority class (**min**).

Datasets	$\|\boldsymbol{D}\|$	$\|\boldsymbol{\mathcal{X}}\|$	$\|\boldsymbol{\mathcal{Y}}\|$	min	Datasets	$\|\boldsymbol{D}\|$	$\|\boldsymbol{\mathcal{X}}\|$	$\|\boldsymbol{\mathcal{Y}}\|$	min
ad	3K	1558	2	0.14	poker	1M	10	2	0.50
eeg	15K	14	2	0.45	ldpa	165K	7	11	0.01
ibn_sina	21K	92	2	0.38	letter	20K	16	26	0.04
zebra	61K	154	2	0.05	pendigits	11K	16	10	0.10
musk	6K	167	2	0.15	har	10K	561	6	0.14
phishing	11K	30	2	0.44	japanese-vowels	10K	14	9	0.08
spam	5K	57	2	0.39	gas-drift	14K	128	6	0.12
ijcnn1	191K	22	2	0.10	walking-activity	150K	4	22	0.01
diabetes	768	8	2	0.35	satimage	6K4	36	6	0.10
credit-g	1K	20	2	0.30	shuttle	58K	9	7	0.00
hiva	43K	1617	2	0.04	usps	9K2	256	10	0.08
svmguide3	1K	22	2	0.24	first-ord-theorem	6K1	51	6	0.08
web	37K	123	2	0.24	artificial-chars	10K	7	10	0.06
mushroom	8K	22	2	0.48	spoken-arabic-digit	263K	14	10	0.10
skin-segmentation	245K	3	2	0.21	isolet	7K8	617	26	0.04
mozilla4	16K	5	2	0.33	covertype	581K	54	7	0.00
electricity	45K	8	2	0.42	connect-4	68K	42	3	0.10
bank-marketing	45K	16	2	0.12	dna	3K2	180	3	0.24
magic-telescope	19K	10	2	0.35	splice	3K2	60	3	0.24
phoeneme	5K	5	2	0.29	mnist	70K	784	10	0.09

We first split each of these datasets into trusted and untrusted datasets completely at random with a ratio of trusted data p of 1%, 2%, and 5%. These values aim to replicate actual use cases where getting trusted samples is a costly process, either time-consuming or expensive. Then we synthetically corrupt the untrusted dataset with the methods described earlier in this Section.

4.6.6 . Competitors

We compare IRBL2 and K-PDR against multiple state-of-the-art competitors and baselines :

- K-KMM, the original version of KDR using KMM as proposed in [49];
- IRBL [134], which is IRBL2 without covariate shift correction;
- PDR [10] the covariate-shift only baseline;
- Trusted-Only baseline, when the model is learned using only the trusted dataset;
- No-Correction baseline, when the model is learned on both datasets without correction applied.

To evaluate the competitors' performance, we use the same probabilistic classifier family, histogram-based gradient boosting trees [91] from Scikit-Learn [140]

with their default hyperparameters, every time an algorithm required a base classifier.

For KMM and K-KMM we use the Radial Basis Function (RBF) kernel [183] with the default $\gamma = \frac{1}{|\mathcal{X}|}$ value from Scikit-Learn. In order to scale KMM to the bigger datasets, we used an ensembling version [119] of the original KMM algorithm with a batch size of 200.

Finally, accuracy is the metric used to quantify their efficiency on the classification tasks.

4.7 . Results

The experiments were divided into two parts corresponding to the distributional shift hierarchy described in Table 4.1 in Section 5.3. The first part of the experiments corresponds to the left side of the table, $\mathbb{P}(Y \mid X)\mathbb{P}(X)$, mixing concept drift and covariate lag. The concept drift is simulated thanks to a labeling noise as described in Section 4.6.1 and the covariate shift is introduced thanks to a class imbalance. The second part of the experiments corresponds to the table's right part, $\mathbb{P}(X \mid Y)\mathbb{P}(Y)$, mixing class-conditional and prior shifts. The class-conditional shift is simulated using our method proposed in Section 4.6.3, and the prior shift is simulated thanks to an artificial class imbalance. Each of these parts is thus composed of two axes of analysis. First, we will analyze each of these axes independently and then study their combined and crossed effects.

4.7.1 . First part: Concept Drift and Covariate Shift

To study one axis independently, we set the value of the other axis to a value that does not alter the training data set. Here, if the first axis of analysis is label noise dictated by r, we set the sub-sampling ratio ρ to 1. Then we compute a critical diagram presented in Figure 4.1 which ranks the competitors on this axis.

The Nemenyi test [125] is used to rank the approaches in terms of mean accuracy. The Nemenyi test consists of two successive steps. First, the Friedman test is applied to the mean accuracy of competing approaches to determine whether their overall performance is similar. Second, if not, the post-hoc test is applied to determine groups of approaches whose overall performance is significantly different from that of the other groups.

Figure 4.1 confirms the results from [134] as IRBL is the best competitor on label noise. IRBL2 follows closely, and the covariate shift adaptation seems not to impact the method's efficiency on label noise.

The second axis of analysis is covariate shift, so we set the label noise r to 0 and compute a new critical diagram in Figure 4.2.

Figure 4.2 shows that K-PDR is the best approach to combat covariate shift and, more generally, KDR as K-KMM is an efficient competitor. However, on covariate shift, IRBL2 does not beat the TrustedOnly baseline and struggles on the task.

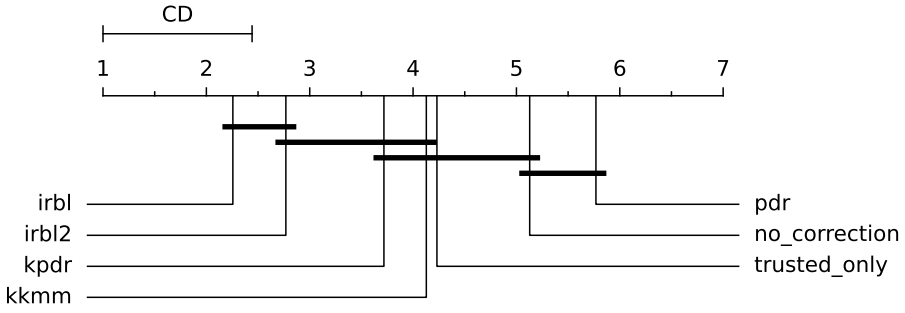


Figure 4.1: Nemenyi test for all datasets $\forall p, r$ with $\rho = 1$.

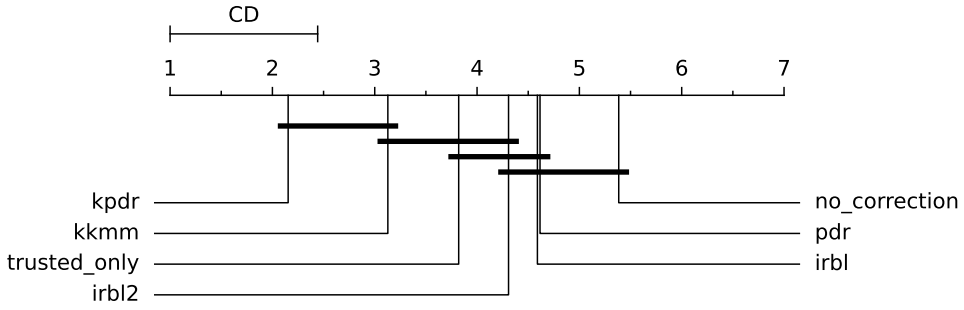
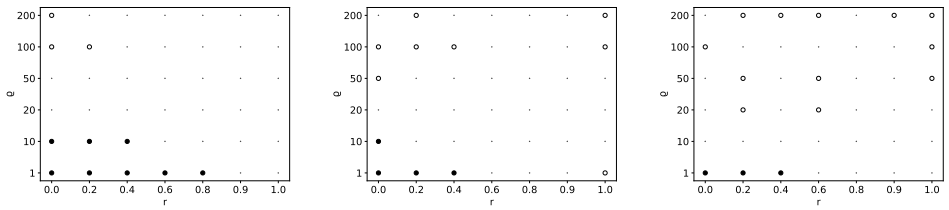


Figure 4.2: Nemenyi test for all datasets $\forall p, \rho$ with $r = 0$.

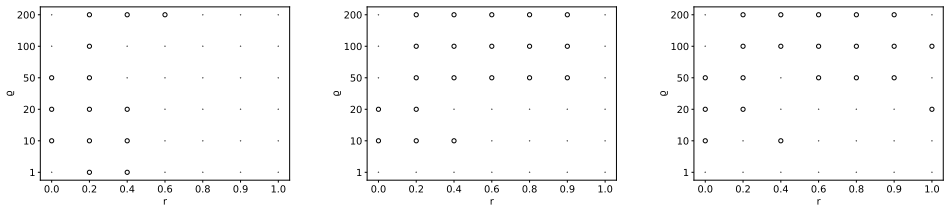
To study the combined effect of both axes, we extend the experiments on a grid with varying strength of label noise r and sub-sampling ρ. Furthermore, as the Nemenyi test between all competitors gives the main trends, we use a Wilcoxon signed-rank test [192] to perform a pairwise analysis between competitors.

Figure 4.3 presents six graphics, each reporting the Wilcoxon test that evaluates one competitor against another, based on the accuracy over all datasets. These graphics form a grid with the horizontal axis representing the label noise strength r and the vertical axis representing the sub-sampling ratio ρ. On each point of a grid (r, ρ), a Wilcoxon rank-signed test [192] is conducted on two competitors for all datasets to determine if there is a significant difference in accuracy between them. If the first competitor wins, "$\circ$" is placed on the grid at this location, "$\cdot$" and "$\bullet$" indicate a tie or a loss respectively.

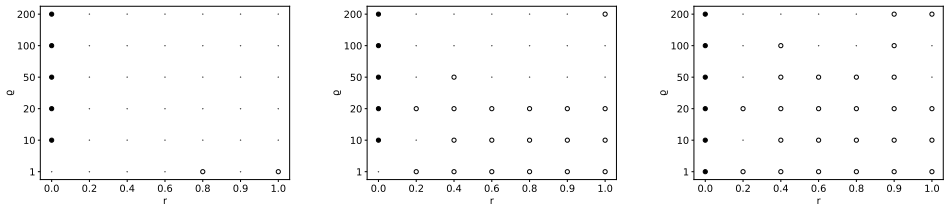
From Figure 4.3 and these pairwise comparisons, we can see interesting patterns emerging. First, on the comparison between IRBL and IRBL2, we can see that IRBL2 is indeed performing significantly better than IRBL when covariate shift is present ($\rho \geq 100$) and significantly worse when there is no covariate shift. The label noise strength does not seem to impact this observation. The classifier used to estimate the density ratio should be close to random guessing as there



(a) IRBL2 vs IRBL $p = 0.01$ (b) IRBL2 vs IRBL $p = 0.02$ (c) IRBL2 vs IRBL $p = 0.05$



(a) K-PDR vs K-KMM $p = 0.01$ (b) K-PDR vs K-KMM $p = 0.02$ (c) K-PDR vs K-KMM $p = 0.05$



(a) IRBL2 vs K-PDR $p = 0.01$ (b) IRBL2 vs K-PDR $p = 0.02$ (c) IRBL2 vs K-PDR $p = 0.05$

Figure 4.3: Results of the Wilcoxon signed rank test computed on all datasets. Each figure compares one competitor versus another for a given trusted data ratio. Figures in the same row are the same competitors against different case of trusted data ratio: $p = 0.01$, $p = 0.02$, $p = 0.05$. In each figure "$\circ$", "$\cdot$" and "$\bullet$" indicate respectively a win, a tie or a loss of the first competitor compared to the second competitor, the vertical axis is ρ and the horizontal axis is r.

is no way to differentiate between a trusted and an untrusted sample from the features only. Checking the quality of this classifier before using it to modify untrusted weights might solve the issue and should be explored. K-PDR is always better or equal to K-KMM, showing solid improvements across the board, especially with a bigger trusted dataset ($p \geq 0.02$). Using a classifier instead of kernels significantly improved the accuracy of KDR across the board. Indeed classifiers are more powerful tools than kernels as they require fewer assumptions about the data distribution. Finally, IRBL2 seems to be significantly better than K-PDR but struggles when there is no label noise ($r = 0$). Seeing such a difference in the performance of K-PDR with label noise is surprising. Indeed when learned per class to estimate the density ratio, classifiers might overfit noisy samples and wrongly estimate the reweighting scheme.

Note that the critical diagrams in Figures 4.1 and 4.2 are an aggregated view of one row or column of graphics from Figure 4.3.

4.7.2 . Second part: Class-Conditional Shift and Prior Shift

The first axis of analysis is the class-conditional shift, driven by the per-class cluster subsampling proposed in Section 4.6.3 of strength ρ_C. The critical diagram presented in Figure 4.4 ranks the competitors on this axis. Figure 4.4 shows that K-PDR is the best approach on class-conditional shifts, followed closely by IRBL, PDR, and the NoCorrection baseline.

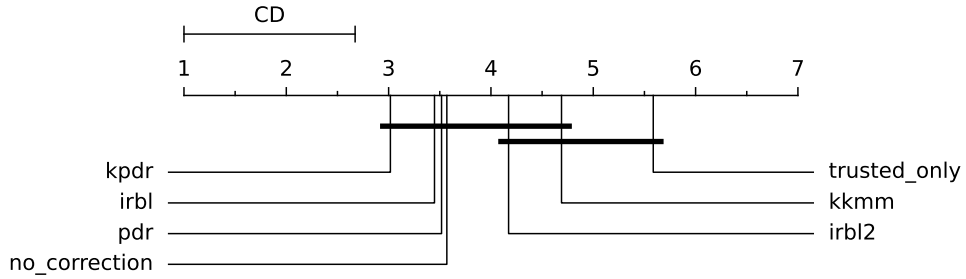


Figure 4.4: Nemenyi test for all datasets $\forall p, \rho_C$ with $\rho = 0$.

The second axis of analysis, about prior shift, is equivalent to the second axis of Section 4.7.1, as explained in Section 4.6.4 (see Figure 4.2 and its analysis).

Figure 4.5 presents six graphics constructed with the same methodology that Figure 4.3.

The conclusions drawn from Figure 4.5 are much more precise than the first part of the experiments. On class-conditional shifts, IRBL2 improves IRBL performances when a prior shift is present ($\rho = 50$) but degrades the performance without a prior shift ($\rho = 0$). The conclusion about the classifier estimating the density ratio being close to random guessing is still meaningful in this case. Again K-PDR is significantly better than K-KMM in all cases of class-conditional and prior shift pairs, experimentally proving the advantage of using classifiers instead of kernels. Finally, IRBL2 is strictly worse than K-PDR in all experiments in class-conditional shifts. Indeed K-PDR is designed to estimate class-conditional shifts. Meanwhile, IRBL only estimates covariate shifts. Thus, K-PDR is designed to solve this task, which is confirmed experimentally.

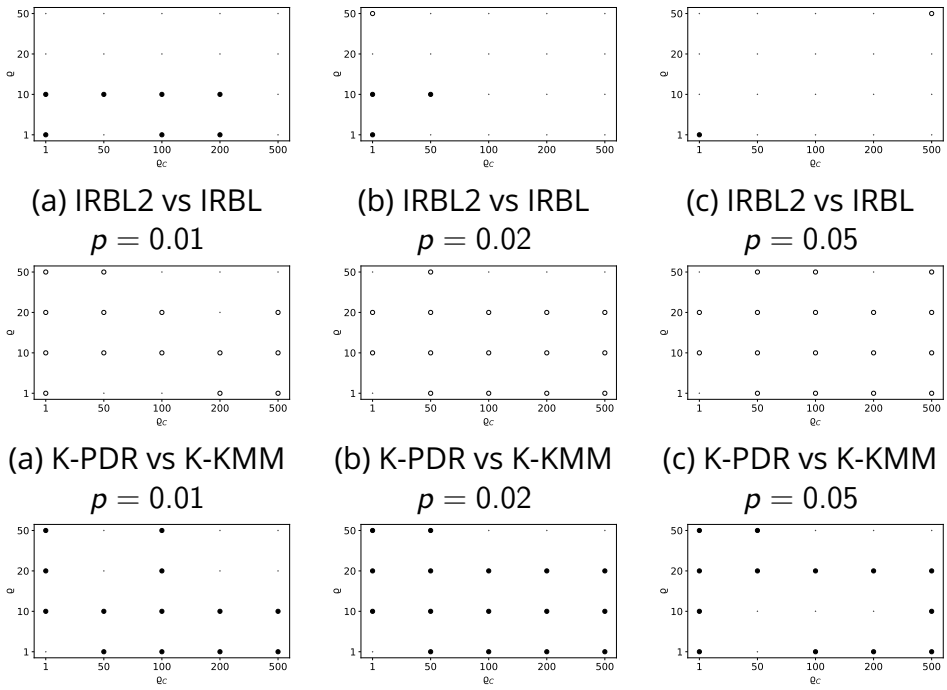


Figure 4.5: Results of the Wilcoxon signed rank test computed on all datasets. Each figure compares one competitor versus another for a given trusted data ratio. Figures in the same row are the same competitors against different case of trusted data ratio: $p = 0.01$, $p = 0.02$, $p = 0.05$. In each figure "$\circ$", "$\cdot$" and "$\bullet$" indicate respectively a win, a tie or a loss of the first competitor compared to the second competitor, the vertical axis is ρ and the horizontal axis is ρ_C.

4.8 . Conclusion

In this Chapter, we have shown the capabilities of the biquality learning framework to design algorithms able to handle closed-set distribution shifts by having access to a trusted and untrusted dataset at training time. We proposed two biquality learning algorithms, IRBL2 and K-PDR, inspired respectively by the *label noise* and the *covariate shift* literature. We reviewed distribution shift sources and their hierarchy and proposed a novel method to create class-conditional shifts in real-world datasets synthetically. Throughout extensive experiments, with numerous simulated distribution shifts and competitors, we demonstrated that K-PDR is the most robust algorithm capable of handling closed-set distribution shifts.

5 - Learning Deep Representations from Weak Supervision

Contents

This Chapter has been published in a workshop of an international conference by the author: Pierre Nodet, Vincent Lemaire, Alexis Bondu, and Antoine Cornuéjols. Contrastive representations for label noise require fine-tuning. In Georg Krempl, Vincent Lemaire, Daniel Kottke, Andreas Holzinger, and Barbara Hammer, editors, *Proceedings of the ECML Workshop on Interactive Adaptive Learning (IAL@ECML PKDD 2021)*, number 3079 in CEUR Workshop Proceedings, pages 89–104, Aachen, 2021.

5.1 . Introduction

The Deep Learning (DL) paradigm has proved very powerful in many tasks. However, recent papers [116, 209] have shown that **"noisy labels"** are a real challenge for end-to-end deep learning architectures. Their test performance is found to deteriorate significantly even if they are able to learn perfectly the train examples. This problem has attracted a lot of suggestions in many recent papers.

Zhang et al. [211] conducted experiments to analyze the impact of label noise on deep architectures, and they found that the performance degradation mainly comes from the representation learning rather than the classification part. It therefore appears very difficult to learn a relevant representation in the presence of label noise, in an end-to-end manner.

To tackle this problem, one option is to exploit an already existing representation which has been learned in an unsupervised way. In particular, Self Supervised Learning [90] (SSL) gathers an ensemble of algorithms which automatically generate supervised tasks from unlabeled data, and, therefore to learn representations from examples that are not affected by label noise. An example of SSL algorithm is Contrastive Learning [86], where a representation of the data is learned by making feature vectors from similar pictures (i.e. generated from the same original picture by using two different transformer functions) to be close in the feature space whereas feature vectors from dissimilar pictures are to be far apart. In [64], the authors propose to initialize the representation with a pre-trained Contrastive Learning one, and then, to use the noisy labels to learn the classification part and fine-tune the representation. It appears that this approach clearly outperforms the end-to-end architecture, where the representation is learned from noisy labels.

But questions remain: is this performance improvement only attributable to the quality of the Contrastive Representation used (i.e. the starting point of fine-tuning)? Or is the fine-tuning step able to promote a better representation? To answer these questions this paper examines the different possibilities to learn a DL architecture in presence of label noise: (i) end-to-end learning (ii) learning only the head part when freezing a contrastive representation and (iii) fine tuning the later representation.

The rest of this paper is organized as follows. The section 2 provides a brief overview of the main families of algorithms dedicated to fight the label noise underlying the issue of preserving a good representation in spite of label noise. Section 3 then describes the experimental protocol. The section 4 will present the results and a deep analysis which will allow us to answer the questions above. The last section raises an interesting conclusion and provides some perspectives for future work.

5.2 . Representation Preserving with Noisy Labels

This section presents a brief overview of the state of the art on *learning deep architecture with noisy labels* emphasizing how these methods *preserve*, to some extent, *the learned representation* in the presence of label noise. For an extended overview, the reader may look [164].

5.2.1 . Preserving by Recovering

The dominant approach to preserve the learned representation is to recover a clean distribution of the data from the noisy dataset. It mostly consists in finding a mapping function from the noisy to the clean distribution thanks to heuristics, algorithms or machine learning models. Three different ways of recovering the clean distribution are usually put forward [134]: (i) sample reweighting; (ii) label correction and (iii) instance moving.

Recovering by Reweighting - The sample reweighting methodology aims at assigning a weight to every samples such that the reweighted population behaves as being sampled from the clean distribution. The Radon-Nikodym derivative (RND) [131] of the clean concept with respect to the noisy concept is the function that defines the perfect reweighting scheme. Many algorithms therefore rely on providing a good estimation of the RND by learning it from the data using Meta Learning [150] or minimizing the Maximum Mean Discrepancy of both distributions in a Reproducing kernel Hilbert space [112, 49]. Many of these methods are inspired by the covariate shift problem [83, 65]. Other algorithms rely on different reweighting schemes that do not involve the RND as done, for instance, in Curriculum Learning [9]. They are described in details later in this section. By doing sample reweighting, algorithms evaluate whether or not a sample is deemed to have been corrupted and assign a lower weight to a suspect sample so that its influence on the training procedure is lowered. The hope is that clean samples are sufficient to learn high-quality representations

Recovering by Relabelling - Another way to recover the clean distribution from the noisy data is to correct the noisy labels. One great advantage over sample reweighting is that corrected samples can be fully used during the training procedure. Indeed, when a sample is corrected, it will count as one entire sample in the training procedure (gradient descent for example), whereas a reweighted noisy sample would get a low weight and would not be used significantly in the training procedure. Thus, when done effectively, label correcting might get better performance. Meta Label Correction (MLC) [162] is an example of this approach where the label correction is done thanks to a model learned using meta learning. One downside of label correction, however, is that the label of a clean sample can get "corrected" or the label of a noisy sample can get changed to a wrong label. Label correcting algorithm assign the same weight to all training examples, even though they might have "corrected" a label based on shaky assumptions. By contrast, Sample reweighting will assign a low weight if the algorithm is not confident in

whether the sample is clean or noisy.

Recovering by Modifying - A third way to recover the clean distribution is by modifying the sample itself so that its position in the feature space gets closer or is moved within an area for its label that seems more appropriate (i.e. obeying regularisation criteria). Finding a transformation in the latent space itself has the advantage to require less labelled samples, or even none at all, as the work is performed on distance between samples themselves, like for example in [159].

5.2.2 . Preserving by Collaboration

Multiple algorithms and agreements measures have been used in many subfields of machine learning such as ensembling [16, 57, 59] or semi supervised learning [204, 12]. They can be adapted to learn with noisy labels by relying on a disagreement method between models in order to detect noisy samples. When the learned models disagree on predictions for the label of a sample, this is considered as a sign that the label of this sample may be noisy. When the models used are diverse enough, these methods are often found to be quite efficient [71, 188].

However these algorithms suffer from learning their own biases and diversity needs to be introduced in the learning procedure. Using algorithms from different classes of models and different origins can increase the diversity among them by introducing more source of biases [100]. Alternating between learning from the data and from the other models is another way to combat the reinforcement of the models' biases [188]. These algorithms rely on carefully made heuristics to be efficient.

5.2.3 . Preserving by Correcting

When learning loss base models, such as neural networks, on label noise, the loss value of a training example can be a discriminative feature to decide if its label is noisy. Deep neural networks seem to have the property that they first learn general and high level patterns from the data before falling prey to overfitting the training samples, especially in the presence of noisy labels [5, 103]. As they are "learned" at a later stage, these noisy examples are often associated with a high loss value [68] which may then highly influences the training procedure and perturb the learned representation [209]. A way to combat label noise is accordingly to focus first on small loss and easy examples and keep the high loss and hard examples for the end of the training procedure. Curriculum Learning [9] is a way to employ this training schema with heuristic based schemes [98, 50, 87, 106] or schemes learned from data [88, 161]. This class of algorithm has the same properties as the ones relying on importance reweighting, but maybe more adapted to training with iterative loss based algorithms such as neural networks or linear models.

Instead of filtering or reweighting samples based on their loss values, one could try to correct the loss for these samples using the underlying noise pattern. Numerous method have been doing so by estimating the noise transition matrix for *Completely at Random* (i.e uniform) and *At Random* (i.e class dependent) noise

[139, 79, 162]. This category of algorithms are still to be tested on more complex noises scenarios such as *Not at Random* (i.e instance and class dependent) noise.

5.2.4 . Preserving by Robustness

The last identified way to preserve the learned representation of a deep neural network in presence of label noise is by using a robust or regularized training procedure. This can take multiple forms from losses to architectures or even optimizers. One of them are Symmetric Losses [177, 63, 24]. A symmetric loss has the property that: $\forall x \in \mathcal{X}$, $\sum_{y \in \mathcal{Y}} L(f(x), y) = c$ where $c \in \mathbb{R}$. These losses have been proven to be theoretically insensitive to Completely at Random (CAR) label noise. Recently, modified versions of the well-known Categorical Cross Entropy (CCE) loss have been designed in order to be more robust and thus more resistant to CAR label noise as is the case for the Symmetric Cross Entropy (SCE) loss [187] or the Generalized Cross Entropy Loss (GCE) [214]. Both of these rely on using the CCE loss combined with a known more robust loss such as the Mean Absolute Error (MAE). However, the resulting algorithms often underfit in presence of too few label noise while they are unable to learn a correct classifier with too much label noise.

All these approaches still adopt the end-to-end learning framework, aiming at fighting the effects of label noise by preserving the learned representation. However, they fail to do so in practice: *decoupling* the learning of the representation, using Self Supervised (SSL) learning, from the classification learning stage itself and then fine tuning the representation with robust algorithms is beneficial for the model performance [211, 64]. A natural question arises about the origin of the performance improvements, and the ability of these algorithms to learn or promote a good representation in presence of label noise. If robust algorithms are unable to learn a representation it should be even better to freeze the SSL representation instead of fine tuning it.

In order to assess the origin of the improvements for different classes of algorithms and different noise levels, we compare the above-mentioned end-to-end approaches against each other when the representation is learned in a self-supervised fashion by either fine tuning or freezing the representation when the classification head is learned. Thus, any difference in the performance would be attributable to the difference in the representation learnt.

5.3 . Experimental Protocol

In [211], the authors showed that when using end-to-end learning, fine tuning the representation on noisy labels harms a lot the final performance, while learning a classifier on frozen embeddings is quite robust to label noise and leads to significant performance improvements over state-of-the-art algorithms if the representation is learned using trustful examples. The latter can be found for instance using confidence and loss value. Nonetheless it is arguable whether these improvements were brought by an efficient self-supervised pretraining (SSL) with SimCLR [25], a contrastive learning method, or by the classification stage of the REED algorithm [211].

The goal of the following experimental protocol is to assess and isolate the role of the contrastive learning stage, in the performance that can be achieved by representative methods as presented in Section 5.2 about state of the art approaches. Specifically, several RLL algorithms have been chosen, one from each of the highlighted families (see Section 5.2 and Table 5.1). For each, the difference in performance between using contrastive learning to learn the representation and the performance reported with the original end-to-end algorithms is measured. These experiments seek to highlight the impact of each RLL algorithms and assess if these are able to promote a better representation than the pretrained contrastive representation through fine-tuning.

The rest of this section describes the experimental protocol used to conduct this set of experiments.

5.3.1 . The tested Algorithms

Section 5.2 presented an overview of the state of the art for learning with label noise organized around families of approaches that we highlighted. Since our experiments aim at studying the properties of each of these approaches, we selected one representative technique from each of these families as indicated in the following.

- In the first family of techniques (*recover the clean distribution*), the algorithms re-weight the noisy examples or attempt to correct their label. One of these algorithm uses what is called Dynamic Importance Reweigthting **(DIW)**. It reweights samples using Kernel Mean Matching (KMM) [83, 65] as is done in covariate shift with Density Ratio Estimators [168]. Because this algorithm adapts well-grounded principles to end-to-end deep learning, it is a particularly relevant algorithm for our experiments.

- CoLearning **(CoL)** [188] is a good representative of the family of *collaborative learning algorithms*. It uses disagreements criteria to detect noisy labels and is tailored for end-to-end deep learning where the two models are branches of a larger neural networks. It appears to be one of the best performing

Algorithms (Date)	Noise Ratio	Clean Validation	Family (Section)
DIW (2020)	×	✓	Reweighting (5.2.1)
CoLearning (2020)	✓	×	Collaborative Learning (5.2.2)
MWNet (2019)	×	✓	Curriculum Learning (5.2.3)
F-Correction (2017)	×	×	Loss Correction (5.2.3)
GLC (2018)	×	✓	Loss Correction (5.2.3)
GCE (2018)	×	×	Robust Loss (5.2.4)

Table 5.1: Taxonomy of robust deep learning algorithms studied in this paper. The **Noise Ratio** column corresponds to whether the algorithm needs the noise rate (✓) to learn from noisy data or not (×). The **Clean Validation** column corresponds to whether the algorithm needs an additional clean validation dataset (✓) to learn from noisy data or not (×).

collaborative algorithm while not resorting to complex methods such as data augmentation or probabilistic modelling like the better known DivideMix [102].

- The third identified way to combat label noise is by *mitigating the effect of high loss samples* [68] by either ditching them or using a loss correction approach. Curriculum learning is often used to remove the examples that are associated with high loss from the training set. **(MWNet)** [161] is one the most recent approach using this technique, which learns the curriculum from the data with meta learning. Besides, Forward Loss Correction **(F-Correction)** [139] and Gold Loss Correction **(GLC)** [79] are two of the most popular approaches to combat label noise by *correcting the loss function*. Both seek to estimate the transition matrix between the noisy labels to the clean labels, the first technique using a supervised approach thanks to a clean validation set, and the second one in an unsupervised manner. Even though many extensions of these algorithm have been developed since then [196, 162], in these experiments, we use F-Correction and GLC since they are way simpler and almost as effective.

- Lastly, in recent literature, a new emphasis is put on the research of new loss functions that are conducive to better risk minimization in presence of noisy labels *for robustness purpose*. For example, [177, 24] show theoretically and experimentally that when the loss function satisfies a symmetry condition, described below, this contributes to the robustness of the classifier. The Generalized Cross Entropy **(GCE)** [214] is the robust loss chosen in this benchmark as it appears to be very effective.

A note *about additional requirements*: These algorithms may have additional requirements, mostly some knowledge about the noise properties. These are described in table 5.1. In the experiments presented below, the clean validation

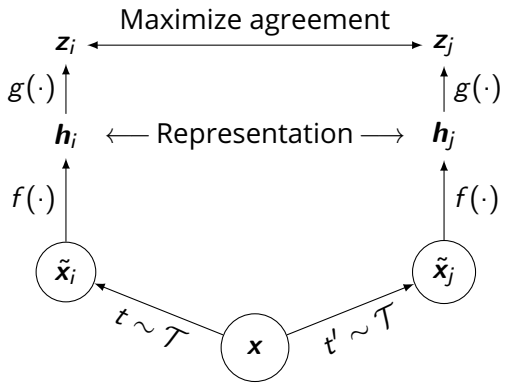


Figure 5.1: Figure from [25]: "A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\boldsymbol{h}$ for downstream tasks."

dataset is set to be 2 percent of the total training data, like in [161, 218], and the noise probability is provided to the algorithms that need it.

A note *about the choice of the pretrained architecture*: We chose to use SimCLR for Self-Supervised Learning (SSL) as done in [211].

SimCLR is a contrastive learning algorithm that is composed of three main components (See Figure 5.1): a family of data augmentation $\mathcal{T}$, an encoder network $f(\cdot)$ and a projection head $g(\cdot)$. Data augmentation is used as a mean to generate positive pairs of samples: a single image $\mathbf{x}$ is transformed into two similar images $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ by using a data augmentation module $\mathcal{T}$ with different seeds t and t'. Then the two images go through an encoder network $f(\cdot)$ to extract an image representation $\mathbf{h}$, such as $\mathbf{h}_i = f(\tilde{\mathbf{x}}_i)$ and $\mathbf{h}_j = f(\tilde{\mathbf{x}}_j)$. Finally a projection head $g(.)$ is used to train the contrastive objective in a smaller sample space $\mathbf{z}$, with $\mathbf{z}_i = g(\tilde{\mathbf{h}}_i)$ and $\mathbf{z}_j = g(\tilde{\mathbf{h}}_j)$. The contrastive loss used is called the NT-Xent, the normalized temperature-scaled cross entropy loss, and defined by the following formula:

$$\ell(\mathbf{z}_i, \mathbf{z}_j) = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \tag{5.1}$$

where τ is the temperature scaling and sim is the cosine similarity. The final loss is computed across all positive pairs, both (i, j) and (j, i), in a mini-batch. When the training of SimCLR is complete, the projection head $g(.)$ is dropped and the embeddings $\mathbf{h}$ are used as an image representation in downstream tasks.

Other SSL algorithms could have been used as well, such as Moco [77, 26] or Bootstrap Your Own Latent (BYOL) [66]. However, we do not expect that the main conclusions of the study would be much changed.

5.3.2 . Datasets

The datasets chosen in this benchmark are two image classification datasets namely CIFAR10, CIFAR100. They are two famous image classification datasets, containing only clean examples and as such, we will simulate symmetric (Completly at Random) and asymmetric (At Random) noise as defined later in section 5.3.3. These benchmarks should be extended to other image classification datasets such as FashionMNIST, Food-101N, Clothing1M and Webvision and to other classification tasks such as text classification or time series classification.

5.3.3 . Simulated Noise

As datasets chosen in Section 5.3.2 contains clean labels, label noise will be introduced synthetically on the training samples. Two artificial noise models will be used, a symmetric (Completely at Random) and asymmetric (At Random) noise. Symmetric noise corrupts a label from one class to any other classes with the same probability, meanwhile the asymmetric corrupts a label to a similar class only. Similar classes are defined through class mappings. For CIFAR-10, the class mappings are TRUCK $\rightarrow$ AUTOMOBILE, BIRD$\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG. For CIFAR-100, the class mappings are generated from the next class in that group (where 100 classes are categorized into 20 super-classes of 5 classes). These class mappings are the ones introduced in [139, 214].

5.3.4 . Implementation Details

We give some implementation details for reproducibility and / or a better understanding of the freezing process in the experiments:

- On CIFAR10 and CIFAR100 the SGD optimizer will be used to train the final Multinomial Logistic Regression with an initial learning rate of 0.01, a weight decay of $1e^{-4}$ and a non-Nesterov momentum of 0.9. The learning rate will be modified during training with cosine annealing [114]. The batch size is 128.

- When doing the *"Freeze"* experiments, the weights of SimCLR from [64] will be used and will not be modified during the training procedure. All the weights up to before the projection head of SimCLR are used, then the dimension output of the feature encoder is 2048 for CIFAR10 and CIFAR100. The classification architecture is composed of a single linear layer with an output dimension of 10 (or 100), corresponding to the number of classes. Thus, when trained with the Categorical Cross Entropy it corresponds to a usual logistic regression. This classifier is going to be learned with multiple algorithms robust to label noise. These algorithms are not modified from their original formulation.

- The *"Fine Tuning"* experiments follow the same implementation as the *"Freeze"* experiments. However, the weights of the same pretrained SimCLR encoder are allowed to be modified by backpropagation.

- Based on their public implementation and / or article we re-implemented all the algorithm tested (DIW [49], CoL [188], MWNet [161], F-Correction [139], GLC [79] and GCE [214]). All these re-implemented algorithms will soon be available as an open source library easily usable by researchers and practitioners. These custom implementations have been verified to produce, under the same condition stated in the corresponding original papers (noise models, network architectures, optimizers, ...), the same results or results in the interval of confidence (for clean or noisy labels). We may thus be confident that results in the different parts of the Tables 5.2 and 5.3 are comparable.

- The experiments have been run multiple times for all algorithms, some datasets, some noise models and some noise ratios with different seeds to see the seed impact on the final performance of the classifier. For all algorithms, the standard deviation of the accuracy was less than 0.1 percent.

5.4 . Results

This section reports the results obtained using the protocol described in section 5.3. They are presented in the tables 5.2 and 5.3 corresponding to the two tested datasets CIFAR10 and CIFAR100. Each table is composed of three row subsections corresponding to the different types of representation used, which can be learned in a *End-to-End* manner (A), be taken from an already existing SSL model, either *Frozen* (B) or *Fine tuned* (C). Moreover they are composed of two columns subsections corresponding to the noise model used to corrupt samples (symmetric or asymmetric).

These tables present the results from different studies: (A) The first part of these tables about *"End-to-End learning"* are results reported in the respective papers [49, 188, 161, 139, 79, 214] or reported in [64]; (B) The second part about *"Freeze"* experiments conducted in this paper, are made by re-implementing the referred algorithms from scratch; (C) The *"Fine Tuning"* experiments are results reported in [64].

The interpretation of the Table 5.2 and 5.3 will be done in two times, first a comparison between whole blocks (as (A) against (B)) will give insights on how deep neural networks learn representations on noisy data and how robust algorithms helps to improve the learning process or helps to preserve a given representation. Then in a second time comparisons in a given block will be made against multiple algorithms to see how well these conclusions works on different preservation families given in Section 5.2.

Algorithms		CIFAR10 Clean	Symmetric						Asymmetric	
		0	20	40	60	80	90	95	20	40
DIW [49]	End-to-End (A)			80.4	76.3					84.4
CoL [188]			93.3	91.2		49.2			88.2	82.9
MWNet [161]		95.6	92.4	89.3	84.1	69.6	25.8	18.5	93.1	89.7
F-Correction [139]		90.5	87.9			63.3	42.9		90.1	
GLC [79]		95.0	95.0	95.0	95.0	90.0	80.0	76.0		
GCE [214]		93.3	89.8	87.1	82.5	64.1			89.3	76.7
DIW	Freeze (B)	91.3	91.2	90.8	90.5	89.8	89.2	88.1	91.0	90.6
CoL		91.1	91.1	90.9	90.6	89.9	89.4	88.8	90.8	89.9
MWNet		91.3	91.2	90.8	90.6	89.8	88.2	82.4	90.9	86.4
F-Correction		90.8	90.5	90.1	89.6	88.4	88.0	88.1	88.9	88.4
GLC		90.7	89.7	90.0	89.5	89.0	88.5	88.3	88.7	88.2
GCE		91.1	90.8	90.7	90.5	90.4	90.0	89.1	90.9	89.0
DIW	Fine Tuning (C)	94.5	94.5	94.5	94.5	94.0	92.0	89.1	94.2	93.6
CoL		93.9	94.6	94.6	94.2	93.6	92.7	91.7	94.0	93.7
MWNet [64]		94.6	93.9	92.9		91.5	90.2	87.2	93.7	92.6
F-Correction		94.0	93.4	93.1	92.9	92.3	91.4	90.0	93.6	92.8
GLC		93.5	93.4	93.5	93.1	92.0	91.2	88.3	93.2	92.1
GCE [64]		94.6	94.0	92.9		90.8	88.4	83.8	93.5	90.3

Table 5.2: Final accuracy for the different models on CIFAR10 under symmetric and asymmetric noises and multiple noise rates.

First, we observe when comparing section (A) and (B) from both tables that *"Freeze"* experiments consistently outperforms *"End-to-End"* experiments as soon as the data stop being perfectly clean. Using a pretrained self-supervised representation such as SimCLR improves significantly the performances of the final classifier. Outside of well controlled and perfectly clean datasets all selected algorithms are not able to learn a good enough representation from the noisy data and are beaten by a representation learned without resorting to using given labels. Robust Learning to Label noise algorithms, especially designed for deep learning, can preserve an already good representation from noisy labels but are unable to learn a good representation from scratch.

Then, we observe when comparing section (B) and (C) from both tables that *"Fine Tuning"* experiments consistently outperforms *"Freeze"* at noise rates less than 80 for the symmetric case and less than 40 for the asymmetric case. The nature of the final classifier used after the learned representation partially explains these results; we used a single dense layer (see Section 5.3.4). This classifier may under-fit as the number of learnable parameters might be too low to actually fit complex datasets such as CIFAR10 and CIFAR100 even with a good given representation. Using more complex classifiers such as Multi-Layer Perceptron could have led to comparable performances than fine tuning even for low noise rates. This point leaves room for further investigation. Having the possibility to fine tune the representation to better fit the classification task induces the risk to actually degrade it.

Outside of well controlled and perfectly clean datasets, practitioners should

Algorithms		CIFAR100 Clean	Symmetric						Asymmetric	
		0	20	40	60	80	90	95	20	40
DIW [49]	End-to-End (A)			53.7	49.1					54.0
CoL [188]			75.8	73.0		32.8				
MWNet [161]		79.9	74.0	67.7	58.7	30.5	5.2	3.0	71.5	56.0
F-Correction [139]		68.1	58.6			19.9	10.2		64.2	
GLC [79]		75.0	75.0	75.0	62.0	44.0	24.0	12.0	75.0	75.0
GCE [214]		76.8	66.8	61.8	53.2	29.2			66.6	47.2
DIW	Freeze (B)	65.6	65.1	64.0	62.9	59.0	53.3	42.5	61.7	49.0
CoL		65.8	65.0	64.0	63.4	62.3	60.0	57.0	64.1	58.6
MWNet		66.6	66.6	66.2	65.4	63.7	59.8	49.5	64.8	54.5
F-Correction		66.5	64.7	61.8	58.8	54.5	51.7	50.8	58.4	56.5
GLC		58.5	57.8	52.3	51.1	41.6	40.1	35.3	51.4	50.3
GCE		63.5	62.9	61.5	60.0	55.7	51.0	49.9	51.2	48.3
DIW	Fine Tuning (C)	73.8	74.9	74.9	74.5	70.2	62.3	50.4	71.8	62.8
CoL		73.7	74.8	74.8	75.0	73.2	67.3	62.0	72.6	70.3
MWNet [64]		75.4	73.2	69.9		64.0	57.6	44.9	72.2	64.9
F-Correction		69.8	70.1	69.1	69.5	66.9	62.1	57.0	70.3	66.2
GLC		69.7	69.4	68.6	62.5	50.4	32.1	18.7	68.2	62.3
GCE [64]		75.4	73.3	70.1		63.3	55.9	45.7	71.3	59.3

Table 5.3: Final accuracy for the different models on CIFAR100 under symmetric and asymmetric noises and multiple noise rates.

first consider to learn a self-supervised representation and then either fine tune it or freeze it with classifier learned with robust algorithms. Self-Supervised Learning (SSL) algorithm such as SimCLR seems to perfectly fit this task, but other SSL algorithms could be used and explored.

Another observation from this benchmark is about the difference in performance between all the tested algorithms. Indeed, if we consider part (B) of Table 5.2, for both noise models and all noise rates, the performances between the algorithms are close, around 0.1 point in accuracy with some exceptional data points. It shows that even complex algorithms have a hard time beating simpler approaches when they are compared with an already learned representation.

The same observation can be done for the part (B) of Table 5.3 (for CIFAR 100), especially for the symmetric noise. However the differences between algorithms are better put in perspective with this more complex dataset which contains 10 time more classes and 10 time less samples per classes. We notice that some algorithms start to struggle at high symmetric noise rate or for the more complex asymmetric noise model. For example, GLC is under-performing against competitors for all cases and is under-performing against its end-to-end version. One reason could be the small size used for the validation dataset as the transition matrix is evaluated on it in a supervised manner. The small number of samples may impact the performance of the transition matrix estimator. Much less so than the estimator proposed by F-Correction which seems to perform fine even on CIFAR100 for all symmetric noises, yet only above average on asymmetric noises. Seeing F-Correction and GLC not performing well on asymmetric noise for both dataset is surprising as these algorithms were both particularly designed for this case.

Lastly, we observe on both Tables 5.2 and 5.3 that algorithms with additional knowledge on the noise model (see Table 5.1) have an edge over algorithms that do not, especially on the hardest cases with more classes, higher noise ratio or more complex noise model. CoL requires the noise ratio as its efficiency relies on the hyper parameters value corresponding to the injection of pseudo labels and confidence in model prediction that are dependent of the noise ratio. CoL emerges among the most well rounded and most efficient algorithm for all noise models, noise rates and datasets thanks partially to this additional knowledge. On the other hand, GLC, DIW and MWNet require an additional clean validation dataset in order to estimate the noise model or a proxy of it to correct the learning procedure on the noisy dataset. We could expect these algorithms to perform better than CoL as they would be able to deal with more complex noise models and have a fine-grained policy for correcting noisy samples. Still these algorithms are not able in these experiments to get a better accuracy than CoL and perform on par with it.

Finally we need to emphasize that only two datasets have been used in this study, specially two datasets about image classification. In order to stronger our claims, more experiments should be conducted.

5.5 . Conclusion

In this paper our contribution was to suggest new insights about *decoupling* against *end-to-end* deep learning architectures to learn, preserve or promote a good representation in case of label noise. We presented (i) a new view on a part of the state of the art: the ways to preserve the representation (ii) and an empirical study which completes the results and the conclusions of other recent papers [217, 64, 211]. Experiments conducted draw a comprehensive picture of performances by featuring six methods and nine noise instances of three different kinds (none, symmetric, and asymmetric). Our added value for the empirical study is the comparison between the "freeze" and the "fine tuning" results.

One conclusion we are able to draw is that designing algorithms that preserve or promote good representation under label noise is not the same as designing algorithms capable of learning from scratch a good representation under label noise. To make end-to-end learning succeed in this setup researchers should take a better approach when designing such algorithms.

Another element that emerged from the experiments was the efficiency of both freeze and fine tuning approaches in comparison to the end-to-end learning approach. Even the most complex algorithms such as DIW when trained in an end-to-end manner are not able to beat simple robust loss as GCE when trained with fine tuning. It questions usual experimental protocols of Robust Learning to Label (RLL) noise papers and questions the recent advances in the field. Evaluating RLL algorithms with pretrained architectures should be the norm as it is easy to do so and the most efficient way for practitioners to train model on noisy data.

One more strong point in this conclusion is that in presence of noise the experiments show that fine tuning of Contrastive representation allows the six methods to achieve better results than their end-to-end learning version and represent a new reference compare to the recent state of art. Results are also remarkable stable versus the noise level.

Since fine-tuned representations are shown to outperform frozen ones, one can conclude that noise-robust classification heads are indeed able to promote meaningful representations if provided with a suitable starting point (contrastingly to readers of [217, 64] who might prematurely jump to the inverse conclusion).

However these experiments could be extended to be more exhaustive in two ways: (i) SimCLR is not the only recent and efficient contrastive learning algorithms, MOCO [77, 26] or Bootstrap Your Own Latent (BYOL) [66] could have been used as said earlier in the paper, but other self-supervised or unsupervised algorithms could have been used such as Auto-Encoder [96] or Flow [93]; (ii) experiments could be extended to datasets from other domains such as text classification or time series classification.

6 - Biquality Learning at Orange

Weakly Supervised Learning at Orange is ubiquitous. First of all because Fraud Detection is one of the most important use case of Machine Learning at Orange. Fraud in Telecommunications is a huge topic and a huge market, it is reported than around 32.7 billion dollars are lost every year in fraud in the telecommunication sector (note that fraud amount is usually under estimated). Orange is a victim of these fraud and combats it. Second, because Cyber-Security is another important topic at Orange for multiple reasons. Since historically Orange has been gathering and storing very sensitive data of their users and had to protect itself against cyber-attacks, and more recently because Orange tried to propose its grown expertise in the domain to others thanks to the consulting services offer by Orange Cyber-Defense. Finally because Orange has more or less always been applying principle from data mining and machine learning, it has long term deployment of machine learning models and experience in MLOps [97], a domain where managing model life cycle is synonym of dealing with various of weak signals corresponding to predictions of past model versions. Thus, it is an important research topic and gaining knowledge on how to develop better algorithm is a key goal.

In practice, when Data Scientists are assigned tasks about attack or fraud detection, they are given an untrusted data where they absolutely cannot trust the quality of the labeling process. Historically, Robust Machine Learning algorithms and methodologies have been deployed to combat this issue, but most importantly experts have been required to help cleanly label the available datasets. Yet, a common pattern was the lack of time and number of such experts to completely label the datasets. That is where we felt the need to review algorithms that were able to leverage such datasets. In practice, we found few alternatives that worked with Orange own machine learning processes, especially the ones revolving about our in-house AutoML platform [14, 15] and Orange use cases centered about tabular classification tasks.

That is why we proposed to study the field of Biquality Learning, in the lenses of tabular classification with agnostic classifiers. In addition to the knowledge provided in this manuscript, we propose a Python library to gather the empirical knowledge gathered during this PhD; a shared knowldege to Orange.

biquality-learn : a Python library for Biquality Learning

We designed a **biquality-learn** library which follows the **scikit-learn** API, meaning that it provides a consistent interface for training and using biquality learning algorithms. It includes a variety of reweighting algorithms, plugin correctors, as well as functions for simulating label noise and generating sample data.

To install **biquality-learn**, a simple command *pip install biquality-learn* is enough and will install all necessaries dependencies.

Overall, the goal of **biquality-learn** is to make well-known and proven biquality learning algorithms accessible and easy to use for everyone, and to enable researchers to experiment in a reproducible way on biquality data.

- Source Code : https://github.com/pierrenodet/biquality-learn/
- Documentation : https://biquality-learn.readthedocs.io/en/stable/

6.0.1 . Design of the API

Scikit-learn [140] is a machine learning library for Python with a design philosophy that emphasizes consistency, simplicity, and performance. The library provides a consistent interface for various algorithms, making it easy for users to switch between models. It also aims to make machine learning easy to get started with through user-friendly API and clear documentation. Additionally, it is built on top of efficient numerical libraries (numpy [72], and SciPy [184]) to ensure that models can be trained and used on large datasets in a reasonable amount of time.

In **biquality-learn**, we followed the same principle, implementing a similar API with *fit* and *predict* functions. In addition to passing the input features and the labels as in **scikit-learn**, in **biquality-learn**, the *sample_quality* property is provided to specify from which dataset the sample is originating from. Especially, values of 0 indicate an untrusted sample and values of 1 indicate a trusted sample.

6.0.2 . Training Biquality Learning Classifiers

To train a biquality learning algorithm using **biquality-learn**, you will need to have a dataset with a trusted and untrusted portion. If you don't have one and want to experiments on synthetic data, you can generate such a dataset using functions from **scikit-learn**, or you can obtain it from external sources.

Here is an example of how to train a biquality classifier using the KPDR (K-Probabilistic Density Ratio) algorithm from **biquality-learn**:

```python
# Generate a dataset with a trusted and untrusted portion
from sklearn.datasets import make_classification
from sklearn.model_selection import StratifiedShuffleSplit

X, y = make_classification()

trusted, untrusted =
↪  next(StratifiedShuffleSplit(train_size=0.1).split(X, y))

# Train a biquality classifier using KPDR
from sklearn.linear_models import LogisticRegression
from bqlearn.kdr import KPDR

bqclf = KPDR(LogisticRegression(), LogisticRegression())

# Specify the sample quality for each sample in the dataset
n_samples = X.shape[0]
sample_quality = np.ones(n_samples)
sample_quality[untrusted] = 0

# Fit the classifier to the data
bqclf.fit(X, y, sample_quality=sample_quality)

# Use the classifier to make predictions on the same data
predictions = bqclf.predict(X)
```

6.0.3 . Criteria of Inclusion in biquality-learn

In Chapter 2 we did an extensive review about Biquality Learning and did a state-of-the-art in Section 2.4. At the end of Chapter in Section 2.8 we stated which subparts of this state-of-the-art was useful for Orange and their particular use cases. That's why we designed a library centered about approaches for tabular data and tabular classifiers. Thus restricting approaches that are truly classifier agnostic or implementable within **scikit-learn**'s API.

We summarized all implemented algorithms and what kind of corruptions they are able to handle in a Table.

6.0.4 . scikit-learn's metadata routing

scikit-learn's metadata routing system can be used to seamlessly incorporate the *sample_quality* property into the training and prediction process of biquality learning algorithms. This allows you to use **biquality-learn**'s algorithms in a similar way to **scikit-learn**'s algorithms, by passing the *sample_quality* property as an

Algorithms	Dataset Shifts	Weaknesses of Supervision
EasyAdapt [37]	✓	×
TrAdaBoost [36]	✓	×
Unhinged (Linear/Kernel) [177]	×	✓
Backward [123, 139] (with GLC [79])	×	✓
IRLNL [111, 186] (with GLC [79])	×	✓
Plugin [213] (with GLC [79])	×	✓
KKMM [49]	✓	✓
IRBL [134]	×	✓
IRBL2	✓	✓
KPDR	✓	✓

Table 6.1: Implemented Algorithms in Biquality Learn

additional argument to the *fit*, *predict*, and other methods.

For example, when https://github.com/scikit-learn/scikit-learn/pull/24250 will be merged, it will be possible to make a bagging ensemble of biquality classifiers thanks to the *BaggingClassifier* implemented in **scikit-learn** without overriding its behaviour on biquality data.

```
from sklearn.ensemble import BaggingClassifier

biquality_bagging = BaggingClassifier(bqclf).fit(X, y,
↪  sample_quality=sample_quality)
print(biquality_bagging.score(X_test, y_test))
```

6.0.5 . Cross-Validating Biquality Classifiers

Any cross-validators working for usual Supervised Learning can work in the case of Biquality Learning. However, when splitting the data into a train and test set, untrusted samples need to be removed from the test set to avoid computing supervised metrics on corrupted labels. That's why *make_biquality_cv* is provided by **biquality-learn** to post-process any **scikit-learn** compatible cross-validators.

Here is an example of how to use **scikit-learn**'s *RandomizedSearchCV* function to perform hyperparameter validation for a biquality learning algorithm in **biquality-learn**:

```
from sklearn.model_selection import RandomizedSearchCV
from sklearn.utils.fixes import loguniform
from bqlearn.model_selection import make_biquality_cv

# Specify parameters and distributions to sample from
param_dist = {"final_estimator__C": loguniform(1e3, 1e5)}

# Run randomized search
```

```
n_iter_search = 20
random_search = RandomizedSearchCV(
   bq_clf,
   param_distributions=param_dist,
   n_iter=n_iter_search,
   cv=make_biquality_cv(X, sample_quality, cv=3)
)
random_search.fit(X, y, sample_quality=sample_quality)
```

6.0.6 . Quality Implementations of Biquality Learning Algorithms

For this library we dedicated a important part of the work to the quality of implementations. An example of such an algorithm is TrAdaBoost. TrAdaBoost [36] is an extension of boosting to transfer learning. TrAdaBoost learns on both trusted and untrusted data and reweights samples at every iteration depending on their trustiness and the error made by the stagging model. On trusted data, samples are exactly reweighted as in AdaBoost [57], where difficult samples get more attention. On untrusted data, the Weighted Majority Algorithm [108] is used, where misclassified samples are deemed useless for the task.

As there is no reference of an actual version of the TrAdaBoost algorithm fitting all these features, we propose a modified version of TrAdaBoost for multi-class classification inspired by SAMME [73] that handles a learning rate hyper-parameter and correct the natural weight drift of TrAdaBoost using Dynamic TrAdaBoost [4] in Algorithm 7).

This novel and improved version of TrAdaBoost is available in **biquality-learn**.

Algorithm 7: Dynamic Multi-class TrAdaBoost

Input: Trusted Dataset D_T, Untrusted Dataset D_U

Parameter: Classifier Familiy $\mathcal{F}$, N number of iterations, λ learning rate

Let $T = |D_T|$ and $U = |D_U|$

$\forall i \in [\![1, T+U]\!], w_i^0 = 1$

for $t \in [\![1, N]\!]$ **do**

  $\forall i \in [\![1, T+U]\!], p_i^t = w_i^{t-1} / \sum_{k=1}^{T+U} w_i^{t-1}$

  Learn $f^t \in \mathcal{F}$ on $D_T \cup D_U$ with distribution p^t

  Let $\epsilon^t = \frac{\sum_{k=1}^{T+U} w_i^{t-1} |f^t(x_i) - y_i|}{\sum_{k=1}^{T+U} w_i^{t-1}}$

  if $\epsilon^t \geq 1 - \frac{1}{K}$ **then**

    break

  Let $\beta^t = \frac{\epsilon^t}{1-\epsilon^t} \frac{1}{K-1}$ and $\beta_c = \frac{1}{1+\sqrt{2 \ln U/N}}$

  Let $C^t = (1 - \epsilon^t)(1 + \epsilon^t \beta^{t(-\lambda)})$

  for $i \in [\![1, T]\!]$ **do**

    $w_i^t = w_i^{t-1} \beta^{t(-\lambda |f^t(x_i) - y_i|)}$

  for $i \in [\![T+1, U]\!]$ **do**

    $w_i^t = w_i^{t-1} C^t \beta_c{}^{\lambda |f^t(x_i) - y_i|}$

Output: $f = \arg\max_k \sum_{t=1}^{N} -\lambda \ln(\beta^t) \mathbb{1}(f^t = k)$

6.0.7 . Simulating Corruptions with the Corruption API

The *corruption* module in **biquality-learn** provides several functions to artificially create biquality datasets by introducing synthetic corruption. These functions can be used to simulate various types of label noise or imbalances in the dataset, which can be useful for testing and evaluating biquality learning algorithms.

Here is a brief overview of the functions available in the corruption module:

- *make_weak_labels*: This function adds weak labels to the dataset by learning a classifier on a subset of the dataset and using its predicitons as a new label.

- *make_label_noise*: This function adds label noise to the dataset by randomly flipping a specified fraction of the labels.

- *make_instance_dependent_label_noise*: This function adds instance-dependent label noise to the dataset by flipping labels with a probability that is dependent on the instance features.

- *uncertainty_noise_probability*: This function computes the probability of corrupting a label based on the predicted class probabilities.

- *make_imbalance*: This function creates an imbalanced dataset by oversampling or undersampling the minority class.
- *make_cluster_imbalance*: This function creates an imbalanced dataset by sampling more or fewer instances from certain clusters.

These functions can be used to artificially create biquality datasets for testing and evaluating biquality learning algorithms.

Conclusion

In conclusion, this manuscript aimed to determine the feasibility of designing a training algorithm that could automatically adapt to various weak supervision scenarios. Through the proposal of a novel taxonomy of Weakly Supervised Learning, named the Weak Supervision Cube, and the definition of a generic learning framework named Biquality Learning, we were able to review the state-of-the-art algorithms that suppose the availability of a small trusted dataset. Under this framework, we proposed an algorithm based on Importance Reweighting for Biquality Learning (IRBL) that is classifier-agnostic, and is based on the empirical estimation of the Radon-Nikodym derivative (RND), which is the sample reweighting scheme to build a risk-consistent estimator on untrusted data. Our results demonstrate that IRBL can improve the performance of various classifiers on biquality datasets. In addition to addressing the issue of weak supervision, this thesis also deals with the problem of dataset shifts. To solve it, we proposed a variant of IRBL called IRBL2, and its covariate-shift version named KPDR both based on the RND estimation. This approach allows the algorithm to better handle the problem of corrupted and untrusted labels in the presence of dataset shifts. We also developed a library called bq-learn that implements the proposed methods and algorithms. This library is designed to make it easy for practitioners to use and apply the proposed solutions to their own datasets and use-cases. It provides a user-friendly interface and a set of pre-built functions that can be easily integrated into existing machine learning pipelines. The biquality-learn library can be used to handle weak supervision and dataset shifts in a unified way. It provides a powerful tool for practitioners to improve the performance of supervised learning algorithms on weakly labeled and corrupted datasets.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Steven P. Abney. Bootstrapping. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 360–367. ACL, 2002.

[3] Charu C. Aggarwal, Xiangnan Kong, Quanquan Gu, Jiawei Han, and Philip S. Yu. Active Learning: A Survey. In Charu C. Aggarwal, editor, *Data Classification: Algorithms and Applications*, chapter 22, pages 571–605. CRC Press, 2014.

[4] Samir Al-Stouhi and Chandan K Reddy. Adaptive boosting for transfer learning using dynamic updates. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 60–75. Springer, 2011.

[5] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, pages 233–242, 2017.

[6] Yoram Baram, Ran El-Yaniv, and Kobi Luz. Online Choice of Active Learning Algorithms. *Journal of Machine Learning Research*, 5:255–291, 2004.

[7] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: a survey. *Machine Learning*, 109:719–760, 2020.

[8] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning*, pages 41–48, 2009.

[10] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88, 2007.

[11] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[12] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[13] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[14] Marc Boullé. Modl: a bayes optimal discretization method for continuous attributes. *Machine learning*, 65(1):131–165, 2006.

[15] Marc Boullé. Compression-based averaging of selective naive bayes classifiers. *The Journal of Machine Learning Research*, 8:1659–1685, 2007.

[16] Leo Breiman. Bagging predictors. *Machine Language*, 24(2):123–140, August 1996.

[17] Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

[18] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329–353, May 2018.

[19] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[20] Chih-Chung Chang and Chih-Jen Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011.

[21] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[22] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-supervised learning*. Adaptive computation and machine learning. MIT Press, Cambridge, Mass, 2006. OCLC: ocm64898359.

[23] Moses Charikar, Jacob Steinhardt, and Gregory Valiant. Learning from untrusted data. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, page 47–60, 2017.

[24] Nontawat Charoenphakdee, Jongyeong Lee, and Masashi Sugiyama. On symmetric losses for learning from corrupted labels. In *International Conference on Machine Learning*, volume 97, pages 961–970, 2019.

[25] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR, 13–18 Jul 2020.

[26] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv:2003.04297*, 2020.

[27] De Cheng, Tongliang Liu, Yixiong Ning, Nannan Wang, Bo Han, Gang Niu, Xinbo Gao, and Masashi Sugiyama. Instance-dependent label-noise learning with manifold-regularized transition matrix estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16630–16639, 2022.

[28] Hong-Min Chu and Hsuan-Tien Lin. Can Active Learning Experience Be Transferred? *2016 IEEE 16th International Conference on Data Mining*, pages 841–846, 2016.

[29] Stephen Clark, James R. Curran, and Miles Osborne. Bootstrapping pos-taggers using unlabelled data. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 49–55. ACL, 2003.

[30] Timothé Collet. *Optimistic Methods in Active Learning for Classification*. PhD thesis, Université de Lorraine, 2018.

[31] D. Conte, P. Foggia, G. Percannella, F. Tufano, and M. Vento. A method for counting people in crowded scenes. In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 225–232, 2010.

[32] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2022.

[33] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. *The Journal of Machine Learning Research*, 12:1501–1536, 2011.

[34] Timothee Cour, Ben Sapp, and Ben Taskar. Learning from partial labels. *Journal of Machine Learning Research*, 12(42):1501–1536, 2011.

[35] Koby Crammer, Michael Kearns, and Jennifer Wortman. Learning from multiple sources. *Journal of Machine Learning Research*, 9(8), 2008.

[36] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *International Conference on Machine Learning*, pages 193–200, 2007.

[37] Hal Daumé III. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815*, 2009.

[38] Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings, 2010.

[39] Michael Davy. A review of active learning and co-training in text classification. Technical report, Trinity College Dublin, Department of Computer Science, 2005.

[40] A Philip Dawid. The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379):605–610, 1982.

[41] Louis Desreumaux and Vincent Lemaire. Learning active learning at the crossroads? evaluation and discussion. In Georg Krempl, Vincent Lemaire, Daniel Kottke, Andreas Holzinger, and Adrian Calma, editors, *Proceedings of the ECML Workshop on Interactive Adaptive Learning (IAL@ECML PKDD 2020)*, number 2660 in CEUR Workshop Proceedings, pages 38–54, Aachen, 2020.

[42] Dheeru Dua and Casey Graff. Uci machine learning repository, 2017.

[43] Lixin Duan, Ivor W Tsang, and Dong Xu. Domain transfer multiple kernel learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(3):465–479, 2012.

[44] Sandra Ebert, Mario Fritz, and Bernt Schiele. Ralf: A reinforced active learning formulation for object class recognition. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3626–3633, 2012.

[45] Charles Elkan. The foundations of cost-sensitive learning. In *International joint conference on artificial intelligence*, volume 17, pages 973–978. Lawrence Erlbaum Associates Ltd, 2001.

[46] Charles Elkan. Log-linear models and conditional random fields. *Tutorial notes at CIKM*, 8:1–12, 2008.

[47] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008.

[48] Abdelatif Ennaji, Driss Mammass, Mostafa El Yassa, et al. Self-training using a k-nearest neighbor as a base classifier reinforced by support vector machines. *International Journal of Computer Applications*, 975:8887, 2012.

[49] Tongtong Fang, Nan Lu, Gang Niu, and Masashi Sugiyama. Rethinking importance weighting for deep learning under distribution shift. In *Neural Information Processing Systems*, volume 33, pages 11996–12007, 2020.

[50] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[51] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[52] Andres Folleco, Taghi M. Khoshgoftaar, Jason Van Hulse, and Lofton Bullard. Identifying learners robust to low quality data. In *2008 IEEE International Conference on Information Reuse and Integration*, pages 190–195, 2008.

[53] James Richard Foulds and Eibe Frank. A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, 2010.

[54] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning*, pages 1568–1577. PMLR, 2018.

[55] Benoit Frenay and Michel Verleysen. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 1994.

[56] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.

[57] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[58] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[59] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29:1189–1232, 2001.

[60] Gabriel Pui Cheong Fung, Jeffrey X. Yu, Hongjun Lu, and Philip S. Yu. Text classification without negative examples revisit. *IEEE Trans. on Knowl. and Data Eng.*, 18(1):6–20, 2006.

[61] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), 2014.

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

[63] Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. Robust loss functions under label noise for deep neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), 2017.

[64] Aritra Ghosh and Andrew Lan. Contrastive learning improves model robustness under label noise. *arXiv:2104.08984 [cs.LG]*, 2021.

[65] Arthur Gretton, Alex Smola, Jiayuan Huang, Marcel Schmittfull, Karsten Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 3(4):5, 2009.

[66] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, koray kavukcuoglu, Remi Munos, and Michal Valko. Bootstrap your own latent - a new approach to self-supervised learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21271–21284, 2020.

[67] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022.

[68] Xian-Jin Gui, Wei Wang, and Zhang-Hao Tian. Towards understanding deep learning from noisy labels with small-loss criterion. In *International Joint Conference on Artificial Intelligence*, 2021.

[69] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[70] Isabelle Guyon. Datasets of the active learning challenge. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2010.

[71] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. page 11, 2018.

[72] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[73] Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360, 2009.

[74] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference and prediction*. Springer, 2 edition, 2009.

[75] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[76] Ryuichiro Hataya and Hideki Nakayama. Unifying semi-supervised and robust learning by mixup. In *The 2nd Learning from Limited Labeled Data Workshop, ICLR*, 2019.

[77] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[78] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dÁlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15663–15674. Curran Associates, Inc., 2019.

[79] Dan Hendrycks, Mantas Mazeika, Duncan Wilson, and Kevin Gimpel. Using trusted data to train deep networks on labels corrupted by severe noise. In *Advances in Neural Information Processing Systems*, volume 31, pages 10456–10465, 2018.

[80] Ray J. Hickey. Noise modelling and evaluating learning from examples. *Artificial Intelligence*, 82(1-2):157–179, 1996.

[81] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[82] Wei-Ning Hsu and Hsuan-Tien Lin. Active Learning by Learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2659–2665. AAAI Press, 2015.

[83] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007.

[84] Eyke Hüllermeier and Jürgen Beringer. Learning from ambiguously labeled examples. In A. Fazel Famili, Joost N. Kok, José M. Peña, Arno Siebes, and Ad Feelders, editors, *Advances in Intelligent Data Analysis VI*, pages 168–179. Springer Berlin Heidelberg, 2005.

[85] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods, 2019.

[86] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2021.

[87] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. Self-paced curriculum learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[88] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International Conference on Machine Learning*, volume 80, pages 2304–2313, 2018.

[89] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *ICML*, 2018.

[90] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[91] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[92] Shehroz S Khan and Michael G Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014.

[93] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[94] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Learning Active Learning from Data. In *Advances in Neural Information Processing Systems 30*, pages 4225–4235. 2017.

[95] Ksenia Konyushkova, Raphael Sznitman, and Pascal Fua. Discovering General-Purpose Active Learning Strategies. *arXiv:1810.04114 [cs.LG]*, 2019.

[96] Mark A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE Journal*, 37(2):233–243, February 1991.

[97] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. *arXiv preprint arXiv:2205.02302*, 2022.

[98] M. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, volume 23, 2010.

[99] Hugo Le Baher, Vincent Lemaire, and Romain Trinquart. On the intrinsic robustness of some leading classifiers and symetric loss function - an empiricalevaluation (under review). *arXiv:2010.13570 [cs.LG]*, 2020.

[100] Jisoo Lee and Sae-Young Chung. Robust training with ensemble consensus. In *International Conference on Learning Representations*, 2020.

[101] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5447–5456, 2018.

[102] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. In *International Conference on Learning Representations*, 2019.

[103] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 4313–4324, 2020.

[104] Wen Li, Limin Wang, Wei Li, Eirikur Agustsson, and Luc Van Gool. Webvision database: Visual learning and understanding from web data. *arXiv preprint arXiv:1708.02862*, 2017.

[105] Yu-Feng Li, Lan-Zhe Guo, and Zhi-Hua Zhou. Towards safe weakly supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):334–346, 2019.

[106] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[107] Or Litany and Daniel Freedman. Soseleto: A unified approach to transfer learning and training with noisy labels, 2018.

[108] Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

[109] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *Third IEEE international conference on data mining*, pages 179–186. IEEE, 2003.

[110] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[111] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on pattern analysis and machine intelligence*, 38(3):447–461, 2015.

[112] Tongliang Liu and Dacheng Tao. Classification with noisy labels by importance reweighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(3):447–461, Mar 2016.

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

[114] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

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

[116] Hartmut Maennel, Ibrahim M. Alabdulmohsin, Ilya O. Tolstikhin, Robert J. N. Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? In *Neural Information Processing Systems*, 2020.

[117] Andrea Malossini, Enrico Blanzieri, and Raymond T. Ng. Detecting potential labeling errors in microarrays by data perturbation. *Bioinformatics*, 22(17):2114–2121, 2006.

[118] N. Matic, I. Guyon, L. Bottou, J. Denker, and V. Vapnik. Computer aided cleaning of large databases for character recognition. In *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol.II. Conference B: Pattern Recognition Methodology and Systems*, pages 330–333, August 1992.

[119] Yun-Qian Miao, Ahmed K Farahat, and Mohamed S Kamel. Ensemble kernel mean matching. In *2015 IEEE International Conference on Data Mining*, pages 330–338. IEEE, 2015.

[120] R. Mihalcea. Co-training and self-training for word sense disambiguation. In *CoNLL*, 2004.

[121] André L. B. Miranda, Luís Paulo F. Garcia, André C. P. L. F. Carvalho, and Ana C. Lorena. Use of Classification Algorithms in Noise Detection and Elimination. In *Hybrid Artificial Intelligence Systems*, Lecture Notes in Computer Science, pages 417–424, 2009.

[122] Jose G Moreno-Torres, Troy Raeder, Rocío Alaiz-Rodríguez, Nitesh V Chawla, and Francisco Herrera. A unifying view on dataset shift in classification. *Pattern recognition*, 45(1):521–530, 2012.

[123] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Cost-sensitive learning with noisy labels. *J. Mach. Learn. Res.*, 18(1):5666–5698, 2017.

[124] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013.

[125] Peter Nemenyi. Distribution-free multiple comparisons. *Biometrics*, 18(2):263, 1962.

[126] David F. Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial Intelligence Review*, 33(4):275–306, 2010.

[127] Vincent Ng and Claire Cardie. Weakly supervised natural language learning without redundant views. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 173–180, 2003.

[128] Nam Nguyen and Rich Caruana. Classification with partial labels. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–559, 2008.

[129] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632, 2005.

[130] Kamal Nigam and Rayid Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 86–93, 2000.

[131] Otton Nikodym. Sur une généralisation des intégrales de m. j. radon. *Fundamenta Mathematicae*, 15(1):131–179, 1930.

[132] Pierre Nodet, Vincent Lemaire, Alexis Bondu, and Antoine Cornuéjols. Contrastive representations for label noise require fine-tuning. In Georg Krempl, Vincent Lemaire, Daniel Kottke, Andreas Holzinger, and Barbara Hammer, editors, *Proceedings of the ECML Workshop on Interactive Adaptive Learning (IAL@ECML PKDD 2021)*, number 3079 in CEUR Workshop Proceedings, pages 89–104, Aachen, 2021.

[133] Pierre Nodet, Vincent Lemaire, Alexis Bondu, Antoine Cornuéjols, and Adam Ouorou. From Weakly Supervised Learning to Biquality Learning: an Introduction. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[134] Pierre Nodet, Vincent Lemaire, Alexis Bondu, Antoine Cornuejols, and Adam Ouorou. Importance reweighting for biquality learning. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[135] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[136] Kunkun Pang, Mingzhi Dong, Yang Wu, and Timothy M. Hospedales. Dynamic Ensemble Active Learning: A Non-Stationary Bandit with Expert

Advice. In *Proceedings of the 24th International Conference on Pattern Recognition*, pages 2269–2276, 2018.

[137] Kunkun Pang, Mingzhi Dong, Yang Wu, and Timothy M. Hospedales. Meta-Learning Transferable Active Learning Policies by Deep Reinforcement Learning. *arXiv:1806.04798 [cs.LG]*, 2018.

[138] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. 2019.

[139] Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. Making deep neural networks robust to label noise: a loss correction approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[140] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12:2825–2830, 2011.

[141] Davi Pereira-Santos and André C.P.L.F. de Carvalho. Comparison of Active Learning Strategies and Proposal of a Multiclass Hypothesis Space Search. In *Proceedings of the 9th International Conference on Hybrid Artificial Intelligence Systems – Volume 8480*, pages 618–629. Springer-Verlag, 2014.

[142] Davi Pereira-Santos, Ricardo Bastos Cavalcante Prudêncio, and André C.P.L.F. de Carvalho. Empirical investigation of active learning strategies. *Neurocomputing*, 326–327:15–27, 2019.

[143] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[144] Rafael Poyiadzi, Daniel Bacaicoa-Barber, Jesus Cid-Sueiro, Miquel Perello-Nieto, Peter Flach, and Raul Santos-Rodriguez. The weak supervision landscape. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, pages 218–223. IEEE, 2022.

[145] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *Proceedings of the european conference on computer vision (eccv)*, pages 135–152, 2018.

[146] J. R. Quinlan. Induction of decision trees. 1(1):81–106.

[147] Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, 29(2):709–730, 2020.

[148] S. Reed, H. Lee, Dragomir Anguelov, Christian Szegedy, D. Erhan, and Andrew Rabinovich. Training deep neural networks on noisy labels with bootstrapping. *CoRR*, abs/1412.6596, 2015.

[149] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

[150] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International Conference on Machine Learning*, volume 80, pages 4334–4343, 2018.

[151] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.

[152] Sebastian Ruder and Barbara Plank. Strong baselines for neural semi-supervised learning under domain shift. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1044–1054, July 2018.

[153] Walter Rudin. Analyse réelle et complexe. 1975.

[154] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997, 2017.

[155] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P Dickerson, and Tom Goldstein. Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In *International Conference on Machine Learning*, pages 9389–9398. PMLR, 2021.

[156] Matthias Seeger. Learning with labeled and unlabeled data. Technical report, 2000.

[157] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[158] Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison, 2009.

[159] Yuan Shi and Fei Sha. Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In *ICML*, 2012.

[160] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[161] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019.

[162] Jun Shu, Qian Zhao, Zongben Xu, and Deyu Meng. Meta transition adaptation for robust deep learning with noisy labels. *arXiv:2006.05697*, 2020.

[163] Ankur Sinha, Pekka Malo, and Kalyanmoy Deb. A review on bilevel optimization: from classical to evolutionary approaches and applications. *IEEE Transactions on Evolutionary Computation*, 22(2):276–295, 2017.

[164] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *arXiv:2007.08199 [cs.LG]*, 2021.

[165] Heinrich von Stackelberg et al. Theory of the market economy. 1952.

[166] Masashi Sugiyama. Talk: Recent advances in weakly-supervised learning and reliable learning, 2019.

[167] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440, 2008.

[168] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density ratio estimation: A comprehensive review (statistical experiment and its related topics). 2010.

[169] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[170] Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir D. Bourdev, and R. Fergus. Training convolutional networks with noisy labels. *arXiv: Computer Vision and Pattern Recognition*, 2014.

[171] Haoliang Sun, Chenhui Guo, Qi Wei, Zhongyi Han, and Yilong Yin. Learning to rectify for robust learning with noisy labels. *Pattern Recognition*, 124:108467, 2022.

[172] Jiang-wen Sun, Feng-ying Zhao, Chong-jun Wang, and Shi-fu Chen. Identifying and Correcting Mislabeled Training Instances. In *Future Generation Communication and Networking (FGCN 2007)*, volume 1, pages 244–250, December 2007. ISSN: 2153-1463.

[173] Luís Torgo, Stan Matwin, Nathalie Japkowicz, Bartosz Krawczyk, Nuno Moniz, and Paula Branco. 2nd workshop on learning with imbalanced domains: Preface. In *Second International Workshop on Learning with Imbalanced Domains: Theory and Applications*, pages 1–7, 2018.

[174] Colin Troisemaine, Joachim Flocon-Cholet, Stéphane Gosselin, Sandrine Vaton, Alexandre Reiffers-Masson, and Vincent Lemaire. A method for discovering novel classes in tabular data. In *IEEE International Conference on Knowledge Graph (ICKG)*, 2022.

[175] Ruth Urner, Shai Ben David, and Ohad Shamir. Learning from weak teachers. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *Proceedings of Machine Learning Research*, pages 1252–1260, 2012.

[176] Jason Van Hulse and Taghi Khoshgoftaar. Knowledge Discovery from Imbalanced and Noisy Data. *Data & Knowledge Engineering*, 68(12):1513–1542, December 2009.

[177] Brendan van Rooyen, Aditya Menon, and Robert C Williamson. Learning with symmetric label noise: The importance of being unhinged. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Neural Information Processing Systems*, pages 10–18. 2015.

[178] Brendan Van Rooyen and Robert C Williamson. A theory of learning with corrupted labels. *J. Mach. Learn. Res.*, 18(1):8501–8550, 2017.

[179] Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. Openml: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15(2):49–60, jun 2014.

[180] VN Vapnik and A Ya Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264, 1971.

[181] Paroma Varma and Christopher Ré. Snuba: Automating weak supervision to label training data. In *International Conference on Very Large Data Bases*, volume 12, 2018.

[182] Kalyan Veeramachaneni, Ignacio Arnaldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. Ai2: Training a big data machine to defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity)*, pages 49–54, 2016.

[183] Jean-Philippe Vert, Koji Tsuda, and Bernhard Schölkopf. A primer on kernel methods. *Kernel methods in computational biology*, 47:35–70, 2004.

[184] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[185] Qian-Wei Wang, Yu-Feng Li, and Zhi-Hua Zhou. Partial label learning with unlabeled data. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3755–3761, 2019.

[186] Ruxin Wang, Tongliang Liu, and Dacheng Tao. Multiclass learning with partially corrupted labels. *IEEE transactions on neural networks and learning systems*, 29(6):2568–2580, 2017.

[187] Yisen Wang, Xingjun Ma, Zaiyi Chen, Yuan Luo, Jinfeng Yi, and James Bailey. Symmetric cross entropy for robust learning with noisy labels. In *IEEE/CVF International Conference on Computer Vision*, pages 322–330, 2019.

[188] Yulin Wang, Rui Huang, Gao Huang, Shiji Song, and Cheng Wu. Collaborative learning with corrupted labels. *Neural Networks*, 125:205–213, 2020.

[189] Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with noisy labels revisited: A study using real-world human annotations. In *International Conference on Learning Representations*, 2022.

[190] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):9, 2016.

[191] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[192] Frank Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

[193] Stephen J Wright. Continuous optimization (nonlinear and linear programming). *Foundations of Computer-Aided Process Design*, 1999.

[194] Pengcheng Wu and Thomas G Dietterich. Improving svm accuracy by training on auxiliary data sources. In *Proceedings of the twenty-first international conference on Machine learning*, page 110, 2004.

[195] Yichen Wu, Jun Shu, Qi Xie, Qian Zhao, and Deyu Meng. Learning to purify noisy labels via meta soft label corrector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10388–10396, 2021.

[196] Xiaobo Xia, T. Liu, N. Wang, B. Han, C. Gong, Gang Niu, and Masashi Sugiyama. Are anchor points really indispensable in label-noise learning? In *NeurIPS*, 2019.

[197] Xiaobo Xia, Tongliang Liu, Bo Han, Nannan Wang, Mingming Gong, Haifeng Liu, Gang Niu, Dacheng Tao, and Masashi Sugiyama. Part-dependent label noise: Towards instance-dependent label noise. *Advances in Neural Information Processing Systems*, 33:7597–7610, 2020.

[198] Chaowei Xiao, Bo Li, Jun Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, pages 3905–3911. International Joint Conferences on Artificial Intelligence, 2018.

[199] Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. Learning from massive noisy labeled data for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2691–2699, 2015.

[200] Jun Yang. Review of multi-instance learning and its applications. *Technical report, School of Computer Science Carnegie Mellon University*, 2005.

[201] Shuo Yang, Erkun Yang, Bo Han, Yang Liu, Min Xu, Gang Niu, and Tongliang Liu. Estimating instance-dependent bayes-label transition matrix using a deep neural network. In *International Conference on Machine Learning*, pages 25302–25312. PMLR, 2022.

[202] Yazhou Yang and Marco Loog. A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, 83:401–415, 2018.

[203] Yu Yao, Tongliang Liu, Bo Han, Mingming Gong, Jiankang Deng, Gang Niu, and Masashi Sugiyama. Dual t: Reducing estimation error for transition matrix in label-noise learning. *Advances in neural information processing systems*, 33:7260–7271, 2020.

[204] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[205] Yinyu Ye and Edison Tse. An extension of karmarkar's projective algorithm for convex quadratic programming. *Mathematical programming*, 44(1):157–179, 1989.

[206] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[207] Man-Ching Yuen, Irwin King, and Kwong-Sak Leung. A survey of crowdsourcing systems. In *2011 IEEE third international conference on privacy, security, risk and trust and 2011 IEEE third international conference on social computing*, pages 766–773. IEEE, 2011.

[208] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, page 609–616, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[209] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

[210] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization, 2018.

[211] Hui Zhang and Quanming Yao. Decoupling representation and classifier for noisy label learning. *arXiv:2011.08145*, 2020.

[212] Min-Ling Zhang and Fei Yu. Solving the partial label learning problem: An instance-based approach. In *IJCAI*, pages 4048–4054, 2015.

[213] Mingyuan Zhang, Jane Lee, and Shivani Agarwal. Learning from noisy labels with no change to the training process. In *International Conference on Machine Learning*, pages 12468–12478. PMLR, 2021.

[214] Zhilu Zhang and Mert Sabuncu. Generalized cross entropy loss for training deep neural networks with noisy labels. In *Neural Information Processing Systems*, volume 31, 2018.

[215] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017.

[216] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations*, 2018.

[217] Evgenii Zheltonozhskii, Chaim Baskin, Avi Mendelson, Alex M. Bronstein, and Or Litany. Contrast to divide: Self-supervised pre-training for learning with noisy labels. *arXiv:2103.13646 [cs.LG]*, 2021.

[218] Guoqing Zheng, Ahmed Hassan Awadallah, and Susan Dumais. Meta label correction for noisy label learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35, 2021.

[219] Zhi-Hua Zhou and Ming Li. Tri-training: exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1529–1541, 2005.

[220] Zhun Zhong, Enrico Fini, Subhankar Roy, Zhiming Luo, Elisa Ricci, and Nicu Sebe. Neighborhood contrastive learning for novel class discovery. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10867–10875, 2021.

[221] Y. Zhou and S. A. Goldman. Democratic co-learning. *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 594–602, 2004.

[222] Yuyin Zhou, Xianhang Li, Fengze Liu, Xuxi Chen, Lequan Yu, Cihang Xie, Matthew P Lungren, and Lei Xing. Learning to bootstrap for combating label noise. *arXiv preprint arXiv:2202.04291*, 2022.

[223] Zhi-Hua Zhou. Multi-instance learning from supervised view. *Journal of Computer Science and Technology*, 21(5):800–809, 2006.

[224] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 08 2017.

[225] Zhi-Hua Zhou and Ming Li. Semi-supervised learning by disagreement. *Knowledge and Information Systems*, 24(3):415–439, 2010.

[226] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[227] Xingquan Zhu and Xindong Wu. Class noise vs. attribute noise: A quantitative study of their impacts. *Artif. Intell. Rev.*, 22(3):177–210, November 2004.

[228] Zhaowei Zhu, Yiwen Song, and Yang Liu. Clusterability as an alternative to anchor points when learning with noisy labels. In *International Conference on Machine Learning*, pages 12912–12923. PMLR, 2021.