Titre Français:

Adaptation de domaine agnostique au modèle : application à la détection de fraude

English Title:

Model Agnostic Domain Adaptation: Application to Fraud Detection

Thèse préparée et soutenue publiquement par **Luxin Zhang** le **07/03/2022**, pour obtenir le grade de Docteur en **Mathématiques et leurs interactions**

Collaboration: Worldline - Inria
Université: Université de Lille
Ecole doctorale: Ecole doctorale MADIS

Directeur de thèse: Christophe Biernacki
Co-encadrants: Pascal Germain, Yacine Kessaci

Membres du jury

Rapporteurs:	Jean-Michel Loubès	Professeur à l'Université Toulouse Paul Sabatier
	Massih-Reza Amini	Professeur à l'Université Grenoble Alpes
Examinateurs:	Elisa Fromont (présidente du jury)	Professeur à l'Université de Rennes 1
	Emilie Morvant	Maître de conférences à l'Université Jean Monnet de Saint-Etienne
Encadrants:	Christophe Biernacki	Professeur à l'Université de Lille
	Pascal Germain	Professeur adjoint à l'Université Laval
	Yacine Kessaci	Docteur chargé de recherche à Worldline

Abstract

Domain adaptation aims to alleviate the gap between different data distributions, commonly referred to as the source distribution and target distribution. Most of the related works seek either a latent space where source and target data share the same distribution or a transformation of the source distribution to match the target one. This thesis studies a realistic domain adaptation setting where one has access to an already existing "black-box" machine learning model. Indeed, in a industrial scenario like for Worldline, an efficient pre-trained source domain predictive model is often available and should be preserved. To leverage such a pre-trained model, we propose a *model-agnostic* adaptation function leveraging the optimal transport theory. Besides, the proposed solution has the asset to provide an interpretable *target to source* transformation, by seeking a sparse and ordered *coordinate-wise* adaptation of the feature space, in addition to elementary mapping functions.

To automatically select the subset of features to be adapted, we first introduce a weakly supervised process relying on scarce labeled target data. Then, we address a more challenging unsupervised version of this domain adaptation scenario. To this end, we propose a new pseudo-label estimator over unlabeled target examples based on rank-stability regarding the source model prediction. Such estimated "labels" are further used in a feature selection process to assess whether each feature needs to be transformed to achieve adaptation. We provide theoretical foundations of our method as well as an efficient implementation.

Furthermore, we extend such weakly supervised and unsupervised adaptation methods to a multi-subdomain adaptation case. Precisely, we consider source and target domain data can be subdivided into data from different distributions. To exploit such hidden subdomains, we propose to use an inter-subdomain divergence maximization criterion. Moreover, we leverage a subdomain aggregation method to get the final predictions. Experimental results over two fraud detection datasets and the *Amazon reviews* sentiment analysis benchmark demonstrate the efficiency of our method.

Résumé

L'adaptation de domaine vise à réduire l'écart entre des données de distributions différentes, communément appelées distribution source et distribution cible. La plupart des travaux associés cherchent soit un espace latent où les données sources et cibles partagent la même distribution, soit une transformation de la distribution source pour qu'elle corresponde à la distribution cible. Cette thèse étudie un cas réaliste d'adaptation de domaine dans le contexte industriel de Worldline où un modèle d'apprentissage automatique est déjà existant. En effet, un modèle prédictif pré-entraîné de domaine source est souvent disponible et doit être préservé. Pour bénéficier d'un tel modèle pré-entraîné, nous proposons une fonction d'adaptation *agnostique du modèle* qui s'appuie sur la théorie du transport optimal. En outre, la solution proposée présente l'avantage de fournir une transformation interprétable de la *cible à la source*, en recherchant une adaptation *ordonnée et parcimonieuse* de l'espace des caractéristiques, en plus des fonctions d'adaptation élémentaires.

Pour sélectionner automatiquement le sous-ensemble de caractéristiques à adapter, nous introduisons d'abord un processus faiblement supervisé reposant sur des données cibles rarement étiquetées. Ensuite, nous abordons une version non supervisée, plus complexe, de ce scénario d'adaptation de domaine. À cette fin, nous proposons un nouvel estimateur de pseudo-étiquettes sur des exemples cibles non étiquetés, basé sur la stabilité du rang dans la prédiction du modèle source. Ces "étiquettes" estimées sont ensuite utilisées dans un processus de sélection de caractéristiques pour évaluer si chaque caractéristique doit être transformée pour réaliser l'adaptation. Nous présentons les fondements théoriques de notre méthode ainsi qu'une mise en œuvre efficace.

En outre, nous étendons ces méthodes d'adaptation faiblement supervisées et non supervisées à un cas d'adaptation multi-sous-domaine. Plus précisément, nous considérons que les données des domaines sources et cibles peuvent être subdivisées en données de différentes distributions. Pour exploiter ces sous-domaines cachés, nous proposons d'utiliser un critère de maximisation de la divergence inter-sous-domaines. De plus, nous proposons une méthode d'agrégation des sous-domaines pour obtenir les prédictions finales. Les résultats expérimentaux sur deux bases des données de détection de fraude et une base de données de référence d'analyse de sentiments *Amazon reviews* démontrent l'efficacité de notre méthode.

Acknowledgement

I would like to thank all my teachers, colleagues, and family members who have been with me during my Ph.D.

First of all, my gratitude goes to my supervisors, Prof. Christophe Biernacki, Dr. Pascal Germain, and Dr. Yacine Kessaci. Thank you for providing me with an excellent platform, helping me determine research directions, and guiding my Ph.D. project. Thanks to Yacine Kessaci for helping me to coordinate my collaboration with the company. Thanks to Christophe Biernacki for welcoming me to the Modal team. Thanks to Pascal Germain for his patient guidance. I am also grateful to my colleagues sharing the office with me, Florent Dewez, Vera Shalaeva, Issam Ali Moindjie, and Guillaume Braun. Thank you for the interesting topics and discussions that you bring in the quotidian work.

Of course, a Ph.D. is not only the result of hard work: this achievement also depends on the support and love of the people close to me. I give my deepest gratitude to my parents and grandparents, who always stood by me. Their companionship helped me get through the long days of confinement.

Finally, I would like to thank the thesis referees and other jury members for evaluating my work and proposing interesting discussions.

Luxin Zhang

List of Publications

1. Zhang, L., Germain, P., Kessaci, Y., & Biernacki, C. (2020, September). Target to Source Coordinate-wise Adaptation of Pre-trained Models. In ECML PKDD 2020-The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases.

2. Zhang, L., Germain, P., Kessaci, Y., & Biernacki, C. (2021). Interpretable Domain Adaptation Using Unsupervised Feature Selection on Pre-trained Source Models. (Under Review)

3. Zhang, L., Germain, P., Kessaci, Y., & Biernacki, C. (2022). Interpretable Domain Adaptation for Hidden Subdomain Alignment in the Context of Pre-trained Source Models. In AAAI 2022-Thirty-Sixth AAAI Conference on Artificial Intelligence.

Contents

List of Notations

$\mathcal{X}$	The input space
$\mathcal{X}_{\text{sub}}$	The input space that encodes hidden subdomains
$\mathcal{X}_d$	The d-th dimension of the input space
$\mathcal{Y}$	The output space
X	The input variable
X_i^t, X_j^s	The marginal variables of target and source subdomains
Y	The output variable
$\boldsymbol{x}$	An input vector
x^d	The d-th dimension of an input vector
$\boldsymbol{x}_{\text{adapt}}$	An adapted input vector
x^d_{adapt}	The d-th dimension of an adapted input vector
y	An output scalar
$\boldsymbol{X}$	An input matrix
$\tilde{\boldsymbol{X}}$	A corrupted input matrix
$\mathcal{D}$	An input subspace
$\mathbb{D}$	A set of input subspaces
$\mathcal{D}^*$	The optimal input subspace of supervised coordinate-wise adaptation
$\hat{\mathcal{D}}^*$	The optimal input subspace of weakly supervised coordinate-wise adaptation
$\mathcal{H}$	The reproducing kernel Hilbert space
$\mathbb{R}$	The real value space
μ_t	A discrete target domain input distribution
μ_s	A discrete source domain input distribution
μ_t^d	The d-th dimension of a discrete target input distribution
μ_s^d	The d-th dimension of a discrete source input distribution

n_t	The number of all target examples
n_i^t	The number of target subdomain examples
n_s	The number of all source examples
n_j^s	The number of source subdomain examples
n_t^l	The number of labeled target examples
n_t^u	The number of unlabeled target examples
m_l^h	The number of hidden neuros of the l-th layer
m_d^c	The number of levels in the d-th categorical feature
n_r	The number of repetitions of stochastic transformations
k	The number of clusters
k_t	The number of target subdomains
k_s	The number of source subdomains
$\boldsymbol{K}$	The couple of (k_t, k_s)

$\boldsymbol{R}$	A joint probability matrix
$\boldsymbol{R}^d$	A joint probability matrix of the d-th dimension
$R_{i,j}$	The i-th row and the j-th column of $\boldsymbol{R}$
$R_{i,j}^d$	The i-th row and the j-th column of $\boldsymbol{R}^d$
Γ	A set of admissible $\boldsymbol{R}$
Γ^d	A set of admissible $\boldsymbol{R}^d$
$\boldsymbol{C}$	A cost matrix
$\boldsymbol{S}$	A mapping matrix that maps hidden target subdomains to the source ones
$\boldsymbol{A}$	An aggregation factor to reweight different numbers of subdomains

$\mathbb{Q}$	A set of input-output pairs in supervised learning
$\mathbb{Q}^s$	A set of source domain input-output pairs
$\mathbb{Q}_l^t$	A set of target domain input-output pairs
$\mathbb{X}^t$	A set of all target domain inputs
$\mathbb{X}_u^t$	A set of unlabeled target domain inputs
$\mathbb{X}^s$	A set of all source domain inputs
$\mathbb{X}_i$	A set of input examples of the i-th cluster
$\mathbb{X}_{\text{sub}}$	A set of all source and target subdomain sets of examples
$\mathbb{X}_{\text{stab}}^t$	A set of stable target domain inputs
$\mathbb{X}_{\delta-\text{stab}}^t$	A set of δ-stable target domain inputs
$\mathbb{E}_d$	A set of categorical levels of the d-th feature

$\mathcal{T}(\cdot)$	A source to target transformation function
$\mathcal{G}(\cdot)$	A target to source transformation function
$\mathcal{G}^*(\cdot)$	An optimal oracle target to source transformation function
$\mathcal{G}^{\mathcal{D}}(\cdot)$	A coordinate-wise transformation on a subset of features $\mathcal{D}$
$\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot)$	The optimal coordinate-wise single-domain adaptation function that transforms data from the i-th target subdomain $\mathbb{X}_i^t$ to the j-th source subdomain $\mathbb{X}_j^s$
$h(\cdot)$	A predictive model
$h_s(\cdot)$	The source domain optimal predictive model
$h_t(\cdot)$	The target domain optimal predictive model
$h_t^{\mathcal{D}}(\cdot)$	A shorthand of $h_s \circ \mathcal{G}^{\mathcal{D}}(\cdot)$
$\hat{h}^{\mathcal{D}}(\cdot)$	A pseudo-label estimator
$h_t^i(\cdot)$	A predictor of the i-th adapted subdomain of target
$h_t^\dagger(\cdot; \boldsymbol{K})$	The optimal target domain classifier with a known number of subdomains
$h_t^*(\cdot; \boldsymbol{A})$	The optimal target domain classifier with unknown number of subdomains
p_0, p_1	Propositions of negative and positive examples
$\boldsymbol{w}, b$	Parameters of a linear transformation. $\boldsymbol{w}$ is a vector and b is a scalar
$\boldsymbol{h}_l$	The output vector of the l-th layer of a neural network
$\boldsymbol{c}_i$	The center of a cluster
y_{thres}	A threshold separating positive and negative examples
$\delta_\cdot$	The Dirac function
a_j^s	The weight of a source domain input
a_i^t	The weight of a target domain input
η_{reg}	The weight of the entropy regularization term in the optimal transport
η_{group}	The weight of the group regularization term in the optimal transport
$d^{\max}$	The maximum number of input space dimensions
$e_\cdot^d$	A categorical level of the d-th feature
$v_\cdot^d$	The global frequency of $e_\cdot^d$ in all domains
$v_\cdot^{s,d}, v_\cdot^{t,d}$	Frequencies of $e_\cdot^d$ in each domain
δ	A tolerance of stable inputs

$\mathrm{tr}(\cdot)$	The function that computes the trace of a matrix
$\mathrm{mean}(\cdot)$	The function that computes the average value of a variable
$\mathrm{var}(\cdot)$	The function that computes the variance of a variable
$\min(\cdot)$	The function that computes the minimum value of a set
$\max(\cdot)$	The function that computes the maximum value of a set
$\mathrm{rank}(\cdot)$	The function that gives the ascending order of one example
$\mathbf{1}(\mathrm{cond})$	An indicator function that gives 1 if "cond" is true

$l(\cdot,\cdot)$	A generic loss function
$d_{\mathcal{H}}(\cdot,\cdot)$	The $\mathcal{H}$-divergence function
$e(\cdot)$	The target domain risk between a pseudo-label predictor and the optimal one
$d_{\mathrm{w}}(\cdot,\cdot)$	The sum of one-dimensional Wasserstein distances over each feature
$d_{\text{1d-w}}(\cdot,\cdot)$	The one-dimensional Wasserstein distance
$r_t^l(\cdot)$	The target domain risk
$r_s^l(\cdot)$	The source domain risk
$\Omega(\cdot)$	A regularization term
$l_{\mathrm{reg}}(\cdot)$	An entropy regularization term in the optimal transport
$l_{\mathrm{group}}(\cdot)$	A group regularization term in the optimal transport
$l_{\mathrm{stab}}(\cdot)$	The maximum bias of pseudo-labels of δ-stable examples
$l_{\mathrm{bias}}(\cdot,\cdot,\delta)$	The expected bias of two pseudo-label estimators over δ-stable examples

$\sigma(\cdot)$	The non-linear activation function
$\phi(\cdot)$	The mapping of function an example to a RKHS
$k(\cdot,\cdot)$	A kernel function
$<\cdot,\cdot>$	A inner product between two vectors
$c(\cdot,\cdot)$	A distance measure
$c^p_{\mathrm{num}}(\cdot,\cdot)$	The l^p norm of numerical features
$f_{\mu_s^d}(\cdot)\,, f_{\mu_t^d}(\cdot)$	Cumulative distribution functions of μ_s^d and μ_t^d
$f_{h_s}(\cdot), f_{h_t^{\mathcal{D}}}(\cdot), f_{h_t}(\cdot)$	Cumulative distribution functions of $h_s(X^s)$, $h_t^{\mathcal{D}}(X^t)$ and $h_t(X^t)$.
$p_i(\cdot)$	The probability that an example belongs to a subdomain

List of Figures

List of Tables

Chapter 1

Introduction

This chapter provides the motivations of this thesis. It starts with a Worldline real-life problem of adapting payment fraud detection systems to different geographical areas. Then it reframes this problem as a general adaptation challenge of machine learning models. We summarize our major contributions to this adaptation challenge and present the organizations of this manuscript.

1.1 From a Worldline Industrial Problem to an Academic Challenge

As a European leader and a major global player in the payment and transactional services sector, Worldline provides next-generation transactional services to its clients. The services can have different forms, from payment transactions in banks to the exchange of information between connected objects. However, the payment transactions remain to play a more important role in Worldline's core business model. Generally, Worldline operates two types of payment transactions: the transaction of payment card through a merchant terminal, the so-called face-to-face, and the online transaction on the Internet, the so-called e-commerce. The purpose of payment transactions is to exchange money. By nature, they are subject to fraud attempts that can take different forms. Worldline employees have experimented with various fraud detection models in order to automatically detect fraudulent transactions, ranging from expert systems to advanced machine learning methods.

Expert-rule-based methods basically build "rules" relying on the experiences of business experts for classifications [155]. A typical structure of rules is an if-then expression. For example, an artificial rule can be "if the transaction amount is over 2,000 euros, then it is fraudulent". Such a model is easy to interpret and

apply. However, it relies highly on business expertise and is not flexible to address all cases.

Machine learning has been proved to be a powerful tool in various real-life applications. Among all machine learning paradigms, supervised learning is one of the most widely used approaches. The principal objective of supervised learning is to leverage labeled information, the so-called training data, to discover the underlying patterns that can be generalized to infer labels of unseen data, the so-called testing data. If one wants to have a good generalization performance, abundant data should be annotated precisely, and testing data should obey the same distribution as the training one. However, in real-life applications, manual annotation is costly, and it is common to have testing data drifts from the training one.

A particular case that violates the two conditions of supervised learning is the payment fraud detection system. There is a drift between training and testing data whenever customer payment habits change. A concrete example is the drift in geographical locations. As Worldline operates payment transactions across plenty of areas (*e.g., countries*), due to various payment habits of customers, transactions of one area are more or less different from the other. Therefore, applying a fraud detection model trained in one area is suboptimal to all other areas (*e.g.,* a fraud detection system trained with data from Belgium cannot be directly applied to the Indian market). Moreover, Worldline often tackles data from new areas that have not enough or even missing labels for supervised learning when expanding the business. Besides drifts in geographical locations, payment transactions in one area also continually shift due to the change of seasons. All such constraints limit the application of classical supervised learning in cross-areas payment fraud detection problems.

Since achieving an efficient fraud detection model is tedious and costly, Worldline aims at capitalizing on the experience gained in its already invested markets. They expect to develop an approach to reuse these pre-trained models in new markets. The idea of reusing pre-trained models is appealing, as it can avoid a tedious process of model retraining. The fraud detection model in real-life applications often aggregates predictive models from different families, such as decision trees and neural networks. All these elementary models are precisely developed by machine learning researchers in the company and involve business experts' cooperations. Retraining such an aggregation of models for every new market can be time-consuming and is unfeasible at the early stage of business expansion. Therefore, we should propose a predictive *model-agnostic* adaptation method that transforms data of new markets to be close to the ones of existing markets. To the best of our knowledge, there was no existing framework that

fits these requirements. Moreover, Worldline is also concerned with data privacy and transparency. The **G**eneral **D**ata **P**rotection **R**egulation (GDPR) has been applied since 2018 to protect personal private data from abuse. In that sense, the interpretability of machine learning models is becoming increasingly important. From a business point of view, an easily interpretable approach can potentially contribute to exploring new markets and help machine learning researchers and business experts to improve the existing pre-trained model. Hence, the proposed adaptation method should be easily interpreted even by practitioners without specific machine learning backgrounds.

Besides such business requirements, fraud detection tasks also have characteristics that complicate the adaptation. First, frauds are rare events that seldom occur. The proposed adaptation method should be able to capture variations of fraudsters' behaviors in different markets and not be confused by their camouflage. Then, the number of transactions is generally large (*e.g.,* Worldline operates more than 80% of card transactions of Belgium). The adaptation methods should be scalable to address the big data adaptation problem. Thirdly, payment transactions contain discrete categorical values. The adaptation methods should be able to transform numerical values as well as categorical ones.

1.2 Adaptation Problem Setup

Although the initial motivation of this work relies on the Worldline fraud detection problem, it is a common machine learning challenge to reuse knowledge on unseen data. The idea of leveraging the knowledge of existing pre-trained models to unknown markets fits the paradigm of *transfer learning* [110]. Furthermore, this thesis tackles a specific case of transfer learning, the so-called *domain adaptation* (detailed discussions are given in Section 2.5). Following the taxonomies of the domain adaptation, existing markets are referred to as *source domains*, and new markets are *target domains*.

The domain adaptation problem commonly occurs in machine learning challenges when target domain data is different from the source ones. Based on the Worldline challenges introduced in the previous section, this manuscript provides solutions having the following assets:

1. *Model-agnostic.* Pre-trained/Legacy machine learning models can belong to various families (details in Section 2.3); thus, the domain adaptation methods should be generic enough to reuse them without retraining.

2. *Interpretable.* As the model interpretability is becoming increasingly important, the proposed method should be easily interpreted by practitioners or

without machine learning backgrounds.

3. *Feature-type free.* Data that one used in various tasks often encompass numerical and categorical feature types. Hence, the proposed solution is expected to tackle both of them.

These three assets meet Worldline's requirements and benefit various machine learning tasks, particularly when a pre-trained model is given and should be preserved.

1.3 Contributions & Organization of the Manuscript

Contributions In order to meet both the aforementioned business requirements and fraud detection problem characteristics, we will develop in the rest of the document the following three main contributions:

1. We propose a new *target to source* domain adaptation modeling and its associated weakly supervised domain adaptation resolution pipeline. The domain adaptation methods based on this new perspective is *model-agnostic*, *interpretable* and *feature-type free*.

2. We extend the proposed adaptation pipeline to an unsupervised case by introducing a new pseudo-labeling method with detailed theoretical studies. The solution can be applied to a wide range of settings, including unsupervised ones.

3. Based on the first and second contributions, we provide an intuitive approach to discover multiple underlying distributions in source and target domains to account for the change of seasons. This problem is known as hidden subdomain adaptation in domain adaptation literature.

Organization Chapter 2 reviews some basic notations and taxonomies of machine learning, domain adaptation, and optimal transport theory. In particular, it introduces different families of machine learning models that Worldline adopts in the existing markets. It gives definitions of domain adaptation and provides details of optimal transport theory applying to domain adaptation problems. Furthermore, it introduces the characteristics of the Worldline adaptation problem, compares our proposition with some well-known domain adaptation methods, and explains the reason why they are not suited to address the Worldline adaptation

problem. The last section of the chapter gives details about datasets that we used for experiments.

We propose a new *target to source* domain adaptation perspective in Chapter 3. We formulate this novel perspective as transforming target domain data to fit the source domain distribution and propose a concrete *coordinate-wise* adaptation function: ***C**oordinate-wise **D**omain **A**daptation* (**CDA**). Additionally, we observe that features contribute differently to domain adaptations. Consequently, we enhance the interpretability and improve the performance of CDA by applying a ***W**eakly supervised feature selection for **C**oordinate-wise **D**omain **A**daptation* (**WCDA**). Results of this chapter have been published in the paper *"Target to Source Coordinate-wise Adaptation of Pre-trained Models"* [167].

We further extend the weakly supervised feature selection process to an unsupervised case leveraging a new pseudo-label estimator in Chapter 4. We name our proposition: ***S**tability-based feature selection for **C**oordinate-wise **D**omain **A**daptation* (**SCDA**). Both theoretical and empirical studies have proved efficiency of the proposed pseudo-label estimator applying to domain adaptation tasks. Results of this chapter have been published in the paper *"Interpretable Domain Adaptation Using Unsupervised Feature Selection on Pre-trained Source Models"* [168].

Moreover, Chapter 5 aggregates proposed single domain adaptation methods to address a multi-subdomain adaptation scenario. We provide a general subdomain division criterion and an aggregation method and then specialize in a real-life case of temporal drift. We name our method: **H***idden* **S***ubdomain* **A***daptation with* **V***ariable Number of Subdomains* (**HSAV**). Results of this chapter have been published in the paper *"Interpretable Domain Adaptation for Hidden Subdomain Alignment in the Context of Pre-trained Source Models"* [169].

Chapter 2

Machine Learning and Fraud Detection Background

This chapter starts by presenting essential backgrounds and terminologies related to machine learning. Then, it gives an overview of supervised and unsupervised learning methods, followed by some well-known key machine learning challenges. It introduces domain adaptation methods according to various taxonomies, followed by a chapter addressing the optimal transport theory and its application to domain adaptation tasks. The last section of this chapter puts into perspective the presented background by addressing them according to Worldline's fraud detection specifications.

2.1 Introduction to Machine Learning

Human beings are known to be able to learn knowledge from past experiences to make reasonable guesses about the future. When it is cloudy and wet, we know there will be a high chance of rain, and we can also foresee a decrease in temperature. Machine learning is a discipline in computer science that aims to teach a machine, or a program, to mimic such a reasoning process without being explicitly programmed [132]. In computer science, experiences are generally represented by data. Hence the essence of machine learning research is to propose learning algorithms to extract generic patterns, the so-called *models*, from the data. Machines can use the extracted models to infer consequences when similar events occur in the future. Mitchell [100] gives a formal definition of machine learning by focusing on the improvements of performances of such inferences.

Definition 2.1 (Machine learning)**.** A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if

its performance at tasks in T, as measured by P, improves with experience E.

To place this definition into the industrial application studied in this manuscript, let us consider a fraud detection task. In such a problem, experiences E are payment transactions with labels indicating fraudulent or not. The task T aims to classify transactions into genuine ones and fraudulent ones. The performance measure P can be an accuracy (or any other performance measure) that reports the percentage of actual frauds among transactions classified as fraudulent. In the machine learning paradigm, a group of records of transactions is a *dataset*. Each transaction is called an *instance*, an *example*, or simply a *point*. Transactions contain *attributes* or *features*, such as currency, amount, etc. All attributes form a so-called *input space* or *attribute space*, with the number of attributes corresponding to such a space's dimensionality. All possible values of labels form a *label space* or *output space*. If an output space is discrete, such as the case of fraud detection, the machine learning task is a *classification* task; otherwise, it is called a *regression* task. The process to estimate a model from datasets is called *training* or *learning*. Datasets that are used during the training process are *training sets*. Every point inside is a *training example*. Once the model is estimated, the prediction process of unseen data is called *testing*, and unseen data are *testing data*.

2.2 Supervised and Unsupervised Learning

2.2.1 Supervised Generative and Discriminative Learning

When every input example is annotated by an output label, one stands in a *supervised learning* scenario. Let $X \in \mathcal{X}, Y \in \mathcal{Y}$ be respectively input and output variables, where $\mathcal{X}$ is an input space, and $\mathcal{Y}$ is an output space. More specifically, in a fraud detection task, one tackles a binary classification problem where $\mathcal{X}$ encompasses numerical and categorical dimensions while $\mathcal{Y} = \{0, 1\}$.

The training dataset is given by $\mathbb{Q} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$, where $\boldsymbol{x}_i \in \mathcal{X}$ is referred to as an input and $y_i \in \mathcal{Y}$ stands for an output. When $y_i = 1$, the example is called positive; otherwise, the example is negative. $\mathcal{X}$ is a multi-dimensional space, and we note $\boldsymbol{x}_i$ in bold to stand for a multi-dimensional vector.

The underlying characteristics of input-output pairs are represented by their joint distribution $P(X, Y)$. Supervised learning models that aim to directly modelize $P(X, Y)$ from given labeled data are known as *generative models*. Others that focus on the reasoning process aim to modelize a conditional distribution $P(Y|X)$ and are known as *discriminative models* [105]. Generative models can create new examples that are never seen. Gaussian mixture model and Hidden

Markov model [160, 122] are two of the most well-known ones. In contrast, discriminative models focus on predicting the labels of given inputs. Some widely used discriminative models are **S**upport **V**ector **M**achine (SVM) [27], logistic regression [13], and decision trees [119]. This manuscript gives more details about discriminative models, as the studied fraud detection task aims to predict fraudsters from given transactions instead of generating new ones from scratch. In this manuscript, the learned discriminative models are also called predictive models or classification models.

2.2.2 Supervised Objective Functions

We let $h(\cdot)$ be a classification model. In most machine learning algorithms, $h(\cdot) : \mathcal{X} \to [0, 1]$ does not solely give a discrete class label but a continuous value in $[0, 1]$ that reflects a probability being classified as positive ($P(Y|X)$). Then, outputs that surpass a given threshold y_{thres} are considered as positive examples. We define an indicator function $\mathbf{1}$(condition) that gives 1 if the "condition" is verified and 0 otherwise. The objective of supervised learning can be formulated as an optimization problem of estimating the optimal $h(\cdot)$ that minimizes a given loss function $l(\cdot, \cdot) : [0, 1] \to \mathbb{R}_+$, that is,

$$\underset{h}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} l(h(\boldsymbol{x}_i), y_i) \,.$$

For a binary classification task, two most commonly used loss functions are 0-1 loss:

$$l(h(\boldsymbol{x}_i), y_i) = \mathbf{1}\Big(\mathbf{1}(h(\boldsymbol{x}_i) > y_{\text{thres}}) \neq y_i\Big) \,,$$

that counts the number of false classifications, and the binary cross-entropy (log-loss):

$$l(h(\boldsymbol{x}_i), y_i) = y_i \log(h(\boldsymbol{x}_i)) + (1 - y_i) \log(1 - h(\boldsymbol{x}_i)) \,,$$

that computes the negative log-likelihood of outputs. From the optimization point of view, log-loss is continuous and derivable, which makes it easier to be minimized. Therefore, it is commonly used during the training process, whereas 0-1 loss is generally used as an evaluation metric.

2.2.2.1 Overfitting and Underfitting

If a model is too well-trained to minimize a training risk, whereas testing risks are high, we say the model is *overfitting*. An example is illustrated in Figure 2.1,

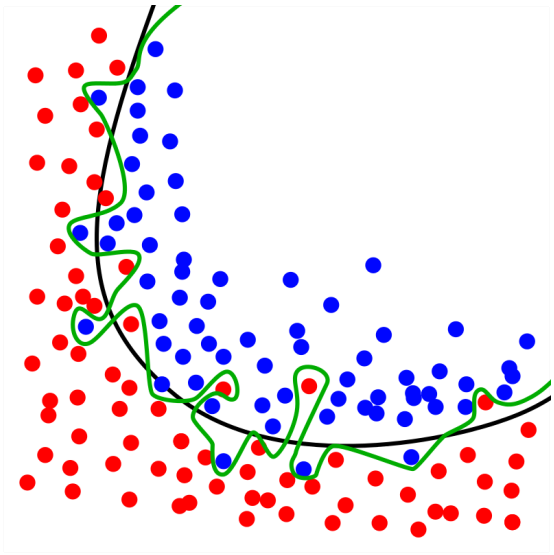


Figure 2.1: Red points and blue points represent training examples of two classes. The green curve is an overfitted model, while the black curve is a regularized one (Chabacano 2008).[1]

where the green curve represents an overfitted model. Typically, overfitting occurs if a predictive model's number of parameters is larger than the number of training examples. An overfitted model captures every minor variation in training datasets, including noises. When unseen data arrive, the overfitted model fails to correctly predict labels, as noises are completely arbitrary and unrelated to the class of labels.

To prevent overfitting, one can adopt an early stopping technique. Early stopping is a form of *regularization* applied by iterative optimization methods such as gradient descent. It stops the minimization process of the objective function when the potential gain is less than a defined threshold. Namely, a small subset of training examples is left out of the training dataset to help search the optimal threshold. Such a subset of examples is called a *validation dataset*. Moreover, one can also apply a cross-validation strategy [140] for such threshold searching.

If a model performs weakly on both training and testing datasets, we say the model is *underfitting*. It occurs when the trained model cannot correctly capture the underlying structure of a dataset. A particular example of underfitting is to apply a linear model on data that are not linearly separable. To prevent this issue, one can leverage more flexible machine learning models such as **G**radient **B**oosting **D**ecision **T**ree (GBDT) and neural networks (Details in Sections 2.3.1 and 2.3.2).

[1] `https://en.wikipedia.org/wiki/Overfitting`

2.2.3 Unsupervised Learning

While supervised learning leverages labeled data to estimate a model that predicts classes of unseen examples, unsupervised learning aims to get semantic patterns in a dataset without using labeled information [54]. In some machine learning paradigm, the inferred patterns are also known as representations and are further used by other learning algorithms (*e.g.,* supervised learning algorithms) to get more robust predictions. Although unsupervised learning reduces the workload of data annotating, it generally needs a huge amount of data to extract efficient patterns. Some famous applications of unsupervised learning include clustering, dimensionality reduction, association rules, etc. (Details in Section 2.3).

2.3 Some Supervised and Unsupervised Models

This section introduces some well-known supervised and unsupervised learning models. Among supervised methods, we provide details of decision-tree-based models and neural networks. These models are efficient when dealing with a huge amount of data and are well-suited for challenging machine learning tasks like fraud detection. Besides, decision trees and neural networks are the exact opposite extremes in terms of model interpretability. Decision trees can be reformed to sequences of simple rules and are easily interpretable. In contrast, neural networks combine complex mathematical transformations and are hardly interpretable. Among unsupervised methods, we detail two families of unsupervised learning: autoencoder and clustering, respectively. Such two methods are particularly important to us, as our propositions and experiments rely on some principal concepts of these methods.

To learn more about other machine learning methods, one can consult the work of Friedman [42].

2.3.1 Supervised Decision Tree

A decision tree can be seen as a flowchart-like structure. The prediction process of a data point starts from the root node. The point follows instructions in each node to reach a leaf node (a node without any child). The prediction of this point is the majority class of the leaf node.

Different algorithms like ID3 [119], C4.5 [120], and CART [14] are proposed to build such a decision tree. All of them seek to increase the purities of leaf nodes. ID3 and C4.5 rely on information entropy [135]: $-p_0 \log(p_0) - p_1 \log(p_1)$, while CART uses Gini index: $1 - p_0^2 - p_1^2$. p_0 and p_1 are respectively propositions of

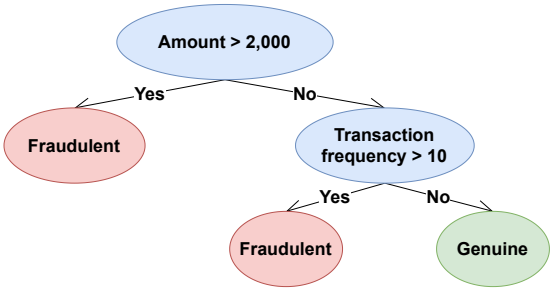


Figure 2.2: Illustration of a simple decision tree.

negative and positive examples in a leaf node. Note that a decision tree is easy to interpret: each top-down flow can be translated into a sequence of rules.

Decision trees adopt *pruning strategies* to avoid overfitting. Classical pruning methods include *pre-pruning* and *post-pruning* [120]. The former stops separating examples of a node during the construction of a tree if prediction performances on the validation dataset do not improve. While the latter removes branches from built trees if it improves validation performances.

2.3.1.1 Gradient Boosting Decision Tree

A variant of decision trees leverages ensemble learning [107, 118] and boosting [134, 118] methods and is known as **G**radient **B**oosting **D**ecision **T**ree (GBDT). Recent efficient implementations like XGBoost [23] and LightGBM [63] have simplified the application of GDBT to real-life industrial contexts.

Ensemble learning uses multiple predictive models to obtain more robust predictions [124]. A well-known example of ensemble models is the random forest, where one builds many decision trees over different partitions of data. The final predictions are obtained through a majority vote process; thus, the risk of overfitting decreases. However, random forest is computationally expensive as it aggregates fully developed trees, the so-called *strong learner*. A strong learner is a classifier that is well-correlated with ground truth labels. Hence, every single strong learner has good prediction performances over the datasets. In contrast, a *weak learner* is defined as a classifier that is just slightly better than a random guess. Nonetheless, weak learners are advantageous as they are much easier to train compared to strong ones. So a natural question is, "Can we aggregate weak learners to get a strong one?" [64]. Boosting methods give a favorable answer to this question [134].

Boosting can be considered as a particular case of ensemble learning where one aggregates multiple weak learners. More precisely, in GBDT, to control the size of aggregated trees, each decision tree is restrained by the maximum number of

depth, the maximum number of leaf nodes, and the minimum number of examples in each leaf node. Moreover, GBDT is an additive model; weak learners are trained to estimate residuals instead of final prediction results. Formally, at the t-th step of training, one looks for $h^{(t)}(\cdot)$ that minimizes the following formula:

$$\sum_{i=1}^{n}(y_i - (\hat{y}_i^{(t-1)} + h^{(t)}(\boldsymbol{x}_i)))^2 + \sum_{j=1}^{t}\Omega(h^{(j)})\,,$$

where

$$\hat{y}_i^{(t-1)} = \sum_{j=1}^{t-1} h^{(j)}(\boldsymbol{x}_i)\,,$$

and $\Omega(\cdot)$ is a regularization term that controls the complexity of the model. The final prediction is given by

$$h(\boldsymbol{x}_i) = \sum_{j=1}^{t_{\max}} h^{(j)}(\boldsymbol{x}_i)\,,$$

where $t_{\max}$ stands for the maximum number of iterations.

2.3.2 Supervised Neural Networks

Deep neural networks have achieved impressive results over a wide range of tasks like computer vision [71] and natural language processing [34]. It is shown to be able to extract representations that are robust and discriminative.

A basic neuron combines two components, a linear transformation parameterized by a vector $\boldsymbol{w}$ and a scaler b, and a non-linear activation function $\sigma(\cdot)$. It has the following formula:

$$\sigma(\boldsymbol{w}^T\boldsymbol{x}_i + b)\,,$$

where $\boldsymbol{w}^T$ refers to the transpose of $\boldsymbol{w}$. The activation function $\sigma(\cdot)$ has different choices, such as the sigmoid function:

$$\frac{1}{1+\exp^{-\boldsymbol{w}^T\boldsymbol{x}_i+b}}\,,$$

and the **Re**ctified **L**inear **U**nit (ReLU) [1]:

$$\max(0, \boldsymbol{w}^T\boldsymbol{x}_i + b)\,.$$

The sigmoid function is highly related to logistic regression and is often used as the activation function of the output node to get probabilities of predictions. In

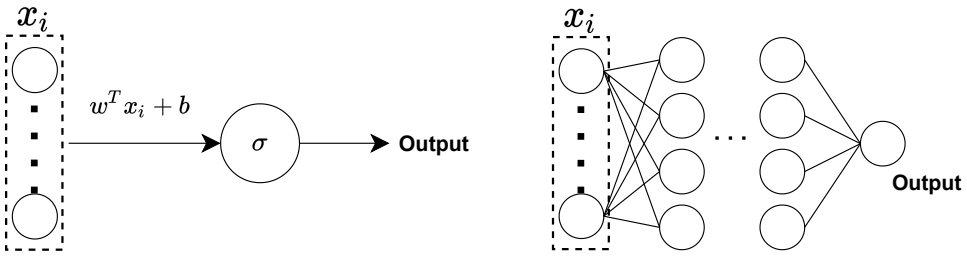


Figure 2.3: Left: illustration of a simple neuron. Right: illustration of a simple fully connected neural network.

contrast, the ReLU function is used in intermediate layers to make a non-linear decision boundary and avoid vanishing gradient problems [55] during the training process.

A simple neuron and a fully connected neural network are illustrated in Figure 2.3. The latter has multiple hidden layers. Particularly, the l-th hidden layer contains m_l^h simple neurons, and the j-th neuron of the hidden layer is associated with a pair of linear transformation parameters ($\boldsymbol{w}_{j,l}^T$, $b_{j,l}$). The outputs of all neurons of the previous layer ($\boldsymbol{h}_{l-1}$) are the inputs of the current layer. Consequently, the output of the current l-th layer is expressed as

$$\boldsymbol{h}_l = \boldsymbol{\sigma}\left(\begin{bmatrix} \boldsymbol{w}_{1,l}^T \\ \boldsymbol{w}_{2,l}^T \\ \vdots \\ \boldsymbol{w}_{m_l^h,l}^T \end{bmatrix} \boldsymbol{h}_{l-1} + \begin{bmatrix} b_{1,l} \\ b_{2,l} \\ \vdots \\ b_{m_l^h,l} \end{bmatrix}\right),$$

where $\boldsymbol{\sigma}(\cdot)$ is an elementwise activation function. During the training process, one estimates all pairs of ($\boldsymbol{w}_{j,l}^T$, $b_{j,l}$) of all layers to correlate outputs with ground truth labels relying on a backpropagation method [65, 84].

2.3.3 Unsupervised Autoencoder

The objective of an autoencoder is to learn efficient representations of unlabeled input data [53] (An example is illustrated in Figure 2.4). It is often represented by a neural network where outputs are expected to be the same as inputs. A typical autoencoder has two components: an encoder that maps the input into a latent space and a decoder that recovers the input from the latent representation. A latent space is also known as a latent feature space in which examples with similar meanings are expected to be closer to each other.

In anomaly detection tasks, autoencoder is widely applied to reconstruct the most salient pattern (genuine pattern) in datasets [131]. The autoencoder is en-

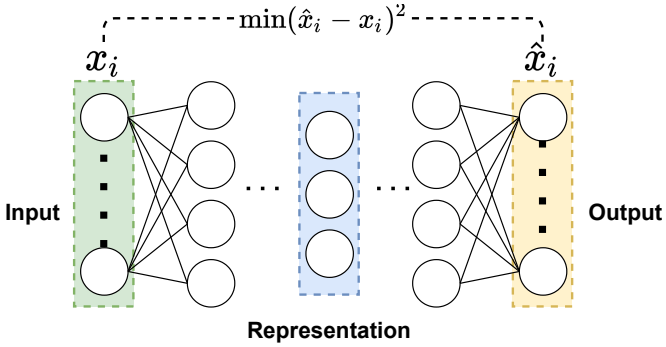


Figure 2.4: Illustration of a simple autoencoder. Outputs of the middle neurons are extracted efficient representations.

couraged to extract representations that encode the most frequent class of examples during the training process. As anomalies are often scarce, they seldom contribute to the extracted representations. Therefore, the trained autoencoder often can precisely reconstruct genuine examples while having a significant reconstruction error over abnormal ones.

Besides classical autoencoders, there are also many variations, such as denoising autoencoders [153] and variational autoencoders [67]. This manuscript provides details of the method **m**arginalized **S**tacked **D**enoising **A**utoencoder (mSDA) [22] that we have used to extract discriminative representations of the Amazon reviews datasets.

2.3.3.1 Marginalized Stacked Denoising Autoencoder

mSDA successively aggregates multiple **m**arginalized **D**enoising **A**utoencoders (mDA). A denoising autoencoder takes corrupted data as inputs and is expected to recover the clean ones on outputs. Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a matrix where each row represents an input example $\boldsymbol{x} \in \mathbb{R}^d$, and $\tilde{\boldsymbol{X}}$ be a copy of $\boldsymbol{X}$ while each dimension has a probability p_c to be set to 0. mDA aims to find a matrix $\boldsymbol{W} \in \mathbb{R}^{d \times d}$ that minimizes in expectation the following term for every possible corrupted $\tilde{\boldsymbol{X}}$:

$$\operatorname{tr}\left((\boldsymbol{X} - \boldsymbol{W}\tilde{\boldsymbol{X}})^T(\boldsymbol{X} - \boldsymbol{W}\tilde{\boldsymbol{X}})\right), \tag{2.1}$$

where $\operatorname{tr}(\cdot)$ returns the trace of a matrix. For a given corrupted $\tilde{\boldsymbol{X}}$, a minimizer of Equation (2.1) can be expressed as the well-known closed-form solution for ordinary least squares [5]:

$$\boldsymbol{W} = \boldsymbol{P}\boldsymbol{Q}^{-1}$$
$$\text{with } \boldsymbol{Q} = \tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T \text{and } \boldsymbol{P} = \boldsymbol{X}\tilde{\boldsymbol{X}}^T.$$

We adopt a bold capital letter to represent a matrix and a plain capital letter with indices to represent elements in the matrix.

Let $\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^T$, for every possible corrupted $\tilde{\boldsymbol{X}}$, the minimizer in expectation of Equation (2.1) is given as

$$\boldsymbol{W} = \mathrm{E}[\boldsymbol{P}]\,\mathrm{E}[\boldsymbol{Q}]^{-1}$$

$$\text{where } ,\mathrm{E}[Q]_{i,j} = \begin{cases} B_{i,j}p_c^2 & \text{if } i = j \\ B_{i,j}p_c & \text{otherwise} \end{cases} ,\text{and } \mathrm{E}[P]_{i,j} = B_{i,j}p_c\,.$$

Then the extracted representation of $\boldsymbol{x}$ by mDA is defined as $\tanh(\boldsymbol{W}\boldsymbol{x})$. The choice of $\tanh(\cdot)$ function is to introduce the nonlinearity into the model.

Additionally, mSDA uses a stack of mDA such that extracted representations of the previous autoencoder are inputs of the following. The final resulting representation is a concatenation of all representations of mDAs. Compared to classical autoencoders, mSDA is not sensitive to noise in datasets and can get more robust representations of inputs.

2.3.4 Unsupervised Clustering

Clustering is the task that groups examples into different sets, the so-called clusters, such that examples in the same cluster are more similar than those in different clusters. Plenty of clustering methods are available [125]: K-means, hierarchical clustering, spectral clustering, to name a few. This manuscript explains in detail the K-means method, as our proposition in Chapter 5 leverages a sibling approach.

K-means is an iterative clustering method. It starts by arbitrarily initializing k points in the input space (denoted by $\boldsymbol{c}_m$). Every point represents the center of a cluster. At each step of iterations, and for every input example $\boldsymbol{x}$, one computes distances between $\boldsymbol{x}$ and centers of clusters $\boldsymbol{c}_m$. $\boldsymbol{x}$ is assigned to the cluster with the shortest distance. Then one updates $\boldsymbol{c}_m$ by the following formula:

$$\boldsymbol{c}_m = \frac{1}{|\mathbb{X}_m|} \sum_{\boldsymbol{x} \in \mathbb{X}_m} \boldsymbol{x}\,,$$

where $\mathbb{X}_m$ is a set of input examples of the m-th cluster. One stops the process when $\boldsymbol{c}_m$ remains unchanged.

The K-means clustering is easy to interpret and implement. However, one needs to know the number of clusters k *a priori*.

2.4 Four Recurrent Challenges in Machine Learning

Previous sections focus on introductions of different families of supervised and unsupervised models. In addition, this section introduces some well-known key challenges that are particularly important to the challenge tackled in this thesis.

2.4.1 Mixed Types of Features

In machine learning tasks, especially when predicting tabular data, it is common to have the input space encompassing numerical and categorical dimensions, the so-called mixed types of features. However, some well-known models such as neural networks cannot directly tackle categorical features. Therefore, numerical representations of categorical values are required. A common approach is to encode one categorical feature into a real-valued vector. Different approaches are proposed to create such vectors.

One-hot encoding is one of the most widely used ones. The one-hot encoding of a categorical value is represented by a vector having the same length as the number of unique values. All dimensions of such a vector are 0 except the position that represents the unique value equals 1 [51]. Although one-hot encoding is easy to interpret, it is not scalable to attributes with plenty of categories. Moreover, distances (*e.g.*, the Euclidean distance) between the one-hot encoding of arbitrary pairs of unique values are always the same. Consequently, one-hot encoding cannot reveal different similarities between different pairs of categorical values.

A more advanced technique is the so-called embedding method, which is widely used in natural language processing tasks to search for real-valued vectors of words. Typically, embedding vectors encode words such that words that are closer, in terms of a given distance measure (*e.g.*, the cosine distance), in the embedding space, are expected to be similar in meaning [144]. The pioneering work word2vec [99] showed the efficiency of such word embedding techniques. Analogously, instance embedding methods [76, 161] are developed to get semantic representations of instance categories.

2.4.2 Imbalanced Dataset

As it is shown in Section 2.2.2, an example $\boldsymbol{x}_i$ is classified as positive if its prediction score $h(\boldsymbol{x}_i) > y_{\text{thres}}$. Note that the predicted labels depend on the values of the threshold y_{thres}. A default choice of y_{thres} is 0.5, supposing that the dataset has balanced classes. However, such a choice of y_{thres} is not suited for the case where numbers of positive and negative examples are different. When

proportions of negative and positive examples are significantly different, one addresses the problem of imbalanced datasets.

The principal idea of dealing with class imbalance is rescaling. Specifically, there are three types of rescaling approaches. i) The first approach is undersampling [89]; that is, one eliminates examples of the majority class such that both classes have nearly the same number of examples. ii) The second approach is oversampling [19], where one generates examples of the minority class through a sampling with replacing process. iii) The third approach is threshold-moving [26]; instead of sampling training datasets, one directly adjusts the threshold of classification y_{thres} such that proportions of different classes in outputs of classifiers correspond to the proportions of positive and negative classes in training datasets.

Note that the undersampling removes training examples and may eliminate important information of classifications. The oversampling makes some examples over-represented than the other, which increases the risk of overfitting these examples.

Concerning evaluation metrics, for a binary classification task, one can combine ground-truth outputs y_i with predictions $\mathbf{1}(h(\boldsymbol{x}_i) > y_{\text{thres}})$ to build a confusion matrix [139]:

Ground truth \ Prediction	Positive	Negative
Positive	TP	FN
Negative	FP	TN

TP, FN, FP, TN stand for the number of true positive, false negative, false positive, and true negative examples, respectively. Based on such a confusion matrix, one can define precision and recall metrics that are widely used in fraud detection tasks.

$$\text{precision} = \frac{TP}{TP + FP}\,,$$
$$\text{recall} = \frac{TP}{TP + FN}\,.$$

Generally speaking, values of precision and recall depend on the choice of y_{thres} and can hardly be maximized together. Setting y_{thres} to a large value results in considering the most confident positive examples; thus, the precision is high. However, only a few examples with $y_i = 1$ are considered positive, which gives a low recall value. In contrast, one can get a high recall value by considering all examples as positive, while the precision would be low.

Note that both precision and recall metrics are sensitive to the choice of y_{thres}, which complicates the comparison of performances between different models. To

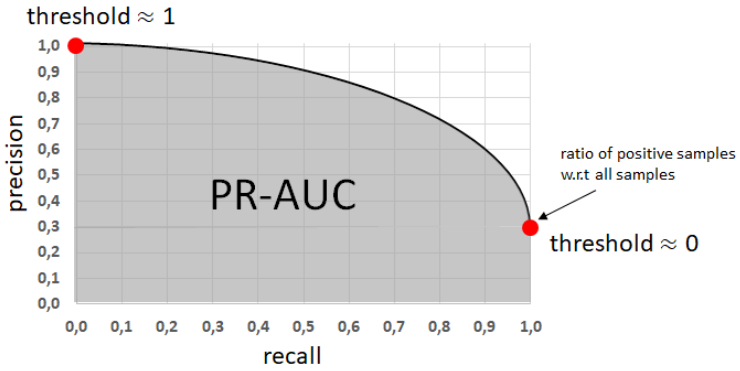


Figure 2.5: An illustration example of precision recall curve (Thomas Kurbiel 2020).[2]

get an evaluation metric independent of the choice of y_{thres}, one can use the area under a precision-recall curve (PR-AUC) to report performances of predictive models.

PR-AUC computes values of precision-recall pairs for different choices of y_{thres} and integrates the area under the formed curve to evaluate the performances of models. An example of computed PR-AUC is illustrated in Figure 2.5. A model has better performance if its PR-AUC is larger.

2.4.3 Feature Selection

In various tasks, input spaces of raw data generally contain noisy, classification irrelevant, or redundant features. For example, in the public Kaggle fraud detection tasks, the raw data have over 400 dimensions while most of them contain over 98% missing values. Identifying and removing such features can reduce the dimensionality of input space, simplify the training process and increase prediction performances. Therefore, feature selection is often an essential step in the machine learning paradigm [50]. Feature selection approaches can be roughly categorized as filter methods, wrapper methods, and embedded methods.

Filter methods select the optimal subset of features before the training process. One filters features with predefined statistical measures to estimate their contributions to classification approximately. Typical measures include mutual information and Pearson correlation [138] between features and outputs. Specifically, for a binary classification problem, Relief [69] is one of the most widely used measures. Given a point $\boldsymbol{x}_i$ of a training dataset, Relief finds the nearest point $\boldsymbol{x}_{i,\text{nh}}$ from the same class, the so-called near-hit, and the nearest point $\boldsymbol{x}_{i,\text{nm}}$ from the opposite

[2] `https://towardsdatascience.com/gaining-an-intuitive-understanding-of-precision-and-recall-3b9df37804a7`

class, the so-called near-miss. Then the Relief measure of the d-th feature is computed by

$$\sum_{i=1}^{n} c(x_i^d, x_{i,\mathrm{nm}}^d)^2 - c(x_i^d, x_{i,\mathrm{nh}}^d)^2 \,,$$

where x_i^d is the value of the d-th feature, and $c(\cdot, \cdot)$ is a distance measure. The computational complexity of Relief increases linearly with the number of features; thus, it is efficient for high-dimensional data.

Wrapper methods are applied after the training process. It uses testing performances as the criterion for the selection of subsets of features. Namely, one trains a classifier for every possible subset of features and evaluates the performance of the estimated classifier over a testing dataset. Clearly, how to create candidate subsets is a critical problem. A naive search of subsets of features explores all combinations of features. It suffers from the combinatorial explosion problem. Therefore, different search strategies such as particle swarm [106], and Las Vegas Wrapper [88] are applied to this problem.

Another feature selection approach is the embedded method. It plugs into the training process a regularization term; hence one automatically performs the feature selection during the estimation of classifiers. Two widely used embedded methods are ridge regression [147] and LASSO [146]. The ridge regression utilizes a L_2 norm to penalize large values of model parameters, whereas LASSO adopts a L_1 norm. Compared to ridge regression, LASSO can shrinkage the weights of certain features to 0; thus, it provides a sparser subset of features. In contrast, ridge regression provides a dense weighting parameter for each feature. One should manually eliminate features having a weighting parameter close to 0.

In Chapter 3, we propose our feature selection process leveraging wrapper methods. Different from feature selection methods for classical supervised learning, our proposition does not change the input space of classifiers and requires no retraining. In contrast, the method consists of finding the optimal subset of features that should be adapted before being given to the pre-trained model.

2.4.4 Interpretability of Machine Learning Models

Although most machine learning models solely aim to make more accurate predictions, one often wants to know "how" and "why" the models work. In many real-world scenarios, such as diagnoses, a single performance metric is incomplete to describe such a task [35]. Additionally, one should be able to explain "why" a model gives such predictions.

Some machine learning models, such as logistic regression and decision trees,

are interpretable by design. In logistic regression, one can directly know the importance of each feature by examining its weighting factor. As for decision trees, each classification result is obtained by following a sequence of simple rules. However, some well-performing models such as neural networks cannot be easily disentangled due to the depth and structural complexity. Therefore, various techniques are developed to help interpret more complex models. According to different levels of granularities, one can categorize such assistant methods as global interpretation methods and local interpretation methods.

Global interpretation methods aim to describe the average behavior of a machine learning model. An example is the permutation feature importance [16, 40] that measures contributions of features relying on the decreasing of prediction performances over permuted datasets. Namely, one chooses one feature from input spaces and shuffles its values of all input examples. Then, one computes the prediction performance of this shuffled dataset. Intuitively, shuffling a significant feature for predicting labels will decrease the performance more; hence the importance of features can be measured by such a performance decrease.

In contrast, local interpretation methods explain the prediction result of every individual example. **L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations (LIME) [123] is one of the most well-known ones. LIME first fixes an example of interest and samples its "neighborhoods" from an artificial Gaussian distribution, ignoring the correlation between features. Then, it builds a surrogate prediction model from classes of interpretable models over sampled data. Hence, the prediction result of the point of interest can be explained by the surrogate model. Another local interpretation method for deep neural networks is the saliency maps [137] method. For image recognition tasks, the saliency maps method can be applied to understand the contributions of each pixel to a classification result. For example, it may highlight pixels forming stripes when predicting a zebra.

2.5 Domain Adaptation

Machine learning methods of previous sections assume that testing data obey the same distribution as the training one. In real-life applications, such an assumption may not always hold [52, 110, 121]. For example, a fraud detection system trained in one country may not be appropriate to predict fraudsters in another country where customers have different payment habits. Clearly, one may not expect a good performance in this case, as testing data drift from the training ones.

The question is, "when the training and testing data are different, how can one leverages the knowledge that one has acquired from labeled training data to predict testing labels?" *Transfer learning* is one of the solutions. In a transfer

learning paradigm, such a training dataset is referred to as the *source domain*, and the testing dataset is referred to as the *target domain*. In real-life applications, source domains represent well-known markets. Generally, one has abundant labeled data in source domains, and a well pre-trained predictive model often exists. In contrast, target domains often represent new markets where one wants to expand their business, and they often have scarce or even missing labels. When source and target domains share the same input and output spaces, while their distributions are different, we stand in a particular case of transfer learning, the so-called *domain adaptation*.

Some other domain adaptation works [112, 130, 38] tackle a slightly different case where the target domain contains classes that are not in the source one. They refer to such a setting as *open set domain adaptation*. As the output space of the target domain is unknown, open set domain adaptation methods assign an artificial out-of-category label to unseen classes instead of true labels.

One can use different taxonomies to categorize domain adaptation methods. According to the family of transformation functions, it can be categorized as *classical domain adaptation* methods and *deep adaptation* methods. According to the number of adapted domains, it can be categorized as *single-source single-target domain (single-domain)* adaptations or *multi-subdomain* adaptations. We further introduce details of current adaptation methods according to different taxonomies and provide discussions of applying these methods to the Worldline adaptation tasks in Section 2.7.2.

2.5.1 Distribution Drift

We denote by $\mathbb{X}^t = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t}$ a set of realizations of target domain inputs that obey a marginal distribution $P(X^t)$, and $\mathbb{Q}^s = \{(\boldsymbol{x}_j^s, y_j^s)\}_{i=1}^{n_s}$ a set of realizations of source domain input-output pairs that obey a joint distribution $P(X^s, Y^s)$. We also define $\mathbb{X}^s = \{\boldsymbol{x}_j^s\}_{j=1}^{n_s}$ a set that contains only input examples of $\mathbb{Q}^s$. The distribution drift between target and source domain is expressed as $P(X^s, Y^s) \neq P(X^t, Y^t)$. As target domain labels are scarce or even missing, one cannot directly align source and target domain joint distributions to mitigate the distribution drift. Alternatively, a common assumption is considering source and target domain distributions are "similar"; thus, any rigorous domain adaptation study must characterize the underlying source-target similarity assumptions.

One common assumption is the so-called *covariate shift* setting [136], which describes the case where the source domain and target domain conditional output distributions coincide, that is $P(Y^s|X^s) = P(Y^t|X^t)$, whereas the marginal input distributions $P(X^s)$ and $P(X^t)$ differ. Typically, marginal input distributions

can be different in two domains due to a label-irrelevant sample selection bias [37, 59, 47], where samples in source and target domains are not collected according to the same criterion.

Label shift describes a case where proportions of classes in source and target domains are different, whereas conditional input distributions are the same [166], that is $P(Y^s) \neq P(Y^t)$ while $P(X^s|Y^s) = P(X^t|Y^t)$. Although target domains labels are not enough to precisely estimate $P(Y^t|X^t)$, one may have an estimation of propositions of different classes ($P(Y^t)$ can be known). For example, although it is costly to annotate all payment transactions in fraud detection tasks, fraudulent proportions are much easier to estimate and can be given by business experts. Under such an assumption, Saerens et al. [127] propose a simple procedure to calibrate outputs of source domain classifiers to fit target domain data. In a more challenging case when $P(Y^t)$ is unknown, an **E**xpectation-**M**aximization (EM) algorithm [33] is applied to iteratively approximate $P(Y^t)$ and adjusts source domain outputs [127, 2].

Besides output calibration methods, another family of approaches leverages the confusion matrix to estimate weighting factors $P(Y^t = y)/P(Y^s = y)$ for different values of $y \in \mathcal{Y}$ and then reweight source domain examples to retrain a predictive model [85]. Recent researches also leverage adversarial learning [80] to mitigate drifts in output distributions.

Another widely studied domain adaptation scenario is the so-called *concept shift* [70]. In such a setting, one assumes the existence of a transformation function $\mathcal{T}(\cdot)$ that aligns source domain conditional output distribution to the target one, that is $P(Y^s|\mathcal{T}(X^s)) = P(Y^t|X^t)$. The scenario addressed by this manuscript built upon a sibling setting and is introduced in Section 2.7.2.

2.5.2 Classical Single-Source Single-Target Domain Adaptation

In the following sections, we first introduce a *single-source single-target* domain adaptation scenario, then we talk about multi-subdomain adaptation in Section 2.5.4.

According to classes of transformation functions, domain adaptation can be roughly categorized as deep adaptation methods that rely on deep neural networks and classical adaptation methods that do not use neural networks to adapt data. In contrast to deep adaptations, classical methods are also referred to as shallow methods in some deep learning works [164, 24].

A sample approach to mitigate differences in marginal input distributions among classical adaptation methods leverages examples reweighting techniques

[141, 142]. One first estimates marginal input distributions in two domains and then reweights source examples to match target ones by minimizing **K**ullback–**L**eibler divergence (KL divergence) [72]:

$$\mathrm{KL}(P(X^t) \parallel P(X^s)) = \int_{\mathcal{X}} P(X^t = \boldsymbol{x}) \log \left(\frac{P(X^t = \boldsymbol{x})}{P(X^s = \boldsymbol{x})} \right) d\boldsymbol{x} \,.$$

Note that KL divergence is asymmetric and can be appropriately defined only if the support of $P(X^t)$ overlaps with the support of $P(X^s)$. Therefore, the scope of application of KL divergence is limited.

Statistical moment matching is another widely used technique. A particular moment matching method leverages the **M**aximum **M**eans **D**iscrepancy (MMD) [12] as a criterion for comparing distributions based on a **R**eproducing **K**ernel **H**ilbert **S**pace (RKHS) [114]. The empirical formula of MMD is expressed as

$$\left\| \frac{1}{n_s} \sum_{j=1}^{n_s} \phi(\boldsymbol{x}_j^s) - \frac{1}{n_t} \sum_{i=1}^{n_t} \phi(\boldsymbol{x}_i^t) \right\|_{\mathcal{H}} ,$$

where $\boldsymbol{x}_j^s \in \mathbb{X}^s, \boldsymbol{x}_i^t \in \mathbb{X}^t$, $\mathcal{H}$ is a universal RKHS and $\phi(\cdot)$ maps inputs to this space. Based on this measure, Pan et al. [111] and Baktashmotlagh et al. [6] propose to match target and source examples into a latent space where the MMD between source and target domains marginal input distributions is minimized. Long et al. [90] introduce pseudo-labeling techniques to "guess" target domain labels and propose to align the conditional distributions instead of the marginal ones.

In practice, MMD is computed relying on a kernel trick. For all input examples $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X} \times \mathcal{X}$, a kernel function $k(\boldsymbol{x}, \boldsymbol{x}') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ of the original input space can express the result of the inner product of $< \boldsymbol{x}, \boldsymbol{x}' >_{\mathcal{H}}$ in a RKHS. Relying on such kernel functions, one can compute MMD without explicitly defining $\phi(\cdot)$ [30].

Gong et al. [46] solve the domain adaptation problem by seeking an average representation of the source and target domains. They suppose that target and source domains are two points on a manifold. Then the average representation is the average of all representations along the geodesic between target and source domains. Other classical methods focus on the adaptation of correlation matrix [143] or principal axes [39]. The work of Ying et al. [163] proposes a new transfer learning perspective and introduces an approach to utilize meta-learning to solve domain transfer problems.

Another intuitive approach to mitigate domain drift is to identify domain invariant features between source and target domains. Blitzer et al. [8] introduce structural correspondence learning to induce correspondences among source and

target domain features automatically. Daumé III [32] leverages augmented input space and semi-supervised learning methods to train a common source and target domains classifier. Satpal and Sarawagi [133], Uguroglu and Carbonell [150], Gautheron et al. [45], Alshawabkeh et al. [3] adopt feature selection techniques to keep only features that do not shift between domains. However, such methods may eliminate discriminative features (features that contribute to label classification) from input spaces and decrease prediction performances.

2.5.3 Deep Single-Source Single-Target Domain Adaptation

Similar to classical domain adaptation methods, some deep adaptation methods also adopt a statistical moment matching method [91, 92] to align deep representations in a latent space. They plug into deep neural networks an adaptation layer to compute MMD between target examples and source ones. To reduce the computational complexity and fit the stochastic optimization regime of deep neural networks, they propose an approximate estimation of MMD, leveraging the stochastic gradient descent method [91]. Namely, they show that the global MMD can be iteratively approximated by MMDs of small batches of data. Therefore, the MMD can be added as a regularization term to the classification loss function. In light of this idea, Chen et al. [20] propose a higher-order moment matching method expecting a more precise alignment. Although the stochastic version of MMD requires less computation time, one still needs lots of iterations to get a precise approximation when dealing with a massive amount of data.

Another family of deep adaptation methods leverages batch normalization techniques [60, 78, 17] to align target and source domain mean and variance (the first and second-order moments). Intuitively, source and target data are respectively normalized in each layer, relying on standard normalization formulas:

$$\frac{X^s - \text{mean}(X^s)}{\sqrt{\text{var}(X^s)}}, \text{ and } \frac{X^t - \text{mean}(X^t)}{\sqrt{\text{var}(X^t)}}.$$

As batch normalization based domain adaptation methods solely focus on the mean and variance of source and target domains, they may not perform a precise adaptation when the underlying drifts between two domains are more complex. Nonetheless, they are easy to implement and far quicker to compute compared to MMD based adaptation methods.

Current deep adaptation methods focus more on the adversarial learning paradigm to generate domain invariant features [44, 149, 94]. A basic structure of adversarial neural networks is illustrated in Figure 2.6. A typical adversarial

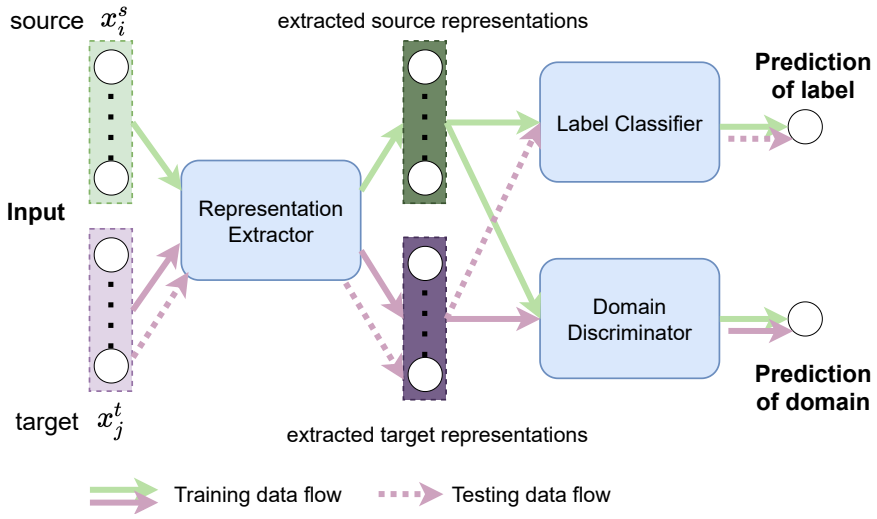


Figure 2.6: A basic structure of deep adversarial domain adaptation neural network.

domain adaptation schema consists of three components: i) A domain discriminator is trained to correctly distinguish source domain data from target ones by minimizing prediction errors of domains. ii) A label classifier is trained using source domain labeled data to classify examples. iii) A representation extractor is trained to generate domain invariant representations to fool the domain discriminator while minimizing classification errors. We often say the representation extractor and the domain discriminator play a minimax game [102] to evolve together. Such a structure encourages the representation extractor to generate classification discriminative while domain invariant features.

Inspired by the idea of adversarial adaptation networks, plenty of deep adaptation methods have been developed. Long et al. [93] propose to align not only source and target domain marginal distributions but also the conditional input distributions over pseudo-labeled outputs. Saito et al. [129] propose to train two classifiers by maximizing divergences of distributions of their outputs instead of explicitly training a domain discriminator.

Besides the aforementioned methods, various domain adaptation approaches leverage deep neural networks to improve their adaptation performances. Saito et al. [128] utilize tri-training to get robust and precise predictions of target examples by separately initializing two deep neural networks with different parameters. Kim et al. [66] translate the maximum output discrepancy principle [129] into the maximum posterior separation through a Gaussian process framework. The work of Teshima et al. [145] performs a casual mechanism transfer by supposing that invertible neural networks [68] can approximate the mechanism to generate labeled data.

Furthermore, pseudo-labeling methods have drawn more and more attention in domain adaptation tasks [74]. A classical yet efficient approach is to annotate examples of the most confident predictions (predictions close to 0 or 1 in a binary classification task) using their predicted labels. Such confidence-based pseudo-labeling approaches have achieved impressive performances in domain adaptation problems over various tasks [93, 157]. However, confident predictions are not guaranteed to be correct, and it is not clear how falsely annotated examples influence adaptation results.

Besides object classification tasks, deep adaptation approaches are also applied to address other challenges, such as visual question answering [18], style translation [75, 86], semantic segmentation [81, 171, 148], etc. Particularly, in the adaptation problem of semantic segmentation tasks, not only extracted representations are aligned between source and target domains, the pixel-level distance between source and target domains is also minimized [170].

2.5.4 Multi-Subdomain Adaptation

Although most domain adaptation works tackle the one source domain to one target domain adaptation problem, it is common to have the source or target domains encompass data from different distributions known as subdomains. Subdividing source and target domains into subdomains transforms the one-to-one adaptation problem to many-to-many (or one-to-many, many-to-one) problems, and can increase the flexibility of adaptation methods. Generally, there are two families of multi-subdomain adaptation problems: The first addresses a setting where separations of subdomains are known, while the second discovers hidden subdomains.

When labels of subdomains are given, similar to the *single-source single-target* domain adaptation scenario, different measures are used to lead multi-subdomain adaptations. Liu et al. [87] and Peng et al. [115] propose to match statistical moments between different source-target subdomain pairs relying on MMD. Zhao et al. [172] leverage adversarial learning to align multi-subdomains. However, building a transformation function between every pair of source-target subdomains is a combination problem. It is undesirable when the number of subdomains is significant. Alternatively, Li et al. [79] explore subdomain relationships by creating a subdomain similarity graph based on the Wasserstein distance [152]. Then transformation functions are only learned between subdomains and their nearest subdomains to decrease the computational complexity.

Furthermore, ideas of representation learning are also applied to multi-subdomain adaptation problems. Zhu et al. [175] propose to search a subspace that separates

the best subdomains and align source and target subdomain distributions in the subspace. Zhao et al. [173] extract subdomain invariant features and adjust the source subdomain predictive model to match target data. Different from the aforementioned methods, instead of aligning explicitly source and target subdomain input distributions, Venkat et al. [151] seamlessly adapt subdomains by encouraging predictions of subdomain classifiers to be similar.

The works of Mansour et al. [97], Xu et al. [158], and Hoffman et al. [57] leverage a weighted distribution combining rule for target label predictions. Basically, they suppose that the target domain distribution can be formed by combining multiple source subdomain distributions. Duan et al. [36] also use a weighting method of pre-trained source subdomain predictors, while their proposition only addresses the setting where a few target domain labels are given.

When dealing with hidden subdomains without subdomain labels, the first step is to reveal such hidden structures. Gong et al. [48] propose to discover them by maximizing subdomain discrepancies, and the number of subdomains is chosen to be the one that maximizes the source domain prediction performances. Hoffman et al. [56] leverage a clustering method of the input space to get hidden subdomains. Xu et al. [159] and Li et al. [77] build source subdomains by including only one positive example, and all negative examples. However, such a method creates lots of subdomains and is not scalable to massive datasets. Recent works of Mancini et al. [95, 96] discover hidden subdomains relying on a neural network of subdomain classifier, while the number of subdomains should be known *a priori*.

2.6 Optimal Transport

This section introduces some essential concepts of the optimal transport theory. The optimal transport has been revisited over the past years to solve a variety of computational problems [117], including many machine learning ones [31]. It is naturally suited for tabular data domain adaptation problems [28], as it offers a principled method to transform numerical and categorical target distributions seamlessly to source ones. Recent works that leverage optimal transport for domain adaptation suppose that adaptation functions between source and target domains can be considered as a transportation plan that minimizes displacement cost in Euclidean distance [116, 28, 29].

2.6.1 Monge-Kantorovich Problem

The optimal transport problem was first introduced by Monge in the 18th century [101] and further developed by Kantorovich in the mid-20th [62]. Intuitively,

the original Monge-Kantorovich problem looks for minimal efforts to move masses of dirt to fill a given collection of pits.

Let a discrete distribution $\mu_s = \sum_j^{n_s} a_j^s \delta_{\boldsymbol{x}_j^s}$ represents the distribution of departure, and $\mu_t = \sum_i^{n_t} a_i^t \delta_{\boldsymbol{x}_i^t}$ be the distribution of destination. $\delta_{\boldsymbol{x}}$ here is a Dirac function on the point $\boldsymbol{x}$, and $\boldsymbol{x}_j^s \in \mathbb{X}^s$, $\boldsymbol{x}_i^t \in \mathbb{X}^t$. a_i^t, a_j^s are weights of $\boldsymbol{x}_i^t$ and $\boldsymbol{x}_j^s$, respectively. We formalize the problem by discrete distributions since distributions are often represented by examples of their realizations in real-life applications.

Definition 2.2 (Monge problem)**.** Monge supposes that there are functions $\mathcal{T}(\cdot) : \mathcal{X} \to \mathcal{X}$ that relates all $\boldsymbol{x}_j^s \in \mathbb{X}^s$ to $\boldsymbol{x}_i^t \in \mathbb{X}^t$. The solution of optimal transport is the $\mathcal{T}(\cdot)$ that minimizes

$$\sum_j^{n_s} c(\boldsymbol{x}_j^s, \mathcal{T}(\boldsymbol{x}_j^s)) \,,$$
$$\text{s.t.} \, \forall i \in \{1, \dots, n_t\} : a_i^t = \sum_{j:\mathcal{T}(\boldsymbol{x}_j^s)=\boldsymbol{x}_i^t} a_j^s \,.$$

Note that this original Monge problem is an assignment problem: One point of $\mathbb{X}^s$ will be assigned to another location represented by a point in $\mathbb{X}^t$. It is known that such an assignment problem is NP-hard to resolve [109]. Moreover, this problem can be ill-posed, as $\mathcal{T}(\cdot)$ is restrained to one-to-one or one-to-many mapping functions and cannot address many-to-one cases.

The main issue of the Monge problem is that the whole weights of one source domain point can only be assigned to one location of another target domain discrete distribution. Kantorovich proposed a relaxation formalization of the original Monge problem to tackle this problem.

Definition 2.3 (Kantorovich relaxation)**.** Kantorovich [62] proposes to dispatch the weight of one source domain point across several locations of target domain. The resulting transportation is no more deterministic but probabilistic. A joint probability matrix $\boldsymbol{R} \in \mathbb{R}^{n_s \times n_t}$ represents the potential assignment between $\boldsymbol{x}_j^s$ and $\boldsymbol{x}_i^t$, and the relaxed minimization problem becomes

$$\begin{aligned} &\underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{i,j} c(\boldsymbol{x}_j^s, \boldsymbol{x}_i^t) R_{j,i} \,, \\ &\text{s.t.} \;\; \sum_j^{n_s} R_{j,i} = a_i^t \,, \text{and} \;\; \sum_i^{n_t} R_{j,i} = a_j^s \,. \end{aligned} \tag{2.2}$$

The relaxed optimal transport formalization is a linear optimization problem that has a polynomial computational complexity [108].

2.6.2 Entropy Regularization

The joint probability matrix obtained by minimizing Equation (2.2) generally gives a sparse solution where $R_{j,i} = 0$ for plenty of pairs of $(\boldsymbol{x}_j^s, \boldsymbol{x}_i^t)$. Such a solution eliminates possible mappings between certain $\boldsymbol{x}_j^s$ and $\boldsymbol{x}_i^t$ and cannot reflect some real cases [156]. Particularly, in a traffic simulation task, the predicted pattern leveraging optimal transport theory is much sparser than the real one. Indeed, one can always expect some traffic flows around the estimated optimal routes. Therefore, entropy regularization is applied to simulate such observations [117]. The regularized formalization of Equation (2.2) is expressed as

$$\begin{aligned} &\underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{i,j} \left(c(\boldsymbol{x}_j^s, \boldsymbol{x}_i^t) R_{j,i} \right) + \eta_{\text{reg}} \times l_{\text{reg}}(\boldsymbol{R}) \,, \\ &\text{s.t.} \;\; \sum_j R_{j,i} = a_i^t \,, \text{and} \;\; \sum_i R_{j,i} = a_j^s \,, \end{aligned} \tag{2.3}$$

where

$$l_{\text{reg}}(\boldsymbol{R}) = \sum_{i,j} R_{j,i} \log(R_{j,i})$$

is an entropy regularization term, and $\eta_{\text{reg}} > 0$ is a weighting hyperparameter to be fixed. As is proposed by Cuturi [31], this minimization problem is less computationally expensive and can be solved by a matrix scaling method [104].

Altschuler et al. [4] show that in the cases where $n_t = n_s = n$, solving the regularized optimal transport of Equation (2.3) requires about $O(n^2 \log(n))$ operations. Although this is less complex than the non-regularized version, it is still too expensive to apply to large dataset learning problems.[3]

2.6.3 Optimal Transport for Domain Adaptation

As optimal transport moves one distribution to another, it is natural to adopt this theory to address the domain adaptation problem, which aims to align source and target domain distributions. From the definition of regularized Kantorovich problem (Equation (2.3)), one can infer a deterministic transformation function to map source domain examples to the target domain:

$$\forall j \in \{1, \ldots, n_s\} \, : \mathcal{T}(\boldsymbol{x}_j^s) = \underset{\boldsymbol{x}' \in \mathcal{X}}{\operatorname{argmin}} \sum_{i=1}^{n_s} R_{j,i}^* c(\boldsymbol{x}', \boldsymbol{x}_i^t) \,, \tag{2.4}$$

where $\boldsymbol{R}^*$ is the minimizer of Equation (2.3). This method is known as the barycentric mapping in optimal transport literatures [11].

[3] The number of transactions in the Worldline fraud detection datasets is around ten millions.

Note that the presented optimal transport does not use any label information, while abundant labels are available in source domains and should be taken into account in adaptation. To this end, Courty et al. [28] propose to estimate a transportation plan by penalizing couplings that match source examples with different labels to the same target points. They add a new regularization term $l_{\text{group}}(\boldsymbol{R})$ to Equation (2.3) and introduce the following optimization problem:

$$\begin{aligned}\underset{\boldsymbol{R}}{\operatorname{argmin}} \sum_{i,j} \left(c(\boldsymbol{x}_j^s, \boldsymbol{x}_i^t) R_{j,i}\right) + \eta_{\text{reg}} \times l_{\text{reg}}(\boldsymbol{R}) + \eta_{\text{group}} \times l_{\text{group}}(\boldsymbol{R}) \,, \\ \text{s.t.} \ \sum_j R_{j,i} = a_i^t \,, \text{and} \ \sum_i R_{j,i} = a_j^s \,,\end{aligned}$$

where η_{group} is a weighting factor. Besides, Courty et al. [28] propose two choices of the regularizer $l_{\text{group}}(\boldsymbol{R})$: The first one is based on the group-LASSO [165] where each target example is only mapped with source examples from the same class, and the second is based on the graph Laplacian regularization [25], which promotes a structural consistency between original source data and the mapped ones. However, as we have already mentioned, computations of all these methods are still too costly to apply to a huge amount of data.

2.6.4 One-dimensional Optimal Transport

If supports of μ_s and μ_t are on the real-value axis, one stands in a special case of optimal transport, the so-called *one-dimensional optimal transport*. Instead of solving the regularized Kantorovich problem (Equation (2.3) and (2.4)) to get a probabilistic assignment matrix, one-dimensional optimal transport has a closed-form deterministic solution:

$$\mathcal{T}(x_j^s) = f_{\mu_t}^{-1}(f_{\mu_s}(x_j^s)) \,, \tag{2.5}$$

where $f_{\mu_s}(\cdot)$ and $f_{\mu_t}(\cdot)$ are cumulative distribution functions of μ_s and μ_t.

Let μ be an arbitrary one-dimensional real-value discrete distribution, and $\mathbb{X}$ a set of n realizations. Of note, $\mathbb{X}$ contains one-dimensional vectors, that is, scalars. A naive definition of the empirical cumulative distribution function of the discrete distribution μ is

$$f_\mu(x) = \frac{\text{rank}(x)}{n} \,; \forall x \in \mathbb{X} \,, \tag{2.6}$$

where $\text{rank}(\cdot)$ gives the ascending order of one example in the dataset it belongs to. Note that such a cumulative distribution function is only defined on a finite number of points. Following this definition, the output space of $f_{\mu_t}(\cdot)$ is discrete.

Hence $f_{\mu_t}^{-1}(\cdot)$ may be undefined on some points of $f_{\mu_s}(x)\,, \exists x \in \mathbb{X}$. Consequently, Equation (2.5) can be ill-posed.

In the computational optimal transport, it is common to extend Equation (2.6) to a continuous interval $\Delta_{\mathbb{X}} = \Big(\min(\mathbb{X}), \max(\mathbb{X})\Big)$ by interpolation [117] such that

$$f_{\mu}(x) = \frac{\frac{x - x_{\text{lower}}}{x_{\text{upper}} - x_{\text{lower}}} + \text{rank}(x_{\text{lower}})}{n}\,; \forall x \in \Delta_{\mathbb{X}}, x \notin \mathbb{X}\,.$$

x_{lower} and x_{upper} are respectively the largest value smaller than x and the smallest value larger than x in $\mathbb{X}$. After the interpolation, $f_{\mu_t}^{-1}(\cdot)$ is properly defined on the interval $[1/n_t, 1]$. For values $f_{\mu_s}(x) < 1/n_t$, one can simply set $f_{\mu_t}^{-1}(f_{\mu_s}(x)) = \min(\mathbb{X}^t)$.

This solution of Equation (2.5) is known as the increasing arrangement. As estimating cumulative distribution functions involves sorting a set of real values, the computational complexity is $O(n \log(n))$, which is far simpler than the original multi-dimensional optimal transport.

In light of the simplicity of one-dimensional optimal transport, when tackling multi-dimensional distributions, Bonneel et al. [10] propose to project them to a large set of random directions and iteratively perform one-dimensional optimal transport among each direction. In comparison, Meng et al. [98] leverage the projection pursuit regression [43] to find the most informative direction of projection at each step of iterations. However, in some real-life applications, features are often generated by business experts and represent meaningful characteristics of transactions. Projections of these features may not be easily interpreted.

2.7 Worldline Fraud Detection Task

This section presents the characteristic of the problem brought by Worldline. For the existing markets where Worldline has abundant labeled data, the company trains well-performed predictive models leveraging supervised learning methods. Whereas for the new markets, Worldline needs domain adaptation strategies to predict fraudsters.

2.7.1 Worldline Pre-trained Source Domain Predictive Model

As we have introduced in Chapter 1, Worldline processes a considerable amount of transactions. Therefore, applied predictive models should be scalable to massive datasets. In addition, as fraudsters pretend to be genuine ones, predictive models are also required to be flexible enough to capture such slight differences. Indeed,

in the source domains, Worldline aggregates GBDT and neural networks (Sections 2.3.1.1, 2.3.2) in complement to pre-defined expert rules to predict fraudsters. This aggregation of models is what we name in the following a "black-box" model.

During the training process, Worldline relies on log-loss to estimate predictive models on existing markets. For neural network models, Worldline leverages instance embedding methods to get real-valued numerical representations of categorical attributes. Embedding features and other numerical attributes are then passed to a fully connected neural network for fraud predictions. Hornik et al. [58] has proved that, if a feedforward neural network is deep enough, it can approximate arbitrary continuous functions at any precision. Theoretically, neural network models are flexible enough for Worldline fraud detection tasks. Additionally, thanks to the improvements of computing power in CPUs and GPUs, the training of deep neural networks over massive datasets is possible. Moreover, the threshold-moving approach is applied to find the optimal threshold of separations in the fraud detection tasks. For evaluation, Worldline adopts PR-AUC to take into account the imbalanced characteristics of fraud detection datasets.

2.7.2 Worldline Domain Adaptation Tasks

In the Worldline fraud detection tasks, as the company wants to predict fraudsters in different countries, the source and target domains' output spaces are always the same. Besides, they are in a homogeneous case where they utilize the same input features for all countries. Therefore, we stand in the domain adaptation paradigm of transfer learning.

In the Worldline domain adaptation problem, proportions of fraudulent transactions are different between markets; thus, we are in a *label shift* setting. Besides, due to different payment habits, we also observe $P(Y^s|X^s) \neq P(Y^t|X^t)$. $P(Y^s|X^s)$ is represented by pre-trained source domain classifiers $h_s(\cdot)$. Such classifiers are required to be preserved and should not be changed. Therefore, we are not in a concept shift scenario (concept shift applies a transformation $\mathcal{T}(\cdot)$ over X^s to change $h_s(\cdot)$). Consequently, we propose our *target to source* domain adaptation formalization in Chapter 3 to address this particular setting.

Although the aforementioned classical and deep adaptation methods have achieved impressive performances over some benchmark datasets of image recognition and sentiment analysis tasks, they are not adequate to address the Worldline domain adaptation problem.

Of note, the introduced classical domain adaptation methods are efficient in transforming numerical features. However, some essential concepts that they

Table 2.1: Comparison of some domain adaptation methods

	Methods	Model-agnostic	Interpretable	Feature-type Free	Huge Dataset	Retraining-Free
Classical	**Ours**	✓	✓	✓	✓	✓
	Pan et al. [111]	✓	✗	✗	✗	✗
	Long et al. [90]	✓	✗	✗	✗	✗
	Courty et al. [28]	✓	✓	✗	✗	✓
	Sun et al. [143]	✓	✓	✗	✓	✗
Deep	Long et al. [92]	✗	✗	✓	✓	✗
	Ganin et al. [44]	✗	✗	✓	✓	✗
	Saito et al. [129]	✗	✗	✓	✓	✗

adopt, such as the covariance matrix and the principal axis, cannot be defined for categorical dimensions. Whereas categorical features commonly exist in machine learning datasets, especially for the Worldline fraud detection tasks. Furthermore, some metrics such as MMD are not scalable to a massive dataset. Indeed, to compute MMD, one should apply the kernel function over every pair of inputs in a dataset. The number of input pairs increases quadratically with the number of inputs. The Worldline fraud detection datasets contain tens of millions of transactions, so it is unfeasible to compute MMD over the entire dataset for classical domain adaptation methods.

Although deep adaptation methods can address the computational problem and tackle categorical features, they are highly dependent on deep neural networks and are not *model-agnostic*. Most deep adaptation methods transform source and target data into a common latent space or adapt source domain data into the target one. Therefore, a retraining process is necessary during the adaptation process. Such a retraining process requires lots of expertise and is unfeasible to apply to every new market. Several recent works [82, 73, 162] that leverage a pre-trained source model for domain adaptation focus on the adaptation of image data. Moreover, they stand in a setting where source domain data are unavailable and assume that the pre-trained models are neural networks. However, such methods do not address the Worldline domain adaptation problem, as the company expects domain adaptation methods to work on tabular data and has no restriction over the family of predictive models. Indeed, in the Worldline industrial applications, the source domain pre-trained model can be from various methods such as decision trees or expert rules. The adaptation method should be *model-agnostic* to address such "black-box" predictive models.

Furthermore, the training process of deep neural networks leverages stochastic gradient descent optimization methods. However, applying such a stochastic method for domain adaptation over highly imbalanced datasets may create bias. Namely, batches of data may contain examples of only one class and fail to represent the global marginal input distribution. For example, the proportion of

fraud in the Worldline datasets is around 0.2%. By a simple computation, one can know that, for a batch of 256 data points, there is a chance of more than 59% that no negative example is presented in this batch.

Consequently, our propositions (Chapters 3, 4 and 5) address the Worldline domain adaptation problem by leveraging optimal transport theory. Specifically, we transform target domain data into the source one by performing one-dimensional optimal transport among each feature dimension.

Table 2.1 highlights the added values of our proposition compared to some typical classical and deep adaptation methods.

2.8 Datasets Used in the Experiments

This section deals with datasets that we utilize in experiments of the following chapters. We evaluate the proposed machine learning methods over three datasets: Worldline fraud detection dataset, Kaggle fraud detection dataset, and Amazon review dataset. The Worldline dataset represents a real-life scenario of applications. However, the dataset is not published for confidential reasons. Besides, we choose to evaluate performances over the Kaggle fraud detection dataset, as it has similar characteristics as the Worldline fraud detection dataset while being public. Both datasets have imbalanced classes and mixed types of features. The Amazon review dataset is a common benchmark of domain adaptation researches. Although most domain adaptation methods using the Amazon review dataset do not fit the requirements of Worldline adaptation tasks, it is instructive to evaluate our proposition and compare it with other adaptation methods over this well-known dataset.

2.8.1 Worldline Fraud Detection Dataset

This dataset consists of real anonymous clients' transactions from July 2018 to September 2018 of two geographical areas: Belgium and Germany. Both datasets have 23 numerical attributes and 7 categorical ones. All features are generated by experts in payment, and a preprocessing process is already applied to guarantee that irrelevant features for fraud detection are removed. The number of examples in the Belgian dataset is over 30 million and is around 15 million in the German dataset. The proportions of fraud are respectively 0.3% and 0.5% in the two countries. Clearly, as proportions are different in the two countries, we can conclude that there is a potential drift between them. Note that classes of labels are highly unbalanced in our fraud detection datasets, thus completing domain adaptation tasks.

We consider the whole Belgian dataset as our source domain (denoted by B). The Belgian dataset is separated into four parts among the axis of time. The first two parts are used to estimate the source domain predictive model, and the third part is served as a validation set for searching of training parameters. We use the fourth part as the testing set to report prediction performances of machine learning models. As for German data, to have a more realistic experiment, we consider every month of data as one target domain, which results in 3 target domains in total (denoted by G-1, G-2, G-3 separately). This setting is based on the realistic fact that the amount of data of a new market is generally far less than existing markets. Then in each target domain, we separate data into training sets, validation sets, and testing sets as the source domain.

In reality, both source domain and target domains are total labeled. To simulate a domain adaptation setting, we use little target domain labeled information in a weakly supervised case and no label in an unsupervised setting when estimating adaptation functions. We only leverage labels of target domain testing sets to evaluate the performances of proposed adaptation methods.

2.8.2 Kaggle Fraud Detection Dataset[4]

The dataset contains payment transactions issued from mobile devices and desktop devices, and one aims to predict if an online transaction is fraudulent or not. The raw data dimension is over 400, while most features contain missing values and some are not discriminative. We discard features with more than 1% of missing values and all transactions containing missing values. To discard label-irrelevant features, we first train a predictive model in a supervised setting and predict test data where one feature's values are randomly shuffled. The feature is considered label-irrelevant if the prediction performance remains nearly the same compared to the not-shuffled test dataset's performance. After preprocessing, the dataset used in experiments has around 400,000 examples with 43 numerical features and 8 categorical ones.

Similar to the Worldline fraud detection dataset, transactions of this dataset follow a chronologic order. We consider the mobile device as the source domain (denoted by M) and separate the dataset into training, validation, and testing sets. Transactions of the desktop device are divided among axis of time to create three target domains denoted by D-1, D-2 and D-3. The proportions of fraud in each domain are respectively 10% and 7%, which are much larger than our fraud detection dataset.

Although we evaluate our methods on two fraud detection datasets, the drifts

[4] `www.kaggle.com/c/ieee-fraud-detection`

between source and target domains of these two datasets are different. In the Kaggle fraud detection task, the drift comes from the change of device. In contrast, in the Worldline fraud detection task, source and target distributions differ as geographical localization (users' payment habits) changes. Hence, they are entirely two different domain adaptation tasks.

2.8.3 Amazon Review Dataset

The dataset contains reviews of buyers on the Amazon website across different categories of products [8]. Each review is a small paragraph of texts, transformed into bags-of-words representation and labeled as positive or negative. Note that the sentiment classification model trained using supervised learning to predict buyers' points of view for one category does not directly generalize to another. Following the setting of Chen et al. [22], we consider 4 domains: Books (B), DVDs (D), Electronics (E), and Kitchen appliances (K). Each domain has 2,000 training examples and around 4,000 test examples with perfectly balanced labels. We keep the most frequent 400 words dimensions and generate features from bags-of-words representations using mSDA unsupervised auto-encoder [22] with 5 layers. Instead of stacking all hidden dimensions as Chen et al. [22] and Ganin et al. [44], we take only the representation of the last layer. Different from the aforementioned two fraud detection datasets, the features of the Amazon reviews dataset are all numerical ones and may not have explicit meaning that can be easily interpreted.

Chapter 3

Single-Target to Single-Source Domain Adaptation

This chapter presents the first contribution of this thesis. Results of this chapter have been published in the paper "Target to Source Coordinate-wise Adaptation of Pre-trained Models" [167]. We propose a new target to source paradigm for domain adaptation without re-training the predictive models. Based on such a new setting, we propose to use one-dimensional optimal transport to transform target domain data. Moreover, instead of adapting all dimensions of input spaces, we propose to seek a sparse and ordered coordinate-wise adaptation of the feature space, in addition to elementary mapping functions. We introduce a weakly supervised process that relies on scarce labeled target data to automatically select the subset of features to be adapted. The proposed final approach is model-agnostic, feature-type free, and easy to interpret.

3.1 Formalization

Recall that $\mathcal{X}$ and $\mathcal{Y}$ are respectively input space and output space. In this manuscript, we are interested in fraud detection problems such that $\mathcal{X}$ encompasses vectors of categorical and numerical types, possibly mixed together. Moreover, we focus on classification problems with binary labels $\mathcal{Y} = \{0, 1\}$, but our method naturally extends to multilabel classification. In the domain adaptation framework, one observes a labeled dataset $\mathbb{Q}^s = \{(\boldsymbol{x}_j^s, y_j^s)\}_{j=1}^{n_s}$ over source domains, and each observed sample $(\boldsymbol{x}_j^s, y_j^s)$ is viewed as a realization of a random variable pair $(X^s, Y^s) \in \mathcal{X} \times \mathcal{Y}$ obeying a joint probability $P(X^s, Y^s)$.

Besides, an unlabeled target domain dataset $\mathbb{X}_u^t = \{\boldsymbol{x}_i^t\}_{i=1}^{n_t^u}$ is available, with

$\boldsymbol{x}_i^t$ realizations of $X^t \in \mathcal{X}$ obeying a marginal probability $P(X^t)$. When addressing a weakly supervised setting, one also have a few labeled target data $\mathbb{Q}_l^t = \{(\boldsymbol{x}_i^t, y_i^t)\}_{i=1}^{n_t^l}$, where $(\boldsymbol{x}_i^t, y_i^t)$ are realizations of $(X^t, Y^t) \in \mathcal{X} \times \mathcal{Y}$ obeying a joint probability $P(X^t, Y^t)$. Recall that $\mathbb{X}^t$ and $\mathbb{X}^s$ are defined to be marginal input sets, we have $\mathbb{X}^t = \mathbb{X}_u^t \cup \{\boldsymbol{x}_i^t | (\boldsymbol{x}_i^t, \cdot) \in \mathbb{Q}_l^t\}$ standing for all input examples of the target domain. The size of $\mathbb{X}^t$ is $n_t = n_t^l + n_t^u$. Analogously, the source domain input set is expressed as $\mathbb{X}^s = \{\boldsymbol{x}_j^s | (\boldsymbol{x}_j^s, \cdot) \in \mathbb{Q}^s\}$.

The aim of domain adaptation classification problem is to infer the target domain predictive model $P(Y^t|X^t)$ relying on such data. Courty et al. [28] assume that there exists a *mapping function* $\mathcal{T}(\cdot) : \mathcal{X} \to \mathcal{X}$ that models the domain drift from the source to the target, such that $P(Y^s|\mathcal{T}(X^s)) = P(Y^t|X^t)$. Our work builds on a sibling assumption, that is the existence of a mapping function $\mathcal{G}(\cdot) : \mathcal{X} \to \mathcal{X}$ that models the *domain drift* from the target to the source:

$$P(Y^s|X^s) = P(Y^t|\mathcal{G}(X^t))\,. \tag{3.1}$$

Note that $P(Y^s|X^s)$ is estimated over source domain data and is often given as a *pre-trained model* in real-life industrial applications. In fraud detection tasks, Y^s takes values in $\{0, 1\}$, and we can further denote the pre-trained predictive model by $h_s(x) = P(Y^s{=}1|X^s{=}x)$ for the sake of simplicity. Precisely, $h_s(x)$ gives the probability that a source transaction x is fraudulent. Relying on Equation (3.1), we give the definition of target domain predictive model $h_t(\cdot)$ in Definition 3.1.

Definition 3.1 (Target domain predictor)**.**

$$\forall x \in \mathcal{X} \;:\quad h_t(x) = h_s \circ \mathcal{G}(x)\,,$$

where $\circ$ represents the composition operation between two functions.

Relying on such a formalization of target domain predictor, we do not need to train a target domain predictive model $h_t(\cdot)$ to predict target labels. Instead, we "move" target domain data to source domains through $\mathcal{G}(\cdot)$ to directly use pre-trained source predictors. However, it is clear that, one cannot directly modelize $P(Y^t|\mathcal{G}(X^t))$ to estimate $\mathcal{G}(\cdot)$ in the lack of annotated target data. Nonetheless, we propose a guarantee over target domain risk when using $\mathcal{G}(\cdot)$ as an adaptation function.

Proposition 3.1 (Guarantee over target domain risk)**.** Under the assumption that

$$P(Y^s) \;=\; P(Y^t)\,, \tag{3.2}$$

if we can find a transformation $\mathcal{G}(\cdot) : \mathcal{X} \to \mathcal{X}$ such that

$$\forall \boldsymbol{x} \in \mathcal{X}\,; \forall y \in \mathcal{Y} : \quad P(X^t = \boldsymbol{x} | Y^t = y) = P(X^s = \mathcal{G}(X^t = \boldsymbol{x}) | Y^s = y)\,, \quad (3.3)$$

then given a loss function $l(\cdot, \cdot)$: $[0, 1] \times \mathcal{Y} \to \mathbb{R}_+$, for any $h_s(\cdot) : \mathcal{X} \to [0, 1]$, we have

$$r_t^l(h_s \circ \mathcal{G}) = r_s^l(h_s)\,, \quad (3.4)$$

where

$$r_t^l(h_s \circ \mathcal{G}) = \mathop{\mathrm{E}}_{(\boldsymbol{x},y)\sim P(X^t,Y^t)} \Big[l(h_s \circ \mathcal{G}(X^t = \boldsymbol{x}), Y^t = y) \Big]\,,$$
$$r_s^l(h_s) = \mathop{\mathrm{E}}_{(\boldsymbol{x},y)\sim P(X^s,Y^s)} \Big[l(h_s(X^s = \boldsymbol{x}), Y^s = y) \Big]\,.$$

Proof. The proof is straightforward by noticing that Equations (3.2) and (3.3) imply

$$\forall \boldsymbol{x} \in \mathcal{X}\,; \forall y \in \mathcal{Y} : \quad P(X^t = \boldsymbol{x}, Y^t = y) = P(X^s = \mathcal{G}(X^t = \boldsymbol{x}), Y^s = y)\,.$$

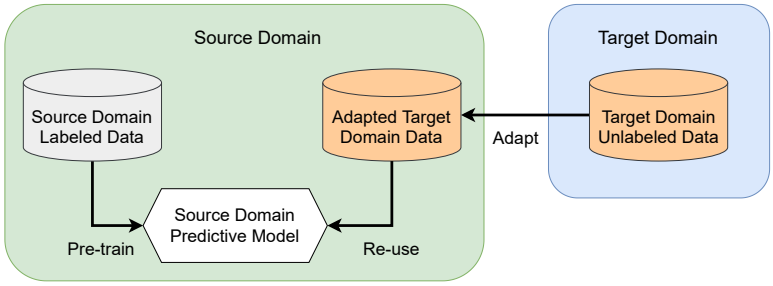
Since the $\mathcal{G}$-mapped target joint probability equals the source joint probability, $P(\mathcal{G}(X^t), Y^t) = P(X^s, Y^s)$. Then Equation (3.4) is obtained by a change of variable. □

As the pre-trained $h_s(\cdot)$ is the source domain optimal Bayes predictor, the risk on source domain data $r_s^l(h_s)$ is generally small, which induces the risk on target domain data $r_t^l(h_s \circ \mathcal{G})$ to be small. However, due to the lack of labeled data, we cannot properly estimate the conditional distribution of the input X^t given the output Y^t. In these conditions, we relax the requirement of Equation (3.3), and we seek for a function $\mathcal{G}(\cdot)$ belonging to the family of transformations that aligns the input marginal distributions:

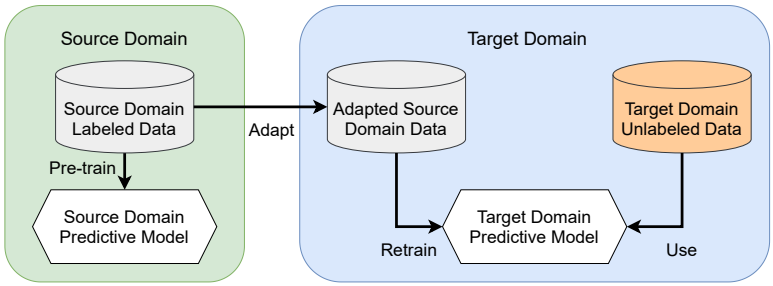
$$P(X^s) = P(\mathcal{G}(X^t))\,. \quad (3.5)$$

As source and target domain inputs are given, estimating $\mathcal{G}(\cdot)$ that aligns Equation (3.5) is possible.

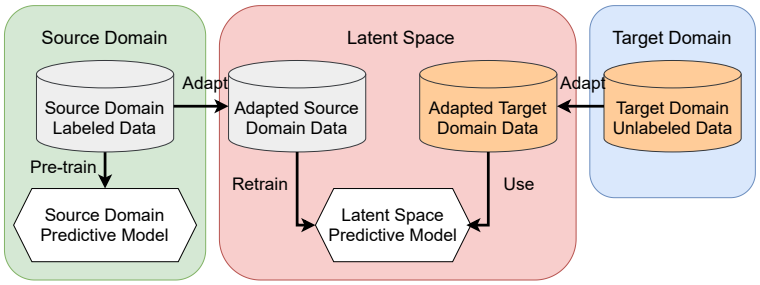
Albeit simple, this training-free *target to source* perspective on domain adaptation is, up to our knowledge, an unexplored problem. Figure 3.1 illustrates three domain adaptation settings. Figure 3.1a shows our proposition. Figure 3.1b is the method applied by Courty et al. [28]. Figure 3.1c is widely used by current deep adaptation methods [44, 129].



(a) Target to source domain adaptation.



(b) Source to target domain adaptation.



(c) Latent space domain adaptation.

Figure 3.1: Three different settings of domain adaptation.

3.2 Label Shift Adjustment

Proposition 3.1 assumes an equality between source and target domain output distributions. However, this assumption can be violated in some real-life applications, such as the Worldline fraud detection tasks. It is shown in Section 2.8 that the proportion of fraud changes for different geological locations. Therefore, we should first correct this *label shift* before seeking the adaptation function $\mathcal{G}(\cdot)$. Although target labels are scarce or missing in weakly and unsupervised domain adaptation scenarios, proportions of each class in the target domain can be known. For example, despite the lack of labels in target domains in a fraud detection system, one may have the proportion of fraud estimated by payment experts. For a classification problem where the output space is discrete, the proportion of classes represents the output distribution $P(Y^t)$.

As input-output pairs (X^s, Y^s) of source domains are given, and target domain

output marginal distribution $P(Y^t)$ is accessible, then, one can get $P(Y^s) = P(Y^t)$ by reweighting source examples by classes. The reweighted source data have no *label shift* compared to target ones. However, one can no longer guarantee the pre-trained black-box model $h_s(\cdot)$ to be the optimal one in the reweighted source domain. Thus, in light of the work of Saerens et al. [127] and Lin et al. [83], we propose to calibrate $h_s(\cdot)$ by the following method such that it gives optimal predictions for examples in the reweighted source domain.

For the simplicity of analysis, we note respectively X^p and Y^p the input and output variables of reweighted source domain. By definition, the output distribution of the reweighted source domain is the same as the target one, and input examples of the same class are reweighted by the same factor. Consequently, we have

$$P(Y^p) = P(Y^t) \text{ and } P(X^p|Y^p) = P(X^s|Y^s)\,.$$

Proposition 3.2 (Calibration of source model)**.** Let $h_s(\cdot)$ be the pre-trained optimal Bayes binary classifier in the source domain. The optimal predictor $h_p(\boldsymbol{x}) = P(Y^p = 1|X^p = \boldsymbol{x})$ in the reweighted source domain is obtained by:

$$h_p(\boldsymbol{x}) = \frac{h_s(\boldsymbol{x})w(1)}{h_s(\boldsymbol{x})w(1) + (1 - h_s(\boldsymbol{x}))w(0)}\,, \tag{3.6}$$

where

$$w(y) = \frac{P(Y^t = y)}{P(Y^s = y)}\,. \tag{3.7}$$

The proof of this proposition is detailed in Appendix A.2. This calibration approach was first proposed by Saerens et al. [127]. However, we get this similar proposition independently.

The proposition suggests that the difference between marginal output distributions of source and target domains can be mitigated by calibrating outputs of the pre-trained black-box model. For the sake of simplicity, hereafter, we use X^s and Y^s instead of X^p and Y^p to express the source domain where data are already reweighted to match the target domain output distribution, and $h_s(\cdot)$ refers to the calibrated optimal predictive model in the reweighted domain. As it is shown in Appendix A.1, to measure the impact of model calibration, PR-AUC is not an adequate measure. Therefore, we report performances in both log-loss and PR-AUC whenever possible.

3.3 Target to Source Optimal Transport for Domain Adaptation

To seek the transformation function $\mathcal{G}(\cdot)$ that maps target examples into the source domain, one should define the family of this transformation. Besides, some constraints should be verified: i) the transformation should be *feature-type free*, ii) the transformation should be scalable to a huge amount of data, iii) the transformation should be easily interpretable. We decide to leverage the optimal transport for domain adaptation, as it naturally satisfies the first requirement through different definitions of numerical and categorical distance functions. We further relax the original multi-dimensional optimal transport in Section 3.4 using *coordinate-wise* adaptation functions to meet the second requirement. Moreover, we introduce a feature selection process in Section 3.5 to make our proposition easily interpretable.

Following the definition of classical optimal transport introduced in Section 2.6, the remainder of this section tailors the formalization of optimal transport to the *target to source* domain adaptation scenario. Central to optimal transport methods is the notion of a *cost function* between a source point and a target point, denoted by

$$c(\cdot,\cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}\,. \tag{3.8}$$

Moreover, we let $\boldsymbol{C} \in \mathbb{R}^{n_t \times n_s}$ denote the *cost matrix* between source and target training points such that $C_{i,j} = c(\boldsymbol{x}_i^t, \boldsymbol{x}_j^s)$ corresponds to the cost of moving weight from $\boldsymbol{x}_i^t \in \mathbb{X}^t$ to $\boldsymbol{x}_j^s \in \mathbb{X}^s$. As discuss in Section 3.4, the cost may be defined both for categorical and numerical features.

Based on these concepts, we present below the target domain to source domain Kantorovich [62] formulation of the multi-dimensional optimal transport problem in the discrete case.

Definition 3.2 (Target to source domain Kantorovich's discrete optimal transport problem)**.** The relationship between source and target examples is encoded as a joint probability *coupling matrix* $\boldsymbol{R} \in \mathbb{R}_+^{n_t \times n_s}$, where $R_{i,j}$ corresponds to the *weight* to be *moved* from $\boldsymbol{x}_i^t \in \mathbb{X}^t$ to $\boldsymbol{x}_j^s \in \mathbb{X}^s$. The set of admissible coupling matrices is given by

$$\Gamma = \left\{ \boldsymbol{R} \in \mathbb{R}_+^{n_t \times n_s} \;\middle|\; a_i^t = \sum_{j'=1}^{n_s} R_{i,j'} \text{ and } a_j^s = \sum_{i'=1}^{n_t} R_{i',j} \right\},$$

where a_i^t (resp. a_j^s) is the weight of $\boldsymbol{x}_i^t \in \mathbb{X}^t$ (resp. $\boldsymbol{x}_j^s \in \mathbb{X}^s$), and we have discrete target (resp. source) domain distributions expressed by $\mu_t = \sum_i^{n_t} a_i^t \delta_{\boldsymbol{x}_i^t}$ (resp. $\mu_s = \sum_j^{n_s} a_j^s \delta_{\boldsymbol{x}_j^s}$).Typically, we consider that the mass is uniformly distributed

among each point, *i.e.* $a_i^t = 1/n_t$ and $a_j^s = 1/n_s$, but the framework allows reweighing the samples, such that

$$\sum_{i=1}^{n_t} a_i^t = \sum_{j=1}^{n_s} a_j^s = 1\,; \quad a_i^t, a_j^s \geq 0\,.$$

Then, the optimal coupling matrix $\boldsymbol{R}^*$ is obtained by solving

$$\boldsymbol{R}^* = \underset{\boldsymbol{R}\in\Gamma}{\operatorname{argmin}} \langle \boldsymbol{C}, \boldsymbol{R} \rangle = \underset{\boldsymbol{R}\in\Gamma}{\operatorname{argmin}} \sum_{i=1}^{n_t}\sum_{j=1}^{n_s} C_{i,j} R_{i,j}\,. \tag{3.9}$$

Following the barycentric mapping introduced in Equation (2.4), the transformation function $\mathcal{G}(\cdot)$ is given by

$$\forall i \in \{1, \ldots, n_t\}\,, \mathcal{G}(\boldsymbol{x}_i^t) = \underset{\boldsymbol{x}'\in\mathcal{X}}{\operatorname{argmin}} \sum_{j=1}^{n_s} \boldsymbol{R}_{i,j}^* c(\boldsymbol{x}', \boldsymbol{x}_j^s)\,. \tag{3.10}$$

The obtained joint probability $\boldsymbol{R}^*$ reveals the allocation of mass from one domain to the other.

Note that Equation (3.10) is only defined on examples in $\mathbb{X}^t$. For unseen target examples $\boldsymbol{x}$ drawn from $P(X^t)$ while $\boldsymbol{x} \notin \mathbb{X}^t$, we first project $\boldsymbol{x}$ to its nearest $\boldsymbol{x}_i^t \in \mathbb{X}^t$ according to $c(\boldsymbol{x}_i^t, \boldsymbol{x})$ before applying the barycentric mapping.

3.4 Coordinate-wise Domain Adaptation

The transformation function given by Equation (3.10) is a sibling form of Equation (2.4) where the essential difference is the domain of input examples. The former transforms target domains data while the latter adapts source domain data. Analogously, they have some common issues that we aim to overcome:

- As we have introduced in Section 2.6, directly optimizing Equation (3.10) is computationally unfeasible when addressing huge amount of data.

- In the case where the input space $\mathcal{X}$ contains mixed attributes, such as a mix of numerical and categorical values, defining a cost function might be difficult.

- Even in the case where the input space contains exclusively numerical attributes (*e.g.*, $\mathcal{X} \subseteq \mathbb{R}^d$), multi-dimensional distance metrics like Euclidean distance is not able to deal properly with the different scaling of each coordinate.

- As performing multi-dimensional optimal transport addresses the dependence across attributes, the estimator of the optimal transformation will have a larger variance, especially when the amount of data is insufficient.

Therefore, we should relax the classical optimal transport problem to get simpler solutions. The proposed domain adaptation method is then performed by solving a sequence of one-dimensional optimal transport method. Doing so, we consider all dimensions as independent and decompose the transformation $\mathcal{G}(\cdot)$ by *coordinate-wise* transformations $\mathcal{G}_d(\cdot)$:

$$\mathcal{G} = [\mathcal{G}_1, ..., \mathcal{G}_d, ..., \mathcal{G}_{d^{\max}}] \,, \tag{3.11}$$

where $d^{\max}$ is equal to the number of features of the input space $\mathcal{X}$.

Each elementary transformation $\mathcal{G}_d(\cdot)$ solves the Kantorovich optimization problem (Equation (3.9)) on one feature only, which is further shown to be more computationally efficient. The distance measure can also be easily defined for each specific feature, especially when each of them has a different significance. Note that this feature by feature transformation is also robust to variation of scaling. We name this adaptation method ***C**oordinate-wise **D**omain **A**daptation* (**CDA**). The next two sections detail how we process the numerical and the categorical features.

3.4.1 Numerical Feature Adaptation

When the d-th feature is numerical, the 1-D optimal transport on the real line has a closed-form solution [117] provided that the cost of moving one point to another is defined with respect to an ℓ^p norm:

$$c^p_{\text{num}}(x^d, x'^d) = |x^d - x'^d|^p \,,$$

where x^d and x'^d are scalars and stand for the d-th dimension of arbitrary examples $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. Let $\mathcal{X}_d$ represent the d-th dimension of the input space. Instead of solving $\mathcal{G}_d(\cdot)$ relying on Equation (3.9) and Equation (3.10), there is a closed-form solution of the Kantorovich optimization problem according to Section 2.6.4.:

$$\mathcal{G}_d(x^d) = ({f_{\mu^d_s}}^{-1} \circ f_{\mu^d_t})(x^d) \,. \tag{3.12}$$

From a domain adaptation perspective, $\boldsymbol{x}$ can be a target domain realization drawn from $P(X^t)$, and x^d is its d-th dimension. $f_{\mu^d_s}(\cdot)$ and $f_{\mu^d_t}(\cdot)$ are respectively cumulative distribution functions (defined by Equation (2.6)) of μ^d_s and μ^d_t. μ^d_s and μ^d_t represent respectively empirical marginal distributions of the d-th dimension

of X^s and X^t.

3.4.2 Categorical Feature Adaptation

In contrast, if the d-th feature is categorical, we have $\mathcal{X}_d = \mathbb{E}_d$, where $\mathbb{E}_d = \{e_1^d, \ldots, e_{m_d^c}^d\}$ is the (non-ordered) set of values taken by the d-th categorical feature, the so-called *levels*, and m_d^c is the number of unique values in $\mathbb{E}_d$.

We use a generic strategy that can be applied to any categorical features, by defining the cost in terms of the occurrence frequency [61]:

$$\forall e_l^d, e_r^d \in \mathbb{E}_d \times \mathbb{E}_d\,,\ c_{\text{cate}}(e_l^d, e_r^d) = C_{l,r}^d = \begin{cases} 0 \text{ if } e_l^d = e_r^d\,, \\ 1 - \dfrac{1}{1 + \log(\frac{1}{v_l^d}) \log(\frac{1}{v_r^d})} \end{cases} \quad \text{otherwise,}$$

with

$$v_l^d = \frac{\left|\{i' \mid x_{i'}^d = e_l^d, \forall i' \in \{1, \ldots, n_t + n_s\}\}\right|}{n_t + n_s}\,, \text{ and}$$
$$v_r^d = \frac{\left|\{j' \mid x_{j'}^d = e_r^d, \forall j' \in \{1, \ldots, n_t + n_s\}\}\right|}{n_t + n_s}$$

respectively representing the global frequencies of e_l^d and e_r^d in all domains. $x_{i'}^d$ and $x_{j'}^d$ are the d-th dimension of realizations $\boldsymbol{x}_{i'}\,, \boldsymbol{x}_{j'} \in \mathbb{X}^t \cup \mathbb{X}^s$.

Furthermore, the admissible set of coupling matrix $\boldsymbol{R}^d$ for the d-th categorical dimension is defined as

$$\Gamma^d = \left\{ \boldsymbol{R}^d \in \mathbb{R}_+^{m_d^c \times m_d^c} \;\middle|\; \sum_{r=1}^{m_d^c} R_{l,r}^d = v_l^{t,d} \text{ and } \sum_{l=1}^{m_d^c} R_{l,r}^d = v_r^{s,d} \right\},$$

with

$$v_l^{t,d} = \frac{\left|\{i' \mid x_{i'}^{t,d} = e_l^d, \forall i' \in \{1, \ldots, n_t\}\}\right|}{n_t}\,, \text{ and}$$
$$v_r^{s,d} = \frac{\left|\{j' \mid x_{j'}^{s,d} = e_r^d, \forall j' \in \{1, \ldots, n_s\}\}\right|}{n_s}$$

respectively representing the frequencies of e_l^d and e_r^d in each domain. $x_{i'}^{t,d}$ (Resp. $x_{j'}^{s,d}$) is the d-th dimension of realizations $\boldsymbol{x}_{i'}^t \in \mathbb{X}^t$ (Resp. $\boldsymbol{x}_{j'}^s \in \mathbb{X}^s$). Therefore, we perform the optimal transport on the m_d^c categorical values instead on the n_t target (and n_s source) examples. Typically, $m_d^c \ll n_t$ (and n_s), and the computation is thus less expensive than the original problem. However, unlike numerical features where we can compute a barycenter thanks to Equation (3.10), the barycenter of

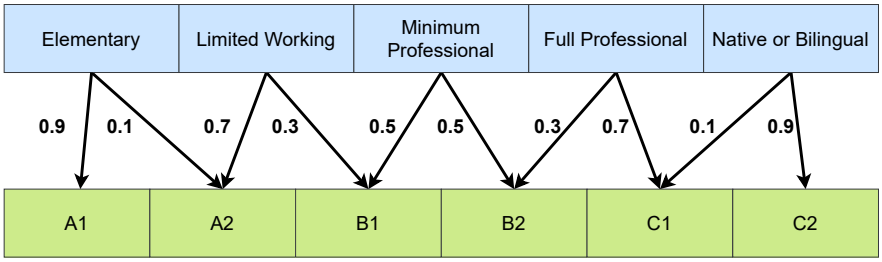


Figure 3.2: Illustration of the categorical stochastic mapping between language proficiencies on Linkedin and the European standard language levels. Values between categories indicate the probability of transformation.

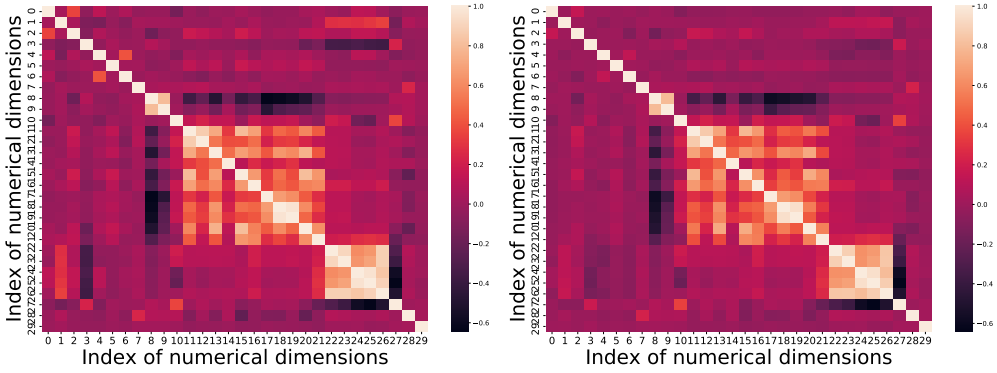


Figure 3.3: Correlation matrices of all numerical dimensions of a Worldline target domain dataset. Left: the correlation matrix before the *coordinate-wise* adaptation. Right: the correlation matrix after the *coordinate-wise* adaptation.

categorical features is difficult to define. Consequently, we propose a *stochastic mapping* strategy to tackle this problem. The probability of transforming one value e_l^d to e_r^d is

$$P(\mathcal{G}_d(e_l^d) = e_r^d) = \frac{R_{l,r}^d}{\sum_{r'=1}^{m_d^c} R_{l,r'}^d}, \tag{3.13}$$

where $\mathcal{G}_d(\cdot)$ is a *stochastic mapping* function. For an arbitrary target example $\boldsymbol{x}$ drawn from $P(X^t)$, the d-th feature x^d and its adapted value x_{adapt}^d take values in $\mathbb{E}_d$. Consequently Equation (3.13) forms a distribution of adapted $\boldsymbol{x}_{\text{adapt}}$ conditioned to $\boldsymbol{x}$ for every categorical dimension. To get the final prediction score of $\boldsymbol{x}$, we perform a Monte Carlo estimation by sampling $\boldsymbol{x}_{\text{adapt}}$ relying on Equation (3.13), and compute the average of $h_s(\boldsymbol{x}_{\text{adapt}})$ on these stochastic adapted examples.

An example of *stochastic categorical mapping* is illustrated in Figure 3.2. Supposing that the Monte Carlo sampling method generates 10 points, in this example, one point (person) having a full processional language level will probably become 3 points of B2 level and 7 points of C1 level under the European standard.

One may argue that the *coordinate-wise* transformation assumes independence

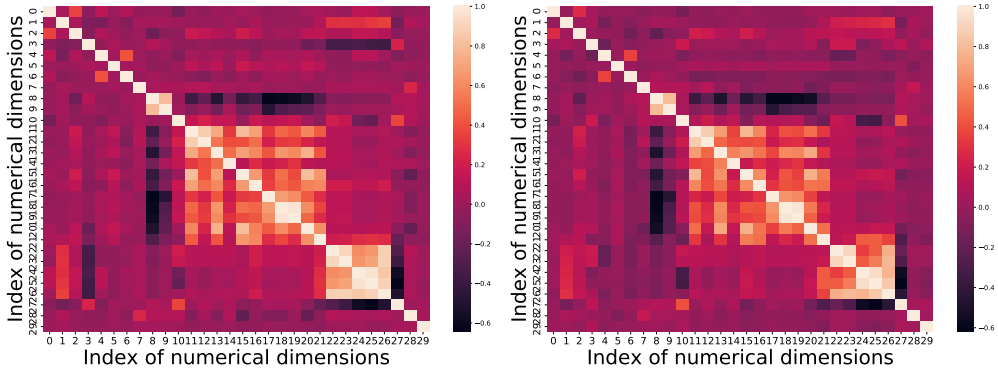


Figure 3.4: Correlation matrices of all numerical dimensions of Worldline target and source domain datasets. Left: the target domain correlation matrix. Right: the source domain correlation matrix.

between input space dimensions, which is hardly verified in real-life applications. However, the one-dimensional optimal transport does not ignore all interactions between features. Indeed, the transformation (Equation (3.12)) preserves the relative orders of features. For example, one point ranked as the second-largest in a training dataset according to the d-th feature is still the second-largest after the adaptation. Such a characteristic of the one-dimensional optimal transport guarantees all dimensions of such a point keep their ranks. We illustrate the correlation matrix of numerical dimensions of a Worldline target domain dataset in Figure 3.3. Note that *coordinate-wise* adaptations preserve most of the correlations between features of this dataset. Moreover, we note that the source domain correlation matrices are similar to that of the target domain (Figure 3.4) in Worldline domain adaptation tasks; thus, the one-dimensional optimal transport is well-suited.

3.5 Weakly Supervised Feature Selection for Domain Adaptation

We have noticed in various experiments on different domain adaptation tasks that some features contribute more to domain adaptation than others. Figure 3.5 illustrates decreasing percentages of log-loss (log-loss improvement) in a feature selection process. We presents performances of different datasets and predictive models to show that it is a common phenomenon. At initialization, no feature is adapted. Then, at each step of the process, we transform one more feature to the source domain that has the minimal value of log-loss over target data. We stop when all features are adapted. Note that we use all target labels to select the feature to transform at each step only for illustration, whereas they

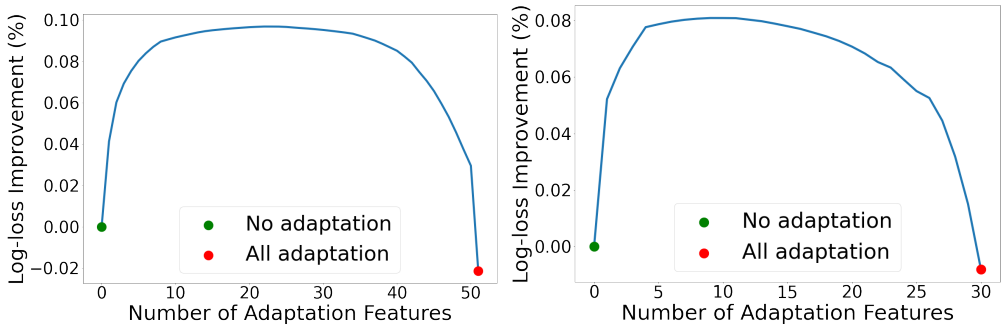


Figure 3.5: Evolution of log-loss improvements according to the number of adapted features. Left: The Kaggle fraud detection dataset with a neural network pre-trained model. Right: The real fraud detection dataset with a tree-based pre-trained model.

are not accessible in practice. Interestingly, instead of adapting all features, the adaptation of a well-selected subset of features has better performance (larger value of log-loss improvement). Therefore, in the *target to source* domain adaptation scenario where a "black-box" source model $h_s(\cdot)$ is available, we aim to seek a subset of features $\mathcal{D} \in \mathbb{D}$ to adapt, where $\mathbb{D}$ contains all possible subsets of features of the input space $\mathcal{X}$, that is,

$$\forall \mathcal{D} \in \mathbb{D} : \mathcal{D} \subseteq \mathcal{X} \,.$$

The selected features are adapted one-by-one using *coordinate-wise* optimal transport mapping functions, while other features remain identical without being excluded from the dataset. Consequently, we can use the source model directly on adapted target data to predict labels. The resulting predictive model of target domain is expressed by $h_t^{\mathcal{D}}(\cdot) = h_s \circ \mathcal{G}^{\mathcal{D}}(\cdot)$, where $\mathcal{G}^{\mathcal{D}}(\cdot)$ is the transformation function that adapts the feature subset $\mathcal{D} \in \mathbb{D}$. Empirically, we show in experiments that $\mathcal{D}$ generally contains just a few features; thus, it is very sparse. Let $\mathcal{G}^*(\cdot)$ be the transformation that verifies Equation (3.3), then the optimal target predictor is expressed by $h_t(\cdot) = h_s \circ \mathcal{G}^*(\cdot)$.

In a supervised setting, one tackles this feature selection problem by leveraging on labeled data in target domains to find the optimal subset of features that minimizes the expected risk, that is,

$$\mathcal{D}^* = \underset{\mathcal{D} \in \mathbb{D}}{\operatorname{argmin}} \underset{(\boldsymbol{x},y)\sim P(X^t,Y^t)}{\mathrm{E}} \left[\left| h_t^{\mathcal{D}}(\boldsymbol{x}) - y \right| \right] . \tag{3.14}$$

The solution $\mathcal{D}^*$ is the optimal subset of features to adapt. One may note that $\mathcal{G}^{\mathcal{D}^*}(\cdot)$ could be different from $\mathcal{G}^*(\cdot)$, as $\mathcal{G}^{\mathcal{D}^*}(\cdot)$ is restricted to the class of *coordinate-wise* transformations, whereas $\mathcal{G}^*(\cdot)$ refers to the optimal transformation among

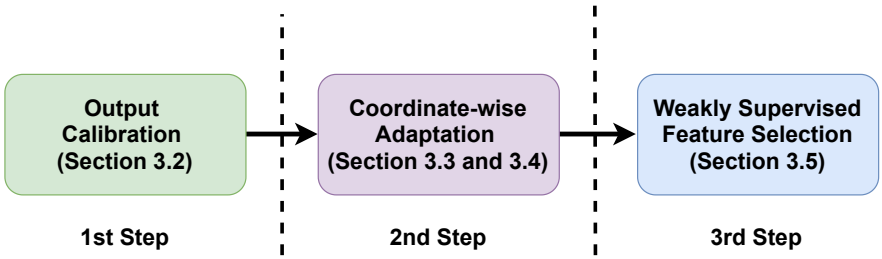


Figure 3.6: Main modules composing our proposed adaptation pipeline in a weakly supervised setting.

all possible adaptation functions.

In typical domain adaptation problems, $P(X^t, Y^t)$ is unknown, thus directly minimizing Equation (3.14) is not feasible. When few labeled target data are available, *i.e.,* in a weekly supervised setting, we propose to seek the subset of features that minimizes the following term:

$$\hat{\mathcal{D}}^* = \underset{\mathcal{D}\in\mathbb{D}}{\operatorname{argmin}} \frac{1}{n_t^l} \sum_{(\boldsymbol{x}^t, y^t)\in\mathbb{Q}_l^t} \left| h_t^{\mathcal{D}}(\boldsymbol{x}^t) - y^t \right|. \tag{3.15}$$

The target domain predictive model is given by $h_t^{\hat{\mathcal{D}}^*}(\cdot) = h_s \circ \mathcal{G}^{\hat{\mathcal{D}}^*}(\cdot)$.

3.6 Implementation

The global pipeline of our proposed transformation is illustrated in Figure 3.6. Our proposition combines an output calibration process, a *coordinate-wise* optimal transport step and a feature selection process. We name our method **WCDA** standing for ***W**eakly Supervised **C**oordinate-wise **D**omain **A**daptation.*

Output Calibration The key to the output calibration process is the estimation of source domain and target domain classes proportions. The target domain classes proportions are given by business experts, and source domain proportions can be easily estimated using source domain labels. Therefore, we do not provide implementation details of this module.

Stochastic Coordinate-wise Adaptation Let a target domain point $\boldsymbol{x} = [x^1, \ldots, x^d, \ldots, x^{d^{\max}}]$. When $\boldsymbol{x}$ contains categorical features, the adaptation function $\mathcal{G}(\cdot)$ becomes stochastic. Every time we apply $\mathcal{G}(\cdot)$ on $\boldsymbol{x}$, it would transform $\boldsymbol{x}$ to a different location. Therefore, we decide to perform the transformation n_r times and save all adapted results in a dataset $\mathbb{X}_{\text{adapt}}$. The detailed processes of CDA are presented in Algorithm 1.

Algorithm 1 Stochastic *Coordinate-wise* Adaptation

1: Initialize $\mathbb{X}_{\text{adapt}} \leftarrow \{\}$.
2: **for** t **in** $\{1, \ldots, n_r\}$ **do**
3: **for** d **in** $\{1, \ldots, d^{\max}\}$ **do**
4: Initialize $\boldsymbol{x}_{\text{adapt}} \leftarrow []$.
5: **if** x^d is numerical **then**
6: get x^d_{adapt} using Equation (3.12).
7: **else**
8: set $e^d_l \leftarrow x^d$ in Equation (3.13).
9: sample one e^d_r.
10: set $x^d_{\text{adapt}} \leftarrow e^d_r$.
11: **end if**
12: $\boldsymbol{x}_{\text{adapt}} \leftarrow [\boldsymbol{x}_{\text{adapt}}, x^d_{\text{adapt}}]$.
13: **end for**
14: add $\boldsymbol{x}_{\text{adapt}}$ to $\mathbb{X}_{\text{adapt}}$.
15: **end for**
16: **return:** $\mathbb{X}_{\text{adapt}}$.

The first loop (step 2) adapts the target example n_r times, and put them into $\mathbb{X}_{\text{adapt}}$. The second loop (step 3) performs *coordinate-wise* adaptation over each dimension of $\boldsymbol{x}$. We use the POT [41] package to compute the optimal transport mapping (steps 6 and 9). The prediction of the point is given by

$$\frac{1}{n_r} \sum_{\boldsymbol{x}_{\text{adapt}} \in \mathbb{X}_{\text{adapt}}} h_s(\boldsymbol{x}_{\text{adapt}}) \,. \tag{3.16}$$

Weakly Supervised Feature Selection $\boldsymbol{X}^t_l$ is a matrix where rows correspond to input examples in $\mathbb{Q}^t_l$, and $\boldsymbol{y}^t_l$ is the associated output vector. We also define their n_r-times repeated version $\boldsymbol{X}^t_{\text{repeat}} = \left[\boldsymbol{X}^t_l; \ldots; \boldsymbol{X}^t_l\right]$ and $\boldsymbol{y}^t_{\text{repeat}} = [\boldsymbol{y}^t_l, \ldots, \boldsymbol{y}^t_l]$ respectively. We apply $\mathcal{G}(\cdot)$ on each row of $\boldsymbol{X}^t_{\text{repeat}}$ and we obtain $\boldsymbol{X}^t_{\text{adapt}}$. Of note, $\mathcal{G}(\cdot)$ is a stochastic transformation; hence repetitions of the same row can be transformed to different locations.

We also introduce some basic selection operations of a matrix and a vector. Taking an arbitrary matrix $\boldsymbol{X}$ as an example, $\boldsymbol{X}_{a,b}$ selects a submatrix of $\boldsymbol{X}$. a and b respectively correspond to the index of rows and columns, and they can be scalars or sets of scalar values. When they are scalars, the corresponding row (column) is selected. When they are sets, all rows (columns) corresponding to indexes in the set are selected. Moreover, we use the ":" symbol to select all dimensions. Following the same convention, we let $\boldsymbol{y}_a$ gives a "subvector" of $\boldsymbol{y}$. a can be a set or a scalar value.

As we have introduced in Section 2.4.3, testing all combinations of feature sets is computationally expensive. Alternatively, we propose to use a greedy

search algorithm. Algorithm 2 illustrates the iterative process to find the subset of features to adapt in a weakly supervised scenario. We stop the process when adapting more features does not improve the adaptation performance on weakly labeled data. Such an iterative loop starts from step 3 and ends at step 17. Moreover, we apply a Bootstrap [15] strategy (step 5) to get a more robust feature subset. We repeat several times the feature selection step (steps 7 to 11) and count the selection frequency of each feature (step 12). Then we add the most frequent feature to the feature subset $\hat{\mathcal{D}}$ (steps 14, 15). If no feature is selected or less than one-half of Bootstrap results agree with the selected result, we stop the iteration process and return the subset of features (steps 17, 18).

Algorithm 2 Weakly Supervised Greedy Search Algorithm

1: Initialize $i \leftarrow 0$
2: Initialize $\hat{\mathcal{D}}^{(i)} \leftarrow \{\}$
3: **repeat**
4: Initialize $\text{Count}[d] \leftarrow 0\,, \forall d \in \mathcal{X}/\hat{\mathcal{D}}^{(i)} \cup \{\varnothing\}$
5: **for** $\mathbb{I}$ **in** Bootstraps of row index of $\boldsymbol{X}^t_{\text{repeat}}$ **do**
6: Initialize $\text{Loss}[d]$ by an empty key-value dictionary
7: **for** d **in** $\mathcal{X}/\hat{\mathcal{D}}^{(i)} \cup \{\varnothing\}$ **do**
8: $\boldsymbol{X}_{\text{boot}} \leftarrow \boldsymbol{X}^t_{\text{repeat}_{\mathbb{I},:}}$; $\boldsymbol{X}_{\text{boot}_{\mathbb{I},\hat{\mathcal{D}}^{(i)}\cup\{d\}}} \leftarrow \boldsymbol{X}^t_{\text{adapt}_{\mathbb{I},\hat{\mathcal{D}}^{(i)}\cup\{d\}}}$
9: $\text{Loss}[d] \leftarrow \sum_{j\in\mathbb{I}} \left| h_s(\boldsymbol{X}_{\text{boot}\,j,:}) - \boldsymbol{y}_j \right|$
10: **end for**
11: $d_{\min} \leftarrow \text{argmin}_d \,\text{Loss}[d]$
12: $\text{Count}[d_{\min}] \leftarrow \text{Count}[d_{\min}] + 1$
13: **end for**
14: $d^* \leftarrow \text{argmax}_d \,\text{Count}[d]$
15: $\hat{\mathcal{D}}^{(i+1)} \leftarrow \hat{\mathcal{D}}^{(i)} \cup \{d^*\}$
16: $v \leftarrow \text{Count}[d^*]/\sum_d \text{Count}[d]$
17: **until** $d^* \leftarrow \varnothing$ or $v < 0.5$; $i \leftarrow i + 1$
18: **return:** $\hat{\mathcal{D}}^{(i)}$

3.7 Experiments

In this section, we evaluate the performances of our adaptation methods CDA and WCDA on 3 different datasets. Details of datasets and separations of source and target domains are provided in Section 2.8.

3.7.1 General Setup

We are given two families of pre-trained source domain models, GBDT and neural networks (NN). The models are implemented respectively in Python using LightGBM [63] and PyTorch [113] packages. The source domain predictive models are trained following a supervised learning paradigm with 10 different random

states. The one that achieves the best performance on source domain testing datasets is given as the pre-trained models.

Adaptation Methods of Comparison We compare our proposed methods (CDA and WCDA) with *deep adaptation methods*: DAN [91], DANN [44], and MCD [129], as well as a *classical adaptation method*: CORAL [143]. As described in Section 3.5, WCDA selects features based on *coordinate-wise* transformations. Whereas CDA skips the weakly supervised feature selection process (Section 3.4). We chose to compare with CORAL as it performs domain adaptations without modifying the input space of data. As a result, we can extend such methods to address *target to source* domain adaptations and leverage pre-trained GBDT models and NN models to predict target labels. However, it transforms only numerical features, while categorical dimensions remain unadapted. In contrast, *deep adaptation methods* can transform categorical attributes, whereas they do not satisfy the *target to source* domain adaptation setting. *Deep adaptation methods* transform source and target domain data into a latent space and require training a predictive model using source labels during adaptation processes. Nonetheless, we report performances of *deep adaptation methods* as they are widely used in current domain adaptation tasks.

Hyper-parameters of NN Models *Deep adaptation methods* like DANN and DAN require to find the optimal weight of the adversarial (regularization) term, and MCD requires fine-tuning the learning rate. To select the optimal hyper-parameter for *deep adaptation methods*, we use a grid search process during the training and take the hyper-parameter that minimizes the classification error on test datasets of source domains. For DANN and DAN methods of the Amazon reviews datasets, we seek this weight in the set of values $\{0.01, 0.05, 0.1\}$. For the Kaggle fraud detection dataset, the set of values that we used to search the hyper-parameter is $\{0.005, 0.01, 0.1\}$ for DAN models, and $\{0.05, 0.1, 0.5\}$ for DANN models. As for MCD models, the Amazon review tasks seek the learning rate among $\{0.0001, 0.0005, 0.001, 0.005\}$, and the Kaggle fraud detection tasks seek the learning rate among $\{0.0005, 0.0007, 0.001\}$.

Weakly Supervised and Unsupervised Setting We compare CDA and WCDA with *deep adaptation methods* in both weakly supervised and unsupervised settings. We annotate respectively 100, 200, and 5000 labeled examples of target domains of the Amazon reviews, Kaggle, and real fraud detection datasets. The original *deep adaptation methods* do not have a weakly supervised version. However, as they integrate a retraining process during the adaptation, we extend

the unsupervised *deep adaptation methods* to weakly supervised scenarios by using labeled target domain data at the retraining step. We also compare our propositions with the FineTune [126] method, where the last layer of pre-trained NN models is adjusted to fit weakly labeled target data. Since CORAL verifies the *target to source* adaptation scenario and does not have a retraining process, we compare CDA with CORAL only in unsupervised settings.

Evaluation Metrics For the Kaggle and Worldline fraud detection tasks, we evaluate predictive models based on the area under the precision-recall curve (PR-AUC) and log-loss. For the Amazon review task, we report performances in accuracy and log-loss, as classes of this dataset are balanced. For the sake of comparison, we set the average performance of source domain NN baseline models as references and report the percentage of improvements compared to such models. For PR-AUC and accuracy metric, the percentage of improvements is the rate of increase, while for the log-loss metric, it is the rate of decrease. All experiments are repeated ten times with different random states. Besides improvements of prediction performances, we also report standard deviations to illustrate the stabilities of adaptation methods.

3.7.2 Adaptation Performance Analysis

3.7.2.1 Kaggle Dataset

Table 3.1 reports adaptation performances in an unsupervised setting. NN-CDA and LGB-CDA respectively stand for our domain adaptation method (without the feature selection step) on two pre-trained models. LGB is a shorthand of the LightGBM package representing the GBDT model. Note that LGB Baseline models have positive values of performance improvements; that is, pre-trained GBDT baseline models without adaptation outperform pre-trained NN models on target domain data.

NN-CDA methods have achieved the best adaptation results on several tasks, such as “D-2 to M” with pre-trained NN and GBDT models and “D-3 to M” with pre-trained GBDT models, in both log-loss metric and PR-AUC metric. However, it does not have a good performance compared to deep adaptation methods when using NN models. DAN achieves the best adaptation performance in PR-AUC and DANN outperforms all the others in log-loss. LGB-CDA slightly decreases performances of LGB Baseline in terms of PR-AUC on average. Whereas it outperforms LGB Baseline in terms of log-loss in all experiments. Occasionally, LGB-CDA decreases performance of baseline model like the case “D-1 to M” of the NN model. We explain such phenomena by the fact that the adaptation of

some dimensions may decrease adaptation performances (Figure 3.5). A feature selection step is essential to overcome this issue.

Table 3.2 shows the adaptation results in a weakly supervised setting. Our proposed pipeline WCDA has achieved the best performance over tasks with pre-trained LGB models. It outperforms all other methods on average with the pre-trained NN model in log-loss metric. However, in terms of PR-AUC, NN-WCDA performs no better than DAN and FineTune on average.

Table 3.1: Adaptation performances of **unsupervised** adaptation methods over **Kaggle** datasets (see Section 2.8.2 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Method	D-1 to M	D-2 to M	D-3 to M	AVG
DAN	**12.31±6.72**	-3.20±7.18	**1.41±3.63**	**3.51**
DANN	3.47±9.53	-2.90±3.82	-4.21±6.51	-1.21
MCD	-11.47±13.38	-6.29±6.09	-6.81±4.83	-8.19
CORAL	9.11±0.26	-2.47±0.08	-8.37±0.14	-0.58
NN-CDA (**ours**)	-8.15±2.51	**3.87±0.35**	0.69±0.38	-1.20

(a) Improvements of PR-AUC of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
DAN	29.57±4.32	-1.58±9.64	4.37±5.39	10.79
DANN	29.58±4.05	2.39±3.98	2.06±5.07	**11.34**
MCD	21.59±9.25	-1.29±5.34	-7.55±10.32	4.25
CORAL	**30.66±0.07**	4.69±0.08	-2.80±0.13	10.85
NN-CDA (**ours**)	18.38±0.62	**6.32±0.34**	**7.40±0.28**	10.70

(b) Improvements of log-loss of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB Baseline	**26.13±1.53**	4.18±1.47	7.58±3.26	**12.63**
CORAL	17.19±0.61	-8.23±0.26	-8.50±0.51	0.15
LGB-CDA (**ours**)	22.55±1.15	**5.24±0.53**	**8.95±0.55**	12.25

(c) Improvements of PR-AUC of LGB predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB Baseline	9.77±6.01	6.53±3.31	12.32±5.05	9.54
CORAL	**33.75±0.26**	-4.67±0.63	-7.76±1.19	7.11
LGB-CDA (**ours**)	32.59±0.54	**8.66±0.56**	**14.10±0.42**	**18.45**

(d) Improvements of log-loss of LGB predictive models.

Table 3.2: Adaptation performances of **weakly supervised** adaptation methods over **Kaggle** datasets (see Section 2.8.2 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%). Unsupervised CDAs are also reported to simplify comparisons.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
DAN	**15.40±6.62**	-0.57±4.53	1.53±3.68	**5.45**
DANN	5.77±10.49	-0.87±3.82	-2.40±4.95	0.83
MCD	8.96±12.36	-2.48±5.03	-2.89±4.84	1.20
FineTune	4.23±0.72	2.12±0.15	**4.95±0.30**	3.76
NN-CDA(unsup) (**ours**)	-8.15±2.51	**3.87±0.35**	0.69±0.38	-1.20
NN-WCDA (**ours**)	1.30±7.57	2.98±2.14	3.72±1.40	2.66

(a) Improvements of PR-AUC of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
DAN	**31.70±3.58**	3.02±5.34	5.94±5.52	13.55
DANN	31.03±4.33	3.16±4.25	5.78±4.62	13.32
MCD	30.28±3.59	1.26±5.39	3.74±5.45	11.76
FineTune	9.22±2.10	5.33±0.83	11.23±1.13	8.59
NN-CDA(unsup) (**ours**)	18.38±0.62	6.32±0.34	7.40±0.28	10.70
NN-WCDA (**ours**)	30.42±3.26	**8.95±2.67**	**12.07±2.04**	**17.14**

(b) Improvements of log-loss of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB Baseline	26.13±1.53	4.18±1.47	7.58±3.26	12.63
LGB-CDA(unsup) (**ours**)	22.55±1.15	5.24±0.53	8.95±0.55	12.25
LGB-WCDA (**ours**)	**27.85±4.41**	**7.18±1.85**	**13.66±1.56**	**16.23**

(c) Improvements of PR-AUC of LGB predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB Baseline	9.77±6.01	6.53±3.31	12.32±5.05	9.54
LGB-CDA(unsup) (**ours**)	32.59±0.54	8.66±0.56	14.10±0.42	18.45
LGB-WCDA (**ours**)	**39.86±1.68**	**14.31±2.40**	**22.06±1.66**	**25.41**

(d) Improvements of log-loss of LGB predictive models.

Table 3.3: Numbers of adapted features of Kaggle fraud detection tasks in a weakly supervised setting.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
NN-WCDA (**ours**)	11.3±2.4	13.6±3.4	15.7±3.7	13.5
LGB-WCDA (**ours**)	15.3±4.3	18.5±4.9	17.5±3.1	17.1

There is a significant difference between NN-WCDA and DAN for the adaptation task "D-1 to M", which may be due to the fact that the pipeline without the feature selection process (NN-CDA) decreases the performance of this task. Moreover, the weakly supervised feature selection process based on 200 labeled target domain examples cannot mitigate such a decrease.

Table 3.3 reports the number of adapted features when using the entire pipeline. We see that the adapted number of features is around 1/3 of the whole input space and LGB-WCDA generally adapts fewer features than NN-WCDA.

Table 3.4: Adaptation performances of **unsupervised** adaptation methods over **Worldline** datasets (see Section 2.8.1 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Method	G-1 to B	G-2 to B	G-3 to B	AVG
DAN	7.31±5.78	5.47±3.96	**10.01±4.70**	**7.60**
DANN	4.38±5.09	6.43±2.69	5.28±3.03	5.37
MCD	6.46±5.70	1.41±8.19	6.84±4.94	4.91
CORAL	**8.31±0.02**	3.83±0.02	3.54±0.02	5.23
NN-CDA (**ours**)	3.86±1.88	**7.93±1.12**	8.87±1.66	6.89

(a) Improvements of PR-AUC of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
DAN	4.40±2.75	9.12±2.60	17.90±3.91	**10.48**
DANN	2.85±2.37	**9.78±1.33**	16.45±2.98	9.69
MCD	4.40±1.57	7.55±3.94	14.16±7.35	8.70
CORAL	**5.45±0.00**	9.43±0.01	14.07±0.01	9.65
NN-CDA (**ours**)	4.11±0.52	8.13±0.34	**18.92±0.27**	10.39

(b) Improvements of log-loss of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB Baseline	**9.91±6.43**	3.59±6.57	-1.94±8.31	3.85
CORAL	0.80±0.13	-8.46±0.16	-10.83±0.29	-6.17
LGB-CDA (**ours**)	8.96±1.98	**10.25±0.86**	**9.76±0.97**	**9.65**

(c) Improvements of PR-AUC of LGB predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB Baseline	5.02±3.54	4.38±4.00	9.43±4.65	6.28
CORAL	1.05±0.03	2.74±0.08	11.40±0.05	5.06
LGB-CDA (**ours**)	**5.35±0.46**	**7.64±0.30**	**17.17±0.59**	**10.05**

(d) Improvements of log-loss of LGB predictive models.

Table 3.5: Adaptation performances of **weakly supervised** adaptation methods over **Worldline** datasets (see Section 2.8.1 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%). Unsupervised CDAs are also reported to simplify comparisons.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
DAN	9.40±3.50	**11.85±1.86**	8.85±1.73	10.03
DANN	9.56±4.54	10.27±5.78	10.46±6.94	10.10
MCD	1.64±14.30	1.77±7.82	12.71±21.75	5.38
FineTune	8.04±1.54	10.91±1.50	5.32±0.63	8.09
NN-CDA(unsup) (**ours**)	3.86±1.88	7.93±1.12	8.87±1.66	6.89
NN-WCDA (**ours**)	**11.75±3.14**	8.89±5.58	**13.62±9.46**	**11.42**

(a) Improvements of PR-AUC of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
DAN	**8.05±1.04**	**12.66±0.88**	18.83±2.07	13.18
DANN	4.41±3.58	10.99±3.05	27.49±4.75	14.30
MCD	-12.28±18.49	-8.05±22.62	8.31±22.13	-4.01
FineTune	6.38±0.40	11.63±0.23	13.99±1.69	10.67
NN-CDA(unsup) (**ours**)	4.11±0.52	8.13±0.34	18.92±0.27	10.39
NN-WCDA (**ours**)	5.23±1.59	9.86±2.14	**28.50±2.30**	**14.53**

(b) Improvements of log-loss of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB Baseline	9.91±6.43	3.59±6.57	-1.94±8.31	3.85
LGB-CDA(unsup) (**ours**)	8.96±1.98	10.25±0.86	9.76±0.97	9.65
LGB-WCDA (**ours**)	**22.65±5.14**	**17.84±4.38**	**15.21±5.65**	**18.56**

(c) Improvements of PR-AUC of LGB predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB Baseline	5.02±3.54	4.38±4.00	9.43±4.65	6.28
LGB-CDA(unsup) (**ours**)	5.35±0.46	7.64±0.30	17.17±0.59	10.05
LGB-WCDA (**ours**)	**11.79±0.64**	**16.59±1.70**	**24.84±2.81**	**17.74**

(d) Improvements of log-loss of LGB predictive models.

Table 3.6: Numbers of adapted features of Worldline fraud detection tasks in a weakly supervised setting.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
NN-WCDA (**ours**)	7.0±2.4	5.1±1.7	7.8±2.3	6.6
LGB-WCDA (**ours**)	7.9±2.4	8.1±1.4	8.0±0.5	8.0

3.7.2.2 Worldline Dataset

Table 3.4 reports adaptation performances of Worldline fraud detection tasks in an unsupervised setting. One observes a similar phenomenon as the Kaggle unsupervised adaptation tasks. Additionally, in both Kaggle and Worldline fraud detection tasks, deep adaptation methods generally have a larger variance than classical adaptation methods including the ours.

In a weakly supervised setting (Table 3.5), WCDA outperforms all other methods on average in all metrics. One can also see that WCDA improves CDA in all Worldline domain adaptation tasks. The observation proves the efficiency of weakly supervised feature selection in domain adaptation tasks. Note that, although some deep adaptation methods have good performances over several adaptation tasks, they do not fit a *target to source* domain adaptation scenario and cannot be applied to our problem in real-life applications.

Concerning the number of adapted features (Table 3.6), similar to the results of Kaggle adaptation tasks (Table 3.3), we improve CDA by adapting 1/4 of dimensions.

Table 3.7: Adaptation performances of **NN unsupervised** adaptation methods over **Amazon datasets** (see Section 2.8.3 for the dataset description). We report percentages of performances improvements, in accuracy and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Task	DAN	DANN	MCD	CORAL	NN-CDA (**ours**)
B to D	**-0.91±0.40**	-3.42±1.31	-2.41±2.13	-0.22±0.00	**-0.91±0.03**
B to E	**6.12±0.58**	4.28±2.59	4.24±1.37	3.97±0.00	5.66±0.04
B to K	6.31±1.40	8.20±3.13	6.73±2.88	10.47±0.00	**10.65±0.04**
D to B	-0.90±0.55	-1.67±0.81	-0.28±0.42	**-0.11±0.00**	-0.28±0.04
D to E	**3.97±0.23**	0.45±1.68	-0.73±2.47	2.18±0.00	3.28±0.04
D to K	**7.20±0.27**	1.74±2.18	-0.02±3.96	6.85±0.00	7.09±0.02
E to B	0.51±1.05	-3.03±1.59	-2.83±1.36	-0.71±0.00	**1.55±0.02**
E to D	-0.05±0.38	-2.18±1.42	-1.53±1.12	0.14±0.00	**0.96±0.02**
E to K	0.14±1.88	0.98±0.73	1.91±0.57	2.49±0.00	**2.55±0.03**
K to B	-0.07±0.48	-0.27±0.50	0.34±0.34	0.45±0.00	**1.06±0.02**
K to D	-1.05±0.45	1.83±0.75	2.23±0.56	1.51±0.00	**2.45±0.03**
K to E	-1.17±0.70	-1.47±0.75	-1.45±1.37	-1.22±0.00	**-0.34±0.02**
avg	1.68	0.45	0.52	2.15	**2.81**

(a) Improvements of accuracy of NN predictive models.

Task	DAN	DANN	MCD	CORAL	NN-CDA (**ours**)
B to D	-0.92±0.30	-4.47±1.14	-2.66±2.64	0.84±0.00	**0.87±0.01**
B to E	**9.64±0.60**	5.72±2.35	6.62±1.96	6.86±0.00	8.78±0.00
B to K	10.37±1.04	11.18±2.56	10.28±3.67	14.12±0.00	**16.27±0.04**
D to B	-0.75±0.46	-1.72±0.77	-0.29±0.55	-0.25±0.00	**-0.21±0.04**
D to E	4.35±0.50	0.19±1.44	0.15±2.72	2.77±0.00	**5.36±0.04**
D to K	7.57±0.50	1.36±1.88	-0.07±4.31	6.40±0.00	**9.23±0.02**
E to B	**2.29±1.46**	-5.45±4.40	-8.01±3.90	-4.60±0.00	2.21±0.00
E to D	-0.11±0.75	-2.90±2.48	-3.91±2.50	-4.67±0.00	**0.64±0.03**
E to K	0.57±3.44	1.04±1.24	3.30±0.97	5.35±0.00	**6.44±0.03**
K to B	-1.24±1.11	1.14±0.87	1.60±0.76	-2.79±0.00	**1.84±0.02**
K to D	-2.64±0.95	1.80±0.94	2.44±0.58	-2.38±0.00	**2.80±0.01**
K to E	-0.96±1.22	-3.84±1.13	-2.28±2.24	-1.63±0.00	**-0.53±0.03**
avg	2.35	0.34	0.60	1.67	**4.47**

(b) Improvements of log-loss of NN predictive models.

Table 3.8: Adaptation performances of **LGB unsupervised** adaptation methods over **Amazon datasets** (see Section 2.8.3 for the dataset description). We report percentages of performances improvements, in accuracy and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Task	LGB Baseline	CORAL	LGB-CDA (**ours**)
B to D	-2.86±0.61	-3.90±0.00	**-2.61±0.06**
B to E	**5.06±0.70**	0.86±0.00	4.84±0.05
B to K	7.75±0.70	6.63±0.00	**8.82±0.03**
D to B	-1.19±0.28	**0.43±0.00**	-0.04±0.09
D to E	-1.81±1.47	-0.30±0.00	**2.23±0.06**
D to K	-0.99±1.45	3.95±0.00	**5.40±0.05**
E to B	-0.04±0.30	-0.52±0.00	**-0.01±0.04**
E to D	**-0.22±0.32**	-1.20±0.00	-0.28±0.08
E to K	0.28±0.35	0.99±0.00	**1.48±0.05**
K to B	-2.43±0.38	-0.20±0.00	**0.12±0.03**
K to D	1.72±0.38	**1.92±0.00**	1.85±0.04
K to E	-2.58±0.18	**-0.90±0.00**	-1.28±0.02
avg	0.22	0.65	**1.71**

(a) Improvements of accuracy of LGB predictive models.

Task	LGB Baseline	CORAL	LGB-CDA (**ours**)
B to D	-4.24±4.55	-3.08±0.00	**-1.97±0.02**
B to E	1.14±5.99	2.77±0.00	**5.58±0.01**
B to K	6.35±5.46	9.02±0.00	**11.75±0.02**
D to B	-3.13±0.33	-2.83±0.00	**-1.59±0.04**
D to E	-2.19±0.81	0.17±0.00	**3.30±0.03**
D to K	-0.68±0.55	4.63±0.00	**6.89±0.03**
E to B	1.57±0.70	-0.46±0.00	**2.07±0.01**
E to D	-0.54±1.00	-1.91±0.00	**0.74±0.02**
E to K	-0.67±2.73	3.44±0.00	**4.83±0.02**
K to B	-0.22±0.41	1.10±0.00	**2.38±0.04**
K to D	1.27±0.35	0.98±0.00	**2.50±0.02**
K to E	-6.55±1.49	-1.41±0.00	**-1.50±0.01**
avg	-0.66	1.04	**2.91**

(b) Improvements of log-loss of LGB predictive models.

Table 3.9: Adaptation performances of **NN weakly supervised** adaptation methods over **Amazon datasets** (see Section 2.8.3 for the dataset description). We report percentages of performances improvements, in accuracy and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%). Unsupervised CDAs are also reported to simplify comparison s.

Task	DAN	DANN	MCD	FineTune	NN-CDA (unsup) (**ours**)	NN-WCDA (**ours**)
B to D	-0.97±0.38	-0.73±0.56	-0.71±0.66	-0.85±1.29	-0.91±0.03	**0.61±0.10**
B to E	6.28±0.52	6.86±0.91	6.85±0.85	4.55±0.43	5.66±0.04	**6.83±0.02**
B to K	11.02±0.48	**11.75±0.53**	11.68±0.65	9.31±0.18	10.65±0.04	11.64±0.02
D to B	-1.40±0.41	-0.12±1.04	-0.26±1.20	-1.24±1.12	-0.28±0.04	**0.25±0.10**
D to E	**4.01±0.23**	2.93±1.08	3.28±0.68	0.99±0.66	3.28±0.04	3.46±0.05
D to K	7.26±0.24	**7.85±0.37**	7.87±0.42	5.32±0.25	7.09±0.02	7.51±0.03
E to B	0.86±0.41	1.38±1.09	1.25±1.20	0.05±0.50	1.55±0.02	**1.73±0.10**
E to D	0.27±0.16	0.05±0.63	0.31±0.49	-0.47±0.35	0.96±0.02	**1.06±0.10**
E to K	0.47±0.77	1.77±0.40	1.35±0.96	**2.89±0.34**	2.55±0.03	2.82±0.01
K to B	0.35±0.38	1.33±0.82	1.44±1.15	-0.57±1.02	1.06±0.02	**1.72±0.07**
K to D	0.17±0.53	2.30±1.05	2.67±0.76	2.85±0.64	2.45±0.03	**3.16±0.06**
K to E	-0.70±0.15	-0.92±0.50	-1.14±1.20	-1.07±0.74	-0.34±0.02	**0.41±0.02**
avg	2.30	2.87	2.88	1.81	2.81	**3.43**

(a) Improvements of accuracy of NN predictive models.

Task	DAN	DANN	MCD	FineTune	NN-CDA (unsup) (**ours**)	NN-WCDA (**ours**)
B to D	-0.84±0.37	-1.06±0.69	-0.88±0.68	-0.22±1.05	0.87±0.01	**1.48±0.01**
B to E	9.77±0.36	10.78±1.16	10.37±1.58	7.95±0.39	8.78±0.00	**11.33±0.01**
B to K	16.12±0.53	17.29±0.62	17.31±0.78	14.66±0.29	16.27±0.00	**18.51±0.00**
D to B	-1.23±0.39	0.12±0.99	0.12±1.06	-1.00±1.25	-0.21±0.00	**0.59±0.01**
D to E	4.69±0.52	6.79±0.50	**6.96±1.07**	2.79±0.49	5.36±0.00	5.85±0.00
D to K	7.72±0.49	10.84±0.60	**11.15±0.81**	6.39±0.29	9.23±0.00	9.80±0.00
E to B	3.08±0.34	4.08±1.43	4.15±1.25	-0.54±2.00	2.21±0.00	**4.75±0.02**
E to D	0.64±0.37	1.25±0.51	1.30±0.63	-1.17±1.32	0.64±0.00	**2.11±0.01**
E to K	1.12±1.30	3.46±0.83	2.45±1.80	5.92±0.61	6.44±0.00	**6.99±0.00**
K to B	1.85±0.50	3.90±0.98	4.21±1.04	-0.20±2.30	1.84±0.00	**4.22±0.03**
K to D	-0.09±0.87	3.40±0.90	3.52±0.54	2.75±0.99	2.80±0.01	**4.80±0.05**
K to E	-0.64±0.50	-1.67±1.03	-2.64±1.74	-0.85±1.02	-0.53±0.00	**1.39±0.00**
avg	3.52	4.93	4.84	3.04	4.47	**5.99**

(b) Improvements of log-loss of NN predictive models.

Table 3.10: Adaptation performances of **LGB weakly supervised** adaptation methods over **Amazon datasets** (see Section 2.8.3 for the dataset description). We report percentages of performances improvements, in accuracy and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%). Unsupervised CDAs are also reported to simplify comparisons.

Task	LGB Baseline	LGB-CDA (unsup) (**ours**)	LGB-WCDA (**ours**)
B to D	-2.86±0.61	-2.61±0.06	**-1.54±0.36**
B to E	5.06±0.70	4.84±0.05	**5.58±0.02**
B to K	7.75±0.70	8.82±0.03	**9.73±0.04**
D to B	-1.19±0.28	-0.04±0.09	**0.46±0.11**
D to E	-1.81±1.47	2.23±0.06	**3.10±0.03**
D to K	-0.99±1.45	5.40±0.05	**7.43±0.09**
E to B	-0.04±0.30	-0.01±0.04	**1.00±0.13**
E to D	-0.22±0.32	-0.28±0.08	**0.42±0.06**
E to K	0.28±0.35	1.48±0.05	**1.92±0.03**
K to B	-2.43±0.38	**0.12±0.03**	-0.23±0.06
K to D	1.72±0.38	1.85±0.04	**3.16±0.05**
K to E	-2.58±0.18	-1.28±0.02	**-1.02±0.02**
avg	0.22	1.71	**2.50**

(a) Improvements of accuracy of LGB predictive models.

Task	LGB Baseline	LGB-CDA (unsup) (**ours**)	LGB-WCDA (**ours**)
B to D	-4.24±4.55	-1.97±0.02	**-1.27±0.19**
B to E	1.14±5.99	5.58±0.01	**8.50±0.02**
B to K	6.35±5.46	11.75±0.02	**14.79±0.02**
D to B	-3.13±0.33	-1.59±0.04	**-0.91±0.08**
D to E	-2.19±0.81	3.30±0.03	**7.11±0.02**
D to K	-0.68±0.55	6.89±0.03	**10.89±0.04**
E to B	1.57±0.70	2.07±0.01	**4.14±0.05**
E to D	-0.54±1.00	0.74±0.02	**2.08±0.04**
E to K	-0.67±2.73	4.83±0.02	**5.11±0.03**
K to B	-0.22±0.41	2.38±0.04	**3.08±0.05**
K to D	1.27±0.35	2.50±0.02	**4.05±0.01**
K to E	-6.55±1.49	-1.50±0.01	**-1.04±0.05**
avg	-0.66	2.91	**4.71**

(b) Improvements of log-loss of LGB predictive models.

Table 3.11: Numbers of adapted features of Amazon review tasks in a weakly supervised setting.

Task	NN-WCDA	LGB-WCDA
B to D	212.6±3.9	313.0±2.0
B to E	227.2±2.2	322.6±0.5
B to K	208.8±1.9	324.2±0.7
D to B	175.0±2.6	264.8±1.7
D to E	249.6±0.8	264.8±0.7
D to K	259.2±2.6	281.4±2.0
E to B	194.2±1.7	317.0±0.9
E to D	202.6±0.8	305.6±0.8
E to K	209.4±1.0	314.2±2.1
K to B	200.2±3.9	327.4±1.4
K to D	215.4±3.2	304.8±0.4
K to E	186.2±3.1	335.2±1.7
AVG	211.7	306.3

3.7.2.3 Amazon Dataset

Tables 3.7 and 3.8 report adaptation performances of Amazon review tasks in an unsupervised setting. CDA without the feature selection step outperforms all other methods in most adaptation tasks in different evaluation metrics. Unlike fraud detection tasks where input spaces are mixed types, the Amazon review datasets only contain numerical values. Furthermore, we observe that adaptation methods based on GBDT pre-trained models are no better than NN pre-trained models.

When using weakly labeled target domain data, both NN-WCDA and LGB-WCDA achieve the best performance on average (Tables 3.9 and 3.10). Deep adaptation methods like DANN and MCD have the best adaptation performances in some sub-tasks. LGB-WCDA outperforms LGB-CDA in all tasks except the "K to B" task in accuracy metric.

Table 3.11 represents the number of adapted features. The average number of adapted features is around 211 for NN-WCDA and around 306 for LGB-WCDA, respectively, representing 1/2 and 3/4 of the total number of input space dimensions. We see that the Amazon review task has a proportion of selected features larger than Worldline and Kaggle fraud detection tasks. This phenomenon may be related to the high feature correlation of the input space (Figure 3.7). Indeed, the feature spaces of the Worldline and Kaggle fraud detection dataset are designed by business experts. Correlated dimensions are removed to reduce redundances. In contrast, features of Amazon review tasks are generated automatically by mSDA; thus, a high correlation may exist between attributes.

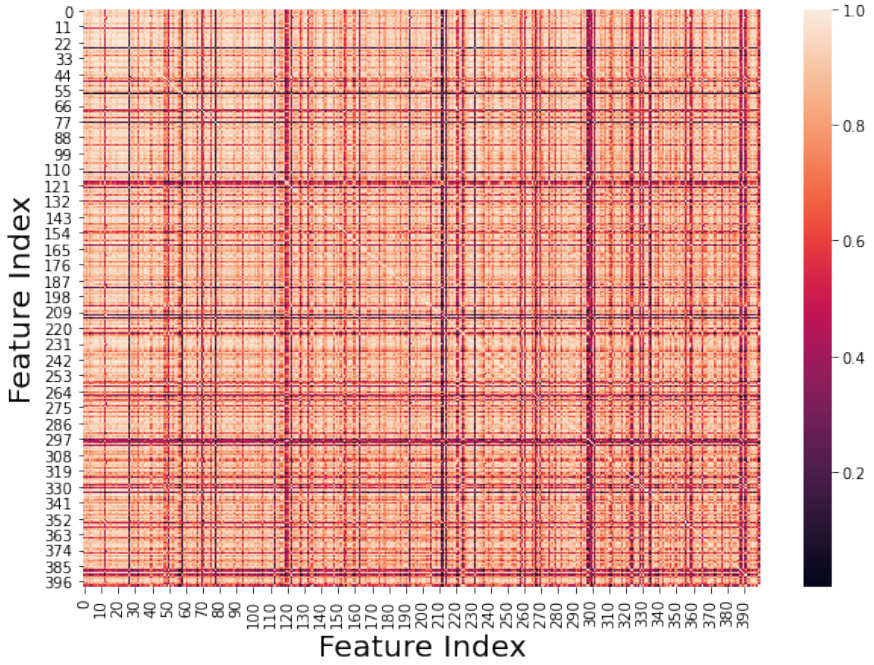


Figure 3.7: Absolute values of the correlation matrix of mSDA representations of Amazon reviews dataset (Electronics).

3.7.3 Interpretability of Adaptation Functions

Compared to classical domain adaptation methods [44, 39], interpretability is one of our *coordinate-wise* optimal transport method assets. For example, by investigating the obtained *stochastic mapping* function of categorical features (Equation (3.13)), one can get details of mapping between each modality. We show one example in Figure 3.8 where the *stochastic mapping* function of a categorical feature in the Kaggle dataset is represented by a transformation matrix. For this categorical feature, source domain has more encoded value 0 than the one of target domain; thus, the encoded values 2 and 3 in target domain have respectively 30.7% and 57.7% of probability to be mapped to the encoded value 0 in the source domain. Values of the Kaggle fraud detection dataset are masked for privacy protection. This example is obtained by using the general occurrence frequency distance [61], while business specific distance between categorical values can also be applied to better fit different real-life industrial cases.

Moreover, the greedy feature selection process enhances this interpretability through the selected subset of features $\hat{\mathcal{D}}$, and their selected orders. Specifically, the feature subset $\hat{\mathcal{D}}$ reveals the source of drifts between source and target domains, and the order provides importance of each feature in domain adaptation tasks intuitively. All this information can provide business experts, with or without machine learning backgrounds, insights to better understand different domains.

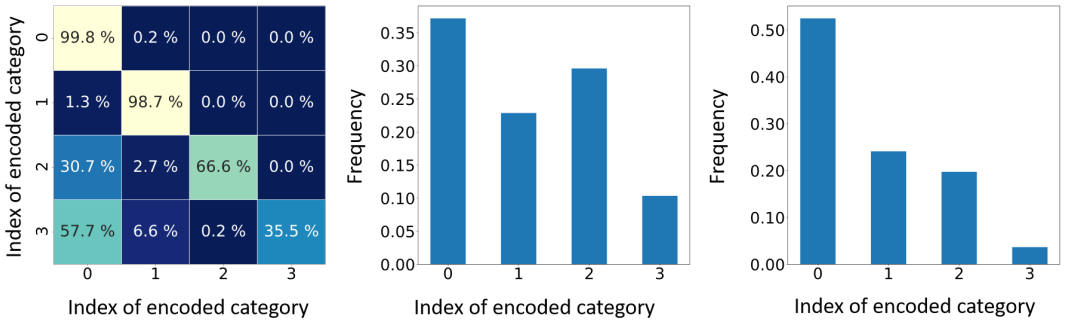


Figure 3.8: Left: The mapping matrix of a categorical feature where different values are encoded by integer numbers. Middle: The target domain distribution of this categorical feature. Right: The source domain distribution of this categorical feature.

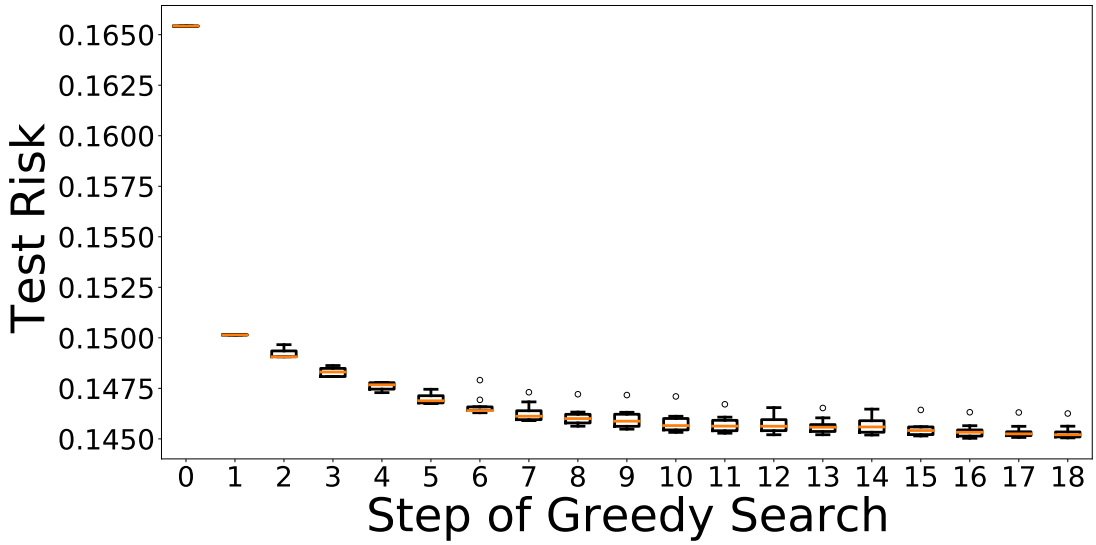


Figure 3.9: The evolution of log-loss risk at different steps of the greedy algorithm on Kaggle datasets in a weakly supervised scenario. We repeat the feature selection process 10 times and report variations at each step by a box plot.

An example is illustrated in Figure 3.9 where evolution of log-loss risks at each step of greedy feature selection is shown over the Kaggle payment dataset, and the contribution of each feature can be measured by the differences of test risk. In this example, the contribution of the first adapted feature is significantly larger than the others. Consequently, one can investigate this feature to further modelize payment habits of customers from different domains.

3.8 Conclusion

This chapter introduced a new *target to source* perspective for domain adaptation tasks. An unsupervised *coordinate-wise* transformation function and a weakly supervised feature selection process were proposed to address this setting. Leveraging the one-dimensional optimal transport, the proposed method was *feature-type free* and *interpretable*. Moreover, efficient implementations were proposed and had achieved state-of-the-art adaptation performances over three adaptation tasks. However, the adaptation pipeline (Figure 3.6) relied on weakly labeled target domain data; thus, the spectrum of use is limited. To generalize the method to an unsupervised case, we further propose a stability-based pseudo-labeling method in the following chapter.

Chapter 4

Unsupervised Feature Selection for Domain Adaptation

This chapter presents the second contribution of this thesis. Results of this chapter have been published in the paper "Interpretable Domain Adaptation Using Unsupervised Feature Selection on Pre-trained Source Models" [168]. We address a more challenging unsupervised target to source domain adaptation scenario. It can be seen as a relaxation of the weakly supervised case considered in the previous chapter. To this end, we propose a new pseudo-label estimator over unlabeled target examples based on rank-stability regarding the source model prediction. Such estimated "labels" are further used in a feature selection process to assess whether each feature needs to be transformed to achieve adaptation. We provide theoretical foundations of our method as well as an efficient implementation.

4.1 Unsupervised Target to Source Domain Adaptation Pipeline

The previous Chapter 3 proposed an adaptation pipeline in a weakly supervised setting. Note that the empirical results (Section 3.7.2) highlighted the impact of the feature selection step in the adaptation pipeline. However, labeled information is not always available. In cases where target domain labels are entirely missing, one needs to skip the feature selection step of the proposed pipeline (Figure 3.6), which gives suboptimal adaptation results (Figure 3.5).

Alternatively, this chapter addresses a more challenging unsupervised domain adaptation setting where target labels, even few labeled examples, are not available ($n_t^l = 0$). However, the set of target inputs $\mathbb{X}^t$ is given. Since $n_t^l = 0$, we have

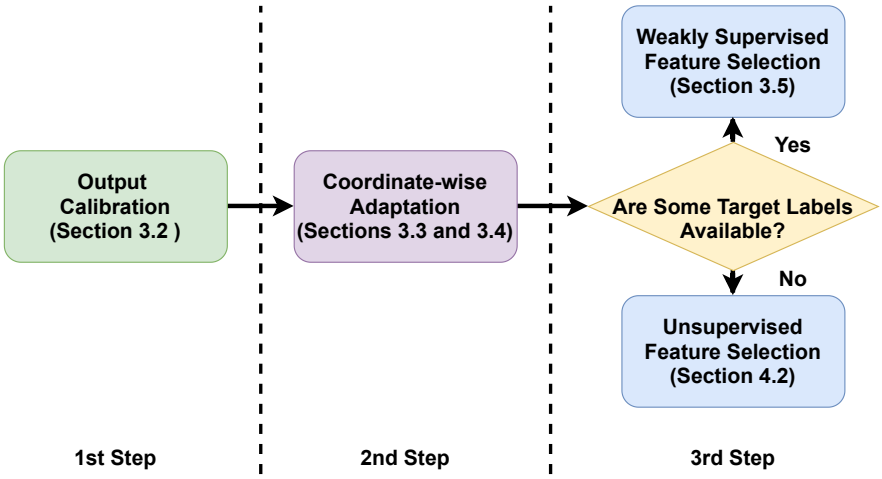


Figure 4.1: Main modules composing our proposed adaptation pipeline.

$\mathbb{X}^t = \mathbb{X}^t_u$ in this setting. Figure 4.1 illustrates the complete adaptation pipeline. Compared to the pipeline of Figure 3.6, we add a condition to check if the target domain includes labeled data. If it is not the case, we use the unsupervised feature selection process that we further detail in this chapter to find the optimal subset of features to adapt.

Intuitively, if one gets an estimator $\hat{h}(\cdot)$ to annotate some specific target examples $\boldsymbol{x} \in \mathbb{X}^t$ approximately, one can solve the feature selection problem by injecting $\hat{h}(\cdot)$ into Equation (3.15) to replace y^t. Since $\hat{h}(\cdot)$ does not generalize to new target examples, one cannot directly use it as the target domain predictor. Nevertheless, it can serve as an adequate "anchor" for the unsupervised feature selection process. Such approximate annotations are the so-called pseudo-labels.

4.2 Pseudo-labeling Methods

One of the most well-known strategies estimates target pseudo-labels using predictions of source models directly, assuming that high-confidence predictions are correct [174, 21, 128]. However, we illustrate further on a toy example (Figure 4.2) that this approach could be unstable and gives incorrect pseudo-labels in some cases. In contrast, instead of pseudo-labeling target examples with confident predictions, we estimate pseudo-labels of examples that have rank-stable predictions under different transformation functions. We name our unsupervised domain adaptation pipeline with a pseudo-labeling method: ***S**tability-based feature selection for **C**oordinate-wise **D**omain **A**daptation* (**SCDA**).

4.2.1 Rank Stability

In this section, we define a notion of *stable inputs* suited for our domain adaptation task, we propose a pseudo-label estimator, and we prove that SCDA gives pseudo-labels equal predictions of the optimal *coordinate-wise* adaptation function, making it legitimate to be applied to the unsupervised feature selection.

Definition 4.1 (Stable inputs)**.** A target input example $\boldsymbol{x} \in \mathbb{X}^t$ is called stable over $\mathbb{D}$ if its rank of prediction remains unchanged after being adapted by *coordinate-wise* transformations over all different feature-subsets from $\mathbb{D}$, that is,

$$\begin{aligned} &\forall \boldsymbol{x}' \in \mathbb{X}^t \,, \boldsymbol{x}' \neq \boldsymbol{x} \,, \forall \mathcal{D}, \mathcal{D}' \in \mathbb{D} \times \mathbb{D} \,: \\ &h_t^{\mathcal{D}}(\boldsymbol{x}) > h_t^{\mathcal{D}}(\boldsymbol{x}') \Longleftrightarrow h_t^{\mathcal{D}'}(\boldsymbol{x}) > h_t^{\mathcal{D}'}(\boldsymbol{x}') \,. \end{aligned} \tag{4.1}$$

Accordingly, we denote by $\mathbb{X}^t_{\text{stab}}$ a set of all such target examples.

Recall that $\mathcal{D}^*$ is the optimal subset of features to adapt (Equation (3.14)), we suppose that predictions of $h_t^{\mathcal{D}^*}(\cdot)$ and $h_t(\cdot)$ on target domain data have the same distribution, that is,

$$P(h_t^{\mathcal{D}^*}(X^t)) = P(h_t(X^t)) \,. \tag{4.2}$$

Such an assumption can be easily verified as $\mathcal{D}^*$ is the optimal subset of features that minimizes the classification loss of target domain data (prediction difference between $h_t^{\mathcal{D}^*}(\boldsymbol{x})$ and $h_t(\boldsymbol{x})$ is marginal). Under this mild assumption, we can further infer that

$$P(h_t^{\mathcal{D}^*}(X^t)) = P(h_t(X^t)) = P(h_s(X^s)) \,. \tag{4.3}$$

Proof. Proposition 3.1 shows that the optimal adaptation function verifies

$$P(\mathcal{G}^*(X^t)) = P(X^s) \,.$$

Furthermore, we have

$$P(h_s \circ \mathcal{G}^*(X^t)) = P(h_s(X^s)) \implies P(h_t(X^t)) = P(h_s(X^s)) \,.$$

Consequently, the equality of source and target domain output distributions is necessary for an optimal *target to source* domain adaptation function. □

Proposition 4.1 (Property of stable inputs)**.** Given that $\mathbb{D}$ contains the optimal

subset of features $\mathcal{D}^*$, we have

$$\forall \boldsymbol{x} \in \mathbb{X}^t_{\text{stab}}\,, \forall \mathcal{D} \in \mathbb{D}\,, \quad {f_{h_s}}^{-1} \circ f_{h_t^{\mathcal{D}}}(h_t^{\mathcal{D}}(\boldsymbol{x})) = h_t^{\mathcal{D}^*}(\boldsymbol{x})\,, \tag{4.4}$$

where $f_{h_s}(\cdot)$ and $f_{h_t^{\mathcal{D}}}(\cdot)$ are respectively cumulative distribution functions of $h_s(X^s)$ and $h_t^{\mathcal{D}}(X^t)$.

Proof. As ranks of predictions can be naturally expressed by cumulative distribution functions, given Equation (4.1), and $\mathcal{D}^*$, $\mathcal{D} \in \mathbb{D}$, we have

$$\forall \boldsymbol{x} \in \mathbb{X}^t_{\text{stab}}\,, f_{h_t^{\mathcal{D}^*}}(h_t^{\mathcal{D}^*}(\boldsymbol{x})) = f_{h_t^{\mathcal{D}}}(h_t^{\mathcal{D}}(\boldsymbol{x}))\,, \tag{4.5}$$

where $f_{h_t^{\mathcal{D}^*}}(\cdot)$ refers to the cumulative distribution function of $h_t^{\mathcal{D}^*}(X^t)$. Proposition 3.1 states that $P(Y^s) = P(Y^t)$, According to Equation (4.3), we have $f_{h_t^{\mathcal{D}^*}}(\cdot) = f_{h_t}(\cdot) = f_{h_s}(\cdot)$. Analogously, $f_{h_t}(\cdot)$ is the cumulative distribution functions of $h_t(X^t)$. Replacing $f_{h_t^{\mathcal{D}^*}}(\cdot)$ by $f_{h_s}(\cdot)$ in Equation (4.5), we get

$$f_{h_s}(h_t^{\mathcal{D}^*}(\boldsymbol{x})) = f_{h_t^{\mathcal{D}}}(h_t^{\mathcal{D}}(\boldsymbol{x}))\,.$$

As $f_{h_s}(\cdot)$ is invertible (see Section 2.6.4), we have

$$h_t^{\mathcal{D}^*}(\boldsymbol{x}) = {f_{h_s}}^{-1} \circ f_{h_t^{\mathcal{D}}}(h_t^{\mathcal{D}}(\boldsymbol{x}))\,,$$

which proves Equation (4.4). □

Note that, $h_t^{\mathcal{D}^*}(\cdot)$ is the optimal *coordinate-wise* adaptation function that we expect to get. Therefore, we define the pseudo-label estimator $\hat{h}^{\mathcal{D}}(\cdot)$ by the following formula:

Definition 4.2 (Rank-stable based pseudo-label estimator)**.**

$$\begin{gathered}\forall \boldsymbol{x} \in \mathbb{X}^t_{\text{stab}}\,, \forall \mathcal{D} \in \mathbb{D}\,, \\ \hat{h}^{\mathcal{D}}(\boldsymbol{x}) = {f_{h_s}}^{-1} \circ f_{h_t^{\mathcal{D}}}(h_t^{\mathcal{D}}(\boldsymbol{x})) = h_t^{\mathcal{D}^*}(\boldsymbol{x})\,.\end{gathered} \tag{4.6}$$

The defined pseudo-label estimator provides the same probability score over stable target domain examples as the prediction results of the optimal adaptation function.

An example is illustrated in Figure 4.2, where we compare two different pseudo-labeling techniques on a toy dataset. In this example, we first identify stable target examples over $\mathbb{D}$, where $\mathbb{D}$ contains all feature subsets of the two-dimensional space. Then we estimate pseudo-labels using $\hat{h}^{\mathcal{D}}(\cdot)$. Stable examples $\boldsymbol{x}$ with $\hat{h}^{\mathcal{D}}(\boldsymbol{x}) > 0.5$ are colored as blue and the others as orange. Note that

Figure 4.2: Left: labeled source domain data. Middle left: unlabeled target domain data. Middle right: pseudo-labels given by confidence-based methods. Right: pseudo-labels provided by our proposition over stable target examples. The ground truth of target data is shown in light colors, and pseudo-labels of the target domain are shown in deep colors. The green line in the sub-figures is the pre-trained source domain predictor.

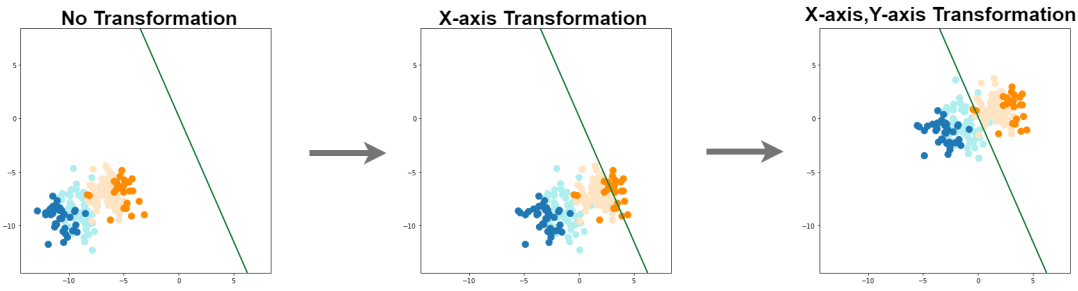


Figure 4.3: Steps of *coordinate-wise* transformations using stability-based pseudo-labels.

according to Equation (4.6), the choice of $\mathcal{D}$ does not affect the pseudo-labels for all $\boldsymbol{x} \in \mathbb{X}^t_{\text{stab}}$. However, if $\boldsymbol{x} \notin \mathbb{X}^t_{\text{stab}}$, pseudo-labels change with respect to the choice of $\mathcal{D}$. The pseudo-labeled examples are further used to adapt target data in order to fit the pre-trained source model (Figure 4.3). We provide details of this process in Algorithm 4. In contrast, confidence-based pseudo-labeling methods (Figure 4.2 middle right) consider predictions of target examples far from the decision boundary as correct; thus, all examples are pseudo-labeled as blue and provide no information to help domain adaptations. One may note that pseudo-labels given by our method can be close to the decision boundary of the two classes. Indeed, our method is agnostic to the prediction value while relying only on the rank-stability over $\mathbb{D}$.

4.2.2 Relaxation of Rank Stability

The method described in the previous section annotates only stable target examples. It is ineffective when the number of examples in $\mathbb{X}^t_{\text{stab}}$ is scarce, as it needs enough *stable* elements to reach a diversity that faithfully expresses the global distribution

of X^t. Ideally, we expect $\mathbb{X}^t_{\text{stab}}$ contains as many examples as possible. Therefore, we introduce a relaxation of Definition 4.1 to compensate for this scarcity. The relaxation tunes the size of $\mathbb{X}^t_{\text{stab}}$ to reach the right trade-off between the similarity to $\mathbb{X}^t$ and the constraint of Equation (4.1).

Definition 4.3 (δ-stable inputs)**.** A target input example $\boldsymbol{x} \in \mathbb{X}^t$ is called δ-stable over $\mathbb{D}$ if

$$l_{\text{stab}}(\boldsymbol{x}) = \max_{\mathcal{D},\mathcal{D}' \in \mathbb{D}} \left(|\hat{h}^{\mathcal{D}}(\boldsymbol{x}) - \hat{h}^{\mathcal{D}'}(\boldsymbol{x})| \right) \leq \delta \,. \tag{4.7}$$

Accordingly, we denote by $\mathbb{X}^t_{\delta-\text{stab}}$ an input set that contains all such target examples. δ here is a tolerance measure that controls the bias between stable and δ-stable input examples. By setting $\delta = 0$, one can retrieve Definition 4.1.

Although one can still use $\hat{h}^{\mathcal{D}}(\cdot)$ to estimate pseudo-labels for $\boldsymbol{x} \in \mathbb{X}^t_{\delta-\text{stab}}$, it is uncertain that Proposition 4.1 is verified. Intuitively, a larger δ results in a richer $\mathbb{X}^t_{\delta-\text{stab}}$ but with a higher risk of violating Proposition 4.1. In the remainder of this section, we formally analyze the effects of this relaxation over the feature selection process and propose the corresponding unsupervised objective function.

In their seminal domain adaptation analysis, Ben-David et al. [7] proposed to upper bound the expected target domain risk by a sum of three terms: (i) the source domain risk, (ii) the $\mathcal{H}$-divergence defined as

$$d_{\mathcal{H}}(P(X^t), P(X^s)) = 2 \sup_{h \in \mathcal{H}} \left| \mathop{\mathrm{E}}_{\boldsymbol{x} \sim P(X^t)} [h(\boldsymbol{x}) \neq 1] - \mathop{\mathrm{E}}_{\boldsymbol{x} \sim P(X^s)} [h(\boldsymbol{x}) \neq 1] \right|$$

to measure the discrepancy between source ($P(X^s)$) and target ($P(X^t)$) input marginal distributions, and (iii) an intrinsic error between true labeling functions of two domains. We denote by

$$e(\mathcal{D}) = \mathop{\mathrm{E}}_{\boldsymbol{x} \sim P(X^t)} \left[|h^{\mathcal{D}}_t(\boldsymbol{x}) - h^{\mathcal{D}^*}_t(\boldsymbol{x})| \right] \tag{4.8}$$

the target domain risk between a label predictor $h^{\mathcal{D}}_t(\cdot)$ and the optimal one $h^{\mathcal{D}^*}_t(\cdot)$. We notice that the drift between $\mathbb{X}^t_{\delta-\text{stab}}$ and $\mathbb{X}^t$ is known as sample selection bias [52]; thus, we can empirically upper bound $e(\mathcal{D})$ by considering δ-stable examples $\boldsymbol{x} \in \mathbb{X}^t_{\delta-\text{stab}}$ as the "source" domain.

Theorem 4.1 (Upper bound of target domain risk)**.** Given a subset of features $\mathcal{D} \in \mathbb{D}$, for all $\mathcal{D}' \in \mathbb{D}$, $\delta \in [0, 1]$, the following inequality holds:

$$e(\mathcal{D}) \leq l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta) + d_{\text{div}}(\delta) \,, \tag{4.9}$$

where

$$l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta) = \mathop{\mathrm{E}}_{\boldsymbol{x}\sim P(X_\delta^t)} \left[|h_t^{\mathcal{D}}(\boldsymbol{x}) - \hat{h}^{\mathcal{D}'}(\boldsymbol{x})| \right], \tag{4.10}$$

$$d_{\text{div}}(\delta) = \mathop{\mathrm{E}}_{\boldsymbol{x}\sim P(X_\delta^t)} \left[l_{\text{stab}}(\boldsymbol{x}) \right] + \frac{1}{2} d_{\mathcal{H}}(P(X^t), P(X_\delta^t)), \tag{4.11}$$

$P(X_\delta^t)$ referring to the distribution of δ-stable target inputs.

Proof. According to the Theorem 1 of Ben-David et al. [7], we have

$$e(\mathcal{D}) \leq \mathop{\mathrm{E}}_{\boldsymbol{x}\sim P(X_\delta^t)} \left[|h_t^{\mathcal{D}}(\boldsymbol{x}) - h_t^{\mathcal{D}^*}(\boldsymbol{x})| \right] + \frac{1}{2} d_{\mathcal{H}}(P(X^t), P(X_\delta^t)) + c_{\text{e}}.$$

As examples in $\mathbb{X}^t$ and in $\mathbb{X}^t_{\delta-\text{stab}}$ have the same true labeling function, the constant term $c_{\text{e}} = 0$. We use the triangle inequality on the expectation term of the upper bound and we get

$$e(\mathcal{D}) \leq l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta) + \mathop{\mathrm{E}}_{\boldsymbol{x}\sim P(X_\delta^t)} \left[|\hat{h}^{\mathcal{D}'}(\boldsymbol{x}) - h_t^{\mathcal{D}^*}(\boldsymbol{x})| \right] + \frac{1}{2} d_{\mathcal{H}}(P(X^t), P(X_\delta^t)).$$

Since $f_{h_t^{\mathcal{D}^*}}(\cdot) = f_{h_s}(\cdot)$, relying on Equations 4.4 and 4.3, we get

$$h_t^{\mathcal{D}^*}(\boldsymbol{x}) = {f_{h_s}}^{-1} \circ f_{h_t^{\mathcal{D}^*}}(h_t^{\mathcal{D}^*}(\boldsymbol{x})) = \hat{h}^{\mathcal{D}^*}(\boldsymbol{x}).$$

As $\mathcal{D}', \mathcal{D}^*$ are subsets of $\mathbb{D}$, by replacing $h_t^{\mathcal{D}^*}(\boldsymbol{x})$ by $\hat{h}^{\mathcal{D}^*}(\boldsymbol{x})$, and relying on Equation (4.7), we have

$$\begin{aligned} &|\hat{h}^{\mathcal{D}'}(\boldsymbol{x}) - h_t^{\mathcal{D}^*}(\boldsymbol{x})| = |\hat{h}^{\mathcal{D}'}(\boldsymbol{x}) - \hat{h}^{\mathcal{D}^*}(\boldsymbol{x})| \leq l_{\text{stab}}(\boldsymbol{x}) \\ \implies &\mathop{\mathrm{E}}_{x\sim P(X_\delta^t)} \left[|\hat{h}^{\mathcal{D}'}(\boldsymbol{x}) - h_t^{\mathcal{D}^*}(\boldsymbol{x})| \right] \leq \mathop{\mathrm{E}}_{x\sim P(X_\delta^t)} \left[l_{\text{stab}}(\boldsymbol{x}) \right]. \end{aligned}$$

Theorem 4.1 is proved. □

In this bound, $l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta)$ refers to the feature selection risk over δ-stable target examples. $d_{\text{div}}(\delta)$ encompasses the risk related to the stable inputs relaxation, and the discrepancy between $P(X^t)$ and $P(X_\delta^t)$. All elements in this upper bound can be computed without target domain labels. Therefore, we define the unsupervised objective function of SCDA as

$$\tilde{\mathcal{D}}^* = \mathop{\mathrm{argmin}}_{\mathcal{D}\in\mathbb{D}} \min_{\mathcal{D}'\in\mathbb{D}} \min_{\delta\in[0,1]} \left(l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta) + d_{\text{div}}(\delta) \right). \tag{4.12}$$

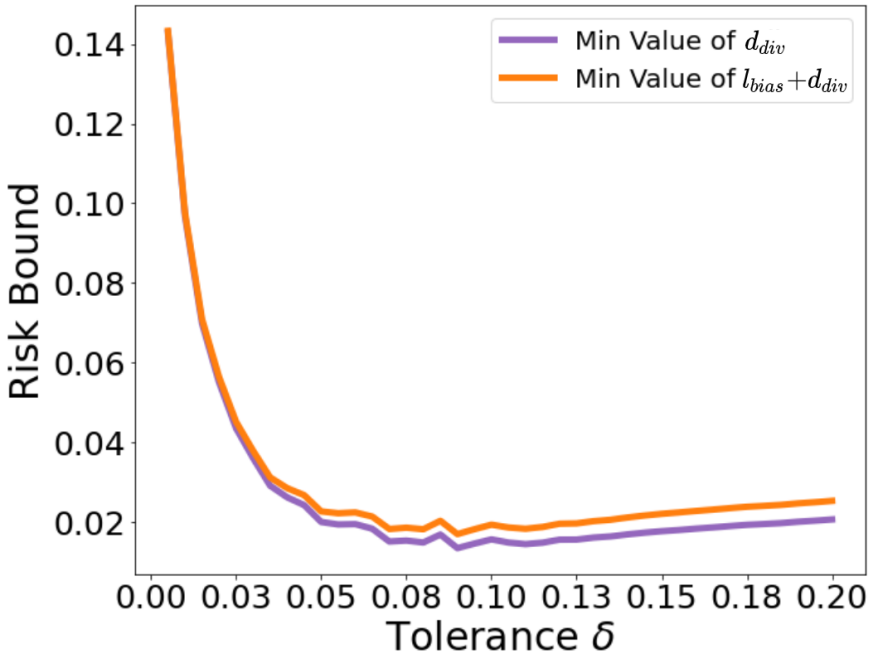


Figure 4.4: Comparison of minimum values of upper bound $l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta) + d_{\text{div}}(\delta)$ and $l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta)$ with respect to δ of a Kaggle fraud detection task.

4.3 Implementation

Set the value of δ. Experimental results show that the optimal δ that minimizes the sum of $l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta) + d_{\text{div}}(\delta)$ is close to the minimizer of $d_{\text{div}}(\delta)$ (Figure 4.4). Therefore, we can rely first on $d_{\text{div}}(\delta)$ to determinate the optimal value of δ, and then find the couple $(\mathcal{D}, \mathcal{D}')$ that minimizes $l_{\text{bias}}(\mathcal{D}, \mathcal{D}', \delta)$. This approach reduces the complexity by simplifying the combination problem of triplets $((\mathcal{D}, \mathcal{D}', \beta))$ to the combination problem of couples $((\mathcal{D}, \mathcal{D}'))$. In practice, we use a grid search algorithm to discretize the value space of δ. The expectation part of Equation (4.11) is empirically estimated by the average value of $l_{\text{stab}}(\boldsymbol{x})$ (Equation (4.7)) over δ-stable inputs. The $\mathcal{H}$-divergence is estimated by training a classifier to distinguish examples between $\mathbb{X}^t$ and $\mathbb{X}^t_{\delta-\text{stab}}$. Detailed processes are presented in Algorithm 3. Of note, a one-dimensional dictionary is a {key:value} structure, while a two-dimensional dictionary uses a one-dimensional dictionary as its value.

Steps 4-6 compute pseudo-labels of $\boldsymbol{x}$ regarding all possible choices of $\mathcal{D}$. When the number of elements in $\mathbb{D}$ is large, we simplify the computation by considering $\mathcal{D}$ with only one input space dimension. Step 10 discretizes the choice of δ. In practice, we apply a log scale to this discretizing function.

Step 5 here is a simplification of notation, as $\hat{h}^{\mathcal{D}}(\cdot)$ (Equation (4.6)) involves a stochastic part $f_{h_t^{\mathcal{D}}}(h_t^{\mathcal{D}}(\cdot))$ ($h_t^{\mathcal{D}}(\cdot) = h_s \circ \mathcal{G}^{\mathcal{D}}(\cdot)$ and the transform $\mathcal{G}^{\mathcal{D}}(\cdot)$ is

stochastic), the true pseudo-label is given by

$$\text{Pseudo}[\boldsymbol{x}][\mathcal{D}] = \frac{1}{n_r} \sum_{n_r} \hat{h}^{\mathcal{D}}(\boldsymbol{x}) \,,$$

where n_r is the repetition times. $\hat{h}^{\mathcal{D}}(\boldsymbol{x})$ gives slightly different pseudo-labels for the same input value $\boldsymbol{x}$ because of the stochastic transformation $\mathcal{G}^{\mathcal{D}}(\cdot)$.

Step 12 empirically computes $d_{\text{div}}(\delta)$, and the term $\hat{d}_{\mathcal{H}}(\mathbb{X}^t, \mathbb{X}^t_{\delta-\text{stab}})$ is estimated relying on a GBDT model. First, one divides $\mathbb{X}^t_{\delta-\text{stab}}$ and $\mathbb{X}^t$ into training datasets and testing datasets. Then, a GBDT model is trained using training data to distinguish examples between $\mathbb{X}^t_{\delta-\text{stab}}$ and $\mathbb{X}^t$. One stops the training process when the testing error is minimized. $\hat{d}_{\mathcal{H}}(\mathbb{X}^t, \mathbb{X}^t_{\delta-\text{stab}})$ is estimated by this empirically optimal model over all examples in $\mathbb{X}^t_{\delta-\text{stab}}$ and $\mathbb{X}^t$. Note that, once δ is fixed, the δ-stable input dataset $\mathbb{X}^t_{\delta-\text{stab}}$ is also determinate.

Algorithm 3 Get δ-stable Set

1: Initialize Pseudo[ ][ ] $\leftarrow$ {{}} by an empty two-dimensional dictionary
2: Initialize Lstab[ ] $\leftarrow$ {} by an empty one-dimensional dictionary
3: Initialize Ddiv[ ] $\leftarrow$ {} by an empty one-dimensional dictionary
4: **for** $\mathcal{D}$ **in** $\mathbb{D}$ **do**
5: Pseudo[$\boldsymbol{x}$][$\mathcal{D}$] $\leftarrow \hat{h}^{\mathcal{D}}(\boldsymbol{x})$
6: **end for**
7: **for** $\boldsymbol{x}$ **in** $\mathbb{X}^t$ **do**
8: Lstab[$\boldsymbol{x}$] $\leftarrow$ $\max_{\mathcal{D}}$ Pseudo[$\boldsymbol{x}$][$\mathcal{D}$] - $\min_{\mathcal{D}}$ Pseudo[$\boldsymbol{x}$][$\mathcal{D}$]
9: **end for**
10: **for** δ **in** discretize$(0, 1)$ **do**
11: $\mathbb{X}^t_{\delta-\text{stab}} \leftarrow \{\boldsymbol{x} | \text{Lstab}[\boldsymbol{x}] \leq \delta\}$
12: Ddiv[δ] $\leftarrow$ $\text{mean}_{\boldsymbol{x} \in \mathbb{X}^t_{\delta-\text{stab}}}$ Lstab[$\boldsymbol{x}$] + $\frac{1}{2}\hat{d}_{\mathcal{H}}(\mathbb{X}^t, \mathbb{X}^t_{\delta-\text{stab}})$
13: **end for**
14: **return:** argmin_{δ} Ddiv[δ], $\mathbb{X}^t_{\delta-\text{stab}}$

Unsupervised Feature Selection The unsupervised feature selection process (Algorithm 4) follows the same paradigm as the weakly supervised one (Algorithm 2). Both feature selection methods leverage a greedy search algorithm and a bootstrap strategy.

Namely, at initialization (step 2), no feature is adapted. We use a dictionary Count[d] to count the frequency of each newly adapted dimension (step 4). The bootstrap process starts from step 5 and ends at step 16. We add the most frequent feature into the selected optimal subset of features (steps 17 and 18). We stop the process when no feature is added or no feature is significantly better than the others (step 20).

The key difference between the unsupervised feature selection algorithm and the weakly supervised one is the objective function. According to Equation (4.12),

Algorithm 4 Unsupervised Greedy Search Algorithm

1: Initialize $i \leftarrow 0$
2: Initialize $\tilde{\mathcal{D}}^{(i)} \leftarrow \{\}$
3: **repeat**
4: Initialize $\text{Count}[d] \leftarrow 0\,, \forall d \in \mathcal{X}/\tilde{\mathcal{D}}^{(i)} \cup \{\varnothing\}$
5: **for** $\mathbb{X}^t_{\text{boot}}$ **in** bootstraps of $\mathbb{X}^t_{\delta-\text{stab}}$ **do**
6: Initialize Loss[ ][ ] $\leftarrow \{\{\}\}$ by a two-dimensional dictionary
7: **for** d **in** $\mathcal{X}/\tilde{\mathcal{D}}^{(i)} \cup \{\varnothing\}$ **do**
8: $\mathcal{D} \leftarrow \tilde{\mathcal{D}}^{(i)} \cup \{d\}$
9: **for** d' **in** $\mathcal{X}/\tilde{\mathcal{D}}^{(i)} \cup \{\varnothing\}$ **do**
10: $\mathcal{D}' \leftarrow \tilde{\mathcal{D}}^{(i)} \cup \{d'\}$
11: $\text{Loss}[d][d'] \leftarrow \sum_{\boldsymbol{x} \in \mathbb{X}^t_{\text{boot}}} |h^{\mathcal{D}}_t(\boldsymbol{x}) - \hat{h}^{\mathcal{D}'}(\boldsymbol{x})|$
12: **end for**
13: **end for**
14: $d_{\min}, d'_{\min} \leftarrow \text{argmin}_{d,d'} \text{Loss}[d][d']$
15: $\text{Count}[d_{\min}] \leftarrow \text{Count}[d_{\min}] + 1$
16: **end for**
17: $d^* \leftarrow \text{argmax}_d \text{Count}[d]$
18: $\tilde{\mathcal{D}}^{(i+1)} \leftarrow \tilde{\mathcal{D}}^{(i)} \cup \{d^*\}$
19: $v \leftarrow \text{Count}[d^*] / \sum_d \text{Count}[d]$
20: **until** $d^* \leftarrow \varnothing$ or $v < 0.5$; $i \leftarrow i + 1$
21: **return:** $\tilde{\mathcal{D}}^{(i)}$

as $d_{\text{div}}(\delta)$ is fixed, the unsupervised feature selection minimization problem focuses on Equation (4.10) (steps 6-13 of Algorithm 4). Whereas the weakly supervised feature selection focus on Equation (3.15) (steps 6-10 of Algorithm 2). Compared to Algorithm 2, Algorithm 4 has a nested loop and is more computationally expensive. Nonetheless, it needs no labeled information.

4.4 Experiments

Similar to Section 3.7, in this section, we evaluate the performances of SCDA on 3 different datasets: Kaggle fraud detection tasks, Worldline fraud detection tasks, and Amazon review tasks. Details of datasets and separations of source and target domains are provided in Section 2.8.

4.4.1 General Setup

We use the same NN and GBDT pre-trained models as Section 3.7 and we compare with our previously proposed CDA, deep adaptation methods, and classical adaptation methods in an unsupervised setting. We also present adaptation performances of weakly supervised feature selection pipeline WCDA. For comparison, we make the best performance of unsupervised adaptation methods bold, while excluding WCDA. All performances are evaluated by PR-AUC (accuracy

for Amazon review tasks) and log-loss. For the sake of comparison, we set the average performance of source domain NN baseline models as references and report the percentage of improvements compared to such models. We also report standard deviations to illustrate the stabilities of adaptation methods. Note that SCDA is the unsupervised version of WCDA. Hence characteristics such as the interpretability of the adaptation method are also the asset of SCDA. However, we do not reintroduce such a point here for the sake of document fluidity.

4.4.2 Adaptation Performance Analysis

4.4.2.1 Kaggle Dataset

Table 4.1 reports adaptation results of Kaggle fraud detection tasks in the unsupervised setting. NN-SCDA outperforms all the other adaptation methods, even the weakly supervised ones, on average. We explain such an observation by the scarcity of weakly labeled examples. Recall that, in a weakly supervised setting, we use 200 examples of labeled target domain data. However, they may not be enough to fully disentangle underlying structures of target domain data. In contrast, SCDA produces pseudo-labels and minimizes an upper bound of the target domain risk. Therefore, it can tune the optimal trade-off between precisions and numbers of pseudo-labels. Besides, SCDA is the best unsupervised domain adaptation method in terms of log-loss improvements of NN models, and in both metrics of LGB models.

Compared to deep adaptation methods that sometimes have a negative improvement (*e.g.,* "D-1 to M of MCD"), SCDA always improves performances of directly applying pre-trained source domain predictive models on target domain data. Additionally, SCDA has a smaller variance than deep adaptation methods, which results in more robust predictions. The high variance of deep adaptation methods confirms our arguments that deep adaptation methods are tedious to train, such that they are not practical for some real-life applications.

Concerning the number of adapted features (Table 4.3), SCDA generally transforms less features than WCDA. Both SCDA and WCDA provide a sparse transformation of the input space.

Table 4.1: Adaptation performances of **unsupervised** adaptation methods over **Kaggle** datasets (see Section 2.8.2 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Method	D-1 to M	D-2 to M	D-3 to M	AVG
DAN	12.31±6.72	-3.20±7.18	1.41±3.63	3.51
DANN	3.47±9.53	-2.90±3.82	-4.21±6.51	-1.21
MCD	-11.47±13.38	-6.29±6.09	-6.81±4.83	-8.19
CORAL	**9.11±0.26**	-2.47±0.08	-8.37±0.14	-0.58
NN-CDA (**ours**)	-8.15±2.51	**3.87±0.35**	0.69±0.38	-1.20
NN-SCDA (**ours**)	3.23±1.87	2.88±0.35	**5.41±0.48**	**3.84**
NN-WCDA(sup) (**ours**)	1.30±7.57	2.98±2.14	3.72±1.40	2.66

(a) Improvements of PR-AUC of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
DAN	29.57±4.32	-1.58±9.64	4.37±5.39	10.79
DANN	29.58±4.05	2.39±3.98	2.06±5.07	11.34
MCD	21.59±9.25	-1.29±5.34	-7.55±10.32	4.25
CORAL	30.66±0.07	4.69±0.08	-2.80±0.13	10.85
NN-CDA (**ours**)	18.38±0.62	6.32±0.34	7.40±0.28	10.70
NN-SCDA (**ours**)	**31.42±0.53**	**9.54±0.35**	**14.05±0.77**	**18.34**
NN-WCDA(sup) (**ours**)	30.42±3.26	8.95±2.67	12.07±2.04	17.14

(b) Improvements of log-loss of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB Baseline	26.13±1.53	4.18±1.47	7.58±3.26	12.63
CORAL	17.19±0.61	-8.23±0.26	-8.50±0.51	0.15
LGB-CDA (**ours**)	22.55±1.15	5.24±0.53	8.95±0.55	12.25
LGB-SCDA (**ours**)	**32.68±1.05**	**7.14±0.32**	**14.31±0.24**	**18.04**
LGB-WCDA(sup) (**ours**)	27.85±4.41	7.18±1.85	13.66±1.56	16.23

(c) Improvements of PR-AUC of LGB predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB Baseline	9.77±6.01	6.53±3.31	12.32±5.05	9.54
CORAL	33.75±0.26	-4.67±0.63	-7.76±1.19	7.11
LGB-CDA (**ours**)	32.59±0.54	8.66±0.56	14.10±0.42	18.45
LGB-SCDA (**ours**)	**41.22±0.64**	**14.34±0.34**	**24.00±0.16**	**26.52**
LGB-WCDA(sup) (**ours**)	39.86±1.68	14.31±2.40	22.06±1.66	25.41

(d) Improvements of log-loss of LGB predictive models.

Table 4.2: Adaptation performances of **unsupervised** adaptation methods over **Worldline** datasets (see Section 2.8.1 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Method	G-1 to B	G-2 to B	G-3 to B	AVG
DAN	7.31±5.78	5.47±3.96	10.01±4.70	7.60
DANN	4.38±5.09	6.43±2.69	5.28±3.03	5.37
MCD	6.46±5.70	1.41±8.19	6.84±4.94	4.91
CORAL	**8.31±0.02**	3.83±0.02	3.54±0.02	5.23
NN-CDA (**ours**)	3.86±1.88	7.93±1.12	**8.87±1.66**	6.89
NN-SCDA (**ours**)	8.02±1.35	**11.72±2.49**	5.95±0.84	**8.56**
NN-WCDA(sup) (**ours**)	11.75±3.14	8.89±5.58	13.62±9.46	11.42

(a) Improvements of PR-AUC of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
DAN	4.40±2.75	9.12±2.60	17.90±3.91	10.48
DANN	2.85±2.37	9.78±1.33	16.45±2.98	9.69
MCD	4.40±1.57	7.55±3.94	14.16±7.35	8.70
CORAL	5.45±0.00	9.43±0.01	14.07±0.01	9.65
NN-CDA (**ours**)	4.11±0.52	8.13±0.34	**18.92±0.27**	10.39
NN-SCDA (**ours**)	**7.26±0.48**	**13.03±1.32**	13.02±0.27	**11.11**
NN-WCDA(sup) (**ours**)	5.23±1.59	9.86±2.14	28.50±2.30	14.53

(b) Improvements of log-loss of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB Baseline	**9.91±6.43**	3.59±6.57	-1.94±8.31	3.85
CORAL	0.80±0.13	-8.46±0.16	-10.83±0.29	-6.17
LGB-CDA (**ours**)	8.96±1.98	**10.25±0.86**	**9.76±0.97**	**9.65**
LGB-SCDA (**ours**)	1.83±6.19	2.77±4.37	7.11±3.32	3.90
LGB-WCDA(sup) (**ours**)	22.65±5.14	17.84±4.38	15.21±5.65	18.56

(c) Improvements of PR-AUC of LGB predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB Baseline	5.02±3.54	4.38±4.00	9.43±4.65	6.28
CORAL	1.05±0.03	2.74±0.08	11.40±0.05	5.06
LGB-CDA (**ours**)	**5.35±0.46**	7.64±0.30	17.17±0.59	10.05
LGB-SCDA (**ours**)	3.14±2.33	**8.58±1.60**	**19.36±1.52**	**10.36**
LGB-WCDA(sup) (**ours**)	11.79±0.64	16.59±1.70	24.84±2.81	17.74

(d) Improvements of log-loss of LGB predictive models.

Table 4.3: Numbers of adapted features of Kaggle fraud detection tasks in an unsupervised setting.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
NN-SCDA (**ours**)	11.0±3.1	11.3±2.9	12.0±3.4	11.4
LGB-SCDA (**ours**)	15.1±3.9	17.1±4.2	17.9±4.2	16.7
NN-WCDA(sup) (**ours**)	11.3±2.4	13.6±3.4	15.7±3.7	13.5
LGB-WCDA(sup) (**ours**)	15.3±4.3	18.5±4.9	17.5±3.1	17.1

Table 4.4: Numbers of adapted features of Worldline fraud detection tasks in an unsupervised setting.

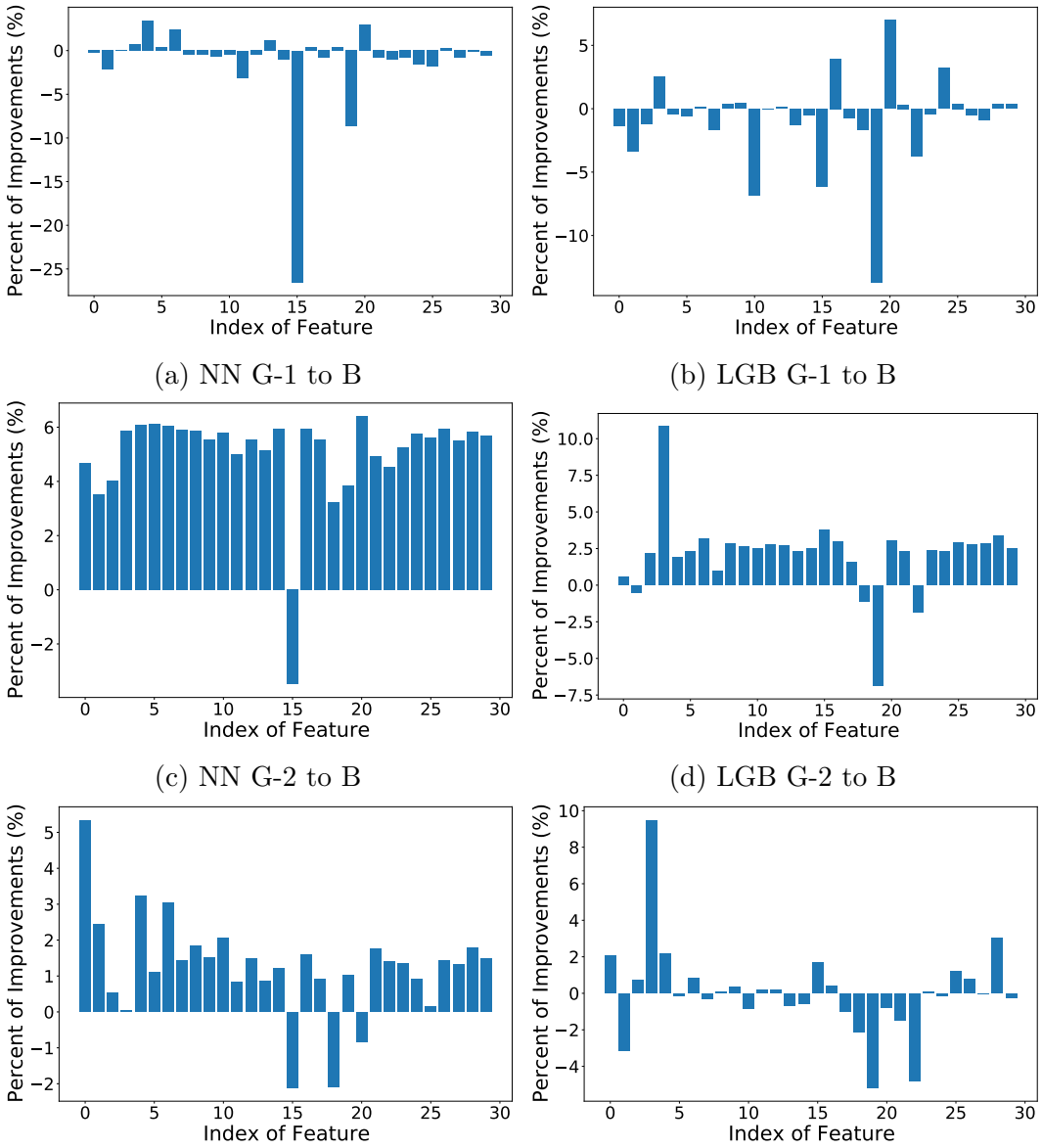
Method	G-1 to B	G-2 to B	G-3 to B	AVG
NN-SCDA (**ours**)	5.7±1.8	4.5±1.4	8.0±1.3	6.0
LGB-SCDA (**ours**)	8.7±3.1	7.7±0.7	7.7±1.0	8.0
NN-WCDA(sup) (**ours**)	7.0±2.4	5.1±1.7	7.8±2.3	6.6
LGB-WCDA(sup) (**ours**)	7.9±2.4	8.1±1.4	8.0±0.5	8.0

4.4.2.2 Worldline Dataset

As for the Worldline fraud detection task, SCDA does not outperform the weakly supervised WCDA. In the case where one has labeled information, WCDA is more suited for the Worldline domain adaptation tasks. We also observe that, among unsupervised adaptation methods, SCDA has the best average performances of NN models in both metrics and LGB models in terms of log-loss improvements. However, it decreases performances of the all adaptation method CDA in terms of PR-AUC when using LGB models. It shows that SCDA is better when using the log-loss improvement as a metric. An intuitive explanation is that SCDA minimizes the absolute error (Equation (4.10)) while PR-AUC focuses on the rank of predictions. Alternatively, log-loss relies on the absolute value of predictions thus can benefit more from SCDA. Additionally, when using the LGB pre-trained model for the Worldline domain adaptation tasks, there is one feature that outperforms all the other dimensions. If the feature selection process does not select the feature, performance improvements can be less significant.

Figure 4.5 illustrates improvements of PR-AUC when adapting only one feature of the Worldline tasks. For some LGB-based pre-trained models (Figures 4.5d and 4.5f), one can find one feature (feature indexed 3) that significantly improves performances than all the others. Without using target domain labels, identifying the only feature is challenging for SCDA. Consequently, SCDA may perform no better than the all adaptation solution CDA if the feature is not selected. In contrast, for NN-based models, there is no such a "must be adapted" feature.

Concerning the number of adapted features (Table 4.4), similar to the weakly supervised case, SCDA adapts around 1/4 of the input dimensions.



(a) NN G-1 to B

(b) LGB G-1 to B

(c) NN G-2 to B

(d) LGB G-2 to B

(e) NN G-3 to B

(f) LGB G-3 to B

Figure 4.5: Improvements of PR-AUC when adapting only one feature of the Worldline tasks.

Table 4.5: Adaptation performances of **NN unsupervised** adaptation methods over **Amazon datasets** (see Section 2.8.3 for the dataset description). We report percentages of performances improvements, in accuracy and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Task	DAN	DANN	MCD	CORAL	NN-CDA (**ours**)	NN-SCDA (**ours**)	NN-WCDA (sup) (**ours**)
B to D	**-0.91±0.40**	-3.42±1.31	-2.41±2.13	-0.22±0.00	**-0.91±0.03**	-0.60±0.10	0.61±0.10
B to E	**6.12±0.58**	4.28±2.59	4.24±1.37	3.97±0.00	5.66±0.04	3.94±0.40	6.83±0.02
B to K	6.31±1.40	8.20±3.13	6.73±2.88	10.47±0.00	**10.65±0.04**	8.89±0.13	11.64±0.02
D to B	-0.90±0.55	-1.67±0.81	-0.28±0.42	-0.11±0.00	-0.28±0.04	**0.40±0.14**	0.25±0.10
D to E	**3.97±0.23**	0.45±1.68	-0.73±2.47	2.18±0.00	3.28±0.04	2.02±0.16	3.46±0.05
D to K	**7.20±0.27**	1.74±2.18	-0.02±3.96	6.85±0.00	7.09±0.02	5.75±0.08	7.51±0.03
E to B	0.51±1.05	-3.03±1.59	-2.83±1.36	-0.71±0.00	**1.55±0.02**	0.08±0.18	1.73±0.10
E to D	-0.05±0.38	-2.18±1.42	-1.53±1.12	0.14±0.00	**0.96±0.02**	-0.02±0.21	1.06±0.10
E to K	0.14±1.88	0.98±0.73	1.91±0.57	2.49±0.00	**2.55±0.03**	2.53±0.06	2.82±0.01
K to B	-0.07±0.48	-0.27±0.50	0.34±0.34	0.45±0.00	**1.06±0.02**	-0.07±0.42	1.72±0.07
K to D	-1.05±0.45	1.83±0.75	2.23±0.56	1.51±0.00	**2.45±0.03**	0.85±0.99	3.16±0.06
K to E	-1.17±0.70	-1.47±0.75	-1.45±1.37	-1.22±0.00	**-0.34±0.02**	-0.40±0.04	0.41±0.02
avg	1.68	0.45	0.52	2.15	**2.81**	1.95	3.43

(a) Improvements of accuracy of NN predictive models.

Task	DAN	DANN	MCD	CORAL	NN-CDA (**ours**)	NN-SCDA (**ours**)	NN-WCDA (sup) (**ours**)
B to D	-0.92±0.30	-4.47±1.14	-2.66±2.64	0.84±0.00	**0.87±0.01**	-0.02±0.06	1.48±0.01
B to E	**9.64±0.60**	5.72±2.35	6.62±1.96	6.86±0.00	8.78±0.00	6.83±0.56	11.33±0.01
B to K	10.37±1.04	11.18±2.56	10.28±3.67	14.12±0.00	**16.27±0.04**	13.82±0.22	18.51±0.00
D to B	-0.75±0.46	-1.72±0.77	-0.29±0.55	-0.25±0.00	-0.21±0.04	**0.31±0.04**	0.59±0.01
D to E	4.35±0.50	0.19±1.44	0.15±2.72	2.77±0.00	**5.36±0.04**	3.16±0.20	5.85±0.00
D to K	7.57±0.50	1.36±1.88	-0.07±4.31	6.40±0.00	**9.23±0.02**	6.68±0.02	9.80±0.00
E to B	**2.29±1.46**	-5.45±4.40	-8.01±3.90	-4.60±0.00	2.21±0.00	-0.06±0.52	4.75±0.02
E to D	-0.11±0.75	-2.90±2.48	-3.91±2.50	-4.67±0.00	**0.64±0.03**	-2.35±0.15	2.11±0.01
E to K	0.57±3.44	1.04±1.24	3.30±0.97	5.35±0.00	**6.44±0.03**	6.23±0.11	6.99±0.00
K to B	-1.24±1.11	1.14±0.87	1.60±0.76	-2.79±0.00	**1.84±0.02**	-0.51±0.41	4.22±0.03
K to D	-2.64±0.95	1.80±0.94	2.44±0.58	-2.38±0.00	**2.80±0.01**	-0.95±1.38	4.80±0.05
K to E	-0.96±1.22	-3.84±1.13	-2.28±2.24	-1.63±0.00	**-0.53±0.03**	0.02±0.07	1.39±0.00
avg	2.35	0.34	0.60	1.67	**4.47**	2.76	5.99

(b) Improvements of log-loss of NN predictive models.

Table 4.6: Adaptation performances of **LGB unsupervised** adaptation methods over **Amazon datasets** (see Section 2.8.3 for the dataset description). We report percentages of performances improvements, in accuracy and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Task	LGB Baseline	CORAL	LGB-CDA (**ours**)	LGB-SCDA (**ours**)	LGB-WCDA (sup) (**ours**)
B to D	-2.86±0.61	-3.90±0.00	**-2.61±0.06**	-2.75±0.00	-1.54±0.36
B to E	5.06±0.70	0.86±0.00	4.84±0.05	**5.26±0.28**	5.58±0.02
B to K	7.75±0.70	6.63±0.00	8.82±0.03	**8.88±0.19**	9.73±0.04
D to B	-1.19±0.28	**0.43±0.00**	-0.04±0.09	-1.12±0.36	0.46±0.11
D to E	-1.81±1.47	-0.30±0.00	**2.23±0.06**	-1.42±1.02	3.10±0.03
D to K	-0.99±1.45	3.95±0.00	**5.40±0.05**	4.57±2.50	7.43±0.09
E to B	-0.04±0.30	-0.52±0.00	**-0.01±0.04**	0.06±0.08	1.00±0.13
E to D	-0.22±0.32	-1.20±0.00	-0.28±0.08	**0.18±0.14**	0.42±0.06
E to K	0.28±0.35	0.99±0.00	1.48±0.05	**1.51±0.09**	1.92±0.03
K to B	-2.43±0.38	-0.20±0.00	**0.12±0.03**	-0.63±0.47	-0.23±0.06
K to D	1.72±0.38	1.92±0.00	1.85±0.04	**2.01±0.26**	3.16±0.05
K to E	-2.58±0.18	**-0.90±0.00**	-1.28±0.02	-0.93±0.05	-1.02±0.02
avg	0.22	0.65	**1.71**	1.30	2.50

(a) Improvements of accuracy of LGB predictive models.

Task	LGB Baseline	CORAL	LGB-CDA (**ours**)	LGB-SCDA (**ours**)	LGB-WCDA (sup) (**ours**)
B to D	-4.24±4.55	-3.08±0.00	-1.97±0.02	**-1.87±0.04**	-1.27±0.19
B to E	1.14±5.99	2.77±0.00	**5.58±0.01**	5.53±0.25	8.50±0.02
B to K	6.35±5.46	9.02±0.00	11.75±0.02	**11.93±0.51**	14.79±0.02
D to B	-3.13±0.33	-2.83±0.00	**-1.59±0.04**	-1.79±0.21	-0.91±0.08
D to E	-2.19±0.81	0.17±0.00	**3.30±0.03**	0.95±0.97	7.11±0.02
D to K	-0.68±0.55	4.63±0.00	**6.89±0.03**	6.24±2.32	10.89±0.04
E to B	1.57±0.70	-0.46±0.00	**2.07±0.01**	1.46±0.07	4.14±0.05
E to D	-0.54±1.00	-1.91±0.00	**0.74±0.02**	0.33±0.15	2.08±0.04
E to K	-0.67±2.73	3.44±0.00	**4.83±0.02**	4.64±0.10	5.11±0.03
K to B	-0.22±0.41	1.10±0.00	**2.38±0.04**	1.64±0.39	3.08±0.05
K to D	1.27±0.35	0.98±0.00	**2.50±0.02**	1.36±0.14	4.05±0.01
K to E	-6.55±1.49	-1.41±0.00	**-1.50±0.01**	-1.79±0.02	-1.04±0.05
avg	-0.66	1.04	**2.91**	2.39	4.71

(b) Improvements of log-loss of LGB predictive models.

Table 4.7: Numbers of adapted features of Amazon review tasks in a weakly supervised setting.

Task	NN-SCDA	LGB-SCDA	NN-WCDA	LGB-WCDA
B to D	7.0±2.8	4.8±2.1	212.6±3.9	313.0±2.0
B to E	11.3±5.3	5.5±1.1	227.2±2.2	322.6±0.5
B to K	10.0±2.6	5.6±1.5	208.8±1.9	324.2±0.7
D to B	1.9±1.1	3.1±0.8	175.0±2.6	264.8±1.7
D to E	9.7±4.6	8.0±2.4	249.6±0.8	264.8±0.7
D to K	7.4±2.1	8.8±3.5	259.2±2.6	281.4±2.0
E to B	3.2±1.6	4.9±2.7	194.2±1.7	317.0±0.9
E to D	19.6±2.3	4.5±0.9	202.6±0.8	305.6±0.8
E to K	5.5±2.7	3.7±1.5	209.4±1.0	314.2±2.1
K to B	7.8±3.4	3.8±2.6	200.2±3.9	327.4±1.4
K to D	8.5±2.5	6.2±2.9	215.4±3.2	304.8±0.4
K to E	30.5±5.0	6.1±2.8	186.2±3.1	335.2±1.7
AVG	10.2	5.4	211.7	306.3

4.4.2.3 Amazon Dataset

As introduced in Section 2.8, different from fraud detection datasets, features of Amazon reviews datasets are generated using a particular neural network: auto-encoder. As a result, individual features may not have interpretable meanings. Tables 4.5 and 4.6 provide results of Amazon reviews datasets using LGB and NN models in log-loss and accuracy. Adaptation results appear to show that SCDA does not improve performances compared to the all adaptation method (CDA). However, when we look at the number of features adapted by SCDA in Table 4.7, LGB-SCDA adapts on average only 5.4 features, and NN-SCDA selects 10.2 features among 400. Compared to WCDA, SCDA selects far fewer dimensions.

The forward feature selection process appears to stop at a very early step and seems to be stuck at local minima. We explain this phenomenon by the fact that features generated by neural networks are highly correlated (see Figure 3.7). Indeed, the greedy algorithm selects only one feature that outperforms all the others in each iteration. When there are many similar dimensions, such as the case of Amazon review datasets, differences of improvement between similar dimensions are marginal. The greedy algorithm stops as no feature is significantly better than the others. One possible solution to this problem is decreasing the acceptance threshold of a feature. Currently, a feature is added into the subset of features to adapt if most bootstrap results agree (the acceptance threshold = 0.5). By decreasing this value, one can include more features whose significances are impacted by correlated features.

In a classical tabular dataset, since all features are generated manually and

redundant features are removed, it is less common to have highly correlated features like the ones generated by neural networks. Nevertheless, by adapting on average 5.4 features for GBDT models and 10.2 features for NN models, our feature selection method SCDA achieves the second-best among other adaptation methods in Tables 4.5b, 4.6a, and 4.6b.

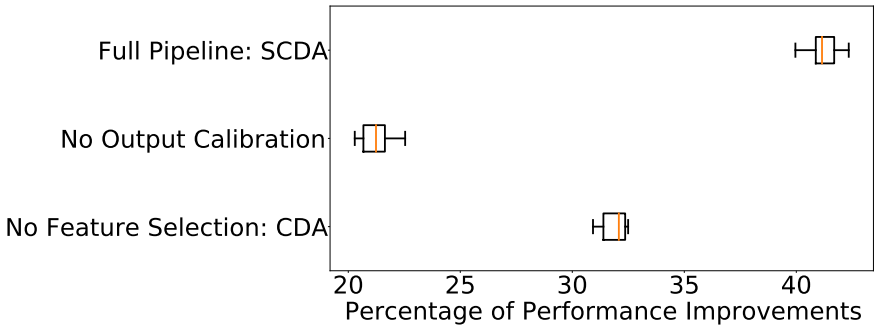
4.4.3 Ablation Study

This section aims to disentangle how each step of the proposed SCDA method helps the adaptation (Recall that the three main steps of our method are illustrated by Figure 4.1) and reveals challenges faced by the unsupervised feature selection method.

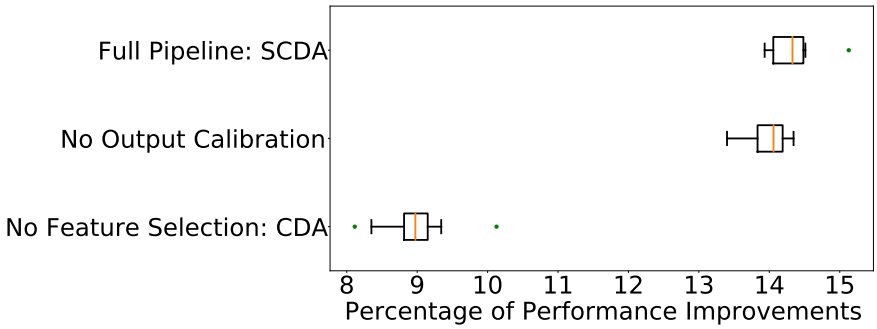
Figure 4.6 illustrates the log-loss improvements of our proposition under three different settings: with full three steps, without the first step, and without the third step. Of note, one cannot solely eliminate the second step of our proposition, as the third step selects the adapted dimensions of the second step. We clearly see that all three steps of our proposition contribute to the domain adaptation. Eliminating one of them results in a decrease in adaptation performances. However, the impact of each step is different. Removing the *label shift* correction step has less impact than removing the feature selection step on two of the studied task (except for Figure 4.6a). Our proposed unsupervised feature selection method, which is one of the core contributions of this work, is well-suited for this task.

4.5 Conclusion

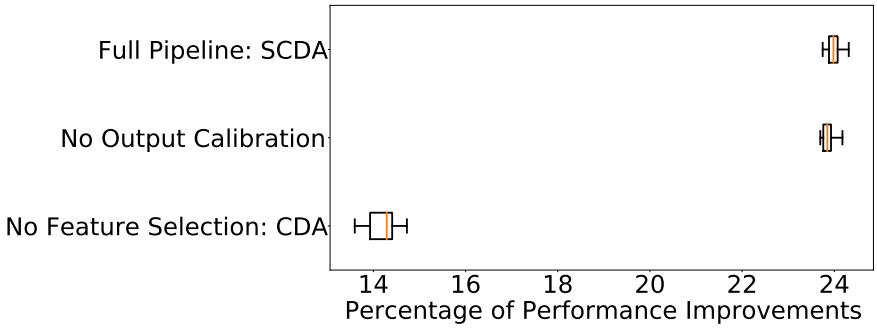
This chapter extended the weakly supervised adaptation pipeline Figure 3.6 to an unsupervised case by providing a new pseudo-label estimator leveraging the rank stability of predictions. Namely, we showed that the minimization problem of relaxed pseudo-labels is an upper bound of the target domain risk minimization problem. Both theoretical and empirical studies showed the efficiency of our proposed unsupervised adaptation pipeline (Figure 4.1). Chapters 3 and 4 addressed a single-target to single-source domain adaptation problem, while multi hidden subdomains commonly exist. Therefore, the following chapter extends our unsupervised (weakly supervised) adaptation pipeline to a multi-subdomain case.



(a) D-1 to M-1



(b) D-2 to M-2



(c) D-3 to M-3

Figure 4.6: The log-loss improvements of Kaggle fraud detection tasks with LGB models. "Full Pipeline: SCDA" represents our adaptation model with full 3 steps (see Figure 4.1). Whereas "No Output Calibration" skips the first step, and "No Feature Selection: CDA" skips the third step.

Chapter 5

Multi-Subdomain Adaptation

This chapter presents the third contribution of this thesis. Results of this chapter have been published in the paper "Interpretable Domain Adaptation for Hidden Subdomain Alignment in the Context of Pre-trained Source Models" [169]. We propose to explore the intra-domain heterogeneity to create subdomains and tackle a multi-subdomain target to source adaptation problem. To this end, we propose both a general subdomain separation criterion and a temporal drift specialized one. We introduce a new subdomain combination method leveraging a variable number (possibly unknown) of subdomains for target label predictions. By providing interpretability at two complementary levels (transformation, similar to previous chapters, and subdomain levels, specificity of this chapter), our method can also be easily understood by business experts with or without machine learning backgrounds.

5.1 Hidden Subdomain Exploration

The previous chapters proposed a *target to source* domain adaptation method leveraging one-dimensional optimal transport to address a domain adaptation problem with a pre-trained source model. Moreover, a feature selection process was applied to identify the most significant features for domain adaptation in both weakly supervised and unsupervised settings. The proposed method was *model-agnostic*, *feature-type free*, and *easy to interpret*.

This previously proposed paradigm tackled a *single-source single-target* (*single-domain*) domain adaptation setting by considering data of a target (source) domain obey the same distribution. However, the intra-domain drifts often naturally exist. A specific example consists of payment habit change according to seasons. Therefore, this chapter addresses a *multi-source multi-target* domain adaptation case to tackle such a situation.

More specifically, we consider both source and target domains can be subdivided into data from different distributions, the so-called subdomains. Furthermore, subdomain labels are rarely provided; thus one should propose methods to annotate them automatically. As discrepancies between subdomains of the same domain are not as significant as discrepancies between source and target domains, such intra-domain drifts may be omitted and even undiscovered as "hidden" when training a predictive model for a single domain. However, identifying such hidden subdomains contributes to domain adaptation by increasing the precision and flexibility of adaptation methods. Some recent works [158, 115] that study *multi-source* (resp. target) domain adaptation problems focus on a scenario where subdomain labels are provided. In contrast, we address a more challenging case where one needs to discover these hidden subdomains. Although Gong et al. [48], Mancini et al. [95] tackle such a hidden subdomain discovering problem, they assume that the number of hidden subdomains is known *a priori*. In this chapter, we propose a method reweighting different target domain classifiers adapted from the best combination of subdomains. Therefore, it is not necessary to know the number of subdomains *a priori*. Precisely, we first provide a general separation criterion to exploit hidden subdomains. Then we specialize our proposition to a practical scenario where data drift in each domain is imputed to time. We name our method **H***idden* **S***ubdomain* **A***daptation with* **V***ariable Number of Subdomains* (HSAV).

5.1.1 Notation

We adopt the same notation as the previous chapters. Recall that, we denote the input (resp. output) space of predictive models of source and target domains as $\mathcal{X}$ (resp. $\mathcal{Y}$). As we focus on a binary classification problem (fraud detection tasks), $\mathcal{Y}=\{0,1\}$, and the pre-trained source domain predictor $h_s(\boldsymbol{x}) : \mathcal{X}\to[0,1]$ gives the probability that one example $\boldsymbol{x}$ is classified as 1 (fraudulent). In a *target to source* domain adaptation setting, $h_s(\boldsymbol{x})$ is given as a black-box classifier of a wide variety of types (*e.g.*, neural networks, GBDT). We denote by $\mathcal{X}_{\text{sub}}$ the feature space that encodes hidden subdomains. Note that features in $\mathcal{X}$ and $\mathcal{X}_{\text{sub}}$ can be different. $\mathcal{X}$ stands for the discriminative attributes in predicting class labels, whereas $\mathcal{X}_{\text{sub}}$ contains attributes that help discover hidden subdomains and may not be discriminative in classification; thus, $\mathcal{X}_{\text{sub}}$ can contain attributes that are not in $\mathcal{X}$.

Recall that, X^t and X^s are respectively the target and source domain input variables over the support $\mathcal{X}$, and $P(X^t)$ and $P(X^s)$ represent their distributions. Analogously, we let X_i^t and X_j^s be the marginal variables of corresponding

subdomains, and $P(X_i^t)$ and $P(X_j^s)$ be subdomain distributions. Thus,

$$P(X^t) = \sum_{i=1}^{k_t} \pi_i^t P(X_i^t)\,, \text{ and } P(X^s) = \sum_{j=1}^{k_s} \pi_j^s P(X_j^s)\,,$$
$$\text{s.t. } \forall i\,, \pi_i^t > 0\,, \text{ and } \forall j\,, \pi_j^s > 0\,,$$
$$\sum_{i=1}^{k_t} \pi_i^t = 1\,, \text{ and } \sum_{j=1}^{k_s} \pi_j^s = 1\,.$$

where π_i^t and π_j^s are proportions of subdomains, and $k_s \in \{1, \ldots, k_s^{\text{sup}}\}$ and $k_t \in \{1, \ldots, k_t^{\text{sup}}\}$ refer to the number of subdomains. $k_t^{\text{sup}} \in \mathbb{N}^*$ and $k_s^{\text{sup}} \in \mathbb{N}^*$ stand respectively for the maximum number of subdomains that we consider. $\mathbb{X}^t$ (resp. $\mathbb{X}^s$) contains target (resp. source) domain examples drawn from $P(X^t)$ (resp. $P(X^s)$). Analogously, we define $\mathbb{X}_i^t = \{\boldsymbol{x}_l^t\}_{l=1}^{n_i^t}$, $\mathbb{X}_j^s = \{\boldsymbol{x}_l^s\}_{l=1}^{n_j^s}$ sets of n_i^t and n_j^s examples of target and source subdomains respectively drawn from $P(X_i^t)$ and $P(X_j^s)$. For compactness, we denote by $\mathbb{X}_{\text{sub}} = \{\mathbb{X}_1^t, \ldots, \mathbb{X}_{k_t}^t, \mathbb{X}_1^s, \ldots, \mathbb{X}_{k_s}^s\}$ a set of all target and source subdomains.

Moreover, we assume that there exists a mapping matrix $\boldsymbol{S} \in \{0,1\}^{k_t \times k_s}$ that relates hidden target subdomains to the source ones. We denote by $S_{i,j}$ the scalar located at the i-th row and the j-th column of $\boldsymbol{S}$; $\boldsymbol{S}_{i,:}$ and $\boldsymbol{S}_{:,j}$ represent the row and column vectors. $S_{i,j}$ takes the value 1 if the target subdomain X_i^t is related to the source subdomain X_j^s, or the value 0 otherwise. Furthermore, to enhance the interpretability, we encourage $\boldsymbol{S}$ to be sparse. Typically, we want one target hidden subdomain maps to only one source hidden subdomain, that is, $\forall i \in \{1, \ldots, k_t\}\,, \sum_{j=1}^{k_s} S_{i,j} = 1$. In the following, we first provide details of our adaptation methods by starting with a known number of subdomains, and then we generalize our method to handle a variable number of subdomains.

5.1.2 Formalization

Definition 5.1 (Target domain predictive model in a multi-subdomain adaptation case)**.** Let first assume that we face a domain adaptation problem from which we know *a priori* the underlying number of target and source subdomains k_t and k_s, and let $\boldsymbol{K} = (k_t, k_s)$. Given $\boldsymbol{x} \in \mathbb{X}^t$, we formalize the target domain classifier as

$$h_t(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}, \boldsymbol{S}) = \sum_{i=1}^{k_t} p_i(\boldsymbol{x}; \mathbb{X}_{\text{sub}}) h_t^i(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}, \boldsymbol{S})\,, \tag{5.1}$$

where $h_t^i(\cdot)$ is a predictor of the i-th adapted target subdomain, and $p_i(\boldsymbol{x}; \mathbb{X}_{\text{sub}})$ is the probability that a target example $\boldsymbol{x}$ belongs to this subdomain.

More precisely, we have

$$p_i(\boldsymbol{x}; \mathbb{X}_{\text{sub}}) = \frac{\pi_i^t P(X_i^t = \boldsymbol{x})}{\sum_{k=1}^{k_t} \pi_k^t P(X_k^t = \boldsymbol{x})} .$$

Of note, our experiments involve both categorical and numerical features. As it is not straightforward to compute densities of examples with mixed types of input space, we consider the i-th subdomain density $P(X_i^t = \boldsymbol{x})$ as a product of densities of each dimension of $\boldsymbol{x}$. Furthermore, we formalize the classifier of the i-th subdomain of target as

$$h_t^i(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}, \boldsymbol{S}) = \sum_{j=1}^{k_s} S_{i,j} h_s \circ \mathcal{G}_{i,j}^{\mathcal{D}^*}(\boldsymbol{x}; \mathbb{X}_{\text{sub}}) , \tag{5.2}$$

where $\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot; \mathbb{X}_{\text{sub}})$ is the optimal *coordinate-wise single-domain* domain adaptation function that transforms data from the i-th target subdomain $\mathbb{X}_i^t$ to the j-th source subdomain $\mathbb{X}_j^s$. $\mathcal{D}^*$ is the optimal subset of features to adapt, and the estimation of $\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot; \mathbb{X}_{\text{sub}})$ follows the pipeline that we proposed in the previous chapters (see Figure 4.1). This chapter focuses on the aggregation of these adaptation functions.

With a known subdomain number $\boldsymbol{K}$, the hidden subdomain adaptation consists of estimating the optimal subdomain separations $\mathbb{X}_{\text{sub}}$ and the mapping relation $\boldsymbol{S}$ between them. In the remainder of this chapter, we first give the criterion to estimate $\mathbb{X}_{\text{sub}}$, as the formula is the same for both weakly supervised and unsupervised adaptation scenarios. Then we detail estimations of $\boldsymbol{S}$ for each case in an individual section.

5.1.2.1 Estimation of $\mathbb{X}_{\text{sub}}$.

Logically, separations of subdomains are significant if inter-subdomain discrepancies are large. If subdomains were similar, they could be adapted using the same transformation, and there would be no need to distinguish them. Moreover, in a predictor weighting formalization as Equations (5.1) and (5.2), one benefits from a diversity between weighted elements. Here, such diversity is inherited from the differences between subdomains, as they likely spawn diverse transformations $\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot; \mathbb{X}_{\text{sub}})$. Following the same convention of Gong et al. [48] and Hoffman et al. [56], we search the optimal $\mathbb{X}_{\text{sub}}$ by maximizing inter-subdomain discrepancies. That is,

$$\mathbb{X}_{\text{sub}}^* = \underset{\mathbb{X}_{\text{sub}}}{\operatorname{argmax}} \left[\sum_{i \neq j}^{k_t} d_{\text{w}}(\mathbb{X}_i^t, \mathbb{X}_j^t) + \sum_{i \neq j}^{k_s} d_{\text{w}}(\mathbb{X}_i^s, \mathbb{X}_j^s) \right] , \tag{5.3}$$

where $d_{\mathrm{w}}(\cdot,\cdot)$ is a domain discrepancy measure. In our case, as we focus on adapting tabular data where the input space contains categorical and numerical attributes, $d_{\mathrm{w}}(\cdot,\cdot)$ is chosen to be the sum of one-dimensional Wasserstein distances over each feature.

5.1.3 Specialization to Temporal Drift

In our fraud detection tasks, we focus on a practical case where the feature $\mathcal{X}_{\mathrm{sub}}$ that encodes hidden subdomains is one temporal dimension (the time). Such a scenario is very common in real-life applications where data arrive as time goes on, and there is a drift between collected data. For example, in a payment fraud detection system, payment habits are different due to the change of seasonality. Moreover, the time is a one-dimensional feature that can be efficiently subdivided. Consequently, in such a setting, subdomains $\mathbb{X}_i^t$ and $\mathbb{X}_j^s$ are parameterized by separations of time $\{t_1^t,\ldots,t_{k_t-1}^t\}$ and $\{t_1^s,\ldots,t_{k_s-1}^s\}$ respectively. We have

$$\begin{aligned}&\forall \boldsymbol{x}\in\mathbb{X}_i^t, t_{i-1}^t < t(\boldsymbol{x}) \leq t_i^t, \text{and}\\ &\forall \boldsymbol{x}\in\mathbb{X}_j^s, t_{j-1}^s < t(\boldsymbol{x}) \leq t_j^s,\end{aligned}$$

where $t(\boldsymbol{x}):\mathcal{X}\to\mathcal{X}_{\mathrm{sub}}$ gives the hidden subdomain feature (the time in our case) of input $\boldsymbol{x}$. As we separate subdomains following the axis of time, instead of computing inter-subdomain discrepancies between every pair of subdomains, we take into account only discrepancies between successive subdomains. Equation (5.3) is redefined as

$$\mathbb{X}_{\mathrm{sub}}^* = \underset{\mathbb{X}_{\mathrm{sub}}}{\operatorname{argmax}}\left[\sum_{i=1}^{k_t-1} d_{\mathrm{w}}(\mathbb{X}_i^t,\mathbb{X}_{i+1}^t)+\sum_{j=1}^{k_s-1} d_{\mathrm{w}}(\mathbb{X}_j^s,\mathbb{X}_{j+1}^s)\right]. \tag{5.4}$$

An example of separations is illustrated in Figure 5.1. Solving Equation (5.4) consists of finding the optimal separations of time $\{t_1^t,\ldots,t_{k_t-1}^t\}$ and $\{t_1^s,\ldots,t_{k_s-1}^s\}$.

Some recent domain adaptation works [9, 154] also address the temporal drift, while they do not apply to our case. The key difference between their propositions and ours is that we adapt from target subdomains to source subdomains, whereas they align subdomains data in the same domain.

5.2 Weakly Supervised Subdomain Aggregation

In a weakly supervised setting where $\mathbb{Q}_l^t$ with n_t^l labeled target data is given, similar to the previously proposed feature selection process (Section 3.5), one can minimize the prediction error over few labeled target domain points to estimate $\boldsymbol{S}$.

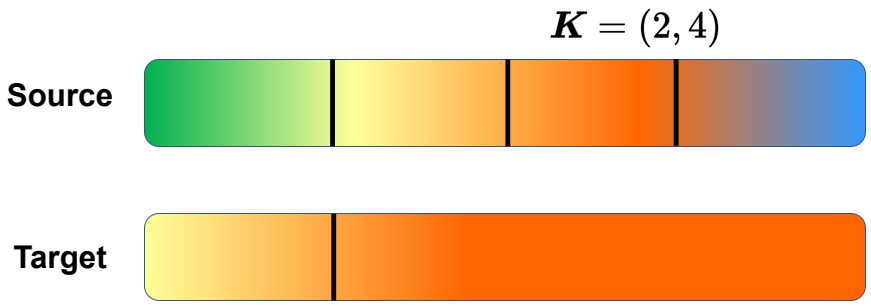


Figure 5.1: Separations of source and target domains into subdomains. Graduated color represents a continuous intra-domain drift. The source domain is subdivided into 4 subdomains, and the target domain is subdivided into 2 subdomains.

5.2.1 Known Number of Subdomains

In a classical case where the number of hidden subdomains $\boldsymbol{K}$ is known (adopted by most of multi-subdomain adaptation methods), the optimization problem is formulated as

$$\begin{aligned} \boldsymbol{S}^* = \underset{\boldsymbol{S}}{\operatorname{argmin}} \frac{1}{n_t^l} \sum_{(\boldsymbol{x},y)\in\mathbb{Q}_l^t} l(h_t(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}^*, \boldsymbol{S}), y)\,, \\ \text{s.t.} \sum_{j=1}^{k_s} S_{i,j} = 1\,, \forall i \in \{1, \ldots, k_t\}\,. \end{aligned} \tag{5.5}$$

For our fraud detection tasks, the loss function $l(\cdot)$ is chosen to be a binary cross-entropy loss. As $S_{i,j} \in \{0, 1\}$, such a discrete optimization problem is computational expensive and not scalable when k_s and k_t are large. Alternatively, we leverage a Softmax function to approximate $\boldsymbol{S}$. We provide implementation details in Section 5.4.

Figure 5.2 illustrates an example of a possible mapping matrix $\boldsymbol{S}$. The first target subdomain is mapped to the second source subdomain, and the second target subdomain is mapped to the third source subdomain.

5.2.2 Unknown Number of Subdomains

In many real-life scenarios, the number of hidden subdomains is unknown. In such cases, a natural choice is to estimate the optimal couple $\boldsymbol{K}$, using weakly labeled target examples, that is,

$$\boldsymbol{K}^* = \underset{\boldsymbol{K}}{\operatorname{argmin}} \frac{1}{n_t^l} \sum_{(\boldsymbol{x},y)\in\mathbb{Q}_l^t} l(h_t^\dagger(\boldsymbol{x}; \boldsymbol{K}), y)\,,$$

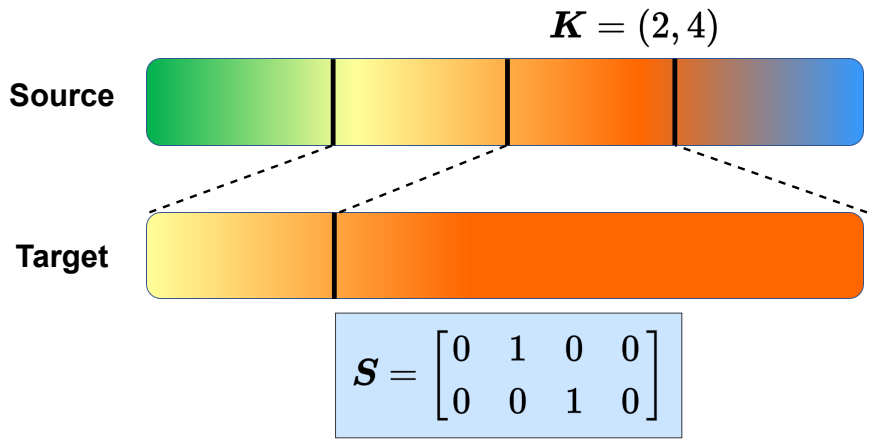


Figure 5.2: An example of mapping between target and source subdomains. Dotted lines map target subdomains to the corresponding source subdomains.

where $h_t^\dagger(\boldsymbol{x}; \boldsymbol{K}) = h_t(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}^*, \boldsymbol{S}^*)$. For a given $\boldsymbol{K}$, $h_t^\dagger(\cdot; \boldsymbol{K})$ is the estimated optimal target domain classifier of the previous section. However, we empirically observe that, when using a Bootstrap strategy, one can hardly find a single $\boldsymbol{K}^*$ that is significantly better than others. Indeed, the scarcity of $\mathbb{Q}_l^t$ leads to high variability in Bootstrap predictive performances.

Therefore, instead of using a single $h_t^\dagger(\boldsymbol{x}; \boldsymbol{K}^*)$ with the optimal estimated $\boldsymbol{K}^*$ as the target domain classifier, we propose aggregate multiple target predictors $h_t^\dagger(\cdot; \boldsymbol{K})$. Each $h_t^\dagger(\cdot; \boldsymbol{K})$ is obtained for different values of $\boldsymbol{K}$. The weight associated with each possible subdomain number is handled by the matrix $\boldsymbol{A} \in \mathbb{R}^{k_t^{\text{sup}} \times k_s^{\text{sup}}}$, such that the target domain predictive model becomes

$$h_t^*(\boldsymbol{x}; \boldsymbol{A}) = \sum_{k_t=1}^{k_t^{\text{sup}}} \sum_{k_s=1}^{k_s^{\text{sup}}} \sigma_{k_t,k_s}(\boldsymbol{A}) h_t^\dagger(\boldsymbol{x}; \overbrace{(k_t, k_s)}^{\boldsymbol{K}}) \,,$$

where $\sigma_{k_t,k_s}(\cdot)$ is a Softmax function that encourages a sparsity of the weighting factor:

$$\sigma_{k_t,k_s}(\boldsymbol{A}) = \frac{\exp(A_{k_t,k_s})}{\sum_{u=1}^{k_t^{\text{sup}}} \sum_{v=1}^{k_s^{\text{sup}}} \exp(A_{u,v})} \,.$$

The Softmax function is a smoothed version of one-hot encoding [49]. By sparsity, we mean that $\boldsymbol{A}$ tends to give most weights to one $\boldsymbol{K}$. Consequently, the objective function becomes

$$\boldsymbol{A}^* = \underset{\boldsymbol{A}}{\text{argmin}} \frac{1}{n_t^l} \sum_{(\boldsymbol{x},y) \in \mathbb{Q}_l^t} l(h_t^*(\boldsymbol{x}; \boldsymbol{A}), y) \,, \tag{5.6}$$

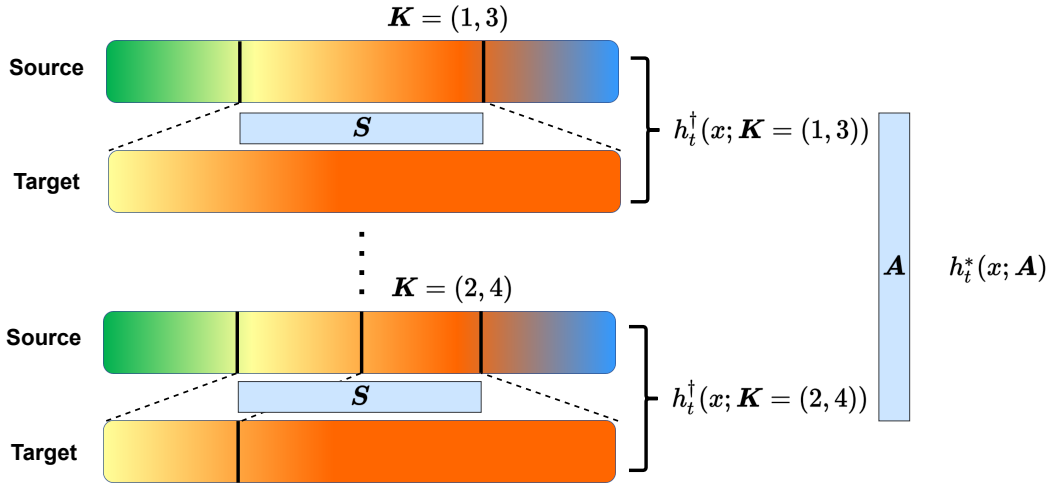


Figure 5.3: An example of subdomain aggregation. The final prediction results are given by a $\boldsymbol{A}$-weighted sum of $h_t^\dagger(\boldsymbol{x}; (k_t, k_s))$.

and the corresponding target domain predictor is $h_t^*(\cdot; \boldsymbol{A}^*)$.

Figure 5.3 illustrates an example of aggregation. As the number of subdomains is unknown, one computes $h_t^\dagger(\boldsymbol{x}; (k_t, k_s))$ for different values of $\boldsymbol{K}$ and aggregates them using $\boldsymbol{A}$.

5.3 Unsupervised Subdomain Aggregation

When $\mathbb{Q}_l^t$ is not available, we face a more challenging unsupervised domain adaptation case. The unsupervised subdomain aggregation method follows the same idea as the supervised one: one separates subdomains, maps subdomains, and aggregates different numbers of subdomains. However, the objective functions to estimate parameters $\boldsymbol{S}$ and $\boldsymbol{A}$ change, as no target domain labels are provided.

It is known that the given pre-trained source domain predictive model $h_s(\cdot)$ is often well trained to be the optimal one in source domains. Under such a setting, Equation (4.3) shows that any optimal *target to source* domain adaptation function should align source and target domain output distributions. Specifically, when $\boldsymbol{K}$ is known, the necessary condition for $\boldsymbol{S}^*$ to be the optimal one is

$$P(h_t(X^t; \boldsymbol{K}, \mathbb{X}_{\text{sub}}^*, \boldsymbol{S}^*)) = P(h_s(X^s))\,. \tag{5.7}$$

When we have a variable number of subdomains, a necessary condition of the optimal $\boldsymbol{A}^*$ is

$$P(h_t^*(X^t; \boldsymbol{A}^*)) = P(h_s(X^s))\,. \tag{5.8}$$

Relying on such conditions, we propose unsupervised objective functions for both known and unknown number of $\boldsymbol{K}$.

5.3.1 Known Number of Subdomains

Inspired by Equation (5.7), given $\boldsymbol{K}\in\{(k_t,k_s)|k_t\in\{1,\dots,k_t^{\text{sup}}\},k_s\in\{1,\dots,k_s^{\text{sup}}\}\}$, the unsupervised optimization problem of $\boldsymbol{S}$ is formulated as

$$\begin{aligned}\boldsymbol{S}^* = \underset{\boldsymbol{S}}{\operatorname{argmin}}\, d_{\text{1d-w}}(h_t(X^t;\boldsymbol{K},\mathbb{X}^*_{\text{sub}},\boldsymbol{S}),h_s(X^s))\,,\\ \text{s.t.}\sum_{j=1}^{k_s} S_{i,j}=1\,,\forall i\in\{1,\dots,k_t\}\,.\end{aligned} \tag{5.9}$$

where $d_{\text{1d-w}}(\cdot,\cdot)$ is the one-dimensional Wasserstein distance over the distribution of positive outputs. Empirically, $d_{\text{1d-w}}(\cdot,\cdot)$ is given by

$$\begin{aligned}d_{\text{1d-w}}(h_t(X^t;\boldsymbol{K},&\mathbb{X}^*_{\text{sub}},\boldsymbol{S}),h_s(X^s)) =\\ &\sum_{x\in\mathbb{X}^t}\Big(h_t(\boldsymbol{x};\boldsymbol{K},\mathbb{X}^*_{\text{sub}},\boldsymbol{S})-f^{-1}_{h_s}(h_t(\boldsymbol{x};\boldsymbol{K},\mathbb{X}^*_{\text{sub}},\boldsymbol{S}))\Big)^2\,,\end{aligned}$$

where $f_{h_s}(\cdot)$ is the cumulative distribution function of $h_s(X^s)$.

5.3.2 Unknown Number of Subdomains

Analogously, for a variable number of subdomains, inspired by Equation (5.8), we propose the following unsupervised objective function to mimic Equation (5.6):

$$\boldsymbol{A}^* = \underset{\boldsymbol{A}}{\operatorname{argmin}}\, d_{\text{1d-w}}(h^*_t(\boldsymbol{x};\boldsymbol{A}),h_s(X^s))\,. \tag{5.10}$$

In the implementation section, we give optimization details of our proposed objective functions.

5.4 Implementation

For both weakly supervised and unsupervised cases, the proposed HSAV consists of 3 steps.

1. We estimate separations of subdomains $\mathbb{X}_{\text{sub}}$ w.r.t. different numbers of subdomains (Figure 5.1).
2. We estimate the corresponding sparse mapping factor $\boldsymbol{S}$ (Figure 5.2).
3. We combine predictions of variable numbers of subdomains relying on $\boldsymbol{A}$ (Figure 5.3).

Algorithm 5 Subdomains Aggregation

for k_t **in** $\{1, \dots, k_t^{\text{sup}}\}$ **do**
  Get target subdomains $\{\mathbb{X}_1^t, \dots, \mathbb{X}_{k_t}^t\}$ relying on Equation (5.4).
  for k_s **in** $\{1, \dots k_s^{\text{sup}}\}$ **do**
    Get source subdomains $\{\mathbb{X}_1^s, \dots, \mathbb{X}_{k_s}^s\}$ relying on Equation (5.4).
    for i **in** $\{1, \dots k_t\}$ **do**
      for j **in** $\{1, \dots k_s\}$ **do**
        Estimate $\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot)$ (Equation (5.2)).
      end for
    end for
    Estimate $\boldsymbol{S}$ (Equations (5.5), (5.9)).
  end for
end for
Estimate $\boldsymbol{A}$.

Algorithm 5 shows the estimation of parameters at each step.

Optimization over $\mathbb{X}_{\text{sub}}$ Since Equation (5.4) is not differentiable due to the existence of categorical features, we rely on the Nelder-Mead method [103] to solve this objective function. Note that, since the two sums of Equation (5.4) are independent, one can solve both parts individually. The maximum number of subdomains is determined by gradually increasing the number of subdomains and solving Equation (5.4). Then we rely on an empirical criterion to define the stop condition. Empirically, every subdomain $\mathbb{X}_i^t$ (or $\mathbb{X}_j^s$) should have a minimum number of examples such that one can precisely estimate its distribution.

Namely, we start from $k_t = 2$ (resp. $k_s = 2$) and compute the subdomain separations. We increase k_t (resp. k_s) by 1 if all subdomains have at least n_m examples, that is, $\forall i \in \{1, \dots, k_t\}, n_i^t > n_m$ (resp. $\forall j \in \{1, \dots, k_s\}, n_j^s > n_m$). Otherwise, we stop the process and take the current value of k_t (resp. k_s) as the maximum number of target (resp. source) subdomains k_t^{sup} (resp. k_s^{sup}).

Optimization over $\boldsymbol{S}$ Since the discrete optimization problems (Equations 5.5 and 5.9) are computationally expensive, instead of directly searching $\boldsymbol{S}$ in the discrete space $\{0,1\}^{k_t \times k_s}$, we relax the constraints over $\boldsymbol{S}$ relying on a Softmax function by row. Specifically, we set

$$\omega_{i,j}(\tilde{\boldsymbol{S}}) = \frac{\exp(\tilde{S}_{i,j})}{\sum_{k=1}^{k_s} \exp(\tilde{S}_{i,k})},$$

with $\tilde{\boldsymbol{S}} \in \mathbb{R}^{k_t \times k_s}$, and $\omega(\tilde{\boldsymbol{S}})$ is a matrix where the i-th row the j-th column is referred to as $\omega_{i,j}(\tilde{\boldsymbol{S}})$.

Therefore, in a weakly supervised scenario, we can optimize $\tilde{\boldsymbol{S}}$ in place of $\boldsymbol{S}$,

and Equation (5.5) becomes

$$\tilde{\boldsymbol{S}}^* = \underset{\tilde{\boldsymbol{S}}}{\operatorname{argmin}} \frac{1}{n_t^l} \sum_{(\boldsymbol{x},y)\in\mathbb{Q}_l^t} l(h_t(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}^*, \omega(\tilde{\boldsymbol{S}})), y)\,,$$

which can be solved by classical optimization methods (*e.g.*, gradient descent).

Analogously, in an unsupervised scenario, Equation (5.9) becomes

$$\tilde{\boldsymbol{S}}^* = \underset{\tilde{\boldsymbol{S}}}{\operatorname{argmin}}\, d_{\text{1d-w}}(h_t(X^t; \boldsymbol{K}, \mathbb{X}_{\text{sub}}^*, \omega(\tilde{\boldsymbol{S}})), h_s(X^s))\,.$$

As $f_{h_s}^{-1}(\cdot)$ is not differentiable, we solve it using an iterative method relying on the gradient descent. At each step (denoted by p) of iterations, the gradient is computed by

$$\nabla g_s = \frac{\partial \sum_{x\in\mathbb{X}^t} \left(h_t(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}^*, \omega(\tilde{\boldsymbol{S}})) - h^{\#}(\boldsymbol{x})\right)^2}{\partial \tilde{\boldsymbol{S}}}\,, \tag{5.11}$$

where $h^{\#}(\boldsymbol{x}) = f_{h_s}^{-1}(h_t(\boldsymbol{x}; \boldsymbol{K}, \mathbb{X}_{\text{sub}}^*, \omega(\tilde{\boldsymbol{S}}^{(p-1)}))$. Note that we compute $h^{\#}(\boldsymbol{x})$ using $\tilde{\boldsymbol{S}}^{(p-1)}$ of the previous iteration, which is considered as a constant with respect to $\tilde{\boldsymbol{S}}$ and does not contribute to the gradient of $\tilde{\boldsymbol{S}}^{(p)}$. Then we set $\tilde{\boldsymbol{S}}^{(p)} = \tilde{\boldsymbol{S}}^{(p-1)} + c_1 * \nabla g_s$ with c_1 as a learning rate and continue the process until the stop criterion is met. For both supervised and unsupervised cases, we set all $\tilde{S}_{i,j} = 0$ at initialization; thus the initial factor $\omega_{i,j}(\tilde{\boldsymbol{S}})$ between each pair of target-source subdomain is uniform.

To further reduce the computational cost of the proposed method, instead of using one $\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot; \mathbb{X}_{\text{sub}})$ between each pair of target-source subdomains, we compute a global transformation function $\mathcal{G}^{\mathcal{D}^*}(\cdot)$ that adapts X^t to X^s and let $\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot; \mathbb{X}_{\text{sub}}) = \mathcal{G}^{\mathcal{D}^*}(\cdot)$ during optimizations. The final discrete solution of $\boldsymbol{S}$ is

$$S_{i,j}^* = \begin{cases} 1 & \text{if } \tilde{S}_{i,j}^* = \max_k \tilde{S}_{i,k}^*\,, \\ 0 & \text{otherwise.} \end{cases} \tag{5.12}$$

Once the discrete factor $\boldsymbol{S}$ between target and source subdomains is determined, we re-estimate transformation functions $\mathcal{G}_{i,j}^{\mathcal{D}^*}(\cdot; \mathbb{X}_{\text{sub}})$ between the i-th subdomain of target and the j-the subdomain of source for all $S_{i,j}^* = 1$.

Optimization over $\boldsymbol{A}$ Following a similar approach as the optimization of $\tilde{\boldsymbol{S}}$, in a supervised setting, we solve Equation (5.6) using a gradient descent method.

Supposing that $\min(k_t^{\text{sup}}, k_s^{\text{sup}}) > 1$, at initialization, we set

$$A_{k_t,k_s} = \begin{cases} \log(3(k_t^{\text{sup}} \times k_s^{\text{sup}} - 1)) & \text{if } k_t{=}k_s{=}1\,, \\ 0 & \text{otherwise,} \end{cases}$$

which results the weighting factor of target classifiers of different numbers of subdomains to be

$$\sigma_{k_t,k_s}(\boldsymbol{A}) = \begin{cases} 3/4 & \text{if } k_t{=}k_s{=}1\,, \\ 1/(4(k_t^{\text{sup}} \times k_s^{\text{sup}} - 1)) & \text{otherwise.} \end{cases}$$

We set $\sigma_{k_t,k_s}(\boldsymbol{A})$=3/4 for k_t=k_s=1 to privilege this choice if subdomains cannot significantly improve prediction performances. Besides, the value of the gradient at $\sigma_{k_t,k_s}(\boldsymbol{A}) = 3/4$ is not too small to conduct an effective training.

In an unsupervised setting, we solve Equation (5.10) by the same approach as Equation (5.11). Namely, the gradient at the p-th step of iterations is computed by

$$\nabla g_u = \frac{\partial \sum_{x \in \mathbb{X}^t} \left(h_t^*(\boldsymbol{x}; \boldsymbol{A}) - h_A^{\#} \right)^2}{\partial \boldsymbol{A}}\,,$$

where $h_A^{\#}(\boldsymbol{x}) = f_{h_s}^{-1}(h_t^*(\boldsymbol{x}; \boldsymbol{A}^{(p-1)})$. Then $\boldsymbol{A}^{(p)} = \boldsymbol{A}^{(p-1)} + c_2 \nabla g_u$ with c_2 as a learning rate until the stop criteria is met.

Furthermore, we adopt a Bootstrap Bagging method [15] to enhance the robustness of estimated parameters. Precisely, in both supervised and unsupervised scenarios, $\tilde{\boldsymbol{S}}$ and $\boldsymbol{A}$ are estimated over 10 Bagging datasets, and the average value is used as the final estimation results. Then $\boldsymbol{S}$ is obtained relying on Equation (5.12) over the average of $\tilde{\boldsymbol{S}}$.

5.5 Experiments

The proposed method HSAV is evaluated over Kaggle and Worldline fraud detection datasets. However, HSAV is not suited to extract hidden subdomains of Amazon review datasets, as the domain separation process (Section 5.1.3) of HSAV addresses a temporal drift, whereas the Amazon review datasets do not have the time dimension. Details of datasets and separations of source and target domains are provided in Section 2.8.

5.5.1 General Setup

We use the same pre-trained NN and GBDT models as Chapters 3 and 4. Performances are evaluated using improvements of PR-AUC and log-loss compared to source domain NN baseline models.

Adaptation Methods of Comparison We compare HSAV with SCDA in an unsupervised setting and with WCDA in a weakly supervised setting. In addition to these *single-source single-target* domain adaptation methods, we also compare with deep multi-subdomain adaptation methods: DCTN [158], and MultiDA [95, 96]. All methods are estimated under different hyper-parameters, and the one that achieves the best performance over source domain test data is chosen.

Both DCTN and MultiDA tackle the case with a fixed number of subdomains. Namely, we fix the number of hidden subdomains to be 2 in the source domain and 1 in target domains for these two methods, as target domains contain a shorter period than the source one (Sections (2.8.2), (2.8.1)). Furthermore, Figure 5.4 confirms this choice. In a weakly supervised case, similar to Section 3.7, we also train a FineTune model by fine-tuning the last layer of NN pre-trained models using weakly labeled target data.

Hyper-parameters of NN models In an unsupervised case, hyper-parameters are chosen based on the prediction performances of test data of the source domain. In contrast, in a weakly supervised case, hyper-parameters are chosen relying on test data of source domain and weakly-labeled target domain data. We define the hyper-parameter searching set as $\mathbb{H} = \{0.001, 0.003, 0.005, 0.007, 0.01\}$. For DAN and DANN, we fix the learning rate to 0.005 and search the weighting parameter between classification error and domain alignment error among $\mathbb{H}$. For all other methods (MCD, MultiDA, DCTN), we explore the learning rate among $\mathbb{H}$. As for the FineTune case, we freeze all layers except the last one and set the learning rate to 0.0005 to fine-tune the layer to fit weakly-labeled target domain data.

Table 5.1: Adaptation performances of **unsupervised** adaptation methods over **Kaggle** datasets (see Section 2.8.2 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%).

Method	D-1 to M	D-2 to M	D-3 to M	AVG
NN-HSAV (**ours**)	5.26±1.91	**3.24±0.31**	**5.51±0.33**	**4.67**
MultiDA	4.59±23.10	-7.29±8.36	-4.64±6.21	-2.45
DCTN	-16.87±18.39	-13.40±10.87	-8.59±5.59	-12.95
NN-SCDA (**ours**)	3.23±1.87	2.88±0.35	5.41±0.48	3.84
DAN	**12.31±6.72**	-3.20±7.18	1.41±3.63	3.51
DANN	3.47±9.53	-2.90±3.82	-4.21±6.51	-1.21
MCD	-11.47±13.38	-6.29±6.09	-6.81±4.83	-8.19

(a) Improvements of PR-AUC of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
NN-HSAV (**ours**)	**31.68±0.64**	**9.76±0.30**	**14.78±0.31**	**18.74**
MultiDA	-460.04±151.52	-219.19±68.03	-182.90±40.92	-287.38
DCTN	-8.65±41.45	-77.26±92.59	-32.67±27.63	-39.53
NN-SCDA (**ours**)	31.42±0.53	9.54±0.35	14.05±0.77	18.34
DAN	29.57±4.32	-1.58±9.64	4.37±5.39	10.79
DANN	29.58±4.05	2.39±3.98	2.06±5.07	11.34
MCD	21.59±9.25	-1.29±5.34	-7.55±10.32	4.25

(b) Improvements of log-loss of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB-HSAV (**ours**)	**32.87±1.31**	**7.41±0.28**	**14.43±0.24**	**18.24**
LGB-SCDA (**ours**)	32.68±1.05	7.14±0.32	14.31±0.24	18.04
LGB Baseline	26.13±1.53	4.18±1.47	7.58±3.26	12.63

(c) Improvements of PR-AUC of LGB predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB-HSAV (**ours**)	**41.64±0.43**	**14.47±0.30**	**24.08±0.16**	**26.73**
LGB-SCDA (**ours**)	41.22±0.64	14.34±0.34	24.00±0.16	26.52
LGB Baseline	9.77±6.01	6.53±3.31	12.32±5.05	9.54

(d) Improvements of log-loss of LGB predictive models.

Table 5.2: Adaptation performances of **weakly supervised** adaptation methods over **Kaggle** datasets (see Section 2.8.2 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%). Dotted lines separate single-domain adaptation and multi-subdomain adaptation methods.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
NN-HSAV (**ours**)	12.61±2.76	**5.64±1.09**	4.87±2.10	**7.70**
MultiDA	-5.71±19.92	-2.87±5.58	-0.25±4.85	-2.94
DCTN	-13.58±18.18	-11.77±7.92	-5.00±6.81	-10.12
NN-WCDA (**ours**)	1.30±7.57	2.98±2.14	3.72±1.40	2.66
DAN	**15.40±6.62**	-0.57±4.53	1.53±3.68	5.45
DANN	5.77±10.49	-0.87±3.82	-2.40±4.95	0.83
MCD	8.96±12.36	-2.48±5.03	-2.89±4.84	1.20
FineTune	4.23±0.72	2.12±0.15	**4.95±0.30**	3.76

(a) Improvements of PR-AUC of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
NN-HSAV (**ours**)	**32.62±1.90**	**11.72±1.18**	**14.31±1.55**	**19.55**
MultiDA	-0.27±26.13	-1.26±8.34	-4.85±11.91	-2.13
DCTN	-8.92±24.68	-39.48±23.07	-42.77±34.84	-30.39
NN-WCDA (**ours**)	30.42±3.26	8.95±2.67	12.07±2.04	17.14
DAN	31.70±3.58	3.02±5.34	5.94±5.52	13.55
DANN	31.03±4.33	3.16±4.25	5.78±4.62	13.32
MCD	30.28±3.59	1.26±5.39	3.74±5.45	11.76
FineTune	9.22±2.10	5.33±0.83	11.23±1.13	8.59

(b) Improvements of log-loss of NN predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB-HSAV (**ours**)	**29.52±3.03**	**8.49±0.62**	**14.51±0.72**	**17.51**
LGB-WCDA (**ours**)	27.85±4.41	7.18±1.85	13.66±1.56	16.23
LGB Baseline	26.13±1.53	4.18±1.47	7.58±3.26	12.63

(c) Improvements of PR-AUC of LGB predictive models.

Method	D-1 to M	D-2 to M	D-3 to M	AVG
LGB-HSAV (**ours**)	**40.61±1.46**	**15.92±0.70**	**23.65±0.78**	**26.73**
LGB-WCDA (**ours**)	39.86±1.68	14.31±2.40	22.06±1.66	25.41
LGB Baseline	9.77±6.01	6.53±3.31	12.32±5.05	9.54

(d) Improvements of log-loss of LGB predictive models.

5.5.2 Adaptation Performance Analysis

5.5.2.1 Kaggle Dataset

Table 5.1 reports performances of HSAV based on NN pre-trained model (NN-HSAV) and LGB pre-trained model (LGB-HSAV) in an unsupervised setting. HSAV outperforms all other adaptation methods in most tasks, while DAN only outperforms HSAV in "D-1 to M" in terms of PR-AUC. Note that performances of HSAV and SCDA are very close in some tasks (*e.g.*, "D-3 to M"). Indeed, SCDA is a particular case of HSAV where $k_t = k_s = 1$. Compared to SCDA, HSAV can automatically decide whether one should use subdomains and the number of subdomains.

Deep subdomain adaptation methods with a fixed number of subdomains (MultiDA, DCTN) have an effect of negative transfer (negative improvements) over this dataset, especially when using the log-loss improvement metric. In the papers that propose DCTN [158] and MultiDA [95, 96], these methods are evaluated on image datasets and are shown to be efficient. We have done our best to train DCTN and MultiDA on the Kaggle tabular datasets. However, the variances of their performances are very high. There could be a set of hyperparameters that favors DCTN and MultiDA, but tedious to find. These experimental results empirically demonstrate the difficulty of generalizing deep multi-subdomain adaptation methods to other non-image tasks.

Although DAN significantly improves the baseline of NN models of the adaptation task "D-1 to M", the improvement is less significant compared to LGB models. The LGB baseline model without adaptation method exceeds all NN models. Indeed, the LGB model is efficient in dealing with tabular data with categorical variables. Moreover, LGB-HSAV further improves the LGB baseline and is also better than the *single-domain* adaptation method LGB-SCDA.

Table 5.2 reports the results in a weakly supervised case. Relying on labeled target data, all adaptation methods based on NN models improve their performances compared to the unsupervised case. However, LGB-HSAV and LGB-SCDA are no better than their unsupervised version, mainly in the adaptation task "D-1 to M". We explain this result by the instability related to the scarcity of weakly labeled data.

Compared to WCDA, HSAV is shown to be more stable. A specific example is the adaptation task "D-1 to M" with NN models in terms of PR-AUC improvements. Indeed, HSAV aggregates multiple adaptation results and can decrease the impact of not well-performed adaptation.

Table 5.3: Adaptation performances of **unsupervised** adaptation methods over **Worldline** datasets (see Section 2.8.1 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%). Dotted lines separate single-domain adaptation and multi-subdomain adaptation methods.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
NN-HSAV (**ours**)	**8.77±1.61**	**12.93±1.92**	7.38±0.85	**9.70**
MultiDA	-7.07±15.18	-6.68±5.89	8.26±4.88	-1.83
DCTN	2.14±4.38	-0.15±7.44	1.98±7.65	1.32
NN-SCDA (**ours**)	8.02±1.35	11.72±2.49	5.95±0.84	8.56
DAN	7.31±5.78	5.47±3.96	**10.01±4.70**	7.60
DANN	4.38±5.09	6.43±2.69	5.28±3.03	5.37
MCD	6.46±5.70	1.41±8.19	6.84±4.94	4.91

(a) Improvements of PR-AUC of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
NN-HSAV (**ours**)	**7.67±0.52**	**13.81±1.03**	14.50±1.14	**12.00**
MultiDA	-787.93±262.34	-776.48±153.09	-555.38±93.78	-706.60
DCTN	-2.90±3.34	-2.63±4.32	4.14±8.87	-0.46
NN-SCDA (**ours**)	7.26±0.48	13.03±1.32	13.02±0.27	11.11
DAN	4.40±2.75	9.12±2.60	**17.90±3.91**	10.48
DANN	2.85±2.37	9.78±1.33	16.45±2.98	9.69
MCD	4.40±1.57	7.55±3.94	14.16±7.35	8.70

(b) Improvements of log-loss of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB-HSAV (**ours**)	5.60±4.23	**6.28±2.64**	**9.97±1.95**	**7.28**
LGB-SCDA (**ours**)	1.83±6.19	2.77±4.37	7.11±3.32	3.90
LGB Baseline	**9.91±6.43**	3.59±6.57	-1.94±8.31	3.85

(c) Improvements of PR-AUC of LGB predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB-HSAV (**ours**)	**5.06±1.30**	**10.60±0.88**	**21.19±0.77**	**12.28**
LGB-SCDA (**ours**)	3.14±2.33	8.58±1.60	19.36±1.52	10.36
LGB Baseline	5.02±3.54	4.38±4.00	9.43±4.65	6.28

(d) Improvements of log-loss of LGB predictive models.

Table 5.4: Adaptation performances of **weakly supervised** adaptation methods over **Worldline** datasets (see Section 2.8.1 for the dataset description). We report percentages of performances improvements, in PR-AUC and log-loss respectively, compared to NN baseline models (average improvements of NN baseline models are considered as 0%). Dotted lines separate single-domain adaptation and multi-subdomain adaptation methods.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
NN-HSAV (**ours**)	**12.57±1.99**	**14.30±1.42**	11.79±1.38	**12.88**
MultiDA	0.90±5.31	-3.85±6.40	**19.18±6.18**	5.41
DCTN	8.30±2.42	10.41±1.03	19.17±2.15	12.63
NN-WCDA (**ours**)	11.75±3.14	8.89±5.58	13.62±9.46	11.42
DAN	9.40±3.50	11.85±1.86	8.85±1.73	10.03
DANN	9.56±4.54	10.27±5.78	10.46±6.94	10.10
MCD	1.64±14.30	1.77±7.82	12.71±21.75	5.38
FineTune	8.04±1.54	10.91±1.50	5.32±0.63	8.09

(a) Improvements of PR-AUC of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
NN-HSAV (**ours**)	**9.03±0.59**	**14.09±0.97**	20.83±2.34	**14.65**
MultiDA	-78.15±38.97	-29.50±23.00	-16.96±27.13	-41.54
DCTN	1.87±2.10	4.43±0.81	21.86±5.85	9.39
NN-WCDA (**ours**)	5.23±1.59	9.86±2.14	**28.50±2.30**	14.53
DAN	8.05±1.04	12.66±0.88	18.83±2.07	13.18
DANN	4.41±3.58	10.99±3.05	27.49±4.75	14.30
MCD	-12.28±18.49	-8.05±22.62	8.31±22.13	-4.01
FineTune	6.38±0.40	11.63±0.23	13.99±1.69	10.67

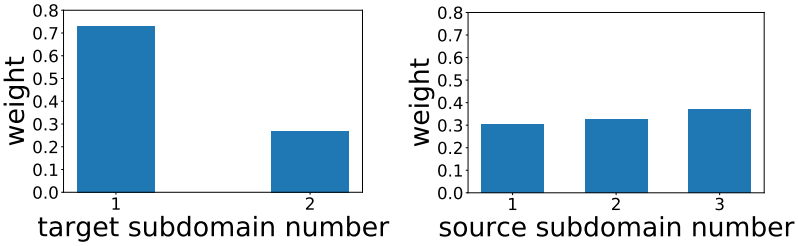
(b) Improvements of log-loss of NN predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB-HSAV (**ours**)	**23.43±5.46**	**19.52±2.75**	**17.39±3.08**	**20.11**
LGB-WCDA (**ours**)	22.65±5.14	17.84±4.38	15.21±5.65	18.56
LGB Baseline	9.91±6.43	3.59±6.57	-1.94±8.31	3.85

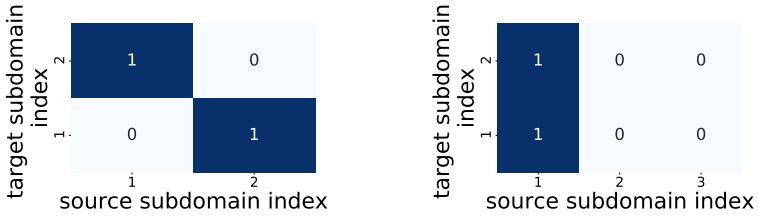
(c) Improvements of PR-AUC of LGB predictive models.

Method	G-1 to B	G-2 to B	G-3 to B	AVG
LGB-HSAV (**ours**)	**12.25±0.78**	**17.22±1.24**	**26.36±1.07**	**18.61**
LGB-WCDA (**ours**)	11.79±0.64	16.59±1.70	24.84±2.81	17.74
LGB Baseline	5.02±3.54	4.38±4.00	9.43±4.65	6.28

(d) Improvements of log-loss of LGB predictive models.



(a) Subdomains weights in the target (left), and the source (right).



(b) Sparse matrix$\boldsymbol{S}$, k_t=k_s=2 (left), and k_t=2 , k_s=3 (right).

Figure 5.4: Interpretability study on the Kaggle task D-2 to M.

5.5.2.2 Worldline Dataset

A similar conclusion can be drawn from Tables 5.3 and 5.4 where we evaluate adaptation methods over the Worldline fraud detection datasets. On average, our propositions (NN-HSAV and LGB-HSAV) outperform other adaptation methods. Although the unsupervised DAN has the best performance on the task "G-3 to B", it performs no better than NN-HSAV on average. In contrast to the Kaggle adaptation tasks, in the unsupervised setting, LGB-HSAV performs no better than LGB Baseline on the task "G-1 to B" in terms of PR-AUC. This may be related to the low performance of LGB-SCDA, as LGB-HSAV combines LGB-SCDA as elementary adaptation methods. Nonetheless, LGB-HSAV largely improvements LGB-SCDA on this task. Except for the task "G-3 to B" with NN models, HSAV always outperforms WCDA.

Compared to the unsupervised case (Table 5.3), labeled target data always increase the performances of all adaptation methods in this case, especially for the *multi-subdomain* adaptation method DCTN. The weakly supervised DCTN achieves the second-best performance among all NN models in terms of PR-AUC improvements.

To sum up, both Kaggle and Worldline adaptation tasks show the efficiency of discovering hidden subdomains relying on HSAV. Besides, HSAV is shown to improve SCDA and WCDA in unsupervised and weakly supervised settings, respectively.

5.5.3 Interpretability of Aggregation Functions

The following shows the interpretability of our method by illustrating the adaptation results of the Kaggle adaptation task D-2 to M. Values of parameter $\sigma(\boldsymbol{A})$ and $\boldsymbol{S}$ are displayed in Figure 5.4. More precisely, Figure 5.4a is obtained by summing $\sigma(\boldsymbol{A})$ by rows and columns, respectively. It shows the weights of each number of subdomains in source and target domains. Specifically, in the target domain, we give more weights to the case where k_t=1 (no subdomain). As for the number of subdomains in the source domain, although k_s=3 has more weights than the others, the difference is not significant. Therefore, one should also take into account the case when k_s=1 and k_s=2 to have a good prediction performance. Such an observation intuitively explains the reason why traditional *multi-subdomain* DA methods with a fixed number of subdomains cannot achieve good results. Figure 5.4b provides the mapping matrix $\boldsymbol{S}$ of the case when k_t=2 and $k_s \in \{2, 3\}$. When k_s=2, the first target subdomain maps to the second source subdomain, while the second target subdomain maps to the first source subdomain. When k_s=3, all subdomains of the target domain are mapped to the first source subdomain.

5.6 Conclusion

Standing in a *target to source* domain adaptation scenario, we provided a predictor aggregation method for a multi-subdomain adaptation scenario in weakly supervised and unsupervised settings. The proposed methods directly extended WCDA and SCDA to a multi-subdomain case. We added a sparse restriction over subdomain reweighing factors to enhance the interpretability of propositions. We first introduced a general subdomain division criterion and then specialized in a real-life temporal drift case. Empirical results showed that the multi-subdomain method HSAV outperforms single-domain methods WCDA and SCDA.

Chapter 6

Conclusion & Perspectives

6.1 Conclusion

This thesis makes contribution to a challenging domain adaptation problem with a focus on a fraud detection task in the Worldline industrial context. Different from existing domain adaptation scenarios, we addressed a setting where a well-performing pre-trained source domain predictive model is given and should be preserved. To this end, we proposed a new *target to source* domain adaptation framework and a concrete solution (CDA) leveraging one-dimensional optimal transport to tackle this adaptation scenario in Chapter 3. Central to optimal transport is the choice of cost function. For numerical values, we adopted Euclidean distance to have a closed-form solution of transportation plans. For categorical values, we relied on a generic cost in terms of the occurrence frequency [61] for the computation of transportation plans. Furthermore, we observed that feature selection helps to improve adaptation performance and enhance the interpretability of CDA.

In the case where few labels are available in target domains, we extended CDA by WCDA to use a feature selection process to automatically identify the features that drift the most between source and target domains. We then built an ordered and sparse mapping of the global input space based on the feature selection results. Moreover, such feature selection results provide business experts insights to explore the new market better. Our propositions were evaluated on three datasets and were shown to obtain state-of-the-art performances.

Chapter 4 extended the adaptation pipeline WCDA proposed in Chapter 3 to an unsupervised case. Namely, we provided an unsupervised feature selection process by guessing labels of stable examples in terms of the order of predictions under different adaptation functions. We argued that minimizing predicting risk on relaxed pseudo-labels equals minimizing an upper bound of target domain

risk. Both theoretical and empirical results proved the efficiency of the proposed method.

Chapter 5 considered a more general case where the source and target domains can contain examples of different distributions, and the number of underlying distributions is unknown. We proposed a method that automatically subdivides source and target domains and creates candidate predictions under different numbers of subdomains. The final predictions are a weighted sum of candidate predictions. We provided solutions in both weakly supervised and unsupervised cases, and an efficient gradient descent optimization algorithm was applied to solve the problem.

6.2 Perspectives

Based on our proposed *target to source* domain adaptation pipeline, we discuss some future research perspectives here. Our adaptation method leverages multiple one-dimensional optimal transport functions by supposing that correlations between features are automatically aligned between source and target domains. In the cases that violate such an assumption, one solution would be to regroup the input space into several subspaces. Features in the same subspace are adapted relying on a multi-dimensional transformation function while that in different subspaces are adapted independently. One potential research topic is to investigate the way to regroup such features. Note that the feature space encompasses categorical and numerical dimensions; thus the method should be *feature-type free*.

Another potential research topic consists of improving the feature selection process. Our current feature selection process supposes having a small subset of features that contribute more to domain adaptations than the others. A greedy algorithm is applied to find such features. However, the greedy algorithm becomes inefficient when input dimensions are very high, and all features contribute equally to domain adaptations. How to identify domain-significant features is a challenging task in this case requiring a deeper analysis.

From the Worldline industrial perspective, an interesting topic is to investigate other machine learning challenges besides the fraud detection, such as recommendation systems, smart routing, etc. Our proposition focuses on a highly class-imbalanced binary classification problem. How to apply such a method to other tasks with a potential domain drift remains to be studied.

Appendix A

Calibration

A.1 Evaluation Metric of Calibrated Models

In this thesis, we suppose that the given black-box model $h_s(\cdot)$ is well-calibrated (optimal Bayes predictor), and it returns the probability that an example belongs to one class. However, we give a brief proof to show that some well-known metrics like *area under the precision recall curve* (PR-AUC) are not suited to evaluate predictive model performances in this case.

Definition A.1 (Precision and recall functions)**.** Given a set of ordered source examples

$$\mathbb{X}^s_{\text{hypo}} = (\boldsymbol{x}^s_1, \ldots, \boldsymbol{x}^s_j, \ldots, \boldsymbol{x}^s_{n_s}), \boldsymbol{x}^s_j \in \mathcal{X},$$
$$\mathbb{Y}^s_{\text{hypo}} = (y_1, \ldots, y_i, \ldots, y_{n_s}), y_i \in \mathcal{Y},$$

where $\mathcal{X}$ and $\mathcal{Y}$ are respectively the input space and the output space. $\mathbb{X}^s_{\text{hypo}}$ is a set of inputs sorted in ascending order of $h_s(\boldsymbol{x}^s_j)$, and $\mathbb{Y}^s_{\text{hypo}}$ is a set of outputs in the same order:

$$\forall i, j \in \{1, \ldots, n_s\} \text{ and } i > j, h_s(\boldsymbol{x}^s_j) \geq h_s(\boldsymbol{x}^s_j). \tag{A.1}$$

The empirical precision function of the source predictor $h_s(\cdot)$ is defined as

$$\text{prec}(j) = \frac{\sum_{i=1}^{n_s} \mathbf{1}(h_s(\boldsymbol{x}^s_j) \geq h_s(\boldsymbol{x}^s_j)) y_j}{\sum_{i=1}^{n_s} \mathbf{1}(h_s(\boldsymbol{x}^s_j) \geq h_s(\boldsymbol{x}^s_j))}, j \in \{1, \ldots, n_s\}, \tag{A.2}$$

and the empirical recall function is defined as

$$\text{recall}(j) = \frac{\sum_{i=1}^{n_s} \mathbf{1}(h_s(\boldsymbol{x}^s_j) \geq h_s(\boldsymbol{x}^s_j)) y_j}{\sum_{i=1}^{n_s} y_j}, j \in \{1, \ldots, n_s\}. \tag{A.3}$$

Equation (3.6) is a non-decreasing function of $h_s(\cdot)$. After the adjustment of

label shift according to Equation (3.6), the order in Equation (A.1) is preserved by $h_p(\cdot)$ for the given set $\mathbb{X}^s_{\text{hypo}}$:

$$\forall i, j \in \{1, \ldots, n_s\} \text{ and } i > j \,, h_p(\boldsymbol{x}^s_j) \geq h_p(\boldsymbol{x}^s_j) \,.$$

Hence, the values of the precision function (Equation (A.2)) and the recall function(Equation (A.3)) remain unchanged. The metric PR-AUC is not able to measure if a model is well-calibrated or not, whereas the log-loss, having the following formula over source domain data:

$$-\frac{1}{n_s}\sum_{i=1}^{n_s}\Big[y_i \log(h_s(\boldsymbol{x}^s_j)) + (1-y_i)\log(1-h_s(\boldsymbol{x}^s_j))\Big]$$

is known to be able to measure errors between the prediction and the probability that a point belongs to a class. However, the metric PR-AUC is widely used in fraud detection tasks. Therefore, we report these two metrics whenever possible.

A.2 Calibration of Pre-trained Model

Proof. By the Bayes' theorem, we have

$$\begin{aligned}
&P(Y^p = y | X^p = \boldsymbol{x}) \\
&= \frac{P(X^p = \boldsymbol{x} | Y^p = y) P(Y^p = y)}{P(X^p = \boldsymbol{x})} \\
&= \frac{P(X^s = \boldsymbol{x} | Y^s = y) P(Y^t = y)}{P(X^p = \boldsymbol{x})} \\
&= \frac{P(Y^s = y | X^s = \boldsymbol{x}) P(Y^t = y) P(X^s = \boldsymbol{x})}{P(Y^s = y) P(X^p = \boldsymbol{x})} \\
&= P(Y^s = y | X^s = \boldsymbol{x}) q(\boldsymbol{x}) w(y) \,, \qquad \text{(A.4)}
\end{aligned}$$

where

$$q(\boldsymbol{x}) = \frac{P(X^s = \boldsymbol{x})}{P(X^p = \boldsymbol{x})} \,.$$

As we have

$$\begin{aligned}
&\sum_y P(Y^p = y | X^p = \boldsymbol{x}) = 1 \\
&\implies \sum_y P(Y^s = y | X^s = \boldsymbol{x}) q(\boldsymbol{x}) w(y) = 1 \,,
\end{aligned}$$

we solve the equation in $q(\boldsymbol{x})$ and we get

$$q(\boldsymbol{x}) = \left[\sum_y P(Y^s = y|X^s = \boldsymbol{x})w(y)\right]^{-1} .$$

We inject this solution into Equation (A.4) and we get

$$P(Y^p = y|X^p = \boldsymbol{x}) = \frac{P(Y^s = y|X^s = \boldsymbol{x})w(y)}{\sum_{y'} P(Y^s = y'|X^s = \boldsymbol{x})w(y')} .$$

In a binary classification problem, y takes values in $\{0, 1\}$. Replacing $P(Y^p = y|X^p = \boldsymbol{x})$ and $P(Y^s = y|X^s = \boldsymbol{x})$ respectively by $h_p(\cdot)$ and $h_s(\cdot)$ proves Equation (3.6). □

Bibliography

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[2] Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Em with bias-corrected calibration is hard-to-beat at label shift adaptation. 2020.

[3] Malak Alshawabkeh, Javed A Aslam, Jennifer G Dy, and David Kaeli. Feature weighting and selection using hypothesis margin of boosting. In *2012 IEEE 12th International Conference on Data Mining*, pages 41–50. IEEE, 2012.

[4] Jason Altschuler, Jonathan Niles-Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *NIPS*, 2017.

[5] Yuichiro Anzai. *Pattern recognition and machine learning*. Elsevier, 2012.

[6] Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. In *ICCV*, pages 769–776, 2013.

[7] Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *NIPS*, 19:137, 2007.

[8] John Blitzer, Ryan McDonald, and Fernando Pereira. Domain adaptation with structural correspondence learning. In *Conference on Empirical Methods in Natural Language Processing*, pages 120–128, 2006.

[9] Andreea Bobu, Eric Tzeng, Judy Hoffman, and Trevor Darrell. Adapting to continuously shifting domains, 2018.

[10] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and radon wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 51(1):22–45, 2015.

[11] Nicolas Bonneel, Gabriel Peyré, and Marco Cuturi. Wasserstein barycentric coordinates: histogram regression using optimal transport. *ACM Trans. Graph.*, 35(4):71–1, 2016.

[12] Karsten M Borgwardt, Arthur Gretton, Malte J Rasch, Hans-Peter Kriegel, Bernhard Schölkopf, and Alex J Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[13] Carl R Boyd, Mary Ann Tolson, and Wayne S Copes. Evaluating trauma care: the triss method. trauma score and the injury severity score. *The Journal of trauma*, 27(4):370–378, 1987.

[14] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Taylor & Francis, 1984. ISBN 9780412048418.

[15] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[16] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[17] Fabio Maria Carlucci, Lorenzo Porzi, Barbara Caputo, Elisa Ricci, and Samuel Rota Bulo. Autodial: Automatic domain alignment layers. In *2017 IEEE international conference on computer vision (ICCV)*, pages 5077–5085. IEEE, 2017.

[18] Wei-Lun Chao, Hexiang Hu, and Fei Sha. Cross-dataset adaptation for visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5716–5725, 2018.

[19] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[20] Chao Chen, Zhihang Fu, Zhihong Chen, Sheng Jin, Zhaowei Cheng, Xinyu Jin, and Xian-Sheng Hua. Homm: Higher-order moment matching for unsupervised domain adaptation. In *AAAI*, volume 34, pages 3422–3429, 2020.

[21] Minmin Chen, Kilian Q Weinberger, and John Blitzer. Co-training for domain adaptation. In *NIPS*, pages 2456–2464, 2011.

[22] Minmin Chen, Zhixiang Eddie Xu, Kilian Q. Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. In *ICML*, 2012.

[23] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, et al. Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1(4):1–4, 2015.

[24] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *ICML*, pages 1081–1090, 2019.

[25] Fan RK Chung and Fan Chung Graham. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[26] Guillem Collell, Drazen Prelec, and Kaustubh R Patil. A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data. *Neurocomputing*, 275:330–340, 2018.

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[28] Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE TPAMI*, 39(9):1853–1865, 2016.

[29] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *NIPS*, 2017.

[30] Nello Cristianini, John Shawe-Taylor, et al. *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[31] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *NIPS*, 2013.

[32] Hal Daumé III. Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[33] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[35] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[36] Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *ICML*, pages 289–296, 2009.

[37] Miroslav Dudík, Steven Phillips, and Robert E Schapire. Correcting sample selection bias in maximum entropy density estimation. *Advances in neural information processing systems*, 18:323–330, 2005.

[38] Zhen Fang, Jie Lu, Feng Liu, Junyu Xuan, and Guangquan Zhang. Open set domain adaptation: Theoretical bound and algorithm. *IEEE transactions on neural networks and learning systems*, 2020.

[39] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *ICCV*, pages 2960–2967, 2013.

[40] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *J. Mach. Learn. Res.*, 20(177):1–81, 2019.

[41] R'emi Flamary and Nicolas Courty. Pot python optimal transport library, 2017.

[42] Jerome H Friedman. *The elements of statistical learning: Data mining, inference, and prediction*. springer open, 2017.

[43] Jerome H Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76(376):817–823, 1981.

[44] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1):2096–2030, 2016.

[45] Léo Gautheron, Ievgen Redko, and Carole Lartizien. Feature selection for unsupervised domain adaptation using optimal transport. In *ECML PKDD*, pages 759–776. Springer, 2018.

[46] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *CVPR*, pages 2066–2073. IEEE, 2012.

[47] Boqing Gong, Kristen Grauman, and Fei Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 222–230. PMLR, 2013.

[48] Boqing Gong, Kristen Grauman, and Fei Sha. Reshaping visual datasets for domain adaptation. *NIPS*, 26:1286–1294, 2013.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[50] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[51] David Harris and Sarah Harris. *Digital design and computer architecture*. Morgan Kaufmann, 2010.

[52] James J Heckman. Sample selection bias as a specification error. *Econometrica: Journal of the Econometric Society*, pages 153–161, 1979.

[53] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, 6:3–10, 1994.

[54] Geoffrey E Hinton, Terrence Joseph Sejnowski, et al. *Unsupervised learning: foundations of neural computation*. MIT press, 1999.

[55] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[56] Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering latent domains for multisource domain adaptation. In *ECCV*, pages 702–715. Springer, 2012.

[57] Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. Algorithms and theory for multiple-source adaptation. In *NeurIPS*, 05 2018.

[58] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[59] Jiayuan Huang, Arthur Gretton, Karsten Borgwardt, Bernhard Schölkopf, and Alex Smola. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601–608, 2006.

[60] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[61] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972.

[62] Leonid Kantorovich. On the translocation of masses. *Management Science*, 5(1):1–4, 1958.

[63] 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. *NIPS*, 30:3146–3154, 2017.

[64] Michael Kearns and Leslie Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM (JACM)*, 41(1): 67–95, 1994.

[65] Henry J Kelley. Gradient theory of optimal flight paths. *Ars Journal*, 30 (10):947–954, 1960.

[66] Minyoung Kim, Pritish Sahu, Behnam Gholami, and Vladimir Pavlovic. Unsupervised visual domain adaptation: A deep max-margin gaussian process approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4380–4390, 2019.

[67] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[68] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[69] Kenji Kira, Larry A Rendell, et al. The feature selection problem: Traditional methods and a new algorithm. In *Aaai*, volume 2, pages 129–134, 1992.

[70] Wouter M Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

[71] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[72] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[73] Vinod K Kurmi, Venkatesh K Subramanian, and Vinay P Namboodiri. Domain impression: A source data free domain adaptation method. In *WACV*, pages 615–625, 2021.

[74] Dong-Hyun Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, 2013.

[75] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European conference on computer vision (ECCV)*, pages 35–51, 2018.

[76] Siyang Li, Bryan Seybold, Alexey Vorobyov, Alireza Fathi, Qin Huang, and C-C Jay Kuo. Instance embedding transfer to unsupervised video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6526–6535, 2018.

[77] Wen Li, Zheng Xu, Dong Xu, Dengxin Dai, and Luc Van Gool. Domain generalization and adaptation using low rank exemplar svms. *IEEE TPAMI*, 40(5):1114–1127, 2017.

[78] Yanghao Li, Naiyan Wang, Jianping Shi, Jiaying Liu, and Xiaodi Hou. Revisiting batch normalization for practical domain adaptation. *arXiv preprint arXiv:1603.04779*, 2016.

[79] Yitong Li, Michael Murias, Samantha Major, Geraldine Dawson, and David E Carlson. Extracting relationships by multi-domain matching. In *NeurIPS*, pages 6799–6810, 2018.

[80] Yitong Li, Michael Murias, Samantha Major, Geraldine Dawson, and David Carlson. On target shift in adversarial domain adaptation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 616–625. PMLR, 2019.

[81] Yunsheng Li, Lu Yuan, and Nuno Vasconcelos. Bidirectional learning for domain adaptation of semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6936–6945, 2019.

[82] Jian Liang, Dapeng Hu, and Jiashi Feng. Do we really need to access the source data? source hypothesis transfer for unsupervised domain adaptation. In *ICML*, pages 6028–6039. PMLR, 2020.

[83] Yi Lin, Yoonkyung Lee, and Grace Wahba. Support vector machines for classification in nonstandard situations. *Machine learning*, 46(1-3):191–202, 2002.

[84] Seppo Linnainmaa. The representation of the cumulative rounding error of an algorithm as a taylor expansion of the local rounding errors. *Master's Thesis (in Finnish), Univ. Helsinki*, pages 6–7, 1970.

[85] Zachary Lipton, Yu-Xiang Wang, and Alexander Smola. Detecting and correcting for label shift with black box predictors. In *International conference on machine learning*, pages 3122–3130. PMLR, 2018.

[86] Alexander H Liu, Yen-Cheng Liu, Yu-Ying Yeh, and Yu-Chiang Frank Wang. A unified feature disentangler for multi-domain image translation and manipulation. *arXiv preprint arXiv:1809.01361*, 2018.

[87] Hongfu Liu, Ming Shao, and Yun Fu. Structure-preserved multi-source domain adaptation. In *ICDM*, pages 1059–1064. IEEE, 2016.

[88] Huan Liu and Rudy Setiono. Feature selection and classification-a probabilistic wrapper approach. In *Proceedings of 9th International Conference on Industrial and Engineering Applications of AI and ES*, pages 419–424, 1997.

[89] Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2):539–550, 2008.

[90] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer feature learning with joint distribution adaptation. In *ICCV*, 2013.

[91] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *ICML*, 2015.

[92] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, pages 2208–2217. PMLR, 2017.

[93] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *NeurIPS*, pages 1640–1650, 2018.

[94] Zelun Luo, Yuliang Zou, Judy Hoffman, and Li Fei-Fei. Label efficient learning of transferable representations across domains and tasks. *arXiv preprint arXiv:1712.00123*, 2017.

[95] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Boosting domain adaptation by discovering latent domains. In *CVPR*, pages 3771–3780, 2018.

[96] Massimiliano Mancini, Lorenzo Porzi, Samuel Rota Bulo, Barbara Caputo, and Elisa Ricci. Inferring latent domains for unsupervised deep domain adaptation. *IEEE TPAMI*, 2019.

[97] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS*, volume 21. Curran Associates, Inc., 2009.

[98] Cheng Meng, Yuan Ke, Jingyi Zhang, Mengrui Zhang, Wenxuan Zhong, and Ping Ma. Large-scale optimal transport map estimation using projection pursuit. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[99] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[100] Tom Mitchell. Machine learning. 1997.

[101] Gaspard Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

[102] Oskar Morgenstern and John Von Neumann. *Theory of games and economic behavior*. Princeton university press, 1953.

[103] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[104] Arkadi Nemirovski and Uriel Rothblum. On complexity of matrix scaling. *Linear Algebra and its Applications*, 302:435–460, 1999.

[105] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[106] Bach Hoai Nguyen, Bing Xue, and Peter Andreae. A particle swarm optimization based feature selection approach to transfer learning in classification. In *Proceedings of the genetic and evolutionary computation conference*, pages 37–44, 2018.

[107] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[108] James B Orlin. A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming*, 78(2):109–129, 1997.

[109] Lale Özbakir, Adil Baykasoğlu, and Pınar Tapkan. Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, 215(11):3782–3795, 2010.

[110] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009.

[111] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010.

[112] Pau Panareda Busto and Juergen Gall. Open set domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 754–763, 2017.

[113] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019.

[114] Vern I Paulsen and Mrinal Raghupathi. *An introduction to the theory of reproducing kernel Hilbert spaces*, volume 152. Cambridge university press, 2016.

[115] Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *ICCV*, pages 1406–1415, 2019.

[116] Michaël Perrot, Nicolas Courty, Rémi Flamary, and Amaury Habrard. Mapping estimation for discrete optimal transport. In *NIPS*, pages 4204–4212, 2016.

[117] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607, 2019.

[118] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[119] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[120] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[121] Joaquin Quiñonero-Candela, Masashi Sugiyama, Neil D Lawrence, and Anton Schwaighofer. *Dataset shift in machine learning*. Mit Press, 2009.

[122] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[123] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[124] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33 (1):1–39, 2010.

[125] Lior Rokach and Oded Maimon. Clustering methods. In *Data mining and knowledge discovery handbook*, pages 321–352. Springer, 2005.

[126] Rasmus Rothe, Radu Timofte, and Luc Van Gool. Dex: Deep expectation of apparent age from a single image. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 10–15, 2015.

[127] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1):21–41, 2002.

[128] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for unsupervised domain adaptation. In *ICML*, 2017.

[129] Kuniaki Saito, Kohei Watanabe, Yoshitaka Ushiku, and Tatsuya Harada. Maximum classifier discrepancy for unsupervised domain adaptation. In *CVPR*, pages 3723–3732, 2018.

[130] Kuniaki Saito, Shohei Yamamoto, Yoshitaka Ushiku, and Tatsuya Harada. Open set domain adaptation by backpropagation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 153–168, 2018.

[131] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014.

[132] Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.

[133] Sandeepkumar Satpal and Sunita Sarawagi. Domain adaptation of conditional probability models via feature subsetting. In *ECML PKDD*, pages 224–235. Springer, 2007.

[134] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5 (2):197–227, 1990.

[135] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[136] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[137] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[138] HE Soper, AW Young, BM Cave, Alice Lee, and Karl Pearson. On the distribution of the correlation coefficient in small samples. appendix ii to the papers of" student" and ra fisher. *Biometrika*, 11(4):328–413, 1917.

[139] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

[140] Mervyn Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological)*, 36 (2):111–133, 1974.

[141] Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(5), 2007.

[142] 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 *NIPS*, 2008.

[143] Baochen Sun, Jiashi Feng, and Kate Saenko. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*, pages 153–171. Springer, 2017.

[144] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, 2014.

[145] Takeshi Teshima, Issei Sato, and Masashi Sugiyama. Few-shot domain adaptation by causal mechanism transfer. In *International Conference on Machine Learning*, pages 9458–9469. PMLR, 2020.

[146] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[147] Andrei Nikolaevich Tikhonov. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, volume 151, pages 501–504. Russian Academy of Sciences, 1963.

[148] Yi-Hsuan Tsai, Wei-Chih Hung, Samuel Schulter, Kihyuk Sohn, Ming-Hsuan Yang, and Manmohan Chandraker. Learning to adapt structured output space for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7472–7481, 2018.

[149] Eric Tzeng, Judy Hoffman, Trevor Darrell, and Kate Saenko. Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE international conference on computer vision*, pages 4068–4076, 2015.

[150] Selen Uguroglu and Jaime Carbonell. Feature selection for transfer learning. In *ECML PKDD*, pages 430–442. Springer, 2011.

[151] Naveen Venkat, Jogendra Kundu, Durgesh Singh, Ambareesh Revanur, and R. Babu. Your classifier can secretly suffice multi-source domain adaptation. In *NeurIPS*, 03 2021.

[152] Cédric Villani. The wasserstein distances. In *Optimal Transport*, pages 93–111. Springer, 2009.

[153] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[154] Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. In Hal Daumé III and Aarti Singh, editors, *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9898–9907. PMLR, 13–18 Jul 2020.

[155] Sholom M Weiss and Nitin Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995.

[156] Alan Geoffrey Wilson. The use of entropy maximising models, in the theory of trip distribution, mode split and route split. *Journal of transport economics and policy*, pages 108–126, 1969.

[157] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *ICML*, pages 5423–5432, 2018.

[158] Ruijia Xu, Ziliang Chen, Wangmeng Zuo, Junjie Yan, and Liang Lin. Deep cocktail network: Multi-source unsupervised domain adaptation with category shift. In *CVPR*, pages 3964–3973, 2018.

[159] Zheng Xu, Wen Li, Li Niu, and Dong Xu. Exploiting low-rank structure from latent domains for domain generalization. In *ECCV*, pages 628–643. Springer, 2014.

[160] Guorong Xuan, Wei Zhang, and Peiqi Chai. Em algorithms of gaussian mixture model and hidden markov model. In *Proceedings 2001 International Conference on Image Processing (Cat. No. 01CH37205)*, volume 1, pages 145–148. IEEE, 2001.

[161] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6210–6219, 2019.

[162] Hao-Wei Yeh, Baoyao Yang, Pong C Yuen, and Tatsuya Harada. Sofa: Source-data-free feature alignment for unsupervised domain adaptation. In *WACV*, pages 474–483, 2021.

[163] Wei Ying, Yu Zhang, Junzhou Huang, and Qiang Yang. Transfer learning via learning to transfer. In *International conference on machine learning*, pages 5085–5094. PMLR, 2018.

[164] Kaichao You, Ximei Wang, Mingsheng Long, and Michael Jordan. Towards accurate model selection in deep unsupervised domain adaptation. In *ICML*, pages 7124–7133. PMLR, 2019.

[165] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[166] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *ICML*, pages 819–827, 2013.

[167] Luxin Zhang, Pascal Germain, Yacine Kessaci, and Christophe Biernacki. Target to source coordinate-wise adaptation of pre-trained models. In *ECML PKDD*, pages 378–394. Springer International Publishing, 2021.

[168] Luxin Zhang, Pascal Germain, Yacine Kessaci, and Christophe Biernacki. Interpretable Domain Adaptation Using Unsupervised Feature Selection on Pre-trained Source Models. August 2021.

[169] Luxin Zhang, Pascal Germain, Yacine Kessaci, and Christophe Biernacki. Interpretable Domain Adaptation for Hidden Subdomain Alignment in the Context of Pre-trained Source Models. 2022.

[170] Qiming Zhang, Jing Zhang, Wei Liu, and Dacheng Tao. Category anchor-guided unsupervised domain adaptation for semantic segmentation. *arXiv preprint arXiv:1910.13049*, 2019.

[171] Yixin Zhang and Zilei Wang. Joint adversarial learning for domain adaptation in semantic segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 6877–6884, 2020.

[172] Han Zhao, Shanghang Zhang, Guanhang Wu, José MF Moura, Joao P Costeira, and Geoffrey J Gordon. Adversarial multiple source domain adaptation. *NeurIPS*, 31:8559–8570, 2018.

[173] Sicheng Zhao, Guangzhi Wang, Shanghang Zhang, Yang Gu, Yaxian Li, Zhichao Song, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-source distilling domain adaptation. In *AAAI*, number 07, pages 12975–12983, 2020.

[174] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[175] Yongchun Zhu, Fuzhen Zhuang, and Deqing Wang. Aligning domain-specific distribution and classifier for cross-domain classification from multiple sources. In *AAAI*, volume 33, pages 5989–5996, 2019.