Towards a Unified Framework for Learning and Reasoning

Han Zhao

August 2020
CMU-ML-20-111

Machine Learning Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee

Geoffrey J. Gordon, Chair
Ruslan Salakhutdinov
Barnabás Póczos
Tommi S. Jaakkola (Massachusetts Institute of Technology)

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was supported by the Office of Naval Research under award numbers N000140911052 and N000141512365, the Air Force Research Laboratory award number FA87501720152, a grant from NCS Pearson, Inc., and a Nvidia GPU grant. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Information Theory, Learning Theory, Probabilistic Graphical Models, Representation Learning, Algorithmic Fairness, Domain Adaptation, Transfer Learning, Multitask Learning, Sum-Product Networks, Probabilistic Circuits

To my parents, my brother, and my lovely girlfriend.

Abstract

The success of supervised machine learning in recent years crucially hinges on the availability of large-scale and unbiased data, which is often time-consuming and expensive to collect. Recent advances in deep learning focus on learning rich and invariant representations that have found abundant applications in domain adaptation, multitask learning, algorithmic fairness, and machine translations, just to name a few. However, it is not clear what price we have to pay in terms of task utility for such universal representations. On the other hand, learning is only one of the two most fundamental cognitive abilities of intelligent agents. An intelligent agent needs to have both the ability to learn from the experience, and the ability to reason from what has been learned. However, classic symbolic reasoning cannot model the inherent uncertainty that ubiquitously exists, and it is not robust to noisy observations. Perhaps more fundamentally, reasoning is computationally intractable in general. As a result, learning, which often takes reasoning as a sub-procedure, is also hard. Building on the fundamental concepts from information theory and theoretical computer science, this thesis aims to understand the inherent tradeoff between utility and invariance in learning the representations, and to develop efficient algorithms for learning tractable and exact probabilistic inference machines.

This thesis contains two parts. The first part is devoted to *understanding and learning invariant representations*. In particular, we will focus on understanding the costs of existing invariant representations by characterizing a fundamental tradeoff between invariance and utility. First, we will use domain adaptation as an example to both theoretically and empirically show such tradeoff in achieving small joint generalization error. This result also implies an inherent tradeoff between demographic parity, a statistical notion of group fairness, and utility in both classification and regression settings. Going beyond, we will further show that such general tradeoff exists in learning with structured data. In particular, we shall derive an impossibility theorem for universal machine translation by learning language-invariant representations. Second, we will focus on designing learning algorithms to escape the existing tradeoff and to utilize the benefits of invariant representations. We will show how the algorithm can be used to guarantee equalized treatment of individuals between groups, and discuss what additional problem structure it requires to permit efficient domain adaptation and machine translation through learning invariant representations.

The second part of the thesis is devoted to *learning tractable and exact circuits for probabilistic reasoning*. It is well-known that exact marginal and conditional inference in classic probabilistic graphical models (PGMs), including Bayesian Networks (BNs) and Markov Networks (MNs), is #P-complete. As a result, practitioners usually need to resort to various approximate inference schemes to ensure computational tractability. Probabilistic circuits, which include Sum-Product Networks (SPNs) as a special case, have been proposed as tractable deep models for exact probabilistic inference. They distinguish themselves from other types of probabilistic graphical models by the fact that inference can be done exactly in linear time with respect to the size of the circuit. This has generated a lot of interest since inference is not only a powerful tool to reason under uncertainty, but also a core task for parameter estimation and structure learning. In this part, we will concentrate on both theoretical and practical parts of learning tractable probabilistic circuits. In particular, we will investigate the representational power of SPNs, as well as its parameter learning procedures in both online and offline settings.

Acknowledgments

First and foremost, I am greatly indebted to my advisor, Geoff Gordon, who, throughout my five-year journey at Carnegie Mellon University, has provided me with constant supports, insightful discussions, as well as the freedom that allows me to pursue interesting research problems to fulfill my own curiosity. In addition to being a fantastic researcher, Geoff is also the kindest and most supportive individual that I have had the pleasure of working with. He has a great sense of research taste, which also shapes mine to work on important and elegant questions. I would like to sincerely thank Geoff for his intuitive insights in understanding complex problems, for sharing with me his broad knowledge in almost every aspect of machine learning, and for his unwavering faith in me that pushes me to become a better researcher. I hope that I could become a researcher, an advisor and a person like him in the future.

I am very grateful to Ruslan Salakhutdinov, Barnabás Póczos and Tommi S. Jaakkola for all the help they have provided during my Ph.D., including serving as my thesis committee members. Russ has been a brilliant collaborator on a number of research projects during my Ph.D. He is extremely knowledgeable and understands almost all aspects of deep learning. I would like to thank him for his tremendous help in research and his countless support in my career. Barnabás has been a great friend and mentor. He is always willing to listen to me on both research and life, and give me his advice and thoughts. Tommi hosted me for a visit at MIT, the results of which include a research paper on the information-theoretic characterization of the fundamental limit of invariant representation learning, and another one on using adversarial learning to defend attribute inference attacks on graphs. During my visit, his sharpness and deep insights helped me achieve a better and clearer understanding about the tradeoff problem I was working on.

I would also like to thank my master thesis advisor, Pascal Poupart at the University of Waterloo, who led me to explore this fantastic field of research on tractable probabilistic reasoning, for his patience and encouragement, and his endless support and guidance. Pascal has always been a kind mentor and friend, and it is my privilege to be able to continue collaborating with him after graduating from Waterloo. The results of our collaboration are presented in Chapter 9.

A huge thanks goes to all of my other collaborators throughout my Ph.D.: Tameem Adel, Brandon Amos, Jianfeng Chi, Adam Coates, Remi Tachet des Combes, João P. Costeira, Amanda Coston, Chen Dan, Junjie Hu, Priyank Jaini, Stefanie Jegelka, Nebojsa Jojic, Chen Liang, Chin-Hui Lee, Peizhao Li, Peiyuan Alex Liao, Hongfu Liu, José M. F. Moura, Renato Negrinho, Rema Padman, Abdullah Rashwan, Pradeep Ravikumar, Andrej Risteski, Nihar B. Shah, Jian Shen, Xiaofei Shi, Alexander J. Smola, Otilia Stretcu, Ivan Tashev, Yuan Tian, Yao-Hung Tsai, Richard E. Turner, Wen Wang, Yu-Xiang Wang, Guanhang Wu, Keyulu Xu, Yichong Xu, Makoto Yamada, Jianbo Ye, Shuayb Zarar, Kun Zhang, Shanghang Zhang, Zhenyao Zhu, and Honglei Zhuang. It would not have been as fun or as productive without their help. Special thanks goes to Andrej Risteski, Nihar B. Shah, Yao-Hung Tsai and Yichong Xu. It has been my great privilege to be able to work with them, from whom I not only benefit inspiring discussions, but also countless fun moments in life.

I also want to thank everyone in the Machine Learning Department at Carnegie Mellon University, who contribute to making it a vibrant and fun place to pursue graduate studies. In particular, I would like to thank Diane Stidle and other amazing staff in our department for their endless efforts in making our everyday life in the department easy and enjoyable. Special thanks to all my friends at CMU: Siddharth Ancha, Ifigeneia Apostolopoulou, Shaojie Bai, Devendra Chaplot, Maria De-Arteaga, Carlton Downey, Simon Du, Avinava Dubey, William Guss, Ahmed Hefny, Hanzhang Hu, Zhiting Hu, Lisa Lee, Leqi Liu, Yangyi Lu, Yifei Ma, Ritesh Noothigattu, Anthony Platanios, Mrinmaya Sachan, Wen Sun, Mariya Toneva, Po-Wei Wang, Yining Wang, Eric Wong, Yifan Wu, Yuexin Wu, Pengtao Xie, Qizhe Xie, Keyang Xu, Diyi Yang, Fan Yang, Zhilin Yang, Ian En-Hsu Yen, Yaodong Yu, Hongyang Zhang and Xun Zheng. Hope our friendship will last forever.

Last but most importantly, I would like to thank my parents, Wei Zhao and Yuxia Wang, my younger brother, Rui Zhao and my lovely girlfriend Lu Sun, for all of your unconditional love and support during my Ph.D. journey. Needless to say, this thesis would not have been possible without your encouragement along the way. This thesis is dedicated to all of you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The recent decade has witnessed a phenomenal success in artificial intelligence. In particular, deep learning has gained an unprecedented impact across both research and industry communities by demonstrating better than human performance on various kinds of real-world competitions, e.g., the ImageNet recognition task (Krizhevsky et al., 2012), the Stanford question answering competition (Rajpurkar et al., 2016), the board game Go (Silver et al., 2017), etc. As a result, machine learning tools have been widely adopted to help decision making in various real-world scenarios, e.g., face recognition, machine translation, college admission, etc. While these empirical achievements are exciting, whether the learned model could deliver its promise in real-world scenarios crucially depends on the data used to train the model. However, often, it is computationally expensive, or sometimes infeasible, to collect labeled data under all the possible real-world scenarios. As a result, due to this *distributional shift* between the training and test data, the learned model may fail dramatically in practice. Perhaps more importantly, in high-stakes settings such as loan approvals, criminal justice and hiring process, if the data used to train the model contain historical bias, then the learned model, without *bias mitigation*, can only exacerbate the existing discrimination. Hence, the ability to learn representations that are invariant to the changes in the environment is crucial and can provide us the robustness against various noise and nuisance factors in real-world.

On the other hand, learning is only one of the two most fundamental cognitive abilities of intelligent agents. An intelligent agent needs to have *the ability to learn from the experience*, as well as *the ability to reason from what has been learned*. As foreseen by the Turing Award Laureate Prof. Leslie Valiant (Valiant, 2018), one of the key challenges for AI in the coming decades is the development of *integrated learning and reasoning mechanisms*. However, classic symbolic reasoning cannot model the inherent uncertainty that ubiquitously exists, and it is not robust to noisy observations. Perhaps more fundamentally, inference and reasoning are computationally intractable in general. As a result, learning, which often takes inference/reasoning as a sub-procedure, is also hard.

This thesis is centered around advancing the frontier of AI in both *representation learning* and *probabilistic reasoning*. As shown in Figure 1.1, we believe an intelligent agent should consist of a knowledge base and an inference engine, and the agent interacts with the environment in loop. In each iteration, the agent receives input from the environment and uses its knowledge base to map the real-world observations or queries to its internal algebraic representations. The inference engine then carries out probabilistic reasoning to answer the query or choose an action to execute in the environment. The long-term research goal of this thesis is thus to build a unified framework that provides a common semantics for learning and reasoning, by developing invariant representations that generalize across different environments as well as efficient inference engine that allows exact and tractable probabilistic reasoning.

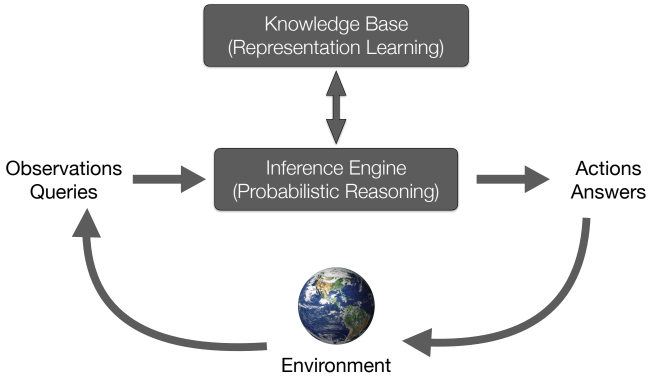


Figure 1.1: A model of an intelligent agent, which consists of a knowledge base and an inference engine. Representation learning serves as a technique to help build our knowledge base that maps from real-world objects to their algebraic representations. The inference engine is powered by probabilistic reasoning that allows interactions between the agent and the environment.

1.1 Main Contributions of This Thesis

Within the broad area of artificial intelligence and machine learning, my Ph.D. research primarily spans two themes: *invariant representation learning* and *tractable probabilistic reasoning*. *Invariant representation learning* serves as a bridge connecting abstract objects in real-world and their corresponding algebraic representations that are amenable for computation and allow generalization across different environments. *Tractable probabilistic reasoning* aims to provide an inference mechanism for *exact and efficient* reasoning under uncertainty. However, it is not well-understood what is the fundamental limit of invariant representations in terms of task utility, and it is well-known that even approximate probabilistic reasoning is computationally intractable (Roth, 1996) in the worst case.

Building on the fundamental concepts from information theory and theoretical computer science, my work aims to understand the inherent tradeoff between utility and invariance in learning the representations, and to develop efficient algorithms for learning tractable and exact probabilistic inference machines. The *key contributions* of my thesis research are as follows and summarized in Figure 1.2.

1. Analyzed and proved the fundamental limit of learning invariant representations in terms of task utility. With this result, we also identify and explain the inherent tradeoffs in learning domain-invariant representations for *unsupervised domain adaptation* (Chapter 4 and Chapter 5), learning fair representations for *algorithmic fairness* (Chapter 6), learning representations for *privacy-preservation under attribute-inference attacks* (Zhao et al., 2019b), and learning *multilingual sentence representations for universal machine translation* (Chapter 8).

2. Developed an algorithm on learning domain-invariant representations for unsupervised domain adaptation under *multiple different source environments* (Chapter 3) using adversarial neural networks. To mitigate bias in automated decision making systems, my coauthors and I also proposed an algorithm to learn fair representations that can simultaneously guarantee accuracy parity and equalized odds (Hardt et al., 2016) among different demographic groups (Chapter 7). Analogously, we also developed an algorithm by learning invariant representations to filter out sensitive information and provided guarantees on inference error from malicious adversaries (Zhao et al., 2019b).

3. Established *the first equivalence* between Sum-Product networks (Poon and Domingos, 2011), Bayesian networks with algebraic decision diagrams, and mixture models (Zhao et al., 2015b, 2016b)

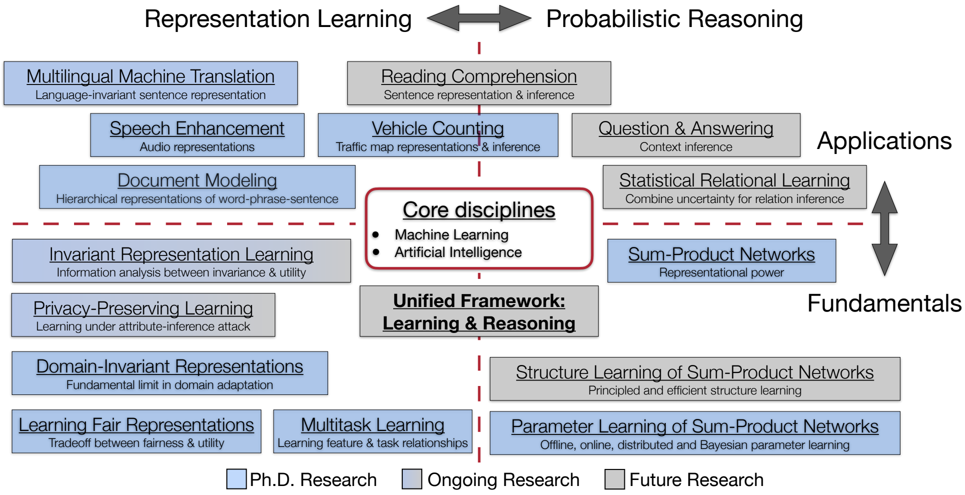

Representation Learning
Probabilistic Reasoning
Multilingual Machine Translation
Language-invariant sentence representation
Reading Comprehension
Sentence representation & inference
Speech Enhancement
Audio representations
Vehicle Counting
Traffic map representations & inference
Question & Answering
Context inference
Applications
Document Modeling
Hierarchical representations of word-phrase-sentence
Statistical Relational Learning
Combine uncertainty for relation inference
Core disciplines
Machine Learning
Artificial Intelligence
Invariant Representation Learning
Information analysis between invariance & utility
Sum-Product Networks
Representational power
Privacy-Preserving Learning
Learning under attribute-inference attack
Unified Framework:
Learning & Reasoning
Fundamentals
Domain-Invariant Representations
Fundamental limit in domain adaptation
Structure Learning of Sum-Product Networks
Principled and efficient structure learning
Learning Fair Representations
Tradeoff between fairness & utility
Multitask Learning
Learning feature & task relationships
Parameter Learning of Sum-Product Networks
Offline, online, distributed and Bayesian parameter learning
□ Ph.D. Research
□ Ongoing Research
□ Future Research

Figure 1.2: Overview of my Ph.D. research (blue), ongoing research (mixed blue and gray) and future plans (gray) around the theme of representation learning and probabilistic reasoning. Artificial Intelligence and machine learning are the core disciplines of my research theme. Topics on the left and right are related to representation learning and probabilistic reasoning, respectively. Topics about the underlying theory and fundamental appear at the bottom. Applications are at the top. My long term research goal is to develop a unified framework that can combine representation learning and probabilistic reasoning together.

(Chapter 9). Inspired by our theoretical results, we proposed efficient learning algorithms for Sum-Product networks in both offline (Zhao et al., 2016b) (Chapter 9) and online (Rashwan et al., 2016b; Zhao and Gordon, 2017) (Chapter 11), discrete (Zhao et al., 2016b) and continuous (Jaini et al., 2016) settings, and from both frequentists' and Bayesian principles (Chapter 10).

Broadly, all the results above enjoy sound theoretical guarantees and provide insights towards better understanding of learning invariant representations and building tractable inference machines. On the practical side, I have also devoted much effort to develop software tools for the proposed algorithms and publicly share them with the community. For instance, MDAN (Zhao et al., 2018b) (for domain adaptation with multiple sources) has been successfully used as a benchmark algorithm in various follow-up work on unsupervised domain adaptation for vision (Zhao et al., 2019i) and language (Chen and Cardie, 2018).

1.2 Overview of Part I: Invariant Representation Learning

Many machine learning applications involve learning representations that achieve two competing goals: To maximize information or accuracy with respect to a subset of features (e.g. for prediction) while simultaneously maximizing invariance or independence with respect to another, potentially overlapping, subset of features (e.g. for fairness, domain identity, etc.). Typical examples include privacy-preserving learning, multilingual machine translation, domain adaptation, and algorithmic fairness, just to name a few. In fact, all of the above problems admit a common minimax game-theoretic formulation, whose equilibrium represents a fundamental tradeoff between accuracy and invariance.

In this part of the thesis, we provide an information theoretic analysis of this general and important problem under different applications settings, including domain adaptation, algorithmic fairness, and multilingual machine translation. In each of the above applications, we analyze the inherent tradeoff between accuracy and invariance by providing lower bounds on the error rates that any classifier has to incur when invariance is satisfied. The general principle here could be understood as a kind of *uncertainty principle* when learning with invariance constraint: if perfect invariance is attained, then any classifier, even with unbounded computational resources and labeled data, has to incur a large error on at least one subset of the overall population. Informally, if we instantiate this general principle to domain adaptation, multilingual machine translation and algorithmic fairness respectively, we immediately obtain the following implications:

1. In domain adaptation with one source and one target domain, any classifier based on domain-invariant representations has to incur a large error on at least one of the two domains.
2. In multilingual machine translation with a set of different languages, any decoder based on language-invariant sentence representations has to incur a large translation error on at least one of the translation pairs.
3. In a population with two groups, e.g., black and white, if the base rates differ between these two subgroups, then any fair classifier (in the sense of statistical parity) has to achieve a large error on at least one of the groups.

To complement our negative results, in this part of the thesis we also discuss what kind of problem structures could allow us to circumvent the above tradeoffs. In particular, we shall discuss how to design algorithms that can be used to guarantee equalized treatment of individuals between groups, and discuss what additional problem structure it requires to permit efficient domain adaptation and machine translation through learning invariant representations.

1.2.1 Learning Domain-Invariant Representations

The success of machine learning has been partially attributed to rich labeled datasets and powerful computations. Unfortunately, collecting and annotating such large-scale training data is prohibitively expensive and time-consuming. To solve these limitations, different labeled datasets can be combined to build a larger one. However, due to the potential distributional shift between different datasets, models trained on the combined one still suffer from large generalization error on a target domain different from the training domain.

Our work on domain adaptation focuses on understanding the limit of knowledge transfer from a labeled source environment to an unlabeled target environment by learning domain-invariant representations to bridge the gap (Adel et al., 2017; Zhao et al., 2018b, 2019f). The main idea of domain-invariant learning is simple and intuitive: we would like to learn representations that are invariant to the difference among different environments while still contain rich information for the desired task. Theoretically, my research sheds new light on this problem by proving an information-theoretic lower bound on the joint error of *any* domain adaptation algorithm that attempts to learn invariant representations: *there is a fundamental tradeoff between learning domain-invariant representations and achieving small joint error in both environments when the difference between environments can be used to explain the target task* (Zhao et al., 2019e). Specifically, our result implies that any algorithm based on domain-invariant representations has to incur a large error on at least one of the environments. This result characterizes the fundamental limit in terms of the joint utility when learning with domain-invariant representations.

1.2.2 Learning Fair Representations

With the prevalence of machine learning applications in high-stakes domains, e.g., criminal judgement, medical testing, online advertising, etc., it is crucial to ensure that the automated decision making systems do not propagate existing bias or discrimination that might exist in historical data used to train these systems. Among many recent proposals for achieving different notions of algorithmic fairness, *learning fair representations* has received increasing attention due to recent advances in learning rich representations with deep neural networks. At a high level the underlying idea is that if representations from different groups are similar to each other, then any predictive model on top of them will certainly make decisions independent of group membership.

On the other hand, while it has long been empirically observed (Calders et al., 2009) that there is an underlying tradeoff between utility and fairness, theoretical understanding is lacking. In my recent work (Zhao and Gordon, 2019), I provided the first theoretical result that characterizes the inherent tradeoff between utility and fairness. More precisely, our result shows that any fair algorithm, in the sense of demographic parity, admits an information-theoretic lower bound on the joint error across different demographic groups. To escape such inherent tradeoff, we also propose an alternative algorithm (Zhao et al., 2019c) to learn conditionally group-invariant representations. The proposed algorithm constructs a classifier that is no worse than the optimal classifier in terms of demographic parity gap, and can achieve equalized false positive/negative rates and accuracy parity across different demographic groups simultaneously.

1.2.3 Learning Multilingual Representations

The goal of universal machine translation is to learn to translate between any pair of languages, given a corpus of paired translated documents for *a small subset* of all pairs of languages. Despite impressive empirical results and an increasing interest in massively multilingual models, theoretical analysis on translation errors made by such universal machine translation models is only nascent. In this thesis we formally prove certain impossibilities of this endeavour in general, as well as prove positive results in the presence of additional (but natural) structure of data. For the former, we derive a lower bound on the translation error in the many-to-many translation setting, which shows that any algorithm aiming to learn shared sentence representations among multiple language pairs has to make a large translation error on at least one of the translation tasks, if no assumption on the structure of the languages is made. For the latter, we show that if the paired documents in the corpus follow a natural *encoder-decoder* generative process, we can expect a natural notion of "generalization": a linear number of language pairs, rather than quadratic, suffices to learn a good representation. Our theory also explains what kinds of connection graphs between pairs of languages are better suited: ones with longer paths result in worse sample complexity in terms of the total number of documents per language pair needed.

1.3 Overview of Part II: Tractable Probabilistic Reasoning

Exact marginal and conditional inference in classic probabilistic graphical models (PGMs), including Bayesian Networks (BNs) and Markov Networks (MNs), is #P-complete (Roth, 1996). As a result, practitioners usually need to resort to various approximate inference schemes to ensure computational tractability. Sum-Product Networks (SPNs) have been proposed as tractable deep models (Poon and Domingos, 2011) for exact probabilistic inference. They belong to a more general family of unified models for learning and reasoning, known as *probabilistic circuits* (Van den Broeck, 2018). SPNs, like other models within the family of probabilistic circuits, distinguish themselves from other types of probabilistic graphical models by the fact that inference can be done exactly in linear time with respect to the size of the circuit. This has generated a lot of interest since inference is not only a powerful tool to reason under uncertainty, but also a core task for parameter estimation and structure learning.

In this part of the thesis, we focus on both theoretical and practical aspects of learning tractable probabilistic circuits, including investigating the representational power of SPNs, and the parameter learning of SPNs from both the frequentist's and Bayesian's perspective. In what follows we first briefly summarize the problems we consider and highlight the results in the rest of this chapter.

1.3.1 A Unified Framework for Parameter Learning

In Chapter 9 we present a unified framework, which treats two of the previous algorithms (Gens and Domingos, 2012; Poon and Domingos, 2011) as special cases, for learning the parameters of SPNs. We prove that any complete and decomposable SPN is equivalent to a mixture of trees where each tree corresponds to a product of univariate distributions. Based on the mixture model perspective, we can precisely characterize the functional form of the objective function based on the network structure. We show that the optimization problem associated with learning the parameters of SPNs based on the MLE principle can be formulated as a signomial program (SP), where both PGD and exponentiated gradient (EG) can be viewed as first order approximations of the signomial program after suitable transformations of the objective function. We also show that the signomial program formulation can be equivalently transformed into a difference of convex functions (DCP) formulation, where the objective function of the program can be naturally expressed as a difference of two convex functions. The DCP formulation allows us to develop two efficient optimization algorithms for learning the parameters of SPNs based on sequential monomial approximations (SMA) and the concave-convex procedure (CCCP), respectively. Both proposed approaches naturally admit multiplicative updates, hence effectively deal with the positivity constraints of the optimization. Furthermore, under our unified framework, we also show that CCCP leads to the same algorithm as EM despite that these two approaches are different from each other in general. Although we mainly focus on MLE based parameter learning, the mixture model interpretation of SPN also helps to develop a Bayesian learning method for SPNs (Zhao et al., 2016a). PGD, EG, SMA and CCCP can all be viewed as different levels of convex relaxation of the original SP. Hence the framework also provides an intuitive way to compare all four approaches.

1.3.2 Collapsed Variational Inference

In Chapter 10 we propose a collapsed variational inference algorithm for SPNs that is robust to overfitting and can be naturally extended into a stochastic variant to scale to large data sets. We call our algorithm CVB-SPN. CVB-SPN is a deterministic approximate Bayesian inference algorithm that is computationally efficient and easy to implement while at the same time allowing us to incorporate prior information into the design. Like other variational techniques, CVB-SPN is based on optimization. Unlike other variational techniques, the number of parameters to be optimized is only linear in the network size ($O(|\mathcal{S}|)$) and is *independent* of the size of the training set, as opposed to $O(D|\mathcal{S}|)$ in ordinary VI, where D is the size of the training set. It is worth noting that, different from traditional collapsed variational approaches in the literature that marginalize out global latent variables (Teh et al., 2006, 2007), here we consider a complementary approach: instead of marginalizing out the global latent variables in order to spread out the interactions among many local latent variables, CVB-SPN takes advantage of the fast exact inference in SPNs and *marginalizes out the local latent variables* in order to maintain a marginal variational posterior distribution directly on the global latent variables, i.e., the model parameters. The posterior mean of the variational distribution over model parameters can then be used as a Bayesian estimator. To the best of our knowledge, this is the first general Bayesian approach to learn the parameters of SPNs efficiently in both batch and stochastic settings.

At first glance, marginalizing out all the local latent variables in graphical models seems to be a bad idea for two reasons. First, by marginalizing out local latent variables we naively appear to incur computation exponential in the tree-width of the graph (Jordan et al., 1999; Wainwright and Jordan, 2008). Except for graphical models with special structures, for example, LDA, Gaussian mixture models, thin junction trees, etc., such exact computation is intractable by itself. Second, marginalization will in general invalidate the conjugacy between the prior distribution and the joint distribution over global and local latent variables, which further makes the expectation over the variational posterior intractable. Fortunately, the ability of SPNs to model context-specific independence helps to solve the first problem, and by using a change of variables CVB-SPN handles the second problem efficiently. Besides the reduced space complexity, we also show that the objective of CVB-SPN forms a strictly better lower bound to be optimized than the evidence lower bound (ELBO) in standard VI. To show the validity of CVB-SPN, we conduct extensive experiments and compare it with maximum likelihood based methods in both batch and stochastic settings.

To tackle the second problem described above, there is a strand of recent work on extending standard VI to general nonconjugate settings using stochastic sampling from the variational posterior (Blei et al., 2012; Kingma and Welling, 2013; Mnih and Gregor, 2014; Ranganath et al., 2014; Titsias and Lázaro-Gredilla, 2014; Titsias, 2015). However, control variates need to be designed in order to reduce the high variance incurred by insufficient samples. Furthermore, for each such sample from the variational posterior, those algorithms need to go through the whole training data set to compute the stochastic gradient, leading to a total computational cost $O(ND|\mathcal{S}|)$, where N is the sample size. This is often prohibitively expensive for SPNs.

1.3.3 Linear Time Computation of Moments

Most existing batch learning algorithms for SPNs can be straightforwardly adapted to the online setting, where the network updates its parameters after it receives one instance at each time step. This online learning setting makes SPNs more widely applicable in various real-world scenarios. This includes the case where either the data set is too large to store at once, or the network needs to adapt to the change of external data distributions. Recently Rashwan et al. (2016b) proposed an online Bayesian Moment Matching (BMM) algorithm to learn the probability distribution of the model parameters of SPNs based on the method of moments. Later Jaini et al. (2016) extended this algorithm to the continuous case where the leaf nodes in the network are assumed to be Gaussian distributions. At a high level BMM can be understood as an instance of the general assumed density filtering framework (Sorenson and Stubberud, 1968) where the algorithm finds an approximate posterior distribution within a tractable family of distributions by the method of moments. Specifically, BMM for SPNs works by matching the first and second order moments of the approximate tractable posterior distribution to the exact but intractable posterior. An essential sub-routine of the above two algorithms (Jaini et al., 2016; Rashwan et al., 2016b) is to efficiently compute the exact first and second order moments of the one-step update posterior distribution (cf. 11.2). Rashwan et al. (2016b) designed a recursive algorithm to achieve this goal in linear time when the underlying network structure is a tree, and this algorithm is also used by Jaini et al. (2016) in the continuous case. However, the algorithm only works when the underlying network structure is a tree, and a naive computation of such moments in a DAG will scale quadratically w.r.t. the network size. Often this quadratic computation is prohibitively expensive even for SPNs with moderate sizes.

In Chapter 11 we propose a linear time (and space) algorithm that is able to compute any moments of all the network parameters simultaneously even when the underlying network structure is a DAG. (Zhao and Gordon, 2017) There are three key ingredients in the design and analysis of our algorithm: 1). A linear time reduction from the moment computation problem to the joint inference problem in SPNs, 2). A succinct evaluation procedure of polynomial by differentiation without expanding it, and 3). A dynamic programming method to further reduce the quadratic computation to linear. The differential approach (Darwiche, 2003) used for polynomial evaluation can also be applied for exact inference in Bayesian networks. This technique has also been implicitly used in the recent development of a concave-convex procedure (CCCP) for optimizing the weights of SPNs (Zhao et al., 2016b). Essentially, by reducing the moment computation problem to a joint inference problem in SPNs, we are able to exploit the fact that the network polynomial of an SPN computes a multilinear function in the model parameters, so we can efficiently evaluate this polynomial by differentiation even if the polynomial may contain exponentially many monomials, provided that the polynomial admits a tractable circuit complexity. Dynamic programming can be further used to trade off a constant factor in space complexity (using two additional copies of the network) to reduce the quadratic time complexity to linear so that all the edge moments can be computed simultaneously in two passes of the network. To demonstrate the usefulness of our linear time sub-routine for computing moments, we apply it to design an efficient assumed density filter (Sorenson and Stubberud, 1968) to learn the parameters of SPNs in an online fashion. ADF runs in linear time and space due to our efficient sub-routine. As an additional contribution, we also show that ADF and BMM can both be understood under a general framework of moment matching, where the only difference lies in the moments chosen to be matched and how to match the chosen moments.

1.4 Bibliographic Notes

The research presented in this thesis is based on joint work with several co-authors, described below. This thesis only includes works for which this author was the, or one of the, primary contributors.

Chapter 3 is based on a joint work with Shanghang Zhang, Guanhang Wu, João P. Costeira, José M. F. Moura and Geoff Gordon. Chapter 4 is based on a joint work with Remi Tachet des Combes, Kun Zhang and Geoff Gordon. Chapter 5 is based on a joint work with Remi Tachet des Combes, Yu-Xiang Wang and Geoff Gordon. Chapter 6 is based on a joint work with Geoff Gordon. Chapter 7 is based on joint work with Amanda Coston, Tameem Adel and Geoff Gordon. Chapter 8 is based on a joint work with Junjie Hu and Andrej Risteski. Chapter 9 is based a joint work with Pascal Poupart and Geoff Gordon. Chapter 10 is based on a joint work with Tameem Adel, Geoff Gordon and Brandon Amos. Finally, Chapter 11 is based on a joint work with Geoff Gordon.

1.5 Excluded Research

In order to keep this thesis succinct and coherent, I excluded a significant port of my research works during my Ph.D. career. In what follows we list the excluded research works.

- Online and Distributed Learning of Sum-Product Networks in both discrete and continuous settings (Jaini et al., 2016; Rashwan et al., 2016b).
- Unsupervised Domain Adaptation (Adel et al., 2017; Zhao et al., 2019f).
- Frank-Wolfe for Symmetric-NMF under Simplicial Constraint (Zhao and Gordon, 2018).
- Convolutional-Recurrent Neural Networks for Speech Enhancement (Zhao et al., 2018a).
- Learning Neural Networks with Adaptive Regularization (Zhao et al., 2019e).
- Multitask Feature and Relationship Learning (Zhao et al., 2017).
- Strategyproof Conference Peer Review (Xu et al., 2018).
- Active learning on graphs (Liang et al., 2018).
- Fair clustering for visual learning (Li et al., 2020).
- Dynamic and interpretable postoperative complication risk scoring (Wang et al., 2020).

- Continual Learning with Adaptive Weights (Adel et al., 2019b).
- Adversarial privacy preservation under attribute inference attack (Zhao et al., 2019a).

Chapter 2

Preliminaries

In this chapter we provide some background knowledge about the learning problems presented in this thesis. In particular, after introducing the notation used throughout the thesis, we shall give a brief introduction to the problem of unsupervised domain adaptation, learning with fairness constraint, Sum-Product Networks (SPNs), and signomial programming. More definitions of these concepts will be introduced in detail in the corresponding chapters when necessary. The purpose of this chapter is to give a brief introduction of the models and preliminaries that we shall use throughout the entire thesis. However, each chapter is self-contained and has more detailed definitions with variants that are tailored to specific problems considered in the chapter. This chapter ends with a brief introduction to the technical tools from statistical learning theory and information theory that will be used in this thesis.

2.1 Notation

We now describe some notation that we employ throughout this thesis.

For an integer $n \in \mathbb{N}$, we use $[n]$ to abbreviate the notation $\{1, 2, \ldots, n\}$. We use a capital letter X to denote a random variable and a bold capital letter $\mathbf{X}$ to denote a random vector. Similarly, a lowercase letter x is used to denote a value taken by X and a bold lowercase letter $\mathbf{x}$ denotes a joint value taken by the corresponding vector $\mathbf{X}$ of random variables. Indicator variable $\mathbb{I}_A$ equals 1 iff the event A is true, otherwise it takes the default value 0. For boolean variable X_i, we use $\mathbb{I}_{x_i}$ to denote $\mathbb{I}_{X_i=1}$ and $\mathbb{I}_{\bar{x}_i}$ to denote $\mathbb{I}_{X_i=0}$, respectively. To simplify the notation, we use $\Pr(x)$ to mean $\Pr(X = x)$ and $\Pr(\mathbf{x})$ to mean $\Pr(\mathbf{X} = \mathbf{x})$. Graphs are often denotes as $\mathcal{G}$. Throughout the thesis, we shall use standard notation $\mathbb{R}_+$ to denote the set of nonnegative reals and $\mathbb{R}_{++}$ to mean the set of strictly positive reals.

For typical learning problems, we use $\mathcal{X}$ and $\mathcal{Y}$ to denote the input and output space, respectively. Similarly, $\mathcal{Z}$ stands for the representation space induced from $\mathcal{X}$ by a feature transformation $g : \mathcal{X} \mapsto \mathcal{Z}$. Accordingly, we use X, Y, Z to denote the random variables which take values in $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$, respectively. Throughout the thesis, we use the words domain and distribution interchangeably. In particular, *domain* corresponds to a distribution $\mathcal{D}$ on the input space $\mathcal{X}$ and a labeling function $f : \mathcal{X} \to [0, 1]$. In the domain adaptation setting, we use $\langle \mathcal{D}_S, f_S \rangle$ and $\langle \mathcal{D}_T, f_T \rangle$ to denote the source and target domains, respectively. A *hypothesis* is a function $h : \mathcal{X} \to \{0, 1\}$. The *error* of a hypothesis h w.r.t. the labeling function f under distribution $\mathcal{D}_S$ is defined as: $\varepsilon_S(h, f) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[|h(\mathbf{x}) - f(\mathbf{x})|]$. When f and h are binary classification functions, this definition reduces to the probability that h disagrees with f under $\mathcal{D}_S$: $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[|h(\mathbf{x}) - f(\mathbf{x})|] = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[\mathbb{I}(f(\mathbf{x}) \neq h(\mathbf{x}))] = \Pr_{\mathbf{x}\sim\mathcal{D}_S}(f(\mathbf{x}) \neq h(\mathbf{x}))$. For two functions g and h with compatible domains and ranges, we use $h \circ g$ to denote the function composition $h(g(\cdot))$.

SPNs and BNs are denoted respectively by $\mathcal{S}$ and $\mathcal{B}$, respectively. For a directed acyclic graph (DAG) $\mathcal{G}$ and a node v in $\mathcal{G}$, we use $\mathcal{G}_v$ to denote the subgraph of $\mathcal{G}$ induced by v and all its descendants. Let $\mathbf{V}$ be a subset of the nodes of $\mathcal{G}$, then $\mathcal{G}|_{\mathbf{V}}$ is a subgraph of $\mathcal{G}$ induced by the node set $\mathbf{V}$. Similarly, we use $\mathbf{X}|_A$ or $\mathbf{x}|_A$ to denote the restriction of a vector to a subset A. We will use node and vertex, arc and edge interchangeably when we refer to a graph. Furthermore, $|\mathcal{G}|$ is used to denote the size of the graph $\mathcal{G}$, which is the sum of the number of nodes and the number of edges in $\mathcal{G}$. We also use $\mathrm{tw}(\mathcal{G})$ to represent the tree-width (Robertson and Seymour, 1984) of $\mathcal{G}$.

2.2 Domain Adaptation

We first introduce the problem setup of domain adaptation and review a theoretical model for domain adaptation when there is one source and one target (Ben-David et al., 2007, 2010; Blitzer et al., 2008; Kifer et al., 2004). The key idea is the $\mathcal{H}$-divergence to measure the discrepancy between two distributions. Other theoretical models for DA exist (Cortes and Mohri, 2014; Cortes et al., 2008; Mansour et al., 2009a,c); we choose to work with the above model because this distance measure has a particularly natural interpretation and can be well approximated using samples from both domains.

We define the *risk* of hypothesis h as the error of h w.r.t. a true labeling function under domain $\mathcal{D}_S$, i.e., $\varepsilon_S(h) := \varepsilon_S(h, f_S)$. As common notation in computational learning theory, we use $\widehat{\varepsilon}_S(h)$ to denote the empirical risk of h on the source domain. Similarly, we use $\varepsilon_T(h)$ and $\widehat{\varepsilon}_T(h)$ to mean the true risk and the empirical risk on the target domain. $\mathcal{H}$-divergence is defined as follows:

Definition 2.2.1. Let $\mathcal{H}$ be a hypothesis class for instance space $\mathcal{X}$, and $\mathcal{A}_\mathcal{H}$ be the collection of subsets of $\mathcal{X}$ that are the support of some hypothesis in $\mathcal{H}$, i.e., $\mathcal{A}_\mathcal{H} := \{h^{-1}(\{1\}) \mid h \in \mathcal{H}\}$. The distance between two distributions $\mathcal{D}$ and $\mathcal{D}'$ based on $\mathcal{H}$ is: $d_\mathcal{H}(\mathcal{D}, \mathcal{D}') := 2\sup_{A\in\mathcal{A}_\mathcal{H}} |\Pr_\mathcal{D}(A) - \Pr_{\mathcal{D}'}(A)|$.

When the hypothesis class $\mathcal{H}$ contains all the possible measurable functions over $\mathcal{X}$, $d_\mathcal{H}(\mathcal{D}, \mathcal{D}')$

reduces to the familiar total variation. Given a hypothesis class $\mathcal{H}$, we define its symmetric difference w.r.t. itself as: $\mathcal{H}\Delta\mathcal{H} = \{h(\mathbf{x}) \oplus h'(\mathbf{x}) \mid h, h' \in \mathcal{H}\}$, where $\oplus$ is the XOR operation. Let h^* be the optimal hypothesis that achieves the minimum combined risk on both the source and the target domains: $h^* := \arg\min_{h\in\mathcal{H}} \varepsilon_S(h) + \varepsilon_T(h)$, and use λ to denote the combined risk of the optimal hypothesis h^*: $\lambda := \varepsilon_S(h^*) + \varepsilon_T(h^*)$. Ben-David et al. (2007) and Blitzer et al. (2008) proved the following generalization bound on the target risk in terms of the source risk and the discrepancy between the single source domain and the target domain:

Theorem 2.2.1 (Blitzer et al. (2008))**.** Let $\mathcal{H}$ be a hypothesis space of VC-dimension d and $\widehat{\mathcal{D}}_S$ ($\widehat{\mathcal{D}}_T$) be the empirical distribution induced by sample of size m drawn from $\mathcal{D}_S$ ($\mathcal{D}_T$). Then w.p.b. at least $1-\delta$, $\forall h \in \mathcal{H}$,

$$\varepsilon_T(h) \leq \widehat{\varepsilon}_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T) + \lambda + O\left(\sqrt{\frac{d\log(m/d) + \log(1/\delta)}{m}}\right). \tag{2.1}$$

The bound depends on λ, the optimal combined risk that can be achieved by hypothesis in $\mathcal{H}$. The intuition is if λ is large, we cannot hope for a successful domain adaptation. One notable feature is that the empirical discrepancy distance between two samples can be approximated by a discriminator to distinguish instances from two domains.

2.3 Algorithmic Fairness

We first introduce various definitions of fairness in the literature under the category of group fairness, where the group membership is given by the sensitive attribute A, e.g., race, gender, etc. Even in this context there are many possible definitions of *fairness* (Narayanan, 2018), and in what follows we provide a brief review of the ones that are mostly relevant to this thesis. To keep the notation consistent, for $a \in \{0, 1\}$, we use $\mathcal{D}_a$ to mean the conditional distribution of $\mathcal{D}$ given $A = a$. For an event E, $\mathcal{D}(E)$ denotes the probability of E under $\mathcal{D}$. In particular, in the literature of fair machine learning, we call $\mathcal{D}(Y = 1)$ the *base rate* of distribution $\mathcal{D}$ and we use $\Delta_{\mathrm{BR}}(\mathcal{D}, \mathcal{D}') := |\mathcal{D}(Y = 1) - \mathcal{D}'(Y = 1)|$ to denote the difference of the base rates between two distributions $\mathcal{D}$ and $\mathcal{D}'$ over the same sample space. Given a feature transformation function $g : \mathcal{X} \to \mathcal{Z}$ that maps instances from the input space $\mathcal{X}$ to feature space $\mathcal{Z}$, we define $g_\sharp\mathcal{D} := \mathcal{D} \circ g^{-1}$ to be the induced (pushforward) distribution of $\mathcal{D}$ under g, i.e., for any event $E' \subseteq \mathcal{Z}$, $g_\sharp\mathcal{D}(E') := \mathcal{D}(g^{-1}(E')) = \mathcal{D}(\{x \in \mathcal{X} \mid g(x) \in E'\})$.

Definition 2.3.1 (Demographic Parity, a.k.a. Statistical parity)**.** Given a joint distribution $\mathcal{D}$, a classifier $\widehat{Y}$ satisfies *demographic parity* if $\widehat{Y}$ is independent of A.

Demographic parity reduces to the requirement that $\mathcal{D}_0(\widehat{Y} = 1) = \mathcal{D}_1(\widehat{Y} = 1)$, i.e., positive outcome is given to the two groups at the same rate. When exact equality does not hold, we use the absolute difference between them as an approximate measure:

Definition 2.3.2 (DP Gap)**.** Given a joint distribution $\mathcal{D}$, the *demographic parity gap* of a classifier $\widehat{Y}$ is $\Delta_{\mathrm{DP}}(\widehat{Y}) := |\mathcal{D}_0(\widehat{Y} = 1) - \mathcal{D}_1(\widehat{Y} = 1)|$.

Demographic parity is also known as *statistical parity*, and it has been adopted as definition of fairness in a series of work (Calders et al., 2009; Edwards and Storkey, 2015; Johndrow et al., 2019; Kamiran and Calders, 2009; Kamishima et al., 2011; Louizos et al., 2015; Madras et al., 2018; Zemel et al., 2013). However, as we shall quantify precisely in Section 6.3, demographic parity may cripple the utility that we hope to achieve, especially in the common scenario where the *base rates* differ between two groups, e.g., $\mathcal{D}_0(Y = 1) \neq \mathcal{D}_1(Y = 1)$ (Hardt et al., 2016). In light of this, an alternative definition is *accuracy parity*:

Definition 2.3.3 (Accuracy Parity)**.** Given a joint distribution $\mathcal{D}$, a classifier h satisfies *accuracy parity* if $\varepsilon_{\mathcal{D}_0}(h) = \varepsilon_{\mathcal{D}_1}(h)$.

In the literature, a break of accuracy parity is also known as disparate mistreatment (Zafar et al., 2017). Again, when h is a deterministic binary classifier, accuracy parity reduces to $\mathcal{D}_0(h(X) = Y) = \mathcal{D}_1(h(X) = Y)$. Different from demographic parity, the definition of accuracy parity does not eliminate the perfect predictor when $Y = A$ when the base rates differ between two groups. When costs of different error types matter, more refined definitions exist:

Definition 2.3.4 (Positive Rate Parity)**.** Given a joint distribution $\mathcal{D}$, a deterministic classifier h satisfies *positive rate parity* if $\mathcal{D}_0(h(X) = 1 \mid Y = y) = \mathcal{D}_1(h(X) = 1 \mid Y = y), \forall y \in \{0, 1\}$.

Positive rate parity is also known as *equalized odds* (Hardt et al., 2016), which essentially requires equal true positive and false positive rates between different groups. Furthermore, Hardt et al. (2016) also defined *true positive parity*, or *equal opportunity*, to be $\mathcal{D}_0(h(X) = 1 \mid Y = 1) = \mathcal{D}_1(h(X) = 1 \mid Y = 1)$ when positive outcome is desirable. Last but not least, *predictive rate parity*, also known as *test fairness* (Chouldechova, 2017), asks for equal chance of positive outcomes across groups given predictions:

Definition 2.3.5 (Predictive Rate Parity)**.** Given a joint distribution $\mathcal{D}$, a probabilistic classifier h satisfies *predictive rate parity* if $\mathcal{D}_0(Y = 1 \mid h(X) = c) = \mathcal{D}_1(Y = 1 \mid h(X) = c), \forall c \in [0, 1]$.

When h is a deterministic binary classifier that only takes value in $\{0, 1\}$, Chouldechova (2017) showed an intrinsic tradeoff between predictive rate parity and positive rate parity:

Theorem 2.3.1 (Chouldechova (2017))**.** Assume $\mathcal{D}_0(Y = 1) \neq \mathcal{D}_1(Y = 1)$, then for any deterministic classifier $h : \mathcal{X} \to \{0, 1\}$ that is not perfect, i.e., $h(X) \neq Y$, positive rate parity and predictive rate parity cannot hold simultaneously.

Similar tradeoff result for probabilistic classifier has also been observed by Kleinberg et al. (2016), where the authors showed that for any non-perfect predictors, calibration and positive rate parity cannot be achieved simultaneously if the base rates are different across groups. Here a classifier h is said to be *calibrated* if $\mathcal{D}(Y = 1 \mid h(X) = c) = c, \forall c \in [0, 1]$, i.e., if we look at the set of data that receive a predicted probability of c by h, we would like c-fraction of them to be positive instances according to Y (Pleiss et al., 2017).

2.4 Sum-Product Networks

A sum-product network (SPN) is a graphical representation of a joint probability distribution over a set of random variables $\mathbf{X} = \{X_1, \ldots, X_n\}$. It is a rooted directed acyclic graph where the interior nodes are sums or products and the leaves are univariate distributions over X_i. Edges emanating from sum nodes are parameterized with positive weights. Each node in an SPN encodes an unnormalized marginal distribution over $\mathbf{X}$. An example of SPN is shown in Fig. 2.1. Let $\mathbf{x}$ be an instantiation of the random vector $\mathbf{X}$. We associate an unnormalized probability $V_k(\mathbf{x} \mid \mathbf{w})$ with each node k when the input to the network is $\mathbf{x}$ with network weights set to be $\mathbf{w}$:

$$V_k(\mathbf{x} \mid \mathbf{w}) = \begin{cases} \mathbb{I}_{x_i}(\mathbf{x}_i) & k \text{ is a leaf indicator variable on } X_i = 1 \\ \mathbb{I}_{\bar{x}_i}(\mathbf{x}_i) & k \text{ is a leaf indicator variable on } X_i = 0 \\ \prod_{j \in \mathrm{Ch}(k)} V_j(\mathbf{x} \mid \mathbf{w}) & k \text{ is a product node} \\ \sum_{j \in \mathrm{Ch}(k)} w_{kj} V_j(\mathbf{x} \mid \mathbf{w}) & k \text{ is a sum node} \end{cases} \tag{2.2}$$

where $\mathrm{Ch}(k)$ is the child list of node k in the graph and w_{kj} is the edge weight associated with sum node k and its child node j. In the example of Fig. 2.1, the root of the graph computes the following polynomial

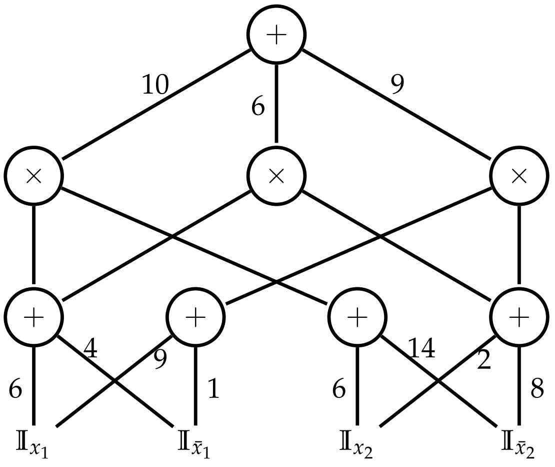


Figure 2.1: An example of a sum-product network over two boolean variables X_1 and X_2.

function over $(\mathbb{I}_{x_1}, \mathbb{I}_{\bar{x}_1}, \mathbb{I}_{x_2}, \mathbb{I}_{\bar{x}_2})$:

$$\begin{aligned} V_{\text{root}}(\mathbf{x} \mid \mathbf{w}) =& 10(6\mathbb{I}_{x_1} + 4\mathbb{I}_{\bar{x}_1}) \times (6\mathbb{I}_{x_2} + 14\mathbb{I}_{\bar{x}_2}) \\ &+ 6(6\mathbb{I}_{x_1} + 4\mathbb{I}_{\bar{x}_1}) \times (2\mathbb{I}_{x_2} + 8\mathbb{I}_{\bar{x}_2}) \\ &+ 9(9\mathbb{I}_{x_1} + 1\mathbb{I}_{\bar{x}_1}) \times (2\mathbb{I}_{x_2} + 8\mathbb{I}_{\bar{x}_2}) \\ =& 594\mathbb{I}_{x_1}\mathbb{I}_{x_2} + 1776\mathbb{I}_{x_1}\mathbb{I}_{\bar{x}_2} + 306\mathbb{I}_{\bar{x}_1}\mathbb{I}_{x_2} + 824\mathbb{I}_{\bar{x}_1}\mathbb{I}_{\bar{x}_2} \end{aligned} \tag{2.3}$$

Given an SPN $\mathcal{S}$, we first define the notion of *network polynomial*, which plays an important role throughout this thesis:

Definition 2.4.1 (Network Polynomial)**.** Let $f(\cdot) \geq 0$ be an unnormalized probability distribution over a Boolean random vector $\mathbf{X}_{[n]}$. The network polynomial of $f(\cdot)$ is a multilinear function $\sum_{\mathbf{x}} f(\mathbf{x}) \prod_{i=1}^{n} \mathbb{I}_{\mathbf{x}_i}$ of indicator variables, where the summation is over all possible instantiations of the Boolean random vector $\mathbf{X}_{[n]}$.

Note that the network polynomial as defined above is a function over $2n$ indicator variables. Intuitively, the network polynomial is a Boolean expansion (Boole, 1847) of the unnormalized probability distribution $f(\cdot)$. For example, the network polynomial of a BN $X_1 \to X_2$ is $\Pr(x_1, x_2)\mathbb{I}_{x_1}\mathbb{I}_{x_2} + \Pr(x_1, \bar{x}_2)\mathbb{I}_{x_1}\mathbb{I}_{\bar{x}_2} + \Pr(\bar{x}_1, x_2)\mathbb{I}_{\bar{x}_1}\mathbb{I}_{x_2} + \Pr(\bar{x}_1, \bar{x}_2)\mathbb{I}_{\bar{x}_1}\mathbb{I}_{\bar{x}_2}$. Similarly, the network polynomial of an SPN is given by $V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$.

To simplify the notation, for any partial assignment $\mathbf{y}$ of a subset of random variables $\mathbf{Y} \subseteq \mathbf{X}$, we slightly abuse the notation $V_{\text{root}}(\mathbf{y} \mid \mathbf{w})$ to actually mean the value of the network where we set all the indicators w.r.t. $X \in \mathbf{X} \backslash \mathbf{Y}$ to be 1 and for $X \in \mathbf{Y}$ we set the value of indicator variable according to the assignment $\mathbf{y}$. For instance, in the SPN given in Fig. 2.1, we use $V_{\text{root}}(X_2 = 1 \mid \mathbf{w})$ to mean the value of the network with inputs $(\mathbb{I}_{x_1}, \mathbb{I}_{\bar{x}_1}, \mathbb{I}_{x_2}, \mathbb{I}_{\bar{x}_2}) = (1, 1, 1, 0)$.

We now proceed to give a formal definition of sum-product network over binary random variables $\mathbf{X}_{[n]}$. In later we will also extend the following definition over binary random variables to arbitrary discrete distributions and continuous distributions.

Definition 2.4.2 (Sum-Product Network (Poon and Domingos, 2011))**.** A Sum-Product Network (SPN) over Boolean variables $\mathbf{X}_{[n]}$ is a rooted DAG whose leaves are the indicators $\mathbb{I}_{x_1}, \ldots, \mathbb{I}_{x_n}$ and $\mathbb{I}_{\bar{x}_1}, \ldots, \mathbb{I}_{\bar{x}_n}$ and whose internal nodes are sums and products. Each edge (v_k, v_j) emanating from a sum node k has a positive weight w_{kj}. The value of a product node k is $V_k := \prod_{j \in \text{Ch}(k)} V_j$ and the value of a sum node k is

$V_k := \sum_{v_j \in \mathrm{Ch}(v_k)} w_{kj} V_j$. The value of an SPN is defined to be the value of its root: $V_{\mathrm{root}}(\cdot \mid \mathbf{w})$.

By definition, it is clear that each internal node of SPN also defines a polynomial function over a subset of $\mathbf{X}_{[n]}$. We use $\mathbb{S}$ to denote the set of sum nodes in an SPN and $\mathbb{P}$ to denote the set of product nodes in an SPN. We now introduce a concept, *scope*, to formally describe the domain of the function computed by an internal node. The definition is recursive:

Definition 2.4.3 (Scope)**.** For an SPN $\mathcal{S}$ over $\mathbf{X}_{[n]}$, the *scope* of a node k is defined as follows:

1. $\mathrm{scope}(k) = \{i\}$ if k is a leaf indicator variable over X_i.
2. $\mathrm{scope}(k) := \bigcup_{j \in \mathrm{Ch}(k)} \mathrm{scope}(j)$.

Two more structural properties about SPN can be further defined based on the notion of scope:

Definition 2.4.4 (Complete)**.** An SPN is *complete* iff $\forall k \in \mathbb{S}, \forall j \in \mathrm{Ch}(k)$, $\mathrm{scope}(k) = \mathrm{scope}(j)$.

Definition 2.4.5 (Decomposable)**.** An SPN is decomposable iff $\forall k \in \mathbb{P}, \forall j_1, j_2 \in \mathrm{Ch}(k), j_1 \neq j_2$, $\mathrm{scope}(j_1) \bigcap \mathrm{scope}(j_2) = \varnothing$.

In other words, *completeness* requires that for each sum node in an SPN, the scope of this sum node is the same as the scope of any of its child. On the other hand, *decomposability* ensures that for any product node, the scopes of any pair of its children are disjoint. It has been shown that every complete and decomposable SPN defines a proper probability distribution (Poon and Domingos, 2011), and this is a sufficient but not necessary condition. In this thesis, we focus only on complete and decomposable SPNs as we are interested in their associated probabilistic semantics. For a complete and decomposable SPN $\mathcal{S}$, each node v in $\mathcal{S}$ defines a network polynomial $f_v(\cdot)$ which corresponds to the sub-SPN rooted at v. The network polynomial defined by the root of the SPN can then be computed recursively by taking a weighted sum of the network polynomials defined by the sub-SPNs rooted at the children of each sum node and a product of the network polynomials defined by the sub-SPNs rooted at the children of each product node.

The joint distribution encoded by an SPN is defined by the graphical structure and the weights. For example, to make the polynomial in Eq. (2.3) a well-defined probability distribution, it suffices to normalize it by dividing the sum of all its coefficients. In the example given in Fig. 2.1, the normalization constant can be computed by $594 + 1776 + 306 + 824 = V_{\mathrm{root}}(\mathbf{1} \mid \mathbf{w})$, where we simply set all the leaf indicator variables to be 1 and the output of the network gives the desired normalization constant.

Formally, the probability/density of a joint assignment $\mathbf{X} = \mathbf{x}$ in an SPN $\mathcal{S}$ is defined to be proportional to the value at the root of the SPN with input $\mathbf{x}$ divided by a normalization constant $V_{\mathrm{root}}(\mathbf{1} \mid \mathbf{w})$:

$$\Pr_{\mathcal{S}}(\mathbf{x}) = \frac{V_{\mathrm{root}}(\mathbf{x} \mid \mathbf{w})}{V_{\mathrm{root}}(\mathbf{1} \mid \mathbf{w})} \tag{2.4}$$

Intuitively, setting all the input leaf indicator variables to be 1 corresponds to integrating/marginalizing out the random vector $\mathbf{X}$, which will ensure Eq. (2.4) defines a proper probability distribution. Eq. (2.4) can also be used to compute the marginal probability of a partial assignment $\mathbf{Y} = \mathbf{y}$: simply set all the leaf indicator variables whose corresponding random variable is not in $\mathbf{Y}$ to be 1 and other leaf nodes based on the assignment $\mathbf{Y} = \mathbf{y}$. Equivalently, this corresponds to integrating out variables outside of the partial assignment. We can compute conditional probabilities by evaluating two partial assignments:

$$\Pr_{\mathcal{S}}(\mathbf{Y} = \mathbf{y} \mid \mathbf{Z} = \mathbf{z}) = \frac{\Pr_{\mathcal{S}}(\mathbf{Y} = \mathbf{y}, \mathbf{Z} = \mathbf{z})}{\Pr_{\mathcal{S}}(\mathbf{Z} = \mathbf{z})} = \frac{V_{\mathrm{root}}(\mathbf{y}, \mathbf{z} \mid \mathbf{w})}{V_{\mathrm{root}}(\mathbf{z} \mid \mathbf{w})}$$

Since joint, marginal and conditional queries can all be computed by two network passes, exact inference takes linear time with respect to the network size:

Theorem 2.4.1 (Linear time exact inference (Poon and Domingos, 2011))**.** Let $\mathcal{S}$ be a complete and decomposable SPN over boolean random vector $\mathbf{X}_{[n]}$, then $\forall \mathbf{Y}, \mathbf{Z} \subseteq \mathbf{X}_{[n]}$, $\mathbf{Y} \bigcap \mathbf{Z} = \varnothing$, $\Pr_{\mathcal{S}}(\mathbf{Y} \mid \mathbf{Z})$ can be computed in $O(|\mathcal{S}|)$.

Thm. 2.4.1 is perhaps the most interesting property about SPN that distinguishes it from classical probabilistic graphical models, including Bayesian networks, Markov random fields, factor graphs and junction trees. It is well known that exact inference in those models is computationally hard. More precisely, for a Bayesian network $\mathcal{B}$, the complexity for exact inference in $\mathcal{B}$ is given by $O(2^{\text{tw}(\mathcal{B})})$, but it is NPC to determine whether a given graph $\mathcal{B}$ has tree-width at most a given number t (Arnborg et al., 1987). Hence practitioners often have to resort to various approximate inference schemes, e.g., loopy belief propagation (Murphy et al., 1999), Markov chain Monte Carlo (MCMC) sampling (Neal, 1993), variational relaxation (Sontag et al., 2011), to name a few. On the other hand, despite the fact that marginal inference is exact and intractable in SPNs, the maximum-a-posteriori (MAP) inference is still NP-hard (Choi and Darwiche, 2017; Mei et al., 2017; Peharz, 2015).

It has been recently shown that any complete and decomposable SPN $\mathcal{S}$ over $\mathbf{X}_{[n]}$ is equivalent to a BN with $O(n|\mathcal{S}|)$ size (Zhao et al., 2015b), if we are allowed to use a more compact data structure, e.g., the algebraic decision diagram (ADD) (Bahar et al., 1997) instead of the usual conditional probability table, to represent the conditional probability distribution at each node of the BN. The insight behind the construction is that each internal sum node in $\mathcal{S}$ corresponds to a latent variable in the constructed BN, and the BN will be a bipartite graph with one layer of local latent variables pointing to one layer of observable variables $\mathbf{X}$. An observable variable is a child of a local latent variable if and only if the observable variable appears as a descendant of the latent variable (sum node) in the original SPN. Equivalently, this shows that the SPN $\mathcal{S}$ can be interpreted as a BN where the number of latent variables per instance is $O(|\mathcal{S}|)$. This BN perspective provides an interesting connection between the two models, and later we shall reply on this equivalence to develop Bayesian variational inference method for learning the parameters of SPNs.

2.5 Geometric and Signomial Programming

Before we introduce signomial programming (SP), we shall first introduce geometric programming (GP), which is a strict subclass of SP. A *monomial* is defined as a function $h : \mathbb{R}^n_{++} \mapsto \mathbb{R}$: $h(\mathbf{x}) = dx_1^{a_1}x_2^{a_2}\cdots x_n^{a_n}$, where the domain is restricted to be the positive orthant ($\mathbb{R}^n_{++}$), the coefficient d is positive and the exponents $a_i \in \mathbb{R}, \forall i$. A *posynomial* is a sum of monomials: $g(\mathbf{x}) = \sum_{k=1}^K d_k x_1^{a_{1k}} x_2^{a_{2k}} \cdots x_n^{a_{nk}}$. One of the key properties of posynomials is positivity, which allows us to transform any posynomial into the log domain. A GP in standard form is defined to be an optimization problem where both the objective function and the inequality constraints are posynomials and the equality constraints are monomials. There is also an implicit constraint that $\mathbf{x} \in \mathbb{R}^n_{++}$.

$$\begin{aligned}
\text{minimize} \quad & \sum_{k=1}^{K_0} d_{0k} \prod_{t=1}^{n} \mathbf{x}_t^{a_{0kt}} \\
\text{subject to} \quad & \sum_{k=1}^{K_i} d_{ik} \prod_{t=1}^{n} \mathbf{x}_t^{a_{ikt}} \leq 1, \quad i \in [p] \\
& d_j \prod_{t=1}^{n} \mathbf{x}_t^{a_{jt}} = 1, \quad j \in [q]
\end{aligned} \tag{2.5}$$

In its standard form GP is not a convex program since posynomials are not convex functions in general. However, we can effectively transform it into a convex problem by using the *logarithmic transformation trick* on $\mathbf{x}$, the multiplicative coefficients of each monomial and also each objective/constraint function (Boyd et al., 2007; Chiang, 2005). We make the following change: let $\mathbf{y} = \log(\mathbf{x})$, $c_{ik} = \log(d_{ik}), \forall i, k$ and take

$\log(\cdot)$ of each function in ((2.5)). Then the standard GP form is equivalent to the following formulation:

$$\begin{aligned} \text{minimize} \quad & \log\left(\sum_{k=1}^{K_0} \exp(\mathbf{a}_{0k}^T \mathbf{y} + c_{0k})\right) \\ \text{subject to} \quad & \log\left(\sum_{k=1}^{K_i} \exp(\mathbf{a}_{ik}^T \mathbf{y} + c_{ik})\right) \leq 0, \quad i \in [p] \\ & \mathbf{a}_j^T \mathbf{y} + c_j = 0, \quad j \in [q] \end{aligned} \tag{2.6}$$

which is a convex program since the *log-sum-exp* function is convex in its argument and $\mathbf{a}^T\mathbf{y} + c$ is affine in $\mathbf{y}$. Furthermore, in the convex formulation of GP we have $\mathbf{y} \in \mathbb{R}^n$, i.e., we naturally remove the positive constraint on $\mathbf{x}$ by taking the log transformation.

An SP has the same form as GP except that the multiplicative constant d inside each monomial is not restricted to be positive, i.e., d can take any real value. Although the difference seems to be small, there is a huge difference between GP and SP from the computational perspective. The negative multiplicative constant in monomials invalidates the logarithmic transformation trick frequently used in GP. As a result, SPs cannot be reduced to convex programs and are believed to be hard to solve in general (Boyd et al., 2007).

2.6 Some Technical Tools

We first introduce the concept of *Rademacher complexity*, which will be frequently used in the following chapters to prove data-dependent generalization bounds.

Definition 2.6.1 (Empirical Rademacher Complexity)**.** Let $\mathcal{H}$ be a family of functions mapping from $\mathcal{X}$ to $[a, b]$ and $\mathbf{S} = \{\mathbf{x}_i\}_{i=1}^n$ a fixed sample of size n with elements in $\mathcal{X}$. Then, the *empirical Rademacher complexity* of $\mathcal{H}$ with respect to the sample X is defined as

$$\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}) := \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i h(\mathbf{x}_i)\right],$$

where $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^n$ and σ_i are i.i.d. uniform random variables taking values in $\{+1, -1\}$.

Roughly speaking, the Rademacher complexity measures the ability of a hypothesis class to fit uniformly random noise, hence it could be understood as a measure of the richness of hypothesis classes. The following lemma is particularly useful to provide data-dependent guarantees in terms of the empirical Rademacher complexity:

Lemma 2.6.1 (Bartlett and Mendelson (2002))**.** Let $\mathcal{H} \subseteq [0, 1]^{\mathcal{X}}$, then for $\forall\delta > 0$, w.p.b. at least $1 - \delta$, the following inequality holds for $\forall h \in \mathcal{H}$:

$$\mathbb{E}[h(\mathbf{x})] \leq \frac{1}{n}\sum_{i=1}^n h(\mathbf{x}_i) + 2\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}) + 3\sqrt{\frac{\log(2/\delta)}{2n}} \tag{2.7}$$

Ledoux-Talagrand's contraction lemma is a useful technique in upper bounding the Rademacher complexity of function compositions:

Lemma 2.6.2 (Ledoux-Talagrand's contraction lemma)**.** Let $\phi : \mathbb{R} \mapsto \mathbb{R}$ be a Lipschitz function with parameter L, i.e., $\forall a, b \in \mathbb{R}$, $|\phi(a) - \phi(b)| \leq L|a - b|$. Then,

$$\mathrm{Rad}_{\mathbf{S}}(\phi \circ \mathcal{H}) = \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i\phi(h(\mathbf{x}_i))\right] \leq L\,\mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i h(\mathbf{x}_i)\right] = L\,\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}),$$

where $\phi \circ \mathcal{H} := \{\phi \circ h \mid h \in \mathcal{H}\}$ is the class of composite functions.

Throughout the thesis, we will frequently use information-theoretic concepts and tools to understand various phenomenons in learning representations. Here we give a brief introduction to the ones that we use in the rest of the chapters.

For two distributions $\mathcal{D}$ and $\mathcal{D}'$, the Jensen-Shannon (JS) divergence $D_{\text{JS}}(\mathcal{D} \parallel \mathcal{D}')$ is defined as:

$$D_{\text{JS}}(\mathcal{D} \parallel \mathcal{D}') := \frac{1}{2}D_{\text{KL}}(\mathcal{D} \parallel \mathcal{D}_M) + \frac{1}{2}D_{\text{KL}}(\mathcal{D}' \parallel \mathcal{D}_M),$$

where $D_{\text{KL}}(\cdot \parallel \cdot)$ is the Kullback–Leibler (KL) divergence and $\mathcal{D}_M := (\mathcal{D} + \mathcal{D}')/2$. The JS divergence can be viewed as a symmetrized and smoothed version of the KL divergence, and it is closely related to the L_1 distance between two distributions through Lin's lemma (Lin, 1991).

Unlike the KL divergence, the JS divergence is bounded: $0 \leq D_{\text{JS}}(\mathcal{D} \parallel \mathcal{D}') \leq 1$. Additionally, from the JS divergence, we can define a distance metric between two distributions as well, known as the JS distance (Endres and Schindelin, 2003):

$$d_{\text{JS}}(\mathcal{D}, \mathcal{D}') := \sqrt{D_{\text{JS}}(\mathcal{D} \parallel \mathcal{D}')}.$$

Lin's lemma gives an upper bound of the JS divergence between two distributions via the L_1 distance (total variation distance).

Lemma 2.6.3 (Theorem. 3, (Lin, 1991))**.** Let $\mathcal{D}$ and $\mathcal{D}'$ be two distributions, then $D_{\text{JS}}(\mathcal{D}, \mathcal{D}') \leq \frac{1}{2}\|\mathcal{D} - \mathcal{D}'\|_1$.

The following inequality is the celebrated data-processing inequality:

Lemma 2.6.4 (Data processing inequality)**.** Let $X \to Z \to Y$ be a Markov chain, then $I(X; Z) \geq I(X; Y)$, where $I(\cdot; \cdot)$ is the mutual information.

The following inequality is a special case of the Fubini's theorem:

Theorem 2.6.1 (Fubini's Theorem)**.** Let X be a nonnegative random variable, then $\mathbb{E}[X] = \int_0^\infty \Pr(X \geq t)\, dt$.

Finally, the following Hoeffding's bound will be a powerful tool to give concentration arguments:

Theorem 2.6.2 (Hoeffding's inequality)**.** Let $\{X_t\}_{t=1}^n$ be a sequence of independent random variables such that $\forall t \in [n], \mathbb{E}[X_t] = 0$ and $X_t \in [a_t, b_t]$ almost surely. Then the following inequality holds:

$$\Pr\left(\left|\frac{1}{n}\sum_{t=1}^{n} X_t\right| \geq \epsilon\right) \leq 2\exp\left(-\frac{2n^2\epsilon^2}{\sum_{t=1}^{n}(b_t - a_t)^2}\right).$$

Other concentration inequalities will be introduced in the corresponding chapters when necessary.

Part I

Understanding and Learning Invariant Representations

Chapter 3

Domain Adaptation with Multiple Source Environments

While domain adaptation has been actively researched, most algorithms focus on the single-source-single-target adaptation setting. In this chapter we propose new generalization bounds and algorithms under both classification and regression settings for unsupervised multiple source domain adaptation. Our theoretical analysis naturally leads to an efficient learning strategy using adversarial neural networks: we show how to interpret it as learning feature representations that are invariant to the multiple domain shifts while still being discriminative for the learning task. To this end, we propose multisource domain adversarial networks (MDAN) that approach domain adaptation by optimizing task-adaptive generalization bounds. To demonstrate the effectiveness of MDAN, we conduct extensive experiments showing superior adaptation performance on both classification and regression problems: sentiment analysis, digit classification, and vehicle counting.

3.1 Introduction

The success of machine learning has been partially attributed to rich datasets with abundant annotations (Russakovsky et al., 2015). Unfortunately, collecting and annotating such large-scale training data is prohibitively expensive and time-consuming. To solve these limitations, different labeled datasets can be combined to build a larger one, or synthetic training data can be generated with explicit yet inexpensive annotations (Shrivastava et al., 2016). However, due to the possible shift between training and test samples, learning algorithms based on these cheaper datasets still suffer from high generalization error. Domain adaptation (DA) focuses on such problems by establishing knowledge transfer from a labeled source domain to an unlabeled target domain, and by exploring domain-invariant structures and representations to bridge the gap (Pan and Yang, 2010). Both theoretical results (Ben-David et al., 2010; Gopalan et al., 2014; Mansour and Schain, 2012; Mansour et al., 2009a; Xu and Mannor, 2012) and algorithms (Adel et al., 2017; Ajakan et al., 2014; Becker et al., 2013; Ghifary et al., 2015; Glorot et al., 2011; Hoffman et al., 2012, 2017b; Jhuo et al., 2012; Long et al., 2015; Pei et al., 2018) for DA have been proposed. Most theoretical results and algorithms with respect to DA focus on the single-source-single-target setting (Ganin et al., 2016; Louizos et al., 2015; Shu et al., 2018; Tzeng et al., 2015, 2017). However, in many application scenarios, the labeled data available may come from multiple domains with different distributions. As a result, naive application of the single-source-single-target DA algorithms may lead to suboptimal solutions. Such problem calls for an efficient technique for multiple source domain adaptation. Some existing multisource DA methods (Gan et al., 2016; Hoffman et al., 2012, 2017a; Sun et al., 2011; Zhang et al., 2015a) cannot lead to effective deep learning based algorithms, leaving much space to be improved for their performance.

In this chapter we analyze the multiple source domain adaptation problem and propose an adversarial learning strategy based on our theoretical results. Specifically, we give new generalization bounds for both classification and regression problems under domain adaptation when there are multiple source domains with labeled instances and one target domain with unlabeled instances. Our theoretical results build on the seminal theoretical model for domain adaptation introduced by Blitzer et al. (2008) and Ben-David et al. (2010), where a divergence measure, known as the $\mathcal{H}$-divergence, was proposed to measure the distance between two distributions based on a given hypothesis space $\mathcal{H}$. Our new result generalizes the bound (Ben-David et al., 2010, Thm. 2) to the case when there are multiple source domains, and to regression problems as well. The new bounds achieve a finite sample error rate of $\tilde{O}(\sqrt{1/km})$, where k is the number of source domains and m is the number of labeled training instances from each domain.

Interestingly, our bounds also lead to an efficient algorithm using adversarial neural networks. This algorithm learns both domain invariant and task discriminative features under multiple domains. Specifically, we propose a novel MDAN model by using neural networks as rich function approximators to instantiate the generalization bound we derive (Fig. 3.1). MDAN can be viewed as computationally efficient approximations to optimize the parameters of the networks in order to minimize the bounds. We introduce two versions of MDAN: The hard version optimizes directly a simple worst-case generalization bound, while the soft version leads to a more data-efficient model and optimizes an average case and task-adaptive bound. The optimization of MDAN is a minimax saddle point problem, which can be interpreted as a zero-sum game with two participants competing against each other to learn invariant features. MDAN combine feature extraction, domain classification, and task learning in one training process. We propose to use stochastic optimization with simultaneous updates to optimize the parameters in each iteration.

Contributions Our contributions in this chapter are three-fold:

1. Theoretically, we provide average case generalization bounds for both classification and regression problems under the multisource domain adaptation setting.

2. Inspired by our theoretical results, we also propose efficient algorithms that tackle multisource domain adaptation problems using adversarial learning strategy.
3. Empirically, to demonstrate the effectiveness of MDAN as well as the relevance of our theoretical results, we conduct extensive experiments on real-world datasets, including both natural language and vision tasks, classification and regression problems. We achieve consistently superior adaptation performances on all the tasks, validating the effectiveness of our models.

3.2 Preliminaries

We first introduce the problem setup of domain adaptation and review a theoretical model for domain adaptation when there is one source and one target (Ben-David et al., 2007, 2010; Blitzer et al., 2008; Kifer et al., 2004). The key idea is the $\mathcal{H}$-divergence to measure the discrepancy between two distributions. Other theoretical models for DA exist (Cortes and Mohri, 2014; Cortes et al., 2008; Mansour et al., 2009a,c); we choose to work with the above model because this distance measure has a particularly natural interpretation and can be well approximated using samples from both domains.

Problem Setup We use *domain* to represent a distribution $\mathcal{D}$ on input space $\mathcal{X}$ and a labeling function $f : \mathcal{X} \to [0, 1]$. In the setting of one source one target domain adaptation, we use $\langle \mathcal{D}_S, f_S \rangle$ and $\langle \mathcal{D}_T, f_T \rangle$ to denote the source and target, respectively. A *hypothesis* is a function $h : \mathcal{X} \to [0, 1]$. The *error* of a hypothesis h w.r.t. a labeling function f under distribution $\mathcal{D}_S$ is defined as: $\varepsilon_S(h, f) := \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[|h(\mathbf{x}) - f(\mathbf{x})|]$. When f and h are binary classification functions, this definition reduces to the probability that h disagrees with f under $\mathcal{D}_S$: $\mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[|h(\mathbf{x}) - f(\mathbf{x})|] = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}_S}[\mathbb{I}(f(\mathbf{x}) \neq h(\mathbf{x}))] = \Pr_{\mathbf{x}\sim\mathcal{D}_S}(f(\mathbf{x}) \neq h(\mathbf{x}))$.

We define the *risk* of hypothesis h as the error of h w.r.t. a true labeling function under domain $\mathcal{D}_S$, i.e., $\varepsilon_S(h) := \varepsilon_S(h, f_S)$. As common notation in computational learning theory, we use $\widehat{\varepsilon}_S(h)$ to denote the empirical risk of h on the source domain. Similarly, we use $\varepsilon_T(h)$ and $\widehat{\varepsilon}_T(h)$ to mean the true risk and the empirical risk on the target domain. $\mathcal{H}$-divergence is defined as follows:

Definition 3.2.1. Let $\mathcal{H}$ be a hypothesis class for instance space $\mathcal{X}$, and $\mathcal{A}_\mathcal{H}$ be the collection of subsets of $\mathcal{X}$ that are the support of some hypothesis in $\mathcal{H}$, i.e., $\mathcal{A}_\mathcal{H} := \{h^{-1}(\{1\}) \mid h \in \mathcal{H}\}$. The distance between two distributions $\mathcal{D}$ and $\mathcal{D}'$ based on $\mathcal{H}$ is: $d_\mathcal{H}(\mathcal{D}, \mathcal{D}') := 2\sup_{A\in\mathcal{A}_\mathcal{H}} |\Pr_\mathcal{D}(A) - \Pr_{\mathcal{D}'}(A)|$.

When the hypothesis class $\mathcal{H}$ contains all the possible measurable functions over $\mathcal{X}$, $d_\mathcal{H}(\mathcal{D}, \mathcal{D}')$ reduces to the familiar total variation. Given a hypothesis class $\mathcal{H}$, we define its symmetric difference w.r.t. itself as: $\mathcal{H}\Delta\mathcal{H} = \{h(\mathbf{x}) \oplus h'(\mathbf{x}) \mid h, h' \in \mathcal{H}\}$, where $\oplus$ is the XOR operation. Let h^* be the optimal hypothesis that achieves the minimum combined risk on both the source and the target domains: $h^* := \arg\min_{h\in\mathcal{H}} \varepsilon_S(h) + \varepsilon_T(h)$, and use λ to denote the combined risk of the optimal hypothesis h^*: $\lambda := \varepsilon_S(h^*) + \varepsilon_T(h^*)$. Ben-David et al. (2007) and Blitzer et al. (2008) proved the following generalization bound on the target risk in terms of the source risk and the discrepancy between the single source domain and the target domain:

Theorem 3.2.1 (Blitzer et al. (2008)). Let $\mathcal{H}$ be a hypothesis space of VC-dimension d and $\widehat{\mathcal{D}}_S$ ($\widehat{\mathcal{D}}_T$) be the empirical distribution induced by sample of size m drawn from $\mathcal{D}_S$ ($\mathcal{D}_T$). Then w.p.b. at least $1 - \delta$, $\forall h \in \mathcal{H}$,

$$\varepsilon_T(h) \leq \widehat{\varepsilon}_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T) + \lambda + O\left(\sqrt{\frac{d\log(m/d) + \log(1/\delta)}{m}}\right). \tag{3.1}$$

The bound depends on λ, the optimal combined risk that can be achieved by hypothesis in $\mathcal{H}$. The intuition is if λ is large, we cannot hope for a successful domain adaptation. One notable feature is that the empirical discrepancy distance between two samples can be approximated by a discriminator to distinguish instances from two domains.

3.3 Generalization Bound for Multiple Source Domain Adaptation

In this section we discuss two approaches to obtain generalization guarantees for multiple source domain adaptation in both classification and regression settings, one by a union bound argument and one using reduction from multiple source domains to single source domain. We conclude this section with a discussion and comparison of our bounds with existing generalization bounds for multisource domain adaptation (Ben-David et al., 2010; Mansour et al., 2009c). We defer the detailed proof in Sec. 3.7 and we mainly focus on discussing the interpretations and implications of the theorems.

Let $\{\mathcal{D}_{S_i}\}_{i=1}^k$ and $\mathcal{D}_T$ be k source domains and the target domain, respectively. One idea to obtain a generalization bound for multiple source domains is to apply Thm. 3.2.1 repeatedly k times, followed by a union bound to combine them. Following this idea, we first obtain the following bound as a corollary of Thm. 3.2.1 in the setting of multiple source domains, serving as a baseline model:

Corollary 3.3.1 (Worst case classification bound)**.** Let $\mathcal{H}$ be a hypothesis class with $VCdim(\mathcal{H}) = d$. If $\widehat{\mathcal{D}}_T$ and $\{\widehat{\mathcal{D}}_{S_i}\}_{i=1}^k$ are the empirical distributions generated with m *i.i.d.* samples from each domain, then, for $0 < \delta < 1$, with probability at least $1 - \delta$, for all $h \in \mathcal{H}$, we have:

$$\varepsilon_T(h) \leq \max_{i\in[k]} \left\{ \widehat{\varepsilon}_{S_i}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T; \widehat{\mathcal{D}}_{S_i}) + \lambda_i \right\} + O\left(\sqrt{\frac{1}{m}\left(\log\frac{k}{\delta} + d\log\frac{m}{d}\right)} \right), \tag{3.2}$$

where λ_i is the combined risk of the optimal hypothesis on domains S_i and T.

Remark This bound is quite pessimistic, as it essentially is a worst case bound, where the generalization on the target only depends on the worst source domain. However, in many real-world scenarios, when the number of related source domains is large, a single irrelevant source domain may not hurt the generalization too much. Furthermore, in the case of multiple source domains, despite the possible discrepancy between the source domains and the target domain, effectively we have a labeled sample of size km, while the asymptotic convergence rate in Corollary. 3.3.1 is of $\tilde{O}(\sqrt{1/m})$. Hence naturally one interesting question to ask is: is it possible to have a generalization bound of finite sample rate $\tilde{O}(\sqrt{1/km})$? In what follows we present a strategy to achieve a generalization bound of rate $\tilde{O}(\sqrt{1/km})$. The idea of this strategy is a reduction using convex combination from multiple domains to single domain by combining all the labeled instances from k domains to one.

Theorem 3.3.1 (Average case classification bound)**.** Let $\mathcal{H}$ be a hypothesis class with $VCdim(\mathcal{H}) = d$. If $\{\widehat{\mathcal{D}}_{S_i}\}_{i=1}^k$ are the empirical distributions generated with m *i.i.d.* samples from each domain, and $\widehat{\mathcal{D}}_T$ is the empirical distribution on the target domain generated from mk samples without labels, then, $\forall \alpha \in \mathbb{R}_+^k, \sum_{i\in[k]} \alpha_i = 1$, and for $0 < \delta < 1$, w.p.b. at least $1 - \delta$, for all $h \in \mathcal{H}$, we have:

$$\varepsilon_T(h) \leq \sum_{i\in[k]} \alpha_i \left(\widehat{\varepsilon}_{S_i}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T; \widehat{\mathcal{D}}_{S_i}) \right) + \lambda_\alpha + O\left(\sqrt{\frac{1}{km}\left(\log\frac{1}{\delta} + d\log\frac{km}{d}\right)} \right), \tag{3.3}$$

where λ_α is the risk of the optimal hypothesis on the mixture source domain $\sum_{i\in[k]} \alpha_i S_i$ and T.

Different from Corollary 3.3.1, Thm. 3.3.1 requires mk unlabeled instances from the target domain. This is a mild requirement since unlabeled data is cheap to collect. Roughly, the bound in Thm. 3.3.1 can be understood as an average case bound if we choose $\alpha_i = 1/k, \forall i \in [k]$. Note that a simple convex combination by applying Thm. 3.2.1 k times can only achieve finite sample rate of $\tilde{O}(\sqrt{1/m})$, while the one in (3.3) achieves $\tilde{O}(\sqrt{1/km})$. On the other hand, the constants $\max_{i\in[k]}\lambda_i$ (in Corollary 3.3.1) and λ_α (in Thm. 3.3.1) are generally not comparable. As a final note, although the proof works for any convex combination α_i, in the next section we will describe a practical method so that we do not need to explicitly choose it. Thm. 3.3.1 upper bounds the generalization error for classification problems. Next we also provide generalization guarantee for regression problem, where instead of VC dimension, we use pseudo-dimension to characterize the structural complexity of the hypothesis class.

Theorem 3.3.2 (Average case regression bound)**.** Let $\mathcal{H}$ be a set of real-valued functions from $\mathcal{X}$ to $[0,1]$[1] with $Pdim(\mathcal{H}) = d$. If $\{\widehat{\mathcal{D}}_{S_i}\}_{i=1}^k$ are the empirical distributions generated with m *i.i.d.* samples from each domain, and $\widehat{\mathcal{D}}_T$ is the empirical distribution on the target domain generated from mk samples without labels, then, $\forall\alpha \in \mathbb{R}_+^k, \sum_{i\in[k]}\alpha_i = 1$, and for $0 < \delta < 1$, with probability at least $1-\delta$, for all $h \in \mathcal{H}$, we have:

$$\varepsilon_T(h) \leq \sum_{i\in[k]}\alpha_i\left(\widehat{\varepsilon}_{S_i}(h) + \frac{1}{2}d_{\bar{\mathcal{H}}}(\widehat{\mathcal{D}}_T;\widehat{\mathcal{D}}_{S_i})\right) + \lambda_\alpha + O\left(\sqrt{\frac{1}{km}\left(\log\frac{1}{\delta} + d\log\frac{km}{d}\right)}\right), \tag{3.4}$$

where λ_α is the risk of the optimal hypothesis on the mixture source domain $\sum_{i\in[k]}\alpha_i S_i$ and T, and $\bar{\mathcal{H}} := \{\mathbb{I}_{|h(x)-h'(x)|>t} : h, h' \in \mathcal{H}, 0 \leq t \leq 1\}$ is the set of threshold functions induced from $\mathcal{H}$.

Comparison with Existing Bounds First, it is easy to see that, the bounds in both (3.2) and (3.3) reduce to the one in Thm. 3.2.1 when there is only one source domain ($k = 1$). Blitzer et al. (2008) give a generalization bound for semi-supervised classification with multiple sources where, besides labeled instances from multiple source domains, the algorithm also has access to a fraction of labeled instances from the target domain. Although in general our bound and the one in (Blitzer et al., 2008, Thm. 3) are incomparable, it is instructive to see the connections and differences between them: our bound works in the unsupervised domain adaptation setting where we do not have any labeled data from the target. As a comparison, their bound in (Blitzer et al., 2008, Thm. 3) is a bound for semi-supervised domain adaptation. As a result, because of the access to labeled instances from the target domain, their bound is expressed relative to the optimal error on the target, while ours is in terms of the empirical error on the source domains, hence theirs is more informative. To the best of our knowledge, our bound in Thm. 3.3.2 is the first one using the idea of $\mathcal{H}$-divergence for regression problems. The proof of this theorem relies on a reduction from regression to classification. Mansour et al. (2009b) give a generalization bound for multisource domain adaptation under the assumption that the target distribution is a mixture of the k sources and the target hypothesis can be represented as a convex combination of the source hypotheses. Also, their generalized discrepancy measure can be applied for other loss functions.

3.4 Multisource Domain Adaptation with Adversarial Neural Networks

Motivated by the bounds given in the last section, in this section we propose our model, multisource domain adversarial networks (MDAN), with two versions: Hard version (as a baseline) and Soft version. Suppose we are given samples drawn from k source domains $\{\mathcal{D}_{S_i}\}$, each of which contains m instance-label pairs. Additionally, we also have access to unlabeled instances sampled from the target domain $\mathcal{D}_T$. Once we fix our hypothesis class $\mathcal{H}$, the last two terms in the generalization bounds (3.2) and (3.3) will be fixed; hence we can only hope to minimize the bound by minimizing the first two terms, i.e., the source training error

[1] This is just for the simplicity of presentation, the range can easily be generalized to any bounded set.

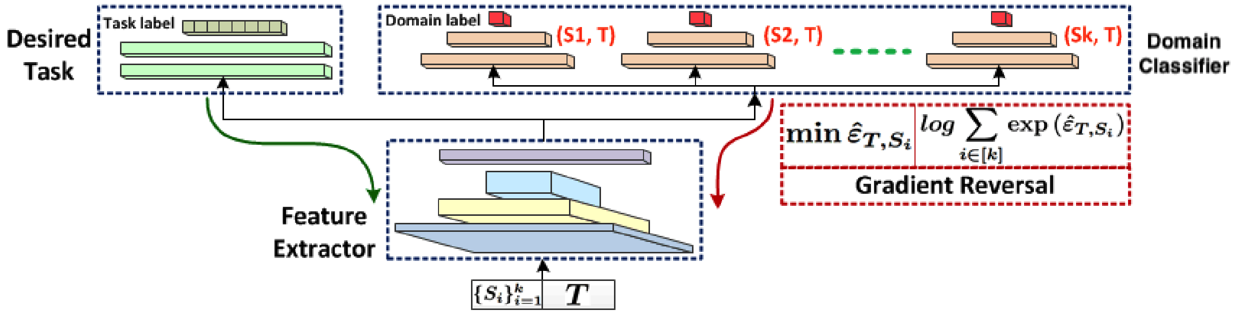


Figure 3.1: MDAN Network architecture. Feature extractor, domain classifier, and task learning are combined in one training process. Hard version: the source that achieves the minimum domain classification error is backpropagated with gradient reversal; Smooth version: all the domain classification risks over k source domains are combined and backpropagated adaptively with gradient reversal.

and the discrepancy between source domains and target domain. The idea is to train a neural network to learn a representation with the following two properties:

1. Indistinguishable between the k source domains and the target domain;
2. Informative enough for our desired task to succeed.

Note that both requirements are necessary: without the second property, a neural network can learn trivial random noise representations for all the domains, and such representations cannot be distinguished by any discriminator; without the first property, the learned representation does not necessarily generalize to the unseen target domain.

One key observation that leads to a practical approximation of $d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T; \widehat{\mathcal{D}}_{S_i})$ from Ben-David et al. (2007) is that computing the discrepancy measure is closely related to learning a classifier that is able to distinguish samples from different domains. Let $\widehat{\varepsilon}_{T,S_i}(h)$ be the empirical risk of hypothesis h in the domain discriminating task. Ignoring the constant terms that do not affect the upper bound, we can minimize the worst case upper bound in (3.2) by solving the following optimization problem:

$$\textbf{Hard version:} \quad \text{minimize} \quad \max_{i\in[k]} \left(\widehat{\varepsilon}_{S_i}(h) - \min_{h'\in\mathcal{H}\Delta\mathcal{H}} \widehat{\varepsilon}_{T,S_i}(h') \right) \tag{3.5}$$

The two terms in (3.5) exactly correspond to the two criteria we just proposed: the first term asks for an informative feature representation for our desired task to succeed, while the second term captures the notion of invariant feature representations between different domains. Inspired by Ganin et al. (2016), we use the gradient reversal layer to effectively implement (3.5) by backpropagation. The network architecture is shown in Fig. 3.1. As discussed in the last section, one notable drawback of the hard version is that the algorithm may spend too much computational resources in optimizing the worst source domain. Furthermore, in each iteration the algorithm only updates its parameter based on the gradient from one of the k domains. This is data inefficient and can waste our computational resources in the forward process.

To avoid both of the problems, we propose the MDAN Soft version that optimizes an upper bound of the convex combination bound given in (3.3). To this end, define $\widehat{\varepsilon}_i(h) := \widehat{\varepsilon}_{S_i}(h) - \min_{h'\in\mathcal{H}\Delta\mathcal{H}} \widehat{\varepsilon}_{T,S_i}(h')$ and let $\gamma > 0$ be a constant. We formulate the following optimization problem:

$$\textbf{Soft version:} \quad \text{minimize} \quad \frac{1}{\gamma} \log \sum_{i\in[k]} \exp\left(\gamma(\widehat{\varepsilon}_{S_i}(h) - \min_{h'\in\mathcal{H}\Delta\mathcal{H}} \widehat{\varepsilon}_{T,S_i}(h')) \right) \tag{3.6}$$

At the first glance, it may not be clear what the above objective function corresponds to. To understand this,

Algorithm 1 Multiple Source Domain Adaptation

```
for $t = 1$ to $\infty$ do
    Sample $\{S_i^{(t)}\}_{i=1}^k$ and $T^{(t)}$ from $\{\widehat{\mathcal{D}}_{S_i}\}_{i=1}^k$ and $\widehat{\mathcal{D}}_T$, each of size $m$
    for $i = 1$ to $k$ do
        $\widehat{\varepsilon}_i^{(t)} \leftarrow \widehat{\varepsilon}_{S_i^{(t)}}(h) - \min_{h' \in \mathcal{H}\Delta\mathcal{H}} \widehat{\varepsilon}_{T^{(t)}, S_i^{(t)}}(h')$
        Compute $w_i^{(t)} := \exp(\widehat{\varepsilon}_i^{(t)})$
    end for
    Select $i^{(t)} := \arg\max_{i \in [k]} \widehat{\varepsilon}_i^{(t)}$                                // Hard version
    Backpropagate gradient of $\widehat{\varepsilon}_{i^{(t)}}^{(t)}$
    for $i = 1$ to $k$ do
        Normalize $w_i^{(t)} \leftarrow w_i^{(t)} / \sum_{i' \in [k]} w_{i'}^{(t)}$                    // Soft version
    end for
    Backpropagate gradient of $\sum_{i \in [k]} w_i^{(t)} \widehat{\varepsilon}_i^{(t)}$
end for
```

if we define $\alpha_i = \exp(\widehat{\varepsilon}_i(h)) / \sum_{j \in [k]} \exp(\widehat{\varepsilon}_j(h))$, then the following inequality holds:

$$\sum_{i \in [k]} \alpha_i \widehat{\varepsilon}_i(h) \leq \log\left(\mathbb{E}_\alpha[\exp(\widehat{\varepsilon}_i(h))]\right) = \log\left(\frac{\sum_{i \in [k]} \exp^2(\widehat{\varepsilon}_i(h))}{\sum_{i \in [k]} \exp(\widehat{\varepsilon}_i(h))}\right) \leq \log \sum_{i \in [k]} \exp(\widehat{\varepsilon}_i(h)).$$

In other words, the objective function in (3.6) is in fact an upper bound of the convex combination bound given in (3.3), with the combination weight α defined above. Compared with the one in (3.3), one advantage of the objective function in (3.6) is that we do not need to explicitly choose the value of α. Instead, it adaptively corresponds to the loss $\widehat{\varepsilon}_i(h)$, and the larger the loss, the heavier the weight.

Alternatively, from the algorithmic perspective, during the optimization (3.6) naturally provides an adaptive weighting scheme for the k source domains depending on their relative error. Use θ to denote all the model parameters:

$$\frac{\partial}{\partial \theta} \frac{1}{\gamma} \log \sum_{i \in [k]} \exp\left(\gamma(\widehat{\varepsilon}_{S_i}(h) - \min_{h' \in \mathcal{H}\Delta\mathcal{H}} \widehat{\varepsilon}_{T,S_i}(h'))\right) = \sum_{i \in [k]} \frac{\exp \gamma \widehat{\varepsilon}_i(h)}{\sum_{i' \in [k]} \exp \gamma \widehat{\varepsilon}_{i'}(h)} \frac{\partial \widehat{\varepsilon}_i(h)}{\partial \theta}. \tag{3.7}$$

Compared with (3.5), the log-sum-exp trick not only smooths the objective, but also provides a principled and adaptive way to combine all the gradients from the k source domains. In words, (3.7) says that the gradient of MDAN is a convex combination of the gradients from all the domains. The larger the error from one domain, the larger the combination weight in the ensemble. As we will see in Sec. 3.5, the optimization problem (3.6) often leads to better generalizations in practice, which may partly be explained by the ensemble effect of multiple sources implied by the upper bound. Pseudocode of both algorithms is listed in Alg. 1.

3.5 Experiments

We evaluate both hard and soft MDAN and compare them with state-of-the-art methods on three real-world datasets: the Amazon benchmark dataset (Chen et al., 2012) for sentiment analysis, a digit classification task that includes 4 datasets: MNIST (LeCun et al., 1998b), MNIST-M (Ganin et al., 2016), SVHN (Netzer et al., 2011), and SynthDigits (Ganin et al., 2016), and a public, large-scale image dataset on vehicle counting from multiple city cameras (Zhang et al., 2017a).

3.5.1 Amazon Reviews

Domains within the dataset consist of reviews on a specific kind of product (Books, DVDs, Electronics, and Kitchen appliances). Reviews are encoded as 5000 dimensional feature vectors of unigrams and bigrams, with binary labels indicating sentiment. We conduct 4 experiments: for each of them, we pick one product as target domain and the rest as source domains. Each source domain has 2000 labeled examples, and the target test set has 3000 to 6000 examples. During training, we randomly sample the same number of unlabeled target examples as the source examples in each mini-batch. We implement both the Hard-Max and Soft-Max methods, and compare them with three baselines: MLPNet, marginalized stacked denoising autoencoders (mSDA) (Chen et al., 2012), and DANN (Ganin et al., 2016). DANN cannot be directly applied in multiple source domains setting. In order to make a comparison, we use two protocols. The first one is to combine all the source domains into a single one and train it using DANN, which we denote as C-DANN. The second protocol is to train multiple DANNs separately, where each one corresponds to a source-target pair. Among all the DANNs, we report the one achieving the best performance on the target domain. We denote this experiment as B-DANN. For fair comparison, all these models are built on the same basic network structure with one input layer (5000 units) and three hidden layers (1000, 500, 100 units).

Table 3.1: Sentiment classification accuracy.

Train/Test	**MLPNet**	**mSDA**	**B-DANN**	**C-DANN**	**MDAN**	
					Hard-Max	Soft-Max
D+E+K/B	0.7655	0.7698	0.7650	0.7789	0.7845	**0.7863**
B+E+K/D	0.7588	0.7861	0.7732	0.7886	0.7797	**0.8065**
B+D+K/E	0.8460	0.8198	0.8381	0.8491	0.8483	**0.8534**
B+D+E/K	0.8545	0.8426	0.8433	**0.8639**	0.8580	0.8626

Results and Analysis We show the accuracy of different methods in Table 3.1. Clearly, Soft-Max significantly outperforms all other methods in most settings. When Kitchen is the target domain, C-DANN performs slightly better than Soft-Max, and all the methods perform close to each other. Hard-Max is typically slightly worse than Soft-Max. This is mainly due to the low data-efficiency of the Hard-Max model (Section 3.4, Eq. 3.5, Eq. 3.6). We observe that with more training iterations, the performance of Hard-Max can be further improved. These results verify the effectiveness of MDAN for multisource domain adaptation. To validate the statistical significance of the results, we also run a non-parametric Wilcoxon signed-ranked test for each task to compare Soft-Max with the other competitors (see more details in appendix). From the statistical test, we see that Soft-Max is convincingly better than other methods.

3.5.2 Digits Datasets

Following the setting in (Ganin et al., 2016), we combine four digits datasets (MNIST, MNIST-M, SVHN, SynthDigits) to build the multisource domain dataset. We take each of MNIST-M, SVHN, and MNIST as target domain in turn, and the rest as sources. Each source domain has $20,000$ labeled images and the target test set has $9,000$ examples.

Baselines We compare Hard-Max and Soft-Max of MDAN with 10 baselines: i). *B-Source*. A basic network trained on each source domain $(20,000$ images) without domain adaptation and tested on the

Table 3.2: Accuracy on digit classification. T: MNIST; M: MNIST-M, S: SVHN, D: SynthDigits.

Method	S+M+D/T	T+S+D/M	M+T+D/S	Method	S+M+D/T	T+S+D/M	M+T+D/S
B-Source	0.964	0.519	0.814	**C-Source**	0.938	0.561	0.771
B-DANN	0.967	0.591	**0.818**	**C-DANN**	0.925	0.651	0.776
B-ADDA	0.968	0.657	0.800	**C-ADDA**	0.927	0.682	0.804
B-MTAE	0.862	0.534	0.703	**C-MTAE**	0.821	0.596	0.701
Hard-Max	0.976	0.663	0.802	**Soft-Max**	**0.979**	**0.687**	0.816
MDAC	0.755	0.563	0.604	**Target**	0.987	0.901	0.898

target domain. Among the three models, we report the one achieves the best performance on the test set. ii). *C-Source*. A basic network trained on a combination of three source domains ($20,000$ images for each) without domain adaptation and tested on the target domain. iii). *B-DANN*. We train DANNs (Ganin et al., 2016) on each source-target domain pair ($20,000$ images for each source) and test it on target. Again, we report the best score among the three. iv). *C-DANN*. We train a single DANN on a combination of three source domains ($20,000$ images for each). v). *B-ADDA*. We train ADDA (Tzeng et al., 2017) on each source-target domain pair ($20,000$ images for each source) and test it on the target domain. We report the best accuracy among the three. vi).*C-ADDA*. We train ADDA on a combination of three source domains ($20,000$ images for each). vii). *B-MTAE*. We train MTAE (Ghifary et al., 2015) on each source-target domain pair ($20,000$ images for each source) and test it on the target domain. We report the best accuracy among the three. viii). *C-MTAE*. We train MTAE on a combination of three source domains ($20,000$ images for each). ix). *MDAC*. MDAC (Zhang et al., 2015a) is a multiple source domain adaptation algorithm that explores causal models to represent the relationship between the features X and class label Y. We directly train MDAC on a combination of three source domains. x). *Target*. It is the basic network trained and tested on the target data. It serves as an upper bound of DA algorithms. All the MDAN and baseline methods are built on the same basic network structure to put them on a equal footing.

Results and Analysis The classification accuracy is shown in Table 3.2. The results show that MDAN outperforms all the baselines in the first two experiments and is comparable with Best-Single-DANN in the third experiment. For the combined sources, MDAN always perform better than the source-only baseline (MDAN vs. Combine-Source). However, a naive combination of different training datasets can sometimes even decrease the performance of the baseline methods. This conclusion comes from three observations: First, directly training DANN on a combination of multiple sources leads to worse results than the source-only baseline (Combine-DANN vs. Combine-Source); Second, The performance of Combine-DANN can be even worse than the Best-Single-DANN (the first and third experiments); Third, directly training DANN on a combination of multiple sources always has lower accuracy compared with our approach (Combine-DANN vs. MDAN). We have similar observations for ADDA and MTAE. Such observations verify that the domain adaptation methods designed for single source lead to suboptimal solutions when applied to multiple sources. It also verifies the necessity and superiority of MDAN for multiple source adaptation. Furthermore, we observe that adaptation to the SVHN dataset (the third experiment) is hard. In this case, increasing the number of source domains does not help. We conjecture this is due to the large dissimilarity between the SVHN data to the others. Surprisingly, using a single domain (best-Single DANN) in this case achieves the best result. This indicates that in domain adaptation the quality of data (how close to the target data) is much more important than the quantity (how many source domains). As a conclusion,

Table 3.3: Counting error statistics. S is the number of source cameras; T is the target camera id.

S	T	MDAN		DANN	FCN	T	MDAN		DANN	FCN
		Hard-Max	Soft-Max				Hard-Max	Soft-Max		
2	A	1.8101	**1.7140**	1.9490	1.9094	B	2.5059	**2.3438**	2.5218	2.6528
3	A	1.3276	**1.2363**	1.3683	1.5545	B	1.9092	**1.8680**	2.0122	2.4319
4	A	1.3868	**1.1965**	1.5520	1.5499	B	**1.7375**	1.8487	2.1856	2.2351
5	A	1.4021	**1.1942**	1.4156	1.7925	B	1.7758	**1.6016**	1.7228	2.0504
6	A	1.4359	**1.2877**	2.0298	1.7505	B	1.5912	**1.4644**	1.5484	2.2832
7	A	1.4381	**1.2984**	1.5426	1.7646	B	1.5989	**1.5126**	1.5397	1.7324

this experiment further demonstrates the effectiveness of MDAN when there are multiple source domains available, where a naive combination of multiple sources using DANN may hurt generalization.

3.5.3 WebCamT Vehicle Counting Dataset

WebCamT is a public dataset for vehicle counting from large-scale city camera videos, which has low resolution (352 × 240), low frame rate (1 frame/second), and high occlusion. It has $60,000$ frames annotated with vehicle bounding box and count, divided into training and testing sets, with $42,200$ and $17,800$ frames, respectively. Here we demonstrate the effectiveness of MDAN to count vehicles from an unlabeled target camera by adapting from multiple labeled source cameras: we select 8 cameras located in different intersections of the city with different scenes, and each has more than $2,000$ labeled images for our evaluations. Among these 8 cameras, we randomly pick two cameras and take each camera as the target camera, with the other 7 cameras as sources. We compute the proxy $\mathcal{A}$-distance (PAD) (Ben-David et al., 2007) between each source camera and the target camera to approximate the divergence between them. We then rank the source cameras by the PAD from low to high and choose the first k cameras to form the k source domains. Thus the proposed methods and baselines can be evaluated on different numbers of sources (from 2 to 7). We implement the Hard-Max and Soft-Max MDAN, based on the basic vehicle counting network FCN (Zhang et al., 2017a). We compare our method with two baselines: FCN (Zhang et al., 2017a), a basic network without domain adaptation, and DANN (Ganin et al., 2016), implemented on top of the same basic network. We record mean absolute error (MAE) between true count and estimated count.

Results and Analysis The counting error of different methods is compared in Table 3.3. The Hard-Max version achieves lower error than DANN and FCN in most settings for both target cameras. The Soft-Max approximation outperforms all the baselines and the Hard-Max in most settings, demonstrating the effectiveness of the smooth and adaptative approximation. The lowest MAE achieved by Soft-Max is 1.1942. Such MAE means that there is only around one vehicle miscount for each frame (the average number of vehicles in one frame is around 20). Fig. 3.2 shows the counting results of Soft-Max for the two target cameras under the 5 source cameras setting. We can see that the proposed method accurately counts the vehicles of each target camera for long time sequences. Does adding more source cameras always help improve the performance on the target camera? To answer this question, we analyze the counting error when we vary the number of source cameras as shown in Fig. 3.3a, where the x-axis refers to number of source cameras and the y-axis includes both the MAE curve on the target camera as well as the PAD distance (bar chart) between the pair of source and target cameras. From the curves, we see the counting error goes down with more source cameras at the beginning, while it goes up when more sources are added

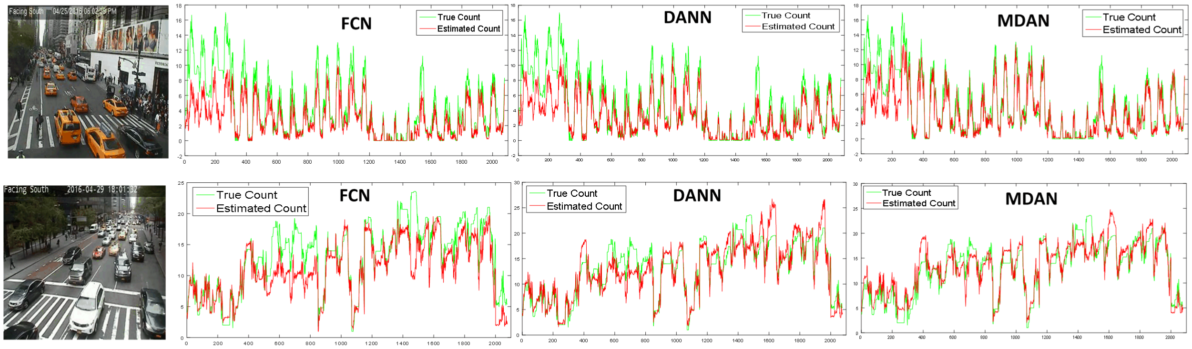


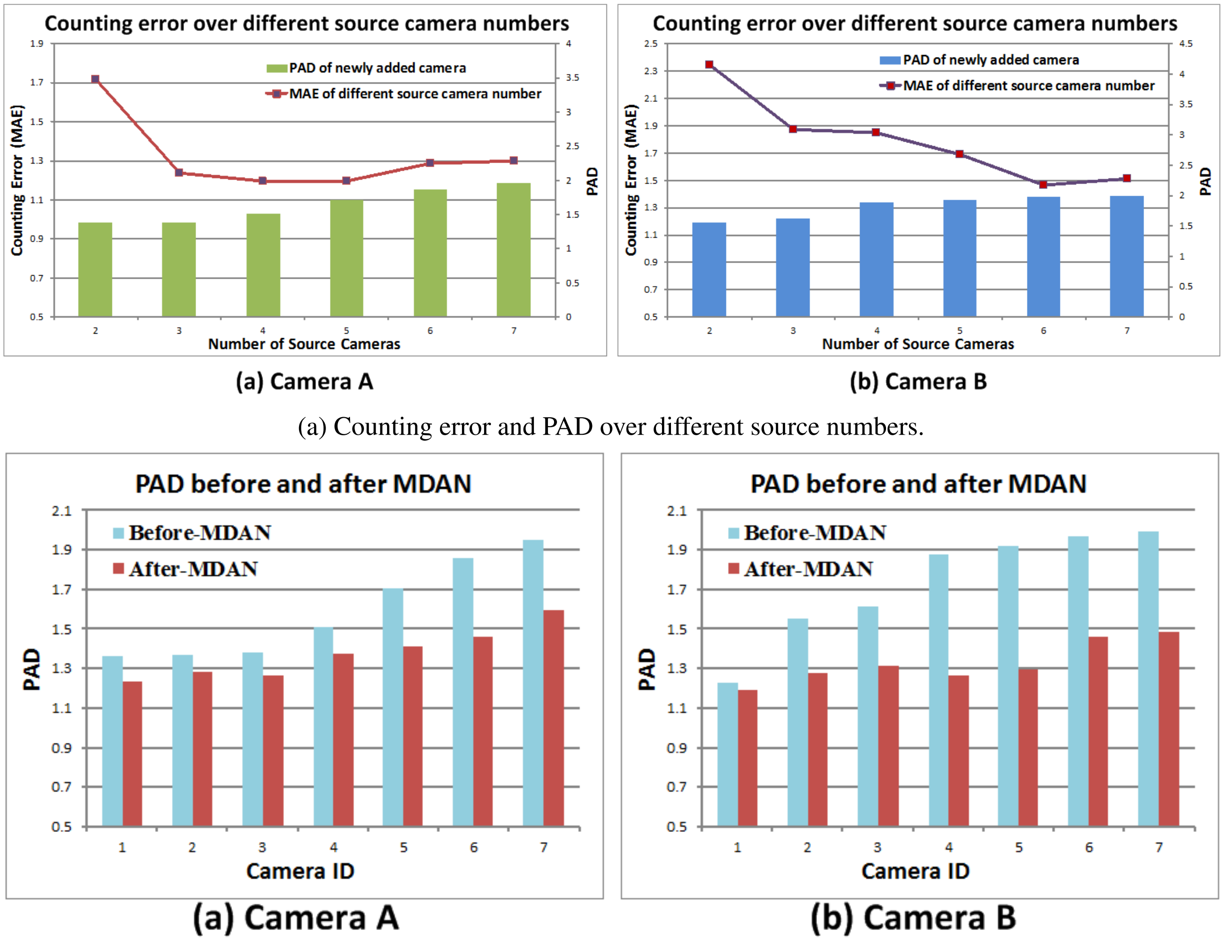
Figure 3.2: Counting results for target camera A (first row) and B (second row). X-frames; Y-Counts.

at the end. This phenomenon shows that the performance on the target domain also depends on the its distance to the added source domain, i.e., it is not always beneficial to naively incorporate more source domains into training. To illustrate this better, we also show the PAD of the newly added camera in the bar chart of Fig. 3.3a. By observing the PAD and the counting error, we see the performance on the target can degrade when the newly added source camera has large divergence from the target camera. To show that MDAN can indeed decrease the divergences between target domain and multiple source domains, in Fig. 3.3b we plot the PAD distances between the target domains and the corresponding source domains. We can see that MDAN consistently decrease the PAD distances between all pairs of target and source domains, for both camera A and camera B. From this experiment we conclude that our proposed MDAN models are effective in multiple source domain adaptation.

3.6 Related Work

A number of adaptation approaches have been studied in recent years. From the theoretical aspect, several theoretical results have been derived in the form of upper bounds on the generalization target error by learning from the source data. A keypoint of the theoretical frameworks is estimating the distribution shift between source and target. Kifer et al. (2004) proposed the $\mathcal{H}$-divergence to measure the similarity between two domains and derived a generalization bound on the target domain using empirical error on the source domain and the $\mathcal{H}$-divergence between the source and the target. This idea has later been extended to multisource domain adaptation (Blitzer et al., 2008) and the corresponding generalization bound has been developed as well. Ben-David et al. (2010) provide a generalization bound for domain adaptation on the target risk which generalizes the standard bound on the source risk. This work formalizes a natural intuition of DA: reducing the two distributions while ensuring a low error on the source domain and justifies many DA algorithms. Based on this work, Mansour et al. (2009a) introduce a new divergence measure: discrepancy distance, whose empirical estimate is based on the Rademacher complexity (Koltchinskii, 2001) (rather than the VC-dim). Other theoretical works have also been studied such as (Mansour and Schain, 2012) that derives the generalization bounds on the target error by taking use of the robustness properties introduced in (Xu and Mannor, 2012). See (Cortes et al., 2008; Mansour et al., 2009c) for more details.

Following the theoretical developments, many DA algorithms have been proposed, such as instance-based methods (Tsuboi et al., 2009); feature-based methods (Becker et al., 2013); and parameter-based methods (Evgeniou and Pontil, 2004). The general approach for domain adaptation starts from algorithms



(a) Counting error and PAD over different source numbers.

(b) PAD distance before and after training MDAN.

Figure 3.3: PAD distance over different source domains along with their changes before and after training MDAN.

that focus on linear hypothesis class (Cortes and Mohri, 2014; Germain et al., 2013). The linear assumption can be relaxed and extended to the non-linear setting using the kernel trick, leading to a reweighting scheme that can be efficiently solved via quadratic programming (Gong et al., 2013). Recently, due to the availability of rich data and powerful computational resources, non-linear representations and hypothesis classes have been increasingly explored (Ajakan et al., 2014; Baktashmotlagh et al., 2013; Chen et al., 2012; Ganin et al., 2016; Glorot et al., 2011). This line of work focuses on building common and robust feature representations among multiple domains using either supervised neural networks (Glorot et al., 2011), or unsupervised pretraining using denoising auto-encoders (Vincent et al., 2008, 2010).

Recent studies have shown that deep neural networks can learn more transferable features for DA (Donahue et al., 2014; Glorot et al., 2011; Yosinski et al., 2014). Bousmalis et al. (2016) develop domain separation networks to extract image representations that are partitioned into two subspaces: domain private component and cross-domain shared component. The partitioned representation is utilized to reconstruct the images from both domains, improving the DA performance. Reference (Long et al., 2015) enables classifier adaptation by learning the residual function with reference to the target classifier. The main-task of this work is limited to the classification problem. Ganin et al. (2016) propose a domain-adversarial neural network to learn the domain indiscriminate but main-task discriminative features. Although these works generally outperform non-deep learning based methods, they only focus on the single-source-single-target DA problem, and much work is rather empirical design without statistical guarantees. Hoffman et al. (2012) present a domain transform mixture model for multisource DA, which is based on non-deep architectures and is difficult to scale up.

Adversarial training techniques that aim to build feature representations that are indistinguishable between source and target domains have been proposed in the last few years (Ajakan et al., 2014; Ganin et al., 2016). Specifically, one of the central ideas is to use neural networks, which are powerful function approximators, to approximate a distance measure known as the $\mathcal{H}$-divergence between two domains (Ben-David et al., 2007, 2010; Kifer et al., 2004). The overall algorithm can be viewed as a zero-sum two-player game: one network tries to learn feature representations that can fool the other network, whose goal is to distinguish representations generated from the source domain between those generated from the target domain. The goal of the algorithm is to find a Nash-equilibrium of the game, or the stationary point of the min-max saddle point problem. Ideally, at such equilibrium state, feature representations from the source domain will share the same distributions as those from the target domain, and, as a result, better generalization on the target domain can be expected by training models using only labeled instances from the source domain.

3.7 Proofs

In this section we provide all the missing proofs in Sec. 3.3. To make it easier to follow, in each part we shall first restate the corresponding theorems in Sec. 3.3 and then provide the proofs.

3.7.1 Proof of Corollary 3.3.1

Corollary 3.3.1 (Worst case classification bound)**.** Let $\mathcal{H}$ be a hypothesis class with $VCdim(\mathcal{H}) = d$. If $\widehat{\mathcal{D}}_T$ and $\{\widehat{\mathcal{D}}_{S_i}\}_{i=1}^k$ are the empirical distributions generated with m *i.i.d.* samples from each domain, then, for $0 < \delta < 1$, with probability at least $1 - \delta$, for all $h \in \mathcal{H}$, we have:

$$\varepsilon_T(h) \leq \max_{i \in [k]} \left\{ \widehat{\varepsilon}_{S_i}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T; \widehat{\mathcal{D}}_{S_i}) + \lambda_i \right\} + O\left(\sqrt{\frac{1}{m}\left(\log\frac{k}{\delta} + d\log\frac{m}{d} \right)} \right), \tag{3.2}$$

where λ_i is the combined risk of the optimal hypothesis on domains S_i and T.

Proof. For each one of the k source domain, from Thm. 4.2.1, for $\forall\delta > 0$, w.p.b $\geq 1 - \delta/k$, we have the following inequality hold:

$$\varepsilon_T(h) \leq \widehat{\varepsilon}_{S_i}(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{S_i}, \widehat{\mathcal{D}}_T) + \lambda_i + O\left(\sqrt{\frac{d\log(m/d) + \log(k/\delta)}{m}} \right)$$

Using a union bound argument, we have:

$$
\begin{aligned}
&\Pr\left(\varepsilon_T(h) > \max_{i\in[k]}\left\{\widehat{\varepsilon}_{S_i}(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T;\widehat{\mathcal{D}}_{S_i}) + \lambda_i\right\} + O\left(\sqrt{\frac{1}{m}\left(\log\frac{k}{\delta} + d\log\frac{m}{d}\right)}\right)\right)\\
\leq &\Pr\left(\bigvee_{i\in[k]} \varepsilon_T(h) > \widehat{\varepsilon}_{S_i}(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T;\widehat{\mathcal{D}}_{S_i}) + \lambda_i + O\left(\sqrt{\frac{1}{m}\left(\log\frac{k}{\delta} + d\log\frac{m}{d}\right)}\right)\right)\\
\leq &\sum_{i\in[k]}\Pr\left(\varepsilon_T(h) > \widehat{\varepsilon}_{S_i}(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T;\widehat{\mathcal{D}}_{S_i}) + \lambda_i + O\left(\sqrt{\frac{1}{m}\left(\log\frac{k}{\delta} + d\log\frac{m}{d}\right)}\right)\right)\\
\leq &\sum_{i\in[k]}\delta/k = \delta
\end{aligned}
$$

which completes the proof. ■

3.7.2 Proof of Theorem 3.3.1

Theorem 3.3.1 (Average case classification bound)**.** Let $\mathcal{H}$ be a hypothesis class with $VCdim(\mathcal{H}) = d$. If $\{\widehat{\mathcal{D}}_{S_i}\}_{i=1}^k$ are the empirical distributions generated with m $i.i.d.$ samples from each domain, and $\widehat{\mathcal{D}}_T$ is the empirical distribution on the target domain generated from mk samples without labels, then, $\forall\alpha \in \mathbb{R}_+^k, \sum_{i\in[k]}\alpha_i = 1$, and for $0 < \delta < 1$, w.p.b. at least $1-\delta$, for all $h \in \mathcal{H}$, we have:

$$
\varepsilon_T(h) \leq \sum_{i\in[k]}\alpha_i\left(\widehat{\varepsilon}_{S_i}(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_T;\widehat{\mathcal{D}}_{S_i})\right) + \lambda_\alpha + O\left(\sqrt{\frac{1}{km}\left(\log\frac{1}{\delta} + d\log\frac{km}{d}\right)}\right), \tag{3.3}
$$

where λ_α is the risk of the optimal hypothesis on the mixture source domain $\sum_{i\in[k]}\alpha_i S_i$ and T.

Proof. Consider a mixture distribution of the k source domains where the mixture weight is given by α. Denote it as $\mathcal{D}_{\tilde{S}} := \sum_{i\in[k]}\alpha_i\mathcal{D}_{S_i}$. Let $\tilde{S}$ be the combined samples from k domains, then equivalently $\tilde{S}$ can be seen as a sample of size km sampled $i.i.d.$ from $\mathcal{D}_{\tilde{S}}$. Apply Thm. 3.2.1 using $\mathcal{D}_T$ as the target domain and $\mathcal{D}_{\tilde{S}}$ as the source domain, we know that for $0 < \delta < 1$, w.p.b. at least $1-\delta$,

$$
\varepsilon_T(h) \leq \widehat{\varepsilon}_{\tilde{S}}(h) + \frac{1}{2}d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{\tilde{S}}, \widehat{\mathcal{D}}_T) + \lambda_\alpha + O\left(\sqrt{\frac{d\log(km/d) + \log(1/\delta)}{km}}\right). \tag{3.8}
$$

On the other hand, for $\forall h \in \mathcal{H}$, we have:

$$
\widehat{\varepsilon}_{\tilde{S}}(h) = \sum_{i\in[k]}\alpha_i\widehat{\varepsilon}_{S_i}(h),
$$

and we can upper bound $d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{\tilde{S}}, \widehat{\mathcal{D}}_T)$ as follows:

$$\begin{aligned}
d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{\tilde{S}}, \widehat{\mathcal{D}}_T) &= 2 \sup_{A \in \mathcal{A}_{\mathcal{H}\Delta\mathcal{H}}} |\Pr_{\widehat{\mathcal{D}}_{\tilde{S}}}(A) - \Pr_{\widehat{\mathcal{D}}_T}(A)| \\
&= 2 \sup_{A \in \mathcal{A}_{\mathcal{H}\Delta\mathcal{H}}} |\sum_{i\in[k]} \alpha_i (\Pr_{\widehat{\mathcal{D}}_{S_i}}(A) - \Pr_{\widehat{\mathcal{D}}_T}(A))| \\
&\leq 2 \sup_{A \in \mathcal{A}_{\mathcal{H}\Delta\mathcal{H}}} \sum_{i\in[k]} \alpha_i | \Pr_{\widehat{\mathcal{D}}_{S_i}}(A) - \Pr_{\widehat{\mathcal{D}}_T}(A)| \\
&\leq 2 \sum_{i\in[k]} \alpha_i \sup_{A \in \mathcal{A}_{\mathcal{H}\Delta\mathcal{H}}} | \Pr_{\widehat{\mathcal{D}}_{S_i}}(A) - \Pr_{\widehat{\mathcal{D}}_T}(A)| \\
&= \sum_{i\in[k]} \alpha_i d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{S_i}, \widehat{\mathcal{D}}_T),
\end{aligned}$$

where the first inequality is due to the triangle inequality and the second inequality is by the sub-additivity of the sup function. Replace $\widehat{\varepsilon}_{\tilde{S}}(h)$ with $\sum_{i\in[k]} \alpha_i \widehat{\varepsilon}_{S_i}(h)$ and upper bound $d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{\tilde{S}}, \widehat{\mathcal{D}}_T)$ by $\sum_{i\in[k]} \alpha_i d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_{S_i}, \widehat{\mathcal{D}}_T)$ in (3.8) completes the proof. ■

3.7.3 Proof of Theorem 3.3.2

Before we give a full proof, we first describe the proof strategy at a high level. Roughly, the proof contains three parts. The first part contains a reduction from regression to classification by relating the pseudo-dimension of the hypothesis class $\mathcal{H}$ and its corresponding threshold binary classifiers. The second part uses $\mathcal{H}$-divergence to relate the source and target domains when they differ. The last part uses standard generalization analysis with pseudo-dimension for regression, when the source and target domains coincide.

First Part of the Proof

To begin with, let $\mathcal{H} = \{h : \mathcal{X} \to [0,1]\}$ be a set of bounded real-valued functions from the input space $\mathcal{X}$ to $[0,1]$. We use $Pdim(\mathcal{H})$ to denote the pseudo-dimension of $\mathcal{H}$, and let $Pdim(\mathcal{H}) = d$. We first prove the following lemma that will be used in proving the main theorem:

Lemma 3.7.1. For $h, h' \in \mathcal{H} := \{h : \mathcal{X} \to [0,1]\}$, where $Pdim(\mathcal{H}) = d$, and for any distribution $\mathcal{D}_S$, $\mathcal{D}_T$ over $\mathcal{X}$,

$$|\varepsilon_S(h,h') - \varepsilon_T(h,h')| \leq \frac{1}{2} d_{\bar{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T)$$

where $\bar{\mathcal{H}} := \{\mathbb{I}_{|h(x)-h'(x)|>t} : h, h' \in \mathcal{H}, 0 \leq t \leq 1\}$.

Proof. By definition, for $\forall h, h' \in \mathcal{H}$, we have:

$$\begin{aligned}
|\varepsilon_S(h,h') - \varepsilon_T(h,h')| &\leq \sup_{h,h' \in \mathcal{H}} |\varepsilon_S(h,h') - \varepsilon_T(h,h')| \\
&= \sup_{h,h' \in \mathcal{H}} \left|\mathbb{E}_{\mathbf{x}\sim S}[|h(\mathbf{x}) - h'(\mathbf{x})|] - \mathbb{E}_{\mathbf{x}\sim T}[|h(\mathbf{x}) - h'(\mathbf{x})|]\right|. \qquad (3.9)
\end{aligned}$$

Since $||h||_\infty \leq 1, \forall h \in \mathcal{H}$, then $0 \leq |h(\mathbf{x}) - h'(\mathbf{x})| \leq 1$, $\forall \mathbf{x} \in \mathcal{X}, h, h' \in \mathcal{H}$. We now use Fubini's theorem to bound $\left|\mathbb{E}_{\mathbf{x}\sim S}[|h(\mathbf{x})-h'(\mathbf{x})|]-\mathbb{E}_{\mathbf{x}\sim T}[|h(\mathbf{x})-h'(\mathbf{x})|]\right|$:

$$\begin{aligned}\left|\mathbb{E}_{\mathbf{x}\sim S}[|h(\mathbf{x})-h'(\mathbf{x})|]-\right.&\left.\mathbb{E}_{\mathbf{x}\sim T}[|h(\mathbf{x})-h'(\mathbf{x})|]\right| \\ &= \left|\int_0^1 \left(\Pr_S(|h(\mathbf{x})-h'(\mathbf{x})|>t)-\Pr_T(|h(\mathbf{x})-h'(\mathbf{x})|>t)\right)\,dt\right| \\ &\le \int_0^1 \left|\Pr_S(|h(\mathbf{x})-h'(\mathbf{x})|>t)-\Pr_T(|h(\mathbf{x})-h'(\mathbf{x})|>t)\right|\,dt \\ &\le \sup_{t\in[0,1]}\left|\Pr_S(|h(\mathbf{x})-h'(\mathbf{x})|>t)-\Pr_T(|h(\mathbf{x})-h'(\mathbf{x})|>t)\right|.\end{aligned}$$

Now in view of (4.6) and $\bar{\mathcal{H}}$, we have:

$$\begin{aligned}&\sup_{h,h'\in\mathcal{H}}\sup_{t\in[0,1]}\left|\Pr_S(|h(\mathbf{x})-h'(\mathbf{x})|>t)-\Pr_T(|h(\mathbf{x})-h'(\mathbf{x})|>t)\right| \\ =\;& \sup_{\bar{h}\in\bar{\mathcal{H}}}|\Pr_S(\bar{h}(\mathbf{x})=1)-\Pr_T(\bar{h}(\mathbf{x})=1)| \\ =\;& \sup_{A\in\mathcal{A}_{\bar{\mathcal{H}}}}|\Pr_S(A)-\Pr_T(A)| \\ =\;& \frac{1}{2}d_{\bar{\mathcal{H}}}(\mathcal{D}_S,\mathcal{D}_T).\end{aligned}$$

Combining all the inequalities above finishes the proof. ■

Next we bound $Pdim(|\mathcal{H}-\mathcal{H}|)$:

Lemma 3.7.2. If $Pdim(\mathcal{H})=d$, then $Pdim(|\mathcal{H}-\mathcal{H}|)\le 2d$.

Proof. By the definition of pseudo-dimension, we immediately have $VCdim(\bar{\mathcal{H}})=Pdim(|\mathcal{H}-\mathcal{H}|)$. Next observe that each function $g\in\bar{\mathcal{H}}$ could be represented as a two layer linear threshold neural network, with two hidden units, one bias input unit and one output unit. Specifically, since

$$|h(\mathbf{x})-h'(\mathbf{x})|>t \iff \max\{h(\mathbf{x})-h'(\mathbf{x})-t, h'(\mathbf{x})-h(\mathbf{x})-t\}>0,$$

which is also equivalent to

$$\operatorname{sgn}(h(\mathbf{x})-h'(\mathbf{x})-t)+\operatorname{sgn}(h'(\mathbf{x})-h(\mathbf{x})-t)>0. \tag{3.10}$$

The above expression can then be implemented by a two layer one output linear threshold neural network. Hence from (Anthony and Bartlett, 2009, Chapter 6, 8), the VC dimension of $\bar{\mathcal{H}}$ is at most twice of the pseudo-dimension of $\mathcal{H}$, completing the proof. ■

Second Part of the Proof

One technical lemma we will frequently use is the triangular inequality w.r.t. $\varepsilon_{\mathcal{D}}(h)$, $\forall h\in\mathcal{H}$:

Lemma 3.7.3. For any hypothesis class $\mathcal{H}$ and any distribution $\mathcal{D}$ on $\mathcal{X}$, the following triangular inequality holds:

$$\forall h,h',f\in\mathcal{H},\quad \varepsilon_{\mathcal{D}}(h,h')\le\varepsilon_{\mathcal{D}}(h,f)+\varepsilon_{\mathcal{D}}(f,h'),$$

Proof.

$$\varepsilon_{\mathcal{D}}(h,h') = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[|h(\mathbf{x}) - h'(\mathbf{x})|] \le \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[|h(\mathbf{x}) - f(\mathbf{x})| + |f(\mathbf{x}) - f(\mathbf{x})|] = \varepsilon_{\mathcal{D}}(h,f) + \varepsilon_{\mathcal{D}}(f,h').$$

■

Now we prove the following lemma in the regression setting when there is only one source domain and one target domain.

Lemma 3.7.4. Let $\mathcal{H}$ be a set of real-valued function from $\mathcal{X}$ to $[0,1]$, and $\mathcal{D}_S$, $\mathcal{D}_T$ be the source and target distributions, respectively. Define $\bar{\mathcal{H}} := \{\mathbb{I}_{|h(x)-h'(x)|>t} : h,h' \in \mathcal{H}, 0 \le t \le 1\}$. Then $\forall h \in \mathcal{H}$, the following inequality holds:

$$\varepsilon_T(h) \le \varepsilon_S(h) + \frac{1}{2}d_{\bar{\mathcal{H}}}(\mathcal{D}_T;\mathcal{D}_S) + \lambda,$$

where $\lambda := \inf_{h'\in\mathcal{H}} \varepsilon_S(h') + \varepsilon_T(h')$.

Proof. Let $h^* := \arg\min_{h'\in\mathcal{H}} \varepsilon_S(h') + \varepsilon_T(h')$. For $\forall h \in \mathcal{H}$:

$$\begin{aligned}
\varepsilon_T(h) &\le \varepsilon_T(h^*) + \varepsilon_T(h,h^*) \\
&= \varepsilon_T(h^*) + \varepsilon_T(h,h^*) - \varepsilon_S(h,h^*) + \varepsilon_S(h,h^*) \\
&\le \varepsilon_T(h^*) + |\varepsilon_T(h,h^*) - \varepsilon_S(h,h^*)| + \varepsilon_S(h,h^*) \\
&\le \varepsilon_T(h^*) + \varepsilon_S(h,h^*) + \frac{1}{2}d_{\bar{\mathcal{H}}}(\mathcal{D}_T,\mathcal{D}_S) \\
&\le \varepsilon_T(h^*) + \varepsilon_S(h) + \varepsilon_S(h^*) + \frac{1}{2}d_{\bar{\mathcal{H}}}(\mathcal{D}_T,\mathcal{D}_S) \\
&= \varepsilon_S(h) + \frac{1}{2}d_{\bar{\mathcal{H}}}(\mathcal{D}_T;\mathcal{D}_S) + \lambda.
\end{aligned}$$

The first and fourth inequalities are by the triangle inequality, and the third one is from Lemma. 3.7.1. ■

Third Part of the Proof

In this part of the proof, we use concentration inequalities to bound the source domain error $\varepsilon_S(h)$ and the divergence $d_{\bar{\mathcal{H}}}(\mathcal{D}_T;\mathcal{D}_S)$ in Lemma 3.7.4.

We first introduce the following generalization theorem in the regression setting when the source and target distributions are the same:

Lemma 3.7.5 (Thm. 10.6, (Mohri et al., 2012))**.** Let $\mathcal{H}$ be a family of real-valued functions from $\mathcal{X}$ to $[0,1]$. Assume that $Pdim(\mathcal{H}) = d$. Then, for $\forall\delta > 0$, w.p.b. at least $1-\delta$ over the choice of a sample of size m, the following inequality holds for all $h \in \mathcal{H}$:

$$\varepsilon(h) \le \widehat{\varepsilon}(h) + O\left(\sqrt{\frac{1}{m}\left(\log\frac{1}{\delta} + d\log\frac{m}{d}\right)}\right).$$

The next lemma bounds the $\mathcal{H}$-divergence between the population distribution and its corresponding empirical distribution:

Lemma 3.7.6. Let $\mathcal{D}_S$ and $\mathcal{D}_T$ be the source and target distribution over $\mathcal{X}$, respectively. Let $\mathcal{H}$ be a class of real-valued functions from $\mathcal{X}$ to $[0,1]$, with $Pdim(\mathcal{H}) = d$. Define $\bar{\mathcal{H}} := \{\mathbb{I}_{|h(x)-h'(x)|>t} : h, h' \in \mathcal{H}, 0 \le t \le 1\}$. If S and T are the empirical distributions of $\mathcal{D}_S$ and $\mathcal{D}_T$ generated with m *i.i.d.* samples from each domain, then, for $0 < \delta < 1$, w.p.b. at least $1 - \delta$, we have:

$$d_{\bar{\mathcal{H}}}(\mathcal{D}_S; \mathcal{D}_T) \le d_{\bar{\mathcal{H}}}(S; T) + O\left(\sqrt{\frac{1}{m}\left(\log\frac{1}{\delta} + d\log\frac{m}{d}\right)}\right).$$

Proof. By the definition of pseudo-dimension, we have $VCdim(\bar{\mathcal{H}}) = Pdim(|\mathcal{H} - \mathcal{H}|)$. Now from Lemma 3.7.2, $Pdim(|\mathcal{H} - \mathcal{H}|) \le 2Pdim(\mathcal{H}) = 2d$, so $VCdim(\bar{\mathcal{H}}) \le 2d$. The lemma then follows from Ben-David et al. (2010, Lemma 1) using standard concentration inequality. ■

Proof of the Generalization Bound under Regression Setting

We prove the following generalization bound for regression problem when there is only one source domain and one target domain. The extension to multiple source domains follows exactly as the proof of Thm. 3.3.1, hence we omit here.

Theorem 3.7.1. Let $\mathcal{H}$ be a set of real-valued function from $\mathcal{X}$ to $[0,1]$ with $Pdim(\mathcal{H}) = d$. Let $\widehat{\mathcal{D}}_S$ ($\widehat{\mathcal{D}}_T$) be the empirical distribution induced by sample of size m drawn from $\mathcal{D}_S$ ($\mathcal{D}_T$). Then w.p.b. at least $1 - \delta$, $\forall h \in \mathcal{H}$,

$$\varepsilon_T(h) \le \widehat{\varepsilon}_S(h) + \frac{1}{2}d_{\bar{\mathcal{H}}}(\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T) + \lambda + O\left(\sqrt{\frac{d\log(m/d) + \log(1/\delta)}{m}}\right) \tag{3.11}$$

where $\lambda = \inf_{h' \in \mathcal{H}} \varepsilon_S(h') + \varepsilon_T(h')$ and $\bar{\mathcal{H}} := \{\mathbb{I}_{|h(x)-h'(x)|>t} : h, h' \in \mathcal{H}, 0 \le t \le 1\}$.

Proof. Combine Lemma 3.7.4, Lemma 3.7.5 and Lemma 3.7.6 using union bound finishes the proof. ■

3.8 Conclusion

In this chapter we theoretically analyze generalization bounds for DA under the setting of multiple source domains with labeled instances and one target domain with unlabeled instances. Specifically, we propose average case generalization bounds for both classification and regression problems. The new bounds have interesting interpretations and the one for classification reduces to an existing bound when there is only one source domain. Following our theoretical results, we propose two MDAN to learn feature representations that are invariant under multiple domain shifts while at the same time being discriminative for the learning task. Both hard and soft versions of MDAN are generalizations of the popular DANN to the case when multiple source domains are available. Empirically, MDAN outperforms the state-of-the-art DA methods on three real-world datasets, including a sentiment analysis task, a digit classification task, and a visual vehicle counting task, demonstrating its effectiveness in multisource domain adaptation for both classification and regression problems.

Appendix

In this section we describe more details about the datasets and the experimental settings. We extensively evaluate the proposed methods on three datasets:

1. We first evaluate our methods on Amazon Reviews dataset (Chen et al., 2012) for sentiment analysis.
2. We evaluate the proposed methods on the digits classification datasets including MNIST (LeCun et al., 1998b), MNIST-M (Ganin et al., 2016), SVHN (Netzer et al., 2011), and SynthDigits (Ganin et al., 2016).
3. We further evaluate the proposed methods on the public dataset WebCamT (Zhang et al., 2017a) for vehicle counting. It contains 60,000 labeled images from 12 city cameras with different distributions. Due to the substantial difference between these datasets and their corresponding learning tasks, we will introduce more detailed dataset description, network architecture, and training parameters for each dataset respectively in the following subsections.

3.A Details on Amazon Reviews Evaluation

Amazon reviews dataset includes four domains, each one composed of reviews on a specific kind of product (Books, DVDs, Electronics, and Kitchen appliances). Reviews are encoded as 5000 dimensional feature vectors of unigrams and bigrams. The labels are binary: 0 if the product is ranked up to 3 stars, and 1 if the product is ranked 4 or 5 stars.

We take one product domain as target and the other three as source domains. Each source domain has 2000 labeled examples and the target test set has 3000 to 6000 examples. We implement the Hard-Max and Soft-Max methods based on a basic network with one input layer (5000 units) and three hidden layers (1000, 500, 100 units). The network is trained for 50 epochs with dropout rate 0.7. We compare Hard-Max and Soft-Max with three baselines: *Baseline 1: MLPNet*. It is the basic network of our methods (one input layer and three hidden layers), trained for 50 epochs with dropout rate 0.01. *Baseline 2: Marginalized Stacked Denoising Autoencoders (mSDA)* (Chen et al., 2012). It takes the unlabeled parts of both source and target samples to learn a feature map from input space to a new representation space. As a denoising autoencoder algorithm, it finds a feature representation from which one can (approximately) reconstruct the original features of an example from its noisy counterpart. *Baseline 3: DANN*. We implement DANN based on the algorithm described in (Ganin et al., 2016) with the same basic network as our methods. Hyper parameters of the proposed and baseline methods are selected by cross validation. Table 3.A.1 summarizes the network architecture and some hyper parameters.

To validate the statistical significance of the results, we run a non-parametric Wilcoxon signed-ranked test for each task to compare Soft-Max with the other competitors, as shown in Table 3.A.2. Each cell corresponds to the p-value of a Wilcoxon test between Soft-Max and one of the other methods, under the null hypothesis that the two paired samples have the same mean. From these p-values, we see Soft-Max is convincingly better than other methods.

Table 3.A.1: Network parameters for proposed and baseline methods

Method	Input layer	Hidden layers	Epochs	Dropout	Domains	Adaptation weight	fl
MLPNet	5000	(1000, 500, 100)	50	0.01	N/A	N/A	N/A
DANN	5000	(1000, 500, 100)	50	0.01	1	0.01	N/A
MDAN	5000	(1000, 500, 100)	50	0.7	3	0.1	10

Table 3.A.2: p-values under Wilcoxon test.

	MLPNet	**mSDA**	**Best-Single-DANN**	**Combine-DANN**	**Hard-Max**
	Soft-Max	**Soft-Max**	**Soft-Max**	**Soft-Max**	**Soft-Max**
B	0.550	0.101	0.521	0.013	0.946
D	0.000	0.072	0.000	0.051	0.000
E	0.066	0.000	0.097	0.150	0.022
K	0.306	0.001	0.001	0.239	0.008

3.B Details on Digit Datasets Evaluation

We evaluate the proposed methods on the digits classification problem. Following the experiments in (Ganin et al., 2016), we combine four popular digits datasets-MNIST, MNIST-M, SVHN, and SynthDigits to build the multi-source domain dataset. MNIST is a handwritten digits database with $60,000$ training examples, and $10,000$ testing examples. The digits have been size-normalized and centered in a 28×28 image. MNIST-M is generated by blending digits from the original MNIST set over patches randomly extracted from color photos from BSDS500 (Arbelaez et al., 2011; Ganin et al., 2016). It has $59,001$ training images and $9,001$ testing images with 32×32 resolution. An output sample is produced by taking a patch from a photo and inverting its pixels at positions corresponding to the pixels of a digit. For DA problems, this domain is quite distinct from MNIST, for the background and the strokes are no longer constant. SVHN is a real-world house number dataset with $73,257$ training images and $26,032$ testing images. It can be seen as similar to MNIST, but comes from a significantly harder, unsolved, real world problem. SynthDigits consists of $500;000$ digit images generated by Ganin et al. (2016) from WindowsTM fonts by varying the text, positioning, orientation, background and stroke colors, and the amount of blur. The degrees of variation were chosen to simulate SVHN, but the two datasets are still rather distinct, with the biggest difference being the structured clutter in the background of SVHN images.

We take MNIST-M, SVHN, and MNIST as target domain in turn, and the remaining three as sources. We implement the Hard-Max and Soft-Max versions according to Alg. 1 based on a basic network, as shown in Fig. 3.B.1. The baseline methods are also built on the same basic network structure to put them on a equal footing. The network structure and parameters of MDAN are illustrated in Fig. 3.B.1. The learning rate is initialized by 0.01 and adjusted by the first and second order momentum in the training process. The domain adaptation parameter of MDAN is selected by cross validation. In each mini-batch of MDAN training process, we randomly sample the same number of unlabeled target images as the number of the source images.

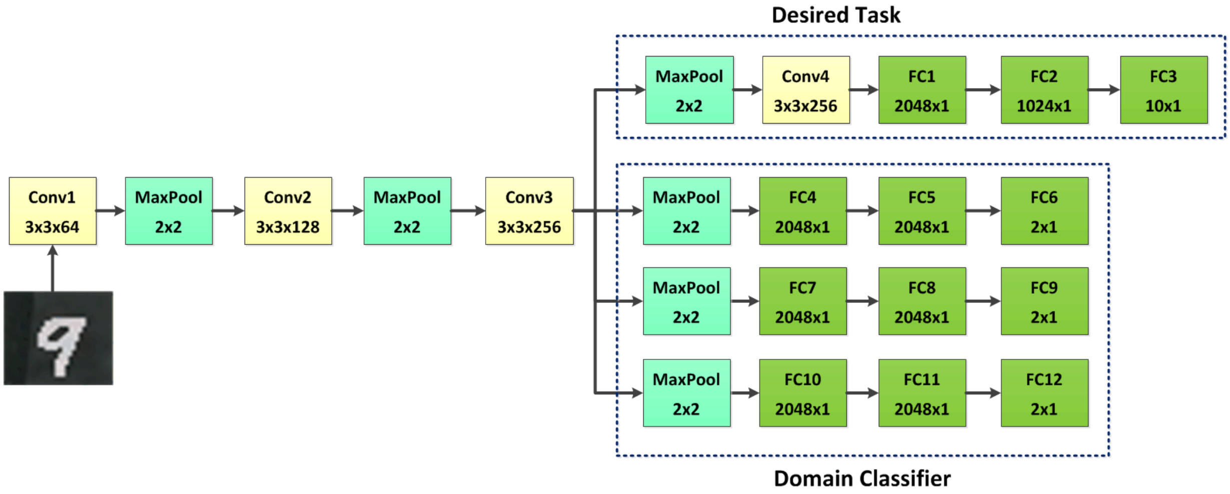


Figure 3.B.1: MDAN network architecture for digit classification

3.C Details on WebCamT Vehicle Counting

WebCamT is a public dataset for large-scale city camera videos, which have low resolution (352×240), low frame rate (1 frame/second), and high occlusion. WebCamT has $60,000$ frames annotated with rich information: bounding box, vehicle type, vehicle orientation, vehicle count, vehicle re-identification, and weather condition. The dataset is divided into training and testing sets, with 42,200 and 17,800 frames, respectively, covering multiple cameras and different weather conditions. WebCamT is an appropriate dataset to evaluate domain adaptation methods, for it covers multiple city cameras and each camera is located in different intersection of the city with different perspectives and scenes. Thus, each camera data has different distribution from others. The dataset is quite challenging and in high demand of domain adaptation solutions, as it has $6,000,000$ unlabeled images from 200 cameras with only $60,000$ labeled images from 12 cameras. The experiments on WebCamT provide an interesting application of our proposed MDAN: when dealing with spatially and temporally large-scale dataset with much variations, it is prohibitively expensive and time-consuming to label large amount of instances covering all the variations. As a result, only a limited portion of the dataset can be annotated, which can not cover all the data domains in the dataset. MDAN provide an effective solution for this kind of application by adapting the deep model from multiple source domains to the unlabeled target domain.

We evaluate the proposed methods on different numbers of source cameras. Each source camera provides 2000 labeled images for training and the test set has 2000 images from the target camera. In each mini-batch, we randomly sample the same number of unlabeled target images as the source images. We implement the Hard-Max and Soft-Max version of MDAN according to Alg. 1, based on the basic vehicle counting network FCN described in (Zhang et al., 2017a). Please refer to (Zhang et al., 2017a) for detailed network architecture and parameters. The learning rate is initialized by 0.01 and adjusted by the first and second order momentum in the training process. The domain adaptation parameter is selected by cross validation. We compare our method with two baselines: *Baseline 1: FCN*. It is our basic network without domain adaptation as introduced in work (Zhang et al., 2017a). *Baseline 2: DANN*. We implement DANN on top of the same basic network following the algorithm introduced in work (Ganin et al., 2016).

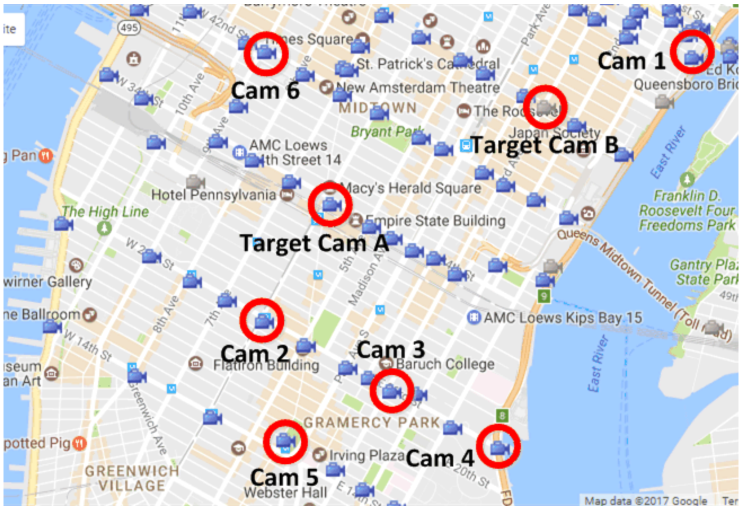


Figure 3.C.1: Locations of the source&target camera map.

Chapter 4

Learning Invariant Representations for Domain Adaptation

Due to the ability of deep neural nets to learn rich representations, recent advances in unsupervised domain adaptation have focused on learning domain-invariant features that achieve a small error on the source domain. The hope is that the learnt representation, together with the hypothesis learnt from the source domain, can generalize to the target domain. In this chapter, we first construct a simple counterexample showing that, contrary to common belief, the above conditions are not sufficient to guarantee successful domain adaptation. In particular, the counterexample exhibits *conditional shift*: the class-conditional distributions of input features change between source and target domains. To give a sufficient condition for domain adaptation, we propose a natural and interpretable generalization upper bound that explicitly takes into account the aforementioned shift. Moreover, we shed new light on the problem by proving an information-theoretic lower bound on the joint error of *any* domain adaptation method that attempts to learn invariant representations. Our result characterizes a fundamental tradeoff between learning invariant representations and achieving small joint error on both domains when the marginal label distributions differ from source to target. Finally, we conduct experiments on real-world datasets that corroborate our theoretical findings.

4.1 Introduction

The recent successes of supervised deep learning methods have been partially attributed to rich datasets and increasing computational power. However, in many critical applications, e.g., self-driving cars or personal healthcare, it is often prohibitively expensive and time-consuming to collect large-scale supervised training data. Unsupervised domain adaptation (DA) focuses on such limitations by trying to transfer knowledge from a labeled source domain to an unlabeled target domain, and a large body of work tries to achieve this by exploring domain-invariant structures and representations to bridge the gap. Theoretical results (Ben-David et al., 2010; Mansour and Schain, 2012; Mansour et al., 2009a) and algorithms (Adel et al., 2017; Ajakan et al., 2014; Becker et al., 2013; Glorot et al., 2011; Pei et al., 2018) under this setting are abundant.

Due to the ability of deep neural nets to learn rich feature representations, recent advances in domain adaptation have focused on using these networks to learn *invariant representations*, i.e., intermediate features whose distribution is the same in source and target domains, while at the same time achieving small error on the source domain. The hope is that the learnt intermediate representation, together with the hypothesis learnt using labeled data from the source domain, can generalize to the target domain. Nevertheless, from a theoretical standpoint, it is not at all clear whether aligned representations and small source error are sufficient to guarantee good generalization on the target domain. In fact, despite being successfully applied in various applications (Hoffman et al., 2017b; Zhang et al., 2017b), it has also been reported that such methods fail to generalize in certain closely related source/target pairs, e.g., digit classification from MNIST to SVHN (Ganin et al., 2016).

Given the wide application of domain adaptation methods based on learning invariant representations, we attempt in this chapter to answer the following important and intriguing question:

> *Is finding invariant representations while at the same time achieving a small source error sufficient to guarantee a small target error? If not, under what conditions is it?*

Contrary to common belief, we give a negative answer to the above question by constructing a simple example showing that these two conditions are not sufficient to guarantee target generalization, even in the case of perfectly aligned representations between the source and target domains. In fact, our example shows that the objective of learning invariant representations while minimizing the source error can actually be hurtful, in the sense that the better the objective, the larger the target error. At a colloquial level, this happens because learning invariant representations can break the originally favorable underlying problem structure, i.e., close labeling functions and conditional distributions. To understand when such methods work, we propose a generalization upper bound as a sufficient condition that explicitly takes into account the conditional shift between source and target domains. The proposed upper bound admits a natural interpretation and decomposition in domain adaptation; we show that it is tighter than existing results in certain cases.

Simultaneously, to understand what the necessary conditions for representation based approaches to work are, we prove an information-theoretic lower bound on the joint error of both domains for *any* algorithm based on learning invariant representations. Our result complements the above upper bound and also extends the constructed example to more general settings. The lower bound sheds new light on this problem by characterizing a fundamental tradeoff between learning invariant representations and achieving small joint error on both domains when the marginal label distributions differ from source to target. Our lower bound directly implies that minimizing source error while achieving invariant representation will only increase the target error. We conduct experiments on real-world datasets that corroborate this theoretical implication. Together with the generalization upper bound, our results suggest that adaptation should be designed to align the label distribution as well when learning an invariant representation (c.f. Sec. 4.4.3).

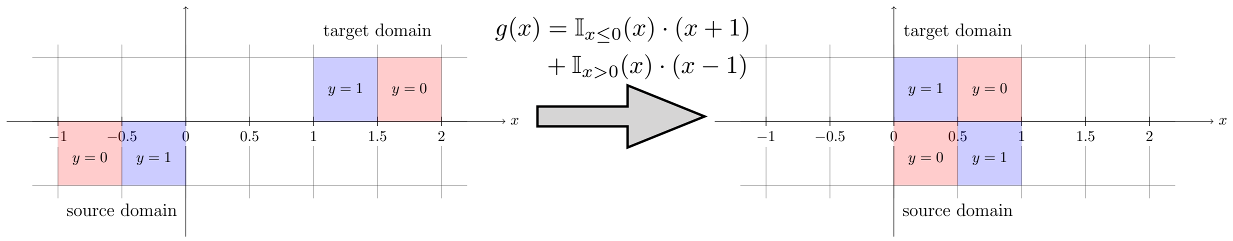


Figure 4.1.1: A counterexample where invariant representations lead to large joint error on source and target domains. Before transformation of $g(\cdot)$, $h^*(x) = 1$ iff $x \in (-1/2, 3/2)$ achieves perfect classification on both domains. After transformation, source and target distributions are perfectly aligned, but no hypothesis can achieve a small joint error.

We believe these insights will be helpful to guide the future design of domain adaptation and representation learning algorithms.

4.2 Preliminaries

We first review a theoretical model for domain adaptation (DA) (Ben-David et al., 2007, 2010; Blitzer et al., 2008; Kifer et al., 2004).

4.2.1 Problem Setup

We consider the unsupervised domain adaptation problem where the learning algorithm has access to a set of n labeled points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \in (\mathcal{X}\times\mathcal{Y})^n$ sampled i.i.d. from the source domain and a set of unlabeled points $\{\mathbf{x}_j\}_{j=1}^m \in \mathcal{X}^m$ sampled i.i.d. from the target domain. At a colloquial level, the goal of an unsupervised domain adaptation algorithm is to generalize well on the target domain by learning from labeled samples from the source domain as well as unlabeled samples from the target domain. Formally, let the *risk* of hypothesis h be the error of h w.r.t. the true labeling function under domain $\mathcal{D}_S$, i.e., $\varepsilon_S(h) := \varepsilon_S(h, f_S)$. As commonly used in computational learning theory, we denote by $\widehat{\varepsilon}_S(h)$ the empirical risk of h on the source domain. Similarly, we use $\varepsilon_T(h)$ and $\widehat{\varepsilon}_T(h)$ to mean the true risk and the empirical risk on the target domain. The problem of domain adaptation considered in this chapter can be stated as: under what conditions and by what algorithms can we guarantee that a small training error $\widehat{\varepsilon}_S(h)$ implies a small test error $\varepsilon_T(h)$? Clearly, this goal is not always possible if the source and target domains are far away from each other.

4.2.2 A Theoretical Model for Domain Adaptation

To measure the similarity between two domains, it is crucial to define a discrepancy measure between them. To this end, Ben-David et al. (2010) proposed the $\mathcal{H}$-divergence to measure the distance between two distributions:

Definition 4.2.1 ($\mathcal{H}$-divergence)**.** Let $\mathcal{H}$ be a hypothesis class on input space $\mathcal{X}$, and $\mathcal{A}_\mathcal{H}$ be the collection of subsets of $\mathcal{X}$ that are the support of some hypothesis in $\mathcal{H}$, i.e., $\mathcal{A}_\mathcal{H} := \{h^{-1}(1) \mid h \in \mathcal{H}\}$. The distance between two distributions $\mathcal{D}$ and $\mathcal{D}'$ based on $\mathcal{H}$ is: $d_\mathcal{H}(\mathcal{D}, \mathcal{D}') := \sup_{A\in\mathcal{A}_\mathcal{H}} |\Pr_\mathcal{D}(A) - \Pr_{\mathcal{D}'}(A)|$. [1]

[1] To be precise, Ben-David et al. (2007)'s original definition of $\mathcal{H}$-divergence has a factor of 2, we choose the current definition as the constant factor is inessential.

$\mathcal{H}$-divergence is particularly favorable in the analysis of domain adaptation with binary classification problems, and it had also been generalized to the *discrepancy distance* (Cortes and Mohri, 2014; Cortes et al., 2008; Mansour et al., 2009a,c) for general loss functions, including the one for regression problems. Both $\mathcal{H}$-divergence and the discrepancy distance can be estimated using finite unlabeled samples from both domains when $\mathcal{H}$ has a finite VC-dimension.

One flexibility of the $\mathcal{H}$-divergence is that its power on measuring the distance between two distributions can be controlled by the richness of the hypothesis class $\mathcal{H}$. To see this, first consider the situation where $\mathcal{H}$ is very restrictive so that it only contains the constant functions $h \equiv 0$ and $h \equiv 1$. In this case, it can be readily verified by the definition that $d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 0, \ \forall\ \mathcal{D}, \mathcal{D}'$. On the other extreme, if $\mathcal{H}$ contains all the measurable binary functions, then $d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 0$ iff $\mathcal{D}(\cdot) = \mathcal{D}'(\cdot)$ almost surely. In this case the $\mathcal{H}$-divergence reduces to the total variation, or equivalently the L_1 distance, between the two distributions.

Given a hypothesis class $\mathcal{H}$, we define its symmetric difference w.r.t. itself as: $\mathcal{H}\Delta\mathcal{H} = \{h(\mathbf{x}) \oplus h'(\mathbf{x}) \mid h, h' \in \mathcal{H}\}$, where $\oplus$ is the `xor` operation. Let h^* be the optimal hypothesis that achieves the minimum joint risk on both the source and target domains: $h^* := \arg\min_{h\in\mathcal{H}} \varepsilon_S(h) + \varepsilon_T(h)$, and let λ^* denote the joint risk of the optimal hypothesis h^*: $\lambda^* := \varepsilon_S(h^*) + \varepsilon_T(h^*)$. Ben-David et al. (2007) proved the following generalization bound on the target risk in terms of the empirical source risk and the discrepancy between the source and target domains:

Theorem 4.2.1 (Ben-David et al. (2007))**.** Let $\mathcal{H}$ be a hypothesis space of VC-dimension d and $\widehat{\mathcal{D}}_S$ (resp. $\widehat{\mathcal{D}}_T$) be the empirical distribution induced by a sample of size n drawn from $\mathcal{D}_S$ (resp. $\mathcal{D}_T$). Then w.p. at least $1-\delta$, $\forall h \in \mathcal{H}$,

$$\varepsilon_T(h) \leq \widehat{\varepsilon}_S(h) + \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T) + \lambda^* + O\left(\sqrt{\frac{d\log n + \log(1/\delta)}{n}}\right). \tag{4.1}$$

The bound depends on λ^*, the optimal joint risk that can be achieved by the hypotheses in $\mathcal{H}$. The intuition is the following: if λ^* is large, we cannot hope for a successful domain adaptation. Later in Sec. 4.4.3, we shall get back to this term to show an information-theoretic lower bound on it for any approach based on learning invariant representations.

Theorem 4.2.1 is the foundation of many recent works on unsupervised domain adaptation via learning invariant representations (Ajakan et al., 2014; Ganin et al., 2016; Pei et al., 2018; Zhao et al., 2018b,c). It has also inspired various applications of domain adaptation with adversarial learning, e.g., video analysis (Hoffman et al., 2016, 2017b; Shrivastava et al., 2016; Tzeng et al., 2017), natural language understanding (Fu et al., 2017; Zhang et al., 2017b), speech recognition (Hosseini-Asl et al., 2018; Zhao et al., 2019g), to name a few.

At a high level, the key idea is to learn a rich and parametrized feature transformation $g : \mathcal{X} \mapsto \mathcal{Z}$ such that the induced source and target distributions (on $\mathcal{Z}$) are close, as measured by the $\mathcal{H}$-divergence. We call g an *invariant representation* w.r.t. $\mathcal{H}$ if $d_{\mathcal{H}}(\mathcal{D}_S^g, \mathcal{D}_T^g) = 0$, where $\mathcal{D}_S^g / \mathcal{D}_T^g$ is the induced source/target distribution. At the same time, these algorithms also try to find new hypothesis (on the representation space $\mathcal{Z}$) to achieve a small empirical error on the source domain. As a whole algorithm, these two procedures corresponds to simultaneously finding invariant representations and hypothesis to minimize the first two terms in the generalization upper bound of Theorem 4.2.1.

4.3 Related Work

A number of adaptation approaches based on learning invariant representations have been proposed in recent years. Although in this chapter we mainly focus on using the $\mathcal{H}$-divergence to characterize the discrepancy between two distributions, other distance measures can be used as well, e.g., the maximum mean discrepancy (MMD) (Long et al., 2014, 2015, 2016), the Wasserstein distance (Courty et al., 2017a,b; Lee and Raginsky, 2018; Shen et al., 2018), etc.

Under the theoretical framework of the $\mathcal{H}$-divergence, Ganin et al. (2016) propose a domain adversarial neural network (DANN) to learn the domain invariant features. Adversarial training techniques that aim to build feature representations that are indistinguishable between source and target domains have been proposed in the last few years (Ajakan et al., 2014; Ganin et al., 2016). Specifically, one of the central ideas is to use neural networks, which are powerful function approximators, to approximate the $\mathcal{H}$-divergence between two domains (Ben-David et al., 2007, 2010; Kifer et al., 2004). The overall algorithm can be viewed as a zero-sum two-player game: one network tries to learn feature representations that can fool the other network, whose goal is to distinguish the representations generated on the source domain from those generated on the target domain. In a concurrent work, Johansson et al. (2019) also identified the insufficiency of learning domain-invariant representation for successful adaptation. They further analyzed the information loss of non-invertible transformations, and proposed a generalization upper bound that directly takes it into account. In our work, by showing an information-theoretic lower bound on the joint error of these methods, we show that although invariant representations can be achieved, it does not necessarily translate to good generalization on the target domain, in particular when the label distributions of the two domains differ significantly.

Causal approaches based on conditional and label shifts for domain adaptation also exist (Azizzadenesheli et al., 2018; Gong et al., 2016; Lipton et al., 2018; Zhang et al., 2013). One typical assumption made to simplify the analysis in this line of work is that the source and target domains share the same generative distribution and only differ at the marginal label distributions. It is worth noting that Zhang et al. (2013) and Gong et al. (2016) showed that both label and conditional shift can be successfully corrected when the changes in the generative distribution follow some parametric families. In this chapter we focus on representation learning and do not make such explicit assumptions.

4.4 Theoretical Analysis

Is finding invariant representations alone a sufficient condition for the success of domain adaptation? Clearly it is not. Consider the following simple counterexample: let $g_{\mathbf{c}} : \mathcal{X} \mapsto \mathcal{Z}$ be a constant function, where $\forall \mathbf{x} \in \mathcal{X}, g_{\mathbf{c}}(\mathbf{x}) = \mathbf{c} \in \mathcal{Z}$. Then for any discrepancy distance $d(\cdot, \cdot)$ over two distributions, including the $\mathcal{H}$-divergence, MMD, and the Wasserstein distance, and for any distributions $\mathcal{D}_S, \mathcal{D}_T$ over the input space $\mathcal{X}$, we have $d(\mathcal{D}_S^{g_{\mathbf{c}}}, \mathcal{D}_T^{g_{\mathbf{c}}}) = 0$, where we use $\mathcal{D}_S^{g_{\mathbf{c}}}$ (resp. $\mathcal{D}_T^{g_{\mathbf{c}}}$) to mean the induced source (resp. target) distribution by the transformation $g_{\mathbf{c}}$ over the representation space $\mathcal{Z}$. Furthermore, it is fairly easy to construct source and target domains $\langle \mathcal{D}_S, f_S \rangle$, $\langle \mathcal{D}_T, f_T \rangle$, such that for any hypothesis $h : \mathcal{Z} \mapsto \mathcal{Y}$, $\varepsilon_T(h \circ g_{\mathbf{c}}) \geq 1/2$, while there exists a classification function $f : \mathcal{X} \to \mathcal{Y}$ that achieves small error, e.g., the labeling function.

One may argue, with good reason, that in the counterexample above, the empirical source error $\widehat{\varepsilon}_S(h \circ g_{\mathbf{c}})$ is also large with high probability. Intuitively, this is because the simple constant transformation function $g_{\mathbf{c}}$ fails to retain the discriminative information about the classification task at hand, despite the fact that it can construct invariant representations.

Is finding invariant representations and achieving a small source error sufficient to guarantee small target error? In this section we first give a negative answer to this question by constructing a counterexample where there exists a nontrivial transformation function $g : \mathcal{X} \mapsto \mathcal{Z}$ and hypothesis $h : \mathcal{Z} \mapsto \mathcal{Y}$ such that both $\varepsilon_S(h \circ g)$ and $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_S^g, \mathcal{D}_T^g)$ are small, while at the same time the target error $\varepsilon_T(h \circ g)$ is large. Motivated by this negative result, we proceed to prove a generalization upper bound that explicitly characterizes a sufficient condition for the success of domain adaptation. We then complement the upper bound by showing an information-theoretic lower bound on the joint error of *any* domain adaptation approach based on learning invariant representations.

4.4.1 Invariant Representation and Small Source Risk are Not Sufficient

In this section, we shall construct a simple 1-dimensional example where there exists a function $h^* : \mathbb{R} \mapsto \{0,1\}$ that achieves zero error on *both* source and target domains. Simultaneously, we show that there exists a transformation function $g : \mathbb{R} \mapsto \mathbb{R}$ under which the induced source and target distributions are perfectly aligned, but *every* hypothesis $h : \mathbb{R} \mapsto \{0,1\}$ incurs a large joint error on the induced source and target domains. The latter further implies that if we find a hypothesis that achieves small error on the source domain, then it has to incur a large error on the target domain. We illustrate this example in Fig. 4.1.1.

Let $\mathcal{X} = \mathcal{Z} = \mathbb{R}$ and $\mathcal{Y} = \{0,1\}$. For $a \leq b$, we use $U(a,b)$ to denote the uniform distribution over $[a,b]$. Consider the following source and target domains:

$$\mathcal{D}_S = U(-1,0), \qquad f_S(x) = \begin{cases} 0, & x \leq -1/2 \\ 1, & x > -1/2 \end{cases}$$

$$\mathcal{D}_T = U(1,2), \qquad f_T(x) = \begin{cases} 0, & x \geq 3/2 \\ 1, & x < 3/2 \end{cases}$$

In the above example, it is easy to verify that the interval hypothesis $h^*(x) = 1$ iff $x \in (-1/2, 3/2)$ achieves perfect classification on *both* domains.

Now consider the following transformation:

$$g(x) = \mathbb{I}_{x \leq 0}(x) \cdot (x+1) + \mathbb{I}_{x>0}(x) \cdot (x-1).$$

Since $g(\cdot)$ is a piecewise linear function, it follows that $\mathcal{D}_S^Z = \mathcal{D}_T^Z = U(0,1)$, and for any distance metric $d(\cdot,\cdot)$ over distributions, we have $d(\mathcal{D}_S^Z, \mathcal{D}_T^Z) = 0$. But now for any hypothesis $h : \mathbb{R} \mapsto \{0,1\}$, and $\forall x \in [0,1]$, $h(x)$ will make an error in exactly one of the domains, hence

$$\forall h : \mathbb{R} \mapsto \{0,1\}, \quad \varepsilon_S(h \circ g) + \varepsilon_T(h \circ g) = 1.$$

In other words, under the above invariant transformation g, the smaller the source error, the larger the target error.

One may argue that this example seems to contradict the generalization upper bound from Theorem 4.2.1, where the first two terms correspond exactly to a small source error and an invariant representation. The key to explain this apparent contradiction lies in the third term of the upper bound, λ^*, i.e., the optimal joint error achievable on *both* domains. In our example, when there is no transformation applied to the input space, we show that h^* achieves 0 error on both domains, hence $\lambda^* = \min_{h \in \mathcal{H}} \varepsilon_S(h) + \varepsilon_T(h) = 0$. However, when the transformation g is applied to the original input space, we prove that every hypothesis has joint error 1 on the representation space, hence $\lambda_g^* = 1$. Since we usually do not have access to the optimal hypothesis on both domains, although the generalization bound still holds on the representation space, it becomes vacuous in our example.

An alternative way to interpret the failure of the constructed example is that the labeling functions (or conditional distributions in the stochastic setting) of source and target domains are far away from each other in the representation space. Specifically, in the induced representation space, the optimal labeling function on the source and target domains are:

$$f'_S(x) = \begin{cases} 0, & x \le 1/2 \\ 1, & x > 1/2 \end{cases}, \quad f'_T(x) = \begin{cases} 0, & x > 1/2 \\ 1, & x \le 1/2 \end{cases},$$

and we have $\|f'_S - f'_T\|_1 = \mathbb{E}_{x \sim U(0,1)}[|f'_S(x) - f'_T(x)|] = 1$.

4.4.2 A Generalization Upper Bound

For most of the practical hypothesis spaces $\mathcal{H}$, e.g., half spaces, it is usually intractable to compute the optimal joint error λ^* from Theorem 4.2.1. Furthermore, the fact that λ^* contains errors from both domains makes the bound very conservative and loose in many cases. In this section, inspired by the constructed example from Sec. 4.4.1, we aim to provide a general, intuitive, and interpretable generalization upper bound for domain adaptation that is free of the pessimistic λ^* term. Ideally, the bound should also explicitly characterize how the shift between labeling functions of both domains affects domain adaptation.

Because of its flexibility in choosing the witness function class $\mathcal{H}$ and its natural interpretation as adversarial binary classification, we still adopt the $\mathcal{H}$-divergence to measure the discrepancy between two distributions. For any hypothesis space $\mathcal{H}$, it can be readily verified that $d_{\mathcal{H}}(\cdot,\cdot)$ satisfies the triangle inequality:

$$d_{\mathcal{H}}(\mathcal{D},\mathcal{D}') \le d_{\mathcal{H}}(\mathcal{D},\mathcal{D}'') + d_{\mathcal{H}}(\mathcal{D}'',\mathcal{D}'),$$

where $\mathcal{D},\mathcal{D}',\mathcal{D}''$ are any distributions over the same space. We now introduce a technical lemma that will be helpful in proving results related to the $\mathcal{H}$-divergence:

Lemma 4.4.1. Let $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ and $\mathcal{D},\mathcal{D}'$ be two distributions over $\mathcal{X}$. Then $\forall h,h' \in \mathcal{H}$, $|\varepsilon_{\mathcal{D}}(h,h') - \varepsilon_{\mathcal{D}'}(h,h')| \le d_{\tilde{\mathcal{H}}}(\mathcal{D},\mathcal{D}')$, where $\tilde{\mathcal{H}} := \{\mathrm{sgn}(|h(\mathbf{x}) - h'(\mathbf{x})| - t) \mid h,h' \in \mathcal{H}, 0 \le t \le 1\}$.

As a matter of fact, the above lemma also holds for any function class $\mathcal{H}$ (not necessarily a hypothesis space) where there exists a constant $M > 0$, such that $\|h\|_\infty \le M$ for all $h \in \mathcal{H}$. Another useful lemma is the following triangle inequality:

Lemma 4.4.2. Let $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ and $\mathcal{D}$ be any distribution over $\mathcal{X}$. For any $h,h',h'' \in \mathcal{H}$, we have $\varepsilon_{\mathcal{D}}(h,h') \le \varepsilon_{\mathcal{D}}(h,h'') + \varepsilon_{\mathcal{D}}(h'',h')$.

Let $f_S : \mathcal{X} \to [0,1]$ and $f_T : \mathcal{X} \to [0,1]$ be the optimal labeling functions on the source and target domains, respectively. In the stochastic setting, $f_S(\mathbf{x}) = \Pr_S(y = 1 \mid \mathbf{x})$ corresponds to the optimal Bayes classifier. With these notations, the following theorem holds:

Theorem 4.4.1. Let $\langle \mathcal{D}_S, f_S \rangle$ and $\langle \mathcal{D}_T, f_T \rangle$ be the source and target domains, respectively. For any function class $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$, and $\forall h \in \mathcal{H}$, the following inequality holds:

$$\varepsilon_T(h) \le \varepsilon_S(h) + d_{\tilde{\mathcal{H}}}(\mathcal{D}_S,\mathcal{D}_T) + \min\{\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|], \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]\}.$$

Remark The three terms in the upper bound have natural interpretations: the first term is the source error, the second one corresponds to the discrepancy between the marginal distributions, and the third measures the distance between the labeling functions from the source and target domains. Altogether, they form a sufficient condition for the success of domain adaptation: besides a small source error, not only do the marginal distributions need to be close, but so do the labeling functions.

Comparison with Theorem 4.2.1 It is instructive to compare the bound in Theorem 4.4.1 with the one in Theorem 4.2.1. The main difference lies in the λ^* in Theorem 4.2.1 and the $\min\{\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|], \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]\}$ in Theorem 4.4.1. λ^* depends on the choice of the hypothesis class $\mathcal{H}$, while our term does not. In fact, our quantity reflects the underlying structure of the problem, i.e., the conditional shift. Finally, consider the example given in the left panel of Fig. 4.1.1. It is easy to verify that we have $\min\{\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|], \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]\} = 1/2$ in this case, while for a natural class of hypotheses, i.e., $\mathcal{H} := \{h(x) = 0 \Leftrightarrow a \leq x \leq b \mid a < b\}$, we have $\lambda^* = 1$. In that case, our bound is tighter than the one in Theorem 4.2.1.

In the covariate shift setting, where we assume the conditional distributions of $Y \mid X$ between the source and target domains are the same, the third term in the upper bound vanishes. In that case the above theorem says that to guarantee successful domain adaptation, it suffices to match the marginal distributions while achieving small error on the source domain. In general settings where the optimal labeling functions of the source and target domains differ, the above bound says that it is not sufficient to simply match the marginal distributions and achieve small error on the source domain. At the same time, we should also guarantee that the optimal labeling functions (or the conditional distributions of both domains) are not too far away from each other. As a side note, it is easy to see that $\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|] = \varepsilon_S(f_T)$ and $\mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|] = \varepsilon_T(f_S)$. In other words, they are essentially the cross-domain errors. When the cross-domain error is small, it implies that the optimal source (resp. target) labeling function generalizes well on the target (resp. source) domain.

Both the error term $\varepsilon_S(h)$ and the divergence $d_{\tilde{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T)$ in Theorem 4.4.1 are with respect to the true underlying distributions $\mathcal{D}_S$ and $\mathcal{D}_T$, which are not available to us during training. In the following, we shall use the Rademacher complexity to provide for both terms a data-dependent bound from empirical samples from $\mathcal{D}_S$ and $\mathcal{D}_T$.

Definition 4.4.1 (Empirical Rademacher Complexity)**.** Let $\mathcal{H}$ be a family of functions mapping from $\mathcal{X}$ to $[a, b]$ and $\mathbf{S} = \{\mathbf{x}_i\}_{i=1}^n$ a fixed sample of size n with elements in $\mathcal{X}$. Then, the *empirical Rademacher complexity* of $\mathcal{H}$ with respect to the sample X is defined as

$$\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}) := \mathbb{E}_{\boldsymbol{\sigma}}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^n \sigma_i h(\mathbf{x}_i)\right],$$

where $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^n$ and σ_i are i.i.d. uniform random variables taking values in $\{+1, -1\}$.

With the empirical Rademacher complexity, we can show that w.h.p., the empirical source error $\widehat{\varepsilon}_S(h)$ cannot be too far away from the population error $\varepsilon_S(h)$ for all $h \in \mathcal{H}$:

Lemma 4.4.3. Let $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$, then for all $\delta > 0$, w.p. at least $1 - \delta$, the following inequality holds for all $h \in \mathcal{H}$: $\varepsilon_S(h) \leq \widehat{\varepsilon}_S(h) + 2\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}) + 3\sqrt{\log(2/\delta)/2n}$.

Similarly, for any distribution $\mathcal{D}$ over $\mathcal{X}$, let $\widehat{\mathcal{D}}$ be its empirical distribution from sample $\mathbf{S} \sim \mathcal{D}^n$ of size n. Then for any two distributions $\mathcal{D}$ and $\mathcal{D}'$, we can also use the empirical Rademacher complexity to provide a data-dependent bound for the perturbation between $d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}')$ and $d_{\mathcal{H}}(\widehat{\mathcal{D}}, \widehat{\mathcal{D}}')$:

Lemma 4.4.4. Let $\tilde{\mathcal{H}}$, $\mathcal{D}$ and $\widehat{\mathcal{D}}$ be defined above, then for all $\delta > 0$, w.p. at least $1 - \delta$, the following inequality holds for all $h \in \tilde{\mathcal{H}}$: $\mathbb{E}_{\mathcal{D}}[\mathbb{I}_h] \leq \mathbb{E}_{\widehat{\mathcal{D}}}[\mathbb{I}_h] + 2\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) + 3\sqrt{\log(2/\delta)/2n}$.

Since $\tilde{\mathcal{H}}$ is a hypothesis class, by definition we have:

$$\begin{aligned} d_{\tilde{\mathcal{H}}}(\mathcal{D}, \widehat{\mathcal{D}}) &= \sup_{A\in\mathcal{A}_{\tilde{\mathcal{H}}}} |\Pr_{\mathcal{D}}(A) - \Pr_{\widehat{\mathcal{D}}}(A)| \\ &= \sup_{h\in\tilde{\mathcal{H}}} |\mathbb{E}_{\mathcal{D}}[\mathbb{I}_h] - \mathbb{E}_{\widehat{\mathcal{D}}}[\mathbb{I}_h]|. \end{aligned}$$

Hence combining the above identity with Lemma 4.4.4, we immediately have w.p. at least $1 - \delta$:

$$d_{\tilde{\mathcal{H}}}(\mathcal{D}, \widehat{\mathcal{D}}) \leq 2\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) + 3\sqrt{\log(2/\delta)/2n}. \tag{4.2}$$

Now use a union bound and the fact that $d_{\tilde{\mathcal{H}}}(\cdot,\cdot)$ satisfies the triangle inequality, we have:

Lemma 4.4.5. Let $\tilde{\mathcal{H}}$, $\mathcal{D}, \mathcal{D}'$ and $\widehat{\mathcal{D}}, \widehat{\mathcal{D}}'$ be defined above, then for $\forall \delta > 0$, w.p. at least $1-\delta$, for $\forall h \in \tilde{\mathcal{H}}$:

$$d_{\tilde{\mathcal{H}}}(\mathcal{D}, \mathcal{D}') \leq d_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}, \widehat{\mathcal{D}}') + 4\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) + 6\sqrt{\log(4/\delta)/2n}.$$

Combine Lemma 4.4.3, Lemma 4.4.5 and Theorem 4.4.1 with a union bound argument, we get the following main theorem that characterizes an upper bound for domain adaptation:

Theorem 4.4.2. Let $\langle \mathcal{D}_S, f_S \rangle$ and $\langle \mathcal{D}_T, f_T \rangle$ be the source and target domains, and let $\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T$ be the empirical source and target distributions constructed from sample $\mathbf{S} = \{\mathbf{S}_S, \mathbf{S}_T\}$, each of size n. Then for any $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ and $\forall h \in \mathcal{H}$:

$$\begin{aligned}\varepsilon_T(h) \leq\ & \widehat{\varepsilon}_S(h) + d_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T) + 2\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}) + 4\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) \\ & + \min\{\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|], \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]\} \\ & + O\left(\sqrt{\log(1/\delta)/n}\right),\end{aligned}$$

where $\tilde{\mathcal{H}} := \{\mathrm{sgn}(|h(\mathbf{x}) - h'(\mathbf{x})| - t) | h, h' \in \mathcal{H}, t \in [0,1]\}$.

Essentially, the generalization upper bound can be decomposed into three parts: the first part comes from the domain adaptation setting, including the empirical source error, the empirical $\mathcal{H}$-divergence, and the shift between labeling functions. The second part corresponds to complexity measures of our hypothesis space $\mathcal{H}$ and $\tilde{\mathcal{H}}$, and the last part describes the error caused by finite samples.

4.4.3 An Information-Theoretic Lower Bound

In Sec. 4.4.1, we constructed an example to demonstrate that learning invariant representations could lead to a feature space where the joint error on both domains is large. In this section, we extend the example by showing that a similar result holds in more general settings. Specifically, we shall prove that for *any* approach based on learning invariant representations, there is an intrinsic lower bound on the joint error of source and target domains, due to the discrepancy between their marginal label distributions. Our result hence highlights the need to take into account task related information when designing domain adaptation algorithms based on learning invariant representations.

Before we proceed to the lower bound, we first define several information-theoretic concepts that will be used in the analysis. For two distributions $\mathcal{D}$ and $\mathcal{D}'$, the Jensen-Shannon (JS) divergence $D_{\mathrm{JS}}(\mathcal{D} \parallel \mathcal{D}')$ is defined as:

$$D_{\mathrm{JS}}(\mathcal{D} \parallel \mathcal{D}') := \frac{1}{2} D_{\mathrm{KL}}(\mathcal{D} \parallel \mathcal{D}_M) + \frac{1}{2} D_{\mathrm{KL}}(\mathcal{D}' \parallel \mathcal{D}_M),$$

where $D_{\mathrm{KL}}(\cdot \parallel \cdot)$ is the Kullback–Leibler (KL) divergence and $\mathcal{D}_M := (\mathcal{D} + \mathcal{D}')/2$. The JS divergence can be viewed as a symmetrized and smoothed version of the KL divergence, and it is closely related to the L_1 distance between two distributions through Lin's lemma (Lin, 1991).

Unlike the KL divergence, the JS divergence is bounded: $0 \leq D_{\mathrm{JS}}(\mathcal{D} \parallel \mathcal{D}') \leq 1$. Additionally, from the JS divergence, we can define a distance metric between two distributions as well, known as the JS distance (Endres and Schindelin, 2003):

$$d_{\mathrm{JS}}(\mathcal{D}, \mathcal{D}') := \sqrt{D_{\mathrm{JS}}(\mathcal{D} \parallel \mathcal{D}')}.$$

With respect to the JS distance and for any (stochastic) mapping $h : \mathcal{Z} \mapsto \mathcal{Y}$, we can prove the following lemma via the celebrated data processing inequality:

Lemma 4.4.6. Let $\mathcal{D}_S^Z$ and $\mathcal{D}_T^Z$ be two distributions over $\mathcal{Z}$ and let $\mathcal{D}_S^Y$ and $\mathcal{D}_T^Y$ be the induced distributions over $\mathcal{Y}$ by function $h : \mathcal{Z} \mapsto \mathcal{Y}$, then

$$d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) \leq d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z). \tag{4.3}$$

For methods that aim to learn invariant representations for domain adaptation, an intermediate representation space $\mathcal{Z}$ is found through feature transformation g, based on which a common hypothesis $h : \mathcal{Z} \mapsto \mathcal{Y}$ is shared between both domains (Ganin et al., 2016; Tzeng et al., 2017; Zhao et al., 2018b). Through this process, the following Markov chain holds:

$$X \xrightarrow{g} Z \xrightarrow{h} \hat{Y}, \tag{4.4}$$

where $\hat{Y} = h(g(X))$ is the predicted random variable of interest. Hence for any distribution $\mathcal{D}$ over $\mathcal{X}$, this Markov chain also induces a distribution $\mathcal{D}^Z$ over $\mathcal{Z}$ and $\mathcal{D}^{\hat{Y}}$ over $\mathcal{Y}$. By Lemma 4.4.6, we know that $d_{\text{JS}}(\mathcal{D}_S^{\hat{Y}}, \mathcal{D}_T^{\hat{Y}}) \leq d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z)$. With these notations, noting that the JS distance is a metric, the following inequality holds:

$$d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) \leq d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_S^{\hat{Y}}) + d_{\text{JS}}(\mathcal{D}_S^{\hat{Y}}, \mathcal{D}_T^{\hat{Y}}) + d_{\text{JS}}(\mathcal{D}_T^{\hat{Y}}, \mathcal{D}_T^Y).$$

Combining the above inequality with Lemma 4.4.6, we immediately have:

$$\begin{aligned} d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) &\leq d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z) \\ &\quad + d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_S^{\hat{Y}}) + d_{\text{JS}}(\mathcal{D}_T^Y, \mathcal{D}_T^{\hat{Y}}). \end{aligned} \tag{4.5}$$

Intuitively, $d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_S^{\hat{Y}})$ and $d_{\text{JS}}(\mathcal{D}_T^Y, \mathcal{D}_T^{\hat{Y}})$ measure the distance between the predicted label distribution and the ground truth label distribution on the source and target domain, respectively. Formally, the following result establishes a relationship between $d_{\text{JS}}(\mathcal{D}^Y, \mathcal{D}^{\hat{Y}})$ and the accuracy of the prediction function h:

Lemma 4.4.7. Let $Y = f(X) \in \{0,1\}$ where $f(\cdot)$ is the labeling function and $\hat{Y} = h(g(X)) \in \{0,1\}$ be the prediction function, then $d_{\text{JS}}(\mathcal{D}^Y, \mathcal{D}^{\hat{Y}}) \leq \sqrt{\varepsilon(h \circ g)}$.

We are now ready to present the key lemma of the section:

Lemma 4.4.8. Suppose the Markov chain $X \xrightarrow{g} Z \xrightarrow{h} \hat{Y}$ holds, then

$$d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) \leq d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z) + \sqrt{\varepsilon_S(h \circ g)} + \sqrt{\varepsilon_T(h \circ g)}.$$

Remark This lemma shows that if the marginal label distributions are significantly different between the source and target domains, then in order to achieve a small joint error, the induced distributions over $\mathcal{Z}$ from source and target domains have to be significantly different as well. Put another way, if we are able to find an invariant representation such that $d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z) = 0$, then the joint error of the composition function $h \circ g$ has to be large:

Theorem 4.4.3. Suppose the condition in Lemma 4.4.8 holds and $d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) \geq d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z)$, then:

$$\varepsilon_S(h \circ g) + \varepsilon_T(h \circ g) \geq \frac{1}{2}\left(d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) - d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z)\right)^2.$$

Remark The lower bound gives us a necessary condition on the success of any domain adaptation approach based on learning invariant representations: if the marginal label distributions are significantly different between source and target domains, then minimizing $d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z)$ and the source error $\varepsilon_S(h \circ g)$ will only increase the target error. In fact, Theorem 8.2.1 can be extended to hold in the setting where different transformation functions are applied in source and target domains:

Corollary 4.4.1. Let g_S, g_T be the source and target transformation functions from $\mathcal{X}$ to $\mathcal{Z}$. Suppose the condition in Lemma 4.4.8 holds and $d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) \geq d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z)$, then:

$$\varepsilon_S(h \circ g_S) + \varepsilon_T(h \circ g_T) \geq \frac{1}{2}\left(d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) - d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z)\right)^2.$$

Recent work has also explored using different transformation functions to achieve invariant representations (Bousmalis et al., 2016; Tzeng et al., 2017), but Corollary 4.4.1 shows that this is not going to help if the marginal label distributions differ between two domains.

We conclude this section by noting that our bound on the joint error of both domains is not necessarily the tightest one. This can be seen from the example in Sec. 4.4.1, where $d_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z) = d_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) = 0$, and we have $\varepsilon_S(h \circ g) + \varepsilon_T(h \circ g) = 1$, but in this case our result gives a trivial lower bound of 0. Nevertheless, our result still sheds new light on the importance of matching marginal label distributions in learning invariant representation for domain adaptation, which we believe to be a promising direction for the design of better adaptation algorithms.

4.5 Experiments

Our theoretical results on the lower bound of the joint error imply that over-training the feature transformation function and the discriminator may hurt generalization on the target domain. In this section, we conduct experiments on real-world datasets to verify our theoretical findings. The task is digit classification on three datasets of 10 classes: MNIST, USPS and SVHN. MNIST contains 60,000/10,000 train/test instances; USPS contains 7,291/2,007 train/test instances, and SVHN contains 73,257/26,032 train/test instances. We show the label distribution of these three datasets in Fig. 4.5.1.

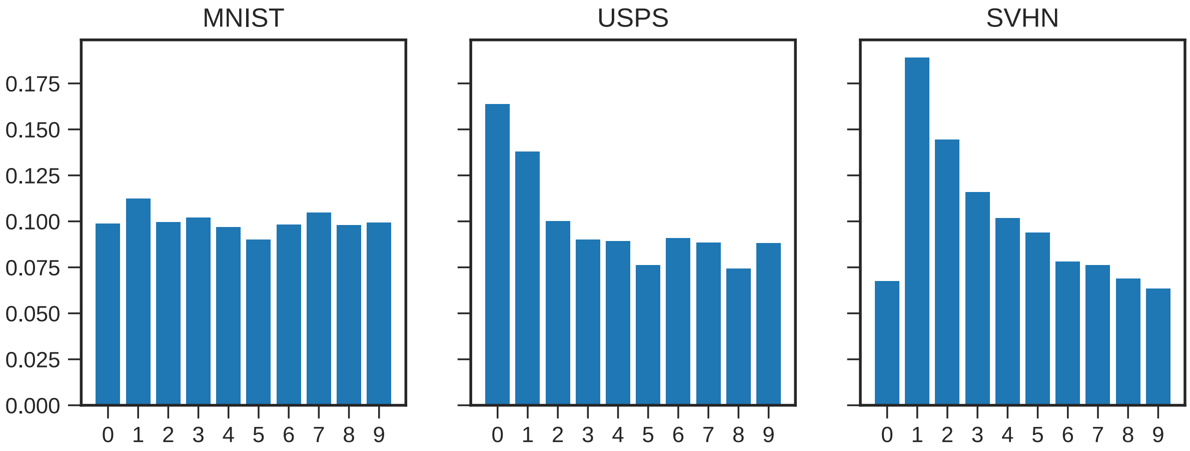
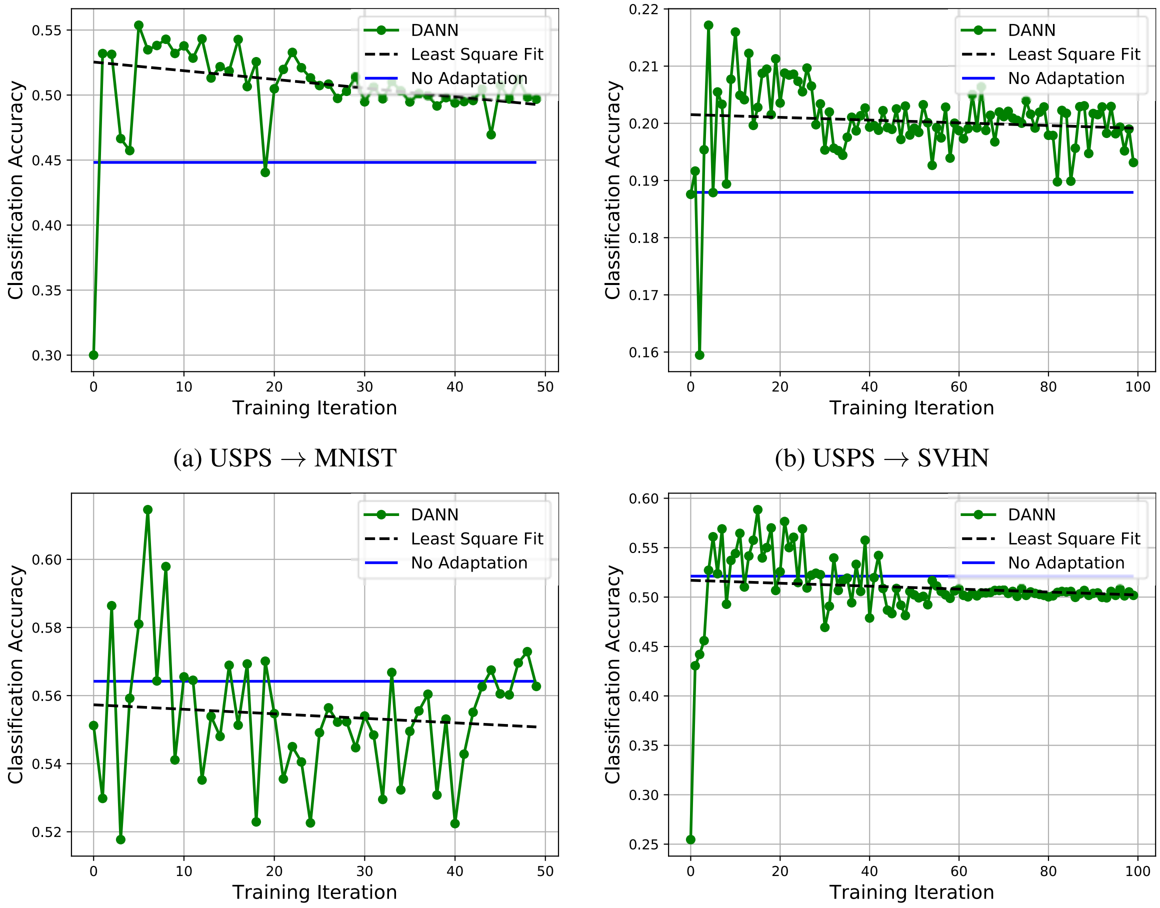


Figure 4.5.1: The label distributions of MNIST, USPS and SVHN.

Before training, we preprocess all the samples into gray scale single-channel images of size 16 × 16, so they can be used by the same network. In our experiments, to ensure a fair comparison, we use the same network structure for all the experiments: 2 convolutional layers, one fully connected hidden layer, followed by a softmax output layer with 10 units. The convolution kernels in both layers are of size 5 × 5, with 10 and 20 channels, respectively. The hidden layer has 1280 units connected to 100 units before classification. For domain adaptation, we use the original DANN (Ganin et al., 2016) with gradient reversal



(a) USPS → MNIST

(b) USPS → SVHN

(c) SVHN → MNIST

(d) SVHN → USPS

Figure 4.5.2: Digit classification on MNIST, USPS and SVHN. The horizontal solid line corresponds to the target domain test accuracy without adaptation. The green solid line is the target domain test accuracy under domain adaptation with DANN. We also plot the least square fit (dashed line) of the DANN adaptation results to emphasize the negative slope.

implementation. The discriminator in DANN takes the output of convolutional layers as its feature input, followed by a 500×100 fully connected layer, and a one-unit binary classification output.

We plot four adaptation trajectories in Fig. 4.5.2. Among the four adaptation tasks, we can observe two phases in the adaptation accuracy. In the first phase, the test set accuracy rapidly grows, in less than 10 iterations. In the second phase, it gradually decreases after reaching its peak, despite the fact that the source training accuracy keeps increasing smoothly. Those phase transitions can be verified from the negative slopes of the least squares fit of the adaptation curves (dashed lines in Fig. 4.5.2). We observe similar phenomenons on additional experiments using artificially unbalanced datasets trained on more powerful networks as well. The above experimental results imply that over-training the feature transformation and discriminator does not help generalization on the target domain, but can instead hurt it when the label distributions differ (as shown in Fig. 4.5.1). These experimental results are consistent with our theoretical findings.

4.6 Proofs

In this section we provide all the missing proofs in Sec. 4.4. Again, we will first restate the corresponding statements and then provide proofs for each of them.

Lemma 4.4.1. Let $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ and $\mathcal{D}, \mathcal{D}'$ be two distributions over $\mathcal{X}$. Then $\forall h, h' \in \mathcal{H}$, $|\varepsilon_{\mathcal{D}}(h,h') - \varepsilon_{\mathcal{D}'}(h,h')| \leq d_{\tilde{\mathcal{H}}}(\mathcal{D}, \mathcal{D}')$, where $\tilde{\mathcal{H}} := \{\text{sgn}(|h(\mathbf{x}) - h'(\mathbf{x})| - t) \mid h, h' \in \mathcal{H}, 0 \leq t \leq 1\}$.

Proof. By definition, for $\forall h, h' \in \mathcal{H}$, we have:

$$
\begin{aligned}
|\varepsilon_S(h,h') - \varepsilon_T(h,h')| &\leq \sup_{h,h' \in \mathcal{H}} |\varepsilon_S(h,h') - \varepsilon_T(h,h')| \\
&= \sup_{h,h' \in \mathcal{H}} \left|\mathbb{E}_{\mathbf{x}\sim S}[|h(\mathbf{x}) - h'(\mathbf{x})|] - \mathbb{E}_{\mathbf{x}\sim T}[|h(\mathbf{x}) - h'(\mathbf{x})|]\right|
\end{aligned}
\tag{4.6}
$$

Since $\|h\|_\infty \leq 1, \forall h \in \mathcal{H}$, then $0 \leq |h(\mathbf{x}) - h'(\mathbf{x})| \leq 1$, $\forall \mathbf{x} \in \mathcal{X}, h, h' \in \mathcal{H}$. We now use Fubini's theorem to bound $\left|\mathbb{E}_{\mathbf{x}\sim S}[|h(\mathbf{x}) - h'(\mathbf{x})|] - \mathbb{E}_{\mathbf{x}\sim T}[|h(\mathbf{x}) - h'(\mathbf{x})|]\right|$:

$$
\begin{aligned}
\big|\mathbb{E}_{\mathbf{x}\sim S}[|h(\mathbf{x}) - h'(\mathbf{x})|] - &\mathbb{E}_{\mathbf{x}\sim T}[|h(\mathbf{x}) - h'(\mathbf{x})|]\big| \\
&= \left|\int_0^1 \left(\Pr_S(|h(\mathbf{x}) - h'(\mathbf{x})| > t) - \Pr_T(|h(\mathbf{x}) - h'(\mathbf{x})| > t)\right) dt\right| \\
&\leq \int_0^1 \left|\Pr_S(|h(\mathbf{x}) - h'(\mathbf{x})| > t) - \Pr_T(|h(\mathbf{x}) - h'(\mathbf{x})| > t)\right| dt \\
&\leq \sup_{t \in [0,1]} \left|\Pr_S(|h(\mathbf{x}) - h'(\mathbf{x})| > t) - \Pr_T(|h(\mathbf{x}) - h'(\mathbf{x})| > t)\right|
\end{aligned}
$$

Now in view of (4.6) and the definition of $\tilde{\mathcal{H}}$, we have:

$$
\begin{aligned}
&\sup_{h,h' \in \mathcal{H}} \sup_{t \in [0,1]} \left|\Pr_S(|h(\mathbf{x}) - h'(\mathbf{x})| > t) - \Pr_T(|h(\mathbf{x}) - h'(\mathbf{x})| > t)\right| \\
&= \sup_{\tilde{h} \in \tilde{\mathcal{H}}} |\Pr_S(\tilde{h}(\mathbf{x}) = 1) - \Pr_T(\tilde{h}(\mathbf{x}) = 1)| \\
&= \sup_{A \in \mathcal{A}_{\tilde{\mathcal{H}}}} |\Pr_S(A) - \Pr_T(A)| \\
&= d_{\tilde{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T)
\end{aligned}
$$

Combining all the inequalities above finishes the proof. ■

Lemma 4.4.2. Let $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ and $\mathcal{D}$ be any distribution over $\mathcal{X}$. For any $h, h', h'' \in \mathcal{H}$, we have $\varepsilon_{\mathcal{D}}(h,h') \leq \varepsilon_{\mathcal{D}}(h,h'') + \varepsilon_{\mathcal{D}}(h'',h')$.

Proof.

$$
\begin{aligned}
\varepsilon_{\mathcal{D}}(h,h') &= \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[|h(\mathbf{x}) - h'(\mathbf{x})|] = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[|h(\mathbf{x}) - h''(\mathbf{x}) + h''(\mathbf{x}) - h'(\mathbf{x})|] \\
&\leq \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[|h(\mathbf{x}) - h''(\mathbf{x})| + |h''(\mathbf{x}) - h'(\mathbf{x})|] = \varepsilon_{\mathcal{D}}(h,h'') + \varepsilon_{\mathcal{D}}(h'',h')
\end{aligned}
$$

■

Theorem 4.4.1. Let $\langle \mathcal{D}_S, f_S \rangle$ and $\langle \mathcal{D}_T, f_T \rangle$ be the source and target domains, respectively. For any function class $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$, and $\forall h \in \mathcal{H}$, the following inequality holds:

$$
\varepsilon_T(h) \leq \varepsilon_S(h) + d_{\tilde{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T) + \min\{\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|], \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]\}.
$$

Proof. On one hand, with Lemma 8.2.1 and Lemma 4.4.2, we have $\forall h \in \mathcal{H}$:

$$\varepsilon_T(h) = \varepsilon_T(h, f_T) \le \varepsilon_S(h, f_T) + d_{\tilde{H}}(\mathcal{D}_S, \mathcal{D}_T) \le \varepsilon_S(h) + \varepsilon_S(f_S, f_T) + d_{\tilde{H}}(\mathcal{D}_S, \mathcal{D}_T).$$

On the other hand, by changing the order of two triangle inequalities, we also have:

$$\varepsilon_T(h) = \varepsilon_T(h, f_T) \le \varepsilon_T(h, f_S) + \varepsilon_T(f_S, f_T) \le \varepsilon_S(h) + \varepsilon_T(f_S, f_T) + d_{\tilde{H}}(\mathcal{D}_S, \mathcal{D}_T).$$

Realize that by definition $\varepsilon_S(f_S, f_T) = \mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|]$ and $\varepsilon_T(f_S, f_T) = \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]$. Combining the above two inequalities completes the proof. ■

Lemma 4.4.3. Let $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$, then for all $\delta > 0$, w.p. at least $1 - \delta$, the following inequality holds for all $h \in \mathcal{H}$: $\varepsilon_S(h) \le \widehat{\varepsilon}_S(h) + 2\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}) + 3\sqrt{\log(2/\delta)/2n}$.

Proof. Consider the source domain $\mathcal{D}_S$. For $\forall h \in \mathcal{H}$, define the loss function $\ell : \mathcal{X} \to [0,1]$ as $\ell(\mathbf{x}) := |h(\mathbf{x}) - f_S(\mathbf{x})|$. First, we know that $\mathrm{Rad}_{\mathbf{S}}(\mathcal{H} - f_S) = \mathrm{Rad}_{\mathbf{S}}(\mathcal{H})$ where we slightly abuse the notation $\mathcal{H} - f_S$ to mean the family of functions $\{h - f_S \mid \forall h \in \mathcal{H}\}$:

$$\begin{aligned}
\mathrm{Rad}_{\mathbf{S}}(\mathcal{H} - f_S) &= \mathbb{E}_{\sigma}\left[\sup_{h' \in \mathcal{H} - f_S} \frac{1}{n}\sum_{i=1}^{n} \sigma_i h'(\mathbf{x}_i)\right] = \mathbb{E}_{\sigma}\left[\sup_{h \in \mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i (h(\mathbf{x}_i) - f_S(\mathbf{x}_i))\right] \\
&= \mathbb{E}_{\sigma}\left[\sup_{h \in \mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i h(\mathbf{x}_i)\right] + \mathbb{E}_{\sigma}\left[\frac{1}{n}\sum_{i=1}^{n} \sigma_i f_S(\mathbf{x}_i)\right] \\
&= \mathrm{Rad}_{\mathbf{S}}(\mathcal{H})
\end{aligned}$$

Observe that the function $\phi : t \to |t|$ is 1-Lipschitz continuous, then by Ledoux-Talagrand's contraction lemma, we can conclude that

$$\mathrm{Rad}_{\mathbf{S}}(\phi \circ (\mathcal{H} - f_S)) \le \mathrm{Rad}_{\mathbf{S}}(\mathcal{H} - f_S) = \mathrm{Rad}_{\mathbf{S}}(\mathcal{H})$$

Using Lemma 2.6.1 with the above arguments and realize that $\varepsilon_S(h) = \mathbb{E}_{\mathbf{x} \sim \mathcal{D}_S}[|h(\mathbf{x}) - f_S(\mathbf{x})|]$ finishes the proof. ■

Lemma 4.4.4. Let $\tilde{\mathcal{H}}$, $\mathcal{D}$ and $\widehat{\mathcal{D}}$ be defined above, then for all $\delta > 0$, w.p. at least $1 - \delta$, the following inequality holds for all $h \in \tilde{\mathcal{H}}$: $\mathbb{E}_{\mathcal{D}}[\mathbb{I}_h] \le \mathbb{E}_{\widehat{\mathcal{D}}}[\mathbb{I}_h] + 2\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) + 3\sqrt{\log(2/\delta)/2n}$.

Proof. Note that $\mathbb{I}_h \in \{0,1\}$, hence this lemma directly follows Lemma 2.6.1. ■

Lemma 4.4.5. Let $\tilde{\mathcal{H}}$, $\mathcal{D}, \mathcal{D}'$ and $\widehat{\mathcal{D}}, \widehat{\mathcal{D}}'$ be defined above, then for $\forall \delta > 0$, w.p. at least $1 - \delta$, for $\forall h \in \tilde{\mathcal{H}}$:

$$d_{\tilde{\mathcal{H}}}(\mathcal{D}, \mathcal{D}') \le d_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}, \widehat{\mathcal{D}}') + 4\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) + 6\sqrt{\log(4/\delta)/2n}.$$

Proof. By the triangular inequality of $d_{\tilde{\mathcal{H}}}(\cdot, \cdot)$, we have:

$$d_{\tilde{\mathcal{H}}}(\mathcal{D}, \mathcal{D}') \le d_{\tilde{\mathcal{H}}}(\mathcal{D}, \widehat{\mathcal{D}}) + d_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}, \widehat{\mathcal{D}}') + d_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}', \mathcal{D}').$$

Now with Lemma 4.4.4, we know that with probability $\ge 1 - \delta/2$, we have:

$$d_{\tilde{\mathcal{H}}}(\mathcal{D}, \widehat{\mathcal{D}}) \le 2\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) + 3\sqrt{\log(4/\delta)/2n}.$$

Similarly, with probability $\ge 1 - \delta/2$, the following inequality also holds:

$$d_{\tilde{\mathcal{H}}}(\mathcal{D}', \widehat{\mathcal{D}}') \le 2\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) + 3\sqrt{\log(4/\delta)/2n}.$$

A union bound to combine the above two inequalities then finishes the proof. ■

Theorem 4.4.2. Let $\langle \mathcal{D}_S, f_S \rangle$ and $\langle \mathcal{D}_T, f_T \rangle$ be the source and target domains, and let $\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T$ be the empirical source and target distributions constructed from sample $\mathbf{S} = \{\mathbf{S}_S, \mathbf{S}_T\}$, each of size n. Then for any $\mathcal{H} \subseteq [0,1]^{\mathcal{X}}$ and $\forall h \in \mathcal{H}$:

$$\begin{aligned}\varepsilon_T(h) \leq \widehat{\varepsilon}_S(h) &+ d_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}_S, \widehat{\mathcal{D}}_T) + 2\mathrm{Rad}_{\mathbf{S}}(\mathcal{H}) + 4\mathrm{Rad}_{\mathbf{S}}(\tilde{\mathcal{H}}) \\ &+ \min\{\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|], \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]\} \\ &+ O\left(\sqrt{\log(1/\delta)/n}\right),\end{aligned}$$

where $\tilde{\mathcal{H}} := \{\mathrm{sgn}(|h(\mathbf{x}) - h'(\mathbf{x})| - t) | h, h' \in \mathcal{H}, t \in [0,1]\}$.

Proof. By Theorem 4.4.1, the following inequality holds:

$$\varepsilon_T(h) \leq \varepsilon_S(h) + d_{\tilde{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T) + \min\{\mathbb{E}_{\mathcal{D}_S}[|f_S - f_T|], \mathbb{E}_{\mathcal{D}_T}[|f_S - f_T|]\}.$$

To get probabilistic bounds for both $\varepsilon_S(h)$ and $d_{\tilde{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T)$, we apply Lemma 4.4.3 and Lemma 4.4.5, respectively. The final step, again, is to use a union bound to combine all the inequalities above, which completes the proof. ■

Lemma 4.4.6. Let $\mathcal{D}_S^Z$ and $\mathcal{D}_T^Z$ be two distributions over $\mathcal{Z}$ and let $\mathcal{D}_S^Y$ and $\mathcal{D}_T^Y$ be the induced distributions over $\mathcal{Y}$ by function $h : \mathcal{Z} \mapsto \mathcal{Y}$, then

$$d_{\mathrm{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y) \leq d_{\mathrm{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z). \tag{4.3}$$

Proof. Let B be a uniform random variable taking value in $\{0,1\}$ and let the random variable Y_B with distribution $\mathcal{D}_B^Y$ (resp. Z_B with distribution $\mathcal{D}_B^Z$) be the mixture of $\mathcal{D}_S^Y$ and $\mathcal{D}_T^Y$ (resp. $\mathcal{D}_S^Z$ and $\mathcal{D}_T^Z$) according to B. We know that:

$$D_{\mathrm{JS}}(\mathcal{D}_S^Z \parallel \mathcal{D}_T^Z) = I(B; Z_B), \quad \text{and} \quad D_{\mathrm{JS}}(\mathcal{D}_S^Y \parallel \mathcal{D}_T^Y) = I(B; Y_B). \tag{4.7}$$

Since $\mathcal{D}_S^Y$ (resp. $\mathcal{D}_T^Y$) is induced by the function $h : \mathcal{Z} \mapsto \mathcal{Y}$ from $\mathcal{D}_S^Z$ (resp. $\mathcal{D}_T^Z$), by linearity, we also have $\mathcal{D}_B^Y$ is induced by h from $\mathcal{D}_B^Z$. Hence $Y_B = h(Z_B)$ and the following Markov chain holds:

$$B \to Z_B \to Y_B.$$

Apply the data processing inequality (Lemma 2.6.4), we have

$$D_{\mathrm{JS}}(\mathcal{D}_S^Z \parallel \mathcal{D}_T^Z) = I(B; Z_B) \geq I(B; Y_B) = D_{\mathrm{JS}}(\mathcal{D}_S^Y \parallel \mathcal{D}_T^Y).$$

Taking square root on both sides of the above inequality completes the proof. ■

Lemma 4.4.7. Let $Y = f(X) \in \{0,1\}$ where $f(\cdot)$ is the labeling function and $\hat{Y} = h(g(X)) \in \{0,1\}$ be the prediction function, then $d_{\mathrm{JS}}(\mathcal{D}^Y, \mathcal{D}^{\hat{Y}}) \leq \sqrt{\varepsilon(h \circ g)}$.

Proof.

$$\begin{aligned}
d_{\mathrm{JS}}(\mathcal{D}^Y,\mathcal{D}^{\hat{Y}}) &= \sqrt{D_{\mathrm{JS}}(\mathcal{D}^Y,\mathcal{D}^{\hat{Y}})} \\
&\le \sqrt{\|\mathcal{D}^Y-\mathcal{D}^{\hat{Y}}\|_1/2} && \text{(Lemma 2.6.3)} \\
&= \sqrt{\left(|\Pr(Y=0)-\Pr(\hat{Y}=0)|+|\Pr(Y=1)-\Pr(\hat{Y}=1)|\right)/2} \\
&= \sqrt{|\Pr(Y=1)-\Pr(\hat{Y}=1)|} \\
&= \sqrt{|\mathbb{E}_X[f(X)]-\mathbb{E}_X[h(g(X))]|} \\
&\le \sqrt{\mathbb{E}_X[|f(X)-h(g(X))|]} \\
&= \sqrt{\varepsilon(h\circ g)}
\end{aligned}$$

■

Lemma 4.4.8. Suppose the Markov chain $X \xrightarrow{g} Z \xrightarrow{h} \hat{Y}$ holds, then

$$d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_T^Y) \le d_{\mathrm{JS}}(\mathcal{D}_S^Z,\mathcal{D}_T^Z) + \sqrt{\varepsilon_S(h\circ g)} + \sqrt{\varepsilon_T(h\circ g)}.$$

Proof. Since $X \xrightarrow{g} Z \xrightarrow{h} \hat{Y}$ forms a Markov chain, by Lemma 4.4.6, the following inequality holds:

$$d_{\mathrm{JS}}(\mathcal{D}_S^{\hat{Y}},\mathcal{D}_T^{\hat{Y}}) \le d_{\mathrm{JS}}(\mathcal{D}_S^Z,\mathcal{D}_T^Z).$$

On the other hand, since $d_{\mathrm{JS}}(\cdot,\cdot)$ is a distance metric, we also have:

$$d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_T^Y) \le d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_S^{\hat{Y}}) + d_{\mathrm{JS}}(\mathcal{D}_S^{\hat{Y}},\mathcal{D}_T^{\hat{Y}}) + d_{\mathrm{JS}}(\mathcal{D}_T^{\hat{Y}},\mathcal{D}_T^Y) \le d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_S^{\hat{Y}}) + d_{\mathrm{JS}}(\mathcal{D}_S^Z,\mathcal{D}_T^Z) + d_{\mathrm{JS}}(\mathcal{D}_T^{\hat{Y}},\mathcal{D}_T^Y).$$

Applying Lemma 4.4.7 to both $d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_S^{\hat{Y}})$ and $d_{\mathrm{JS}}(\mathcal{D}_T^{\hat{Y}},\mathcal{D}_T^Y)$ then finishes the proof. ■

Theorem 5.2.1. Suppose the condition in Lemma 4.4.8 holds and $d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_T^Y) \ge d_{\mathrm{JS}}(\mathcal{D}_S^Z,\mathcal{D}_T^Z)$, then:

$$\varepsilon_S(h\circ g) + \varepsilon_T(h\circ g) \ge \frac{1}{2}\left(d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_T^Y) - d_{\mathrm{JS}}(\mathcal{D}_S^Z,\mathcal{D}_T^Z)\right)^2.$$

Proof. In view of the result in Theorem 4.4.8, applying the AM-GM inequality, we have:

$$\sqrt{\varepsilon_S(h\circ g)} + \sqrt{\varepsilon_T(h\circ g)} \le \sqrt{2\left(\varepsilon_S(h\circ g)+\varepsilon_T(h\circ g)\right)}.$$

Now since $d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_T^Y) \ge d_{\mathrm{JS}}(\mathcal{D}_S^Z,\mathcal{D}_T^Z)$, simple algebra shows

$$\varepsilon_S(h\circ g) + \varepsilon_T(h\circ g) \ge \frac{1}{2}\left(d_{\mathrm{JS}}(\mathcal{D}_S^Y,\mathcal{D}_T^Y) - d_{\mathrm{JS}}(\mathcal{D}_S^Z,\mathcal{D}_T^Z)\right)^2.$$

■

4.7 Conclusion

In this chapter we theoretically and empirically study the important problem of learning invariant representations for domain adaptation. We show that learning an invariant representation and achieving a small source error is not enough to guarantee target generalization. We then prove both upper and lower bounds for the target and joint errors, which directly translate to sufficient and necessary conditions for the success of adaptation. We believe our results take an important step towards understanding deep domain adaptation, and also stimulate future work on the design of stronger deep domain adaptation algorithms that align conditional distributions. Another interesting direction for future work is to characterize what properties the feature transformation function should have in order to decrease the conditional shift. It is also worth investigating under which conditions the label distributions can be aligned without explicit labeled data from the target domain.

Chapter 5

Domain Adaptation with Conditional Distribution Matching under Generalized Label Shift

In this chapter, we extend a recent upper-bound on the performance of adversarial domain adaptation to multi-class classification and more general discriminators. We then propose *generalized label shift* (GLS) as a way to improve robustness against mismatched label distributions. GLS states that, conditioned on the label, there exists a representation of the input that is invariant between the source and target domains. Under GLS, we provide theoretical guarantees on the transfer performance of any classifier. We also devise necessary and sufficient conditions for GLS to hold. The conditions are based on the estimation of the relative class weights between domains and on an appropriate reweighting of samples. Guided by our theoretical insights, we modify three widely used algorithms, JAN, DANN and CDAN and evaluate their performance on standard domain adaptation tasks where our method outperforms the base versions. We also demonstrate significant gains on artificially created tasks with large divergences between their source and target label distributions.

5.1 Introduction

In spite of impressive successes, most deep learning models (Goodfellow et al., 2017) rely on huge amounts of labelled data and their features have proven brittle to distribution shifts (McCoy et al., 2019; Yosinski et al., 2014). Building more robust models, that learn from fewer samples and/or generalize better out-of-distribution is the focus of many recent works (Arjovsky et al., 2019; Bachman et al., 2019; Yaghoobzadeh et al., 2019). The research direction of interest to this paper is that of domain adaptation, which aims at learning features that transfer well between domains.

We focus in particular on the unsupervised domain adaptation setting (UDA), where the algorithm has access to labelled samples from a source domain and unlabelled data from a target domain. Its objective is to train a model that generalizes well to the target domain. Building on advances in adversarial learning (Goodfellow et al., 2014), adversarial domain adaptation (ADA) leverages the use of a discriminator to learn an intermediate representation that is invariant between the source and target domains. Simultaneously, the representation is paired with a classifier, trained to perform well on the source domain (Ganin et al., 2016; Liu et al., 2019; Tzeng et al., 2017; Zhao et al., 2018b). ADA is rather successful on a variety of tasks, however, recent work has proven an upper bound on the performance of existing algorithms when source and target domains have mismatched label distributions (Zhao et al., 2019h). Label, or prior probability, shift is a property of two domains for which the marginal label distributions differ, but the conditional distributions of input given label stay the same across domains (Storkey, 2009; Zhang et al., 2015b).

In this chapter, we study domain adaptation under mismatched label distributions and design methods that are robust in that setting. Our contributions are the following. First, we extend the upper bound by Zhao et al. (2019h) to k-class classification and to conditional domain adversarial networks, a recently introduced domain adaptation algorithm (Long et al., 2018). Second, we introduce *generalized label shift* (GLS), a broader version of the standard label shift where conditional invariance between source and target domains is placed in representation rather than input space. Third, we derive performance guarantees for algorithms that seek to enforce GLS via learnt feature transformations, in the form of upper bounds on the error gap and the joint error of the classifier on the source and target domains. Those performance guarantees suggest principled modifications to ADA to improve its robustness to mismatched label distributions. The modifications rely on estimating the class ratios between source and target domains and use those as importance weights in the adversarial and classification objectives. The importance weights estimation is performed using a method from Lipton et al. (2018). Following the theoretical insights, we devise three new algorithms, based on DANNs (Ganin et al., 2016), JANs (Long et al., 2017) and CDANs (Long et al., 2018). We apply our variants to artificial UDA tasks with large divergences between label distributions, and demonstrate significant performance gains compared to the algorithms' base versions. Finally, we evaluate them on standard domain adaptation tasks (for which the divergence between label distribution is rather limited) and show improved performance as well.

5.2 Preliminaries

Notation and Setup In this chapter we focus on the general k-class classification problem. We use $\mathcal{X}$ and $\mathcal{Y}$ to denote the input and output space, respectively. Similarly, $\mathcal{Z}$ stands for the representation space induced from $\mathcal{X}$ by a feature transformation $g : \mathcal{X} \mapsto \mathcal{Z}$. Accordingly, we use X, Y, Z to denote random variables which take values in $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$. In this work, *domain* corresponds to a distribution on the input space $\mathcal{X}$ and output space $\mathcal{Y}$, and we use $\mathcal{D}_S$ (resp. $\mathcal{D}_T$) to denote the source (resp. target) domain. Noticeably, this corresponds to a stochastic setting, which is stronger than the deterministic one studied in Ben-David et al. (2007, 2010); Zhao et al. (2019h). A *hypothesis* is a function $h : \mathcal{X} \to [k]$. The *error* of a hypothesis h under distribution $\mathcal{D}_S$ is defined as: $\varepsilon_S(h) := \Pr_{\mathcal{D}_S}(h(X) \neq Y)$, i.e., the probability that h disagrees with Y under $\mathcal{D}_S$.

Domain Adaptation via Invariant Representations For source ($\mathcal{D}_S$) and target ($\mathcal{D}_T$) domains, we use $\mathcal{D}_S^X$, $\mathcal{D}_T^X$, $\mathcal{D}_S^Y$ and $\mathcal{D}_T^Y$ to denote the marginal data and label distributions. In UDA, the algorithm has access to n labeled points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$ and m unlabeled points $\{\mathbf{x}_j\}_{j=1}^m \in \mathcal{X}^m$ sampled i.i.d. from the source and target domains. Inspired by Ben-David et al. (2010), a common approach is to learn representations invariant to the domain shift. Letting $g : \mathcal{X} \mapsto \mathcal{Z}$ be a feature transformation and $h : \mathcal{Z} \mapsto \mathcal{Y}$ a hypothesis on the feature space, the goal of domain invariant representations (Ganin et al., 2016; Tzeng et al., 2017; Zhao et al., 2018c) is to find a function g that induces similar distributions on $\mathcal{D}_S$ and $\mathcal{D}_T$. Simultaneously, g is required to preserve rich information about the target task so that $\varepsilon_S(h \circ g)$ is small. The above process results in the following Markov chain:

$$X \xrightarrow{g} Z \xrightarrow{h} \widehat{Y}, \tag{5.1}$$

with $\widehat{Y} = h(g(X))$. We let $\mathcal{D}_S^Z$, $\mathcal{D}_T^Z$, $\mathcal{D}_S^{\widehat{Y}}$ and $\mathcal{D}_T^{\widehat{Y}}$ denote the pushforwards of $\mathcal{D}_S$ and $\mathcal{D}_T$ by g and $h \circ g$. Invariance in feature space is defined as minimizing a distance or divergence between the source and target feature distributions.

Table 5.2.1: Common assumptions in the domain adaptation literature.

Covariate Shift	**Label Shift**
$\mathcal{D}_S^X \neq \mathcal{D}_T^X$	$\mathcal{D}_S^Y \neq \mathcal{D}_T^Y$
$\forall \mathbf{x} \in \mathcal{X}, \mathcal{D}_S(Y \mid X = \mathbf{x}) = \mathcal{D}_T(Y \mid X = \mathbf{x})$	$\forall y \in \mathcal{Y}, \mathcal{D}_S(X \mid Y = y) = \mathcal{D}_T(X \mid Y = y)$

Adversarial Domain Adaptation Invariance is often attained by training a discriminator $d : \mathcal{Z} \mapsto [0, 1]$ to predict whether a representation z is from the source or target domain. g is then trained both to maximize the discriminator loss and to minimize the classification loss of $h \circ g$ on the source domain (h is also trained with the latter objective).

This leads in particular to domain-adversarial neural networks (Ganin et al., 2016, DANN), where g, h and d are parameterized with neural networks: g_θ, h_ϕ and d_ψ. d_ψ outputs the probability to be from the source domain, while h_ϕ outputs the probability to belong to each class. The discriminator loss $\mathcal{L}_{DA}$ and classification loss $\mathcal{L}_C$ are simply cross-entropies. d_ψ, resp. g_θ, are then trained to minimize, resp. maximize $\mathcal{L}_{DA}$, while h_ϕ and g_θ minimize $\mathcal{L}_C$ (see Algo. 2 and Appendix 5.A.4 for details).

Building on DANN, conditional domain adversarial networks (Long et al., 2018, CDAN) use the same adversarial paradigm. However, the discriminator now takes as input the outer product, for a given x, between the predictions of the network $h(g(x))$ and its representation $g(x)$. In other words, d acts on the outer product:

$$h \otimes g(x) := (h_1(g(x)) \cdot g(x), \dots, h_k(g(x)) \cdot g(x))$$

rather than on $g(x)$ (where h_i denotes the i-th element of vector h). We now highlight a limitation of DANNs and CDANs.

An Information-Theoretic Lower Bound Let D_{JS} denote the Jensen-Shanon divergence between two distributions (see Appendix 5.5.1 for details), and let $\widetilde{Z}$ correspond to Z (for DANN) or to $\widehat{Y} \otimes Z$ (for CDAN). The following theorem gives a lower bound on the joint error of the classifier on the source and target domains:

Theorem 5.2.1. Suppose that the Markov chain in (5.1) holds, and that $D_{\text{JS}}(\mathcal{D}_S^Y \parallel \mathcal{D}_T^Y) \geq D_{\text{JS}}(\mathcal{D}_S^{\widetilde{Z}} \parallel \mathcal{D}_T^{\widetilde{Z}})$, then:

$$\varepsilon_S(h \circ g) + \varepsilon_T(h \circ g) \geq \frac{1}{2}\left(\sqrt{D_{\text{JS}}(\mathcal{D}_S^Y \parallel \mathcal{D}_T^Y)} - \sqrt{D_{\text{JS}}(\mathcal{D}_S^{\widetilde{Z}} \parallel \mathcal{D}_T^{\widetilde{Z}})}\right)^2.$$

Remark Remarkably, the above lower bound is algorithm-independent. It is also a population-level result and holds asymptotically with increasing data; large data does not help. Zhao et al. (2019h) prove the theorem for $k = 2$ and $\widetilde{Z} = Z$, *i.e.*, for DANN on binary classification. We extend it to CDAN and arbitrary k (see Appendix 5.5.3 for the proof). Assuming that label distributions differ between source and target domains, the lower bound in Theorem 5.2.1 says that:

> *For both DANN and CDAN, the better the alignment of marginal feature distributions, the worse the sum of errors on source and target domains.*

Notably, for an invariant representation ($D_{\text{JS}}(\mathcal{D}_S^{\widetilde{Z}}, \mathcal{D}_T^{\widetilde{Z}}) = 0$) with no source error, the target error will be larger than $D_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y)/2$. Put another way, algorithms learning invariant representations and minimizing the source empirical risk are fundamentally flawed when marginal label distributions differ between source and target domains.

Intuitively, CDAN also suffers from an intrinsic lower bound because while its discriminator d takes into account the predicted output distribution, $\widehat{Y}$ is still a function of X[1]. All the information available to the discriminator comes from X. From an information-theoretic perspective, to circumvent the above tradeoff between distribution alignment and target error minimization, it is necessary to incorporate the ground-truth label distributions ($\mathcal{D}_S^Y$ and $\mathcal{D}_T^Y$) into the discriminator.

Common Assumptions to Tackle Domain Adaptation Domain adaptation requires assumptions about the data to be possible. Two common ones are *covariate shift* and *label shift*. They correspond to different ways of decomposing the joint distribution over $X \times Y$, as detailed in Table 5.2.1. From the perspective of representation learning, it has been demonstrated that covariate shift is not robust to feature transformation, and can lead to an effect called negative transfer (Zhao et al., 2019h). At the same time, label shift clearly fails in most practical applications. Consider, for instance, transferring knowledge from synthetic to real images (Visda, 2017): the supports of the input distributions are actually disjoint. In this chapter, we focus on label shift and propose a solution to the above problem.

5.3 Main Results

In light of the limitations of existing assumptions, (e.g. covariate shift and label shift), we propose *generalized label shift* (GLS), a relaxation of label shift that substantially improves its applicability. We first discuss some of its properties and explain why the assumption is favorable in domain adaptation based on representation learning. Motivated by GLS, we then present a novel error decomposition theorem that directly suggests a bound minimization framework for domain adaptation. The framework is naturally compatible with $\mathcal{F}$-integral probability metrics (Müller, 1997, $\mathcal{F}$-IPM) and generates a family of domain

[1] Thm.5.2.1 actually holds for any $\tilde{Z}$ s.t. $\widehat{Y} = \tilde{h}(\tilde{Z})$, see Appendix 5.5.3.

adaptation algorithms by choosing various function classes $\mathcal{F}$. In a nutshell, the proposed framework applies Lipton et al. (2018)'s method-of-moments to estimate the importance weight $\mathbf{w}$ of the *marginal label distributions* by solving a quadratic program (QP), and then uses $\mathbf{w}$ to align the weighted source feature distribution with the target feature distribution.

5.3.1 Generalized Label Shift

Definition 5.3.1 (Generalized Label Shift, GLS)**.** A representation $Z = g(X)$ satisfies GLS if

$$\mathcal{D}_S(Z \mid Y = y) = \mathcal{D}_T(Z \mid Y = y), \ \forall y \in \mathcal{Y}. \tag{5.2}$$

First, we note that when g is the identity map, i.e. $Z = X$, the above definition of GLS reduces to the original label shift assumption. Next, GLS is always achievable for any distribution pair $(\mathcal{D}_S, \mathcal{D}_T)$: any constant function $g \equiv c \in \mathbb{R}$ satisfies the above definition. The most important property of GLS is arguably that, unlike label shift, the above definition is compatible with a perfect classifier in the noiseless case. More specifically, suppose there exists a ground-truth labeling function h^* such that $Y = h^*(X)$; then h^* satisfies GLS. As a comparison, without conditioning on $Y = y$, the optimal labeling function does not satisfy $\mathcal{D}_S(h^*(X)) = \mathcal{D}_T(h^*(X))$ if the marginal label distributions are different across domains. This observation is also consistent with the lower bound in Theorem 5.2.1, which holds for arbitrary marginal label distributions.

GLS imposes label shift in the feature space $\mathcal{Z}$ instead of the original input space $\mathcal{X}$. Conceptually, although samples from the same classes in the source and target domain can be dramatically different, the hope is to find an intermediate representation for both domains in which samples from a given class look similar to one another. Taking digit classification as an example and assuming the feature variable Z corresponds to the contour of a digit, it is possible that by using different contour extractors for e.g. MNIST and USPS, those contours look roughly the same in both domains. Technically, GLS can be facilitated by having separate representation extractors g_S and g_T for source and target[2] (Bousmalis et al., 2016; Tzeng et al., 2017).

5.3.2 An Error Decomposition Theorem based on GLS

Before delving into practical ways to enforce GLS, we provide performance guarantees for models that satisfy it, in the form of upper bounds on the error gap and on the joint error between source and target domains. The bound requires the following two concepts:

Definition 5.3.2 (Balanced Error Rate)**.** The *balanced error rate* (BER) of predictor $\widehat{Y}$ on domain $\mathcal{D}_S$ is:

$$\mathrm{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y) := \max_{j \in [k]} \mathcal{D}_S(\widehat{Y} \neq Y \mid Y = j). \tag{5.3}$$

Definition 5.3.3 (Conditional Error Gap)**.** Given a joint distribution $\mathcal{D}$, the *conditional error gap* of a classifier $\widehat{Y}$ is $\Delta_{\mathrm{CE}}(\widehat{Y}) := \max_{y \neq y' \in \mathcal{Y}^2} \ |\mathcal{D}_S(\widehat{Y} = y' \mid Y = y) - \mathcal{D}_T(\widehat{Y} = y' \mid Y = y)|$.

When GLS and the Markov chain in (5.1) hold, the conditional error gap is equal to 0. The next theorem gives an upper bound on the error gap between source and target; it can also be used to obtain a generalization upper bound on the target risk.

Theorem 5.3.1. (Error Decomposition Theorem) For any classifier $\widehat{Y} = (h \circ g)(X)$,

$$|\varepsilon_S(h \circ g) - \varepsilon_T(h \circ g)| \leq \ \|\mathcal{D}_S^Y - \mathcal{D}_T^Y\|_1 \cdot \mathrm{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y) + 2(k-1)\Delta_{\mathrm{CE}}(\widehat{Y}),$$

where $\|\mathcal{D}_S^Y - \mathcal{D}_T^Y\|_1 := \sum_{i=1}^{k} |\mathcal{D}_S(Y = i) - \mathcal{D}_T(Y = i)|$ is the L_1 distance between $\mathcal{D}_S^Y$ and $\mathcal{D}_T^Y$.

[2] For $\mathbf{x} \in \mathcal{D}_S$ (resp. $\mathbf{x} \in \mathcal{D}_T$), $z = g_S(\mathbf{x})$ (resp. $z = g_T(\mathbf{x})$).

Remark The upper bound in Theorem 5.3.1 provides a way to decompose the error gap between source and target domains. Additionally, with such a bound, we can immediately obtain a generalization bound of the target risk $\varepsilon_T(h)$. The upper bound contains two terms. The first one, $\|\mathcal{D}_S^Y - \mathcal{D}_T^Y\|_1$, measures the distance between the marginal label distributions across domains, and is a constant that only depends on the adaptation problem itself. It also contains BER, a reweighted classification performance on the source domain. The second term, $\Delta_{\text{CE}}(\widehat{Y})$, by definition, measures the distance between the family of conditional distributions $\widehat{Y} \mid Y$. In other words, the above upper bound is oblivious to the optimal labeling functions in feature space. This is in sharp contrast with upper bounds from previous work (Ben-David et al., 2010, Theorem 2), (Zhao et al., 2019h, Theorem 4.1), which essentially decompose the error gap in terms of the distance between the marginal feature distributions ($\mathcal{D}_S^Z$, $\mathcal{D}_T^Z$) and the optimal labeling functions (f_S^Z, f_T^Z). Because the optimal labeling function in feature space depends on Z and is unknown in practice, such decomposition is not very informative. As a comparison, Theorem 5.3.1 provides a decomposition orthogonal to previous results and does not require knowledge about unknown optimal labeling functions in feature space.

Notably, the balanced error rate, $\text{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y)$, only depends on samples from the source domain, hence we can seek to minimize it in order to minimize the upper bound. Furthermore, using a data-processing argument, the conditional error gap $\Delta_{\text{CE}}(\widehat{Y})$, can be minimized by aligning the conditional feature distributions across domains. Putting everything together, the upper bound on the error difference between source and target domains suggests that, in order to minimize the error gap, it suffices to align the conditional distributions $Z \mid Y = y$ while simultaneously minimizing the balanced error rate. In fact, under the assumption that the conditional distributions are perfectly aligned (i.e., under GLS), we can prove a stronger result, guaranteeing that the joint error is small:

Theorem 5.3.2. If $Z = g(X)$ satisfies GLS, then for any $h : \mathcal{Z} \to \mathcal{Y}$ and letting $\widehat{Y} = h(Z)$ be the predictor, we have $\varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) \leq 2\text{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y)$.

Remark Theorems 5.3.1 and 5.3.2 imply that if the conditional feature distributions are aligned, then both the source and target error will be bounded by $\text{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y)$. This suggests seeking models that simultaneously verify GLS and minimize $\text{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y)$. Since the balanced error rate only depends on the source domain, using labeled samples from the source domain is sufficient to minimize it.

5.3.3 Conditions for Generalized Label Shift

The main difficulty in applying a bound minimization algorithm inspired by Theorem 5.3.1 is that we do not have access to labels from the target domain in UDA[3], so we cannot directly align the conditional label distributions. Below, we provide a necessary condition for GLS that avoids the need to explicitly align the conditional feature distributions.

Definition 5.3.4. Assuming $\mathcal{D}_S(Y = y) > 0, \forall y \in \mathcal{Y}$, we let $\mathbf{w} \in \mathbb{R}^k$ denote the importance weights of the target and source label distributions:

$$\mathbf{w}_y := \frac{\mathcal{D}_T(Y = y)}{\mathcal{D}_S(Y = y)}, \quad \forall y \in \mathcal{Y}. \tag{5.4}$$

Given the importance weights vector, a necessary condition implied by GLS is expressed in the following lemma.

Lemma 5.3.1. Assuming $Z = g(X)$ satisfies GLS, then $\mathcal{D}_T(Z) = \sum_{y \in \mathcal{Y}} \mathbf{w}_y \cdot \mathcal{D}_S(Z, Y = y) =: \mathcal{D}_S^{\mathbf{w}}(Z)$.

[3] Though it could be used directly if we have a few target labels.

Compared to previous work that attempts to align $\mathcal{D}_T(Z)$ with $\mathcal{D}_S(Z)$ using adversarial discriminators (Ganin et al., 2016) or maximum mean discrepancy (MMD) (Long et al., 2015), Lemma 5.3.1 suggests that we should instead align $\mathcal{D}_T(Z)$ with the *reweighted* marginal distribution $\mathcal{D}_S^{\mathbf{w}}(Z)$.

Reciprocally, one may be interested to know when perfectly aligned target feature distribution and reweighted source feature distribution imply GLS. The following theorem gives a sufficient condition to answer this question:

Theorem 5.3.3. (Clustering structure implies sufficiency) Let $Z = g(X)$ such that $\mathcal{D}_T(Z) = \mathcal{D}_S^{\mathbf{w}}(Z)$. Assume $\mathcal{D}_T(Y = y) > 0, \forall y \in \mathcal{Y}$. If there exists a partition of $\mathcal{Z} = \cup_{y\in\mathcal{Y}}\mathcal{Z}_y$ such that $\forall y \in \mathcal{Y}$, $\mathcal{D}_S(Z \in \mathcal{Z}_y \mid Y = y) = \mathcal{D}_T(Z \in \mathcal{Z}_y \mid Y = y) = 1$, then $Z = g(X)$ satisfies GLS.

Remark Theorem 5.3.3 shows that if there exists a partition of the feature space such that instances with the same label are within the same component, then aligning the target feature distribution with the reweighted source feature distribution implies GLS. While this clustering assumption may seem strong, it is consistent with the goal of reducing classification error: if such a clustering exists, then there also exists a perfect predictor based on the feature $Z = g(X)$, i.e., the cluster index.

We now consider CDAN, an algorithm particularly well-suited for conditional alignment. As described in Section 5.2, the CDAN discriminator seeks to match $\mathcal{D}_S(\widehat{Y} \otimes Z)$ with $\mathcal{D}_T(\widehat{Y} \otimes Z)$. This objective is very aligned with GLS: let us first assume for argument's sake that $\widehat{Y}$ is a perfect classifier on both domains. For any sample (x, y), $\hat{y} \otimes z$ is thus a matrix of 0s except on the y-th row, which contains z. When label distributions match, the effect of fooling the discriminator will result in representations such that the matrices $\widehat{Y} \otimes Z$ are equal on the source and target domains. In other words, the model is such that $Z \mid Y$ match: it verifies GLS (see Thm. 5.3.4 below with $\mathbf{w} = 1$). On the other hand, if the label distributions differ, fooling the discriminator actually requires mislabelling certain samples (a fact quantified in Thm. 5.2.1). We now provide a sufficient condition for GLS under a modified CDAN objective (see proofs in Appendix 5.5.8).

Theorem 5.3.4. Let $\widehat{Y} = h(Z)$, $\gamma := \min_{y\in\mathcal{Y}} \mathcal{D}_T(Y = y)$ and $\mathbf{w}_M := \max_{y\in\mathcal{Y}} \mathbf{w}_y$. For $\tilde{Z} = \hat{Y} \otimes Z$, we have:

$$\max_{y\in\mathcal{Y}} d_{\text{TV}}(\mathcal{D}_S(Z \mid Y = y), \mathcal{D}_T(Z \mid Y = y)) \leq \frac{1}{\gamma}\left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) + \sqrt{2D_{\text{JS}}(\mathcal{D}_S^{\mathbf{w}}(\tilde{Z}), \mathcal{D}_T(\tilde{Z}))}\right).$$

Theorem 5.3.4 suggests that CDANs should match $\mathcal{D}_S^{\mathbf{w}}(\widehat{Y} \otimes Z)$ with $\mathcal{D}_T(\widehat{Y} \otimes Z)$ to make them robust to mismatched label distributions. In fact, the above upper bound not only applies to CDAN, but also to any algorithm that aims at learning domain-invariant representations[4].

Theorem 5.3.5. With the same notation as Thm. 5.3.4:

$$\max_{y\in\mathcal{Y}} d_{\text{TV}}(\mathcal{D}_S(Z \mid Y = y), \mathcal{D}_T(Z \mid Y = y)) \leq \frac{1}{\gamma} \times \left\{\inf_{\widehat{Y}}\left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y})\right) + \sqrt{8D_{\text{JS}}(\mathcal{D}_S^{\mathbf{w}}(Z), \mathcal{D}_T(Z))}\right\}.$$

Remark It is worth pointing out that Theorem 5.3.5 extends Theorem 5.3.3 by incorporating the clustering assumption as the optimal joint error that is achievable by any classifier based on the representations. In

[4] Thm. 5.3.4 does not stem from Thm. 5.3.5 since the LHS of Thm. 5.3.4 applies to Z, not to $\tilde{Z}$, and in its RHS, $\tilde{Z}$ depends on $\hat{Y}$.

particular, if the clustering structure assumption holds in Theorem 5.3.3, then the optimal joint error is 0, which means that the first term on the R.H.S of Thm 5.3.5 becomes 0, hence in this case aligning the reweighted feature distributions also implies GLS.

5.3.4 Estimating the Importance Weights w

Inspired by the moment matching technique to estimate **w** under label shift (Lipton et al., 2018), we propose a method to get **w** under GLS by solving a quadratic program (QP).

Definition 5.3.5. We let $\mathbf{C} \in \mathbb{R}^{|\mathcal{Y}|\times|\mathcal{Y}|}$ denote the confusion matrix of the classifier on the source domain and $\boldsymbol{\mu} \in \mathbb{R}^{|\mathcal{Y}|}$ the distribution of predictions on the target one, $\forall y, y' \in \mathcal{Y}$:

$$\mathbf{C}_{y,y'} := \mathcal{D}_S(\widehat{Y} = y, Y = y'), \qquad \boldsymbol{\mu}_y := \mathcal{D}_T(\widehat{Y} = y).$$

The following lemma is adapted from Lipton et al. (2018) to give a consistent estimate of **w** under GLS; its proof can be found in Appendix 5.5.9.

Lemma 5.3.2. If GLS is verified, and if the confusion matrix **C** is invertible, then $\mathbf{w} = \mathbf{C}^{-1}\boldsymbol{\mu}$.

The key insight from Lemma 5.3.2 is that, in order to estimate the importance vector **w** under GLS, we do not need access to labels from the target domain. It is however well-known that matrix inversion is numerically unstable, especially with finite sample estimates $\hat{\mathbf{C}}$ and $\hat{\boldsymbol{\mu}}$ of **C** and $\boldsymbol{\mu}_y$[5]. We propose to solve instead the following QP (written as $QP(\hat{\mathbf{C}}, \hat{\boldsymbol{\mu}})$), whose solution will be consistent if $\hat{\mathbf{C}} \to \mathbf{C}$ and $\hat{\boldsymbol{\mu}} \to \boldsymbol{\mu}$:

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & \frac{1}{2}\,\|\hat{\boldsymbol{\mu}} - \hat{\mathbf{C}}\mathbf{w}\|_2^2 \\ \text{subject to} \quad & \mathbf{w} \geq 0,\ \mathbf{w}^T\mathcal{D}_S(Y) = 1. \end{aligned} \tag{5.5}$$

The above QP can be efficiently solved in time $O(|\mathcal{Y}|^3)$, with $|\mathcal{Y}|$ small and constant. Furthermore, by construction, the solution of the above QP is element-wise non-negative, even with limited amounts of data to estimate **C** and $\boldsymbol{\mu}_y$.

5.3.5 $\mathcal{F}$-IPM for Distributional Alignment

In order to align the target feature distribution and the reweighted source feature distribution as suggested by Lemma 5.3.1, we now provide a general framework using the integral probability metric (Müller, 1997, IPM).

Definition 5.3.6. Let $\mathcal{F}$ be a family of real-value functions. The $\mathcal{F}$-IPM between two distributions $\mathcal{D}$ and $\mathcal{D}'$ is

$$d_{\mathcal{F}}(\mathcal{D}, \mathcal{D}') := \sup_{f\in\mathcal{F}} |\mathbb{E}_{X\sim\mathcal{D}}[f(X)] - \mathbb{E}_{X\sim\mathcal{D}'}[f(X)]|. \tag{5.6}$$

By approximating any function class $\mathcal{F}$ using parametrized models, e.g., neural networks, we obtain a general framework for domain adaptation by aligning reweighted source feature distribution and target feature distribution, i.e. by minimizing $d_{\mathcal{F}}(\mathcal{D}_T(\tilde{Z}), \mathcal{D}_S^{\mathbf{w}}(\tilde{Z}))$. In particular, by choosing $\mathcal{F} = \{f : \|f\|_\infty \leq 1\}$, $d_{\mathcal{F}}$ reduces to total variation and the definition (5.6) of IPM becomes the negative sum of Type-I and Type-II errors (up to a constant) in distinguishing between $\mathcal{D}$ and $\mathcal{D}'$. This leads to our first algorithm IWDAN (cf. Section 5.4.1), an improved variant DANN algorithm that also takes into account the difference between label distributions. Similarly, by instantiating $\mathcal{F}$ to be the set of bounded norm functions in a RKHS $\mathcal{H}$ (Gretton et al., 2012), we obtain maximum mean discrepancy methods, leading to IWJAN (cf. Section 5.4.1), a variant of JAN (Long et al., 2017) for UDA. Below, we provide a comprehensive empirical evaluation of these variants.

[5] In fact, $\hat{\mathbf{w}} = \hat{\mathbf{C}}^{-1}\hat{\boldsymbol{\mu}}$ is not even necessarily non-negative.

Algorithm 2 Importance-Weighted Domain Adaptation

```
1:  Input: source data $(x_S, y_S)$, target data $x_T$, representation $g_\theta$, classifier $h_\phi$ and discriminator $d_\psi$
2:  Input: epochs $E$, batches per epoch $B$, batch size $s$
3:  Initialize $\mathbf{w}_1 = 1$
4:  for $t = 1$ to $E$ do
5:      Initialize $\hat{\mathbf{C}} = 0, \hat{\boldsymbol{\mu}} = 0$
6:      for $b = 1$ to $B$ do
7:          Sample batches $(x_S^i, y_S^i)$ and $(x_T^i)$
8:          Maximize $\mathcal{L}_{DA}^{\mathbf{w}_t}$ w.r.t. $\theta$, minimize $\mathcal{L}_{DA}^{\mathbf{w}_t}$ w.r.t. $\psi$ and minimize $\mathcal{L}_C^{\mathbf{w}_t}$ w.r.t. $\theta$ and $\phi$
9:          for $i = 1$ to $s$ do
10:             $\hat{\mathbf{C}}_{\cdot y_S^i} \leftarrow \hat{\mathbf{C}}_{\cdot y_S^i} + h_\phi(g_\theta(x_S^i))$ ($y_S^i$-th column)
11:             $\hat{\boldsymbol{\mu}} \leftarrow \hat{\boldsymbol{\mu}} + h_\phi(g_\theta(x_T^i))$
12:         end for
13:     end for
14:     $\hat{\mathbf{C}} \leftarrow \hat{\mathbf{C}}/sB$ and $\hat{\boldsymbol{\mu}} \leftarrow \hat{\boldsymbol{\mu}}/sB$
15:     $\mathbf{w}_{t+1} = \lambda \cdot QP(\hat{\mathbf{C}}, \hat{\boldsymbol{\mu}}) + (1-\lambda)\mathbf{w}_t$
16: end for
```

5.4 Practical Implementation

5.4.1 Algorithms

In the sections above, we have shown a way to estimate the reweighting vector $\mathbf{w}$ and defined necessary and sufficient conditions on the source and target feature distributions for GLS to hold. Together, they suggest simple algorithms based on representation learning:

1. Estimate $\mathbf{w}$ on the fly during training,
2. Align the feature distributions $\tilde{Z}$ of the target domain with the reweighted feature distribution of the source domain and,
3. Minimize the balanced error rate.

Computing $\mathbf{w}$ requires building estimators $\hat{\mathbf{C}}$ and $\hat{\boldsymbol{\mu}}$ from finite samples of $\mathbf{C}$ and $\boldsymbol{\mu}$. We do so by averaging during each successive epochs the predictions of the classifier on the source and target data. This step corresponds to the inner-most loop of Algorithm 2 (lines 9 to 12) and leads to estimations of $\hat{\mathbf{C}}$ and $\hat{\boldsymbol{\mu}}$. At the end of each epoch, the reweighting vector $\mathbf{w}$ is updated, and the estimators reset to 0. We have found empirically that using an exponential moving average of $\mathbf{w}$ performs better (line 15 in Alg. 2). The results of our experiments all use a factor $\lambda = 0.5$.

With the importance weights $\mathbf{w}$ in hand, we can now define our first algorithm, *Importance-Weighted Domain Adversarial Network* (IWDAN), that seeks to enforce the necessary condition in Lemma 5.3.1 (*i.e.* to align $\mathcal{D}_S^w(Z)$ and $\mathcal{D}_T(Z)$) using a discriminator. All it requires is to modify the DANN losses $\mathcal{L}_{DA}$ and $\mathcal{L}_C$. For batches (x_S^i, y_S^i) and (x_T^i) of size s, the weighted domain adaptation loss of IWDAN is:

$$\mathcal{L}_{DA}^{\mathbf{w}}(x_S^i, y_S^i, x_T^i; \theta, \psi) = -\frac{1}{s}\sum_{i=1}^{s} \mathbf{w}_{y_S^i} \log(d_\psi(g_\theta(x_S^i))) + \log(1 - d_\psi(g_\theta(x_T^i))). \tag{5.7}$$

We verify in the Appendix, Lemma 5.5.1, that the standard adversarial domain adaptation framework to $\mathcal{L}_{DA}^{\mathbf{w}}$ indeed minimizes the JSD between $\mathcal{D}_S^w(Z)$ and $\mathcal{D}_T(Z)$. Our second algorithm, *Importance-Weighted Joint Adaptation Networks* (IWJAN) is based on JAN (Long et al., 2017) and follows the reweighting

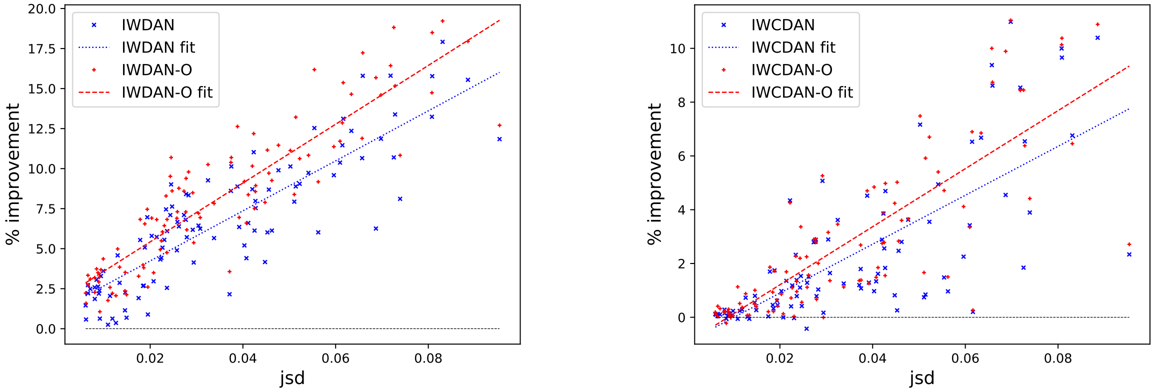


Figure 5.4.1: Gains of our algorithms versus their base versions for the 100 tasks described in Section 5.4 (IWDAN/IWCDAN on the left/right). The x-axis represents $D_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y)$, the JSD between label distributions. Lines represent linear fits. The mean improvements over DANN (resp. CDAN) for IWDAN and IWDAN-O (resp. IWCDAN and IWCDAN-O) are 6.55% and 8.14% (resp. 2.25% and 2.81%).

principle described in Section 5.3.5 with $\mathcal{F}$ a learnt RKHS (the exact JAN and IWJAN losses are specified in Appendix 5.A.4). Finally, our third algorithm, based on CDAN, is *Importance-Weighted Conditional Domain Adversarial Network* (IWCDAN). It follows Thm. 5.3.4, which suggests matching $\mathcal{D}_S^{\mathbf{w}}(\hat{Y} \otimes Z)$ with $\mathcal{D}_T(\hat{Y} \otimes Z)$. This can be done by replacing the standard adversarial loss in CDAN with the one on Eq. 5.7, where d_ψ takes as input $(h_\phi \circ g_\theta) \otimes g_\theta$ instead of g_θ. The classifier loss for our three variants is:

$$\mathcal{L}_C^{\mathbf{w}}(x_S^i, y_S^i; \theta, \phi) = -\frac{1}{s}\sum_{i=1}^{s} \frac{1}{k\mathcal{D}_S(Y=y)} \log(h_\phi(g_\theta(x_S^i))_{y_S^i}). \tag{5.8}$$

This reweighting is suggested by our theoretical analysis from Section 5.3, where we seek to minimize the balanced error rate $\text{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)$. We also define oracle versions, IWDAN-O, IWJAN-O and IWCDAN-O where the weights $\mathbf{w}$ used in the losses are not estimated but computed using the true target label distribution. It gives an idealistic version of the reweighting method, and allows to assess the soundness of GLS. IWDAN, IWJAN and IWCDAN correspond to Alg. 2 with their respective loss functions on line 8, the oracle versions simply use the true weights $\mathbf{w}$ instead of $\mathbf{w}_t$.

5.4.2 Experiments

We apply our three base algorithms, their importance weighted versions, and the associated oracles to domain adaptation problems from the following datasets: Digits (MNIST ↔ USPS (Dheeru and Karra Taniskidou, 2017; LeCun and Cortes, 2010)), Visda (2017), Office-31 (Saenko et al., 2010) and Office-Home (Venkateswara et al., 2017)[6]. All values are averages over 5 runs.

Performance vs $\mathbf{D_{JS}}$ First, we artificially generate a family of tasks from MNIST and USPS by considering various random subsets of the classes in either the source or target domain (see Appendix 5.A.5 for details). This results in 100 domain adaptation tasks, with Jensen-Shannon divergences varying between 0 and 0.1. Applying IWDAN and IWCDAN results in Figure 5.4.1. We see a clear correlation between the

[6] Except for JAN, which is not available on Digits.

Table 5.4.1: Average results on the various domains (Digits has 2 tasks, Visda 1, Office-31 6 and Office-Home 12). The prefix s denotes the experiment where the source domain is subsampled to increase $D_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y)$. Each number is a mean over 5 seeds, the subscript denotes the fraction of times (out of 5 *seeds* × *#tasks*) our algorithms outperform their base versions.

Method	Digits	*s*Digits	Visda	*s*Visda	O-31	*s*O-31	O-H	*s*O-H
No DA	77.17	75.67	48.39	49.02	77.81	75.72	56.39	51.34
DANN	93.15	83.24	61.88	52.85	82.74	76.17	59.62	51.83
IWDAN	**94.90**$_{100\%}$	**92.54**$_{100\%}$	**63.52**$_{100\%}$	**60.18**$_{100\%}$	**83.90**$_{87\%}$	**82.60**$_{100\%}$	**62.27**$_{97\%}$	**57.61**$_{100\%}$
IWDAN-O	95.27$_{100\%}$	94.46$_{100\%}$	64.19$_{100\%}$	62.10$_{100\%}$	85.33$_{97\%}$	84.41$_{100\%}$	64.68$_{100\%}$	60.87$_{100\%}$
CDAN	95.72	88.23	65.60	60.19	87.23	81.62	64.59	56.25
IWCDAN	**95.90**$_{80\%}$	**93.22**$_{100\%}$	**66.49**$_{60\%}$	**65.83**$_{100\%}$	**87.30**$_{73\%}$	**83.88**$_{100\%}$	**65.66**$_{70\%}$	**61.24**$_{100\%}$
IWCDAN-O	95.85$_{90\%}$	94.81$_{100\%}$	68.15$_{100\%}$	66.85$_{100\%}$	88.14$_{90\%}$	85.47$_{100\%}$	67.64$_{98\%}$	63.73$_{100\%}$
JAN	N/A	N/A	56.98	50.64	85.13	78.21	59.59	53.94
IWJAN	N/A	N/A	**57.56**$_{100\%}$	**57.12**$_{100\%}$	**85.32**$_{60\%}$	**82.61**$_{97\%}$	**59.78**$_{63\%}$	**55.89**$_{100\%}$
IWJAN-O	N/A	N/A	61.48$_{100\%}$	61.30$_{100\%}$	87.14$_{100\%}$	86.24$_{100\%}$	60.73$_{92\%}$	57.36$_{100\%}$

Table 5.4.2: Ablation study on the original and subsampled Digits data.

Method	Digits	sDigits
DANN	93.15	83.24
DANN + $\mathcal{L}_C^{\mathbf{w}}$	93.27	84.52
DANN + $\mathcal{L}_{DA}^{\mathbf{w}}$	**95.31**	**94.41**
IWDAN-O	**95.27**	**94.46**
CDAN	95.72	88.23
CDAN + $\mathcal{L}_C^{\mathbf{w}}$	95.65	91.01
CDAN + $\mathcal{L}_{DA}^{\mathbf{w}}$	95.42	93.18
IWCDAN-O	**95.85**	**94.81**

improvements provided by our algorithms and $D_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y)$, which is well aligned with Theorem 5.2.1. Moreover, IWDAN outperforms DANN on the 100 tasks and IWCDAN bests CDAN on 94. Even on small divergences, our algorithms do not suffer compared to their base versions.

Original Datasets Average results on each dataset are shown in Table 5.4.1 ((see Tables in Appendix 5.A.2 for the per-task breakdown). Our weighted version IWDAN outperforms the basic algorithm DANN by 1.75%, 1.64%, 1.16% and 2.65% on the Digits, Visda, Office-31 and Office-Home tasks respectively. Gains for IWCDAN are more limited, but still present: 0.18%, 0.89%, 0.07% and 1.07% respectively. This can be explained by the fact that, as mentioned above, CDAN already enforces a weak form of *GLS*. Gains for JAN are 0.58%, 0.19% and 0.19%. Beyond mean performance, we show the fraction of times (over all seeds and tasks) our variants outperform the original algorithms[7]. Even if gains are small, our variants provide consistent improvements. Additionally, the oracle versions show larger

[7] On the original datasets, the variance between seeds is larger than the difference between algorithms, making it uninformative.

improvements, which strongly supports enforcing GLS.

Subsampled datasets The original datasets have fairly balanced classes, making the JSD between source and target label distributions $D_{\text{JS}}(\mathcal{D}_S^Y \parallel \mathcal{D}_T^Y)$ rather small. To evaluate our algorithms on larger divergences, we arbitrarily modify the source domains on all the tasks above by considering only 30% of the samples coming from the first half of the classes. This results in much larger divergences. Performance is shown in Table 5.4.1. For IWDAN, we see gains of 9.3%, 7.33%, 6.43% and 5.58% on the digits, Visda, Office-31 and Office-Home datasets respectively. For IWCDAN, improvements are 4.99%, 5.64%, 2.26% and 4.99%, and IWJAN shows gains of 6.48%, 4.40% and 1.95%. Moreover, on all seeds and tasks but one, our variants outperform their base versions. Here as well, the oracles perform even better.

Ablation Study Our algorithms have two components, a weighted adversarial loss $\mathcal{L}_{DA}^{\mathbf{w}}$ and a weighted classification loss $\mathcal{L}_C^{\mathbf{w}}$. In Table 5.4.2, we augment DANN and CDAN using those losses separately (with the true weights). We observe that DANN benefits essentially from the reweighting of its adversarial loss $\mathcal{L}_{DA}^{\mathbf{w}}$, the classification loss has little effect. For CDAN, gains are essentially seen on the subsampled datasets. Both losses help, with a $+2\%$ extra gain for $\mathcal{L}_{DA}^{\mathbf{w}}$.

5.5 Proofs

In this section, we provide the theoretical material that completes the main text.

5.5.1 Definition

Definition 5.5.1. Let us recall that for two distributions $\mathcal{D}$ and $\mathcal{D}'$, the Jensen-Shannon (JSD) divergence $D_{\text{JS}}(\mathcal{D} \parallel \mathcal{D}')$ is defined as:

$$D_{\text{JS}}(\mathcal{D} \parallel \mathcal{D}') := \frac{1}{2} D_{\text{KL}}(\mathcal{D} \parallel \mathcal{D}_M) + \frac{1}{2} D_{\text{KL}}(\mathcal{D}' \parallel \mathcal{D}_M),$$

where $D_{\text{KL}}(\cdot \parallel \cdot)$ is the Kullback–Leibler (KL) divergence and $\mathcal{D}_M := (\mathcal{D} + \mathcal{D}')/2$.

5.5.2 Consistency of the Weighted Domain Adaptation Loss (5.7)

For the sake of conciseness, we verify here that the domain adaptation training objective does lead to minimizing the Jensen-Shannon divergence between the weighted feature distribution of the source domain and the feature distribution of the target domain.

Lemma 5.5.1. Let $p(x, y)$ and $q(x)$ be two density distributions, and $w(y)$ be a positive function such that $\int p(y)w(y)dy = 1$. Let $p^w(x) = \int p(x, y)w(y)dy$ denote the w-reweighted marginal distribution of x under p. The minimum value of

$$I(d) := \mathbb{E}_{(x,y)\sim p, x' \sim q}[-w(y)\log(d(x)) - \log(1 - d(x'))]$$

is $\log(4) - 2D_{\text{JS}}(p^w(x) \parallel q(x))$, and is attained for $d^*(x) = \frac{p^w(x)}{p^w(x)+q(x)}$.

Proof. We see that:

$$I(d) = -\iiint [w(y)\log(d(x)) + \log(1-d(x'))]p(x,y)q(x')dxdx'dy \tag{5.9}$$

$$= -\int [\int w(y)p(x,y)dy]\log(d(x)) + q(x)\log(1-d(x))dx \tag{5.10}$$

$$= -\int p^w(x)\log(d(x)) + q(x)\log(1-d(x))dx. \tag{5.11}$$

From the last line, we follow the exact method from Goodfellow et al. (2014) to see that point-wise in x the minimum is attained for $d^*(x) = \frac{p^w(x)}{p^w(x)+q(x)}$ and that $I(d^*) = \log(4) - 2D_{\text{JS}}(p^w(x) \parallel q(x))$. ■

Applying Lemma 5.5.1 to $\mathcal{D}_S(Z,Y)$ and $\mathcal{D}_T(Z)$ proves that the domain adaptation objective leads to minimizing $D_{\text{JS}}(\mathcal{D}_S^w(Z) \parallel \mathcal{D}_T(Z))$.

5.5.3 k-class information-theoretic lower bound

In this section, we prove Theorem 5.2.1 that extends previous result to the general k-class classification problem.

Theorem 5.2.1. Suppose that the Markov chain in (5.1) holds, and that $D_{\text{JS}}(\mathcal{D}_S^Y \parallel \mathcal{D}_T^Y) \geq D_{\text{JS}}(\mathcal{D}_S^{\widetilde{Z}} \parallel \mathcal{D}_T^{\widetilde{Z}})$, then:

$$\varepsilon_S(h \circ g) + \varepsilon_T(h \circ g) \geq \frac{1}{2}\left(\sqrt{D_{\text{JS}}(\mathcal{D}_S^Y \parallel \mathcal{D}_T^Y)} - \sqrt{D_{\text{JS}}(\mathcal{D}_S^{\widetilde{Z}} \parallel \mathcal{D}_T^{\widetilde{Z}})}\right)^2.$$

Proof. We essentially follow the proof from Zhao et al. (2019h), except for Lemmas 4.6 that needs to be adapted to the CDAN framework and Lemma 4.7 to k-class classification.

Lemma 4.6 from Zhao et al. (2019h) states that $D_{\text{JS}}(\mathcal{D}_S^{\widehat{Y}}, \mathcal{D}_T^{\widehat{Y}}) \leq D_{\text{JS}}(\mathcal{D}_S^Z, \mathcal{D}_T^Z)$, which covers the case $\widetilde{Z} = Z$.

When $\widetilde{Z} = \widehat{Y} \otimes Z$, let us first recall that we assume h or equivalently $\widehat{Y}$ to be a one-hot prediction of the class. We have the following Markov chain:

$$\begin{tikzcd} X \arrow[r, "g"] & Z \arrow[r, "\tilde{h}"] & \widetilde{Z} \arrow[r, "l"] & \widehat{Y}, \end{tikzcd}$$

where $\tilde{h}(z) = h(z) \otimes z$ and $l : \mathcal{Y} \otimes \mathcal{Z} \to \mathcal{Y}$ returns the index of the non-zero block in $\tilde{h}(z)$. There is only one such block since h is a one-hot, and its index corresponds to the class predicted by h. We can now apply the same proof than in Zhao et al. (2019h) to conclude that:

$$D_{\text{JS}}(\mathcal{D}_S^{\widehat{Y}}, \mathcal{D}_T^{\widehat{Y}}) \leq D_{\text{JS}}(\mathcal{D}_S^{\widetilde{Z}}, \mathcal{D}_T^{\widetilde{Z}}). \tag{5.12}$$

It essentially boils down to a data-processing argument: the discrimination distance between two distributions cannot increase after the same (possibly stochastic) channel (kernel) is applied to both. Here, the channel corresponds to the (potentially randomized) function l.

Remark Additionally, we note that the above inequality holds for *any* $\tilde{Z}$ such that $\widehat{Y} = l(\widetilde{Z})$ for a (potentially randomized) function l. This covers any and all potential combinations of representations at various layers of the deep net, including the last layer (which corresponds to its predictions $\widehat{Y}$).

Let us move to the second part of the proof. We wish to show that $D_{\mathrm{JS}}(\mathcal{D}^{Y},\mathcal{D}^{\widehat{Y}}) \le \varepsilon(h\circ g)$, where $\mathcal{D}$ can be either $\mathcal{D}_S$ or $\mathcal{D}_T$:

$$\begin{aligned}
2D_{\mathrm{JS}}(\mathcal{D}^{Y},\mathcal{D}^{\widehat{Y}}) &\le \|\mathcal{D}^{Y}-\mathcal{D}^{\widehat{Y}}\|_1 && \text{(Lin, 1991)}\\
&= \sum_{i=1}^{k}|\mathcal{D}(\widehat{Y}=i)-\mathcal{D}(Y=i)|\\
&= \sum_{i=1}^{k}|\sum_{j=1}^{k}\mathcal{D}(\widehat{Y}=i|Y=j)\mathcal{D}(Y=j)-\mathcal{D}(Y=i)|\\
&= \sum_{i=1}^{k}|\mathcal{D}(\widehat{Y}=i|Y=i)\mathcal{D}(Y=i)-\mathcal{D}(Y=i)+\sum_{j\neq i}\mathcal{D}(\widehat{Y}=i|Y=j)\mathcal{D}(Y=j)|\\
&\le \sum_{i=1}^{k}|\mathcal{D}(\widehat{Y}=i|Y=i)-1|\mathcal{D}(Y=i)+\sum_{i=1}^{k}\sum_{j\neq i}\mathcal{D}(\widehat{Y}=i|Y=j)\mathcal{D}(Y=j)\\
&= \sum_{i=1}^{k}\mathcal{D}(\widehat{Y}\neq Y|Y=i)\mathcal{D}(Y=i)+\sum_{j=1}^{k}\sum_{i\neq j}\mathcal{D}(\widehat{Y}=i|Y=j)\mathcal{D}(Y=j)\\
&= 2\sum_{i=1}^{k}\mathcal{D}(\widehat{Y}\neq Y|Y=i)\mathcal{D}(Y=i)=2\mathcal{D}(\widehat{Y}\neq Y)=2\varepsilon(h\circ g). && (5.13)
\end{aligned}$$

We can now apply the triangular inequality to $\sqrt{D_{\mathrm{JS}}}$, which is a distance metric (Endres and Schindelin, 2003), called the Jensen-Shannon distance. This gives us:

$$\begin{aligned}
\sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{Y},\mathcal{D}_T^{Y})} &\le \sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{Y},\mathcal{D}_S^{\widehat{Y}})}+\sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{\widehat{Y}},\mathcal{D}_T^{\widehat{Y}})}+\sqrt{D_{\mathrm{JS}}(\mathcal{D}_T^{\widehat{Y}},\mathcal{D}_T^{Y})}\\
&\le \sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{Y},\mathcal{D}_S^{\widehat{Y}})}+\sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{\widetilde{Z}},\mathcal{D}_T^{\widetilde{Z}})}+\sqrt{D_{\mathrm{JS}}(\mathcal{D}_T^{\widehat{Y}},\mathcal{D}_T^{Y})}\\
&\le \sqrt{\varepsilon_S(h\circ g)}+\sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{\widetilde{Z}},\mathcal{D}_T^{\widetilde{Z}})}+\sqrt{\varepsilon_T(h\circ g)}.
\end{aligned}$$

where we used Equation (5.12) for the second inequality and (5.13) for the third.

Finally, assuming that $D_{\mathrm{JS}}(\mathcal{D}_S^{Y},\mathcal{D}_T^{Y}) \ge D_{\mathrm{JS}}(\mathcal{D}_S^{\widetilde{Z}},\mathcal{D}_T^{\widetilde{Z}})$, we get:

$$\left(\sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{Y},\mathcal{D}_T^{Y})}-\sqrt{D_{\mathrm{JS}}(\mathcal{D}_S^{\widetilde{Z}},\mathcal{D}_T^{\widetilde{Z}})}\right)^2 \le \left(\sqrt{\varepsilon_S(h\circ g)}+\sqrt{\varepsilon_T(h\circ g)}\right)^2 \le 2\left(\varepsilon_S(h\circ g)+\varepsilon_T(h\circ g)\right).$$

which concludes the proof. ■

5.5.4 Proof of Theorem 5.3.1

To simplify the notation, we define the error gap $\Delta_\varepsilon(\widehat{Y})$ as follows:

$$\Delta_\varepsilon(\widehat{Y}) := |\varepsilon_S(\widehat{Y})-\varepsilon_T(\widehat{Y})|.$$

Also, in this case we use $\mathcal{D}_a,\ a\in\{S,T\}$ to mean the source and target distributions respectively. Before we give the proof of Theorem 5.3.1, we first prove the following two lemmas that will be used in the proof.

Lemma 5.5.2. Define $\gamma_{a,j} := \mathcal{D}_a(Y = j), \forall a \in \{S, T\}, \forall j \in [k]$, then $\forall \alpha_j, \beta_j \geq 0$ such that $\alpha_j + \beta_j = 1$, and $\forall i \neq j$, the following upper bound holds:

$$|\gamma_{S,j}\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \gamma_{T,j}\mathcal{D}_T(\widehat{Y} = i \mid Y = j)| \leq$$
$$|\gamma_{S,j} - \gamma_{T,j}| \cdot \Big(\alpha_j\mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j\mathcal{D}_T(\widehat{Y} = i \mid Y = j)\Big) + \gamma_{S,j}\beta_j\Delta_{\mathrm{CE}}(\widehat{Y}) + \gamma_{T,j}\alpha_j\Delta_{\mathrm{CE}}(\widehat{Y}).$$

Proof. To make the derivation uncluttered, define $\mathcal{D}_j(\widehat{Y} = i) := \alpha_j\mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j\mathcal{D}_T(\widehat{Y} = i \mid Y = j)$ to be the mixture conditional probability of $\widehat{Y} = i$ given $Y = j$, where the mixture weight is given by α_j and β_j. Then in order to prove the upper bound in the lemma, it suffices if we give the desired upper bound for the following term

$$\Big||\gamma_{S,j}\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \gamma_{T,j}\mathcal{D}_T(\widehat{Y} = i \mid Y = j)| - |(\gamma_{S,j} - \gamma_{T,j})\mathcal{D}_j(\widehat{Y} = i)|\Big|$$
$$\leq \Big|\Big(\gamma_{S,j}\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \gamma_{T,j}\mathcal{D}_T(\widehat{Y} = i \mid Y = j)\Big) - (\gamma_{S,j} - \gamma_{T,j})\mathcal{D}_j(\widehat{Y} = i)\Big|$$
$$= \Big|\gamma_{S,j}(\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)) - \gamma_{T,j}(\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i))\Big|,$$

following which we will have:

$$|\gamma_{S,j}\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \gamma_{T,j}\mathcal{D}_T(\widehat{Y} = i \mid Y = j)| \leq |(\gamma_{S,j} - \gamma_{T,j})\mathcal{D}_j(\widehat{Y} = i)|$$
$$+ \Big|\gamma_{S,j}(\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)) - \gamma_{T,j}(\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i))\Big|$$
$$\leq |\gamma_{S,j} - \gamma_{T,j}| \Big(\alpha_j\mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j\mathcal{D}_T(\widehat{Y} = i \mid Y = j)\Big)$$
$$+ \gamma_{S,j}\Big|\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)\Big| + \gamma_{T,j}\Big|\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)\Big|.$$

To proceed, let us first simplify $\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)$. By definition of $\mathcal{D}_j(\widehat{Y} = i) = \alpha_j\mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j\mathcal{D}_T(\widehat{Y} = i \mid Y = j)$, we know that:

$$\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)$$
$$= \mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \big(\alpha_j\mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j\mathcal{D}_T(\widehat{Y} = i \mid Y = j)\big)$$
$$= \big(\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \alpha_j\mathcal{D}_S(\widehat{Y} = i \mid Y = j)\big) - \beta_j\mathcal{D}_T(\widehat{Y} = i \mid Y = j)$$
$$= \beta_j\big(\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_T(\widehat{Y} = i \mid Y = j)\big).$$

Similarly, for the second term $\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)$, we can show that:

$$\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i) = \alpha_j\big(\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_S(\widehat{Y} = i \mid Y = j)\big).$$

Plugging these two identities into the above, we can continue the analysis with

$$\Big|\gamma_{S,j}(\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i)) - \gamma_{T,j}(\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_j(\widehat{Y} = i))\Big|$$
$$= \Big|\gamma_{S,j}\beta(\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_T(\widehat{Y} = i \mid Y = j)) - \gamma_{T,j}\alpha_j(\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_S(\widehat{Y} = i \mid Y = j))\Big|$$
$$\leq \Big|\gamma_{S,j}\beta_j(\mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \mathcal{D}_T(\widehat{Y} = i \mid Y = j))\Big| + \Big|\gamma_{T,j}\alpha_j(\mathcal{D}_T(\widehat{Y} = i \mid Y = j) - \mathcal{D}_S(\widehat{Y} = i \mid Y = j))\Big|$$
$$\leq \gamma_{S,j}\beta_j\Delta_{\mathrm{CE}}(\widehat{Y}) + \gamma_{T,j}\alpha_j\Delta_{\mathrm{CE}}(\widehat{Y}).$$

The first inequality holds by the triangle inequality and the second by the definition of the conditional error gap. Combining all the inequalities above completes the proof. ■

We are now ready to prove the theorem:

Theorem 5.3.1. (Error Decomposition Theorem) For any classifier $\widehat{Y} = (h \circ g)(X)$,

$$|\varepsilon_S(h \circ g) - \varepsilon_T(h \circ g)| \leq \ \|\mathcal{D}_S^Y - \mathcal{D}_T^Y\|_1 \cdot \mathrm{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y) + 2(k-1)\Delta_{\mathrm{CE}}(\widehat{Y}),$$

where $\|\mathcal{D}_S^Y - \mathcal{D}_T^Y\|_1 := \sum_{i=1}^k |\mathcal{D}_S(Y = i) - \mathcal{D}_T(Y = i)|$ is the L_1 distance between $\mathcal{D}_S^Y$ and $\mathcal{D}_T^Y$.

Proof of Theorem 5.3.1. First, by the law of total probability, it is easy to verify that following identity holds for $a \in \{S, T\}$:

$$\mathcal{D}_a(\widehat{Y} \neq Y) = \sum_{i \neq j} \mathcal{D}_a(\widehat{Y} = i, Y = j) = \sum_{i \neq j} \gamma_{a,j} \mathcal{D}_a(\widehat{Y} = i \mid Y = j).$$

Using this identity, to bound the error gap, we have:

$$\begin{aligned} &|\mathcal{D}_S(Y \neq \widehat{Y}) - \mathcal{D}_T(Y \neq \widehat{Y})| \\ &= \Big| \sum_{i \neq j} \gamma_{S,j} \mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \sum_{i \neq j} \gamma_{T,j} \mathcal{D}_T(\widehat{Y} = i \mid Y = j) \Big| \\ &\leq \sum_{i \neq j} \Big| \gamma_{S,j} \mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \gamma_{T,j} \mathcal{D}_T(\widehat{Y} = i \mid Y = j) \Big|. \end{aligned}$$

Invoking Lemma 7.6.1 to bound the above terms, and since $\forall j \in [k], \gamma_{S,j}, \gamma_{T,j} \in [0, 1]$, $\alpha_j + \beta_j = 1$, we get:

$$\begin{aligned} &|\mathcal{D}_S(Y \neq \widehat{Y}) - \mathcal{D}_T(Y \neq \widehat{Y})| \\ &\leq \sum_{i \neq j} \Big| \gamma_{S,j} \mathcal{D}_S(\widehat{Y} = i \mid Y = j) - \gamma_{T,j} \mathcal{D}_T(\widehat{Y} = i \mid Y = j) \Big| \\ &\leq \sum_{i \neq j} |\gamma_{S,j} - \gamma_{T,j}| \cdot \Big(\alpha_j \mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j \mathcal{D}_T(\widehat{Y} = i \mid Y = j) \Big) + \gamma_{S,j}\beta_j \Delta_{\mathrm{CE}}(\widehat{Y}) + \gamma_{T,j}\alpha_j \Delta_{\mathrm{CE}}(\widehat{Y}) \\ &\leq \sum_{i \neq j} |\gamma_{S,j} - \gamma_{T,j}| \cdot \Big(\alpha_j \mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j \mathcal{D}_T(\widehat{Y} = i \mid Y = j) \Big) + \gamma_{S,j} \Delta_{\mathrm{CE}}(\widehat{Y}) + \gamma_{T,j} \Delta_{\mathrm{CE}}(\widehat{Y}) \\ &= \sum_{i \neq j} |\gamma_{S,j} - \gamma_{T,j}| \cdot \Big(\alpha_j \mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j \mathcal{D}_T(\widehat{Y} = i \mid Y = j) \Big) + \sum_{i=1}^{k} \sum_{j \neq i} \gamma_{S,j} \Delta_{\mathrm{CE}}(\widehat{Y}) + \gamma_{T,j} \Delta_{\mathrm{CE}}(\widehat{Y}) \\ &= \sum_{i \neq j} |\gamma_{S,j} - \gamma_{T,j}| \cdot \Big(\alpha_j \mathcal{D}_S(\widehat{Y} = i \mid Y = j) + \beta_j \mathcal{D}_T(\widehat{Y} = i \mid Y = j) \Big) + 2(k-1)\Delta_{\mathrm{CE}}(\widehat{Y}). \end{aligned}$$

Note that the above holds $\forall \alpha_j, \beta_j \geq 0$ such that $\alpha_j + \beta_j = 1$. By choosing $\alpha_j = 1, \forall j \in [k]$ and $\beta_j = 0, \forall j \in [k]$, we have:

$$\begin{aligned} &= \sum_{i \neq j} |\gamma_{S,j} - \gamma_{T,j}| \cdot \mathcal{D}_S(\widehat{Y} = i \mid Y = j) + 2(k-1)\Delta_{\mathrm{CE}}(\widehat{Y}) \\ &= \sum_{j=1}^{k} |\gamma_{S,j} - \gamma_{T,j}| \cdot \left(\sum_{i=1, i \neq j}^{k} \mathcal{D}_S(\widehat{Y} = i \mid Y = j) \right) + 2(k-1)\Delta_{\mathrm{CE}}(\widehat{Y}) \\ &= \sum_{j=1}^{k} |\gamma_{S,j} - \gamma_{T,j}| \cdot \mathcal{D}_S(\widehat{Y} \neq Y \mid Y = j) + 2(k-1)\Delta_{\mathrm{CE}}(\widehat{Y}) \\ &\leq \|\mathcal{D}_S^Y - \mathcal{D}_T^Y\|_1 \cdot \mathrm{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y) + 2(k-1)\Delta_{\mathrm{CE}}(\widehat{Y}), \end{aligned}$$

where the last line is due to Holder's inequality, completing the proof. ■

5.5.5 Proof of Theorem 5.3.2

Theorem 5.3.2. If $Z = g(X)$ satisfies GLS, then for any $h : \mathcal{Z} \to \mathcal{Y}$ and letting $\widehat{Y} = h(Z)$ be the predictor, we have $\varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) \leq 2\mathrm{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y)$.

Proof. First, by the law of total probability, we have:

$$\begin{aligned}\varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) &= \mathcal{D}_S(Y \neq \widehat{Y}) + \mathcal{D}_T(Y \neq \widehat{Y}) \\ &= \sum_{j=1}^{k}\sum_{i\neq j}\mathcal{D}_S(\widehat{Y} = i|Y = j)\mathcal{D}_S(Y = j) + \mathcal{D}_T(\widehat{Y} = i|Y = j)\mathcal{D}_T(Y = j).\end{aligned}$$

Now, since $\widehat{Y} = (h \circ g)(X) = h(Z)$, $\widehat{Y}$ is a function of Z. Given the generalized label shift assumption, this guarantees that:

$$\forall y, y' \in \mathcal{Y}, \quad \mathcal{D}_S(\widehat{Y} = y' \mid Y = y) = \mathcal{D}_T(\widehat{Y} = y' \mid Y = y).$$

Thus:

$$\begin{aligned}\varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) &= \sum_{j=1}^{k}\sum_{i\neq j}\mathcal{D}_S(\widehat{Y} = i|Y = j)(\mathcal{D}_S(Y = j) + \mathcal{D}_T(Y = j)) \\ &= \sum_{j\in[k]} \mathcal{D}_S(\widehat{Y} \neq Y \mid Y = j) \cdot (\mathcal{D}_S(Y = j) + \mathcal{D}_T(Y = j)) \\ &\leq \max_{j\in[k]} \mathcal{D}_S(\widehat{Y} \neq Y \mid Y = j) \cdot \sum_{j\in[k]} \mathcal{D}_S(Y = j) + \mathcal{D}_T(Y = j) \\ &= 2\mathrm{BER}_{\mathcal{D}_S}(\widehat{Y} \parallel Y).\end{aligned}$$

■

5.5.6 Proof of Lemma 5.3.1

Lemma 5.3.1. Assuming $Z = g(X)$ satisfies GLS, then $\mathcal{D}_T(Z) = \sum_{y\in\mathcal{Y}} \mathbf{w}_y \cdot \mathcal{D}_S(Z, Y = y) =: \mathcal{D}_S^{\mathbf{w}}(Z)$.

Proof. Using (5.2) and (5.4) on the second line:

$$\begin{aligned}\mathcal{D}_T(Z) &= \sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y = y) \cdot \mathcal{D}_T(Z \mid Y = y) \\ &= \sum_{y\in\mathcal{Y}} \mathbf{w}_y \cdot \mathcal{D}_S(Y = y) \cdot \mathcal{D}_S(Z \mid Y = y) \\ &= \sum_{y\in\mathcal{Y}} \mathbf{w}_y \cdot \mathcal{D}_S(Z, Y = y).\end{aligned}$$

■

5.5.7 Proof of Theorem 5.3.3

Theorem 5.3.3. (Clustering structure implies sufficiency) Let $Z = g(X)$ such that $\mathcal{D}_T(Z) = \mathcal{D}_S^{\mathbf{w}}(Z)$. Assume $\mathcal{D}_T(Y = y) > 0, \forall y \in \mathcal{Y}$. If there exists a partition of $\mathcal{Z} = \cup_{y\in\mathcal{Y}}\mathcal{Z}_y$ such that $\forall y \in \mathcal{Y}$, $\mathcal{D}_S(Z \in \mathcal{Z}_y \mid Y = y) = \mathcal{D}_T(Z \in \mathcal{Z}_y \mid Y = y) = 1$, then $Z = g(X)$ satisfies GLS.

Proof. Follow the condition that $\mathcal{D}_T(Z) = \mathcal{D}_S^{\mathbf{w}}(Z)$, by definition of $\mathcal{D}_S^{\mathbf{w}}(Z)$, we have:

$$\begin{aligned}\mathcal{D}_T(Z) = \sum_{y\in\mathcal{Y}} \frac{\mathcal{D}_T(Y=y)}{\mathcal{D}_S(Y=y)}\mathcal{D}_S(Z, Y=y) &\iff \mathcal{D}_T(Z) = \sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)\mathcal{D}_S(Z \mid Y=y) \\ &\iff \sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)\mathcal{D}_T(Z \mid Y=y) = \sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)\mathcal{D}_S(Z \mid Y=y).\end{aligned}$$

Note that the above equation holds for all measurable subsets of $\mathcal{Z}$. Now by the assumption that $\mathcal{Z} = \cup_{y\in\mathcal{Y}}\mathcal{Z}_y$ is a partition of $\mathcal{Z}$, consider $\mathcal{Z}_{y'}$:

$$\sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)\mathcal{D}_T(Z\in\mathcal{Z}_{y'} \mid Y=y) = \sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)\mathcal{D}_S(Z\in\mathcal{Z}_{y'} \mid Y=y).$$

Due to the assumption $\mathcal{D}_S(Z\in\mathcal{Z}_y \mid Y=y) = \mathcal{D}_T(Z\in\mathcal{Z}_y \mid Y=y) = 1$, we know that $\forall y' \neq y$, $\mathcal{D}_T(Z \in \mathcal{Z}_{y'} \mid Y=y) = \mathcal{D}_S(Z\in\mathcal{Z}_{y'} \mid Y=y) = 0$. This shows that both the supports of $\mathcal{D}_S(Z \mid Y=y)$ and $\mathcal{D}_T(Z \mid Y=y)$ are contained in $\mathcal{Z}_y$. Now consider an arbitrary measurable set $E \subseteq \mathcal{Z}_y$, since $\cup_{y\in\mathcal{Y}}\mathcal{Z}_y$ is a partition of $\mathcal{Z}$, we know that

$$\mathcal{D}_S(Z\in E \mid Y=y') = \mathcal{D}_T(Z\in E \mid Y=y') = 0, \quad \forall y' \neq y.$$

Plug $Z \in E$ into the following identity:

$$\begin{aligned}&\sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)\mathcal{D}_T(Z\in E \mid Y=y) = \sum_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)\mathcal{D}_S(Z\in E \mid Y=y) \\ \Longrightarrow\ & \mathcal{D}_T(Y=y)\mathcal{D}_T(Z\in E \mid Y=y) = \mathcal{D}_T(Y=y)\mathcal{D}_S(Z\in E \mid Y=y) \\ \Longrightarrow\ & \mathcal{D}_T(Z\in E \mid Y=y) = \mathcal{D}_S(Z\in E \mid Y=y),\end{aligned}$$

where the last line holds because $\mathcal{D}_T(Y=y) \neq 0$. Realize that the choice of E is arbitrary, this shows that $\mathcal{D}_S(Z \mid Y=y) = \mathcal{D}_T(Z \mid Y=y)$, which completes the proof. ■

5.5.8 Sufficient Conditions for *GLS*

Theorem 5.3.4. Let $\widehat{Y} = h(Z)$, $\gamma := \min_{y\in\mathcal{Y}} \mathcal{D}_T(Y=y)$ and $\mathbf{w}_M := \max\limits_{y\in\mathcal{Y}} \mathbf{w}_y$. For $\tilde{Z} = \hat{Y} \otimes Z$, we have:

$$\max_{y\in\mathcal{Y}} d_{\mathrm{TV}}(\mathcal{D}_S(Z \mid Y=y), \mathcal{D}_T(Z \mid Y=y)) \leq \\ \frac{1}{\gamma}\left(\mathbf{w}_M\varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) + \sqrt{2D_{\mathrm{JS}}(\mathcal{D}_S^{\mathbf{w}}(\tilde{Z}), \mathcal{D}_T(\tilde{Z}))}\right).$$

Proof. For a given class y, we let R_y denote the y-th row of $\hat{Y} \otimes Z$. R_y is a random variable that lives in $\mathcal{Z}$. Let us consider a measurable set $E \subseteq \mathcal{Z}$. Two options exist:

- $\mathbf{0} \in E$, in which case: $\{R_y \in E\} = (\{Z \in E\} \cap \{\hat{Y} = y\}) \cup \{\hat{Y} \neq y\}$,
- $\mathbf{0} \notin E$, in which case: $\{R_y \in E\} = \{Z \in E\} \cap \{\hat{Y} = y\}$.

This allows us to write:

$$\begin{aligned}\mathcal{D}_S^{\mathbf{w}}(R_y \in E) &= \mathcal{D}_S^{\mathbf{w}}(Z \in E, \hat{Y} = y) + \mathbb{1}_{\{\mathbf{0} \in E\}} \mathcal{D}_S^{\mathbf{w}}(\hat{Y} \neq y) \qquad (5.14)\\ &= \sum_{y'} \mathcal{D}_S(Z \in E, \hat{Y} = y, Y = y')\mathbf{w}_{y'} + \mathbb{1}_{\{\mathbf{0} \in E\}} \sum_{y', y'' \neq y} \mathcal{D}_S(\hat{Y} = y'', Y = y')\mathbf{w}_{y'}\\ \mathcal{D}_T(R_y \in E) &= \mathcal{D}_T(Z \in E, \hat{Y} = y) + \mathbb{1}_{\{\mathbf{0} \in E\}} \mathcal{D}_T(\hat{Y} \neq y) \qquad (5.15)\\ &= \sum_{y'} \mathcal{D}_T(Z \in E, \hat{Y} = y, Y = y') + \mathbb{1}_{\{\mathbf{0} \in E\}} \sum_{y', y'' \neq y} \mathcal{D}_T(\hat{Y} = y'', Y = y').\end{aligned}$$

We are interested in the quantity

$$|\mathcal{D}_S(Z \in E \mid Y = y) - \mathcal{D}_T(Z \in E \mid Y = y)| = \frac{1}{\mathcal{D}_T(Y = y)} |\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_T(Z \in E, Y = y)|,$$

which we bound below:

$$\begin{aligned}&|\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_T(Z \in E, Y = y)|\\ &= |\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_S^{\mathbf{w}}(R_y \in E) + \mathcal{D}_S^{\mathbf{w}}(R_y \in E) - \mathcal{D}_T(R_y \in E)\\ &\qquad + \mathcal{D}_T(R_y \in E) - \mathcal{D}_T(Z \in E, Y = y)|\\ &= |\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_S^{\mathbf{w}}(Z \in E, \hat{Y} = y) - \mathbb{1}_{\{\mathbf{0} \in E\}} \mathcal{D}_S^{\mathbf{w}}(\hat{Y} \neq y) + \mathcal{D}_S^{\mathbf{w}}(R_y \in E) - \mathcal{D}_T(R_y \in E)\\ &\qquad + \mathbb{1}_{\{\mathbf{0} \in E\}} \mathcal{D}_T(\hat{Y} \neq y) + \mathcal{D}_T(Z \in E, \hat{Y} = y) - \mathcal{D}_T(Z \in E, Y = y)|\\ &\leq |\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_S^{\mathbf{w}}(Z \in E, \hat{Y} = y)| + \mathbb{1}_{\{\mathbf{0} \in E\}} |\mathcal{D}_S^{\mathbf{w}}(\hat{Y} \neq y) - \mathcal{D}_T(\hat{Y} \neq y)|\\ &\qquad + |\mathcal{D}_S^{\mathbf{w}}(R_y \in E) - \mathcal{D}_T(R_y \in E)| + |\mathcal{D}_T(Z \in E, \hat{Y} = y) - \mathcal{D}_T(Z \in E, Y = y)|\\ &\leq |\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_S^{\mathbf{w}}(Z \in E, \hat{Y} = y)| + |\mathcal{D}_S^{\mathbf{w}}(\hat{Y} \neq y) - \mathcal{D}_T(\hat{Y} \neq y)|\\ &\qquad + D_{TV}(\mathcal{D}_S^{\mathbf{w}}(R_y), \mathcal{D}_T(R_y)) + |\mathcal{D}_T(Z \in E, \hat{Y} = y) - \mathcal{D}_T(Z \in E, Y = y)|, \quad (5.16)\end{aligned}$$

where we used Eqs.5.14 and 5.15 on the third line, the triangle inequality on the fourth and $\sup_E |\mathcal{D}_S^{\mathbf{w}}(R_y \in E) - \mathcal{D}_T(R_y \in E)| = D_{TV}(\mathcal{D}_S^{\mathbf{w}}(R_y), \mathcal{D}_T(R_y))$ on the last. Let us start by upper-bounding the first term:

$$\begin{aligned}&|\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_S^{\mathbf{w}}(Z \in E, \hat{Y} = y)|\\ &\qquad = |\sum_{y'} \mathcal{D}_S(Z \in E, \hat{Y} = y, Y = y')\mathbf{w}_{y'} - \sum_{y'} \mathcal{D}_S(Z \in E, \hat{Y} = y', Y = y)\mathbf{w}_y|\\ &\qquad \leq |\sum_{y' \neq y} \mathcal{D}_S(Z \in E, \hat{Y} = y, Y = y')\mathbf{w}_{y'} - \mathcal{D}_S(Z \in E, \hat{Y} = y', Y = y)\mathbf{w}_y|\\ &\qquad \leq \mathbf{w}_M \sum_{y' \neq y} \mathcal{D}_S(Z \in E, \hat{Y} = y, Y = y') + \mathcal{D}_S(Z \in E, \hat{Y} = y', Y = y)\\ &\qquad \leq \mathbf{w}_M \mathcal{D}_S(Z \in E, \hat{Y} \neq Y) \leq \mathbf{w}_M \varepsilon_S(\widehat{Y}).\end{aligned}$$

Similarly, we can prove that: $|\mathcal{D}_T(R_y \in E) - \mathcal{D}_T(Z \in E, Y = y)| \leq \varepsilon_T(\widehat{Y})$ which bounds the third term of 5.16. As far as the second term is concerned:

$$\begin{aligned}
|\mathcal{D}_S^{\mathbf{w}}(\hat{Y} \neq y) - \mathcal{D}_T(\hat{Y} \neq y)| =& |\sum_{y',y''\neq y} \mathcal{D}_S(\hat{Y} = y'', Y = y')\mathbf{w}_{y'} - \sum_{y',y''\neq y} \mathcal{D}_T(\hat{Y} = y'', Y = y')| \\
\leq& |\sum_{y'} \mathcal{D}_S(Y = y')\mathbf{w}_{y'} - \sum_{y'} \mathcal{D}_T(Y = y')| \\
&+ |\sum_{y'\neq y} \mathcal{D}_S(\hat{Y} = y, Y = y')\mathbf{w}_{y'} - \sum_{y'\neq y} \mathcal{D}_T(\hat{Y} = y, Y = y') \\
&+ \mathcal{D}_S(\hat{Y} = y, Y = y)\mathbf{w}_y - \mathcal{D}_T(\hat{Y} = y, Y = y)| \\
\leq& |\sum_{y'\neq y} \mathcal{D}_S(\hat{Y} = y, Y = y')\mathbf{w}_{y'} - \sum_{y'\neq y} \mathcal{D}_T(\hat{Y} = y, Y = y') \\
&+ \mathcal{D}_S(Y = y)\mathbf{w}_y - \mathcal{D}_S(\hat{Y} \neq y, Y = y)\mathbf{w}_y \\
&- \mathcal{D}_T(Y = y) + \mathcal{D}_T(\hat{Y} \neq y, Y = y)| \\
\leq& \sum_{y'\neq y} \mathcal{D}_S(\hat{Y} = y, Y = y')\mathbf{w}_{y'} + \mathcal{D}_S(\hat{Y} \neq y, Y = y)\mathbf{w}_y \\
&+ \sum_{y'\neq y} \mathcal{D}_T(\hat{Y} = y, Y = y') + \mathcal{D}_T(\hat{Y} \neq y, Y = y) \\
\leq& \mathbf{w}_M \varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y})
\end{aligned}$$

where the first term of the first inequality disappeared because $\forall y' \in \mathcal{Y}, \mathcal{D}_S(Y = y')\mathbf{w}_{y'} = \mathcal{D}_T(Y = y')$ (we also used that property in the second line of the second inequality). Combining these in Eq.5.16, this guarantees that for any measurable set E:

$$|\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_T(Z \in E, Y = y)| \leq 2\mathbf{w}_M \varepsilon_S(\widehat{Y}) + D_{TV}(\mathcal{D}_S^{\mathbf{w}}(R_y), \mathcal{D}_T(R_y)) + 2\varepsilon_T(\widehat{Y}).$$

Finally, we have $D_{TV}(\mathcal{D}_S^{\mathbf{w}}(R_y), \mathcal{D}_T(R_y)) \leq D_{TV}(\mathcal{D}_S^{\mathbf{w}}(\hat{Y} \otimes Z) \parallel \mathcal{D}_T(\hat{Y} \otimes Z))$ and from Briët and Harremoës (2009)), $D_{TV}(\mathcal{D}_S^{\mathbf{w}}(\hat{Y} \otimes Z) \parallel \mathcal{D}_T(\hat{Y} \otimes Z)) \leq \sqrt{8 D_{\mathrm{JS}}(\mathcal{D}_S^{\mathbf{w}}(\hat{Y} \otimes Z) \parallel \mathcal{D}_T(\hat{Y} \otimes Z))}$ (the total variation and Jensen-Shannon distance are equivalent), which gives us straightforwardly:

$$\begin{aligned}
|\mathcal{D}_S(Z \in E \mid Y = y) - \mathcal{D}_T(Z \in E \mid Y = y)| &= \frac{1}{\mathcal{D}_T(Y = y)} |\mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y - \mathcal{D}_T(Z \in E, Y = y)| \\
&\leq \frac{1}{\mathcal{D}_T(Y = y)} \left(2\mathbf{w}_M \varepsilon_S(\widehat{Y}) + D_{TV}(\mathcal{D}_S^{\mathbf{w}}(R_y), \mathcal{D}_T(R_y)) + 2\varepsilon_T(\widehat{Y})\right) \\
&\leq \frac{1}{\mathcal{D}_T(Y = y)} \left(2\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \sqrt{8 D_{\mathrm{JS}}(\mathcal{D}_S^{\mathbf{w}}(\tilde{Z}) \parallel \mathcal{D}_T(\tilde{Z}))} + 2\varepsilon_T(\widehat{Y})\right).
\end{aligned}$$

Using the fact that $D_{TV}(\mathcal{D}_S(Z \mid Y = y), \mathcal{D}_T(Z \mid Y = y)) = \sup_E |\mathcal{D}_S(Z \in E \mid Y = y) - \mathcal{D}_T(Z \in E \mid Y = y)|$ gives:

$$\begin{aligned}
D_{TV}(\mathcal{D}_S(Z \mid Y = y), \mathcal{D}_T(Z \mid Y = y)) &\leq \frac{2}{\mathcal{D}_T(Y = y)} \left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \sqrt{2 D_{\mathrm{JS}}(\mathcal{D}_S^{\mathbf{w}}(\tilde{Z}) \parallel \mathcal{D}_T(\tilde{Z}))} + \varepsilon_T(\widehat{Y})\right) \\
&\leq \frac{2}{\gamma} \left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \sqrt{2 D_{\mathrm{JS}}(\mathcal{D}_S^{\mathbf{w}}(\tilde{Z}) \parallel \mathcal{D}_T(\tilde{Z}))} + \varepsilon_T(\widehat{Y})\right).
\end{aligned}$$

Taking the maximum over y on the left-hand side concludes the proof. ■

Theorem 5.3.5. With the same notation as Thm. 5.3.4:

$$\max_{y\in\mathcal{Y}} d_{\text{TV}}(\mathcal{D}_S(Z \mid Y=y), \mathcal{D}_T(Z \mid Y=y)) \leq \\ \frac{1}{\gamma} \times \left\{ \inf_{\widehat{Y}} \left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) \right) + \sqrt{8 D_{\text{JS}}(\mathcal{D}_S^{\mathbf{w}}(Z), \mathcal{D}_T(Z))} \right\}.$$

Proof. To prove the above upper bound, let us first fix a $y \in \mathcal{Y}$ and fix a classifier $\widehat{Y} = h(Z)$ for some $h : \mathcal{Z} \to \mathcal{Y}$. Now consider any measurable subset $E \subseteq \mathcal{Z}$, we would like to upper bound the following quantity:

$$\begin{aligned} |\mathcal{D}_S(Z \in E \mid Y=y) - \mathcal{D}_T(Z \in E \mid Y=y)| &= \frac{1}{\mathcal{D}_T(Y=y)} \cdot |\mathcal{D}_S(Z \in E, Y=y)\mathbf{w}_y - \mathcal{D}_T(Z \in E, Y=y)| \\ &\leq \frac{1}{\gamma} \cdot |\mathcal{D}_S(Z \in E, Y=y)\mathbf{w}_y - \mathcal{D}_T(Z \in E, Y=y)|. \end{aligned}$$

Hence it suffices if we can upper bound $|\mathcal{D}_S(Z \in E, Y=y)\mathbf{w}_y - \mathcal{D}_T(Z \in E, Y=y)|$. To do so, consider the following decomposition:

$$\begin{aligned} |\mathcal{D}_T(Z \in E, Y=y) - \mathcal{D}_S(Z \in E, Y=y)\mathbf{w}_y| &= |\mathcal{D}_T(Z \in E, Y=y) - \mathcal{D}_T(Z \in E, \widehat{Y}=y) \\ &\quad + \mathcal{D}_T(Z \in E, \widehat{Y}=y) - \mathcal{D}_S^{\mathbf{w}}(Z \in E, \widehat{Y}=y) \\ &\quad + \mathcal{D}_S^{\mathbf{w}}(Z \in E, \widehat{Y}=y) - \mathcal{D}_S(Z \in E, Y=y)\mathbf{w}_y| \\ &\leq |\mathcal{D}_T(Z \in E, Y=y) - \mathcal{D}_T(Z \in E, \widehat{Y}=y)| \\ &\quad + |\mathcal{D}_T(Z \in E, \widehat{Y}=y) - \mathcal{D}_S^{\mathbf{w}}(Z \in E, \widehat{Y}=y)| \\ &\quad + |\mathcal{D}_S^{\mathbf{w}}(Z \in E, \widehat{Y}=y) - \mathcal{D}_S(Z \in E, Y=y)\mathbf{w}_y|. \end{aligned}$$

We bound the above three terms in turn. First, consider $|\mathcal{D}_T(Z \in E, Y=y) - \mathcal{D}_T(Z \in E, \widehat{Y}=y)|$:

$$\begin{aligned} |\mathcal{D}_T(Z \in E, Y=y) - \mathcal{D}_T(Z \in E, \widehat{Y}=y)| &= |\sum_{y'} \mathcal{D}_T(Z \in E, Y=y, \widehat{Y}=y') - \sum_{y'} \mathcal{D}_T(Z \in E, \widehat{Y}=y, Y=y')| \\ &\leq \sum_{y' \neq y} |\mathcal{D}_T(Z \in E, Y=y, \widehat{Y}=y') - \mathcal{D}_T(Z \in E, \widehat{Y}=y, Y=y')| \\ &\leq \sum_{y' \neq y} \mathcal{D}_T(Z \in E, Y=y, \widehat{Y}=y') + \mathcal{D}_T(Z \in E, \widehat{Y}=y, Y=y') \\ &\leq \sum_{y' \neq y} \mathcal{D}_T(Y=y, \widehat{Y}=y') + \mathcal{D}_T(\widehat{Y}=y, Y=y') \\ &\leq \mathcal{D}_T(Y \neq \widehat{Y}) \\ &= \varepsilon_T(\widehat{Y}), \end{aligned}$$

where the last inequality is due to the fact that the definition of error rate corresponds to the sum of all the off-diagonal elements in the confusion matrix while the sum here only corresponds to the sum of all the elements in two slices. Similarly, we can bound the third term as follows:

$$\begin{aligned}
&|\mathcal{D}_S^{\mathbf{w}}(Z \in E, \widehat{Y} = y) - \mathcal{D}_S(Z \in E, Y = y)\mathbf{w}_y| \\
&\quad = |\sum_{y'} \mathcal{D}_S(Z \in E, \widehat{Y} = y, Y = y')\mathbf{w}_{y'} - \sum_{y'} \mathcal{D}_S(Z \in E, \widehat{Y} = y', Y = y)\mathbf{w}_y| \\
&\quad \leq | \sum_{y' \neq y} \mathcal{D}_S(Z \in E, \widehat{Y} = y, Y = y')\mathbf{w}_{y'} - \mathcal{D}_S(Z \in E, \widehat{Y} = y', Y = y)\mathbf{w}_y| \\
&\quad \leq \mathbf{w}_M \sum_{y' \neq y} \mathcal{D}_S(Z \in E, \widehat{Y} = y, Y = y') + \mathcal{D}_S(Z \in E, \widehat{Y} = y', Y = y) \\
&\quad \leq \mathbf{w}_M \mathcal{D}_S(Z \in E, \widehat{Y} \neq Y) \\
&\quad \leq \mathbf{w}_M \varepsilon_S(\widehat{Y}).
\end{aligned}$$

Now we bound the last term. Recall the definition of total variation, we have:

$$\begin{aligned}
|\mathcal{D}_T(Z \in E, \widehat{Y} = y) - \mathcal{D}_S^{\mathbf{w}}(Z \in E, \widehat{Y} = y)| &= |\mathcal{D}_T(Z \in E \wedge Z \in \widehat{Y}^{-1}(y)) - \mathcal{D}_S^{\mathbf{w}}(Z \in E \wedge Z \in \widehat{Y}^{-1}(y))| \\
&\leq \sup_{E' \text{ is measurable}} |\mathcal{D}_T(Z \in E') - \mathcal{D}_S^{\mathbf{w}}(Z \in E')| \\
&= d_{\text{TV}}(\mathcal{D}_T(Z), \mathcal{D}_S^{\mathbf{w}}(Z)).
\end{aligned}$$

Combining the above three parts yields

$$|\mathcal{D}_S(Z \in E \mid Y = y) - \mathcal{D}_T(Z \in E \mid Y = y)| \leq \frac{1}{\gamma} \cdot \left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) + d_{\text{TV}}(\mathcal{D}_S^{\mathbf{w}}(Z), \mathcal{D}_T(Z)) \right).$$

Now realizing that the choice of $y \in \mathcal{Y}$ and the measurable subset E on the LHS is arbitrary, this leads to

$$\max_{y \in \mathcal{Y}} \sup_E |\mathcal{D}_S(Z \in E \mid Y = y) - \mathcal{D}_T(Z \in E \mid Y = y)| \leq \frac{1}{\gamma} \cdot \left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) + d_{\text{TV}}(\mathcal{D}_S^{\mathbf{w}}(Z), \mathcal{D}_T(Z)) \right).$$

Furthermore, notice that the above upper bound holds for any classifier $\widehat{Y} = h(Z)$, hence we have

$$\max_{y \in \mathcal{Y}} d_{\text{TV}}(\mathcal{D}_S(Z \in E \mid Y = y), \mathcal{D}_T(Z \in E \mid Y = y)) \leq \frac{1}{\gamma} \cdot \inf_{\widehat{Y}} \left(\mathbf{w}_M \varepsilon_S(\widehat{Y}) + \varepsilon_T(\widehat{Y}) + d_{\text{TV}}(\mathcal{D}_S^{\mathbf{w}}(Z), \mathcal{D}_T(Z)) \right),$$

which completes the proof. ∎

5.5.9 Proof of Lemma 5.3.2

Lemma 5.3.2. If *GLS* is verified, and if the confusion matrix $\mathbf{C}$ is invertible, then $\mathbf{w} = \mathbf{C}^{-1}\boldsymbol{\mu}$.

Proof. Given (5.2), and with the joint hypothesis $\widehat{Y} = h(Z)$ over both source and target domains, it is straightforward to see that the induced conditional distributions over predicted labels match between the source and target domains, i.e.:

$$\begin{aligned}
&\mathcal{D}_S(\widehat{Y} = h(Z) \mid Y = y) = \\
&\qquad \mathcal{D}_T(\widehat{Y} = h(Z) \mid Y = y), \ \forall y \in \mathcal{Y}.
\end{aligned} \tag{5.17}$$

This allows us to compute $\mu_y, \forall y \in \mathcal{Y}$ as

$$\begin{aligned}
\mathcal{D}_T(\widehat{Y} = y) &= \sum_{y' \in \mathcal{Y}} \mathcal{D}_T(\widehat{Y} = y \mid Y = y') \cdot \mathcal{D}_T(Y = y') \\
&= \sum_{y' \in \mathcal{Y}} \mathcal{D}_S(\widehat{Y} = y \mid Y = y') \cdot \mathcal{D}_T(Y = y') \\
&= \sum_{y' \in \mathcal{Y}} \mathcal{D}_S(\widehat{Y} = y, Y = y') \cdot \frac{\mathcal{D}_T(Y = y')}{\mathcal{D}_S(Y = y')} \\
&= \sum_{y' \in \mathcal{Y}} \mathbf{C}_{y,y'} \cdot \mathbf{w}_{y'}.
\end{aligned}$$

where we used (5.17) for the second line. We thus have $\boldsymbol{\mu} = \mathbf{C}\mathbf{w}$ which concludes the proof. ■

5.5.10 $\mathcal{F}$-IPM for Distributional Alignment

In Table 5.5.1, we list different instances of IPM with different choices of the function class $\mathcal{F}$ in the above definition, including the total variation distance, Wasserstein-1 distance and the Maximum mean discrepancy (Gretton et al., 2012).

Table 5.5.1: List of IPMs with different $\mathcal{F}$. $\|\cdot\|_{\text{Lip}}$ denotes the Lipschitz seminorm and $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS).

$\mathcal{F}$	$d_{\mathcal{F}}$
$\{f : \|f\|_\infty \leq 1\}$	Total Variation
$\{f : \|f\|_{\text{Lip}} \leq 1\}$	Wasserstein-1 distance
$\{f : \|f\|_{\mathcal{H}} \leq 1\}$	Maximum mean discrepancy

5.6 Conclusion

We have introduced the generalized label shift assumption and theoretically-grounded variations of existing algorithms to handle mismatched label distributions. On (rather) balanced tasks from classic benchmarks, our algorithms outperform (by small margins) their base versions. On unbalanced datasets, the gain becomes significant and, as expected theoretically, correlates well with the JSD between label distributions. We now discuss potential improvements.

Improved importance weights estimation All the results above were obtained with $\lambda = 0.5$; we favored simplicity of the algorithm over raw performance. We notice however, that the oracle sometimes shows substantial improvements over the estimated weights algorithm. It suggests that $\mathbf{w}$ is not perfectly estimated and that e.g. fine-tuning λ or updating $\mathbf{w}$ more or less often could lead to better performance. One can also think of settings (e.g. semi-supervised learning) where estimations of $\mathcal{D}_T^Y$ can be obtained via other means.

Extensions The framework we define relies on appropriately reweighting the domain adversarial losses. It can be straightforwardly applied to settings where multiple source and/or target domains are used, by simply maintaining one importance weights vector **w** for each source/target pair (Peng et al., 2019; Zhao et al., 2018c). In particular, label shift could explain the observation from Zhao et al. (2018c) that too many source domains sometimes hurt performance, and our framework might alleviate the issue.

Appendix

5.A More Experiments

5.A.1 Description of the domain adaptation tasks

Digits We follow a widely used evaluation protocol (Hoffman et al., 2017b; Long et al., 2018). For the digits datasets MNIST (M, LeCun and Cortes (2010)) and USPS (U, Dheeru and Karra Taniskidou (2017)), we consider the DA tasks: M → U and U → M. Performance is evaluated on the 10,000/2,007 examples of the MNIST/USPS test sets.

Visda (2017) is a sim-to-real domain adaptation task. The synthetic domain contains 2D rendering of 3D models captured at different angles and lighting conditions. The real domain is made of natural images. Overall, the training, validation and test domains contain 152,397, 55,388 and 5,534 images, from 12 different classes.

Office-31 (Saenko et al., 2010) is one of the most popular dataset for domain adaptation . It contains 4,652 images from 31 classes. The samples come from three domains: Amazon (A), DSLR (D) and Webcam (W), which generate six possible transfer tasks, A → D, A → W, D → A, D → W, W → A and W → D, which we all evaluate.

Office-Home (Venkateswara et al., 2017) is a more complex dataset than Office-31. It consists of 15,500 images from 65 classes depicting objects in office and home environments. The images form four different domains: Artistic (A), Clipart (C), Product (P), and Real-World images (R). We evaluate the 12 possible domain adaptation tasks.

5.A.2 Full results on the domain adaptation tasks

Tables 5.A.1, 5.A.2, 5.A.3, 5.A.4, 5.A.5 and 5.A.6 show the detailed results of all the algorithms on each task of the domains described above. The subscript denotes the fraction of seeds for which our variant outperforms the base algorithm. More precisely, by outperform, we mean that for a given seed (which fixes the network initialization as well as the data being fed to the model) the variant has a larger accuracy on the test set than its base version. Doing so allows to assess specifically the effect of the algorithm, all else kept constant.

5.A.3 Jensen-Shannon divergence of the original and subsampled domain adaptation datasets

Tables 5.A.7, 5.A.8 and 5.A.9 show $D_{\text{JS}}(\mathcal{D}_S(Z)||\mathcal{D}_T(Z))$ for our four datasets and their subsampled versions, rows correspond to the source domain, and columns to the target one. We recall that subsampling simply consists in taking 30% of the first half of the classes in the source domain (which explains why $D_{\text{JS}}(\mathcal{D}_S(Z)||\mathcal{D}_T(Z))$ is not symmetric for the subsampled datasets).

Table 5.A.1: Results on the Digits tasks. M and U stand for MNIST and USPS, the prefix s denotes the experiment where the source domain is subsampled to increase $D_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y)$.

METHOD	M → U	U → M	AVG.	sM → U	sU → M	AVG.
NO AD.	79.04	75.30	77.17	76.02	75.32	75.67
DANN	90.65	95.66	93.15	79.03	87.46	83.24
IWDAN	**93.28**$_{100\%}$	**96.52**$_{100\%}$	**94.90**$_{100\%}$	**91.77**$_{100\%}$	**93.32**$_{100\%}$	**92.54**$_{100\%}$
IWDAN-O	93.73$_{100\%}$	96.81$_{100\%}$	95.27$_{100\%}$	92.50$_{100\%}$	96.42$_{100\%}$	94.46$_{100\%}$
CDAN	94.16	97.29	95.72	84.91	91.55	88.23
IWCDAN	**94.36**$_{60\%}$	**97.45**$_{100\%}$	**95.90**$_{80\%}$	**93.42**$_{100\%}$	**93.03**$_{100\%}$	**93.22**$_{100\%}$
IWCDAN-O	94.34$_{80\%}$	97.35$_{100\%}$	95.85$_{90\%}$	93.37$_{100\%}$	96.26$_{100\%}$	94.81$_{100\%}$

Table 5.A.2: Results on the Visda domain. The prefix s denotes the experiment where the source domain is subsampled to increase $D_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y)$.

METHOD	VISDA	SVISDA
NO AD.	48.39	49.02
DANN	61.88	52.85
IWDAN	**63.52**$_{100\%}$	**60.18**$_{100\%}$
IWDAN-O	64.19$_{100\%}$	62.10$_{100\%}$
CDAN	65.60	60.19
IWCDAN	**66.49**$_{60\%}$	**65.83**$_{100\%}$
IWCDAN-O	68.15$_{100\%}$	66.85$_{100\%}$
JAN	56.98$_{100\%}$	50.64$_{100\%}$
IWJAN	**57.56**$_{100\%}$	**57.12**$_{100\%}$
IWJAN-O	61.48$_{100\%}$	61.30$_{100\%}$

5.A.4 Losses

For batches of data (x_S^i, y_S^i) and (x_T^i) of size s, the DANN losses are:

$$\mathcal{L}_{DA}(x_S^i, y_S^i, x_T^i; \theta, \psi) = -\frac{1}{s}\sum_{i=1}^{s} \log(d_\psi(g_\theta(x_S^i))) + \log(1 - d_\psi(g_\theta(x_T^i))), \tag{5.18}$$

$$\mathcal{L}_C(x_S^i, y_S^i; \theta, \phi) = -\frac{1}{s}\sum_{i=1}^{s} \log(h_\phi(g_\theta(x_S^i)_{y_S^i})). \tag{5.19}$$

Similarly, the CDAN losses are:

$$\mathcal{L}_{DA}(x_S^i, y_S^i, x_T^i; \theta, \psi) = -\frac{1}{s}\sum_{i=1}^{s} \log(d_\psi(h_\phi(g_\theta(x_S^i)) \otimes g_\theta(x_S^i))) + \log(1 - d_\psi(h_\phi(g_\theta(x_T^i)) \otimes g_\theta(x_T^i))), \tag{5.20}$$

$$\mathcal{L}_C(x_S^i, y_S^i; \theta, \phi) = -\frac{1}{s}\sum_{i=1}^{s} \log(h_\phi(g_\theta(x_S^i)_{y_S^i})), \tag{5.21}$$

where $h_\phi(g_\theta(x_S^i)) \otimes g_\theta(x_S^i) := (h_1(g(x_S^i))g(x_S^i), \ldots, h_k(g(x_S^i))g(x_S^i))$ and $h_1(g(x_S^i))$ is the i-th element of vector $h(g(x_S^i))$.

Table 5.A.3: Results on the Office dataset.

Method	A → D	A → W	D → A	D → W	W → A	W → D	Avg.
No DA	79.60	73.18	59.33	96.30	58.75	99.68	77.81
DANN	84.06	85.41	64.67	96.08	66.77	99.44	82.74
IWDAN	**84.30**$_{60\%}$	**86.42**$_{100\%}$	**68.38**$_{100\%}$	**97.13**$_{100\%}$	**67.16**$_{60\%}$	**100.0**$_{100\%}$	**83.90**$_{87\%}$
IWDAN-O	87.23$_{100\%}$	88.88$_{100\%}$	69.92$_{100\%}$	98.09$_{100\%}$	67.96$_{80\%}$	99.92$_{100\%}$	85.33$_{97\%}$
CDAN	**89.56**	93.01	71.25	99.24	70.32	100.0	87.23
IWCDAN	88.91$_{60\%}$	**93.23**$_{60\%}$	**71.90**$_{80\%}$	**99.30**$_{80\%}$	**70.43**$_{60\%}$	**100.0**$_{100\%}$	**87.30**$_{73\%}$
IWCDAN-O	90.08$_{60\%}$	94.52$_{100\%}$	73.11$_{100\%}$	99.30$_{80\%}$	71.83$_{100\%}$	100.0$_{100\%}$	88.14$_{90\%}$
JAN	85.94	**85.66**	**70.50**	97.48	**71.5**	99.72	85.13
IWJAN	**87.68**$_{100\%}$	84.86$_{0\%}$	70.36$_{60\%}$	**98.98**$_{100\%}$	70.06$_{0\%}$	**100.0**$_{100\%}$	**85.32**$_{60\%}$
IWJAN-O	89.68$_{100\%}$	89.18$_{100\%}$	71.96$_{100\%}$	99.02$_{100\%}$	73.0$_{100\%}$	100.0$_{100\%}$	87.14$_{100\%}$

Table 5.A.4: Results on the Subsampled Office dataset.

Method	sA → D	sA → W	sD → A	sD → W	sW → A	sW → D	Avg.
No DA	75.82	70.69	56.82	95.32	58.35	97.31	75.72
DANN	75.46	77.66	56.58	93.76	57.51	96.02	76.17
IWDAN	**81.61**$_{100\%}$	**88.43**$_{100\%}$	**65.00**$_{100\%}$	**96.98**$_{100\%}$	**64.86**$_{100\%}$	**98.72**$_{100\%}$	**82.60**$_{100\%}$
IWDAN-O	84.94$_{100\%}$	91.17$_{100\%}$	68.44$_{100\%}$	97.74$_{100\%}$	64.57$_{100\%}$	99.60$_{100\%}$	84.41$_{100\%}$
CDAN	82.45	84.60	62.54	96.83	65.01	98.31	81.62
IWCDAN	**86.59**$_{100\%}$	**87.30**$_{100\%}$	**66.45**$_{100\%}$	**97.69**$_{100\%}$	**66.34**$_{100\%}$	**98.92**$_{100\%}$	**83.88**$_{100\%}$
IWCDAN-O	87.39$_{100\%}$	91.47$_{100\%}$	69.69$_{100\%}$	97.91$_{100\%}$	67.50$_{100\%}$	98.88$_{100\%}$	85.47$_{100\%}$
JAN	77.74	77.64	64.48	91.68	92.60	65.10	78.21
IWJAN	**84.62**$_{100\%}$	**83.28**$_{100\%}$	**65.30**$_{80\%}$	**96.30**$_{100\%}$	**98.80**$_{100\%}$	**67.38**$_{100\%}$	**82.61**$_{97\%}$
IWJAN-O	88.42$_{100\%}$	89.44$_{100\%}$	72.06$_{100\%}$	97.26$_{100\%}$	98.96$_{100\%}$	71.30$_{100\%}$	86.24$_{100\%}$

The JAN losses (Long et al., 2017) are :

$$\mathcal{L}_{DA}(x_S^i, y_S^i, x_T^i; \theta, \psi) = -\frac{1}{s^2}\sum_{i,j=1}^{s} k(x_S^i, x_S^j) - \frac{1}{s^2}\sum_{i,j=1}^{s} k(x_T^i, x_T^j) + \frac{2}{s^2}\sum_{i,j=1}^{s} k(x_S^i, x_T^j) \tag{5.22}$$

$$\mathcal{L}_C(x_S^i, y_S^i; \theta, \phi) = -\frac{1}{s}\sum_{i=1}^{s} \log(h_\phi(g_\theta(x_S^i)_{y_S^i})), \tag{5.23}$$

where k corresponds to the kernel of the RKHS $\mathcal{H}$ used to measure the discrepancy between distributions. Exactly as in Long et al. (2017), it is the product of kernels on various layers of the network $k(x_S^i, x_S^j) = \prod_{l \in \mathcal{L}} k^l(x_S^i, x_S^j)$. Each individual kernel k^l is computed as the dot-product between two transformations of the representation: $k^l(x_S^i, x_S^j) = \langle d_\psi^l(g_\theta^l(x_S^i)), d_\psi^l(g_\theta^l(x_S^j))\rangle$ (in this case, d_ψ^l outputs vectors in a high-dimensional space). See Section 5.A.6 for more details.

Table 5.A.5: Results on the Office-Home dataset.

Method	A → C	A → P	A → R	C → A	C → P	C → R	P → A	P → C	P → R	R → A	R → C	R → P	Avg.
No DA	41.02	62.97	71.26	48.66	58.86	60.91	47.1	35.94	68.27	61.79	44.42	75.5	56.39
DANN	46.03	62.23	70.57	49.06	63.05	64.14	48.29	44.06	72.62	63.81	53.93	77.64	59.62
IWDAN	**48.65**$_{100\%}$	**69.19**$_{100\%}$	**73.60**$_{100\%}$	**53.59**$_{100\%}$	**66.25**$_{100\%}$	**66.09**$_{100\%}$	**52.81**$_{100\%}$	**46.24**$_{80\%}$	**73.97**$_{100\%}$	**64.90**$_{100\%}$	**54.02**$_{80\%}$	**77.96**$_{100\%}$	**62.27**$_{97\%}$
IWDAN-O	50.19$_{100\%}$	70.53$_{100\%}$	75.44$_{100\%}$	56.69$_{100\%}$	67.40$_{100\%}$	67.98$_{100\%}$	59.33$_{100\%}$	48.28$_{100\%}$	76.37$_{100\%}$	69.42$_{100\%}$	56.09$_{100\%}$	78.45$_{100\%}$	64.68$_{100\%}$
CDAN	49.00	69.23	74.55	54.46	68.23	68.9	56.77	**48.8**	76.83	**71.27**	**55.72**	**81.27**	64.59
IWCDAN	**49.81**$_{100\%}$	**73.41**$_{100\%}$	**77.56**$_{100\%}$	**56.5**$_{100\%}$	**69.64**$_{80\%}$	**70.33**$_{100\%}$	**58.99**$_{100\%}$	48.41$_{0\%}$	**77.94**$_{100\%}$	69.48$_{0\%}$	54.73$_{0\%}$	81.07$_{60\%}$	**65.66**$_{70\%}$
IWCDAN-O	52.31$_{100\%}$	74.54$_{100\%}$	78.46$_{100\%}$	60.33$_{100\%}$	70.78$_{100\%}$	71.47$_{100\%}$	62.60$_{100\%}$	50.73$_{100\%}$	78.88$_{100\%}$	72.44$_{100\%}$	57.79$_{100\%}$	81.31$_{80\%}$	67.64$_{98\%}$
JAN	**41.64**	67.20	73.12	51.02	62.52	64.46	54.5	40.36	**73.10**	**64.54**	**45.98**	**76.58**	59.59
IWJAN	41.12$_{0\%}$	**67.56**$_{80\%}$	**73.14**$_{60\%}$	**51.70**$_{100\%}$	**63.42**$_{100\%}$	**65.22**$_{100\%}$	**55.26**$_{80\%}$	**40.38**$_{60\%}$	73.08$_{80\%}$	64.40$_{60\%}$	45.68$_{0\%}$	76.36$_{40\%}$	**59.78**$_{63\%}$
IWJAN-O	41.88$_{80\%}$	68.72$_{100\%}$	73.62$_{100\%}$	53.04$_{100\%}$	63.88$_{100\%}$	66.48$_{100\%}$	57.78$_{100\%}$	41.32$_{100\%}$	73.66$_{100\%}$	65.40$_{100\%}$	46.68$_{100\%}$	76.36$_{20\%}$	60.73$_{92\%}$

The IWJAN losses are:

$$\mathcal{L}^{\mathbf{w}}_{DA}(x^i_S, y^i_S, x^i_T; \theta, \psi) = -\frac{1}{s^2}\sum_{i,j=1}^{s} \mathbf{w}_{y^i_S}\mathbf{w}_{y^j_S} k(x^i_S, x^j_S) - \frac{1}{s^2}\sum_{i,j=1}^{s} k(x^i_T, x^j_T) + \frac{2}{s^2}\sum_{i,j=1}^{s} \mathbf{w}_{y^i_S} k(x^i_S, x^j_T) \quad (5.24)$$

$$\mathcal{L}^{\mathbf{w}}_{C}(x^i_S, y^i_S; \theta, \phi) = -\frac{1}{s}\sum_{i=1}^{s} \frac{1}{k\mathcal{D}_S(Y=y)} \log(h_\phi(g_\theta(x^i_S))_{y^i_S}). \quad (5.25)$$

5.A.5 Generation of domain adaptation tasks with varying $D_{\mathbf{JS}}(\mathcal{D}_S(Z) \parallel \mathcal{D}_T(Z))$

We consider the MNIST → USPS task and generate a set $\mathcal{V}$ of 50 vectors in $[0.1, 1]^{10}$. Each vector corresponds to the fraction of each class to be trained on, either in the source or the target domain (to assess the impact of both). The left bound is chosen as 0.1 to ensure that classes all contain some samples.

This methodology creates 100 domain adaptation tasks, 50 for *subsampled*-MNIST → USPS and 50 for MNIST → *subsampled*-USPS, with Jensen-Shannon divergences varying from 6.1e−3 to 9.53e−2[8]. They are then used to evaluate our algorithms, see Section 5.4 and Figures 5.4.1 and 5.A.1. Fig 5.A.1 shows the absolute performance of the 6 algorithms we consider here. We see the sharp decrease in performance of the base versions DANN and CDAN. Comparatively, our importance-weighted algorithms maintain good performance even for large divergences between the marginal label distributions.

[8] We manually rejected some samples to guarantee a rather uniform set of divergences.

Table 5.A.6: Results on the subsampled Office-Home dataset.

Method	A → C	A → P	A → R	C → A	C → P	C → R	P → A	P → C	P → R	R → A	R → C	R → P	Avg.
No DA	35.70	54.72	62.61	43.71	52.54	56.62	44.29	33.05	65.20	57.12	40.46	70.0	
DANN	36.14	54.16	61.72	44.33	52.56	56.37	44.58	37.14	65.21	56.70	43.16	69.86	51.83
IWDAN	**39.81**$_{100\%}$	**63.01**$_{100\%}$	**68.67**$_{100\%}$	**47.39**$_{100\%}$	**61.05**$_{100\%}$	**60.44**$_{100\%}$	**50.44**$_{100\%}$	**41.63**$_{100\%}$	**72.46**$_{100\%}$	**61.00**$_{100\%}$	**49.40**$_{100\%}$	**76.07**$_{100\%}$	**57.61**$_{100\%}$
IWDAN-O	42.79$_{100\%}$	66.22$_{100\%}$	71.40$_{100\%}$	53.39$_{100\%}$	61.47$_{100\%}$	64.97$_{100\%}$	56.05$_{100\%}$	43.39$_{100\%}$	74.87$_{100\%}$	66.73$_{100\%}$	51.72$_{100\%}$	77.46$_{100\%}$	60.87$_{100\%}$
CDAN	38.90	56.80	64.77	48.02	60.07	61.17	49.65	41.36	70.24	62.35	46.98	74.69	56.25
IWCDAN	**42.96**$_{100\%}$	**65.01**$_{100\%}$	**71.34**$_{100\%}$	**52.89**$_{100\%}$	**64.65**$_{100\%}$	**66.48**$_{100\%}$	**54.87**$_{100\%}$	**44.80**$_{100\%}$	**75.91**$_{100\%}$	**67.02**$_{100\%}$	**50.45**$_{100\%}$	**78.55**$_{100\%}$	**61.24**$_{100\%}$
IWCDAN-O	45.76$_{100\%}$	68.61$_{100\%}$	73.18$_{100\%}$	56.88$_{100\%}$	66.61$_{100\%}$	68.48$_{100\%}$	59.63$_{100\%}$	46.98$_{100\%}$	77.54$_{100\%}$	69.24$_{100\%}$	53.77$_{100\%}$	78.11$_{100\%}$	63.73$_{100\%}$
JAN	34.52	56.86	64.54	46.18	56.84	59.06	50.64	37.24	69.98	58.72	40.64	72.00	53.94
IWJAN	**36.24**$_{100\%}$	**61.00**$_{100\%}$	**66.34**$_{100\%}$	**48.66**$_{100\%}$	**59.92**$_{100\%}$	**61.88**$_{100\%}$	**52.92**$_{100\%}$	**37.68**$_{100\%}$	**70.88**$_{100\%}$	**60.32**$_{100\%}$	**41.54**$_{100\%}$	**73.26**$_{100\%}$	**55.89**$_{100\%}$
IWJAN-O	37.46$_{100\%}$	62.68$_{100\%}$	66.88$_{100\%}$	49.82$_{100\%}$	60.22$_{100\%}$	62.54$_{100\%}$	56.54$_{100\%}$	39.66$_{100\%}$	71.78$_{100\%}$	62.36$_{100\%}$	44.56$_{100\%}$	73.76$_{100\%}$	57.36$_{100\%}$

5.A.6 Implementation details

For MNIST and USPS, the architecture is akin to LeNet (LeCun et al., 1998a), with two convolutional layers, ReLU and MaxPooling, followed by two fully connected layers. The representation is also taken as the last hidden layer, and has 500 neurons. The optimizer for those tasks is SGD with a learning rate of 0.02, annealed by 0.5 every five training epochs for M → U and 6 for U → M. The weight decay is also $5e{-}4$ and the momentum 0.9.

For the Office and Visda experiments with IWDAN and IWCDAN, we train a ResNet-50, optimized using SGD with momentum. The weight decay is also $5e{-}4$ and the momentum 0.9. The learning rate is $3e{-}4$ for the Office-31 tasks A → D and D → W, $1e{-}3$ otherwise (default learning rates from the CDAN implementation[9]).

For the IWJAN experiments, we use the default implementation of Xlearn codebase[10] and simply add the weigths estimation and reweighted objectives to it, as described in Section 5.A.4. Parameters, configuration and networks remain the same.

Finally, for the Office experiments, we update the importance weights **w** every 15 passes on the dataset (in order to improve their estimation on small datasets). On Digits and Visda, the importance weights are updated every pass on the source dataset. Here too, fine-tuning that value might lead to a better estimation

[9] `https://github.com/thuml/CDAN/tree/master/pytorch`

[10] `https://github.com/thuml/Xlearn/tree/master/pytorch`

Table 5.A.7: Jensen-Shannon divergence between the label distributions of the Digits and Visda tasks.

(a) Full Dataset

	MNIST	USPS	Real
MNIST	0	6.64e−3	-
USPS	6.64e−3	0	-
Synth.	-	-	2.61e−2

(b) Subsampled

	MNIST	USPS	Real
MNIST	0	6.52e−2	-
USPS	2.75e−2	0	-
Synth.	-	-	6.81e−2

Table 5.A.8: Jensen-Shannon divergence between the label distributions of the Office-31 tasks.

(a) Full Dataset

	Amazon	DSLR	Webcam
Amazon	0	1.76e−2	9.52e−3
DSLR	1.76e−2	0	2.11e−2
Webcam	9.52e−3	2.11e−2	0

(b) Subsampled

	Amazon	DSLR	Webcam
Amazon	0	6.25e−2	4.61e−2
DSLR	5.44e−2	0	5.67e−2
Webcam	5.15e−2	7.05e−2	0

of $\mathbf{w}$ and help bridge the gap with the oracle versions of the algorithms.

We use the cvxopt package[11] to solve the quadratic program 5.5.

5.A.7 Weight Estimation

We estimate importance weights using Lemma 5.3.2, which relies on the GLS assumption. However, there is no guarantee that GLS is verified at any point during training, so the exact dynamics of $\mathbf{w}$ are unclear. Below we discuss those dynamics and provide some intuition about them.

In Fig. 5.A.2b, we plot the Euclidean distance between the moving average of weights estimated using the equation $\mathbf{w} = \mathbf{C}^{-1}\boldsymbol{\mu}$ and the true weights (note that this can be done for any algorithm). As can be seen in the figure, the distance between the estimated and true weights is highly correlated with the performance of the algorithm (see Fig.5.A.2a). In particular, we see that the estimations for IWDAN is more accurate than for DANN. The estimation for DANN exhibits an interesting shape, improving at first, and then getting worse. At the same time, the estimation for IWDAN improves monotonously. The weights for IWDAN-O get very close to the true weights which is in line with our theoretical results: IWDAN-O gets close to zero error on the target error, Thm. 5.3.4 thus guarantees that GLS is verified, which in turns implies that the weight estimation is accurate (Lemma 5.3.2). Finally, without domain adaptation, the estimation is very poor. The following two lemmas shed some light on the phenomena observed for DANN and IWDAN:

Lemma 5.A.1. If the source error $\varepsilon_S(h \circ g)$ is zero and the source and target marginals verify $D_{\text{JS}}(\mathcal{D}_S^{\tilde{\mathbf{w}}}(Z), \mathcal{D}_T(Z)) = 0$, then the estimated weight vector $\mathbf{w}$ is equal to $\tilde{\mathbf{w}}$.

[11] `http://cvxopt.org/`

Table 5.A.9: Jensen-Shannon divergence between the label distributions of the Office-Home tasks.

(a) Full Dataset

	ART	CLIPART	PRODUCT	REAL WORLD
ART	0	3.85e−2	4.49e−2	2.40e−2
CLIPART	3.85e−2	0	2.33e−2	2.14e−2
PRODUCT	4.49e−2	2.33e−2	0	1.61e−2
REAL WORLD	2.40e−2	2.14e−2	1.61e−2	0

(b) Subsampled

	ART	CLIPART	PRODUCT	REAL WORLD
ART	0	8.41e−2	8.86e−2	6.69e−2
CLIPART	7.07e−2	0	5.86e−2	5.68e−2
PRODUCT	7.85e−2	6.24e−2	0	5.33e−2
REAL WORLD	6.09e−2	6.52e−2	5.77e−2	0

Proof. If $\varepsilon_S(h \circ g) = 0$, then the confusion matrix $\mathbf{C}$ is diagonal and its y-th line is $\mathcal{D}_S(Y = y)$. Additionally, if $D_{\text{JS}}(\mathcal{D}_S^{\tilde{\mathbf{w}}}(Z), \mathcal{D}_T(Z)) = 0$, then from a straightforward extension of Eq. 5.12, we have $D_{\text{JS}}(\mathcal{D}_S^{\tilde{\mathbf{w}}}(\hat{Y}), \mathcal{D}_T(\hat{Y})) = 0$. In other words, the distribution of predictions on the source and target domains match, *i.e.* $\boldsymbol{\mu}_y = \mathcal{D}_T(\hat{Y} = y) = \sum_{y'} \tilde{\mathbf{w}}_{y'} \mathcal{D}_S(\hat{Y} = y, Y = y') = \tilde{\mathbf{w}}_y \mathcal{D}_S(Y = y), \forall y$ (where the last equality comes from $\varepsilon_S(h \circ g) = 0$). Finally, we get that $\mathbf{w} = \mathbf{C}^{-1}\boldsymbol{\mu} = \tilde{\mathbf{w}}$. ■

In particular, applying this lemma to DANN (*i.e.* with $\tilde{\mathbf{w}}_{y'} = \mathbf{1}$) suggests that at convergence, the estimated weights should tend to $\mathbf{1}$. Empirically, Fig. 5.A.2b shows that as the marginals get matched, the estimation for DANN does get closer to $\mathbf{1}$ ($\mathbf{1}$ corresponds to a distance of 2.16)[12]. We now attempt to provide some intuition on the behavior of IWDAN, with the following lemma:

Lemma 5.A.2. If $\varepsilon_S(h \circ g) = 0$ and if for a given y:

$$\min(\tilde{\mathbf{w}}_y \mathcal{D}_S(Y = y), \mathcal{D}_T(Y = y)) \leq \boldsymbol{\mu}_y \leq \max(\tilde{\mathbf{w}}_y \mathcal{D}_S(Y = y), \mathcal{D}_T(Y = y)), \tag{5.26}$$

then, letting $\mathbf{w} = \mathbf{C}^{-1}\boldsymbol{\mu}$ be the estimated weight:

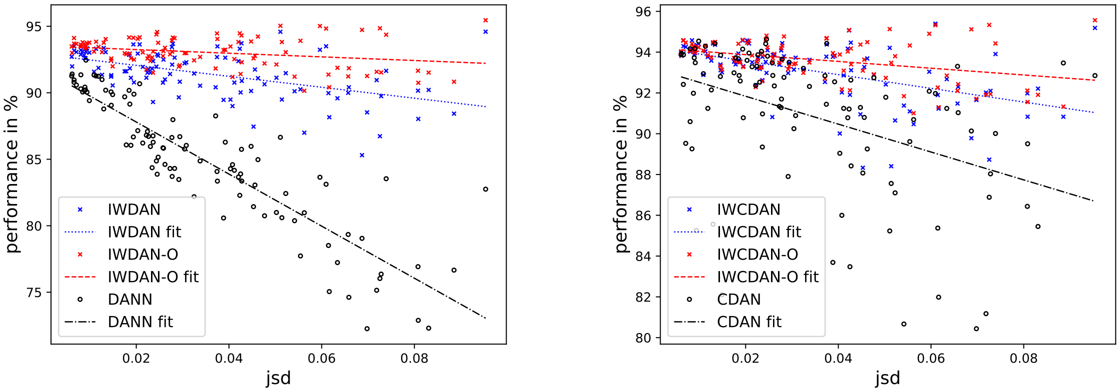
$$|\mathbf{w}_y - \mathbf{w}_y^*| \leq |\tilde{\mathbf{w}}_y - \mathbf{w}_y^*|.$$

Applying this lemma to $\tilde{\mathbf{w}}_y = \mathbf{w}_t$, and assuming that (5.26) holds for all the classes y (we discuss what the assumption implies below), we get that:

$$\|\mathbf{w}_{t+1} - \mathbf{w}_y^*\| \leq \|\mathbf{w}_t - \mathbf{w}_y^*\|, \tag{5.27}$$

or in other words, the estimation improves monotonously. Combining this with Lemma 5.A.2 suggests an explanation for the shape of the IWDAN estimated weights on Fig. 5.A.2b: the monotonous improvement of the estimation is counter-balanced by the matching of weighted marginals which, when reached, makes $\mathbf{w}_t$ constant (Lemma 5.A.1 applied to $\tilde{\mathbf{w}} = \mathbf{w}_t$). However, we wish to emphasize that the exact dynamics

[12] It does not reach it as the learning rate is decayed to 0.



(a) Performance of DANN, IWDAN and IWDAN-O.

(b) Performance of CDAN, CDAN and IWCDAN.

Figure 5.A.1: Performance in % of our algorithms and their base versions. The x-axis represents $D_{\text{JS}}(\mathcal{D}_S^Y, \mathcal{D}_T^Y)$, the Jensen-Shannon distance between label distributions. Lines represent linear fits to the data. For both sets of algorithms, the larger the jsd, the larger the improvement.

of $\mathbf{w}$ are complex, and we do not claim understand them fully. In all likelihood, they are the by-product of regularity in the data, properties of deep neural networks and their interaction with stochastic gradient descent. Additionally, the dynamics are also inherently linked to the success of domain adaptation, which to this day remains an open problem.

As a matter of fact assumption (5.26) itself relates to successful domain adaptation. Setting aside $\tilde{\mathbf{w}}$, which simply corresponds to a class reweighting of the source domain, (5.26) states that predictions on the target domain fall between a successful prediction (corresponding to $\mathcal{D}_T(Y = y)$) and the prediction of a model with matched marginals (corresponding to $\mathcal{D}_S(Y = y)$). In other words, we assume that the model is naturally in between successful domain adaptation and successful marginal matching. Empirically, we observed that it holds true for most classes (with $\tilde{\mathbf{w}} = \tilde{\mathbf{w}}_t$ for IWDAN and with $\tilde{\mathbf{w}} = \mathbf{1}$ for DANN), but not all early in training[13].

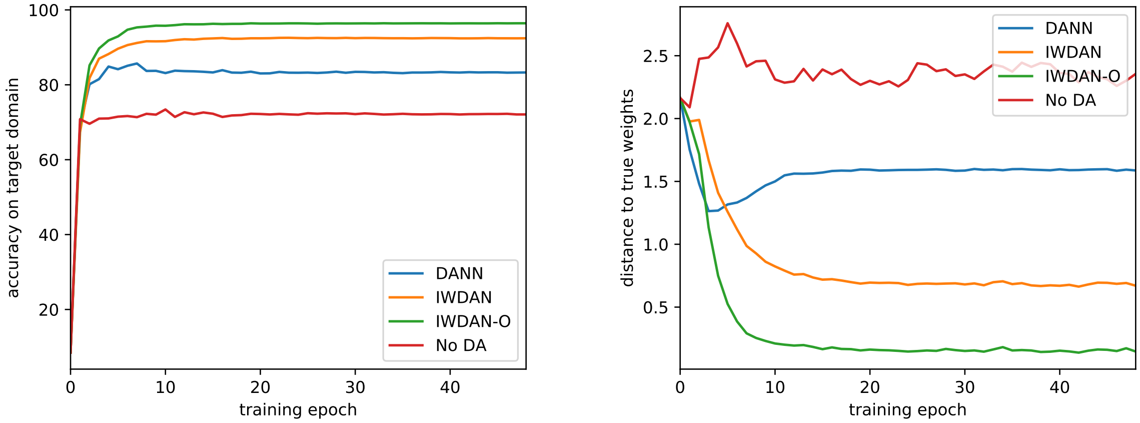
To conclude this section, we prove Lemma 5.A.2.

Proof. From $\varepsilon_S(h \circ g) = 0$, we know that $\mathbf{C}$ is diagonal and that its y-th line is $\mathcal{D}_S(Y = y)$. This gives us: $\mathbf{w}_y = (\mathbf{C}^{-1}\boldsymbol{\mu})_y = \frac{\boldsymbol{\mu}_y}{\mathcal{D}_S(Y=y)}$. Hence:

$$\begin{aligned}
& \min(\tilde{\mathbf{w}}_y \mathcal{D}_S(Y = y), \mathcal{D}_T(Y = y)) \leq \boldsymbol{\mu}_y \leq \max(\tilde{\mathbf{w}}_y \mathcal{D}_S(Y = y), \mathcal{D}_T(Y = y)) \\
\iff \quad & \frac{\min(\tilde{\mathbf{w}}_y \mathcal{D}_S(Y = y), \mathcal{D}_T(Y = y))}{\mathcal{D}_S(Y = y)} \leq \frac{\boldsymbol{\mu}_y}{\mathcal{D}_S(Y = y)} \leq \frac{\max(\tilde{\mathbf{w}}_y \mathcal{D}_S(Y = y), \mathcal{D}_T(Y = y))}{\mathcal{D}_S(Y = y)} \\
\iff \quad & \min(\tilde{\mathbf{w}}_y, \mathbf{w}_y^*) \leq \mathbf{w}_y \leq \max(\tilde{\mathbf{w}}_y, \mathbf{w}_y^*) \\
\iff \quad & \min(\tilde{\mathbf{w}}_y, \mathbf{w}_y^*) - \mathbf{w}_y^* \leq \mathbf{w}_y - \mathbf{w}_y^* \leq \max(\tilde{\mathbf{w}}_y, \mathbf{w}_y^*) - \mathbf{w}_y^* \\
\iff \quad & |\mathbf{w}_y - \mathbf{w}_y^*| \leq |\tilde{\mathbf{w}}_y - \mathbf{w}_y^*|,
\end{aligned}$$

which concludes the proof. ■

[13] In particular at initialization, one class usually dominates the others.



(a) Transfer accuracy during training.

(b) Distance to true weights during training.

Figure 5.A.2: *Left* Accuracy of various algorithms during training. *Right* Euclidean distance between the weights estimated using Lemma 5.3.2 and the true weights. Those plots correspond to averages over 5 seeds.

Chapter 6

Learning Fair Representations

In this chapter, through the lens of information theory, we provide the first result that quantitatively characterizes the tradeoff between demographic parity and the joint utility across different population groups. Specifically, when the base rates differ between groups, we show that any method aiming to learn fair representations admits an information-theoretic lower bound on the joint error across these groups. To complement our negative results, we also prove that if the optimal decision functions across different groups are close, then learning fair representations leads to an alternative notion of fairness, known as the accuracy parity, which states that the error rates are close between groups. Finally, our theoretical findings are also confirmed empirically on real-world datasets.

6.1 Introduction

With the prevalence of machine learning applications in high-stakes domains, e.g., criminal judgement, medical testing, online advertising, etc., it is crucial to ensure that the automated decision making systems do not propagate existing bias or discrimination that might exist in historical data (Barocas and Selbst, 2016; Berk et al., 2018; of the President, 2016). Among many recent proposals for achieving different notions of algorithmic fairness (Dwork et al., 2012; Hardt et al., 2016; Zafar et al., 2015, 2017; Zemel et al., 2013), learning fair representations has received increasing attention due to recent advances in learning rich representations with deep neural networks (Beutel et al., 2017; Edwards and Storkey, 2015; Louizos et al., 2015; Madras et al., 2018; Song et al., 2019; Zhang et al., 2018). In fact, a line of work has proposed to learn group-invariant representations with adversarial learning techniques in order to achieve statistical parity, also known as the demographic parity in the literature. This line of work dates at least back to Zemel et al. (2013) where the authors proposed to learn predictive models that are independent of the group membership attribute. At a high level, the underlying idea is that if representations of instances from different groups are similar to each other, then any predictive model on top of them will certainly make decisions independent of group membership.

On the other hand, it has long been observed that there is an underlying tradeoff between utility and demographic parity:

> *"All methods have in common that to some extent accuracy must be traded-off for lowering the dependency." (Calders et al., 2009)*

In particular, it is easy to see that in an extreme case where the group membership coincides with the target task, a call for exact demographic parity will inevitably remove the perfect predictor (Hardt et al., 2016). Empirically, it has also been observed that a tradeoff exists between accuracy and fairness in binary classification (Zliobaite, 2015). Clearly, methods based on learning fair representations are also bound by such inherent tradeoff between utility and fairness. But how does the fairness constraint trade for utility? Will learning fair representations help to achieve other notions of fairness besides the demographic parity? If yes, what is the fundamental limit of utility that we can hope to achieve under such constraint?

To answer the above questions, through the lens of information theory, in this chapter we provide the first result that quantitatively characterizes the tradeoff between demographic parity and the joint utility across different population groups. Specifically, when the base rates differ between groups, we provide a tight information-theoretic lower bound on the joint error across these groups. Our lower bound is algorithm-independent so it holds for all methods aiming to learn fair representations. When only approximate demographic parity is achieved, we also present a family of lower bounds to quantify the tradeoff of utility introduced by such approximate constraint. As a side contribution, our proof technique is simple but general, and we expect it to have broader applications in other learning problems using adversarial techniques, e.g., unsupervised domain adaptation (Ganin et al., 2016; Zhao et al., 2019e), privacy-preservation under attribute inference attacks (Hamm, 2017; Zhao et al., 2019a) and multilingual machine translation (Johnson et al., 2017).

To complement our negative results, we show that if the optimal decision functions across different groups are close, then learning fair representations helps to achieve an alternative notion of fairness, i.e., the accuracy parity, which states that the error rates are close between groups. Empirically, we conduct experiments on a real-world dataset that corroborate both our positive and negative results. We believe our theoretical insights contribute to better understanding of the tradeoff between utility and different notions of fairness, and they are also helpful in guiding the future design of representation learning algorithms to achieve algorithmic fairness.

6.2 Preliminaries

We first introduce the notation used throughout the chapter and formally describe the problem setup. We then briefly discuss some information-theoretic concepts that will be used in our analysis.

Notation We use $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$ to denote the input and output space. Accordingly, we use X and Y to denote the random variables which take values in $\mathcal{X}$ and $\mathcal{Y}$, respectively. Lower case letters $\mathbf{x}$ and y are used to denote the instantiation of X and Y. To simplify the presentation, we use $A \in \{0, 1\}$ as the sensitive attribute, e.g., race, gender, etc. [1] Let $\mathcal{H}$ be the hypothesis class of classifiers. In other words, for $h \in \mathcal{H}$, $h : \mathcal{X} \to \mathcal{Y}$ is the predictor that outputs a prediction. Note that even if the predictor does not explicitly take the sensitive attribute A as input, this *fairness through blindness* mechanism can still be biased due to the potential correlations between X and A. In this chapter we study the stochastic setting where there is a joint distribution $\mathcal{D}$ over X, Y and A from which the data are sampled. To keep the notation consistent, for $a \in \{0, 1\}$, we use $\mathcal{D}_a$ to mean the conditional distribution of $\mathcal{D}$ given $A = a$. For an event E, $\mathcal{D}(E)$ denotes the probability of E under $\mathcal{D}$. In particular, in the literature of fair machine learning, we call $\mathcal{D}(Y = 1)$ the *base rate* of distribution $\mathcal{D}$ and we use $\Delta_{\mathrm{BR}}(\mathcal{D}, \mathcal{D}') := |\mathcal{D}(Y = 1) - \mathcal{D}'(Y = 1)|$ to denote the difference of the base rates between two distributions $\mathcal{D}$ and $\mathcal{D}'$ over the same sample space. Given a feature transformation function $g : \mathcal{X} \to \mathcal{Z}$ that maps instances from the input space $\mathcal{X}$ to feature space $\mathcal{Z}$, we define $g_\sharp\mathcal{D} := \mathcal{D} \circ g^{-1}$ to be the induced (pushforward) distribution of $\mathcal{D}$ under g, i.e., for any event $E' \subseteq \mathcal{Z}$, $g_\sharp\mathcal{D}(E') := \mathcal{D}(g^{-1}(E')) = \mathcal{D}(\{x \in \mathcal{X} \mid g(x) \in E'\})$.

Problem Setup Given a joint distribution $\mathcal{D}$, the error of a predictor h under $\mathcal{D}$ is defined as $\varepsilon_{\mathcal{D}}(h) := \mathbb{E}_{\mathcal{D}}[|Y - h(X)|]$. Note that for binary classification problems, when $h(X) \in \{0,1\}$, $\varepsilon_{\mathcal{D}}(h)$ reduces to the true error rate of binary classification. To make the notation more compact, we may drop the subscript $\mathcal{D}$ when it is clear from the context. In this chapter we focus on group fairness where the group membership is given by the sensitive attribute A. Even in this context there are many possible definitions of *fairness* (Narayanan, 2018), and in what follows we provide a brief review of the ones that are mostly relevant to this chapter.

Definition 6.2.1 (Demographic Parity)**.** Given a joint distribution $\mathcal{D}$, a classifier $\widehat{Y}$ satisfies *demographic parity* if $\widehat{Y}$ is independent of A.

Demographic parity reduces to the requirement that $\mathcal{D}_0(\widehat{Y} = 1) = \mathcal{D}_1(\widehat{Y} = 1)$, i.e., positive outcome is given to the two groups at the same rate. When exact equality does not hold, we use the absolute difference between them as an approximate measure:

Definition 6.2.2 (DP Gap)**.** Given a joint distribution $\mathcal{D}$, the *demographic parity gap* of a classifier $\widehat{Y}$ is $\Delta_{\mathrm{DP}}(\widehat{Y}) := |\mathcal{D}_0(\widehat{Y} = 1) - \mathcal{D}_1(\widehat{Y} = 1)|$.

Demographic parity is also known as *statistical parity*, and it has been adopted as definition of fairness in a series of work (Calders et al., 2009; Edwards and Storkey, 2015; Johndrow et al., 2019; Kamiran and Calders, 2009; Kamishima et al., 2011; Louizos et al., 2015; Madras et al., 2018; Zemel et al., 2013). However, as we shall quantify precisely in Section 6.3, demographic parity may cripple the utility that we hope to achieve, especially in the common scenario where the *base rates* differ between two groups, e.g., $\mathcal{D}_0(Y = 1) \neq \mathcal{D}_1(Y = 1)$ (Hardt et al., 2016). In light of this, an alternative definition is *accuracy parity*:

Definition 6.2.3 (Accuracy Parity)**.** Given a joint distribution $\mathcal{D}$, a classifier h satisfies *accuracy parity* if $\varepsilon_{\mathcal{D}_0}(h) = \varepsilon_{\mathcal{D}_1}(h)$.

In the literature, a break of accuracy parity is also known as disparate mistreatment (Zafar et al., 2017). Again, when h is a deterministic binary classifier, accuracy parity reduces to $\mathcal{D}_0(h(X) = Y) =$

[1] Our main results could also be straightforwardly extended to the setting where A is a categorical variable.

$\mathcal{D}_1(h(X) = Y)$. Different from demographic parity, the definition of accuracy parity does not eliminate the perfect predictor when $Y = A$ when the base rates differ between two groups. When costs of different error types matter, more refined definitions exist:

Definition 6.2.4 (Positive Rate Parity)**.** Given a joint distribution $\mathcal{D}$, a deterministic classifier h satisfies *positive rate parity* if $\mathcal{D}_0(h(X) = 1 \mid Y = y) = \mathcal{D}_1(h(X) = 1 \mid Y = y), \forall y \in \{0, 1\}$.

Positive rate parity is also known as *equalized odds* (Hardt et al., 2016), which essentially requires equal true positive and false positive rates between different groups. Furthermore, Hardt et al. (2016) also defined *true positive parity*, or *equal opportunity*, to be $\mathcal{D}_0(h(X) = 1 \mid Y = 1) = \mathcal{D}_1(h(X) = 1 \mid Y = 1)$ when positive outcome is desirable. Last but not least, *predictive rate parity*, also known as *test fairness* (Chouldechova, 2017), asks for equal chance of positive outcomes across groups given predictions:

Definition 6.2.5 (Predictive Rate Parity)**.** Given a joint distribution $\mathcal{D}$, a probabilistic classifier h satisfies *predictive rate parity* if $\mathcal{D}_0(Y = 1 \mid h(X) = c) = \mathcal{D}_1(Y = 1 \mid h(X) = c), \forall c \in [0, 1]$.

When h is a deterministic binary classifier that only takes value in $\{0, 1\}$, Chouldechova (2017) showed an intrinsic tradeoff between predictive rate parity and positive rate parity:

Theorem 6.2.1 (Chouldechova (2017))**.** Assume $\mathcal{D}_0(Y = 1) \neq \mathcal{D}_1(Y = 1)$, then for any deterministic classifier $h : \mathcal{X} \to \{0, 1\}$ that is not perfect, i.e., $h(X) \neq Y$, positive rate parity and predictive rate parity cannot hold simultaneously.

Similar tradeoff result for probabilistic classifier has also been observed by Kleinberg et al. (2016), where the authors showed that for any non-perfect predictors, calibration and positive rate parity cannot be achieved simultaneously if the base rates are different across groups. Here a classifier h is said to be *calibrated* if $\mathcal{D}(Y = 1 \mid h(X) = c) = c, \forall c \in [0, 1]$, i.e., if we look at the set of data that receive a predicted probability of c by h, we would like c-fraction of them to be positive instances according to Y (Pleiss et al., 2017).

f-divergence Introduced by Ali and Silvey (1966) and Csiszár (1964, 1967), f-divergence, also known as the Ali-Silvey distance, is a general class of statistical divergences to measure the difference between two probability distributions $\mathcal{P}$ and $\mathcal{Q}$ over the same measurable space.

Definition 6.2.6 (f-divergence)**.** Let $\mathcal{P}$ and $\mathcal{Q}$ be two probability distributions over the same space and assume $\mathcal{P}$ is absolutely continuous w.r.t. $\mathcal{Q}$. Then for any convex function $f : (0, \infty) \to \mathbb{R}$ that is strictly convex at 1 and $f(1) = 0$, the f-divergence of $\mathcal{Q}$ from $\mathcal{P}$ is defined as

$$D_f(\mathcal{P} \parallel \mathcal{Q}) := \mathbb{E}_{\mathcal{Q}}\left[f\left(\frac{d\mathcal{P}}{d\mathcal{Q}}\right)\right]. \tag{6.1}$$

The function f is called the *generator function* of $D_f(\cdot \parallel \cdot)$.

Different choices of the generator function f recover popular statistical divergence as special cases, e.g., the KL-divergence. From Jensen's inequality it is easy to verify that $D_f(\mathcal{P} \parallel \mathcal{Q}) \geq 0$ and $D_f(\mathcal{P} \parallel \mathcal{Q}) = 0$ iff $\mathcal{P} = \mathcal{Q}$ almost surely. Note that f-divergence does not necessarily leads to a distance metric, and it is not symmetric in general, i.e., $D_f(\mathcal{P} \parallel \mathcal{Q}) \neq D_f(\mathcal{Q} \parallel \mathcal{P})$ provided that $\mathcal{P} \sim \mathcal{Q}$. We list some common choices of the generator function f and their corresponding properties in Table 6.2.1. Notably, Khosravifard et al. (2007) proved that total variation is the only f-divergence that serves as a metric, i.e., satisfying the triangle inequality.

Table 6.2.1: List of different f-divergences and their corresponding properties. $D_{\mathrm{KL}}(\mathcal{P} \mid\mid \mathcal{Q})$ denotes the KL-divergence of $\mathcal{Q}$ from $\mathcal{P}$ and $\mathcal{M} := (\mathcal{P} + \mathcal{Q})/2$ is the average distribution of $\mathcal{P}$ and $\mathcal{Q}$. Symm. stands for Symmetric and Tri. stands for Triangle Inequality.

Name	$D_f(\mathcal{P} \parallel \mathcal{Q})$	**Generator** $f(t)$	**Symm.**	**Tri.**
Kullback-Leibler	$D_{\mathrm{KL}}(\mathcal{P} \parallel \mathcal{Q})$	$t \log t$	✗	✗
Reverse-KL	$D_{\mathrm{KL}}(\mathcal{Q} \parallel \mathcal{P})$	$-\log t$	✗	✗
Jensen-Shannon	$D_{\mathrm{JS}}(\mathcal{P}, \mathcal{Q}) := \frac{1}{2}(D_{\mathrm{KL}}(\mathcal{P} \Vert \mathcal{M}) + D_{\mathrm{KL}}(\mathcal{Q} \Vert \mathcal{M}))$	$t \log t - (t+1)\log(\frac{t+1}{2})$	✓	✗
Squared Hellinger	$H^2(\mathcal{P}, \mathcal{Q}) := \frac{1}{2}\int(\sqrt{d\mathcal{P}} - \sqrt{d\mathcal{Q}})^2$	$(1-\sqrt{t})^2/2$	✓	✗
Total Variation	$d_{\mathrm{TV}}(\mathcal{P}, \mathcal{Q}) := \sup_E \lvert\mathcal{P}(E) - \mathcal{Q}(E)\rvert$	$\lvert t-1\rvert/2$	✓	✓

6.3 Main Results

As we briefly mentioned in Section 6.2, it is impossible to have imperfect predictor that is both calibrated and preserves positive rate parity when the base rates differ between two groups. Similar impossibility result also holds between positive rate parity and predictive rate parity. On the other hand, while it has long been observed that demographic parity may eliminate perfect predictor (Hardt et al., 2016), and previous work has empirically verified that tradeoff exists between accuracy and demographic parity (Calders et al., 2009; Kamiran and Calders, 2009; Zliobaite, 2015) on various datasets, so far a quantitative characterization on the exact tradeoff between accuracy and various notions of parity is still missing. In what follows we shall prove a family of information theoretic lower bounds on the accuracy that hold for *all* algorithms.

6.3.1 Tradeoff between Fairness and Utility

Essentially, every prediction function induces a Markov chain: $X \xrightarrow{g} Z \xrightarrow{h} \widehat{Y}$, where g is the feature transformation, h is the classifier on feature space, Z is the feature and $\widehat{Y}$ is the predicted target variable by $h \circ g$. Note that simple models, e.g., linear classifiers, are also included by specifying g to be the identity map. With this notation, we first state the following theorem that quantifies an inherent tradeoff between fairness and utility.

Theorem 6.3.1. Let $\widehat{Y} = h(g(X))$ be the predictor. If $\widehat{Y}$ satisfies demographic parity, then $\varepsilon_{\mathcal{D}_0}(h \circ g) + \varepsilon_{\mathcal{D}_1}(h \circ g) \geq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1)$.

Remark First of all, $\Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1)$ is the difference of base rates across groups, and it achieves its maximum value of 1 iff $Y = A$ almost surely, i.e., Y indicates group membership. On the other hand, if Y is independent of A, then $\Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) = 0$ so the lower bound does not make any constraint on the joint error. Second, Theorem 6.3.1 applies to all possible feature transformation g and predictor h. In particular, if we choose g to be the identity map, then Theorem 6.3.1 says that when the base rates differ, *no algorithm* can achieve a small joint error on both groups, and it also recovers the previous observation that demographic parity can eliminate the perfect predictor (Hardt et al., 2016). Third, the lower bound in Theorem 6.3.1 is insensitive to the marginal distribution of A, i.e., it treats the errors from both groups equally. As a comparison, let $\alpha := \mathcal{D}(A = 1)$, then $\varepsilon_{\mathcal{D}}(h \circ g) = (1-\alpha)\varepsilon_{\mathcal{D}_0}(h \circ g) + \alpha\varepsilon_{\mathcal{D}_1}(h \circ g)$. In this case $\varepsilon_{\mathcal{D}}(h \circ g)$ could still be small even if the minority group suffers a large error.

Furthermore, by the pigeonhole principle, the following corollary holds:

Corollary 6.3.1. If the predictor $\widehat{Y} = h(g(X))$ satisfies demographic parity, then $\max\{\varepsilon_{\mathcal{D}_0}(h \circ g), \varepsilon_{\mathcal{D}_1}(h \circ g)\} \geq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1)/2$.

In words, this means that for fair predictors in the demographic parity sense, at least one of the subgroups has to incur an error of at least $\Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)/2$ which could be large in settings like criminal justice where $\Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)$ is large.

Before we give the proof, we first present a useful lemma that lower bounds the prediction error by the total variation distance.

Lemma 6.3.1. Let $\widehat{Y}=h(g(X))$ be the predictor, then for $a\in\{0,1\}$, $d_{\mathrm{TV}}(\mathcal{D}_a(Y),\mathcal{D}_a(\widehat{Y}))\leq\varepsilon_{\mathcal{D}_a}(h\circ g)$.

Proof. For $a\in\{0,1\}$, we have:

$$\begin{aligned}d_{\mathrm{TV}}(\mathcal{D}_a(Y),\mathcal{D}_a(\widehat{Y}))&=|\mathcal{D}_a(Y=1)-\mathcal{D}_a(h(g(X))=1)|=|\mathbb{E}_{\mathcal{D}_a}[Y]-\mathbb{E}_{\mathcal{D}_a}[h(g(X))]|\\&\leq\mathbb{E}_{\mathcal{D}_a}[|Y-h(g(X))|]=\varepsilon_{\mathcal{D}_a}(h\circ g).\end{aligned}\qquad\blacksquare$$

Now we are ready to prove Theorem 6.3.1:

Proof of Theorem 6.3.1. First of all, we show that if $\widehat{Y}=h(g(X))$ satisfies demographic parity, then:

$$\begin{aligned}d_{\mathrm{TV}}(\mathcal{D}_0(\widehat{Y}),\mathcal{D}_1(\widehat{Y}))&=\max\left\{|\mathcal{D}_0(\widehat{Y}=0)-\mathcal{D}_1(\widehat{Y}=0)|,\ |\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)|\right\}\\&=|\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)|\\&=|\mathcal{D}(\widehat{Y}=1\mid A=0)-\mathcal{D}(\widehat{Y}=1\mid A=1)|=0,\end{aligned}$$

where the last equality follows from the definition of demographic parity. Now from Table 6.2.1, $d_{\mathrm{TV}}(\cdot,\cdot)$ is symmetric and satisfies the triangle inequality, we have:

$$\begin{aligned}d_{\mathrm{TV}}(\mathcal{D}_0(Y),\mathcal{D}_1(Y))&\leq d_{\mathrm{TV}}(\mathcal{D}_0(Y),\mathcal{D}_0(\widehat{Y}))+d_{\mathrm{TV}}(\mathcal{D}_0(\widehat{Y}),\mathcal{D}_1(\widehat{Y}))+d_{\mathrm{TV}}(\mathcal{D}_1(\widehat{Y}),\mathcal{D}_1(Y))\\&=d_{\mathrm{TV}}(\mathcal{D}_0(Y),\mathcal{D}_0(\widehat{Y}))+d_{\mathrm{TV}}(\mathcal{D}_1(\widehat{Y}),\mathcal{D}_1(Y)).\end{aligned}\tag{6.2}$$

The last step is to bound $d_{\mathrm{TV}}(\mathcal{D}_a(Y),\mathcal{D}_a(\widehat{Y}))$ in terms of $\varepsilon_{\mathcal{D}_a}(h\circ g)$ for $a\in\{0,1\}$ using Lemma 6.3.1:

$$d_{\mathrm{TV}}(\mathcal{D}_0(Y),\mathcal{D}_0(\widehat{Y}))\leq\varepsilon_{\mathcal{D}_0}(h\circ g),\quad d_{\mathrm{TV}}(\mathcal{D}_1(Y),\mathcal{D}_1(\widehat{Y}))\leq\varepsilon_{\mathcal{D}_1}(h\circ g).$$

Combining the above two inequalities and (6.2) completes the proof. $\blacksquare$

It is not hard to show that our lower bound in Theorem 8.2.1 is tight. To see this, consider the case $A=Y$, where the lower bound achieves its maximum value of 1. Now consider a constant predictor $\widehat{Y}\equiv1$ or $\widehat{Y}\equiv0$, which clearly satisfies demographic parity by definition. But in this case either $\varepsilon_{\mathcal{D}_0}(h\circ g)=1,\varepsilon_{\mathcal{D}_1}(h\circ g)=0$ or $\varepsilon_{\mathcal{D}_0}(h\circ g)=0,\varepsilon_{\mathcal{D}_1}(h\circ g)=1$, hence $\varepsilon_{\mathcal{D}_0}(h\circ g)+\varepsilon_{\mathcal{D}_1}(h\circ g)\equiv1$, achieving the lower bound.

To conclude this section, we point out that the choice of total variation in the lower bound is not unique. As we will see shortly in Section 6.3.2, similar lower bounds could be attained using specific choices of the general f-divergence with some desired properties.

6.3.2 Tradeoff in Fair Representation Learning

In the last section we show that there is an inherent tradeoff between fairness and utility when a predictor *exactly* satisfies demographic parity. In practice we may not be able to achieve demographic parity exactly. Instead, most algorithms (Adel et al., 2019a; Beutel et al., 2017; Edwards and Storkey, 2015; Louizos et al., 2015) build an adversarial discriminator that takes as input the feature vector $Z=g(X)$, and the goal is to learn fair representations such that it is hard for the adversarial discriminator to infer the group membership from Z. In this sense due to the limit on the capacity of the adversarial discriminator, only approximate demographic parity can be achieved in the equilibrium. Hence it is natural to ask what is the tradeoff between fair representations and accuracy in this scenario? In this section we shall answer this question by generalizing our previous analysis with f-divergence to prove a family of lower bounds on the joint target prediction error. Our results also show how approximate DP helps to reconcile but not remove the tradeoff between fairness and utility. Before we state and prove the main results in this section, we first introduce the following lemma by Liese and Vajda (2006) as a generalization of the data processing inequality for f-divergence:

Lemma 6.3.2 (Liese and Vajda (2006))**.** Let $\mu(\mathcal{Z})$ be the space of all probability distributions over $\mathcal{Z}$. Then for any f-divergence $D_f(\cdot \,||\, \cdot)$, any stochastic kernel $\kappa : \mathcal{X} \to \mu(\mathcal{Z})$, and any distributions $\mathcal{P}$ and $\mathcal{Q}$ over $\mathcal{X}$, $D_f(\kappa\mathcal{P} \,||\, \kappa\mathcal{Q}) \leq D_f(\mathcal{P} \,||\, \mathcal{Q})$.

Roughly speaking, Lemma 6.3.2 says that data processing cannot increase discriminating information. Define $d_{\text{JS}}(\mathcal{P}, \mathcal{Q}) := \sqrt{D_{\text{JS}}(\mathcal{P}, \mathcal{Q})}$ and $H(\mathcal{P}, \mathcal{Q}) := \sqrt{H^2(\mathcal{P}, \mathcal{Q})}$. It is well-known in information theory that both $d_{\text{JS}}(\cdot, \cdot)$ and $H(\cdot, \cdot)$ form a bounded distance metric over the space of probability distributions. Realize that $d_{\text{TV}}(\cdot, \cdot)$, $H^2(\cdot, \cdot)$ and $D_{\text{JS}}(\cdot, \cdot)$ are all f-divergence. The following corollary holds:

Corollary 6.3.2. Let $h : \mathcal{Z} \to \mathcal{Y}$ be any hypothesis, and $g_\sharp\mathcal{D}_a$ be the pushforward distribution of $\mathcal{D}_a$ by g, $\forall a \in \{0, 1\}$. Let $\widehat{Y} = h(g(X))$ be the predictor, then all the following inequalities hold:

1. $d_{\text{TV}}(\mathcal{D}_0(\widehat{Y}), \mathcal{D}_1(\widehat{Y})) \leq d_{\text{TV}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1)$
2. $H(\mathcal{D}_0(\widehat{Y}), \mathcal{D}_1(\widehat{Y})) \leq H(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1)$
3. $d_{\text{JS}}(\mathcal{D}_0(\widehat{Y}), \mathcal{D}_1(\widehat{Y})) \leq d_{\text{JS}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1)$

Now we are ready to present the following main theorem of this section:

Theorem 6.3.2. Let $\widehat{Y} = h(g(X))$ be the predictor. Assume $d_{\text{JS}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1) \leq d_{\text{JS}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y))$ and $H(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1) \leq H(\mathcal{D}_0(Y), \mathcal{D}_1(Y))$, then the following three inequalities hold:

1. Total variation lower bound:

$$\varepsilon_{\mathcal{D}_0}(h \circ g) + \varepsilon_{\mathcal{D}_1}(h \circ g) \geq d_{\text{TV}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y)) - d_{\text{TV}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1).$$

2. Jensen-Shannon lower bound:

$$\varepsilon_{\mathcal{D}_0}(h \circ g) + \varepsilon_{\mathcal{D}_1}(h \circ g) \geq \big(d_{\text{JS}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y)) - d_{\text{JS}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1)\big)^2/2.$$

3. Hellinger lower bound:

$$\varepsilon_{\mathcal{D}_0}(h \circ g) + \varepsilon_{\mathcal{D}_1}(h \circ g) \geq \big(H(\mathcal{D}_0(Y), \mathcal{D}_1(Y)) - H(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1)\big)^2/2.$$

Remark All the three lower bounds in Theorem 6.3.2 imply a tradeoff between the joint error across demographic subgroups and learning group-invariant feature representations. In a nutshell:

> *For fair representations, it is not possible to construct a predictor that simultaneously minimizes the errors on both demographic subgroups.*

When $g_\sharp\mathcal{D}_0 = g_\sharp\mathcal{D}_1$, which also implies $\mathcal{D}_0(\widehat{Y}) = \mathcal{D}_1(\widehat{Y})$, all three lower bounds get larger, in this case we have

$$\begin{aligned}\max\left\{d_{\text{TV}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y)), \frac{1}{2}d_{\text{JS}}^2(\mathcal{D}_0(Y), \mathcal{D}_1(Y)), \frac{1}{2}H^2(\mathcal{D}_0(Y), \mathcal{D}_1(Y))\right\} &= d_{\text{TV}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y)) \\ &= \Delta_{\text{BR}}(\mathcal{D}_0, \mathcal{D}_1),\end{aligned}$$

and this reduces to Theorem 6.3.1. Now we give a sketch of the proof for Theorem 6.3.2:

Proof Sketch of Theorem 6.3.2. We prove the three inequalities respectively. The total variation lower bound follows the same idea as the proof of Theorem 8.2.1 and the inequality $d_{\mathrm{TV}}(\mathcal{D}_0(\widehat{Y}), \mathcal{D}_1(\widehat{Y})) \leq d_{\mathrm{TV}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1)$ from Corollary 6.3.2. To prove the Jensen-Shannon lower bound, realize that $d_{\mathrm{JS}}(\cdot,\cdot)$ is a distance metric over probability distributions. Combining with the inequality $d_{\mathrm{JS}}(\mathcal{D}_0(\widehat{Y}), \mathcal{D}_1(\widehat{Y})) \leq d_{\mathrm{JS}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1)$ from Corollary 6.3.2, we have:

$$d_{\mathrm{JS}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y)) \leq d_{\mathrm{JS}}(\mathcal{D}_0(Y), \mathcal{D}_0(\widehat{Y})) + d_{\mathrm{JS}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1) + d_{\mathrm{JS}}(\mathcal{D}_1(\widehat{Y}), \mathcal{D}_1(Y)).$$

Now by Lin's lemma (Lin, 1991, Theorem 3), for any two distributions $\mathcal{P}$ and $\mathcal{Q}$, we have $d^2_{\mathrm{JS}}(\mathcal{P}, \mathcal{Q}) \leq d_{\mathrm{TV}}(\mathcal{P}, \mathcal{Q})$. Combine Lin's lemma with Lemma 6.3.1, we get the following lower bound:

$$\sqrt{\varepsilon_{\mathcal{D}_0}(h \circ g)} + \sqrt{\varepsilon_{\mathcal{D}_1}(h \circ g)} \geq d_{\mathrm{JS}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y)) - d_{\mathrm{JS}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1).$$

Apply the AM-GM inequality, we can further bound the L.H.S. by

$$\sqrt{2\big(\varepsilon_{\mathcal{D}_0}(h \circ g) + \varepsilon_{\mathcal{D}_1}(h \circ g)\big)} \geq \sqrt{\varepsilon_{\mathcal{D}_0}(h \circ g)} + \sqrt{\varepsilon_{\mathcal{D}_1}(h \circ g)}.$$

Under the assumption that $d_{\mathrm{JS}}(g_\sharp\mathcal{D}_0, g_\sharp\mathcal{D}_1) \leq d_{\mathrm{JS}}(\mathcal{D}_0(Y), \mathcal{D}_1(Y))$, taking a square at both sides then completes the proof for the second inequality. The proof for Hellinger's lower bound follows exactly as the one for Jensen-Shannon's lower bound, except that instead of Lin's lemma, we need to use the fact that $H^2(\mathcal{P}, \mathcal{Q}) \leq d_{\mathrm{TV}}(\mathcal{P}, \mathcal{Q}) \leq \sqrt{2}H(\mathcal{P}, \mathcal{Q}), \forall \mathcal{P}, \mathcal{Q}$. ■

As a simple corollary of Theorem 6.3.2, the following result shows how approximate DP (in terms of the DP gap) helps to reconcile the tradeoff between fairness and utility:

Corollary 6.3.3. Let $\widehat{Y} = h(g(X))$ be the predictor, then $\varepsilon_{\mathcal{D}_0}(h \circ g) + \varepsilon_{\mathcal{D}_1}(h \circ g) \geq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) - \Delta_{\mathrm{DP}}(\widehat{Y})$.

In a sense Corollary 6.3.3 means that in order to lower the joint error, the DP gap of the predictor cannot be too small. Of course, since the above inequality is a lower bound, it only serves as a necessary condition for small joint error. Hence an interesting question would be to ask whether it is possible to have a sufficient condition that guarantees a small joint error such that the DP gap of the predictor is no larger than that of the perfect predictor, i.e., $\Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1)$. We leave this as a future work.

6.3.3 Fair Representations Lead to Accuracy Parity

In the previous sections we prove a family of information-theoretic lower bounds that demonstrate an inherent tradeoff between fair representations and joint error across groups. A natural question to ask then, is, what kind of parity can fair representations bring us? To complement our negative results, in this section we show that learning group-invariant representations help to reduce discrepancy of errors (utilities) across groups.

First of all, since we work under the stochastic setting where $\mathcal{D}_a$ is a joint distribution over X and Y conditioned on $A = a$, then any function mapping $h : \mathcal{X} \to \mathcal{Y}$ will inevitably incur an error due to the noise existed in the distribution $\mathcal{D}_a$. Formally, for $a \in \{0, 1\}$, define the optimal function $h^*_a : \mathcal{X} \to \mathcal{Y}$ under the absolute error to be $h^*_a(X) := m_{\mathcal{D}_a}(Y \mid X)$, where $m_{\mathcal{D}_a}(Y \mid X)$ denotes the median of Y given X under distribution $\mathcal{D}_a$. Now define the noise of distribution $\mathcal{D}_a$ to be $n_{\mathcal{D}_a} := \mathbb{E}_{\mathcal{D}_a}[|Y - h^*_a(X)|]$. With these notations, we are now ready to present the following theorem:

Theorem 6.3.3 (Error Decomposition Theorem)**.** For any hypothesis $\mathcal{H} \ni h : \mathcal{X} \to \mathcal{Y}$, the following inequality holds:

$$\begin{aligned}|\varepsilon_{\mathcal{D}_0}(h) - \varepsilon_{\mathcal{D}_1}(h)| \le |n_{\mathcal{D}_0} - n_{\mathcal{D}_1}| &+ d_{\mathrm{TV}}(\mathcal{D}_0(X), \mathcal{D}_1(X)) \\ &+ \min\left\{\mathbb{E}_{\mathcal{D}_0}[|h_0^* - h_1^*|], \mathbb{E}_{\mathcal{D}_1}[|h_0^* - h_1^*|]\right\}.\end{aligned}$$

Remark Theorem 6.3.3 upper bounds the discrepancy of accuracy across groups by three terms: the noise difference, the distance of representations across groups and the discrepancy of optimal decision functions. In an ideal setting where both distributions are noiseless, i.e., same people in the same group are always treated equally, the upper bound simplifies to the latter two terms:

$$|\varepsilon_{\mathcal{D}_0}(h) - \varepsilon_{\mathcal{D}_1}(h)| \le d_{\mathrm{TV}}(\mathcal{D}_0(X), \mathcal{D}_1(X)) + \min\left\{\mathbb{E}_{\mathcal{D}_0}[|h_0^* - h_1^*|], \mathbb{E}_{\mathcal{D}_1}[|h_0^* - h_1^*|]\right\}.$$

If we further require that the optimal decision functions h_0^* and h_1^* are close to each other, i.e., optimal decisions are insensitive to the group membership, then Theorem 6.3.3 implies that a sufficient condition to guarantee accuracy parity is to find group-invariant representation that minimizes $d_{\mathrm{TV}}(\mathcal{D}_0(X), \mathcal{D}_1(X))$. We now present the proof for Theorem 6.3.3:

Proof of Theorem 6.3.3. First, we show that for $a \in \{0, 1\}$, $\varepsilon_{\mathcal{D}_a}(h)$ cannot be too large if h is close to h_a^*:

$$\begin{aligned}|\varepsilon_{\mathcal{D}_a}(h) - \mathbb{E}_{\mathcal{D}_a}[|h(X) - h_a^*(X)|]| &= |\mathbb{E}_{\mathcal{D}_a}[|h(X) - Y|] - \mathbb{E}_{\mathcal{D}_a}[|h(X) - h_a^*(X)|]| \\ &\le \mathbb{E}_{\mathcal{D}_a}[||h(X) - Y| - |h(X) - h_a^*(X)||] \\ &\le \mathbb{E}_{\mathcal{D}_a}[|Y - h_a^*(X)|] = n_{\mathcal{D}_a},\end{aligned}$$

where both inequalities are due to the triangle inequality. Next, we bound $|\varepsilon_{\mathcal{D}_0}(h) - \varepsilon_{\mathcal{D}_1}(h)|$ by:

$$|\varepsilon_{\mathcal{D}_0}(h) - \varepsilon_{\mathcal{D}_1}(h)| \le |n_{\mathcal{D}_0} - n_{\mathcal{D}_1}| + \big|\mathbb{E}_{\mathcal{D}_0}[|h(X) - h_0^*(X)|] - \mathbb{E}_{\mathcal{D}_1}[|h(X) - h_1^*(X)|]\big|.$$

In order to show this, define $\varepsilon_a(h, h') := \mathbb{E}_{\mathcal{D}_a}[|h(X) - h'(X)|]$ so that

$$\big|\mathbb{E}_{\mathcal{D}_0}[|h(X) - h_0^*(X)|] - \mathbb{E}_{\mathcal{D}_1}[|h(X) - h_1^*(X)|]\big| = \big|\varepsilon_0(h, h_0^*) - \varepsilon_1(h, h_1^*)\big|.$$

To bound $\big|\varepsilon_0(h, h_0^*) - \varepsilon_1(h, h_1^*)\big|$, realize that $|h(X) - h_a^*(X)| \in \{0, 1\}$. On one hand, we have:

$$\begin{aligned}\big|\varepsilon_0(h, h_0^*) - \varepsilon_1(h, h_1^*)\big| &= \big|\varepsilon_0(h, h_0^*) - \varepsilon_0(h, h_1^*) + \varepsilon_0(h, h_1^*) - \varepsilon_1(h, h_1^*)\big| \\ &\le \big|\varepsilon_0(h, h_0^*) - \varepsilon_0(h, h_1^*)\big| + \big|\varepsilon_0(h, h_1^*) - \varepsilon_1(h, h_1^*)\big| \\ &\le \varepsilon_0(h_0^*, h_1^*) + d_{\mathrm{TV}}(\mathcal{D}_0(X), \mathcal{D}_1(X)),\end{aligned}$$

where the last inequality is due to $\big|\varepsilon_0(h, h_1^*) - \varepsilon_1(h, h_1^*)\big| = \big|\mathcal{D}_0(|h - h_1^*| = 1) - \mathcal{D}_1(|h - h_1^*| = 1)\big| \le \sup_E |\mathcal{D}_0(E) - \mathcal{D}_1(E)| = d_{\mathrm{TV}}(\mathcal{D}_0, \mathcal{D}_1)$. Similarly, by subtracting and adding back $\varepsilon_1(h, h_0^*)$ instead, we can also show that $\big|\varepsilon_0(h, h_0^*) - \varepsilon_1(h, h_1^*)\big| \le \varepsilon_1(h_0^*, h_1^*) + d_{\mathrm{TV}}(\mathcal{D}_0(X), \mathcal{D}_1(X))$.

Combine the above two inequalities yielding:

$$\big|\varepsilon_0(h, h_0^*) - \varepsilon_1(h, h_1^*)\big| \le \min\{\varepsilon_0(h_0^*, h_1^*), \varepsilon_1(h_0^*, h_1^*)\} + d_{\mathrm{TV}}(\mathcal{D}_0(X), \mathcal{D}_1(X)).$$

Incorporating the difference of noise back to the above inequality then completes the proof. ■

6.4 Empirical Validation

Our theoretical results on the lower bound imply that over-training the feature transformation function to achieve group-invariant representations will inevitably lead to large joint errors. On the other hand, our upper bound also implies that group-invariant representations help to achieve accuracy parity. To verify these theoretical implications, in this section we conduct experiments on a real-world benchmark dataset, the UCI Adult dataset, to present empirical results with various metrics.

Dataset The Adult dataset contains 30,162/15,060 training/test instances for income prediction. Each instance in the dataset describes an adult from the 1994 US Census. Attributes include gender, education level, age, etc. In this experiment we use gender (binary) as the sensitive attribute, and we preprocess the dataset to convert categorical variables into one-hot representations. The processed data contains 114 attributes. The target variable (income) is also binary: 1 if $\geq$ 50K/year otherwise 0. For the sensitive attribute A, $A = 0$ means Male otherwise Female. In this dataset, the base rates across groups are different: $\Pr(Y = 1 \mid A = 0) = 0.310$ while $\Pr(Y = 1 \mid A = 1) = 0.113$. Also, the group ratios are different: $\Pr(A = 0) = 0.673$.

Experimental Protocol To validate the effect of learning group-invariant representations with adversarial debiasing techniques (Beutel et al., 2017; Madras et al., 2018; Zhang et al., 2018), we perform a controlled experiment by fixing the baseline network architecture to be a three hidden-layer feed-forward network with ReLU activations. The number of units in each hidden layer are 500, 200, and 100, respectively. The output layer corresponds to a logistic regression model. This baseline without debiasing is denoted as NoDebias. For debiasing with adversarial learning techniques, the adversarial discriminator network takes the feature from the last hidden layer as input, and connects it to a hidden-layer with 50 units, followed by a binary classifier whose goal is to predict the sensitive attribute A. This model is denoted as AdvDebias. Compared with NoDebias, the only difference of AdvDebias in terms of objective function is that besides the cross-entropy loss for target prediction, the AdvDebias also contains a classification loss from the adversarial discriminator to predict the sensitive attribute A. In the experiment, all the other factors are fixed to be the same between these two methods, including learning rate, optimization algorithm, training epoch, and also batch size. To see how the adversarial loss affects the joint error, the demographic parity as well as the accuracy parity, we vary the coefficient ρ for the adversarial loss between 0.1, 1.0, 5.0 and 50.0.

Results and Analysis The experimental results are listed in Table 6.4.1. Note that in the table $|\varepsilon_{\mathcal{D}_0} - \varepsilon_{\mathcal{D}_1}|$ could be understood as measuring an approximate version of accuracy parity, and similarly $\Delta_{\mathrm{DP}}(\widehat{Y})$ measures the closeness of the classifier to satisfy demographic parity. From the table, it is then clear that with increasing ρ, both the overall error $\varepsilon_{\mathcal{D}}$ (sensitive to the marginal distribution of A) and the joint error $\varepsilon_{\mathcal{D}_0} + \varepsilon_{\mathcal{D}_1}$ (insensitive to the imbalance of A) are increasing. As expected, $\Delta_{\mathrm{DP}}(\widehat{Y})$ is drastically decreasing with the increasing of ρ. Furthermore, $|\varepsilon_{\mathcal{D}_0} - \varepsilon_{\mathcal{D}_1}|$ is also gradually decreasing, but much slowly than $\Delta_{\mathrm{DP}}(\widehat{Y})$. This is due to the existing noise in the data as well as the shift between the optimal decision functions across groups, as indicated by our upper bound. To conclude, all the empirical results are consistent with our theoretical findings.

6.5 Related Work

Fairness Frameworks Two central notions of fairness have been extensively studied, i.e., group fairness and individual fairness. In a seminal work, Dwork et al. (2012) define individual fairness as a measure of

Table 6.4.1: Adversarial debiasing on demographic parity, joint error across groups, and accuracy parity.

	$\varepsilon_{\mathcal{D}}$	$\varepsilon_{\mathcal{D}_0} + \varepsilon_{\mathcal{D}_1}$	$\lvert\varepsilon_{\mathcal{D}_0} - \varepsilon_{\mathcal{D}_1}\rvert$	$\Delta_{\mathrm{DP}}(\widehat{Y})$
NoDebias	0.157	0.275	0.115	0.189
AdvDebias, $\rho = 0.1$	0.159	0.278	0.116	0.190
AdvDebias, $\rho = 1.0$	0.162	0.286	0.106	0.113
AdvDebias, $\rho = 5.0$	0.166	0.295	0.106	0.032
AdvDebias, $\rho = 50.0$	0.201	0.360	0.112	0.028

smoothness of the classification function. Under the assumption that number of individuals is finite, the authors proposed a linear programming framework to maximize the utility under their fairness constraint. However, their framework requires apriori a distance function that computes the similarity between individuals, and their optimization formulation does not produce an inductive rule to generalize to unseen data. Based on the definition of positive rate parity, Hardt et al. (2016) proposed a post-processing method to achieve fairness by taking as input the prediction and the sensitive attribute. In a concurrent work, Kleinberg et al. (2016) offer a calibration technique to achieve the corresponding fairness criterion as well. However, both of the aforementioned two approaches require sensitive attribute during the inference phase, which is not available in many real-world scenarios.

Regularization Techniques The line of work on fairness-aware learning through regularization dates at least back to Kamishima et al. (2012), where the authors argue that simple deletion of sensitive features in data is insufficient for eliminating biases in automated decision making, due to the possible correlations among attributes and sensitive information (Lum and Johndrow, 2016). In light of this, the authors proposed a *prejudice remover* regularizer that essentially penalizes the mutual information between the predicted goal and the sensitive information. In a more recent approach, Zafar et al. (2015) leveraged a measure of decision boundary fairness and incorporated it via constraints into the objective function of logistic regression as well as support vector machines. As discussed in Section 7.2, both approaches essentially reduce to achieving demographic parity through regularization.

Representation Learning In a pioneer work, Zemel et al. (2013) proposed to preserve both group and individual fairness through the lens of representation learning, where the main idea is to find a good representation of the data with two competing goals: to encode the data for utility maximization while at the same time to obfuscate any information about membership in the protected group. Due to the power of learning rich representations offered by deep neural nets, recent advances in building fair automated decision making systems focus on using adversarial techniques to learn fair representation that also preserves enough information for the prediction vendor to achieve his utility (Adel et al., 2019a; Beutel et al., 2017; Edwards and Storkey, 2015; Louizos et al., 2015; Song et al., 2019; Zhang et al., 2018; Zhao et al., 2019d). Madras et al. (2018) further extended this approach by incorporating reconstruction loss given by an autoencoder into the objective function to preserve demographic parity, equalized odds, and equal opportunity.

6.6 Conclusion

In this chapter we theoretically and empirically study the important problem of quantifying the tradeoff between utility and fairness in learning group-invariant representations. Specifically, we prove a novel lower bound to characterize the tradeoff between demographic parity and the joint utility across different population groups when the base rates differ between groups. In particular, our results imply that any method aiming to learn fair representations admits an information-theoretic lower bound on the joint error, and the better the representation, the larger the joint error. Complementary to our negative results, we also show that learning fair representations leads to accuracy parity if the optimal decision functions across different groups are close. These theoretical findings are also confirmed empirically on real-world datasets. We believe our results take an important step towards better understanding the tradeoff between utility and different notions of fairness. Inspired by our lower bound, one interesting direction for future work is to design instance-weighting algorithm to balance the base rates during representation learning.

Chapter 7

Conditional Learning of Fair Representations

In this chapter we propose a novel algorithm for learning fair representations that can simultaneously mitigate two notions of disparity among different demographic subgroups in the classification setting. Two key components underpinning the design of our algorithm are balanced error rate and conditional alignment of representations. We show how these two components contribute to ensuring accuracy parity and equalized false-positive and false-negative rates across groups without impacting demographic parity. Furthermore, we also demonstrate both in theory and on two real-world experiments that the proposed algorithm leads to a better utility-fairness trade-off on balanced datasets compared with existing algorithms on learning fair representations for classification.

7.1 Introduction

High-stakes settings, such as loan approvals, criminal justice, and hiring processes, use machine learning tools to help make decisions. A key question in these settings is whether the algorithm makes fair decisions. In settings that have historically had discrimination, we are interested in defining fairness with respect to a protected group, the group which has historically been disadvantaged. The rapidly growing field of algorithmic fairness has a vast literature that proposes various fairness metrics, characterizes the relationship between fairness metrics, and describes methods to build classifiers that satisfy these metrics (Chouldechova and Roth, 2018; Corbett-Davies and Goel, 2018). Among many recent attempts to achieve algorithmic fairness (Dwork et al., 2012; Hardt et al., 2016; Zafar et al., 2015; Zemel et al., 2013), learning fair representations has attracted increasing attention due to its flexibility in learning rich representations based on advances in deep learning (Beutel et al., 2017; Edwards and Storkey, 2015; Louizos et al., 2015; Madras et al., 2018). The backbone idea underpinning this line of work is very intuitive: if the representations of data from different groups are similar to each other, then any classifier acting on such representations will also be agnostic to the group membership.

However, it has long been empirically observed (Calders et al., 2009) and recently been proved (Zhao and Gordon, 2019) that fairness is often at odds with utility. For example, consider demographic parity, which requires the classifier to be independent of the group membership attribute. It is clear that demographic parity will cripple the utility if the demographic group membership and the target variable are indeed correlated. To escape such inherent trade-off, other notions of fairness, such as equalized odds (Hardt et al., 2016), which asks for equal false positive and negative rates across groups, and accuracy parity (Zafar et al., 2017), which seeks equalized error rates across groups, have been proposed. It is a well-known result that equalized odds is incompatible with demographic parity (Barocas et al., 2017) except in degenerate cases where group membership is independent of the target variable. Accuracy parity and the so-called predictive rate parity (c.f. Definition 7.2.4) could be simultaneously achieved, e.g., the COMPAS tool (Dieterich et al., 2016). Furthermore, under some conditions, it is also known that demographic parity can lead to accuracy parity (Zhao and Gordon, 2019). However, whether it is possible to simultaneously guarantee equalized odds and accuracy parity remains an open question.

In this chapter, we provide an affirmative answer to the above question by proposing an algorithm to align the conditional distributions (on the target variable) of representations across different demographic subgroups. The proposed formulation is a minimax problem that admits a simple reduction to cost-sensitive learning. The key component underpinning the design of our algorithm is the *balanced error rate* (BER, c.f. Section 7.2) (Feldman et al., 2015; Menon and Williamson, 2018), over the target variable and protected attributes. We demonstrate both in theory and on two real-world experiments that together with the conditional alignment, BER helps our algorithm to simultaneously ensure accuracy parity and equalized odds across groups. Our key contributions are summarized as follows:

- We prove that BER plays a fundamental role in ensuring accuracy parity and a small joint error across groups. Together with the conditional alignment of representations, this implies that we can simultaneously achieve equalized odds and accuracy parity. Furthermore, we also show that when equalized odds is satisfied, BER serves as an upper bound on the error of each subgroup. These results help to justify the design of our algorithm in using BER instead of the marginal error as our loss functions.
- We provide theoretical results that our method achieves equalized odds without impacting demographic parity. This result shows that we can preserve the demographic parity gap for free while simultaneously achieving equalized odds.
- Empirically, among existing fair representation learning methods, we demonstrate that our algorithm

is able to achieve a better utility on balanced datasets. On an imbalanced dataset, our algorithm is the only method that achieves accuracy parity; however it does so at the cost of decreased utility.

We believe our theoretical results contribute to the understanding on the relationship between equalized odds and accuracy parity, and the proposed algorithm provides an alternative in real-world scenarios where accuracy parity and equalized odds are desired.

7.2 Preliminaries

We first introduce the notations used throughout this chapter and formally describe the problem setup and various definitions of fairness explored in the literature.

Notation We use $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$ to denote the input and output space. Accordingly, we use X and Y to denote the random variables which take values in $\mathcal{X}$ and $\mathcal{Y}$, respectively. Lower case letters $\mathbf{x}$ and y are used to denote the instantiation of X and Y. To simplify the presentation, we use $A \in \{0, 1\}$ as the sensitive attribute, e.g., race, gender, etc. [1] Let $\mathcal{H}$ be the hypothesis class of classifiers. In other words, for $h \in \mathcal{H}$, $h : \mathcal{X} \to \mathcal{Y}$ is the predictor that outputs a prediction. Note that even if the predictor does not explicitly take the sensitive attribute A as input, this *fairness through blindness* mechanism can still be biased due to the potential correlations between X and A. In this chapter we study the stochastic setting where there is a joint distribution $\mathcal{D}$ over X, Y and A from which the data are sampled. To keep the notation consistent, for $a, y \in \{0, 1\}$, we use $\mathcal{D}_a$ to mean the conditional distribution of $\mathcal{D}$ given $A = a$ and $\mathcal{D}^y$ to mean the conditional distribution of $\mathcal{D}$ given $Y = y$. For an event E, $\mathcal{D}(E)$ denotes the probability of E under $\mathcal{D}$. In particular, in the literature of fair machine learning, we call $\mathcal{D}(Y = 1)$ the *base rate* of distribution $\mathcal{D}$ and we use $\Delta_{\text{BR}}(\mathcal{D}, \mathcal{D}') := |\mathcal{D}(Y = 1) - \mathcal{D}'(Y = 1)|$ to denote the difference of the base rates between two distributions $\mathcal{D}$ and $\mathcal{D}'$ over the same sample space.

Given a feature transformation function $g : \mathcal{X} \to \mathcal{Z}$ that maps instances from the input space $\mathcal{X}$ to feature space $\mathcal{Z}$, we define $g_\sharp\mathcal{D} := \mathcal{D} \circ g^{-1}$ to be the induced (pushforward) distribution of $\mathcal{D}$ under g, i.e., for any event $E' \subseteq \mathcal{Z}$, $g_\sharp\mathcal{D}(E') := \mathcal{D}(g^{-1}(E')) = \mathcal{D}(\{x \in \mathcal{X} \mid g(x) \in E'\})$. To measure the discrepancy between distributions, we use $d_{\text{TV}}(\mathcal{D}, \mathcal{D}')$ to denote the total variation between them: $d_{\text{TV}}(\mathcal{D}, \mathcal{D}') := \sup_E |\mathcal{D}(E) - \mathcal{D}'(E)|$. In particular, for binary random variable Y, it can be readily verified that the total variation between the marginal distributions $\mathcal{D}(Y)$ and $\mathcal{D}'(Y)$ reduces to the difference of their base rates, $\Delta_{\text{BR}}(\mathcal{D}, \mathcal{D}')$. To see this, realize that $d_{\text{TV}}(\mathcal{D}(Y), \mathcal{D}'(Y)) = \max\{|\mathcal{D}(Y = 1) - \mathcal{D}'(Y = 1)|, |\mathcal{D}(Y = 0) - \mathcal{D}'(Y = 0)|\} = |\mathcal{D}(Y = 1) - \mathcal{D}'(Y = 1)| = \Delta_{\text{BR}}(\mathcal{D}, \mathcal{D}')$ by definition.

Given a joint distribution $\mathcal{D}$, the error of a predictor h under $\mathcal{D}$ is defined as $\varepsilon_{\mathcal{D}}(h) := \mathbb{E}_{\mathcal{D}}[|Y - h(X)|]$. Note that for binary classification problems, when $h(X) \in \{0,1\}$, $\varepsilon_{\mathcal{D}}(h)$ reduces to the true error rate of binary classification. To make the notation more compact, we may drop the subscript $\mathcal{D}$ when it is clear from the context. Furthermore, we use $\text{CE}_{\mathcal{D}}(\widehat{Y} \parallel Y)$ to denote the cross-entropy loss function between the predicted variable $\widehat{Y}$ and the true label distribution Y over the joint distribution $\mathcal{D}$. For binary random variables Y, we define $\text{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)$ to be the *balanced error rate* of predicting Y using $\widehat{Y}$, e.g., $\text{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y) := \mathcal{D}(\widehat{Y} = 0 \mid Y = 1) + \mathcal{D}(\widehat{Y} = 1 \mid Y = 0)$. Realize that $\mathcal{D}(\widehat{Y} = 0 \mid Y = 1)$ is the false negative rate (FNR) of $\widehat{Y}$ and $\mathcal{D}(\widehat{Y} = 1 \mid Y = 0)$ corresponds to the false positive rate (FPR). So the balanced error rate can also be understood as the sum of FPR and FNR using the predictor $\widehat{Y}$. We can similarly define $\text{BER}_{\mathcal{D}}(\widehat{A} \parallel A)$ as well.

[1] Our main results could also be straightforwardly extended to the setting where A is a categorical variable.

Problem Setup We focus on group fairness where the group membership is given by the sensitive attribute A. We assume that the sensitive attribute A is available to the learner during training phase, but not inference phase. As a result, post-processing techniques to ensure fairness are not feasible under our setting. In the literature, there are many possible definitions of *fairness* (Narayanan, 2018), and in what follows we provide a brief review of the ones that are mostly relevant to this work.

Definition 7.2.1 (Demographic Parity (DP))**.** Given a joint distribution $\mathcal{D}$, a classifier $\widehat{Y}$ satisfies *demographic parity* if $\widehat{Y}$ is independent of A.

When $\widehat{Y}$ is a deterministic binary classifier, demographic parity reduces to the requirement that $\mathcal{D}_0(\widehat{Y}=1)=\mathcal{D}_1(\widehat{Y}=1)$, i.e., positive outcome is given to the two groups at the same rate. Demographic parity is also known as *statistical parity*, and it has been adopted as the default definition of fairness in a series of work (Adel et al., 2019a; Calders and Verwer, 2010; Calders et al., 2009; Edwards and Storkey, 2015; Johndrow et al., 2019; Kamiran and Calders, 2009; Kamishima et al., 2011; Louizos et al., 2015; Madras et al., 2018; Zemel et al., 2013). It is not surprising that demographic parity may cripple the utility that we hope to achieve, especially in the common scenario where the *base rates* differ between two groups (Hardt et al., 2016). Formally, the following theorem characterizes the trade-off in terms of the joint error across different groups:

Theorem 7.2.1. (Zhao and Gordon, 2019) Let $\widehat{Y}=h(g(X))$ be the classifier. Then $\varepsilon_{\mathcal{D}_0}(h\circ g)+\varepsilon_{\mathcal{D}_1}(h\circ g)\geq \Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)-d_{\mathrm{TV}}(g_\sharp\mathcal{D}_0,g_\sharp\mathcal{D}_1)$.

In this case of representations that are independent of the sensitive attribute A, then the second term $d_{\mathrm{TV}}(g_\sharp\mathcal{D}_0,g_\sharp\mathcal{D}_1)$ becomes 0, and this implies:

$$\varepsilon_{\mathcal{D}_0}(h\circ g)+\varepsilon_{\mathcal{D}_1}(h\circ g)\geq \Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1).$$

At a colloquial level, the above inequality could be understood as an uncertainty principle which says:

> *For fair representations, it is not possible to construct a predictor that simultaneously minimizes the errors on both demographic subgroups.*

More precisely, by the pigeonhole principle, the following corollary holds:

Corollary 7.2.1. If $d_{\mathrm{TV}}(g_\sharp\mathcal{D}_0,g_\sharp\mathcal{D}_1)=0$, then for any hypothesis h, $\max\{\varepsilon_{\mathcal{D}_0}(h\circ g),\varepsilon_{\mathcal{D}_1}(h\circ g)\}\geq \Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)/2$.

In words, this means that for fair representations in the demographic parity sense, at least one of the subgroups has to incur a prediction error of at least $\Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)/2$ which could be large in settings like criminal justice where $\Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)$ is large. In light of such inherent trade-off, an alternative definition is *accuracy parity*, which asks for equalized error rates across different groups:

Definition 7.2.2 (Accuracy Parity)**.** Given a joint distribution $\mathcal{D}$, a classifier $\widehat{Y}$ satisfies *accuracy parity* if $\mathcal{D}_0(\widehat{Y}\neq Y)=\mathcal{D}_1(\widehat{Y}\neq Y)$.

A violation of accuracy parity is also known as disparate mistreatment (Zafar et al., 2017). Different from the definition of demographic parity, the definition of accuracy parity does not eliminate the perfect predictor even when the base rates differ across groups. Of course, other more refined definitions of fairness also exist in the literature, such as *equalized odds* (Hardt et al., 2016).

Definition 7.2.3 (Equalized Odds, a.k.a. Positive Rate Parity)**.** Given a joint distribution $\mathcal{D}$, a classifier $\widehat{Y}$ satisfies *equalized odds* if $\widehat{Y}$ is independent of A conditioned on Y.

The definition of equalized odds essentially requires equal true positive and false positive rates between different groups, hence it is also known as *positive rate parity*. Analogous to accuracy parity, equalized odds does not eliminate the perfect classifier (Hardt et al., 2016), and we will also justify this observation by formal analysis shortly. Last but not least, we have the following definition for predictive rate parity:

Definition 7.2.4 (Predictive Rate Parity)**.** Given a joint distribution $\mathcal{D}$, a classifier $\widehat{Y}$ satisfies *predictive rate parity* if $\mathcal{D}_0(Y = 1 \mid \widehat{Y} = c) = \mathcal{D}_1(Y = 1 \mid \widehat{Y} = c),\ \forall c \in [0, 1]$.

Note that in the above definition we allow the classifier $\widehat{Y}$ to be probabilistic, meaning that the output of $\widehat{Y}$ could be any value between 0 and 1. For the case where $\widehat{Y}$ is deterministic, Chouldechova (2017) showed that no deterministic classifier can simultaneously satisfy equalized odds and predictive rate parity when the base rates differ across subgroups and the classifier is not perfect.

7.3 Algorithm and Analysis

In this section we first give the proposed optimization formulation and then discuss through formal analysis the motivation of our algorithmic design. Specifically, we show in Section 7.3.1 why our formulation helps to escape the utility-fairness trade-off. We then in Section 7.3.2 formally prove that the BERs in the objective function could be used to guarantee a small joint error across different demographic subgroups. In Section 7.3.3 we establish the relationship between equalized odds and accuracy parity by providing an upper bound of the error gap in terms of both BER and the equalized odds gap. We conclude this section with a brief discussion on the practical implementation of the proposed optimization formulation in Section 7.3.4. Due to the space limit, we defer all the proofs to the appendix and focus on explaining the high-level intuition and implications of our results.

As briefly discussed in Section 7.5, a dominant approach in learning fair representations is via adversarial training. Specifically, the following objective function is optimized:

$$\min_{h,g} \max_{h'} \quad \mathrm{CE}_{\mathcal{D}}(h(g(X)) \parallel Y) - \lambda\, \mathrm{CE}_{\mathcal{D}}(h'(g(X)) \parallel A) \tag{7.1}$$

In the above formulation, the first term corresponds to minimization of prediction loss of the target variable and the second term represents the loss incurred by the adversary h'. Overall this minimax optimization problem expresses a trade-off (controlled by the hyperparameter $\lambda > 0$) between utility and fairness through the representation learning function g: on one hand g needs to preserve sufficient information related to Y in order to minimize the first term, but on the other hand g also needs to filter out information related to A in order to maximize the second term.

7.3.1 Conditional Learning of Fair Representations

However, as we introduced in Section 7.2, the above framework is still subjective to the inherent trade-off between utility and fairness. To escape such a trade-off, we advocate for the following optimization formulation instead:

$$\min_{h,g} \max_{h',h''} \quad \mathrm{BER}_{\mathcal{D}}(h(g(X)) \parallel Y) - \lambda \left(\mathrm{BER}_{\mathcal{D}^0}(h'(g(X)) \parallel A) + \mathrm{BER}_{\mathcal{D}^1}(h''(g(X)) \parallel A)\right) \tag{7.2}$$

Note that here we optimize over two distinct adversaries, one for each conditional distribution $\mathcal{D}^y,\ y \in \{0, 1\}$. Intuitively, the main difference between (7.2) and (7.1) is that we use BER as our objective function in both terms. By definition, since BER corresponds to the sum of Type-I and Type-II errors in classification, the proposed objective function essentially minimizes the conditional errors instead of the original marginal error:

$$\begin{aligned} \mathcal{D}(\widehat{Y} \neq Y) &= \mathcal{D}(Y = 0)\mathcal{D}(\widehat{Y} \neq Y \mid Y = 0) + \mathcal{D}(Y = 1)\mathcal{D}(\widehat{Y} \neq Y \mid Y = 1) \\ \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y) &\propto \frac{1}{2}\mathcal{D}(\widehat{Y} \neq Y \mid Y = 0) + \frac{1}{2}\mathcal{D}(\widehat{Y} \neq Y \mid Y = 1), \end{aligned} \tag{7.3}$$

which means that the loss function gives equal importance to the classification error from both groups. Note that the BERs in the second term of (7.2) are over $\mathcal{D}^y,\ y \in \{0, 1\}$. Roughly speaking, the second term encourages alignment of the the conditional distributions $g_\sharp \mathcal{D}_0^y$ and $g_\sharp \mathcal{D}_1^y$ for $y \in \{0,1\}$. The following proposition shows that a perfect conditional alignment of the representations also implies that any classifier based on the representations naturally satisfies the equalized odds criterion:

Proposition 7.3.1. For $g : \mathcal{X} \to \mathcal{Z}$, if $d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0^y, g_\sharp \mathcal{D}_1^y) = 0, \forall y \in \{0,1\}$, then for any classifier $h : \mathcal{Z} \to \{0,1\}$, $h \circ g$ satisfies equalized odds.

To understand why we aim for conditional alignment of distributions instead of aligning the marginal feature distributions, the following proposition characterizes why such alignment will help us to escape the previous trade-off:

Proposition 7.3.2. For $g : \mathcal{X} \to \mathcal{Z}$, if $d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0^y, g_\sharp \mathcal{D}_1^y) = 0, \forall y \in \{0,1\}$, then for any classifier $h : \mathcal{Z} \to \{0,1\}$, $d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0, (h \circ g)_\sharp \mathcal{D}_1) \leq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1)$.

As a corollary, this implies that the lower bound given in Theorem 8.2.1 now vanishes if we instead align the conditional distributions of representations:

$$\varepsilon_{\mathcal{D}_0}(h \circ g) + \varepsilon_{\mathcal{D}_1}(h \circ g) \geq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) - d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0, (h \circ g)_\sharp \mathcal{D}_1) = 0,$$

where the first inequality is due to Lemma 3.1 (Zhao and Gordon, 2019) and the triangle inequality by the $d_{\mathrm{TV}}(\cdot, \cdot)$ distance. Of course, the above lower bound can only serve as a necessary condition but not sufficient to ensure a small joint error across groups. Later (c.f. Theorem 7.3.2) we will show that together with a small BER on the target variable, it becomes a sufficient condition as well.

7.3.2 The Preservation of Demographic Parity Gap and Small Joint Error

In this section we show that learning representations by aligning the conditional distributions across groups cannot increase the DP gap as compared to the DP gap of Y. Before we proceed, we first introduce a metric to measure the deviation of a predictor from satisfying demographic parity:

Definition 7.3.1 (DP Gap)**.** Given a joint distribution $\mathcal{D}$, the *demographic parity gap* of a classifier $\widehat{Y}$ is $\Delta_{\mathrm{DP}}(\widehat{Y}) := |\mathcal{D}_0(\widehat{Y} = 1) - \mathcal{D}_1(\widehat{Y} = 1)|$.

Clearly, if $\Delta_{\mathrm{DP}}(\widehat{Y}) = 0$, then $\widehat{Y}$ satisfies demographic parity. To simplify the exposition, let $\gamma_a := \mathcal{D}_a(Y = 0), \forall a \in \{0,1\}$. We first prove the following lemma:

Lemma 7.3.1. Assume the conditions in Proposition 7.3.1 hold and let $\widehat{Y} = h(g(X))$ be the classifier, then $|\mathcal{D}_0(\widehat{Y} = y) - \mathcal{D}_1(\widehat{Y} = y)| \leq |\gamma_0 - \gamma_1| \cdot \big(\mathcal{D}^0(\widehat{Y} = y) + \mathcal{D}^1(\widehat{Y} = y)\big), \forall y \in \{0,1\}$.

Lemma 7.3.1 gives an upper bound on the difference of the prediction probabilities across different subgroups. Applying Lemma 7.3.1 twice for $y = 0$ and $y = 1$, we can prove the following theorem:

Theorem 7.3.1. Assume the conditions in Proposition 7.3.1 hold and let $\widehat{Y} = h(g(X))$ be the classifier, then $\Delta_{\mathrm{DP}}(\widehat{Y}) \leq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) = \Delta_{\mathrm{DP}}(Y)$.

Remark Theorem 7.3.1 shows that aligning the conditional distributions of representations between groups will not add more bias in terms of the demographic parity gap. In particular, the DP gap of any classifier that satisfies equalized odds will be at most the DP gap of the perfect classifier. This is particularly interesting as it is well-known in the literature (Barocas et al., 2017) that demographic parity is not compatible with equalized odds except in degenerate cases. Despite this result, Theorem 7.3.1 says that we can still achieve equalized odds and simultaneously preserve the DP gap.

In Section 7.3.1 we show that aligning the conditional distributions of representations between groups helps reduce the lower bound of the joint error, but nevertheless that is only a necessary condition. In the next theorem we show that together with a small Type-I and Type-II error in inferring the target variable Y, these two properties are also sufficient to ensure a small joint error across different demographic subgroups.

Theorem 7.3.2. Assume the conditions in Proposition 7.3.1 hold and let $\widehat{Y} = h(g(X))$ be the classifier, then $\varepsilon_{\mathcal{D}_0}(\widehat{Y}) + \varepsilon_{\mathcal{D}_1}(\widehat{Y}) \leq 2\mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)$.

The above bound means that in order to achieve small joint error across groups, it suffices for us to minimize the BER if a classifier satisfies equalized odds. Note that by definition, the BER in the bound equals to the sum of Type-I and Type-II classification errors using $\widehat{Y}$ as a classifier. Theorem 7.3.2 gives an upper bound of the joint error across groups and it also serves as a motivation for us to design the optimization formulation (7.2) that simultaneously minimizes the BER and aligns the conditional distributions.

7.3.3 Conditional Alignment and Balanced Error Rates Lead to Small Error

In this section we will see that a small BER and equalized odds together not only serve as a guarantee of a small joint error, but they also lead to a small error gap between different demographic subgroups. Recall that we define $\gamma_a := \mathcal{D}_a(Y = 0), \forall a \in \{0, 1\}$. Before we proceed, we first formally define the accuracy gap and equalized odds gap of a classifier $\widehat{Y}$:

Definition 7.3.2 (Error Gap)**.** Given a joint distribution $\mathcal{D}$, the *error gap* of a classifier $\widehat{Y}$ is $\Delta_\varepsilon(\widehat{Y}) := |\mathcal{D}_0(\widehat{Y} \neq Y) - \mathcal{D}_1(\widehat{Y} \neq Y)|$.

Definition 7.3.3 (Equalized Odds Gap)**.** Given a joint distribution $\mathcal{D}$, the *equalized odds gap* of a classifier $\widehat{Y}$ is $\Delta_{\mathrm{EO}}(\widehat{Y}) := \max_{y\in\{0,1\}} |\mathcal{D}_0^y(\widehat{Y} = 1) - \mathcal{D}_1^y(\widehat{Y} = 1)|$.

By definition the error gap could also be understood as the accuracy parity gap between different subgroups. The following theorem characterizes the relationship between error gap, equalized odds gap and the difference of base rates across subgroups:

Theorem 7.3.3. For any classifier $\widehat{Y}$, $\Delta_\varepsilon(\widehat{Y}) \leq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) \cdot \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y) + 2\Delta_{\mathrm{EO}}(\widehat{Y})$.

As a direct corollary of Theorem 7.3.3, if the classifier $\widehat{Y}$ satisfies equalized odds, then $\Delta_{\mathrm{EO}}(\widehat{Y}) = 0$. In this case since $\Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1)$ is a constant, minimizing the balanced error rate $\mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)$ also leads to minimizing the error gap. Furthermore, if we combine Theorem 7.3.2 and Theorem 7.3.3 together, we can guarantee that each of the errors cannot be too large:

Corollary 7.3.1. For any joint distribution $\mathcal{D}$ and classifier $\widehat{Y}$, if $\widehat{Y}$ satisfies equalized odds, then

$$\max\{\varepsilon_{\mathcal{D}_0}(\widehat{Y}), \varepsilon_{\mathcal{D}_1}(\widehat{Y})\} \leq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) \cdot \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)/2 + \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y).$$

Remark It is a well-known fact that out of the three fairness criteria, i.e., demographic parity, equalized odds, and predictive rate parity, any two of them cannot hold simultaneously (Barocas et al., 2017) except in degenerate cases. By contrast, Theorem 7.3.3 suggests it *is* possible to achieve both equalized odds and accuracy parity. In particular, among all classifiers that satisfy equalize odds, it suffices to minimize the sum of Type-I and Type-II error $\mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)$ in order to achieve accuracy parity. It is also worth pointing out that Theorem 7.3.3 provides only an upper bound, but not necessarily the tightest one. In particular, the error gap could still be 0 while $\mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)$ is greater than 0. To see this, we have

$$\begin{cases} \varepsilon_{\mathcal{D}_0}(\widehat{Y}) = \mathcal{D}_0(Y = 0) \cdot \mathcal{D}_0(\widehat{Y} = 1 \mid Y = 0) + \mathcal{D}_0(Y = 1) \cdot \mathcal{D}_0(\widehat{Y} = 0 \mid Y = 1) \\ \varepsilon_{\mathcal{D}_1}(\widehat{Y}) = \mathcal{D}_1(Y = 0) \cdot \mathcal{D}_1(\widehat{Y} = 1 \mid Y = 0) + \mathcal{D}_1(Y = 1) \cdot \mathcal{D}_1(\widehat{Y} = 0 \mid Y = 1). \end{cases}$$

Now if the predictor $\widehat{Y}$ satisfies equalized odds, then

$$\mathcal{D}_0(\widehat{Y} = 1 \mid Y = 0) = \mathcal{D}_1(\widehat{Y} = 1 \mid Y = 0) = \mathcal{D}(\widehat{Y} = 1 \mid Y = 0),$$
$$\mathcal{D}_0(\widehat{Y} = 0 \mid Y = 1) = \mathcal{D}_1(\widehat{Y} = 0 \mid Y = 1) = \mathcal{D}(\widehat{Y} = 0 \mid Y = 1).$$

Hence the error gap $\Delta_\varepsilon(\widehat{Y})$ admits the following identity:

$$\begin{aligned}\Delta_\varepsilon(\widehat{Y}) &= \left|\mathcal{D}(\widehat{Y}=1 \mid Y=0)\big(\mathcal{D}_0(Y=0)-\mathcal{D}_1(Y=0)\big)+\mathcal{D}(\widehat{Y}=0 \mid Y=1)\big(\mathcal{D}_0(Y=1)-\mathcal{D}_1(Y=1)\big)\right| \\ &= \Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)\cdot\left|\mathcal{D}(\widehat{Y}=1 \mid Y=0)-\mathcal{D}(\widehat{Y}=0 \mid Y=1)\right| \\ &= \Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)\cdot\left|\mathrm{FPR}(\widehat{Y})-\mathrm{FNR}(\widehat{Y})\right|.\end{aligned}$$

In other words, if the predictor $\widehat{Y}$ satisfies equalized odds, then in order to have equalized accuracy, $\widehat{Y}$ only needs to have equalized FPR and FNR *globally* when the base rates differ across groups. This is a much weaker condition to ask for than the one asking $\mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)=0$.

7.3.4 Practical Implementation

We cannot directly optimize the proposed optimization formulation (7.2) since the binary $0/1$ loss is NP-hard to optimize, or even approximately optimize over a wide range of hypothesis classes (Ben-David et al., 2003). However, observe that for any classifier $\widehat{Y}$ and $y \in \{0,1\}$, the log-loss (cross-entropy loss) $\mathrm{CE}_{\mathcal{D}^y}(\widehat{Y} \parallel Y)$ is a convex relaxation of the binary loss:

$$\mathcal{D}(\widehat{Y}\neq y \mid Y=y)=\frac{\mathcal{D}(\widehat{Y}\neq y, Y=y)}{\mathcal{D}(Y=y)}\leq\frac{\mathrm{CE}_{\mathcal{D}^y}(\widehat{Y} \parallel Y)}{\mathcal{D}(Y=y)}. \tag{7.4}$$

Hence in practice we can relax the optimization problem (7.2) to a cost-sensitive cross-entropy loss minimization problem, where the weight for each class is given by the inverse marginal probability of the corresponding class. This allows us to equivalently optimize the objective function without explicitly computing the conditional distributions.

7.4 Empirical Studies

In light of our theoretic findings, in this section we verify the effectiveness of the proposed algorithm in simultaneously ensuring equalized odds and accuracy parity using real-world datasets. We also analyze the impact of imposing such parity constraints on the utility of the target classifier, as well as its relationship to the intrinsic structure of the binary classification problem, e.g., the difference of base rates across groups, the global imbalance of the target variable, etc. We analyze how this imbalance affects the utility-fairness trade-off. As we shall see shortly, we will empirically demonstrate that, in many cases, especially the ones where the dataset is imbalanced in terms of the target variable, this will inevitably compromise the target utility. While for balanced datasets, this trend is less obvious: the proposed algorithm achieves a better fairness-utility trade-off when compared with existing fair representation learning methods and we can hope to achieve fairness without sacrificing too much on utility.

Table 7.4.1: Statistics about the Adult and COMPAS datasets.

	Train / Test	$\mathcal{D}_0(Y=1)$	$\mathcal{D}_1(Y=1)$	$\Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)$	$\mathcal{D}(Y=1)$	$\mathcal{D}(A=1)$
Adult	$30,162/15,060$	0.310	0.113	0.196	0.246	0.673
COMPAS	$4,320/1,852$	0.400	0.529	0.129	0.467	0.514

7.4.1 Experimental Setup

To this end, we perform experiments on two popular real-world datasets in the literature of algorithmic fairness, including an income-prediction dataset, known as the *Adult* dataset, from the UCI Machine Learning Repository (Dua and Graff, 2017), and the Propublica *COMPAS* dataset (Dieterich et al., 2016). The basic statistics of these two datasets are listed in Table 7.4.1.

Adult Each instance in the Adult dataset describes an adult, e.g., gender, education level, age, etc, from the 1994 US Census. In this dataset we use gender as the sensitive attribute, and the processed data contains 114 attributes. The target variable (income) is also binary: 1 if $\geq$ 50K/year otherwise 0. For the sensitive attribute A, $A = 0$ means male otherwise female. From Table 7.4.1 we can see that the base rates are quite different (0.310 vs. 0.113) across groups in the Adult dataset. The dataset is also imbalanced in the sense that only around 24.6% of the instances have target label 1. Furthermore, the group ratio is also imbalanced: roughly 67.3% of the data are male.

COMPAS The goal of the COMPAS dataset is binary classification on whether a criminal defendant will recidivate within two years or not. Each instance contains 12 attributes, including age, race, gender, number of prior crimes, etc. For this dataset, we use the race (white $A = 0$ vs. black $A = 1$) as our sensitive attribute and target variable is 1 iff recidivism. The base rates are different across groups, but the COMPAS dataset is balanced in both the target variable and the sensitive attribute.

To validate the effect of ensuring equalized odds and accuracy parity, for each dataset, we perform controlled experiments by fixing the baseline network architecture so that it is shared among all the fair representation learning methods. We term the proposed method CFAIR (for conditional fair representations) that minimizes conditional errors both the target variable loss function and adversary loss function. To demonstrate the importance of using BER in the loss function of target variable, we compare with a variant of CFAIR that only uses BER in the loss of adversaries, denoted as CFAIR-EO. To see the relative effect of using cross-entropy loss vs L_1 loss, we also show the results of LAFTR (Madras et al., 2018), a state-of-the-art method for learning fair representations. Note that LAFTR is closely related to CFAIR-EO but slightly different: LAFTR uses global cross-entropy loss for target variable, but conditional L_1 loss for the adversary. Also, there is only one adversary in LAFTR, while there are two adversaries, one for $\mathcal{D}^0$ and one for $\mathcal{D}^1$, in both CFAIR and CFAIR-EO. Lastly, we also present baseline results of FAIR (Edwards and Storkey, 2015), which aims for demographic parity representations, and NODEBIAS, the baseline network without any fairness constraint. For all the fair representation learning methods, we use the gradient reversal layer (Ganin et al., 2016) to implement the gradient descent ascent (GDA) algorithm to optimize the minimax problem. All the experimental details, including network architectures, learning rates, batch sizes, etc. are provided in the appendix.

7.4.2 Results and Analysis

In Figure 7.4.1 and Figure 7.4.2 we show the error gap Δ_ε, equalized odds gap Δ_{EO}, demographic parity gap Δ_{DP} and the joint error across groups $\varepsilon_0 + \varepsilon_1$ of the aforementioned fair representation learning algorithms on both the Adult and the COMPAS datasets. For each algorithm and dataset, we also gradually increase the value of the trade-off parameter λ and compute the corresponding metrics.

Adult Due to the imbalance of A in the Adult dataset, in the first plot of Figure 7.4.1 we can see that all the algorithms except CFAIR have a large error gap of around 0.12. As a comparison, observe that the error gap of CFAIR when $\lambda = 1e3$ almost reduces to 0, confirming the effectiveness of our algorithm

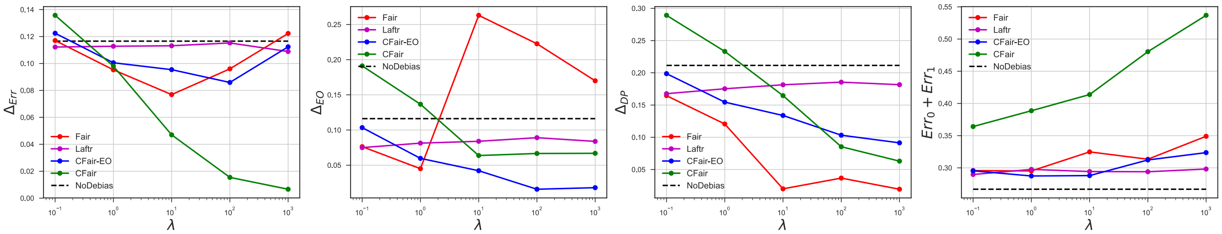


Figure 7.4.1: The error gap Δ_ε, equalized odds gap Δ_{EO}, demographic parity gap Δ_{DP} and joint error $\varepsilon_0 + \varepsilon_1$ on the Adult dataset with $\lambda \in \{0.1, 1.0, 10.0, 100.0, 1000.0\}$.

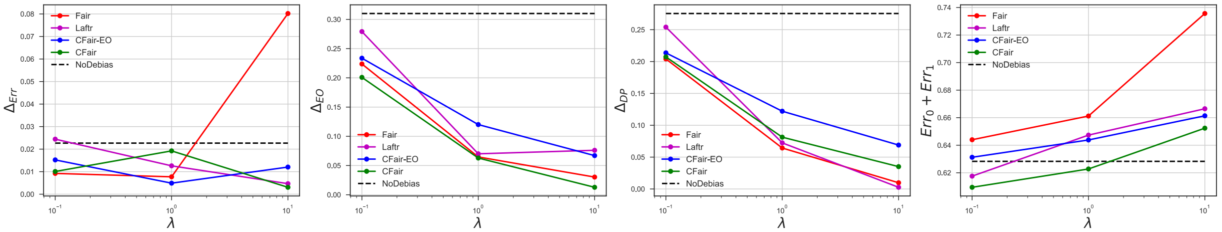


Figure 7.4.2: The error gap Δ_ε, equalized odds gap Δ_{EO}, demographic parity gap Δ_{DP} and joint error $\varepsilon_0 + \varepsilon_1$ on the COMPAS dataset with $\lambda \in \{0.1, 1.0, 10.0\}$.

in ensuring accuracy parity. From the second plot, we can verify that all the three methods, including CFAIR, CFAIR-EO and LAFTR successfully ensure a small equalized odds gap, and they also decrease demographic parity gaps (the third plot). FAIR is the most effective one in mitigating Δ_{DP} since its objective function directly optimizes for that goal. Note that from the second plot we can also confirm that CFAIR-EO is more effective than LAFTR in reducing Δ_{EO}. The reason is two-fold. First, CFAIR-EO uses two distinct adversaries and hence it effectively competes with stronger adversaries than LAFTR. Second, CFAIR-EO uses the cross-entropy loss instead of the L_1 loss for the adversary, and it is well-known that the maximum-likelihood estimator (equivalent to using the cross-entropy loss) is asymptotically efficient and optimal. On the other hand, since the Adult dataset is imbalanced (in terms of Y), using BER in the loss function of the target variable can thus to a large extent hurt the utility, and this is also confirmed from the last plot, where we show the joint error.

COMPAS The first three plots of Figure 7.4.2 once again verify that CFAIR successfully leads to reduced error gap, equalized odds gap and also demographic parity gap. These experimental results are consistent with our theoretical findings where we show that if the representations satisfy equalized odds, then its Δ_{DP} cannot exceed that of the optimal classifier, as shown by the horizontal dashed line in the third plot. In the fourth plot of Figure 7.4.2, we can see that as we increase λ, all the fair representation learning algorithms sacrifice utility. However, in contrast to Figure 7.4.1, here the proposed algorithm CFAIR has the smallest trade-off: this shows that CFAIR is particularly suited in the cases when the dataset is balanced and we would like to simultaneously ensure accuracy parity and equalized odds. As a comparison, while CFAIR-EO is still effective, it is slightly worse than CFAIR in terms of both ensuring parity and achieving small joint error.

7.5 Related Work

Algorithmic Fairness In the literature of algorithmic fairness, two key notions of fairness have been extensively proposed and explored, i.e., *group fairness*, including various variants defined in Section 7.2, and *individual fairness*, which means that similar individuals should be treated similarly. Due to the complexity in defining a distance metric to measure the similarity between individuals (Dwork et al., 2012), most recent research focuses on designing efficient algorithms to achieve group fairness (Creager et al., 2019; Hardt et al., 2016; Madras et al., 2018, 2019; Zafar et al., 2015, 2017; Zemel et al., 2013). In particular, Hardt et al. (2016) proposed a post-processing technique to achieve equalized odds by taking as input the model's prediction and the sensitive attribute. However, the post-processing technique requires access to the sensitive attribute during the inference phase, which is often not available in many real-world scenarios. Another line of work uses causal inference to define notions of causal fairness and to formulate procedures for achieving these notions (Kilbertus et al., 2017; Kusner et al., 2017; Madras et al., 2019; Nabi and Shpitser, 2018; Wang et al., 2019; Zhang et al., 2018). These approaches require making untestable assumptions. Of particular note is the observation in Coston et al. (2019) that fairness-adjustment procedures based on Y in settings with treatment effects may lead to adverse outcomes. To apply our method in such settings, we would need to match conditional counterfactual distributions, which could be a direction of future research.

Theoretical Results on Fairness Theoretical work studying the relationship between different kinds of fairness notions are abundant. Motivated by the controversy of the potential discriminatory bias in recidivism prediction instruments, Chouldechova (2017) showed an intrinsic incompatibility between equalized odds and predictive rate parity. In the seminal work of Kleinberg et al. (2016), the authors demonstrated that when the base rates differ between different groups, then a non-perfect classifier cannot simultaneously be statistically calibrated and satisfy equalized odds. In the context of cost-sensitive learning, Menon and Williamson (2018) show that if the optimal decision function is dissimilar to a fair decision, then the fairness constraint will not significantly harm the target utility. The idea of reducing fair classification to cost-sensitive learning is not new. Agarwal et al. (2018) explored the connection between fair classification and a sequence of cost-sensitive learning problems where each stage corresponds to solving a linear minimax saddle point problem. In a recent work (Zhao and Gordon, 2019), the authors proved a lower bound on the joint error across different groups when a classifier satisfies demographic parity. They also showed that when the decision functions are close between groups, demographic parity also implies accuracy parity. The theoretical results in this work establish a relationship between accuracy parity and equalized odds: these two fairness notions are fundamentally related by the base rate gap and the balanced error rate. Furthermore, we also show that for any predictor that satisfies equalized odds, the balanced error rate also serves as an upper bound on the joint error across demographic subgroups.

Fair Representations Through the lens of representation learning, recent advances in building fair algorithmic decision making systems focus on using adversarial methods to learn fair representations that also preserve sufficient information for the prediction task (Adel et al., 2019a; Beutel et al., 2017; Edwards and Storkey, 2015; Madras et al., 2018; Zhang et al., 2018). In a nutshell, the key idea is to frame the problem of learning fair representations as a two-player game, where the data owner is competing against an adversary. The goal of the adversary is to infer the group attribute as much as possible while the goal of the data owner is to remove information related to the group attribute and simultaneously to preserve utility-related information for accurate prediction. Apart from using adversarial classifiers to enforce group fairness, other distance metrics have also been used to learn fair representations, e.g., the maximum mean discrepancy (Louizos et al., 2015), and the Wasserstein-1 distance (Jiang et al., 2019). In contrast to these methods, in this chapter we advocate for optimizing BER on both the target loss and adversary loss in order to simultaneously achieve accuracy parity and equalized odds. We also show that this leads to better utility-fairness trade-off for balanced datasets.

7.6 Proofs

We provide all the missing proofs in this section.

7.6.1 Proof of Proposition 7.3.1

Proposition 7.3.1. For $g : \mathcal{X} \to \mathcal{Z}$, if $d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0^y, g_\sharp \mathcal{D}_1^y) = 0, \forall y \in \{0,1\}$, then for any classifier $h : \mathcal{Z} \to \{0,1\}$, $h \circ g$ satisfies equalized odds.

Proof. To prove this proposition, we first show that for any pair of distributions $\mathcal{D}$, $\mathcal{D}'$ over $\mathcal{Z}$ and any hypothesis $h : \mathcal{Z} \to \{0,1\}$, $d_{\mathrm{TV}}(h_\sharp \mathcal{D}, h_\sharp \mathcal{D}') \le d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}')$. Note that since h is a hypothesis, there are only two events in the induced probability space, i.e., $h(\cdot) = 0$ or $h(\cdot) = 1$. Hence by definition of the induced (pushforward) distribution, we have:

$$\begin{aligned} d_{\mathrm{TV}}(h_\sharp \mathcal{D}, h_\sharp \mathcal{D}') &= \max_{E = h^{-1}(0), \text{ or } E = h^{-1}(1)} |\mathcal{D}(E) - \mathcal{D}'(E)| \\ &\le \sup_{E \text{ is measurable subset of } \mathcal{Z}} |\mathcal{D}(E) - \mathcal{D}'(E)| \\ &= d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}'). \end{aligned}$$

Apply the above inequality twice for $y \in \{0,1\}$:

$$0 \le d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0^y, (h \circ g)_\sharp \mathcal{D}_1^y) \le d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0^y, g_\sharp \mathcal{D}_1^y) = 0,$$

meaning

$$d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0^y, (h \circ g)_\sharp \mathcal{D}_1^y) = 0,$$

which further implies that $h(g(X))$ is independent of A given $Y = y$ since $h(g(X))$ is binary. ■

7.6.2 Proof of Proposition 7.3.2

Proposition 7.3.2. For $g : \mathcal{X} \to \mathcal{Z}$, if $d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0^y, g_\sharp \mathcal{D}_1^y) = 0, \forall y \in \{0,1\}$, then for any classifier $h : \mathcal{Z} \to \{0,1\}$, $d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0, (h \circ g)_\sharp \mathcal{D}_1) \le \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1)$.

Proof. Let $\widehat{Y} = (h \circ g)(X)$ and note that $\widehat{Y}$ is binary, we have

$$d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0, (h \circ g)_\sharp \mathcal{D}_1) = \frac{1}{2}\left(|\mathcal{D}_0(\widehat{Y} = 0) - \mathcal{D}_1(\widehat{Y} = 0)| + |\mathcal{D}_0(\widehat{Y} = 1) - \mathcal{D}_1(\widehat{Y} = 1)|\right).$$

Now, by Proposition 7.3.1, if $d_{\mathrm{TV}}(g_\sharp \mathcal{D}_0^y, g_\sharp \mathcal{D}_1^y) = 0$, $\forall y \in \{0,1\}$, it follows that $d_{\mathrm{TV}}((h \circ g)_\sharp \mathcal{D}_0^y, (h \circ g)_\sharp \mathcal{D}_1^y) = 0$, $\forall y \in \{0,1\}$ as well. Applying Lemma 7.3.1, we know that $\forall y \in \{0,1\}$,

$$|\mathcal{D}_0(\widehat{Y} = y) - \mathcal{D}_1(\widehat{Y} = y)| \le |\mathcal{D}_0(Y = 0) - \mathcal{D}_1(Y = 0)| \cdot \left(\mathcal{D}^0(\widehat{Y} = y) + \mathcal{D}^1(\widehat{Y} = y)\right).$$

Hence,

$$
\begin{aligned}
d_{\mathrm{TV}}((h\circ g)_\sharp\mathcal{D}_0,(h\circ g)_\sharp\mathcal{D}_1) &= \frac{1}{2}\left(|\mathcal{D}_0(\widehat{Y}=0)-\mathcal{D}_1(\widehat{Y}=0)| + |\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)|\right)\\
&\leq \frac{|\mathcal{D}_0(Y=0)-\mathcal{D}_1(Y=0)|}{2}\left(\left(\mathcal{D}^0(\widehat{Y}=0)+\mathcal{D}^1(\widehat{Y}=0)\right)+\left(\mathcal{D}^0(\widehat{Y}=1)+\mathcal{D}^1(\widehat{Y}=1)\right)\right)\\
&= \frac{|\mathcal{D}_0(Y=0)-\mathcal{D}_1(Y=0)|}{2}\left(\left(\mathcal{D}^0(\widehat{Y}=0)+\mathcal{D}^0(\widehat{Y}=1)\right)+\left(\mathcal{D}^1(\widehat{Y}=0)+\mathcal{D}^1(\widehat{Y}=1)\right)\right)\\
&= \frac{|\mathcal{D}_0(Y=0)-\mathcal{D}_1(Y=0)|}{2}\cdot 2\\
&= |\mathcal{D}_0(Y=0)-\mathcal{D}_1(Y=0)|\\
&= \Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1).
\end{aligned}
$$

■

7.6.3 Proof of Lemma 7.3.1

Recall that we define $\gamma_a := \mathcal{D}_a(Y=0), \forall a\in\{0,1\}$.

Lemma 7.3.1. Assume the conditions in Proposition 7.3.1 hold and let $\widehat{Y}=h(g(X))$ be the classifier, then $|\mathcal{D}_0(\widehat{Y}=y)-\mathcal{D}_1(\widehat{Y}=y)| \leq |\gamma_0-\gamma_1|\cdot\left(\mathcal{D}^0(\widehat{Y}=y)+\mathcal{D}^1(\widehat{Y}=y)\right), \forall y\in\{0,1\}$.

Proof. To bound $|\mathcal{D}_0(\widehat{Y}=y)-\mathcal{D}_1(\widehat{Y}=y)|$, for $y\in\{0,1\}$, by the law of total probability, we have:

$$
\begin{aligned}
&|\mathcal{D}_0(\widehat{Y}=y)-\mathcal{D}_1(\widehat{Y}=y)| =\\
&\quad= \left|\left(\mathcal{D}_0^0(\widehat{Y}=y)\mathcal{D}_0(Y=0)+\mathcal{D}_0^1(\widehat{Y}=y)\mathcal{D}_0(Y=1)\right)-\left(\mathcal{D}_1^0(\widehat{Y}=y)\mathcal{D}_1(Y=0)+\mathcal{D}_1^1(\widehat{Y}=y)\mathcal{D}_1(Y=1)\right)\right|\\
&\quad\leq \left|\gamma_0\mathcal{D}_0^0(\widehat{Y}=y)-\gamma_1\mathcal{D}_1^0(\widehat{Y}=y)\right| + \left|(1-\gamma_0)\mathcal{D}_0^1(\widehat{Y}=y)-(1-\gamma_1)\mathcal{D}_1^1(\widehat{Y}=y)\right|,
\end{aligned}
$$

where the above inequality is due to the triangular inequality. Now apply Proposition 7.3.1, we know that $\widehat{Y}$ satisfies equalized odds, so we have $\mathcal{D}_0^0(\widehat{Y}=y)=\mathcal{D}_1^0(\widehat{Y}=y)=\mathcal{D}^0(\widehat{Y}=y)$ and $\mathcal{D}_0^1(\widehat{Y}=y)=\mathcal{D}_1^1(\widehat{Y}=y)=\mathcal{D}^1(\widehat{Y}=y)$, leading to:

$$
\begin{aligned}
&= \left|\gamma_0-\gamma_1\right|\cdot\mathcal{D}^0(\widehat{Y}=y)+\left|(1-\gamma_0)-(1-\gamma_1)\right|\cdot\mathcal{D}^1(\widehat{Y}=y)\\
&= |\gamma_0-\gamma_1|\cdot\left(\mathcal{D}^0(\widehat{Y}=y)+\mathcal{D}^1(\widehat{Y}=y)\right),
\end{aligned}
$$

which completes the proof. ■

7.6.4 Proof of Theorem 7.3.1

Theorem 7.3.1. Assume the conditions in Proposition 7.3.1 hold and let $\widehat{Y}=h(g(X))$ be the classifier, then $\Delta_{\mathrm{DP}}(\widehat{Y})\leq\Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)=\Delta_{\mathrm{DP}}(Y)$.

Proof. To bound $\Delta_{\mathrm{DP}}(\widehat{Y})$, realize that $|\mathcal{D}_0(\widehat{Y}=0)-\mathcal{D}_1(\widehat{Y}=0)|=|\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)|$, so we can rewrite the DP gap as follows:

$$\Delta_{\mathrm{DP}}(\widehat{Y})=\frac{1}{2}\left(|\mathcal{D}_0(\widehat{Y}=0)-\mathcal{D}_1(\widehat{Y}=0)|+|\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)|\right).$$

Now apply Lemma 7.3.1 twice for $y=0$ and $y=1$, we have:

$$
\begin{aligned}
|\mathcal{D}_0(\widehat{Y}=0)-\mathcal{D}_1(\widehat{Y}=0)| &\leq |\gamma_0-\gamma_1|\cdot\left(\mathcal{D}^0(\widehat{Y}=0)+\mathcal{D}^1(\widehat{Y}=0)\right)\\
|\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)| &\leq |\gamma_0-\gamma_1|\cdot\left(\mathcal{D}^0(\widehat{Y}=1)+\mathcal{D}^1(\widehat{Y}=1)\right).
\end{aligned}
$$

Taking sum of the above two inequalities yields

$$
\begin{aligned}
&|\mathcal{D}_0(\widehat{Y}=0)-\mathcal{D}_1(\widehat{Y}=0)|+|\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)| \\
&\qquad \le |\gamma_0-\gamma_1|\left(\left(\mathcal{D}^0(\widehat{Y}=0)+\mathcal{D}^1(\widehat{Y}=0)\right)+\left(\mathcal{D}^0(\widehat{Y}=1)+\mathcal{D}^1(\widehat{Y}=1)\right)\right) \\
&\qquad = |\gamma_0-\gamma_1|\left(\left(\mathcal{D}^0(\widehat{Y}=0)+\mathcal{D}^0(\widehat{Y}=1)\right)+\left(\mathcal{D}^1(\widehat{Y}=0)+\mathcal{D}^1(\widehat{Y}=1)\right)\right) \\
&\qquad = 2\big|\gamma_0-\gamma_1\big|.
\end{aligned}
$$

Combining all the inequalities above, we know that

$$
\begin{aligned}
\Delta_{\mathrm{DP}}(\widehat{Y}) &= \frac{1}{2}\left(|\mathcal{D}_0(\widehat{Y}=0)-\mathcal{D}_1(\widehat{Y}=0)|+|\mathcal{D}_0(\widehat{Y}=1)-\mathcal{D}_1(\widehat{Y}=1)|\right) \\
&\le \big|\gamma_0-\gamma_1\big| \\
&= |\mathcal{D}_0(Y=0)-\mathcal{D}_1(Y=0)| \\
&= |\mathcal{D}_0(Y=1)-\mathcal{D}_1(Y=1)| \\
&= \Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)=\Delta_{\mathrm{DP}}(Y),
\end{aligned}
$$

completing the proof. ■

7.6.5 Proof of Theorem 7.3.2

Theorem 7.3.2. Assume the conditions in Proposition 7.3.1 hold and let $\widehat{Y}=h(g(X))$ be the classifier, then $\varepsilon_{\mathcal{D}_0}(\widehat{Y})+\varepsilon_{\mathcal{D}_1}(\widehat{Y})\le 2\mathrm{BER}_{\mathcal{D}}(\widehat{Y}\parallel Y)$.

Proof. First, by the law of total probability, we have:

$$
\begin{aligned}
\varepsilon_{\mathcal{D}_0}(\widehat{Y})+\varepsilon_{\mathcal{D}_1}(\widehat{Y}) &= \mathcal{D}_0(Y\neq\widehat{Y})+\mathcal{D}_1(Y\neq\widehat{Y}) \\
&= \mathcal{D}_0^1(\widehat{Y}=0)\mathcal{D}_0(Y=1)+\mathcal{D}_0^0(\widehat{Y}=1)\mathcal{D}_0(Y=0)+\mathcal{D}_1^1(\widehat{Y}=0)\mathcal{D}_1(Y=1)+\mathcal{D}_1^0(\widehat{Y}=1)\mathcal{D}_1(Y=0)
\end{aligned}
$$

Again, by Proposition 7.3.1, the classifier $\widehat{Y}=(h\circ g)(X)$ satisfies equalized odds, so we have $\mathcal{D}_0^1(\widehat{Y}=0)=\mathcal{D}^1(\widehat{Y}=0)$, $\mathcal{D}_0^0(\widehat{Y}=1)=\mathcal{D}^0(\widehat{Y}=1)$, $\mathcal{D}_1^1(\widehat{Y}=0)=\mathcal{D}^1(\widehat{Y}=0)$ and $\mathcal{D}_1^0(\widehat{Y}=1)=\mathcal{D}^0(\widehat{Y}=1)$:

$$
\begin{aligned}
&= \mathcal{D}^1(\widehat{Y}=0)\mathcal{D}_0(Y=1)+\mathcal{D}^0(\widehat{Y}=1)\mathcal{D}_0(Y=0)+\mathcal{D}^1(\widehat{Y}=0)\mathcal{D}_1(Y=1)+\mathcal{D}^0(\widehat{Y}=1)\mathcal{D}_1(Y=0) \\
&= \mathcal{D}^1(\widehat{Y}=0)\cdot\left(\mathcal{D}_0(Y=1)+\mathcal{D}_1(Y=1)\right)+\mathcal{D}^0(\widehat{Y}=1)\cdot\left(\mathcal{D}_0(Y=0)+\mathcal{D}_1(Y=0)\right) \\
&\le 2\mathcal{D}^1(\widehat{Y}=0)+2\mathcal{D}^0(\widehat{Y}=1) \\
&= 2\mathrm{BER}_{\mathcal{D}}(\widehat{Y}\parallel Y),
\end{aligned}
$$

which completes the proof. ■

7.6.6 Proof of Theorem 7.3.3

Theorem 7.3.3. For any classifier $\widehat{Y}$, $\Delta_\varepsilon(\widehat{Y})\le\Delta_{\mathrm{BR}}(\mathcal{D}_0,\mathcal{D}_1)\cdot\mathrm{BER}_{\mathcal{D}}(\widehat{Y}\parallel Y)+2\Delta_{\mathrm{EO}}(\widehat{Y})$.

Before we give the proof of Theorem 7.3.3, we first prove the following two lemmas that will be used in the following proof.

Lemma 7.6.1. Define $\gamma_a:=\mathcal{D}_a(Y=0),\forall a\in\{0,1\}$, then $|\gamma_0\mathcal{D}_0^0(\widehat{Y}=1)-\gamma_1\mathcal{D}_1^0(\widehat{Y}=1)|\le|\gamma_0-\gamma_1|\cdot\mathcal{D}^0(\widehat{Y}=1)+\gamma_0\mathcal{D}^0(A=1)\Delta_{\mathrm{EO}}(\widehat{Y})+\gamma_1\mathcal{D}^0(A=0)\Delta_{\mathrm{EO}}(\widehat{Y})$.

Proof. In order to prove the upper bound in the lemma, it suffices if we could give the desired upper bound for the following term

$$\begin{aligned}
&\Big||\gamma_0\mathcal{D}_0^0(\widehat{Y}=1)-\gamma_1\mathcal{D}_1^0(\widehat{Y}=1)|-|(\gamma_0-\gamma_1)\mathcal{D}^0(\widehat{Y}=1)|\Big| \\
&\leq \Big|\Big(\gamma_0\mathcal{D}_0^0(\widehat{Y}=1)-\gamma_1\mathcal{D}_1^0(\widehat{Y}=1)\Big)-(\gamma_0-\gamma_1)\mathcal{D}^0(\widehat{Y}=1)\Big| \\
&= \Big|\gamma_0(\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1))-\gamma_1(\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1))\Big|,
\end{aligned}$$

following which we will have:

$$\begin{aligned}
|\gamma_0\mathcal{D}_0^0(\widehat{Y}=1)-\gamma_1\mathcal{D}_1^0(\widehat{Y}=1)| \leq{} & |(\gamma_0-\gamma_1)\mathcal{D}^0(\widehat{Y}=1)| \\
&+\Big|\gamma_0(\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1))-\gamma_1(\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1))\Big|,
\end{aligned}$$

and an application of the Bayes formula could finish the proof. To do so, let us first simplify $\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1)$. Applying the Bayes's formula, we know that:

$$\begin{aligned}
\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1) &= \mathcal{D}_0^0(\widehat{Y}=1)-\big(\mathcal{D}_0^0(\widehat{Y}=1)\mathcal{D}^0(A=0)+\mathcal{D}_1^0(\widehat{Y}=1)\mathcal{D}^0(A=1)\big) \\
&= \big(\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}_0^0(\widehat{Y}=1)\mathcal{D}^0(A=0)\big)-\mathcal{D}_1^0(\widehat{Y}=1)\mathcal{D}^0(A=1) \\
&= \mathcal{D}^0(A=1)\big(\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}_1^0(\widehat{Y}=1)\big).
\end{aligned}$$

Similarly, for the second term $\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1)$, we can show that:

$$\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1)=\mathcal{D}^0(A=0)\big(\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}_0^0(\widehat{Y}=1)\big).$$

Plug these two identities into above, we can continue the analysis with

$$\begin{aligned}
&\Big|\gamma_0(\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1))-\gamma_1(\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}^0(\widehat{Y}=1))\Big| \\
&= \Big|\gamma_0\mathcal{D}^0(A=1)(\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}_1^0(\widehat{Y}=1))-\gamma_1\mathcal{D}^0(A=0)(\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}_0^0(\widehat{Y}=1))\Big| \\
&\leq \Big|\gamma_0\mathcal{D}^0(A=1)(\mathcal{D}_0^0(\widehat{Y}=1)-\mathcal{D}_1^0(\widehat{Y}=1))\Big|+\Big|\gamma_1\mathcal{D}^0(A=0)(\mathcal{D}_1^0(\widehat{Y}=1)-\mathcal{D}_0^0(\widehat{Y}=1))\Big| \\
&\leq \gamma_0\mathcal{D}^0(A=1)\Delta_{\mathrm{EO}}(\widehat{Y})+\gamma_1\mathcal{D}^0(A=0)\Delta_{\mathrm{EO}}(\widehat{Y}).
\end{aligned}$$

The first inequality holds by triangular inequality and the second one holds by the definition of equalized odds gap. ■

Lemma 7.6.2. Define $\gamma_a := \mathcal{D}_a(Y=0), \forall a \in \{0,1\}$, then $|(1-\gamma_0)\mathcal{D}_0^1(\widehat{Y}=0)-(1-\gamma_1)\mathcal{D}_1^1(\widehat{Y}=0)| \leq |\gamma_0-\gamma_1|\cdot\mathcal{D}^1(\widehat{Y}=0)+(1-\gamma_0)\mathcal{D}^1(A=1)\Delta_{\mathrm{EO}}(\widehat{Y})+(1-\gamma_1)\mathcal{D}^1(A=0)\Delta_{\mathrm{EO}}(\widehat{Y})$.

Proof. The proof of this lemma is symmetric to the previous one, so we omit it here. ■

Now we are ready to prove Theorem 7.3.3:

Proof of Theorem 7.3.3. First, by the law of total probability, it is easy to verify that following identity holds for $a \in \{0,1\}$:

$$\begin{aligned}
\mathcal{D}_a(\widehat{Y}\neq Y) &= \mathcal{D}_a(Y=1,\widehat{Y}=0)+\mathcal{D}_a(Y=0,\widehat{Y}=1) \\
&= (1-\gamma_a)\mathcal{D}_a^1(\widehat{Y}=0)+\gamma_a\mathcal{D}_a^0(\widehat{Y}=1).
\end{aligned}$$

Using this identity, to bound the error gap, we have:

$$\begin{aligned}|\mathcal{D}_0(Y \neq \widehat{Y}) - \mathcal{D}_1(Y \neq \widehat{Y})| &= \left|((1-\gamma_0)\mathcal{D}_0^1(\widehat{Y}=0) + \gamma_0\mathcal{D}_0^0(\widehat{Y}=1)) - ((1-\gamma_1)\mathcal{D}_1^1(\widehat{Y}=0) + \gamma_1\mathcal{D}_1^0(\widehat{Y}=1))\right| \\ &\leq \left|\gamma_0\mathcal{D}_0^0(\widehat{Y}=1) - \gamma_1\mathcal{D}_1^0(\widehat{Y}=1)\right| + \left|(1-\gamma_0)\mathcal{D}_0^1(\widehat{Y}=0) - (1-\gamma_1)\mathcal{D}_1^1(\widehat{Y}=0)\right|.\end{aligned}$$

Invoke Lemma 7.6.1 and Lemma 7.6.2 to bound the above two terms:

$$\begin{aligned}&|\mathcal{D}_0(Y \neq \widehat{Y}) - \mathcal{D}_1(Y \neq \widehat{Y})| \\ &\qquad\leq \left|\gamma_0\mathcal{D}_0^0(\widehat{Y}=1) - \gamma_1\mathcal{D}_1^0(\widehat{Y}=1)\right| + \left|(1-\gamma_0)\mathcal{D}_0^1(\widehat{Y}=0) - (1-\gamma_1)\mathcal{D}_1^1(\widehat{Y}=0)\right| \\ &\qquad\leq \gamma_0\mathcal{D}^0(A=1)\Delta_{\mathrm{EO}}(\widehat{Y}) + \gamma_1\mathcal{D}^0(A=0)\Delta_{\mathrm{EO}}(\widehat{Y}) \\ &\qquad+ (1-\gamma_0)\mathcal{D}^1(A=1)\Delta_{\mathrm{EO}}(\widehat{Y}) + (1-\gamma_1)\mathcal{D}^1(A=0)\Delta_{\mathrm{EO}}(\widehat{Y}) \\ &\qquad+ \left|\gamma_0 - \gamma_1\right|\mathcal{D}^0(\widehat{Y}=1) + \left|\gamma_0 - \gamma_1\right|\mathcal{D}^1(\widehat{Y}=0),\end{aligned}$$

Realize that both $\gamma_0, \gamma_1 \in [0,1]$, we have:

$$\begin{aligned}&\leq \mathcal{D}^0(A=1)\Delta_{\mathrm{EO}}(\widehat{Y}) + \mathcal{D}^0(A=0)\Delta_{\mathrm{EO}}(\widehat{Y}) + \mathcal{D}^1(A=1)\Delta_{\mathrm{EO}}(\widehat{Y}) + \mathcal{D}^1(A=0)\Delta_{\mathrm{EO}}(\widehat{Y}) \\ &+ \left|\gamma_0 - \gamma_1\right|\mathcal{D}^0(\widehat{Y}=1) + \left|\gamma_0 - \gamma_1\right|\mathcal{D}^1(\widehat{Y}=0) \\ &= 2\Delta_{\mathrm{EO}}(\widehat{Y}) + \left|\gamma_0 - \gamma_1\right|\mathcal{D}^0(\widehat{Y}=1) + \left|\gamma_0 - \gamma_1\right|\mathcal{D}^1(\widehat{Y}=0) \\ &= 2\Delta_{\mathrm{EO}}(\widehat{Y}) + \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) \cdot \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y),\end{aligned}$$

which completes the proof. ■

We also provide the proof of Corollary 7.3.1:

Corollary 7.3.1. For any joint distribution $\mathcal{D}$ and classifier $\widehat{Y}$, if $\widehat{Y}$ satisfies equalized odds, then

$$\max\{\varepsilon_{\mathcal{D}_0}(\widehat{Y}), \varepsilon_{\mathcal{D}_1}(\widehat{Y})\} \leq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) \cdot \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)/2 + \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y).$$

Proof. We first invoke Theorem 7.3.3, if $\widehat{Y}$ satisfies equalized odds, then $\Delta_{\mathrm{EO}}(\widehat{Y}) = 0$, which implies:

$$\Delta_\varepsilon(\widehat{Y}) = \left|\varepsilon_{\mathcal{D}_0}(\widehat{Y}) - \varepsilon_{\mathcal{D}_1}(\widehat{Y})\right| \leq \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) \cdot \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y).$$

On the other hand, by Theorem 7.3.2, we know that

$$\varepsilon_{\mathcal{D}_0}(\widehat{Y}) + \varepsilon_{\mathcal{D}_1}(\widehat{Y}) \leq 2\mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y).$$

Combine the above two inequalities and recall that $\max\{a, b\} = (|a+b| + |a-b|)/2, \forall a, b \in \mathbb{R}$, yielding:

$$\begin{aligned}\max\{\varepsilon_{\mathcal{D}_0}(\widehat{Y}), \varepsilon_{\mathcal{D}_1}(\widehat{Y})\} &= \frac{|\varepsilon_{\mathcal{D}_0}(\widehat{Y}) - \varepsilon_{\mathcal{D}_1}(\widehat{Y})| + |\varepsilon_{\mathcal{D}_0}(\widehat{Y}) + \varepsilon_{\mathcal{D}_1}(\widehat{Y})|}{2} \\ &\leq \frac{\Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) \cdot \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y) + 2\mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)}{2} \\ &= \Delta_{\mathrm{BR}}(\mathcal{D}_0, \mathcal{D}_1) \cdot \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y)/2 + \mathrm{BER}_{\mathcal{D}}(\widehat{Y} \parallel Y),\end{aligned}$$

completing the proof. ■

7.7 Conclusion

In this chapter we propose a novel representation learning algorithm that aims to simultaneously ensure accuracy parity and equalized odds. The main idea underlying the design of our algorithm is to align the conditional distributions of representations (rather than marginal distributions) and use balanced error rate (i.e., the conditional error) on both the target variable and the sensitive attribute. Theoretically, we prove how these two concepts together help to ensure accuracy parity and equalized odds without impacting demographic parity, and we also show how these two can be used to give a guarantee on the joint error across different demographic subgroups. Empirically, we demonstrate on two real-world experiments that the proposed algorithm effectively leads to the desired notions of fairness, and it also leads to better utility-fairness trade-off on balanced datasets.

Calibration and Utility Our work takes a step towards better understanding the relationships between different notions of fairness and their corresponding trade-off with utility. In some scenarios, e.g., the COMPAS tool, it is desired to have a decision making system that is also well calibrated. While it is well-known that statistical calibration is not compatible with demographic parity or equalized odds, from a theoretical standpoint it is still not clear whether calibration will harm utility and if so, what is the fundamental limit of a calibrated tool on utility.

Fairness and Privacy Future work could also investigate how to make use of the close relationship between privacy and group fairness. At a colloquial level, fairness constraints require a predictor to be (to some extent) agnostic about the group membership attribute. The membership query attack in privacy asks the same question – is it possible to guarantee that even an optimal adversary cannot steal personal information through inference attacks. Prior work (Dwork et al., 2012) has described the connection between the notion of individual fairness and differential privacy. Hence it would be interesting to exploit techniques developed in the literature of privacy to develop more efficient fairness-aware learning algorithms. On the other hand, results obtained in the algorithmic fairness literature could also potentially lead to better privacy-preserving machine learning algorithms (Zhao et al., 2019a).

Appendix

7.A Experimental Details

7.A.1 The Adult Experiment

For the baseline network NoDebias, we implement a three-layer neural network with ReLU as the hidden activation function and logistic regression as the target output function. The input layer contains 114 units, and the hidden layer contains 60 hidden units. The output layer only contain one unit, whose output is interpreted as the probability of $\mathcal{D}(\widehat{Y} = 1 \mid X = x)$.

For the adversary in Fair and Laftr, again, we use a three-layer feed-forward network. Specifically, the input layer of the adversary is the hidden representations of the baseline network that contains 60 units. The hidden layer of the adversary network contains 50 units, with ReLU activation. Finally, the output of the adversary also contains one unit, representing the adversary's inference probability $\mathcal{D}(\widehat{A} = 1 \mid Z = z)$. The network structure of the adversaries in both CFair and CFair-EO are exactly the same as the one used in Fair and Laftr, except that there are two adversaries, one for $\mathcal{D}^0(\widehat{A} = 1 \mid Z = z)$ and one for $\mathcal{D}^1(\widehat{A} = 1 \mid Z = z)$.

The hyperparameters used in the experiment are listed in Table 7.A.1.

Table 7.A.1: Hyperparameters used in the Adult experiment.

Optimization Algorithm	AdaDelta
Learning Rate	1.0
Batch Size	512
Training Epochs $\lambda \in \{0.1, 1.0, 10.0, 100.0, 1000.0\}$	100

7.A.2 The COMPAS Experiment

Again, for the baseline network NoDebias, we implement a three-layer neural network with ReLU as the hidden activation function and logistic regression as the target output function. The input layer contains 11 units, and the hidden layer contains 10 hidden units. The output layer only contain one unit, whose output is interpreted as the probability of $\mathcal{D}(\widehat{Y} = 1 \mid X = x)$.

For the adversary in Fair and Laftr, again, we use a three-layer feed-forward network. Specifically, the input layer of the adversary is the hidden representations of the baseline network that contains 60 units. The hidden layer of the adversary network contains 10 units, with ReLU activation. Finally, the output of the adversary also contains one unit, representing the adversary's inference probability $\mathcal{D}(\widehat{A} = 1 \mid Z = z)$. The network structure of the adversaries in both CFair and CFair-EO are exactly the same as the one used in Fair and Laftr, except that there are two adversaries, one for $\mathcal{D}^0(\widehat{A} = 1 \mid Z = z)$ and one for $\mathcal{D}^1(\widehat{A} = 1 \mid Z = z)$.

The hyperparameters used in the experiment are listed in Table 7.A.2.

Table 7.A.2: Hyperparameters used in the COMPAS experiment.

Optimization Algorithm	AdaDelta
Learning Rate	1.0
Batch Size	512
Training Epochs $\lambda \in \{0.1, 1.0\}$	20
Training Epochs $\lambda = 10.0$	15

Chapter 8

Learning Language-Invariant Representations for Universal Machine Translation

The goal of universal machine translation is to learn to translate between any pair of languages, given a corpus of paired translated documents for *a small subset* of all pairs of languages. Despite impressive empirical results and an increasing interest in massively multilingual models, theoretical analysis on translation errors made by such universal machine translation models is only nascent. In this chapter, we formally prove certain impossibilities of this endeavour in general, as well as prove positive results in the presence of additional (but natural) structure of data. For the former, we derive a lower bound on the translation error in the many-to-many translation setting, which shows that any algorithm aiming to learn shared sentence representations among multiple language pairs has to make a large translation error on at least one of the translation tasks, if no assumption on the structure of the languages is made. For the latter, we show that if the paired documents in the corpus follow a natural *encoder-decoder* generative process, we can expect a natural notion of "generalization": a linear number of language pairs, rather than quadratic, suffices to learn a good representation. Our theory also explains what kinds of connection graphs between pairs of languages are better suited: ones with longer paths result in worse sample complexity in terms of the total number of documents per language pair needed.

8.1 Introduction

Despite impressive improvements in neural machine translation (NMT), training a large multilingual NMT model with hundreds of millions of parameters usually requires a collection of *parallel corpora* at a large scale, on the order of millions or even billions of aligned sentences (Arivazhagan et al., 2019; Johnson et al., 2017) for supervised training. Although it is possible to automatically crawl the web (Nie et al., 1999; Resnik, 1999; Resnik and Smith, 2003) to collect parallel sentences for high-resource language pairs such as German-English and Chinese-English, it is often infeasible or expensive to manually translate large amounts of documents for low-resource language pairs, e.g., Nepali-English, Sinhala-English (Guzmán et al., 2019). Much recent progress in low-resource machine translation, has been driven by the idea of *universal machine translation* (UMT), also known as *multilingual machine translation* (Gu et al., 2018; Johnson et al., 2017; Zoph and Knight, 2016), which aims at training one single NMT to translate between multiple source and target languages. Typical UMT models leverage either a single shared encoder or language-specific encoders to map all source languages to a shared space, and translate the source sentences to a target language by a decoder. Inspired by the idea of UMT, there has been a recent trend towards learning language-invariant embeddings for multiple source languages in a shared latent space, which eases the cross-lingual generalization from high-resource languages to low-resource languages on many tasks, e.g., parallel corpus mining (Artetxe and Schwenk, 2019; Schwenk, 2018), sentence classification (Conneau et al., 2018b), cross-lingual information retrieval (Litschko et al., 2018), and dependency parsing (Kondratyuk and Straka, 2019), just to name a few.

The idea of finding an abstract "lingua franca" is very intuitive and the empirical results are impressive, yet theoretical understanding of various aspects of universal machine translation is limited. In this chapter, we particularly focus on two basic questions:

1. *How can we measure the inherent tradeoff between the quality of translation and how language-invariant a representation is?*
2. *How many language pairs do we need aligned sentences for, to be able to translate between any pair of languages?*

Toward answering the first question, we show that in a completely assumption-free setup on the languages and distribution of the data, it is impossible to avoid making a large translation error on at least one pair of the translation tasks. Informally we highlight our first theorem as follows, and provide the formal statements in Theorems 8.2.1 and 8.2.2.

Theorem 8.1.1 (Impossibility, Informal)**.** There exist a choice of distributions over documents from different languages, s.t. for any choice of maps from the language to a common representation, at least one of the translation pairs must incur a high cost. In addition, there is an inherent tradeoff between the translation quality and the degree of representation invariance w.r.t. languages: the better the language invariance, the higher the cost on at least one of the translation pairs.

To answer the second question, we show that under fairly mild generative assumptions on the aligned documents for the pairwise translations, it is possible to not only do well on all of the pairwise translations, but also be able to do so after *only seeing* aligned documents of a *linear* number of languages, rather than a *quadratic* one. We summarize the second theorem as follows, and provide a formal statement in Theorem 8.3.1.

Theorem 8.1.2 (Sample complexity, Informal)**.** Under a generative model where the documents for each language are generated from a "ground-truth" encoder-decoder model, after seeing aligned documents for a *linear* number of pairs of languages, we can learn encoders/decoders that perform well on any unseen language pair.

Notation and Setup We first introduce the notation used throughout the chapter and then briefly describe the problem setting of universal machine translation.

We use $\mathcal{L}$ to denote the set of all possible languages, e.g., $\{\text{English}, \text{French}, \text{German}, \text{Chinese}, \ldots\}$. For any language $L \in \mathcal{L}$, we associate with L an alphabet Σ_L that contains all the symbols from L. Note that we assume $|\Sigma_L| < \infty$, $\forall L \in \mathcal{L}$, but different languages could potentially share part of the alphabet. Given a language L, a sentence x in L is a sequence of symbols from Σ_L, and we denote Σ_L^* as the set of all sentences generated from Σ_L. Note that since in principle different languages could share the same alphabet, to avoid ambiguity, for each language L, there is a unique token $\langle L\rangle \in \Sigma_L$ and $\langle L\rangle \notin \Sigma'_L, \forall L' \neq L$. The goal of the unique token $\langle L\rangle$ is used to denote the source sentence, and a sentence x in L will have a unique prefix $\langle L\rangle$ to indicate that $x \in \Sigma_L^*$. Also, in this manuscript we will use sentence and string interchangeably.

Formally, let $\{L_i\}_{i\in[K]}$[1] be the set of K source languages and $L \notin \{L_i\}_{i\in[K]}$ be the target language we are interested in translating to. For a pair of languages L and L', we use $\mathcal{D}_{L,L'}$ to denote the joint distribution over the parallel sentence pairs from L and L'. Given this joint distribution, we also use $\mathcal{D}_{L,L'}(L)$ to mean the marginal distribution over sentences from L. Likewise we use $\mathcal{D}_{L,L'}(L')$ to denote the corresponding marginal distribution over sentences from L'. Finally, for two sets A and B, we use $A \sqcup B$ to denote the disjoint union of A and B. In particular, when A and B are disjoint, their disjoint union equals the usual set union, i.e., $A \sqcup B = A \bigcup B$.

8.2 An Impossibility Theorem

In this section, for the clarity of presentation, we first focus the deterministic setting where for each language pair $L \neq L'$, there exists a ground-truth translator $f^*_{L\to L'} : \Sigma_L^* \to \Sigma_{L'}^*$ that takes an input sentence x from the source language L and outputs the ground-truth translation $f^*_{L\to L'}(x) \in \Sigma_{L'}^*$. Later we shall extend the setup to allow a probabilistic extension as well. Before we proceed, we first describe some concepts that will be used in the discussion.

Given a feature map $g : \mathcal{X} \to \mathcal{Z}$ that maps instances from the input space $\mathcal{X}$ to feature space $\mathcal{Z}$, we define $g_\sharp\mathcal{D} := \mathcal{D} \circ g^{-1}$ to be the induced (pushforward) distribution of $\mathcal{D}$ under g, i.e., for any event $E' \subseteq \mathcal{Z}$, $\Pr_{g_\sharp\mathcal{D}}(E') := \Pr_{\mathcal{D}}(g^{-1}(E')) = \Pr_{\mathcal{D}}(\{x \in \mathcal{X} \mid g(x) \in E'\})$. For two distribution $\mathcal{D}$ and $\mathcal{D}'$ over the same sample space, we use the total variation distance to measure the discrepancy them: $d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}') := \sup_E |\Pr_{\mathcal{D}}(E) - \Pr_{\mathcal{D}'}(E)|$, where E is taken over all the measurable events under the common sample space. We use $\mathbb{I}(E)$ to denote the indicator function which takes value 1 iff the event E is true otherwise 0.

In general, given two sentences x and x', we use $\ell(x, x')$ to denote the loss function used to measure their distance. For example, we could use a $0-1$ loss function $\ell_{0-1}(x, x') = 0$ iff $x = x'$ else 1. If both x and x' are embedded in the same Euclidean space, we could also use the squared loss $\ell_2(x, x')$ as a more refined measure. To measure the performance of a translator f on a given language pair $L \to L'$ w.r.t. the ground-truth translator $f^*_{L\to L'}$, we define the error function of f as

$$\varepsilon_{\mathcal{D}}^{L\to L'}(f) := \mathbb{E}_{\mathcal{D}}\left[\ell_{0-1}(f(X), f^*_{L\to L'}(X))\right],$$

which is the translation error of f as compared to the ground-truth translator $f^*_{L\to L'}$. For universal machine translation, the input string of the translator can be any sentence from any language. To this end, let Σ^* be the union of all the sentences/strings from all the languages of interest: $\Sigma^* := \bigcup_{L\in\mathcal{L}} \Sigma_L^*$. Then a universal machine translator of target language $L \in \mathcal{L}$ is a mapping $f_L : \Sigma^* \to \Sigma_L^*$. In words, f_L takes as input a

[1] We use $[K]$ to denote the set $\{0, 1, \ldots, K-1\}$.

string (from one of the possible languages) and outputs the corresponding translation in target language L. It is not hard to see that for such task there exists a perfect translator f_L^*:

$$f_L^*(x) = \sum_{L' \in \mathcal{L}} \mathbb{I}(x \in \Sigma_{L'}^*) \cdot f_{L' \to L}^*(x). \tag{8.1}$$

Note that $\{\Sigma_{L'}^* \mid L' \in \mathcal{L}\}$ forms a partition of Σ^*, so exactly one of the indicator $\mathbb{I}(x \in \Sigma_{L'}^*)$ in (8.1) will take value 1.

Given a target language L, existing approaches for universal machine seek to find an intermediate space $\mathcal{Z}$, such that source sentences from different languages are aligned within $\mathcal{Z}$. In particular, for each source language L', the goal is to find a feature mapping $g_{L'} : \Sigma_{L'}^* \to \mathcal{Z}$ so that the induced distributions of different languages are close in $\mathcal{Z}$. The next step is to construct a decoder $h : \mathcal{Z} \to \Sigma_L^*$ that maps feature representation in $\mathcal{Z}$ to sentence in the target language L.

One interesting question about the idea of learning language-invariant representations is that, whether such method will succeed even under the benign setting where there is a ground-truth universal translator and the learner has access to infinite amount of data with unbounded computational resources. That is, we are interested in understanding the information-theoretic limit of such methods for universal machine translation.

In this section we first present an impossibility theorem in the restricted setting of translating from two source languages L_0 and L_1 to a target language L. Then we will use this lemma to prove a lower bound of the universal translation error in the general many-to-many setting. We will mainly discuss the implications and intuition of our theoretical results and use figures to help illustrate the high-level idea of the proof.

8.2.1 Two-to-One Translation

Recall that for each translation task $L_i \to L$, we have a joint distribution $\mathcal{D}_{L_i,L}$ (parallel corpora) over the aligned source-target sentences. For convenience of notation, we use $\mathcal{D}_i$ to denote the marginal distribution $\mathcal{D}_{L_i,L}(L_i)$, $\forall i \in [K]$ when the target language L is clear from the context. Given a fixed constant $\epsilon > 0$, we first define the *ϵ-universal language mapping*:

Definition 8.2.1 (ϵ-Universal Language Mapping)**.** A map $g : \bigcup_{i \in [K]} \Sigma_{L_i}^* \to \mathcal{Z}$ is called an ϵ-universal language mapping if $d_{\text{TV}}(g_\sharp \mathcal{D}_i, g_\sharp \mathcal{D}_j) \leq \epsilon$, $\forall i \neq j$.

In particular, if $\epsilon = 0$, we call the corresponding feature mapping a universal language mapping. In other words, a universal language mapping perfectly aligns the feature representations of different languages in feature space $\mathcal{Z}$. The following lemma provides a useful tool to connect the 0-1 translation error and the TV distance between the corresponding distributions.

Lemma 8.2.1. Let $\Sigma := \bigcup_{L \in \mathcal{L}} \Sigma_L$ and $\mathcal{D}_\Sigma$ be a language model over Σ^*. For any two string-to-string maps $f, f' : \Sigma^* \to \Sigma^*$, let $f_\sharp \mathcal{D}_\Sigma$ and $f'_\sharp \mathcal{D}_\Sigma$ be the corresponding pushforward distributions. Then $d_{\text{TV}}(f_\sharp \mathcal{D}_\Sigma, f'_\sharp \mathcal{D}_\Sigma) \leq \Pr_{\mathcal{D}_\Sigma}(f(X) \neq f'(X))$ where $X \sim \mathcal{D}_\Sigma$.

Proof. Note that the sample space Σ^* is countable. For any two distributions $\mathcal{P}$ and $\mathcal{Q}$ over Σ^*, it is a well-known fact that $d_{\mathrm{TV}}(\mathcal{P}, \mathcal{Q}) = \frac{1}{2}\sum_{y\in\Sigma^*} |\mathcal{P}(y) - \mathcal{Q}(y)|$. Using this fact, we have:

$$\begin{aligned}
d_{\mathrm{TV}}(f_\sharp\mathcal{D}, f'_\sharp\mathcal{D}) &= \frac{1}{2}\sum_{y\in\Sigma^*} \left|f_\sharp\mathcal{D}(y) - f'_\sharp\mathcal{D}(y)\right| \\
&= \frac{1}{2}\sum_{y\in\Sigma^*} \left|\Pr_{\mathcal{D}}(f(X) = y) - \Pr_{\mathcal{D}}(f'(X) = y)\right| \\
&= \frac{1}{2}\sum_{y\in\Sigma^*} \left|\mathbb{E}_{\mathcal{D}}[\mathbb{I}(f(X) = y)] - \mathbb{E}_{\mathcal{D}}[\mathbb{I}(f'(X) = y)]\right| \\
&\le \frac{1}{2}\sum_{y\in\Sigma^*} \mathbb{E}_{\mathcal{D}}\left[\left|\mathbb{I}(f(X) = y) - \mathbb{I}(f'(X) = y)\right|\right] \\
&= \frac{1}{2}\sum_{y\in\Sigma^*} \mathbb{E}_{\mathcal{D}}[\mathbb{I}(f(X) = y, f'(X) \neq y) + \mathbb{I}(f(X) \neq y, f'(X) = y)] \\
&= \frac{1}{2}\sum_{y\in\Sigma^*} \mathbb{E}_{\mathcal{D}}\left[\mathbb{I}(f(X) = y, f'(X) \neq f(X))\right] + \mathbb{E}_{\mathcal{D}}\left[\mathbb{I}(f'(X) = y, f'(X) \neq f(X))\right] \\
&= \sum_{y\in\Sigma^*} \mathbb{E}_{\mathcal{D}}\left[\mathbb{I}(f(X) = y, f'(X) \neq f(X))\right] \\
&= \sum_{y\in\Sigma^*} \Pr_{\mathcal{D}}\left(f(X) = y, f'(X) \neq f(X)\right) \\
&= \Pr_{\mathcal{D}}(f(X) \neq f'(X)).
\end{aligned}$$

The second equality holds by the definition of the pushforward distribution. The inequality on the fourth line holds due to the triangle inequality and the equality on the seventh line is due to the symmetry between $f(X)$ and $f'(X)$. The last equality holds by the total law of probability. ■

The next lemma follows from the data-processing inequality for total variation and it shows that if languages are close in a feature space, then any decoder cannot increase the corresponding discrepancy of these two languages in the output space.

Lemma 8.2.2. (Data-processing inequality) Let $\mathcal{D}$ and $\mathcal{D}'$ be any distributions over $\mathcal{Z}$, then for any decoder $h : \mathcal{Z} \to \Sigma_L^*$, $d_{\mathrm{TV}}(h_\sharp\mathcal{D}, h_\sharp\mathcal{D}') \le d_{\mathrm{TV}}(\mathcal{D}, \mathcal{D}')$.

As a direct corollary, this implies that any distributions induced by a decoder over ϵ-universal language mapping must also be close in the output space:

Corollary 8.2.1. If $g : \Sigma^* \to \mathcal{Z}$ is an ϵ-universal language mapping, then for any decoder $h : \mathcal{Z} \to \Sigma_L^*$, $d_{\mathrm{TV}}((h \circ g)_\sharp\mathcal{D}_0, (h \circ g)_\sharp\mathcal{D}_1) \le \epsilon$.

With the above tools, we can state the following theorem that characterizes the translation error in a two-to-one setting:

Theorem 8.2.1. (Lower bound, Two-to-One) Consider a restricted setting of universal machine translation task with two source languages where $\Sigma^* = \Sigma_{L_0}^* \bigcup \Sigma_{L_1}^*$ and the target language is L. Let $g : \Sigma^* \to \mathcal{Z}$ be an ϵ-universal language mapping, then for any decoder $h : \mathcal{Z} \to \Sigma_L^*$, we have

$$\varepsilon_{\mathcal{D}_0}^{L_0\to L}(h \circ g) + \varepsilon_{\mathcal{D}_1}^{L_1\to L}(h \circ g) \ge d_{\mathrm{TV}}(\mathcal{D}_{L_0,L}(L), \mathcal{D}_{L_1,L}(L)) - \epsilon. \tag{8.2}$$

Proof of Theorem 8.2.1. First, realize that $d_{\mathrm{TV}}(\cdot, \cdot)$ is a distance metric, the following chain of triangle

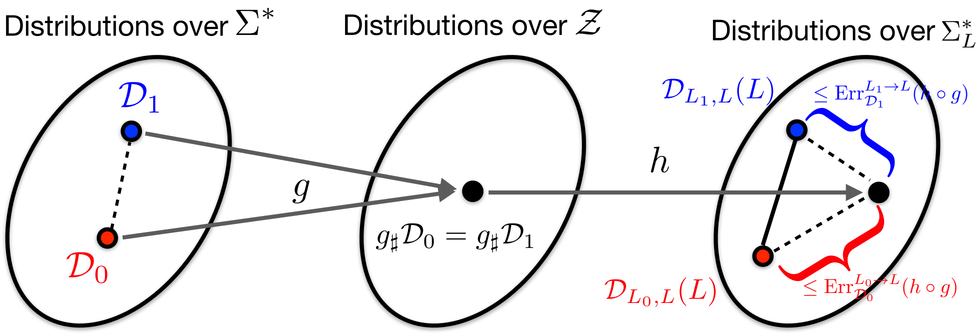

Distributions over Σ^*
Distributions over $\mathcal{Z}$
Distributions over Σ_L^*
$\mathcal{D}_1$
$\mathcal{D}_0$
g
$g_\sharp\mathcal{D}_0 = g_\sharp\mathcal{D}_1$
h
$\mathcal{D}_{L_1,L}(L)$
$\leq \mathrm{Err}_{\mathcal{D}_1}^{L_1\to L}(h\circ g)$
$\mathcal{D}_{L_0,L}(L)$
$\leq \mathrm{Err}_{\mathcal{D}_0}^{L_0\to L}(h\circ g)$

Figure 8.2.1: Proof by picture: Language-invariant representation g induces the same feature distribution over $\mathcal{Z}$, which leads to the same output distribution over the target language Σ_L^*. However, the parallel corpora of the two translation tasks in general have different marginal distributions over the target language, hence a triangle inequality over the output distributions gives the desired lower bound.

inequalities hold:

$$\begin{aligned} d_{\mathrm{TV}}(\mathcal{D}_{L_0,L}(L), \mathcal{D}_{L_1,L}(L)) \leq\ & d_{\mathrm{TV}}(\mathcal{D}_{L_0,L}(L), (h\circ g)_\sharp\mathcal{D}_0) \\ & + d_{\mathrm{TV}}((h\circ g)_\sharp\mathcal{D}_1, \mathcal{D}_{L_1,L}(L)) \\ & + d_{\mathrm{TV}}((h\circ g)_\sharp\mathcal{D}_0, (h\circ g)_\sharp\mathcal{D}_1). \end{aligned}$$

Now by the assumption that g is an ϵ-universal language mapping and Corollary 8.2.1, the third term on the RHS of the above inequality, $d_{\mathrm{TV}}((h\circ g)_\sharp\mathcal{D}_0, (h\circ g)_\sharp\mathcal{D}_1)$, is upper bounded by ϵ. Furthermore, note that since the following equality holds:

$$\mathcal{D}_{L_i,L}(L) = f^*_{L_i\to L_\sharp}\mathcal{D}_i, \quad \forall i\in\{0,1\},$$

we can further simplify the above inequality as

$$d_{\mathrm{TV}}(\mathcal{D}_{L_0,L}(L), \mathcal{D}_{L_1,L}(L)) \leq d_{\mathrm{TV}}(f^*_{L_0\to L_\sharp}\mathcal{D}_0, (h\circ g)_\sharp\mathcal{D}_0) + d_{\mathrm{TV}}((h\circ g)_\sharp\mathcal{D}_1, f^*_{L_1\to L_\sharp}\mathcal{D}_1) + \epsilon.$$

Now invoke Lemma 8.2.1 for $i\in\{0,1\}$ to upper bound the first two terms on the RHS, yielding:

$$d_{\mathrm{TV}}(f^*_{L_i\to L_\sharp}\mathcal{D}_i, (h\circ g)_\sharp\mathcal{D}_i) \leq \Pr_{\mathcal{D}_i}\left((h\circ g)(X) \neq f^*_{L_i\to L}(X)\right) = \varepsilon_{\mathcal{D}_i}^{L_i\to L}(h\circ g).$$

A simple rearranging then completes the proof. ■

Remark Recall that under our setting, there exists a perfect translator $f_L^* : \Sigma^* \to \Sigma_L^*$ in (8.1) that achieves zero translation error on both translation tasks. Nevertheless, the lower bound in Theorem 8.2.1 shows that one cannot hope to simultaneously minimize the joint translation error on both tasks through universal language mapping. Second, the lower bound is algorithm-independent and it holds even with unbounded computation and data. Third, the lower bound also holds even if all the data are perfect, in the sense that all the data are sampled from the perfect translator on each task. Hence, the above result could be interpreted as a kind of *uncertainty principle* in the context of universal machine translation, which says that any decoder based on language-invariant representations has to achieve a large translation error on at least one pair of translation task. We provide a proof-by-picture in Fig. 8.2.1 to illustrate the main idea underlying the proof of Theorem 8.2.1 in the special case where $\epsilon = 0$.

The lower bound is large whenever the distribution over target sentences differ between these two translation tasks. This often happens in practical scenarios where the parallel corpus of high-resource language pair contains texts over a diverse domain whereas as a comparison, parallel corpus of low-resource language pair only contains target translations from a specific domain, e.g., sports, news, product reviews, etc. Such negative impact on translation quality due to domain mismatch between source and target sentences has also recently been observed and confirmed in practical universal machine translation systems, see Shen et al. (2019) and Pires et al. (2019) for more empirical corroborations.

8.2.2 Many-to-Many Translation

Theorem 8.2.1 presents a negative result in the setting where we have two source languages and one target language for translation. Nevertheless universal machine translation systems often involve multiple input and output languages simultaneously (Artetxe and Schwenk, 2019; Johnson et al., 2017; Ranzato et al., 2019; Wu et al., 2016). In this section we shall extend the previous lower bound in the simple two-to-one setting to the more general translation task of many-to-many setting.

To enable such extension, i.e., to be able to make use of multilingual data within a single system, we need to modify the input sentence to introduce the language token $\langle L \rangle$ at the beginning of the input sentence to indicate the target language L the model should translate to. This simple modification has already been proposed and used in practical MT systems (Johnson et al., 2017, Section 3). To give an example, consider the following English sentence to be translated to French,

⟨English⟩ Hello, how are you?

It will be modified to:

⟨French⟩⟨English⟩ Hello, how are you?

Note that the first token is used to indicate the target language to translate to while the second one is used to indicate the source language to avoid the ambiguity due to the potential overlapping alphabets between different languages.

Recall in Definition 8.2.1 we define a language map g to be ϵ-universal iff $d_{\text{TV}}(g_\sharp \mathcal{D}_i, g_\sharp \mathcal{D}_j) \leq \epsilon, \forall i, j$. This definition is too stringent in the many-to-many translation setting since this will imply that the feature representations lose the information about which target language to translate to. In what follows we shall first provide a relaxed definition of ϵ-universal language mapping in the many-to-many setting and then show that even under this relaxed definition, learning universal machine translator via language-invariant representations is impossible in the worst case.

Definition 8.2.2 (ϵ-Universal Language Mapping, Many-to-Many). Let $\mathcal{D}_{L_i,L_k}$, $i, k \in [K]$ be the joint distribution of sentences (parallel corpus) in translating from L_i to L_k. A map $g : \bigcup_{i \in [K]} \Sigma^*_{L_i} \to \mathcal{Z}$ is called an ϵ-universal language mapping if there exists a partition of $\mathcal{Z} = \sqcup_{k \in [K]} \mathcal{Z}_k$ such that $\forall k \in [K]$ and $\forall i \neq j$, $g_\sharp \mathcal{D}_{L_i,L_k}(L_i)$ and $g_\sharp \mathcal{D}_{L_j,L_k}(L_j)$ are supported on $\mathcal{Z}_k$ and $d_{\text{TV}}(g_\sharp \mathcal{D}_{L_i,L_k}(L_i), g_\sharp \mathcal{D}_{L_j,L_k}(L_j)) \leq \epsilon$.

First of all, it is clear that when there is only one target language, then Definition 8.2.2 reduces to Definition 8.2.1. Next, the partition of the feature space $\mathcal{Z} = \sqcup_{k \in [K]} \mathcal{Z}_k$ essentially serves as a way to determine the target language L the model should translate to. Note that it is important here to enforce the partitioning condition of the feature space $\mathcal{Z}$, otherwise there will be ambiguity in determining the target language to translate to. For example, if the following two input sentences

⟨French⟩⟨English⟩ Hello, how are you?

⟨Chinese⟩⟨English⟩ Hello, how are you?

are mapped to the same feature representation $z \in \mathcal{Z}$, then it is not clear whether the decoder h should translate z to French or Chinese.

With the above extensions, now we are ready to present the following theorem which gives a lower bound for both the maximum error as well as the average error in the many-to-many universal translation setting.

Theorem 8.2.2. (Lower bound, Many-to-Many) Consider a universal machine translation task where $\Sigma^* = \bigcup_{i\in[K]} \Sigma^*_{L_i}$. Let $\mathcal{D}_{L_i,L_k}$, $i,k \in [K]$ be the joint distribution of sentences (parallel corpus) in translating from L_i to L_k. If $g : \Sigma^* \to \mathcal{Z}$ be an ϵ-universal language mapping, then for any decoder $h : \mathcal{Z} \to \Sigma^*$, we have

$$\max_{i,k\in[K]} \varepsilon^{L_i\to L_k}_{\mathcal{D}_{L_i,L_k}}(h\circ g) \geq \frac{1}{2}\max_{k\in[K]}\max_{i\neq j}\; d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k)) - \frac{\epsilon}{2},$$

$$\frac{1}{K^2}\sum_{i,k\in[K]} \varepsilon^{L_i\to L_k}_{\mathcal{D}_{L_i,L_k}}(h\circ g) \geq \frac{1}{K^2(K-1)}\sum_{k\in[K]}\sum_{i<j} d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k)) - \frac{\epsilon}{2}.$$

Proof of Theorem 8.2.2. First let us fix a target language L_k. For each pair of source languages L_i, L_j, $i \neq j$ translating to L_k, applying Theorem 8.2.1 gives us:

$$\varepsilon^{L_i\to L_k}_{\mathcal{D}_{L_i,L_k}}(h\circ g) + \varepsilon^{L_j\to L_k}_{\mathcal{D}_{L_j,L_k}}(h\circ g) \geq d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k)) - \epsilon. \tag{8.3}$$

Now consider the pair of source languages (L_{i^*}, L_{j^*}) with the maximum $d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k))$:

$$\begin{aligned} 2\max_{i\in[K]} \varepsilon^{L_i\to L_k}_{\mathcal{D}_{L_i,L_k}}(h\circ g) &\geq \varepsilon^{L_{i^*}\to L_k}_{\mathcal{D}_{L_{i^*},L_k}}(h\circ g) + \varepsilon^{L_{j^*}\to L_k}_{\mathcal{D}_{L_{j^*},L_k}}(h\circ g) \\ &\geq \max_{i\neq j}\; d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k)) - \epsilon. \end{aligned} \tag{8.4}$$

Since the above lower bound (8.4) holds for any target language L_k, taking a maximum over the target languages yields:

$$2\max_{i,k\in[K]} \varepsilon^{L_i\to L_k}_{\mathcal{D}_{L_i,L_k}}(h\circ g) \geq \max_{k\in[K]}\max_{i\neq j}\; d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k)) - \epsilon,$$

which completes the first part of the proof. For the second part, again, for a fixed target language L_k, to lower bound the average error, we apply the triangle inequality in (8.3) iteratively for all pairs $i < j$, yielding:

$$(K-1)\sum_{i\in[K]} \varepsilon^{L_i\to L_k}_{\mathcal{D}_{L_i,L_k}}(h\circ g) \geq \sum_{i<j} d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k)) - \frac{K(K-1)}{2}\epsilon.$$

Dividing both sides by $K(K-1)$ gives the average translation error to L_k. Now summing over all the possible target language L_k yields:

$$\frac{1}{K^2}\sum_{i,k\in[K]} \varepsilon^{L_i\to L_k}_{\mathcal{D}_{L_i,L_k}}(h\circ g) \geq \frac{1}{K^2(K-1)}\sum_{k\in[K]}\sum_{i<j} d_{\mathrm{TV}}(\mathcal{D}_{L_i,L_k}(L_k), \mathcal{D}_{L_j,L_k}(L_k)) - \frac{\epsilon}{2}. \qquad \blacksquare$$

It is clear from the proof above that both lower bounds in Theorem 8.2.2 include the many-to-one setting as a special case. The proof of Theorem 8.2.2 essentially applies the lower bound in Theorem 8.2.1 iteratively. Again, the underlying reason for such negative result to hold in the worst case is due to the mismatch of distributions of the target language in different pairs of translation tasks. It should also be noted that the results in Theorem 8.2.2 hold even if language-dependent encoders are used, as long as they induce invariant feature representations for the source languages.

How to Bypass this Limitation? There are various ways to get around the limitations pointed out by the theorems in this section.

One way is to allow the decoder h to have access to the input sentences (besides the language-invariant representations) during the decoding process – e.g. via an attention mechanism on the input level. Technically, such information flow from input sentences during decoding would break the Markov structure of "input-representation-output" in Fig. 8.2.1, which is an essential ingredient in the proof of Theorem 8.2.1 and Theorem 8.2.2. Intuitively, in this case both language-invariant (hence language-independent) and language-dependent information would be used.

Another way would be to assume extra structure on the distributions $\mathcal{D}_{L_i,L_j}$, i.e., by assuming some natural language generation process for the parallel corpora that are used for training (Cf. Section 8.3). Since languages share a lot of semantic and syntactic characteristics, this would make a lot of sense — and intuitively, this is what universal translation approaches are banking on. In the next section we will do exactly this — we will show that under a suitable generative model, not only will there be a language-invariant representation, but it will be learnable using corpora from a very small (linear) number of pairs of language.

8.3 Sample Complexity under a Generative Model

The results from the prior sections showed that absent additional assumptions on the distributions of the sentences in the corpus, there is a fundamental limitation on learning language-invariant representations for universal machine translation. Note that our negative result also holds in the setting where there exists a ground-truth universal machine translator – it's just that learning language-invariant representations cannot lead to the recovery of this ground-truth translator.

In this section we show that with additional natural structure on the distribution of the corpora we can resolve this issue. The structure is a natural underlying generative model from which sentences from different languages are generated, which "models" a common encoder-decoder structure that has been frequently used in practice (Cho et al., 2014; Ha et al., 2016; Sutskever et al., 2014). Under this setting, we show that it is not only possible to learn the optimal translator, but it is possible to do so only seeing documents from only a small subset of all the possible language pairs.

Moreover, we will formalize a notion of "*sample complexity*" in terms of number of pairs of languages for which parallel corpora are necessary, and how it depends on the structure of the connection graph between language pairs.

We first describe our generative model for languages and briefly talk about why such generative model could help to overcome the negative result in Theorem 8.2.2.

8.3.1 Language Generation Process and Setup

Language Generative Process The language generation process is illustrated in Fig. 8.3.1. Formally, we assume the existence of a shared "semantic space" $\mathcal{Z}$. Furthermore, for every language $L \in \mathcal{L}$, we have a "ground truth" pair of encoder and decoder $(\mathbf{E}_L, \mathbf{D}_L)$, where $\mathbf{E}_L : \mathbb{R}^d \to \mathbb{R}^d$, $\mathbf{E}_L \in \mathcal{F}$ is *bijective* and $\mathbf{D}_L = \mathbf{E}_L^{-1}$. We assume that $\mathcal{F}$ has a group structure under function composition: namely, for $\forall f_1, f_2 \in \mathcal{F}$, we have that $f_1^{-1}, f_2^{-1} \in \mathcal{F}$ and $f_1 \circ f_2^{-1}, f_2 \circ f_1^{-1} \in \mathcal{F}$ (e.g., a typical example of such group is the general linear group $\mathcal{F} = GL_d(\mathbb{R})$).

To generate a pair of aligned sentences for two languages L, L', we first sample a $z \sim \mathcal{D}$, and subsequently generate

$$x = \mathbf{D}_L(z), \quad x' = \mathbf{D}_{L'}(z), \tag{8.5}$$

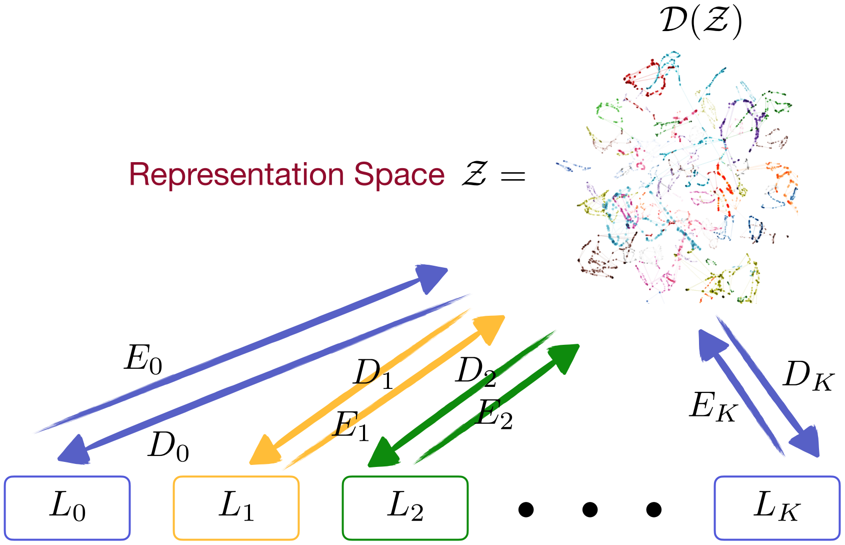

$\mathcal{D}(\mathcal{Z})$
Representation Space $\mathcal{Z} =$
E_0
D_0
D_1
E_1
D_2
E_2
E_K
D_K
L_0
L_1
L_2
L_K

Figure 8.3.1: An encoder-decoder generative model of translation pairs. There is a global distribution $\mathcal{D}$ over representation space $\mathcal{Z}$, from which sentences of language L_i are generated via decoder D_i. Similarly, sentences could also be encoded via E_i to $\mathcal{Z}$.

where the x is a vector encoding of the appropriate sentence in L (e.g., a typical encoding is a frequency count of the words in the sentence, or a sentence embedding using various neural network models (Kiros et al., 2015; Wang et al., 2017; Zhao et al., 2015a)). Similarly, x' is the corresponding sentence in L'. Reciprocally, given a sentence x from language L, the encoder $\mathbf{E}_L$ maps the sentence x into its corresponding latent vector in $\mathcal{Z}$: $z = \mathbf{E}_L(x)$.

We note that we assume this deterministic map between z and x for simplicity of exposition—in Section 8.3.4 we will extend the results to the setting where x has a conditional distribution given z of a parametric form.

We will assume the existence of a graph H capturing the pairs of languages for which we have aligned corpora – we can think of these as the "high-resource" pairs of languages. For each edge in this graph, we will have a corpus $S = \{(x_i, x_i')\}_{i=1}^n$ of aligned sentences.[2] The goal will be to learn encoder/decoders that perform well on the potentially unseen pairs of languages. To this end, we will be providing a sample complexity analysis for the number of paired sentences for each pair of languages with an edge in the graph, so we will need a measure of the complexity of $\mathcal{F}$. We will use the covering number, though our proofs are flexible, and similar results would hold for Rademacher complexity, VC dimension, or any of the usual complexity measures.

Definition 8.3.1 (Covering number)**.** For any $\epsilon > 0$, the *covering number* $\mathcal{N}(\mathcal{F}, \epsilon)$ of the function class $\mathcal{F}$ under the ℓ_∞ norm is the minimum number $k \in \mathbb{N}$ such that $\mathcal{F}$ could be covered with k (ℓ_∞) balls of radius ϵ, i.e., there exists $\{f_1, \ldots, f_k\} \subseteq \mathcal{F}$ such that, for all $f \in \mathcal{F}$, there exists $i \in [k]$ with $\|f - f_i\|_\infty = \max_{x \in \mathbb{R}^d} \|f(x) - f_i(x)\|_2 \leq \epsilon$.

Finally, we will assume that the functions in $\mathcal{F}$ are bounded and Lipschitz:

Assumption 8.3.1 (Smoothness and Boundedness)**.** $\mathcal{F}$ is bounded under the $\|\cdot\|_\infty$ norm, i.e., there exists $M > 0$, such that $\forall f \in \mathcal{F}$, $\|f\|_\infty \leq M$. Furthermore, there exists $0 \leq \rho < \infty$, such that for $\forall x, x' \in \mathbb{R}^d$,

[2] In general each edge can have different number of aligned sentences. We use the same number of aligned sentences n just for the ease of presentation.

$\forall f \in \mathcal{F}, \|f(x) - f(x')\|_2 \le \rho \cdot \|x - x'\|_2$.

Training Procedure Turning to the training procedure, we will be learning encoders $E_L \in \mathcal{F}$ for each language L. The decoder for that language will be E_L^{-1}, which is well defined since $\mathcal{F}$ has a group structure. Since we are working with a vector space, rather than using the (crude) 0-1 distance, we will work with a more refined loss metric for a translation task $L \to L'$:

$$\varepsilon(E_L, E_{L'}) := \|E_{L'}^{-1} \circ E_L - \mathbf{E}_{L'}^{-1} \circ \mathbf{E}_L\|^2_{\ell_2(\mathbf{D}_{L\sharp}\mathcal{D})}. \tag{8.6}$$

Note that the ℓ_2 loss is taken over the distribution of the input samples $\mathbf{D}_{L\sharp}\mathcal{D} = \mathbf{E}_L^{-1}{}_\sharp\mathcal{D}$, which is the natural one under our generative process. Again, the above error measures the discrepancy between the predicted translation w.r.t. the one give by the ground-truth translator, i.e., the composition of encoder $\mathbf{E}_L$ and decoder $\mathbf{D}_{L'}$. Straightforwardly, the empirical error over a corpus $S = \{(x_i, x_i')\}_{i=1}^n$ of aligned sentences for a pair of languages (L, L') is defined by

$$\widehat{\varepsilon}_S(E_L, E_{L'}) := \frac{1}{n}\sum_{i\in[n]} \|E_{L'}^{-1} \circ E_L(x_i) - x_i'\|_2^2, \tag{8.7}$$

where S is generated by the generation process. Following the paradigm of empirical risk minimization, the loss to train the encoders will be the obvious one:

$$\min_{\{E_L, L\in\mathcal{L}\}} \sum_{(L,L')\in H} \widehat{\varepsilon}_S(E_L, E_{L'}). \tag{8.8}$$

Remarks Before we proceed, one natural question to ask here is that, how does this generative model assumption circumvent the lower bound in Theorem 8.2.2? To answer this question, note the following easy proposition:

Proposition 8.3.1. Under the encoder-decoder generative assumption, $\forall i, j \in [K], d_{\mathrm{TV}}(\mathcal{D}_{L_i,L}(L), \mathcal{D}_{L_j,L}(L)) = 0$.

Proposition 8.3.1 holds because the marginal distribution of the target language L under any pair of translation task equals the pushforward of $\mathcal{D}(Z)$ under $\mathbf{D}_L$: $\forall i \in [K], \mathcal{D}_{L_i,L}(L) = \mathbf{D}_{L\sharp}\mathcal{D}(\mathcal{Z})$. Hence the lower bounds gracefully reduce to 0 under our encoder-decoder generative process, meaning that there is no loss of translation accuracy using universal language mapping.

8.3.2 Main Result: Translation between Arbitrary Pairs of Languages

The main theorem we prove is that if the graph H capturing the pairs of languages for which we have aligned corpora is connected, given sufficiently many sentences for each pair, we will learn encoder/decoders that perform well on the unseen pairs. Moreover, we can characterize how good the translation will be based on the distance of the languages in the graph. Concretely:

Theorem 8.3.1 (Sample complexity under generative model)**.** Suppose H is connected. Furthermore, suppose the trained $\{E_L\}_{L\in\mathcal{L}}$ satisfy

$$\forall L, L' \in H : \widehat{\varepsilon}_S(E_L, E_{L'}) \le \epsilon_{L,L'},$$

for $\epsilon_{L,L'} > 0$. Furthermore, for $0 < \delta < 1$ suppose the number of sentences for each aligned corpora for each training pair (L, L') is $\Omega\left(\frac{1}{\epsilon^2_{L,L'}} \cdot \left(\log \mathcal{N}(\mathcal{F}, \frac{\epsilon_{L,L'}}{16M}) + \log(K/\delta)\right)\right)$. Then, with probability $1 - \delta$, for any pair of languages $(L, L') \in \mathcal{L} \times \mathcal{L}$ and $L = L_1, L_2, \ldots, L_m = L'$ a path between L and L' in H, we have $\varepsilon(E_L, E_{L'}) \le 2\rho^2 \sum_{k=1}^{m-1} \epsilon_{L_k, L_{k+1}}$.

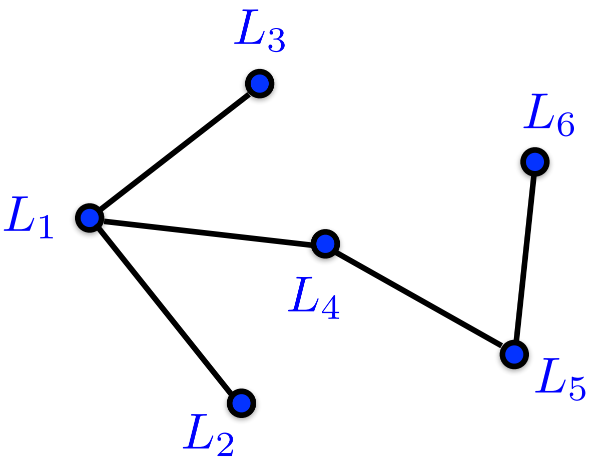


Figure 8.3.2: A translation graph H over $K = 6$ languages. The existence of an edge between a pair of nodes L_i and L_j means that the learner has been trained on the corresponding language pair. In this example the diameter of the graph $\mathrm{diam}(H) = 4$: L_3, L_1, L_4, L_5, L_6.

Remark We make several remarks about the theorem statement. Note that the guarantee is in terms of translation error rather than parameter recovery. In fact, due to the identifiability issue, we cannot hope to recover the ground truth encoders $\{\mathbf{E}_L\}_{L\in\mathcal{L}}$: it is easy to see that composing all the encoders with an invertible mapping $f \in \mathcal{F}$ and composing all the decoders with $f^{-1} \in \mathcal{F}$ produces exactly the same outputs.

Furthermore, the upper bound is adaptive, in the sense that for any language pair (L_i, L_j), the error depends on the sum of the errors connecting (L_i, L_j) in the translation graph H. One can think naturally as the low-error edges as resource-rich pairs: if the function class $\mathcal{F}$ is parametrized by finite-dimensional parameter space with dimension p, then using standard result on the covering number of finite-dimensional vector space (Anthony and Bartlett, 2009), we know that $\log \mathcal{N}(\mathcal{F}, \frac{\epsilon}{16M}) = \Theta(p \log(1/\epsilon))$; as a consequence, the number of documents needed for a pair scales as $\log(1/\epsilon_{L,L'})/\epsilon^2_{L,L'}$.

Furthermore, as an immediate corollary of the theorem, if we assume $\epsilon_{L,L'} \leq \epsilon$ for all $(L, L') \in H$, we have $\varepsilon(E_L, E_{L'}) \leq 2\rho^2 d_{L,L'} \cdot \epsilon$, where $d_{L,L'}$ is the length of the shortest path connecting L and L' in H. It also immediately follows that for any pair of languages L, L', we have $\varepsilon(E_L, E_{L'}) \leq 2\rho^2 \mathrm{diam}(H) \cdot \epsilon$ where $\mathrm{diam}(H)$ is the diameter of H – thus the intuitive conclusion that graphs that do not have long paths are preferable.

The upper bound in Theorem 8.3.1 also provides a counterpoint to the lower-bound, showing that under a generative model for the data, it is possible to learn a pair of encoder/decoder for each language pair after seeing aligned corpora only for a linear number of pairs of languages (and not quadratic!), corresponding to those captured by the edges of the translation graph H. As a final note, we would like to point out that an analogous bound can be proved easily for other losses like the 0-1 loss or the general ℓ_p loss as well.

8.3.3 Proof Sketch of the Theorem

Before we provide the proof for the theorem, we first state several useful lemmas that will be used during our analysis.

Concentration Bounds The first step is to prove a concentration bound for the translation loss metric on each pair of languages. In this case, it will be easier to write the losses in terms of one single function: namely notice that in fact $\varepsilon(E_L, E'_L)$ only depends on $E_{L'}^{-1} \circ E_L$, and due to the group structure, $\mathcal{F} \ni f := E_{L'}^{-1} \circ E_L$. To that end, we will abuse the notation somewhat and denote $\varepsilon(f) := \varepsilon(E_L, E'_L)$. The following lemma is adapted from Bartlett (1998) where the bound is given in terms of binary classification error while here we present a bound using ℓ_2 loss. At a high level, the bound uses covering number to concentrate the empirical loss metric to its corresponding population counterpart.

Lemma 8.3.1. If $S = \{(x_i, x'_i)\}_{i=1}^n$ is sampled i.i.d. according to the encoder-decoder generative process, the following bound holds:

$$\Pr_{S \sim \mathcal{D}^n} \left(\sup_{f \in \mathcal{F}} |\varepsilon(f) - \widehat{\varepsilon}_S(f)| \geq \epsilon \right) \leq 2\mathcal{N}(\mathcal{F}, \frac{\epsilon}{16M}) \cdot \exp\left(\frac{-n\epsilon^2}{16M^4} \right).$$

This lemma can be proved using a ϵ-net argument with covering number. With this lemma, we can bound the error given by an empirical risk minimization algorithm:

Theorem 8.3.2. (Generalization, single task) Let S be a sample of size n according to our generative process. Then for any $0 < \delta < 1$, for any $f \in \mathcal{F}$, w.p. at least $1 - \delta$, the following bound holds:

$$\varepsilon(f) \leq \widehat{\varepsilon}_S(f) + O\left(\sqrt{\frac{\log \mathcal{N}(\mathcal{F}, \frac{\epsilon}{16M}) + \log(1/\delta)}{n}} \right). \tag{8.9}$$

Theorem 8.3.2 is a finite sample bound for generalization on a single pair of languages. This bound gives us an error measure on an edge in the translation graph in Fig. 8.3.2. Now, with an upper bound on the translation error of each *seen* language pair, we are ready to prove the main theorem (Theorem 8.3.1) which bounds the translation error for all possible pairs of translation tasks:

Proof of Theorem 8.3.1. First, under the assumption of Theorem 8.3.1, for any pair of language (L, L'), we know that the corpus contains at least $\Omega\left(\frac{1}{\epsilon_{L,L'}^2} \cdot \left(\log \mathcal{N}(\mathcal{F}, \frac{\epsilon_{L,L'}}{16M}) + \log(K/\delta) \right) \right)$ parallel sentences. Then by Theorem 8.3.2, with probability $1 - \delta$, for any L, L' connected by an edge in H, we have

$$\varepsilon(E_L, E_{L'}) \leq \widehat{\varepsilon}(E_L, E_{L'}) + \epsilon_{L,L'} \leq \epsilon_{L,L'} + \epsilon_{L,L'} = 2\epsilon_{L,L'},$$

where the last inequality is due to the assumption that $\widehat{\varepsilon}(E_L, E_{L'}) \leq \epsilon_{L,L'}$. Now consider any $L, L' \in \mathcal{L} \times \mathcal{L}$, connected by a path

$$L' = L_1, L_2, L_3, \dots, L_m = L$$

of length at most m. We will bound the error

$$\varepsilon(E_L, E_{L'}) = \|E_{L'}^{-1} \circ E_L - \mathbf{E}_{L'}^{-1} \circ \mathbf{E}_L\|_{\ell_2(\mathbf{D}_{L\sharp}\mathcal{D})}^2$$

by a judicious use of the triangle inequality. Namely, let's denote

$$\begin{aligned} I_1 &:= \mathbf{E}_{L_1}^{-1} \circ \mathbf{E}_{L_m}, \\ I_k &:= E_{L_1}^{-1} \circ E_{L_k} \circ \mathbf{E}_{L_k}^{-1} \circ \mathbf{E}_{L_m}, \qquad 2 \leq k \leq m-1, \\ I_m &:= E_{L_1}^{-1} \circ E_{L_m}. \end{aligned}$$

Then, we can write

$$\|E_{L'}^{-1} \circ E_L - \mathbf{E}_{L'}^{-1} \circ \mathbf{E}_L\|_{\ell_2(\mathbf{D}_{L\sharp}\mathcal{D})} = \|\sum_{k=1}^{m-1} I_k - I_{k+1}\|_{\ell_2(\mathbf{D}_{L\sharp}\mathcal{D})} \leq \sum_{k=1}^{m-1} \|I_k - I_{k+1}\|_{\ell_2(\mathbf{D}_{L\sharp}\mathcal{D})}. \tag{8.10}$$

Furthermore, notice that we can rewrite $I_k - I_{k+1}$ as

$$E_{L_1}^{-1} \circ E_{L_k} \left(\mathbf{E}_{L_k}^{-1} \circ \mathbf{E}_{L_{k+1}} - E_{L_k}^{-1} \circ E_{L_{k+1}}\right) \mathbf{E}_{L_{k+1}}^{-1} \circ \mathbf{E}_{L_m}.$$

Given that $E_{L_1}^{-1}$ and E_{L_k} are ρ-Lipschitz we have

$$\begin{aligned}
\|I_k - I_{k+1}\|_{\ell_2(\mathbf{D}_{L\sharp}\mathcal{D})} &= \left\|E_{L_1}^{-1} \circ E_{L_k} \left(E_{L_k}^{-1} \circ E_{L_{k+1}} - \mathbf{E}_{L_k}^{-1} \circ \mathbf{E}_{L_{k+1}}\right)\right\|_{\ell_2(\mathbf{D}_{L_{k+1}\sharp}\mathcal{D})} \\
&\leq \rho^2 \left\|\left(E_{L_k}^{-1} \circ E_{L_{k+1}} - \mathbf{E}_{L_k}^{-1} \circ \mathbf{E}_{L_{k+1}}\right)\right\|_{\ell_2(\mathbf{D}_{L_{k+1}\sharp}\mathcal{D})} \\
&\leq 2\rho^2 \epsilon_{L_k, L_{k+1}},
\end{aligned}$$

where the first line is from the definition of pushforward distribution, the second line is due to the Lipschitzness of $\mathcal{F}$ and the last line follows since all (L_k, L_{k+1}) are edges in H. Plugging this into (8.10), we have

$$\|E_{L'}^{-1} \circ E_L - \mathbf{E}_{L'}^{-1} \circ \mathbf{E}_L\|_{\ell_2(\mathbf{D}_{L\sharp}\mathcal{D})} \leq 2\rho^2 \sum_{k=1}^{m} \epsilon_{L_k, L_{k+1}}.$$

To complete the proof, realize that we need the events $|\varepsilon(L_k, L_{k+1}) - \widehat{\varepsilon}(L_k, L_{k+1})| \leq \epsilon_{L_k, L_{k+1}}$ to hold simultaneously for all the edges in the graph H. Hence it suffices if we can use a union bound to bound the failing probability. To this end, for each edge, we amplify the success probability by choosing the failure probability to be δ/K^2, and we can then bound the overall failure probability as:

$$\begin{aligned}
&\Pr\left(\text{At least one edge in the graph } H \text{ fails to satisfy (8.9)}\right) \\
&\leq \sum_{(i,j)\in H} \Pr\left(|\varepsilon(L_i, L_j) - \widehat{\varepsilon}(L_i, L_j)| > \epsilon_{L_i, L_j}\right) \\
&\leq \sum_{(i,j)\in H} \delta/K^2 \\
&\leq \frac{K(K-1)}{2} \cdot \frac{\delta}{K^2} \\
&\leq \delta.
\end{aligned}$$

The first inequality above is due to the union bound, and the second one is from Theorem 8.3.2 by choosing the failing probability to be δ/K^2. ■

8.3.4 Extension to Randomized Encoders and Decoders

Our discussions so far on the sample complexity under the encoder-decoder generative process assume that the ground-truth encoders and decoders are *deterministic* and *bijective*. This might seem to be a quite restrictive assumption, but nevertheless our underlying proof strategy using transitions on the translation graph still works in more general settings. In this section we shall provide an extension of the previous deterministic encoder-decoder generative process to allow randomness in the generation process. Note that this extension simultaneously relaxes both the deterministic and bijective assumptions before.

As a first step of the extension, since there is not a notion of inverse function anymore in the randomized setting, we first define the ground-truth encoder-decoder pair $(\mathbf{E}_L, \mathbf{D}_L)$ for a language $L \in \mathcal{L}$.

Definition 8.3.2. Let $\mathcal{D}_r$ and $\mathcal{D}_{r'}$ be two distributions over random seeds r and r' respectively. A randomized decoder $\mathbf{D}_{L_i}$ is a deterministic function that maps a feature z along with a random seed r to a sentences in language L_i. Similarly, a randomized encoder $\mathbf{E}_{L_i}$ maps a sentence $x \in \Sigma^*_{L_i}$ and a random seed r' to a representation in $\mathcal{Z}$. $(\mathbf{E}_{L_i}, \mathbf{D}_{L_i})$ is called an encoder-decoder pair if it keeps the distribution $\mathcal{D}$ over $\mathcal{Z}$ invariant under the randomness of $\mathcal{D}_r$ and $\mathcal{D}_{r'}$:

$$\mathbf{E}_{L_i \sharp}(\mathbf{D}_{L_i \sharp}(\mathcal{D} \times \mathcal{D}_r) \times \mathcal{D}_{r'}) = \mathcal{D}, \tag{8.11}$$

where we use $\mathcal{D} \times \mathcal{D}'$ to denote the product measure of distributions $\mathcal{D}$ and $\mathcal{D}'$.

Just like the deterministic setting, here we still assume that $\mathbf{E}_{L_i}, \mathbf{D}_{L_i} \in \mathcal{F}$ where $\mathcal{F}$ is closed under function composition. Furthermore, in order to satisfy Definition 8.3.2, we assume that $\forall\ \mathbf{D}_{L_i} \in \mathcal{F}$, there exists a corresponding $\mathbf{E}_{L_i} \in \mathcal{F}$, such that $(\mathbf{E}_{L_i}, \mathbf{D}_{L_i})$ is an encoder-decoder pair that verifies Definition 8.3.2. It is clear that the deterministic encoder-decoder pair in Section 8.3.1 is a special case of that in Definition 8.3.2: in that case $\mathbf{D}_{L_i} = \mathbf{E}_{L_i}^{-1}$ so that $\mathbf{E}_{L_i} \circ \mathbf{D}_{L_i} = \mathrm{id}_{\mathcal{Z}}$, the identity map over feature space $\mathcal{Z}$. Furthermore there is no randomness from r and r', hence the invariant criterion becomes $\mathbf{E}_{L_i \sharp}\mathbf{D}_{L_i \sharp}\mathcal{D} = (\mathbf{E}_{L_i} \circ \mathbf{D}_{L_i})_\sharp \mathcal{D} = \mathrm{id}_{\mathcal{Z}\sharp}\mathcal{D} = \mathcal{D}$, which trivially holds.

The randomness mechanism in Definition 8.3.2 has several practical implementations in practice. For example, the denoising autoencoder (Vincent et al., 2008), the encoder part of the conditional generative adversarial network (Mirza and Osindero, 2014), etc. Again, in the randomized setting we still need to have an assumption on the structure of $\mathcal{F}$, but this time a relaxed one:

Assumption 8.3.2 (Smoothness and Boundedness)**.** $\mathcal{F}$ is bounded under the $\|\cdot\|_\infty$ norm, i.e., there exists $M > 0$, such that $\forall f \in \mathcal{F}$, $\|f\|_\infty \leq M$. Furthermore, there exists $0 \leq \rho < \infty$, such that for $\forall x, x' \in \mathbb{R}^d$, $\forall f \in \mathcal{F}$, $\|\mathbb{E}_{\mathcal{D}_r}[f(x,r) - f(x',r)]\|_2 \leq \rho \cdot \|x - x'\|_2$.

Correspondingly, we also need to slightly extend our loss metric under the randomized setting to the following:

$$\varepsilon(E_L, D_{L'}) := \mathbb{E}_{r,r'}\|D_{L'} \circ E_L - \mathbf{D}_{L'} \circ \mathbf{E}_L\|^2_{\ell_2(\mathbf{D}_{L\sharp}(\mathcal{D}\times\mathcal{D}_r))},$$

where the expectation is taken over the distributions over random seeds r and r'. The empirical error could be extended in a similar way by replacing the population expectation with the empirical expectation. With the above extended definitions, now we are ready to state the following generalization theorem under randomized setting:

Theorem 8.3.3. (Sample complexity under generative model, randomized setting) Suppose H is connected and the trained $\{E_L\}_{L\in\mathcal{L}}$ satisfy

$$\forall L, L' \in H : \widehat{\varepsilon}_S(E_L, D_{L'}) \leq \epsilon_{L,L'},$$

for $\epsilon_{L,L'} > 0$. Furthermore, for $0 < \delta < 1$ suppose the number of sentences for each aligned corpora for each training pair (L, L') is $\Omega\left(\frac{1}{\epsilon^2_{L,L'}} \cdot \left(\log \mathcal{N}(\mathcal{F}, \frac{\epsilon_{L,L'}}{16M}) + \log(K/\delta)\right)\right)$. Then, with probability $1 - \delta$, for any pair of languages $(L, L') \in \mathcal{L} \times \mathcal{L}$ and $L = L_1, L_2, \ldots, L_m = L'$ a path between L and L' in H, we have $\varepsilon(E_L, D_{L'}) \leq 2\rho^2 \sum_{k=1}^{m-1} \epsilon_{L_k, L_{k+1}}$.

We comment that Theorem 8.3.3 is completely parallel to Theorem 8.3.1, except that we use generalized definitions under the randomized setting instead. Hence all the discussions before on Theorem 8.3.1 also apply here.

8.4 Related Work

Multilingual Machine Translation Early studies on multilingual machine translation mostly focused on pivot methods (Cohn and Lapata, 2007; De Gispert and Marino; Och and Ney, 2001; Utiyama and

Isahara, 2007) that use one pivot language to connect the translation between ultimate source and target languages, and train two separate statistical translation models (Koehn et al., 2003) to perform source-to-pivot and pivot-to-target translations. Since the successful application of encoder-decoder architectures in sequential tasks (Sutskever et al., 2014), neural machine translation (Bahdanau et al., 2014; Wu et al., 2016) has made it feasible to jointly learn from parallel corpora in multiple language pairs, and perform translation to multiple languages by a single model. Existing studies have been proposed to explore different variants of encoder-decoder architectures by using separate encoders (decoders) for multiple source (target) languages (Dong et al., 2015; Firat et al., 2016a,b; Platanios et al., 2018; Zoph and Knight, 2016) or sharing the weight of a single encoder (decoder) for all source (target) languages (Ha et al., 2016; Johnson et al., 2017). Recent advances of universal neural machine translation have also been applied to improve low-resource machine translation (Aharoni et al., 2019; Arivazhagan et al., 2019; Gu et al., 2018; Neubig and Hu, 2018) and downstream NLP tasks (Artetxe and Schwenk, 2019; Schwenk and Douze, 2017). Despite the recent empirical success in the literature, theoretical understanding is only nascent. Our work takes a first step towards better understanding the limitation of existing approaches and proposes a sufficient generative assumption that guarantees the success of universal machine translation.

Invariant Representations The line of work on seeking a shared multilingual embedding space started from learning cross-lingual word embeddings from parallel corpora (Gouws et al., 2015; Luong et al., 2015) or a bilingual dictionary (Artetxe et al., 2017; Conneau et al., 2018a; Faruqui and Dyer, 2014; Mikolov et al., 2013), and later extended to learning cross-lingual contextual representations (Conneau et al., 2019; Devlin et al., 2019; Huang et al., 2019; Lample and Conneau, 2019) from monolingual corpora. The idea of learning invariant representations is not unique in machine translation. In fact, similar ideas have already been used in other contexts, including domain adaptation (Combes et al., 2020; Ganin et al., 2016; Zhao et al., 2018b, 2019e), fair representations (Zemel et al., 2013; Zhang et al., 2018; Zhao and Gordon, 2019; Zhao et al., 2019c) and counterfactual reasoning in causal inference (Johansson et al., 2016; Shalit et al., 2017). Different from these existing work which mainly focuses on binary classification, our work provides the first impossibility theorem on learning language-invariant representations in terms of recovering a perfect translator under the setting of seq-to-seq learning.

8.5 Proofs

In this section we provide all the missing proofs in Section 8.3. Again, in what follows we will first restate the corresponding theorems for the ease of reading and then provide the detailed proofs.

Lemma 8.3.1. If $S = \{(x_i, x_i')\}_{i=1}^n$ is sampled i.i.d. according to the encoder-decoder generative process, the following bound holds:

$$\Pr_{S\sim\mathcal{D}^n}\left(\sup_{f\in\mathcal{F}} |\varepsilon(f) - \widehat{\varepsilon}_S(f)| \geq \epsilon\right) \leq 2\mathcal{N}(\mathcal{F}, \frac{\epsilon}{16M}) \cdot \exp\left(\frac{-n\epsilon^2}{16M^4}\right).$$

Proof. For $f \in \mathcal{F}$, define $\ell_S(f) := \varepsilon(f) - \widehat{\varepsilon}_S(f)$ to be the generalization error of f on sample S. The first step is to prove the following inequality holds for $\forall f_1, f_2 \in \mathcal{F}$ and any sample S:

$$|\ell_S(f_1) - \ell_S(f_2)| \leq 8M \cdot \|f_1 - f_2\|_\infty.$$

In other words, $\ell_S(\cdot)$ is a Lipschitz function in $\mathcal{F}$ w.r.t. the ℓ_∞ norm. To see, by definition of the generalization error, we have

$$\begin{aligned}
&|\ell_S(f_1) - \ell_S(f_2)| \\
&= |\varepsilon(f_1) - \widehat{\varepsilon}_S(f_1) - \varepsilon(f_2) + \widehat{\varepsilon}_S(f_2)| \\
&\le |\varepsilon(f_1) - \varepsilon(f_2)| + |\widehat{\varepsilon}_S(f_1) - \widehat{\varepsilon}_S(f_2)|.
\end{aligned}$$

To get the desired upper bound, it suffices for us to bound $|\varepsilon(f_1) - \varepsilon(f_2)|$ by $\|f_1 - f_2\|_\infty$ and the same technique could be used to upper bound $|\widehat{\varepsilon}_S(f_1) - \widehat{\varepsilon}_S(f_2)|$ since the only difference lies in the measure where the expectation is taken over. We now proceed to upper bound $|\varepsilon(f_1) - \varepsilon(f_2)|$:

$$\begin{aligned}
|\varepsilon(f_1) - \varepsilon(f_2)| &= \left|\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[\|f_1(\mathbf{x}) - \mathbf{x}'\|_2^2] - \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[\|f_2(\mathbf{x}) - \mathbf{x}'\|_2^2]\right| \\
&= \left|\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[\|f_1(\mathbf{x})\|_2^2 - \|f_2(\mathbf{x})\|_2^2 - 2\mathbf{x}'^T(f_1(\mathbf{x}) - f_2(\mathbf{x}))]\right| \\
&\le \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left|(f_1(\mathbf{x}) - f_2(\mathbf{x}))^T(f_1(\mathbf{x}) + f_2(\mathbf{x})) - 2\mathbf{x}'^T(f_1(\mathbf{x}) - f_2(\mathbf{x}))\right| \\
&\le \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\left|(f_1(\mathbf{x}) - f_2(\mathbf{x}))^T(f_1(\mathbf{x}) + f_2(\mathbf{x}))\right|\right] + 2\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\left|\mathbf{x}'^T(f_1(\mathbf{x}) - f_2(\mathbf{x}))\right|\right] \\
&\le \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\|f_1(\mathbf{x}) - f_2(\mathbf{x})\| \cdot \|f_1(\mathbf{x}) + f_2(\mathbf{x})\|\right] + 2\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\|\mathbf{x}'\| \cdot \|f_1(\mathbf{x}) - f_2(\mathbf{x})\|\right] \\
&\le 2M\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\|f_1(\mathbf{x}) - f_2(\mathbf{x})\|\right] + 2M\mathbb{E}_{\mathbf{z}\sim\mathcal{D}}\left[\|f_1(\mathbf{x}) - f_2(\mathbf{x})\|\right] \\
&\le 4M\|f_1 - f_2\|_\infty.
\end{aligned}$$

In the proof above, the first inequality holds due to the monotonicity property of integral. The second inequality holds by triangle inequality. The third one is due to Cauchy-Schwarz inequality. The fourth inequality holds by the assumption that $\forall f \in \mathcal{F}$, $\max_{x\in\mathcal{X}} \|f(\mathbf{x})\| \le M$ and the identity mapping is in $\mathcal{F}$ so that $\|\mathbf{x}'\| = \|\mathrm{id}(\mathbf{x}')\| \le \|\mathrm{id}(\cdot)\|_\infty \le M$. The last one holds due to the monotonicity property of integral.

It is easy to see that the same argument could also be used to show that $|\widehat{\varepsilon}_S(f_1) - \widehat{\varepsilon}_S(f_2)| \le 4M\|f_1 - f_2\|_\infty$. Combine these two inequalities, we have

$$\begin{aligned}
|\ell_S(f_1) - \ell_S(f_2)| &\le |\varepsilon(f_1) - \varepsilon(f_2)| + |\widehat{\varepsilon}_S(f_1) - \widehat{\varepsilon}_S(f_2)| \\
&\le 8M\|f_1 - f_2\|_\infty.
\end{aligned}$$

In the next step, we show that suppose $\mathcal{F}$ could be covered by k subsets $\mathcal{C}_1, \ldots, \mathcal{C}_k$, i.e., $\mathcal{F} = \cup_{i\in[k]}\mathcal{C}_i$. Then for any $\epsilon > 0$, the following upper bound holds:

$$\Pr_{S\sim\mathcal{D}^n}\big(\sup_{f\in\mathcal{F}} |\ell_S(f)| \ge \epsilon\big) \le \sum_{i\in[k]} \Pr_{S\sim\mathcal{D}^n}\big(\sup_{f\in\mathcal{C}_i} |\ell_S(f)| \ge \epsilon\big).$$

This follows from the union bound:

$$\begin{aligned}
\Pr_{S\sim\mathcal{D}^n}\big(\sup_{f\in\mathcal{F}} |\ell_S(f)| \ge \epsilon\big) &= \Pr_{S\sim\mathcal{D}^n}\big(\bigcup_{i\in[k]} \sup_{f\in\mathcal{C}_i} |\ell_S(f)| \ge \epsilon\big) \\
&\le \sum_{i\in[k]} \Pr_{S\sim\mathcal{D}^n}\big(\sup_{f\in\mathcal{C}_i} |\ell_S(f)| \ge \epsilon\big).
\end{aligned}$$

Next, within each L_∞ ball $\mathcal{C}_i$ centered at f_i with radius $\frac{\epsilon}{16M}$ such that $\mathcal{F} \subseteq \cup_{i\in[k]}\mathcal{C}_i$, we bound each term in the above union bound as:

$$\Pr_{S\sim\mathcal{D}^n}\big(\sup_{f\in\mathcal{C}_i} |\ell_S(f)| \ge \epsilon\big) \le \Pr_{S\sim\mathcal{D}^n}\big(|\ell_S(f_i)| \ge \epsilon/2\big).$$

To see this, realize that $\forall f \in \mathcal{C}_i$, we have $\|f - f_i\|_\infty \leq \epsilon/16M$, which implies

$$|\ell_S(f) - \ell_S(f_i)| \leq 8M\|f - f_i\|_\infty \leq \frac{\epsilon}{2}.$$

Hence we must have $|\ell_S(f_i)| \geq \epsilon/2$, otherwise $\sup_{f\in\mathcal{C}_i} |\ell_S(f)| < \epsilon$. This argument means that

$$\Pr_{S\sim\mathcal{D}^n}\big(\sup_{f\in\mathcal{C}_i} |\ell_S(f)| \geq \epsilon\big) \leq \Pr_{S\sim\mathcal{D}^n}\big(|\ell_S(f_i)| \geq \epsilon/2\big).$$

To finish the proof, we use the standard Hoeffding inequality to upper bound $\Pr_{S\sim\mathcal{D}^n}\big(|\ell_S(f_i)| \geq \epsilon/2\big)$ as follows:

$$\begin{aligned}\Pr_{S\sim\mathcal{D}^n}\big(|\ell_S(f_i)| \geq \epsilon/2\big) &= \Pr_{S\sim\mathcal{D}^n}\big(|\varepsilon(f_i) - \widehat{\varepsilon}_S(f_i)| \geq \epsilon/2\big) \\ &\leq 2\exp\left(-\frac{2n^2(\epsilon/2)^2}{n((2M)^2-0)^2}\right) \\ &= 2\exp\left(-\frac{n\epsilon^2}{16M^4}\right).\end{aligned}$$

Now combine everything together, we obtain the desired upper bound as stated in the lemma.

$$\Pr_{S\sim\mathcal{D}^n}\left(\sup_{f\in\mathcal{F}} |\varepsilon(f) - \widehat{\varepsilon}_S(f)| \geq \epsilon\right) \leq 2\mathcal{N}(\mathcal{F}, \frac{\epsilon}{16M}) \cdot \exp\left(\frac{-n\epsilon^2}{16M^4}\right). \qquad \blacksquare$$

We next prove the generalization bound for a single pair of translation task:

Theorem 8.3.2. (Generalization, single task) Let S be a sample of size n according to our generative process. Then for any $0 < \delta < 1$, for any $f \in \mathcal{F}$, w.p. at least $1 - \delta$, the following bound holds:

$$\varepsilon(f) \leq \widehat{\varepsilon}_S(f) + O\left(\sqrt{\frac{\log\mathcal{N}(\mathcal{F}, \frac{\epsilon}{16M}) + \log(1/\delta)}{n}}\right). \tag{8.9}$$

Proof. This is a direct corollary of Lemma 8.3.1 by setting the upper bound in Lemma 8.3.1 to be δ and solve for ϵ. $\blacksquare$

We now provide the proof sketch of Theorem 8.3.3. The main proof idea is exactly the same as the one we have in the deterministic setting, except that we replace the original definitions of errors and Lipschitzness with the generalized definitions under the randomized setting.

Theorem 8.3.3. (Sample complexity under generative model, randomized setting) Suppose H is connected and the trained $\{E_L\}_{L\in\mathcal{L}}$ satisfy

$$\forall L, L' \in H : \widehat{\varepsilon}_S(E_L, D_{L'}) \leq \epsilon_{L,L'},$$

for $\epsilon_{L,L'} > 0$. Furthermore, for $0 < \delta < 1$ suppose the number of sentences for each aligned corpora for each training pair (L, L') is $\Omega\left(\frac{1}{\epsilon^2_{L,L'}} \cdot \left(\log\mathcal{N}(\mathcal{F}, \frac{\epsilon_{L,L'}}{16M}) + \log(K/\delta)\right)\right)$. Then, with probability $1 - \delta$, for any pair of languages $(L, L') \in \mathcal{L} \times \mathcal{L}$ and $L = L_1, L_2, \ldots, L_m = L'$ a path between L and L' in H, we have $\varepsilon(E_L, D_{L'}) \leq 2\rho^2 \sum_{k=1}^{m-1} \epsilon_{L_k,L_{k+1}}$.

Proof Sketch. The first step is prove the corresponding error concentration lemma using covering numbers as the one in Lemma 8.3.1. Again, due to the assumption that $\mathcal{F}$ is closed under composition, we have $D_{L'} \circ E_L \in \mathcal{F}$, hence it suffices if we could prove a uniform convergence bound for an arbitrary function $f \in \mathcal{F}$. To this end, for $f \in \mathcal{F}$, define $\ell_S(f) := \varepsilon(f) - \widehat{\varepsilon}_S(f)$ to be the generalization error of f on sample S. The first step is to prove the following inequality holds for $\forall f_1, f_2 \in \mathcal{F}$ and any sample S:

$$|\ell_S(f_1) - \ell_S(f_2)| \leq 8M \cdot \|f_1 - f_2\|_\infty.$$

In other words, $\ell_S(\cdot)$ is a Lipschitz function in $\mathcal{F}$ w.r.t. the ℓ_∞ norm. To see this, by definition of the generalization error, we have

$$|\ell_S(f_1) - \ell_S(f_2)| = |\varepsilon(f_1) - \widehat{\varepsilon}_S(f_1) - \varepsilon(f_2) + \widehat{\varepsilon}_S(f_2)| \leq |\varepsilon(f_1) - \varepsilon(f_2)| + |\widehat{\varepsilon}_S(f_1) - \widehat{\varepsilon}_S(f_2)|.$$

To get the desired upper bound, it suffices for us to bound $|\varepsilon(f_1) - \varepsilon(f_2)|$ by $\|f_1 - f_2\|_\infty$ and the same technique could be used to upper bound $|\widehat{\varepsilon}_S(f_1) - \widehat{\varepsilon}_S(f_2)|$ since the only difference lies in the measure where the expectation is taken over.

Before we proceed, in order to make the notation uncluttered, we first simplify $\varepsilon(f)$:

$$\varepsilon(f) = \mathbb{E}_{r,r'}\left[\|f - \mathbf{D}_{L'} \circ \mathbf{E}_L\|^2_{\ell_2(\mathbf{D}_{L\sharp}(\mathcal{D}\times\mathcal{D}_r))}\right].$$

Define $\mathbf{z} \sim \mathcal{D}$ to mean the sampling process of $(x, r, r') \sim \mathbf{D}_{L\sharp}(\mathcal{D} \times \mathcal{D}_r) \times D_r \times D_{r'}$, $\mathbf{x} := (x, r, r')$ and $\mathbf{x}' := \mathbf{D}_{L'}(\mathbf{E}_L(x, r'), r)$. Then

$$\begin{aligned}\varepsilon(f) &= \mathbb{E}_{r,r'}\left[\|f - \mathbf{D}_{L'} \circ \mathbf{E}_L\|^2_{\ell_2(\mathbf{D}_{L\sharp}(\mathcal{D}\times\mathcal{D}_r))}\right] \\ &= \mathbb{E}_{\mathbf{z}\sim\mathcal{D}}[\|f(\mathbf{x}) - \mathbf{x}'\|_2^2].\end{aligned}$$

With the simplified notation, it is now clear that we essentially reduce the problem in the randomized setting to the original one in the deterministic setting. Hence by using exactly the same proof as the one of Lemma 8.3.1, we can obtain the following high probability bound:

$$\Pr\left(\sup_{f\in\mathcal{F}} |\varepsilon(f) - \widehat{\varepsilon}_S(f)| \geq \epsilon\right) \leq 2\mathcal{N}(\mathcal{F}, \frac{\epsilon}{16M}) \cdot \exp\left(\frac{-n\epsilon^2}{16M^4}\right).$$

As a direct corollary, a similar generalization bound for a single pair of translation task like the one in Theorem 8.3.2 also holds. To finish the proof, by the linearity of the expectation $\mathbb{E}_{r,r'}$, it is clear that exactly the same chaining argument in the proof of Theorem 8.3.1 could be used as well as the only thing we need to do is to take an additional expectation $\mathbb{E}_{r,r'}$ at the most outside level. ■

8.6 Conclusion

In this chapter we provided the first theoretical study on using language-invariant representations for universal machine translation. Our results are two-fold. First, we showed that without appropriate assumption on the generative structure of languages, there is an inherent tradeoff between learning language-invariant representations versus achieving good translation performance jointly in general. In particular, our results show that if the distributions (language models) of the target language differ between different translation pairs, then any machine translation method based on learning language-invariant representations is bound to achieve a large error on at least one of the translation tasks, even with unbounded computational resources.

On the positive side, we also show that, under appropriate generative model assumption of languages, e.g., a typical encoder-decoder model, it is not only possible to recover the ground-truth translator between any pair of languages that appear in the parallel corpora, but also we can hope to achieve a small translation error on sentences from unseen pair of languages, as long as they are connected in the so-called translation graph. This result holds in both deterministic and randomized settings. In addition, our result also characterizes how the relationship (distance) between these two languages in the graph affects the quality of translation in an intuitive manner: a graph with long connections results in a poorer translation.

Part II

Learning Tractable Circuits for Probabilistic Reasoning

Chapter 9

A Unified Framework for Parameter Learning

In this chapter we present a unified approach for learning the parameters of Sum-Product networks (SPNs). We prove that any complete and decomposable SPN is equivalent to a mixture of trees where each tree corresponds to a product of univariate distributions. Based on the mixture model perspective, we characterize the objective function when learning SPNs based on the maximum likelihood estimation (MLE) principle and show that the optimization problem can be formulated as a signomial program. We construct two parameter learning algorithms for SPNs by using sequential monomial approximations (SMA) and the concave-convex procedure (CCCP), respectively. The two proposed methods naturally admit multiplicative updates, hence effectively avoiding the projection operation. With the help of the unified framework, we also show that, in the case of SPNs, CCCP leads to the same algorithm as Expectation Maximization (EM) despite the fact that they are different in general.

9.1 Sum-Product Networks as a Mixture of Trees

We introduce the notion of *induced trees* from SPNs and use it to show that every complete and decomposable SPN can be interpreted as a mixture of induced trees, where each induced tree corresponds to a product of univariate distributions. From this perspective, an SPN can be understood as a huge mixture model where the effective number of components in the mixture is determined by its network structure. The method we describe here is not the first method for interpreting an SPN (or the related arithmetic circuit) as a mixture distribution Chan and Darwiche; Dennis and Ventura (2015); Zhao et al. (2015b); but, the new method can result in an exponentially smaller mixture, see the end of this section for more details.

Definition 9.1.1 (Induced Sum-Product Network)**.** Given a complete and decomposable SPN $\mathcal{S}$ over $\mathbf{X}_{[n]}$, let $\mathcal{T} = (\mathcal{T}_V, \mathcal{T}_E)$ be a subgraph of $\mathcal{S}$. $\mathcal{T}$ is called an *induced SPN* from $\mathcal{S}$ if

1. $Root(\mathcal{S}) \in \mathcal{T}_V$.
2. If $v \in \mathcal{T}_V$ is a sum node, then exactly one child of v in $\mathcal{S}$ is in $\mathcal{T}_V$, and the corresponding edge is in $\mathcal{T}_E$.
3. If $v \in \mathcal{T}_V$ is a product node, then all the children of v in $\mathcal{S}$ are in $\mathcal{T}_V$, and the corresponding edges are in $\mathcal{T}_E$.

For notational convenience we will call $\mathcal{T}$ an induced SPN by omitting the fact that $\mathcal{T}$ is induced from $\mathcal{S}$ if there is no confusion in the context.

Theorem 9.1.1. If $\mathcal{T}$ is an induced SPN from a complete and decomposable SPN $\mathcal{S}$, then $\mathcal{T}$ is a tree that is complete and decomposable.

As a result of Thm. 9.1.1, we will use the terms *induced SPNs* and *induced trees* interchangeably. With some abuse of notation, we use $\mathcal{T}(\mathbf{x})$ to mean the value of the network polynomial of $\mathcal{T}$ with input vector $\mathbf{x}$.

Theorem 9.1.2. If $\mathcal{T}$ is an induced tree from $\mathcal{S}$ over $\mathbf{X}_{[n]}$, then $\mathcal{T}(\mathbf{x}) = \prod_{(v_i,v_j)\in\mathcal{T}_E} w_{ij} \prod_{i=1}^{n} \mathbb{I}_{x_i}$, where w_{ij} is the edge weight of (v_i, v_j) if v_i is a sum node and $w_{ij} = 1$ if v_i is a product node.

Remark. Although we focus our attention on Boolean random variables for the simplicity of discussion and illustration, Thm. 9.1.2 can be extended to the case where the univariate distributions at the leaf nodes are continuous or discrete distributions with countably infinitely many values, e.g., Gaussian distributions or Poisson distributions. We can simply replace the product of univariate distributions term, $\prod_{i=1}^{n} \mathbb{I}_{x_i}$, in Thm. 9.1.2 to be the general form $\prod_{i=1}^{n} \Pr_i(X_i)$, where $\Pr_i(X_i)$ is a univariate distribution over X_i. Also note that it is possible for two unique induced trees to share the same product of univariate distributions, but in this case their weight terms $\prod_{(v_i,v_i)\in\mathcal{T}_E} w_{ij}$ are guaranteed to be different. As we will see shortly, Thm. 9.1.2 implies that the joint distribution over $\{X_i\}_{i=1}^{n}$ represented by an SPN is essentially a mixture model with potentially exponentially many components in the mixture.

Definition 9.1.2 (Network cardinality)**.** The network cardinality $\tau_{\mathcal{S}}$ of an SPN $\mathcal{S}$ is the number of unique induced trees.

Theorem 9.1.3. $\tau_{\mathcal{S}} = V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$, where $V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$ is the value of the network polynomial of $\mathcal{S}$ with input vector $\mathbf{1}$ and all edge weights set to be 1.

Theorem 9.1.4. $\mathcal{S}(\mathbf{x}) = \sum_{t=1}^{\tau_{\mathcal{S}}} \mathcal{T}_t(\mathbf{x})$, where $\mathcal{T}_t$ is the tth unique induced tree of $\mathcal{S}$.

Remark. The above four theorems prove the fact that an SPN $\mathcal{S}$ is an ensemble or mixture of trees, where each tree computes an unnormalized distribution over $\mathbf{X}_{[n]}$. The total number of unique trees in $\mathcal{S}$ is the network cardinality $\tau_{\mathcal{S}}$, which only depends on the structure of $\mathcal{S}$. Each component is a simple product of univariate distributions. We illustrate the theorems above with a simple example in Fig. 9.1.1.

Zhao et al. (2015b) show that every complete and decomposable SPN is equivalent to a bipartite Bayesian network with a layer of hidden variables and a layer of observable random variables. The number

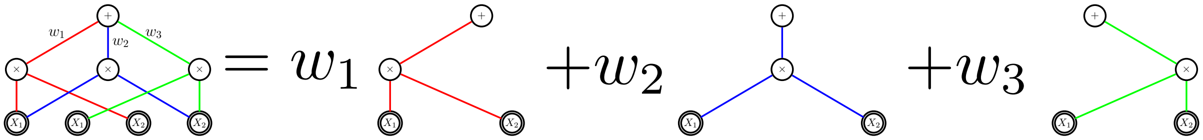


Figure 9.1.1: A complete and decomposable SPN is a mixture of induced trees. Double circles indicate univariate distributions over X_1 and X_2. Different colors are used to highlight unique induced trees; each induced tree is a product of univariate distributions over X_1 and X_2.

of hidden variables in the bipartite Bayesian network is equal to the number of sum nodes in $\mathcal{S}$. A naive expansion of such Bayesian network to a mixture model will lead to a huge mixture model with $2^{O(|\$|)}$ components, where $|\$|$ is the number of sum nodes in $\mathcal{S}$. Here we complement the theory and show that each complete and decomposable SPN is essentially a mixture of trees and the effective number of unique induced trees is given by $\tau_{\mathcal{S}}$. Note that $\tau_{\mathcal{S}} = V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$ depends only on the network structure, and can often be much smaller than $2^{O(|\$|)}$. Without loss of generality, assuming that in $\mathcal{S}$ layers of sum nodes are alternating with layers of product nodes, then $V_{\text{root}}(\mathbf{1} \mid \mathbf{1}) = \Omega(2^h)$, where h is the height of $\mathcal{S}$. However, the exponentially many trees are recursively merged and combined in $\mathcal{S}$ such that the overall network size is still tractable.

9.2 Maximum Likelihood Estimation as Signomial Programming

Let's consider the likelihood function computed by an SPN $\mathcal{S}$ over n binary random variables with model parameters $\mathbf{w}$ and input vector $\mathbf{x} \in \{0,1\}^n$. Here the model parameters in $\mathcal{S}$ are edge weights from every sum node, and we collect them together into a long vector $\mathbf{w} \in \mathbb{R}^p_{++}$, where p corresponds to the number of edges emanating from sum nodes in $\mathcal{S}$. By definition, the probability distribution induced by $\mathcal{S}$ can be computed by

$$\Pr_{\mathcal{S}}(\mathbf{x} \mid \mathbf{w}) := \frac{V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{\sum_{\mathbf{x}} V_{\text{root}}(\mathbf{x} \mid \mathbf{w})} = \frac{V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{V_{\text{root}}(\mathbf{1} \mid \mathbf{w})}$$

Corollary 9.2.1. Let $\mathcal{S}$ be an SPN with weights $\mathbf{w} \in \mathbb{R}^p_{++}$ over input vector $\mathbf{x} \in \{0,1\}^n$, the network polynomial $V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$ is a posynomial: $V_{\text{root}}(\mathbf{x} \mid \mathbf{w}) = \sum_{t=1}^{V_{\text{root}}(\mathbf{1}|\mathbf{1})} \prod_{i=1}^n \mathbb{I}_{x_i}^{(t)} \prod_{d=1}^p w_d^{\mathbb{I}_{w_d \in \mathcal{T}_t}}$, where $\mathbb{I}_{w_d \in \mathcal{T}_t}$ is the indicator variable whether w_d is in the t-th induced tree $\mathcal{T}_t$ or not. Each monomial corresponds exactly to a unique induced tree SPN from $\mathcal{S}$.

The above statement is a direct corollary of Thm. 9.1.2, Thm. 9.1.3 and Thm. 9.1.4. From the definition of network polynomial, we know that $V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$ is a multilinear function of the indicator variables. Corollary 9.2.1 works as a complement to characterize the functional form of a network polynomial in terms of $\mathbf{w}$. It follows that the likelihood function $\mathcal{L}_{\mathcal{S}}(\mathbf{w}) := \Pr_{\mathcal{S}}(\mathbf{x} \mid \mathbf{w})$ can be expressed as the ratio of two posynomial functions. We now show that the optimization problem based on MLE is an SP. Using the definition of $\Pr_{\mathcal{S}}(\mathbf{x} \mid \mathbf{w})$ and Corollary 9.2.1, let $\tau = V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$, the MLE problem can be rewritten as:

$$\begin{aligned} \text{maximize}_{\mathbf{w}} \quad & \frac{V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{V_{\text{root}}(\mathbf{1} \mid \mathbf{w})} = \frac{\sum_{t=1}^{\tau} \prod_{i=1}^n \mathbb{I}_{x_i}^{(t)} \prod_{d=1}^p w_d^{\mathbb{I}_{w_d \in \mathcal{T}_t}}}{\sum_{t=1}^{\tau} \prod_{d=1}^p w_d^{\mathbb{I}_{w_d \in \mathcal{T}_t}}} \\ \text{subject to} \quad & \mathbf{w} \in \mathbb{R}^p_{++} \end{aligned} \tag{9.1}$$

Proposition 9.2.1. The MLE problem for SPNs is a signomial program.

Being nonconvex in general, SP is essentially hard to solve from a computational perspective Boyd et al. (2007); Chiang (2005). However, despite the hardness of SP in general, the objective function in the MLE formulation of SPNs has a special structure, i.e., it is the ratio of two posynomials, which makes the design of efficient optimization algorithms possible.

9.3 Difference of Convex Functions

Both projected gradient descent (PGD) and exponentiated gradient (EG) are first-order methods and they can be viewed as approximating the SP after applying a logarithmic transformation to the objective function only. Although ((9.1)) is a signomial program, its objective function is expressed as the ratio of two posynomials. Hence, we can still apply the *logarithmic transformation trick* used in geometric programming to its objective function and to the variables to be optimized. More concretely, let $w_d = \exp(y_d), \forall d \in [p]$ and take the log of the objective function; it becomes equivalent to maximize the following new objective without any constraint on $\mathbf{y}$:

$$\text{maximize} \quad \log\left(\sum_{t=1}^{\tau(\mathbf{x})} \exp\left(\sum_{d=1}^{p} y_d \mathbb{I}_{y_d \in \mathcal{T}_t}\right)\right) - \log\left(\sum_{t=1}^{\tau} \exp\left(\sum_{d=1}^{p} y_d \mathbb{I}_{y_d \in \mathcal{T}_t}\right)\right) \tag{9.2}$$

Note that in the first term of Eq. (9.2) the upper index $\tau(\mathbf{x}) \leq \tau := V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$ depends on the current input $\mathbf{x}$. By transforming into the log-space, we naturally guarantee the positivity of the solution at each iteration, hence transforming a constrained optimization problem into an unconstrained optimization problem without any sacrifice. Both terms in Eq. (9.2) are convex functions in $\mathbf{y}$ after the transformation. Hence, the transformed objective function is now expressed as the difference of two convex functions, which is called a DC function (Hartman et al., 1959). This helps us to design two efficient algorithms to solve the problem based on the general idea of sequential convex approximations for nonlinear programming.

9.3.1 Sequential Monomial Approximation

Let's consider the linearization of both terms in Eq. (9.2) in order to apply first-order methods in the transformed space. To compute the gradient with respect to different components of $\mathbf{y}$, we view each node of an SPN as an intermediate function of the network polynomial and apply the chain rule to back-propagate the gradient.

The differentiation of $V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$ with respect to the root node of the network is set to be 1. The differentiation of the network polynomial with respect to a partial function at each node can then be computed in two passes of the network: the bottom-up pass evaluates the values of all partial functions given the current input $\mathbf{x}$ and the top-down pass differentiates the network polynomial with respect to each partial function. Since the model parameters $\mathbf{y}(\mathbf{w})$ are only associated with sum nodes, they can be easily computed once we have obtained the differentiations for each node as

$$\begin{aligned}\frac{\partial \log V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{\partial y_{ij}} &= \frac{\partial \log V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{\partial V_{v_i}(\mathbf{x} \mid \mathbf{w})} \frac{\partial V_{v_i}(\mathbf{x} \mid \mathbf{w})}{\partial w_{ij}} \frac{\partial w_{ij}}{\partial y_{ij}} \\ &= \frac{\partial \log V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{\partial V_{v_i}(\mathbf{x} \mid \mathbf{w})} V_{v_j}(\mathbf{x} \mid \mathbf{w}) w_{ij} \end{aligned} \tag{9.3}$$

where v_i is restricted to be a sum node and v_j is a child of v_i. Following the evaluation-differentiation passes, the gradient of the objective function in ((9.2)) can be computed in $O(|\mathcal{S}|)$. Furthermore, although the computation is conducted in $\mathbf{y}$, the results are fully expressed in terms of $\mathbf{w}$, which suggests that in practice we do not need to explicitly construct $\mathbf{y}$ from $\mathbf{w}$. An illustration of the process is provided in Fig. 9.3.1.

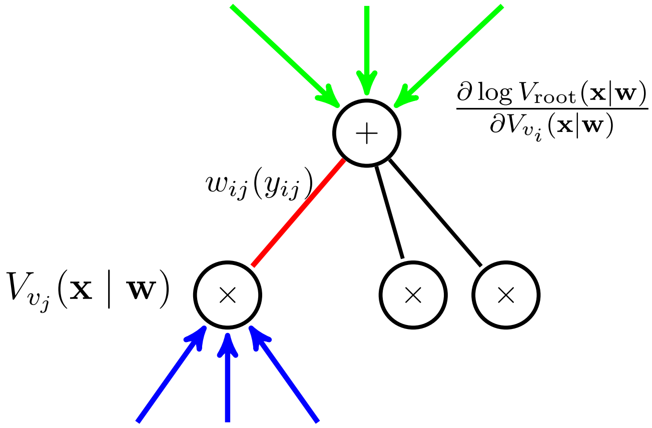

$\frac{\partial \log V_{\mathrm{root}}(\mathbf{x}|\mathbf{w})}{\partial V_{v_i}(\mathbf{x}|\mathbf{w})}$
+
$w_{ij}(y_{ij})$
$V_{v_j}(\mathbf{x} \mid \mathbf{w})$
×
×
×

Figure 9.3.1: Information flow about the computation of the gradient of log-network polynomial with respect to $y_{ij}(w_{ij})$. Each edge $y_{ij}(w_{ij})$ collects the evaluation value from v_j in bottom-up pass and also differentiation value from v_i in top-down pass.

Let $f(\mathbf{y}) := \log V_{\mathrm{root}}(\mathbf{x} \mid \exp(\mathbf{y})) - \log V_{\mathrm{root}}(\mathbf{1} \mid \exp(\mathbf{y}))$. Consider the optimal first-order approximation of $f(\mathbf{y})$ at point $\mathbf{y}^{(k)}$:

$$\hat{f}(\mathbf{y}) := f(\mathbf{y}^{(k)}) + \nabla_{\mathbf{y}} f(\mathbf{y}^{(k)})^T (\mathbf{y} - \mathbf{y}^{(k)}) \tag{9.4}$$

which is equivalent to $\exp(\hat{f}(\mathbf{w})) = C_1 \prod_{d=1}^{p} w_d^{\nabla_{y_d} f(\mathbf{y}^{(k)})}$ in the original space, where C_1 is a positive constant w.r.t. $\mathbf{y}(\mathbf{w})$. It follows that approximating $f(\mathbf{y})$ with the best linear function is equivalent to using the best monomial approximation of the signomial program ((9.1)). This leads to a sequential monomial approximations of the original SP formulation: at each iteration $\mathbf{y}^{(k)}$, we linearize both terms in Eq. (9.2) and form the optimal monomial function in terms of $\mathbf{w}^{(k)}$. The additive update of $\mathbf{y}^{(k)}$ leads to a multiplicative update of $\mathbf{w}^{(k)}$ since $\mathbf{w}^{(k)} = \exp(\mathbf{y}^{(k)})$, and we use a backtracking line search to determine the step size of the update in each iteration.

9.3.2 Concave-convex Procedure

Sequential monomial approximation fails to use the structure of the problem when learning SPNs. Here we propose another approach based on the concave-convex procedure (CCCP) (Yuille et al., 2002) to use the fact that the objective function is expressed as the difference of two convex functions. At a high level CCCP solves a sequence of concave surrogate optimizations until convergence. In many cases, the maximum of a concave surrogate function can only be solved using other convex solvers and as a result the efficiency of the CCCP highly depends on the choice of the convex solvers. However, we show that by a suitable transformation of the network we can compute the maximum of the concave surrogate in closed form in time that is linear in the network size, which leads to a very efficient algorithm for learning the parameters of SPNs. We also prove the convergence properties of our algorithm.

Consider the objective function to be maximized in DCP: $f(\mathbf{y}) = \log V_{\text{root}}(\mathbf{x} \mid \exp(\mathbf{y})) - \log V_{\text{root}}(\mathbf{1} \mid \exp(\mathbf{y})) := f_1(\mathbf{y}) + f_2(\mathbf{y})$ where $f_1(\mathbf{y}) := \log V_{\text{root}}(\mathbf{x} \mid \exp(\mathbf{y}))$ is a convex function and $f_2(\mathbf{y}) := -\log V_{\text{root}}(\mathbf{1} \mid \exp(\mathbf{y}))$ is a concave function. We can linearize only the convex part $f_1(\mathbf{y})$ to obtain a surrogate function

$$\hat{f}(\mathbf{y}, \mathbf{z}) = f_1(\mathbf{z}) + \nabla_{\mathbf{z}} f_1(\mathbf{z})^T(\mathbf{y} - \mathbf{z}) + f_2(\mathbf{y}) \tag{9.5}$$

for $\forall \mathbf{y}, \mathbf{z} \in \mathbb{R}^p$. Now $\hat{f}(\mathbf{y}, \mathbf{z})$ is a concave function in $\mathbf{y}$. Due to the convexity of $f_1(\mathbf{y})$ we have

$$f_1(\mathbf{y}) \geq f_1(\mathbf{z}) + \nabla_{\mathbf{z}} f_1(\mathbf{z})^T(\mathbf{y} - \mathbf{z}), \quad \forall \mathbf{y}, \mathbf{z}$$

and as a result the following two properties always hold for $\forall \mathbf{y}, \mathbf{z}$:

$$\hat{f}(\mathbf{y}, \mathbf{z}) \leq f(\mathbf{y}) \quad \text{and} \quad \hat{f}(\mathbf{y}, \mathbf{y}) = f(\mathbf{y}) \tag{9.6}$$

CCCP updates $\mathbf{y}$ at each iteration k by solving $\mathbf{y}^{(k)} \in \arg\max_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$ unless we already have $\mathbf{y}^{(k-1)} \in \arg\max_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$, in which case a generalized fixed point $\mathbf{y}^{(k-1)}$ has been found and the algorithm stops.

It is easy to show that at each iteration of CCCP we always have $f(\mathbf{y}^{(k)}) \geq f(\mathbf{y}^{(k-1)})$. Note also that $f(\mathbf{y})$ is computing the log-likelihood of input $\mathbf{x}$ and therefore it is bounded above by 0. By the monotone convergence theorem, $\lim_{k\to\infty} f(\mathbf{y}^{(k)})$ exists and the sequence $\{f(\mathbf{y}^{(k)})\}$ converges.

We now discuss how to compute a closed form solution for the maximization of the concave surrogate $\hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$. Since $\hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$ is differentiable and concave for any fixed $\mathbf{y}^{(k-1)}$, a sufficient and necessary condition to find its maximum is

$$\nabla_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)}) = \nabla_{\mathbf{y}^{(k-1)}} f_1(\mathbf{y}^{(k-1)}) + \nabla_{\mathbf{y}} f_2(\mathbf{y}) = 0 \tag{9.7}$$

In the above equation, if we consider only the partial derivative with respect to $y_{ij}(w_{ij})$, we obtain

$$\frac{w_{ij}^{(k-1)} V_{v_j}(\mathbf{x} \mid \mathbf{w}^{(k-1)})}{V_{\text{root}}(\mathbf{x} \mid \mathbf{w}^{(k-1)})} \frac{\partial V_{\text{root}}(\mathbf{x} \mid \mathbf{w}^{(k-1)})}{\partial V_{v_i}(\mathbf{x} \mid \mathbf{w}^{(k-1)})} = \frac{w_{ij} V_{v_j}(\mathbf{1} \mid \mathbf{w})}{V_{\text{root}}(\mathbf{1} \mid \mathbf{w})} \frac{\partial V_{\text{root}}(\mathbf{1} \mid \mathbf{w})}{\partial V_{v_i}(\mathbf{1} \mid \mathbf{w})} \tag{9.8}$$

Eq. (9.8) leads to a system of p nonlinear equations, which is hard to solve in closed form. However, if we do a change of variable by considering locally normalized weights w'_{ij} (i.e., $w'_{ij} \geq 0$ and $\sum_j w'_{ij} = 1\ \forall i \in \mathbb{S}$), then a solution can be easily computed. As described in (Peharz et al., 2015; Zhao et al., 2015b), any SPN can be transformed into an equivalent *normal* SPN with locally normalized weights in a bottom up pass as follows:

$$w'_{ij} = \frac{w_{ij} V_{v_j}(\mathbf{1} \mid \mathbf{w})}{\sum_j w_{ij} V_{v_j}(\mathbf{1} \mid \mathbf{w})} \tag{9.9}$$

We can then replace $w_{ij} V_{v_j}(\mathbf{1} \mid \mathbf{w})$ in the above equation by the expression it is equal to in Eq. (9.8) to obtain a closed form solution:

$$w'_{ij} \propto w_{ij}^{(k-1)} \frac{V_{v_j}(\mathbf{x} \mid \mathbf{w}^{(k-1)})}{V_{\text{root}}(\mathbf{x} \mid \mathbf{w}^{(k-1)})} \frac{\partial V_{\text{root}}(\mathbf{x} \mid \mathbf{w}^{(k-1)})}{\partial V_{v_i}(\mathbf{x} \mid \mathbf{w}^{(k-1)})} \tag{9.10}$$

Note that in the above derivation both $V_{v_i}(\mathbf{1} \mid \mathbf{w}) / V_{\text{root}}(\mathbf{1} \mid \mathbf{w})$ and $\partial V_{\text{root}}(\mathbf{1} \mid \mathbf{w}) / \partial V_{v_i}(\mathbf{1} \mid \mathbf{w})$ can be treated as constants and hence absorbed since $w'_{ij}, \forall j$ are constrained to be locally normalized. In order to obtain a solution to Eq. (9.8), for each edge weight w_{ij}, the sufficient statistics include only three terms, i.e, the evaluation value at v_j, the differentiation value at v_i and the previous edge weight $w_{ij}^{(k-1)}$, all of which

Table 9.3.1: Summary of PGD, EG, SMA and CCCP. Var. means the optimization variables.

Algo	**Var.**	**Update Type**	**Update Formula**
PGD	$\mathbf{w}$	Additive	$w_d^{(k+1)} \leftarrow P_{\mathbb{R}^{\epsilon}_{++}} \left\{ w_d^{(k)} + \gamma(\nabla_{w_d} f_1(\mathbf{w}^{(k)}) - \nabla_{w_d} f_2(\mathbf{w}^{(k)})) \right\}$
EG	$\mathbf{w}$	Multiplicative	$w_d^{(k+1)} \leftarrow w_d^{(k)} \exp\{\gamma(\nabla_{w_d} f_1(\mathbf{w}^{(k)}) - \nabla_{w_d} f_2(\mathbf{w}^{(k)}))\}$
SMA	$\log \mathbf{w}$	Multiplicative	$w_d^{(k+1)} \leftarrow w_d^{(k)} \exp\{\gamma w_d^{(k)} \times (\nabla_{w_d} f_1(\mathbf{w}^{(k)}) - \nabla_{w_d} f_2(\mathbf{w}^{(k)}))\}$
CCCP	$\log \mathbf{w}$	Multiplicative	$w_{ij}^{(k+1)} \propto w_{ij}^{(k)} \times \nabla_{v_i} V_{\text{root}}(\mathbf{w}^{(k)}) \times V_{v_j}(\mathbf{w}^{(k)})$

can be obtained in two passes of the network for each input $\mathbf{x}$. Thus the computational complexity to obtain a maximum of the concave surrogate is $O(|\mathcal{S}|)$. Interestingly, Eq. (9.10) leads to the same update formula as in the EM algorithm Peharz (2015) despite the fact that CCCP and EM start from different perspectives. We show that all the limit points of the sequence $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$ are guaranteed to be stationary points of DCP in (9.2).

Theorem 9.3.1. Let $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$ be any sequence generated using Eq. (9.10) from any positive initial point, then all the limiting points of $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$ are stationary points of the DCP in (9.2). In addition, $\lim_{k\to\infty} f(\mathbf{y}^{(k)}) = f(\mathbf{y}^*)$, where $\mathbf{y}^*$ is some stationary point of (9.2).

We summarize all four algorithms and highlight their connections and differences in Table 9.3.1. Although we mainly discuss the batch version of those algorithms, all of the four algorithms can be easily adapted to work in stochastic and/or parallel settings.

9.4 Experiments

9.4.1 Experimental Setting

We conduct experiments on 20 benchmark data sets from various domains to compare and evaluate the convergence performance of the four algorithms: PGD, EG, SMA and CCCP (EM). We list here the detailed statistics of the 20 data sets used in the experiments in Table 10.4.1. All the features in the 20 data sets are binary features. All the SPNs that are used for comparisons of PGD, EG, SMA and CCCP are trained using LearnSPN (Gens and Domingos, 2013). We discard the weights returned by LearnSPN and use random weights as initial model parameters. The random weights are determined by the same random seed in all four algorithms. The sizes of different SPNs produced by LearnSPN and ID-SPN are shown in Table 9.4.2.

9.4.2 Parameter Learning

We implement all four algorithms in C++. For each algorithm, we set the maximum number of iterations to 50. If the absolute difference in the training log-likelihood at two consecutive steps is less than 0.001, the algorithms are stopped. For PGD, EG and SMA, we combine each of them with backtracking line search and use a weight shrinking coefficient set at 0.8. The learning rates are initialized to 1.0 for all three methods. For PGD, we set the projection margin ϵ to 0.01. There is no learning rate and no backtracking line search in CCCP. We set the smoothing parameter to 0.001 in CCCP to avoid numerical issues.

We show in Fig. 9.4.1 the average log-likelihood scores on 20 training data sets to evaluate the convergence speed and stability of PGD, EG, SMA and CCCP. Clearly, CCCP wins by a large margin

Table 9.4.1: Statistics of data sets and models. N is the number of variables modeled by the network, $|\mathcal{S}|$ is the size of the network and D is the number of parameters to be estimated in the network. $N \times V/D$ means the ratio of training instances times the number of variables to the number parameters.

Data set	N	$\|\mathcal{S}\|$	D	Train	Valid	Test	$N \times V/D$
NLTCS	16	13,733	1,716	16,181	2,157	3,236	150.871
MSNBC	17	54,839	24,452	291,326	38,843	58,265	202.541
KDD 2k	64	48,279	14,292	180,092	19,907	34,955	806.457
Plants	69	132,959	58,853	17,412	2,321	3,482	20.414
Audio	100	739,525	196,103	15,000	2,000	3,000	7.649
Jester	100	314,013	180,750	9,000	1,000	4,116	4.979
Netflix	100	161,655	51,601	15,000	2,000	3,000	29.069
Accidents	111	204,501	74,804	12,758	1,700	2,551	18.931
Retail	135	56,931	22,113	22,041	2,938	4,408	134.560
Pumsb-star	163	140,339	63,173	12,262	1,635	2,452	31.638
DNA	180	108,021	52,121	1,600	400	1,186	5.526
Kosarak	190	203,321	53,204	33,375	4,450	6,675	119.187
MSWeb	294	68,853	20,346	29,441	3,270	5,000	425.423
Book	500	190,625	41,122	8,700	1,159	1,739	105.783
EachMovie	500	522,753	188,387	4,524	1,002	591	12.007
WebKB	839	1,439,751	879,893	2,803	558	838	2.673
Reuters-52	889	2,210,325	1,453,390	6,532	1,028	1,540	3.995
20 Newsgrp	910	14,561,965	8,295,407	11,293	3,764	3,764	1.239
BBC	1058	1,879,921	1,222,536	1,670	225	330	1.445
Ad	1556	4,133,421	1,380,676	2,461	327	491	2.774

over PGD, EG and SMA, both in convergence speed and solution quality. Furthermore, among the four algorithms, CCCP is the most stable one due to its guarantee that the log-likelihood (on training data) will not decrease after each iteration. As shown in Fig. 9.4.1, the training curves of CCCP are more smooth than the other three methods in almost all the cases. These 20 experiments also clearly show that CCCP often converges in a few iterations. On the other hand, PGD, EG and SMA are on par with each other since they are all first-order methods. SMA is more stable than PGD and EG and often achieves better solutions than PGD and EG. On large data sets, SMA also converges faster than PGD and EG. Surprisingly, EG performs worse than PGD in some cases and is quite unstable despite the fact that it admits multiplicative updates. The "hook shape" curves of PGD in some data sets, e.g. Kosarak and KDD, are due to the projection operations.

The computational complexity per update is $O(|\mathcal{S}|)$ in all four algorithms. CCCP often takes less time than the other three algorithms because it takes fewer iterations to converge. We list detailed running time statistics for all four algorithms on the 20 data sets

9.4.3 Fine Tuning

We combine CCCP as a "fine tuning" procedure with the structure learning algorithm LearnSPN and compare it to the state-of-the-art structure learning algorithm ID-SPN (Rooshenas and Lowd, 2014). More concretely, we keep the model parameters learned from LearnSPN and use them to initialize CCCP. We then update the model parameters globally using CCCP as a fine tuning technique. This normally helps

Table 9.4.2: Sizes of SPNs produced by LearnSPN and ID-SPN.

Data set	LearnSPN	ID-SPN
NLTCS	13,733	24,690
MSNBC	54,839	579,364
KDD 2k	48,279	1,286,657
Plants	132,959	2,063,708
Audio	739,525	2,643,948
Jester	314,013	4,225,471
Netflix	161,655	7,958,088
Accidents	204,501	2,273,186
Retail	56,931	60,961
Pumsb-star	140,339	1,751,092
DNA	108,021	3,228,616
Kosarak	203,321	1,272,981
MSWeb	68,853	1,886,777
Book	190,625	1,445,501
EachMovie	522,753	2,440,864
WebKB	1,439,751	2,605,141
Reuters-52	2,210,325	4,563,861
20 Newsgrp	14,561,965	3,485,029
BBC	1,879,921	2,426,602
Ad	4,133,421	2,087,253

to obtain a better generative model since the original parameters are learned greedily and locally during the structure learning algorithm. We use the validation set log-likelihood score to avoid overfitting. The algorithm returns the set of parameters that achieve the best validation set log-likelihood score as output. Experimental results are reported in Table. 9.4.3. As shown in Table 9.4.3, the use of CCCP after LearnSPN always helps to improve the model performance. By optimizing model parameters on these 20 data sets, we boost LearnSPN to achieve better results than state-of-the-art ID-SPN on 7 data sets, where the original LearnSPN only outperforms ID-SPN on 1 data set. Note that the sizes of the SPNs returned by LearnSPN are much smaller than those produced by ID-SPN. Hence, it is remarkable that by fine tuning the parameters with CCCP, we can achieve better performance despite the fact that the models are smaller. For a fair comparison, we also list the size of the SPNs returned by ID-SPN in the supplementary material. As a result, we suggest using CCCP after structure learning algorithms to fully exploit the expressiveness of the constructed model.

9.5 Proofs

In this section we provide all the omitted proofs in Sec. 9.1.

9.5.1 Structural Properties of SPNs

Theorem 9.1.1. If $\mathcal{T}$ is an induced SPN from a complete and decomposable SPN $\mathcal{S}$, then $\mathcal{T}$ is a tree that is complete and decomposable.

Table 9.4.3: Average log-likelihoods on test data. Highest log-likelihoods are highlighted in bold. ↑ shows statistically better log-likelihoods than CCCP and ↓ shows statistically worse log-likelihoods than CCCP. The significance is measured based on the Wilcoxon signed-rank test.

Data set	CCCP	LearnSPN	ID-SPN	**Data set**	CCCP	LearnSPN	ID-SPN
NLTCS	**-6.029**	↓-6.099	↓-6.050	DNA	-84.921	↓-85.237	↑**-84.693**
MSNBC	**-6.045**	↓-6.113	-6.048	Kosarak	-10.880	↓-11.057	**-10.605**
KDD 2k	**-2.134**	↓-2.233	↓-2.153	MSWeb	-9.970	↓-10.269	**-9.800**
Plants	-12.872	↓-12.955	↑**-12.554**	Book	-35.009	↓-36.247	↑**-34.436**
Audio	-40.020	↓-40.510	**-39.824**	EachMovie	-52.557	↓-52.816	↑**-51.550**
Jester	**-52.880**	↓-53.454	↓-52.912	WebKB	-157.492	↓-158.542	↑**-153.293**
Netflix	-56.782	↓-57.385	↑**-56.554**	Reuters-52	-84.628	↓-85.979	↑**-84.389**
Accidents	-27.700	↓-29.907	↑**-27.232**	20 Newsgrp	-153.205	↓-156.605	↑**-151.666**
Retail	**-10.919**	↓-11.138	-10.945	BBC	**-248.602**	↓-249.794	↓-252.602
Pumsb-star	-24.229	↓-24.577	↑**-22.552**	Ad	**-27.202**	↓-27.409	↓-40.012

Proof. Argue by contradiction that $\mathcal{T}$ is not a tree, then there must exist a node $v \in \mathcal{T}$ such that v has more than one parent in $\mathcal{T}$. This means that there exist at least two paths $R, p_1, \ldots, v$ and $R, q_1, \ldots, v$ that connect the root of $\mathcal{S}(\mathcal{T})$, which we denote by R, and v. Let t be the last node in $R, p_1, \ldots, v$ and $R, q_1, \ldots, v$ such that $R, \ldots, t$ are common prefix of these two paths. By construction we know that such t must exist since these two paths start from the same root node R (R will be one candidate of such t). Also, we claim that $t \neq v$ otherwise these two paths overlap with each other, which contradicts the assumption that v has multiple parents. This shows that these two paths can be represented as $R, \ldots, t, p, \ldots, v$ and $R, \ldots, t, q, \ldots, v$ where $R, \ldots, t$ are the common prefix shared by these two paths and $p \neq q$ since t is the last common node. From the construction process defined in Def. 9.1.1, we know that both p and q are children of t in $\mathcal{S}$. Recall that for each sum node in $\mathcal{S}$, Def. 9.1.1 takes at most one child, hence we claim that t must be a product node, since both p and q are children of t. Then the paths that $t \to p \rightsquigarrow v$ and $t \to q \rightsquigarrow v$ indicate that $\mathrm{scope}(v) \subseteq \mathrm{scope}(p) \subseteq \mathrm{scope}(t)$ and $\mathrm{scope}(v) \subseteq \mathrm{scope}(q) \subseteq \mathrm{scope}(t)$, leading to $\varnothing \neq \mathrm{scope}(v) \subseteq \mathrm{scope}(p) \cap \mathrm{scope}(q)$, which is a contradiction of the decomposability of the product node t. Hence as long as $\mathcal{S}$ is complete and decomposable, $\mathcal{T}$ must be a tree.

The completeness of $\mathcal{T}$ is trivially satisfied because each sum node has only one child in $\mathcal{T}$. It is also straightforward to verify that $\mathcal{T}$ satisfies the decomposability as $\mathcal{T}$ is an induced subgraph of $\mathcal{S}$, which is decomposable. ■

Theorem 9.1.2. If $\mathcal{T}$ is an induced tree from $\mathcal{S}$ over $\mathbf{X}_{[n]}$, then $\mathcal{T}(\mathbf{x}) = \prod_{(v_i,v_j)\in\mathcal{T}_E} w_{ij} \prod_{i=1}^{n} \mathbb{I}_{x_i}$, where w_{ij} is the edge weight of (v_i, v_j) if v_i is a sum node and $w_{ij} = 1$ if v_i is a product node.

Proof. First, the scope of $\mathcal{T}$ is the same as the scope of $\mathcal{S}$ because the root of $\mathcal{S}$ is also the root of $\mathcal{T}$. This shows that for each X_i there is at least one indicator $\mathbb{I}_{x_i}$ in the leaves otherwise the scope of the root node of $\mathcal{T}$ will be a strict subset of the scope of the root node of $\mathcal{S}$. Furthermore, for each variable X_i there is at most one indicator $\mathbb{I}_{x_i}$ in the leaves. This is observed by the fact that there is at most one child collected from a sum node into $\mathcal{T}$ and if $\mathbb{I}_{x_i}$ and $\mathbb{I}_{\bar{x}_i}$ appear simultaneously in the leaves, then their least common ancestor must be a product node. Note that the least common ancestor of $\mathbb{I}_{x_i}$ and $\mathbb{I}_{\bar{x}_i}$ is guaranteed to exist because of the tree structure of $\mathcal{T}$. However, this leads to a contradiction of the fact that $\mathcal{S}$ is decomposable. As a result, there is exactly one indicator $\mathbb{I}_{x_i}$ for each variable X_i in $\mathcal{T}$. Hence the multiplicative constant of the monomial admits the form $\prod_{i=1}^{n} \mathbb{I}_{x_i}$, which is a product of univariate distributions. More specifically, it is a product of indicator variables in the case of Boolean input variables.

We have already shown that $\mathcal{T}$ is a tree and only product nodes in $\mathcal{T}$ can have multiple children. It follows that the functional form of $f_{\mathcal{T}}(\mathbf{x})$ must be a monomial, and only edge weights that are in $\mathcal{T}$ contribute to the monomial. Combing all the above, we know that $f_{\mathcal{T}}(\mathbf{x}) = \prod_{(v_i,v_i)\in\mathcal{T}_E} w_{ij} \prod_{n=1}^N \mathbb{I}_{x_n}$. ■

Theorem 9.1.3. $\tau_{\mathcal{S}} = V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$, where $V_{\text{root}}(\mathbf{1} \mid \mathbf{1})$ is the value of the network polynomial of $\mathcal{S}$ with input vector $\mathbf{1}$ and all edge weights set to be 1.

Theorem 9.1.4. $\mathcal{S}(\mathbf{x}) = \sum_{t=1}^{\tau_{\mathcal{S}}} \mathcal{T}_t(\mathbf{x})$, where $\mathcal{T}_t$ is the tth unique induced tree of $\mathcal{S}$.

Proof. We prove by induction on the height of $\mathcal{S}$. If the height of $\mathcal{S}$ is 2, then depending on the type of the root node, we have two cases:

1. If the root is a sum node with K children, then there are $C_K^1 = K$ different subgraphs that satisfy Def. 9.1.1, which is exactly the value of the network by setting all the indicators and edge weights from the root to be 1.
2. If the root is a product node then there is only 1 subgraph which is the graph itself. Again, this equals to the value of $\mathcal{S}$ by setting all indicators to be 1.

Assume the theorem is true for SPNs with height $\leq h$. Consider an SPN $\mathcal{S}$ with height $h+1$. Again, depending on the type of the root node, we need to discuss two cases:

1. If the root is a sum node with K children, where the kth sub-SPN has $f_{\mathcal{S}_k}(\mathbf{1}|\mathbf{1})$ unique induced trees, then by Def. 9.1.1 the total number of unique induced trees of $\mathcal{S}$ is $\sum_{k=1}^K f_{\mathcal{S}_k}(\mathbf{1}|\mathbf{1}) = \sum_{k=1}^K 1 \cdot f_{\mathcal{S}_k}(\mathbf{1}|\mathbf{1}) = f_{\mathcal{S}}(\mathbf{1}|\mathbf{1})$.
2. If the root is a product node with K children, then the total number of unique induced trees of $\mathcal{S}$ can then be computed by $\prod_{k=1}^K f_{\mathcal{S}_k}(\mathbf{1}|\mathbf{1}) = f_{\mathcal{S}}(\mathbf{1}|\mathbf{1})$.

The second part of the theorem follows by using distributive law between multiplication and addition to combine unique trees that share the same prefix in bottom-up order. ■

9.5.2 MLE as Signomial Programming

Proposition 9.2.1. The MLE problem for SPNs is a signomial program.

Proof. Using the definition of $\Pr(\mathbf{x}|\mathbf{w})$ and Corollary 9.2.1, let $\tau = f_{\mathcal{S}}(\mathbf{1}|\mathbf{1})$, the MLE problem can be rewritten as

$$\begin{aligned} \text{maximize}_{\mathbf{w}} \quad & \frac{f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{f_{\mathcal{S}}(\mathbf{1}|\mathbf{w})} = \frac{\sum_{t=1}^{\tau} \prod_{n=1}^N \mathbb{I}_{x_n}^{(t)} \prod_{d=1}^D w_d^{\mathbb{I}_{w_d\in\mathcal{T}_t}}}{\sum_{t=1}^{\tau} \prod_{d=1}^D w_d^{\mathbb{I}_{w_d\in\mathcal{T}_t}}} \\ \text{subject to} \quad & \mathbf{w} \in \mathbb{R}_{++}^D \end{aligned} \tag{9.11}$$

which we claim is equivalent to:

$$\begin{aligned} \text{minimize}_{\mathbf{w},z} \quad & -z \\ \text{subject to} \quad & \sum_{t=1}^{\tau} z \prod_{d=1}^{D} w_d^{\mathbb{I}_{w_d\in\mathcal{T}_t}} - \sum_{l=1}^{\tau} \prod_{n=1}^{N} \mathbb{I}_{x_n}^{(t)} \prod_{d=1}^{D} w_d^{\mathbb{I}_{w_d\in\mathcal{T}_t}} \leq 0 \\ & \mathbf{w} \in \mathbb{R}_{++}^D, z > 0 \end{aligned} \tag{9.12}$$

It is easy to check that both the objective function and constraint function in (9.12) are signomials. To see the equivalence of (9.11) and (9.12), let p^* be the optimal value of (9.11) achieved at $\mathbf{w}^*$. Choose $z = p^*$ and $\mathbf{w} = \mathbf{w}^*$ in (9.12), then $-z$ is also the optimal solution of (9.12) otherwise we can find feasible $(z', \mathbf{w}')$ in (9.12) which has $-z' < -z \Leftrightarrow z' > z$. Combined with the constraint function in (9.12), we have $p^* = z < z' \leq \frac{f_S(\mathbf{x}|\mathbf{w}')}{f_S(\mathbf{1}|\mathbf{w}')}$, which contradicts the optimality of p^*. In the other direction, let $z^*, \mathbf{w}^*$ be the solution that achieves optimal value of (9.12), then we claim that z^* is also the optimal value of (9.11), otherwise there exists a feasible $\mathbf{w}$ in (9.11) such that $z \triangleq \frac{f_S(\mathbf{x}|\mathbf{w})}{f_S(\mathbf{1}|\mathbf{w})} > z^*$. Since $(\mathbf{w}, z)$ is also feasible in (9.12) with $-z < -z^*$, this contradicts the optimality of z^*. ■

The transformation from (9.11) to (9.12) does not make the problem any easier to solve. Rather, it destroys the structure of (9.11), i.e., the objective function of (9.11) is the ratio of two posynomials. However, the equivalent transformation does reveal some insights about the intrinsic complexity of the optimization problem, which indicates that it is hard to solve (9.11) efficiently with the guarantee of achieving a globally optimal solution.

9.5.3 Convergence of CCCP for SPNs

We discussed before that the sequence of function values $\{f(\mathbf{y}^{(k)})\}$ converges to a limiting point. However, this fact alone does not necessarily indicate that $\{f(\mathbf{y}^{(k)})\}$ converges to $f(\mathbf{y}^*)$ where $\mathbf{y}^*$ is a stationary point of $f(\cdot)$ nor does it imply that the sequence $\{\mathbf{y}^{(k)}\}$ converges as $k \to \infty$. *Zangwill's global convergence theory* (Zangwill, 1969) has been successfully applied to study the convergence properties of many iterative algorithms frequently used in machine learning, including EM (Wu, 1983), generalized alternating minimization (Gunawardana and Byrne, 2005) and also CCCP (Lanckriet and Sriperumbudur, 2009). Here we also apply Zangwill's theory and combine the analysis from Lanckriet and Sriperumbudur (2009) to show the following theorem:

Theorem 9.3.1. Let $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$ be any sequence generated using Eq. (9.10) from any positive initial point, then all the limiting points of $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$ are stationary points of the DCP in (9.2). In addition, $\lim_{k\to\infty} f(\mathbf{y}^{(k)}) = f(\mathbf{y}^*)$, where $\mathbf{y}^*$ is some stationary point of (9.2).

Proof. We will use Zangwill's global convergence theory for iterative algorithms (Zangwill, 1969) to show the convergence in our case. Before showing the proof, we need to first introduce the notion of "point-to-set mapping", where the output of the mapping is defined to be a set. More formally, a point-to-set map Φ from a set $\mathcal{X}$ to $\mathcal{Y}$ is defined as $\Phi : \mathcal{X} \mapsto \mathcal{P}(\mathcal{Y})$, where $\mathcal{P}(\mathcal{Y})$ is the power set of $\mathcal{Y}$. Suppose $\mathcal{X}$ and $\mathcal{Y}$ are equipped with the norm $\|\cdot\|_{\mathcal{X}}$ and $\|\cdot\|_{\mathcal{Y}}$, respectively. A point-to-set map Φ is said to be *closed* at $x^* \in \mathcal{X}$ if $x_k \in \mathcal{X}$, $\{x_k\}_{k=1}^{\infty} \to x^*$ and $y_k \in \mathcal{Y}, \{y_k\}_{k=1}^{\infty} \to y^*, y_k \in \Phi(x_k)$ imply that $y^* \in \Phi(x^*)$. A point-to-set map Φ is said to be closed on $S \subseteq \mathcal{X}$ if Φ is closed at every point in S. The concept of *closedness* in the point-to-set map setting reduces to *continuity* if we restrict that the output of Φ to be a set of singleton for every possible input, i.e., when Φ is a point-to-point mapping.

Theorem 9.5.1 (Global Convergence Theorem (Zangwill, 1969))**.** Let the sequence $\{x_k\}_{k=1}^{\infty}$ be generated by $x_{k+1} \in \Phi(x_k)$, where $\Phi(\cdot)$ is a point-to-set map from $\mathcal{X}$ to $\mathcal{X}$. Let a solution set $\Gamma \subseteq \mathcal{X}$ be given, and suppose that:

1. all points x_k are contained in a compact set $S \subseteq \mathcal{X}$.
2. Φ is closed over the complement of Γ.
3. there is a continuous function α on $\mathcal{X}$ such that:
   (a) if $x \notin \Gamma$, $\alpha(x') > \alpha(x)$ for $\forall x' \in \Phi(x)$.
   (b) if $x \in \Gamma, \alpha(x') \geq \alpha(x)$ for $\forall x' \in \Phi(x)$.

Then all the limit points of $\{x_k\}_{k=1}^{\infty}$ are in the solution set Γ and $\alpha(x_k)$ converges monotonically to $\alpha(x^*)$ for some $x^* \in \Gamma$.

Let $\mathbf{w} \in \mathbb{R}_{+}^{D}$. Let $\Phi(\mathbf{w}^{(k-1)}) = \exp(\arg\max_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)}))$ and let $\alpha(\mathbf{w}) = f(\log \mathbf{w}) = f(\mathbf{y}) = \log f_{\mathcal{S}}(\mathbf{x}| \exp(\mathbf{y})) - \log f_{\mathcal{S}}(\mathbf{1}| \exp(\mathbf{y}))$. Here we use $\mathbf{w}$ and $\mathbf{y}$ interchangeably as $\mathbf{w} = \exp(\mathbf{y})$ or each component is a one-to-one mapping. Note that since the $\arg\max_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$ given $\mathbf{y}^{(k-1)}$ is achievable, $\Phi(\cdot)$ is a well defined point-to-set map for $\mathbf{w} \in \mathbb{R}_{+}^{D}$.

Specifically, in our case given $\mathbf{w}^{(k-1)}$, at each iteration of Eq. 9.10 we have

$$w'_{ij} = \frac{w_{ij} f_{v_j}(\mathbf{1}|\mathbf{w})}{\sum_j w_{ij} f_{v_j}(\mathbf{1}|\mathbf{w})} \propto w_{ij}^{(k-1)} \frac{f_{v_j}(\mathbf{x}|\mathbf{w}^{(k-1)})}{f_{\mathcal{S}}(\mathbf{x}|\mathbf{w}^{(k-1)})} \frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w}^{(k-1)})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w}^{(k-1)})}$$

i.e., the point-to-set mapping is given by

$$\Phi_{ij}(\mathbf{w}^{(k-1)}) = \frac{w_{ij}^{(k-1)} f_{v_j}(\mathbf{x}|\mathbf{w}^{(k-1)}) \frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w}^{(k-1)})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w}^{(k-1)})}}{\sum_{j'} w_{ij'}^{(k-1)} f_{v_{j'}}(\mathbf{x}|\mathbf{w}^{(k-1)}) \frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w}^{(k-1)})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w}^{(k-1)})}}$$

Let $S = [0, 1]^D$, the D dimensional hyper cube. Then the above update formula indicates that $\Phi(\mathbf{w}^{(k-1)}) \in S$. Furthermore, if we assume $\mathbf{w}^{(1)} \in S$, which can be obtained by local normalization before any update, we can guarantee that $\{\mathbf{w}_k\}_{k=1}^{\infty} \subseteq S$, which is a compact set in $\mathbb{R}_{+}^{D}$.

The solution to $\max_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$ is not unique. In fact, there are infinitely many solutions to this nonlinear equations. However, as we define above, $\Phi(\mathbf{w}^{(k-1)})$ returns *one* solution to this convex program in the D dimensional hyper cube. Hence in our case $\Phi(\cdot)$ reduces to a point-to-point map, where the definition of *closedness* of a point-to-set map reduces to the notion of *continuity* of a point-to-point map. Define $\Gamma = \{\mathbf{w}^* \mid \mathbf{w}^* \text{ is a stationary point of } \alpha(\cdot)\}$. Hence we only need to verify the continuity of $\Phi(\mathbf{w})$ when $\mathbf{w} \in S$. To show this, we first characterize the functional form of $\frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})}$ as it is used inside $\Phi(\cdot)$. We claim that for each node v_i, $\frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})}$ is again, a posynomial function of $\mathbf{w}$. A graphical illustration is given in Fig. 9.5.1 to explain the process. This can also be derived from the sum rules and product rules used during top-down differentiation. More specifically, if v_i is a product node, let $v_j, j = 1, \ldots, J$ be its parents in the network, which are assumed to be sum nodes, the differentiation of $f_{\mathcal{S}}$ with respect to f_{v_i} is given by $\frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})} = \sum_{j=1}^{J} \frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_j}(\mathbf{x}|\mathbf{w})} \frac{\partial f_{v_j}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})}$. We reach

$$\frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})} = \sum_{j=1}^{J} w_{ij} \frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_j}(\mathbf{x}|\mathbf{w})} \tag{9.13}$$

Similarly, if v_i is a sum node and its parents $v_j, j = 1, \ldots, J$ are assumed to be product nodes, we have

$$\frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})} = \sum_{j=1}^{J} \frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_j}(\mathbf{x}|\mathbf{w})} \frac{f_{v_j}(\mathbf{x}|\mathbf{w})}{f_{v_i}(\mathbf{x}|\mathbf{w})} \tag{9.14}$$

Since v_j is a product node and v_j is a parent of v_i, so the last term in Eq. 9.14 can be equivalently expressed as

$$\frac{f_{v_j}(\mathbf{x}|\mathbf{w})}{f_{v_i}(\mathbf{x}|\mathbf{w})} = \prod_{h \neq i} f_{v_h}(\mathbf{x}|\mathbf{w})$$

where the index is range from all the children of v_j except v_i. Combining the fact that the partial differentiation of $f_{\mathcal{S}}$ with respect to the root node is 1 and that each f_v is a posynomial function, it follows by induction in top-down order that $\frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})}$ is also a posynomial function of $\mathbf{w}$.

We have shown that both the numerator and the denominator of $\Phi(\cdot)$ are posynomial functions of $\mathbf{w}$. Because posynomial functions are continuous functions, in order to show that $\Phi(\cdot)$ is also continuous on $S\backslash\Gamma$, we need to guarantee that the denominator is not a degenerate posynomial function, i.e., the denominator of $\Phi(\mathbf{w}) \neq 0$ for all possible input vector $\mathbf{x}$. Recall that $\Gamma = \{\mathbf{w}^* \mid \mathbf{w}^* \text{ is a stationary point of } \alpha(\cdot)\}$, hence $\forall \mathbf{w} \in S\backslash\Gamma$, $\mathbf{w} \notin \text{bd } S$, where bd S is the boundary of the D dimensional hyper cube S. Hence we have $\forall \mathbf{w} \in S\backslash\Gamma \Rightarrow \mathbf{w} \in \text{int } S \Rightarrow \mathbf{w} > 0$ for each component. This immediately leads to $f_v(\mathbf{x}|\mathbf{w}) > 0, \forall v$. As a result, $\Phi(\mathbf{w})$ is continuous on $S\backslash\Gamma$ since it is the ratio of two strictly positive posynomial functions.

We now verify the third property in Zangwill's global convergence theory. At each iteration of CCCP, we have the following two cases to consider:

1. If $\mathbf{w}^{(k-1)} \notin \Gamma$, i.e., $\mathbf{w}^{(k-1)}$ is not a stationary point of $\alpha(\mathbf{w})$, then $\mathbf{y}^{(k-1)} \notin \arg\max_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$, so we have $\alpha(\mathbf{w}^{(k)}) = f(\mathbf{y}^{(k)}) \geq \hat{f}(\mathbf{y}^{(k)}, \mathbf{y}^{(k-1)}) > \hat{f}(\mathbf{y}^{(k-1)}, \mathbf{y}^{(k-1)}) = f(\mathbf{y}^{(k-1)}) = \alpha(\mathbf{w}^{(k-1)})$.
2. If $\mathbf{w}^{(k-1)} \in \Gamma$, i.e., $\mathbf{w}^{(k-1)}$ is a stationary point of $\alpha(\mathbf{w})$, then $\mathbf{y}^{(k-1)} \in \arg\max_{\mathbf{y}} \hat{f}(\mathbf{y}, \mathbf{y}^{(k-1)})$, so we have $\alpha(\mathbf{w}^{(k)}) = f(\mathbf{y}^{(k)}) \geq \hat{f}(\mathbf{y}^{(k)}, \mathbf{y}^{(k-1)}) = \hat{f}(\mathbf{y}^{(k-1)}, \mathbf{y}^{(k-1)}) = f(\mathbf{y}^{(k-1)}) = \alpha(\mathbf{w}^{(k-1)})$.

By Zangwill's global convergence theory, we now conclude that all the limit points of $\{\mathbf{w}_k\}_{k=1}^\infty$ are in Γ and $\alpha(\mathbf{w}_k)$ converges monotonically to $\alpha(\mathbf{w}^*)$ for some stationary point $\mathbf{w}^* \in \Gamma$. ■

Remark Technically we need to choose $\mathbf{w}_0 \in \text{int } S$ to ensure the continuity of $\Phi(\cdot)$. This initial condition combined with the fact that inside each iteration of CCCP the algorithm only applies positive multiplicative update and renormalization, ensure that after any finite k steps, $\mathbf{w}_k \in \text{int}S$. Theoretically, only in the limit it is possible that some components of $\mathbf{w}_\infty$ may become 0. However in practice, due to the numerical precision of float numbers on computers, it is possible that after some finite update steps some of the components in $\mathbf{w}_k$ become 0. So in practical implementation we recommend to use a small positive number ϵ to smooth out such 0 components in $\mathbf{w}_k$ during the iterations of CCCP. Such smoothing may hurt the monotonic property of CCCP, but this can only happens when $\mathbf{w}_k$ is close to $\mathbf{w}^*$ and we can use early stopping to obtain a solution in the interior of S.

Thm. 9.3.1 only implies that any limiting point of the sequence $\{\mathbf{w}_k\}_{k=1}^\infty(\{\mathbf{y}_k\}_{k=1}^\infty)$ must be a stationary point of the log-likelihood function and $\{f(\mathbf{y})_k\}_{k=1}^\infty$ must converge to some $f(\mathbf{y}^*)$ where $\mathbf{y}^*$ is a stationary point. Thm. 9.3.1 does not imply that the sequence $\{\mathbf{w}_k\}_{k=1}^\infty(\{\mathbf{y}_k\}_{k=1}^\infty)$ is guaranteed to converge. Lanckriet and Sriperumbudur (2009) studies the convergence property of general CCCP procedure. Under more strong conditions, i.e., the strict concavity of the surrogate function or that $\Phi()$ to be a contraction mapping, it is possible to show that the sequence $\{\mathbf{w}_k\}_{k=1}^\infty(\{\mathbf{y}_k\}_{k=1}^\infty)$ also converges. However, none of such conditions hold in our case. In fact, in general there are infinitely many fixed points of $\Phi(\cdot)$, i.e., the equation $\Phi(\mathbf{w}) = \mathbf{w}$ has infinitely many solutions in S. Also, for a fixed value t, if $\alpha(\mathbf{w}) = t$ has at least one solution, then there are infinitely many solutions. Such properties of SPNs make it generally very hard to guarantee the convergence of the sequence $\{\mathbf{w}_k\}_{k=1}^\infty(\{\mathbf{y}_k\}_{k=1}^\infty)$. We give a very simple example below to illustrate the hardness in SPNs in Fig. 9.5.2. Consider applying the CCCP procedure to learn the parameters on the SPN given in Fig. 9.5.2 with three instances $\{(0,1), (1,0), (1,1)\}$. Then if we choose the initial parameter $\mathbf{w}_0$ such that the weights over the indicator variables are set as shown in Fig. 9.5.2, then any assignment of (w_1, w_2, w_3) in the probability simplex will be equally optimal in terms of likelihood on inputs. In this example, there are uncountably infinite equal solutions, which invalidates the finite solution set requirement given in (Lanckriet and Sriperumbudur, 2009) in order to show the convergence of $\{\mathbf{w}_k\}_{k=1}^\infty$. However, we emphasize that the convergence of the sequence $\{\mathbf{w}_k\}_{k=1}^\infty$ is not as important as the convergence of $\{\alpha(\mathbf{w})_k\}_{k=1}^\infty$ to desired locations on the log-likelihood surface as in practice any $\mathbf{w}^*$ with equally good log-likelihood may suffice for the inference/prediction task.

It is worth to point out that the above theorem *does not* imply the convergence of the sequence $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$. Thm. 9.3.1 only indicates that all the limiting points of $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$, i.e., the limits of subsequences of $\{\mathbf{w}^{(k)}\}_{k=1}^{\infty}$, are stationary points of the DCP in (9.2). We also present a negative example in Fig. 9.5.2 that invalidates the application of Zangwill's global convergence theory on the analysis in this case.

The convergence rate of general CCCP is still an open problem (Lanckriet and Sriperumbudur, 2009). Salakhutdinov et al. (2002) studied the convergence rate of unconstrained bound optimization algorithms with differentiable objective functions, of which our problem is a special case. The conclusion is that depending on the curvature of f_1 and f_2 (which are functions of the training data), CCCP will exhibit either a quasi-Newton behavior with superlinear convergence or first-order convergence. We show in experiments that CCCP normally exhibits a fast, superlinear convergence rate compared with PGD, EG and SMA. Both CCCP and EM are special cases of a more general framework known as Majorization-Maximization. We show that in the case of SPNs these two algorithms coincide with each other, i.e., they lead to the same update formulas despite the fact that they start from totally different perspectives.

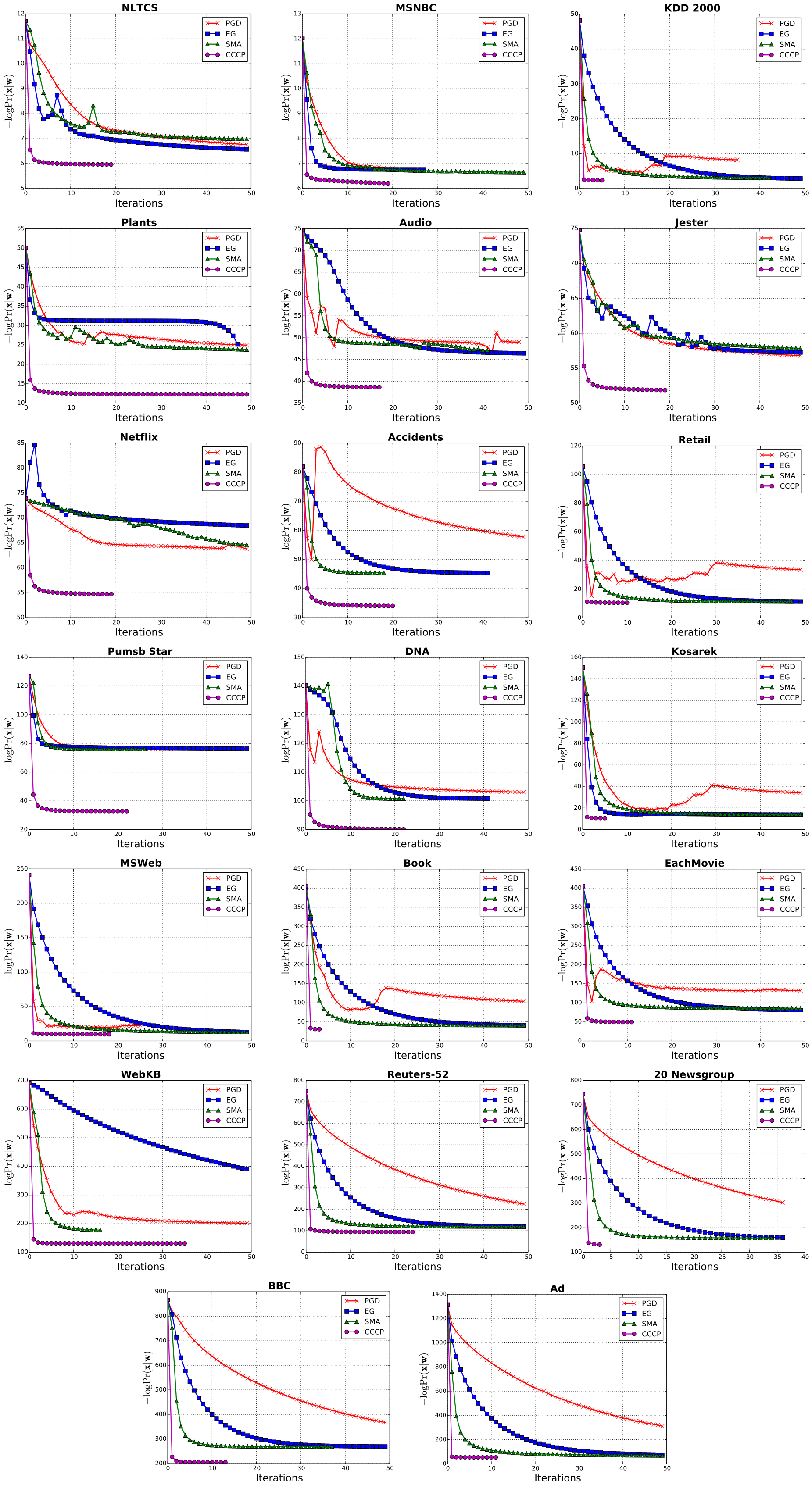


Figure 9.4.1: Negative log-likelihood values versus number of iterations for PGD, EG, SMA and CCCP.

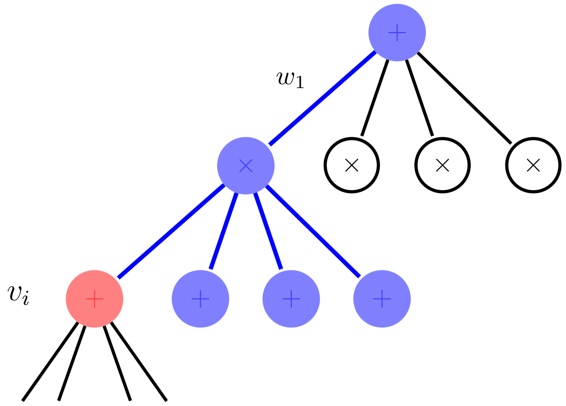


Figure 9.5.1: Graphical illustration of $\frac{\partial f_{\mathcal{S}}(\mathbf{x}|\mathbf{w})}{\partial f_{v_i}(\mathbf{x}|\mathbf{w})}$. The partial derivative of $f_{\mathcal{S}}$ with respect to f_{v_i} (in red) is a posynomial that is a product of edge weights lying on the path from root to v_i and network polynomials from nodes that are children of product nodes on the path (highlighted in blue).

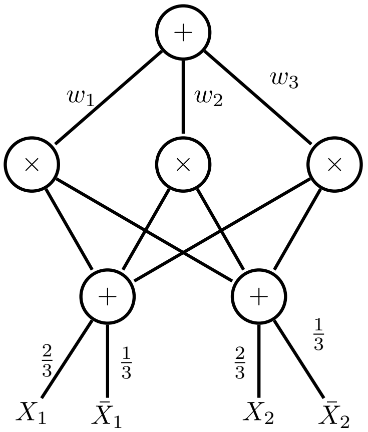


Figure 9.5.2: A counterexample of SPN over two binary random variables where the weights w_1, w_2, w_3 are symmetric and indistinguishable.

Appendix

9.A Experiment Details

9.A.1 Methods

We will briefly review the current approach for training SPNs using projected gradient descent (PGD). Another related approach is to use exponentiated gradient (EG) (Kivinen and Warmuth, 1997) to optimize (9.11). PGD optimizes the log-likelihood by projecting the intermediate solution back to the positive orthant after each gradient update. Since the constraint in (9.11) is an open set, we need to manually create a closed set on which the projection operation can be well defined. One feasible choice is to project on to $\mathbb{R}^D_\epsilon \triangleq \{\mathbf{w} \in \mathbb{R}^D_{++} \mid w_d \geq \epsilon, \forall d\}$ where $\epsilon > 0$ is assumed to be very small. To avoid the projection, one direct solution is to use the exponentiated gradient (EG) method (Kivinen and Warmuth, 1997), which was first applied in an online setting and latter successfully extended to batch settings when training with convex models. EG admits a multiplicative update at each iteration and hence avoids the need for projection in PGD. However, EG is mostly applied in convex setting and it is not clear whether the convergence guarantee still holds or not in nonconvex setting.

9.A.2 Experimental Setup

Table 9.A.1 shows the detailed running time of PGD, EG, SMA and CCCP on 20 data sets, measured in seconds.

Table 9.A.1: Running time of 4 algorithms on 20 data sets, measured in seconds.

Data set	PGD	EG	SMA	CCCP
NLTCS	438.35	718.98	458.99	206.10
MSNBC	2720.73	2917.72	8078.41	2008.07
KDD 2k	46388.60	22154.10	27101.50	29541.20
Plants	12595.60	10752.10	7604.09	13049.80
Audio	19647.90	3430.69	12801.70	14307.30
Jester	6099.44	6272.80	4082.65	1931.41
Netflix	29573.10	27931.50	15080.50	8400.20
Accidents	14266.50	3431.82	5776.00	20345.90
Retail	28669.50	7729.89	9866.94	5200.20
Pumsb-star	3115.58	13872.80	4864.72	2377.54
DNA	599.93	199.63	727.56	1380.36
Kosarak	122204.00	112273.00	49120.50	42809.30
MSWeb	136524.00	13478.10	65221.20	45132.30
Book	190398.00	6487.84	69730.50	23076.40
EachMovie	30071.60	32793.60	17751.10	60184.00
WebKB	123088.00	50290.90	44004.50	168142.00
Reuters-52	13092.10	5438.35	20603.70	1194.31
20 Newsgrp	151243.00	96025.80	173921.00	11031.80
BBC	20920.60	18065.00	36952.20	3440.37
Ad	12246.40	2183.08	12346.70	731.48

Chapter 10

Collapsed Variational Inference

Existing parameter learning approaches for SPNs are largely based on the maximum likelihood principle and are subject to overfitting compared to more Bayesian approaches. Exact Bayesian posterior inference for SPNs is computationally intractable. Even approximation techniques such as standard variational inference and posterior sampling for SPNs are computationally infeasible even for networks of moderate size due to the large number of local latent variables per instance. In this chapter, we propose a novel deterministic collapsed variational inference algorithm for SPNs that is computationally efficient, easy to implement and at the same time allows us to incorporate prior information into the optimization formulation. Extensive experiments show a significant improvement in accuracy compared with a maximum likelihood based approach.

10.1 Introduction

Standard variational Bayes inference is an optimization-based approach to approximate the full posterior distribution of a probabilistic model (Jordan et al., 1999). It works by constructing and maximizing an evidence lower bound (ELBO) of the log marginal likelihood function $\log p(\mathbf{x})$. Equivalently, one can minimize the Kullback-Leibler (KL) divergence between the true posterior distribution and the variational posterior distribution. To review, let $\Theta = \{\mathbf{H}, \mathbf{W}\}$ represent the set of latent variables in a probabilistic model (including both the local, $\mathbf{H}$, and the global, $\mathbf{W}$, latent variables) and let $\mathbf{X}$ represent the data. For example, in mixture models such as LDA, $\mathbf{H}$ corresponds to the topic assignments of the words and $\mathbf{W}$ corresponds to the topic-word distribution matrix. The joint probability distribution of Θ and $\mathbf{X}$ is $p(\mathbf{X}, \Theta \mid \boldsymbol{\alpha})$ where $\boldsymbol{\alpha}$ is the set of hyperparameters of the model. Standard VI approximates the true posterior $p(\Theta \mid \mathbf{X}, \boldsymbol{\alpha})$ with a variational distribution $q(\Theta \mid \boldsymbol{\beta})$ with a set of variational parameters $\boldsymbol{\beta}$. This reduces an inference problem into an optimization problem in which the objective function is given by the ELBO:

$$\log p(\mathbf{x} \mid \boldsymbol{\alpha}) \geq \mathbb{E}_q[\log p(\mathbf{x}, \Theta|\boldsymbol{\alpha})] + \mathbb{H}[q(\Theta|\boldsymbol{\beta})] =: \widehat{\mathcal{L}}(\boldsymbol{\beta}).$$

The variational distribution $q(\Theta \mid \boldsymbol{\beta})$ is typically assumed to be fully factorized, with each variational parameter β_i governing one latent variable θ_i. The variational parameters are then optimized to maximize the ELBO, and the optimal variational posterior will be used as a surrogate to the true posterior distribution $p(\Theta \mid \mathbf{X}, \boldsymbol{\alpha})$. It is worth noting that in standard VI the number of variational parameters is linearly proportional to the number of latent variables, including both the global and local latent variables.

10.1.1 Motivation

Standard VI methods are computationally infeasible for SPNs due to the large number of local latent variables for each training instance, as shown in Fig. 10.1.1. Let $\mathbf{W}$ denote the set of global latent variables (model parameters) and $\mathbf{H}_d$ denote the set of local latent variables, where d indexes over the training instances. In standard VI we need to maintain a set of variational parameters for each of the latent variables, i.e., $\mathbf{W}$ and $\{\mathbf{H}_d\}_{d=1}^{D}$. In the case of SPNs, the number of local latent variables is exactly the number of internal sum nodes in the network, which can be linearly proportional to the size of the network, $|\mathcal{S}|$. Together with the global latent variables, this leads to a total number of $O(D|\mathcal{S}| + |\mathcal{S}|)$ variational parameters to be maintained in standard VI. As we will see in the experiments, this is prohibitively expensive for SPNs that range from tens of thousands to millions of nodes.

To deal with the expensive computation and storage in standard VI, we develop CVB-SPN, a collapsed variational algorithm, which, instead of assuming independence among the local latent variables, models the dependence among them in an exact fashion. More specifically, we marginalize all the local latent variables out of the joint posterior distribution and maintain a marginal posterior distribution over the global latent variables directly. This approach would not be an option for many graphical models, but as we will see later, we can take advantage of fast exact inference in SPNs. On the other hand, we still variationally approximate the posterior distribution of the global variables (model parameters) using a mean-field approach. Note this basic assumption made in CVB-SPN is different from typical collapsed variational approaches developed for LDA and HDP (Teh et al., 2006, 2007), where global latent variables are integrated out and local latent variables are assumed to be mutually independent. As a result, CVB-SPN models the marginal posterior distribution over global latent variables only and hence the number of variational parameters to be optimized is $O(|\mathcal{S}|)$ compared with $O(D|\mathcal{S}|)$ in the standard case.

Intuitively, this is not an unreasonable assumption to make since compared with local latent variables, the global latent variables in Fig. 10.1.1 are further away from the influence of observations of $\mathbf{X}$ and they are mutually independent a priori. Besides the computational consideration, another motivation to collapse

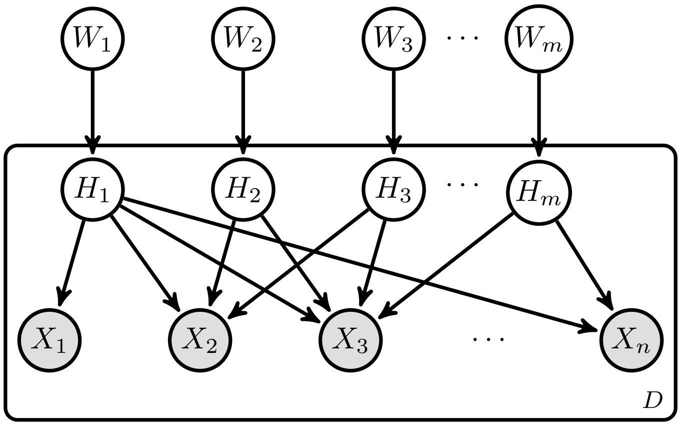


Figure 10.1.1: Graphical model representation of SPN $\mathcal{S}$. The box is a plate that represents replication over D training instances. $m = O(|\mathcal{S}|)$ corresponds to the number of sum nodes and n is the number of observable variables in $\mathcal{S}$. Typically, $m \gg n$. W, H, X correspond to global latent variables, local latent variables and observable variables, respectively.

out the local latent variables is that no interpretation associated with the local latent variables in an SPN has so far proven to be useful in practice. Unlike other mixture models such as LDA, where local latent variables correspond to topic assignment of words, the sum nodes in SPNs do not share typical statistical interpretations that may be useful in real-world applications.

CVB-SPN makes fewer assumptions about the independence among random variables and has fewer variational variables to be optimized, but to achieve these benefits, we must overcome the following difficulties: the cost of exact marginalization over local latent variables and the non-conjugacy between the prior distribution and the likelihood function introduced by the marginalization. The first problem is elegantly handled by the property of SPN that exact inference over $\mathbf{X}$ is always tractable. In what follows we proceed to derive the CVB-SPN algorithm that efficiently solves the second problem.

10.2 Collapsed Variational Inference

Throughout the derivation we will assume that the weights w_{kj} associated with a sum node k are locally normalized, i.e., $\sum_{j\in\text{Ch}(k)} w_{kj} = 1, \forall k$. This can be achieved by a bottom-up pass of the network in time $O(|\mathcal{S}|)$ without changing the joint probability distribution over $\mathbf{X}$; see Peharz et al. (2015) and Zhao et al. (2015b) for more details. For SPNs with locally normalized weights $\mathbf{w}$, it can be verified that $V_{\text{root}}(\mathbf{1} \mid \mathbf{w}) = 1$, hence the marginal likelihood function $p(\mathbf{x} \mid \mathbf{w})$ that marginalizes out local latent variables $\mathbf{h}$ is given by $V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$.

Since the weights associated with each sum node k are locally normalized, we can interpret each sum node as a multinomial random variable with one possible value for each child of the sum node. It follows that we can specify a Dirichlet prior $\text{Dir}(w_k \mid \alpha_k)$ for each sum node k. Since all the global latent variables are d-separated before we get the observation $\mathbf{x}$, a prior distribution over all the weights can be factorized as:

$$p(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{k=1}^{m} p(w_k \mid \alpha_k) = \prod_{k=1}^{m} \text{Dir}(w_k \mid \alpha_k). \tag{10.1}$$

The true posterior distribution after a sequence of observations $\{\mathbf{x}_d\}_{d=1}^D$ is:

$$p(\mathbf{w} \mid \{\mathbf{x}_d\}_{d=1}^D, \boldsymbol{\alpha}) \propto p(\mathbf{w} \mid \boldsymbol{\alpha}) \prod_{d=1}^{D} V_{\text{root}}(\mathbf{x}_d \mid \mathbf{w}) = \prod_{k=1}^{m} \text{Dir}(w_k \mid \alpha_k) \prod_{d=1}^{D} \sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{kj} \prod_{i=1}^{n} p_t(x_{di}). \tag{10.2}$$

Intuitively, Eq. (10.2) indicates that the true posterior distribution of the model parameters is a mixture model where the number of components scales as $\tau_{\mathcal{S}}^D$ and each component is a product of m Dirichlets, one for each sum node. Here $\tau_{\mathcal{S}}$ corresponds to the number of induced trees in $\mathcal{S}$ (cf. Thm. 9.1.4). For discrete SPNs the leaf univariate distributions are simply point mass distributions, i.e., indicator variables. For continuous distributions such as Gaussians, we also need to specify the priors for the parameters of those leaf distributions, but the analysis goes the same as the discrete case. Eq. (10.2) is intractable to compute exactly, hence we resort to collapsed variational inference. To simplify notation, we will assume that there is only one instance in the data set, i.e., $D = 1$. Extension of the following derivation to multiple training instances is straightforward. Consider the log marginal probability over observable variables that upper bounds the new ELBO $\mathcal{L}(\boldsymbol{\beta})$:

$$\begin{aligned} \log p(\mathbf{x} \mid \boldsymbol{\alpha}) &\geq \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log p(\mathbf{x}, \mathbf{w} \mid \boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w} \mid \boldsymbol{\beta})] \\ &= \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log \sum_{\mathbf{h}} p(\mathbf{x}, \mathbf{h}, \mathbf{w} \mid \boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w} \mid \boldsymbol{\beta})] \\ &=: \mathcal{L}(\boldsymbol{\beta}), \end{aligned} \tag{10.3}$$

where $q(\mathbf{w} \mid \boldsymbol{\beta}) = \prod_{k=1}^m \text{Dir}(w_k \mid \beta_k)$ is the factorized variational distribution over $\mathbf{w}$ to approximate the true marginal posterior distribution $p(\mathbf{w} \mid \mathbf{x}, \boldsymbol{\alpha}) = \sum_{\mathbf{h}} p(\mathbf{w}, \mathbf{h} \mid \mathbf{x}, \boldsymbol{\alpha})$. Note that here we are using $q(\mathbf{w})$ as opposed to $q(\mathbf{w})q(\mathbf{h})$ in standard VI. We argue that $\mathcal{L}(\boldsymbol{\beta})$ gives us a better lower bound to optimize than the one given by standard VI. To see this, let $\widehat{\mathcal{L}}(\boldsymbol{\beta})$ be the ELBO given by standard VI, i.e.,

$$\widehat{\mathcal{L}}(\boldsymbol{\beta}) = \mathbf{E}_{q(\mathbf{w})q(\mathbf{h})}[\log p(\mathbf{x}, \mathbf{w}, \mathbf{h}|\boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w})q(\mathbf{h})],$$

we have:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\beta}) &= \mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{x}, \mathbf{w} \mid \boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w})] \\ &= \mathbb{E}_{q(\mathbf{w})}\left[\mathbb{E}_{p(\mathbf{h}|\mathbf{w},\mathbf{x},\boldsymbol{\alpha})}[\log \frac{p(\mathbf{h}, \mathbf{w}, \mathbf{x} \mid \boldsymbol{\alpha})}{p(\mathbf{h} \mid \mathbf{w}, \mathbf{x}, \boldsymbol{\alpha})}]\right] + \mathbb{H}[q(\mathbf{w})] \\ &= \max_{q(\mathbf{h}|\mathbf{w})} \mathbb{E}_{q(\mathbf{w})}\left[\mathbb{E}_{q(\mathbf{h}|\mathbf{w})}[\log \frac{p(\mathbf{h}, \mathbf{w}, \mathbf{x} \mid \boldsymbol{\alpha})}{q(\mathbf{h} \mid \mathbf{w})}]\right] + \mathbb{H}[q(\mathbf{w})] \\ &\geq \max_{q(\mathbf{h})} \mathbb{E}_{q(\mathbf{w})}\left[\mathbb{E}_{q(\mathbf{h})}[\log \frac{p(\mathbf{h}, \mathbf{w}, \mathbf{x} \mid \boldsymbol{\alpha})}{q(\mathbf{h})}]\right] + \mathbb{H}[q(\mathbf{w})] \\ &\geq \mathbb{E}_{q(\mathbf{w})q(\mathbf{h})}[\log p(\mathbf{x}, \mathbf{w}, \mathbf{h} \mid \boldsymbol{\alpha})] + \mathbb{H}[q(\mathbf{w})q(\mathbf{h})] \\ &= \widehat{\mathcal{L}}(\boldsymbol{\beta}). \end{aligned}$$

The first equality holds by the definition of $\mathcal{L}(\boldsymbol{\beta})$ and the second equality holds since $\log p(\mathbf{x}, \mathbf{w} \mid \boldsymbol{\alpha}) = \log \frac{p(\mathbf{h},\mathbf{w},\mathbf{x}|\boldsymbol{\alpha})}{p(\mathbf{h}|\mathbf{w},\mathbf{x},\boldsymbol{\alpha})}$ is constant w.r.t. $\mathbb{E}_{p(\mathbf{h}|\mathbf{w},\mathbf{x},\boldsymbol{\alpha})}[\cdot]$. The third equality holds because the inner expectation is the maximum that can be achieved by the negative KL divergence between the approximate posterior $q(\mathbf{h}|\mathbf{w})$ and the true posterior $p(\mathbf{h}|\mathbf{w}, \mathbf{x}, \boldsymbol{\alpha})$. The inequality in the fourth line is due to the fact that the maximization over $q(\mathbf{h} \mid \mathbf{w})$ is free of constraint hence the optimal posterior is given by the true conditional posterior $q^*(\mathbf{h} \mid \mathbf{w}) = p(\mathbf{h} \mid \mathbf{w}, \mathbf{x}, \boldsymbol{\alpha})$, while the maximization over $q(\mathbf{h})$ in the fourth line needs to satisfy the independence assumption made in standard VI, i.e., $q(\mathbf{h} \mid \mathbf{w}) = q(\mathbf{h})$. Combining the inequality above with (10.3), we have

$$\log p(\mathbf{x} \mid \boldsymbol{\alpha}) \geq \mathcal{L}(\boldsymbol{\beta}) \geq \widehat{\mathcal{L}}(\boldsymbol{\beta}), \tag{10.4}$$

which shows that the ELBO given by the collapsed variational inference is a better lower bound than the one given by standard VI. This conclusion is also consistent with the one obtained in collapsed variational inference for LDA and HDP (Teh et al., 2006, 2007) where the authors collapsed out the global latent variables instead of the local latent variables. It is straightforward to verify that the difference between $\log p(\mathbf{x} \mid \boldsymbol{\alpha})$ and $\mathcal{L}(\boldsymbol{\beta})$ is given by $\mathbb{KL}(q(\mathbf{w} \mid \boldsymbol{\beta}) \parallel p(\mathbf{w} \mid \mathbf{x}, \boldsymbol{\alpha}))$, i.e., the KL-divergence between the variational marginal posterior and the exact marginal posterior distribution. Substituting $q(\mathbf{w} \mid \boldsymbol{\beta})$ into the KL-divergence objective and simplifying it, we reach the following optimization objective:

$$\underset{\boldsymbol{\beta}}{\text{minimize}} \quad \mathbb{KL}(q(\mathbf{w} \mid \boldsymbol{\beta}) \parallel p(\mathbf{w} \mid \boldsymbol{\alpha})) - \mathbb{E}_{q(\mathbf{w}\mid\boldsymbol{\beta})}[\log p(\mathbf{x} \mid \mathbf{w})], \tag{10.5}$$

where the first part can be interpreted as a regularization term that penalizes a variational posterior that is too far away from the prior, and the second part is a data-fitting term that requires the variational posterior to have a good fit to the training data set. The first part in ((10.5)) can be efficiently computed due to the factorization assumption of both the prior and the variational posterior. However, the second part does not admit an analytic form because after we marginalize out all the local latent variables, $q(\mathbf{w} \mid \boldsymbol{\beta})$ is no longer conjugate to the likelihood function $p(\mathbf{x}|\mathbf{w}) = V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$. We address this problem in the next section.

10.3 Upper Bound by Logarithmic Transformation

The hardness of computing $\mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log p(\mathbf{x} \mid \mathbf{w})]$ makes the direct optimization of (10.5) infeasible in practice. While nonconjugate inference with little analytic work is possible with recent innovations (Blei et al., 2012; Kingma and Welling, 2013; Mnih and Gregor, 2014; Ranganath et al., 2014; Titsias and Lázaro-Gredilla, 2014; Titsias, 2015), they fail to make use of a key analytic property of SPNs, i.e., easy marginalization, that allows for the optimal variational distributions of local variables to be used. In this section we show how to use a logarithmic transformation trick to develop an upper bound of the objective in (10.5) which leads to the efficient CVB-SPN algorithm.

The key observation is that the likelihood of $\mathbf{w}$ is a posynomial function (Boyd et al., 2007). To see this, as shown in Thm. 9.1.4, we have $p(\mathbf{x} \mid \mathbf{w}) = \sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{kj} \prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i)$. The product term with respect to $\mathbf{x}$, $\prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i)$, is guaranteed to be nonnegative and can be treated as a constant w.r.t. $\mathbf{w}$. Hence each component in $p(\mathbf{x} \mid \mathbf{w})$, in terms of $\mathbf{w}$, is a monomial function with positive multiplicative constant, and it follows that $p(\mathbf{x} \mid \mathbf{w})$ is a posynomial function of $\mathbf{w}$. A natural implication of this observation is that we can do a change of variables transformation to $\mathbf{w}$ such that $\log p(\mathbf{x} \mid \mathbf{w})$ becomes a convex function in terms of the transformed variables. More specifically, let $\mathbf{w}' = \log \mathbf{w}$, where $\log(\cdot)$ is taken elementwise. The log likelihood, now expressed in terms of $\mathbf{w}'$, is

$$\begin{aligned}
\log p(\mathbf{x} \mid \mathbf{w}) &= \log \left(\sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{kj} \prod_{i=1}^{n} p_t(X_i = \mathbf{x}_i) \right) \\
&= \log \left(\sum_{t=1}^{\tau_{\mathcal{S}}} \exp \left(c_t + \sum_{(k,j)\in\mathcal{T}_{tE}} w'_{kj} \right) \right) \\
&=: \log p(\mathbf{x} \mid \mathbf{w}'),
\end{aligned} \tag{10.6}$$

where c_t is defined as $c_t = \log \prod_{i=1}^n p_t(X_i = \mathbf{x}_i)$. For each unique tree $\mathcal{T}_t$, $c_t + \sum_{(k,j)\in\mathcal{T}_{tE}} w'_{kj}$ is an affine function of $\mathbf{w}'$. Since the log-sum-exp function is convex in its argument, $\log p(\mathbf{x} \mid \mathbf{w}')$ is convex in $\mathbf{w}'$. Such a change of variables trick is frequently applied in the geometric programming literature to transform a non-convex posynomial optimization problem into a convex programming problem (Boyd et al., 2007; Chiang, 2005).

The convexity of $\log p(\mathbf{x} \mid \mathbf{w}')$ helps us to develop a lower bound of $\mathbb{E}_{q(\mathbf{w}|\boldsymbol{\beta})}[\log p(\mathbf{x} \mid \mathbf{w})]$ that is efficient to compute. Let $q'(\mathbf{w}')$ be the corresponding variational distribution over $\mathbf{w}'$ induced from $q(\mathbf{w})$ by the bijective transformation between $\mathbf{w}$ and $\mathbf{w}'$. We have

$$\begin{aligned}\mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{x} \mid \mathbf{w})] &= \int q(\mathbf{w}) \log p(\mathbf{x} \mid \mathbf{w})\, d\mathbf{w} \\ &= \int q'(\mathbf{w}') \log p(\mathbf{x} \mid \mathbf{w}')\, d\mathbf{w}' \\ &\geq \log p(\mathbf{x} \mid \mathbb{E}_{q'(\mathbf{w}')}[\mathbf{w}']),\end{aligned}$$

where $\log p(\mathbf{x} \mid \mathbb{E}_{q'(\mathbf{w}')}[\mathbf{w}'])$ means the log-likelihood of $\mathbf{x}$ with the edge weights set to be $\exp(\mathbb{E}_{q'(\mathbf{w}')}[\mathbf{w}']))$. Since $q(\mathbf{w}) = \prod_{k=1}^m \mathrm{Dir}(w_k|\beta_k)$ is a product of Dirichlets, we can compute the weight $\exp\left(\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}]\right)$ for each edge (k,j) as

$$\begin{aligned}\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}] &= \int q'(\mathbf{w}')w'_{kj}\, d\mathbf{w}' = \int q'(w'_k)w'_{kj}\, dw'_k \\ &= \int q(w_k) \log w_{kj}\, dw_k = \psi(\beta_{kj}) - \psi(\sum_{j'} \beta_{kj'}),\end{aligned}$$

where $\psi(\cdot)$ is the digamma function. The equation above then implies the new edge weight can be computed by

$$\exp\left(\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}]\right) = \exp\left(\psi(\beta_{kj}) - \psi(\sum_{j'} \beta_{kj'})\right) \approx \frac{\beta_{kj} - \frac{1}{2}}{\sum_{j'} \beta_{kj'} - \frac{1}{2}}, \tag{10.7}$$

where the approximation is by $\exp(\psi(x)) \approx x - \frac{1}{2}$ when $x > 1$. Note that the mean of the variational posterior is given by $\bar{w}_{kj} = \mathbb{E}_{q(\mathbf{w})}[w_{kj}] = \beta_{kj}/\sum_{j'} \beta_{kj'}$, which is close to $\exp\left(\mathbb{E}_{q'(\mathbf{w}')}[w'_{kj}]\right)$ when $\beta_{kj} > 1$. Roughly speaking, this shows that the lower bound we obtained for $\mathbb{E}_{q(\mathbf{w})}[\log p(\mathbf{x} \mid \mathbf{w})]$ by utilizing the logarithmic transformation is trying to optimize the variational parameters $\boldsymbol{\beta}$ such that the variational posterior mean has a good fit to the training data. This is exactly what we hope for since at the end we need to use the variational posterior mean as a Bayesian estimator of our model parameter. Combining all the analysis above, we formulate the following objective function to be minimized that is an upper bound of the objective function given in (10.5):

$$\mathbb{KL}(q(\mathbf{w} \mid \boldsymbol{\beta}) \parallel p(\mathbf{w} \mid \boldsymbol{\alpha})) - \log p(\mathbf{x} \mid \mathbb{E}_{q'(\mathbf{w}'|\boldsymbol{\beta})}[\mathbf{w}']). \tag{10.8}$$

Note that we will use the approximation given by Eq. (10.7) in the above optimization formulation and in Alg. 3. This is a non-convex optimization problem due to the digamma function involved. Nevertheless we can still achieve a local optimum with projected gradient descent in the experiments. We summarize our algorithm, CVB-SPN, in Alg. 3. For each instance, the gradient of the objective function in (10.8) with respect to $\boldsymbol{\beta}$ can be computed by a bottom-up and top-down pass of the network in time $O(|\mathcal{S}|)$. Please refer to (Peharz et al., 2016; Zhao et al., 2016b) for more details.

Algorithm 3 CVB-SPN

```
Input: Initial $\boldsymbol{\beta}$, prior hyperparameter $\boldsymbol{\alpha}$, training instances $\{\mathbf{x}_d\}_{d=1}^{D}$.
Output: Locally optimal $\boldsymbol{\beta}^*$.
 1: while not converged do
 2:     Update $\mathbf{w} = \exp(\mathbb{E}_{q'(\mathbf{w}'|\boldsymbol{\beta})}[\mathbf{w}'])$ with Eq. (10.7).
 3:     Set $\nabla_{\boldsymbol{\beta}} = 0$.
 4:     for $d = 1$ to $D$ do
 5:         Bottom-up evaluation of $\log p(\mathbf{x}_d|\mathbf{w})$.
 6:         Top-down differentiation of $\frac{\partial}{\partial \mathbf{w}} \log p(\mathbf{x}_d|\mathbf{w})$.
 7:         Update $\nabla_{\boldsymbol{\beta}}$ based on $\mathbf{x}_d$.
 8:     end for
 9:     Update $\nabla_{\boldsymbol{\beta}}$ based on $\mathbb{KL}(q(\mathbf{w}|\boldsymbol{\beta}) \parallel p(\mathbf{w}|\boldsymbol{\alpha}))$.
10:     Update $\boldsymbol{\beta}$ with projected GD.
11: end while
```

Parallel and Stochastic Variants Note that Alg. 3 is easily parallelizable by splitting training instances in the loop (Line 4–Line 8) across threads. It can also be extended to the stochastic setting where at each round we sample one instance or a mini-batch of instances $\{\mathbf{x}_{i_k}\}_{k=1}^{s}$ (s is the size of the mini-batch) from the training set. The stochastic formulation will be helpful when the size of the full training data set is too large to be stored in the main memory as a whole, or when the training instances are streaming so that at each time we only have access to one instance (Hoffman et al., 2013).

10.4 Experiments

10.4.1 Experimental Setting

We conduct experiments on a set of 20 benchmark data sets to compare the performance of the proposed collapsed variational inference method with maximum likelihood estimation (Gens and Domingos, 2012). The 20 real-world data sets used in the experiments have been widely used (Rooshenas and Lowd, 2014) to assess the modeling performance of SPNs. All the features are binary. The 20 data sets also cover both low dimensional and high dimensional statistical estimation, hence, they enable a thorough experimental comparison. Detailed information about each data set as well as the SPN models is shown in Table 10.4.1. All the SPNs are built using LearnSPN (Gens and Domingos, 2013).

We evaluate and compare the performance of CVB-SPN with other parameter learning algorithms in both the batch and online learning settings. For a baseline, we compare to the state-of-the-art parameter learning algorithm for SPNs where the algorithm optimizes the training set log-likelihood directly in order to find a maximum likelihood estimator, which we will denote as MLE-SPN. We compare CVB-SPN and MLE-SPN in both batch and online cases. To have a fair comparison, we apply the same optimization method, i.e., projected gradient descent, to optimize the objective functions in CVB-SPN and MLE-SPN. CVB-SPN optimizes over the variational parameters of the posterior distribution based on (3.5) while MLE-SPN optimizes directly over the model parameters to maximize the log-likelihood of the training set. Since the optimization variables are constrained to be positive in both CVB-SPN and MLE-SPN, we need to project the parameters back onto the positive orthant after every iteration. We fix the projection margin ϵ to 0.01, i.e., $\mathbf{w} = \max\{\mathbf{w}, 0.01\}$ to avoid numerical issues. We implement both methods with backtracking line search to automatically adjust the learning rate at each iteration. In all experiments, the maximum number of iterations is fixed to 50 for both methods. We discard the model parameters

Table 10.4.1: Statistics of data sets and models. n is the number of observable random variables modeled by the network, $|\mathcal{S}|$ is the size of the network and p is the number of parameters to be estimated. $n \times D/p$ means the ratio of training instances times the number of variables to the number of parameters.

Data set	n	$\|\mathcal{S}\|$	p	Train	Valid	Test	$n \times D/p$
NLTCS	16	13,733	1,716	16,181	2,157	3,236	150.871
MSNBC	17	54,839	24,452	291,326	38,843	58,265	202.541
KDD 2k	64	48,279	14,292	180,092	19,907	34,955	806.457
Plants	69	132,959	58,853	17,412	2,321	3,482	20.414
Audio	100	739,525	196,103	15,000	2,000	3,000	7.649
Jester	100	314,013	180,750	9,000	1,000	4,116	4.979
Netflix	100	161,655	51,601	15,000	2,000	3,000	29.069
Accidents	111	204,501	74,804	12,758	1,700	2,551	18.931
Retail	135	56,931	22,113	22,041	2,938	4,408	134.560
Pumsb-star	163	140,339	63,173	12,262	1,635	2,452	31.638
DNA	180	108,021	52,121	1,600	400	1,186	5.526
Kosarak	190	203,321	53,204	33,375	4,450	6,675	119.187
MSWeb	294	68,853	20,346	29,441	3,270	5,000	425.423
Book	500	190,625	41,122	8,700	1,159	1,739	105.783
EachMovie	500	522,753	188,387	4,524	1,002	591	12.007
WebKB	839	1,439,751	879,893	2,803	558	838	2.673
Reuters-52	889	2,210,325	1,453,390	6,532	1,028	1,540	3.995
20 Newsgrp	910	14,561,965	8,295,407	11,293	3,764	3,764	1.239
BBC	1058	1,879,921	1,222,536	1,670	225	330	1.445
Ad	1556	4,133,421	1,380,676	2,461	327	491	2.774

returned by LearnSPN and use random weights as initial model parameters. CVB-SPN is more flexible to incorporate those model weights returned by LearnSPN as the hyperparameters for prior distributions. In the experiments we multiply the weights returned by LearnSPN by a positive scalar and treat them as the hyperparameters of the prior Dirichlet distributions. This is slightly better than using randomly initialized priors, but the differences are negligible on most data sets. MLE-SPN can also incorporate the model weights returned by LearnSPN by treating them as the hyperparameters of fixed prior Dirichlets. This corresponds to an MAP formulation. However in practice we find the MAP formulation performs no better than MLE-SPN, and on small data sets, MAP-SPN gives consistently worse results than MLE-SPN; so, we report only results for MLE-SPN. For both MLE-SPN and CVB-SPN we use a held-out validation set to pick the best solution during the optimization process. We report average log-likelihood on each data set for each method.

Both CVB-SPN and MLE-SPN are easily extended to the online setting where training instances are coming in a streaming fashion. In this case we also compare CVB-SPN and MLE-SPN to an online Bayesian moment matching method (OBMM) (Rashwan et al., 2016a) that is designed for learning SPNs. OBMM is a purely online algorithm in the sense that it can only process one instance in each update in order to avoid the exponential blow-up in the number of mixture components. OBMM constructs an approximate posterior distribution after seeing each instance by matching the first and second moments of the approximate posterior to the exact posterior. In this experiment, we compare both CVB-SPN (OCVB-SPN) and MLE-SPN (OMLE-SPN) to OBMM where only one pass over the training set is allowed for learning and at each round only one instance is available to each of the algorithms.

Table 10.4.2: Average log-likelihoods on test data. Highest average log-likelihoods are highlighted in bold. ↑ / ↓ are used to represent statistically better/worse results than (O)CVB-SPN respectively.

Data set	MLE-SPN	CVB-SPN	OBMM	OMLE-SPN	OCVB-SPN
NLTCS	↓ -6.44	**-6.08**	↑**-6.07**	↓-7.78	-6.12
MSNBC	↓-7.02	**-6.29**	-6.35	↓-6.94	**-6.34**
KDD 2k	↓-4.24	**-2.14**	**-2.14**	↓-27.99	-2.16
Plants	↓-28.78	**-12.86**	↑**-15.14**	↓-30.23	-16.03
Audio	↓-46.42	**-40.36**	↓-40.70	↓-48.90	**-40.58**
Jester	↓-59.55	**-54.26**	-53.86	↓-63.67	**-53.84**
Netflix	↓-64.88	**-60.69**	-57.99	↓-65.72	**-57.96**
Accidents	↓-50.14	**-29.55**	↓-42.66	↓-58.63	**-38.07**
Retail	↓-15.53	**-10.91**	↓-11.42	↓-82.42	**-11.31**
Pumsb-star	↓-80.61	**-25.93**	↓-45.27	↓-80.19	**-37.05**
DNA	↓-102.62	**-86.73**	↓-99.61	↓-96.84	**-91.52**
Kosarak	↓-47.16	**-10.70**	-11.22	↓-111.95	**-11.12**
MSWeb	↓-19.69	**-9.89**	↓-11.33	↓-140.86	**-10.73**
Book	↓-88.16	**-34.44**	↓-35.55	↓-299.02	**-34.77**
EachMovie	↓-97.15	**-52.63**	↑**-59.50**	↓-284.92	-64.75
WebKB	↓-199.15	**-161.46**	↑**-165.57**	↓-413.94	-169.31
Reuters-52	↓-218.97	**-85.45**	**-108.01**	↓-513.97	-108.04
20 Newsgrp	↓-260.69	**-155.61**	↓-158.01	↓-728.11	**-156.63**
BBC	↓-372.45	**-251.23**	↓-275.43	↓-517.36	**-272.56**
Ad	↓-311.87	**-19.00**	↓-63.81	↓-572.01	**-57.56**

10.4.2 Results

All experiments are run on a server with Intel Xeon CPU E5 2.00GHz. The running time ranges from 2 min to around 5 days depending on the size of the data set and the size of the network. All algorithms have roughly the same running time on the same data set as they scale linearly in the size of the training set and the size of the network. Table 10.4.2 shows the average joint log-likelihood scores of different parameter learning algorithms on 20 data sets. For each configuration, we use bold numbers to highlight the best score among the offline methods and among the online methods. We also use ↑ / ↓ to indicate whether the competitor methods achieve statistically significant better/worse results than CVB-SPN on the corresponding test data set under the Wilcoxon signed-rank test (Wilcoxon, 1950) with p-value ≤ 0.05. In the batch learning experiment, CVB-SPN consistently dominates MLE-SPN on every data set with a large margin. The same results can be observed in the online learning scenarios: both OCVB-SPN and OBMM significantly outperform OMLE-SPN on all the 20 experiments. These experiments demonstrate the effectiveness and robustness of Bayesian inference methods in parameter estimation on large graphical models like SPNs. In the online learning scenario, OCVB-SPN beats OBMM on 14 out of the 20 data sets, and achieves statistically better results on 10 out of the 14. On the other hand, OBMM obtains statistically better results than OCVB-SPN on 4 of the data sets. This is partly due to the fact that OCVB-SPN explicitly optimizes over an objective function that includes the evaluation of the variational posterior mean on the likelihood function, while OBMM only tries to match the first and second moments of the distribution after each update.

10.5 Conclusion

We develop a collapsed variational inference method, CVB-SPN, to learn the parameters of SPNs. CVB-SPN directly maintains a variational posterior distribution over the global latent variables by marginalizing out all the local latent variables. As a result, CVB-SPN is more memory efficient than standard VI. We also show that the collapsed ELBO in CVB-SPN is a better lower bound than the standard ELBO. We construct a logarithmic transformation trick to avoid the intractable computation of a high-dimensional expectation. We conduct experiments on 20 data sets to compare the proposed algorithm with state-of-the-art learning algorithms in both batch and online settings. The results demonstrate the effectiveness of CVB-SPN in both cases.

Chapter 11

Linear Time Computation of Moments

Bayesian online algorithms for Sum-Product Networks (SPNs) need to update their posterior distribution after seeing one single additional instance. To do so, they must compute moments of the model parameters under this distribution. The best existing method for computing such moments scales quadratically in the size of the SPN, although it scales linearly for trees. This unfortunate scaling makes Bayesian online algorithms prohibitively expensive, except for small or tree-structured SPNs. In this chapter we propose an optimal linear-time algorithm that works even when the SPN is a general directed acyclic graph (DAG), which significantly broadens the applicability of Bayesian online algorithms for SPNs. There are three key ingredients in the design and analysis of our algorithm:

1. For each edge in the graph, we construct a linear time reduction from the moment computation problem to a joint inference problem in SPNs.
2. Using the property that each SPN computes a multilinear polynomial, we give an efficient procedure for polynomial evaluation by differentiation without expanding the network that may contain exponentially many monomials.
3. We propose a dynamic programming method to further reduce the computation of the moments of all the edges in the graph from quadratic to linear.

We demonstrate the usefulness of our linear time algorithm by applying it to develop a linear time assume density filter (ADF) for SPNs.

11.1 Exact Posterior Has Exponentially Many Modes

Let m be the number of sum nodes in $\mathcal{S}$. Suppose we are given a fully factorized prior distribution $p_0(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{k=1}^{m} p_0(w_k \mid \alpha_k)$ over $\mathbf{w}$. It is worth pointing out the fully factorized prior distribution is well justified by the bipartite graph structure of the equivalent BN constructed from the SPN. We are interested in computing the moments of the posterior distribution after we receive one observation from the world. Essentially, this is the Bayesian online learning setting where we update the belief about the distribution of model parameters as we observe data from the world sequentially. Note that w_k corresponds to the weight vector associated with sum node k, so w_k is a vector that satisfies $w_k > 0$ and $\mathbf{1}^T w_k = 1$. Let us assume that the prior distribution for each w_k is Dirichlet, i.e.,

$$p_0(\mathbf{w} \mid \boldsymbol{\alpha}) = \prod_{k=1}^{m} \mathrm{Dir}(w_k \mid \alpha_k) = \prod_{k=1}^{m} \frac{\Gamma(\sum_j \alpha_{k,j})}{\prod_j \Gamma(\alpha_{k,j})} \prod_j w_{k,j}^{\alpha_{k,j}-1}$$

After observing one instance $\mathbf{x}$, we have the exact posterior distribution to be: $p(\mathbf{w} \mid \mathbf{x}) = p_0(\mathbf{w}; \boldsymbol{\alpha}) p(\mathbf{x} \mid \mathbf{w}) / p(\mathbf{x})$. Let $Z_{\mathbf{x}} \triangleq p(\mathbf{x})$ and realize that the network polynomial also computes the likelihood $p(\mathbf{x} \mid \mathbf{w})$. Plugging the expression for the prior distribution as well as the network polynomial into the above Bayes formula, we have:

$$p(\mathbf{w} \mid \mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{k=1}^{m} \mathrm{Dir}(w_k \mid \alpha_k) \prod_{(k,j) \in \mathcal{T}_{tE}} w_{k,j} \prod_{i=1}^{n} p_t(x_i)$$

Since Dirichlet is a conjugate distribution to the multinomial, each term in the summation is an updated Dirichlet with a multiplicative constant. So, the above equation suggests that the exact posterior distribution becomes a mixture of $\tau_{\mathcal{S}}$ Dirichlets after one observation. In a data set of D instances, the exact posterior will become a mixture of $\tau_{\mathcal{S}}^D$ components, which is intractable to maintain since $\tau_{\mathcal{S}} = \Omega(2^{H(\mathcal{S})})$.

The hardness of maintaining the exact posterior distribution appeals for an approximate scheme where we can sequentially update our belief about the distribution while at the same time efficiently maintain the approximation. Assumed density filtering (Sorenson and Stubberud, 1968) is such a framework: the algorithm chooses an approximate distribution from a tractable family of distributions after observing each instance. A typical choice is to match the moments of an approximation to the exact posterior.

11.2 The Hardness of Computing Moments

In order to find an approximate distribution to match the moments of the exact posterior, we need to be able to compute those moments under the exact posterior. This is not a problem for traditional mixture models including mixture of Gaussians, latent Dirichlet allocation, etc., since the number of mixture components in those models are assumed to be small constants. However, this is not the case for SPNs, where the effective number of mixture components is $\tau_{\mathcal{S}} = \Omega(2^{H(\mathcal{S})})$, which also depends on the input network $\mathcal{S}$.

To simplify the notation, for each $t \in [\tau_{\mathcal{S}}]$, we define $c_t := \prod_{i=1}^{n} p_t(x_i)$[1] and

$$u_t := \int_{\mathbf{w}} p_0(\mathbf{w}) \prod_{(k,j) \in \mathcal{T}_{tE}} w_{k,j} \, d\mathbf{w}.$$

That is, c_t corresponds to the product of leaf distributions in the tth induced tree $\mathcal{T}_t$, and u_t is the moment of $\prod_{(k,j) \in \mathcal{T}_{tE}} w_{k,j}$, i.e., the product of tree edges, under the prior distribution $p_0(\mathbf{w})$. Realizing that the

[1] For ease of notation, we omit the explicit dependency of c_t on the instance $\mathbf{x}$.

posterior distribution needs to satisfy the normalization constraint, we have:

$$\sum_{t=1}^{\tau_{\mathcal{S}}} c_t \int_{\mathbf{w}} p_0(\mathbf{w}) \prod_{(k,j)\in\mathcal{T}_{tE}} w_{k,j}\, d\mathbf{w} = \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t = Z_{\mathbf{x}} \tag{11.1}$$

Note that the prior distribution for a sum node is a Dirichlet distribution. In this case we can compute a closed form expression for u_t as:

$$u_t = \prod_{(k,j)\in\mathcal{T}_{tE}} \int_{w_k} p_0(w_k) w_{k,j}\, dw_k = \prod_{(k,j)\in\mathcal{T}_{tE}} \mathbb{E}_{p_0(w_k)}[w_{k,j}] = \prod_{(k,j)\in\mathcal{T}_{tE}} \frac{\alpha_{k,j}}{\sum_{j'} \alpha_{k,j'}} \tag{11.2}$$

More generally, let $f(\cdot)$ be a function applied to each edge weight in an SPN. We use the notation $M_p(f)$ to mean the moment of function f evaluated under distribution p. We are interested in computing $M_p(f)$ where $p = p(\mathbf{w} \mid \mathbf{x})$, which we call the *one-step update posterior distribution*. More specifically, for each edge weight $w_{k,j}$, we would like to compute the following quantity:

$$M_p(f(w_{k,j})) = \int_{\mathbf{w}} f(w_{k,j}) p(\mathbf{w} \mid \mathbf{x})\, d\mathbf{w} = \frac{1}{Z_{\mathbf{x}}} \sum_{t=1}^{\tau_{\mathcal{S}}} c_t \int_{\mathbf{w}} p_0(\mathbf{w}) f(w_{k,j}) \prod_{(k',j')\in\mathcal{T}_{tE}} w_{k',j'}\, d\mathbf{w} \tag{11.3}$$

We note that (11.3) is not trivial to compute as it involves $\tau_{\mathcal{S}} = \Omega(2^{H(\mathcal{S})})$ terms. Furthermore, in order to conduct moment matching, we need to compute the above moment for each edge (k, j) from a sum node. A naive computation will lead to a total time complexity $\Omega(|\mathcal{S}| \cdot 2^{H(\mathcal{S})})$. A linear time algorithm to compute these moments has been designed by Rashwan et al. (2016b) when the underlying structure of $\mathcal{S}$ is a tree. This algorithm recursively computes the moments in a top-down fashion along the tree. However, this algorithm breaks down when the graph is a DAG.

In what follows we will present a $O(|\mathcal{S}|)$ time and space algorithm that is able to compute all the moments simultaneously for general SPNs with DAG structures. We will first show a linear time reduction from the moment computation in (11.3) to a joint inference problem in $\mathcal{S}$, and then proceed to use the differential trick to efficiently compute (11.3) for each edge in the graph. The final component will be a dynamic program to simultaneously compute (11.3) for all edges $w_{k,j}$ in the graph by trading constant factors of space complexity to reduce time complexity.

11.3 Linear Time Reduction from Moment Computation to Joint Inference

Let us first compute (11.3) for a fixed edge (k, j). Our strategy is to partition all the induced trees based on whether they contain the tree edge (k, j) or not. Define $\mathcal{T}_F = \{\mathcal{T}_t \mid (k, j) \notin \mathcal{T}_t, t \in [\tau_{\mathcal{S}}]\}$ and $\mathcal{T}_T = \{\mathcal{T}_t \mid (k, j) \in \mathcal{T}_t, t \in [\tau_{\mathcal{S}}]\}$. In other words, $\mathcal{T}_F$ corresponds to the set of trees that do not contain edge (k, j) and $\mathcal{T}_T$ corresponds to the set of trees that contain edge (k, j). Then,

$$\begin{aligned} M_p(f(w_{k,j})) = &\frac{1}{Z_{\mathbf{x}}} \sum_{\mathcal{T}_t\in\mathcal{T}_T} c_t \int_{\mathbf{w}} p_0(\mathbf{w}) f(w_{k,j}) \prod_{(k',j')\in\mathcal{T}_{tE}} w_{k',j'}\, d\mathbf{w} \\ &+ \frac{1}{Z_{\mathbf{x}}} \sum_{\mathcal{T}_t\in\mathcal{T}_F} c_t \int_{\mathbf{w}} p_0(\mathbf{w}) f(w_{k,j}) \prod_{(k',j')\in\mathcal{T}_{tE}} w_{k',j'}\, d\mathbf{w} \end{aligned} \tag{11.4}$$

For the induced trees that contain edge (k, j), we have

$$\frac{1}{Z_{\mathbf{x}}}\sum_{\mathcal{T}_t\in\mathcal{T}_T} c_t \int_{\mathbf{w}} p_0(\mathbf{w}) f(w_{k,j}) \prod_{(k',j')\in\mathcal{T}_{tE}} w_{k',j'}\, d\mathbf{w} = \frac{1}{Z_{\mathbf{x}}}\sum_{\mathcal{T}_t\in\mathcal{T}_T} c_t u_t M_{p'_{0,k}}(f(w_{k,j})) \tag{11.5}$$

where $p'_{0,k}$ is the one-step update posterior Dirichlet distribution for sum node k after absorbing the term $w_{k,j}$. Similarly, for the induced trees that do not contain the edge (k, j):

$$\frac{1}{Z_{\mathbf{x}}}\sum_{\mathcal{T}_t\in\mathcal{T}_F} c_t \int_{\mathbf{w}} p_0(\mathbf{w}) f(w_{k,j}) \prod_{(k',j')\in\mathcal{T}_{tE}} w_{k',j'}\, d\mathbf{w} = \frac{1}{Z_{\mathbf{x}}}\sum_{\mathcal{T}_t\in\mathcal{T}_F} c_t u_t M_{p_{0,k}}(f(w_{k,j})) \tag{11.6}$$

where $p_{0,k}$ is the prior Dirichlet distribution for sum node k. The above equation holds by changing the order of integration and realize that since (k, j) is not in tree $\mathcal{T}_t$, $\prod_{(k',j')\in\mathcal{T}_{tE}} w_{k',j'}$ does not contain the term $w_{k,j}$. Note that both $M_{p_{0,k}}(f(w_{k,j}))$ and $M_{p'_{0,k}}(f(w_{k,j}))$ are independent of specific induced trees, so we can combine the above two parts to express $M_p(f(w_{k,j}))$ as:

$$M_p(f(w_{k,j})) = \left(\frac{1}{Z_{\mathbf{x}}}\sum_{\mathcal{T}_t\in\mathcal{T}_F} c_t u_t\right) M_{p_{0,k}}(f(w_{k,j})) + \left(\frac{1}{Z_{\mathbf{x}}}\sum_{\mathcal{T}_t\in\mathcal{T}_T} c_t u_t\right) M_{p'_{0,k}}(f(w_{k,j})) \tag{11.7}$$

From (11.1) we have

$$\frac{1}{Z_{\mathbf{x}}}\sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t = 1 \quad \text{and} \quad \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t = \sum_{\mathcal{T}_t\in\mathcal{T}_T} c_t u_t + \sum_{\mathcal{T}_t\in\mathcal{T}_F} c_t u_t$$

This implies that $M_p(f)$ is in fact a convex combination of $M_{p_{0,k}}(f)$ and $M_{p'_{0,k}}(f)$. In other words, since both $M_{p_{0,k}}(f)$ and $M_{p'_{0,k}}(f)$ can be computed in closed form for each edge (k, j), so in order to compute (11.3), we only need to be able to compute the two coefficients efficiently. Recall that for each induced tree $\mathcal{T}_t$, we have the expression of u_t as $u_t = \prod_{(k,j)\in\mathcal{T}_{tE}} \alpha_{k,j} / \sum_{j'} \alpha_{k,j'}$. So the term $\sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t$ can thus be expressed as:

$$\sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t = \sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} \frac{\alpha_{k,j}}{\sum_{j'} \alpha_{k,j'}} \prod_{i=1}^{n} p_t(x_i) \tag{11.8}$$

The key observation that allows us to find the linear time reduction lies in the fact that (11.8) shares exactly the same functional form as the network polynomial, with the only difference being the specification of edge weights in the network. The following lemma formalizes our argument.

Lemma 11.3.1. $\sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t$ can be computed in $O(|\mathcal{S}|)$ time and space in a bottom-up evaluation of $\mathcal{S}$.

Proof. Compare the form of (11.8) to the network polynomial:

$$p(\mathbf{x} \mid \mathbf{w}) = V_{\text{root}}(\mathbf{x} \mid \mathbf{w}) = \sum_{t=1}^{\tau_{\mathcal{S}}} \prod_{(k,j)\in\mathcal{T}_{tE}} w_{k,j} \prod_{i=1}^{n} p_t(x_i) \tag{11.9}$$

Clearly (11.8) and (11.9) share the same functional form and the only difference lies in that the edge weight used in (11.8) is given by $\alpha_{k,j} / \sum_{j'} \alpha_{k,j'}$ while the edge weight used in (11.9) is given by $w_{k,j}$, both of which are constrained to be positive and locally normalized. This means that in order to compute the value of (11.8), we can replace all the edge weights $w_{k,j}$ with $\alpha_{k,j} / \sum_{j'} \alpha_{k,j'}$, and a bottom-up pass evaluation of $\mathcal{S}$ will give us the desired result at the root of the network. The linear time and space complexity then follows from the linear time and space inference complexity of SPNs. ■

In other words, we reduce the original moment computation problem for edge (k, j) to a joint inference problem in $\mathcal{S}$ with a set of weights determined by $\boldsymbol{\alpha}$.

11.4 Efficient Polynomial Evaluation by Differentiation

To evaluate (11.7), we also need to compute $\sum_{\mathcal{T}_t \in \mathcal{T}_T} c_t u_t$ efficiently, where the sum is over a subset of induced trees that contain edge (k, j). Again, due to the exponential lower bound on the number of unique induced trees, a brute force computation is infeasible in the worst case. The key observation is that we can use the *differential trick* to solve this problem by realizing the fact that $Z_{\mathbf{x}} = \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t$ is a multilinear function in $\alpha_{k,j} / \sum_{j'} \alpha_{k,j'}, \forall k, j$ and it has a tractable circuit representation since it shares the same network structure with $\mathcal{S}$.

Lemma 11.4.1. $\sum_{\mathcal{T}_t \in \mathcal{T}_T} c_t u_t = w_{k,j} \left(\partial \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t / \partial w_{k,j} \right)$, and it can be computed in $O(|\mathcal{S}|)$ time and space in a top-down differentiation of $\mathcal{S}$.

Proof. Define $w_{k,j} \triangleq \alpha_{k,j} / \sum_{j'} \alpha_{k,j'}$, then

$$\begin{aligned}
\sum_{\mathcal{T}_t \in \mathcal{T}_T} c_t u_t &= \sum_{\mathcal{T}_t \in \mathcal{T}_T} \prod_{(k',j') \in \mathcal{T}_{tE}} w_{k',j'} \prod_{i=1}^{n} p_t(x_i) \\
&= w_{k,j} \sum_{\mathcal{T}_t \in \mathcal{T}_T} \prod_{\substack{(k',j') \in \mathcal{T}_{tE} \\ (k',j') \neq (k,j)}} w_{k',j'} \prod_{i=1}^{n} p_t(x_i) + 0 \cdot \sum_{\mathcal{T}_t \in \mathcal{T}_F} c_t u_t \\
&= w_{k,j} \left(\frac{\partial}{\partial w_{k,j}} \sum_{\mathcal{T}_t \in \mathcal{T}_T} c_t u_t + \frac{\partial}{\partial w_{k,j}} \sum_{\mathcal{T}_t \in \mathcal{T}_F} c_t u_t \right) = w_{k,j} \left(\frac{\partial}{\partial w_{k,j}} \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t \right)
\end{aligned}$$

where the second equality is by the observation that the network polynomial is a multilinear function of $w_{k,j}$ and the third equality holds because $\mathcal{T}_F$ is the set of trees that do not contain $w_{k,j}$. The last equality follows by simple algebraic transformations. In summary, the above lemma holds because of the fact that differential operator applied to a multilinear function acts as a selector for all the monomials containing a specific variable. Hence, $\sum_{\mathcal{T}_t \in \mathcal{T}_F} c_t u_t = \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t - \sum_{\mathcal{T}_t \in \mathcal{T}_T} c_t u_t$ can also be computed. To show the linear time and space complexity, recall that the differentiation w.r.t.$w_{k,j}$ can be efficiently computed by back-propagation in a top-down pass of $\mathcal{S}$ once we have computed $\sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t$ in a bottom-up pass of $\mathcal{S}$. ■

Remark The fact that we can compute the differentiation w.r.t. $w_{k,j}$ using the original circuit without expanding it underlies many recent advances in the algorithmic design of SPNs. Zhao et al. (2016a,b) used the above differential trick to design linear time collapsed variational algorithm and the concave-convex produce for parameter estimation in SPNs. A different but related approach, where the differential operator is taken w.r.t. input indicators, not model parameters, is applied in computing the marginal probability in Bayesian networks and junction trees (Darwiche, 2003; Park and Darwiche, 2004). We finish this discussion by concluding that when the polynomial computed by the network is a multilinear function in terms of model parameters or input indicators (such as in SPNs), then the differential operator w.r.t. a variable can be used as an efficient way to compute the sum of the subset of monomials that contain the specific variable.

11.5 Dynamic Programming: from Quadratic to Linear

Define $D_k(\mathbf{x} \mid \mathbf{w}) = \partial V_{\text{root}}(\mathbf{x} \mid \mathbf{w}) / \partial V_k(\mathbf{x} \mid \mathbf{w})$. Then the differentiation term $\partial \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t / \partial w_{k,j}$ in Lemma 11.4.1 can be computed via back-propagation in a top-down pass of the network as follows:

$$\frac{\partial \sum_{t=1}^{\tau_{\mathcal{S}}} c_t u_t}{\partial w_{k,j}} = \frac{\partial V_{\text{root}}(\mathbf{x} \mid \mathbf{w})}{\partial V_k(\mathbf{x} \mid \mathbf{w})} \frac{\partial V_k(\mathbf{x} \mid \mathbf{w})}{\partial w_{k,j}} = D_k(\mathbf{x} \mid \mathbf{w}) V_j(\mathbf{x} \mid \mathbf{w}) \tag{11.10}$$

Let $\lambda_{k,j} = \left(w_{k,j}V_j(\mathbf{x} \mid \mathbf{w})D_k(\mathbf{x} \mid \mathbf{w})\right) / V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$ and $f_{k,j} = f(w_{k,j})$, then the final formula for computing the moment of edge weight $w_{k,j}$ under the one-step update posterior p is given by

$$M_p(f_{k,j}) = \left(1 - \lambda_{k,j}\right) M_{p_0}(f_{k,j}) + \lambda_{k,j} M_{p_0'}(f_{k,j}) \tag{11.11}$$

Corollary 11.5.1. For each edges (k, j), (11.7) can be computed in $O(|\mathcal{S}|)$ time and space.

The corollary simply follows from Lemma 11.3.1 and Lemma 11.4.1 with the assumption that the moments under the prior has closed form solution. By definition, we also have $\lambda_{k,j} = \sum_{\mathcal{T}_t \in \mathcal{T}_T} c_t u_t / Z_{\mathbf{x}}$, hence $0 \leq \lambda_{k,j} \leq 1, \forall (k, j)$. This formula shows that $\lambda_{k,j}$ computes the ratio of all the induced trees that contain edge (k, j) to the network. Roughly speaking, this measures how important the contribution of a specific edge is to the whole network polynomial. As a result, we can interpret (11.11) as follows: the more important the edge is, the more portion of the moment comes from the new observation.

CCCP for SPNs was originally derived using a sequential convex relaxation technique, where in each iteration a concave surrogate function is constructed and optimized. The key update in each iteration of CCCP (Zhao et al. (2016b), (7)) is given as follows: $w_{k,j}' \propto w_{k,j}V_j(\mathbf{x} \mid \mathbf{w})D_k(\mathbf{x} \mid \mathbf{w}) / V_{\text{root}}(\mathbf{x} \mid \mathbf{w})$, where the R.H.S. is exactly the same as $\lambda_{k,j}$ defined above. From this perspective, CCCP can also be understood as implicitly applying the differential trick to compute $\lambda_{k,j}$, i.e., the relative importance of edge (k, j), and then take updates according to this importance measure.

In order to compute the moments of all the edge weights $w_{k,j}$, a naive computation would scale $O(|\mathcal{S}|^2)$ because there are $O(|\mathcal{S}|)$ edges in the graph and from Cor. 11.5.1 each such computation takes $O(|\mathcal{S}|)$ time. The key observation that allows us to further reduce the complexity to linear comes from the structure of $\lambda_{k,j}$: $\lambda_{k,j}$ only depends on three terms, i.e., the forward evaluation value $V_j(\mathbf{x} \mid \mathbf{w})$, the backward differentiation value $D_k(\mathbf{x} \mid \mathbf{w})$ and the original weight of the edge $w_{k,j}$. This implies that we can use dynamic programming to cache both $V_j(\mathbf{x} \mid \mathbf{w})$ and $D_k(\mathbf{x} \mid \mathbf{w})$ in a bottom-up evaluation pass and a top-down differentiation pass, respectively. At a high level, we trade off a constant factor in space complexity (using two additional copies of the network) to reduce the quadratic time complexity to linear.

Theorem 11.5.1. For all edges (k, j), (11.7) can be computed in $O(|\mathcal{S}|)$ time and space.

Proof. During the bottom-up evaluation pass, in order to compute the value $V_{\text{root}}(\mathbf{x}; \mathbf{w})$ at the root of $\mathcal{S}$, we will also obtain all the values $V_j(\mathbf{x}; \mathbf{w})$ at each node j in the graph. So instead of discarding these intermediate $V_j(\mathbf{x}; \mathbf{w})$, we cache them by allocating additional space at each node j. So after one bottom-up evaluation pass of the network, we will also have all the $V_j(\mathbf{x}; \mathbf{w})$ for each node j, at the cost of one additional copy of the network. Similarly, during the top-down differentiation pass of the network, because of the chain rule, we will also obtain all the intermediate $D_k(\mathbf{x}; \mathbf{w})$ at each node k. Again, we cache them. Once we have both $V_j(\mathbf{x}; \mathbf{w})$ and $D_k(\mathbf{x}; \mathbf{w})$ for each edge (k, j), from (11.11), we can get all the moments for all the weighted edges in $\mathcal{S}$ simultaneously. Because the whole process only requires one bottom-up evaluation pass and one top-down differentiation pass of $\mathcal{S}$, the time complexity is $2|\mathcal{S}|$. Since we use two additional copies of $\mathcal{S}$, the space complexity is $3|\mathcal{S}|$. ■

We summarize the linear time algorithm for moment computation in Alg. 4.

11.6 Applications in Online Moment Matching

In this section we use Alg. 4 as a sub-routine to develop a new Bayesian online learning algorithm for SPNs based on assumed density filtering (Sorenson and Stubberud, 1968). To do so, we find an approximate distribution by minimizing the KL divergence between the one-step update posterior and the approximate

Algorithm 4 Linear Time Exact Moment Computation

```
Input: Prior $p_0(\mathbf{w} \mid \boldsymbol{\alpha})$, moment $f$, SPN $\mathcal{S}$ and input $\mathbf{x}$.
Output: $M_p(f(w_{k,j})), \forall(k,j)$.
1: $w_{k,j} \leftarrow \alpha_{k,j} / \sum_{j'} \alpha_{k,j'}, \forall(k,j)$.
2: Compute $M_{p_0}(f(w_{k,j}))$ and $M_{p'_0}(f(w_{k,j})), \forall(k,j)$.
3: Bottom-up evaluation pass of $\mathcal{S}$ with input $\mathbf{x}$. Record $V_k(\mathbf{x};\mathbf{w})$ at each node $k$.
4: Top-down differentiation pass of $\mathcal{S}$ with input $\mathbf{x}$. Record $D_k(\mathbf{x};\mathbf{w})$ at each node $k$.
5: Compute the exact moment for each $(k,j)$: $M_p(f_{k,j}) = \left(1-\lambda_{k,j}\right) M_{p_0}(f_{k,j}) + \lambda_{k,j} M_{p'_0}(f_{k,j})$.
```

distribution. Let $\mathcal{P} = \{q \mid q = \prod_{k=1}^m \mathrm{Dir}(w_k;\beta_k)\}$, i.e., $\mathcal{P}$ is the space of product of Dirichlet densities that are decomposable over all the sum nodes in $\mathcal{S}$. Note that since $p_0(\mathbf{w};\boldsymbol{\alpha})$ is fully decomposable, we have $p_0 \in \mathcal{P}$. One natural choice is to try to find an approximate distribution $q \in \mathcal{P}$ such that q minimizes the KL-divergence between $p(\mathbf{w}|\mathbf{x})$ and q, i.e.,

$$\hat{p} = \operatorname*{arg\,min}_{q \in \mathcal{P}} D_{\mathrm{KL}}(p(\mathbf{w} \mid \mathbf{x}) \parallel q).$$

It is not hard to show that when q is an exponential family distribution, which is the case in our setting, the minimization problem corresponds to solving the following moment matching equation:

$$\mathbb{E}_{p(\mathbf{w}|\mathbf{x})}[T(w_k)] = \mathbb{E}_{q(\mathbf{w})}[T(w_k)] \tag{11.12}$$

where $T(w_k)$ is the vector of sufficient statistics of $q(w_k)$. When $q(\cdot)$ is a Dirichlet, we have $T(w_k) = \log w_k$, where the log is understood to be taken elementwise. This principle of finding an approximate distribution is also known as *reverse information projection* in the literature of information theory (**?**). As a comparison, information projection corresponds to minimizing $D_{\mathrm{KL}}(q \parallel p(\mathbf{w} \mid \mathbf{x}))$ within the same family of distributions $q \in \mathcal{P}$. By utilizing our efficient linear time algorithm for exact moment computation, we propose a Bayesian online learning algorithm for SPNs based on the above moment matching principle, called assumed density filtering (ADF). The pseudocode is shown in Alg. 5.

In the ADF algorithm, for each edge $w_{k,j}$ the above moment matching equation amounts to solving the following equation:

$$\psi(\beta_{k,j}) - \psi(\sum_{j'} \beta_{k,j'}) = \mathbb{E}_{p(\mathbf{w}|\mathbf{x})}[\log w_{k,j}]$$

where $\psi(\cdot)$ is the digamma function. This is a system of nonlinear equations about β where the R.H.S. of the above equation can be computed using Alg. 4 in $O(|\mathcal{S}|)$ time for all the edges (k,j). To efficiently solve it, we take $\exp(\cdot)$ at both sides of the equation and approximate the L.H.S. using the fact that $\exp(\psi(\beta_{k,j})) \approx \beta_{k,j} - \frac{1}{2}$ for $\beta_{k,j} > 1$. Expanding the R.H.S. of the above equation using the identity from (11.11), we have:

$$\exp\left(\psi(\beta_{k,j}) - \psi(\sum_{j'} \beta_{w,j'})\right) = \exp\left(\mathbb{E}_{p(\mathbf{w}|\mathbf{x})}[\log w_{k,j}]\right)$$

$$\Leftrightarrow \frac{\beta_{k,j} - \frac{1}{2}}{\sum_{j'} \beta_{k,j'} - \frac{1}{2}} = \left(\frac{\alpha_{k,j} - \frac{1}{2}}{\sum_{j'} \alpha_{k,j'} - \frac{1}{2}}\right)^{(1-\lambda_{k,j})} \times \left(\frac{\alpha_{k,j} + \frac{1}{2}}{\sum_{j'} \alpha_{k,j'} + \frac{1}{2}}\right)^{\lambda_{k,j}} \tag{11.13}$$

Note that $(\alpha_{k,j} - 0.5)/(\sum_{j'} \alpha_{k,j'} - 0.5)$ is approximately the mean of the prior Dirichlet under p_0 and $(\alpha_{k,j} + 0.5)/(\sum_{j'} \alpha_{k,j'} + 0.5)$ is approximately the mean of p'_0, where p'_0 is the posterior by adding one

Algorithm 5 Assumed Density Filtering for SPN

```
Input: Prior $p_0(\mathbf{w} \mid \boldsymbol{\alpha})$, SPN $\mathcal{S}$ and input $\{\mathbf{x}_i\}_{i=1}^{\infty}$.
$p(\mathbf{w}) \leftarrow p_0(\mathbf{w} \mid \boldsymbol{\alpha})$
for $i = 1, \dots, \infty$ do
    Apply Alg. 4 to compute $\mathbb{E}_{p(\mathbf{w}|\mathbf{x}_i)}[\log w_{k,j}]$ for all edges $(k, j)$.
    Find $\hat{p} = \arg\min_{q \in \mathcal{P}} D_{\mathrm{KL}}(p(\mathbf{w} \mid \mathbf{x}_i) \parallel q)$ by solving the moment matching equation (11.12).
    $p(\mathbf{w}) \leftarrow \hat{p}(\mathbf{w})$.
end for
```

pseudo-count to $w_{k,j}$. So (11.13) is essentially finding a posterior with hyperparameter β such that the posterior mean is approximately the weighted geometric mean of the means given by p_0 and p'_0, weighted by $\lambda_{k,j}$.

Instead of matching the moments given by the sufficient statistics, also known as the natural moments, BMM tries to find an approximate distribution q by matching the first order moments, i.e., the mean of the prior and the one-step update posterior. Using the same notation, we want q to match the following equation:

$$\mathbb{E}_{q(\mathbf{w})}[w_k] = \mathbb{E}_{p(\mathbf{w}|\mathbf{x})}[w_k] \quad \Leftrightarrow \quad \frac{\beta_{k,j}}{\sum_{j'} \beta_{k,j'}} = (1 - \lambda_{k,j}) \frac{\alpha_{k,j}}{\sum_{j'} \alpha_{k,j'}} + \lambda_{k,j} \frac{\alpha_{k,j} + 1}{\sum_{j'} \alpha_{k,j'} + 1} \tag{11.14}$$

Again, we can interpret the above equation as to find the posterior hyperparameter β such that the posterior mean is given by the weighted arithmetic mean of the means given by p_0 and p'_0, weighted by $\lambda_{k,j}$. Notice that due to the normalization constraint, we cannot solve for β directly from the above equations, and in order to solve for β we will need one more equation to be added into the system. However, from line 1 of Alg. 4, what we need in the next iteration of the algorithm is not β, but only its normalized version. So we can get rid of the additional equation and use (11.14) as the update formula directly in our algorithm.

Using Alg. 4 as a sub-routine, both ADF and BMM enjoy linear running time, sharing the same order of time complexity as CCCP. However, since CCCP directly optimizes over the data log-likelihood, in practice we observe that CCCP often outperforms ADF and BMM in log-likelihood scores.

11.7 Conclusion

In this chapter we propose an optimal linear time algorithm to efficiently compute the moments of model parameters in SPNs under online settings. The key techniques used in the design of our algorithm include the liner time reduction from moment computation to joint inference, the differential trick that is able to efficiently evaluate a multilinear function, and the dynamic programming to further reduce redundant computations. Using the proposed algorithm as a sub-routine, we are able to improve the time complexity of BMM from quadratic to linear on general SPNs with DAG structures. We also use the proposed algorithm as a sub-routine to design a new online algorithm, ADF. As a future direction, we hope to apply the proposed moment computation algorithm in the design of efficient structure learning algorithms for SPNs. We also expect that the analysis techniques we develop might find other uses for learning SPNs.

Part III

Conclusion

Chapter 12

Conclusion and Future Work

My thesis research contributes to two main themes in artificial intelligence: invariant representation learning and tractable probabilistic reasoning. Moving forward, I will continue working along these two themes towards the long-term goal of building a unified framework that provides a common semantics for learning and reasoning, and also branch out to explore applications related to algorithmic fairness and multilingual natural language understanding.

12.1 Information Analysis of Invariant Representation Learning

Invariant representation learning has abundant applications in domain adaptation, algorithmic fairness and privacy-preservation under attribute-inference attacks. Recently, the idea of learning language-invariant representations has also been actively explored in neural machine translation in order to enable knowledge transfer from high-resource language pairs to low-resource language pairs. Despite its broad applications, many fundamental questions remain open. Our work (Zhao and Gordon, 2019; Zhao et al., 2019b,e) has shown that utility has lower bound if exact invariance is attained. However, it is not clear what is the general form of tradeoff between utility and invariance. In particular, under a budget for approximate invariance, what is the maximum utility we can hope to achieve? This question calls for a characterization of the Pareto frontier between utility and invariance. In my future research, I want to apply tools from information theory to provide a precise answer to the above question, and to use the theory of invariant representation learning to design Pareto-optimal algorithms in the above mentioned applications.

12.2 Efficient Structure Learning of Probabilistic Circuits

SPNs distinguish themselves from other probabilistic graphical models, including both Bayesian Networks and Markov Networks, by the fact that reasoning can be performed exactly in linear time with respect to the size of the network. Similar to traditional graphical models, there are two main problems when learning SPNs: structure learning and parameter learning. In structure learning the goal is to infer the structure of SPNs directly from the data. As a direction for future work, I am highly interested in developing principled structure learning algorithms that can produce compact SPNs with directed acyclic structures. To date, most structure learning algorithms can only produce SPNs with tree structures, and they are based on various kinds of heuristics to generate SPNs from data without performance guarantees. In light of the existing limitations, it is favorable to come up with an algorithm that is able to produce SPNs with directed acyclic structures to fully exploit their representational power. I am also excited about extending the domain of SPNs from discrete/continuous data to more structured ones, e.g., string, graph, etc., and apply them to problems that require the capability of reasoning, including *question answering*, *reading comprehension* and *statistical relation learning over graphs*.

12.3 Unified Framework for Learning and Reasoning

I believe the holy grail of artificial intelligence is to build intelligent agents that have the ability to learn from the experience as well as to reason from what has been learned. In order to achieve this goal, we need to have a robust and probabilistic framework to unify learning and reasoning. Such framework is drastically different from the traditional one where symbolic representations are used to construct the knowledge base and first-order logic is used to build the inference engine. Instead, as shown in Figure 1.1, I propose to use invariant representation that maps real-world objects to their corresponding algebraic representations to serve as the foundation of knowledge base, and to use tractable probabilistic inference machine, e.g., Sum-Product networks, to act as the inference engine. Compared with the classic symbolic and logic-based framework, such new framework is inherently probabilistic and hence can handle the ubiquitous uncertainty. In particular, representations that are invariant to the change in the environment can provide us the robustness against various noise and nuisance factors in real-world, and the tractability of exact probabilistic inference machine can further allow us to efficiently deal with the uncertainty existing in real-world logic deduction.

Of course, the goal is challenging. First, in order to learn invariant representations, one needs to explicitly specify a set of nuisance factors that the representations should be invariant to. Due to the complexity of the real-world, such supervision is not always available or well-defined. Furthermore, when the invariant representations contain some internal structures, e.g., the hierarchical structure of sentence representations, it is not clear how to combine such structured data with existing tractable probabilistic inference machines. These problems are both fascinating and challenging, and I believe that being able to solve them could take us a significant step towards the goal of a unified framework for learning and reasoning.

Bibliography

Tameem Adel, Han Zhao, and Alexander Wong. Unsupervised domain adaptation with a relaxed covariate shift assumption. In *AAAI*, pages 1691–1697, 2017. 1.2.1, 1.5, 3.1, 4.1

Tameem Adel, Isabel Valera, Zoubin Ghahramani, and Adrian Weller. One-network adversarial fairness. In *33rd AAAI Conference on Artificial Intelligence*, 2019a. 6.3.2, 6.5, 7.2, 7.5

Tameem Adel, Han Zhao, and Richard E Turner. Continual learning with adaptive weights (claw). In *International Conference on Learning Representations*, 2019b. 1.5

Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. A reductions approach to fair classification. In *International Conference on Machine Learning*, pages 60–69, 2018. 7.5

Roee Aharoni, Melvin Johnson, and Orhan Firat. Massively multilingual neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3874–3884, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1388. 8.4

Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. Domain-adversarial neural networks. *arXiv preprint arXiv:1412.4446*, 2014. 3.1, 3.6, 4.1, 4.2.2, 4.3

Syed Mumtaz Ali and Samuel D Silvey. A general class of coefficients of divergence of one distribution from another. *Journal of the Royal Statistical Society: Series B (Methodological)*, 28(1):131–142, 1966. 6.2

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009. 3.7.3, 8.3.2

Pablo Arbelaez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):898–916, 2011. 3.B

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, et al. Massively multilingual neural machine translation in the wild: Findings and challenges. *arXiv preprint arXiv:1907.05019*, 2019. 8.1, 8.4

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2019. URL `http://arxiv.org/abs/1907.02893`. cite arxiv:1907.02893. 5.1

Stefan Arnborg, Derek G Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in ak-tree. *SIAM Journal on Algebraic Discrete Methods*, 8(2):277–284, 1987. 2.4

Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610,

2019. 8.1, 8.2.2, 8.4

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of ACL 2017*, pages 451–462, 2017. 8.4

Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. 2018. 4.3

Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *CoRR*, abs/1906.00910, 2019. 5.1

R Iris Bahar, Erica A Frohm, Charles M Gaona, Gary D Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Algebraic decision diagrams and their applications. *Formal methods in system design*, 10(2-3): 171–206, 1997. 2.4

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. URL `http://arxiv.org/abs/1409.0473`. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. 8.4

Mahsa Baktashmotlagh, Mehrtash T Harandi, Brian C Lovell, and Mathieu Salzmann. Unsupervised domain adaptation by domain invariant projection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 769–776, 2013. 3.6

Solon Barocas and Andrew D Selbst. Big data's disparate impact. *Calif. L. Rev.*, 104:671, 2016. 6.1

Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *NIPS Tutorial*, 2017. 7.1, 7.3.2, 7.3.3

Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002. 2.6.1

PeterL Bartlett. Thesamplecomplexityofp atternclassification withneuralnetworks: Thesizeoftheweightsismo reimportantthan thesizeofthenetwork. *IEEETrans. Inf. Theory*, 44(2), 1998. 8.3.3

Carlos J Becker, Christos M Christoudias, and Pascal Fua. Non-linear domain adaptation with boosting. In *Advances in Neural Information Processing Systems*, pages 485–493, 2013. 3.1, 3.6, 4.1

Shai Ben-David, Nadav Eiron, and Philip M Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003. 7.3.4

Shai Ben-David, John Blitzer, Koby Crammer, Fernando Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007. 2.2, 2.2, 3.2, 3.2, 3.4, 3.5.3, 3.6, 4.2, 1, 4.2.2, 4.2.1, 4.3, 5.2

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. 2.2, 3.1, 3.2, 3.3, 3.6, 3.7.3, 4.1, 4.2, 4.2.2, 4.3, 5.2, 5.2, 5.3.2

Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, page 0049124118782533, 2018. 6.1

Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *arXiv preprint arXiv:1707.00075*, 2017. 6.1, 6.3.2, 6.4, 6.5, 7.1, 7.5

David M Blei, Michael I Jordan, and John W Paisley. Variational Bayesian Inference with Stochastic Search. In *ICML*, 2012. 1.3.2, 10.3

John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136, 2008. 2.2, 2.2, 2.2.1, 3.1, 3.2, 3.2, 3.2.1, 3.3, 3.6, 4.2

George Boole. *The mathematical analysis of logic*. Philosophical Library, 1847. 2.4

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016. 3.6, 4.4.3, 5.3.1

Stephen Boyd, Seung-Jean Kim, Lieven Vandenberghe, and Arash Hassibi. A tutorial on geometric programming. *Optimization and Engineering*, 8(1):67–127, 2007. 2.5, 2.5, 9.2, 10.3, 10.3

Jop Briët and Peter Harremoës. Properties of classical and quantum jensen-shannon divergence. *Phys. Rev. A*, 79:052311, May 2009. doi: 10.1103/PhysRevA.79.052311. URL `https://link.aps.org/doi/10.1103/PhysRevA.79.052311`. 5.5.8

Toon Calders and Sicco Verwer. Three naive bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery*, 21(2):277–292, 2010. 7.2

Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. Building classifiers with independency constraints. In *2009 IEEE International Conference on Data Mining Workshops*, pages 13–18. IEEE, 2009. 1.2.2, 2.3, 6.1, 6.2, 6.3, 7.1, 7.2

Hei Chan and Adnan Darwiche. On the robustness of most probable explanations. In *In Proceedings of the Twenty Second Conference on Uncertainty in Artificial Intelligence*. 9.1

Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. Marginalized denoising autoencoders for domain adaptation. *arXiv preprint arXiv:1206.4683*, 2012. 3.5, 3.5.1, 3.6, 1, 3.A

Xilun Chen and Claire Cardie. Multinomial adversarial networks for multi-domain text classification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, volume 1, 2018. 1.1

Mung Chiang. *Geometric programming for communication systems*. Now Publishers Inc, 2005. 2.5, 9.2, 10.3

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 8.3

Arthur Choi and Adnan Darwiche. On relaxing determinism in arithmetic circuits. In *International Conference on Machine Learning*, pages 825–833, 2017. 2.4

Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017. 2.3, 2.3, 2.3.1, 6.2, 6.2, 6.2.1, 7.2, 7.5

Alexandra Chouldechova and Aaron Roth. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*, 2018. 7.1

Trevor Cohn and Mirella Lapata. Machine translation by triangulation: Making effective use of multi-parallel corpora. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 728–735, Prague, Czech Republic, June 2007. Association for Computational Linguistics. 8.4

Remi Tachet des Combes, Han Zhao, Yu-Xiang Wang, and Geoff Gordon. Domain adaptation with conditional distribution matching and generalized label shift. *arXiv preprint arXiv:2003.04475*, 2020. 8.4

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. Word translation without parallel data. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*, 2018a. 8.4

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, October-November 2018b. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. 8.1

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019. 8.4

Sam Corbett-Davies and Sharad Goel. The measure and mismeasure of fairness: A critical review of fair machine learning. *arXiv preprint arXiv:1808.00023*, 2018. 7.1

Corinna Cortes and Mehryar Mohri. Domain adaptation and sample bias correction theory and algorithm for regression. *Theoretical Computer Science*, 519:103–126, 2014. 2.2, 3.2, 3.6, 4.2.2

Corinna Cortes, Mehryar Mohri, Michael Riley, and Afshin Rostamizadeh. Sample selection bias correction theory. In *International Conference on Algorithmic Learning Theory*, pages 38–53. Springer, 2008. 2.2, 3.2, 3.6, 4.2.2

Amanda Coston, Alexandra Chouldechova, and Edward H Kennedy. Counterfactual risk assessments, evaluation, and fairness. *arXiv preprint arXiv:1909.00066*, 2019. 7.5

Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *Advances in Neural Information Processing Systems*, pages 3730–3739, 2017a. 4.3

Nicolas Courty, Rémi Flamary, Devis Tuia, and Alain Rakotomamonjy. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 39(9):1853–1865, 2017b. 4.3

Elliot Creager, David Madras, Joern-Henrik Jacobsen, Marissa Weis, Kevin Swersky, Toniann Pitassi, and Richard Zemel. Flexibly fair representation learning by disentanglement. In *International Conference on Machine Learning*, pages 1436–1445, 2019. 7.5

Imre Csiszár. Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten. *Magyer Tud. Akad. Mat. Kutato Int. Koezl.*, 8:85–108, 1964. 6.2

Imre Csiszár. Information-type measures of difference of probability distributions and indirect observation. *studia scientiarum Mathematicarum Hungarica*, 2:229–318, 1967. 6.2

Adnan Darwiche. A differential approach to inference in Bayesian networks. volume 50, pages 280–305. ACM, 2003. 1.3.3, 11.4

Adrià De Gispert and Jose B Marino. Catalan-english statistical machine translation without parallel corpus: bridging through spanish. Citeseer. 8.4

Aaron Dennis and Dan Ventura. Greedy structure search for sum-product networks. In *International Joint Conference on Artificial Intelligence*, volume 24, 2015. 9.1

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. 8.4

Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. 5.4.2, 5.A.1

William Dieterich, Christina Mendoza, and Tim Brennan. Compas risk scales: Demonstrating accuracy equity and predictive parity. *Northpoint Inc*, 2016. 7.1, 7.4.1

Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Icml*, volume 32, pages 647–655, 2014. 3.6

Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1166. 8.4

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`. 7.4.1

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012. 6.1, 6.5, 7.1, 7.5, 7.7

Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, 2015. 2.3, 6.1, 6.2, 6.3.2, 6.5, 7.1, 7.2, 7.4.1, 7.5

Dominik Maria Endres and Johannes E Schindelin. A new metric for probability distributions. *IEEE Transactions on Information theory*, 2003. 2.6, 4.4.3, 5.5.3

Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM, 2004. 3.6

Manaal Faruqui and Chris Dyer. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-1049. 8.4

Michael Feldman, Sorelle A Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. Certifying and removing disparate impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 259–268. ACM, 2015. 7.1

Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California, June 2016a. Association for Computational Linguistics. doi: 10.18653/v1/N16-1101. 8.4

Orhan Firat, Baskaran Sankaran, Yaser Al-onaizan, Fatos T. Yarman Vural, and Kyunghyun Cho. Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 268–277, Austin, Texas, November 2016b. Association for Computational Linguistics. doi: 10.18653/v1/D16-1026. 8.4

Lisheng Fu, Thien Huu Nguyen, Bonan Min, and Ralph Grishman. Domain adaptation for relation extraction with domain adversarial neural network. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 425–429, 2017. 4.2.2

Chuang Gan, Tianbao Yang, and Boqing Gong. Learning attributes equals multi-source domain generalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 87–97, 2016. 3.1

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016. 3.1, 3.4, 3.5, 3.5.1, 3.5.2, 3.5.2, 3.5.3, 3.6, 2, 3.A, 3.B, 3.C, 4.1, 4.2.2, 4.3, 4.4.3, 4.5, 5.1, 5.2, 5.2, 5.3.3, 6.1, 7.4.1, 8.4

Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 3248–3256, 2012. 1.3.1, 10.4.1

Robert Gens and Pedro Domingos. Learning the structure of sum-product networks. In *Proceedings of The 30th International Conference on Machine Learning*, pages 873–880, 2013. 9.4.1, 10.4.1

Pascal Germain, Amaury Habrard, François Laviolette, and Emilie Morvant. A pac-bayesian approach for domain adaptation with specialization to linear classifiers. In *ICML (3)*, pages 738–746, 2013. 3.6

Muhammad Ghifary, W Bastiaan Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. In *Proceedings of the IEEE international conference on computer vision*, pages 2551–2559, 2015. 3.1, 3.5.2

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 513–520, 2011. 3.1, 3.6, 4.1

Boqing Gong, Kristen Grauman, and Fei Sha. Connecting the dots with landmarks: Discriminatively learning domain-invariant features for unsupervised domain adaptation. In *ICML (1)*, pages 222–230, 2013. 3.6

Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *International conference on machine learning*, pages 2839–2848, 2016. 4.3

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. 2017. ISBN 9780262035613 0262035618. URL `https://www.worldcat.org/title/deep-learning/oclc/985397543&referer=brief_results`. 5.1

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. URL `http://arxiv.org/abs/1406.2661`. cite arxiv:1406.2661. 5.1, 5.5.1

Raghuraman Gopalan, Ruonan Li, and Rama Chellappa. Unsupervised adaptation across domain shifts by generating intermediate data representations. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2288–2302, 2014. 3.1

Stephan Gouws, Yoshua Bengio, and Greg Corrado. BilBOWA: Fast bilingual distributed representations without word alignments. In *Proceedings of ICML 2015*, pages 748–756, 2015. 8.4

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773, 2012. 5.3.5, 5.5.10

Jiatao Gu, Hany Hassan, Jacob Devlin, and Victor OK Li. Universal neural machine translation for extremely low resource languages. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 344–354, 2018. 8.1, 8.4

Asela Gunawardana and William Byrne. Convergence theorems for generalized alternating minimization procedures. *The Journal of Machine Learning Research*, 6:2049–2073, 2005. 9.5.3

Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. The FLORES evaluation datasets for low-resource machine translation: Nepali–English and Sinhala–English. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6098–6111, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1632. 8.1

Thanh-Le Ha, Jan Niehues, and Alexander Waibel. Toward multilingual neural machine translation with universal encoder and decoder. *arXiv preprint arXiv:1611.04798*, 2016. 8.3, 8.4

Jihun Hamm. Minimax filter: Learning to preserve privacy from inference attacks. *The Journal of Machine Learning Research*, 18(1):4704–4734, 2017. 6.1

Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *Advances in neural information processing systems*, pages 3315–3323, 2016. 2, 2.3, 2.3, 6.1, 6.2, 6.2, 6.3, 6.3.1, 6.5, 7.1, 7.2, 7.2, 7.2, 7.5

Philip Hartman et al. On functions representable as a difference of convex functions. *Pacific J. Math*, 9(3): 707–713, 1959. 9.3

Judy Hoffman, Brian Kulis, Trevor Darrell, and Kate Saenko. Discovering latent domains for multisource domain adaptation. In *Computer Vision–ECCV 2012*, pages 702–715. Springer, 2012. 3.1, 3.6

Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *arXiv preprint arXiv:1612.02649*, 2016. 4.2.2

Judy Hoffman, Mehryar Mohri, and Ningshan Zhang. Multiple-source adaptation for regression problems. *arXiv preprint arXiv:1711.05037*, 2017a. 3.1

Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017b. 3.1, 4.1, 4.2.2, 5.A.1

Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. *The Journal of Machine Learning Research*, 14(1):1303–1347, 2013. 10.3

Ehsan Hosseini-Asl, Yingbo Zhou, Caiming Xiong, and Richard Socher. Augmented cyclic adversarial learning for domain adaptation. *arXiv preprint arXiv:1807.00374*, 2018. 4.2.2

Haoyang Huang, Yaobo Liang, Nan Duan, Ming Gong, Linjun Shou, Daxin Jiang, and Ming Zhou. Unicoder: A universal language encoder by pre-training with multiple cross-lingual tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2485–2494, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1252. 8.4

Priyank Jaini, Abdullah Rashwan, Han Zhao, Yue Liu, Ershad Banijamali, Zhitang Chen, and Pascal Poupart. Online algorithms for sum-product networks with continuous variables. In *Conference on Probabilistic Graphical Models*, pages 228–239, 2016. 3, 1.3.3, 1.5

I-Hong Jhuo, Dong Liu, DT Lee, and Shih-Fu Chang. Robust visual domain adaptation with low-rank reconstruction. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2168–2175. IEEE, 2012. 3.1

Ray Jiang, Aldo Pacchiano, Tom Stepleton, Heinrich Jiang, and Silvia Chiappa. Wasserstein fair classification. *arXiv preprint arXiv:1907.12059*, 2019. 7.5

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *International conference on machine learning*, pages 3020–3029, 2016. 8.4

Fredrik D Johansson, Rajesh Ranganath, and David Sontag. Support and invertibility in domain-invariant representations. *arXiv preprint arXiv:1903.03448*, 2019. 4.3

James E Johndrow, Kristian Lum, et al. An algorithm for removing sensitive information: application to race-independent recidivism prediction. *The Annals of Applied Statistics*, 13(1):189–220, 2019. 2.3, 6.2, 7.2

Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017. 6.1, 8.1, 8.2.2, 8.4

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999. 1.3.2, 10.1

Faisal Kamiran and Toon Calders. Classifying without discriminating. In *2009 2nd International Conference on Computer, Control and Communication*, pages 1–6. IEEE, 2009. 2.3, 6.2, 6.3, 7.2

Toshihiro Kamishima, Shotaro Akaho, and Jun Sakuma. Fairness-aware learning through regularization approach. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 643–650. IEEE, 2011. 2.3, 6.2, 7.2

Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012. 6.5

Mohammadali Khosravifard, Dariush Fooladivanda, and T Aaron Gulliver. Confliction of the convexity and metric properties in f-divergences. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 90(9):1848–1853, 2007. 6.2

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 180–191. VLDB Endowment, 2004. 2.2, 3.2, 3.6, 4.2, 4.3

Niki Kilbertus, Mateo Rojas Carulla, Giambattista Parascandolo, Moritz Hardt, Dominik Janzing, and Bernhard Schölkopf. Avoiding discrimination through causal reasoning. In *Advances in Neural Information Processing Systems*, pages 656–666, 2017. 7.5

Diederik P Kingma and Max Welling. Auto-encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1.3.2, 10.3

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302, 2015. 8.3.1

Jyrki Kivinen and Manfred K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997. 9.A.1

Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *arXiv preprint arXiv:1609.05807*, 2016. 2.3, 6.2, 6.5, 7.5

Philipp Koehn, Franz J. Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 127–133, 2003. 8.4

Vladimir Koltchinskii. Rademacher penalties and structural risk minimization. *IEEE Transactions on Information Theory*, 47(5):1902–1914, 2001. 3.6

Dan Kondratyuk and Milan Straka. 75 languages, 1 model: Parsing universal dependencies universally. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2779–2795, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1279. 8.1

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*, pages 4066–4076, 2017. 7.5

Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 8.4

Gert R Lanckriet and Bharath K Sriperumbudur. On the convergence of the concave-convex procedure. pages 1759–1767, 2009. 9.5.3, 9.5.3, 9.5.3

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998a. ISSN 0018-9219. doi: 10.1109/5.726791. 5.A.6

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010. URL `http://yann.lecun.com/exdb/mnist/`. 5.4.2, 5.A.1

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998b. 3.5, 2

Jaeho Lee and Maxim Raginsky. Minimax statistical learning with wasserstein distances. In *Advances in Neural Information Processing Systems*, pages 2692–2701, 2018. 4.3

Peizhao Li, Han Zhao, and Hongfu Liu. Deep fair clustering for visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9070–9079, 2020. 1.5

Chen Liang, Jianbo Ye, Han Zhao, Bart Pursel, and C Lee Giles. Active learning of strict partial orders: A case study on concept prerequisite relations. *arXiv preprint arXiv:1801.06481*, 2018. 1.5

Friedrich Liese and Igor Vajda. On divergences and informations in statistics and information theory. *IEEE Transactions on Information Theory*, 52(10):4394–4412, 2006. 6.3.2, 6.3.2

Jianhua Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991. 2.6, 2.6.3, 4.4.3, 5.5.3, 6.3.2

Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018. 4.3, 5.1, 5.3, 5.3.4, 5.3.4

Robert Litschko, Goran Glavaš, Simone Paolo Ponzetto, and Ivan Vulić. Unsupervised cross-lingual information retrieval using monolingual data only. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1253–1256, 2018. 8.1

Hong Liu, Mingsheng Long, Jianmin Wang, and Michael I. Jordan. Transferable adversarial training: A general approach to adapting deep classifiers. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 4013–4022. PMLR, 2019. 5.1

Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. Transfer joint matching for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1410–1417, 2014. 4.3

Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105, 2015. 3.1, 3.6, 4.3, 5.3.3

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144, 2016. 4.3

Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2208–2217. JMLR, 2017. 5.1, 5.3.5, 5.4.1, 5.A.4, 5.A.4

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *NeurIPS*, pages 1647–1657, 2018. 5.1, 5.2, 5.A.1

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. *arXiv preprint arXiv:1511.00830*, 2015. 2.3, 3.1, 6.1, 6.2, 6.3.2, 6.5, 7.1, 7.2, 7.5

Kristian Lum and James Johndrow. A statistical framework for fair predictive algorithms. *arXiv preprint arXiv:1610.08077*, 2016. 6.5

Thang Luong, Hieu Pham, and Christopher D. Manning. Bilingual word representations with monolingual quality in mind. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, pages 151–159, 2015. 8.4

David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. In *International Conference on Machine Learning*, pages 3381–3390, 2018. 2.3, 6.1, 6.2, 6.4, 6.5, 7.1, 7.2, 7.4.1, 7.5, 7.5

David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Fairness through causal awareness: Learning causal latent-variable models for biased data. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 349–358. ACM, 2019. 7.5

Yishay Mansour and Mariano Schain. Robust domain adaptation. In *ISAIM*, 2012. 3.1, 3.6, 4.1

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009a. 2.2, 3.1, 3.2, 3.6, 4.1, 4.2.2

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pages 1041–1048, 2009b. 3.3

Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Multiple source adaptation and the rényi divergence. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 367–374. AUAI Press, 2009c. 2.2, 3.2, 3.3, 3.6, 4.2.2

R. Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *Proceedings of the ACL*, 2019. 5.1

Jun Mei, Yong Jiang, and Kewei Tu. Maximum a posteriori inference in sum-product networks. *arXiv preprint arXiv:1708.04846*, 2017. 2.4

Aditya Krishna Menon and Robert C Williamson. The cost of fairness in binary classification. In *Conference on Fairness, Accountability and Transparency*, pages 107–118, 2018. 7.1, 7.5

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013. 8.4

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 8.3.4

Andriy Mnih and Karol Gregor. Neural Variational Inference and Learning in Belief Networks. In *ICML*, 2014. 1.3.2, 10.3

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning*. MIT press, 2012. 3.7.5

Alfred Müller. Integral probability metrics and their generating classes of functions. *Advances in Applied Probability*, 29(2):429–443, 1997. 5.3, 5.3.5

Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999. 2.4

Razieh Nabi and Ilya Shpitser. Fair inference on outcomes. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 7.5

Arvind Narayanan. Translation tutorial: 21 fairness definitions and their politics. In *Proc. Conf. Fairness Accountability Transp., New York, USA*, 2018. 2.3, 6.2, 7.2

Radford M Neal. Probabilistic inference using markov chain monte carlo methods. 1993. 2.4

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 5, 2011. 3.5, 2

Graham Neubig and Junjie Hu. Rapid adaptation of neural machine translation to new languages. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 875–880, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1103. 8.4

Jian-Yun Nie, Michel Simard, Pierre Isabelle, and Richard Durand. Cross-language information retrieval based on parallel texts and automatic mining of parallel texts from the web. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 74–81, 1999. 8.1

Franz Josef Och and Hermann Ney. Statistical multi-source translation. pages 253–258, 2001. 8.4

Executive Office of the President. *Big data: A report on algorithmic systems, opportunity, and civil rights*. Executive Office of the President, 2016. 6.1

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. 3.1

James D Park and Adnan Darwiche. A differential semantics for jointree algorithms. *Artificial Intelligence*, 156(2):197–216, 2004. 11.4

Robert Peharz. *Foundations of sum-product networks for probabilistic modeling*. PhD thesis, 2015. 2.4, 9.3.2

Robert Peharz, Sebastian Tschiatschek, Franz Pernkopf, Pedro Domingos, and BEEPRI BioTechMed-Graz.

On theoretical properties of sum-product networks. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 744–752, 2015. 9.3.2, 10.2

Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the Latent Variable Interpretation in Sum-Product Networks. *arXiv preprint arXiv:1601.06180*, 2016. 10.3

Zhongyi Pei, Zhangjie Cao, Mingsheng Long, and Jianmin Wang. Multi-adversarial domain adaptation. 2018. 3.1, 4.1, 4.2.2

Xingchao Peng, Zijun Huang, Ximeng Sun, and Kate Saenko. Domain agnostic learning with disentangled representations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 5102–5112. PMLR, 2019. 5.6

Telmo Pires, Eva Schlinger, and Dan Garrette. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*, 2019. 8.2.1

Emmanouil Antonios Platanios, Mrinmaya Sachan, Graham Neubig, and Tom Mitchell. Contextual parameter generation for universal neural machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Brussels, Belgium, November 2018. 8.4

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. In *Advances in Neural Information Processing Systems*, pages 5680–5689, 2017. 2.3, 6.2

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 2551–2558, 2011. 3, 1.3, 1.3.1, 2.4.2, 2.4, 2.4.1

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, 2016. 1

Rajesh Ranganath, Sean Gerrish, and David Blei. Black Box Variational Inference. In *AISTATS*, pages 814–822, 2014. 1.3.2, 10.3

Marc'Aurelio Ranzato, Paco Guzmán, Vishrav Chaudhary, Peng-Jen Chen, Jiajun Shen, and Xian Li. Recent advances in low-resource machine translation. 2019. 8.2.2

Abdullah Rashwan, Han Zhao, and Pascal Poupart. Online and Distributed Bayesian Moment Matching for Parameter Learning in Sum-Product Networks. In *International Conference on Artificial Intelligence and Statistics*, 2016a. 10.4.1

Abdullah Rashwan, Han Zhao, and Pascal Poupart. Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1469–1477, 2016b. 3, 1.3.3, 1.5, 11.2

Philip Resnik. Mining the web for bilingual text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 527–534, College Park, Maryland, USA, June 1999. Association for Computational Linguistics. doi: 10.3115/1034678.1034757. 8.1

Philip Resnik and Noah A. Smith. The web as a parallel corpus. *Computational Linguistics*, 29(3):349–380, 2003. doi: 10.1162/089120103322711578. 8.1

Neil Robertson and Paul D Seymour. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984. 2.1

Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of The 31st International Conference on Machine Learning*, pages 710–718, 2014. 9.4.3, 10.4.1

Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273–302, 1996. 1.1, 1.3

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 3.1

Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV (4)*, volume 6314 of *Lecture Notes in Computer Science*, pages 213–226. Springer, 2010. ISBN 978-3-642-15560-4. 5.4.2, 5.A.1

Ruslan Salakhutdinov, Sam Roweis, and Zoubin Ghahramani. On the convergence of bound optimization algorithms. *UAI*, 2002. 9.5.3

Holger Schwenk. Filtering and mining parallel data in a joint multilingual space. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–234, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-2037. 8.1

Holger Schwenk and Matthijs Douze. Learning joint multilingual sentence representations with neural machine translation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 157–167, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-2619. 8.4

Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3076–3085. JMLR. org, 2017. 8.4

Jiajun Shen, Peng-Jen Chen, Matt Le, Junxian He, Jiatao Gu, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. The source-target domain mismatch problem in machine translation. *arXiv preprint arXiv:1909.13151*, 2019. 8.2.1

Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 4.3

Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. Learning from simulated and unsupervised images through adversarial training. *arXiv preprint arXiv:1612.07828*, 2016. 3.1, 4.2.2

Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018. 3.1

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017. 1

Jiaming Song, Pratyusha Kalluri, Aditya Grover, Shengjia Zhao, and Stefano Ermon. Learning controllable fair representations. In *Artificial Intelligence and Statistics*, pages 2164–2173, 2019. 6.1, 6.5

David Sontag, Amir Globerson, and Tommi Jaakkola. Introduction to dual composition for inference. In *Optimization for Machine Learning*. MIT Press, 2011. 2.4

Harold W Sorenson and Allen R Stubberud. Non-linear filtering by approximation of the a posteriori density. *International Journal of Control*, 8(1):33–51, 1968. 1.3.3, 11.1, 11.6

Amos Storkey. When training and test sets are different: Characterising learning transfer. *Dataset shift in machine learning.*, 2009. 5.1

Qian Sun, Rita Chattopadhyay, Sethuraman Panchanathan, and Jieping Ye. A two-stage weighting framework for multi-source domain adaptation. In *Advances in neural information processing systems*, pages 505–513, 2011. 3.1

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc., 2014. 8.3, 8.4

Yee W Teh, David Newman, and Max Welling. A Collapsed Variational Bayesian Inference Algorithm for Latent Dirichlet Allocation. In *Advances in neural information processing systems*, 2006. 1.3.2, 10.1.1, 10.2

Yee W Teh, Kenichi Kurihara, and Max Welling. Collapsed Variational Inference for HDP. In *Advances in neural information processing systems*, pages 1481–1488, 2007. 1.3.2, 10.1.1, 10.2

Michalis Titsias and Miguel Lázaro-Gredilla. Doubly Stochastic Variational Bayes for Non-conjugate Inference. In *Proceedings of the 31st International Conference on Machine Learning*, 2014. 1.3.2, 10.3

Michalis K Titsias. Local Expectation Gradients for Doubly Stochastic Variational Inference. *arXiv preprint arXiv:1503.01494*, 2015. 1.3.2, 10.3

Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama. Direct density ratio estimation for large-scale covariate shift adaptation. *Journal of Information Processing*, 17:138–155, 2009. 3.6

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. 3.1

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. *arXiv preprint arXiv:1702.05464*, 2017. 3.1, 3.5.2, 4.2.2, 4.4.3, 4.4.3, 5.1, 5.2, 5.3.1

Masao Utiyama and Hitoshi Isahara. A comparison of pivot methods for phrase-based statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 484–491, Rochester, New York, April 2007. Association for Computational Linguistics. 8.4

Leslie Valiant. What needs to be added to machine learning? In *Proceedings of ACM Turing Celebration Conference-China*, pages 6–6. ACM, 2018. 1

Guy Van den Broeck. Probabilistic and logistic circuits: A new synthesis of logic and machine learning. 2018. 1.3

Hemanth Venkateswara, Jose Eusebio, Shayok Chakraborty, and Sethuraman Panchanathan. Deep hashing network for unsupervised domain adaptation. In *(IEEE) Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5.4.2, 5.A.1

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. ACM, 2008. 3.6, 8.3.4

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11(Dec):3371–3408, 2010. 3.6

Visda. Visual domain adaptation challenge, 2017. URL `http://ai.bu.edu/visda-2017/`. 5.2, 5.4.2, 5.A.1

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008. 1.3.2

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566, 2017. 8.3.1

Wen Wang, Han Zhao, Honglei Zhuang, Nirav Shah, and Rema Padman. Dycrs: Dynamic interpretable postoperative complication risk scoring. In *Proceedings of The Web Conference 2020*, pages 1839–1850, 2020. 1.5

Yixin Wang, Dhanya Sridhar, and David M Blei. Equal opportunity and affirmative action via counterfactual predictions. *arXiv preprint arXiv:1905.10870*, 2019. 7.5

Frank Wilcoxon. Some Rapid Approximate Statistical Procedures. *Annals of the New York Academy of Sciences*, pages 808–814, 1950. 10.4.2

CF Jeff Wu. On the convergence properties of the EM algorithm. *The Annals of Statistics*, pages 95–103, 1983. 9.5.3

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. 8.2.2, 8.4

Huan Xu and Shie Mannor. Robustness and generalization. *Machine learning*, 86(3):391–423, 2012. 3.1, 3.6

Yichong Xu, Han Zhao, Xiaofei Shi, and Nihar B Shah. On strategyproof conference peer review. *arXiv preprint arXiv:1806.06266*, 2018. 1.5

Yadollah Yaghoobzadeh, Remi Tachet des Combes, Timothy J. Hazen, and Alessandro Sordoni. Robust natural language inference models with example forgetting. *CoRR*, abs/1911.03861, 2019. 5.1

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014. 3.6, 5.1

Alan L Yuille, Anand Rangarajan, and AL Yuille. The concave-convex procedure (CCCP). *Advances in Neural Information Processing Systems*, 2:1033–1040, 2002. 9.3.2

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. *arXiv preprint arXiv:1507.05259*, 2015. 6.1, 6.5, 7.1, 7.5

Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness beyond disparate treatment & disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1171–1180. International World Wide Web Conferences Steering Committee, 2017. 2.3, 6.1, 6.2, 7.1, 7.2, 7.5

Willard I Zangwill. *Nonlinear programming: a unified approach*, volume 196. Prentice-Hall Englewood Cliffs, NJ, 1969. 9.5.3, 9.5.3, 9.5.1

Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *International Conference on Machine Learning*, pages 325–333, 2013. 2.3, 6.1, 6.2, 6.5, 7.1, 7.2, 7.5, 8.4

Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340.

ACM, 2018. 6.1, 6.4, 6.5, 7.5, 7.5, 8.4

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827, 2013. 4.3

Kun Zhang, Mingming Gong, and Bernhard Schölkopf. Multi-source domain adaptation: A causal view. In *AAAI*, pages 3150–3157, 2015a. 3.1, 3.5.2

Shanghang Zhang, Guanhang Wu, Joao P Costeira, and Jose MF Moura. Understanding traffic density from large-scale web camera data. *arXiv preprint arXiv:1703.05868*, 2017a. 3.5, 3.5.3, 3, 3.C

Xu Zhang, Felix X. Yu, Shih-Fu Chang, and Shengjin Wang. Deep transfer network: Unsupervised domain adaptation. *CoRR*, abs/1503.00591, 2015b. 5.1

Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. Aspect-augmented adversarial networks for domain adaptation. *arXiv preprint arXiv:1701.00188*, 2017b. 4.1, 4.2.2

Han Zhao and Geoff Gordon. Frank-wolfe optimization for symmetric-nmf under simplicial constraint. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, 2018. 1.5

Han Zhao and Geoffrey J Gordon. Linear time computation of moments in sum-product networks. In *Advances in Neural Information Processing Systems*, pages 6894–6903, 2017. 3, 1.3.3

Han Zhao and Geoffrey J Gordon. Inherent tradeoffs in learning fair representations. In *Advances in neural information processing systems*, 2019. 1.2.2, 7.1, 7.2.1, 7.3.1, 7.5, 8.4, 12.1

Han Zhao, Zhengdong Lu, and Pascal Poupart. Self-adaptive hierarchical sentence model. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015a. 8.3.1

Han Zhao, Mazen Melibari, and Pascal Poupart. On the relationship between sum-product networks and bayesian networks. In *International Conference on Machine Learning*, pages 116–124, 2015b. 3, 2.4, 9.1, 9.1, 9.3.2, 10.2

Han Zhao, Tameem Adel, Geoff Gordon, and Brandon Amos. Collapsed variational inference for sum-product networks. In *International Conference on Machine Learning*, pages 1310–1318, 2016a. 1.3.1, 11.4

Han Zhao, Pascal Poupart, and Geoffrey J Gordon. A unified approach for learning the parameters of sum-product networks. In *Advances in neural information processing systems*, pages 433–441, 2016b. 3, 1.3.3, 10.3, 11.4, 11.5

Han Zhao, Otilia Stretcu, Renato Negrinho, Alex Smola, and Geoff Gordon. Efficient multi-task feature and relationship learning. *arXiv preprint arXiv:1702.04423*, 2017. 1.5

Han Zhao, Shuayb Zarar, Ivan Tashev, and Chin-Hui Lee. Convolutional-recurrent neural networks for speech enhancement. In *Acoustics, Speech and Signal Processing (ICASSP), 2018 IEEE international Conference on*. IEEE, 2018a. 1.5

Han Zhao, Shanghang Zhang, Guanhang Wu, Geoffrey J Gordon, et al. Adversarial multiple source domain adaptation. In *Advances in neural information processing systems*, 2018b. 1.1, 1.2.1, 4.2.2, 4.4.3, 5.1, 8.4

Han Zhao, Shanghang Zhang, Guanhang Wu, Geoffrey J Gordon, et al. Multiple source domain adaptation with adversarial learning. 2018c. 4.2.2, 5.2, 5.6

Han Zhao, Jianfeng Chi, Yuan Tian, and Geoffrey J Gordon. Adversarial privacy preservation under attribute inference attack. *arXiv preprint arXiv:1906.07902*, 2019a. 1.5, 6.1, 7.7

Han Zhao, Jianfeng Chi, Yuan Tian, and Geoffrey J. Gordon. Adversarial privacy preservation under attribute inference attack. *arXiv preprint arXiv:1906.07902*, 2019b. 1, 2, 12.1

Han Zhao, Amanda Coston, Tameem Adel, and Geoffrey J Gordon. Conditional learning of fair representations. *arXiv preprint arXiv:1910.07162*, 2019c. 1.2.2, 8.4

Han Zhao, Amanda Coston, Tameem Adel, and Geoffrey J. Gordon. Conditional learning of fair representations. *arXiv preprint arXiv:1910.07162*, 2019d. 6.5

Han Zhao, Remi Tachet Des Combes, Kun Zhang, and Geoffrey Gordon. On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532, 2019e. 1.2.1, 1.5, 6.1, 8.4, 12.1

Han Zhao, Junjie Hu, Zhenyao Zhu, Adam Coates, and Geoff Gordon. Deep generative and discriminative domain adaptation. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2315–2317. International Foundation for Autonomous Agents and Multiagent Systems, 2019f. 1.2.1, 1.5

Han Zhao, Junjie Hu, Zhenyao Zhu, Adam Coates, and Geoffrey J. Gordon. Deep generative and discriminative domain adaptation. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, 2019g. 4.2.2

Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon. On learning invariant representations for domain adaptation. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, volume 97 of *Proceedings of Machine Learning Research*, pages 7523–7532. PMLR, 2019h. 5.1, 5.2, 5.2, 5.2, 5.3.2, 5.5.3

Sicheng Zhao, Bo Li, Xiangyu Yue, Yang Gu, Pengfei Xu, Runbo Hu, Hua Chai, and Kurt Keutzer. Multi-source domain adaptation for semantic segmentation. In *Advances in Neural Information Processing Systems*, pages 7285–7298, 2019i. 1.1

Indre Zliobaite. On the relation between accuracy and fairness in binary classification. *arXiv preprint arXiv:1505.05723*, 2015. 6.1, 6.3

Barret Zoph and Kevin Knight. Multi-source neural translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 30–34, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1004. 8.1, 8.4