Towards More Efficient and Data-Driven Domain Adaptation

Petar Stojanov

CMU-CS-21-136

August 2021

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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

This research was sponsored by MT Research, IBM, the Air Force under award number FA8702-15-D-0002, the Defense Advanced Research Projects Agency under award number HR001115C0114, and Mt. Sinai under grant number 025578554609

Keywords: machine learning, transfer learning, graphical models, domain adaptation, causal discovery, kernel methods, neural networks

For my family

Acknowledgments

This work would not have been possible without the tireless support, guidance, and perseverance of my advisors and mentors, Jaime Carbonell and Kun Zhang. It all started with Jaime Carbonell accepting me in his group and giving me the freedom to pursue various directions in AI and machine learning. While he is no longer with us, I will never forget his role in my life, my career and his contribution to the scientific community. The freedom and guidance he provided quickly led me to read the ground-breaking work by Kun Zhang, who put an enormous amount of trust in me and agreed to be my co-advisor. More importantly than all the technical guidance, he taught me so much about systematic scientific thought, principled methodology and pursuit of bold, fundamental ideas. Therefore, I will forever be grateful for everything Jaime Carbonell and Kun Zhang have done for my growth as a scientist and a person.

I would also like to acknowledge my mentor, co-author and collaborator Mingming Gong, for tirelessly working through all difficulties work with me and teaching me how to improve as a scientist. In addition, I would like to thank my other collaborators and co-authors: Biwei Huang, Shaoan Xie and Zeyu Tang for all the help, advice, hard work and meaningful discussions. Furthermore, I am deeply grateful to the members of the thesis committee: Barnabas Poczos, Eric Xing and Aapo Hyvärinen for their invaluable feedback and effort in guiding this work to its current state.

I would express my deepest gratitude to my family: my mother Slavica Stojanova, my father Ljupco Stojanov, and my brother Stefan Stojanov, who were constantly by my side and supporting me through all of the ups and downs of this long challenging process. I would not be who I am, and none of this would have been achieved without them. Thank you for always being there for me, for listening, for enduring and for giving me the strength and courage to keep growing.

Finally, I would like to thank all of my dear friends: Siddhartha Jain, Rohan Varma, Hemank Lamba, Yasmine Kotturi, Christoph Dann, Benedikt Boecking, Brian Atwood, Nathan Saucier, Olga Opojevici, Nicholas Wilton, Stefan Andjelkovic, Naji Shajarisales, among many others. Thank you immensely for all of the support, be it technical, emotional or both. I could have not achieved any of the work presented of this thesis without you, and I will forever be grateful to you.

Abstract

In recent years with the fast progress made in neural networks research, supervised machine learning approaches have become increasingly powerful in finding flexible functions to predict target variable Y from input features X. However, most of these complex models require a large amounts of data to train, and often work under the assumption that the data points are i.i.d. In reality these assumptions are very likely to be violated. A simplified notion of this violation is when the training and the test datasets come from different joint distributions (i.e. $P^{train}(X, Y) \neq P^{test}(X, Y)$). In this setting, where the training and test datasets are also known as source and target domains respectively, domain adaptation is required to obtain good performance. In particular, when only unlabeled features are observed in the target domain, this setting is referred to as *unsupervised domain adaptation*, and it will be the main focus of this thesis.

Domain adaptation is a wide sub-field of machine learning with the task of designing algorithms to account for this distributional difference under specific assumptions, for the purpose of better prediction performance in the target domain. In this thesis we make use of the data-generating process to address several sub-problems of unsupervised domain adaptation. Namely, we first address the problem of unsupervised domain adaptation with multiple labeled source domains and an unlabeled target domain under the conditional-target shift setting, and we present an approach to capture the low-dimensional changes of the joint distribution across domains in order to perform prediction in the target domain. Secondly, we introduce an algorithm to reduce the dimensionality of the data when performing domain adaptation under the covariate shift setting. In particular, we make use of the particular properties of the covariate shift setting in order to reduce the dimensionality of the data such that we preserve relevant predictive information about the target variable Y.

We further investigate domain adaptation from the perspective of the data-generating process when addressing the problem using neural networks. Deep neural architectures are commonly used to extract domain-invariant representations from the observed features. However, without labels in the target domain, there is no guarantee that these representations will have relevant predictive information for the target domain data. In this thesis, we investigate techniques to regularize this invariant representation in order to enforce that it has non-trivial structure which contains information which is relevant for predicting Y in the target domain. The first of these techniques is based on mutual information, and the second technique makes use of a novel criterion of distortion of the marginal distribution when transforming it from the source to the target domain.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In the modern age of powerful computers, mobile phones, and high-bandwidth internet, AI has found an ever-present role in many aspects of everyday life. It is present in the advertisements that are shown to us when using many internet applications, in the applications that editors use to refine photographs and films that we enjoy, in tools used for automatic translations of text, in surveillance and security systems, operations of vehicles, healthcare applications, among many others . AI systems often use large amounts of data to train, and in many cases such data is readily available. For example, Facebook and Google have access to tremendous amounts of diverse user data. However, even for these technological giants, there is one major problem that constantly arises when they deploy AI for use in the real world, and that is the fact that the real world is constantly changing, or in other words, it is *nonstationary*. This means that there is a very high chance that when an AI system is deployed to be used for prediction, it may encounter phenomena that were not present in the data that was used to train it.

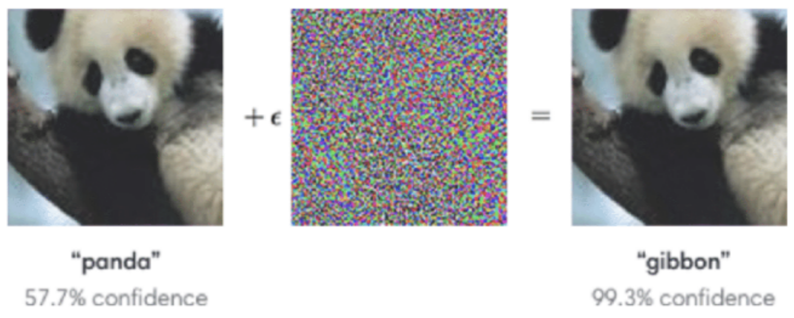


Figure 1.1: A simple illustration how an small adversarial attack can throw off a learning algorithm.

In the quite egregious case, when the domain of application consists of human subjects, training data may have only consisted of Caucasian people, and the system is unable to be of proper use to people of other ethnicities, raising serious ethical concerns. To see just how detrimental even a tiny change in environment can be to a common AI algorithm, consider the simple illustration in Figure 1.1, which was encountered and presented by (Goodfellow et al. [45]). Here, a very small amount of Gaussian noise was applied to the image, and the change is invisible to the human eye, yet the learning algorithm completely fails to classify the image. Therefore, in order to make AI systems more robust, reliable, and most importantly, fair and accessible to everyone, one needs to tackle the problem of data nonstationarity or heterogeneity.

To formalize the learning setting and the basic learning task, we represent our data in terms of features $X \in \mathbb{R}^d$ and target variable $Y \in \mathbb{R}$ (or a discrete variable in the case of classification), which follow a joint distribution $P(X, Y)$. In practice, we are given labeled training data $(\mathbf{x}^{tr}, \mathbf{y}^{tr}) = (\mathbf{x}_k^{tr}, y^{tr})_{k=1}^{m_{tr}}$, and unlabeled test data $\mathbf{x}^{te} = (\mathbf{x}^{te})_{k=1}^{m_{te}}$. The goal is to learn to predict Y from X via a predictor function $\hat{h}$: $\hat{Y} = \hat{h}(X)$, as accurately as possible on the test data. Generally, the learning objective can be summarized as minimizing the risk:

$$h^* = \arg\min_h \mathbb{E}_{X,Y \sim P(X,Y)}[l(X, h(Y))],$$

where l is a loss function to evaluate the degree of error that h makes. In the finite-sample case, the goal is to minimize the empirical risk:

$$\hat{h} = \arg\min_h \sum_{i=1}^{m_{tr}} [l(\mathbf{x}_i^{tr}, h(y_i^{tr}))].$$

If X^{te} also follows $P(X, Y)$, then as we accumulate more and more training data, we would expect the learned predictor $\hat{h}$ to perform increasingly well on the test data if it is properly regularized. However, the environment can change at test time, and when it does, one way to formalize this phenomenon is that the train and test joint distributions are not equal: $P^{tr}(X, Y) \neq P^{te}(X, Y)$, and in this situation $\hat{h}$ cannot be expected to perform well on the test data.

Domain adaptation is a sub-field of machine learning tasked with studying how to deal with this setting and adapt the learning machine to account for the difference in the joint distribution at training and test time. In the field of domain adaptation, the training and test data distributions are referred to as the source and target domain distributions respectively, and this is the terminology that will be used throughout this thesis. We will denonte the source domain joint distribution as $P^{\mathcal{S}}(X, Y)$, and the target domain distribution as $P^{\mathcal{T}}(X, Y)$.

1.1 Why is adaptation possible? - A Causal View

If one solely considers the mathematical formulation of the above problem along with the fact that $P^{\mathcal{S}}(X, Y) \neq P^{\mathcal{T}}(X, Y)$, one would easily become pessimistic and give up on trying to tackle the problem of adaptation. However, humans and other intelligent beings can easily and quickly learn concepts that generalize across environments. For example, humans learn at a very early age to distinguish a dog from a bird in an image, no matter whether the image is at daytime, night time, in an urban environment or the countryside. A high-level explanation why this is possible for us is that there are some core semantic *similarities* across the different environments that we can easily recognize. This means that the distribution $P(X, Y)$ changes across different domains only in some small, manageable way, such that key information for making decisions was not lost. In order to reason about parsimonious ways in which changes like this can occur, one has to go beyond the statistical information observed in the data and consider some kind of structural (ontological) information. It turns out that causal information is naturally the kind of view that can provide us with a way to start to reason about *minimal changes*.

1.1.1 Causality and Its Relationship with Prediction

In order to reason about parsimonious changes of joint distributions we will first take a causal view on the two random variables X and Y. If we think of these variables from a purely statistical aspect, we care about making observational statements about their relationship and its uncertainty. For example, we may want to know with what probability we can *observe* specific values of X and Y (as specified by the joint distribution $P(X, Y)$), or the probability of a certain value of $Y = y$, given that a specific value $X = x$ is *observed* (as specified by the conditional distribution $P(Y|X = x)$).

Causality, on the other hand, deals with how this set of variables behaves when subject to *interventions* (Pearl et al. [95], Woodward [134]). To illustrate this, let us suppose that we only have two variables in the system and they are observed. Specifically, let Y represent a class of object in a room (out of a finite discrete set of objects), and let X represent the pixels of a digital photo taken from the individual object. It is possible to *intervene* on Y by removing all objects from the room except for toy cars. This intervention will have a strong effect on the distribution of images X that can be obtained when taking pictures. On the other hand, intervening on X by modifying the taken digital image in Adobe Photoshop has no effect on the distribution of objects in the room. This asymmetric behavior under interventions establishes Y as the *cause* and X as its *effect*.

One rigorous way to represent causal relationships are functional causal models (Pearl [94]). Suppose that the variables X and Y have a causal relationship in addition to their statistical one. In the case when Y is the cause of X and they do not have a latent common cause, functional causal models express the generating process of Y and X in terms of functions and independent noise terms:

$$\begin{aligned} Y &= E_Y, \\ X &= f_X(Y, E_X), \end{aligned}$$

where $E_X \perp\!\!\!\perp E_Y$. The effect X is a function of Y, along with some random variable E_X which represents all unmeasured factors unrelated to Y. Now, we can make one key observation: that the way in which Y is generated is independent from the way in which X is generated from Y, and we refer to this as the *modularity condition*. Functional causal models have been studied in great depth for various classes of functions that f_X can represent, such as additive noise models (Hoyer et al. [59]), and post-nonlinear additive noise models (Zhang and Hyvärinen [142]). By the modularity condition, the functional properties of f_X are unrelated to Y, and furthermore, it follows that Y and E_X are statistically independent. Under this representation, it is then easy to see that when changing Y, X might also change, but not vice versa. In particular, an intervention may be a change in the parameters of E_Y, E_X and/or the functional form of f_X, and such an intervention would lead to a different joint distribution $P(X, Y)$. Furthermore, by representing X and Y in terms of this causal relationship, the distributions $P(Y)$ and $P(X|Y)$ and their respective parameters correspond to not only statistical properties, but also physical processes regarding how the data was generated (Schölkopf et al. [102]).

Now let us consider non-stationarity from the view of the data-generating process outlined above. Namely, let us suppose that the joint distribution $P(X, Y)$ changes across domains. This means that its factors $P(Y)$ and $P(X|Y)$ may change. Let us use θ_Y and θ_X to denote the changing aspects (parameters) of $P(Y)$ and $P(X|Y)$, respectively (Bareinboim and Pearl [1], Zhang et al. [145]), as shown in Figure 1.5. Since $P(Y)$ and $P(X|Y)$ represent physical processes through which the variables are generated, the changes in the process through which X is generated from Y are independent from the changes in the process that generated Y itself. To give an illustrative example, suppose that Y is an object (a cat vs. a dog) that we wish to classify in an image, and that X is the image itself. The changes in the physical mechanism $P(Y)$ may entail different proportions of cats and dogs in different environments, and the changes in $P(X|Y)$ may represent different camera resolution or illumination conditions. If we assume that Y and X are the only variables in the system and that there are no unmeasured confounding factors, then it is clear that the way the proportion of cats and dogs changes is completely unrelated to the resolution of the camera or the illumination condition. Thus, factoring the joint distribution in terms of the physical data-generating processes $P(X, Y) = P(Y)P(X|Y)$ yields two independently-changing *causal modules* (Zhang et al. [145]). Consequently, via Bayes rule, it becomes apparent that generally speaking, the factors obtained by the other factorization, $P(X)$ and $P(Y|X)$, have changes which are coupled (Daniusis et al. [27], Janzing and Schölkopf [65], Zhang et al. [144]).

We can now return to the notion of *minimal changes* on which we rely in order to do domain adaptation. Since $P(Y)$ and $P(X|Y)$ are the independently-changing causal modules of the joint distribution, the changes that they undergo are naturally minimal. On the other hand, the changes in $P(X)$ and $P(Y|X)$ are generally not minimal, because a change in one factor can depend on one or more changes in the other factor in non-trivial ways. Therefore, one key advantage of having the causal view of the data is that we can decompose the joint distribution in terms of independently-changing factors whose changes we know are minimal, so we can hope to adapt each module individually, and the change we would have to model would not be complex, because they correspond to (natural) causal generating processes.

However, the property of minimal changes is not the only advantage of the causal view of the data. In addition to telling us how the minimal changes may have occurred, the causal information also provides us with knowledge of how we can use the available data for learning. Recall that we are given labeled training data $(X^{\mathcal{S}}, Y^{\mathcal{S}})$ in the source domain (from the distribution $P^{\mathcal{S}}(X, Y)$), and unlabeled test data $X^{\mathcal{T}}$ from $P^{\mathcal{T}}$, since the joint distribution $P^{\mathcal{T}}(X, Y)$ is not observed. It turns out that the causal view can also tell us whether we can use unlabeled data from $P^{\mathcal{T}}(X)$ in the target domain to facilitate adaptation and guide the learning process towards inferring anything about $P^{\mathcal{T}}(X, Y)$. Namely, as we mentioned, if Y is the cause of X then $P(X)$ and $P(Y|X)$ have parameters which are dependent on each other, and $P^{\mathcal{T}}(X)$ has useful information about the unobserved $P^{\mathcal{T}}(Y|X)$, which is the quantity we are interested for prediction in the target domain (Gong et al. [42], Schölkopf et al. [104], Zhang et al. [148]).

To provide an example of how this dependency is useful, consider the illustration on Figure 1.2. Here, we have different individuals (domains), and we collect handwritten digit data from them. We are given two variables: Y represents the number that the person intends to write, and X represents the image of the digit they wrote down. Clearly, the generating process in this example is $Y \to X$, because if at a later time-point, the person changes their intended concept for writing Y, this will directly impact the observed image X. On the other hand, us changing the pixels of the image X has no bearing on the abstract concept in the mind of the person who drew it. Each domain (person) has a different style of writing, which represents the changes of $P(X|Y)$ across domains. Similarly, each person may have a different favorite number they are more likely

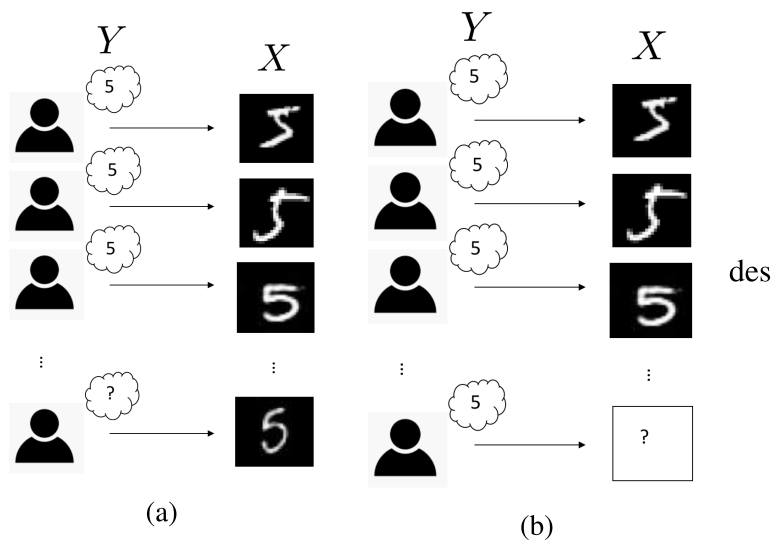


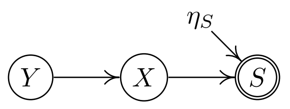
Figure 1.2: Illustrative example regarding prediction in the anti-causal vs. causal direction, in the case when $Y \rightarrow X$.

to think of, and this represents the variability of $P(Y)$ across domains.

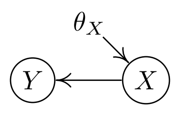
Now, let us consider the task of predicting the abstract object Y from the image X. In this case, we are predicting the cause from the effect (*anti-causal direction*). As shown in Figure 1.2a, we are given labeled source domains (people) for whom we know the number they meant to write, and the image that they wrote of that number. At test time, we have one or more people that only write down an image, and the task is to predict the number they thought of. It is clear that the distribution of images for the target domain(s), $P^{\mathcal{T}}(X)$ has valuable information about the label Y. Namely, based on what we know from the source domains, we can make use of the image in the target domain to deduce that its contours have a familiar shape, and in the case shown in the Figure 1.2a, that the number is most likely "five".

On the other hand, let us consider the opposite scenario - predicting the effect X from the cause Y (*causal direction*). In this case, at test time, we know the abstract object the person thought of, and we have to predict what the image of the written-down digit looks like, as shown in Figure 1.2b. It is evident that we have no way of knowing anything about the person's style from just the concept "five", and the best we can do is take an average of the styles observed in the source domains, as proven in (Zhang et al. [149]). Therefore, knowledge of the causal direction tells us not only what the independently changing causal modules are, but also what information we can make use of for prediction in the target domain.

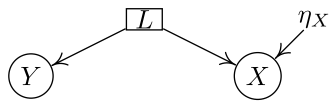
So far, we have discussed causal and anti-causal learning when considering only two random variables: the features X and the target variable Y. However, causality is generally not restricted to two variables, and is typically defined as a directed acyclic graph with multiple (d) vertices, represented by the random variables $X_1, ..., X_d$, and the directed edges represent direct cause-effect relationships. In the graph, each vertex X_i is independent of its non-decendants given its parents (also called the *causal Markov condition*). Furthermore, when observing data drawn from the joint distribution of the d variables, a common assumption is that the conditional-independence



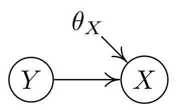
(a) Example 1: selection bias (a data point is included if and only if $S = 1$.



(b) The augmented graph explaining the changes of $P(X, Y)$ across domains in Example 1.



(c) Example 2: generating process under the presence of a confounder; the mechanism of X changes across domains, as indicated by η_X



(d) The augmented graph explaining the changes of $P(X, Y)$ across domains in Example 2.

Figure 1.3: Examples how selection bias and latent confounders may affect the properties of the observed distribution (Zhang et al. [150]).

relations observed in the data correspond to the structure of the causal graph (this property is also known as *faithfulness*) (Pearl et al. [95], Spirtes et al. [113]). In the case with multiple variables, functional causal models, or structural equation models, can also be used to describe the same causal relationships.

If the causal graph underlying the observed data is known, there is no confounder (hidden direct common cause of two variables), and the observed data are perfect random samples from the populations, then one can directly benefit from the causal model for transfer learning, as claimed in (Zhang et al. [150]). However, in practice the observed data undergo an imperfect sampling process such as a selection bias (Bareinboim et al. [2]), or there may be unobserved confounders. In this case the learned graph that represents the statistical dependence relations between the observed variables may not be causal, and thus may not be used to answer questions regarding what would happen under interventions on a subset of the variables. Fortunately, it turns out that for passive prediction under distribution shift, it suffices to consider the independently changing factors of the joint distribution across domains.

To illustrate this more clearly, let us consider the examples on Figure 1.3, introduced by (Zhang et al. [150]). In Figure 1.3a, we are presented with the underlying causal graph of generating X from Y along with a biased sampling process. Here, S corresponds to a binary selection variable whose value is $S = 1$ if and only if a data-point is selected. The selection process, given by $P(S = 1|X)$ undergoes changes across domains, as denoted by the auxiliary variable η_S. However, if this selection process is unobserved, we will observe two phenomena instead: **(1)** that the marginal distribution $P(X)$ changes across domains, and **(2)** that $P(Y|X)$ does not change across domains, because in the original generating process, Y is conditionally independent of the auxiliary variable η_X given X. In Figure 1.3b, we see exactly an *augmented* graphical model which represents the *observed* joint distribution along with its changes across domains. Here, θ_X indicates the property of distribution shift of $P(X)$ across domains (and has the same value for each observation within the same domain, but may have different values across domains). Thus we observe two independently changing factors of the joint distributions, where $P(X)$ changes and $P(Y|X)$ stays the same (and thus undergoes the trivial identity map). This compact representation of the joint distribution reveals to us that given sufficient amount of observed data and a flexible enough model, it may not be needed to perform domain adaptation at all because the optimal predictor which is specified by $P(Y|X)$ stays the same across domains.

Similarly, suppose that the true generating process of the variables X and Y involves an unobserved common cause, as illustrated in Figure 1.3c. If we consider only the observed variables and their joint distribution in two or more domains, it is apparent that: **(1)** X and Y are dependent on each other, **(2)** $P(Y)$ does not change across domains, because in the true generating process Y and η_X are marginally independent, and **(3)** $P(X|Y)$ changes across domains. This yields two independently changing factors of the joint distributions $P(Y)$ and $P(X|Y)$ (where $P(Y)$ undergoes the trivial identity map). Given these observations, it becomes clear that in order to deduce what aspects of the joint distribution need to be adapted to a new domain of interest, one only cares about the joint distribution and its compact representation regarding its independently-changing factors. For the case when there are only two variables X and Y, there are various possible scenarios of domain shift which are presented on Figure 1.5, each of which has been studied in detail and has its own respective techniques to address the domain shift, some of which we shall discuss below.

1.2 Domain Adaptation Techniques

Causal and anti-causal learning have been analyzed and discussed in detail in (Schölkopf et al. [102], Zhang et al. [148, 149]). In this thesis, we aim to tackle problems in both the anti-causal and causal learning paradigms. We now provide an overview of the fundamental techniques used to tackle domain adaptation in various settings.

1.2.1 Covariate Shift

One of the most widely studied and classical domain-adaptation settings is the one of *covariate shift* (Cortes et al. [21], Shimodaira [106], Sugiyama et al. [118, 119]). In this setting it is assumed that the joint distribution $P(X, Y)$ changes across domains in a very specific way: $P(X)$ changes across domains whereas $P(Y|X)$ stays the same. From the causal perspective, since $P(X)$ changes and $P(Y|X)$ stays the same, they act as independently changing causal modules in a rather specific way; in which $P(Y|X)$ undergoes a trivial identity transformation. Therefore, this kind of change is consistent with the causal direction: $X \rightarrow Y$ (Zhang et al. [149]), shown in Figure 1.5a (assuming there is no selection bias in the data). This assumption is convenient because what changes is what is observed, whereas the unobserved quantity is invariant. At first sight though, it may seem that since what we wish to predict (given by $P(Y|X)$) is the same across domains, there is no need for adaptation. This would be true if we had a very flexible class of functions and a large amount of data. However, in many areas of application, there are small datasets that require fitting simple, and thus mis-specified models.

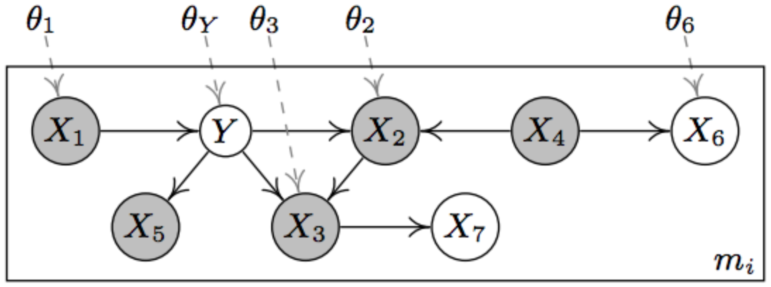
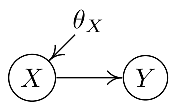
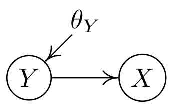


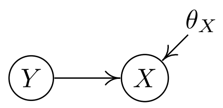
Figure 1.4: Example of an augmented graphical model (presented in (Zhang et al. [150])), learned from observed data via the algorithm introduced in (Zhang et al. [145]).



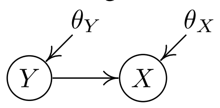
(a) Covariate shift.



(b) Target shift.



(c) Conditional shift.



(d) Conditional-target shift.

Figure 1.5: Various scenarios of distribution shift from the causal view, when considering two variables X and Y.

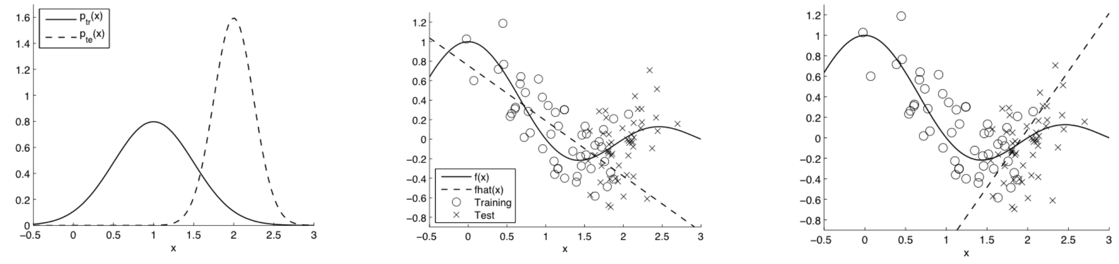


Figure 1.6: An example of covariate shift, first introduced in (Sugiyama et al. [119]).

To have a clearer picture about covariate shift and the need for correcting it, let us consider the example on Figure 1.6, first introduced in (Sugiyama et al. [119]). On the left panel we are given the marginal distributions of X, $P^{\mathcal{S}}(X)$ and $P^{\mathcal{T}}(X)$, with the solid and dotted bell curve respectively. In the middle panel, we see points from $P^{\mathcal{S}}(X,Y)$ labeled with circles and points from $P^{\mathcal{T}}(X,Y)$ labeled with crosses, along with a solid curve representing $\mathbb{E}[Y|X]$, the optimal predictor. In this middle panel, there is also a simpler (linear) predictor shown by the dotted line, which was fit only on the points from the source domain. It is obvious that this predictor would perform fairly poorly on the target-domain data-points. The objective of covariate shift correction is to make use of the unlabeled data in the target domain and correct this simple predictor to take into account the shift in $P(X)$. The result of this correction can be seen in the right panel, in which the dotted line now fits the target domain points well.

In order to perform covariate shift correction, one must find a way to prioritize the subset of the points in the source domain that are most similar in marginal distribution to the target domain, when constructing the model. In particular, one population way to achieve it is to compute importance weights for the source domain data-points, which are then incorporated into the learning algorithm. An importance weight can be thought of as a function of X, given by $\beta(X)$, such that when multiplied with the data X, results in the distribution $P^{\mathcal{T}}(X)$ in the target domain. When factoring the joint distribution in the target domain, it is easy to see their form:

$$P^{\mathcal{T}}(X,Y) = P^{\mathcal{T}}(X)P^{\mathcal{T}}(Y|X) = \frac{P^{\mathcal{T}}(X)}{P^{\mathcal{S}}(X)}P^{\mathcal{S}}(X)P^{\mathcal{T}}(Y|X) = \beta(X)P^{\mathcal{S}}(X)P^{\mathcal{T}}(Y|X),$$

where $\beta(X) = \frac{P^{\mathcal{T}}(X)}{P^{\mathcal{S}}(X)}$, is the importance weight which can be calculated from the observed features X. Then, the learning algorithm can make use of the source domain data along with its labels and the importance weights to learn a predictor f^*:

$$f^* = \arg\min_{h\in\mathcal{H}} \mathbb{E}_{(X,Y)\sim P^{\mathcal{S}}(X,Y)}[\beta(X)l(f(X),Y)],$$

where l is an appropriate loss function, and $\mathcal{H}$ is a hypothesis space.

The bulk of the work done on covariate shift correction deals with efficient computation of these importance weights from observed data. In Chapter 3 of this thesis, we will examine in detail the technical underpinnings and challenges of this domain adaptation setting, and introduce a novel approach of reducing the dimensions so that computing the importance weights is statistically more efficient, without sacrificing important information for prediction in the target domain.

1.2.2 Conditional-Target Shift

While covariate-shift setting undergoes a change that can be corrected by focusing on the features X (which are observed in both domains), the assumption that the optimal predictor $P(Y|X)$ does not change is very strong, and often may not hold in practice. To make a domain adaptation algorithm more applicable, one is faced with the following question: is there an approach which does not assume that $P(Y|X)$ is the same across domains, but at the same time does not require any class labels in the target domain?

As illustrated in the example on Figure 1.2, in the anti-causal learning setting in which $Y \rightarrow X$, $P(X)$ (which is observed in the target domain) contains valuable information about the optimal predictor $P(Y|X)$ (which is only observed in the source domain). By assuming anti-causal learning, this allows us to factorize the joint distribution $P(X,Y)$ into two independently changing causal modules $P(Y)$ and $P(X|Y)$. In prior work (Gong et al. [42], Zhang et al. [145, 148]), it was reasoned that since these causal modules change independently, the change they undergo is minimal. Therefore, various assumptions can be made about parametric changes these modules can undergo across domains, and then try to learn these parametric changes in order to infer the modules in the target domain (labeled as $P^{\mathcal{T}}(X|Y)$ and $P^{\mathcal{T}}(Y)$) under the constraint that they can successfully reconstruct the observed distribution $P^{\mathcal{T}}(X)$ in the target domain. To provide a basic blueprint for the procedure, let us consider a transformation function $\mathcal{T}$ which transforms the data from the source to the target domain:

$$(\mathcal{T}^*) = \arg\min_{\mathcal{T}} d(P^{\mathcal{S}}(X^{new}), P^{\mathcal{T}}(X)) \iff$$
$$\arg\min_{\mathcal{T}} d(\int P^{\mathcal{S}}(\mathcal{T}(X), y)dy, P^{\mathcal{T}}(X)),$$

where d is any valid distribution divergence metric. In essence, this framework searches for an optimal transformation $\mathcal{T}^*$ such that when it is applied on the source-domain data, the induced distribution matches the observed target-domain marginal distribution of the features X. Examples for parametric forms of $\mathcal{T}^*$ are identity transformation (Zhang et al. [148]), location-scale transformation (Zhang et al. [148]) and a projection to an invariant subspace (Gong et al. [42]). Since $P(Y)$ and $P(X|Y)$ are independently changing causal modules, and due to the simple parametric form of $P(Y)$ in many supervised learning settings, they can be adapted independently. Typically $P(Y)$ can be adapted using importance weights $\beta(Y)$, in a similar fashion to covariate shift (Zhang et al. [148]), and $P(X|Y)$ can be adapted using various parametric transformations such as location-scale transformation (Zhang et al. [148]), a linear projection to an invariant subspace (Gong et al. [42]), or as a linear mixture of multiple observed conditional distributions (Zhang et al. [149]).

One of the main drawbacks of this line of work is the parametric assumptions that are made regarding the transformation $\mathcal{T}$ when matching the source and target domains. In Chapter 2 of this thesis, we explore a non-parametric way of making use of multiple labeled source domains in order to map to an unlabeled target domain and perform prediction.

1.2.3 Domain Adaptation via Inference on Learned Graphical Models

Given multiple domains with their respective joint distributions, it is possible use conditional-independence testing (Zhang et al. [147]), to find a probabilistic graphical model over the variables of interest, in which each conditional distribution that changes across domains is associated with an additional (latent) variable θ which represents its changing properties (Zhang et al. [145]), as presented in Figure 1.4. Here, we are shown a graphical model that consists of a target variable Y and seven other features X, and variables with incoming edges from θ are the ones whose conditional distributions given their parents change across domains ($P(X_1)$, $P(Y|X_1)$, $P(X_3|Y, X_2)$, $P(X_2|X_4, Y)$, $P(X_6|X_4)$), whereas shaded variables are the ones that are in the Markov Blanket of Y and are relevant for its prediction. Once the graph is learned, Generative Adversarial Networks (GAN) (Goodfellow et al. [44], Mirza and Osindero [87]) can be used to learn the conditional distributions, and stochastic variational inference (Hoffman et al. [58]) can be used to perform inference using the conditional distributions, in order to infer the posterior of Y given the relevant features (in this example, $P(Y|X_1, X_5, X_3, X_2, X_4)$), in the unlabeled target domain (Zhang et al. [150]).

1.2.4 Invariant Representation Learning via Neural Networks

So far we have discussed ways to break down the joint distribution into independently-changing factors, and depending on the change, apply an appropriate correction. However, as described so far, these approaches operate directly on the observed features X. When working with popular real-world applications such as text and images, the observed features have thousands of dimensions, many of which are redundant and noisy. Thus, even if the distribution matching approach described above for conditional-target shift can successfully reconstruct the target domain joint distribution in the infinite-sample setting, in practice it would suffer from very high estimation error. Specifically, a distribution metric d (such as the Jensen-Shannon Divergence) may suffer from curse of dimensionality when operating on high-dimensional raw features.

In the past decade deep neural networks have become a universally-used methodology to extract features from complex high-dimensional data for various supervised and unsupervised learning tasks. For example, convolutional neural networks (CNNs, (LeCun et al. [71])) have become ubiquitous in using as a backbone feature extractor for many supervised learning tasks in computer vision. One example is object classification in images, where a CNN can be represented as a function $f : \mathcal{X} \to \mathcal{Z}$ from the raw feature space to a hidden space $\mathcal{Z}$, on top of which a classifier like a multi-layer perceptron operates, given by a function $h : \mathcal{Z} \to \mathcal{Y}$ which predicts the label from the hidden representation. Such a framework for prediction has yielded state-of-the-art performance for most supervised learning on images and text. Given the promising empirical results, there was a clear need to combine the deep representation learning paradigm with the notion of distribution matching and domain adaptation.

One of the main motivations for formulating domain adaptation as a deep representation learning problem was a seminal theoretical analysis, presented by (Ben-David et al. [4]). In this study, given a hypothesis space $\mathcal{H}$ with VC dimension d, the authors provided a bound on the true risk in the target domain, given by $\epsilon_T(h) := \mathbb{E}_{X,Y\sim P^{\mathcal{T}}(X,Y)}[\mathbb{I}(h(X) \neq Y)]$, in terms of the following quantities:

- $\epsilon_S(h) := \mathbb{E}_{X,Y\sim P^{\mathcal{S}}(X,Y)}[\mathbb{I}(h(X) \neq Y)]$, the source domain risk, which can be estimated from the labeled source domain data,
- $d_{\mathcal{H}\Delta\mathcal{H}}(P^{\mathcal{S}}(X), P^{\mathcal{T}}(X))$, a divergence metric between the marginal distributions $P^{\mathcal{S}}(X)$ and $P^{\mathcal{T}}(X)$,
- $\lambda^* := \epsilon_S(h^*) + \epsilon_T(h^*)$, the *best achievable* joint risk in both domains in terms of the optimal hypothesis $h^* := \arg\min_{h\in\mathcal{H}} \epsilon_S(h) + \epsilon_T(h)$.

Given these three quantities, the bound is as follows:

Theorem 1 ((Ben-David et al. [4]))**.** *Let $\mathcal{H}$ be a hypothesis space of VC dimension d and $\hat{P}^{\mathcal{S}}(X)$ and $\hat{P}^{\mathcal{T}}(X)$ be the empirical distributions induced by n samples from $P^{\mathcal{S}}(X)$ and $P^{\mathcal{T}}(X)$ respectively. Then, with probability at least $1-\delta$, $h \in \mathcal{H}$:*

$$\epsilon_T(h) \leq \epsilon_S(h) + d_{\mathcal{H}\Delta\mathcal{H}}(\hat{P}^{\mathcal{S}}(X), \hat{P}^{\mathcal{T}}(X)) + \lambda^* + O(\sqrt{\frac{d\log n + \log 1/\delta}{n}}).$$

From this theorem, one can appreciate that the first two terms are entirely dependent on data that is observed and available for training in the unsupervised domain adaptation setting. Therefore, a plethora of studies ((Long et al. [77, 79, 81], Na et al. [89], Shu et al. [108])) have used this theoretical result as point of departure for designing deep learning architectures for domain adaptation. The first framework to do so was introduced by (Ganin and Lempitsky [38]), in which:

(1) novel techniques motivated by Generative Adversarial Networks (Goodfellow et al. [44]) were introduced used to learn a function $\phi : X \to \mathcal{Z}$ such that $d_{\mathcal{H}\Delta\mathcal{H}}(\hat{P}^{\mathcal{S}}(Z), \hat{P}^{\mathcal{T}}(Z)) = 0$, and

(2) the inferred representation Z is predictive of Y in the labeled source domain data (which can be done by training a predictor $g : \mathcal{Z} \to \mathcal{Y}$ simultaneously with ϕ).

Ensuring that $d_{\mathcal{H}\Delta\mathcal{H}}(\hat{P}^{\mathcal{S}}(Z), \hat{P}^{\mathcal{T}}(Z)) = 0$ essentially implies that $P^{\mathcal{S}}(Z) = P^{\mathcal{T}}(Z)$, which is why we refer to this sub-area of domain adaptation as *invariant representation learning*. The hope is that the overall mapping $h : \mathcal{X} \to \mathcal{Y}$ such that $h = g \circ \phi$ is predictive of Y in the target domain. Unfortunately, since labels are not observed in the target domain data, this cannot be guaranteed. This desired property is represented by λ^* in Theorem 1, which can be minimized if the model can ensure that $P^{\mathcal{S}}(Z|Y) = P^{\mathcal{T}}(Z|Y)$. Since the feature map ϕ and predictor g can be arbitrarily complex and are modeled by neural networks, this equality of the induced conditional distributions in latent feature space cannot be guaranteed, and the bound can be arbitrarily loose. The problem that results from $P^{\mathcal{S}}(Z|Y) \neq P^{\mathcal{T}}(Z|Y)$ and the lack of control of the λ^* term in the bound of Theorem 1 has prompted many attempts to mitigate this issue. For example, the study by (Long et al. [83]) introduces pseudo-labels and a joint distribution divergence measure in an attempt to align the joint distributions $P^{\mathcal{S}}(Z, Y)$ and $P^{\mathcal{T}}(Z, Y)$. In (Bousmalis et al. [12]), a decoder $\tilde{\phi} : \mathcal{Z} \to \mathcal{X}$ is used to ensure that Z contains meaningful invariant information, including predictive information about the target variable Y. Another approach, introduced in (Shu et al. [108]), is to ensure that the decision boundary of the predictor g does not cross high-density regions in the latent representation Z, thereby taking advantage of the cluster structure of the data.

While ongoing work has produced promising and increasing empirical performance on benchmark datasets, there is lack of reasoning what the latent features Z represent (from an interpretability stand-point), what information is required to infer them, and what properties of the data make it possible/impossible to do so. In Chapters 4 and 5 of this thesis, we view the invariant representation learning paradigm through the lens of the data-generating process, which allows us to reason about which information is needed to infer Z, how we can incorporate it into the invariant representation learning framework, and what other regularization techniques we can use to ensure that Z contains sufficient label-specific information in the target domain.

1.2.5 Semi-Supervised Learning Approaches to Domain Adaptation

Semi-supervised learning is a long-studied field of machine learning which specializes in making use of a large amount of unlabeled data to improve a predictor which has a very few labeled data-points for training (Chapelle et al. [16], Van Engelen and Hoos [127], Zhang et al. [153], Zhu [159]). Formally, instead of a source and target domain, the data-set is divided into labeled points $(\mathbf{x}^l, \mathbf{y}^l) = (\mathbf{x}^l_k, y^l_k)_{k=1}^{n_l}$, and unlabeled points $\mathbf{x}^u = (\mathbf{x}^u)_{k=1}^{n_u}$, drawn from $P(X, Y)$ and $P(X)$ respectively (note that here we only have one joint distribution, instead of two or more which is the case in domain adaptation). The approaches take advantage of the structure of the features X to inform the search for a predictor trained on the few labeled data-points. For example, this structure can be defined in terms of high-density and low-density regions of $P(X)$, and constraints can be enforced to ensure that the predictor does not cross high-density regions. This reasoning is exploited for the semi-supervised approaches of entropy minimization (Grandvalet et al. [46])

and self-training (Lee et al. [72]). Another way to define the structure is by assuming that the input space can be broken down into multiple low-dimensional manifolds, and samples on the same manifold should share the same class label (Zhou et al. [157]). These techniques have been recently adopted to unsupervised domain adaptation (French et al. [36], Jin et al. [67], Tarvainen and Valpola [125]), and shown to achieve state-of-art performance on several benchmark datasets (Zhang et al. [153]).

1.3 Thesis Objective and Outline

The main goal of our thesis is to incorporate the causal view (although some representations we use for domain adaptation are not necessarily causal without additional assumptions) into the unsupervised domain adaptation setting in order to contribute to the understanding of the problem, by reasoning under which settings it is (or is not) solvable. Furthermore, our aim is to provide methodology that has strong performance, intuitive justification and can be extended to novel neural architectures and more general/challenging settings, in a principled way. We make use of this view to contribute to more classical domain adaptation approaches/settings (such as covariate shift and conditional-target shift using classical prediction methods such as SVMs), and invariant representation learning via neural networks. Our hope is that this thesis will provide a unified picture for all three of these approaches. The outline of the thesis can be summarized as follows:

Chapter 2: "Data-Driven Approach to Multiple-Source Domain Adaptation" (Stojanov et al. [114]) tackles the unsupervised domain adaptation problem in which thare are multiple labeled source domains and a single unlabeled target domain. We make use of the data-generating process under the conditional-target shift setting, to design an algorithm that extracts the low-dimensional changes across domains, and makes use of them for prediction in the target domain.

Chapter 3: "Low-Dimensional Density Ratio Estimation for Covariate Shift Correction" (Stojanov et al. [115]) deals with the unsupervised domain adaptation problem under the covariate-shift setting. In particular, we develop an approach that makes use of the assumptions of covariate shift, along with reproducing kernel techniques, in order to reduce the dimensionality of the data for the purposes of density ratio estimation.

Chapter 4: "Domain Adaptation with Invariant Representation Learning: What Transformations to Learn?" (in submission) provides a novel approach for unsupervised domain adaptation using invariant representation learning, in which we make use of the data-generating process to formulate a notion of minimal change across domains based on mutual information, and use it to provide a principled regularizer that ensures that the latent representation Z contains useful information about the target variable Y.

Chapter 5: "Domain Adaptation via Direct Transformation" (pre-print) extends on the idea that more general techniques are needed to enforce minimal change of the joint distribution, within a neural architecture framework. In this chapter, we provide a more general notion of minimal change than mutual information, and provide two approaches to incorporate it within a deep learning domain adaptation algorithm.

Our hope is that each chapter of this thesis will provide the reader with sufficient insight regarding the respective domain adaptation sub-problem, its real-world applications, the core challenges that the state-of-the-art methods are facing, and how the view introduced here can be used to provide advances towards solving each problem. We also recognize that the approaches presented here are only scratching the surface regarding how generating process information and the concept of minimal changes can be used to improve transfer learning, and we further hope that this thesis contributes to novel and bold ideas not only in domain adaptation, but other areas of machine learning as well.

Chapter 2

Data-Driven Approach to Multiple-Source Domain Adaptation

A key problem in domain adaptation is determining what to transfer across different domains. We propose a data-driven method to represent these changes across multiple source domains and perform unsupervised domain adaptation. We assume that the joint distributions follow a specific generating process and have a small number of identifiable changing parameters, and develop a data-driven method to identify the changing parameters by learning low-dimensional representations of the changing class-conditional distributions across multiple source domains. The learned low-dimensional representations enable us to reconstruct the target-domain joint distribution from unlabeled target-domain data, and further enable predicting the labels in the target domain. We demonstrate the efficacy of this method by conducting experiments on synthetic and real datasets.

2.1 Introduction

In recent years machine learning techniques have become ubiquitous in solving real-world problems. For many of these applications obtaining new labeled data can be difficult, time-consuming, or expensive. Moreover, the training and test data are collected during different time periods and/or under different conditions, often yielding a shift in the distribution across datasets. For example, the distribution of medical data regarding a particular disease may vary from patient to patient because of heritable factors and different laboratory and measurement conditions. Furthermore, image datasets are collected in more than one setting, with different viewpoints and illumination conditions. Suppose we have one or more labeled datasets called source domains, and a new unlabeled dataset called target domain which has a different distribution from the source domain(s). Domain adaptation is the problem of accounting for the shift in distribution across domains such that relevant information is transferred from source domain(s) to the target domain so as to predict the target domain labels. Let X denote the features and Y denote the labels. In the multiple-source domain adaptation setting, there are $M > 1$ source domains in the training data generated from multiple respective joint distributions $P_{XY}^{(1)}, ..., P_{XY}^{(M)}$. The goal is to learn a classifier for a new target domain with unlabeled data generated from $P_X^{\mathfrak{T}}$. To enable successful domain transfer, one needs to make some assumptions about the joint distribution and take into account the generating process of the data. Following (Gong et al. [42], Zhang et al. [148, 149]), we assume that the causal direction is $Y \to X$; then $P_{X|Y}$ corresponds to the causal mechanism that generates features from the label. According to the modularity property of a causal model, $Y \to X$ implies that P_Y and $P_{X|Y}$ change independently across domains (Pearl [93], Schölkopf et al. [102], Spirtes et al. [112]). The generating process is illustrated on Figure 2.1. For example, in image classification, the class label can be considered as the cause of images. If we change the label distribution, this would not change the causal mechanism $P_{X|Y}$ that generates images from labels. The change of $P_{X|Y}$ can be due to other factors such as illumination and viewpoint. Thus, the factorization of the joint distribution following the causal direction (given by$P_Y P_{X|Y}$) is more favorable, because the other factorization yields factors P_X and $P_{Y|X}$ which arise from independent modules P_Y and $P_{X|Y}$ (via Bayes rule), and are thus coupled and change dependently across domains in the generic case.

Determination of what information to transfer from source domains to the target is a crucial issue in domain adaptation. In this paper, we propose a nonparametric approach to capture distribution changes and recover the target domain joint distribution. Since the causal direction is $Y \to X$, it is not surprising that the changes in the data generating process, $P_{X|Y}$, are usually simple and relatively easy to model. More specifically, we assume an infinite-dimensional nonparametric paradigm for the causal mechanism of all domains, i.e., $\{P_{X|Y;\Theta} : \Theta \in \mathit{\Theta}^{\infty}\}$, where $\mathit{\Theta}^{\infty}$ is an infinite-dimensional space of parameters. We show that if the number of changing parameters in $P_{X|Y}$ is small, $P^{(1)}_{X|Y}, \cdots, P^{(M)}_{X|Y}$ lie in a low-dimensional manifold. Given enough source domains, we can identify the manifold of the d changing parameters by learning low-dimensional representations of the distributions $P^{(1)}_{X|Y}, \cdots, P^{(M)}_{X|Y}$. Furthermore, we can make use of the low-dimensional representation to reconstruct the target-domain causal mechanism $P^{\mathcal{T}}_{X|Y}$, which can then be used to construct the target-domain classifier.

Therefore, the motivation of our approach is two-fold: **(1)** Working with a plausible representation for the generating process of the data allows us to observe a low-dimensional change across domains. **(2)** When factorized according to the generative process, the factors of the distribution (i.e. P_Y and $P_{X|Y}$) change independently, and their respective low-dimensional changes across domains can be learned separately. The proposed method leverages these properties to extract the low-dimensional representations of the changing parameters across domains, and make use of it for predicting target-domain labels.

2.1.1 Related Work

Classical single-source domain adaptation focuses on the setting where there are two domains, one labeled training dataset and one unlabeled test dataset (termed source and target domain), arising from two joint distributions $P^{\mathcal{S}}_{X,Y}$ and $P^{\mathcal{T}}_{X,Y}$, respectively. In order for classification in the test domain to be feasible, there must be some connection between the source and target domains, and this connection is reflected in the respective joint distributions. Therefore, domain adaptation approaches generally focus on understanding what aspects of the joint distribution change and leveraging this knowledge to account for the difference and construct an appropriate hypothesis in the target domain. Single-source domain adaptation has been extensively studied, with some of its

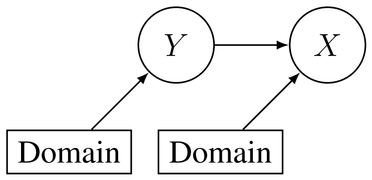


Figure 2.1: Generating process $Y \to X$ across domains with domain index variable $D = 1, ..., M$

theoretical underpinnings analyzed in (Jiang [66], Pan and Yang [90]) and (Ben-David et al. [8]).

When considering the change of P_{XY} across domains, prior approaches generally make some assumptions of changes in its factors. Namely, a large body of work has focused on the setting in which it is assumed that P_X changes and $P_{Y|X}$ remains the same. Approaches in this setting focus on minimizing the discrepancy in P_X between the weighted source domain and the target. This setting is called covariate shift or sample selection bias (Zadrozny [140]), and has been thoroughly studied in (Ben-David and Urner [6], Shimodaira [106], Sugiyama et al. [120]). However, in practice both P_X and $P_{Y|X}$ can change across domains. An alternative assumption to address this is the setting in which P_Y changes and $P_{X|Y}$ remains the same, a setting termed target shift (Zhang et al. [148]) or prior probability shift (Du Plessis and Sugiyama [31], Storkey [117]).

A particular line of work in single-source domain adaptation , (Gong et al. [42], Zhang et al. [148]) makes assumptions on the generative process across domains. With the assumption that the labels generate the features ($Y \to X$) (which is reasonable for many real-world scenarios including digit recognition and medical diagnosis) , the authors work with the factors $P_{X|Y}$ and P_Y, which are assumed to change independently under this causal generative process assumption (Woodward [134]). This property is leveraged to model the changes in $P_{X|Y}$ and P_Y separately, naturally reducing the complexity of the problem.

Recently there have also been deep learning approaches to single-source domain adaptation. They are based on constructing domain adaptation layers with the aim of learning transferable representations for classification under the covariate shift setting (Bousmalis et al. [13], Ganin et al. [39], Long et al. [78, 80], Shu et al. [108]). Recent work by (Long et al. [79]) focuses on the setting when multiple components of the joint distribution may change, and presents an architecture that extracts transferable representations that reduce the discrepancy of the joint distribution across domains.

There is a diverse body of work in multiple-source domain adaptation. Similar to single domain adaptation, (Muandet et al. [88]) learns domain invariant components that are shared by all domains and uses them for prediction in the target domain. Other approaches focus on combining multiple hypotheses from the source domains and weighing them based on the source-domain marginal distributions, $P_X^{(1)}, ..., P_X^{(M)}$, (Mansour et al. [85]), where the weights are determined in various ways (Chattopadhyay et al. [17], Duan et al. [32], Gao et al. [40]). Another approach (Blanchard et al. [11]) focuses on incorporating the marginal distribution P_X as an additional input of the classifier. However, P_X is an infinite-dimensional object, and performing direct comparisons on it across domains may lead to high estimation error and overfitting.

This fact is addressed by a method which assumes that the generating process is $Y \to X$ and that the change across domains follows the Conditional-Target Shift setting (described above) (Zhang et al. [149]). This approach performs domain adaptation by assuming that the target conditional distribution $P_{X|Y}^{\mathcal{T}}$ is a linear mixing of the conditional distributions in the source domains $P_{X|Y}^{(1)}, ..., P_{X|Y}^{(M)}$, and solving for the mixing weights. However, the linear mixture assumption imposes a rather strong constraint on the type of low-dimensional changes that can be modeled and accounted for across domains. The same can be said about another related body of work that aims to find a low-dimensional representation of relevant changes, called metric learning (Weinberger and Saul [133]) learns a distance matrix that operates on a learnt low-dimensional subspace of the data, such that relevant information for prediction is contained. However, this distance metric has a particular parametric form, and may discard meaningful information about the change in distribution across domains, if applied to domain adaptation.

In our approach, we follow the same domain adaptation setting, and we aim to automatically discover the (potentially nonlinear) low-dimensional changes across domains from data. This work consists of the following main contributions:
(1) We present a data-driven approach to capturing the low-dimensional manifold of the changes in the distribution across domains.
(2) We show that if the source- and target-domain joint distributions lie on a low-dimensional manifold, then the joint distribution in the target domain $P_{XY}^{\mathcal{T}}$ can be identified from the marginal distribution $P_X^{\mathcal{T}}$.
(3) We provide an algorithm that makes use of the low-dimensional manifold in order to reconstruct the joint distribution in the target domain and perform classification.

2.2 Theoretical Foundation

Closely following the multiple-source settings of binary classification of (Blanchard et al. [11]) and (Muandet et al. [88]), we let $\mathcal{X}$ be the input feature space, and let $\mathcal{Y} = \{-1, 1\}$ be the output space. Let $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$ be the family of joint distributions over $\mathcal{X} \times \mathcal{Y}$. Also, let $\mathcal{P}_{\mathcal{Y}}$ and $\mathcal{P}_{\mathcal{X}|\mathcal{Y}}$ be the respective families of distributions. Let there be a distribution μ on $\mathcal{P}_{\mathcal{X}\times\mathcal{Y}}$, where $P_{XY}^{(1)}, ..., P_{XY}^{(M)}$ are independent and identically distributed (i.i.d.) realizations from this family for the source domains, and $P_{XY}^{(\mathcal{T})}$ is the realization for the target domain. In what follows, we shall describe some of the mathematical tools required to represent and make use of the changing parameters across source domains.

2.2.1 Representing Distributions in Hilbert Space

To perform domain adaptation, one needs to compare probability distributions. Kernel mean embeddings provide a convenient way to represent probability distributions as points in a Reproducing Kernel Hilbert Space (RKHS) associated with some positive semi-definite kernel, where the distance between them can be easily computed (Smola et al. [109]).

Given a positive semi-definite kernel function k with corresponding RKHS $\mathcal{H}_k$ and a feature map $\psi : \mathcal{X} \to \mathcal{H}_k$ (s.t. for $x_1, x_2 \in \mathcal{X}$, $k(x_1, x_2) = \langle\psi(x_1), \psi(x_2)\rangle_{\mathcal{H}_k}$), the kernel mean

random variable	X	Y
domain	$\mathcal{X}$	$\mathcal{Y}$
feature map	$\psi(x)$	$\rho(y)$
kernel	$k(x, x')$	$l(y, y')$
i-th domain data point	$\mathbf{x}^{(i)}$	$\mathbf{y}^{(i)}$
empirical estimates of $P_X(x)$ and $P_Y(y)$	$\hat{P}_X(x)$	$\hat{P}_Y(y)$
kernel mean embedding on i-th domain	$\mu_X^{(i)}$	$\mu_Y^{(i)}$
feature map on kernel mean embedding	$\Phi(\mu_X)$	

Table 2.1: Notation used

embedding of the marginal distribution P_X is given by:

$$\mu_X := \int_{\mathcal{X}} k(x, \cdot) dP_X(x) = \mathbb{E}_{P_X}[\psi(x)]. \tag{2.1}$$

When k is a characteristic kernel (such as the Gaussian kernel), μ_X is a point in $\mathcal{H}_k$ that captures all the moments of P_X. A computationally convenient distance metric between two distributions $P_X^{(1)}$ and $P_X^{(2)}$ is their Euclidean distance in the high-dimensional embedding space, given by $d(P_X^{(1)}, P_X^{(2)}) \equiv ||\mu_X^{(1)} - \mu_X^{(2)}||^2$. It is also known as the Maximum Mean Discrepancy (MMD) (Gretton et al. [51]). A consistent estimator of the kernel mean embedding with finite n data points is $\hat{\mu}_X = \frac{1}{n}\sum_{i=1}^{n} \psi(x_i)$.

While the marginal distribution is fully represented as a single point in Hilbert space, the conditional distribution P_{XY} is represented by a set of a family of points in RKHS indexed by the conditioning variable Y (Song et al. [110]). Namely, given a kernel l corresponding to a feature map $\rho : \mathcal{Y} \to \mathcal{H}_l$ and kernel k corresponding to feature map $\psi : \mathcal{X} \to \mathcal{H}_k$, the conditional kernel mean embedding is given by the operator $\mathcal{U}_{X|Y}$, a mapping from $\mathcal{H}_l$ to $\mathcal{H}_k$. Using this operator, the kernel sum rule (Song et al. [110]) can be used to express the embedding of the marginal distribution P_X in terms of the independently changing factors $P_{X|Y}$ and P_Y: $\mu_X = \mathcal{U}_{X|Y}\mu_Y$. For a fixed value of the conditioning variable $Y = c$, the kernel mean embedding of $P_{X|Y=c}$ is given by:

$$\mu_{X|Y=c} := \int_{\mathcal{X}} k(x, \cdot) dP_{X|Y=c}(x) = \mathbb{E}_{P_{X|Y=c}}[\psi(x)].$$

It can be shown that when Y is discrete and $l(y_1, y_2) = \delta(y_1, y_2)$ is the Kronecker delta kernel, $\mathcal{U}_{X|Y} = [\mu_{X|Y=1}, ..., \mu_{X|Y=C}]^T$. Furthermore, similarly to the marginal case, the conditional kernel mean embedding for a fixed $Y = c$ can be estimated by $\hat{\mu}_{X|Y=c} = \frac{1}{n_c}\sum_{i=1}^{n_c} \psi(x_i)$, where n_c is the number of observations which have class c.

2.2.2 Identifying Low-Dimensional Changing Parameters

The goal of our method is to mathematically express and utilize the identifiable changing parameters in P_{XY} across domains via its independent factors, in our case $P_{X|Y}$ and P_Y. We work with $P_{X|Y}$ and demonstrate how this can be achieved. When performing kernel mean embedding of the conditional distributions of the features given a class label c in the the source domains, $P^{(1)}_{X|Y=c}, ..., P^{(M)}_{X|Y=c}$, we obtain M points in $\mathcal{H}_k$ given by $\mu^{(1)}_{X|Y=c}, ..., \mu^{(M)}_{X|Y=c}$. Let there be a kernel k_μ with an RKHS $\mathcal{H}_{k_\mu}$ and a corresponding feature map $\Phi : \mathcal{H}_k \rightarrow \mathcal{H}_{k_\mu}$. To extract the nonstationary components (parameters) of these distributions, one needs to find the transformations of the distributions with maximal variability. This can be achieved by performing Kernel Principcal Component Analysis (KPCA) (Schölkopf et al. [103]) on $\mu^{(1)}_{X|Y=c} \cdots, \mu^{(M)}_{X|Y=c}$, using an additional kernel k_μ, resulting in a centered kernel Gram Matrix: $\tilde{\mathbf{K}}_{ij} = k_\mu(\mu^{(i)}_{X|Y=c}, \mu^{(j)}_{X|Y=c})$. To show that this is the case, we first need the following lemma regarding linear PCA.

Lemma 2 *Let points $\phi_1, ..., \phi_M$ be p-dimensional vectors (where p could be infinite). Let $\lambda_1, ..., \lambda_q$ be the set of all non-zero eigenvalues after performing PCA on these vectors. If $P_\lambda(\phi_i)$ is the projection of ϕ_i on the principal eigenvectors corresponding to $\lambda_1, ..., \lambda_q$, then $\phi_i \neq \phi_j \iff P_\lambda(\phi_i) \neq P_\lambda(\phi_j)$.*

The proof can be found in section 2.7. Using this lemma we are now ready to formally establish the connection between the changing parameters of distributions and the outcome of KPCA performed on their kernel mean embeddings:

Theorem 1 *Let $P^{(1)}_{X|Y=c}, ..., P^{(M)}_{X|Y=c}$ be probability distributions with d identifiable changing parameters $\Theta_d = \theta_1, .., \theta_d$, and $\xi_1, ..., \xi_q$ be principal components resulting from KPCA with kernel k_μ on kernel mean embeddings $\mu^{(1)}_{X|Y=c} \cdots \mu^{(M)}_{X|Y=c}$. If k and k_μ are characteristic kernels, then $\xi_1, ..., \xi_q$ are a one-to-one mapping of the d changing parameters (i.e. $\xi_1, .., \xi_q = f(\theta_1, ..., \theta_d)$, where f is a bijective mapping.)*

Proof: The characteristic property of kernel k and identifiability of parameters $\theta_1, ..., \theta_d$ imply that $\Theta^{(1)}_d \neq \Theta^{(2)}_d \implies \mu^{(1)}_{X|Y=c} \neq \mu^{(2)}_{X|y=c}$ for the KMEs of two distributions $P^{(1)}_{X|Y=c}$ and $P^{(2)}_{X|Y=c}$, where $\Theta^{(1)}_d$ and $\Theta^{(2)}_d$ are their respective realizations of the d parameters. Performing KPCA on $\mu^{(1)}_{X|y=c} \cdots \mu^{(M)}_{X|y=c}$ using kernel k_μ results in non-zero eigenvalues $\lambda_1, ..., \lambda_q$. An important observation is that for a particular $\mu^{(i)}_{X|Y=c}$, its principal components corresponding to the non-zero eigenvalues, given by $\xi^{(i)}_{1,c}, ..., \xi^{(i)}_{q,c}$ are a function of $\mu^{(i)}_{X|Y=c}$ (i.e. $\xi^{(i)}_{1,c}, ..., \xi^{(i)}_{q,c} = g(\mu^{(i)}_{X|Y=c})$). Therefore, it suffices to show that g is a one-to-one function. By the characteristic property of k_μ, it follows that $\Theta^{(1)}_d \neq \Theta^{(2)}_d \implies \Phi(\mu^{(1)}_{X|Y=c}) \neq \Phi(\mu^{(2)}_{X|Y=c})$, and $\Phi(\mu^{(1)}_{X|Y=c}), ..., \Phi(\mu^{(M)}_{X|Y=c})$ are points in a q-dimensional subspace in $\mathcal{H}_{k_\mu}$. Since KPCA performs linear PCA on infinite-dimensional points $\Phi(\mu^{(1)}_{X|Y=c}), ..., \Phi(\mu^{(M)}_{X|Y=c})$, by Lemma 2, $\Phi(\mu^{(1)}_{X|Y=c}) \neq \Phi(\mu^{(2)}_{X|Y=c}) \iff \xi^{(1)}_{1,c}, ..., \xi^{(1)}_{q,c} \neq \xi^{(2)}_{1,c}, ..., \xi^{(2)}_{q,c}$, so g is a one-to-one function. This means that $\Theta^{(1)}_d \neq \Theta^{(2)}_d \implies \xi^{(1)}_{1,c}, ..., \xi^{(1)}_{q,c} \neq \xi^{(2)}_{1,c}, ..., \xi^{(2)}_{q,c}$, implying that f is a composition of one-to-one functions, and is itself a one-to-one function.$\square$

By establishing this one-to-one correspondence between the d changing parameters and the q principal components of KPCA, we have shown that the resulting q-dimensional manifold contains valuable low-dimensional information regarding the change of a particular factor of the joint distribution (in the proof treated as $P_{X|Y=c}$) across source domains $i = 1, ..., M$.

2.3 Algorithm

Now that we have a way of representing the changes of distributions across domains, we can use them to reconstruct the factors of the joint distribution in the target domain that will be used for classification. Given class labels $c = 1, ..., C$, the first step of the algorithm is to reconstruct the marginal distribution $P_X^{\mathcal{T}}$ by using the q-dimensional manifold of change across domains, such that the relevant factors are identified. The second step uses the reconstructed components $P_Y^{\mathcal{T}}$ and $P_{X|Y}^{\mathcal{T}}$ from the reconstructed marginal distribution in order to calculate $P_{Y|X}^{\mathcal{T}}$ and thus do classification in the target domain.

2.3.1 Reconstruction in the Target Domain

The main objective of our method is to identify the two factors of the joint distribution: $P_{X|Y=c}^{\mathcal{T}}$ and $P_{Y=c}^{\mathcal{T}}$, $\forall c$. All of the information about these two factors is contained in the marginal distribution of the target-domain $P_X^{\mathcal{T}} = \sum_{c=1}^{C} P^{\mathcal{T}}(X|y = c)P^{\mathcal{T}}(Y = c)$. Since we have access to unlabeled data points $\mathbf{x}_1^{\mathcal{T}}, ..., \mathbf{x}_{n_t}^{\mathcal{T}}$in the target domain, we can estimate the marginal distribution $\hat{P}_X^{\mathcal{T}}$, and search for factors $\hat{P}^{new}(X|y = c)$ and $\hat{P}^{new}(Y = c)$ that best reconstruct the marginal distribution estimate in terms of: $\hat{P}_X^{new} = \sum_{c=1}^{C} \hat{P}^{new}(X|y = c)\hat{P}^{new}(Y = c)$. Thus, we aim to find the respective factors which minimize a distance metric $d(\hat{P}_X^{\mathcal{T}}, \hat{P}_X^{new})$.

A computationally and statistically efficient procedure for minimization of the distance between the reconstructed and true marginal distribution is via Maximum Mean Discrepancy (MMD): $||\mu_X^{\mathcal{T}} - \mu_X^{new}||^2$, where $\mu_X^{\mathcal{T}}$ and μ_X^{new} are the kernel mean embeddings of $P_X^{\mathcal{T}}$ and P_X^{new} respectively (Gretton et al. [51]).

We parameterize conditional distribution mean embedding in the target domain as $\mu_{X|Y=c}^{new} = \mathbb{E}_{X\sim P_X^{\mathcal{T}}}[\beta_c(x)\psi(x)]$, where $\beta_c(x)$ represents the class-specific density ratio $P_{X|Y=c}^{\mathcal{T}}/P_X^{\mathcal{T}}$ which needs to be learned. Here, ψ is the feature transform into Hilbert space corresponding to the Gaussian kernel or another characteristic kernel, while for the label Y we have a feature map $\rho(y)$ corresponding to the Kronecker Delta kernel $k(x, y) = \delta(x, y)$, so the kernel mean embedding for the label is $\mu_Y = \mathbb{E}_{Y\sim P_Y^{\mathcal{T}}}[\rho(y)]$. For possible labels $y = 1, .., C$, the feature map of this kernel is the standard basis $\rho(y) = e_Y$ and the corresponding kernel mean embedding is: $\mu_Y = \mathbb{E}_{Y\sim P_Y^{\mathcal{T}}}[\rho(y)] = [P_{Y=1}, ..., P_{Y=C}]$.

In addition to this parameterization of the target domain, we are also given a q-dimensional manifold in $\mathcal{H}_k$ of the changing parameters of $P_{X|Y=c}$ across domains. We minimize the maximum mean discrepancy (MMD) (Gretton et al. [51]) between the marginal distribution of the target domain and its reconstruction $\mu_{X|Y=c}^{new}$, such that the reconstruction is as close as possible to the q-dimensional manifold. For this purpose, we introduce the following minimization criterion, given in population version:

$$\min_{\beta,\mu_Y} ||\mu_X^{\mathfrak{T}} - \mathcal{U}_{X|Y}^{new}\mu_Y^{new}||^2$$

$$\iff \min_{\beta,\mu_Y} ||\mu_X^{\mathfrak{T}} - \sum_{c=1}^{C} \mu_{X|Y=c}^{new}(\mu_Y)_c||^2 \tag{2.2}$$

$$\iff \min_{\beta,\mu_Y} ||\mu_X^{\mathfrak{T}} - \sum_{c=1}^{C} \mathbb{E}_{X\sim P_X^{\mathfrak{T}}}[\beta_c(x)\psi(x)](\mu_Y)_c||^2 \tag{2.3}$$

$$\text{s.t.} \sum_{c=1}^{C} ||\Phi(\mu_{X|Y=c}^{new}) - P_q\Phi(\mu_{X|Y=c}^{new})||^2 \leq \epsilon \tag{2.4}$$

$$\beta_c(x) \geq 0, \mathbb{E}_{X\sim P_X^{\mathfrak{T}}}[\beta_c(x)] = 1 \,\forall c \tag{2.5}$$

In the first constraint, (2.4), Φ represents an additional feature map corresponding to the Gaussian kernel k_μ (which we also use to perform Kernel PCA), and we use $P_q\Phi(\hat{\mu}_{X|Y=c}^{new})$ to represent the reconstruction of $\mu_{X|Y=c}^{new}$ onto the q-dimensional manifold described by the principal components of the source domains $(\mu_{X|Y=c}^{1}, ..., \mu_{X|Y=c}^{M})$ in the Gaussian Kernel feature space. Namely, if $\mathbf{v}_1, ..., \mathbf{v}_q$ are the eigenvectors corresponding to the nonzero eigenvalues in that feature space, then we let $\xi_{k,c}^{new} = (\mathbf{v}_k \cdot \Phi(\hat{\mu}_{X|Y=c}^{new})) = \sum_{i=1}^{M} \alpha_{i,c}^{k} k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{(i)})$ be the projection of $\hat{\mu}_{X|Y=c}^{new}$ on the k-th principal component, where $\boldsymbol{\alpha}_c$ vectors are eigenvectors of the centered Gaussian Kernel Gram Matrix $\tilde{\mathbf{K}}$ which was used to perform Kernel PCA on the source domains (Mika et al. [86]). Then $P_q\Phi(\mu_{X|Y=c}^{new}) = \sum_{k=1}^{n} \xi_k \mathbf{v}_k$, and the k-th eigenvector $\mathbf{v}_k$ can be expressed as the following linear combination: $\mathbf{v}_k = \sum_{l=1}^{M} \alpha_{i,c}^{k} \Phi(\mu_{X|Y=c}^{i})$. One should note that for each class label c we try to identify a separate low-dimensional manifold corresponding to the conditional distributions $P_{X|Y=c}^{(i)}$ of the source domains, and the regularizer penalizes the sum of reconstruction errors across all label-specific manifolds.

The last two constraints, given in (2.5), ensure that $P_{X|Y=c}^{\mathfrak{T}} = \beta_c(x)P_X^{\mathfrak{T}}$ is a valid distribution. The empirical version of the objective is:

$$\min_{\mathbf{B},\boldsymbol{\gamma}} ||\hat{\mu}_X^{\mathfrak{T}} - \hat{\mathcal{U}}_{X|Y}^{new}\hat{\mu}_Y^{new}||^2 \tag{2.6}$$

$$\iff \min_{\mathbf{B},\boldsymbol{\gamma}} ||\hat{\mu}_X^{\mathfrak{T}} - \sum_{c=1}^{C} \boldsymbol{\gamma}_j \frac{1}{n_{\mathfrak{T}}} \sum_{i=1}^{n_{\mathfrak{T}}} \mathbf{B}_{ic}\psi(x_i^{\mathfrak{T}})||^2 \tag{2.7}$$

$$\text{s.t.} \sum_{c=1}^{C} ||\Phi(\hat{\mu}_{X|Y=c}^{new}) - P_q\Phi(\hat{\mu}_{X|Y=c}^{new})||^2 \leq \epsilon \tag{2.8}$$

$$\mathbf{B}_{ic} \in [0, B_{max}] \text{ and } |\sum_{i=1}^{n_{\mathfrak{T}}} \mathbf{B}_{ic}| = n_{\mathfrak{T}}, \tag{2.9}$$

$$\forall c \in 1, 2, \ldots, C. \tag{2.10}$$

Here, $\mathbf{B} \in \mathbb{R}^{n_{\mathfrak{T}} \times C}$ contain the re-weighting coefficients that help reconstruct (estimate) the target conditional distribution given a specific class c: $\hat{P}_{X|Y=c}^{\mathfrak{T}} = \mathbf{B}_{:,c}\hat{P}_X^{\mathfrak{T}}$.

$\boldsymbol{\gamma}$ is used to estimate class probabilities (given by $\hat{P}_Y^{\mathfrak{T}}$, as a result of applying the Kronecker

Delta Kernel feature map) across all source domains, resulting in a new estimated marginal class probability in the target domain: $\hat{P}^{new}_{Y=c} = \boldsymbol{\gamma}_c$.

In order to make sure that $\mathbf{B}$ is a smooth function of the data, we reparameterize it as in (Zhang et al. [148]); namely, we let $\mathbf{B} = \mathbf{RA}$ where $\mathbf{R} = \mathbf{K}_B(\mathbf{K}_B + \lambda_B \mathbf{I})^{-1}$, where $\mathbf{K}_B$ is calculated using the Gaussian kernel with a separate width parameter σ_B, and regularized with a separate λ_B. We then minimize over $\mathbf{A} \in \mathbb{R}^{n_{\mathcal{T}} \times K}$ instead. After incorporating this reparameterization and putting the objective in Lagrange form, we have:

$$\begin{aligned}
&\min_{\mathbf{A},\boldsymbol{\gamma}} ||\hat{\mu}^{\mathcal{T}}_X - \hat{\mathcal{U}}^{new}_{X|Y}\hat{\mu}^{new}_Y||^2 + \\
&\quad \lambda_f (\sum_{c=1}^{C} ||\Phi(\hat{\mu}^{new}_{X|Y=c}) - P_q \Phi(\hat{\mu}^{new}_{X|Y=c})||^2) \\
\iff &\min_{\mathbf{A},\boldsymbol{\gamma}} ||\hat{\mu}^{\mathcal{T}}_X - \sum_{c=1}^{C} \boldsymbol{\gamma}_c \frac{1}{n_{\mathcal{T}}} \sum_{i=1}^{n_{\mathcal{T}}} (\mathbf{RA})_{ic} \psi(x^{\mathcal{T}}_i)||^2 + \qquad (2.11) \\
&\quad \lambda_f (\sum_{c=1}^{C} ||\Phi(\hat{\mu}^{new}_{X|Y=c}) - P_q \Phi(\hat{\mu}^{new}_{X|Y=c})||^2) \qquad (2.12) \\
&\text{s.t. } (\mathbf{RA})_{ic} \in [0, B_{max}] \text{ and } |\sum_{i=1}^{n_{\mathcal{T}}} (\mathbf{RA})_{ic}| = n_{\mathcal{T}}.
\end{aligned}$$

We use alternating optimization for this task; optimizing w.r.t $\boldsymbol{\gamma}$ is a straight-forward quadratic programming problem and optimizing w.r.t $\mathbf{A}$ can be done using a barrier method (for details see Section 2.6). The procedure is outlined in Algorithm 1.

2.3.2 Generative Classifier

Once we have the key components of the joint distribution in the target domain, we can perform classification in the target domain. The probability of the label given the data is

$$\begin{aligned}
\hat{P}^{\mathcal{T}}_Y(\mathbf{y}_i = c|\mathbf{x}^{\mathcal{T}}) &= \frac{\hat{P^{new}_Y}(\mathbf{y}_i = c)\hat{P}^{new}(\mathbf{x}^{\mathcal{T}}|\mathbf{y}_i = c)}{P^{\mathcal{T}}_X(\mathbf{x}^{\mathcal{T}})} \\
=& \hat{P^{new}_Y}(\mathbf{Y_i} = c)\mathbf{B}_{ic} \qquad (2.13)
\end{aligned}$$

We test the efficacy of this approach in the following section.

Algorithm 1 Classification Routine for Data-Driven Multi-Source Domain Adaptation

```
Input: (1) $M$ source domains with $n_i$ labeled training data-points: $(x_1, y_1), ..., (x_{n_i}, y_{n_i}) \sim P_{XY}^{(i)}$
       $\forall i \in 1, ..., M$.
       (2) A target domain with unlabeled data-points: $x_1, ..., x_{n_{\mathcal{T}}} \sim P_X^{\mathcal{T}}$
Output: predicted class labels in target domain: $\hat{\mathbf{y}}$
1: while not converged do
2:     Solve MMD problem given by (2.11) for $\boldsymbol{\gamma}$ using quadratic programming.
3:     Solve MMD problem given by (2.11) for $\mathbf{A}$ using barrier method.
4: end while
5: $\mathbf{B} = \mathbf{RA}$,
6: Return $\hat{P}_Y^{\mathcal{T}}(y_i = c|\mathbf{x}_i) = \boldsymbol{\gamma}_c \mathbf{B}_{ic}, \forall i \in 1, ..., n_{\mathcal{T}}, \forall c \in 1, ..., C$.
```

2.3.3 Identifability of Target Domain Conditional Distribution

The above algorithm identifies the separate components $P_{X|Y}$ and P_Y while reconstructing the marginal distribution P_X. Before presenting the identifiability result, we make some assumptions:

$\mathbf{A}_1$: For each value of c, the distribution $P_{X|Y=c}$ has only a finite number of parameters that change across possible domains. Suppose we have enough source domains, and let q be the number of non-zero eigenvalues of Gram matrix on $\mu_{X|Y=c}$ across all source domains.

Assumption $\mathbf{A}_1$ implies that there exists a nonlinear one-to-one transformation $h : \mathcal{P}_{\mathcal{X}|\mathcal{Y}} \to \mathbb{R}^q$. Then, the conditional distribution in each domain j is a linear combination of the other domains after such a transformation: $h(P_{X|Y=c}^{(j)}) = \sum_{i=1, i \neq j}^{M} \eta_{ic}^{j} h(P_{X|Y=c}^{(i)})$ for some weights $\eta_{1c}^{j}, .., \eta_{Mc}^{j}$. Furthermore, for the target domain $\mathcal{T}$, $\exists\ \boldsymbol{\eta}_c^*$ s.t. $h(P_{X|Y=c}^{\mathcal{T}}) = \sum_{i=1, i \neq j}^{M} \eta_{ic}^{*} h(P_{X|Y=c}^{(i)})$. In other words, all domain-specific conditional distributions for label c lie in a q-dimensional subspace of $\mathcal{H}_\mu$. This means that each conditional distribution corresponding to domain j can be uniquely determined by the mixture weights $\eta_{1c}^{j}, .., \eta_{Mc}^{j}$.

$\mathbf{A}_2$: Let $P_{X|Y=c}^{\boldsymbol{\eta}_c}$ be a distribution determined by weights $\boldsymbol{\eta}_c$, and $P_{X|Y=c}^{\boldsymbol{\eta}'_c}$ be determined by $\boldsymbol{\eta}'_c$. Then the elements of the set $\{p_{1c} P_{X|Y=c}^{\boldsymbol{\eta}_c} + p_{2c} P_{X|Y=c}^{\boldsymbol{\eta}'_c}; c = 1, .., C\}$ are linearly independent for $\forall \boldsymbol{\eta}_c, \boldsymbol{\eta}'_c, p_{1c}, p_{2c}, p_{1c}^2 + p_{2c}^2 \neq 0$.

We can now state the following identifiability theorem:

Theorem 2 *Let $\mathbf{A}_1$ and $\mathbf{A}_2$ hold, and $\hat{\eta}_c$ be the weights such that $P_{X|Y=c}^{new} = P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c}$ is the reconstructed distribution, namely $P_{X|Y=c}^{new} = \mathbf{B}_{:,c} P_X^{\mathcal{T}}$. If $\exists\ \hat{\boldsymbol{\eta}}_c$ s.t $P_X^{\mathcal{T}} = \sum_{c=1}^{C} P_Y^{new}(Y = c) P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c} = \sum_{c=1}^{C} \boldsymbol{\gamma}_c P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c}$, then we have $\forall$ c, $P_Y^{\mathcal{T}}(Y = c) = \boldsymbol{\gamma}_c$ and $P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c} = P_{X|Y=c}^{\mathcal{T}}$.*

2.4 Empirical Results

2.4.1 Baselines

We consider several baselines that can be used to perform classification in the target domain using data from multiple source domains:
(1) The simplest and most straightforward approach is to combine the data from all source domains and treat it as if it arose from a single joint distribution P_{XY} and use it for training via SVM. This approach is called “poolSVM”.

(2) The method introduced by (Mansour et al. [85]), in which the target-domain conditional distribution $P^{\mathfrak{T}}_{Y|X}$ is represented as a linear mixture of the source-domain marginal distributions, $P^{\mathfrak{T}}_{Y|X} = \sum_{i=1}^{M} \lambda_i P^{(i)}_{Y|X}$, where the weights are functions of the marginal distributions of the source domain, namely $\lambda_i = \frac{\tilde{\alpha}_i P^{(i)}_X}{\sum_{q=1}^{M} \tilde{\alpha}_q P^{(q)}_X}$. When they introduced the method, (Mansour et al. [85]) used uniform weights $\tilde{\alpha}_i = \frac{1}{M}\ \forall i \in 1, ..., M$. As described in (Zhang et al. [149]), the weights can also be learned using a kernel mean matching approach, such that $\sum_{i=1}^{M} \tilde{\alpha}_i P^{(i)}_X$ is as close to $P^{\mathfrak{T}}_X$ as possible (we refer to this approach as “dist-weight”).

(3) Treating the target conditional distribution as a uniform mixture of the source-domain conditional distributions, $P^{\mathfrak{T}}_{X|Y} = \frac{1}{M} \sum_{i=1}^{M} P^{(i)}_{X|Y}$. We refer to this baseline as "uniform".

(4) The algorithm proposed by (Zhang et al. [149]) which, like our approach, assumes the generative process $Y \to X$, and aims to use the relevant low-dimensional factors P_Y and $P_{X|Y}$, where the kernel mean embedding of $P_{X|Y}$ in the target domain is a linear mixture of the kernel mean embeddings in the source domains, namely: $\mu^{\mathfrak{T}}_{X|Y=c} = \sum_{i=1}^{M} \lambda_i \mu^{(i)}_{X|Y=c}$. The mixing weights are learned jointly with $P^{\mathfrak{T}}_Y$, and this information is used to do distribution-weighted combination of the classifiers in the source domains, like in the method by (Mansour et al. [85]). We denote this method by "dist-comb".

(5) The method proposed by (Blanchard et al. [11]), which uses a kernel SVM approach. The authors used the canonical SVM framework with a product kernel which, in addition to comparing data points, also compares marginal distributions across domains. This kernel is given by a product of two kernel functions: $k_B((P^{(i)}_X X_{iq}), (P^{(j)}_X, X_{jl})) = k_P(P^{(i)}_X, P^{(j)}_X) k_X(X_{iq}, X_{jl})$ between two points X_{iq} and X_{jl} of domains i and j. Here, k_P is a characteristic kernel that operates on probability distributions, and k_X is a kernel applied directly on the data points. We refer to this method as "marg-kernel".

2.4.2 Synthetic Datasets

In order to test the effectiveness of our proposed method, we perform the task of handwritten digit recongition on the MNIST (LeCun et al. [71]) dataset. This task satisfies the assumption of the generative process $Y \to X$, and is thus suitable for application of our approach. We performed two classification tasks; in the first one we classify digits 4 and 9, and in the second one we try to discern between digits 1 and 7. For each task, we create a multiple-source domain adaptation setting, where each domain represents a rotation of a digit with a different angle. We establish 20 such angles, with the difference of two adjacent domains (angles) being 18 degrees . Thus, in this setting, rotation is the only changing parameter across domains. Because of the choice of the changing parameter, this dataset violates the commonly required assumption that the target domain must be contained in the support of the source-domain joint distributions. We conduct 20 experiments, where each angle is treated as a target domain, and 10 other source angles are sampled randomly, while ensuring that the nearest source angle is at least 36 degrees away from the target. We sample 350 points for each source domain and the target domain. Because the dimensionality of images is high and we used a very simple approach to reduce it, we fixed $P(Y = c)$ to range between 0.2 and 0.8 for the two classes in order to prevent instability when estimating $P_{X|Y}/P_X$ in our generative approach via MMD.

We present the accuracies of all the baselines and the proposed method (termed "generative") in Figures 2.2 and 2.3, for the classification of digits "4" vs. "9" and "1" vs. "7" respectively. In addition, we also provide the average accuracy together with standard deviations and Wilcoxon signed rank tests in Table 2.2.

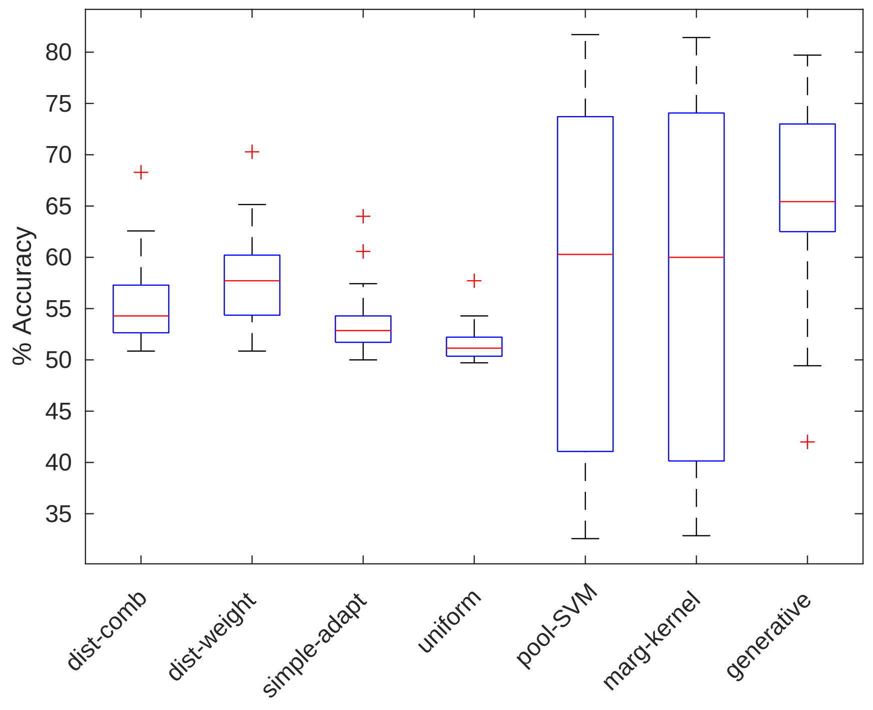


Figure 2.2: Accuracies of the baselines and the proposed method for the task of classifying between digits 4 and 9, for handwritten digit recognition

From the figures, one can see that the proposed method outperforms all the baselines. In particular, this performance gap is drastic in the task with digits 4 and 9, where the digits are difficult to classify and there is no possibility for support overlap between the source and

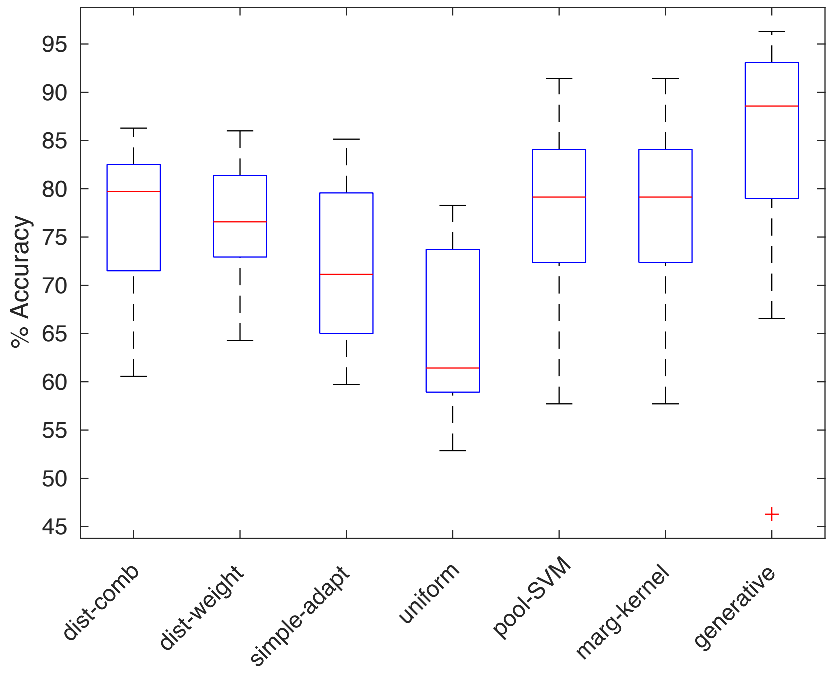


Figure 2.3: Accuracies of the baselines and the proposed method for the task of classifying between digits 1 and 7, for handwritten digit recognition

target domains because of reflexion. It demonstrates that our method is capable of utilizing the one-dimensional rotational changes across domains to perform classification on datasets with a complex decision boundary.

	dist-comb	dist-weight	simple-adapt	uniform	poolSVM	marg-kernel	generative
MNIST 4/9							
% accuracy	55.6391(4.43)	58.0(4.8)	54.0 (3.4)	51.4 (1.8)	57.93 (15.9)	58.0 (17.0)	**65.8 (9)**
p-value	0.0006	0.0033	0.0003	0.0001	0.0795	0.0766	–
MNIST 1/7							
% accuracy	76.81 (7.4)	76.76 (7.7)	72.0 (8)	64.96 (8.8)	77.74 (8.8)	77.72(12.3)	**84.4 (2.7)**
p-value	0.01	0.009	0.001	0.0003	0.035	0.035	–
Medical							
% accuracy	75.07(14.4)	79.39 (15.0)	81.76 (14.2)	81.75 (13)	76.7 (13.9)	81.41 (14.7)	**85.62 (7.4)**
p-value	0.0002	0.015	0.07	0.05	0.0005	0.08	–

Table 2.2: Accuracies and p values for Wilcoxon signed rank test across the baselines and the proposed method performed on: the MNIST dataset where we classify 4 vs. 9 (top), he MNIST dataset where we classify 1 vs. 7 (top), and the real dataset from lung lobe images (bottom). The p-values displayed are comparing the proposed method with each respective baseline.

2.4.3 Real Dataset

We also applied our method to lung phenotype data (CT images) from the COPDGene cohort, which is a public dataset for lung disease study. The task here is to detect the fissure between two lung lobes, which is a binary classification problem. This task is an important intermediate step towards understanding which genes are responsible for certain lung diseases. The fissure is represented by a 3D point set obtained by the method proposed in (Ross et al. [99]) and further refined by manual annotations. The goal is to classify whether the 3D points belong to one fissure region or another (represented by the positive and negative labels). Since the lung and fissure shape varies from patient to patient, the distributions of the points for the two fissures change across different patients. Furthermore, since labeling the points is costly and expensive, it would be very useful to be able to learn an optimal classifier on lung image data for a target patient by using existing labeled data from a few other patients by applying our method.

We conducted 40 experiments, in which randomly picked 7 source patients, and for each experiment we randomly sampled a target patient. We then subsampled 250 points for each patient (domain), such that $P(Y = 1)$ varies uniformly between 0.2 and 0.8 across all patients (both sources and target) for each experiment. We then performed classification using the generative method in each of the 40 target patients, and we present the accuracies in Figure 2.4 and Table 2.2. From these real dataset experiments, we see that our method outperforms all of the baselines. We

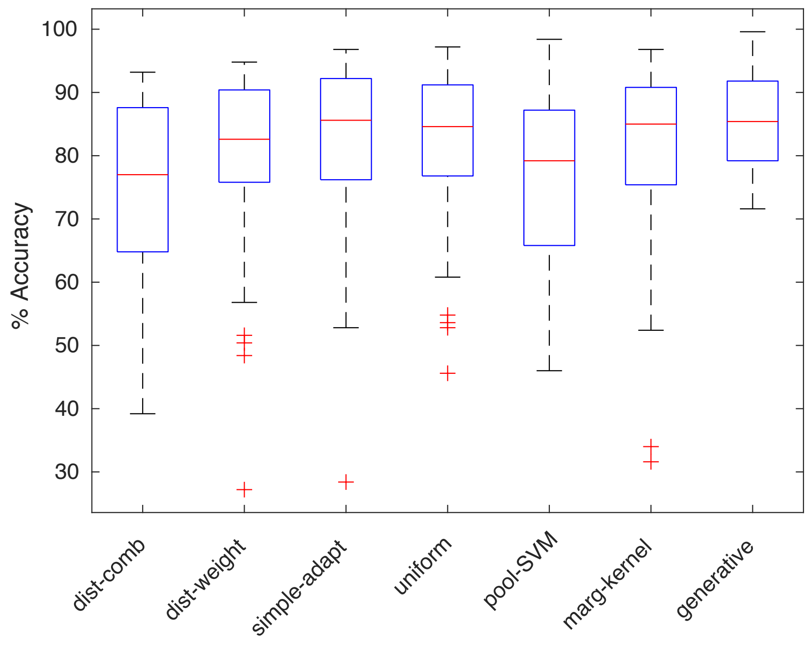


Figure 2.4: Accuracies of the baselines and the proposed method for the task of classifying between two different lung fissures in the real dataset

also note that all of the baselines have a much higher variance probably due to larger differences between the distributions of the target patient and source patients in some of the experiments.

2.5 Conclusion

We developed a data-driven method to discover and utilize low-dimensional changes of the joint distribution across domains for the purpose of domain adaptation. We did so by representing and exploiting the low-dimensionality of the change of the causal mechanism $P_{X|Y}$ across source domains. Out approach consists of two steps: (1) reconstructing the marginal distribution in the target-domain $P_X^{\mathcal{T}}$ such that $P_{X|Y=c}^{\mathcal{T}}$ and $P_Y^{\mathcal{T}}$ can be identified, and (2) using the reconstructed joint distribution in the target domain to perform classification. We have proven that this method is theoretically well grounded and have demonstrated its increased efficacy compared to the baselines via synthetic and real data experiments. We believe that this method opens the door for more flexible and principled data-driven approaches to domain adaptation.

2.6 Derivation of Algorithm and Implementation Details

2.6.1 Derivations of Algorithm and Reconstruction of Joint Dinstribution

We now expand the squares of equations (2.11) and (2.12) to express the objective only in terms of kernel Gram matrices:

$$\min_{\mathbf{A},\boldsymbol{\gamma}} \sum_{j=1}^{C}\sum_{j'=1}^{C} \gamma_j \gamma_{j'} \frac{1}{n_t^2} \sum_{i=1}^{n_t}\sum_{i'=1}^{n_t} k(x_i^t, x_{i'}^t)(\mathbf{RA})_{ij}(\mathbf{RA})_{i'j'} \tag{2.14}$$

$$-2\sum_{j=1}^{C} \gamma_j \frac{1}{n_t^2} \sum_{i=1}^{n_t}\sum_{i'=1}^{n_t} (\mathbf{RA})_{ij} k(x_i^t, x_{i'}^t) \tag{2.15}$$

$$+\lambda_f[-\sum_{c=1}^{C} 2 \sum_{k=1}^{n_d} \xi_{k,c}^{new} \sum_{i=1}^{M} \alpha_{i,c}^k k_\mu(\hat{\mu}_{X|Y=c}^i, \hat{\mu}_{X|Y=c}^{new})) \tag{2.16}$$

$$+\sum_{k=1}^{n_d}\sum_{k'=1}^{n_d} \xi_{k,c}^{new}\xi_{k',c}^{new} \sum_{i=1}^{M}\sum_{i'=1}^{M} \alpha_{i,c}^k \alpha_{i',c}^{k'} k_\mu(\hat{\mu}_{X|Y=c}^i, \hat{\mu}_{X|Y=c}^{i'})] \tag{2.17}$$

where k_μ corresponds to the Gaussian kernel function used to perform Kernel PCA, and $(\mathbf{RA})_{:,i}$ denotes the i-th column of $\mathbf{RA}$. We observe that $k_\mu(\hat{\mu}_{X|Y}^{new}, \hat{\mu}_{X|Y}^{new}) = 1$, so we omit this term as well. Here, n_d represents the number of dimensions we use to reconstruct $\Phi(\hat{\mu}_{X|Y=c}^{new})$ on the low-dimensional manifold. For completeness, we fully expand the regularization term:

$$-2\sum_{c=1}^{C}\sum_{k=1}^{n_d}\sum_{i=1}^{M}\sum_{j=1}^{M} \alpha_{i,c}^k \alpha_{j,c}^k k_\mu(\hat{\mu}_{X|Y=c}^i, \hat{\mu}_{X|Y=c}^{new}) k_\mu(\hat{\mu}_{X|Y=c}^j, \hat{\mu}_{X|Y=c}^{new}) +$$

$$\sum_{k=1}^{n_d}\sum_{k'=1}^{n_d}\sum_{j=1}^{M}\sum_{j'=1}^{M} \alpha_{j,c}^k \alpha_{j',c}^{k'} k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^j) k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{j'}) \sum_{i=1}^{M}\sum_{i'=1}^{M} \alpha_{i,c}^k \alpha_{i',c}^{k'} k_\mu(\hat{\mu}_{X|Y=c}^i, \hat{\mu}_{X|Y=c}^{i'})$$

For simplicity, we assume binary classification and we differentiate with respect to one of the columns of $\mathbf{A}$ (ex. the second row corresponding to the label 1, denoted by $\mathbf{a}_1$). To maintain clarity, we denote the above terms as $T_{1,..,4}$ and we will differentiate each w.r.t $\mathbf{a}_1$. For the first two terms T_1 and T_2 we have:

$$\frac{\partial T_1}{\partial \mathbf{a}_1} = \frac{2}{n_t^2}[\gamma_2 \sum_{j'=1}^{C} \gamma_{j'} \sum_{i=1}^{n_t} \sum_{i'=1}^{n_t} k(x_i^t, x_{i'}^t) \mathbf{R}_{i,:}(\mathbf{R}_{i',:}^T \mathbf{a}_{j'}) + \gamma_2^2 \sum_{i=1}^{n_t} \sum_{i'=1}^{n_t} k(x_i^t, x_{i'}^t) \mathbf{R}_{i,:}(\mathbf{R}_{i',:}^T \mathbf{a}_1)] \quad (2.18)$$

$$\frac{\partial T_2}{\partial \mathbf{a}_1} = 2\gamma_2 \frac{1}{n_t^2} \sum_{i=1}^{n_t} \sum_{i'=1}^{n_t} k(x_i^t, x_{i'}^t) \mathbf{R}_{i,:} \quad (2.19)$$

$$(2.20)$$

We now address the second two terms (T_3 and T_4) that correspond to the regularizer:

$$\frac{\partial T_3}{\partial \mathbf{a}_1} = -2 \sum_{i=1}^{M} \sum_{i'=1}^{M} [\frac{\partial k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i})}{\partial \mathbf{a}_1} k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i'}) + \quad (2.21)$$

$$\frac{\partial k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i'})}{\partial \mathbf{a}_1} k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i})] \sum_{k=1}^{n_d} \alpha_{i,c}^k \alpha_{i',c}^k \quad (2.22)$$

$$\frac{\partial T_4}{\partial \mathbf{a}_1} = \sum_{k=1}^{n_d} \sum_{k'=1}^{n_d} \sum_{i=1}^{M} \sum_{i'=1}^{M} \sum_{j=1}^{M} \sum_{j'=1}^{M} \alpha_{i,c}^k \alpha_{i',c}^k \alpha_j^{k'} \alpha_{j',c}^{k'} [[\frac{\partial k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i})}{\partial \mathbf{a}_1} k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{j}) + \quad (2.23)$$

$$\frac{\partial k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{j})}{\partial \mathbf{a}_1} k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i})] \quad (2.24)$$

where:

$$\frac{\partial k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i})}{\partial \mathbf{a}_1} = -\frac{1}{2\sigma^2}(\frac{(\hat{\mu}_{X|Y=c}^{new})^T(\hat{\mu}_{X|Y=c}^{i})}{\partial \mathbf{a}_1} + \frac{(\hat{\mu}_{X|Y=c}^{new})^T(\hat{\mu}_{X|Y=c}^{new})}{\partial \mathbf{a}_1}) k_\mu(\hat{\mu}_{X|Y=c}^{new}, \hat{\mu}_{X|Y=c}^{i}) \quad (2.25)$$

$$(2.26)$$

and:

$$\frac{(\hat{\mu}_{X|Y=c}^{new})^T(\hat{\mu}_{X|Y=c}^{i})}{\partial \mathbf{a}_1} = \frac{\partial[\frac{1}{n_1^i n_t} \mathbf{1}^T \mathbf{K}_1^{i,c} \text{diag}((\mathbf{RA})_{:,1})\mathbf{1}]}{\partial \mathbf{a}_1} \quad (2.27)$$

$$= \frac{\partial[\frac{1}{n_1^i n_t} \mathbf{1}^T \mathbf{K}_1^{i,c} (\mathbf{RA})_{:,1}]}{\partial \mathbf{a}_1} = \frac{1}{n_1^i n_t}(\mathbf{K}_1^{i,c}\mathbf{R})^T \mathbf{1} \quad (2.28)$$

$$\frac{(\hat{\mu}_{X|Y}^{new})^T(\hat{\mu}_{X|Y}^{new})}{\partial \mathbf{a}_1} = \frac{\partial[\frac{1}{n_t^2} \mathbf{1}^T \text{diag}((\mathbf{RA})_{:,1} \mathbf{K}_t \text{diag}((\mathbf{RA})_{:,1})\mathbf{1}]}{\partial \mathbf{a}_1} \quad (2.29)$$

$$= \frac{\partial[\frac{1}{n_t^2}((\mathbf{RA})_{:,1})^T \mathbf{K}_t (\mathbf{RA})_{:,1}]}{\partial \mathbf{a}_1} = \frac{1}{n_t^2} \frac{\partial[\mathbf{a}_1^T \mathbf{R}^T \mathbf{K}_t \mathbf{R} \mathbf{a}_1]}{\partial \mathbf{a}_1} = \frac{1}{n_t^2} 2\mathbf{R}^T K_t \mathbf{R} \mathbf{a}_1 \quad (2.30)$$

Here, $\mathbf{K}_t$ is the Gram matrix for the kernel mean embedding kernel of the target data, and $\mathbf{K}_1^{i,c}$ is the cross-kernel matrix between the target-domain data, and the data of source i with label 1. n_t is the number of samples in the target domain, and n_1^i is the number of samples in the i-th source domain that have a label 1. Furthermore, for the mean embeddings of conditional distributions conditioned on discrete labels, we have:

$$\hat{\mu}^{new}_{X|Y=1} = [\frac{1}{n_t}(\psi(\mathbf{x}^T)\text{diag}((\mathbf{RA})_{:,1}))\mathbf{1}] \tag{2.31}$$

$$\hat{\mu}^{i}_{X|Y=1} = [\frac{1}{n_1^i}\psi(\mathbf{x}^T)\mathbf{1}] \tag{2.32}$$

Vectorization of Objective

We rewrite the objective function in terms of matrix operations:

$$\min_{\boldsymbol{\gamma},\mathbf{A}} \frac{1}{n_t^2}\boldsymbol{\gamma}^T F\boldsymbol{\gamma} - \frac{2}{n_t}\boldsymbol{\gamma}^T(\mathbf{K1})^T(\mathbf{RA}) + \lambda_f R(\mathbf{A}) \tag{2.33}$$

where $\mathbf{F}_{j,j'} = (\mathbf{RA})^T_{:,j}[(\mathbf{K1}) \odot (\mathbf{RA})_{:,j'}]$. $R(\mathbf{A})$ represents a regularization term, now given by a summation over labels:

$$\sum_{c=1}^{C} ||\Phi(\hat{\mu}^{new}_{X|Y=c}) - P_n\Phi(\hat{\mu}^{new}_{X|Y=c})||^2 = \sum_{c=1}^{C}[k_\mu(\hat{\mu}^{new}_{X|Y=c}, \hat{\mu}^{new}_{X|Y=c}) - \tag{2.34}$$

$$2\sum_{k=1}^{n_d}\boldsymbol{\alpha}_c^{kT}\mathbf{D}\boldsymbol{\alpha}_c^k + \sum_{k=1}^{n_d}\sum_{k'=1}^{n_d}(\boldsymbol{\alpha}_c^{kT}\mathbf{D}\boldsymbol{\alpha}_c^{k'})(\boldsymbol{\alpha}_c^{kT}\tilde{\mathbf{K}}\boldsymbol{\alpha}_c^{k'})] \tag{2.35}$$

$k_\mu(\hat{\mu}^{new}_{X|Y=c}, \hat{\mu}^{new}_{X|Y=c}) = 1$ so it is a constant we can ignore for the optimization. Here, $\mathbf{D}_{ij} = k_\mu(\hat{\mu}^{new}_{X|Y=c}, \hat{\mu}^{i}_{X|Y=c})k_\mu(\hat{\mu}^{new}_{X|Y=c}, \hat{\mu}^{j}_{X|Y=c})$, and $\tilde{\mathbf{K}}$ is the Gram matrix over the embeddings of the source domains. Now we write down the derivatives in vectorized form:

$$\frac{\partial T_1}{\partial \mathbf{a}_1} = \frac{2\gamma_{\mathbf{1}}}{n_t}((\mathbf{Ra}_1) \odot (\mathbf{K1})^T\mathbf{R} + \frac{2\gamma_{\mathbf{1}}}{n_t^2}\sum_{j'=1}^{K}\gamma_{j'}((\mathbf{Ra}_{j'}) \odot (\mathbf{K1})^T\mathbf{R} \tag{2.36}$$

$$\frac{\partial T_2}{\partial \mathbf{a}_1} = \frac{2\gamma_{\mathbf{1}}}{n_t^2}(\mathbf{K1})^T\mathbf{R} \tag{2.37}$$

$$\frac{\partial T_3}{\partial \mathbf{a}_1} = \sum_{k=1}^{n_d}\alpha_1^{kT}\mathbf{Q}\alpha_1^k \tag{2.38}$$

$$\frac{\partial T_4}{\partial \mathbf{a}_1} = \sum_{k=1}^{n_d}\alpha_1^{kT}\mathbf{Q}\alpha_1^k\alpha_1^{kT}\tilde{\mathbf{K}}\alpha_1^k \tag{2.39}$$

where:

$$\mathbf{Q}_{ii'} = [\frac{\partial k_\mu(\hat{\mu}^{new}_{X|Y=1}, \hat{\mu}^{i}_{X|Y=1})}{\partial \mathbf{a}_1}k_\mu(\hat{\mu}^{new}_{X|Y=1}, \hat{\mu}^{i'}_{X|Y=1}) + \frac{\partial k_\mu(\hat{\mu}^{new}_{X|Y=1}, \hat{\mu}^{i'}_{X|Y=1})}{\partial \mathbf{a}_1}k_\mu(\hat{\mu}^{new}_{X|Y=1}, \hat{\mu}^{i}_{X|Y=1})] \tag{2.40}$$

Having the first derivatives of the objective with respect to colums of $\mathbf{A}$ is sufficient for implementing a barrier method using most packages (we used fmincon() in MATLAB), which then use the first derivatives to approximate the second derivatives during optimization.

2.6.2 Hyperparameter Tuning

Here we describe the hyperparameter settings in the baselines and the proposed method in the experiments. For the "poolSVM" method, we tuned the slackness parameter and the kernel width using 5-fold cross-validation on the pooled training data. For the "marg-kernel" method, we used the optimal slackness and k_X kernel width according to cross-validation via "poolSVM". For the k_P kernel, we varied the kernel width on a grid of values proportional to the median pairwise distance of the training data pooled together. For each dataset, we present the results with a k_P kernel width with the best average performance. For the methods "dist-weight" and "dist-comb", for all kernel mean embeddings we also tried several different values proportional to the median pairwise distance of the training data, and we report the results corresponding to the kernel width with best average performance for each respective baseline.

For our method, we set the kernel mean embedding kernel widths corresponding to k and k_μ, in the same manner as "dist-weight" and "dist-comb",with the only difference that we used only 5 percent of the experiments in each dataset in order to pick kernel widths with the best average performance, and then report the accuracies for all experiments using the selected kernel widths. Regarding the other hyperparameters of our method, in all experiments we fixed σ_B to the median pairwise distance of the data in the target domain. We kept $\lambda_f = 1$ and $\lambda_R = 0.1$.

2.7 Proofs of Theorems and Lemmata

2.7.1 Proof of Lemma 2

We restate the lemma here:

Let points $\phi_1, ..., \phi_M$, be p-dimensional vectors (where p could be infinite). Let $\lambda_1, ..., \lambda_q$ be the set of all non-zero eigenvalues after performing PCA on these vectors. If $P_\lambda(\phi_i)$ is the projection of ϕ_i on the principal eigenvectors corresponding to $\lambda_1, ..., \lambda_q$, then $\phi_i \neq \phi_j \iff P_\lambda(\phi_i) \neq P_\lambda(\phi_j)$

Proof: Let $\phi_i \neq \phi_j$. Then expressing ϕ_i and ϕ_j in terms of a Fourier expansion on eigenvectors corresponding to the nonzero eigevalues ($\mathbf{v}_1, ..., \mathbf{v}_q$), we obtain $P_\lambda(\phi_i) = \sum_{k=1}^{q}(\phi_i^T \mathbf{v}_k)\mathbf{v}_k$, $P_\lambda(\phi_j) = \sum_{k=1}^{q}(\phi_j^T \mathbf{v}_k)\mathbf{v}_k$, and there are no residual terms because the rest of the eigenvalues are zero. Then it immediately follows that $P_\lambda(\phi_i) \neq P_\lambda(\phi_j)$.

2.7.2 Proof of Theorem 2

We restate the theorem here:

Theorem 2 *Let $\mathbf{A}_1$ and $\mathbf{A}_2$ hold, and $\hat{\eta}_c$ be the weights such that $P^{new}_{X|Y=c} = P^{\hat{\boldsymbol{\eta}}_c}_{X|Y=c}$ is the reconstructed distribution, namely $P^{new}_{X|Y=c} = \mathbf{B}_{:,c}P^{\mathfrak{T}}_X$. If $\exists\ \hat{\boldsymbol{\eta}}_c$ s.t $P^{\mathfrak{T}}_X = \sum_{c=1}^{C} P^{new}_Y(Y=c)P^{\hat{\boldsymbol{\eta}}_c}_{X|Y=c} = \sum_{c=1}^{C} \boldsymbol{\gamma}_c P^{\hat{\boldsymbol{\eta}}_c}_{X|Y=c}$, then we have $\forall$ c, $P^{\mathfrak{T}}_Y(Y=c) = \boldsymbol{\gamma}_c$ and $P^{\hat{\boldsymbol{\eta}}_c}_{X|Y=c} = P^{\mathfrak{T}}_{X|Y=c}$.*

Proof: Making use of $\mathbf{A}_1$ to express $P_X^{\mathfrak{T}}$ as $P_X^{\mathfrak{T}} = \sum_{c=1}^{C} P_Y^{\mathfrak{T}}(Y = c)P_{X|Y=c}^{\boldsymbol{\eta}_c^*}$. After solving (6), we can achieve: $P_X^{\mathfrak{T}} = P_X^{new} \implies \sum_{c=1}^{C} P_Y^{\mathfrak{T}}(Y = c)P_{X|Y=c}^{\boldsymbol{\eta}_c^*} = \sum_{c=1}^{C} \boldsymbol{\gamma}_c P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c} \implies \sum_{c=1}[\boldsymbol{\gamma}_c P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c} - P_Y^{\mathfrak{T}}(Y = c)P_{X|Y=c}^{\boldsymbol{\eta}_c^*}] = 0$. By $\mathbf{A}_2$ we can conclude that $\forall\, c, \boldsymbol{\gamma}_c P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c} - P_Y^{\mathfrak{T}}(Y = c)P_{X|Y=c}^{\boldsymbol{\eta}_c^*} = 0$. Taking the integral of the last expression yields $P_Y^{\mathfrak{T}}(Y = c) = \boldsymbol{\gamma}_c$, and therefore $P_{X|Y=c}^{\hat{\boldsymbol{\eta}}_c} = P_{X|Y=c}^{\boldsymbol{\eta}_c^*} = P_{X|Y=c}^{\mathfrak{T}}$.

Chapter 3

Low-Dimensional Density Ratio Estimation for Covariate Shift Correction

Covariate shift is a prevalent setting for supervised learning in the wild when the training and test data are drawn from different time periods, different but related domains, or via different sampling strategies. This paper addresses a transfer learning setting, with covariate shift between source and target domains. Most existing methods for correcting covariate shift exploit density ratios of the features to reweight the source-domain data, and when the features are high-dimensional, the estimated density ratios may suffer large estimation variances, leading to poor performance of prediction under covariate shift. In this work, we investigate the dependence of covariate shift correction performance on the dimensionality of the features, and propose a correction method that finds a low-dimensional representation of the features, which takes into account feature relevant to the target Y, and exploits the density ratio of this representation for importance reweighting. We discuss the factors that affect the performance of our method, and demonstrate its capabilities on both pseudo-real data and real-world applications.

3.1 Introduction

We are concerned with the learning problem where we are given labeled training (source-domain) data $(x_1^{tr}, y_1^{tr}), ..., (x_n^{tr}, y_{n_{tr}}^{tr}) \subseteq \mathcal{X} \times \mathcal{Y}$, generated from joint distribution P_{XY}^{tr}, and aim to find a function that can predict the target Y from the features X on test (target-domain) data $(x_1^{te}, y_1^{te}), ..., (x_n^{te}, y_{n_{te}}^{te}) \subseteq \mathcal{X} \times \mathcal{Y}$, generated by P_{XY}^{te}, where the labels y^{te} are not observed. While most off-the-shelf supervised learning algorithms assume that $P_{XY}^{tr} = P_{XY}^{te}$, this might not be the case in practice. For example, consider the task of predicting a prognostic outcome in cancer patient cohorts given abundant clinical and molecular data such as gene expression. The data would often be collected from different populations and may be generated and processed under different lab conditions for the training and test cohorts. In this case, assuming that the joint distributions in the two domains are identical may lead to poor prediction performance.

Covariate shift (aka sample selection bias) (Heckman [56], Storkey [117], Zadrozny [140]) is the transfer learning setting in which $P_{XY}^{tr} \neq P_{XY}^{te}$ where the distribution of the features changes between the training and test domains ($P_X^{tr} \neq P_X^{te}$), with the assumption that $P_{Y|X}^{tr} = P_{Y|X}^{te}$.

The general approach to accounting for this particular distribution difference is to re-weight the source-domain labeled data such that the weighted data and the target-domain data have the same distribution, and then incorporate this weight information into the appropriate supervised learning procedure (Gretton et al. [50], Kanamori et al. [68], Sugiyama et al. [120], Yamada et al. [138]). More formally, the goal is to the minimize the risk under the test data distribution, given by $R^{te}(l) = \mathbb{E}_{(X,Y)\sim P^{te}_{XY}}[l(x, y; \theta)]$. Density ratio-based covariate shift correction aims to find a re-weighting function $\beta(x)$ such that the reweighted risk in the source domain given by $R^{tr}_{\beta}(l) = \mathbb{E}_{(X,Y)\sim P^{tr}_{XY}}[\beta(x)l(x, y; \theta)]$ matches the risk under the test data distribution (i.e. $R^{tr}_{\beta}(l) = R^{te}(l)$). The optimal function β is given by the density ratio $\beta(x) = \frac{P^{te}_X(x)}{P^{tr}_X(x)}$.

In density ratio-based covariate shift correction, while $\hat{\beta}$ is a consistent estimator of the density ratio, it can suffer high variance in the finite sample case, as initially demonstrated in (Shimodaira [106]). A key contributing factor to this variance in the estimate is the dimensionality of the data, and this is very apparent if one attempts to estimate the densities $p^{te}(x)$ and $p^{tr}(x)$ from data and then calculate their ratio. In high dimensions, dividing by an estimated quantity like a density can amplify the error (Sugiyama et al. [122]). To avoid estimating the density and performing the division explicitly, various methods have been developed to find the density ratio directly via criteria such as moment matching (Gretton et al. [50]), KL divergence, (Sugiyama et al. [120]), and relative Pearson divergence via least squares density estimation (Yamada et al. [138]), and thus achieve better statistical and/or computational efficiency. However, even if β is estimated very accurately, the prediction risk in the target domain may suffer high variance if the dimensionality of the features is high. This indicates that reducing the data dimensionality may improve prediction performance in the target domain.

To cope with this problem, there have also been efforts to reduce the dimensionality used in estimating the density ratio by searching for a low-dimensional subspace where the marginal distributions of the source and target domains are different; see, e.g., the method of Least-Squares Hetero-distributional Subspace Search (LHSS) (Sugiyama et al. [121]). Another way to cope with high dimensionality is by expanding the density ratio in terms of eigenfunctions of a kernel-based operator (Izbicki et al. [64]). While these directions have shown improvements in estimating the density ratio, they do not take into account the relevance of the features to the target variable Y; as a consequence, they may risk discarding useful information for prediction, and may still have unnecessarily high variance in the estimated density ratio (e.g., consider the case where a particular feature is independent from Y and the remaining features but has very different distributions across domains). Finding alternative ways of reducing the dimensionality of the features to improve prediction under covariate shift is our goal. Furthermore, the finite-sample generalization bounds performed thus far focus on the effect of sample size in the source and target domains. In this study, we extend some of these results and analyze them in terms of dimensionality, in order to provide insight into the relationship between covariate shift correction performance and the number of features.

3.1.1 Related Work

The theory of domain adaptation has been studied extensively in several settings; for instance, see (Ben-David et al. [7, 8], Li and Bilmes [73]). There has also been a rich body of work done regarding covariate shift (sample selection bias) both from a theoretical and empirical points of view. The consistency of the density ratio importance weights was established (Shimodaira [106]), and it was demonstrated that in the finite sample scenario, the estimate suffers higher variance. Sample selection bias was approached from a learning theoretic point of view (Zadrozny [140]), and how various supervised learning algorithms behave was studied in this setting. Maximum-entropy density estimation was also investigated under sample selection bias (Dudík et al. [33]). As previously mentioned, several prior studies have attempted to avoid estimating the densities of the target and source domains and calculating the ratio explicitly with various methodologies; see, e.g., (Bickel et al. [10], Dai et al. [26], Huang et al. [62], Kanamori et al. [68], Sugiyama et al. [120]). Regarding the theoretical properties of covariate shift correction, finite sample analyses of the risk in the target domain have been conducted (Gretton et al. [50]), producing a transductive bound of the empirical weighted risk for the kernel mean matching (KMM) method (this result was stated in Corollary 1 below). Furthermore, the effects of the estimation error of $\hat{\beta}$ on the risk in the target domain have been analyzed for KMM (Cortes et al. [21]). The generalization error under covariate shift has been provided without assuming boundedness on the weights β, but instead assuming that the second moment is bounded (Cortes et al. [22]).

3.2 Motivation

To illustrate the problem more clearly, let us consider one of the main transductive results on the empirical weighted risk in the training data, proven in (Gretton et al. [50]):

Corollary 1(Gretton et al. [50]) *With probability* $1-\delta$ *the following bound on the expected risk in the target domain holds:*

$$\begin{aligned}\sup_{l(\cdot,\cdot,\theta)} &|\frac{1}{n_{tr}}\sum_{i=1}^{n_{tr}}\beta_i l(x_i^{tr}, y_i^{tr}, \theta) - \mathbb{E}_{Y|X}[\frac{1}{n_{te}}\sum_{i=1}^{n_{te}} l(x_i^{te}, y_i^{te}, \theta)]| \\ &\leq ||\beta|| CR\frac{2+\sqrt{4\log(1/\delta)}}{n_{tr}} + \\ &C(1+\sqrt{4\log(1/\delta)})R\sqrt{B^2/n_{tr}+1/n_{te}},\end{aligned}$$

where C and R are constants specific to the model class of $l(\cdot,\cdot,\theta)$, and B is the upper bound of β.

Now consider an example in which data were generated according to a simple model given by: $X_{tr} \sim 0.5\mathcal{N}([0,...,0]^d, \Sigma) + 0.5\mathcal{N}([0.5,...,0.5]^d, \Sigma)$, $X_{te} \sim 0.5\mathcal{N}([1.5,...,1.5]^d, \Sigma) + 0.5\mathcal{N}([2,...,2]^d, \Sigma)$, $P(Y=1|X) = \text{sigmoid}(X_1 + 5\tan(X_1))$, where we can range d to explore the behavior of importance re-weighting with varying dimensionality. One can appreciate that in this toy example, the target Y only depends on the first feature X_1. However, when we plot this bound, we see that the variance of the empirical risk estimate grows with increasing dimensionality (as seen in Figure 3.1b). Furthermore, there are additional difficulties that arise with the task of estimating $\hat{\beta}$

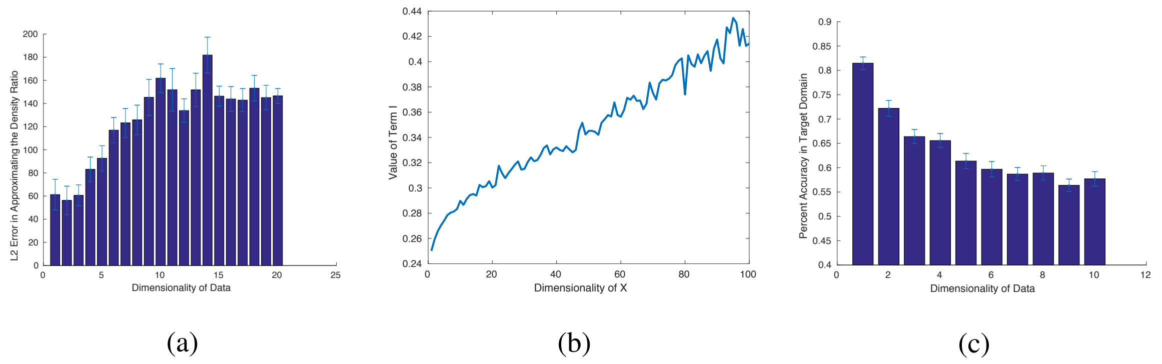


Figure 3.1: (a): Estimation L_2 error of β in Gaussian mixture toy dataset (b): Term 1 of bound. (c): Classification accuracy.

from data, in which the estimation error $||\beta - \hat{\beta}||^2$ is also larger with increasing dimensionality, as shown in Figure 3.1a. Finally, to observe the effect of these phenomena on prediction accuracy, we plotted the classification accuracy using logistic regression for the above-mentioned dataset in Figure 3.1c. It is clear that as the dimensionality increases, the prediction accuracy deteriorates.

Given these observations, a question arises: is there a way to automatically derive and make use of the relevant low-dimensional representation of the features for the purpose of covariate shift correction? If we could find such a low-dimensional representation to capture all of the relevant information in the features X relative to the target Y, then we would be able to perform covariate shift correction on this representation and enjoy a low variance and high estimation accuracy of the importance weights, as well as low variance of the empirical risk. Note that the target-domain risk can be expressed as the re-weighted source domain risk: $R^{te}(l) = \int P^{tr}_{XY} \cdot \frac{P^{te}_{XY}}{P^{tr}_{XY}} l(x, y; \theta) dxdy = \int P^{tr}_{XY} \cdot \frac{P^{te}_{X}}{P^{tr}_{X}} l(x, y; \theta) dxdy = R^{tr}_{\beta}(l)$. Given features $X \in \mathbb{R}^D$, is it possible to find a function of the features X, $h : \mathbb{R}^D \to \mathbb{R}^d$ such that the ratio $\beta_h(x) = \frac{p_{te}(h(x))}{p_{tr}(h(x))}$ can be used to express the target-domain risk in term of the re-weighted source domain risk (i.e. such that $R^{te}(l) = R^{tr}_{\beta_h}(l)$)?

3.3 A Low-Dimensional Reweighting Approach

Since covariate shift correction can suffer in high dimensions, the goal is to find a principled way to represent X in a low-dimensional space, which means that for $X \in \mathbb{R}^D$, we need to find a function $h : \mathbb{R}^D \to \mathbb{R}^d$ s.t. $D > d$, such that $\beta_h(x) = \frac{p_{te}(h(x))}{p_{tr}(h(x))}$ is a density ratio that can be used to express the risk in the target domain in the population case. For this purpose, we develop the following result, inspired by the idea of propensity score in causal effect estimation (Rosenbaum and Rubin [98]). It identifies some key properties that an appropriate function $h(x)$ needs to have.

Theorem 1: *Suppose i) $X \perp\!\!\!\perp Y \,|\, h(X)$ and that ii) the loss $l(x, y, ; \theta)$ can be rewritten as $l_h(h(x), y, ; \theta')$, which involves $h(x)$ instead of x. Then density ratio $\beta_h(x) = \frac{p_{te}(h(x))}{p_{tr}(h(x))}$ and $\beta(x) = \frac{p^{te}(x)}{p^{tr}(x)}$ are loss-equivalent for covariate shift correction, in the sense that* $\mathbb{E}_{(X,Y)\sim P^{te}_{XY}}[l(x, y; \theta)] =$

$\mathbb{E}_{(h(X),Y)\sim P^{tr}_{h(X),Y}}[\beta(h(x))l_h(h(x),y;\theta')]$.

This result implies that $\beta_h := \beta(h(x))$ is just as optimal as β in terms of minimizing the target-domain risk in the infinite sample case, but $h(X)$ could potentially have lower dimensionality and thus avoid negative effects that high dimensionality has on prediction performance in the covariate shift setting. Condition *ii)* will hold if the optimal function $f(x)$ can be rewritten as a function of $h(x)$–intuitively, if $X \perp\!\!\!\perp Y \mid h(X)$, $h(X)$ contains all information in X that is relevant to Y, and hence the optimal prediction function $f(x)$ is also a function of $h(x)$. (The effect of the functional class of $f(x)$ will be discussed later.)

Now that we have established the main property required for $h(X)$, we need to find a function h that satisfies it. Thus, we identify two functions that satisfy these properties for the purposes of classification and regression respectively, in the following proposition:

Proposition 1: *Suppose Y is binary. Then $h(X) = p(Y = 1|X)$ satisfies $X \perp\!\!\!\perp Y \mid h(X)$. Suppose Y is continuous and that $Y = f(X) + \epsilon$, where ϵ is noise and is independent from X. Then $h(X) = \mathbb{E}[Y|X]$.*

In the covariate shift setting, the main premise is that although $P(Y = 1|X)$ and $\mathbb{E}[Y|X]$ do not change, they are too complex to be reliably estimated by a simple method from a finite labeled sample in the source domain (otherwise, there would be no need for covariate shift correction since $P^{tr}_{Y|X} = P^{te}_{Y|X}$). We need a way to estimate a rather simple function $\hat{h}(X)$ that satisfies the conditional independence property required by Theorem 1 using source-domain data.

3.3.1 Procedure for Approximating the Low-Dimensional Representation

Our approach involves finding a low-dimensional representation of X via a random vector $h(X) = \mathbf{h} = [h_1(X)...h_d(X)]$, where $d < D$, such that $X \perp\!\!\!\perp Y|h(X)$. We can use kernel methods and covariance operators in Hilbert Space to express the degree to which $h(X)$ satisfies the conditional independence property, which was widely applied in sufficient dimension reduction (Fukumizu et al. [37], Suzuki and Sugiyama [124]).

We approximate $h(X)$ by assuming that it is given by a linear transformation $\hat{h}(X) = \mathbf{W^T}X$, where $\mathbf{W} \in \mathbb{R}^{D\times d}$ is a projection matrix to a d-dimensional space, then $\hat{h}(X)$ can be found by minimizing a magnitude of the conditional covariance operator $\hat{\mathfrak{U}}_{YY|\hat{h}(X)}$. In our case, we minize the trace of this operator:

$$\arg\min_{W} C(\mathbf{W}) = \mathrm{Tr}[\hat{\mathfrak{U}}_{YY|\hat{h}(X)}] \tag{3.1}$$

$$\text{s.t. } \mathbf{W}^T\mathbf{W} = \mathbf{I} \tag{3.2}$$

After solving for $\mathbf{W}$, we can use an out-of-box density ratio estimation procedure, to estimate $\hat{\beta}_W$

d-dimensional space. In our procedure, we use Kernel Mean Matching:

$$\hat{\beta}_W = \arg\min_{\beta} ||\frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta_i \phi(\mathbf{x}_{Wi}^{tr}) - \frac{1}{n_{te}} \sum_{i=1}^{n_{te}} \phi(\mathbf{x}_{Wi}^{te})|| \tag{3.3}$$

$$\text{s.t. } \beta_i = [0, B], \forall i, \ |\sum_{i=1}^{n_{tr}} \beta_i - n_{tr}| \leq n_{tr}\xi \tag{3.4}$$

where a good value for ξ is $O(B/\sqrt{n_{tr}})$ (Gretton et al. [50]). Thus, the procedure for finding the importance weights using a low-dimensional representation of X consists of two steps:

(1) Use the source-domain labeled training data to solve problem in equation 3.1 to obtain $\mathbf{W}$ such that $Y \perp\!\!\!\perp X|\mathbf{W}^T X$.

(2) Obtain $\hat{\beta}_W$ by solving problem in equation 3.3 on the projected unlabeled data $\mathbf{x}_W$ in the source and target domains using the operator $\mathbf{W}$.
After these two steps are completed, $\hat{\beta}_W$ can be used along with the projections $\mathbf{x}_{Wi}, ..., .\mathbf{x}_{Wn}$ to do covariate shift correction and subsequently apply a supervised learning algorithm on the reweighted projected source-domain data points.

An important design choice of this algorithm is d, the dimensionality of the projection $\mathbf{W}^T X$. To select this value, we perform 5-fold cross-validation on the source-domain data, and select the dimensionality that yields the lowest average cost $C(\mathbf{W})$ (from equation 3.1) across the hold-out samples.

3.4 Theoretical Analysis

In order to gain insight into the implications of dimensionality reduction on learning across domains, we derived a transductive bound on the generalization error given by: $|R^{te}(l_{\hat{\beta}_W}) - R^{te}(l^*)|$, where:

(1) $R^{te}(l^*)$ is the optimal risk in the target domain and it is given by

$$R^{te}(l^*) = \mathbb{E}_{Y|X}[\frac{1}{n_{te}} \sum_{i=1}^{n_{te}} l^*(x_i^{te}, y_i^{te}, \theta)]$$

for test data pairs $(x_1^{te}, y_1^{te}), ..., (x_{n_{te}}^{te}, y_{n_{te}}^{te})$, where $l^* = \arg\min_{l \in \mathcal{H}} R^{te}(l)$;

(2) $R^{te}(l_{\hat{\beta}_W})$ is the true risk arising from the loss applied on reweighted and dimensionality-reduced data using estimated weights $\hat{\beta}_W$, and it is given by

$$R^{te}(l_{\hat{\beta}_W}) = \mathbb{E}_{Y|X}[\frac{1}{n_{te}} \sum_{i=1}^{n_{te}} l_{\hat{\beta}_W}(x_i^{te}, y_i^{te}, \theta)].$$

Here, $l_{\hat{\beta}_W}(x_i^{te}, y_i^{te}, \theta) = l(h_{\hat{\beta}_W}(x_i^{te}), y_i^{te})$, where $h_{\hat{\beta}_W}$ is a hypothesis function on X that has been learned from re-weighted projected training data using $\hat{\beta}_W$.

(3) The expected risk in the target domain with projected features:

$$R_W^{te}(l) = \mathbb{E}_{Y|X}[\frac{1}{n_{te}} \sum_{i=1}^{n_{te}} l(W^{\intercal} x_i^{te}, y_i^{te}, \theta)],$$

with the optimal function $l_W^* = \arg\min_{l \in \mathcal{G}} R_W^{te}(l)$.

In order to derive the aforementioned transductive generalization bound, we rely on the following assumptions which were initially also made by were first made in (Cortes et al. [21], Gretton et al. [50]):

A1 The kernel k is a product kernel, and it satisfies $k(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{d} k(x^{(i)}, y^{(i)})$. It is also bounded: $k(\mathbf{x}, \mathbf{x}) \leq \kappa < \infty$.

A2 (Gretton et al. [50]): The loss function $l(x, \theta)$ satisfies: $l(x, \theta) = \langle \Phi(X), \Theta \rangle$, and such that $\Theta \leq C$. Similarly, $l(x, y, \theta) = \langle \Psi(x, y), \Lambda \rangle$, where $||\Lambda|| \leq C$ and $||\Psi(x, y)|| \leq R$,$||\Phi(x)|| \leq R$ (same constant is used for convenience). Thus, $l(x, \theta)$ and $l(x, y, \theta)$ each belong to a corresponding RKHS. We shall further assume that this RKHS corresponds to a product kernel as defined in **A1**. We also assume the loss l is σ-admissible, as defined by Cortes et al. (Cortes et al. [21]), and differentiable. We provide more detail on this assumption in the Appendix, and it is satisfied by many loss functions including the quadratic cost.

A3 The features after projection $w_1^T X, ..., w_d^T X$, where w_i is the i-th column of $\mathbf{W}$, are independent.

This is assumed for the sake of simplicity of the analysis (if needed, one can further apply a linear transformation to make the outputs independent). Using these quantities defined in the target domain along with the assumptions stated above, we provide a bound on the generalization error in terms of the dimensionality of the features X:

Theorem 2: *Assume that **A1**, **A2** and **A3** hold and let for each projected feature i,* $||\beta_W(w_i^T x)||_2^2 \leq Q, \beta_W(w_i^T x) \leq T \; \forall i \in 1, ..., d$. *Furthermore, let the importance weights* $\hat{\beta}_W$ *be a result of the KMM procedure using a feature map* $\Phi : \mathcal{X} \to \mathcal{H}$ *which corresponds to a kernel function k that satisfies **A1**, and such that* $||\Phi(X_j)|| \leq U \; \forall j \in 1, ..., d$. *Let* $\mathbf{K}$ *be the kernel Gram matrix for kernel k,* $\mathbf{K}_1$*,* $\mathbf{K}_2$*,...,*$\mathbf{K}_d$ *be the kernel Gram matrices of* $k(x^{\mathcal{T}}, y^{\mathcal{T}}), .., k(x^{(d)}, y^{(d)})$ *respectively, and let* $\tilde{\lambda}$ *be the smallest among the minimum eigenvalues* $\lambda_{min}(\boldsymbol{K}_1), ..., \lambda_{min}(\boldsymbol{K}_d)$*. Then with probability*

$1 - \delta$ *the following bound in the target domain holds:*

$$|R^{te}(l_{\hat{\beta}_W}) - R^{te}(l^*)| \leq |R^{te}(l^*_W) - R^{te}(l^*)| + \frac{(2 + \sqrt{2\log(6/\delta)})CU^d}{\frac{n_{tr}}{\sqrt{Q^d}}} + C(1 + \sqrt{2\log(6/\delta)})U^d\sqrt{T^{2d}/n_{tr} + 1/n_{te}} + \frac{\sigma^2\kappa^2}{\lambda}\left(\frac{\xi T^d}{\sqrt{n_{tr}}} + \frac{\kappa^{\frac{1}{2}}}{\tilde{\lambda}^{d/2}}\sqrt{\frac{T^{2d}}{n_{tr}} + \frac{1}{n_{te}}}(1 + \sqrt{2\log(6/\delta)})\right),$$

where λ is a hyper-parameter that controls regularization over the hypothesis set. The first term on the RHS corresponds to the bias that arises from the difference between the optimal hypothesis function given by covariate shift correction in the original D-dimensional space and the one given by covariate shift correction in the reduced d-dimensional space (our method). The second and third terms correspond to the variance of the empirical risk estimate, and the fourth term corresponds to the estimation error in $\hat{\beta}_W$.

We can see from this bound that the dimensionality of the dataset is present in each term. Let us first assume that there is no bias in covariate shift correction after dimensionality reduction. For instance, consider an example where the true generating process for Y is $Y = f(\mathbf{R}^T X) + \epsilon$, where $\mathbb{E}[\epsilon] = 0$. This implies that $X \perp\!\!\!\perp Y|\mathbf{R}^T X$. Let the sample size be infinite. Suppose that we use the correct functional form for prediction, resulting in the optimal function under original covariate shift correction (using all of the features) given by $\hat{Y} = f^*(\mathbf{R}^{*T} X)$, where $\mathbf{R}^*$ is a projection matrix and f^* is a nonlinear function. Our method can find $\mathbf{W}$ such that $X \perp\!\!\!\perp Y|\mathbf{W}^T X$, so it follows that:

$$P(Y|\mathbf{R}^T X) = P(Y|\mathbf{R}^T X, X) = P(Y|\mathbf{W}^T X, X) = P(Y|\mathbf{W}^T X)$$

(because the information of $\mathbf{W}^T X$ and $\mathbf{R}^T X$ is contained in X and because of the conditional independence relations). This implies that we have f' such that $\mathbb{E}(Y|\mathbf{R}^T X) = \mathbb{E}(Y|\mathbf{W}^T X)$, indicating that the optimal decision function under the original covariate shift setting and the one after dimensionality reduction are the same. This means that $f^*(\mathbf{R}^{*T} X) = f'(\mathbf{W}^T X)$. Therefore, one only needs to use a low-dimensional representation $\mathbf{W}^T X$ and learn f' instead of using all of the features of X and learn both $\mathbf{R}$ and f^*. If f' and f^* are in the same function class, there will be no bias, implying that the first term of the RHS will be equal to 0. In this case, the first term in the risk will vanish.

One should note, however, that there are cases in which performing dimensionality reduction with a linear transformation can incur large bias. Consider the case where X has two variables, and the generating process is $X_2 = X_1^3 + \epsilon_1$, $Y = X_2^{1/3} + \epsilon_2$. If under covariate shift, we use a linear model to predict Y, then both X_1 and X_2 are relevant. However, our method would select only feature X_2, which has a nonlinear relationship in Y, resulting in a large bias.

Even though in certain cases our method can have some bias, it can enjoy smaller variance in the risk estimate and smaller estimation error of the weights as a result of low dimensionality. First, the effective sample size $M := n_{tr}^2/||\beta||^2 \geq n_{tr}^2/Q^d$ in the second term, as defined by Gretton et al. (Gretton et al. [50]), can get exponentially smaller as d increases, which explains one of the main reasons why performance may suffer in the target domain when the dimensionality of the data is high. d is also present in the exponent of constants T and U in the second and third term. This means that the variance increases exponentially with respect to d.

Furthermore, the estimation error of the weights β_W also gets exponentially larger as d increases, as can be seen in the third term of the RHS. In the denominator, we have the value $\tilde{\lambda}^{d/2}$, which is the minimum of the smallest eigenvalues corresponding to the kernel Gram matrices for each feature. This number can often be smaller than 1. For example, for the Gaussian RBF kernel this is guaranteed unless the kernel width used is so small that the kernel Gram matrix becomes the identity matrix. This follows from the fact that for the RBF kernel, $\mathrm{Tr}(\mathbf{K}) = n = \sum_{i=1}^{n} \lambda_i(\mathbf{K})$ where $\lambda_i(\mathbf{K})$ is the i-th largest eigenvalue, thus guaranteeing that $\lambda_{min} < 1$ if there is more than one unique eigenvalue.

3.5 Empirical Evaluation

For the purposes of evaluating our method we performed experiments on both pseudo-real and real-world data: T for pseudo-real regression datasets, we created a source domain and a target domain from real datasets with an artificial sample selection bias, and (2) two real datasets consists of a classification problem and a regression problem. We compare our method, i.e., finding the low-dimensional representation $\mathbf{W}^T\mathbf{X}$ and using it to compute the importance weights, with four prediction schemes, including: (i) no reweighting, which treats both the source and the target domains as if they came from the same distribution, (ii) using all the features to compute the importance weights (corresponding to original covariate shift correction), (iii) LHSS (Sugiyama et al. [121]), as a marginal distribution-based dimensionality reduction method, and (iv) using a low-dimensional representation obtained by performing PCA and its density ratio for covariate shift correction. For computing importance weights in schemes (ii) and (iii), we used the three above-mentioned algorithms: KMM ((Huang et al. [62])), KLIEP ((Sugiyama et al. [120])) and RuLSIF (Yamada et al. [138]), which were briefly described in the Related Work section above. We obtained the code for each of these baselines, and ran it on our datasets after tuning the methods to the best of our ability.

3.5.1 Pseudo-Real Data Experiments

We used benchmark regression datasets[1] to generate pseudo-real data, which were also used in (Huang et al. [62]) and (Cortes et al. [21]). We biased the data in the following way, as in (Cortes et al. [21]). We made use of a sample selection variable s, and we calculate a conditional probability of selecting a data point to be observed in the source domain given its features as $p(s = 1|x) = \frac{e^v}{1+e^v}$, where $v = \frac{4w\cdot(x-\bar{x})}{\sigma_{w\cdot(x-\bar{x})}}$ and where w is a random projection vector chosen uniformly from $[-1, 1]^d$. As done in (Cortes et al. [21]), we chose random directions w such that the selection probabilities yield sufficiently different performance between using no weights and using an ideal weight given by $\beta(x) = \frac{1}{P(s=1|x)}$, thus ensuring that the biased dataset is a good candidate dataset for covariate shift correction. We performed this biased sampling scheme on 10 random subsamples of size 2000 from each of the original datasets.

[1]https://www.dcc.fc.up.pt/ ltorgo/Regression/DataSets.html

We performed covariate shift correction on these biased datasets using the above-mentioned approaches and used KRR for regression; we present these results on Table 3.1 in terms of normalized mean-squared error (NMSE): $\frac{1}{n_{te}}\sum_{i=1}^{n_{te}}\frac{(y_i-\hat{y}_i)^2}{\sigma_y^2}$, as performed in (Cortes et al. [21]). Regarding hyperparameters selection, here for KRR we used a kernel width $\sigma_r = \sqrt{\frac{D}{2}}$ where D is the number of features of the dataset, as done in (Cortes et al. [21]) (please see Appendix for more details on hyper-parameter settings). When using PCA for the baselines, we either reduced the dimensionality to 95 percent of the cumulative energy content or to the same number of dimensions used by our method, and report the best results. A table with standard deviations is included in the Appendix due to space constraints.

	Unweight.	KMM-all	KLIEP-all	RuLSIF-all	LHSS	KMM-PCA	KLIEP-PCA	RuLSIF-PCA	KMM-W	D-W	D-original	p-value
Ailerons	2.26	2.01	2.09	2.18	2.06	2.72	2.74	2.81	**0.92**	9	40	**0.0010**
Bank32NH	0.79	0.79	0.82	0.78	0.73	0.91	0.92	0.91	**0.62**	11	32	**0.0527**
Bank8FM	0.81	0.78	0.84	0.79	0.74	0.92	0.99	0.99	**0.32**	1	8	**0.0010**
Abalone	0.99	*0.87*	0.90	0.95	0.85	1.27	1.05	0.96	*0.87*	7	7	0.7842
Elevators	1.14	*1.02*	1.07	1.12	*1.02*	1.50	1.10	1.12	*1.02*	16	18	0.4229
CPU-Act	1.56	1.29	1.46	1.44	1.36	2.12	2.10	2.22	0.53	13	21	**0.0010**
California	0.99	0.93	1.00	0.99	0.94	1.21	1.55	1.23	0.78	5	8	**0.0049**
Puma8NH	0.45	0.38	0.43	0.44	0.38	0.74	0.74	0.74	0.33	3	8	**0.0049**

Table 3.1: NMSE results on the baselines and the proposed method, on the pseudo-synthetic datasets. The methods with the suffix "-all" use all the features to calculate importance weights. The "-PCA" suffix means that PCA was used to represent the data in lower dimensions before estimating the weights $\hat{\beta}$; the suffix "-W" means that the proposed low-dimensional representation given by $\mathbf{W}^T\mathbf{X}$.

There are several take-aways from these experimental results. One can appreciate that the proposed method outperforms the baselines in the majority of the datasets, and it does so by a large margin. In the cases in which it does not outperform the baselines (such as "Abalone" and "Elevators"), it selects almost all the features and reduces to regular covariate shift correction. This may be because on these datasets, $h(X)$ s.t. $X \perp\!\!\!\perp Y|h(X)$ cannot be summarized in a lower-dimensional linear projection of X.

Furthermore, we see that on these datasets PCA as a dimensionality reduction technique is not effective for the purposes of covariate shift correction. This unreliability as a method for covariate shift correction likely comes from the fact that PCA does not take into account the relationship of the dimensions of X with the target Y, and thus is likely to disregard relevant features that explain less of the variance in the data.

	Unweight.	KMM	KLIEP	RuLSIF	LHSS	KMM-W	p-value
$T_1 \rightarrow T_2$	95.4(0.9)	95.2(0.6)	95.7(0.6)	95.8(0.6)	95.9(0.6)	**97.3(0.4)**	**0.014**
$T_2 \rightarrow T_1$	90(1.2)	92.4(1.2)	91.4(1.3)	91.2(1.2)	91.7(1.3)	**94.8(0.7)**	**0.006**
$M \rightarrow F$	94.4(1.0)	*95.4(0.8)*	93.5(1.1)	92.8(1.5)	95.1(0.7)	*95.4(0.9)*	0.548
$F \rightarrow M$	91(1.5)	92.1(1.1)	90.4(2.1)	90.9(2)	92.7(0.8)	**93.7**(1.0)	**0.082**

Table 3.2: SVM accuracy results on the baselines and the proposed method on the cancer gene expression dataset. We performed PCA on the data before testing all of the baselines and the proposed method, due to the high dimensionality of the original dataset. Standard error is in parentheses.

Direction	Unweighted	KMM	KLIEP	RuLSIF	LHSS	KMM-W	p-value
$A \rightarrow C$	*75.93(1.1)*	75.67(1.2)	*76.27(1.1)*	75.8(1.1)	75(1.4)	74.27(1.9)	0.746
$A \rightarrow D$	*76.6(1.5)*	75.53(1.5)	*77.33(1.8)*	75.33(1.1)	70.93(1.8)	70.27(3.8)	0.96
$A \rightarrow W$	67(1.9)	66.67(1.8)	66.47(1.9)	66.4(2)	62.67(2.2)	**71.67(1.9)**	**0.037**
$C \rightarrow A$	86.93(1.6)	86.13(1.2)	86.87(2.3)	*88.53(0.9)*	86.27T	*88.4(0.5)*	0.535
$C \rightarrow D$	*78.2(1.1)*	77.53(1.8)	77.53(1.2)	*78.2(1.3)*	73.8(3.2)	77.13(2.1)	0.582
$C \rightarrow W$	67.07(1.8)	68(1.7)	67.73(1.8)	67.8(2.1)	66.27(1.8)	**73.27(1.8)**	**0.009**
$D \rightarrow A$	75.8(1.3)	78.93(1.4)	77.47(1.2)	78.87(1.3)	71.8(1.1)	**83.87(0.9)**	**0.005**
$D \rightarrow C$	63(1.2)	67.67(1.2)	67.53(1.6)	67.6(1.1)	60.53(1.5)	**71.33(0.9)**	**0.001**
$D \rightarrow W$	93.67(0.6)	*96.33(0.8)*	*96.4(0.9)*	*96.47(0.9)*	93.27(0.8)	*95.8(0.8)*	0.891
$W \rightarrow A$	71.33(0.8)	70.33(0.8)	71.13(0.7)	71.13(0.9)	71.4(0.6)	**72.27(2.3)**	**0.191**
$W \rightarrow C$	63.87(1.5)	65.53(1.5)	65.6(2.1)	64.8(2)	63.13(1.5)	**70.93(1.3)**	**0.041**
$W \rightarrow D$	97.53(0.4)	*97.6(0.4)*	97.4(0.3)	97.27(0.4)	96.93(0.5)	*97.8(0.3)*	0.445

Table 3.3: SVM accuracy results on the baselines and the proposed method on the Office-Caltech dataset. Here we did PCA on all baselines before performing covariate shift, due to the high dimensionality of the dataset.

3.5.2 Experiments on Real Data

In order to further examine the efficacy of our approach, we performed experiments on two real datasets. In addition to the object recognition task, we also performed experiments on a publicly available cancer gene expression dataset, provided by The Cancer Genome Atlas (TCGA) Network [2]. The data were collected from large set of patients with five different tumor types (colon cancer, breast cancer, stomach cancer, glioblastoma, and kidney cancer), and various clinical parameters were collected for each patient. In this dataset, each patient is a data point, and the features are 20531 annotated genes. This dataset is very high-dimensional, yet the task is simple and boils down to identifying the different organs where the tumor took place, which can be identified by a limited set of genes (features). We performed prediction of the tumor type based on the gene

[2]http://cancergenome.nih.gov/ and http://firebrowse.org

expression profile across domains, which are obtained as follows.

The different domain subdivisions we consider are time (patients diagnosed before and after the year of 2008), and gender (excluding breast cancer). For both domain subdivisions, there may be an overall change in the distribution of gene expression, i.e., $P^{source}(X) \neq P^{target}(X)$. For example, methodologies of collecting biopsies and measuring gene expression evolve over time. Similarly, the overall gene expression profile across genders may be different due to different epigenetic factors. Furthermore, it is safe to assume that $P^{source}(Y|X) = P^{target}(Y|X)$, because various time points at which the patients were diagnosed or their gender, are not supposed to affect the biology of the tumor tissue. Therefore, this dataset and task correspond to the covariate shift setting.

Before running each method, we performed PCA as a pre-processing step. We tested our method against LHSS and SVM without re-weighting, on the PCA-derived features. We also ran the baselines KMM, KLIEP, and RuLSIF by using: **(1)** all $n-1$ PCA-derived features for estimating the weights (that is the highest possible number of features since $D > n$ in the original dataset), **(2)** the dimensions corresponding to the 95 percent of the cumulative energy content, **(3)** the same number of dimensions that our method selected. For the baselines, we report best accuracy of the three dimensionality reduction schemes. For each transfer direction we performed 20 replicates, in each of which we subsample 50 points in each domain, and the average accuracies for each direction are reported in Table 3.2. The dimensionality of our method selected was $d = 3$ in all transfer directions. The results show that our method outperforms the baselines in three of the four transfer directions with statistical significance, and in one of them it ties with the KMM baseline.

In addition to the cancer dataset, we also evaluated our method on the Office-Caltech datase (Gong et al. [41]), and it is concerned with the task of object recognition. This dataset was constructed from two prior datasets: Office (Saenko et al. [100]) and Caltech (Griffin et al. [52]), and has four domains with images: Amazon images, webcam (low-resolution), DSLR (high-resolution), and Caltech-256. We used the DeCAF$_6$ features extracted by a convolutional neural network described in (Donahue et al. [30]). We conducted experiments with each source-target ordered pair of the domains in this dataset, using SVM to classify the data after covariate shift correction. Since each data point (image) in this dataset has 4096 CNN features, PCA was used for preprocessing. For each transfer direction, we performed 10 replicate experiments by subsampling 150 points in the corresponding source and target domains. We used the same experimental setting as in the cancer dataset. However, we note that different from the cancer data, in this dataset the assumption of covariate shift may not hold true: $p(Y|X)$ may change across the domains, in which the images were collected under different conditions.

In order to fully assess the reliability of our method, for each baseline we used SVM classification in which we either set the kernel width and the slackness parameter C to fixed values and assuming a misspecified model, or selected them via 5-fold CV. We reported the best accuracy of the two options. For our method we used a simple model, where we set a kernel width proportional to the median pairwise distance of all the unlabeled data points (with constants of proportionality 4 and 1 for the cancer and the Office-Caltech datasets respectively), and fixing $C = 10$.

The average accuracies for each experiment are given in Table 3.3. It shows that our method outperforms the baselines in 6 of the settings. In the settings where our method does not outperform the baselines, it is either tied with at least one more of the baselines, or there is no single baseline that significantly outperforms the others. These results suggest that even when the covariate shift assumption is likely to be violated, our method still reliably improves the accuracy. For all source-target domain pairs, the most often selected dimensionality by our method was 10.

3.6 Discussion

This study aimed to tackle dimensionality reduction for covariate shift correction, by taking into account the target variable Y and the features that are relevant for predicting it. We provided some theoretical insights in terms of the role that high dimensionality plays in poor generalization in the target domain. We focused on a linear projection as the low-dimensional representation of X. However, this might not satisfy the conditional independence properties for some datasets, and the importance weights derived from this representation may not be as useful. This may have contributed to the cases in our experiments when using all features performed better than using the $\mathbf{W}$ projection to reduce the dimensionality. A potentially fruitful future direction of research would be to develop methodology which can identify nonlinear functions of X that satisfy the necessary conditional independence property and reduce the dimensionality efficiently, as this would broaden the applicability of this type of approach to more domains and datasets. Another line of our future work is to extend the idea to hand other settings for domain adaptation, such as target shift (Zhang et al. [148]).

3.7 Proofs of Theoretical Results

In this appendix we provide proofs for some theoretical results in this chapter.

3.7.1 Proofs of Theorem 1 and Proposition 1

1. Proof for Theorem 1

Proof: *If the conditions in the theorem hold true, we can rewrite the expected loss as follows:*

$$\begin{aligned}
&l_{P_{XY}}[f] \triangleq \mathbb{E}_{P_{XY}}[l(f(X),Y)] \\
&= \int l(f(x),y)p(y|x)p(x)dxdy \\
&= \int l_h(\tilde{f}(h(x)),y)p(y|x,h(x))p(x|h(x))p(h(x))dhdxdy \\
&= \int l_h(\tilde{f}(h(x)),y)p(y|h(x))p(h(x))dhdy.
\end{aligned}$$

Since $p^{te}(y|x) = p^{tr}(y|x)$, implied by the covariate shift setting, and $p(y|h(x)) = p(y|h(x),x) = p(y|x)$, we have $p^{te}(y|h(x)) = p^{tr}(y|h(x))$. Let $\beta(h) \triangleq \frac{p^{te}(h(x))}{p^{tr}(h(x))}$, and the expected loss in the *target domain, $l_{P^{te}_{XY}}[f]$, further becomes*

$$\begin{aligned} l_{P^{te}_{XY}}[f] &= \int l_h(\tilde{f}(h(x)), y)p^{te}(y|h(x))p^{te}(h(x))\beta(h)dhdy \\ &= \int l_h(\tilde{f}(h(x)), y)p^{tr}(y|h(x))p^{tr}(h(x))\beta(h)dhdy \\ &= \mathbb{E}_{P^{tr}_{XY}}[l_h(\tilde{f}(h(x)), y)\beta(h)]. \end{aligned}$$

□

2. We also present a proof for Proposition 1

Proof: *For the case when Y is binary, we have: $p(Y = 1 \,|\, X) = p(Y = 1 \,|\, X, h(X)) = h(X) = p(Y = 1 \,|\, h(X))$, and similarly $p(Y = 0 \,|\, X) = p(Y = 0 \,|\, X, h(X)) = 1 - h(X) = p(Y = 0 \,|\, h(X))$. For the case when Y is continuous, it follows trivially because $f(X) = \mathbb{E}[Y|X]$, and $Y \perp\!\!\!\perp f(X)|f(X)$, which implies that $Y \perp\!\!\!\perp X|f(X)$ due to the fact that X and ϵ are independent.* □

3.7.2 Proof of Theorem 2

We shall first introduce some relevant quantities:

- Expected risk in target domain

$$R^{te}(l) = \mathbb{E}_{y|x}[\frac{1}{n_{te}}\sum_{i=1}^{n_{te}} l(x_i^{te}, y_i^{te}, \theta)],$$

  the optimal function $l^* = \arg\min_{l\in\mathcal{G}} R^{te}(l)$

- Expected risk in the target domain with projected features

$$R^{te}_W(l) = \mathbb{E}_{y|x}[\frac{1}{n_{te}}\sum_{i=1}^{n_{te}} l(W^\intercal x_i^{te}, y_i^{te}, \theta)],$$

  the optimal function $l^*_W = \arg\min_{l\in\mathcal{G}} R^{te}_W(l)$

- Expected risk in the weighted source domain with original features

$$R^{tr}_\beta(l) = \mathbb{E}_{y|x}[\frac{1}{n_{tr}}\sum_{i=1}^{n_{tr}} \beta_i l(x_i^{tr}, y_i^{tr}, \theta)],$$

  the estimated function $l^*_\beta = \arg\min_{l\in\mathcal{G}} R^{tr}_\beta(l)$

- Expected risk in the weighted source domain with projected features

$$R^{tr}_{\beta_W}(l) = \mathbb{E}_{y|x}[\frac{1}{n_{tr}}\sum_{i=1}^{n_{tr}} \beta_{W,i} l(W^\intercal x_i^{tr}, y_i^{tr}, \theta)],$$

  the estimated function $l^*_{\beta_W} = \arg\min_{l\in\mathcal{G}} R^{tr}_{\beta_W}(l)$

- Empirical risk in the weighted source domain with true weights

$$\hat{R}^{tr}_{\beta}(l) = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta_i l(x_i^{tr}, y_i^{tr}, \theta),$$

  the estimated function $l_\beta = \arg\min_{l \in \mathcal{G}} \hat{R}^{tr}_{\beta}(l)$

- Empirical risk in the weighted source domain with projected features and true weights

$$\hat{R}^{tr}_{\beta_W}(l) = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta_{W,i} l(W^\intercal x_i^{tr}, y_i^{tr}, \theta),$$

  the estimated function $l_{\beta_W} = \arg\min_{l \in \mathcal{G}} \hat{R}^{tr}_{\beta_W}(l)$

- Empirical risk in the weighted source domain with original features and estimated weights

$$\hat{R}^{tr}_{\hat{\beta}}(l) = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \hat{\beta}_i l(x_i^{tr}, y_i^{tr}, \theta),$$

  the estimated function $l_{\hat{\beta}} = \arg\min_{l \in \mathcal{G}} \hat{R}^{tr}_{\hat{\beta}}(l)$

- Empirical risk in the weighted source domain with projected features and estimated weights

$$\hat{R}^{tr}_{\hat{\beta}_W}(l) = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \hat{\beta}_{W,i} l(W^\intercal x_i^{tr}, y_i^{tr}, \theta)$$

  , the estimated function $l_{\hat{\beta}_W} = \arg\min_{l \in \mathcal{G}} \hat{R}^{tr}_{\hat{\beta}_W}(l)$

Before we proceed to the proof of Theorem 2, we present some lemmata that are required to analyze the generalization in the target domain. We we also need a general variant of **A3**:

A3': The features of X, given by $X_1, ..., X_D$ are independent.

Both of them are proven assuming we have features X, in with D dimensions.

We first introduce a lemma which is a modification of the result by Gretton et al., in which we analyze the impact of dimensionality on the variance of the empirical weighted risk in the source domain, under the assumptions made in this study. Please note that we prove the lemmata assuming an input feature space of D, regardless if it is an original feature space of observations or a result of a transformation.

Lemma 1 (Adapted from Corollary 1.9 in Gretton et al): *Assume **A1, A2** and **A3'** hold. Then the following bound on the reweighted risk in the source domain holds with probability at least* $1 - \delta$:

$$\sup_{l \in \mathcal{G}} |\hat{R}^{tr}_{\beta}(l) - R^{te}(l)| = |\frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \beta_i l(x_i^{tr}, y_i^{tr}, \theta) - \mathbb{E}_{Y|X}[\frac{1}{n_{te}} \sum_{i=1}^{n_{te}} l(x_i^{te}, y_i^{te}, \theta)]| \leq \quad (3.5)$$

$$\frac{(2 + \sqrt{4\log(1/\delta)})CU^D}{\frac{n_{tr}}{\sqrt{Q^D}}} + C(1 + \sqrt{4\log(1/\delta)})U^D \sqrt{T^{2D}/n_{tr} + 1/n_{te}} \quad (3.6)$$

Proof: The proof will examine the constants R, B and provide an upper bound on $||\beta||^2$, which can then be used to modify the bound in the corollary by Gretton et al. Assume for now that all the dimensions of X are independent: $p(x) = \prod_{i=1}^{D} p(x^{(i)})$. This means that the weights on all the features $\beta(x) = \beta(x^{(1)})\beta(x^{(2)})...\beta(x^{(D)})$. Therefore, the squared norm of the weights is: $||\beta(x)||_2^2 = \int \beta(x)^2 dx = \int \prod_{i=1}^{D} \beta(x^{(i)})^2 dx = \prod_{i=1}^{D} \int \beta(x^{(i)})^2 dx = \prod_{i=1}^{D} ||\beta(x^{(i)})||_2^2 \leq Q^D$ where the third equality is due to the independence of the features.

Furthermore, from the assumptions on the feature transform Φ and its corresponding reproducing kernel k, we see that $||\Phi(X_i)||^2 \leq R^2 \implies k(x_i, x_i) \leq R^2 \ \forall i \in \{1, ..., D\} \implies k(\mathbf{x}, \mathbf{x}) \leq U^{2D} \implies ||\Phi(\mathbf{x})|| \leq U^D$, meaning we can substitute R with U^d in the bound. By the same derivation, $||\Psi(\mathbf{x}, \mathbf{y})|| \leq U^D$ as well.

Finally, it can be easily seen that from the assumption of independence of the features, $\beta(x_i) \leq T$ $\forall i \in \{1, ..., D\} \implies \beta(\mathbf{X}) \leq T^D$.

Plugging the bound on $||\beta(x)||_2^2 \leq Q^D$, U^D for R, and T^D for B in the bound by Gretton et al. yields the result. □

In addition to the variance of the empirical weighted risk in the source domain, another component that is important for analysis of the generalization in the target domain is the estimation error of the weights obtained by KMM, given by $\hat{\beta}$. For this purpose, we analyze the role of the dimensionality on the estimation error as studied by Theorem 4 in Cortes et al. (Cortes et al. [21]). We first restate the formal definition of admissibility given by Cortes et al. (Cortes et al. [21])

Definition 1:(Cortes et al. [21])*Let H be a hypothesis set. The loss l is σ-admissible if there exists $\sigma \in \mathbb{R}_+$ such that for any two hypotheses h, $h' \in H$ and for all $(x, y) \in X \times Y$,*

$$|l(x, y, h) - l(x, y, h')| \leq \sigma |h(x) - h'(x)|$$

We now present Theorem 4 proved by Cortes et al (Cortes et al. [21]), before we analyze it in terms of the dimensionality of the dataset:

Theorem 4(Cortes et al. [21]) *Let k be a strictly positive definite symmetric universal kernel suck that $k(x, x) \leq \kappa < \infty$. Let h_β be the hypothesis returned by a kernel-based regularization algorithm using a weighted sample S_β using weights β, and let $\hat{h}_\beta$ be a hypothesis obtained the same way using a weighted sample $S_{\hat{\beta}}$ with weights $\hat{\beta}$. Let l_β be the loss on a hypothesis function h_β and let $l_{\hat{\beta}}$ be the loss on hypothesis function $h_{\hat{\beta}}$, where the loss function $l(\cdot, \cdot, h)$ is σ-admissible. Then, for any $\delta > 0$ with probability at least $1 - \delta$, the difference in generalization error of the hypotheses is bounded as:*

$$|R(l_\beta) - R(l_{\hat{\beta}})| = \frac{\sigma^2 \kappa^2}{\lambda} \left(\frac{\xi B}{\sqrt{n_{tr}}} + \frac{\kappa^{\frac{1}{2}}}{\lambda_{min}^{\frac{1}{2}}(\mathbf{K})} \sqrt{\frac{B^2}{n_{tr}} + \frac{1}{n_{te}}} (1 + \sqrt{2 \log(\frac{2}{\delta})}) \right) \tag{3.7}$$

Now we can make use of this estimation error result and analyze it in terms of dimensinality in the following lemma :

Lemma 2: *Let the σ-admissibility assumption hold on the loss function $l(\cdot,\cdot,\theta)$, and let assumptions **A1**, **A2** and **A3** hold. Let k be a kernel function that satisfies **A2**, and such that $||\Phi(X_j)|| \leq U \;\forall j \in 1,...,d$. Let $\mathbf{K}$ be the kernel Gram matrix for kernel k, $\mathbf{K}_1$, $\mathbf{K}_2$,...,$\mathbf{K}_d$ be the kernel Gram matrices of $k(x^{(1)},y^{(1)}),..,k(x^{(d)},y^{(d)})$ respectively, and let $\tilde{\lambda}$ be the smallest among the minimum eigenvalues $\lambda_{min}(\boldsymbol{K}_1),...,\lambda_{min}(\boldsymbol{K}_d)$. Then the following bound on the estimation error of the risk in the test domain holds:*

$$|R^{te}(l_{\hat{\beta}}) - R^{te}(l_{\beta})| \leq \frac{\sigma^2\kappa^2}{\lambda}(\frac{\xi T^d}{\sqrt{n_{tr}}} + \frac{\kappa^{\frac{1}{2}}}{\tilde{\lambda}^{\frac{d}{2}}}\sqrt{\frac{T^{2d}}{n_{tr}} + \frac{1}{n_{te}}}(1+\sqrt{2\log(\frac{2}{\delta})})) \tag{3.8}$$

Proof: Since k satisfies **A1**, from the positive-semidefinite property of the kernel gram matrix, the following holds:

$$\lambda_{min}(\mathbf{K}) \geq \lambda_{min}(\mathbf{K}_1...\mathbf{K}_d) \geq \prod_{i=1}^{d}\lambda_{min}(\mathbf{K}_i) \geq \tilde{\lambda}^d$$

Thus, we can insert $\tilde{\lambda}^d$ for $\lambda_{min}(\mathbf{K})$ in the bound by Cortes et al (Cortes et al. [21]) to obtain the result. Similarly, we can insert T^d for B with the same argument used in the proof of Lemma 1. $\square$

Now that we have the bounds on the variance of the empirical risk in the source domain and the estimation error of β, we can combine them to prove Theorem 2 which we presented in the main text, and we restate here:

Theorem 2: *Assume that **A1**, **A2** and **A3** hold and let for each feature i, $||\beta_W(x^{(i)})||_2^2 \leq Q, \beta_W(x^{(i)}) \leq T \;\forall i \in 1,...,d$. Furthermore, let the importance weights β_W be a result of the KMM procedure using a feature map $\Phi : \mathcal{X} \to \mathcal{H}$ which corresponds to a kernel function k that satisfies **A1**, and such that $||\Phi(X_j)|| \leq U \;\forall j \in 1,...,d$. Let $\mathbf{K}$ be the kernel Gram matrix for kernel k, $\mathbf{K}_1$, $\mathbf{K}_2$,...,$\mathbf{K}_d$ be the kernel Gram matrices of $k(x^{(1)},y^{(1)}),..,k(x^{(d)},y^{(d)})$ respectively, and let $\tilde{\lambda}$ be the smallest among the minimum eigenvalues $\lambda_{min}(\boldsymbol{K}_1),...,\lambda_{min}(\boldsymbol{K}_d)$. Then the generalization bound in the target domain satisfies:*

$$\begin{aligned}|R^{te}(l_{\hat{\beta}_W}) - R^{te}(l^*)| \leq &|R^{te}(l_W^*) - R^{te}(l^*)| + \frac{(2+\sqrt{2\log(6/\delta)})CU^d}{\frac{n_{tr}}{\sqrt{Q^d}}}\\ &+ C(1+\sqrt{2\log(6/\delta)})U^d\sqrt{T^{2d}/n_{tr} + 1/n_{te}}\\ &+ \frac{\sigma^2\kappa^2}{\lambda}(\frac{\xi T^d}{\sqrt{n_{tr}}} + \frac{\kappa^{\frac{1}{2}}}{\tilde{\lambda}^{d/2}}\sqrt{\frac{T^{2d}}{n_{tr}} + \frac{1}{n_{te}}}(1+\sqrt{2\log(6/\delta)}))\end{aligned}$$

Proof: By expanding the risk in the target domain and using triangle inequality, we have:

$$
\begin{aligned}
&|R^{te}(l_{\hat{\beta}_W}) - R^{te}(l^*)| \\
=&|R^{te}(l_{\hat{\beta}_W}) - R^{te}(l_{\beta_W}) + R^{te}(l_{\beta_W}) - R^{tr}_{\beta_W}(l_{\beta_W}) \qquad (3.9)\\
&+ R^{tr}_{\beta_W}(l_{\beta_W}) - R^{tr}_{\beta_W}(l^*_{\beta_W}) + R^{tr}_{\beta_W}(l^*_{\beta_W}) - R^{te}(l^*_W) + R^{te}(l^*_W) - R^{te}(l^*)| \\
\leq&|R^{te}(l_{\hat{\beta}_W}) - R^{te}(l_{\beta_W})| + |R^{te}(l_{\beta_W}) - R^{tr}_{\beta_W}(l_{\beta_W})| + |R^{tr}_{\beta_W}(l_{\beta_W}) \qquad (3.10)\\
&- R^{tr}_{\beta_W}(l^*_{\beta_W})| + |R^{tr}_{\beta_W}(l^*_{\beta_W}) - R^{te}(l^*_W)| + |R^{te}(l^*_W) - R^{te}(l^*)| \\
\leq&|R^{te}(l_{\hat{\beta}_W}) - R^{te}(l_{\beta_W})| + 2\sup_{l\in\mathcal{G}} |\hat{R}^{tr}_{\beta_W}(l) - R^{tr}_{\beta_W}(l)| \qquad (3.11)\\
&+ 2\sup_{l\in\mathcal{G}} |R^{tr}_{\beta_W}(l) - R^{te}(l)| + |R^{te}(l^*_W) - R^{te}(l^*)| \\
\leq&|R^{te}(l^*_W) - R^{te}(l^*)| + |R^{te}(l_{\hat{\beta}_W}) - R^{te}(l_{\beta_W})| + 2\sup_{l\in\mathcal{G}} |\hat{R}^{tr}_{\beta_W}(l) - R^{te}(l)| \qquad (3.12)
\end{aligned}
$$

Substituting the bound of Lemma 1 for the second term in the RHS, and the bound of Lemma 2 in the first term of the RHS yields the final result. □

3.7.3 Covariance Operators

In the main text we discussed estimating the trace of the conditional covariance operator. It can be empirically estimated as (using notation as defined in the main text):

$$
\begin{aligned}
&\mathrm{Tr}[\hat{\mathcal{U}}_{YY|\hat{h}(X)}] \\
&= \mathrm{Tr}[\hat{\mathcal{U}}_{YY}] - \mathrm{Tr}[\hat{\mathcal{U}}_{Y,\hat{h}(X)} \hat{\mathcal{U}}^{-1}_{\hat{h}(X),\hat{h}(X)} \hat{\mathcal{U}}_{\hat{h}(X),Y}] \\
&= \frac{1}{n_{tr}} \mathrm{Tr}[\rho(\mathbf{Y})\rho(\mathbf{Y})^T] - \frac{1}{n_{tr}} \mathrm{Tr}[\rho(\mathbf{Y})\phi(\hat{h}(\mathbf{X}))^T (\phi(\hat{h}(\mathbf{X}))\phi(\hat{h}(\mathbf{X}))^T + n_{tr}\epsilon\mathbf{I})^{-1}\phi(\hat{h}(\mathbf{X}))\rho(\mathbf{Y})^T],
\end{aligned}
$$

where ϵ is a small number to prevent ill conditioning of the matrix, and fixed to 0.01.

Chapter 4

Domain Adaptation with Invariant Representation Learning: What Transformations to Learn?

Unsupervised domain adaptation, as a prevalent transfer learning setting, spans many real-world applications. With the increasing representational power and applicability of neural networks, state-of-the-art domain adaptation methods make use of deep architectures to map the input features X to a latent representation Z that has the same marginal distribution across domains. This has been shown to be insufficient for generating optimal representation for classification, and to find conditionally invariant representations, usually strong assumptions are needed. We provide reasoning why when the supports of the source and target data from overlap, any map of X that is fixed across domains may not be suitable for domain adaptation via invariant features. Furthermore, we develop an efficient technique in which the optimal map from X to Z also takes domain-specific information as input, in addition to the features X. By using the property of minimal changes of causal mechanisms across domains, our model also takes into account the domain-specific information to ensure that the latent representation Z does not discard valuable information about Y. We demonstrate the efficacy of our method via synthetic and real-world data experiments.

4.1 Introduction

Unsupervised domain adaptation (UDA) is a common setting for supervised learning, in which the labeled training and unlabeled test data come from different distributions. More formally, given features $X \in \mathbb{R}^d$ and labels $Y \in \mathbb{R}$, we observe labeled source and unlabeled target domain instances, represented by $(\mathbf{x}^{\mathcal{S}}, \mathbf{y}^{\mathcal{S}}) = (\mathbf{x}^{\mathcal{S}}_k, y^{\mathcal{S}}_k)_{k=1}^{m_S}$ and $\mathbf{x}^{\mathcal{T}} = (\mathbf{x}^{\mathcal{T}})_{k=1}^{m_T}$ respectively, where $P^{\mathcal{S}}(X,Y) \neq P^{\mathcal{T}}(X,Y)$ (and m_S and m_T are the number of observations of the source and target domain instances, respectively). The main challenge of domain adaptation is to use the given source domain observations for learning a predictor that will perform well in the target domain. To do so, the procedure needs to make use of some similarities between the two domains. One way to formalize this is by making assumptions about how the joint distribution joint $P(X,Y)$

changes across domains. For example, in the well-studied setting of covariate shift (Ben-David et al. [5], Cortes et al. [20], Huang et al. [61], Shimodaira [105], Sugiyama et al. [119], Zadrozny [139]), the marginal distribution $P(X)$ changes while the conditional distribution $P(Y|X)$ (i.e., the optimal predictor) is shared across domains.

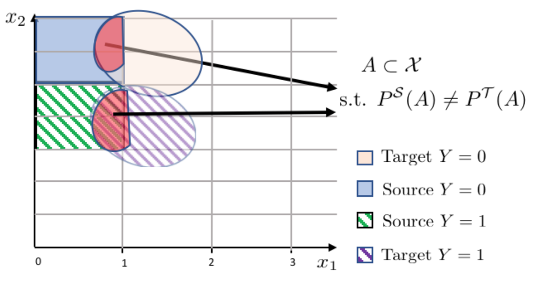
However, in many real-world applications, $P(Y|X)$ can also change, and this requires making use of further assumptions. One such assumption is that the factorization $P(X, Y) = P(Y)P(X|Y)$ allows for addressing the changes in $P(Y)$ and $P(X|Y)$ independently in a situation where their respective changes are simple and easier to capture (Schölkopf et al. [102], Zhang et al. [148]). In this particular setting, the problem is generally broken down into: **(1)** *Target shift:* $P(Y)$ changes across domains while $P(X|Y)$ stay the same (Lipton et al. [74], Plessis and Sugiyama [96], Storkey [116], Zhang et al. [148]) **(2)** *Conditional shift:* $P(X|Y)$ changes across domains but $P(Y)$ stays the same (Long et al. [75], M. Gong and Schölkopf [84], Zhang et al. [148]). **(3)** *Conditional-target shift:* Both $P(X|Y)$ and $P(Y)$ change independently across domains - the most general setting under this generating process assumption (Chen et al. [18], Zhang et al. [148]). Then, assumptions about the changes of the factors of the joint distrubution can be made so that the problem is solvable, such as location-scale transformation (M. Gong and Schölkopf [84], Zhang et al. [143]), or that the changing parameters lie on a low-dimensional manifold (Stojanov et al. [114]), and algorithms can be designed to enforce these constraints and make use of them for prediction in the target domain.

Another fruitful view of the problem is through the lens of representation learning, due to wide-spread applicability of neural architectures for many real-world problems. In particular, state of the art deep learning techniques harness the high representational capacity of neural networks to transform the input data into a latent feature representation which is predictive of the target variable Y in the source domain, and has the same distribution across domains. Formally, this means learning a function (encoder) $\phi : \mathcal{X} \rightarrow \mathcal{Z}$ from the input space to a latent space $\mathcal{Z}$, such that $P^S(Z) = P^T(Z)$. At the same time, a function $h : \mathcal{Z} \rightarrow \mathcal{Y}$ can be learnt to minimize the risk in the labeled source domainn (Ben-David et al. [4]). The hope is then, that the overall function $g := h \circ \phi$ will have low prediction risk in the target domain.However, the above-described theoretical and methodological framework does not guarantee that the learnt representation Z will have any relevant information for predicting Y in the target domain. Namely, one can easily have a situation in which the learnt representation Z is *marginally invariant* ($P^S(Z) = P^T(Z)$), but not *conditionally invariant* ($P^S(Z|Y) \neq P^T(Z|Y)$), as discussed in (Zhao et al. [156]). This means that the learnt function $g := h \circ \phi$ can have very good prediction performance in the source domain, but generalize very poorly to the target domain. In many methods, the same

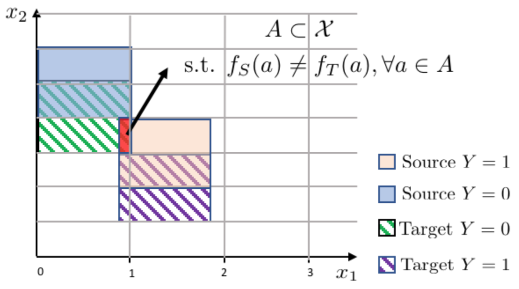
$$\begin{tikzcd}
 & & \theta_X \arrow[dr] & \\
Y \arrow[r] & Z \arrow[rr] & & X
\end{tikzcd}$$

Figure 4.1: The underlying data-generating process under conditional shift, of the observed variables Y and X, and the latent variable Z. θ_X represents the changing parameters of $P(X|Y)$ across domains.

encoding function $\phi(X)$ across domains is used to learn invariant latent representations. This enjoys computational benefits and makes the learning procedure relatively simple, and the vast majority of approaches ((Ganin and Lempitsky [38], Gu et al. [54], Long et al. [78, 82, 83])



(a)



(b)

Figure 4.2: Two different scenarios in which a single encoder $\phi(X)$ is not sufficient to learn Z.

among many) employ this technique. However, in certain situations, the same encoding function across domains cannot learn a marginally invariant representation that is optimal for classification in the target domain. There are certain studies ((Bousmalis et al. [12], Tzeng et al. [126], Wang et al. [129])) which implement domain-specific encoders ϕ_S and ϕ_T. Unfortunately, such methods suffer from the following drawbacks: **(1)** the exact motivation behind having separate encoders for each domain is not clear; **(2)** including two separate encoders may be inefficient because it greatly increases the number of parameters that we need to learn; **(3)** in the field of UDA based on invariant representation learning, there is still no principled way to guarantee that the learned marginally invariant representation Z has sufficient structural (semantic) information, and has the potential to be conditionally invariant.

In this paper we assume the setting of *conditional shift*, and we make use of the data-generating process to: **(i)** justify the use of two separate encoding functions in order to infer the latent representation, **(ii)** implement the two encoding functions more efficiently, and **(iii)** constrain the latent representation Z to have meaningful structure which is useful for prediction in the target domain. In Section 2 of this chapter, we first motivate the use of two separate encoders to infer Z, via rigorous treatment and an illustrative example. Subsequently, in Section 3, we introduce an efficient way to implement two separate functions for inferring Z. In Section 3 we shall also introduce a principled way to ensure that the latent representation Z to contain useful information for prediction, and finally, in Section 4 we provide empirical evaluation.

4.1.1 Related Work

As previously mentioned, there is a vast body of work exploring approaches to make use of latent invariant representations for the purposes of unsupervised domain adaptation, using both deep learning and more traditional techniques. Namely, invariant representations were considered as a linear projection in (M. Gong and Schölkopf [84]), where theoretical guarantees were established regarding the conditional invariance of the latent representation and the identifiability of the joint distribution in the target domain. Furthermore, this framework was generalized to nonlinear maps to invariant latent representations, parameterized by neural network termed DANN (Ganin and Lempitsky [38]). In this framework, the constraint that $P^S(Z) = P^T(Z)$ can be enforced in various ways, including adversarial training, Maximum Mean Discrepancy (Long et al. [76]), Wassertstein Distance (Courty et al. [23]) and Margin Disparity Discrepancy (Zhang et al. [155]).

However, none of these studies have considered using two separate encoders for the two domains.

Subsequently, this reasoning has been considered in combination with image-to-image translation (CYCADA) (Hoffman et al. [57]). Another assumption frequently made in recent studies is that the input features X have a clustering structure in which each cluster has a different label Y. Therefore, further directions were pursued by incorporating additional information contained in the unlabeled target domain data, such as pseudo-labels provided by the classifier which is initially trained on the labeled source domain data (Long et al. [83], Wang and Breckon [132]) (termed CDAN and SLPP, respectively). In addition, one can also make use of this assumption to enforce that the prediction function g does not pass through high-density regions in input space, as performed by the DIRT-T algorithm (Shu et al. [108]). The method by (Tzeng et al. [126]) makes use of two separate encoding functions, but does not regularize their output to preserve semantic information in any way. In addition to these, approaches, the DSN algorithm (Bousmalis et al. [12]) takes into models the changing parameters of $P(X, Y)$ in the latent representation and enforces that they are different from the invariant representation Z. The only method that we are aware of, which takes domain-specific information into account is the study by (Wang et al. [129]), in which domain information is given and continuous, and therefore contains a lot more information than the discrete domain index that is typically considered. In (Gu et al. [54]), a spherical architecture is used for ϕ, however there is only one encoder for both domains. The study by (Na et al. [89]) performs bridging between the source and the target domains using data-augmentation, but this is a technique specific to image data.

While the above-mentioned methods use various approaches to improve the quality of the learnt invariant latent representation and show competitive performance, there is no guarantee that these approaches are general enough to encompass all scenarios. Namely, the challenge that existing methods face is two-fold: **(1)** there is no principled way to ensure that marginal invariance ($P^S(Z) = P^T(Z)$) preserves conditional invariance ($P^S(Z|Y) = P^T(Z|Y)$), **(2)** there are cases (especially in lower-dimensional datasets) where the encoding function ϕ needs domain-specific information, in addition to the features X. The aim of this paper is to address these two issues.In this paper, we introduce the first invariant representation learning method that makes use of the data-generating process to justify and efficiently use domain-specific information in the encoder ϕ, and to properly constrain it to ensure that Z contains relevant label-specific information.

4.2 What Does it Take to Learn a Useful Invariant Representation?

To gain insights into what information is necessary for learning an intermediate representation Z via ϕ, let us consider the data-generating process depicted on Figure 4.1. In this graphical model, the label Y is generated first from its prior distribution $P(Y)$. Then, the invariant representation Z is generated from Y via the generating mechanism $P(Z|Y)$ and X is subsequently generated from $P(X|Z; \theta_X)$, where θ_X are the changing parameters of $P(X|Y)$ across domains. We consider θ_X as a parameter vector with a constant value for each point within a domain, and we generally assume that the change across domains in $P(X|Y)$ is *minimal*. This generating process, in part or in whole, can be exploited in many classification applications. For example, in object classification in images, Y is the label (the object the picture was taken of), X is the observed pixels, and Z can correspond to latent features with semantic relevance, such as some function of the edges and contours on the image. In this case, θ_X can correspond to different resolution, illumination conditions, or other environment-specific changes that are not relevant for predicting the class Y. In this graphical representation, we see that generally speaking, Z is conditionally dependent from θ_X given X, although they may be marginally independent. This implies that in order to recover Z, given that X is considered, the information of θ_X should also be considered in the transformation, because it is in the Markov Blanket of Z.

To gain intuition why considering the Markov Blanket is important, consider the following scenario, depicted in Figure 4.2a. In this setting, we are given the source domain with Gaussian distribution, and the target domain with Uniform distribution. Here, we have a region $A \subset \mathcal{X}$ (highlighted in red), in which both of the domains' distributions have support, so there is support overlap. However, each domain has a different density in this region. The following proposition claims that in this situation, we cannot infer an invariant representation Z from X with a single encoder ϕ:

Proposition 1: *Let $A \subset \mathcal{X}$ be a region in input space such that $P^{\mathcal{S}}(X \in A) > 0$ and $P^{\mathcal{T}}(X \in A) > 0$, and let there be $a \subset A$ such that $P^{\mathcal{S}}(X \in a) \neq P^{\mathcal{T}}(X \in a)$. Furthermore, let $\phi : \mathcal{X} \rightarrow Z$ be an encoder s.t. $A = \{x : \phi(x) \in B\}$ for some $B \subset \mathcal{Z}$. Then, there is no function ϕ s.t. $P^{\mathcal{S}}(\phi(X) \in B) = P^{\mathcal{T}}(\phi(X) \in B)$.*

All omitted proofs can be found in the appendix of this chapter. This proposition implies that if the supports of the marginal distributions of the source and target domains overlap, and if the distributions are different in the specific region of the support overlap (set A), then we cannot use a fixed transformation to transform the data in that region to an invariant representation in $\mathcal{Z}$ space.

There is an additional case in which a single encoder ϕ is not sufficient to learn an optimal latent representation Z. The scenario is illustrated on Figure 4.2b. In this case, even though both domains have the same distribution $P(X)$ in the region of support overlap between the source and the target domain (labeled in red), the value of Y is opposite between the two domains. The following proposition demonstrates that in this setting, a single encoding function ϕ is inherently sub-optimal:

Proposition 2: *Let the true labeling functions in the source and target domain be $f_S, f_T : \mathcal{X} \rightarrow \mathcal{Y}$, respectively. Let $A \subset \mathcal{X}$ be a region s.t. $f_S(a) \neq f_T(a), \forall a \in A$. Let $g : \mathcal{X} \rightarrow \mathcal{Y}$ be a composition of a representation learner $\phi : \mathcal{X} \rightarrow \mathcal{Z}$ and a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$. If ϕ is the same function across domains, then for a 0-1 loss, the risk over the region A: $\epsilon^A(g) = \epsilon^A(g)_S + \epsilon^A(g)_T \geq 1$.*

In this scenario, no matter what criterion one uses to find the representation for downstream classification using h, as long as the two domains use the same transformation ϕ for the representation, the points in A cannot be correctly classified in the target domain, if they are correctly classfied in the source domain. This is because the final classifer of the data, $g = h \circ \phi$, is fixed across domains. As a consequence, any point $a \in A$ will be mapped to the same class label, no matter which domain it is in. These two scenarios generally occur in low-dimensional datasets, such as scientific and healthcare applications. In very high-dimensional structured data such as images and text, there will be some domain-specific dimensions which will have very different values in the source and the target domain (such as brightness in images in daytime vs. nighttime).

This will result in no overlap between the supports of the source and target domain distributions, and thus the domain-specific dimensions will act as domain specific information. In this case using ϕ which is only a function of X is sufficient. However, even in this scenario, we need to find a principled way to constrain the encoder such that Z contains relevant label-specific information for the prediction in the target domain.

The observations from the perspective of the causal mechanism of generating X from Z and θ_X provide hints how to tackle the problem of UDA using invariant latent representations in a general way. First of all, we observe that θ_X needs to be an input to the encoder ϕ, in addition to X. Secondly, in this causal mechanism, we can safely assume that the influence of θ_X on the relationship between X and Z (and therefore the relationship between X and Y) is minimal. This allows us to: **(1)** model θ_X as a latent variable and provide it is an input to $\phi(X, \theta_X)$, instead of learning two separate encoders $\phi_{\mathcal{S}}$ and $\phi_{\mathcal{T}}$; **(2)** mimic the causal mechanism of generating X, given by $X = \tilde{\phi}(Z, \theta_X)$, via a decoder $\tilde{\phi} : \mathcal{Z} \times \Theta \rightarrow \mathcal{X}$, in which we constrain the influence of θ_X to be *minimal*. The hope is then that such a decoder would act as a regularizer on the encoder ϕ, forcing it to preserve important semantic information when inferring Z. In this paper, we design a method based on this reasoning in order to address the problem of UDA in a principled manner.

4.3 Proposed Method: Domain-Specific Adversarial Training

Motivated by the above discussion, we aim to design a domain-adversarial network which can infer the latent domain-specific parameters θ_X. Let $\theta_X \in \{\theta_X^{\mathcal{S}}, \theta_X^{\mathcal{T}}\}$, which are its possible values in the source and target domains. Our goal is to design a neural network which can provide good estimates for them, given by $\hat{\theta}_X^{\mathcal{S}}$ and $\hat{\theta}_X^{\mathcal{T}}$, and subsequently make use of them for learning an appropriate invariant representation $Z = \phi(X, \theta_X)$ which is predictive of Y. The model for the proposed method DSAN (Domain-Specific Adversarial Network) is depicted in Figure 4.3.

We first consider the domain index of the data: $j \in \{\mathcal{S}, \mathcal{T}\}$ (which indicates whether a data-point comes from the source or target domain), and parameterize the estimated latent domain-specific parameters $\hat{\theta}_X^{(j)}$ as a function of the domain index of the i-the data-point j_i (0 for source, 1 for target domain). We make use of $\hat{\theta}_X^{(j)}$ and data encoder ϕ to obtain $\mathbf{z}_i^{(j)} = \phi(\mathbf{x}_i^{(j)}, \hat{\theta}_X^{(j)})$, where $\mathbf{x}_i^{(j)}$ and $\mathbf{z}_i^{(j)}$ represent the the i-th input point and latent representation point respectively, from the j-th domain. We can then use the latent representation $\mathbf{z}_i^{(j)}$ to obtain a softmax probability for the predicted label label $\tilde{y}_i^{(j)} = h(\mathbf{z}_i^{(j)})$, and this predictor can be trained in the source domain. To enforce the condition $P^S(Z) = P^T(Z)$, we also have an adversarial predictor which predicts the domain index, given by $\tilde{j} = h_a(\mathbf{z}_i^{(j)})$. In order to successfully infer the latent changes θ_X, our model also needs to make use of (or mimic) the data-generating process depicted on Figure 4.1, in particular the process of generating X from Z and θ_X. This can be achieved via a decoder $\tilde{\phi}$ which reconstructs the features X in both the source and the target domain, from Z and θ_X, given by $\tilde{\mathbf{x}}_i^{(j)} = \tilde{\phi}(\mathbf{z}_i^{(j)}, \hat{\theta}_X^{(j)})$. As described so far, our model consists of the following loss functions:

$$\begin{aligned}
\mathcal{L}_{reconst.} &= \frac{1}{m_S + m_T} \sum_{i=1}^{m_S+m_T} ||\mathbf{x}_i - \tilde{\mathbf{x}}||^2, \\
\mathcal{L}_{classif.} &= -\frac{1}{m_S} \sum_{c=1}^{C} \sum_{i=1}^{m_S} y_{ic} log(\tilde{y}_{ic}), \\
\mathcal{L}_{inv.} &= - \sum_{i=1}^{m_S+m_T} \{j_i \log \tilde{j}_i + (1 - j_i) \log(1 - \tilde{j}_i)\}, \\
\mathcal{L}_{cent.} &= - \sum_{i=1}^{m_T} h(\mathbf{z}_i^{\mathfrak{T}})^T \log(h(\mathbf{z}_i^{\mathfrak{T}}))
\end{aligned}$$

Here, $\mathcal{L}_{reconst.}$, $\mathcal{L}_{reconst.}$, and $\mathcal{L}_{inv.}$ are the reconstruction, classification and adversarial loss respectively. $\mathcal{L}_{cent.}$ is the conditional cross-entropy of the predictions in the target domain, and intuitively, it serves to enforce that the decision boundary does not cross data-dense regions (Shu et al. [108]). The current loss functions do not involve a constraint on the encoding transformation ϕ which would prevent it from discarding valuable information about Y. We now describe how we can make use of the data-generating process and the assumption of minimal change across domains to enforce such a constraint.

4.3.1 Enforcing Minimal Change by Mutual Information Minimization

Before we delve into the details of how to enforce minimality of the influence of θ_X on the generating process of X from Y, we need to find a way to measure it. One way to formally do so is the joint mutual information $I(\theta_X; (X, Y))$. Intuitively, this means that the change of the joint distribution (conditional distribution $P(X|Y)$ in the case of covariate shift) across domains is *minimal.* Conveniently, this term is known to be related to a distributional divergence measure that can be minimized, as stated by the following lemma (and is a well-established result):

Lemma 1: *Let $X, Y \sim P(X, Y; \theta_X)$, and let θ_X be a discrete random variable with possible values $\theta_X^{\mathfrak{S}}$ and $\theta_X^{\mathfrak{T}}$ in the source and target domains respectively, with a uniform prior $P(\theta_X = \theta_X^{\mathfrak{S}}) = 0.5$, $P(\theta_X = \theta_X^{\mathfrak{T}}) = 0.5$. Then, $I(\theta_X; (X, Y)) =$ JSD$(P_{X,Y;\theta_X=\theta_X^{\mathfrak{S}}} || P_{X,Y;\theta_X=\theta_X^{\mathfrak{T}}})$, where JSD is the Jensen-Shannon Divergence, and $P_{X,Y;\theta_X=\theta_X^{\mathfrak{S}}}$ and $P_{X,Y;\theta_X=\theta_X^{\mathfrak{T}}}$ are the source and target domain joint distributions respectively.*

Therefore, to measure the influence of θ_X on the generating process as mimicked by our decoder, we need to measure the JSD between the joint distributions as implied by the reconstructed data, given by JSD$(P_{X,Y;\theta_X=\hat{\theta}_X^{\mathfrak{S}}} || P_{X,Y;\theta_X=\hat{\theta}_X^{\mathfrak{T}}})$ (note that here we are using inferred domain-specific variables $\hat{\theta}_X^{\mathfrak{S}}$ and $\hat{\theta}_X^{\mathfrak{T}}$). In theory, given enough data and expressiveness of the decoder, we can assume that we can learn to reconstruct the data perfectly. However, due to lack of labeled data in the target domain, we can only reconstruct the unlabeled features in the target domain (and have no access to $P_{X,Y;\theta_X=\hat{\theta}_X^{\mathfrak{T}}}$). To cope with this, we can use the inferred domain-specific variable

$\hat{\theta}_X^{\mathcal{T}}$ to translate the source domain data to the target domain via the decoder: $\tilde{\mathbf{x}}^{\mathcal{S}}_{trans.} = \tilde{\phi}(\mathbf{z}^{\mathcal{S}}, \hat{\theta}_X^{\mathcal{T}})$. To minimize the Jensen-Shannon Divergence, one could use an adversarial approach to match the joint distributions $P^{\mathcal{S}}(\tilde{X}, Y)$ and $P^{\mathcal{S}}(\tilde{X}_{trans.}, Y)$ (Biau et al. [9]) (the joint distribution of the source reconstruction and the source labels, and the joint distribution of the translation from source to target and the source labels, respectively). This can be achieved by providing concatenated data and label vectors: $[\tilde{\mathbf{x}}^{\mathcal{S}}, y^{\mathcal{S}}]^T$ and $[\tilde{\mathbf{x}}^{\mathcal{S}}_{trans.}, y^{\mathcal{S}}]^T$ to an adversarially-trained classifier.

The following Theorem, adopted from (Gong et al. [43]), gives an upper bound of the JSD which is easy to minimize:

Theorem 1 (Gong et al. [43]): *Let $P_{YX|\theta_X=\theta^{\mathcal{S}}}$ and $P_{YX|\theta_X=\theta^{\mathcal{T}}}$ denote the source and target domain distributions respectively. Let $Q^{h_c}_{Y|X}$ denote the conditional distribution of Y given X specified by an auxiliary classifier h_c. We have:*

$$\begin{aligned} \mathit{JSD}(P_{XY;\theta_X=\theta_X^{\mathcal{S}}}||P_{XY;\theta_X=\theta_X^{\mathcal{T}}}) &\leq 2c_1\sqrt{\mathit{JSD}(P_{X|\theta_X=\theta_X^{\mathcal{S}}}||P_{X|\theta_X=\theta_X^{\mathcal{T}}})} \\ &+ c_2\sqrt{\mathit{KL}(P_{Y|X,\theta_X=\theta_X^{\mathcal{S}}}||Q^{h_c}_{Y|X})} + c_1\sqrt{\mathit{KL}(P_{Y|X,\theta_X=\theta_X^{\mathcal{T}}}||Q^{h_c}_{Y|X})} \end{aligned}$$

Here, JSD$(P_{X|\theta_X=\hat{\theta}_X^{\mathcal{S}}}||P_{X|\theta_X=\hat{\theta}_X^{\mathcal{T}}})$ is equal to the JSD between the source and target domain marginal distributions of X and is fixed, we can ignore this term when we minimize the upper bound using our algorithm. Therefore, to minimize the two KL divergences on the RHS:

$$\text{KL}(P_{Y|X,\hat{\theta}_X=\theta_X^{\mathcal{S}}}||Q^{h_c}_{Y|X}) \text{ and } \text{KL}(P_{Y|X,\theta_X=\hat{\theta}_X^{\mathcal{T}}}||Q^{h_c}_{Y|X}),$$

it suffices to train a joint auxiliary classifier h_c on the reconstructed source domain data $\tilde{x}^{\mathcal{S}}$ and the translated data $\tilde{x}^{\mathcal{T}}_{trans.}$. Namely, these divergences can be expressed as:

$$\begin{aligned} KL(P^{\mathcal{S}}(Y|f(X))||Q^{h_c}_{Y|X}) &= H(P^{\mathcal{S}}(Y|f(X)), Q^{h_c}_{Y|X}) + H(P^{\mathcal{S}}(Y|f(X)) \\ KL(P^{\mathcal{S}}(Y|X)||Q^{h_c}_{Y|X}) &= H(P^{\mathcal{S}}(Y|X), Q^{h_c}_{Y|X}) + H(P^{\mathcal{S}}(Y|X)), \end{aligned}$$

where,

$$\begin{aligned} H(P^{\mathcal{S}}(Y|f(X)), Q^{h_c}_{Y|X}) &= -\sum_{j=1}^{C}\sum_{i=1}^{m_S} y_{ij}\log(\tilde{y}^c_{ij,trans.}), \\ H(P^{\mathcal{S}}(Y|X), Q^{h_c}_{Y|X}) &= -\sum_{j=1}^{C}\sum_{i=1}^{m_S} y_{ij}\log(\tilde{y}^c_{ij}) \end{aligned}$$

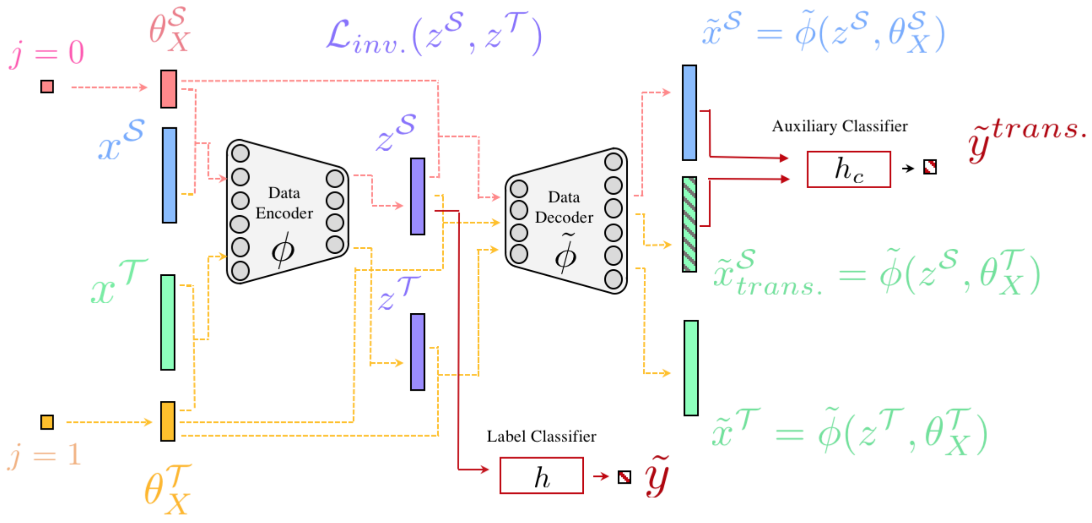
are the cross-entropy terms which need to be minimized. Therefore, we introduce a cross-entropy loss $\mathcal{L}^{trans.}_{inv.}$ for the auxiliary classifier, whose two terms can be used to minimize the two KL divergences respectively:

$$\mathcal{L}^{trans}_{inv.} = -\sum_{j=1}^{C}\sum_{i=1}^{m_S} y_{ij}\log(\tilde{y}^c_{ij}) - \sum_{j=1}^{C}\sum_{i=1}^{m_S} y_{ij}\log(\tilde{y}^c_{ij,trans.}), \text{ where } \tilde{y}^c_i = h_c(\tilde{x}^{\mathcal{S}}_i),$$

and $\tilde{y}^c_{i,trans.} = h_c(\tilde{x}^{\mathcal{S}}_{i,trans.})$ are the softmax predictions of the reconstruction and the translation of the source domain respectively, using predictor h_c. Therefore, the total loss function can be formulated in the following manner: $\mathcal{L}_{total} = \lambda_\alpha \mathcal{L}_{reconst.} + \lambda_\delta \mathcal{L}_{classif.} - \lambda_\gamma \mathcal{L}_{inv.} + \lambda_\tau \mathcal{L}^{trans.}_{inv.}$, and can be minimized by alternating optimization:

$$\min_{\phi,\tilde{\phi},\rho,h,h_c} \mathcal{L}_{total}$$

$$\min_{h_a} \lambda_\gamma \mathcal{L}_{inv.}.$$



$j = 0$
$\theta_X^{\mathcal{S}}$
$\mathcal{L}_{inv.}(z^{\mathcal{S}}, z^{\mathcal{T}})$
$\tilde{x}^{\mathcal{S}} = \tilde{\phi}(z^{\mathcal{S}}, \theta_X^{\mathcal{S}})$
$x^{\mathcal{S}}$
$z^{\mathcal{S}}$
Data Encoder ϕ
Data Decoder $\tilde{\phi}$
Auxiliary Classifier
h_c
$\tilde{y}^{trans.}$
$x^{\mathcal{T}}$
$z^{\mathcal{T}}$
$\tilde{x}^{\mathcal{S}}_{trans.} = \tilde{\phi}(z^{\mathcal{S}}, \theta_X^{\mathcal{T}})$
Label Classifier
h
$\tilde{y}$
$\tilde{x}^{\mathcal{T}} = \tilde{\phi}(z^{\mathcal{T}}, \theta_X^{\mathcal{T}})$
$j = 1$
$\theta_X^{\mathcal{T}}$

Figure 4.3: A diagram of the proposed autoencoder framework. Here, the domain index is mapped to θ_X. Subsequently, the input data $\mathbf{x}$ and θ_X are mapped via ϕ to a latent representation $\mathbf{z}$, which in turn is reconstructed $\tilde{\mathbf{x}}$ using the decoder $\tilde{\phi}$, and θ_X as additional input. In addition, the penalty $\mathcal{L}_{inv.}$ is enforced to ensure invariance between Z^S and Z^T. The hidden representation is also used to predict $\tilde{\mathbf{y}}$ in the source domain. Finally, $\mathcal{L}^{trans-y}_{inv.}$ enforces the minimal change of $P(X|Y)$ across domains. The pink and orange lines depict the flow of the source and target data and domain-specific information respectively.

This alternating minimization problem can be solved using Gradient-Reversal Layers (GRL), as shown in (Ganin et al. [39]). $\mathcal{L}_{inv.}$ can also be expressed in terms of Maximum Mean Discrepancy (MMD) (Gretton et al. [49]). Our model is now equipped with a way to ensure that when reconstructing the data, θ_X cannot play a major role in the decoder $\tilde{\phi}$. Since θ_X correspond to minimal changes of the conditional distribution $P(X|Y)$, this forces Z to retain Y-specific structure, and in turn implicitly regularizes the encoder ϕ, resulting in a conditionally invariant representation. In the following section, we will empirically demonstrate this property of the constraint.

4.4 Empirical Evaluation

In order to provide intuition about the proposed method's advantages in the challenging scenarios described above, we first designed a simple but informative 2D simulated example for domain adaptation using mixtures of Gaussian distributions. This simulated dataset is presented on Figure 4.4(a), and is representative of the example discussed in Figure 4.2, since there is a significant region of overlap between the two domains, in which the points have opposite labels across domains. In this experiment, we compared our method DSAN, with the following baselines:

(1) DSAN-unreg., which is our proposed method but without regularizing the transformation ϕ using the mutual information minimization in our loss;

(2) DSN, the method proposed in (Bousmalis et al. [12]), which makes use of domain-specific private encodings, but which are only used for reconstruction of the data, and not for inferring the shared invariant representation Z;

(3) DANN, the classical domain-adversarial method proposed by (Ganin et al. [39]). We used a 2D hidden representation Z (more details about implementation and tuning can be found in the supplement).

The results are presented in Table 4.3, demonstrating that our approach greatly outperforms the baselines. To visualize how our algorithm is able to solve this challenging example with much greater success, we also present the 2D invariant representations in the source and target domains on Figure 5.5. At the top, we have the hidden representations Z of DSAN-unreg., and on the bottom the ones yielded by our method DSAN. On the left side we present Z in the source domain, labeled by the true values, in the middle we show this representation in the target domain labeled with the prediction of the algorithm, and on the right-hand side we present Z in the target domain labeled with the true labels. From this figure, one can appreciate that the unregularized version of our approach yields a representation Z which discards a lot of information about Y, as can be seen in parts 4.5b and 4.6a. On the other hand, parts 4.5e and 4.5f show that our algorithm with the mutual information minimization constraint can successfully preserve label-specific information in the invariant representation.

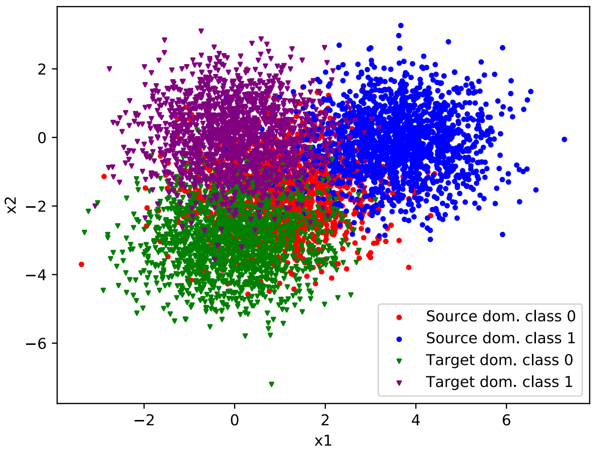
4.4.1 Real Data Experiments

To evaluate our method on real datasets, we consider three datasets and respective tasks from various domains of applications: cross-domain Wi-Fi localization, Amazon product reviews and image classification. For detailed descriptions of the experimental design, hyperparameter tuning and neural network architectures, as well as ablation studies, we refer the interested reader to the supplementary materials. We performed evaluation on the following datasets: **Wi-Fi localization**: Introduced in (Zhang et al. [146]), is a dataset in which wireless signal data was collected in a hallway area discretized as a grid. In each grid, data was recorded from 67 access points (dimensions). The task is to predict the location from the signals (which has been converted to a classification problem with 19 classes). There are three domains in this dataset (with 1140 points each), collected in three different time points. This is a low-dimensional dataset, in which we verified via manual inspection and tSNE and PCA projections that both scenarios of support overlap described in Chapter 2 hold; **Amazon Review:** Amazon Review is another benchmark for multi-domain sentiment analysis. It contains positive and negative reviews of four kinds of products: Kitchen appliance (K), DVDs (D), Electronics (E), and Books (B). **ImageCLEF:** A standard UDA benchmark dataset for image classification, consisting of three domains: *Caltech-256*(C), *ImageNet ILSVRC*(I), and *Pascal VOC2012*(P), consisting of 12 classes. **Baselines:** We compare DSAN with classical approaches like TCA (Pan et al. [91]) and GFK (Gong et al. [41]), as well as with some deep transfer learning models. Three recently proposed methods, DSR (Cai et al. [14]), DIRT-T (Shu et al. [107]), MDD (Zhang et al. [154]), BSP (Chen et al. [19]), and RSDA (Gu et al. [54]) are included. For both the real-world and simulated experiments, we ran and tuned the baseline methods ourselves (all baselines for Wi-Fi and simulated, and MDD and DIRT-T for Amazon dataset).

The classification accuracies on the Amazon-Review Dataset for unsupervised domain adaptation are shown in Table 4.2, which shows that our algorithm model outperforms the baselines in the vast majority of source-target directions. We also perform a Wilcoxon signed-rank test on the experiment result of different random seed, and the p-value is 0.0216. The performance of our method on the Wi-Fi localization dataset is shown on Table 4.1. In this experiment, we sub-sampled 950 points in each domain to create 10 replicate experiments, and we present the accuracies and standard deviations across all replicates. From the results, one can appreciate that our method outperforms all baselines, for the majority of the pairs. For one of the remaining two pairs in which it does not, the performance is within margin of error with the strongest baseline. We calculated a Wilcoxon p value $p = 0.002$ across all replicate experiments combined. The performance of our method on the ImageCLEF dataset is presented on Table 4.4(b), in which one can appreciate that we achieve state-of-the-art performance on all transfer directions.

4.5 Conclusion

In this paper, we have demonstrated that the existing paradigm of representation learning for domain adaptation using a fixed encoder ϕ of the data, can be sub-optimal when the same points can have different labels across domains. We have introduced a method which takes domain-specific information in order to provide the data encoder with the necessary flexibility and appropriate constraints, in order to retain valuable information about Y in the latent representation Z. We note that in many real-world applications with high-dimensional datasets, the supports of the source and target domains may not overlap. In this case, domain-specific information is contained in the features X and can be automatically used by the encoder $\phi(X)$. However, $\phi(X)$ would then have the same problem with excessive flexibility and can still suffer from finding trivial reprepresentatons of Z that discard information about Y. Therefore, our principle for constraining domain-specific encoders would still be beneficial in this scenario. For future work, a promising direction is combining this framework motivated by the data-generating process with more powerful encoders, such as the spherical neural network proposed by (Gu et al. [54]).



(a) Scatter plot of simulated data

(b) Accuracy (%) on ImageCLEF dataset for unsupervised domain adaptation (reproduced by (Gu et al. [54])

)

Models	I → P	P → I	I → C	C → I	C → P	P → C	Average
DResNet-50 (He et al. [55])	74.8	83.9	91.5	78.0	65.5	91.2	80.7
iCAN (Zhang et al. [151])	79.5	89.7	94.7	89.9	78.5	92.0	87.4
CDAN (Long et al. [81])	77.7	90.7	97.7	91.3	74.2	94.3	87.7
SymNets (Zhang et al. [152])	80.2	93.6	97.0	93.4	78.7	96.4	89.9
SAFN+ENT (Xu et al. [137])	79.3	93.3	96.3	91.7	77.6	95.3	88.9
CAT (Deng et al. [28])	77.2	91.6	95.5	91.3	75.3	93.6	87.3
DANN (Ganin and Lempitsky [38])	75.0	86.0	96.2	87.0	74.3	91.5	85.0
DANN+S (Gu et al. [54])	78.3	91.0	96.8	91.8	77.7	95.2	88.5
RSDA-DANN (Gu et al. [54])	79.2	93.0	**98.3**	93.6	78.5	**98.2**	90.1
MSTN (Xie et al. [135])	77.3	91.3	96.8	91.2	77.7	95.0	88.2
DSAN-U	79.4	94.5	96.8	94.5	76.1	88.5	88.3
DSAN	**81.0**	**95.7**	**97.3**	**95.3**	**80.5**	**97.0**	**91.1**

Figure 4.4

Table 4.1: Accuracy (%) on Wi-Fi localization dataset for unsupervised domain adaptation

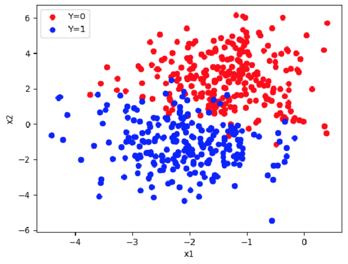
Models	t1 → t2	t1 → t3	t2 → t1	t2 → t3	t3 → t1	t3 → t2	Average
DANN (Ganin and Lempitsky [38])	31.30((1.3)	33.15(1.9)	*38.03(2.0)*	32.41(2.3)	28.54((0.8)	31.12(1.6)	32.59
DSN (Bousmalis et al. [12])	31.25(1.5)	34.23(2.4)	34.32(2.8)	29.47(0.7)	27.0(1.6)	31.8(1.3)	31.35
DSR (Cai et al. [14])	34.20(1.3)	34.62(4.8)	30.61(3.5)	34.72(1.1)	33.13(0.5)	31.61(0.9)	31.13
MDD (Zhang et al. [154])	30.92(2.1)	34.42(1.8)	30.01(1.5)	27.58(1.8)	33.25(1.0)	27.34(2.3)	30.59
DIRT-T (Shu et al. [107])	36.38(1.2)	33.66(2.7)	37.12(3.4)	31.85(2.87)	33.98(1.42)	28.48(2.68)	33.58
BSP (Shu et al. [107])	35.5(1.3)	32.16(2.1)	31.15(1.4)	31.06(1.9)	32.91(1.1)	33.83(2.2)	32.72
DSAN-U	31.16(1.1)	29.01(1.7)	32.76(0.9)	32.18(1.1)	31.02(1.7)	**33.62(1.8)**	31.45
DSAN	**40.45(1.6)**	**37.35(2.3)**	**38.22(2.4)**	**39.94(2.1)**	**36.47(3.5)**	**34.18(2.9)**	**37.77**

Table 4.2: Accuracy (%) on Amazon Review dataset for unsupervised domain adaptation

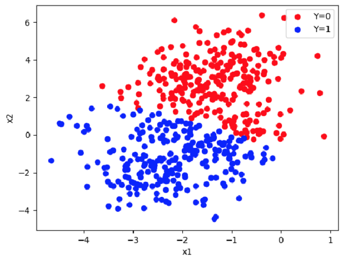
Methods	B→D	B→E	B→K	D→B	D→E	D→K	E→D	E→B	E→K	K→B	K→D	K→E	Avg
NN	49.6	49.8	50.3	53.3	51.0	53.1	50.8	50.9	51.2	52.2	51.2	52.3	51.3
TCA (Pan et al. [91])	63.6	60.9	64.2	63.3	64.2	69.1	59.5	62.1	74.8	64.1	65.4	74.5	65.5
GFK (Gong et al. [41])	66.4	65.5	69.2	66.3	63.7	67.7	62.4	63.4	73.8	65.5	65.0	73.0	66.8
SA (Fernando et al. [34])	67.0	70.8	72.2	67.5	67.1	69.4	61.4	64.9	70.4	64.4	64.6	68.2	67.3
BDA (Wang et al. [130])	64.2	62.1	65.4	62.4	66.3	68.9	59.6	61.6	74.7	62.7	64.3	74.0	65.5
CORAL (Sun et al. [123])	71.6	65.1	67.3	70.1	65.6	67.1	67.1	66.2	77.6	68.2	68.9	75.4	69.1
JGSA (Zhang et al. [141])	66.6	75.0	72.1	55.5	67.3	65.6	51.6	50.8	55.0	58.3	56.4	51.7	60.5
DANN (Ganin and Lempitsky [38])	78.4	73.3	77.9	72.3	75.4	78.3	71.3	73.8	85.4	70.9	74.0	84.3	64.9
MDD (Zhang et al. [154])	77.1	74.4	77.0	74.7	74.1	76.3	72.4	70.2	83.3	69.3	73.2	82.8	75.4
DIRT-T (Shu et al. [107])	78.6	76.1	75.5	76.8	75.2	79.1	69.6	71.0	84.2	69.2	73.3	79.5	75.7
EasyTL (Wang et al. [131])	79.8	79.7	80.9	79.9	80.8	82.0	75.0	**75.3**	84.9	**76.5**	76.3	82.5	79.5
BSP (Chen et al. [19])	79.33	73.86	75.86	75.67	74.77	77.2	72.81	71.3	84.02	70.92	73.59	84.29	76.14
RSDA (Gu et al. [54])	80.1	77.8	84.4	76.6	79.4	82.3	73.6	71.1	86.6	71.8	74.7	84.4	77.9
DSAN-U	81.0	78.6	78.9	78.4	79.4	83.3	76.2	74.5	87.4	73.1	77.0	83.0	78.9
DSAN	**82.7**	**80.8**	**82.6**	79.5	**81.4**	**85.3**	**76.7**	75.1	**88.0**	73.8	**77.3**	**85.0**	**80.7**

Table 4.3: Results on the simulated data.

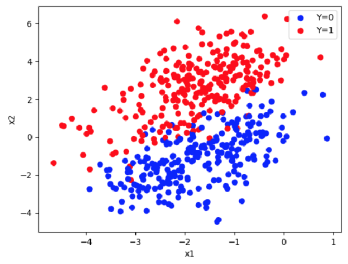
Model	DANN	DSN	DSAN-unreg.	DSAN
Accuracy	58.2(4.3)	58.8(5.2)	79.4(2.2)	**87(1.0)**



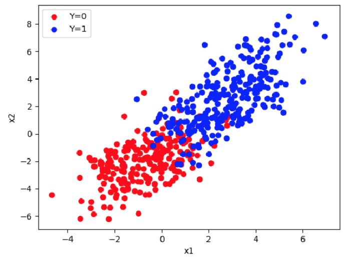
(a)



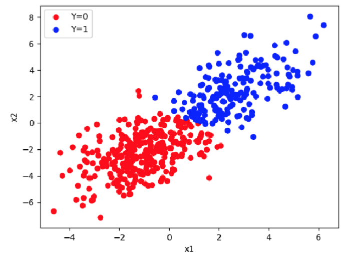
(b)



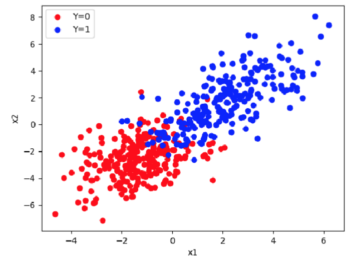
(c)



(d)



(e)



(f)

Figure 4.5:
Z in the simulations. First three scatter-plots show it without regularization: (a) Z of the source domain labeled with true labels, (b) Z of the target domain labeled with the predictions (c) Z of the target domain with the true labels. (d), (e), and (f) show the same, but with regularization.

4.6 Proofs for Theoretical Statements

In this section we provide proofs for theoretical statements in the chapter.

4.6.1 Proof of Proposition 1:

Before proving this proposition, let us re-state it:

Proposition 1: *Let $A \subset \mathcal{X}$ be a region in input space such that $P^{\mathcal{S}}(X \in A) > 0$ and $P^{\mathcal{T}}(X \in A) > 0$. Furthermore, let $\phi : \mathcal{X} \rightarrow Z$ be an encoder s.t. $A = \{x : \phi(x) \in B\}$ for some $B \subset \mathcal{Z}$. Then, there is no function ϕ s.t. $P^{\mathcal{S}}(\phi(X) \in B) = P^{\mathcal{T}}(\phi(X) \in B)$.*

Proof. Let $\mathcal{X}$ be the input feature space, and $\mathcal{Z}$ be the latent space in which the invariant representation is learned via encoder $\phi : \mathcal{X} \rightarrow \mathcal{Z}$.

Let there be a subset in the invariant space $B \subset \mathcal{Z}$, and suppose that we have marginal invariance in the latent space: $P^{\mathcal{S}}(\phi(X) \in B) = P^{\mathcal{T}}(\phi(X) \in B), \forall B$. Define the pre-image of B

as:

$$A = \{x \in \mathcal{X} : \phi(x) \in B\}.$$

Then, we have the following equality relationships:

$$\begin{aligned} P^{\mathcal{S}}(\phi(X) \in B) &= P^{\mathcal{S}}(X \in A) \\ = P^{\mathcal{T}}(\phi(X) \in B) &= P^{\mathcal{T}}(X \in A), \end{aligned}$$

which implies that $P^{\mathcal{S}}(X \in A) = P^{\mathcal{T}}(X \in A)$ must hold. This further implies that if there is difference in the source and target distributions in the support-overlapping region A, given by:

$$P^{\mathcal{S}}(X \in A) \neq P^{\mathcal{T}}(X \in A),$$

then the following must hold: $P^{\mathcal{S}}(\phi(X) \in B) \neq P^{\mathcal{T}}(\phi(X) \in B)$. Therefore, if there is a region of support overlap for which the distributions of the source and target domain are different, a single transformation ϕ cannot be used to map data in this region to a marginally-invariant representation. □

4.6.2 Proof of Proposition 2:

Before proving this proposition, let us re-state it:

Proposition 2: *Let the true labeling functions in the source and target domain be $f_S, f_T : \mathcal{X} \rightarrow \mathcal{Y}$, respectively. Let $\mathcal{A} \subseteq \mathcal{X}$ be a region s.t. $f_S(a) \neq f_T(a), \forall a \in \mathcal{A}$. Let $g : \mathcal{X} \rightarrow \mathcal{Y}$ be a composition of a representation learner $\phi : \mathcal{X} \rightarrow \mathcal{Z}$ and a classifier $h : \mathcal{X} \rightarrow \mathcal{Y}$. If ϕ is the same function across domains, then for a 0-1 loss, the risk over the region $\mathcal{A}$ is $\epsilon^{\mathcal{A}}(g) = \epsilon^{\mathcal{A}}_S(g) + \epsilon^{\mathcal{A}}_T(g) \geq 1$.*

Proof: The total risk for the 0-1 loss is:

$$\epsilon^{\mathcal{A}}(g) = \epsilon^{\mathcal{A}}_S(g) + \epsilon^{\mathcal{A}}_T(g) = \text{Pr}(g(a) \neq f_S(a)) + \text{Pr}(g(a) \neq f_T(a)). \tag{4.1}$$

Since $f_S(a) \neq f_T(a), \forall a \in \mathcal{A}$, g either predicts the correct label for a in one domain and not the other, or predicts the wrong label in both. Therefore, for the multiple-class classification problem, the following inequality holds:

$$\text{Pr}(g(a) \neq f_T(a)) \geq 1 - \text{Pr}(g(a) \neq f_S(a)), \tag{4.2}$$

so it follows that:

$$\epsilon^{\mathcal{A}}(g) = \epsilon^{\mathcal{A}}_S(g) + \epsilon^{\mathcal{A}}_T(g) = \text{Pr}(g(a) \neq f_S(a)) + \text{Pr}(g(a) \neq f_T(a)) \geq 1.$$

□

4.6.3 Proof of Lemma 1:

Before proving this lemma, we shall first state it here:

Lemma 1: *Let $X, Y \sim P(X, Y; \theta_X)$, and let θ_X be a discrete random variable with possible values $\theta_X^{\mathcal{S}}$ and $\theta_X^{\mathcal{T}}$ in the source and target domains respectively, with a uniform prior $P(\theta_X = \theta_X^{\mathcal{S}}) = 0.5$, $P(\theta_X = \theta_X^{\mathcal{T}}) = 0.5$. Then, $I(\theta_X; (X, Y)) =$ JSD$(P_{X,Y;\theta_X=\theta^{\mathcal{S}}}||P_{X,Y;\theta_X=\theta^{\mathcal{T}}})$, where JSD is the Jensen-Shannon Divergence.*

Proof: From the general definition of mutual information, and the by factoring the joint distribution according to the generating process in Figure 4.1 we have:

$$\begin{aligned}I(\theta_X; (X, Y)) &= \int p_{XY\theta_X}(x, y, \theta) \log \frac{p_{XY\theta_X}(x, y, \theta)}{p_{\theta_X}(\theta) p_{XY}(x, y)} dx dy d\theta \\ &= \int p_{XY\theta_X}(x, y, \theta) \log \frac{p_{XY|\theta_X}(x, y|\theta)}{p_{XY}(x, y)} dx dy d\theta.\end{aligned}$$

Since θ_X is a discrete random variable with a uniform prior, the above expression can be rewritten as:

$$\begin{aligned}I(\theta_X; (X, Y)) &= \frac{1}{2} \int p_{XY|\theta_X=\theta_X^{\mathcal{S}}}(x, y) \log \frac{p_{XY|\theta_X=\theta_X^{\mathcal{S}}}(x, y)}{p_{XY}(x, y)} dx dy \\ &+ \frac{1}{2} \int p_{XY|\theta_X=\theta_X^{\mathcal{T}}}(x, y) \log \frac{p_{XY|\theta_X=\theta_X^{\mathcal{T}}}(x, y)}{p_{XY}(x, y)} dx dy \\ &= \frac{1}{2} \text{KL}(P_{XY|\theta_X=\theta_X^{\mathcal{S}}}||P_{XY}) + \frac{1}{2} \text{KL}(P_{XY|\theta_X=\theta_X^{\mathcal{T}}}||P_{XY}) \\ &= \text{JSD}(P_{XY|\theta_X=\theta_X^{\mathcal{S}}}||P_{XY|\theta_X=\theta_X^{\mathcal{T}}}).\square \qquad (4.3)\end{aligned}$$

4.7 Implementation Details

In this section we provide implementation details for the experiments performed in this chapter.

4.7.1 Enforcing Invariance with Maximum Mean Discrepancy

When the input dataset is low-dimensional, it is also possible to enforce marginal invariance for the representation Z using MMD (Maximum Mean Discrepancy) (Gretton et al. [48]) between the representations Z in the source and target domains, instead of adversarial training. Let $\mathbf{z}_i^{\mathcal{S}}$, $\mathbf{z}_i^{\mathcal{T}}$ denote the i-th instance of Z in the source and target domains respectively. Then, the MMD loss can be written as follows:

$$\mathcal{L}_{inv.}^{MMD} = \frac{1}{m_S^2} \sum_{i,j=1}^{m_S} \kappa(\mathbf{z}_i^{\mathcal{S}}, \mathbf{z}_j^{\mathcal{S}}) + \frac{1}{m_S m_T} \sum_{i,j=1}^{m_S, m_T} \kappa(\mathbf{z}_i^{\mathcal{S}}, \mathbf{z}_j^{\mathcal{T}}) + \frac{1}{m_T^2} \sum_{i,j=1}^{m_T} \kappa(\mathbf{z}_i^{\mathcal{T}}, \mathbf{z}_j^{\mathcal{T}}).$$

We followed the procedure in (Bousmalis et al. [12]), and used a mixture kernel function of q RBF kernels: $\kappa(\mathbf{z}_1, \mathbf{z}_2) = \sum_{i=1}^{q} \eta_i \exp\{-||\mathbf{z}_1 - \mathbf{z}_2||^2\}/\sigma_i^2$, where σ_i^2 is the kernel width of the i-th kernel, and η_i is a mixing weight which we set to $1/q$. For all experiments where we used MMD, we used the following RBF kernels (indexed by the respective kernel width): $\sigma_i^2 = [10^{-2}, 10^{-1}, 1, 5, 10, 50, 100, 500, 1000, 5000, 10000]$. Then, the total loss can be modified as follows:

$$\mathcal{L}_{total} = \lambda_\alpha \mathcal{L}_{reconst.} + \lambda_\delta \mathcal{L}_{classif.} + \lambda_\gamma \mathcal{L}_{inv.}^{MMD} + \lambda_\tau \mathcal{L}_{inv.}^{trans.}, \tag{4.4}$$

and in this case, alternating optimization is not needed. Then, we can simply minimize this total loss:

$$\min_{\phi, \tilde{\phi}, \rho, h, h_c} \mathcal{L}_{total}. \tag{4.5}$$

4.7.2 Using Pseudo-Labels and Ablation Study

In the main text, for the ImageCLEF dataset, in addition to $\mathcal{L}_{total}$ as described in the methods section, we make use of pseudo-labels to refine the quality of the invariant latent representation Z. For this purpose, we make use of the centroid procedure described in (Xie et al. [135]). Namely, for this dataset we introduce an additional term to the loss. Given C class labels and a distance metric Φ (we use Euclidean distance), we have:

$$\mathcal{L}_{SM} = \sum_{j=1}^{C} \Phi(C_j^{\mathcal{S}}, C_j^{\mathcal{T}}),$$

where $C_j^{\mathcal{S}}, C_j^{\mathcal{T}}$ are the centroids for the j-the class in the source and target domains respectively. Thus, we add this loss function to our total loss, in addition to the other terms described in the main text.

Ablation Study

To demonstrate the efficacy of the proposed procedure, we performed ablation studies to assess the contribution of the following design choices:

- the various terms in our total loss function,
- the need to have domain-specific information in the encoder and decoder

In Table 4.4 we present the results. Here, the columns are described as follows:

- "Dec." indicates whether a decoder is being used at all.
- "θ-Enc." indicates whether domain-specific information is being used in the encoder.
- "θ-Dec." indicates whether domain-specific information is being used in the decoder.
- "Entropy" indicates whether the conditional-entropy loss is being used.
- "Pseudo-labels" indicates whether pseudo-labels are being used.

From the results, a simple conclusion that can be derived is that the constraining the decoder according to the minimality of the generating process (via mutual information minimization) is instrumental to the performance of our method, and to attaining state-of-the-art performance.

Dec.	θ-Enc.	θ-Dec.	Entropy	Pseudo	Reg.	I → P	P → I	I → C	C → I	C → P	P → C	Avg.
×	✓	N/A	✓	✓	✓	77.5	89.2	95	91.8	74.2	92.3	87.53
✓	✓	×	✓	✓	✓	78.8	93.2	95.7	93.7	77.83	92.6	88.63
✓	×	✓	✓	✓	✓	79.7	93.5	96.5	93.00	79.5	95.3	89.58
✓	✓	✓	×	×	✓	80.0	90.3	95.0	93.5	72.1	94.3	87.53
✓	✓	✓	×	✓	✓	80.0	90.7	95.1	94.00	79.00	94.0	89.7
✓	✓	✓	✓	×	✓	80	95.0	96.1	93.7	78.3	96.7	89.96
✓	✓	✓	✓	✓	×	79.4	94.5	96.8	94.5	76.1	88.5	88.3
✓	✓	✓	✓	✓	✓	**81.0**	**95.7**	**97.3**	**95.3**	**80.5**	**97.0**	**91.1**

Table 4.4: Ablation study on the ImageCLEF dataset.

4.7.3 Experimental Setting, Neural Network Architectures, and Hyperparameters

In this section we provide details regarding the empirical evaluation in the main text. We note that unless mentioned otherwise, all non-linearity activations of hidden units in the neural network model are ReLU. We also note that in all of our experiments θ_X is one-dimensional and parameterized by a linear function of j, the domain index. Furthermore, in our implementation, the inputs X and Z were concatenated with θ_X before being passed to the encoder and decoder ϕ and $\tilde{\phi}$ respectively.

Simulated Experiments

The simulated dataset was generated in the following manner: we first let $Z \sim 0.5\mathcal{N}([0,0]^T, \mathbf{I}_2) + 0.5\mathcal{N}([0,3]^T, \mathbf{I}_2)$, where the two mixtures correspond to labels $Y = 0$ and $Y = 1$ respectively, where $P(Y = 0) = P(Y = 1) = 0.5$, and $\mathbf{I_2}$ corresponds to a 2×2 identity matrix. Then, we let the source domain data $X^S = \mathbf{R}(Z + [2,0]^T)$ and $X^T = Z + [0,-3]^T$, where $\mathbf{R}$ is a rotation matrix which rotates the data 45 degrees clockwise.

We performed 10 replicate experiments, in which we generated 3200 points in the source and target domains each, as described above. For each replicate, the same random seed was used for each method. As mentioned in the main text, all baselines and the proposed method used the same architecture:

- **The encoder** ϕ is a multi-layer perceptron (MLP) with two hidden layers of 128 dimensions each.
- **The invariant representation** Z has 2 dimensions, so that it can be easily visualized.
- **The classifier** h is an MLP of a single hidden layer of 128 dimensions.
- **The decoder** $\tilde{\phi}$ (for DSAN and DSAN-unreg.) is an MLP with two hidden layers of 128 dimensions each.
- **The regularization (auxiliary) classifier:** h_c is an MLP with with two hidden layers of 512 dimensions each.

For this experiment, to enforce invariance we used the MMD formulation of our approach instead of adversarial training (hence, no details regarding h_a are provided). The hyper-parameters used for training are presented on Table 4.5.

Hyperparameter:	**Value:**
learning rate	0.011
λ_α	0.1
λ_γ	0.2
λ_δ	0.5
batch size	128
optimizer	SGD
L_2 weight decay	1.00E-06
dropout	0
batch normalization	0
λ_τ	1

Table 4.5: Hyperparameter settings for the simulated dataset experiments.

Wireless Localization Experiments

Similarly to the simulated experiment, we performed 10 replicate experiments subsampling 950 points from a total of 1140, for each of the three domains. Furthermore, we used the same architecture of the neural network for the baselines and for the proposed method, where applicable. The neural network architecture can be summarized as follows:

- **The encoder** ϕ is a multi-layer perceptron (MLP) with two hidden layers of 512 dimensions each.
- **The invariant representation** Z is of 20 dimensions.
- **The classifier** h is an MLP of a single hidden layer of 128 dimensions with ReLU activation.
- **The decoder** $\tilde{\phi}$ (for DSAN, DSAN-unreg., and DSN) is an MLP with two hidden layers of 512 dimensions each.
- **The regularization (auxiliary) classifier:** h_c is an MLP with with two hidden layers of 512 dimensions each.

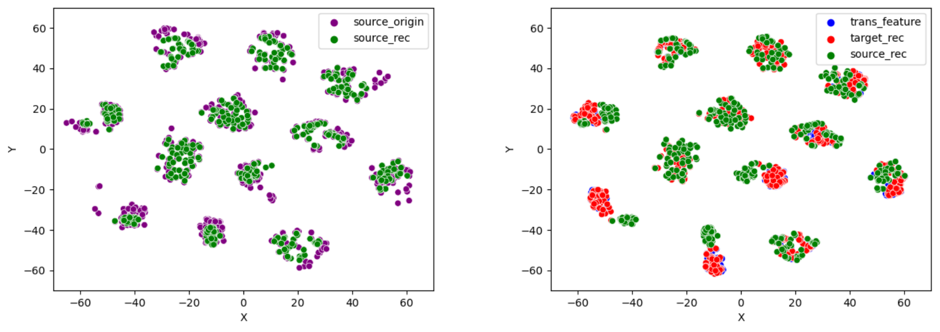
For this experiment, to enforce invariance we used the MMD formulation of our approach instead of adversarial training. Each replicate experiment was run for 400 epochs, and the accuracies reported are the ones at the last epoch. The hyperparameter settings can be found on Table 4.6.

We also considered MDD (Zhang et al. [155]) and DIRT-T (Shu et al. [108]) on this dataset. For each baseline, the encoder was selected among MLP dimensionalities 64, 128 and 512, and between one and two layers in depth. The same dimensionalities were used as options for the hidden representation Z. We selected the one with best performance in each respective baseline. The hyperparameter settings used are as follows:

- for MDD, the encoder consists of two layers of 128 dimensions each. The hidden representation is 128, and the learning rate is 0.01. For this experiment, we used batch normalization
- for DIRT-T, the encoder consists of two layers of 64 dimensions each. The hidden representation is 64, and the learning rate is 2e-03. For this experiment, we did not use batch normalization

Hyperparameter:	Value:
learning rate	0.001
λ_α	0.01
λ_γ	0.4
λ_δ	0.4
batch size	128
optimizer	SGD
L_2 weight decay	1.00E-06
dropout	0
batch normalization	0
λ_τ	2

Table 4.6: Hyperparameter settings for the wireless localization dataset experiments.



(a) (b)

Figure 4.6: tSNE figures of representations and features of the ImageCLEF dataset: (a) source original features and source reconstructed features (b) the source reconstructed features (green), the target reconstructed features (red), and the source translated data to the target domain (blue).

4.7.4 Visualization of Reconstructions

For the ImageCLEF dataset, we performed a dimensionality reduction via tSNE, and we visualize the reconstructions of the source and target domain data, as well as the translation from the source to the target domain on Figure 4.6.

Sentiment Prediction in the Amazon Product Reviews

For the experiments on the Amazon dataset, we performed 5 runs on different random seeds using the entire dataset. Each replicate experiment was run for 150 epochs, and the accuracies reported are the best ones across epochs. We provide the details regarding the architecture used below:

- **The encoder** ϕ is a multi-layer perceptron (MLP) with a single hidden layer of 4000 dimensions.
- **The invariant representation** Z is of 4096 dimensions.
- **The classifier** h is an MLP of a single hidden layer with 100 dimensions.

- **The decoder** $\tilde{\phi}$ is an MLP with a single layer of 5000 dimensions (which is the input dimensionality of the Amazon Product Reviews data. It is only a linear transformation with ReLU activation, and no hidden layers).
- **The domain-adversarial discriminator** h_a is an MLP of a single hidden layer with 100 dimensions.
- **The regularization (auxiliary) classifier:** h_c is an MLP with two hidden layers of 3000 dimensions each.

The hyperparameter setting used for this experiment is presented on Table 4.9.

Hyperparameter:	Value:
learning rate	1.00E-04
λ_α	1
λ_γ	1
λ_δ	1
batch size	128
optimizer	Adam
L_2 weight decay	1.00E-05
dropout	0.75
batch normalization	0
λ_τ	0.005

Table 4.7: Hyperparameter settings for the Amazon Product Reviews dataset experiment.

Regarding the baselines for this experiment, for a fair comparison, the result of most baselines are directly reported from their original papers wherever available. In order to include more recent state-of-the-art baselines, we also compare our model with DIRT (Shu et al. [108]) and MDD (Zhang et al. [155]). For these two baselines, we ran the experiments ourselves using the respective source codes. The hyper-parameter settings are shown in Table 4.8. To select the

	RSDA	DIRT-T	MDD
dropout rate	0.5	0.5	0.5
feature demension	4096	4096	4096
batch size	64	128	128
learning rate	0.04	0.002	0.01
optimizer	SGD	Adam	SGD

Table 4.8: Hyperparameter settings for the Amazon Product Reviews dataset experiment for baselines

hyperparameters in Table 4.8, we try different hyperparameter combinations according to various ranges for each hyper-parameter. For DIRT-T, the range of learning rate considered is 0.001-0.01, the range of dropout rate is 0.1-0.9. For MDD, the range of learning rate is [0.005,0.015], the range of dropout rate is 0.1-0.9 and the range of gamma is [1,2,3,4,5]. We try 5 different random seeds for each hyper-parameters combination and choose the best result.
We use the same dimensionality of the latent representation Z for both baselines, and we used the same encoder architecture as used in the proposed method. The encoder architecture was selected

for the proposed method and for these baselines according to the best performance obtained on the basic baseline DANN (Ganin et al. [39]). The dimensionality of the hidden representation Z was selected among the values: 512, 1024, 2048, 4096 and the one with the best performance in DANN was retained.

Image Classification in the ImageCLEF Dataset

Here we present details regarding the experiment on the ImageCLEF dataset. For all baselines, we used reproduced results reported by previous studies.

- **The encoder** ϕ is a multi-layer perceptron (MLP) with a single hidden layer of 2048 dimensions.
- **The invariant representation** Z is of 2048 dimensions.
- **The classifier** h is an MLP of a single hidden layer with 2048 dimensions.
- **The decoder** $\tilde{\phi}$ is an MLP with a single of 2048 dimensions (which is the input dimensionality of the ImageCLEF data.).
- **The domain-adversarial discriminator** h_a is an MLP of two hidden layer with 100 dimensions.
- **The regularization (auxiliary) classifier:** h_c is an MLP with two hidden layers of 2048 dimensions each.

Hyperparameter:	Value:
learning rate	0.05
λ_α	0.01
λ_γ	1
λ_δ	0.001
batch size	64
optimizer	Adam
L_2 weight decay	1.00E-04
dropout	0.70
batch normalization	0
λ_τ	1.0
vada hyperparameter 1	1.0
vada hyperparameter 2	0.5
pseudo label hyperparameter 2	0.5

Table 4.9: Hyperparameter settings for the ImageCLEF dataset experiment.

4.8 Extended Discussion on Related Work

In this section, we briefly discuss the relationship between our work and other related methods for domain adaptation using neural architectures. In the main text, we mentioned recent advances in neural architectures such as spherical CNNs (Gu et al. [54]), source-target domain bridging via data augmentation (Na et al. [89]) and pseudo-labels (Xie et al. [135]). We note that these techniques do not mimic the data-generating process and ensure that all information is available to infer Z and reconstruct the original data. However, the methods operate mainly on image datasets, and the techniques that they use are complementary to ours. We have attempted to incorporate some of them in our method, and we show in our ablation study that combining them with our approach is as a fruitful direction for improving performance in UDA. In this section, we focus on related studies that use domain-specific information when learning latent representations, in some shape or form.

Relationship to "Domain Separation Networks"(Bousmalis et al. [12])

"Domain Separation Networks" (DSN) is a study which is in principle most similar to ours. It relies on a autoencoder framework, in which there are two separate private encoders, in addition to a shared encoder, which together constitute in two separate encoding functions for the source and target domains, in order to infer Z. In addition to this, the authors explicitly model a private-part of Z which can be thought of as analogous of our use of θ_X in our encoding and decoding functions, and the decoder is then a function of both the shared and private parts.

The main conceptual difference between this work and ours is that our work is motivated by the causal mechanism of generating X from θ_X and Z. This inspired us to try to measure the influence of θ_X in this generating process (via the mutual information) and restrict it, thereby ensuring that Z will not be a trivial and potentially useless representation for predicting Y. The method DSN achieves this as a by-product of using addition of the private and shared parts before providing them as input to the decoder which reconstructs the original features. This addition is a heuristic technique which is expected to specifically work with image datasets, whereas our mutual information minimization approach does not make such assumptions.

Relationship to ADDA (Tzeng et al. [126])

The method ADDA is a well known domain adaptation technique which uses two separate encoders ϕ_S and ϕ_T to learn the latent representation Z. However, as mentioned in the main text, this is an inefficient way to parameterize the two separate encoding functions. Furthermore, this study lacks a decoder, and therefore does not attempt to regularize the latent representation Z to have meaningful structure.

Relationship to "Continuously-Indexed Domain Adaptation" CIDA (Wang et al. [129])

The method CIDA, like our method, takes domain index as input to the encoder ϕ. However, in this method θ (the domain-specific information) is thought of a continuous random variable which is: **(1)**: of different value for each data-point, and **(2)**: has predictive information about the random variable Y. This reasoning assumes a different generating assumes a different generating process of the data than the one presented on Figure 4.1, in which Y and θ are conditionally independent given Z. By assuming that θ has predictive information about Y, the authors implicitly assume that θ and Y are dependent conditioned on Y in the data-generating process, and thus their algorithm is tailored to benefit this information in θ. On the other hand, our method only assumes that domain-specific information comes in the form of a discrete domain index unrelated to Y, and tries to make the best of that information to ensure the invariant representation Z has sufficient quality.

Chapter 5

Domain Adaptation via Direct Transformation

Most real-world data comes from different distributions, and many widely-used prediction tasks have to cope with this non-stationarity in the datasets. To do so, the main challenge is how to relate the various distributions the data comes from, and how to reason about what information is meaningful across different distributions. In this thesis, we have so far introduced two ways to do so:

(1) in Chapter 2, we postulated that the change across distributions is low-dimensional, and we have presented a way to extract it so we can reconstruct the joint distribution in the target domain;

(2) in Chapter 4 we assumed that the generating process of the features X from the label Y undergoes *minimal change*, as measured by the mutual information.

Despite the encouraging empirical results, these two approaches have significant drawbacks in principle, and expressing the minimality of change (or semantic similarity across distributions) in a way that is sensible and tractable remains a challenge in unsupervised domain adaptation and other related tasks. For example, image-to-image translation could also benefit from a more principled approach to represent the minimal changes across distributions, in order to ensure that meaningful semantic information in the image is not lost. In this chapter, we introduce a novel measure of change across domains which may have utility in several different cross-distributional tasks such as domain adaptation and image-to-image translation.

5.1 Introduction

In this study, we again assume the unsupervised domain adaptation setting, in which we are given labeled source and unlabeled target domain instances, represented by $(\mathbf{x}^{\mathcal{S}}, \mathbf{y}^{\mathcal{S}}) = (\mathbf{x}^{\mathcal{S}}_k, y^{\mathcal{S}})_{k=1}^{m_S}$ and $\mathbf{x}^{\mathcal{T}} = (\mathbf{x}^{\mathcal{T}})_{k=1}^{m_T}$ respectively, where $P^{\mathcal{S}}(X,Y) \neq P^{\mathcal{T}}(X,Y)$. Furthermore, we assume the conditional shift setting as presented in Figure 4.1, and was previously assumed in Chapter 4.

In domain adaptation, preservation of meaningful information is most commonly thought of as preserving the relationship between X and Y as much as possible. One way to measure this quantity is via the difference between $P^{\mathcal{S}}(X,Y)$ and $P^{\mathcal{T}}(X,Y)$. However, measuring this difference directly involves taking into account all features X regardless of their redundancy of relationship with Y, and therefore may not capture the minimal changes that are relevant for prediction. To mitigate this issue, in Chapter 2 we presented an approach that breaks down the joint distribution into two independently-changing modules, $P(X|Y)$ and $P(Y)$, and extracts the low-dimensional changes in them in a non-parametric way. However, a major drawback of this approach is that it operates on the original features, and does not use neural architectures, and therefore does not take advantage of the expressive power of neural networks. Furthermore, the dimensionality it can capture is restricted to the reproducing kernel function type of the kernel-PCA approach, and the number of labeled source domains available to perform kernel-PCA.

In the previous chapter of this thesis, we tried to mitigate these issues by designing a neural network architecture to explicitly model the changes of the joint distribution $P(X,Y)$ as a function of the domain index, and make use of them for regularizing the latent invariant representation Z. However, to do so, we relied on calculating the Jensen-Shannon Divergence between reconstruction and translation outputs of the decoder, which assumes that the source and the target domain distributions have substantial support overlap. However, in practice domain adaptation is frequently required and used in very high-dimensional data such as images or text. For example, in image datasets, a simple transformation such as increased brightness may affect the majority of the pixels in the image when going from source to the target domain, resulting in non-overlapping supports of their respective distributions. A similar case can be found in natural language data, where the target domain can make use of frequent vocabulary not present in the source and vice versa, which may happen if the two domains represent two different jargons.

Another way to reason about low-dimensional changes is by automatically discovering which subset of the features X have a dependence relationship with the target variable Y via a conditional-independence testing method (Zhang et al. [145]). This method yields a probabilistic graphical model of the Markov Blanket of Y, augmented with information regarding which of its conditional distributions change across domains. This information can then be provided to a conditional GAN which can be used to learn/represent the conditional distributions, and stochastic variatonal inference framework to perform inference of Y given the features X in the target domain (Zhang et al. [150]). The main drawback of this approach is that, similarly to the method presented in Chapter 2, it performs graph learning and inference on the observed features X, which may be high-dimensional, noisy and redundant.

Throughout this thesis, particularly in Chapter 4, we emphasized the popularity and promising performance of approaches that learn an invariant latent representation Z, which is then used for prediction in the target domain. Another way to represent the relationship between X and Y in this framework (and hopefully improve the quality of the learned representation Z) is to train a predictor on the source domain and apply it to the target domain to obtain pseudo-labels. These pseudo-labels allow for an initial approximation of the target domain joint distribution and can then be refined in various ways, and within several related deep learning frameworks based on invariant representation learning (Gu et al. [54], Long et al. [83], Xie et al. [135]). The hope is that a sufficiently well-approximated joint distribution in the target domain via pseudo-labels can be used to learn a good predictor in the target domain within the respective deep learning framework. However, this technique depends heavily on the quality of the initial pseudo-labels learnt from the source domain predictor, and thus it implicitly assumes that relationship between X and Y is preserved across domains, at least to some extent. Similarly to pseudo-labels, one can make use of an assumption that the data in both domains from the same label belongs to its own respective cluster. Thus, a specific penalty can be used to ensure that the learned predictor does not cross high-density regions in the target domain (Shu et al. [108]), and we used such a penalty in the algorithm presented in Chapter 4 as an additional regularizer.

5.1.1 Relating the Domains via a Direct Transformation

In addition to the aforementioned approaches to transfer information about predicting Y from X from the source to the target domain, yet another strategy is to assume that the two domains can be related through a direct transformation. Namely, one can postulate that there is a true function ϕ^*, such that $X^{\mathcal{T}} = \phi^*(X^{\mathcal{S}})$. Then, the problem reduces to finding the mapping ϕ^* such that $P^{\mathcal{S}}(\phi^*(X^{\mathcal{S}})) = P^{\mathcal{T}}(X)$. In general, a search for such a function from the source to the target domain data is an ill-posed problem because there may be many non-linear functions that satisfy this criterion.

Since in domain adaptation we rely on fundamental similarities between the joint distributions $P^{\mathcal{S}}(X, Y)$ and $P^{\mathcal{T}}(X, Y)$, the function ϕ^* is assumed to have specific properties to ensure that the relevant information for predicting Y is preserved after applying the function on source domain data. For simplicity, we in this study we assume that $P(Y)$ is invariant, and $P(X|Y)$ changes (i.e. *conditional shift*), and consequently, they are independently changing modules of the joint distribution. As discussed in Chapter 1, this implies that there is a dependence between the marginal distribution of $P(X)$ and the optimal predictor $P(Y|X)$, and that $P(X)$ may contain invaluable information about the joint distribution that can be exploited when searching for the optimal transformation ϕ^*.

In addition to the assumptions regarding the specific changes of the distribution $P(X, Y)$ across domains, one also needs to make some constraints about the function ϕ^* itself. One such assumption is that for each class c, the true ϕ^* transforms each class-conditional distribution $P(X|Y = c)$ as a location-scale transformation, as done in (Zhang et al. [148]). Under this assumption (and other mild assumptions), it has been shown that ϕ^* is identifiable asymptotically, and an optimization algorithm was presented to search for this function in a finite-sample setting.

One popular way to relax these strong linearity assumptions on ϕ^* in domain adaptation is Optimal Transport. Optimal Transport is a more general optimization problem which aims to map one probability measure μ_S to another, μ_T (with respective supports $\mathcal{X}^{\mathcal{S}}$ and $\mathcal{X}^{\mathcal{T}}$) at a minimum cost c, which is a function which can be defined in various ways. This problem was first introduced by French mathematician Gaspard Monge in the 18-th century, and revisited and refined by Kantorovich in the 20th century (Kantorovich [69]). It is defined as follows:

$$\begin{aligned} \phi^* = \arg\min_{\phi} \int_{\mathcal{X}} c(x, \phi(x)) d\mu_S(x), \\ \text{s.t. } \phi^* \# \mu_S = \mu_T, \end{aligned}$$

where $\phi^* \# \mu_S(x) = \mu_S(\phi^{*-1}(x))$, is the push-forward operator, and using it in the constraint of the problem ensures that the transformation ϕ^* transforms one probability measure to another. A convenient feature of this problem is that the supports $\mathcal{X}^{\mathcal{S}}$ and $\mathcal{X}^{\mathcal{T}}$ do not need to overlap. Optimal transport was first adapted to domain adaptation in (Courty et al. [24, 25]), and subsequently used within the framework of invariant representation learning and combined with pseudo-label techniques (Xu et al. [136]), yielding strong empirical results. Therefore, this direction of searching for a direct transformation ϕ^* from the source to the target domain may be a promising way to reason about the minimal change between the joint distributions that does not assume support overlap between the source and target domain marginal distributions.

While promising, optimal transport relies on a distance metric as a cost function in order to evaluate how complex the transport map ϕ^* may be. In this work, we introduce another measurement of how complex ϕ^* is, without relying on a choice of a cost function. The choice of a cost function may greatly affect the usefulness of the transport map for domain adaptation. Furthermore, since optimal transport is based on measuring distance, it is affected by the distance between $\mathcal{X}^{\mathcal{S}}$ and $\mathcal{X}^{\mathcal{T}}$, the supports of the source and the target domains. However, the two supports could be very far away in practice, and an example of this is when one domain represents images in day-time and the other domain represents images at night. In such cases, there is a risk that the optimal transport-based loss may be dominated by the distance between the supports rather than comparing the statistical properties of the source and target domain distributions. To see an example of problems arising in optimal transport, consider the simulated dataset on Figure 5.1. Here we have the source domain data, $X^{\mathcal{S}}$, which follows a mixture of Gaussian distributions, and the target domain data is a linear function of the source domain: $X^{\mathcal{T}} = \mathbf{A}X^{\mathcal{S}}$. On the figure, the optimal coupling yielded by solving the optimal transport problem (using the toolkit provided by (Flamary et al. [35])) is shown with blue lines, and one can appreciate that there are samples with opposite classes across domains being coupled with each other (some of them are shown in red circles). Therefore, the OT cost can be heavily influenced by affine transformations.

In this study, we assume that the two domains are related through a relatively simple transformation ϕ^* ($X^{\mathcal{T}} = \phi^*(X^{\mathcal{S}})$), and we try to learn this function via a novel criterion. Namely, in this chapter we argue that the variance of the log-determinant of the Jacobian of a given transformation ϕ, $\mathbb{V}[\log|\det \mathbf{J}_\phi|]$, is a suitable measure of how complex ϕ is in regards to its influence on the change of the distribution across domains. In the following section, we will show motivating examples demonstrating why this quantity is a good measure for preserving predictive information about Y when transforming the source domain data to the target domain. We will then present approaches to incorporate this criterion into existing domain-adaptation frameworks. Lastly, we will conclude this chapter with related work and empirical results.

5.2 Related Work

In addition to the many approaches mentioned in this thesis regarding learning domain-invariant representations, there is a lot of ongoing work tackling the problem of domain shift via direct transformations (translations). One subset of this body of work deals with unsupervised translation of text and images. For images, one of the widely used and adapted approaches is CycleGAN (Zhu et al. [158]), which imposes a so-called cycle consistency constraint, which is imposed by learning an inverse-map $\tilde{X}^{\mathcal{T}} = \psi(X^{\mathcal{T}})$ s.t. $P(\tilde{X}^{\mathcal{T}}) = P(X^{\mathcal{S}})$ and s.t. $\phi(\tilde{X}^{\mathcal{T}}) = X^{\mathcal{T}}$, thus ensuring that ϕ is a bijection and hoping that it preserves relevant semantic information after translating the image. However, under these contraints, ϕ and ψ can still be arbitrarily flexible and there is no guarantee that they will not discard semantic information (and thus information about Y if the aim is subsequent classification). For example, under the given constraints, when translating digit datasets, such as MNIST to MNIST-M, the model will successfully learn to capture the style change from one domain to another, but there is no guarantee that it will not flip one digit number to another as well. Our method provides a specific constraint on the translation function ϕ that aim to ensure that the main structural information in the distribution is preserved, and at the same time does not rely on CNNs or specific properties of image data.

Another sub-field of research that is related to this approach are generative models which use normalizing flows for a variety of tasks, such as more general variational inference (Rezende and Mohamed [97]), density estimation and generation (Dinh et al. [29], Gresele et al. [47], Kingma and Dhariwal [70], Papamakarios et al. [92]), and image-to-image translation and domain adaptation (Grover et al. [53]). As we shall we shall discuss in the following section, density estimation via a transformation involves the determinant of the Jacobian. The bulk of the literature of normalizing flows deals with the computation and differentiation of the log-determinant of the Jacobian matrix in deep neural networks. Namely, the task is to design neural network architectures such that the Jacobian matrix is ensured to be simpler, with a triangular or diagonal structure.

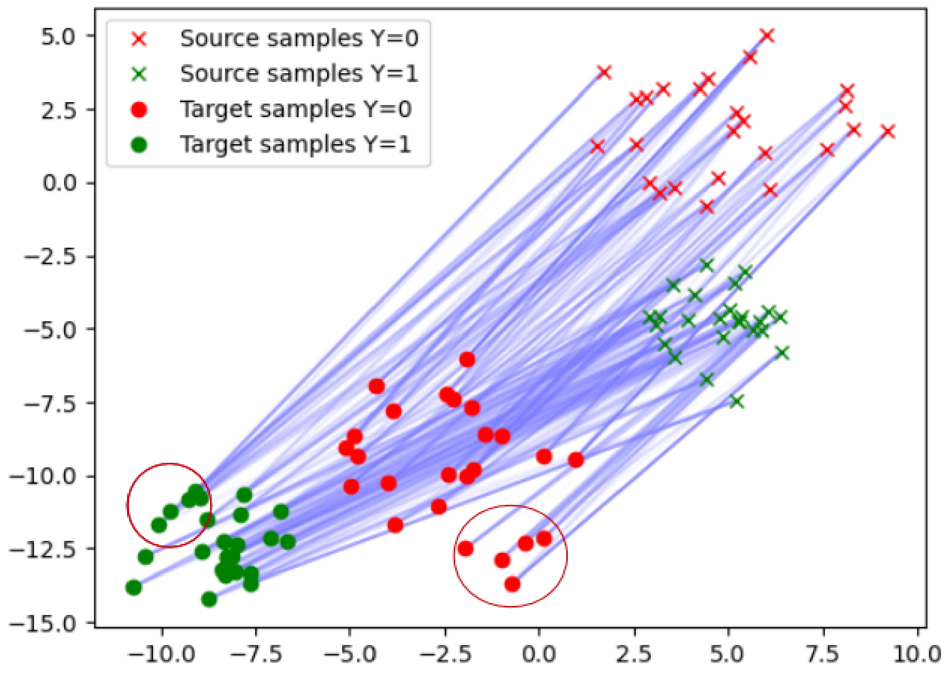


Figure 5.1: A simulated example showing the drawbacks of optimal transport, and the advantages of the Jacobian determinant.

source domain variables	$X^{\mathcal{S}}$	$Y^{\mathcal{S}}$
target domain variables	$X^{\mathcal{T}}$	$Y^{\mathcal{T}}$
i-th domain data point	$\mathbf{x}^{(i)}$	$y^{(i)}$
translation function	$\phi(X)$	
translated source domain data	$\tilde{X}^{\mathcal{S}}$	
class label classifier	h_c	
domain label classifier	h_α	
domain index	$j \in \mathcal{S}, \mathcal{T}$	
Jacobian matrix of ϕ evaluated at point i	$\mathbf{J}_\phi^{(i)}$	

Table 5.1: Notation used

5.3 Characterizing Structural Change when Applying Transformations

Before we examine how the determinant Jacobian matrix is related to preserving predictive information when performing transformations from the source to the target domain, we first describe its role in change of variables via functions. Suppose we have two 2d-dimensional vectors $[x_1, x_2]^T$ and $[y_1, y_2]^T$ related through a function g, $\mathbf{y} = g(\mathbf{x})$. When performing integration in calculus, it is often necessary to change one variable into another inside the integral. For example, if we consider integrating a function f with respect to y, the change of variables rule is as follows:

$$\int_{g(\mathbf{a})}^{g(\mathbf{b})} f(\mathbf{y}) d\mathbf{y} = \int_{\mathbf{a}}^{\mathbf{b}} f(g(\mathbf{x})) |\det \mathbf{J}_g| d\mathbf{y},$$

where $|\det \mathbf{J}_g|$ is the the absolute value of the determinant of the Jacobian matrix of g. Intuitively, this value represents the degree to which a very small rectangular region in the x_1-x_2 coordinate plane gets distorted by g.

An example of this process can be seen on Figure 5.2. Here, we define $g : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ as follows: $y_1 = x_1^2 - x_2^2$, $y_2 = 2x_1x_2$. In the figure, the blue curves in the middle and right-most panel correspond to $x_1^2 - x_2^2 = 3$ and $x_1^2 - x_2^2 = 4$, whereas the red curves correspond to $2x_1x_2 = 3$ and $2x_1x_2 = 4$. We see that by applying g, the rectangle in the x_1-x_2 plane on the left-most gets distorted to form the purple area in the middle panel. The right-most panel of the figure shows a parallelogram with a gray rectangle over the purple region. This rectangle is a linear approximation of the purple region that the Jacobian performs. The log-determinant of the Jacobian is the ratio of the areas of the gray linear approximation on the right-most plot and the purple rectangle in original space on the left-most plot and thus it approximates the level of distortion applied on this region by the function g.

5.3.1 Statistical Interpretation of the Jacobian Determinant

The same reasoning of change of variables naturally extends to random variables. Given two continuous random vectors $X^{\mathcal{S}}$ and $X^{\mathcal{T}}$, such that $Y = g(X)$, with cumulative distribution

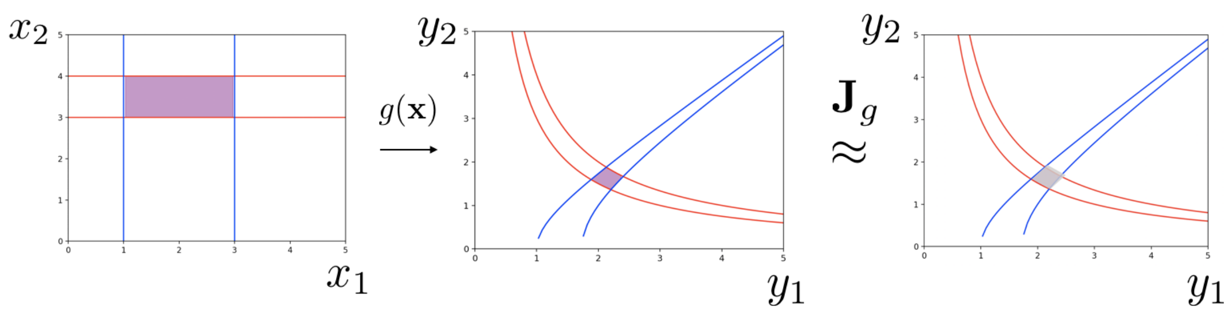


Figure 5.2: Example 2D transformation and visualization of the Jacobian and the effect of its determinant.

functions (CDFs) F_X and F_Y and probability density functions (PDFs) $p_X^{\mathcal{S}}$ and $p_X^{\mathcal{T}}$, the above change of variables principle can be applied to the CDF (Casella and Berger [15]), in order to derive $p_X^{\mathcal{T}}$ in terms of $p_X^{\mathcal{S}}$:

$$p_X^{\mathcal{T}}(\phi^*(X^{\mathcal{S}})) = p_X^{\mathcal{S}}(X^{\mathcal{S}})/|\det(\mathbf{J}_{\phi^*})| \implies |\det(\mathbf{J}_{\phi^*})| = \frac{p_X^{\mathcal{T}}(\phi^*(X^{\mathcal{S}}))}{p_X^{\mathcal{S}}(X^{\mathcal{S}})}.$$

From this expression we see that, the determinant of the Jacobian of transformations of random variables expresses distortion by the density ratio before and after applying the transformation. It is immediately apparent that this measure of distortion implied by the transformation ϕ does not require a distance metric, and is concerned with the change in density applied in each small region of $\mathcal{X}^{\mathcal{S}}$, the support of the source domain. It is worth mentioning that change of variables is heavily utilized in density estimation using generative models, as mentioned in the related work section. In that case, ϕ represents a map from original feature space to a simplified latent space which follows a simple parametric distribution, and the objective is to maximize its parametric likelihood. However, the calculation involves the log-determinant of the Jacobian of ϕ, so the key technical contributions consist of using normalizing flows to construct neural architectures the Jacobian matrix has simple diagonal or triangular structure, so that its determinant can be easily calculated and differentiated.

Given this, it would be useful to summarize the information conveyed by the determinant of the Jacobian matrix across the entire source dataset. Therefore, one may consider the expected value as a suitable criterion:

$$\mathbb{E}[\log|\det(\mathbf{J}_{\phi^*})|] = H(\phi^*(X^{\mathcal{S}})) - H(X^{\mathcal{S}}),$$

where H is the entropy. However, this quantity is very sensitive to simple changes in scale that may affect the entire dataset, because the overall level of density across the entire target domain support $\mathcal{X}^{\mathcal{T}}$ can change drastically, whereas the shapes and the other characteristics of the source and target domain distributions, $P(X)^{\mathcal{S}}$ and $P(X)^{\mathcal{T}}$ may be similar. To ensure that the criterion is agnostic to such changes in scale, we opt to use the variance of the log-determinant of the Jacobian, $\mathcal{L}_{\mathbf{J}_\phi} := \mathbb{V}[\log|\det \mathbf{J}_\phi|]$. This value measures the overall degree to which there are *local* changes in density over the entire support of of the source domain $\mathcal{X}^{\mathcal{S}}$, regardless of global differences in scale.

To see how this measure can be effective, consider again the example on Figure 5.1, and recall that the relationship between the source and the target domain, ϕ^* is linear: $X^{\mathcal{T}} = \mathbf{A}X^{\mathcal{S}}$. Then, each entry in the Jacobian of this transformation, $\mathbf{J}_\phi$ is a constant, and so is the log-determinant of its Jacobian. Therefore, in this case, $\mathcal{L}_{\mathbf{J}} = 0$, and the search for a non-parametric flexible function ϕ^* under the constraint that $P^{\mathcal{T}}(X) = P^{\mathcal{S}}(\phi(X))$. Thus, we can formulate a simple optimization problem that searches for an optimal translator as follows:

$$\begin{aligned} \phi^* = \arg\min_{\phi} \mathcal{L}_{\mathbf{J}_\phi} &:= \mathbb{V}[\log|\det \mathbf{J}_\phi|] \\ \text{s.t. } P^{\mathcal{T}}(X) &= P^{\mathcal{S}}(\phi(X)). \end{aligned}$$

After solving this problem and finding ϕ^*, one can use this to transfer the source domain data to a translation $\tilde{X}^{\mathcal{S}}_{trans.}$ in the target domain, and pair it up with source domain labels $Y^{\mathcal{S}}$ to train a classifier which should be optimal for prediction for the unlabeled target domain data. In practice, one can also make use of the source domain labels to facilitate the search for the optimal translator. In the following section, we shall present two ways to make use of this criterion in a deep learning framework.

5.4 Method

Given the new criterion for minimal change introduced in the previous section, we now aim to design a deep architecture to make use of it. The first approach for doing so is via training a direct translation function ϕ from the source to the target domain, instead of relying on invariant representation learning. We present the proposed method on Figure 5.3. The model consists of four main components which are trained jointly:

- The first component maps the source domain data using a transformation $\phi : \mathbb{R}^d \to \mathbb{R}^d$ and yields a translation $\tilde{X}^{\mathcal{S}} = \phi(X^S)$.
- The second component learns to predict class label $\tilde{Y}^{\mathcal{S}}$ from the transformed source domain data $\tilde{X}^{\mathcal{S}}$ via a class predictor: $\tilde{Y}^{\mathcal{S}} = h_c(\tilde{X}^{\mathcal{S}})$.
- The third component ensures that the transformed data matches the target domain data in marginal distribution $P^S(\phi(X)) = P^{\mathcal{T}}(X)$, and is usually modeled by an adversarial domain label classifier: $\tilde{j} = h_\alpha(\tilde{X}^{\mathcal{S}})$ and $\tilde{j} = h_\alpha(X^{\mathcal{T}})$.
- The fourth component minimizes the level of distortion via the newly introduced criterion: $\mathcal{L}_{\mathbf{J}_\phi} = \mathbb{V}[\log|\det \mathbf{J}_\phi|]$.

To enforce the above-mentioned constraints, we need the following loss functions:

$$\begin{aligned} \mathcal{L}_{classif.} &= -\sum_{j=1}^{C}\sum_{i=1}^{m_S} y_{ij}\log(\tilde{y}^c_{ij}), \\ \mathcal{L}_{inv.} &= -\sum_{i=1}^{m_S+m_T}\{j_i\log(\tilde{j}_i) + (1-j_i)\log(1-\tilde{j}_i)\}, \end{aligned}$$

and the total loss can be minimized in an alternate fashion:

$$\min_{\phi} \lambda_c \mathcal{L}_{classif.} + \lambda_j \mathcal{L}_{\mathbf{J}_\phi} - \lambda_h \mathcal{L}_{inv.}$$
$$\min_{h_\alpha} \lambda_h \mathcal{L}_{inv.}.$$

Here, after training, we can obtain the predicted labels $\tilde{Y}^{\mathcal{T}} = h_c(X^{\mathcal{T}})$, and if needed one can also train a classifier from scratch on the translated features $\tilde{X}^S$ and source domain labels $Y^{\mathcal{S}}$.

5.4.1 Alternative Implementation via Invariant Representation Learning

In addition to implementing the search for ϕ^* by directly mapping the source domain to the target domain, the same constraint for minimum distortion via the variance of the Jacobian matrix can be implemented within the framework of invariant latent representation learning. Let us assume that the data-generating process is as in Chapter 4, depicted in Figure 4.1. Motivated by this generating process assumption, in Chapter 4 we introduced an autoencoder framework (depicted on Figure 4.3) in which we map the input data X (along with appropriate domain-specific information) to an invariant representation Z via an encoder ϕ. Subsequently, the invariant features Z (also together with appropriate domain-specific information) are inputs to a decoder function $\tilde{\phi}(Z, \theta_X)$. As described in Chapter 4, depending on the domain-specific input θ_X, the decoder may either produce a reconstruction $\tilde{X}^{\mathcal{S}} = \tilde{\phi}(Z^{\mathcal{S}}, \theta_X^{\mathcal{S}})$, or a translation $\tilde{X}^{\mathcal{S}}_{trans.} = \tilde{\phi}(Z^{\mathcal{S}}, \theta_X^{\mathcal{T}})$. This presents an overall translation function $\psi(X^{\mathcal{S}}) = \tilde{\phi}(\phi(X^{\mathcal{S}}, \theta_X^{\mathcal{S}}), \theta_X^{\mathcal{T}})$, whose influence on the distribution change can be regularized via the constraints on the variance of the log-determinant of the Jacobian, given by $\mathcal{L}_{\mathbf{J}_\psi}$. The only change to the algorithm in Chapter 4 is providing $\mathcal{L}_{\mathbf{J}_\psi}$ to its total loss instead of the cross-entropy loss $\mathcal{L}^{trans}_{inv.}$ of the auxiliary classifier h_c.

By applying this kind of change to the algorithm in Chapter 4, we are effectively using a different criterion for a minimal change of the generating process of X from Y across domains. Namely, instead of mutual information, we are now using a measurement of minimal distortion of the marginal distribution. As mentioned before, this criterion has an advantage over mutual information, in that it does not require there to be overlap between the supports $\mathcal{X}^{\mathcal{S}}$ and $\mathcal{X}^{\mathcal{T}}$.

5.4.2 Challenges with Scalability of the Jacobian Determinant

The main drawback of the proposed framework is the computational cost of computing the Jacobian during the forward pass. The Jacobian involves derivatives of ϕ with respect to the input features X, and we perform their evaluation using automatic differentiation. It can be shown (Baydin et al. [3], Gresele et al. [47]) that for a D-dimensional dataset and a fully-connected neural network consisting of L layers, the cost of computing the Jacobian is in the order of $\mathcal{O}(LD^3)$. Therefore, this computation may be prohibitively expensive for high-dimensional datasets such as images, so in the following section, we restrict our empirical evaluation to relatively low-dimensional datasets.

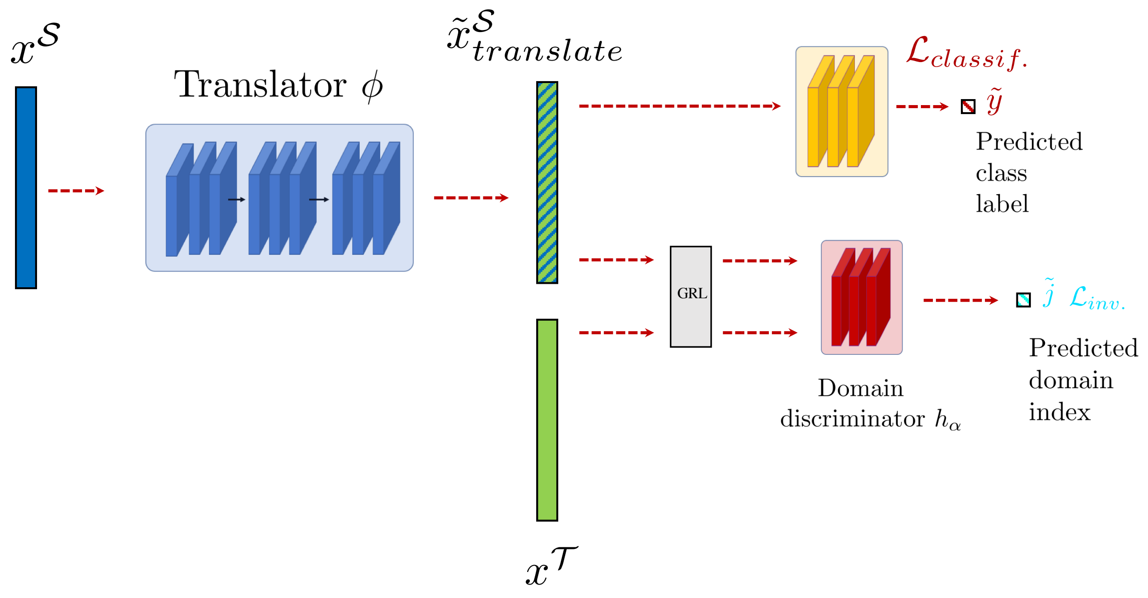


Figure 5.3: Diagram of the proposed model.

5.5 Empirical Evaluation

To evaluate the efficacy of the proposed method, we conduct experiments on a synthetic and real dataset. We consider the following baseline methods in our experiments:

- no-transfer: a simple classifier only trained on the source domain
- DANN (Ganin et al. [39]): a classic invariant training method which learns a marginally invariant representation that is predictive of Y in the source domain,
- DSN (Bousmalis et al. [12]): an invariant training method that explicitly models the invariant and the changing parts of the data in its latent representation, and regularizes them via a reconstruction loss,
- DSAN: our recently-proposed method which for learning invariant latent representations, which uses mutual information minimization to enforce minimal change on the encoding function,
- DSAN: optimal transport mapping (Flamary et al. [35]) of the source domain data to the target domain and subsequent classification done by SVM,
- TRANS-U: unregularized translation method without constraining the variance of the Jacobian matrix of the translation function ϕ.

5.5.1 Experiments on Simulated Data

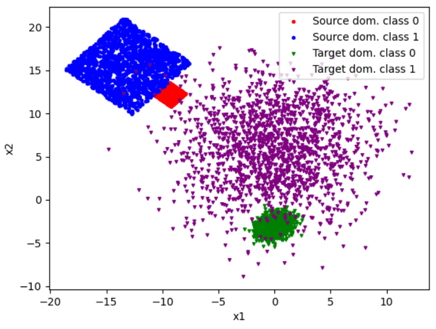
To evaluate our method, we created a binary classification dataset in which there is minimal change of $P(X|Y)$ across domains and there is little to no support overlap across the two domains. The dataset is shown on Figure 5.4, in which the source and target domains follow mixtures of uniform and Gaussian distributions respectively, ensuring that the true transformation function ϕ^* is nonlinear and requires a neural network to learn. For this experiment, we implemented the algorithm as a direct translation as opposed to an invariant representation learning approach.

We implemented each of the components of our proposed model using a multi-layer perceptron, and we sampled 3200 points from each domain in 10 separate experiments using different random initializations. We present the results on Table 5.4c, where we provide the accuracy and the standard deviation across the different replicate experiments (shown in parentheses). From the results, one can appreciate that our proposed method significantly outperforms the baselines by a significant margin. Furthermore, it is apparent that the translation-based baseline TRANS-U does significantly better than the other baselines. We postulate that this is due to the fact that in this example, a well-initialized MLP can still perform translation and preserve meaningful information of the target variable Y in its output for certain random initializations of the network. However, baseline methodologies that perform invariant representation learning are unlikely to maintain knowledge of this information in the latent representation, since they perform transformations on both the source and the target domain data. On Figure 5.1, we see the source domain data after mapping them to the target domain using optimal transport, colored by their respective source domain labels. From this figure, it is apparent that the optimal transport approach does not successfully capture the properties of the joint distribution in the target domain, and discards meaningful information about the target variable Y.

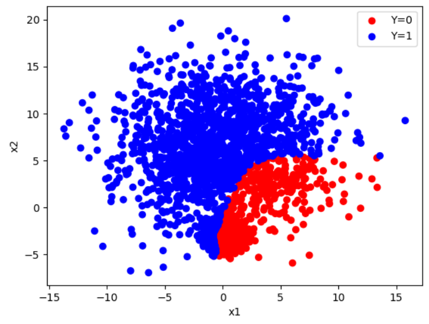
To see the effects of the Jacobian criterion, on Figure 5.5 we show the source domain data mapped to the target domain when we use the constraint on the variance of the Jacobian determinant (TRANS) versus when we do not (TRANS-U). Here, the top row shows the translation and the performance on the target domain data, when the constraint on the variance on the log-determinant of the Jacobian matrix is not imposed: (a) the target domain data with the true labels, (b) the output $\tilde{X}^{\mathcal{S}} = \phi(X^{\mathcal{S}})$ when run without the constraint on the variance of the Jacobian, colored by the true source domain labels (c) the target domain data $X^{\mathcal{T}}$, colored by the the predictions $\tilde{Y}^{\mathcal{T}}$ after training. Top row shows the translation and the performance on the target domain data, when the constraint on the variance on the log-determinant of the Jacobian matrix is enabled: (d) the translation run with the constraint on the variance of the Jacobian, (e) the target domain data colored by the prediction after training. From these visualizations, one can appreciate that the proposed approach is indeed necessary for preserving label-specific information and successful prediction in the target domain.

5.5.2 Experiments on Real Data

To further examine the efficacy of the proposed approach, we also perform experiments on real-world datasets. The first dataset we consider the 67-dimensional wireless localization dataset used in Chapter 4. To ease the process of translation and to reduce dimensionality for the purpose of calculating the gradient of $\mathcal{L}_{\mathbf{J}}$, we use the conditional independence-based graph discovery



(a)



(b)

Model	Accuracy
no-transfer	50.4(4.3)
DANN	51.1(4.3)
DSN	52.3(5.2)
OT	82.65(1.1)
TRANS-U	84.1(15.1)
TRANS	**93.3(1.3)**

(c)

Figure 5.4: Scatter-plot of the simulated dataset and performance of the proposed method compared to baselines (percent accuracy).

Table 5.2: Accuracy (%) on Wi-Fi localization dataset for unsupervised domain adaptation.

Models	t1 → t2	t1 → t3	t2 → t1	t2 → t3	t3 → t1	t3 → t2	Average
CDAN (Long et al. [83])	37.95(2.3)	30.31(2.2)	35.79 (1.9)	34.21(1.2)	31.03(0.8)	32.44(1.1)	33.62
MSTN (Xie et al. [135])	32.32(1.2)	26.56(1.8)	25.68 (2.2)	27.66(2.3)	29.3(1.3)	28.66(1.2)	28.36
MDD (Zhang et al. [154])	29.33(2.2)	31.61(2.9)	28.905(3.1)	27.62(2.3)	29.57(3.0)	26.31(3.3)	28.89
BSP (Shu et al. [107])	37.8(2.2)	32.07(1.8)	35.76(2.8)	34.23(1.5)	31.49 (1.2)	31.63(1.1)	33.83
DSAN-U	44.33(2.9)	33.63(4.2)	37.10(1.8)	34.77(2.1)	36.77(2.6)	34.55(3.5)	36.96
DSAN-Jac	*44.50(2.5)*	**36.3(4.3)**	*37.75(2.8)*	**36.7(4.0)**	**37.28(1.7)**	**35.55(2.8)**	**37.89**

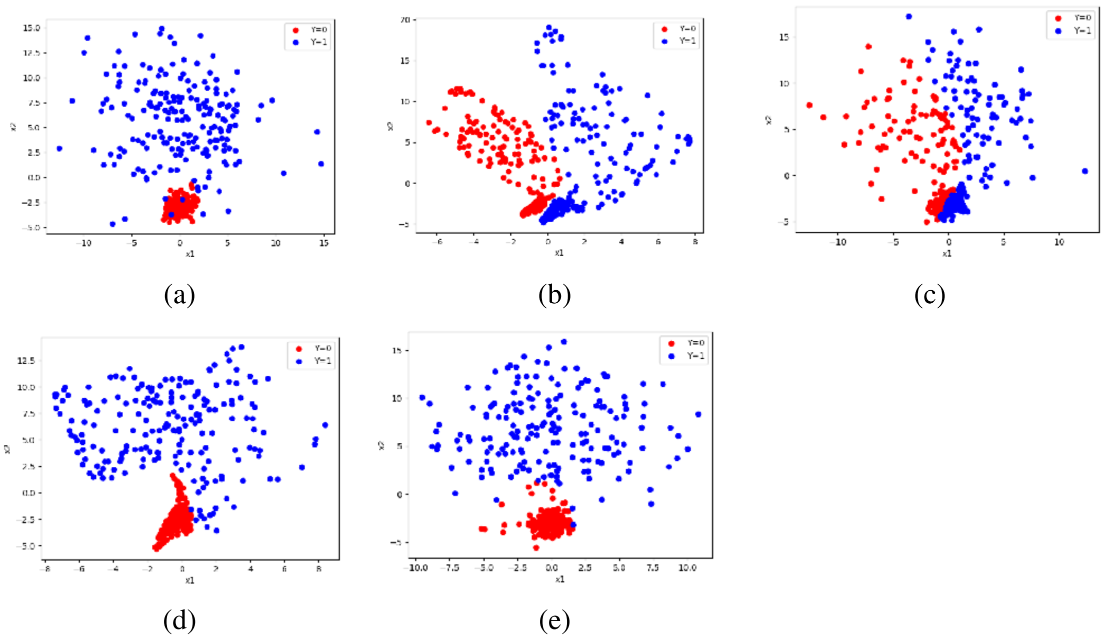


Figure 5.5: The translated data $\tilde{X}^S$ in the simulated experiments at the end of the run.

approach by (Zhang et al. [145]) to reduce dimensionality of the dataset to the 16 features which are in the Markov Blanket of Y. We used the following recently-proposed baselines: MDD (Zhang et al. [154]), BSP (Shu et al. [107]), as well as the established pseudo-label based methods: CDAN (Long et al. [83]) and MSTN (Xie et al. [135]). In addition to these baselines, we also include the baseline of an autoencoder approach as described in Chapter 4, without any regularization (we refer to it as DSAN-U). For this experiment, we implemented the proposed method as a regularizer of an invariant representation learning approach (as described as an alternative implementation to a direct translation approach, in subsection 5.3.1 of this chapter). Since in this setting, the baseline methods and our proposed method use an encoding function ϕ, we used the same MLP architecture for each baseline as well as the proposed method (which we refer to here as DSAN-Jac). When comparing to DSAN-U, we ensure that both the encoder and decoder have the same architecture.

The results of the wireless localization dataset experiment are presented in Table 5.2. From these results, one can appreciate that the proposed method marginally outperforms the baselines. While the results show an encouraging trend when using our approach, they are not statistically significant. We found that the most likely reason for this is that the proposed method is challenging to tune – in particular the penalty parameter that corresponds to the $\mathcal{L}_{\mathbf{J}}$ (given by λ_j). Given the significant results on simulated data, we are confident that the proposed method can be tuned to outperform the baselines more convincingly.

5.5.3 Challenges in High-Dimensional Settings

The criterion introduced in this study relies on computing the Jacobian matrix of the translation function. When the translation function is parameterized by a neural network architecture, the Jacobian matrix can be calculated by backpropagation. However, when there are many dimensions and limited number of observations, the euclidean distance between each pair of points may be

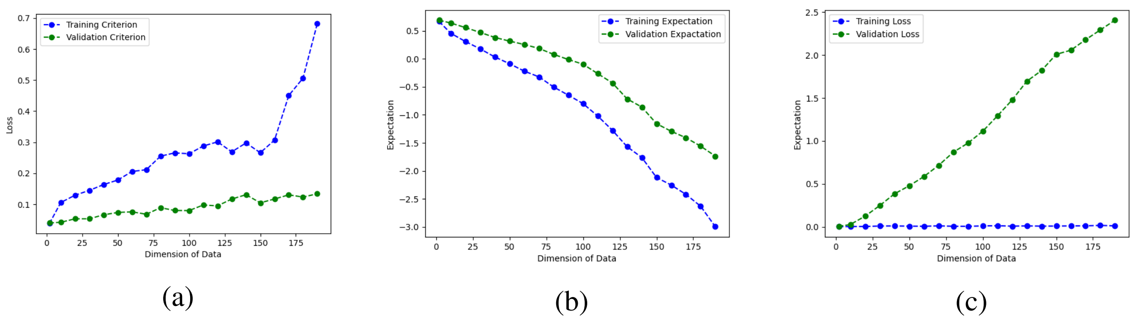


Figure 5.6: Analysis of the effect of dimensionality on the variance and expectation of the log-determinant of the Jacobian of the transformation in a simple simulated setting.

very large, and the estimated translation function can be arbitrarily flexible between them, thus overfitting on the training data. Therefore, the variance of the Jacobian may suffer from estimation error when calculated this way.

To demonstrate this phenomenon, we conducted a simple simulation experiment. Namely, we simulated a dataset consisting of 100 data-points from the standard normal distribution $X^{\mathcal{S}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}^d)$, where we vary the dimensionality of the data given by d. We then apply a simple scale transformation $X^{\mathcal{T}} = 2X^{\mathcal{S}}$, thus generating paired source and target domain data. We learned this transformation via a deep fully-connected feed-forward network and a mean-squared error loss between the paired instances. In Figure 5.6, we present the effect of the dimensionality of the data on the moments of the Jacobian log-determinant. To generate simulated data, we sampled 200 points from the above model, and we vary the dimensionality between 5 and 195.

On the right panel in Figure 5.6c we observe that the mean squared error on the training dataset is low and that the loss evaluated on a held out validation dataset is increasing, indicating that there is gradually increasing overfitting on the training data with increasing dimensionality. On the middle panel (Figure 5.6b), we present the expectation of the log-determinant of the Jacobian of the learnt transformation, and we see that it drops rapidly when evaluated on the training data, indicating that the average determinant (without the log transformation) approaches zero, and the function is not invertible. Furthermore, we can also observe that this phenomenon begins to occur even before significant overfitting takes place, when the dimensionality ranges from 10 to 30. Consequently, due to the logarithm transformation on the determinant in our objective, the increase in scale of the expectation in the middle panel is reflected on an increase in variance of the log-determinant in the left panel (Figure 5.6a), which is the criterion we use in our approach. Therefore, we can conclude that with limited number of data-points, using a very flexible model to learn the transformation may result in an unreliable measurement of distortion with imposed by the transformation as calculated by our proposed criterion. In conclusion, further investigation is required to adapt the proposed criterion for use in high-dimensional settings.

5.6 Conclusion

In this chapter we introduced a novel criterion for change implied on the distribution of the features by a transformation from the source to the target domain. We have performed simulated and real-world dataset experiments to examine the effectiveness of this criterion, and to show that it has promising potential when added to existing domain adaptation frameworks. One direction of future work would be to combine this criterion with methods that rely on pseudo-labels, and to see whether it improves the quality the of the pseudo-labels and thus the prediction performance in the target domain. Furthermore, as currently implemented the current approach performs back-propagation through a log-determinant operation, which is computationally prohibitive for any dataset beyond a few dimensions. Therefore, combining this criterion with the machinery and computational efficiency of normalizing flows would be instrumental in incorporating it in high-dimensional datasets such as text and images, and for using to facilitate applications such as image-to-image translation.

Chapter 6

Conclusion and Discussions

In this thesis, we have approached the problem of unsupervised domain adaptation from various and often orthogonal viewpoints. However, all of the studies presented in this thesis are motivated and follow one underlying line of reasoning: that considering the data-generating process can lead to a compact representation of the minimal changes across different domains, which can then be adapted to various learning techniques and algorithms. In each of the studies our assumption is that the joint distribution $P(X, Y)$ can be broken down into two independently-changing modules.

In Chapters 2, 4, and 5 we assumed that these modules are $P(Y)$ and $P(X|Y)$, which can result from two scenarios: **(1)** the true underlying causal process is $Y \rightarrow X$ and there are no hidden confounders or selection bias in the observed data, or **(2)** the two independent modules in the joint distribution can result from a hidden common cause L in the true causal process, as demonstrated in example 2 of Figure 1.3. On the other hand, in Chapter 3 we assumed that the two independently-changing modules of the joint distribution are $P(X)$ are $P(Y|X)$. This can arise due to: **(1)** the true underlying causal process is $X \rightarrow Y$ and there are no hidden confounders or selection bias in the observed data or **(2)** the true underlying causal process is $Y \rightarrow X$ and there is selection bias on X which varies across domains, as demonstrated in example 1 of Figure 1.3.

Taking into consideration the data-generating process and coming up with a compact representation of the joint distribution was invaluable in deciding how to represent the changing parts of the distribution in the learning procedure, and make use of them for adaptation. The manner in which we achieved this varied depending on the generating process assumptions and the specific properties of the machine learning techniques we used. In the context of this overarching theme, we shall now briefly revisit the various approaches used in this thesis and point out some of their main drawbacks.

In Chapter 2, we worked in a classification setting under the *conditional-target shift*, meaning that both $P(Y)$ and $P(X|Y)$ change across domains, and they do so independently. In this study, we assumed that there are multiple labeled source domains and a single unlabeled target domain. We designed an algorithm that makes use of reproducing kernel techniques such as Kernel PCA (Schölkopf et al. [103]) and Maximum Mean Discrepancy (MMD) (Smola et al. [109]) which accomplishes two tasks simultaneously: **(1)** we extracted the low-dimensional manifold of changing parameters of $P(X|Y)$ in the source domains, and **(2)** we made use of this manifold to successfully reconstruct the joint distribution in the target domain, and we showed that the joint distribution in the target domain is identifiable in the asymptotic case. Despite the promising performance, there are some drawbacks of the proposed approach. Firstly, the approach is heavily dependent on the number of source domains in the datasets. If this number is smaller than the number of effective changing parameters of $P(X|Y)$, then it will not be possible to capture all of the changes of the proposed approach in the low-dimensional manifold learned via Kernel PCA. Secondly, the approach operates on the raw features of the data, and therefore may suffer from estimation error when dealing with high-dimensional and noisy datasets such as images or text. Finally, we applied this algorithm on all of the features of the data,

In Chapter 3, we assumed that the two independently-changing modules of the joint distribution across domains are $P(X)$ and $P(Y|X)$. Specifically, we worked in the setting of covariate shift, in which $P(X)$ changes, but $P(Y|X)$ stays the same (it undergoes a trivial identity change). We demonstrated that correcting for covariate shift can have deteriorating performance with increasing number of dimensions in the data, and we introduced an approach to reduce the dimensionality while ensuring that relevant information for predicting Y is not discarded. In particular, we made use of reproducing kernel techniques in order to express the dependence between the target variable Y and a linear projection of the features X, given by $\mathbf{W}^T X$. Furthermore, we established an algorithm that minimizes this dependence by searching over the manifold of linear projections. The main drawback of this approach is that the relationship between X and Y may be non-linear, and in this case our approach will suffer from bias, as demonstrated by the main theoretical results of this study.

In Chapter 4, we again worked with the independently-changing modules $P(Y)$ and $P(X|Y)$, but we assumed that $P(Y)$ undergoes the trivial identity transformation (i.e. is invariant), and we worked within the realm of invariant representation learning methods using neural architectures. In this study, we further assumed that the generating process is known and there is no selection bias and/or unobserved confounders (as shown in Figure 4.1). This assumption allowed us to interpret the invariant representation being learned in the network as corresponding to a latent variable Z in the generating process, and we were able to dissect what information is necessary to infer this variable from observed data. This view of the problem allowed us to accomplish two main tasks: **(1)** infer the latent representation from X and domain-specific information θ_X via an encoder, and **(2)** mimic the data-generating process of X from Z and θ_X via a decoder, thus ensuring that Z has non-trivial structure which preserves information for predicting Y in the target domain. To ensure this, a key ingredient to our method was a way to represent the notion of minimal change in the generating process of X from Y (given by $P(X|Y)$) across domains. Namely, we expressed this dependence in terms of the mutual information of $I((X, Y); \theta_X)$, and we demonstrated a convenient way to implement it within the invariant representation learning framework. The main drawback of this approach is that this constraint for minimal change across domains comes with the assumption that the source and target domain distribution have a significant amount of support overlap. This is a problem when working with data which is high-dimensional, as it is likely that there are dimensions which have values that are different by orders of magnitude between the two domains.

In Chapter 5, we attempt to mitigate this issue by viewing the relationship between the source and the target domain in terms of a direct transformation from the source to the target. We make the same generating process assumptions as in Chapter 4, and we represent the notion of minimal change across domains in terms of the variance of the Jacobian determinant of this transformation, which measures the degree of local changes of the density $p(X)$ imposed by the transformation. Since in this setting, $P(X)$ has information about $P(Y|X)$, our conjecture was that this measure can be used as a proxy to assess to which degree label-specific information is being lost when applying the direct transformation. The main disadvantage of this approach is the poor scalability to high-dimensions, due to the high computational cost of computing and differentiating the determinant of the Jacobian.

6.1 Future Work

In this thesis, we have made contributions to several sub-branches of unsupervised domain adaptation, and provided new strategies and insight through which this challenging problem can be tackled. For each of them, there are several natural directions to go further and improve on them. We shall now provide a survey of some of these future directions.

6.1.1 Refining the Search for a Low-Dimensional Manifold

In Chapter 2, we presented a novel approach of capturing low-dimensional changes across domains via kernel methods. This approach relied on embedding M source domain distributions $P^{(1)}(X|Y), .., .P^{(M)}(X|Y)$ into Hilbert space, and then performing PCA and projecting (representing) these distributions as points a hyperplane in this infinite-dimensional space. However, we do not impose any constraints regarding *where* on this hyperplane these points may lie. In practice, it could be that these distributions (points) may be close to each other on the hyper-plane. In fact, it has been suggested that it is reasonable to assume that points mapped to an RKHS using the Gaussian kernel follow approximately a Gaussian distribution when projected on a random element in the RKHS (Huang et al. [63]). This property can be used to establish an additional Gaussian likelihood-based constraint on the distributions upon mapping them into Hilbert space and projecting them on the hyperplane in that space.

Furthermore, we previously mentioned that the number of labeled source domains is instrumental for the effectiveness in capturing more complex changes of $P(X|Y)$ across domains. When the total number of domains available is small, it may be beneficial to make use of the target domain when learning the low-dimensional manifold of changes. Since $P(X|Y)$ is not observed in the target domain, a pseudolabel-based procedure may be used to assign initial labels, and refine them iteratively by re-learning the manifold and performing prediction on the target domain.

Finally, as discussed throughout this thesis, using all of the observed features for prediction may be sub-optimal because a large fraction of them may be irrelevant for predicting Y, and taking all of them into account may introduce more complex changes for the algorithm to capture via Kernel-PCA. Thus, the algorithm may require more labeled source domains for reliable transfer learning. To prevent this from happening, one could discover a compact representation of the changes of $P(X, Y)$ across domains using conditional-independence testing to discover an augmented graphical model of the data, as presented in Figure 1.4 (Zhang et al. [145]). This representation will yield independently-changing modules of the distribution, and a low-dimensional manifold of changes may be learnt for each of them. Then, the task would be to combine this information for prediction in the target domain.

6.1.2 Future Directions for Neural Network-Based Domain Adaptation

In Chapters 4 and 5 of this thesis we dealt with building on existing deep learning frameworks for unsupervised domain adaptation. In particular, we dealt with ways to represent the property of minimal change of $P(X|Y)$ across domains and incorporate it in a deep learning architecture. In Chapter 5 we introduced a way to measure the change of $P(X|Y)$ across domains by only using the marginal distributions of X in the source and the target domain. The main building blocks of this approach was to assume that there is a simple direct transformation from the source to the target domain, and to try to approximating via a constraint on its Jacobian determinant. However, as discussed before, there are scalability issues with this criterion, rendering it unviable for high-dimensional datasets without previously compressing them to a low-dimensional space, and therefore this criterion cannot be explored for applications such as image-to-image translation at the moment.

Furthermore, when working with a limited number of points in a very high-dimensional space, calculating the Jacobian (and its determinant) of the transformation by back-propagation may not be an accurate representation of the smoothness of the transformation. Therefore, efficient nearest-neighbor approaches to approximate the Jacobian matrix via finite differences in this scenario is a crucial next step for making this method directly applicable to high-dimensional datasets. In addition to this future direction, another next step towards deploying this method in a high-dimensional setting is to ensure that the determinant of the Jacobian can be calculated and differentiated computationally efficiently. Therefore, a fruitful direction to investigate would be how to incorporate a normalizing flow into the neural architecture corresponding to the transformation in the model, in order to ensure that the determinant of the Jacobian can be calculated easily.

In the introduction of this thesis we mentioned that semi-supervised learning approaches have recently been explored in the unsupervised domain-adaptation setting and have shown very strong empirical performance on several benchmark datasets. A large subset of these techniques rely on making use of large unlabeled data in order to guide a predictor towards regions of low density. Examples of these techniques are self-training (Lee et al. [72]) and cross-entropy minimization (Grandvalet et al. [46]), which have been successfully incorporated into unsupervised domain adaptation (Saito et al. [101], Shu et al. [108], Zhang et al. [153]). When used in unsupervised domain adaptation, the semi-supervised learning techniques are applied on the unlabeled target domain data in order to inform the predictor which is trained on the source-domain labeled data. When using self-training, the general approach is to train a predictor on the labeled source domain data, and then apply it on the target domain data to generate pseudo-labels which are then iteratively refined. However, the initial pseudo-labels obtained this way may be very noisy due to the different joint distributions between the source and target domains, and this may hurt the overall prediction performance in the target domain. Therefore, a fruitful future direction would be to explore how the constraint on the variance of the Jacobian determinant can be used to reduce the noise of the pseudo-labels by determining the distortion that they impose on $P(X|Y)$ (instead of simply using the constraint on $P(X)$ like we did in Chapter 5).

In addition to designing the criterion for minimal change (such as mutual information or variance of the Jacobian determinant), another promising direction to investigate is the degree to which novel neural network architectures such as transformers (Vaswani et al. [128]), can be used to apply these ideas more efficiently on structured data such as text and images. In particular, the attention mechanisms and their respective hidden representations may have valuable information about the change of the distribution across domains, and would be worthwhile to explore how to harness them in a domain adaptation framework.

Finally, the ultimate goal of our line of thinking is to work towards the most general neural network framework for domain adaptation which can provably capture the changes of the joint distribution across domains, given enough labeled source domains. In Chapter 2, we introduced a method to capture the changes of the conditional distribution $P(X|Y)$ across domains via Reproducing Kernel techniques. Similarly, (Zhang et al. [150]), generative models were used to capture the changes of specific conditional distributions relevant for predicting Y. However, both of these lines of work only capture limited factors of change across domains. Ultimately, the goal would be to design generative models that can capture various latent factors of variation of the joint distribution across domains and explore what would be needed for their identifiability. This line of reasoning is related to learning latent variable graphical models, and one of the approaches developed for this task is spectral learning (Hsu et al. [60], Song et al. [111]). Therefore, a promising direction to study would be making use of these established approaches to learn a latent variable model under simpler assumptions, after which one can move to flexible neural-network based generative models with various disentanglement properties.

Bibliography

[1] Elias Bareinboim and Judea Pearl. Controlling selection bias in causal inference. In *Artificial Intelligence and Statistics*, pages 100–108. PMLR, 2012. 1.1.1

[2] Elias Bareinboim, Jin Tian, and Judea Pearl. Recovering from selection bias in causal and statistical inference. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. 1.1.1

[3] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of machine learning research*, 18, 2018. 5.4.2

[4] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Proceedings of NIPS 2006*, 2007. 1.2.4, 1, 4.1

[5] S. Ben-David, S. Shalev-Shwartz, and Ruth Urner. Domain adaptation–can quantity compensate for quality? In *ISAIM 2012*, 2012. 4.1

[6] Shai Ben-David and Ruth Urner. Domain adaptation–can quantity compensate for quality? *Annals of Mathematics and Artificial Intelligence*, 70(3):185–202, 2014. 2.1.1

[7] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007. 3.1.1

[8] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010. 2.1.1, 3.1.1

[9] Gérard Biau, Benoît Cadre, MAXIME Sangnier, and Ugo Tanielian. Some theoretical properties of gans. *arXiv preprint arXiv:1803.07819*, 2018. 4.3.1

[10] Steffen Bickel, Michael Brückner, and Tobias Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM, 2007. 3.1.1

[11] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Generalizing from several related classification tasks to a new unlabeled sample. In *Advances in neural information processing systems*, pages 2178–2186, 2011. 2.1.1, 2.2, 2.4.1

[12] 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. 1.2.4, 4.1, 4.1.1, 4.4, 4.1, 4.7.1, 4.8, 5.5

[13] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, page 7, 2017. 2.1.1

[14] Ruichu Cai, Zijian Li, Pengfei Wei, Jie Qiao, Kun Zhang, and Zhifeng Hao. Learning disentangled semantic representation for domain adaptation. In *IJCAI: proceedings of the conference*, volume 2019, page 2060. NIH Public Access, 2019. 4.4.1, 4.1

[15] George Casella and Roger L Berger. *Statistical inference*. Cengage Learning, 2021. 5.3.1

[16] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20 (3):542–542, 2009. 1.2.5

[17] Rita Chattopadhyay, Qian Sun, Wei Fan, Ian Davidson, Sethuraman Panchanathan, and Jieping Ye. Multisource domain adaptation and its application to early detection of fatigue. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4):18, 2012. 2.1.1

[18] Q. Chen, Y. Liu, Z. Wang, I. Wassell, and K. Chetty. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7976–7985, 2018. 4.1

[19] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In *International conference on machine learning*, pages 1081–1090. PMLR, 2019. 4.4.1, 4.2

[20] C. Cortes, Y. Mansour, and M. Mohri. Learning bounds for importance weighting. In *NIPS 23*, 2010. 4.1

[21] 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. 1.2.1, 3.1.1, 3.4, 3.5.1, 3.7.2, 3.7.2

[22] Corinna Cortes, Yishay Mansour, and Mehryar Mohri. Learning bounds for importance weighting. In *Advances in neural information processing systems*, pages 442–450, 2010. 3.1.1

[23] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. In *NIPS*, 2017. 4.1.1

[24] 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, 2016. 5.1.1

[25] Nicolas Courty, Rémi Flamary, Amaury Habrard, and Alain Rakotomamonjy. Joint distribution optimal transportation for domain adaptation. *arXiv preprint arXiv:1705.08848*, 2017. 5.1.1

[26] Wenyuan Dai, Qiang Yang, Gui-Rong Xue, and Yong Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007. 3.1.1

[27] Povilas Daniusis, Dominik Janzing, Joris Mooij, Jakob Zscheischler, Bastian Steudel, Kun

Zhang, and Bernhard Schölkopf. Inferring deterministic causal relations. *arXiv preprint arXiv:1203.3475*, 2012. 1.1.1

[28] Zhijie Deng, Yucen Luo, and Jun Zhu. Cluster alignment with a teacher for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9944–9953, 2019. 4.4b

[29] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016. 5.2

[30] 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 *International conference on machine learning*, pages 647–655, 2014. 3.5.2

[31] Marthinus Christoffel Du Plessis and Masashi Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50: 110–119, 2014. 2.1.1

[32] Lixin Duan, Ivor W Tsang, Dong Xu, and Tat-Seng Chua. Domain adaptation from multiple sources via auxiliary classifiers. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 289–296. ACM, 2009. 2.1.1

[33] Miroslav Dudík, Steven J Phillips, and Robert E Schapire. Correcting sample selection bias in maximum entropy density estimation. In *Advances in neural information processing systems*, pages 323–330, 2006. 3.1.1

[34] Basura Fernando, Amaury Habrard, Marc Sebban, and Tinne Tuytelaars. Unsupervised visual domain adaptation using subspace alignment. In *Proceedings of the IEEE international conference on computer vision*, pages 2960–2967, 2013. 4.2

[35] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021. URL `http://jmlr.org/papers/v22/20-451.html`. 5.1.1, 5.5

[36] Geoffrey French, Michal Mackiewicz, and Mark Fisher. Self-ensembling for visual domain adaptation. *arXiv preprint arXiv:1706.05208*, 2017. 1.2.5

[37] Kenji Fukumizu, Francis R Bach, and Michael I Jordan. Dimensionality reduction for supervised learning with reproducing kernel hilbert spaces. *Journal of Machine Learning Research*, 5(Jan):73–99, 2004. 3.3.1

[38] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. *arXiv preprint arXiv:1409.7495*, 2014. 1.2.4, 4.1, 4.1.1, 4.4b, 4.1, 4.2

[39] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 2.1.1, 4.3.1, 4.4, 4.7.4, 5.5

[40] Jing Gao, Wei Fan, Jing Jiang, and Jiawei Han. Knowledge transfer via multiple model local structure mapping. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 283–291. ACM, 2008. 2.1.1

[41] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. Geodesic flow kernel for unsupervised domain adaptation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2066–2073. IEEE, 2012. 3.5.2, 4.4.1, 4.2

[42] Mingming Gong, Kun Zhang, Tongliang Liu, Dacheng Tao, Clark Glymour, and Bernhard Schölkopf. Domain adaptation with conditional transferable components. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2839–2848, 2016. 1.1.1, 1.2.2, 2.1, 2.1.1

[43] Mingming Gong, Yanwu Xu, Chunyuan Li, Kun Zhang, and Kayhan Batmanghelich. Twin auxilary classifiers gan. In *Advances in Neural Information Processing Systems*, pages 1328–1337, 2019. 4.3.1

[44] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Conditional generative moment-matching networks. In *NIPS*, pages 2672–2680, 2014. 1.2.3, 1.2.4

[45] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1

[46] Yves Grandvalet, Yoshua Bengio, et al. Semi-supervised learning by entropy minimization. *CAP*, 367:281–296, 2005. 1.2.5, 6.1.2

[47] Luigi Gresele, Giancarlo Fissore, Adrián Javaloy, Bernhard Schölkopf, and Aapo Hyvarinen. Relative gradient optimization of the jacobian term in unsupervised deep learning. *Advances in Neural Information Processing Systems*, 33, 2020. 5.2, 5.4.2

[48] A. Gretton, K. Fukumizu, Z. Harchaoui, and B. K. Sriperumbudur. A fast, consistent kernel two-sample test. In *NIPS 23*, pages 673–681, Cambridge, MA, 2009. MIT Press. 4.7.1

[49] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *JMLR*, 13:723–773, 2012. 4.3.1

[50] Arthur Gretton, Alexander J Smola, Jiayuan Huang, Marcel Schmittfull, Karsten M Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. 2009. 3.1, 3.1.1, 3.2, 3.3.1, 3.4

[51] 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. 2.2.1, 2.3.1

[52] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007. 3.5.2

[53] Aditya Grover, Christopher Chute, Rui Shu, Zhangjie Cao, and Stefano Ermon. Alignflow: Cycle consistent learning from multiple domains via normalizing flows. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4028–4035, 2020. 5.2

[54] Xiang Gu, Jian Sun, and Zongben Xu. Spherical space domain adaptation with robust pseudo-label loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and*

Pattern Recognition, pages 9101–9110, 2020. 4.1, 4.1.1, 4.4.1, 4.5, 4.4b, 4.2, 4.8, 5.1

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4.4b

[56] James J Heckman. Sample selection bias as a specification error (with an application to the estimation of labor supply functions), 1977. 3.1

[57] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *International conference on machine learning*, pages 1989–1998. PMLR, 2018. 4.1.1

[58] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013. 1.2.3

[59] P.O. Hoyer, D. Janzing, J. Mooji, J. Peters, and B. Schölkopf. Nonlinear causal discovery with additive noise models. In *Advances in Neural Information Processing Systems 21*, Vancouver, B.C., Canada, 2009. 1.1.1

[60] Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012. 6.1.2

[61] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *NIPS 19*, pages 601–608, 2007. 4.1

[62] Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2006. 3.1.1, 3.5, 3.5.1

[63] Su-Yun Huang, Chii-Ruey Hwang, and Miao-Hsiang Lin. Kernel fisher's discriminant analysis in gaussian reproducing kernel hilbert space. *Taiwan: Academia Sinica*, 2005. 6.1.1

[64] Rafael Izbicki, Ann Lee, and Chad Schafer. High-dimensional density ratio estimation with extensions to approximate likelihood computation. In *Artificial Intelligence and Statistics*, pages 420–429, 2014. 3.1

[65] Dominik Janzing and Bernhard Schölkopf. Causal inference using the algorithmic markov condition. *IEEE Transactions on Information Theory*, 56(10):5168–5194, 2010. 1.1.1

[66] Jing Jiang. A literature survey on domain adaptation of statistical classifiers. *URL: http://sifaka. cs. uiuc. edu/jiang4/domainadaptation/survey*, 3, 2008. 2.1.1

[67] Ying Jin, Ximei Wang, Mingsheng Long, and Jianmin Wang. Minimum class confusion for versatile domain adaptation. In *European Conference on Computer Vision*, pages 464–480. Springer, 2020. 1.2.5

[68] Takafumi Kanamori, Shohei Hido, and Masashi Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10(Jul):1391–1445, 2009. 3.1, 3.1.1

[69] Leonid V Kantorovich. On the translocation of masses. In *Dokl. Akad. Nauk. USSR (NS)*, volume 37, pages 199–201, 1942. 5.1.1

[70] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018. 5.2

[71] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1.2.4, 2.4.2

[72] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013. 1.2.5, 6.1.2

[73] Xiao Li and Jeff Bilmes. A bayesian divergence prior for classiffier adaptation. In *Artificial Intelligence and Statistics*, pages 275–282, 2007. 3.1.1

[74] Z. C. Lipton, Y. Wang, and A. Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018. 4.1

[75] M. Long, J. Wang, G. Ding, J. Sun, and P. S. Yu. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*, pages 2200–2207, 2013. 4.1

[76] M. Long, Y. Cao, J. Wang, and M. Jordan. Learning transferable features with deep adaptation networks. In *ICML-15*, pages 97–105, 2015. 4.1.1

[77] M. Long, H. Zhu, J. Wang, and M. I. Jordan. Deep transfer learning with joint adaptation networks. In *Proc. 34th International Conference on Machine Learning (ICML 2017)*, 2017. 1.2.4

[78] 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. 2.1.1, 4.1

[79] Mingsheng Long, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. *arXiv preprint arXiv:1605.06636*, 2016. 1.2.4, 2.1.1

[80] 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. 2.1.1

[81] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. *arXiv preprint arXiv:1705.10667*, 2017. 1.2.4, 4.4b

[82] 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. org, 2017. 4.1

[83] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018. 1.2.4, 4.1, 4.1.1, 5.1, 5.2, 5.5.2

[84] T. Liu D. Tao C. Glymour M. Gong, K. Zhang and B. Schölkopf. Domain adaptation with conditional transferable components. In *ICML 2016*, 2016. 4.1, 4.1.1

[85] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation with

multiple sources. In *Advances in neural information processing systems*, pages 1041–1048, 2009. 2.1.1, 2.4.1

[86] Sebastian Mika, Bernhard Schölkopf, Alexander J Smola, Klaus-Robert Müller, Matthias Scholz, and Gunnar Rätsch. Kernel pca and de-noising in feature spaces. In *NIPS*, volume 11, pages 536–542, 1998. 2.3.1

[87] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1.2.3

[88] Krikamol Muandet, David Balduzzi, and Bernhard Schölkopf. Domain generalization via invariant feature representation. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 10–18, 2013. 2.1.1, 2.2

[89] Jaemin Na, Heechul Jung, HyungJin Chang, and Wonjun Hwang. Fixbi: Bridging domain spaces for unsupervised domain adaptation. *arXiv preprint arXiv:2011.09230*, 2020. 1.2.4, 4.1.1, 4.8

[90] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010. 2.1.1

[91] Sinno Jialin Pan, Ivor W Tsang, James T Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22(2):199–210, 2010. 4.4.1, 4.2

[92] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017. 5.2

[93] J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, Cambridge, 2000. 2.1

[94] Judea Pearl. *Causality*. Cambridge university press, 2009. 1.1.1

[95] Judea Pearl et al. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009. 1.1.1, 1.1.1

[96] M. D. Plessis and M. Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. In *ICML-12*, pages 823–830, 2012. 4.1

[97] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015. 5.2

[98] P. R. Rosenbaum and D. B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70:41–55, 1983. 3.3

[99] James C Ross, Gordon L Kindlmann, Yuka Okajima, Hiroto Hatabu, Alejandro A Díaz, Edwin K Silverman, George R Washko, Jennifer Dy, and Raúl San José Estépar. Pulmonary lobe segmentation based on ridge surface sampling and shape model fitting. *Medical physics*, 40(12), 2013. 2.4.3

[100] Kate Saenko, Brian Kulis, Mario Fritz, and Trevor Darrell. Adapting visual category models to new domains. In *European conference on computer vision*, pages 213–226. Springer, 2010. 3.5.2

[101] Kuniaki Saito, Yoshitaka Ushiku, and Tatsuya Harada. Asymmetric tri-training for un-

supervised domain adaptation. In *International Conference on Machine Learning*, pages 2988–2997. PMLR, 2017. 6.1.2

[102] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij. On causal and anticausal learning. In *Proc. 29th International Conference on Machine Learning (ICML 2012)*, Edinburgh, Scotland, 2012. 1.1.1, 1.2, 2.1, 4.1

[103] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *International Conference on Artificial Neural Networks*, pages 583–588. Springer, 1997. 2.2.2, 6

[104] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. *arXiv preprint arXiv:1206.6471*, 2012. 1.1.1

[105] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90:227–244, 2000. 4.1

[106] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. 1.2.1, 2.1.1, 3.1, 3.1.1

[107] Rui Shu, Hung Bui, Hirokazu Narui, and Stefano Ermon. A DIRT-t approach to unsupervised domain adaptation. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=H1q-TM-AW`. 4.4.1, 4.1, 4.2, 5.2, 5.5.2

[108] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018. 1.2.4, 2.1.1, 4.1.1, 4.3, 4.7.3, 4.7.4, 5.1, 6.1.2

[109] Alex Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A hilbert space embedding for distributions. In *International Conference on Algorithmic Learning Theory*, pages 13–31. Springer, 2007. 2.2.1, 6

[110] Le Song, Kenji Fukumizu, and Arthur Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013. 2.2.1

[111] Le Song, Han Liu, Ankur Parikh, and Eric Xing. Nonparametric latent tree graphical models: Inference, estimation, and structure learning. *arXiv preprint arXiv:1401.3940*, 2014. 6.1.2

[112] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction, and Search.* MIT Press, Cambridge, MA, 2nd edition, 2001. 2.1

[113] Peter Spirtes, Clark N Glymour, Richard Scheines, and David Heckerman. *Causation, prediction, and search.* MIT press, 2000. 1.1.1

[114] Petar Stojanov, Mingming Gong, Jaime Carbonell, and Kun Zhang. Data-driven approach to multiple-source domain adaptation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3487–3496, 2019. 1.3, 4.1

[115] Petar Stojanov, Mingming Gong, Jaime Carbonell, and Kun Zhang. Low-dimensional

density ratio estimation for covariate shift correction. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3449–3458, 2019. 1.3

[116] A. Storkey. When training and test sets are different: Characterizing learning transfer. In J. Candela, M. Sugiyama, A. Schwaighofer, and N. Lawrence, editors, *Dataset Shift in Machine Learning*, pages 3–28. MIT Press, 2009. 4.1

[117] Amos Storkey. When training and test sets are different: characterizing learning transfer. *Dataset shift in machine learning*, pages 3–28, 2009. 2.1.1, 3.1

[118] M. Sugiyama, M. Krauledat, and K. R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8:985–1005, December 2007. 1.2.1

[119] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60:699–746, 2008. (document), 1.2.1, 1.6, 1.2.1, 4.1

[120] Masashi Sugiyama, Shinichi Nakajima, Hisashi Kashima, Paul V Buenau, and Motoaki Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in neural information processing systems*, pages 1433–1440, 2008. 2.1.1, 3.1, 3.1.1, 3.5

[121] Masashi Sugiyama, Makoto Yamada, Paul Von Buenau, Taiji Suzuki, Takafumi Kanamori, and Motoaki Kawanabe. Direct density-ratio estimation with dimensionality reduction via least-squares hetero-distributional subspace search. *Neural Networks*, 24(2):183–198, 2011. 3.1, 3.5

[122] Masashi Sugiyama, Taiji Suzuki, and Takafumi Kanamori. Density-ratio matching under the bregman divergence: a unified framework of density-ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012. 3.1

[123] Baochen Sun, Jiashi Feng, and Kate Saenko. Return of frustratingly easy domain adaptation. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 4.2

[124] Taiji Suzuki and Masashi Sugiyama. Sufficient dimension reduction via squared-loss mutual information estimation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 804–811, 2010. 3.3.1

[125] A Tarvainen and H Valpola. Weight-averaged consistency targets improve semi-supervised deep learning results. corr abs/1703.01780. *arXiv preprint arXiv:1703.01780*, 2017. 1.2.5

[126] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017. 4.1, 4.1.1, 4.8

[127] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020. 1.2.5

[128] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017. URL `https://arxiv.org/pdf/1706.03762.pdf`. 6.1.2

[129] Hao Wang, Hao He, and Dina Katabi. Continuously indexed domain adaptation. *arXiv*

preprint arXiv:2007.01807, 2020. 4.1, 4.1.1, 4.8

[130] Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. Balanced distribution adaptation for transfer learning. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 1129–1134. IEEE, 2017. 4.2

[131] Jindong Wang, Yiqiang Chen, Han Yu, Meiyu Huang, and Qiang Yang. Easy transfer learning by exploiting intra-domain structures. In *2019 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1210–1215. IEEE, 2019. 4.2

[132] Qian Wang and Toby P Breckon. Unsupervised domain adaptation via structured prediction based selective pseudo-labeling. *arXiv preprint arXiv:1911.07982*, 2019. 4.1.1

[133] Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2), 2009. 2.1.1

[134] James Woodward. *Making things happen: A theory of causal explanation*. Oxford university press, 2005. 1.1.1, 2.1.1

[135] Shaoan Xie, Zibin Zheng, Liang Chen, and Chuan Chen. Learning semantic representations for unsupervised domain adaptation. In *International Conference on Machine Learning*, pages 5423–5432, 2018. 4.4b, 4.7.2, 4.8, 5.1, 5.2, 5.5.2

[136] Renjun Xu, Pelen Liu, Liyan Wang, Chao Chen, and Jindong Wang. Reliable weighted optimal transport for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4394–4403, 2020. 5.1.1

[137] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. Larger norm more transferable: An adaptive feature norm approach for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1426–1435, 2019. 4.4b

[138] Makoto Yamada, Taiji Suzuki, Takafumi Kanamori, Hirotaka Hachiya, and Masashi Sugiyama. Relative density-ratio estimation for robust distribution comparison. In *Advances in neural information processing systems*, pages 594–602, 2011. 3.1, 3.5

[139] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *ICML-04*, pages 114–121, Banff, Canada, 2004. 4.1

[140] Bianca Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the twenty-first international conference on Machine learning*, page 114. ACM, 2004. 2.1.1, 3.1, 3.1.1

[141] Jing Zhang, Wanqing Li, and Philip Ogunbona. Joint geometrical and statistical alignment for visual domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1859–1867, 2017. 4.2

[142] K. Zhang and A. Hyvärinen. On the identifiability of the post-nonlinear causal model. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, Montreal, Canada, 2009. 1.1.1

[143] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In *ICML-13*, 2013. 4.1

[144] K Zhang, J Zhang, and B Schölkopf. Distinguishing cause from effect based on exogeneity.

In *Fifteenth Conference on Theoretical Aspects of Rationality and Knowledge (TARK), 2015*, pages 261–271. Carnegie Mellon University, 2015. 1.1.1

[145] K. Zhang, B. Huang, J. Zhang, C. Glymour, and B. Schölkopf. Causal discovery from nonstationary/heterogeneous data: Skeleton estimation and orientation determination. In *IJCAI*, volume 2017, page 1347, 2017. (document), 1.1.1, 1.4, 1.2.2, 1.2.3, 5.1, 5.5.2, 6.1.1

[146] Kai Zhang, Vincent Zheng, Qiaojun Wang, James Kwok, Qiang Yang, and Ivan Marsic. Covariate shift in hilbert space: A solution via sorrogate kernels. In *International Conference on Machine Learning*, pages 388–395, 2013. 4.4.1

[147] Kun Zhang, Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. Kernel-based conditional independence test and application in causal discovery. *arXiv preprint arXiv:1202.3775*, 2012. 1.2.3

[148] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *ICML (3)*, pages 819–827, 2013. 1.1.1, 1.2, 1.2.2, 2.1, 2.1.1, 2.3.1, 3.6, 4.1, 5.1.1

[149] Kun Zhang, Mingming Gong, and Bernhard Schölkopf. Multi-source domain adaptation: A causal view. In *AAAI*, pages 3150–3157, 2015. 1.1.1, 1.2, 1.2.1, 1.2.2, 2.1, 2.1.1, 2.4.1

[150] Kun Zhang, Mingming Gong, Petar Stojanov, Biwei Huang, Qingsong Liu, and Clark Glymour. Domain adaptation as a problem of inference on graphical models. *arXiv preprint arXiv:2002.03278*, 2020. (document), 1.3, 1.1.1, 1.4, 1.2.3, 5.1, 6.1.2

[151] Weichen Zhang, Wanli Ouyang, Wen Li, and Dong Xu. Collaborative and adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3801–3809, 2018. 4.4b

[152] Yabin Zhang, Hui Tang, Kui Jia, and Mingkui Tan. Domain-symmetric networks for adversarial domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5031–5040, 2019. 4.4b

[153] Yabin Zhang, Haojian Zhang, Bin Deng, Shuai Li, Kui Jia, and Lei Zhang. Semi-supervised models are strong unsupervised domain adaptation learners. *arXiv preprint arXiv:2106.00417*, 2021. 1.2.5, 6.1.2

[154] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. Bridging theory and algorithm for domain adaptation. volume 97 of *Proceedings of Machine Learning Research*, pages 7404–7413, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/zhang19i.html`. 4.4.1, 4.1, 4.2, 5.2, 5.5.2

[155] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael I Jordan. Bridging theory and algorithm for domain adaptation. *arXiv preprint arXiv:1904.05801*, 2019. 4.1.1, 4.7.3, 4.7.4

[156] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J Gordon. On learning invariant representation for domain adaptation. *arXiv preprint arXiv:1901.09453*, 2019. 4.1

[157] Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing*

systems, pages 321–328, 2004. 1.2.5

[158] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. 5.2

[159] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005. 1.2.5