Using Prior Data to Facilitate Learning and Inference in New Environments

by

Junfeng Wen

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Statistical Machine Learning

Department of Computing Science

University of Alberta

Abstract

This dissertation demonstrates how to utilize data collected previously from different sources to facilitate learning and inference for a target task. Learning from scratch for a target task or environment can be expensive and time-consuming. To address this problem, we make three contributions in this dissertation: (1) improving the efficiency of classical domain adaptation methods, (2) developing a novel theory and algorithm for multi-source adaptation, and (3) proposing a theoretically sound approach to estimate the stationary distribution of a Markov chain from batch data. For (1), specifically in the covariate shift scenario, classical methods that compute importance weights suffer from computational issues when the sample size is large. We resolve such issues from an optimization perspective by applying the Frank-Wolfe algorithm. For (2), to fully utilize data from not one but multiple sources, we develop a theory and a corresponding algorithm that are suitable for selecting the most relevant sources for adaptation. Finally, for (3), we propose a method to estimate a target stationary distribution from batch data without interacting with the environment. It is a general method that can be applied to many use cases such as off-policy evaluation in reinforcement learning and post-processing MCMC samples. For all three contributions, we provide empirical studies on various tasks and environments, which show that utilizing prior data effectively can indeed improve learning for a target task.

To my parents.

Acknowledgements

I want to express my sincerest gratitude to my two fantastic supervisors, Russell Greiner and Dale Schuurmans, for their inspiration, guidance and support. I especially thank them for providing me academic freedom to explore and pursue my own interests. I am very grateful to Bo Dai and Lihong Li, who have provided valuable advice and help. Without them, this thesis would not be possible. I also want to thank my committee members, Prof. Csaba Szepesvári, Prof. Nilanjan Ray, and Prof. Masashi Sugiyama whose prior work has inspired much of this thesis.

I want to thank the friends I made during my Ph.D. studies in Greiner's group, Schuurmans' group, Alberta Machine Intelligence Institute (Amii) and Borealis AI. Pursuing a Ph.D. is a unique experience, and they have made this journey much more enjoyable.

Finally, I would like to thank my parents for their love and understanding, and my girlfriend Fushan Li for her company and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine learning has achieved phenomenal advancements in recent years. For instance, deep neural networks have conquered large scale image recognition challenges and revolutionized how machine perceives images (Deng et al., 2009; Krizhevsky et al., 2012); AlphaGo (Silver et al., 2016) and its successors (Silver et al., 2017, 2018) showed their capabilities of surpassing human experts in playing Go, Shogi, and Chess. However, these remarkable achievements come with a price. Deep neural networks or reinforcement learning agents are usually data-hungry: *e.g.*, learning image classifier requires millions of labelled images from the ImageNet[1] dataset, and learning to play Atari games requires millions of interactions with the environment. Acquiring such a huge amount of data for a specific task can be difficult or even impossible due to, for example, ethical regulations in medical domains or physical constraints on robotic arms.

To mitigate the aforementioned data-hungry issue, one idea would be to utilize data from a different but related *source* domain to facilitate learning/inference for a specific *target* domain. By using previously collected data effectively, one can avoid the burden of acquiring a significant amount of (labelled) target data. This idea has been applied to domain adaptation and off-policy reinforcement learning.

[1] www.image-net.org

1.1 Domain Adaptation

One common scenario of domain adaptation (Mansour et al., 2009a) or transfer learning (Pan and Yang, 2009) is to use a labelled source dataset and an unlabelled target dataset to learn a model that performs well in the target domain. Of course, such adaptation from one domain to another would typically fail if the source and the target have nothing in common. Therefore, researchers have imposed certain assumptions (Zhang et al., 2013; Wang and Schneider, 2014), characterizing how the joint distribution $p(x, y)$ of input x and output y are allowed to differ across domains. As an example, covariate shift (Shimodaira, 2000; Sugiyama and Kawanabe, 2012) assumes that the conditional distribution $p(y|x)$ remains the same across domains while the marginal distributions $p(x)$ are different. To attain provable generalization across domains, much existing analysis focus on developing theories based on various discrepancy measures (Banerjee et al., 2005; Gretton et al., 2012), which have provided justifications for the algorithms (Gretton et al., 2009; Sugiyama et al., 2007, 2008).

Importance sampling (Shimodaira, 2000) is one of the mainstream techniques for domain adaption. It learns an importance weight for each source instance and uses these weights to compute a weighted loss for downstream training. Although importance sampling techniques have demonstrated noticeable improvement over non-adapting methods in the literature, applying them to large-scale datasets is often difficult due to their computational cost (Pan et al., 2009). As a result, researchers have resorted to heuristic remedies such as sub-sampling (Sugiyama et al., 2007).

Besides computational issues, it may be critical to adapt from *multiple* sources. Adapting from one source to one target is relatively well studied, and we have well-developed techniques and theories to address (and often solve) this problem. However, in practice, we may encounter a more challenging scenario in which data are collected from not one but multiple sources, and we need to effectively combine them in order to learn a

model to predict data in a target domain. This is ubiquitous, for example, in medical problems, as the number of patients in one site is usually small and providing the labels (here, the diagnosis) for patients requires non-trivial efforts from medical experts. Therefore, utilizing electronic health records from several other sites becomes necessary in the hope that this information can significantly improve the quality of the learned model.

1.2 Off-policy Reinforcement Learning

Reinforcement learning (Sutton and Barto, 2018) focuses on learning by interacting with the environment. By observing the state of the environment, the learning agent takes actions and in return, the environment provides feedback such as a reward signal to guide the agent. The policy of an agent is defined as a mapping from the environment state to an action. The agent usually needs to interact with the environment for millions of steps before mastering a superior policy. Unlike simulation-based environments that enable fast and cheap interactions, real-world interactions take much more time and energy. For example, the movement of real robotic arms is restricted due to motor limits and component configurations. Therefore, using data collected from different but related environments is not only necessary but also essential for fast learning.

The field of batch reinforcement learning (Lange et al., 2012) investigates this idea and focuses on how to learn a target policy from data collected from some behaviour policy. As a sub-problem in this field, evaluating the target policy's performance using behaviour data (obtained using a different policy) is known as off-policy evaluation (OPE), which finds many practical applications such as A/B testing (Li et al., 2015a) and treatment effect estimation (Leacy and Stuart, 2014).

Similar to covariate shift, importance sampling also plays a critical role in OPE. In the simpler contextual bandit setting where there is no temporal structure and the decision is one-shot, OPE can be carried out by

reweighting each observed reward by the action probability ratio of the target policy over the behaviour policy. Applying importance sampling can be challenging in practice due to two difficulties. First, the behaviour action probability is not always available. We may not know why the behaviour actions are chosen as they are. Second, the product of probability ratios will incur an unbearably high variance when it comes to multi-step decisions in general RL setting (Li et al., 2015b; Jiang and Li, 2016), which is known as the "curse of horizon" (Liu et al., 2018). How to reliably solve behaviour-agnostic OPE for multi-step RL problems remains unclear in the literature.

1.3 Contributions

In this dissertation, I will present three contributions outlined as follows:

- To solve the computational issue of popular covariate shift correction algorithms such as Kernel Mean Matching (KMM) (Gretton et al., 2009) and Kullback-Leibler Importance Estimation Procedure (KLIEP) (Sugiyama et al., 2008), we apply the Frank-Wolfe algorithm (Jaggi, 2013) to improve their computational efficiency. The proposed optimization procedure is suitable for these algorithms because it can handle the optimization constraints in a projection-free way. Specifically, the time complexity of solving KMM is reduced from $O(n^2)$ to $O(n)$ per iteration where n is the number of training examples. In addition, the proposed method can bypass the need for sub-sampling when solving KLIEP, which our empirical studies show is critical in finding better solutions. Chapter 2 summarizes this work, which has been published in a refereed conference proceeding (Wen et al., 2015).

- For multi-source adaptation, we theoretically analyze how to combine data from different sources effectively and derive an end-to-end algorithm that is suitable for training deep neural networks. The

new algorithm can choose the most relevant source data during training and produce interpretable domain weights. Experiments on real-world problems such as image classification and sentiment analysis show that the proposed method outperforms state-of-the-art alternatives. Chapter 3 summarizes this work, which has been published in a refereed conference proceeding (Wen et al., 2020b).

- We propose a variational power method for estimating stationary distribution of a Markov chain, which can be used for solving behaviour-agnostic OPE in RL. It is a general method that exploits the eigen-structure of the state transitions to estimate the stationary distribution using batch data, without any further interaction with the environment. We theoretically prove that this method converges. In addition to OPE, this variational power method has a wide range of applications for many other fundamental problems, such as solving stochastic differential equations and post-processing MCMC samples. Chapter 4 summarizes this work, which has been published in a refereed conference proceeding (Wen et al., 2020a).

Chapter 2

Correcting Covariate Shift with the Frank-Wolfe Algorithm

2.1 Overview

Covariate shift is a fundamental problem for learning in non-stationary environments where the conditional distribution $p(y|x)$ is the same between training and test data while their marginal distributions $p_{tr}(x)$ and $p_{te}(x)$ are different. Although many covariate shift correction techniques work effectively for real world problems, most do not scale well in practice. In this chapter, using inspiration from recent optimization techniques, we apply the Frank-Wolfe algorithm to two well-known covariate shift correction techniques, Kernel Mean Matching (KMM) and Kullback-Leibler Importance Estimation Procedure (KLIEP), and identify an important connection between kernel herding and KMM. Our complexity analysis shows the benefits of the Frank-Wolfe approach over projected gradient methods in solving KMM and KLIEP. An empirical study then demonstrates the effectiveness and efficiency of the Frank-Wolfe algorithm for correcting covariate shift in practice.

2.2 Introduction

Many machine learning algorithms assume that training and test data come from the same distribution, which is often violated in practical appli-

cations. Researchers have thus endeavoured to resolve the distribution shift problem under varied assumptions (Zadrozny, 2004; Bickel et al., 2007; Quionero-Candela et al., 2009). In this chapter, we focus on the *covariate shift* scenario (Shimodaira, 2000) in which the marginal data distributions are different between training and test domains ($p_{tr}(x) \neq p_{te}(x)$) while their conditional distributions remain the same ($p_{tr}(y|x) = p_{te}(y|x)$). Correcting covariate shift has a wide range of applications, such as in natural language processing (Jiang and Zhai, 2007), off-policy reinforcement learning (Hachiya et al., 2009), computer vision (Yamada et al., 2012) and signal processing (Yamada et al., 2010).

A common approach to correcting covariate shift is *importance sampling/reweighting*: each individual training point is assigned a non-negative weight intended to diminish the discrepancy between training and test marginals by some criterion (Sugiyama et al., 2012a). If a reweighting function is modelled properly, covariate shift can be effectively corrected in learning. For example, for a given prediction function $f(x)$ and a loss function $l(f(x), y)$, reweighting training points with weight $w(x) = p_{te}(x)/p_{tr}(x)$ can minimize the loss over the test distribution:

$$\text{loss}(f) = \mathbb{E}_{x\sim p_{te}}\mathbb{E}_{y|x}[l(f(x), y)] = \mathbb{E}_{x\sim p_{tr}}\mathbb{E}_{y|x}[w(x)\ l(f(x), y)],$$

which suggests that one should seek the function $f^\star = \text{argmin}_f \text{loss}(f)$ that minimizes this expected loss. While these correction techniques remain effective for many real world applications, most of them do not exploit the structure of the solution and as a result, do not scale well in practice.

Recently, the Frank-Wolfe (FW) algorithm has begun to gain popularity in the machine learning community (Zhang et al., 2012; Bach, 2015). It has been proven to be an efficient algorithm for many optimization problems, particularly when the solution has sparse structure (Jaggi, 2013), and has also been shown effective and efficient in many applications (Joulin et al., 2014; Salamatian et al., 2014). However, whether Frank-Wolfe can be applied to the covariate shift problem has remained unexplored.

In this work, we show that herding (Welling, 2009; Chen et al., 2010) (as a Frank-Wolfe algorithm) can be applied to the covariate shift scenario, and point out a connection between kernel herding and Kernel Mean Matching (KMM) (Gretton et al., 2009). Moreover, we exploit the structure of another commonly used covariate shift correction technique, Kullback-Leibler Importance Estimation Procedure (KLIEP) (Sugiyama et al., 2008), to speed up the algorithm using Frank-Wolfe. We also analyse the convergence rate and complexity of KMM(FW) and KLIEP(FW), and present an empirical study that demonstrates the efficiency of Frank-Wolfe in solving KMM and KLIEP over the traditional projected gradient approach.

2.3 Related Work

This work is closely related to kernel herding (Chen et al., 2010), which is a sampling technique for moment approximation. In the original setting, data points are iteratively generated to approximate the population mean (first moment), either from a population when the distribution p is known, or selected from existing dataset when p is unknown. This work is different in that we apply herding to the covariate shift scenario, where the goal is to approximate the *test* mean by sub-sampling existing training data.

Even though there are many ways to correct covariate shift, few methods scale well, which hampers their usage in practice. Tsuboi et al. (2009) modifies KLIEP with a different parametric form of the weight function. Although the computation time of the new model is independent of the number of test points, which is computationally advantageous, the proposed parametric form is less interpretable than the original KLIEP. Here we apply Frank-Wolfe to the original KLIEP, where the weight function is parametrized by a mixture of Gaussians. Another attempt has been made to solve KMM and KLIEP via online learning (Agarwal et al., 2011). However, the convergence rate of that approach, $O(1/\sqrt{t})$, is slower than our proposed method, $O(1/t)$, and assumes the availability of explicit feature

representation of $\phi(\mathbf{x})$, contrary to the idea of Hilbert space estimation.

There are also several other techniques that correct covariate shift. For instance, RuLSIF (Yamada et al., 2011) minimizes the α-relative Pearson divergence using least squares, while RCSA (Wen et al., 2014) corrects covariate shift in an adversarial setting. We do not extensively investigate the possibility of applying Frank-Wolfe to these techniques, since Frank-Wolfe is efficient primarily when a convex problem has sparse solution structure and the corresponding linearised problem can be solve easily.

2.4 A Brief Review of Herding

Kernel herding (Chen et al., 2010) is a greedy algorithm that sequentially generates instances, $\phi(\mathbf{x}_t)$ in the t^{th} iteration, to minimize a squared error $\mathcal{E}_T^2$:

$$\mathcal{E}_T^2 = \left\| \frac{1}{T} \sum_{t=1}^{T} \phi(\mathbf{x}_t) - \mu_p \right\|_{\mathcal{H}}^2,$$

where $\phi(\cdot)$ is a feature map to a reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ and $\mu_p = \mathbb{E}_{x\sim p}[\phi(\mathbf{x})]$ is the population mean. Algorithmically, it generates instances according to:

$$\begin{aligned} \mathbf{x}_{t+1} &\in \operatorname*{argmax}_{\mathbf{x}\in\mathcal{X}} \langle u_t, \phi(\mathbf{x}) \rangle, \\ u_{t+1} &= u_t + \mu_p - \phi(\mathbf{x}_{t+1}), \end{aligned} \tag{2.1}$$

for some properly chosen u_0, where $\mathcal{X}$ is the sampling space. In the original kernel herding algorithm, the expectation is taken with respect to the training population, i.e., $p = p_{tr}$.

Bach et al. (2012) showed that herding is equivalent to a Frank-Wolfe algorithm (also known as conditional gradient descent) for the objective

$$\min_{\hat{\mu}\in\mathcal{M}} \frac{1}{2} \|\hat{\mu} - \mu_p\|_{\mathcal{H}}^2,$$

where $\mathcal{M}$ is the marginal polytope (the convex hull of all $\phi(x)$ for $x \in \mathcal{X}$). Specifically, it employs the following updates

$$\begin{aligned} s_{t+1} &\in \operatorname*{argmin}_{s\in\mathcal{M}} \langle \hat{\mu}_t - \mu_p, s \rangle, \\ \hat{\mu}_{t+1} &= (1 - \rho_t)\hat{\mu}_t + \rho_t s_{t+1}, \end{aligned} \tag{2.2}$$

where $\rho_t \in [0, 1]$ is step size. They pointed out that herding corresponds to using a step size $\rho_t = 1/(t+1)$, while other choices (e.g., via line search) are also valid. With $\rho_t = 1/(t+1)$, s_t can be seen as $\phi(\mathbf{x}_t)$ while $\widehat{\mu}_t = \mu - u_t/t$ as in (2.1).

2.5 Frank-Wolfe for Covariate Shift

2.5.1 Kernel Herding and KMM

In this work, we apply kernel herding to the covariate shift scenario, where the training and test marginal distributions are different ($p_{tr}(\mathbf{x}) \neq p_{te}(\mathbf{x})$) while their conditional distributions are the same ($p_{tr}(y|\mathbf{x}) = p_{te}(y|\mathbf{x})$). We use herding to generate instances from the training pool in order to approximate the *test* mean, i.e., where $\mathcal{X}$ is the training set but $p = p_{te}$ is the test marginal distribution. Intuitively, herding actively selects representative training points to approximate the test population mean. With proper substitution of empirical estimates, our objective is

$$\min_{\widehat{\mu} \in \widehat{\mathcal{M}}} \frac{1}{2} \left\| \widehat{\mu} - \frac{1}{m} \sum_{j=1}^{m} \phi(\mathbf{x}_j) \right\|_{\mathcal{H}}^2,$$

where $\widehat{\mathcal{M}}$ is marginal polytope of training dataset and m is the number of (unlabelled) test points. According to (2.2), in each update step, a representative training point $\phi(\mathbf{x}_t)$ will be chosen from $\widehat{\mathcal{M}}$ and $\widehat{\mu}_t$ will be moved toward that direction. This objective is very similar to that of Kernel Mean Matching (KMM) (Gretton et al., 2009)[1]:

$$\begin{aligned} \min_{\mathbf{w} \in \mathcal{W}} \quad & \left\| \sum_{i=1}^{n} w_i \phi(\mathbf{x}_i) - \frac{1}{m} \sum_{j=1}^{m} \phi(\mathbf{x}_j) \right\|_{\mathcal{H}}^2 \\ & \mathcal{W} = \left\{ \mathbf{w} \middle| \sum_{i=1}^{n} w_i = 1, w_i \geqslant 0 \right\}, \end{aligned} \tag{2.3}$$

[1]Here the predefined bound on the weight ($w_i \leqslant B$) and the derivation tolerance ($|\sum_{i=1}^{n} w_i - 1| \leqslant \epsilon$) are ignored for the sake of clarity and simplicity. However, they can be easily incorporated into the herding scheme.

where n training points and m test points are given[2]. Problem (2.3) (a linearly constrained quadratic program) is generally solved by standard solver with gradient descent and feasibility projection. However, according to the interpretation in (2.2), herding for KMM is projection-free: with a proper step size $\rho_t \in [0, 1]$, the estimated mean $\widehat{\mu}_t$ is a convex combination of previous chosen training points, and thus the constraint $\mathcal{W}$ in (2.3) is automatically satisfied.

2.5.2 Frank-Wolfe for KLIEP

Once herding is viewed as a Frank-Wolfe algorithm, it can be applied to other covariate shift correction procedures. For instance, the Kullback-Leibler Importance Estimation Procedure (KLIEP) (Sugiyama et al., 2008), a popular choice for covariate shift correction, can be efficiently solved by Frank-Wolfe. The objective of KLIEP is to minimize the Kullback-Leibler divergence from $p_{te}(\mathbf{x})$ to $\widehat{p}_{te}(\mathbf{x})$, where $\widehat{p}_{te}(\mathbf{x})$ is attained by reweighting the training marginal:

$$\widehat{p}_{te}(\mathbf{x}) = \widehat{w}(\mathbf{x})\, p_{tr}(\mathbf{x}).$$

The reweighting function is further parametrized as a mixture of Gaussians:

$$\widehat{w}(\mathbf{x}) = \sum_{l=1}^{b} \alpha_l \varphi_l(\mathbf{x}) = \sum_{l=1}^{b} \alpha_l \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_l\|^2}{2\sigma^2}\right), \tag{2.4}$$

where α_l are the mixing coefficients, $\mathbf{c}_l$ are fixed/predefined centres (usually test points) and σ is a parameter selected by some criterion. After some derivation and substitution of empirical marginals, the final objective becomes

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & F(\boldsymbol{\alpha}) = \sum_{j=1}^{m} \log\left(\sum_{l=1}^{b} \alpha_l \varphi_l(\mathbf{x}_j)\right) \\ \text{s.t.} \quad & \sum_{i=1}^{n}\sum_{l=1}^{b} \alpha_l \varphi_l(\mathbf{x}_i) = n;\ \alpha_1, \alpha_2, \cdots, \alpha_b \geqslant 0. \end{aligned} \tag{2.5}$$

[2]With a slight abuse of notation, we use indices i and j to distinguish data points from training and test datasets respectively.

This is a convex optimization problem. To apply Frank-Wolfe, we first derive the gradient of the objective

$$\frac{\partial F(\boldsymbol{\alpha})}{\partial \alpha_l} = \sum_{j=1}^{m} \frac{\varphi_l(\mathbf{x}_j)}{\sum_{l'=1}^{b} \alpha_{l'} \varphi_{l'}(\mathbf{x}_j)}. \tag{2.6}$$

Considering the constraint, one can observe that the possible range for α_l is $[0, n/\sum_{i=1}^{n} \varphi_l(\mathbf{x}_i)]$. For the t^{th} iteration, we choose l_t such that

$$l_t = \underset{l}{\operatorname{argmax}} \left\langle \frac{\partial F(\boldsymbol{\alpha})}{\partial \alpha_l}, \frac{n}{\sum_{i=1}^{n} \varphi_l(\mathbf{x}_i)} \right\rangle,$$

then update the current $\boldsymbol{\alpha}_t$ by

$$\boldsymbol{\alpha}_{t+1} = (1 - \rho_t) \cdot \boldsymbol{\alpha}_t + \rho_t \sum_{i=1}^{n} \varphi_{l_t}(\mathbf{x}_i) \cdot \mathbf{e}_{l_t}, \tag{2.7}$$

where ρ_t is the step size and $\mathbf{e}_l = [0, \cdots, 0, 1, 0, \cdots, 0]^T$ (all zeros except for a one in the l^{th} entry). Compared to KMM(FW), this method chooses representative Gaussians to match test marginal, instead of choosing individual points.

The chosen Gaussian centres (i.e., $\{\mathbf{c}_l\}$ in Eq. (2.4)) are crucial for KLIEP. In the original paper (Sugiyama et al., 2008), the $\{\mathbf{c}_l\}$ are set to be the test points $\{\mathbf{x}_j\}$, which can be computationally intensive when one has many test points. Therefore, for large test data, the original authors proposed to randomly select $b = 100$ test points as Gaussian centres. However, such an approach can significantly reduce the performance of the algorithm, since the 100 chosen centres might not appropriately model the distribution difference, a phenomenon we observed in our experiments (see Section 2.7.2). If Frank-Wolfe is applied, only one Gaussian is amplified per iteration and the optimal solution can be efficiently attained (Jaggi, 2013), since it tends to be sparse (Sugiyama et al., 2008). More importantly, we can use *all* the test points as Gaussian centres in a computationally efficient way, so that their information can be used to capture the distribution difference.

2.6 Convergence and Complexity

2.6.1 Convergence Rate

In this section, we compare the convergence and complexity of projected gradient (PG) and Frank-Wolfe methods. For convergence, since both KMM and KLIEP are convex problems, it is well known that the convergence rate of PG is $O(1/t)$ (Bertsekas, 1999). The convergence rate of Frank-Wolfe for KMM and KLIEP is also $O(1/t)$ without further assumptions (Jaggi, 2013). However, under some circumstances, $O(1/t^2)$ and even $O(e^{-t})$ are possible for KMM(FW) (Chen et al., 2010; Beck and Teboulle, 2004). Moreover, the solutions to KMM and KLIEP tend to be sparse in practice (for example, see Section Section 2.7.1), which is a very suitable scenario for the Frank-Wolfe algorithm.

2.6.2 Complexity Analysis

Next we analyze the time and space complexities of the algorithms. Since both KMM and KLIEP are convex, the computation of the objective is not necessary in each iteration. Instead, one can simply check the gradient of the objective and stop when the norm of the gradient is sufficiently small.

We first focus on KMM. Ignoring the term independent of $\mathbf{w}$, one can rewrite the objective in Eq. (2.3) in matrix form as

$$\min_{\mathbf{w}\in\mathcal{W}} \quad \frac{1}{2}\mathbf{w}^\top K_{tr}\mathbf{w} - \mathbf{k}_{te}^\top \mathbf{w}, \tag{2.8}$$

where

$$(K_{tr})_{ii'} = \langle\, \phi(\mathbf{x}_i),\ \phi(\mathbf{x}_{i'}) \,\rangle, \quad (\mathbf{k}_{te})_i = \frac{1}{m}\sum_{j=1}^{m} \langle\, \phi(\mathbf{x}_i),\ \phi(\mathbf{x}_j) \,\rangle.$$

For the projected gradient method, the bottleneck is the computation of the gradient. From Eq. (2.8) it is clear that $O(n^2)$ multiplications are required to compute the gradient in each iteration. This cannot be simplified because, unlike the Frank-Wolfe algorithm which will be discussed below, the projection to $\mathcal{W}$ creates a nonlinear operation on $\mathbf{w}_t$ in each iteration. Projected

gradient requires additional $O(n)$ space to store the gradient. For Frank-Wolfe using a $1/(t+1)$ step size, we can see from Eq. (2.2) that the time complexity is $O(n)$ per iteration. As s_{t+1} must be a single $\phi(\mathbf{x}_{i^\star})$ for some $i^\star$, we only need to maintain a score list of size n to store $\{\langle \widehat{\mu}_t - \mu_p, \phi(\mathbf{x}_i)\rangle\}$ for all i. The scores can be distributedly stored, and updated in parallel if needed. After decomposition, $\langle \mu_p, \phi(\mathbf{x}_i)\rangle$ is simply $(\mathbf{k}_{te})_i$ and can be precomputed, while $\langle \widehat{\mu}_t, \phi(\mathbf{x}_i)\rangle$ can be easily maintained for each t as

$$\langle\ \widehat{\mu}_{t+1},\ \phi(\mathbf{x}_i)\ \rangle = (1-\rho_t)\cdot\langle\ \widehat{\mu}_t,\ \phi(\mathbf{x}_i)\ \rangle + \rho_t \cdot (K_{tr})_{ii^\star}.$$

This is faster than the projected gradient approach because there is no projection step in Frank-Wolfe, and the score list can be maintained efficiently with only $O(n)$ additional space. Finally, when line search is applied to ρ_t, the time complexity is still $O(n)$, since the line search step size is attained in closed form $\rho_t = \frac{\langle \mu_p - \widehat{\mu}_t,\ s_{t+1} - \widehat{\mu}_t\rangle}{\|s_{t+1} - \widehat{\mu}_t\|^2}$. We can decompose the inner products as before and see that all operations can be performed in $O(n)$ time. While the computation of $\|\widehat{\mu}_{t+1}\|^2 = \mathbf{w}_{t+1}^\top K_{tr}\mathbf{w}_{t+1}$ seems to require $O(n^2)$ multiplications, we can further decompose it, using the fact that $\mathbf{w}_{t+1} = (1-\rho_t)\mathbf{w}_t + \rho_t\delta\mathbf{w}_t$ for some increment $\delta\mathbf{w}_t$, as

$$\begin{aligned}
&(1-\rho_t)^2\cdot\mathbf{w}_t^\top K_{tr}\mathbf{w}_t + 2(1-\rho_t)\cdot\rho_t\cdot\mathbf{w}_t^\top K_{tr}\delta\mathbf{w}_t + \rho_t^2\cdot\delta\mathbf{w}_t^\top K_{tr}\delta\mathbf{w}_t \\
&= (1-\rho_t)^2\cdot\|\widehat{\mu}_t\|^2 + 2(1-\rho_t)\cdot\rho_t\cdot\mathbf{w}_t^\top(K_{tr})_{:i^\star} + \rho_t^2\cdot(K_{tr})_{i^\star i^\star},
\end{aligned}$$

where $(K_{tr})_{:i^\star}$ means the $i^\star$th column of K_{tr}. Therefore, only $O(n)$ multiplications are required since the first term can be used recursively and $\delta\mathbf{w}_t$ only has one singular non-zero entry. Similar tricks cannot be applied to projected gradient because its increment $\delta\mathbf{w}_t$ is not sparse, and $\delta\mathbf{w}_t^\top K_{tr}\delta\mathbf{w}_t$ still requires $O(n^2)$ computation in general. An additional $O(n)$ storage is still needed for the score list in the line search case.

Next we investigate the complexity of KLIEP. Similar to KMM, the bottleneck is the computation of the gradient in Eq. (2.6). The projected gradient method requires $2mb$ multiplications to compute Eq. (2.6) for all l, while Frank-Wolfe only requires $2m + mb$ multiplications, since the denominator is linear in $\boldsymbol{\alpha}$ and can be updated efficiently due to Eq. (2.7).

Table 2.1: Complexities of projected gradient (PG) and Frank-Wolfe with $1/(t+1)$ or line search step size. Time complexity is the complexity of one iteration. The additional required space is shared across iterations.

		PG	$1/(t+1)$	Line
KMM	Time	$O(n^2)$	$O(n)$	$O(n)$
	Space	$O(n)$	$O(n)$	$O(n)$
KLIEP	Time	$O(mb)$	$O(mb)$	$O(mb)$
	Space	$O(m+b)$	$O(m+b)$	$O(m+b)$

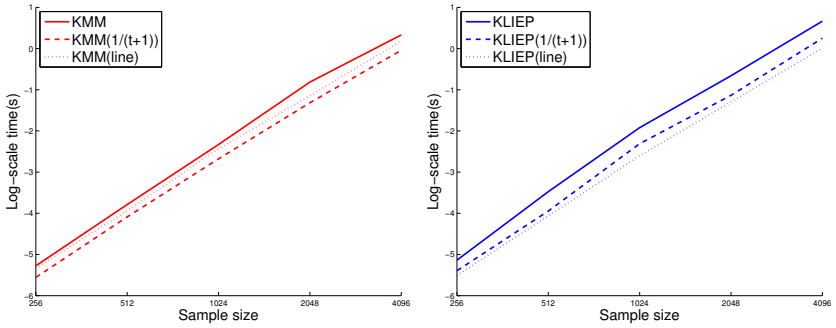


Figure 2.1: Runtime plot over different sample sizes. Time is measured in seconds.

This can lead to different run time in practice (see Section 2.7). Line search is less efficient for KLIEP(FW) compared to KMM(FW) as there is no closed form solution to ρ_t. To find the optimal step size of line search, note that $G(\rho_t) := F(\boldsymbol{\alpha}_{t+1})$ is a concave function of ρ_t. Moreover, the gradient $G'(\cdot)$ is decreasing in (0,1) with $G'(0) > 0$. Therefore, if $G'(1) \geqslant 0$, a step size of 1 is used, otherwise $G'(1) < 0$ and we use binary search on the interval $[0, 1]$ to find ρ_t such that $G'(\rho_t) = 0$. Such a binary search requires $O(m)$ multiplications. All three versions require $O(m+b)$ space to compute and store the gradient. Table 2.1 summarizes the results of this section.

2.7 Experimental Evaluation

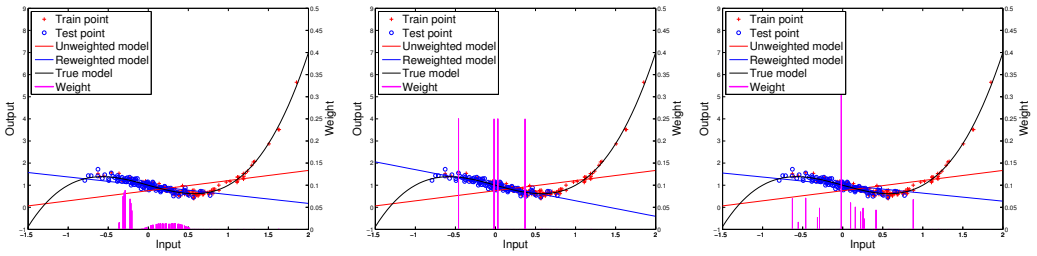
In this section, we demonstrate the results of the proposed approach on both synthetic and some benchmark datasets. In particular, we compare the proposed Frank-Wolfe method on KMM and KLIEP with the projected

gradient (PG) method. In the experiments, a Gaussian kernel is applied to KMM where the kernel width is chosen to be the median of the pairwise distances over the training set. For KLIEP, the width is chosen according to the criterion of Sugiyama et al. (2008).

2.7.1 Synthetic Datasets

To investigate the efficiency of the different methods, we compare their runtime to reach the same accuracy. Synthetic data is generated from $y = x^3 - x + 1 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 0.1^2)$. Training points are generated from $\mathcal{N}(0.5, 0.5^2)$ while test points are generated from $\mathcal{N}(0, 0.3^2)$ (Shimodaira, 2000). In the experiment, $n = m$. We first run the projected gradient method to convergence, then run the Frank-Wolfe methods (with two different step sizes: $\rho_t = 1/(t+1)$ versus line search) until it reaches the same objective. Fig. 2.1 shows the runtime in log scale with varied sample sizes. The Frank-Wolfe methods are consistently faster than the projected gradient method. In the case of KLIEP, although Frank-Wolfe and projected gradient have the same theoretical complexity and convergence as shown in Section 2.6, their actual runtime can differ in practice, as the constant factor in the big-O notation can be different. Also, note that line search has better efficiency in the case of KLIEP, since the final solution is usually extremely sparse, meaning that line search can find it in a few iterations while $1/(t+1)$ takes some more time.

We next check the solution quality of the different methods on the same problems, to determine the effectiveness of Frank-Wolfe. We generated $n = m = 128$ data points and show the final training weights in Figures 2.2 and 2.3. On one hand, the weights of the three KMM methods differ from each other significantly, but their objective values remain very close. This implies that although the objective of KMM (2.3) is convex, there are many "optimal" solutions (in terms of objective values) that minimize the discrepancy between the means in the RKHS. This also leads to different behaviours in the underling task (regression or classification,



(a) Projected gradient (b) FW $1/(t+1)$ step size (c) FW line search

Figure 2.2: Illustrating the training point weights produced by different KMM optimization methods. Without reweighting, least squares outputs the red line as the best linear fit. By adjusting the weights of the training points (as shown by the purple vertical bars), a new linear fit (blue line) can be found for better test set prediction. Notice that the three methods produce significantly different training weights (purple vertical bars), but their final objective values are very close.

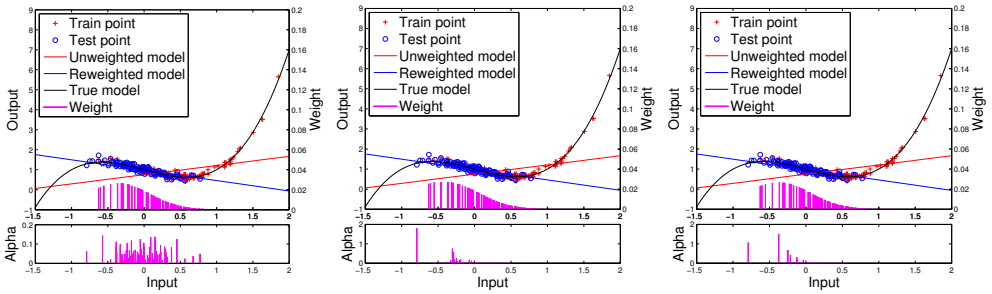
etc). No further criteria is proposed in the literature to distinguish which solution is preferable for KMM. For example, we include a red line and a blue line in each graph, showing the linear models learned by least squares from the unweighted (i.e., with uniform weights) and reweighted training set, respectively. The blue lines of the three KMM methods are slightly different from each other. More results can be found on benchmark datasets in Section 2.7.2. On the other hand, although KLIEP also demonstrates multiple solutions with close objective values, the final weights produced by the three KLIEP methods are very similar, as shown in Fig. 2.3. Note that Frank-Wolfe tends to find sparser solutions than projected gradient in general, which suggests the effectiveness of Frank-Wolfe in finding sparse solutions if they exist.

2.7.2 Benchmark Datasets

Next we applied the reweighting methods on some benchmark datasets from the libsvm[3] and delve[4] libraries to show their performance in correcting covariate shift on reweighted learning. All methods are run until convergence.

[3] www.csie.ntu.edu.tw/~cjlin/libsvm

[4] www.cs.toronto.edu/~delve/data/datasets.html



(a) Projected gradient (b) FW $1/(t+1)$ step size (c) FW line search

Figure 2.3: Illustrating the training point weights produced by different KLIEP optimization methods. These methods choose very different Gaussian centres (the values shown in the α graph at the bottom), but their training weights (purple vertical bars in the top graphs) are very similar to each other, obtaining very close final objective values. Note that the mixing coefficients α in (a) have a different scale for better visualization. The solutions obtained by Frank-Wolfe are sparser than that of projected gradient. The final solution using line search in (c) has only 7 Gaussian components, which is extremely sparse compared to all 128 possible Gaussian candidates. This shows the capability of Frank-Wolfe method in finding sparse solutions.

For each dataset, we first normalize the input features and the output to $[-1, 1]$ if it is significantly out of scale. We introduce a covariate shift by the following scheme. First we compute the first (with largest eigenvalue) principle component of the dataset. Let z_{max} and z_{min} be the maximum and minimum projected values along the principle direction and σ_z be the standard deviation of the projected values. In each trial, we first draw $m = 2000$ test points without replacement from the pool, with probability proportional to $\mathcal{N}(z_{max}, \sigma_z^2/4)$. After removing these test points from the dataset, we then draw $n = 5000$ training points without replacement from the pool, with probability proportional to $\mathcal{N}(z_{min}, \sigma_z^2/4)$. After computing the weights for the training points, we learn a linear model from the reweighted set and evaluate the model performance on the test set, using hinge loss for classification and squared loss for regression. We compare the performance of the reweighting methods with the unweighted method (i.e., with uniform weights for training points). For KMM(FW) we use $1/(t+1)$ as the step size, while for KLIEP(FW) we use the line

search step size, since these choices are faster in practice. In addition to the projected gradient solver KLIEP(PG), we also compare the KLIEP(FW) method with the implementation of the original authors KLIEP(100), where the "$b = 100$" trick is used for projected gradient when there are too many test instances (only randomly select 100 Gaussian centres; see Section 2.5.2 for more details).

Table 2.2 shows the test loss and Table 2.3 shows the runtime over 50 trials. First, note that the performance of KMM is not very reliable because (1) the kernel width used in the experiment is difficult to tune, as observed by others (Cortes et al., 2008) and (2) the solution to KMM tends to be sensitive to random initialization – i.e., different solutions may have similar objective values – which leads to unstable behaviour on the underlying task. Next, we can see that KLIEP(FW) is effective in correcting covariate shift. In most cases its test loss is smaller than unweighted learning as well as KLIEP(PG) on the test set. One possible reason is that KLIEP(PG) may fail to improve the objective on a step and stop prematurely. Comparing KLIEP(PG) and KLIEP(100), we can see that using all information from the test set will probably reduce test loss, although it may suffer from increased runtime. Finally, judging from Table 2.3, Frank-Wolfe is a very efficient algorithm compared to projected gradient in the current setting where sparse solutions are possible. KLIEP(100) is extremely fast but its task performance is so restrictive that it becomes impractical. We may apply values other than $b = 100$, but the best value of b is dataset-dependent and could be hard to identify. Instead, FW does not require parameter tuning and can utilize all Gaussians without computational issues.

2.8 Conclusion

We have proposed an efficient method that can be incorporated into two well-known covariate shift correction techniques, Kernel Mean Matching (KMM) and Kullback-Leibler Importance Estimation Procedure (KLIEP).

Table 2.2: Test losses for the different covariate shift correction methods over 50 trials (mean with standard error in parentheses). Weights are computed using either Frank-Wolfe (FW) or projected gradient (PG). Losses are normalized so that the mean loss of the unweighted method is 1.0. The upper half of the table considers classification datasets (hinge loss), while the lower half considers regression datasets (squared loss). The best method(s) according to the Wilcoxon signed-rank test at the 5% significance level is(are) shown in bold for each dataset.

Dataset	Unweighted	KLIEP(FW)	KLIEP(PG)	KLIEP(100)	KMM(FW)	KMM(PG)
a7a	1.000 (0.006)	0.978 (0.007)	0.998 (0.015)	0.999 (0.009)	0.968 (0.024)	**0.955** (0.023)
a8a	1.000 (0.006)	**0.971** (0.008)	1.002 (0.020)	0.997 (0.015)	**0.969** (0.027)	**0.959** (0.025)
a9a	1.000 (0.005)	**0.968** (0.010)	0.983 (0.023)	0.998 (0.012)	0.963 (0.025)	0.965 (0.023)
mushrooms	1.000 (0.077)	1.010 (0.078)	0.981 (0.072)	1.078 (0.109)	**0.800** (0.080)	0.923 (0.060)
cpusmall	1.000 (0.001)	**0.893** (0.023)	0.943 (0.010)	0.936 (0.016)	1.034 (0.034)	0.978 (0.023)
kin-8fh	**1.000** (0.003)	**1.014** (0.043)	4.462 (0.511)	5.872 (1.212)	7.686 (2.566)	3.594 (1.059)
kin-8fm	1.000 (0.004)	**0.738** (0.048)	5.060 (0.690)	5.329 (0.909)	3.028 (1.070)	1.786 (0.522)
kin-8nh	**1.000** (0.002)	**1.018** (0.040)	2.032 (0.329)	2.659 (0.454)	2.752 (0.398)	1.913 (0.370)
kin-8nm	1.000 (0.004)	**0.803** (0.017)	1.172 (0.112)	1.396 (0.219)	1.031 (0.093)	0.899 (0.055)

Table 2.3: Average runtime in seconds over 50 trials (mean with standard error in parenthesis). KLIEP(FW) is generally faster than KLIEP(PG). Although KLIEP(100) is extremely fast, its corresponding performance on the learning task is very restrictive. KMM(FW) is faster than KMM(PG) for every dataset.

Dataset	KLIEP(FW)	KLIEP(PG)	KLIEP(100)	KMM(FW)	KMM(PG)
a7a	6.22 (0.03)	5.29 (2.59)	0.07 (0.04)	1.32 (0.02)	1.76 (0.05)
a8a	6.25 (0.04)	7.17 (2.92)	0.04 (0.00)	1.31 (0.02)	1.78 (0.04)
a9a	6.28 (0.04)	4.39 (2.38)	0.06 (0.04)	1.30 (0.02)	1.77 (0.05)
mushrooms	4.19 (0.21)	23.90 (0.02)	0.89 (0.10)	1.18 (0.02)	1.97 (0.10)
cpusmall	1.24 (0.02)	3.18 (1.13)	0.08 (0.03)	1.12 (0.03)	3.32 (0.21)
kin-8fh	5.61 (0.21)	23.79 (0.02)	0.76 (0.13)	1.06 (0.02)	1.56 (0.06)
kin-8fm	5.60 (0.21)	23.78 (0.03)	0.60 (0.14)	1.07 (0.02)	1.53 (0.06)
kin-8nh	5.46 (0.30)	23.85 (0.02)	0.72 (0.13)	1.11 (0.02)	1.53 (0.06)
kin-8nm	5.69 (0.03)	23.76 (0.02)	0.73 (0.13)	1.08 (0.02)	1.55 (0.05)

Our approach is inspired by noticing a connection between kernel herding and KMM. By exploiting sparse solution structure, we apply the Frank-Wolfe algorithm to KMM and KLIEP. Since Frank-Wolfe is projection-free, its time complexity per iteration is in general smaller than the traditional projected gradient method. Our convergence and complexity analysis show that KMM(FW) has an advantage over the projected gradient method,

while KLIEP(FW) has runtime performance comparable to the projected gradient method. Our empirical studies show that Frank-Wolfe is very suitable and practical for finding optimal solutions and correcting covariate shift as the sample size grows.

In addition to the mixture of Gaussian parametrization, Frank-Wolfe can also be applied to different variants of KLIEP, or even other density-ratio estimation methods (Sugiyama et al., 2012b). As long as the objective function is convex and differentiable, and there is a compact convex set constraint, Frank-Wolfe is applicable and potentially more efficient than the projected gradient method. We may gain additional benefits by using more sophisticated Frank-Wolfe variants (Lacoste-Julien and Jaggi, 2015), or their stochastic counterparts (Goldfarb et al., 2017) to handle large-scale data. These suggest that Frank-Wolfe is a practical and powerful tool for learning density-ratio for covariate shift problems.

Chapter 3

Domain Aggregation Networks for Multi-Source Domain Adaptation

3.1 Overview

In many real-world applications, we want to exploit multiple source datasets to build a model for a different but related target dataset. Despite the recent empirical success, most existing research has used ad-hoc methods to combine multiple sources, leading to a gap between theory and practice. In this chapter, we develop a finite-sample generalization bound based on domain discrepancy and accordingly propose a theoretically justified optimization procedure. Our algorithm, Domain AggRegation Network (DARN), can automatically and dynamically balance between including more data to increase effective sample size and excluding irrelevant data to avoid negative effects during training. We find that DARN can significantly outperform the state-of-the-art alternatives on multiple real-world tasks, including digit/object recognition and sentiment analysis.

3.2 Introduction

Many machine learning algorithms assume the learned predictor will be tested on the data from the same distribution as the training data. This assumption, although reasonable, is not necessarily true for many real-

world applications. For example, patients from one hospital may have a different distribution of gender, height and weight from another hospital. Consequently, a diagnostic model constructed at one location may not be directly applicable to another location without proper adjustment. The situation becomes even more challenging when we want to use data from *multiple source domains* to build a model for a *target domain*, as this requires deciding, e.g., how to select the relevant source domains and how to effectively aggregate these domains when training complex models like neural networks. Including irrelevant data from certain source domains can severely reduce the performance on the target domain.

To address this problem, researchers have been exploring methods of *transfer learning* (Pan and Yang, 2009) or *domain adaptation* (Mansour et al., 2009a; Cortes et al., 2019), where a model is trained based on labelled source data and unlabelled target data. Most existing works have focused on single-source to single-target ("one-to-one") domain adaptation, using different assumptions such as covariate shift (Shimodaira, 2000; Gretton et al., 2009; Sugiyama and Kawanabe, 2012) or concept drift (Jiang and Zhai, 2007; Gama et al., 2014). When dealing with multiple source domains, one may attempt to directly use these approaches by combining all source data into a large joint dataset and then apply one-to-one adaptation. This naïve aggregation method will often fail as not all source domains are equally important when transferring to a specific target domain. There are some works on multi-source to single-target adaptation. Although some of them are theoretically motivated with cross-domain generalization bounds, they either use ad-hoc aggregation rules when developing practical algorithms (Zhao et al., 2018; Li et al., 2018) or lack finite-sample analysis (Mansour et al., 2009b,c; Hoffman et al., 2018a). This leaves a gap between the theory for multi-source adaptation and practical algorithm for domain aggregation.

This research has three contributions: First, we develop a finite-sample cross-domain generalization bound for multi-source adaptation, which is

based on the discrepancy introduced by Cortes et al. (2019). We show that in order to improve performance on the specific target domain, there is a trade-off between including all source domains to increase effective sample size, versus excluding less relevant source domains to avoid negative outcomes. Second, motivated by our theory and domain adversarial method (Ganin and Lempitsky, 2015; Ganin et al., 2016), we propose Domain AggRegation Network (DARN), which can dynamically adjust the importance weights of the source domains during the course of training. Unlike previous works, our aggregation scheme (Eq. (3.7)), which itself is of independent interest in some other contexts, is a direct optimization of our generalization upper bound (Eq. (3.4)) without resorting to surrogates. Third, our experiments on digit/object recognition and sentiment analysis show that DARN can significantly outperform state-of-the-art methods.

This chapter is organized as follows. Section 3.3 introduces necessary background on one-to-one adaptation based on discrepancy. Section 3.4 elaborates our theoretical analysis and the corresponding algorithm deployment. Section 3.5 discusses about related approaches in the literature, highlighting the key differences to our method. Section 3.6 empirically compares the performance of the proposed method to other alternatives. Finally, Section 3.7 concludes this work.

3.3 Background on Cross-domain Generalization

This section provides background from previous work on one-to-one domain adaptation (Cortes et al., 2019). Let $\mathcal{X}$ be the input space, $\mathcal{Y} \subseteq \mathbb{R}$ be output space and $\mathcal{H} \subseteq \{h : \mathcal{X} \mapsto \mathcal{Y}\}$ be a hypothesis class. A loss function $L : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ is μ-admissible[1] if

$$\forall y, y', y'' \in \mathcal{Y} \quad |L(y', y) - L(y'', y)| \leqslant \mu\, |y' - y''|. \tag{3.1}$$

[1]For example, the common L_p losses are μ-admissible (Cortes et al., 2019, Lemma 23).

The *discrepancy* (Mansour et al., 2009a) between two distributions P, Q over $\mathcal{X}$ is defined as

$$\begin{aligned} \operatorname{disc}(P, Q) &= \max_{h, h' \in \mathcal{H}} |\mathcal{L}_P(h, h') - \mathcal{L}_Q(h, h')| \\ \text{where} \quad \mathcal{L}_P(h, h') &= \mathbb{E}_{x \sim P}[L(h(x), h'(x))]. \end{aligned} \tag{3.2}$$

This quantity can be computed or approximated empirically given samples from both distributions. For classification problems with 0-1 loss, it reduces to the well-known $d_{\mathcal{A}}$-distance (Kifer et al., 2004; Blitzer et al., 2008; Ben-David et al., 2010a) and can be approximated using a domain-classifier loss w.r.t. $\mathcal{H}$ (Zhao et al., 2018; Ben-David et al., 2007, Sec.4), while for regression problems with L_2 loss, it reduces to the maximum eigenvalue (Cortes and Mohri, 2011, Sec.5); see Section 3.4.2 for more details. For two domains $(P, f_P), (Q, f_Q)$ where $f_P, f_Q : \mathcal{X} \mapsto \mathcal{Y}$ are the corresponding labeling functions, we have the following cross-domain generalization bound:

Theorem 1 (Proposition 5 & 8, Cortes et al. (2019))**.** *Let $\mathfrak{R}_m(\mathcal{H})$ be the Rademacher complexity of $\mathcal{H}$ given sample size m, $\mathcal{H}_Q = \{x \mapsto L(h(x), f_Q(x)) : h \in \mathcal{H}\}$ be the set of functions mapping x to its loss w.r.t. f_Q and $\mathcal{H}$,*

$$\eta_{\mathcal{H}} = \mu \times \min_{h \in \mathcal{H}} \left(\max_{x \in \operatorname{supp}(\widehat{P})} |f_P(x) - h(x)| + \max_{x \in \operatorname{supp}(\widehat{Q})} |f_Q(x) - h(x)| \right), \tag{3.3}$$

be a constant measuring how well $\mathcal{H}$ can fit the true models where $\operatorname{supp}(\widehat{P})$ is the support of the empirical distribution $\widehat{P}$ (using the μ from Eq. (3.1)), and $M_Q = \sup_{x \in \mathcal{X}, h \in \mathcal{H}} L(\, h(x),\ f_Q(x)\,)$ be the upper bound on loss for Q. Given $\widehat{Q}$ with m points sampled iid from Q labelled according to f_Q, for $\delta \in (0, 1)$ w.p. at least $1 - \delta$, the following holds for all $h \in \mathcal{H}$,

$$\mathcal{L}_P(h, f_P) \;\leqslant\; \mathcal{L}_{\widehat{Q}}(h, f_Q) + \operatorname{disc}(P, Q) \;+ 2\mathfrak{R}_m(\mathcal{H}_Q) + \eta_{\mathcal{H}} + M_Q \sqrt{\frac{\log(1/\delta)}{2m}}.$$

This theorem provides a way to generalize across domains when we have sample $\widehat{Q}$ labelled according to f_Q and an unlabelled sample $\widehat{P}$. The first term is the usual loss function for the sample $\widehat{Q}$, while the second term

$\mathrm{disc}(P, Q)$ can be estimated based on the unlabelled data $\widehat{Q}, \widehat{P}$. $\eta_{\mathcal{H}}$ measures how well the model family $\mathcal{H}$ can fit the examples from the datasets, and it is not controllable once $\mathcal{H}$ is given. The final term, as a function of sample size m, determines the convergence speed.

3.4 Domain Aggregation Network

3.4.1 Theory

Suppose we are given k source domains $\{(S_i, f_{S_i}) : i \in [k] := \{1, 2, \ldots, k\}\}$ and a target domain (T, f_T) where S_i, T are distributions over $\mathcal{X}$ and $f_{S_i}, f_T : \mathcal{X} \mapsto \mathcal{Y}$ are their respective labelling functions. For simplicity, assume that each sample $\widehat{S}_i$ has m points, drawn iid from S_i and labelled according to f_{S_i}. We are also given m unlabelled points $\widehat{T}$ drawn iid from T. We want to leverage all source domains' information to learn a model $h \in \mathcal{H}$ minimizing $\mathcal{L}_T(h, f_T)$.

One naïve approach could be to combine all the source domains into a large joint dataset and conduct one-to-one adaptation to the target domain using Theorem 1. However, including data from irrelevant or even adversarial domains is likely to jeopardize the performance on the target domain, which is sometimes referred to as *negative transfer* (Pan and Yang, 2009). Moreover, as certain source domains may be more similar or relevant to the target domain than the others, it makes more sense to adjust their importance according to their utilities. We propose to find domain weights $\alpha_i \geqslant 0$ such that $\sum_{i=1}^{k} \alpha_i = 1$ to achieve this. Our main theorem below sheds some light on how we should combine the source domains (the proof is provided in Appendix A.1):

Theorem 2. *Given* k *source domains datasets* $\{(x_j^{(i)}, y_j^{(i)}) : i \in [k], j \in [m]\}$ *with* m *iid examples each where* $\widehat{S}_i = \{x_j^{(i)}\}$ *and* $y_j^{(i)} = f_{S_i}(x_j^{(i)})$*, for any* $\boldsymbol{\alpha} \in \Delta = \{\boldsymbol{\alpha} : \alpha_i \geqslant 0, \sum_i \alpha_i = 1\}$ *and* $\delta \in (0, 1)$*, w.p. at least* $1 - \delta$*, the following holds for all*

$h \in \mathcal{H}$,

$$\begin{aligned}\mathcal{L}_T(h, f_T) \leqslant \sum_i \alpha_i \left[\mathcal{L}_{\widehat{S}_i}(h, f_{S_i}) + \mathrm{disc}(T, S_i) + 2\mathfrak{R}_m(\mathcal{H}_{S_i}) + \eta_{\mathcal{H},i}\right] \\ + \|\boldsymbol{\alpha}\|_2 M_S \sqrt{\frac{\log(1/\delta)}{2m}},\end{aligned} \tag{3.4}$$

where $\mathcal{H}_{S_i} = \{x \mapsto L(h(x), f_{S_i}(x)) : h \in \mathcal{H}\}$ *is the set of functions mapping* x *to the corresponding loss,*

$$\eta_{\mathcal{H},i} = \mu \times \min_{h \in \mathcal{H}} \left(\max_{x \in \mathrm{supp}(\widehat{T})} |f_T(x) - h(x)| + \max_{x \in \mathrm{supp}(\widehat{S}_i)} |f_{S_i}(x) - h(x)| \right),$$

and $M_S = \sup_{i \in [k], x \in \mathcal{X}, h \in \mathcal{H}} L(h(x), f_{S_i}(x))$ *is the upper bound on loss on the source domains.*

There are several remarks.

- For the last term of the bound, moving $\|\boldsymbol{\alpha}\|_2$ into the square root will give us $\frac{m}{\|\boldsymbol{\alpha}\|_2^2}$, which serves as the *effective sample size*. If $\boldsymbol{\alpha}$ is uniform (i.e., $[1/k, \ldots, 1/k]^\top$), the effective sample size is km; if $\boldsymbol{\alpha}$ is one-hot, we effectively only have m points from exactly one domain.
- Let $g_{h,i} := \mathcal{L}_{\widehat{S}_i}(h, f_{S_i}) + \mathrm{disc}(T, S_i)$. Small $g_{h,i}$ indicates that we can achieve small loss on domain S_i, and it is similar to the target domain (i.e., small $\mathrm{disc}(T, S_i)$, estimated from $\widehat{T}, \widehat{S}_i$). We may want to focus on S_i by setting α_i close to 1, but this will reduce the effective sample size. Therefore, we have to trade-off between the terms in this bound by choosing a proper $\boldsymbol{\alpha}$.
- When S_i and T are only partially overlapped, it might be difficult to find a suitable α_i. In such cases, the discrepancy could be large because the performance gap on the two domains can be big (see Eq. (3.2)). Then it would be important for the model to learn certain overlapping representations to control the bound. We can also artificially split the source domains into smaller datasets (e.g., by clustering) then apply our method on the finer scale.
- $\mathfrak{R}_m(\mathcal{H}_{S_i})$ determines how expressive the model family $\mathcal{H}$ is w.r.t. the source data S_i. Although it can be estimated from samples (Bartlett

and Mendelson, 2002, Theorem 11), the computation is non-trivial for a model family like neural networks. $\eta_{\mathcal{H},i}$ is assumed to be small, which is a necessary condition for success adaptation (Ben-David et al., 2010b, Theorem 2), and estimating $\eta_{\mathcal{H},i}$ is impossible without labelled target data. As removing these two terms does not seem to be empirically significant, the following analysis ignores them for simplicity.

Before we proceed to develop an algorithm based on the theorem, we compare Eq. (3.4) to existing finite-sample bounds. The bound of Blitzer et al. (2008, Theorem 3) is informative when we have access to a small set of *labelled* target examples. In such cases, we can improve our bound by using this small labelled target set as an additional source domain S_{k+1}. The bound of Zhao et al. (2018, Theorem 2) is based on the $d_{\mathcal{A}}$-distance, which is a special case of disc in our bound. Moreover, Eq. (3.4) uses sample-based Rademacher complexity, which is generally tighter than other complexity measures such as VC-dimension (Bartlett and Mendelson, 2002; Koltchinskii et al., 2002; Bousquet et al., 2003). Peng et al. (2019, Theorem 1) provide a bound based on binary classification, which depends on the *unknown* target minimizer. In comparison, our bound uses the *attainable* source loss. Moreover, we relate our theorem to the optimization procedure, as we show next.

3.4.2 Algorithm

In this section, we develop a practical algorithm, Domain AggRegation Network (DARN), based on Theorem 2. Ignoring the constants, we would like to minimize the upper bound of Eq. (3.4):

$$\min_{h\in\mathcal{H}} \min_{\boldsymbol{\alpha}\in\Delta} \quad U_h(\boldsymbol{\alpha}) = \langle \mathbf{g}_h, \boldsymbol{\alpha}\rangle + \tau\|\boldsymbol{\alpha}\|_2 \tag{3.5}$$

where $\mathbf{g}_h = [g_{h,1}, \ldots, g_{h,k}]^\top$ and $\tau > 0$ is a hyper-parameter. If we can solve the inner minimization exactly given h, then we can treat the optimal $\boldsymbol{\alpha}^*(h)$ as a function of h and solve the outer minimization over h effectively. In

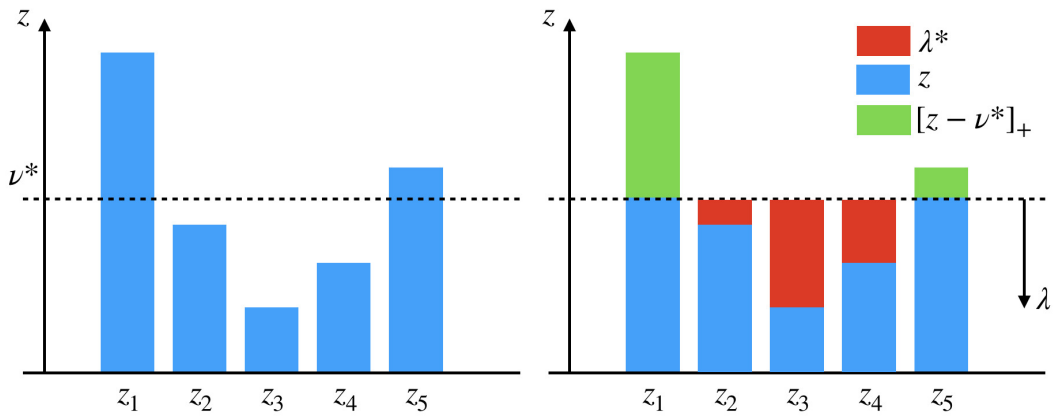


Figure 3.1: Optimal solution to Eq. (3.6) (best viewed in color). λ^* compensates what is below the ν^*, while ν^* is chosen such that the vector of the green bars has L_2 norm of 1.

the following, we show how to achieve this.

Given $\mathbf{g}_h$, the inner minimization can be reformulated as a second-order cone programming problem, but it has no closed-form solution due to the $\tau\|\boldsymbol{\alpha}\|_2$ term. Consider the following problem ($\mathbf{z} = -\mathbf{g}_h/\tau$ recovers the inner minimization):

$$\min_{\boldsymbol{\alpha}\in\Delta} \quad -\langle \mathbf{z}, \boldsymbol{\alpha}\rangle + \|\boldsymbol{\alpha}\|_2 \tag{3.6}$$

The Lagrangian for its dual problem is

$$\Lambda(\boldsymbol{\alpha}, \boldsymbol{\lambda}, \nu) = -\mathbf{z}^\top\boldsymbol{\alpha} + \|\boldsymbol{\alpha}\|_2 - \boldsymbol{\lambda}^\top\boldsymbol{\alpha} + \nu(\mathbf{1}^\top\boldsymbol{\alpha} - 1),$$

with $\nu \in \mathbb{R}, \boldsymbol{\lambda} \geq 0$. Taking the derivative w.r.t. $\boldsymbol{\alpha}$ and setting it to zero gives

$$\frac{\partial\Lambda}{\partial\boldsymbol{\alpha}} = -\mathbf{z} + \frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|_2} - \boldsymbol{\lambda} + \nu\mathbf{1} = 0 \Rightarrow \frac{\boldsymbol{\alpha}^*}{\|\boldsymbol{\alpha}^*\|_2} = \mathbf{z} - \nu\mathbf{1} + \boldsymbol{\lambda}.$$

Note that $\boldsymbol{\alpha} \neq \mathbf{0}$ so we have the constraint $\|\mathbf{z} - \nu\mathbf{1} + \boldsymbol{\lambda}\|_2 = 1$. Using this $\boldsymbol{\alpha}^*$ in Λ gives the following dual problem

$$\max_{\nu,\boldsymbol{\lambda}} \quad -\nu \qquad \text{s.t.} \quad \|\mathbf{z} - \nu\mathbf{1} + \boldsymbol{\lambda}\|_2 = 1 \quad \text{and} \quad \boldsymbol{\lambda} \geq 0$$

We would like to *decrease* ν as much as possible, and the best we can attain is the ν^* satisfying $\|[\mathbf{z} - \nu^*\mathbf{1}]_+\|_2 = 1$ where $[\mathbf{v}]_+ = \max(\mathbf{0}, \mathbf{v})$. In this case, the optimal $\boldsymbol{\lambda}^*$ can be attained as $\lambda_i^* = 0$ if $z_i - \nu^* > 0$, otherwise $\lambda_i^* = \nu^* - z_i$; see Fig. 3.1. Although we do not have a closed-form expression

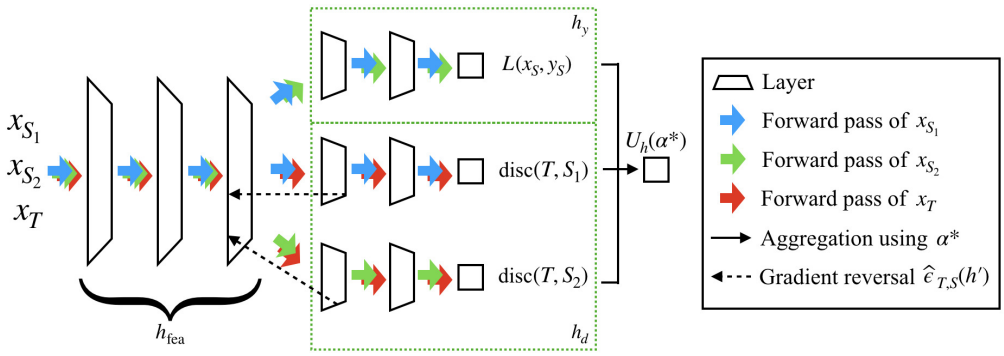


Figure 3.2: DARN architecture with two source domains (best viewed in color). Mini-batches of x_{S_i} and x_T are fed to the network. x_{S_i} will go through the classification/regression path on the upper h_y box, while all x will go through to the corresponding discrepancies (in the lower h_d box). The gradients from the discrepancies will be reverted during backpropagation.

for the optimal ν^*, we can use binary search to find it, starting from the interval $[z_{\min} - 1, z_{\max}]$ where $z_{\min}, z_{\max}$ are the minimum and maximum of $\mathbf{z}$ respectively. Then we can recover the primal solution as

$$\boldsymbol{\alpha}^* = [\mathbf{z} - \nu^* \mathbf{1}]_+ / \|[\mathbf{z} - \nu^* \mathbf{1}]_+\|_1. \tag{3.7}$$

Eq. (3.7) gives rise to a new way to project any vector $\mathbf{z}$ to the probability simplex, which is of independent interest and may be used in some other contexts.[2] It resembles the standard projection onto the simplex based on *squared* Euclidean distance (Duchi et al., 2008, Eq.(3)). One subtle but crucial difference is that Eq. (3.6) uses $\|\boldsymbol{\alpha}\|_2$ instead of $\|\boldsymbol{\alpha}\|_2^2$. Recall that $\mathbf{z} = -\mathbf{g}_h/\tau$. Here τ can be interpreted as a temperature parameter. On one hand, if $\tau \gg 0$, all $\mathbf{z}$ will have similar values and thus the optimal ν^* will be close to $z_{\max}$ and $\boldsymbol{\alpha}^*$ will be close to uniform. On the other hand, as $\tau \to 0$, $z_{\max}$ will stand out from the rest z_i and eventually $\nu^* = z_{\max} - 1$. This means the $g_{h,i}$ corresponding to the $z_{\max}$ is small enough so we focus solely on this domain and ignore all other domains (even though this will reduce effective sample size as discussed in Section 3.4.1).

[2]For example, if $\mathbf{z}$ are the logits of the final classification layer of a neural network, this projection provides another way to produce class probabilities similar to the softmax transformation.

We can optimize our objective and train a neural network h using gradient-based optimizer. The optimal $\boldsymbol{\alpha}^*(h)$ is a function of h and we can backprop through it. To facilitate the gradient computation and show that we can efficiently backprop through the projection of Eq. (3.7), we derive the Jacobian $J = \partial\boldsymbol{\alpha}/\partial\mathbf{z}$ in Appendix A.2. This ensures effective end-to-end training.

Now we elaborate on how to compute $g_{h,i}$. The $\mathcal{L}_{\widehat{S}_i}(h, f_{S_i})$ is the task loss. The $\mathrm{disc}(T, S_i)$ depends on whether the task is classification or regression. For classification, disc coincides with the $d_{\mathcal{A}}$-distance, so we use the domain classification loss (Ben-David et al., 2007; Zhao et al., 2018):

$$\mathrm{disc}(\widehat{T}, \widehat{S}_i) = 2\left(1 - \min_{h_d} \widehat{\epsilon}_{T,S_i}(h_d)\right)$$
$$\widehat{\epsilon}_{T,S_i}(h_d) = \frac{1}{2m}\sum_{i=1}^{2m} \left|h_d(x) - \delta_{x\in\widehat{T}}\right|$$

where $\widehat{\epsilon}_{T,S_i}(h_d)$ is the sample domain classification loss of a domain classifier $h_d : \mathcal{X} \mapsto \{0, 1\}$. This minimization over h_d will become maximization once we move it outside of the disc due to the minus sign. Then our objective consists of $\min_h$ and $\max_{h_d}$, which resembles adversarial training (Goodfellow et al., 2014): learning a task classifier h to minimize loss and a domain classifier h_d to maximize domain confusion. More specifically, if we decompose the neural network h into a feature extractor h_{fea} and a label predictor h_y (i.e., $h(x) = (h_y \circ h_{\mathrm{fea}})(x)$), we can learn a domain-classifier h_d on top of h_{fea} to classify $h_{\mathrm{fea}}(x)$ between S_i and T as a binary classification problem, where the domain itself is the label (see Fig. 3.2). To achieve this, we use the logistic loss to approximate $\widehat{\epsilon}_{T,S_i}(h_d)$ and apply the gradient reversal layer (Ganin and Lempitsky, 2015; Ganin et al., 2016) when optimizing disc through backpropagation.

If we are solving regression problems with L_2 loss, then $\mathrm{disc}(\widehat{T}, \widehat{S}_i) = \|M_T - M_{S_i}\|_2$ is the largest eigenvalue (in magnitude) of the difference of

two matrices (Mansour et al., 2009a; Cortes and Mohri, 2011, Sec.5)

$$M_T = \frac{1}{m}\sum_j h_{\text{fea}}(x_j^{(T)})h_{\text{fea}}^\top(x_j^{(T)})$$
$$M_{S_i} = \frac{1}{m}\sum_j h_{\text{fea}}(x_j^{(i)})h_{\text{fea}}^\top(x_j^{(i)}),$$

which can be conveniently approximated using mini-batches and a few steps of power iteration. The overall algorithm is summarized in Algorithm 1.

3.4.3 Complexity Analysis

Here we analyze the time and space complexities of the algorithm in each iteration. Similar to MDAN (Zhao et al., 2018), in each gradient step, we need to compute the task loss $\mathcal{L}_{\hat{S}_i}(h, f_{S_i})$ and the $\text{disc}(T, S_i)$ (or $d_{\mathcal{A}}$-distance) using mini-batches from each source domain $i \in [k]$. The question is whether one can maintain the $O(k)$ complexity given that we need to compute the weights using Eq. (3.7) and backprop through it. For the forward computation of $\boldsymbol{\alpha}^*$, in order to compute the threshold ν^* to the $\epsilon > 0$ relative precision, the binary search will cost $O(k \log(1/\epsilon))$. As for the backward pass of gradient computation, according to our calculation in Appendix A.2, the Jacobian $J = \partial\boldsymbol{\alpha}/\partial\mathbf{z}$ has a concise form, meaning that it is possible to compute the matrix-vector product $J\mathbf{v}$ for a given vector $\mathbf{v}$ in $O(k)$ time and space. Therefore, our space complexity is the same as MDAN and our time complexity is slightly slower by a factor of $\log(1/\epsilon)$. In comparison, the time complexity for MDMN (Li et al., 2018) is $O(k^2)$ because it requires computing the pairwise weights within the k source domains. When there are a lot of source domains, MDMN will be noticeably slower than MDAN and DARN.

3.5 Related Work

The idea of utilizing data from the source domain (S, f_S) to train a model for a different but related target domain (T, f_T) has been explored exten-

Algorithm 1 Domain Aggregation Network

Input:

- k labelled source datasets $\{(x_j^{(i)}, y_j^{(i)}) : i \in [k], j \in [m]\}$
- One unlabelled target dataset $\{x_j^{(T)} : j \in [m]\}$
- Temperature parameter $\tau > 0$
- Optimizer learning rate $\eta > 0$

1: Initialize a neural network h with feature extractor h_{fea}, one label prediction head h_y and k domain classification heads $\{h_{d,i} : i \in [k]\}$, as in Fig. 3.2

2: **repeat**

3: Sample a mini-batch of size B for each dataset

4: Compute the losses for the source mini-batches

$$\widehat{\mathcal{L}}_{S_i}(h) \leftarrow \frac{1}{B}\sum_{j=1}^{B} L\left[(h_y \circ h_{fea})(x_j^{(i)}),\ y_j^{(i)}\right] \tag{3.8}$$

5: Compute the sample discrepancies $\widehat{disc}(T, S_i)$

- If classification, compute as

$$\frac{1}{2B}\sum_{j=1}^{B}\left[L_d\left((h_{d,i} \circ h_{fea}^r)(x_j^{(T)}), 1\right) + L_d\left((h_{d,i} \circ h_{fea}^r)(x_j^{(i)}), 0\right)\right] \tag{3.9}$$

where L_d is the domain classification loss and $h_{fea}^r = r \circ h_{fea}$ with r being the GRL

- If regression, compute as

$$\|M_T^r - M_{S_i}^r\|_2 \tag{3.10}$$

$$\text{where } M_T^r = \frac{1}{B}\sum_{j=1}^{B} h_{fea}^r(x_j^{(T)}) h_{fea}^r(x_j^{(T)})^\top$$

$$\text{and } M_{S_i}^r = \frac{1}{B}\sum_{j=1}^{B} h_{fea}^r(x_j^{(i)}) h_{fea}^r(x_j^{(i)})^\top$$

6: With $\widehat{g}_{h,i} = \widehat{\mathcal{L}}_{S_i}(h) + \widehat{disc}(T, S_i)$, compute the optimal weights $\boldsymbol{\alpha}^*(h)$ using Eq. (3.7) and temperature τ

7: Compute the objective value $U_h = \langle \widehat{\mathbf{g}}_h, \boldsymbol{\alpha}^*(h) \rangle + \tau \|\boldsymbol{\alpha}^*(h)\|_2$ and take a gradient step $h \leftarrow h - \eta \frac{\partial U_h}{\partial h}$

8: **until** convergence

9: **return** h

sively for the last decade using different assumptions (Pan and Yang, 2009; Zhang et al., 2015). For instance, the covariate shift setting (Shimodaira, 2000; Gretton et al., 2009; Sugiyama and Kawanabe, 2012; Wen et al., 2014) assumes $S \neq T$ but $f_S = f_T$, while concept drift (Jiang and Zhai, 2007;

Gama et al., 2014) assumes $S = T$ but $f_S \neq f_T$. More specifically, both domain discrepancy and hypothesis class contribute to the adaptation performance (Ben-David et al., 2010b).

Finding a domain-invariant feature space by minimizing a distance measure is common practice in domain adaptation, especially for training neural networks. Tzeng et al. (2017) provided a comprehensive framework that subsumes several prior efforts on learning shared representations across domains (Tzeng et al., 2015; Ganin et al., 2016). DARN uses adversarial domain classifier and the gradient reversal trick from Ganin et al. (2016). Instead of proposing a new loss for each pair of the source and target domains, one of our contributions is the aggregation technique of computing the mixing coefficients $\boldsymbol{\alpha}$, which is derived from theoretical guarantees. When dealing with multiple source domains, our aggregation method can certainly be applied to other forms of discrepancies such as MMD (Gretton et al., 2012; Long et al., 2015, 2016), and other model architectures such as Domain Separation Network (Bousmalis et al., 2016), cycle-consistent model (Hoffman et al., 2018b), class-dependent adversarial domain classifier (Pei et al., 2018) and Known Unknown Discrimination (Schoenauer-Sebag et al., 2019).

Our work focuses on multi-source to single-target adaptation, which has been investigated in the literature. Sun et al. (2011) developed a generalization bound but resorted to heuristic algorithms to adjust distribution shifts. Zhao et al. (2018) proposed a certain ad-hoc scheme for the combination coefficients $\boldsymbol{\alpha}$, which, unlike ours, are not theoretically justified. Multiple Domain Matching Network (MDMN) (Li et al., 2018) computes domain similarities not only between the source and target domains but also within the source domain themselves based on Wasserstein-like measure. Calculating such pairwise weights can be computationally demanding when we have a lot of source domains. Their bound requires additional smooth assumptions on the labelling functions f_{S_i}, f_T, and is not a finite-sample bound, as opposed to ours. As for the actual algorithm, they also

use ad-hoc coefficients $\boldsymbol{\alpha}$ without theoretical justification. Mansour et al. (2009b,c) consider multi-source adaptation where $T = \sum_i \beta_i S_i$ is a convex mixture of source distributions with some weights β_i. Our analysis does not require this assumption. Hoffman et al. (2018a) provides similar guarantees with different assumptions, but unlike ours, their bounds are not finite-sample bounds.

3.6 Experimental Evaluation

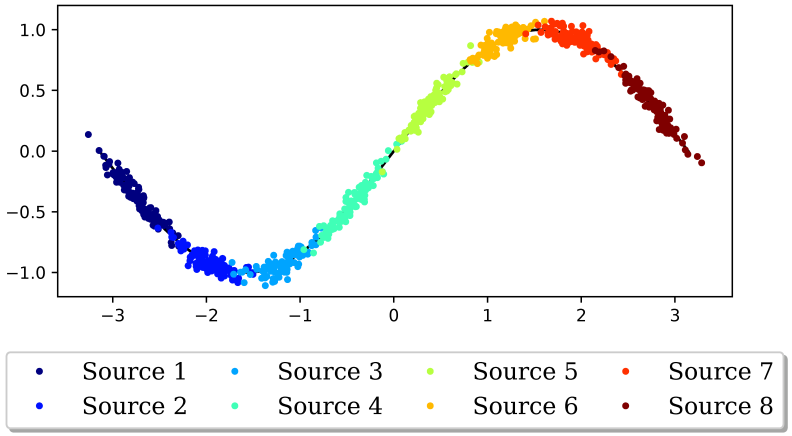
In this section, we demonstrate some of the key properties of DARN on a synthetic regression problem, then compare DARN to several state-of-the-art methods on multiple challenging real-world tasks. Additional experiment details can be found in Appendix A.3.

3.6.1 Regression on Synthetic Data

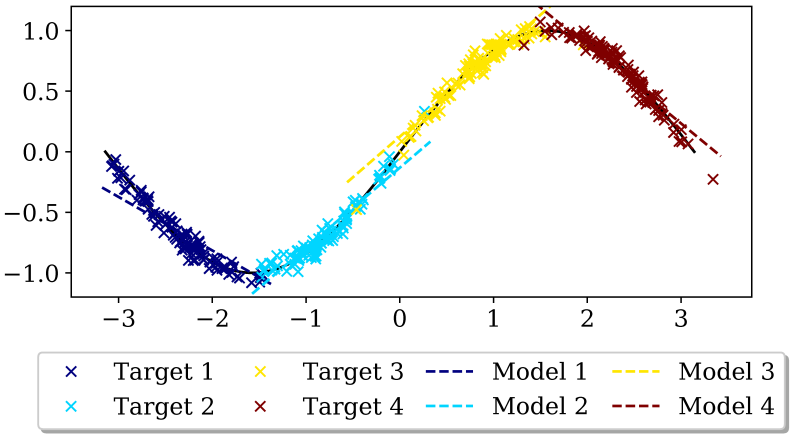
Setup. We construct eight source domains that evenly cover the sin function on $[-\pi, \pi]$ (see Fig. 3.3a). Similarly, we construct four target domains on the same region (see Fig. 3.3b). Each source/target domain has 100 data points.

We use labelled source data and unlabelled target data for learning. We take on one target domain at a time, and learn a linear model from *all* eight source domains with MSE loss. The goal is to see whether DARN can focus on the relevant source domains and learn a linear model that can perform well on the target domain.

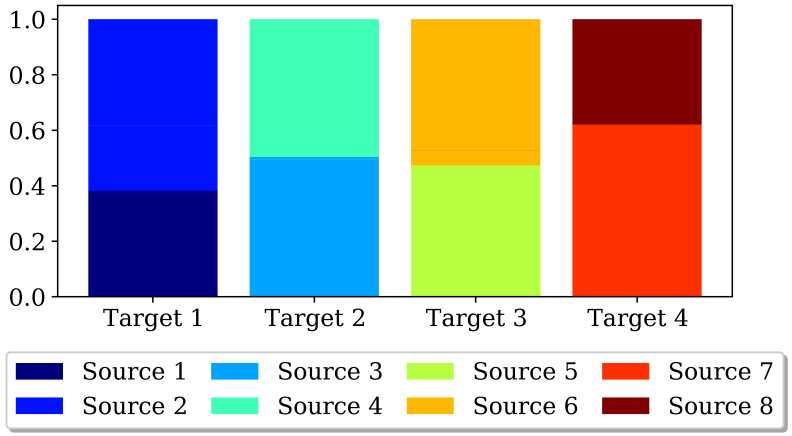
Results and Analysis. First, Fig. 3.3b shows the learned models. The linear models can fit the target data very well. This shows that DARN can learn a meaningful model for a specific target domain, using only labelled source data and unlabelled target data. Second, Fig. 3.3c shows the source domain weights (the $\boldsymbol{\alpha}$) for each target domain after training. The weight colors correspond to the colors in Fig. 3.3a. Noticeably DARN can focus well on the respective relevant source domains for each target domain and



(a) Source domains



(b) Target domains & learned models



(c) Domain weights (α values)

Figure 3.3: Regression experiment (best viewed in color).

ignore the rest (with zero weights). Note that these domain weights are automatically learned during the training of the model.

3.6.2 Digit Recognition

Table 3.1: Classification accuracy (%) of the target digit datasets. Mean and standard error over 20 runs. The best method (excluding TAR) based on one-sided Wilcoxon signed-rank test at the 5% significance level is shown in bold for each domain.

Method	MNIST	MNIST-M	SVHN	Synth	Avg.
SRC	96.78 ± 0.08	60.80 ± 0.21	68.99 ± 0.69	84.09 ± 0.27	77.66 ± 0.14
DANN	96.41 ± 0.13	60.10 ± 0.27	70.19 ± 1.30	83.83 ± 0.25	77.63 ± 0.35
M3SDA	96.95 ± 0.06	65.03 ± 0.80	71.66 ± 1.16	80.12 ± 0.56	78.44 ± 0.36
MDAN	97.10 ± 0.10	64.09 ± 0.31	77.72 ± 0.60	85.52 ± 0.19	81.11 ± 0.21
MDMN	97.15 ± 0.09	64.34 ± 0.27	76.43 ± 0.48	85.80 ± 0.21	80.93 ± 0.16
DARN	**98.09** ± 0.03	**67.06** ± 0.14	**81.58** ± 0.14	**86.79** ± 0.09	**83.38** ± 0.06
TAR	99.02 ± 0.02	94.66 ± 0.10	87.40 ± 0.17	96.90 ± 0.09	94.49 ± 0.07

Setup. Following previous works (Ganin et al., 2016; Zhao et al., 2018), we use the four digit recognition datasets in this experiment (MNIST, MNIST-M, SVHN and Synth). MNIST is a well-known gray-scale images for digit recognition, and MINST-M (Ganin and Lempitsky, 2015) is a variant where the black and white pixels are masked with color patches. Street View House Number (SVHN) (Netzer et al., 2011) is a standard digit dataset taken from house numbers in Google Street View images. Synthetic Digits (Synth) (Ganin and Lempitsky, 2015) is a synthetic dataset that mimic SVHN using various transformations. One of the four datasets is chosen as unlabelled target domain in turn and the other three are used as labelled source domains.

Baselines. We compare DARN to several baselines and state-of-the-art methods. There are many approaches in the literature dealing with single-source to single-target adaption. Since our focus is on the *multi-source* setting, we mainly compare DARN to the most relevant methods that utilize multiple sources for adaptation. (1) The **SRC** (for source) method uses only labelled source data to train the model. It merges all available source examples to form a large dataset to perform training without adaptation.

(2) **TAR** (for target) is another baseline that only uses labelled target data. It serves as performance upper bound as if we had access to the true label of the target data. (3) Domain Adversarial Neural Network (**DANN**) (Ganin et al., 2016) is similar to our method in that we both use adversarial training objectives. Here we follow the previous protocol (Zhao et al., 2018) and merge all source data to form a large joint source dataset of km instances for DANN. (4) Moment Matching for Multi-Source Domain Adaptation (**M3SDA**) (Peng et al., 2019) is a recent state-of-the-art method that combines moment matching and maximizing classifier discrepancy (Saito et al., 2018). We use their public code with a few necessary adjustments (change classification head based on the number of classes etc.) (5) Multisource Domain Adversarial Network (**MDAN**) (Zhao et al., 2018) resembles our method in that we both dynamically assign each source domain an importance weight during training. However, unlike ours, their weights are not theoretically justified. We use the soft version from their code since it is reported to perform better than the hard version. (6) Multiple Domain Matching Network (**MDMN**) (Li et al., 2018) computes weights not only between source and target domains but also within source domain themselves. We use their code of computing weights in our implementation. All the methods are applied to the same neural network structure to ensure fair comparison.

Results and Analysis. Table 3.1 shows the classification accuracy of each target dataset. The last column is the average accuracy of four domains, and the standard errors are calculated based on 20 runs. (1) Most methods can consistently outperform SRC, which has no adaptation. This shows the improvement when using unlablelled target data for adaptation. (2) Without proper weighting for each source domain, DANN with joint source data can sometimes perform worse than SRC. This suggests the importance of ignoring irrelevant data to avoid negative transfer. (3) DARN significantly outperforms other methods across all four domains, especially on the MNIST-M and SVHN domains. Notice that even though MDAN

and MDMN have generalization guarantees, they both resort to ad-hoc aggregation rules to combine the source domains during training. Instead, our aggregation (Eq. (3.7)) is a direct optimization of the upper bound (Theorem 2) thus is theoretically justified and empirically superior for this problem.

3.6.3 Object Recognition: Office-Home

Table 3.2: Classification accuracy (%) of the Office-Home datasets. Mean and standard error over 20 runs. The best method (excluding TAR) based on one-sided Wilcoxon signed-rank test at the 5% significance level is shown in bold for each domain.

Method	Art	Clipart	Product	Real-World	Avg.
SRC	58.02 ± 0.47	57.29 ± 0.30	74.26 ± 0.22	77.98 ± 0.25	66.89 ± 0.16
DANN	57.39 ± 0.69	57.35 ± 0.35	73.78 ± 0.27	78.12 ± 0.21	66.66 ± 0.19
M3SDA	64.05 ± 0.61	62.79 ± 0.37	76.21 ± 0.30	78.63 ± 0.22	70.42 ± 0.18
MDAN	68.14 ± 0.58	67.04 ± 0.21	81.03 ± 0.22	82.79 ± 0.15	74.75 ± 0.18
MDMN	68.67 ± 0.55	67.75 ± 0.20	81.37 ± 0.18	83.32 ± 0.14	75.28 ± 0.15
DARN	**70.00** ± 0.38	**68.42** ± 0.14	**82.75** ± 0.21	**83.88** ± 0.16	**76.26** ± 0.13
TAR	71.19 ± 0.38	79.16 ± 0.16	90.66 ± 0.15	85.60 ± 0.14	81.65 ± 0.12

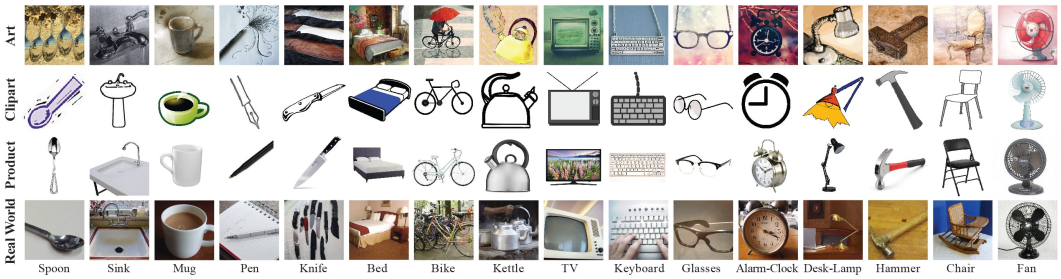


Figure 3.4: Example images of the Office-Home dataset.

Setup. To show the applicability of our method to more complicated real-world tasks, we use the Office-Home dataset (Venkateswara et al., 2017). Some example images are shown in Fig. 3.4. It contains images of 65 classes such as spoon, sink, mug and pen from four different domains: Art, Clipart, Product and Real-World. One of the four datasets is chosen as unlabelled target domain in turn and the other three are used as labelled source domains.

Results and Analysis. Table 3.2 shows the classification accuracy of each target dataset over 20 runs. Most existing works in the literature focused on single-source adaptation for this problem (e.g., see Long et al. (2018)). Compared to them, using multi-source methods can significantly boost performance, revealing the importance of properly combining multiple source domains. Even though this is a significantly more challenging problem with more classes and much fewer images compared to the digit datasets, DARN achieves state-of-the-art performance in this setting, beating existing methods by a noticeable margin.

3.6.4 Sentiment Analysis

Table 3.3: Classification accuracy (%) of the target sentiment datasets. Mean and standard error over 20 runs. The best method(s) (excluding TAR) based on one-sided Wilcoxon signed-rank test at the 5% significance level is(are) shown in bold for each domain.

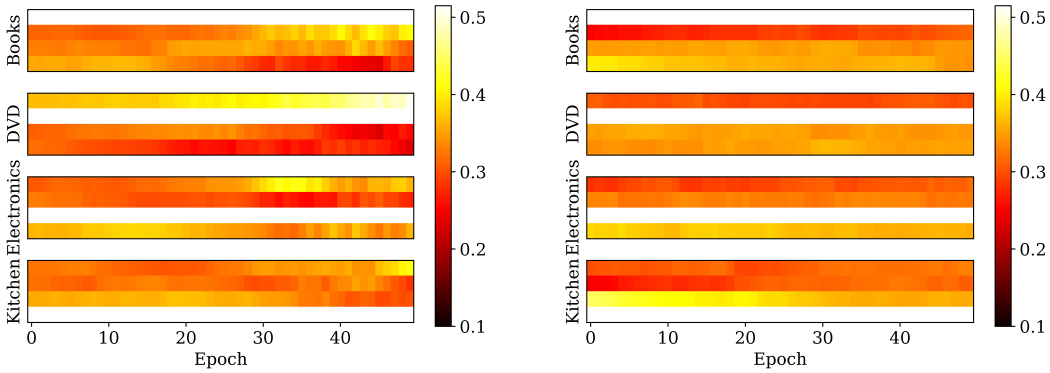
Method	Books	DVD	Electronics	Kitchen	Avg.
SRC	79.15 ± 0.39	80.38 ± 0.30	85.48 ± 0.10	85.46 ± 0.34	82.62 ± 0.20
DANN	79.13 ± 0.29	80.60 ± 0.29	85.27 ± 0.14	85.56 ± 0.28	82.64 ± 0.14
M3SDA	79.42 ± 0.17	80.82 ± 0.35	85.52 ± 0.19	86.45 ± 0.43	83.05 ± 0.14
MDAN	**79.99** ± 0.20	**81.66** ± 0.19	84.76 ± 0.17	86.82 ± 0.13	83.31 ± 0.08
MDMN	**80.13** ± 0.20	**81.58** ± 0.21	85.61 ± 0.13	**87.13** ± 0.11	**83.61** ± 0.07
DARN	**79.93** ± 0.19	**81.57** ± 0.16	**85.75** ± 0.16	**87.15** ± 0.14	**83.60** ± 0.08
TAR	84.10 ± 0.13	83.68 ± 0.12	86.11 ± 0.32	88.72 ± 0.14	85.65 ± 0.09

Setup. We use the Amazon review dataset (Blitzer et al., 2007; Chen et al., 2012) that consists of positive and negative product reviews from four domains (Books, DVD, Electronics and Kitchen). Each of them is used in turn as the target domain and the other three are used as source domains. We follow the common protocol (Chen et al., 2012; Zhao et al., 2018) of using the top-5000 frequent unigrams/bigrams of all reviews as bag-of-words features.

Results and Analysis. Table 3.3 summarizes the classification accuracies over 20 runs. (1) Some domains are harder to adapt to than the others. For example, the accuracies of SRC and TAR on the Electronics domain are

very close to each other, indicating that this requires little to no adaptation. Yet, DARN is the closest to the TAR performance here. The Books domain is more challenging. Even though there exists a large gap between SRC and TAR, the improvements over the SRC method are very small for all methods. (3) DARN is always within the best performing methods and significantly outperforms others in the Electronics domain. Note that MDMN additionally computes similarities within source domains in each iteration, which can be computationally expensive ($O(k^2)$ per iteration) if the number of source domains is large. Instead, DARN focuses on the discrepancy between source and target domains ($O(k)$ per iteration) so it is more efficient.

3.6.5 Visualizing Domain Importance



(a) MDMN (b) DARN

Figure 3.5: Domain weights for the Amazon data.

To show how DARN can aggregate multiple source domains effectively, we visualize the source domain weights (i.e., α in DARN) for the Amazon dataset. We also compared to the weights of MDMN, using the original authors' code.

Fig. 3.5a and Fig. 3.5b compare the evolution of source domain weights during training. In each subfigure, every row corresponds to the weights of the source domains when learning for one target domain. Brighter color indicates larger weight and the target domain itself has no weight. They are

evaluated at the end of each epoch over 50 epochs. To avoid noisy values due to small mini-batch size, the values are exponential moving averages with a decay rate of 0.95. There are a few observations. (1) The domain weights produced by MDMN are not very stable. We can see that their weights change drastically, especially towards the end of the training (e.g., epochs 40-50 for the Books target domain). After examining the MDMN weights, we notice that it can produce alternating *one-hot* vectors $\boldsymbol{\alpha}$, constantly changing from one domain to a different domain and ignoring the rest. This instability makes their domain weights hard to interpret. (2) In comparison, DARN has smoother weights during training. In Fig. 3.5b, as Electronics and Kitchen are more related to each other than Books and DVD, their respective weights remain higher during training. This is reasonable since they have overlapping products (e.g., blenders). (3) The $\boldsymbol{\alpha}$ of DARN is changing dynamically during training, showing the flexibility of DARN to adjust domain importance when necessary.

3.7 Conclusion

This work uses the discrepancy (Mansour et al., 2009a; Cortes et al., 2019) to derive a finite-sample generalization bound for multi-source to single-target adaptation. We show that, in order to achieve the best possible generalization upper bound for a target domain, we need to trade-off between including all source domains to increase effective sample size and excluding less relevant domains to avoid negative transfer. Based on the theory, we develop an algorithm, Domain AggRegation Network (DARN), that can dynamically adjust the weight of each source domain during end-to-end training. Experiments on digit/object recognition and sentiment analysis show that DARN outperforms state-of-the-art alternatives. Recent analysis (Zhao et al., 2019; Johansson et al., 2019) show that solely focusing on learning domain invariant features can be insufficient when the marginal label distributions are significantly different. Thus it makes

sense to take $\eta_{\mathcal{H}}$ into consideration when a small amount of labelled data is available for the target domain, which we will explore in the future.

Chapter 4

Batch Stationary Distribution Estimation

4.1 Overview

This chapter considers the problem of approximating the stationary distribution of an ergodic Markov chain given a set of sampled transitions. Classical simulation-based approaches assume access to the underlying process so that trajectories of sufficient length can be gathered to approximate stationary sampling. Instead, we consider an alternative setting where a *fixed* set of transitions has been collected beforehand, by a separate, possibly unknown procedure. The goal is still to estimate properties of the stationary distribution, but without additional access to the underlying system. We propose a consistent estimator that is based on recovering a correction ratio function over the given data. In particular, we develop a variational power method (VPM) that provides provably consistent estimates under general conditions. In addition to unifying a number of existing approaches from different subfields, we also find that VPM yields significantly better estimates across a range of problems, including queueing, stochastic differential equations, post-processing MCMC, and off-policy evaluation in reinforcement learning.

4.2 Introduction

Markov chains are a pervasive modeling tool in applied mathematics of particular importance in stochastic modeling and machine learning. A key property of an *ergodic* Markov chain is the existence of a unique *stationary distribution*; *i.e.*, the long-run distribution of states that remains invariant under the transition kernel. In this chapter, we consider a less well studied but still important version of the stationary distribution estimation problem, where one has access to a set of sampled transitions from a given Markov chain, but does not know the mechanism by which the probe points were chosen, nor is able to gather additional data from the underlying process. Nevertheless, one would still like to estimate target properties of the stationary distribution, such as the expected value of a random variable of interest.

This setting is inspired by many practical scenarios where sampling from the Markov process is costly or unavailable, but data has already been collected and available for analysis. A simple example is a queueing system consisting of a service desk that serves customers in a queue. Queue length changes stochastically as customers arrive or leave after being served. The long-term distribution of queue length (*i.e.*, the stationary distribution of the underlying Markov chain) is the object of central interest for managing such a service (Haviv, 2009; Serfozo, 2009). In practice, however, queue lengths are physical quantities that can only be measured for moderate periods, perhaps on separate occasions, but rarely for sufficient time to ensure the (stochastic) queue length has reached the stationary distribution. Since the measurement process itself is expensive, it is essential to make reasonable inferences about the stationary distribution from the collected data alone.

We investigate methods for estimating properties of the stationary distribution solely from a batch of previously collected data. The key idea is to first estimate a correction ratio function over the given data, which can

then be used to estimate expectations of interest with respect to the stationary distribution. To illustrate, consider an ergodic Markov chain with state space $\mathcal{X}$, transition kernel $\mathcal{T}$, and a unique stationary distribution μ that satisfies

$$\mu\left(x'\right) = \int \mathcal{T}\left(x'|x\right) \mu\left(x\right) dx := \left(\mathcal{T}\mu\right)\left(x'\right) . \tag{4.1}$$

Assume we are given a *fixed* sample of state transitions, $\mathcal{D} = \left\{(x, x')_{i=1}^{n}\right\} \sim \mathcal{T}\left(x'|x\right) p\left(x\right)$, such that each x has been sampled according to an *unknown* probe distribution p, but each x' has been sampled according to the true underlying transition kernel, $x'|x \sim \mathcal{T}\left(x'|x\right)$. Below we investigate procedures for estimating the point-wise ratios, $\widehat{\tau}\left(x_i\right) \approx \frac{\mu(x_i)}{p(x_i)}$, such that the weighted empirical distribution

$$\widehat{\mu}(x) := \Big(\sum_{i=1}^{n} \widehat{\tau}\left(x_i\right)\Big)^{-1} \sum_{i=1}^{n} \widehat{\tau}\left(x_i\right) \mathbb{I}\{x = x_i\}$$

can be used to approximate μ directly, or further used to estimate the expected value of some target function(s) of x with respect to μ. Crucially, the approach we propose does not require knowledge of the probe distribution p, nor does it require additional access to samples drawn from the transition kernel $\mathcal{T}$, yet we will be able to establish consistency of the estimation strategy under general conditions.

In addition to developing the fundamental approach, we demonstrate its applicability and efficacy in a range of important scenarios beyond queueing, including:

- **Stochastic differential equations (SDEs)** SDEs are an essential modeling tool in many fields like statistical physics (Kadanoff, 2000), finance (Oksendal, 2013) and molecular dynamcis (Liu, 2008). An autonomous SDE describes the instantaneous change of a random variable X by

$$dX = f\left(X\right) dt + \sigma\left(X\right) dW\,, \tag{4.2}$$

  where $f\left(X\right)$ is a drift term, $\sigma\left(X\right)$ a diffusion term, and W the Wiener process. Given data $\mathcal{D} = \left\{ \left(x, x'\right)_{i=1}^{n} \right\}$ such that $x \sim p\left(x\right)$ is drawn from an unknown probe distribution and x' is the next state after a small time

step according to (4.2), we consider the problem of estimating quantities of the stationary distribution μ when one exists.

- **Off-policy evaluation (OPE)** Another important application is *behavior-agnostic off-policy evaluation* (Nachum et al., 2019) in reinforcement learning (RL). Consider a Markov decision process (MDP) specified by $M = \langle \mathcal{S}, \mathcal{A}, P, R \rangle$, such that $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, P is the transition function, and R is the reward function (Puterman, 2014). Given a policy π that maps $s \in \mathcal{S}$ to a distribution over $\mathcal{A}$, a random trajectory can be generated starting from an initial state s_0: $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, where $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim P(\cdot|s_t, a_t)$ and $r_t \sim R(s_t, a_t)$. The *value* of a policy π is defined to be its long-term average per-step reward:

$$\rho(\pi) := \lim_{T \to \infty} \mathbb{E}\left[\frac{1}{T}\sum_{t=0}^{T-1} r_t\right] = \mathbb{E}_{(s,a)\sim d_\pi \circ \pi}\left[R(s,a)\right],$$

  where d_π denotes the limiting distribution over states $\mathcal{S}$ of the Markov process induced by π. In behavior-agnostic off-policy evaluation, one is given a target policy π and a set of transitions $\mathcal{D} = \left\{(s, a, r, s')_{i=1}^n\right\} \sim P(s'|s,a)\, p(s,a)$, potentially generated by multiple behavior policies. From such data, an estimate for $\rho(\pi)$ can be formed in terms of a stationary ratio estimator:

$$\rho(\pi) = \mathbb{E}_{(s,a)\sim p}\left[\frac{d_\pi(s)\,\pi(a|s)}{p(s,a)} r(s,a)\right] \approx \frac{1}{n}\sum_{i=1}^{n} \widehat{\tau}(s_i, a_i) r_i\,. \tag{4.3}$$

  We refer the interested readers to Section 4.6.4 and Appendix C for further discussion.

For the remainder of the chapter, we will outline four main contributions. First, we generalize the classical power iteration method to obtain an algorithm, the *Variational Power Method* (VPM), that can work with arbitrary parametrizations in a functional space, allowing for a flexible yet practical approach. Second, we prove the consistency and convergence of VPM. Third, we illustrate how a diverse set of stationary distribution estimation problems, including those above, can be addressed by VPM in

a unified manner. Finally, we demonstrate empirically that VPM significantly improves estimation quality in a range of applications, including queueing, sampling, SDEs and OPE.

4.3 Variational Power Method

To develop our approach, first recall the definition of $\mathcal{T}$ and μ in (4.1). We make the following assumption about $\mathcal{T}$ and μ throughout the chapter.

Assumption 1 (ergodicity)**.** *The transition operator $\mathcal{T}$ has a unique stationary distribution, denoted μ.*

Conditions under which this assumption holds are mild, and have been extensively discussed in standard textbooks (Meyn et al., 2009; Levin and Peres, 2017).

Next, to understand the role of the probe distribution p, note that we can always rewrite the stationary distribution as $\mu = p \circ \tau$ (i.e., $\mu(x) = p(x)\tau(x)$, hence $\tau(x) = \frac{\mu(x)}{p(x)}$), provided the following assumption holds.

Assumption 2 (absolute continuity)**.** *The stationary distribution μ is absolutely continuous w.r.t. p. That is, there exists $C < \infty$ such that $\|\tau\|_\infty \leqslant C$.*

Assumption 2 follows previous work (Liu and Lee, 2017; Nachum et al., 2019), and is common in density ratio estimation (Sugiyama et al., 2008; Gretton et al., 2009) and off-policy evaluation (Wang et al., 2017; Xie et al., 2019).

Combining these two assumptions, definition (4.1) yields

$$\begin{aligned}
\mu(x') &= \int \mathcal{T}(x'|x)\,\mu(x)\,dx = \int \mathcal{T}(x'|x)\,p(x)\,\frac{\mu(x)}{p(x)}dx \\
&:= \int \mathcal{T}_p(x, x')\,\tau(x)\,dx, \quad \text{which implies} \\
p(x')\,\tau(x') &= \int \mathcal{T}_p(x, x')\,\tau(x)\,dx := \mathcal{T}_p\tau(x'). \qquad (4.4)
\end{aligned}$$

This development reveals how, under the two stated assumptions, there is sufficient information to determine the unique ratio function τ that ensures

$p \circ \tau = \mu$ in principle. Given such a function τ, we can then base inferences about μ solely on data sampled from p and τ.

4.3.1 Variational Power Iteration

To develop a practical algorithm for recovering τ from the constraint (4.4), in function space, we first consider the classical power method for recovering the μ that satisfies (4.1). From (4.1) it can be seen that the stationary distribution μ is an eigenfunction of $\mathcal{T}$. Moreover, it is the *principal* eigenfunction, corresponding to the largest eigenvalue $\lambda_1 = 1$. In the simpler case of finite $\mathcal{X}$, the vector μ is the principal (right) eigenvector of the transposed transition matrix. A standard approach to computing μ is then the power method:

$$\mu_{t+1} = \mathcal{T}\mu_t, \tag{4.5}$$

whose iterates converge to μ at a rate linear in $|\lambda_2|$, where λ_2 is the second largest eigenvalue of $\mathcal{T}$. For ergodic Markov chains, one has $|\lambda_2| < 1$ (Meyn et al., 2009, Chap 20).

Our initial aim is to extend this power iteration approach to the constraint (4.4) without restricting the domain $\mathcal{X}$ to be finite. This can be naturally achieved by the update

$$\tau_{t+1} = \frac{\mathcal{T}_p \tau_t}{p}, \tag{4.6}$$

where the division is element-wise. Clearly the fixed point of (4.6) corresponds to the solution of (4.4) under the two assumptions stated above. Furthermore, just as for μ_t in (4.5), τ_t in (4.6) also converges to τ at a linear rate for finite $\mathcal{X}$. Unfortunately, the update (4.6) cannot be used directly in a practical algorithm for two important reasons. First, we do not have a point-wise evaluator for $\mathcal{T}_p$, but only samples from $\mathcal{T}_p$. Second, the operator $\mathcal{T}_p$ is applied to a function τ_t, which typically involves an intractable integral over $\mathcal{X}$ in general. To overcome these issues, we propose a variational method that considers a series of reformulated problems whose optimal solutions correspond to the updates (4.6).

To begin to develop a practical variational approach, first note that (4.6) operates directly on the density ratio, which implies the density ratio estimation techniques of Nguyen et al. (2007) and Sugiyama et al. (2012a) can be applied. Let ϕ be a lower semicontinuous, convex function satisfying $\phi(1) = 0$, and consider the induced f-divergence,

$$D_\phi(\tilde{p}\|\tilde{q}) = \int \tilde{p}(x)\,\phi\left(\frac{\tilde{q}(x)}{\tilde{p}(x)}\right)dx = -\left(\min_\nu \mathbb{E}_{\tilde{p}}[\phi^*(\nu)] - \mathbb{E}_{\tilde{q}}[\nu]\right), \quad (4.7)$$

where $\phi^*(x) = \sup_{y\in\mathbb{R}} x^\top y - \phi(y)$ is the conjugate function of ϕ. The key property of this formulation is that for any distributions $\tilde{p}$ and $\tilde{q}$, the inner optimum in ν satisfies $\partial\phi^*(\nu) = \tilde{q}/\tilde{p}$ (Nguyen et al., 2007); that is, the optimum in (4.7) can be used to directly recover the distribution ratio.

To apply this construction to our setting, first consider solving a problem of the following form in the dual space:

$$\nu_{t+1} = \arg\min_\nu \mathbb{E}_{p(x')}\left[\phi^*\left(\nu\left(x'\right)\right)\right] - \mathbb{E}_{\mathcal{T}_p(x,x')}\left[\partial\phi^*\left(\nu_t(x)\right)\cdot\nu\left(x'\right)\right] \quad (4.8)$$

$$= \arg\min_\nu \mathbb{E}_{p(x')}\left[\phi^*\left(\nu\left(x'\right)\right)\right] - \mathbb{E}_{\mathcal{T}_p(x,x')\tau_t(x)}\left[\nu\left(x'\right)\right], \quad (4.9)$$

where to achieve (4.9) we have applied the inductive assumption that $\tau_t = \partial\phi^*(\nu_t)$. Then, by the optimality property of ν_{t+1}, we know that the solution ν_{t+1} must satisfy

$$\partial\phi^*(\nu_{t+1}) = \frac{\mathcal{T}_p\tau_t}{p} = \tau_{t+1}, \quad (4.10)$$

hence the updated ratio τ_{t+1} in (4.6) can be directly recovered from the dual solution ν_{t+1}, while also retaining the inductive property that $\tau_{t+1} = \partial\phi^*(\nu_{t+1})$ for the next iteration.

These developments can be further simplified by considering the specific choice $\phi(\tau) = (\tau-1)^2$, which leads to $\phi^*(\nu) = \nu + \frac{\nu^2}{4}$, $\tau = \partial\phi^*(\nu) = 1 + \frac{\nu}{2}$ and

$$\tau_{t+1} = \arg\min_{\tau\geqslant 0} \tfrac{1}{2}\mathbb{E}_{p(x')}\left[\tau^2\left(x'\right)\right] - \mathbb{E}_{\mathcal{T}_p(x,x')}\left[\tau_t(x)\tau\left(x'\right)\right]. \quad (4.11)$$

Crucially, this variational update (4.11) determines the same update as (4.6), but overcomes the two aforementioned difficulties. First, it bypasses

the direct evaluation of $\mathcal{T}_p$ and p, and allows these to be replaced by unbiased estimates of expectations extracted from the data. Second, it similarly bypasses the intractability of the operator application $\mathcal{T}_p\tau_t$ in the functional space, replacing this with an expectation of $\tau_t \circ \tau$ that can also be directly estimated from the data.

We now discuss some practical refinements of the approach.

4.3.2 Normalization

For τ_t to be a proper ratio $\frac{\mu_t}{p}$, it should be normalized w.r.t. p, *i.e.* $\mathbb{E}_p[\tau_t] = 1$. To address this issue, we explicitly ensure normalization by considering a constrained optimization in place of (4.11):

$$\begin{aligned} \min_{\tau \geqslant 0} \ & \tfrac{1}{2}\mathbb{E}_{p(x')}\left[\tau^2\left(x'\right)\right] - \mathbb{E}_{\mathcal{T}_p(x,x')}\left[\tau_t\left(x\right)\tau\left(x'\right)\right], \\ & \text{s.t.} \quad \mathbb{E}_{p(x)}\left[\tau\left(x\right)\right] = 1. \end{aligned} \tag{4.12}$$

We can tackle this by solving its Lagrangian. To avoid instability, we add a regularization term:

$$\begin{aligned} \min_{\tau \geqslant 0} \max_{\nu \in \mathbb{R}} \ J(\tau, \nu) = & \tfrac{1}{2}\mathbb{E}_{p(x')}\left[\tau^2\left(x'\right)\right] - \mathbb{E}_{\mathcal{T}_p(x,x')}\left[\tau_t\left(x\right)\tau\left(x'\right)\right] \\ & + \nu\left(\mathbb{E}_p\left[\tau\right] - 1\right) - \tfrac{\lambda}{2}\nu^2. \end{aligned} \tag{4.13}$$

where $\lambda > 0$ is a regularization parameter. Crucially, the dual variable ν is a scalar, making this problem much simpler than dual embedding (Dai et al., 2017), where the dual variables form a parameterized function that introduces approximation error. The problem (4.13) is a straightforward convex-concave objective with respect to (τ, ν) that can be optimized by stochastic gradient descent.

The following theorem shows that under certain conditions, the normalization will be maintained for any $\lambda > 0$.

Theorem 3 (Normalization of solution)**.** *If* $\mathbb{E}_p[\tau_t] = 1$, *then for any* $\lambda > 0$, *the estimator* (4.13) *has the same solution as* (4.12), *hence* $\mathbb{E}_p[\tau_{t+1}] = 1$.

Hence, we can begin with any τ_0 satisfying $\mathbb{E}_p[\tau_0] = 1$, and the theorem ensures that the normalization of τ_{t+1} will be inductively maintained using any fixed $\lambda > 0$. The proof is given in Appendix B.1.

4.3.3 Damped Iteration

The next difficulty to be addressed arises from the fact that, in practice, we need to optimize the variational objective based on sampled data, which induces approximation error since we are replacing the true operator $\mathcal{T}_p$ by a stochastic estimate $\widehat{\mathcal{T}}_p$ such that $\mathbb{E}[\widehat{\mathcal{T}}_p] = \mathcal{T}_p$. Without proper adjustment, such estimation errors can accumulate over the power iterations, and lead to inaccurate results.

To control the error due to sampling, we introduce a damped version of the update (Ryu and Boyd, 2016), where instead of performing a stochastic update $\tau_{t+1} = \frac{\widehat{\mathcal{T}}_p}{p}\tau_t$, we instead perform a damped update given by

$$\tau_{t+1} = (1 - \alpha_{t+1}) \cdot \tau_t + \alpha_{t+1} \cdot \frac{\widehat{\mathcal{T}}_p}{p}\tau_t \tag{4.14}$$

where $\alpha_t \in (0, 1)$ is a stepsize parameter. Intuitively, the update error introduced by the stochasticity of $\widehat{\mathcal{T}}_p$ is now controlled by the stepsize α_t. The choice of stepsize and convergence of the algorithm is discussed in Section 4.4.

The damped iteration can be conveniently implemented with minor modifications to the previous objective. We only need to change the sample from $\mathcal{T}_p$ in (4.13) by a weighted sample:

$$\begin{aligned}\min_{\tau \geqslant 0} \max_{\nu \in \mathbb{R}} \; J(\tau, \nu) = \tfrac{1}{2}\mathbb{E}_{p(x')}\left[\tau^2\left(x'\right)\right] - (1 - \alpha_{t+1})\mathbb{E}_{p(x')}\left[\tau_t\left(x'\right)\tau\left(x'\right)\right] \\ - \alpha_{t+1}\mathbb{E}_{\mathcal{T}_p(x,x')}\left[\tau_t(x)\tau\left(x'\right)\right] + \nu\left(\mathbb{E}_p\left[\tau\right] - 1\right) - \tfrac{\lambda}{2}\nu^2.\end{aligned} \tag{4.15}$$

4.3.4 A Practical Algorithm

A practical version of VPM is described in Algorithm 2. It solves (4.15) using a parameterized $\tau : \mathcal{X} \mapsto \mathbb{R}$ expressed as a neural network τ_θ with parameters θ. Given the constraint $\tau \geqslant 0$, we added a softplus activation $\log(1 + \exp(\cdot))$ to the final layer to ensure positivity. The expectations with respect to p and $\mathcal{T}_p$ are directly estimated from sampled data. When optimizing τ_θ by stochastic gradient methods, we maintain a copy of the previous network τ_t as the reference network to compute the second and

Algorithm 2 Variational Power Method

Input: Transition data $\mathcal{D} = \{(x, x')_{i=1}^{n}\}$, learning rate $\alpha_\theta, \alpha_\nu$, number of power steps T, number of inner optimization steps M, batch size B
Initialize τ_θ
for $t = 1 \dots T$ **do**
    Update and fix the reference network $\tau_t = \tau_\theta$
    for $m = 1 \dots M$ **do**
        Sample transition data $\{(x, x')_{i=1}^{B}\}$
        Compute gradients $\nabla_\theta J$ and $\nabla_\nu J$ from (4.16)
        $\theta = \theta - \alpha_\theta \nabla_\theta J$ ▷ gradient descent
        $\nu = \nu + \alpha_\nu \nabla_\nu J$ ▷ gradient ascent
    end for
end for
Return τ_θ

third terms of (4.15). The gradients of $J(\tau, \nu)$ with respect to θ and ν are given by

$$\begin{aligned} \nabla_\theta J(\tau, \nu) &= \mathbb{E}_p\left[\tau \nabla_\theta \tau\right] - (1 - \alpha_{t+1})\mathbb{E}_p\left[\tau_t \nabla_\theta \tau\right] \\ &\quad - \alpha_{t+1}\mathbb{E}_{\mathcal{T}_p}\left[\tau_t \nabla_\theta \tau\right] + \nu \mathbb{E}_p\left[\nabla_\theta \tau\right], \\ \nabla_\nu J(\tau, \nu) &= \mathbb{E}_p\left[\tau\right] - 1 - \lambda\nu. \end{aligned} \tag{4.16}$$

After convergence of τ_θ in each iteration, the reference network is updated by setting $\tau_{t+1} = \tau_\theta$. Note that one may apply other gradient-based optimizers instead of SGD.

4.4 Convergence Analysis

We now demonstrate that the final algorithm obtains sufficient control over error accumulation to achieve consistency. For notation brevity, we discuss the result for the simpler form (4.5) instead of the ratio form (4.6). The argument easily extends to the ratio form.

Starting from the plain stochastic update $\mu_t = \widehat{\mathcal{T}}\mu_{t-1}$, the damped update can be expressed by

$$\begin{aligned} \mu_t &= (1 - \alpha_t)\mu_{t-1} + \alpha_t \widehat{\mathcal{T}}\mu_{t-1} \\ &= (1 - \alpha_t)\mu_{t-1} + \alpha_t \mathcal{T}\mu_{t-1} + \alpha_t \epsilon, \end{aligned} \tag{4.17}$$

where ϵ is the error due to stochasticity in $\widehat{\mathcal{T}}$. The following theorem establishes the convergence properties of the damped iteration.

Theorem 4 (Informal)**.** *Under mild conditions, after* t *iteration with step-size* $\alpha_t = 1/\sqrt{t}$*, we have*

$$\mathbb{E}\left[\|\mu_R - \mathcal{T}\mu_R\|_2^2\right] \leqslant \frac{C_1}{\sqrt{t}}\|\mu_0 - \mu\|_2^2 + \frac{C_2 \ln t}{\sqrt{t}}\|\epsilon\|_2^2,$$

for some constants $C_1, C_2 > 0$*, where the expectation is taken over the distribution of iterates* $(\mu_R)_{R=1}^t$*. In other words,* $\mathbb{E}\left[\|\mu_R - \mathcal{T}\mu_R\|_2^2\right] = \widetilde{\mathcal{O}}\left(t^{-1/2}\right)$*, and consequently* μ_R *converges to* μ *for ergodic* $\mathcal{T}$*.*

The precise version of the theorem statement, together with a complete proof, is given in Appendix B.2.

Note that the optimization quality depends on the number of samples, the approximation error of the parametric family, and the optimization algorithm. There is a complex trade-off between these factors (Bottou and Bousquet, 2008). On one hand, with more data, the statistical error is reduced, but the computational cost of the optimization increases. On the other hand, with a more flexible parametrization, such as neural networks, reduces the approximation error, but adds to the difficulty of optimization as the problem might no longer be convex. Alternatively, if the complexity of the parameterized family is increased, the consequences of statistical error also increases.

Representing τ in a reproducing kernel Hilbert space (RKHS) is a particularly interesting case, because the problem (4.13) becomes convex, hence the optimization error of the empirical surrogate is reduced to zero. Nguyen et al. (2007, Theorem 2) show that, under mild conditions, the statistical error can be bounded in rate $\mathcal{O}\left(n^{-\frac{1}{2+\beta}}\right)$ in terms of Hellinger distance (β denotes the exponent in the bracket entropy of the RKHS), while the approximation error will depend on the RKHS (Bach, 2017).

4.5 Related Work

The algorithm we have developed reduces distribution estimation to density ratio estimation, which has been extensively studied in numerous contexts. One example is learning under covariate shift (Shimodaira, 2000), where the ratio τ can be estimated by different techniques (Gretton et al., 2009; Nguyen et al., 2007; Sugiyama et al., 2008; Sugiyama and Kawanabe, 2012). These previous works differ from the current setting in that they require data to be sampled from both the target and proposal distributions. By contrast, we consider a substantially more challenging problem, where only data sampled from the proposal is available, and the target distribution is given only *implicitly* by (4.1) through the transition kernel $\mathcal{T}$. A more relevant approach is Stein importance sampling (Liu and Lee, 2017), where the ratio is estimated by minimizing the kernelized Stein discrepancy (Liu et al., 2016). However, it requires additional gradient information about the target potential, whereas our method only requires sampled transitions. Moreover, the method of Liu and Lee (2017) is computationally expensive and does not extrapolate to new examples.

The algorithm we develop in this chapter is inspired by the classic power method for finding principal eigenvectors. Many existing works have focused on the finite-dimension setting (Balsubramani et al., 2013; Hardt and Price, 2014; Yang et al., 2019), while Kim et al. (2005) and Xie et al. (2015) have extended the power method to the infinite-dimension case using RKHS. Not only do these algorithms require access to the transition kernel $\mathcal{T}$, but they also require tractable operator multiplications. In contrast, our method avoids direct interaction with the operator $\mathcal{T}$, and can use flexible parametrizations (such as neural networks) to learn the density ratio without per-step renormalization.

Another important class of methods for estimating or sampling from stationary distributions are based on simulations. A prominent example is Markov chain Monte Carlo (MCMC), which is widely used in many

statistical inference scenarios (Andrieu et al., 2003; Koller and Friedman, 2009; Welling and Teh, 2011). Existing MCMC methods (*e.g.*, Neal et al., 2011; Hoffman and Gelman, 2014) require repeated, and often many, interactions with the transition operator $\mathcal{T}$ to acquire a single sample from the stationary distribution. Instead, VPM can be applied when only a fixed sample is available. Interestingly, this suggests that VPM can be used to "post-process" samples generated from typical MCMC methods to possibly make more effective use of the data. We demonstrated this possibility empirically in Section 4.6. Unlike VPM, other post-processing methods (Oates et al., 2017) require additional information about the target distribution (Robert and Casella, 2013). Recent advances have also shown that learning parametric samplers can be beneficial (Song et al., 2017; Li et al., 2019), but require the potential function. In contrast, VPM directly learns the stationary density ratio solely from transition data.

One important application of VPM is off-policy RL (Precup et al., 2001). In particular, in off-policy evaluation (OPE), one aims to evaluate a target policy's performance, given data collected from a different behavior policy. This problem matches our proposed framework as the collected data naturally consists of transitions from a Markov chain, and one is interested in estimating quantities computed from the stationary distribution of a different policy. (See Appendix C for a detailed description of how the VPM algorithm can be applied to OPE, even when $\gamma = 1$.) Standard importance weighting is known to have high variance, and various techniques have been proposed to reduce variance (Precup et al., 2001; Jiang and Li, 2016; Rubinstein and Kroese, 2016; Thomas and Brunskill, 2016; Guo et al., 2017). However, these methods still exhibit exponential variance in the trajectory length (Li et al., 2015b; Jiang and Li, 2016).

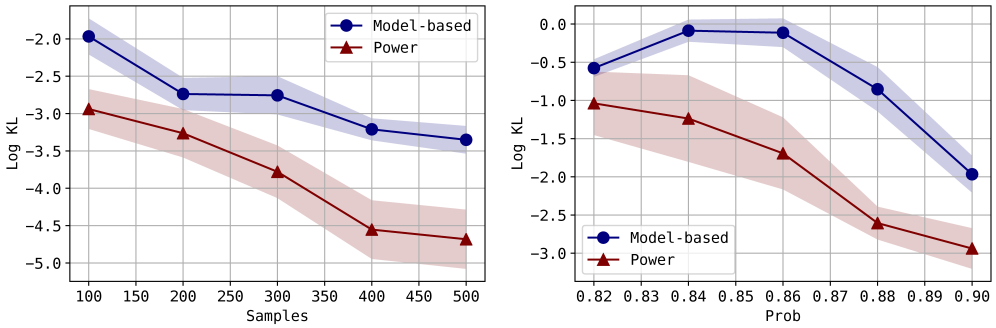
More related to the present work is the recent work on off-policy RL that avoids the exponential blowup of variance. It is sufficient to adjust observed rewards according to the ratio between the target and behavior stationary distributions (Hallak and Mannor, 2017; Liu et al., 2018; Gelada

and Bellemare, 2019). Unfortunately, these methods require knowledge of the behavior policy, $p(a|s)$, in addition to the transition data, which is not always available in practice. In this work, we focus on the behavior-agnostic scenario where $p(a|s)$ is unknown. Although the recent work of Nachum et al. (2019) considers the same scenario, their approach is only applicable when the discount factor $\gamma < 1$, whereas the method in this work can handle any $\gamma \in [0, 1]$.

4.6 Experimental Evaluation

In this section, we demonstrate the advantages of VPM in four representative applications. Experiment details are provided in Appendix B.4.

4.6.1 Queueing



(a) Number of samples (b) Finish probability

Figure 4.1: Log KL divergence between estimation and the truth.

In this subsection, we use VPM to estimate the stationary distribution of queue length. Following the standard Kendall's notation in queueing theory (Haviv, 2009; Serfozo, 2009), we analyze the discrete-time Geo/Geo/1 queue, which is commonly used in the literature (Atencia and Moreno, 2004; Li and Tian, 2008; Wang et al., 2014). Here the customer inter-arrival time and service time are geometrically distributed with one service desk. For this problem, x represents the queue length, and the observed transition (x, x') is the change in queue length after one time step. For example,

a transition $(3, 4)$ indicates that the queue changes from 3 customers to 4 customers after one time step (recall that this is a discrete-time queue). The probe distribution $p(x)$ is a uniform distribution over the lengths in a predefined range $[0, B)$, meaning that we have some information on how the queue length can change within this range. This queue setting has a closed-form stationary distribution that we can compare to (Serfozo, 2009, Sec.1.11).

Fig. 4.1 provides the log KL divergence between the estimated and true stationary distributions. We compare VPM to a model-based approach, which estimates the transition matrix $\widehat{\mathcal{T}}(x'|x)$ from the same set of data, then simulates a long trajectory using $\widehat{\mathcal{T}}$. It can be seen that our method can be more effective across different sample sizes and queue configurations.

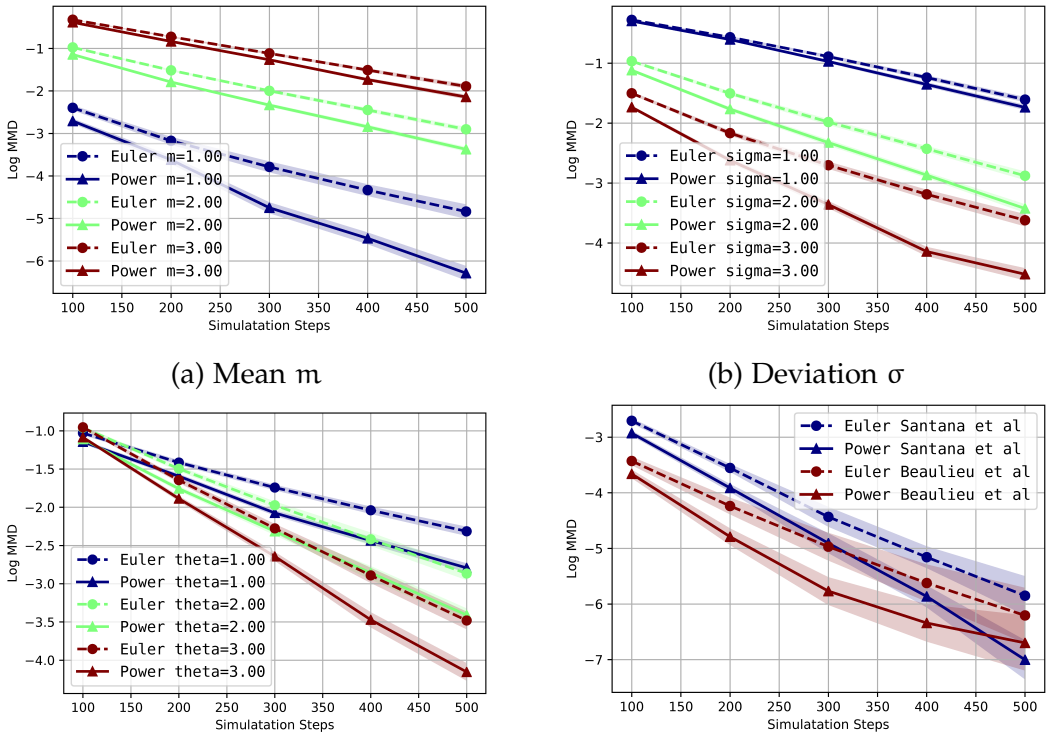
4.6.2 Solving SDEs

We next apply VPM to solve a class of SDEs known as the Ornstein-Uhlenbeck process (OUP), which finds many applications in biology (Butler and King, 2004), financial mathematics and physical sciences (Oksendal, 2013). The process is described by the equation:

$$dX = \theta(m - X)dt + \sigma dW$$

where m is the asymptotic mean, $\sigma > 0$ is the deviation, $\theta > 0$ determines the strength, and W is the Wiener process. The OUP has a closed-form solution, which converges to the stationary distribution, a normal distribution $\mathcal{N}(m, \sigma^2/2\theta)$, as $t \to \infty$. This allows us to conveniently calculate the Maximum Mean Discrepancy (MMD) between the adjusted sample to a true sample. We compare our method with the Euler-Maruyama (EM) method (Gardiner, 2009), which is a standard simulation-based method for solving SDEs. VPM uses samples from the EM steps to train the ratio network and the learned ratio is used to compute weighted MMD.

The results are shown in Fig. 4.2, with different configurations of parameters (m, σ, θ). It can be seen that VPM consistently improves over the



(a) Mean m

(b) Deviation σ

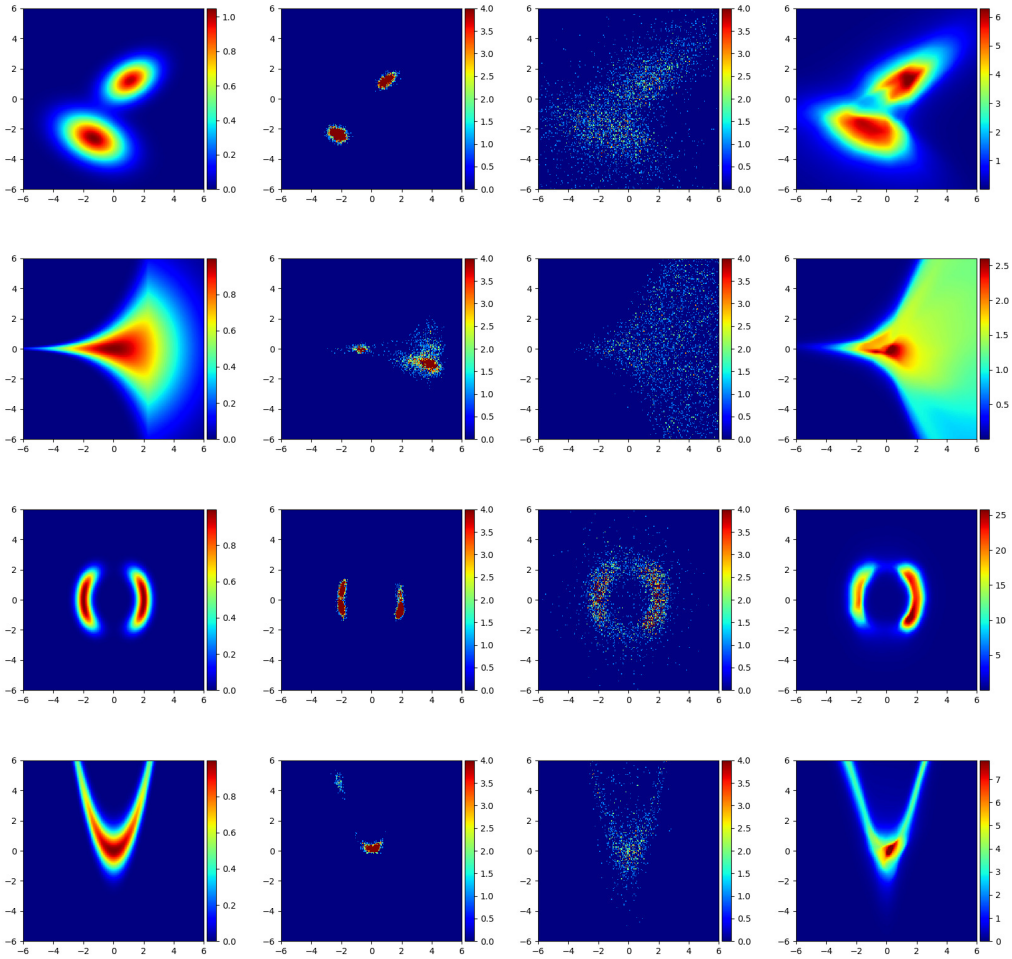
(c) Strength θ

(d) Phylogeny Studies

Figure 4.2: Log MMD versus number of EM steps across different settings, default $(m, \sigma, \theta) = (2, 2, 2)$. (d) is based on the real-world phylogeny studies (Beaulieu et al., 2012; Santana et al., 2012) with $(m, \sigma, \theta) = (0.618, 1.584, 3.85), (0.661, 0.710, 8.837)$ respectively.

EM method in terms of the log MMD to a true sample from the normal distribution. The EM method only uses the most recent data, which can be wasteful since the past data can carry additional information about the system dynamics.

In addition, we perform experiment on real-world phylogeny studies. OUP is widely used to model the evolution of various organism traits. The results of two configurations (Beaulieu et al., 2012; Santana et al., 2012, Tab.3&1 resp.) are shown in Fig. 4.2d. Notably VPM can improve over the EM method by correcting the sample with learned ratio.



(a) Potentials (b) Model (c) VPM (d) Estimated τ

Figure 4.3: The 2nd and 3rd columns are samples from the model-based method and VPM respectively. Rows (from top to bottom) correspond to data sets: 2gauss, funnel, kidney, banana.

4.6.3 Post-processing MCMC

In this experiment, we demonstrate how VPM can post-process MCMC to use transition data more effectively in order to learn the target distributions. We use four common potential functions as shown in the first column of Fig. 4.3 (Neal, 2003; Rezende and Mohamed, 2015; Wenliang et al., 2019). A point is sampled from the uniform distribution $p(x) = \text{Unif}(x; [-6, 6]^2)$, then transitioned through an HMC operator (Neal et al., 2011). The transitioned pairs are used as training set $\mathcal{D}$.

We compare VPM to a model-based method that explicitly learns a transition model $\widehat{\mathcal{T}}(x'|x)$, parametrized as a neural network to produce Gaussian mean (with fixed standard deviation of 0.1). Then, we apply $\widehat{\mathcal{T}}$ 100 times to a hold-out set drawn from $p(x)$, and use the final instances as limiting samples (second column of Fig. 4.3). As for VPM, since p is uniform, the estimated $\hat{\tau}$ is proportional to the true stationary distribution. To obtain limiting samples (third column of Fig. 4.3), we resample from a hold-out set drawn from $p(x)$ with probability proportional to $\hat{\tau}$.

The results are shown in Fig. 4.3. Note that the model-based method quickly collapses all training data into high-probability regions as stationary distributions, which is an inevitable tendency of restricted parametrized $\widehat{\mathcal{T}}$. Our learned ratio faithfully reconstructs the target density as shown in the right-most column of Fig. 4.3. The resampled data of VPM are much more accurate and diverse than that of the model-based method. These experiments show that VPM can indeed effectively use a fixed set of data to recover the stationary distribution without additional information.

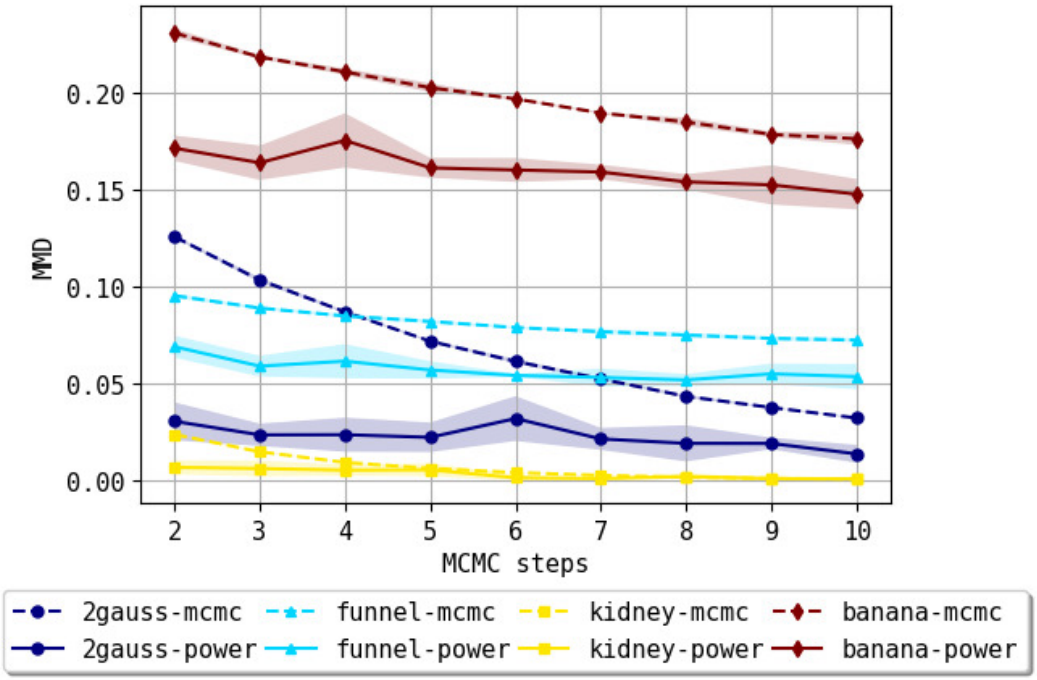
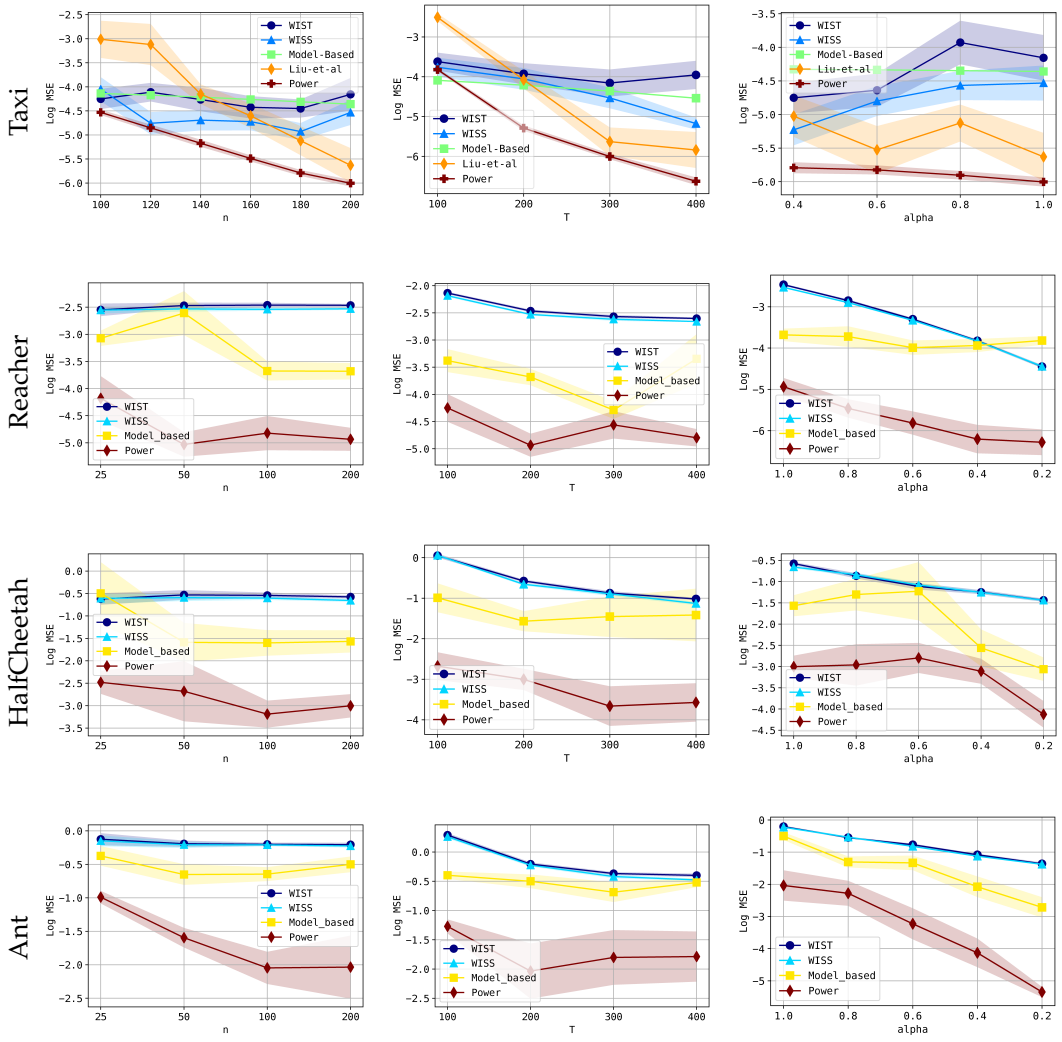


Figure 4.4: MMD before and after ratio correction using VPM.

To compare the results quantitatively, Fig. 4.4 shows the MMD of the estimated sample to a "true" sample. Since there is no easy way to sample from the potential function, the "true" sample consists of data after 2k HMC steps with rejection sampler. After each MCMC step, VPM takes the transition pairs as input and adjusts the sample importance according to the learned ratio. As we can see, after each MCMC step, VPM is able to post-process the data and further reduce MMD by applying the ratio. The improvement is consistent along different MCMC steps across different datasets.



(a) # of trajectories (b) Trajectory length (c) Policy deviation

Figure 4.5: Log MSE of different methods for various datasets and settings.

4.6.4 Off-Policy Evaluation

Finally, we apply our method to behavior-agnostic off-policy evaluation outlined in Section 4.2, in which only the transition data and the target policy are given, while the behavior policy is *unknown*. Concretely, given a sample $\mathcal{D} = \{(s, a, r, s')_{i=1}^n\}$ from the behavior policy, we compose each transition in $\mathcal{D}$ with a target action $a' \sim \pi(\cdot|s')$. Denoting $x = (s, a)$, the data set can be expressed as $\mathcal{D} = \{(x, x')_{i=1}^n\}$. Applying the proposed VPM with $\mathcal{T}(x'|x)$, we can estimate $\frac{\mu(s,a)}{p(s,a)}$, hence the average accumulated reward can be obtained via (4.3). Additional derivation and discussion can be found in Appendix B.3.

We conduct experiments on the (discrete) Taxi environment as in Liu et al. (2018), and the challenging (continuous) environments including the Reacher, HalfCheetah and Ant.

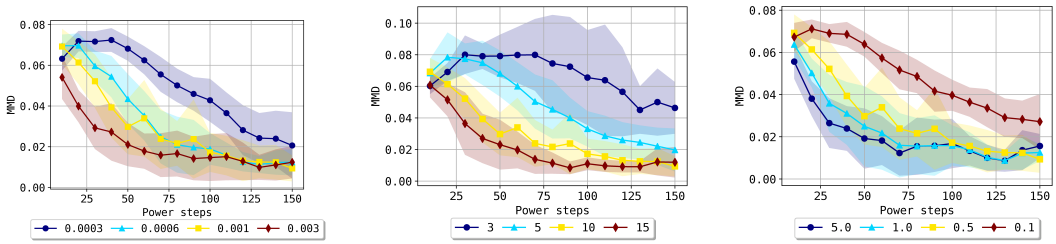
Taxi is a gridworld environment in which the agent navigates to pick up and drop off passengers in specific locations. The target and behavior policies are set as in Liu et al. (2018). For the continuous environments, the Reacher agent tries to reach a specified location by swinging an robotic arm, while the HalfCheetah/Ant agents are complex robots that try to move forward as much as possible. The target policy is a pre-trained PPO or A2C neural network, which produces a Gaussian action distribution $\mathcal{N}(m_t, \Sigma_t)$. The behavior policy is the same as target policy but using a larger action variance $\Sigma_b = (1 - \alpha)\Sigma_t + 2\alpha\Sigma_t, \alpha \in (0, 1]$. We collect T trajectories of n steps each, using the behavior policy.

We compare VPM to a model-based method that estimates both the transition $\mathcal{T}$ and reward R functions. Using *behavior cloning*, we also compare to the trajectory-wise and step-wise weighted importance sampling (WIST,WISS) (Precup et al., 2001), as well as Liu et al. (2018) with their public code for the Taxi environment.

The results are shown in Fig. 4.5. The x-axes are different configurations and the y-axes are the log Mean Square Error (MSE) to the true average tar-

get policy reward, estimated from abundant on-policy data collected from the target policy. As we can see, VPM outperforms all baselines significantly across different settings, including number of trajectories, trajectory length and behavior policies. The method by Liu et al. (2018) can suffer from not knowing the behavior policy, as seen in the Taxi environment. Weighted importance sampling methods (WIST,WISS) also require access to the behavior policy.

4.6.5 Ablation Study



(a) Learning rate (b) # of inner steps M (c) Regularization λ

Figure 4.6: Ablation study. MMD versus number of power iterations for the funnel dataset. Default $(\mathrm{lr}, M, \lambda) = (0.001, 10, 0.5)$.

In this section, we conduct an ablation study to show that VPM is robust to different choices of the parameters. Fig. 4.6 shows the MMD curves for the MCMC funnel dataset in Section 4.6.3, using different learning rates, number of inner optimization steps and the regularization λ. Other datasets show similar trends.

Learning rates. In all experiments, we use Adam optimizer (Kingma and Ba, 2014). Fig. 4.6a shows the convergent behaviour with different learning rates in $\{0.0003, 0.0006, 0.001, 0.003\}$. The algorithm can take a longer time to converge when using a small learning rate. Even though large learning rate (*e.g.*, 0.003) seems to converge faster, its final solution can be noisy. We can see that VPM can work using different learning rates around the default Adam learning rate of 0.001.

Number of inner optimization steps. Recall that in each power iteration, VPM solves an inner optimization Eq. (4.15). Fig. 4.6b shows the

the effect of different number of inner optimization steps M. Larger M can produce more accurate power iterator and converge faster in terms of number of power iterations, but the time per iteration will also increase accordingly. If M is too small (*e.g.*, 3), the learning can be unstable and the final ratio network can be inaccurate. Due to the damped update, the error in each power iteration can be controlled effectively and VPM can converge to the optimal ratio as long as M is reasonably large.

Regularization. Finally, we investigate the effect of the regularization parameter λ. Intuitively, λ controls the capability of the dual variable ν in Eq. (4.13). The results are shown in Fig. 4.6c. Although different λ values can have different convergence speeds, their final solutions can achieve low MMD given sufficient iterations, as suggested by Theorem 3.

4.7 Conclusion

We have formally considered the problem of estimating stationary distribution of an ergodic Markov chain using a fixed set of transition data. We extended a classical power iteration approach to the batch setting, using an equivalent variational reformulation of the update rule to bypass the agnosticity of transition operator and the intractable operations in a functional space, yielding a new algorithm *Variational Power Method (VPM)*. We characterized the convergence of VPM theoretically, and demonstrated its empirical advantages for improving existing methods on several important problems such as queueing, solving SDEs, post-processing MCMC and behavior-agnostic off-policy evaluation.

Chapter 5

Perspectives and Prospects

In this dissertation, we showed how to utilize data from prior sources to facilitate learning and inference in new environments. Specifically, we demonstrated this idea for domain adaptation and reinforcement learning. For domain adaptation, we first improved the computational efficiency of classical algorithms, KMM and KLIEP, that compute importance weights for covariate shift problems in Chapter 2. Then, we presented a novel theoretical analysis and algorithmic development for multi-source adaptation problems in Chapter 3. For reinforcement learning, we developed a variational power method for behaviour-agnostic off-policy evaluation and beyond in Chapter 4. Although the proposed methods have their respective assumptions, they all require that the prior data is "similar" to the target data in some sense. Specifically, for methods based on importance sampling (Chapter 2 and Chapter 4), it is assumed that the density ratio is bounded from above. This corresponds to the case that we always have a chance to see any target example in the source/behaviour dataset. As for Theorem 2 in Chapter 3, if none of the sources is similar to the target data, then finding the optimal domain weights for the upper bound is not very meaningful. It is important to identify and verify the required characteristics of the prior data, otherwise, it would not make sense to use them to facilitate learning for a target domain/environment.

The proposed methods have improved over existing work in different ways. The optimization procedure in Chapter 2 for KMM and KLIEP is

based on the Frank-Wolfe algorithm (a.k.a. conditional gradient descent). It reduced the computational complexity of KMM from $O(n^2)$ to $O(n)$ per iteration, and improved the KLIEP so that it can utilize all source data points without sub-sampling. The domain aggregation network in Chapter 3 is based on a novel theoretical analysis extended from single-source to single-target adaptation. The algorithm can combine data from multiple domains effectively and train a model end-to-end with interpretable domain importance. Finally, the variational power method in Chapter 4 is suitable for several fundamental estimation problems including off-policy evaluation even in the behaviour-agnostic setting. Besides, it is provably convergent, unlike similar techniques in the literature.

The methods introduced in this dissertation are fundamental and can be applied to various practical problems. For example, correcting sample bias is critical for treatment effect estimation using observational data. Unlike randomized control trials (RCT), observational data is often biased due to clinical constraints. Thus bridging the gap between RCT and observational studies is essential to obtain an accurate treatment effect. As another example, off-policy evaluation can be applied to A/B testing in the web/app industries. Evaluating new product variants offline is appealing since conducting A/B testing with real-world users can be expensive, and we are likely to have plenty of log data collected from prior interactions. These are just two examples of potential applications, and there are many other possibilities for us to explore.

Nevertheless, the methods developed here have limitations. It is clear that if the source/behaviour data has nothing in common with the target data, using prior data may even hurt learning in the target domain. This can happen in domain adaptation, for example, when the source and target domains have contradicting/adversarial concepts: given the same image, the source domain claims it is a dog, while the target domain claims it is only a cat. Of course, this is an extreme case, and the source and the target may only partially disagree for real-world problems. However, this shows

that, aligning datasets without looking at their labels would be insufficient under some circumstances. The most straightforward amendment is to acquire and utilize a small labelled set from the target domain, which is indeed practical in most use cases. A similar problem can happen for policy evaluation in the RL. If the behaviour data doesn't have full support over the target data, evaluating the target policy would be difficult because there exists some part of the state space that has never been explored before. One possible solution to extrapolate beyond source/behaviour data is to provide confidence bounds: the further away from source/behaviour, the less confident about the prediction. This can be helpful or even essential for many decision-making applications.

5.1 Future Directions

The methods introduced in this dissertation are fundamental and it is possible to extend the ideas to several related fields of research in the literature.

Federated Learning In federated learning, we have a centralized model on the server and decentralized data scattered over different user devices like cell phones or smartwatches. The model needs to be transferred to a local device for prediction, and it also needs to be updated periodically for better prediction. However, user privacy is crucial in federated learning, so accessing user data directly from the server is prohibited. One of the prominent applications of federated learning is text typing, in which the user types in some text and the prediction model offers several autocompletes. Learning from different users, the model tries to provide accurate predictions without transferring sensitive data to the server. There are two challenges from the two directions of the communication: (1) after the model is transferred to the local device, how to further improve its performance *locally* using user data for better personalization, and (2) what information should be transferred to the server to update the centralized model. The techniques discussed in this dissertation are not immediately

applicable due to the following reasons. For the first challenge, to better fit a particular user's data, one needs to compare them (*e.g.*, user's typing history) to a larger set of other users' data (others' histories). Such a comparison is difficult as it is likely to cause privacy violations. For the second challenge, aggregating information from different users is difficult without direct access. We need to explore ways to adjust existing algorithms for such a privacy-sensitive situation.

Imitation Learning Unlike batch RL that learns/evaluates from arbitrary experience, the agent in imitation learning wants to learn from expert's demonstrations. The expert's actions are supposed to be close to optimal, so behaviour cloning (Ross et al., 2011) might be a good strategy for the agent. However, mimicking the expert's behaviour might not be sufficient unless the expert is truly optimal. Moreover, in most cases, demonstration is limited, and the environment is large and complicated. Therefore, it would be essential for the agent to explore beyond the expert's demonstration.

There are other scenarios in which the demonstrations come from multiple experts, or the same expert with different intents. This is common in the multi-task setting where each collected trajectory is associated with a task. For example, one trajectory is to go to the kitchen and cook, while another one is to go to the living room and clean. In the same state/position, the expert may head to different directions depending on the task in mind. Figuring out the underlying intent of each trajectory and then apply the most relevant demonstrations for a specific target task is similar to what was presented in this dissertation.

Sim2Real One may learn a model from simulation since collecting data from a simulator is relatively cheap and easy. However, applying such a model to real-world environments can be problematic due to the so-called reality gap. A model built on synthetic data may not work well for real-world data because of their differences, and we may need to refine the synthetic data in some way so that they are more realistic. Such refinement

can be carried out not only at the data level but also at the code level: we can modify the simulator itself so that the generated data become more realistic. Besides, using simulated data can improve data efficiency for solving real-world problems. For example, we can generate as many data as we want using simulated robot arms. Instead of running a real robot for months, using simulation data collected under different conditions can accelerate learning without interacting with the physical environment. Borrowing ideas from domain adaptation can alleviate the burden of collecting a large amount of real-world data, and the ideas presented in this dissertation may be used to select the most relevant simulations. In short, utilizing simulation data can be beneficial if we can close the reality gap.

5.2 Summary

This dissertation demonstrated how to utilize data collected from different sources to assist learning and inference for domain adaptation, reinforcement learning and beyond. We have developed novel theories and efficient algorithms for both one-to-one and multi-source adaptation. We have also proposed new algorithms to compute importance ratios in different contexts, and empirical studies show that the proposed methods can outperform popular alternatives in the literature. These methods can be used for a wide range of applications and may also inspire advancements for related fields of research in the future.

Bibliography

Agarwal, D., Li, L., and Smola, A. J. (2011). Linear-time estimators for propensity scores. In *International Conference on Artificial Intelligence and Statistics*, pages 93–100.

Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43.

Atencia, I. and Moreno, P. (2004). The discrete-time geo/geo/1 queue with negative customers and disasters. *Computers & Operations Research*, 31(9):1537–1548.

Bach, F. (2015). Duality between subgradient and conditional gradient methods. *SIAM Journal on Optimization*, 25(1):115–129.

Bach, F. (2017). Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681.

Bach, F., Lacoste-Julien, S., and Obozinski, G. (2012). On the equivalence between herding and conditional gradient algorithms. In *International Conference on Machine Learning*, pages 1359–1366.

Balsubramani, A., Dasgupta, S., and Freund, Y. (2013). The fast convergence of incremental pca. In *Advances in neural information processing systems*, pages 3174–3182.

Banerjee, A., Merugu, S., Dhillon, I. S., and Ghosh, J. (2005). Clustering with bregman divergences. *The Journal of Machine Learning Research*, 6:1705–1749.

Bartlett, P. L. and Mendelson, S. (2002). Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482.

Beaulieu, J. M., Jhwueng, D.-C., Boettiger, C., and O'Meara, B. C. (2012). Modeling stabilizing selection: expanding the ornstein–uhlenbeck model of adaptive evolution. *Evolution: International Journal of Organic Evolution*, 66(8):2369–2383.

Beck, A. and Teboulle, M. (2004). A conditional gradient method with linear rate of convergence for solving convex linear systems. *Mathematical Methods of Operations Research*, 59(2):235–247.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010a). A theory of learning from different domains. *Machine learning*, 79(1-2):151–175.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2007). Analysis of representations for domain adaptation. In *Advances in Neural Information Processing Systems*, volume 19, pages 137–144.

Ben-David, S., Lu, T., Luu, T., and Pál, D. (2010b). Impossibility theorems for domain adaptation. In *International Conference on Artificial Intelligence and Statistics*, pages 129–136.

Bertsekas, D. P. (1999). *Nonlinear programming*. Athena Scientific, 2nd edition.

Bickel, S., Brückner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In *International Conference on Machine Learning*, pages 81–88.

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *Advances in neural information processing systems*, pages 129–136.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 440–447.

Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In *Advances in neural information processing systems*, pages 161–168.

Bousmalis, K., Trigeorgis, G., Silberman, N., Krishnan, D., and Erhan, D. (2016). Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351.

Bousquet, O., Boucheron, S., and Lugosi, G. (2003). Introduction to statistical learning theory. In *Summer School on Machine Learning*, pages 169–207. Springer.

Butler, M. A. and King, A. A. (2004). Phylogenetic comparative analysis: a modeling approach for adaptive evolution. *The American Naturalist*, 164(6):683–695.

Chen, M., Xu, Z., Weinberger, K. Q., and Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 1627–1634. Omnipress.

Chen, Y., Welling, M., and Smola, A. (2010). Super-samples from kernel herding. In *Uncertainty in Artificial Intelligence*, pages 109–116.

Cortes, C. and Mohri, M. (2011). Domain adaptation in regression. In *International Conference on Algorithmic Learning Theory*, pages 308–323. Springer.

Cortes, C., Mohri, M., and Medina, A. M. (2019). Adaptation based on generalized discrepancy. *The Journal of Machine Learning Research*, 20(1):1–30.

Cortes, C., Mohri, M., Riley, M., and Rostamizadeh, A. (2008). Sample selection bias correction theory. *Algorithmic Learning Theory*, 5254:38–53.

Dai, B., He, N., Pan, Y., Boots, B., and Song, L. (2017). Learning from conditional distributions via dual embeddings. In *Artificial Intelligence and Statistics*, pages 1458–1467.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Duchi, J., Shalev-Shwartz, S., Singer, Y., and Chandra, T. (2008). Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):44.

Ganin, Y. and Lempitsky, V. (2015). Unsupervised domain adaptation by backpropagation. In *International Conference on Machine Learning*, pages 1180–1189.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030.

Gardiner, C. (2009). *Stochastic methods*, volume 4. Springer Berlin.

Gelada, C. and Bellemare, M. G. (2019). Off-policy deep reinforcement learning by bootstrapping the covariate shift. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3647–3655.

Goldfarb, D., Iyengar, G., and Zhou, C. (2017). Linear convergence of stochastic frank wolfe variants. In *Artificial Intelligence and Statistics*, pages 1066–1074.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(Mar):723–773.

Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., and Schölkopf, B. (2009). Covariate shift by kernel mean matching. *Dataset Shift in Machine Learning*, pages 131–160.

Guo, Z., Thomas, P. S., and Brunskill, E. (2017). Using options and covariance testing for long horizon off-policy policy evaluation. In *Advances in Neural Information Processing Systems*, pages 2492–2501.

Hachiya, H., Akiyama, T., Sugiayma, M., and Peters, J. (2009). Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, 22(10):1399–1410.

Hallak, A. and Mannor, S. (2017). Consistent on-line off-policy evaluation. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1372–1383. JMLR. org.

Hardt, M. and Price, E. (2014). The noisy power method: A meta algorithm with applications. In *Advances in Neural Information Processing Systems*, pages 2861–2869.

Haviv, M. (2009). Queues–a course in queueing theory. *The Hebrew University, Jerusalem*, 91905.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hoffman, J., Mohri, M., and Zhang, N. (2018a). Algorithms and theory for multiple-source adaptation. In *Advances in Neural Information Processing Systems*, pages 8246–8256.

Hoffman, J., Tzeng, E., Park, T., Zhu, J.-Y., Isola, P., Saenko, K., Efros, A., and Darrell, T. (2018b). Cycada: Cycle-consistent adversarial domain adaptation. In *International Conference on Machine Learning*, pages 1994–2003.

Hoffman, M. D. and Gelman, A. (2014). The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *Journal of Machine Learning Research*, 15(1):1593–1623.

Jaggi, M. (2013). Revisiting frank-wolfe: Projection-free sparse convex optimization. In *International Conference on Machine Learning*, pages 427–435.

Jiang, J. and Zhai, C. (2007). Instance weighting for domain adaptation in NLP. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 264–271.

Jiang, N. and Li, L. (2016). Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 652–661.

Johansson, F., Sontag, D., and Ranganath, R. (2019). Support and invertibility in domain-invariant representations. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 527–536.

Joulin, A., Tang, K., and Fei-Fei, L. (2014). Efficient image and video co-localization with frank-wolfe algorithm. In *Computer Vision–ECCV 2014*, pages 253–268. Springer.

Kadanoff, L. P. (2000). *Statistical physics: statics, dynamics and renormalization*. World Scientific Publishing Company.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *International Conference on Very Large Data Bases*, volume 30, pages 180–191.

Kim, K. I., Franz, M. O., and Schölkopf, B. (2005). Iterative kernel principal component analysis for image modeling. *IEEE transactions on pattern analysis and machine intelligence*, 27(9):1351–1366.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT press.

Koltchinskii, V., Panchenko, D., et al. (2002). Empirical margin distributions and bounding the generalization error of combined classifiers. *The Annals of Statistics*, 30(1):1–50.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lacoste-Julien, S. and Jaggi, M. (2015). On the global linear convergence of frank-wolfe optimization variants. In *Advances in Neural Information Processing Systems*, pages 496–504.

Lange, S., Gabel, T., and Riedmiller, M. (2012). Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer.

Leacy, F. P. and Stuart, E. A. (2014). On the joint use of propensity and prognostic scores in estimation of the average treatment effect on the treated: a simulation study. *Statistics in medicine*, 33(20):3488–3508.

Levin, D. A. and Peres, Y. (2017). *Markov chains and mixing times*, volume 107. American Mathematical Soc.

Li, C., Bai, K., Li, J., Wang, G., Chen, C., and Carin, L. (2019). Adversarial learning of a sampler based on an unnormalized distribution. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3302–3311.

Li, J.-h. and Tian, N.-s. (2008). Analysis of the discrete time geo/geo/1 queue with single working vacation. *Quality Technology & Quantitative Management*, 5(1):77–89.

Li, L., Chen, S., Kleban, J., and Gupta, A. (2015a). Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th International Conference on World Wide Web*, pages 929–934.

Li, L., Munos, R., and Szepesvari, C. (2015b). Toward minimax off-policy value estimation. In *Artificial Intelligence and Statistics*, pages 608–616.

Li, Y., Carlson, D. E., et al. (2018). Extracting relationships by multi-domain matching. In *Advances in Neural Information Processing Systems*, pages 6798–6809.

Liu, J. S. (2008). *Monte Carlo strategies in scientific computing*. Springer Science & Business Media.

Liu, Q. and Lee, J. (2017). Black-box importance sampling. In *Artificial Intelligence and Statistics*, pages 952–961.

Liu, Q., Lee, J., and Jordan, M. (2016). A kernelized stein discrepancy for goodness-of-fit tests. In *International conference on machine learning*, pages 276–284.

Liu, Q., Li, L., Tang, Z., and Zhou, D. (2018). Breaking the curse of horizon:

Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366.

Long, M., Cao, Y., Wang, J., and Jordan, M. (2015). Learning transferable features with deep adaptation networks. In *International Conference on Machine Learning*, pages 97–105.

Long, M., Cao, Z., Wang, J., and Jordan, M. I. (2018). Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650.

Long, M., Zhu, H., Wang, J., and Jordan, M. I. (2016). Unsupervised domain adaptation with residual transfer networks. In *Advances in Neural Information Processing Systems*, pages 136–144.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009a). Domain adaptation: Learning bounds and algorithms. In *Annual Conference on Learning Theory*.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009b). Domain adaptation with multiple sources. In *Advances in neural information processing systems*, pages 1041–1048.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009c). Multiple source adaptation and the Rényi divergence. In *UAI*, pages 367–374.

Meyn, S., Tweedie, R. L., and Glynn, P. W. (2009). *Markov Chains and Stochastic Stability*. Cambridge Mathematical Library. Cambridge University Press, 2 edition.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press.

Nachum, O., Chow, Y., Dai, B., and Li, L. (2019). Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In *Advances in Neural Information Processing Systems*, pages 2318–2328.

Neal, R. M. (2003). Slice sampling. *Annals of statistics*, pages 705–741.

Neal, R. M. et al. (2011). Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2007). Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *Advances in neural information processing systems*, pages 1089–1096.

Oates, C. J., Girolami, M., and Chopin, N. (2017). Control functionals for monte carlo integration. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(3):695–718.

Oksendal, B. (2013). *Stochastic differential equations: an introduction with applications*. Springer Science & Business Media.

Pan, S. J., Tsang, I. W., Kwok, J. T., and Yang, Q. (2009). Domain adaptation via transfer component analysis. In *International Joint Conference on Artifical Intelligence*, pages 1187–1192.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Pei, Z., Cao, Z., Long, M., and Wang, J. (2018). Multi-adversarial domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. (2019). Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415.

Precup, D., Sutton, R. S., and Dasgupta, S. (2001). Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424.

Puterman, M. L. (2014). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538.

Robert, C. and Casella, G. (2013). *Monte Carlo statistical methods*. Springer Science & Business Media.

Ross, S., Gordon, G., and Bagnell, D. (2011). A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635.

Rubinstein, R. Y. and Kroese, D. P. (2016). *Simulation and the Monte Carlo method*, volume 10. John Wiley & Sons.

Ryu, E. K. and Boyd, S. (2016). Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43.

Saito, K., Watanabe, K., Ushiku, Y., and Harada, T. (2018). Maximum classifier discrepancy for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3723–3732.

Salamatian, S., Fawaz, N., Kveton, B., and Taft, N. (2014). Sppm: Sparse privacy preserving mappings. In *Conference on Uncertainty in Artificial Intelilgence*.

Santana, S. E., Grosse, I. R., and Dumont, E. R. (2012). Dietary hardness, loading behavior, and the evolution of skull form in bats. *Evolution: International Journal of Organic Evolution*, 66(8):2587–2598.

Schoenauer-Sebag, A., Heinrich, L., Schoenauer, M., Sebag, M., Wu, L., and Altschuler, S. (2019). Multi-domain adversarial learning. In *International Conference on Learning Representations*.

Serfozo, R. (2009). *Basics of applied stochastic processes*. Springer Science & Business Media.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Song, J., Zhao, S., and Ermon, S. (2017). A-nice-mc: Adversarial training for mcmc. In *Advances in Neural Information Processing Systems*, pages 5140–5150.

Sugiyama, M. and Kawanabe, M. (2012). *Machine learning in non-stationary environments: introduction to covariate shift adaptation*. The MIT Press.

Sugiyama, M., Nakajima, S., Kashima, H., Von Buenau, P., and Kawanabe, M. (2007). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012a). *Density ratio estimation in machine learning*. Cambridge University Press.

Sugiyama, M., Suzuki, T., and Kanamori, T. (2012b). 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. Publisher: Springer.

Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., von Bünau, P., and Kawanabe, M. (2008). Direct importance estimation for covariate shift adaptation. *Annals of the Institute of Statistical Mathematics*, 60(4):699–746.

Sun, Q., Chattopadhyay, R., Panchanathan, S., and Ye, J. (2011). A two-stage weighting framework for multi-source domain adaptation. In *Advances in neural information processing systems*, pages 505–513.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Thomas, P. and Brunskill, E. (2016). Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148.

Tsuboi, Y., Kashima, H., Hido, S., Bickel, S., and Sugiyama, M. (2009). Direct density ratio estimation for large-scale covariate shift adaptation. *Information and Media Technologies*, 4(2):529–546.

Tzeng, E., Hoffman, J., Darrell, T., and Saenko, K. (2015). Simultaneous deep transfer across domains and tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4068–4076.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. (2017). Deep hashing network for unsupervised domain adaptation. In *(IEEE) Conference on Computer Vision and Pattern Recognition (CVPR)*.

Wang, F., Wang, J., and Zhang, F. (2014). Equilibrium customer strategies in the geo/geo/1 queue with single working vacation. *Discrete Dynamics in Nature and Society*, 2014.

Wang, X. and Schneider, J. (2014). Flexible transfer learning under support and model shift. In *Advances in Neural Information Processing Systems*, pages 1898–1906.

Wang, Y.-X., Agarwal, A., and Dudik, M. (2017). Optimal and adaptive off-policy evaluation in contextual bandits. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3589–3597. JMLR.org.

Welling, M. (2009). Herding dynamical weights to learn. In *International Conference on Machine Learning*, pages 1121–1128.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

Wen, J., Dai, B., Li, L., and Schuurmans, D. (2020a). Batch stationary distribution estimation. In *International Conference on Machine Learning*.

Wen, J., Greiner, R., and Schuurmans, D. (2015). Correcting covariate shift with the frank-wolfe algorithm. In *International Joint Conference on Artifical Intelligence*, pages 1010–1016.

Wen, J., Greiner, R., and Schuurmans, D. (2020b). Domain aggregation networks for multi-source domain adaptation. In *International Conference on Machine Learning*.

Wen, J., Yu, C.-N., and Greiner, R. (2014). Robust learning under uncertain test distributions: Relating covariate shift to model misspecification. In *International Conference on Machine Learning*, pages 631–639.

Wenliang, L., Sutherland, D., Strathmann, H., and Gretton, A. (2019). Learning deep kernels for exponential family densities. In *International Conference on Machine Learning*, pages 6737–6746.

Xie, B., Liang, Y., and Song, L. (2015). Scale up nonlinear component analysis with doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 2341–2349.

Xie, T., Ma, Y., and Wang, Y.-X. (2019). Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. In *Advances in Neural Information Processing Systems 32*, pages 9665–9675.

Yamada, M., Sigal, L., and Raptis, M. (2012). No bias left behind: Covariate shift adaptation for discriminative 3d pose estimation. In *Computer Vision–ECCV 2012*, pages 674–687. Springer.

Yamada, M., Sugiyama, M., and Matsui, T. (2010). Semi-supervised speaker identification under covariate shift. *Signal Processing*, 90(8):2353–2361.

Yamada, M., Suzuki, T., Kanamori, T., Hachiya, H., and Sugiyama, M. (2011). Relative density-ratio estimation for robust distribution comparison. In *Advances in Neural Information Processing Systems*, pages 594–602.

Yang, L. F., Yu, Z., Braverman, V., Zhao, T., and Wang, M. (2019). Online factorization and partition of complex networks by random walk. In Globerson, A. and Silva, R., editors, *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, UAI 2019, Tel Aviv, Israel, July 22-25, 2019*. AUAI Press.

Zadrozny, B. (2004). Learning and evaluating classifiers under sample selection bias. In *International Conference on Machine Learning*, page 114. ACM.

Zhang, K., Gong, M., and Schölkopf, B. (2015). Multi-source domain adaptation: A causal view. In *Twenty-ninth AAAI conference on artificial intelligence*.

Zhang, K., Muandet, K., Wang, Z., et al. (2013). Domain adaptation under target and conditional shift. In *Proceedings of the 30th International Conference on Machine Learning*, pages 819–827.

Zhang, X., Schuurmans, D., and Yu, Y.-l. (2012). Accelerated training for matrix-norm regularization: A boosting approach. In *Advances in Neural Information Processing Systems*, pages 2906–2914.

Zhao, H., Des Combes, R. T., Zhang, K., and Gordon, G. (2019). On learning invariant representations for domain adaptation. In *International Conference on Machine Learning*, pages 7523–7532.

Zhao, H., Zhang, S., Wu, G., Moura, J. M., Costeira, J. P., and Gordon, G. J. (2018). Adversarial multiple source domain adaptation. In *Advances in Neural Information Processing Systems*, pages 8559–8570.

Appendix A

Details on DARN

A.1 Proof of Theorem 2

Theorem 2 *Given* k *source domains datasets* $\{(x_j^{(i)}, y_j^{(i)}) : i \in [k], j \in [m]\}$ *with* m *iid examples each where* $\widehat{S}_i = \{x_j^{(i)}\}$ *and* $y_j^{(i)} = f_{S_i}(x_j^{(i)})$*, for any* $\boldsymbol{\alpha} \in \Delta = \{\boldsymbol{\alpha} : \alpha_i \geqslant 0, \sum_i \alpha_i = 1\}$ *and* $\delta \in (0,1)$*, w.p. at least* $1-\delta$*, the following holds for all* $h \in \mathcal{H}$*,*

$$\mathcal{L}_T(h, f_T) \leqslant \sum_i \alpha_i \left[\mathcal{L}_{\widehat{S}_i}(h, f_{S_i}) + \mathrm{disc}(T, S_i) + 2\mathfrak{R}_m(\mathcal{H}_{S_i}) + \eta_{\mathcal{H},i}\right] + \|\boldsymbol{\alpha}\|_2 M_S \sqrt{\frac{\log(1/\delta)}{2m}},$$

where $\mathcal{H}_{S_i} = \{x \mapsto L(h(x), f_{S_i}(x)) : h \in \mathcal{H}\}$ *is the set of functions mapping* x *to the corresponding loss,*

$$\eta_{\mathcal{H},i} = \mu \times \min_{h \in \mathcal{H}} \left(\max_{x \in \mathrm{supp}(\widehat{T})} |f_T(x) - h(x)| + \max_{x \in \mathrm{supp}(\widehat{S}_i)} |f_{S_i}(x) - h(x)| \right),$$

and $M_S = \sup_{i \in [k], x \in \mathcal{X}, h \in \mathcal{H}} L(h(x), f_{S_i}(x))$ *is the upper bound on loss on the source domains.*

Proof. Given $\boldsymbol{\alpha} \in \Delta$, the mixture $\widehat{S} = \sum_i \alpha_i \widehat{S}_i$ can be considered as the joint source data with km points, where a point $x^{(i)}$ from $\widehat{S}_i$ has weight α_i/m. Define $\Phi = \sup_{h \in \mathcal{H}} \mathcal{L}_T(h, f_T) - \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})$. Changing a point $x^{(i)}$ from $\widehat{S}_i$ will change Φ at most $\frac{M_S \alpha_i}{m}$. Using the McDiarmid's inequality, we have $\Pr(\Phi - \mathbb{E}[\Phi] > \epsilon) \leqslant \exp\left(-\frac{2\epsilon^2 m}{M_S^2 \|\boldsymbol{\alpha}\|_2^2}\right)$. As a result, for $\delta \in (0,1)$, w.p. at least

$1-\delta$, the following holds for any $h \in \mathcal{H}$

$$\mathcal{L}_T(h, f_T) \leqslant \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i}) + \mathbb{E}[\Phi] + \|\boldsymbol{\alpha}\|_2 M_S \sqrt{\frac{\log(1/\delta)}{2m}}.$$

Now we bound $\mathbb{E}[\Phi]$. Let $\mathcal{H}_{S_i} = \{x \mapsto L(h(x), f_{S_i}(x)) : h \in \mathcal{H}\}$.

$$\begin{aligned}
\mathbb{E}[\Phi] &= \mathbb{E}_{\widehat{S}}\left[\sup_{h\in\mathcal{H}} \mathcal{L}_T(h, f_T) - \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right] \\
&\leqslant \mathbb{E}_{\widehat{S}}\left[\sup_{h\in\mathcal{H}} \sum_i \alpha_i \mathcal{L}_{S_i}(h, f_{S_i}) - \sum_i \alpha_i \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right] \\
&\qquad + \sup_{h\in\mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \sum_i \alpha_i \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&\leqslant \mathbb{E}_{\widehat{S}}\left[\sum_i \alpha_i \sup_{h\in\mathcal{H}}\left(\mathcal{L}_{S_i}(h, f_{S_i}) - \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right)\right] \\
&\qquad + \sum_i \alpha_i \sup_{h\in\mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&= \sum_i \alpha_i \mathbb{E}_{\widehat{S}_i}\left[\sup_{h\in\mathcal{H}}\left(\mathcal{L}_{S_i}(h, f_{S_i}) - \mathcal{L}_{\widehat{S}_i}(h, f_{S_i})\right)\right] \\
&\qquad + \sum_i \alpha_i \sup_{h\in\mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&\leqslant 2\sum_i \alpha_i \mathfrak{R}_m(\mathcal{H}_{S_i}) + \sum_i \alpha_i \sup_{h\in\mathcal{H}}\left(\mathcal{L}_T(h, f_T) - \mathcal{L}_{S_i}(h, f_{S_i})\right) \\
&\leqslant \sum_i \alpha_i \left(2\mathfrak{R}_m(\mathcal{H}_{S_i}) + \operatorname{disc}(T, S_i) + \eta_{\mathcal{H},i}\right).
\end{aligned}$$

where first and second inequalities are using the subadditivity of sup, followed by the equality using the independence between the domains $\{\widehat{S}_i\}$, the second last inequality is due to the standard "ghost sample" argument in terms of the Rademacher complexity and the last inequality is due to Cortes et al. (2019, Proposition 8) for each individual S_i. □

A.2 Jacobian

Here we calculate the Jacobian $J_{ij} = \partial\alpha_i/\partial z_j$ for Eq. (3.7) in the main text:

$$\boldsymbol{\alpha}^* = [\mathbf{z} - \nu^*\mathbf{1}]_+ / \|[\mathbf{z} - \nu^*\mathbf{1}]_+\|_1.$$

In the following, we write $\boldsymbol{\alpha} = \boldsymbol{\alpha}^*, \nu = \nu^*$ to simplify notations. Let $S = \{i : z_i - \nu > 0\}$ be the support of the probability vector $\boldsymbol{\alpha}$. $J_{ij} = 0$ if $i \notin S$ or $j \notin S$ since $\alpha_i = 0$ in the former case while z_j does not contribute to the $\boldsymbol{\alpha}$ in the latter case. Now consider the case $i, j \in S$. Let $K = \|[\mathbf{z} - \nu\mathbf{1}]_+\|_1 = \sum_{j\in S}(z_j - \nu)$. Then $\alpha_i = (z_i - \nu)\cdot\frac{1}{K}$ and

$$\begin{aligned}\frac{\partial\alpha_i}{\partial z_j} &= \left(\delta_{i=j} - \frac{\partial\nu}{\partial z_j}\right)\cdot\frac{1}{K} - \frac{1}{K^2}\cdot\frac{\partial K}{\partial z_j}\cdot(z_i - \nu) \\ &= \frac{1}{K}\left(\delta_{i=j} - \frac{\partial\nu}{\partial z_j} - \frac{\partial K}{\partial z_j}\cdot\alpha_i\right),\end{aligned} \tag{A.1}$$

where $\delta_{i=j}$ is the indicator or delta function. Now we compute $\frac{\partial\nu}{\partial z_j}$ and $\frac{\partial K}{\partial z_j}$. By the definition of ν, we know that

$$\begin{aligned}&\sum_{j\in S}(z_j - \nu)^2 = |S|\nu^2 - 2\nu\sum_{j\in S} z_j + \sum_{j\in S} z_j^2 = 1 \\ \implies\quad &\nu = \frac{\sum_{j\in S} z_j}{|S|} - \frac{\sqrt{A}}{|S|} \quad \text{where} \quad A = \left(\sum_{j\in S} z_j\right)^2 - |S|\left(\sum_{j\in S} z_j^2 - 1\right) \\ \implies\quad &\frac{\partial\nu}{\partial z_j} = \frac{1}{|S|} - \frac{B_j}{|S|\sqrt{A}} \quad \text{where} \quad B_j = \sum_{j'\in S} z_{j'} - |S|z_j\end{aligned} \tag{A.2}$$

The first line is due to the quadratic formula and realizing that $\sum_{j\in S} z_j/|S|$ is the mean of the supported z_j so ν must be smaller than it (i.e., we take $-$ in the $\pm$ of the quadratic formula, otherwise some of the z_j will not be in the support anymore). And

$$\frac{\partial K}{\partial z_j} = 1 - |S|\cdot\frac{\partial\nu}{\partial z_j} = \frac{B_j}{\sqrt{A}}. \tag{A.3}$$

Plugging Eq. (A.2) and Eq. (A.3) in Eq. (A.1) gives

$$\frac{\partial\alpha_i}{\partial z_j} = \frac{1}{K}\left[\delta_{i=j} - \frac{1}{|S|} + \frac{B_j}{\sqrt{A}}\cdot\left(\frac{1}{|S|} - \alpha_i\right)\right]$$

Note that

$$\frac{B_j}{K} = \frac{\sum_{j' \in S} z_{j'} - |S| z_j}{\sum_{j' \in S}(z_{j'} - \nu)} = \frac{\sum_{j' \in S}(z_{j'} - \nu) + |S|(\nu - z_j)}{\sum_{j' \in S}(z_{j'} - \nu)} = 1 - |S|\alpha_j.$$

Then

$$J_{ij} = \frac{\partial \alpha_i}{\partial z_j} = \frac{1}{K}\left(\delta_{i=j} - \frac{1}{|S|}\right) + \frac{|S|}{\sqrt{A}}\left(\frac{1}{|S|} - \alpha_i\right)\left(\frac{1}{|S|} - \alpha_j\right).$$

In matrix form,

$$J = \frac{1}{K}\left(\mathrm{Diag}(\mathbf{s}) - \frac{\mathbf{s}\mathbf{s}^\top}{|S|}\right) + \frac{|S|}{\sqrt{A}}\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)^\top,$$

where $\mathbf{s} = [s_1, \ldots, s_k]^\top$ is a vector indicating the support $s_i = \delta_{i \in S}$ and $\circ$ is element-wise multiplication. More often, we need to compute its multiplication with a vector $\mathbf{v}$

$$J\mathbf{v} = \frac{\mathbf{s}}{K} \circ \left(\mathbf{v} - \frac{\mathbf{s}^\top \mathbf{v}}{|S|}\right) + \frac{|S|}{\sqrt{A}}\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)\left(\frac{\mathbf{s}}{|S|} - \boldsymbol{\alpha} \circ \mathbf{s}\right)^\top \mathbf{v}.$$

Note that all quantities except A have been computed during the forward pass of calculating Eq. (3.7). A can be computed in $O(|S|)$ time so the overall computation is still $O(k)$ since $|S| \leqslant k$.

A.3 Experiment Details

The following provides additional details of the experiments.

A.3.1 Regression

For the eight source domains, the ith ($i = \{0, 1, \ldots, 7\}$) domain data is generated by $x_i \sim \mathcal{N}(\frac{\pi}{4}i - \frac{7\pi}{8}, 0.2^2)$ and the output is $y = \sin(x) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, 0.05^2)$ is random noise. For the four target domains, the jth ($j = \{0, 1, 2, 3\}$) domain is generated by $x_j \sim \mathcal{N}(\frac{\pi}{2}j - \frac{3\pi}{4}, 0.4^2)$.

A.3.2 Digit Recognition

MNIST images are resized to 32×32 and represented as 3-channel color images in order to match the shape of the other three datasets. Each domain has its own given training and test sets when downloaded. Their respective training sample sizes are 60000, 59001, 73257, 479400, and the respective test sample sizes are 10000, 9001, 26032, 9553. In each run, 20000 images are randomly sampled from each domain's training set as actual labelled source or unlabelled target training examples, and 9000 images are randomly sampled from each domain's test set as actual test examples for evaluation. The model structure is shown in Fig. A.1. There is no dropout and the hyper-parameters are chosen based on cross-validation. It is trained for 50 epochs and the mini-batch size is 128 per domain. The optimizer is Adadelta with a learning rate of 1.0. The soft version of MDAN has an additional parameter $\gamma = 1/\tau$ which is the inverse of our temperature τ. $\gamma = 0.5$ is used for MDAN and $\gamma = 0.1$ for DARN.

A.3.3 Object Recognition: Office-Home

For the four domains, Art, Clipart, Product and Real-World, the respective sample sizes are 2427, 4365, 4439, 4357. In each run, 2000 images are randomly sampled from each domain as labelled source or unlabelled target training examples, and the rest images are used as test images for evalu-

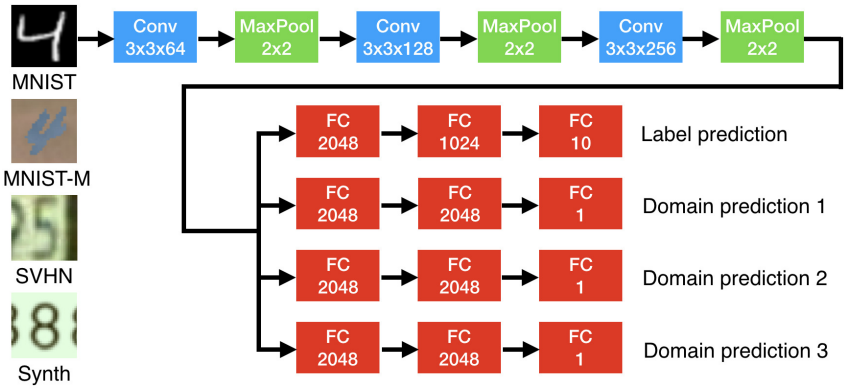


Figure A.1: Model architecture for the digit recognition.

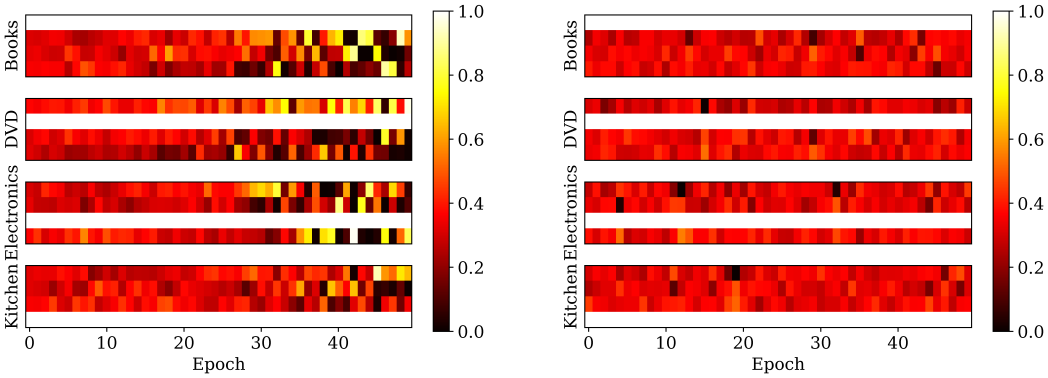
ation. We use the ResNet50 (He et al., 2016) pretrained features from the ImageNet as the base network for feature learning and put an MLP with [1000, 500, 100, 65] units on top for classification. It is trained for 50 epochs and the mini-batch size is 32 per domain. The optimizer is Adadelta with a learning rate of 1.0. MDAN uses $\gamma = 1.0$ while DARN uses $\gamma = 0.5$.

A.3.4 Sentiment Analysis

The respective sample sizes for the Books, DVD, Electronics and Kitchen domains are 6465, 5586, 7681, 7945. We train a fully connected model (MLP) with [1000, 500, 100] hidden units for classifying positive versus negative reviews. The dropout drop rate is 0.7 for the input and hidden layers. In each run, we randomly sample 2000 reviews from each domain as labelled source or unlabelled target training examples, while the remaining instances are used as test examples for evaluation. The hyper-parameters are chosen based on cross-validation. The model is trained for 50 epochs and the mini-batch size is 20 per domain. The optimizer is Adadelta with a learning rate of 1.0. The chosen parameters are $\gamma = 10.0$ for MDAN and $\gamma = 0.9$ for our DARN, which are selected from a wide range of candidate values.

Fig. A.2a and Fig. A.2b show the domain weights of MDMN and DARN *without* exponential average smoothing. They correspond to Fig. 3.5a and Fig. 3.5b in the main text. Although both can have certain instability due

to small mini-batch size, MDMN is noticeably less stable, especially towards the end of the training, in which alternating one-hot weights can occur (e.g., epochs 40-50 for the Books target domain). This makes their weights hard to interpret.



(a) MDMN (b) DARN

Figure A.2: Domain weights for the Amazon data without smoothing.

Appendix B

Details on VPM

B.1 Consistency of the Objectives

Theorem 3 (Normalization of solution) *If* $\mathbb{E}_p[\tau_t] = 1$, *then for any* $\lambda > 0$, *the estimator* (4.13) *has the same solution as* (4.12), *hence* $\mathbb{E}_p[\tau_{t+1}] = 1$.

Proof. Taking derivative of the objective function in (4.12) and setting it to zero, we can see that the unconstrained solution is $\frac{\mathcal{T}_p \tau_t}{p}$. Moreover, it satisfies the constraint when $\mathbb{E}_p[\tau_t] = 1$: we can rewrite $\tau_t = \frac{\mu_t}{p}$ for some distribution μ_t and $\mathbb{E}_p\left[\frac{\mathcal{T}_p \tau_t}{p}\right] = \int \mathcal{T}(x'|x)\, \mu_t(x)\, dx dx' = 1$.

We just need to show $\frac{\mathcal{T}_p \tau_t}{p}$ is also the solution to (4.13). First, note that for any primal τ, the optimal dual ν can be attained at $\nu = \frac{1}{\lambda}(\mathbb{E}_p[\tau] - 1)$. Plugging it to (4.13), we have for any $\lambda > 0$

$$\begin{aligned}
&\min_{\tau \geqslant 0} \frac{1}{2} \mathbb{E}_{p(x')}\left[\tau^2(x')\right] - \mathbb{E}_{\mathcal{T}_p(x,x')}\left[\tau_t(x)\, \tau(x')\right] + \frac{1}{2\lambda}(\mathbb{E}_p[\tau] - 1)^2 \\
&\geqslant \min_{\tau \geqslant 0} \frac{1}{2} \mathbb{E}_{p(x')}\left[\tau^2(x')\right] - \mathbb{E}_{\mathcal{T}_p(x,x')}\left[\tau_t(x)\, \tau(x')\right] + \min_{\tau \geqslant 0} \frac{1}{2\lambda}(\mathbb{E}_p[\tau] - 1)^2 \\
&= -\tfrac{1}{2}\mathbb{E}_p\left[\left(\frac{\mathcal{T}_p \tau_t}{p}\right)^2\right].
\end{aligned} \tag{B.1}$$

This lower bound is attainable by plugging $\tau = \frac{\mathcal{T}_p \tau_t}{p}$ in (4.13). Finally, we conclude the proof by noticing that (4.13) is strictly convex so the optimal solution is unique. □

B.2 Convergence Analysis

Let (X, Σ, ν) be a measure space. The $\mathcal{L}^2(X)$ space consists of measurable functions $f : X \mapsto \mathbb{R}$ such that $\|f\| = (\int |f|^2 d\nu)^{1/2} < \infty$. Suppose the initial $\mu_0 \in \mathcal{L}^2(X)$, we want to show the converging behavior of the following damped iteration:

$$\begin{aligned} \mu_t &= (1-\alpha_t)\mu_{t-1} + \alpha_t \widehat{\mathcal{T}}\mu_{t-1} \\ &= (1-\alpha_t)\mu_{t-1} + \alpha_t \mathcal{T}\mu_{t-1} + \alpha_t \epsilon \end{aligned} \tag{B.2}$$

with suitable step-sizes $\alpha_t \in (0, 1)$, where $\epsilon \in \mathcal{L}^2(X)$ is a random field due to stochacity in $\widehat{\mathcal{T}}$. To this end, we will use the following lemma.

Lemma 5. *For $\alpha \in \mathbb{R}, f, g \in \mathcal{L}^2(X)$*

$$\|(1-\alpha)f + \alpha g\|^2 = (1-\alpha)\|f\|^2 + \alpha\|g\|^2 - \alpha(1-\alpha)\|f-g\|^2.$$

This can be proved by expanding both sides. Now we state our main convergence result.

Theorem 4 *Suppose $\mu_0 \in \mathcal{L}^2(X)$, the step size is $\alpha_t = 1/\sqrt{t}$, $\epsilon \in \mathcal{L}^2(X)$ is a random field and $\mathcal{T}$ has a unique stationary distribution μ. After t iterations, define the probability distribution over the iterations as*

$$\Pr(R = k) = \frac{\alpha_k(1-\alpha_k)}{\sum_{k'=1}^{t} \alpha_{k'}(1-\alpha_{k'})}$$

Then there exist some constants $C_1, C_2 > 0$ such that

$$\mathbb{E}\left[\|\mu_R - \mathcal{T}\mu_R\|_2^2\right] \leqslant \frac{C_1}{\sqrt{t}}\|\mu_0 - \mu\|_2^2 + \frac{C_2 \ln t}{\sqrt{t}}\|\epsilon\|_2^2,$$

where the expectation is taken over R. Consequently, μ_R converges to μ for ergodic $\mathcal{T}$.

Proof. Using Lemma 5 and the fact that $\mathcal{T}$ is non-expansive, we have

$$\begin{aligned} \|\mu_t - \mu\|^2 &= \|(1-\alpha_t)(\mu_{t-1} - \mu) + \alpha_t(\mathcal{T}\mu_{t-1} - \mu) + \alpha_t\epsilon\|^2 \\ &\leqslant \|(1-\alpha_t)(\mu_{t-1} - \mu) + \alpha_t(\mathcal{T}\mu_{t-1} - \mu)\|^2 + \alpha_t^2\|\epsilon\|^2 \\ &\leqslant (1-\alpha_t)\|\mu_{t-1} - \mu\|^2 + \alpha_t\|\mathcal{T}\mu_{t-1} - \mu\|^2 \\ &\quad - \alpha_t(1-\alpha_t)\|\mu_{t-1} - \mathcal{T}\mu_{t-1}\|^2 + \alpha_t^2\|\epsilon\|^2 \\ &\leqslant \|\mu_{t-1} - \mu\|^2 - \alpha_t(1-\alpha_t)\|\mu_{t-1} - \mathcal{T}\mu_{t-1}\|^2 + \alpha_t^2\|\epsilon\|^2. \end{aligned}$$

Then telescoping sum gives

$$0 \leqslant \|\mu_t - \mu\|^2 \leqslant \|\mu_0 - \mu\|^2 + \sum_{k=1}^{t} \alpha_k^2 \|\epsilon\|^2 - \sum_{k=1}^{t} \alpha_k (1 - \alpha_k) \|\mu_k - \mathcal{T}\mu_k\|^2$$

So

$$\sum_{k=1}^{t} \alpha_k (1 - \alpha_k) \|\mu_k - \mathcal{T}\mu_k\|^2 \leqslant \|\mu_0 - \mu\|^2 + \sum_{k=1}^{t} \alpha_k^2 \|\epsilon\|^2.$$

Divide both sides by $\sum_{k=1}^{t} \alpha_k (1 - \alpha_k)$ (taking expectation over iterations) gives

$$\begin{aligned}
\mathbb{E}[\|\mu_R - \mathcal{T}\mu_R\|^2] &= \sum_{k=1}^{t} \frac{\alpha_k (1 - \alpha_k)}{\sum_{k'} \alpha_{k'} (1 - \alpha_{k'})} \|\mu_k - \mathcal{T}\mu_k\|^2 \\
&\leqslant \frac{\|\mu_0 - \mu\|^2 + \sum_{k=1}^{t} \alpha_k^2 \|\epsilon\|^2}{\sum_{k=1}^{t} \alpha_k (1 - \alpha_k)}.
\end{aligned}$$

When $\alpha_t = 1/\sqrt{t}$, we have

$$\begin{aligned}
\sum_{k=1}^{t} \alpha_k^2 \|\epsilon\|^2 &= \sum_{k=1}^{t} \frac{1}{k} \|\epsilon\|^2 \leqslant (\ln t + 1) \|\epsilon\|^2 \\
\sum_{k=4}^{t} \alpha_k (1 - \alpha_k) &= \sum_{k=4}^{t} \frac{1}{\sqrt{k}} - \frac{1}{k} \geqslant \int_4^t \left(\frac{1}{\sqrt{k+1}} - \frac{1}{k+1} \right) dk = \Omega\left(t^{\frac{1}{2}} \right)
\end{aligned}$$

So for big enough t, there exists $C_0 > 0$ such that

$$\mathbb{E}\left[\|\mu_R - \mathcal{T}\mu_R\|^2 \right] \leqslant \frac{\|\mu_0 - \mu\|^2 + \ln(t+1) \|\epsilon\|^2}{C_0 \sqrt{t}},$$

which leads to the the bound in the theorem and $\mathbb{E}\left[\|\mu_R - \mathcal{T}\mu_R\|^2 \right] = \widetilde{\mathcal{O}}\left(t^{-1/2} \right)$. Additionally, since $\mathcal{T}$ has a unique stationary distribution $\mu = \mathcal{T}\mu$, we have μ_R converges to μ. □

B.3 Application to Off-policy Stationary Ratio Estimation

We provide additional details describing how the variational power method we have developed in the main body can be applied to the behavior-agnostic off-policy estimation problem (OPE). The general framework has been introduced in Section 4.2 and the implementation for the undiscounted case ($\gamma = 1$) is demonstrated in Section 4.6.4. Specifically, given a sample $\mathcal{D} = \left\{(s, a, r, s')_{i=1}^n\right\}$ from the behavior policy, we compose each transition in $\mathcal{D}$ with a target action $a' \sim \pi(\cdot|s')$. Denoting $x = (s, a)$, the data set can be expressed as $\mathcal{D} = \left\{(x, x')_{i=1}^n\right\}$. Applying the proposed VPM with $\mathcal{T}(x'|x)$, we can estimate $\frac{\mu(s,a)}{p(s,a)}$. Here the $\mu(s, a) = d_\pi(s)\pi(a|s)$ consists of the stationary state occupancy d_π and the target policy π, while $p(s, a)$ is the data-collecting distribution. Then the average accumulated reward can be obtained via (4.3).

Here we elaborate on how the discounted case (*i.e.*, $\gamma \in (0, 1)$) can be handled by our method. We first introduce essential quantities similar to the undiscounted setting. For a trajectory generated stochastically using policy π from an initial state s_0: $(s_0, a_0, r_0, s_1, a_1, r_1, \ldots)$, where $a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim P(\cdot|s_t, a_t)$ and $r_t \sim R(s_t, a_t)$, the the policy value is

$$\rho_\gamma(\pi) := (1-\gamma)\,\mathbb{E}_{s_0\sim\mu_0, a\sim\pi, s'\sim P}\left[\sum_{t=0}^\infty \gamma^t r_t\right],$$

where μ_0 is the initial-state distribution. Denote

$$d_t^\pi(s, a) = \mathbb{P}\left(s_t = s, a_t = a \middle| \begin{bmatrix} s_0 \sim \mu_0, \forall i < t, \\ a_i \sim \pi(\cdot|s_i), \\ s_{i+1} \sim P(\cdot|s_i, a_i) \end{bmatrix}\right).$$

The discounted occupancy distribution is

$$\mu_\gamma(s, a) := (1-\gamma)\sum_{t=0}^\infty \gamma^t d_t^\pi(s, a). \tag{B.3}$$

Then, we can re-express the discounted accumulated reward via μ_γ and the stationary density ratio,

$$\rho_\gamma(\pi) = \mathbb{E}_{(s,a)\sim\mu_\gamma(s,a)}[r(s, a)] = \mathbb{E}_{(s,a)\sim p(s,a)}\left[\frac{\mu_\gamma(s, a)}{p(s, a)} r(s, a)\right]. \tag{B.4}$$

The proposed VPM is applicable to estimating the density ratio in this discounted case. Denoting $x = (s, a)$, $x' = (s', a')$ respectively for notational consistency, we expand μ_γ and use the definition of d_t^π:

$$\begin{aligned}
&\mu_\gamma\left(s', a'\right) = (1-\gamma)\mu_0\left(s'\right)\pi\left(a'|s'\right) + \gamma\int \pi\left(a'|s'\right) P\left(s'|s, a\right)\mu_\gamma\left(s, a\right) ds\ da \\
&\implies p\left(x'\right)\tau^*\left(x'\right) = (1-\gamma)\,\mu_0\pi\left(x'\right) + \gamma\int \mathcal{T}_p\left(x, x'\right)\tau^*\left(x\right) dx, \qquad \text{(B.5)}
\end{aligned}$$

where $\mu_0\pi\left(x'\right) = \mu_0\left(s'\right)\pi\left(a'|s'\right)$ and $\mathcal{T}_p\left(x, x'\right) = \pi\left(a'|s'\right) P\left(s'|s, a\right) p\left(s, a\right)$.

It has been shown that the RHS of (B.5) is contractive (Sutton and Barto, 2018; Mohri et al., 2018), therefore, the fix-point iteration,

$$p\left(x'\right)\tau_{t+1}\left(x'\right) = (1-\gamma)\,\mu_0\pi\left(x'\right) + \gamma\int \mathcal{T}_p\left(x, x'\right)\tau_t\left(x\right) dx, \qquad \text{(B.6)}$$

converges to the true τ as $t \to \infty$, provided the update above is carried out exactly. Compared to (4.6), we can see that the RHS of (B.6) is now a mixture of $\mu_0\pi$ and $\mathcal{T}_p$, with respective coefficients $(1-\gamma)$ and γ.

Similarly, we construct the $(t+1)$-step variational update as

$$\begin{aligned}
\tau_{t+1} = \arg\min_{\tau \geqslant 0}\ &\tfrac{1}{2}\mathbb{E}_{p(x')}\left[\tau^2\left(x'\right)\right] - \gamma\mathbb{E}_{\mathcal{T}_p(x,x')}\left[\tau_t\left(x\right)\tau\left(x'\right)\right] \\
&- (1-\gamma)\,\mathbb{E}_{\mu_0 p(x')}\left[\tau\left(x'\right)\right] + \lambda\left(\mathbb{E}_p\left[\tau\right] - 1\right)^2.
\end{aligned} \qquad \text{(B.7)}$$

Compared to (4.11), we see that the main difference is the third term of (B.7) involves the initial distribution. As $\gamma \to 1$, (B.7) reduces to (4.11).

B.4 Experiment Details

Here we provide additional details about the experiments. In all experiments, the regularization $\lambda = 0.5$ and the optimizer is Adam with $\beta_1 = 0.5$. The τ model is a neural network with 2 hidden layers of 64 units each with ReLU activation and softplus activation for the output.

B.4.1 Queueing

For Geo/Geo/1 queue, when the arrival and finish probabilities are $q_a, q_f \in (0, 1)$ respectively with $q_f > q_a$, the stationary distribution is $P(X = i) = (1 - \rho)\rho^i$ where $\rho = q_a(1 - q_f)/[q_f(1 - q_a)]$ (Serfozo, 2009, Sec.1.11). The defaults are $(n, q_a, q_f) = (100, 0.8, 0.9)$ for the figures. ρ is called *traffic intensity* in the queueing literature and we set $B = \lceil 40\rho \rceil$ in the experiment. The mean and standard error of the log KL divergence is computed based on 10 runs. We conduct closed-form update for 1000 steps. As for the model-based method, we simulate the transition chain for 200 steps to attain the estimated stationary distribution.

B.4.2 Solving SDEs

Using initial samples are uniformly spaced in $[0, 1]$, we run the Euler-Maruyama (EM) method and evaluate the MMD along the path. The τ model is a neural network with 2 hidden layers of 64 units each with ReLU, and Softplus for the final layer. Numbers of outer and inner steps are $T = 50, M = 10$. The learning rate is 0.0005. At each evaluation time step t, we use the most recent 1% of evolution data to train our model τ. The plots are reporting the mean and standard deviation over 10 runs. For the phylogeny studies, the number of particles is 1k and $dt = 0.0005$ for the EM simulation, while the rest settings using $dt = 0.001$.

B.4.3 Post-processing MCMC

The potential functions are collected from several open-source projects[1][2]. 50k examples are sampled from the uniform distribution

$$p(x) = \text{Unif}(x; [-6, 6]^2),$$

then transition each x through an HMC operator (one leapfrog step of size 0.5). The model-based $\widehat{\mathcal{T}}$ has a similar structure as τ except the final layer has 2D output without activation to estimate the Gaussian mean. The mini-batch size is $B = 1k$, the maximum number of power iterations $T = 150$ and the number of inner optimization steps is $M = 10$. The model-based $\mathcal{T}$ is given the same number of iterations ($MT = 1500$). The learning rate is 0.001 for τ and 0.0005 for $\widehat{\mathcal{T}}$. To compute the model-based sample, we apply the estimated transition 100 time steps. The MMD plot is based on a "true sample" of size 2k from the stationary distribution (estimated by 2k HMC transition steps). The numbers are mean and standard deviation over 10 runs. The MMD is computed by the Gaussian kernel with the median pairwise distance as kernel width.

The quality of the transition kernel and the generated data is critical. Since x and x' are supposed to be related, we use an HMC kernel with one leap-frog step. The initial x is effectively forgotten if using too many leap-frog steps. The main point is to show that our method can utilize the intermediate samples from the chain other than the final point. Moreover, to conform with Assumption 2, the potential functions are numerically truncated.

To verify the convergent behavior of our method, Fig. B.1 shows how the ratio network improves as we train the model. It can be seen that the our method quickly concentrates its mass to the region with high potentials.

[1] https://github.com/kamenbliznashki/normalizing_flows
[2] https://github.com/kevin-w-li/deep-kexpfam

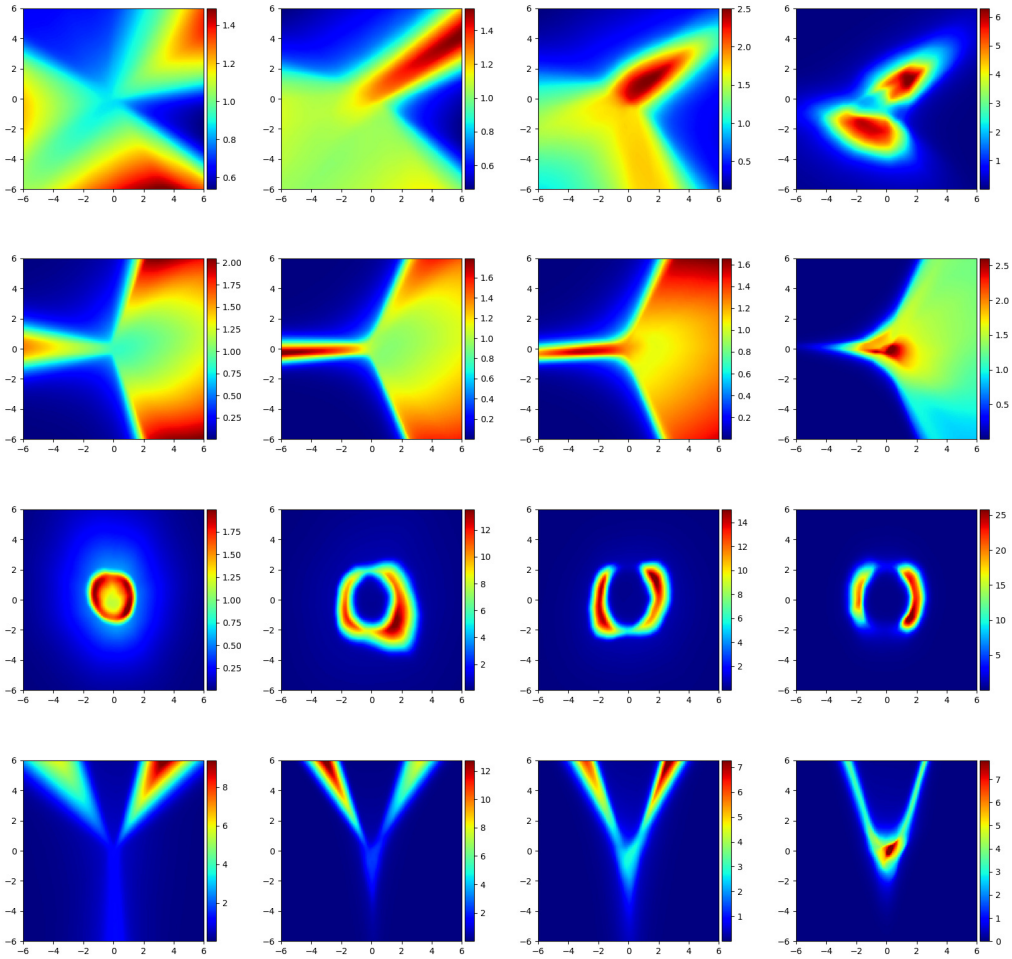


Figure B.1: The VPM estimates after $\{10, 20, 30, 150\}$ iterations on the datasets. As we can see, with the algorithm proceeds, the learned stationary density ratio is getting closer to the ground-truth.

B.4.4 Off-policy Evaluation

Taxi is a 5×5 gridworld in which the taxi agent navigates to pick up and drop off passengers in specific locations. It has a total of 2000 states and 6 actions. Each step incurs a -1 reward unless the agent picks up or drops off a passenger in the correct locations. The behavior policy is set to be the policy after 950 Q-learning iterations and the target policy is the policy after 1000 iterations. In the Taxi experiment, given a transition (s, a, s'), instead of sampling one single action from the target policy $\pi(a'|s')$, we use the whole distribution $\pi(\cdot|s')$ for estimation. We conduct closed-form

update in the power method and the number of steps is $T = 100$.

Continuous experiments. The environments are using the open-source PyBullet engine. The state spaces are in $\mathbb{R}^9, \mathbb{R}^{26}, \mathbb{R}^{28}$ respectively and the action spaces are in $\mathbb{R}^2, \mathbb{R}^6, \mathbb{R}^8$ respectively. the τ model is the same as in the SDE experiment (except for input, which depends on the environment). $T = 200, M = 10, B = 1k$ and the learning rate is 0.0003. The model-based method has a similar neural network structure and is trained for $MT = 2k$ steps with a learning rate of 0.0005. The target policy for the Reacher agent is pretrained using PPO while the HalfCheetah and Ant agents are pretrained using A2C (all with two hidden layers of 64 units each).

The results in the plots are mean and standard deviation from 10 runs.