Learning to Predict and Make Decisions under Distribution Shift

Yifan Wu

September 2021
CMU-ML-21-110

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Zachary Lipton, Chair
Andrej Risteski
Sivaraman Balakrishnan
Alexander Smola (Amazon)

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

This research was supported by the Department of the Interior award D17AP00001 and grants from PricewaterhouseCoopers, the University of California, and UPMC Enterprises.

Keywords: Distribution Shift, Domain Adaptation, Label Shift, PU Learning, Reinforcement Learning, Bandit

Abstract

A common use case of machine learning in real world settings is to learn a model from historical data and then deploy the model on future unseen examples. When the data distribution for the future examples differs from the historical data distribution, machine learning techniques that depend precariously on the i.i.d. assumption tend to fail. So dealing with distribution shift is an important challenge when developing machine learning techniques for practical use. While it is unrealistic to expect a learned model to predict accurately under any form of distribution shift, well chosen research objectives may still lead to effective machine learning algorithms that handle distribution shift properly. For example, when facing distribution shift, we may expect to build machine learning models that (i) make accurate predictions under specific assumptions on how the distribution shift happens; (ii) identify out-of-distribution inputs where the model may not be able to predict well; and/or (iii) act conservatively according to what it can predict well. While recent research has produced practically successful methods in machine learning settings with independent and identically distributed data, progress on settings where dealing with distribution shift is necessary has remained in a comparatively developmental stage.

In this thesis, we study the problem of learning under distribution shift in two scenarios: prediction and decision making. The first part of the thesis addresses the prediction problem, focusing on exploiting specific assumptions such as covariate shift and label shift. We develop theoretical understanding and effective techniques in these scenarios. In the second part of this thesis, we study the problem of offline policy optimization, where the goal is to learn a good policy from a fixed set of data, whose distribution may not be rich enough to inform the optimal policy. We first present an extensive empirical study on behavior regularized offline reinforcement learning algorithms. We then present a theoretical study on whether/why one should follow the pessimistic principle in the offline policy optimization problem.

Acknowledgments

I would like to thank my advisor Zachary Lipton for agreeing to be my advisor and providing all kinds of support during my PhD study. I would like to thank my Master's degree advisors Csaba Szepesvári and András György for getting me prepared for research before I started my PhD. I would like to thank Ofir Nachum, George Tucker, Yanqi Zhou, and Hanxiao Liu for hosting my internships at Google. I would like to thank Chenjun Xiao, Saurabh Garg, and Ruosong Wang for the research collaboration. I would like to thank Andrej Risteski, Sivaraman Balakrishnan, and Alex Smola for being on my thesis committee and providing insightful feedback for my thesis proposal and dissertation. I would like to thank Diane Stidle for taking care of the Machine Learning Department so that I never worry about any administrative process in the PhD program. I would also like to thank everyone I have interacted with during my PhD, who has made my life as a PhD student better in various ways. Last but most importantly, I would like to thank my family for their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Despite success across a variety of challenging tasks, current machine learning techniques depend on massive amounts of training data that is representative of potential test cases. Real world settings, however, usually demand that learned models make predictions or decisions under scenarios that may or may not be well covered by the training data. The problem that the collected dataset is non-representative cannot simply be dealt with by collecting more data from the same source. Moreover, the performance of learned models tend to drop significantly even with a tiny amount of distribution shift between training and test [115, 135], which makes it challenging to reliably deploy machine learning in real world applications. Although one can always increase training coverage by adding more sources of data [32], data augmentation [100, 124], or injecting structural bias into models [45, 60, 145], the combinatorial nature of real world data and tasks makes it unrealistic to collect sufficient data or inject perfect model bias for generalization to any potential input for the learned model. On the other hand, even if it is hard to build a model that is able to predict accurately under all potential distribution shift scenarios, under appropriate assumptions and with appropriate modifications, machine learning may still be applicable in the presence of distribution shift. For example, adapting a model to a specific type of distribution shift might be more approachable than adapting to any potential shifts. Also by knowing where the model can predict well, one can use the model to make conservative predictions or decisions, and to guide future active data collection to covered shifted distributions. Therefore, in addition to improving models' generalization performance in general, methods that explicitly deal with the presence of distribution shift are also desirable for machine learning to be used in practice.

1.2 Outline and Contributions

This thesis focuses on understanding and developing machine learning methods that explicitly take distribution shift into consideration. More specifically, we investigate two research directions where addressing distribution shift plays an important role: *(i) unsupervised domain adaptation*; and *(ii) offline policy optimization*. In the rest of this section, we give a brief introduction to these two directions as well as summarize the contributions of thesis in these directions.

1.2.1 Part I Unsupervised Domain Adaptation

Domain adaptation addresses the common situation in which the *target* distribution $p_T(x, y)$ generating our test data differs from the *source* distribution $p_S(x, y)$ generating our training data. *unsupervised domain adaptation* aims at learning a classifier from labeled *source domain* data, i.e., i.i.d. samples from $p_S(x, y)$ and unlabeled *target domain* data, i.e., i.i.d. samples from $p_T(x)$, to maximize performance on the target distribution $p_T(x, y)$, e.g., by attempting to predict the conditional label distribution $p_T(y|x)$ or its mean/mode, depending on the actual prediction task.

Without further assumptions on the relationship between p_S and p_T, guarantees of target-domain accuracy are impossible [14]. However, well-chosen assumptions can make possible algorithms with non-vacuous performance guarantees. For example, under the *covariate shift* assumption [59, 123], although the input marginals can vary between source and target ($p_S(x) \neq p_T(x)$), the conditional distribution of the labels (given features) exhibits invariance across domains ($p_S(y|x) = p_T(y|x)$). Some consider the reverse setting, *label shift* [94, 116, 159], where although the label distribution shifts ($p_S(y) \neq p_T(y)$), the class-conditional input distribution is invariant ($p_S(x|y) = p_T(x|y)$). In the first part of this thesis, we investigate both covariate shift and label shift assumptions. We develop better understanding of the limitations and properties of existing methods in these scenarios, as well as propose new methods with both theoretical guarantees and improved performance.

Chapter 2 Domain Adaptation with Asymmetrically-Relaxed Distribution Alignment Traditional approaches to the *covariate shift* problem require the source distribution's support to cover the target support, estimating adapted classifiers via importance-weighted risk minimization [54, 62, 123, 156]. In the covariate shift setting without the contained support assumption, recently-proposed domain-adversarial approaches consist of aligning source and target encodings, an approach often motivated as minimizing two (of three) terms in a theoretical bound on target error [46]. Unfortunately, this minimization can cause arbitrary increases in the third term, a problem guaranteed to arise under shifting label distributions. We propose *asymmetrically-relaxed distribution alignment*, a new approach that overcomes some limitations of standard domain-adversarial algorithms. Moreover, we characterize precise assumptions under which our algorithm is theoretically principled and demonstrate empirical benefits on both synthetic and real datasets. This chapter is largely based on previous work published in [149]. The thesis author was the primary investigator and author of this paper.

Chapter 3 A Unified View of Label Shift Estimation In the *label shift* setting, the label distribution might change (i.e, $p_S(y) \neq p_T(y)$) across domains but the class-conditional distributions $p(x|y)$ remains the same. We first consider the easier scenario where the source support covers the target support (i.e., $p_T(y)/p_S(y) \leq \beta$). With the contained support assumption, learning a target domain classifier can also be done by importance risk minimization [94], where the first task is to estimate the label marginal $p_T(y)$, in order to compute the importance ratios. There are two dominant approaches for estimating the label marginal. BBSE [94], a moment-matching approach based on confusion matrices, is provably consistent and provides interpretable error bounds. However, a maximum likelihood estimation approach, which we call MLLS, dominates empirically [4]. In this paper, we present a unified view of the two methods and the first

theoretical characterization of MLLS. Our contributions include (i) consistency conditions for MLLS, which include calibration of the classifier and a confusion matrix invertibility condition that BBSE also requires; (ii) a unified framework, casting BBSE as roughly equivalent to MLLS for a particular choice of calibration method; and (iii) a decomposition of MLLS's finite-sample error into terms reflecting miscalibration and estimation error. Our analysis attributes BBSE's statistical inefficiency to a loss of information due to coarse calibration. Experiments on synthetic data, MNIST, and CIFAR10 support our findings. This chapter is largely based on previous work published in [47]. The thesis author and Saurabh Garg jointly developed the theoretical part of this paper. Saurabh Garg is the primary contributor of the experiments in this paper.

Chapter 4 Learning from Positive and Unlabeled Data We then consider the more challenging label shift setting where the contained support assumption is violated (i.e., there exists some y such that $p_T(y) > 0$ while $p_S(y) = 0$). The simplest version of this scenario is that the source distribution only contains a single *positive* class while the target distribution contains a mixture of the *positive* class and an additional *negative* class, which is also known as Positive and Unlabeled learning (PU learning). Given only positive examples and unlabeled examples (from both positive and negative classes), we might hope nevertheless to estimate an accurate positive-versus-negative classifier. Similarly to the contained support setting, this task is broken down into two subtasks: (i) *Label Shift Estimation* (also known as *Mixture Proportion Estimation* (MPE) in the literature)—determining the fraction of positive examples in the unlabeled data; and (ii) *Classification*— learning the desired positive-versus-negative classifier. Unfortunately, classical methods for both problems break down in settings high-dimensional and complex inputs. Meanwhile, recently proposed heuristics lack theoretical coherence and depend precariously on hyperparameter tuning. We propose two simple techniques: *Best Bin Estimation* (BBE) (for MPE); and *Conditional Value Under Optimism* (CVuO), a simple objective for classification using the MPE output. Both methods outperform previous approaches empirically, and for BBE, we establish theoretical guarantees demonstrating that its good performance only requires access to a blackbox classifier that approximately separates out a small subset of positive examples. Our final algorithm (TED)^n, alternates between the two procedures, significantly improving both our mixture proportion estimator and classifier. This chapter is largely based on [48] which is under submission. The thesis author and Saurabh Garg jointly developed the theoretical and algorithmic part of this paper. Saurabh Garg is the primary contributor of the experiments in this paper.

1.2.2 Part II Offline Policy Optimization

Decision making problems in machine learning are often formulated as learning a policy π that takes actions to maximize future returns. For example, in *multi-armed bandits* the goal is to identify the action that achieves the best return, while in *contextual bandits* or *reinforcement learning* the desired policy takes actions conditioned on the observed state, maximizing the one-step or long-term returns respectively. [1] *Offline policy optimization* (also known as *batch policy opti-*

[1] In this thesis, since we focus on the offline setting, we only consider optimizing the performance of the policy in the end rather than balancing the exploration-exploitation trade-off during the learning process.

mization) aims at learning a good policy from a fixed dataset $\mathcal{D}$ of logged experience, as opposed to *online policy optimization* where continual interaction with the environment is allowed for active data collection.

A major challenge in offline policy optimization is to deal with distribution shift [89]. Although here we may not have an explicit definition of a *target distribution*, the target distribution in offline policy optimization can be regarded as an implicit distribution that is sufficient to inform the learning objective [52, 79, 84], which is the optimal policy. In this scenario, *distribution shift* refers to the situation where the *source distribution* (training data distribution) is insufficient to inform the optimal policy (i.e., different from any target distribution in hindsight). When learning the optimal policy becomes unrealistic because of the distribution shift, one may expect to learn a good (but maybe suboptimal) policy from the limited training distribution. To achieve this, one widely adopt principle in existing algorithms is *pessimism in the face of uncertainty* (referred to as the *pessimistic principle*, as opposed to the *optimistic principle* in online exploration), which aims at finding the best policy among those whose return can be well estimated from the data. In the second part of this thesis, we attempt to understand the implementation, properties and limitations of the pessimistic principle in the offline policy optimization problem.

Arguably, the simplest and the hardest problems of decision making in machine learning correspond to multi-armed bandits and reinforcement learning respectively. This thesis looks at both end of the spectrum: We use the multi-armed bandit setting to understand the theoretical limits and properties of the offline policy optimization problem and use the reinforcement learning setting to explore the empirical effectiveness of certain algorithms. We first present an extensive empirical study on offline reinforcement learning, understanding the effectiveness, limitations, and the role of each building components in a family of offline RL algorithms based on the pessimistic principle. We then ask the question of whether following the pessimistic principle is optimal for offline policy optimization. We show that a simple theoretical argument for its optimality is not possible even in the multi-armed bandit setting. We also propose alternative arguments to justify the use of pessimistic algorithms in practice.

Chapter 5 An Empirical Study on Behavior regularized offline RL In the offline reinforcement learning settings, standard RL algorithms have been shown to diverge or otherwise yield poor performance. Accordingly, recent work has suggested a number of remedies to these issues. In this chapter, we introduce a general framework, *behavior regularized actor critic* (BRAC), to empirically evaluate recently proposed methods as well as a number of simple baselines across a variety of offline continuous control tasks. Surprisingly, we find that many of the technical complexities introduced in recent methods are unnecessary to achieve strong performance. Additional ablations provide insights into which design choices – target value construction, type of divergence regularizer, etc. – matter most in the offline RL setting. This chapter is largely based on previous work published in [148]. The thesis author was the primary investigator and author of this paper.

Chapter 6 On the Optimality of Offline Policy Optimization Algorithms Although interest in the offline policy optimization problem has grown significantly in recent years, its theoretical foundations remain under-developed. To advance the understanding of this problem, in this chap-

ter we provide results that characterize the limits and possibilities of offline policy optimization in the finite-armed stochastic bandit setting. First, we introduce a class of *confidence-adjusted index* algorithms that unifies optimistic and pessimistic principles in a common framework, which enables a general analysis. For this family, we show that *any* confidence-adjusted index algorithm is minimax optimal, whether it be optimistic, pessimistic or neutral. Our analysis reveals that instance-dependent optimality, commonly used to establish optimality of *on-line* stochastic bandit algorithms, *cannot be achieved by any algorithm* in the offline setting. In particular, for any algorithm that performs optimally in some environment, there exists another environment where the same algorithm suffers arbitrarily larger regret. In addition, we propose alternates arguments which incorporates prior or weights over instances and can be potentially used to support the use of pessimistic algorithms. This chapter is largely based on previous work published in [150]. The thesis author and Chenjun Xiao were the joint-primary investigators and authors of this paper.

Part I

Unsupervised Domain Adaptation

Chapter 2

Domain Adaptation with Asymmetrically-Relaxed Distribution Alignment

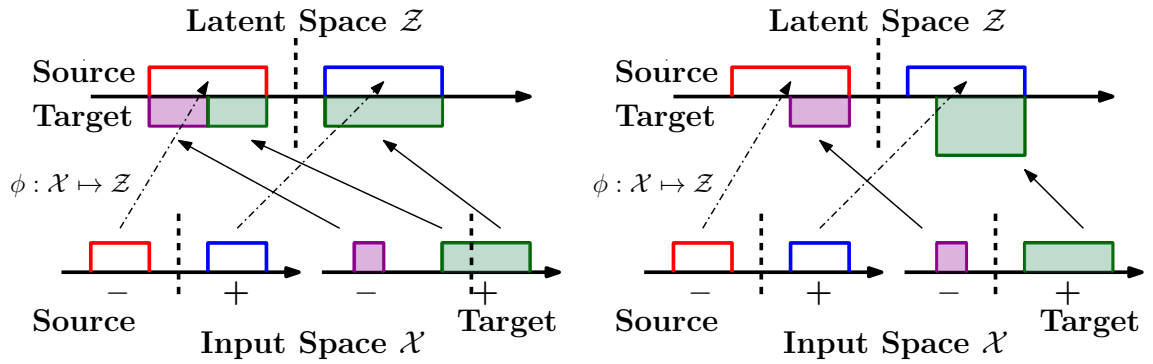
2.1 Overview

Despite breakthroughs in supervised deep learning across a variety of challenging tasks, current techniques depend precariously on the i.i.d. assumption. Unfortunately, real-world settings often demand not just generalization to *unseen examples* but robustness under a variety of shocks to the data distribution. Ideally, our models would leverage unlabeled test data, adapting in real time to produce improved predictions. *Unsupervised domain adaptation* formalizes this problem as learning a classifier from labeled *source domain* data and unlabeled data from a *target domain*, to maximize performance on the target distribution.

Without further assumptions, guarantees of target-domain accuracy are impossible [14]. However, well-chosen assumptions can make possible algorithms with non-vacuous performance guarantees. For example, under the *covariate shift* assumption [59, 123], although the input marginals can vary between source and target ($p_S(x) \neq p_T(x)$), the conditional distribution of the labels (given features) exhibits invariance across domains ($p_S(y|x) = p_T(y|x)$). Some consider the reverse setting *label shift* [94, 116, 159], where although the label distribution shifts ($p_S(y) \neq p_T(y)$), the class-conditional input distribution is invariant ($p_S(x|y) = p_T(x|y)$). Traditional approaches to both problems require the source distribution's support to cover the target support, estimating adapted classifiers via importance-weighted risk minimization [54, 62, 94, 123, 156].

Problematically, assumptions of contained support are violated in practice. Moreover, most theoretical analyses do not guaranteed target accuracy when the source distribution support does not cover that of the target. A notable exception, Ben-David et al. [13] leverages capacity constraints on the hypothesis class to enable generalization to out-of-support samples. However, their results (i) do not hold for high-capacity hypothesis classes, e.g., neural networks; and (ii) do not provide intuitive interpretations on what is sufficient to guarantee a good target domain performance.

A recent sequence of deep learning papers have proposed empirically-supported adversarial



(a) Exact matching (b) Relaxed matching

Figure 2.1: (a) In order to match the latent space distributions exactly, a model must map some elements of positive class in the target domain to some elements of negative class in the source domain. (b) A better mapping is achieved by requiring only that the source covers the target in the latent space.

training schemes aimed at practical problems with non-overlapping supports [46, 139]. Example problems include generalizing from gray-scale images to colored images or product images on white backgrounds to photos of products in natural settings. While importance-weighting solutions are useless here (with non-overlapping support, weights are unbounded), *domain-adversarial networks* [46] and subsequently-proposed variants report favorable empirical results on a variety of image recognition challenges.

The key idea of domain-adversarial networks is to simultaneously minimize the source error and align the two distributions in representation space. The scheme consists of an encoder, a *label classifier*, and a *domain classifier*. During training, the *domain classifier* is optimized to predict each image's domain given its encoding. The *label classifier* is optimized to predict labels from encodings (for source images). The encoder weights are optimized for the twin objectives of accurate label classification (of source data) and *fooling* the domain classifier (for all data).

Although Ganin et al. [46] motivate their idea via theoretical results due to Ben-David et al. [13], the theory is insufficient to justify their method. Put simply, Ben-David et al. [13] bound the test error by a sum of three terms. The domain-adversarial objective minimizes two among these, but this minimization may cause the third term to increase. This is guaranteed to happen when the label distribution shifts between source and target. Consider the case of cat-dog classification with non-overlapping support. Say that the source distribution contains 50% dogs and 50% cats, while the target distribution contains 25% dogs and 75% cats. Successfully aligning these distributions in representation space requires the classifier to predict the same fraction of dogs and cats on source and target. If one achieves 100% accuracy on the source data, then target accuracy will be at most 75% (Figure 2.1(a)).

In this chapter, we propose asymmetrically-relaxed distribution alignment, a relaxed distance for aligning data across domains that can be minimized without requiring latent-space distributions to match exactly. The new distance is minimized whenever the density ratios in representation space from target to source are upper-bounded by a certain constant, such that the target representation support is contained in the source representation's. The relaxed distribution align-

ment need not lead to a poor classifier on the target domain under label distribution mismatch (Figure 2.1(b)). We demonstrate theoretically that the relaxed alignment is sufficient to ensure low error on the target domain under a concrete set of assumptions on the data distributions. Further, we propose several practical ways to achieve the relaxed distribution alignment, translating the new distance into adversarial learning objectives. Empirical results on synthetic and real datasets show that incorporating our relaxed distribution alignment loss into adversarial domain adaptation gives better classification performance on the target domain. We make the following key contributions:

- We propose an asymmetrically relaxed distribution matching objective, overcoming the limitation of standard objectives under label distribution shift.
- We provide theoretical analysis demonstrating that under a clear set of assumptions, the asymmetrically-relaxed distribution alignment can provide target-domain performance guarantees.
- We propose several distances that satisfy the desired properties and are optimizable by adversarial training.
- We empirically show that our asymmetrically-relaxed distribution matching losses improve target performance when there is a label distribution shift in the target domain and perform comparably otherwise.

2.2 Preliminaries

We use subscripts S and T to distinguish between source and target domains, e.g., p_S and p_T, and employ the notation U for statements that are true for any domain $U \in \{S, T\}$. For simplicity, we dispense with some rigorousness in notating probability measures. For example, we use the terms measure and distribution interchangeably and assume that a density function exists when necessary without explicitly stating the base measure and required regularity conditions. We use a single lowercase letter, e.g. p, to denote both the probability measure function and the probability density function: $p(x)$ is a density when the input x is a single point while $p(C)$ is a probability when the input C is a set. We will use $\mathrm{Supp}(p)$ to denote the support of distribution p, i.e., the set of points where the density is positive. Similarly, for a function mapping ϕ, $\phi(x)$ denotes an *output* if x is a point and $\phi(C)$ denotes the *image* if C is a set. The inverse mapping ϕ^{-1} always outputs a set (the inverse image) regardless of whether its input is a point or a set. We will also be less careful about the use of $\sup$ v.s. $\max$, $\inf$ v.s. $\min$ and "everywhere" v.s. "almost everywhere". For two functions f and g we use $f \equiv g$ to denote that $f(x) = g(x)$ for every input x.

Unsupervised domain adaptation For simplicity, we address the binary classification scenario. Let $\mathcal{X}$ be the input space and $f : \mathcal{X} \mapsto \{0, 1\}$ be the (domain-invariant) ground truth labeling function. Let p_S and p_T be the input distributions over $\mathcal{X}$ for source and target domain respectively. Let $\mathcal{Z}$ be a latent space and Φ denote a class of mappings from $\mathcal{X}$ to $\mathcal{Z}$. For a domain U, let $p_U^\phi(\cdot)$ be the induced probability distribution over $\mathcal{Z}$ such that $p_U^\phi(C) = p_U(\phi^{-1}(C))$ for any $C \subset \mathcal{Z}$. Given $z \in \mathcal{Z}$ let $\phi_U(\cdot|z)$ be the conditional distribution induced by p_U and ϕ

such that $\int \mathrm{d}z p_U^\phi(z)\phi_U(x|z) = p_U(x)$ holds for all $x \in \mathcal{X}$. Define $\mathcal{H}$ to be a class of predictors over the latent space $\mathcal{Z}$, i.e., each $h \in \mathcal{H}$ maps from $\mathcal{Z}$ to $\{0, 1\}$. Given a representation mapping $\phi \in \Phi$, classifier $h \in \mathcal{H}$, and input $x \in \mathcal{X}$, our prediction is $h(\phi(x))$. The risk for a single input x can be written as $|h(\phi(x)) - f(x)|$ and the expected risk for a domain U is

$$\begin{aligned}\mathcal{E}_U(\phi, h) &= \int \mathrm{d}x p_U(x) \left|h(\phi(x)) - f(x)\right| \\ &\doteq \int \mathrm{d}z p_U^\phi(z) \left|h(z) - f_U^\phi(z)\right| \\ &\doteq \int \mathrm{d}z p_U^\phi(z) r_U(z; \phi, h) \end{aligned} \tag{2.1}$$

where we define a domain-dependent latent space labeling function $f_U^\phi(z) = \int \mathrm{d}x \phi_U(x|z) f(x)$ and the risk for a classifier h as $r_U(z; \phi, h) = \left|h(z) - f_U^\phi(z)\right| \in [0, 1]$.

We are interested in bounding the classification risk of a (ϕ, h)-pair on the target domain:

$$\begin{aligned}\mathcal{E}_T(\phi, h) &= \int \mathrm{d}z p_T^\phi(z) r_T(z; \phi, h) = \mathcal{E}_S(\phi, h) \\ &+ \int \mathrm{d}z p_T^\phi(z) r_T(z; \phi, h) - \int \mathrm{d}z p_S^\phi(z) r_S(z; \phi, h) \\ &= \mathcal{E}_S(\phi, h) + \int \mathrm{d}z p_T^\phi(z) \left(r_T(z; \phi, h) - r_S(z; \phi, h)\right) \\ &+ \int \mathrm{d}z \left(p_T^\phi(z) - p_S^\phi(z)\right) r_S(z; \phi, h) \,. \end{aligned} \tag{2.2}$$

The second term in (2.2) becomes zero if the latent space labeling function is domain-invariant. To see this, we apply

$$\begin{aligned} r_T(z; \phi, h) - r_S(z; \phi, h) &= \left|h(z) - f_T^\phi(z)\right| - \left|h(z) - f_S^\phi(z)\right| \\ &\leq \left|f_T^\phi(z) - f_S^\phi(z)\right| \,. \end{aligned} \tag{2.3}$$

The third term in (2.2) is zero when p_T^ϕ and p_S^ϕ are the same.

In the *unsupervised domain adaptation* setting, we have access to labeled source data $(x, f(x))$ for $x \sim p_S$ and unlabeled target data $x \sim p_T$, from which we can calculate[1] the first and third term in (2.2). For $x \in \mathrm{Supp}(p_T) \setminus \mathrm{Supp}(p_S)$, we have no information about its true label $f(x)$ and thus $f_T^\phi(z)$ becomes inaccessible when $z = \phi(x)$ for such x. So the second term in (2.2) is not directly controllable.

Domain-adversarial learning *Domain-adversarial* approaches focus on minimizing the first and third term in (2.2) jointly. Informally, these approaches minimize the source domain classification risk and the distance between the two distributions in the latent space:

$$\min_{\phi, h} \; \mathcal{E}_S(\phi, h) + \lambda D(p_S^\phi, p_T^\phi) + \Omega(\phi, h) \,, \tag{2.4}$$

[1] In this work we focus on how domain adaption are able to generalize across distributions with different supports so we will not talk about finite-sample approximations.

where D is a distance metric between distributions and Ω is a regularization term. Standard choices of D such as a domain classifier (Jensen-Shannon (JS) divergence [2]) [46], Wasserstein distance [122] or Maximum Mean Discrepancy [62] have the property that $D(p_S^\phi, p_T^\phi) = 0$ if $p_S^\phi \equiv p_T^\phi$ and $D(p_S^\phi, p_T^\phi) > 0$ otherwise. In the next section, we will show that minimizing (2.4) with such D will lead to undesirable performance and propose an alternative objective to align p_S^ϕ and p_T^ϕ instead of driving them to be identically distributed.

2.3 A Motivating Scenario

To motivate our approach, we formally show how exact distribution matching can lead to undesirable performance. More specifically, we will lower bound $\mathcal{E}_T(\phi, h)$ when both $\mathcal{E}_S(\phi, h)$ and $D(p_S^\phi, p_T^\phi)$ are zero with respect to the shift in the label distribution. Let ρ_S and ρ_T be the proportion of data with positive label, i.e., $\rho_U = \int \mathrm{d}x p_U(x) f(x)$. We formalize the result as follows.

Proposition 2.3.1. If $D(p_S^\phi, p_T^\phi) = 0$ if and only if $p_S^\phi \equiv p_T^\phi$, $\mathcal{E}_S(\phi, h) = D(p_S^\phi, p_T^\phi) = 0$ indicates $\mathcal{E}_T(\phi, h) \geq |\rho_S - \rho_T|$.

The proof follows the intuition of Figure 2.1(a): If $\rho_S < \rho_T$, the best we can do is to map $\rho_T - \rho_S$ proportion of positive samples from the target inputs to regions of latent space corresponding to negative examples from the source domain while maintaining the label consistency for remaining ones. Switching the term positive/negative gives a similar argument for $\rho_T < \rho_S$. Proposition 2.3.1 says that if there is a label distribution mismatch $\rho_T \neq \rho_S$, minimizing the objective (2.4) to zero imposes a positive lower bound on the target error. This is especially problematic in cases where a perfect pair ϕ, h may exist, achieving zero error on both source and target data (Figure 2.1(b)).

Asymmetrically-relaxed distribution alignment It may appear contradictory that minimizing the first and third term of (2.2) to zero guarantees a positive $\mathcal{E}_T(\phi, h)$ and thus a positive second term when there exists a pair of ϕ, h such that $\mathcal{E}_T(\phi, h) = 0$ (all three terms are zero). However, this happens because although $D(p_S^\phi, p_T^\phi) = 0$ is a sufficient condition for the third term of (2.2) to be zero, *it is not a necessary condition*. We now examine the third term of (2.2):

$$\begin{aligned} &\int \mathrm{d}z \left(p_T^\phi(z) - p_S^\phi(z) \right) r_S(z; \phi, h) \\ &\leq \left(\sup_{z \in \mathcal{Z}} \frac{p_T^\phi(z)}{p_S^\phi(z)} - 1 \right) \mathcal{E}_S(\phi, h). \end{aligned} \tag{2.5}$$

This expression (2.5) shows that if the source error $\mathcal{E}_S(\phi, h)$ is zero then it is sufficient to say the third term of (2.2) is zero when the density ratio $p_T^\phi(z)/p_S^\phi(z)$ is upper bounded by some constant for all z. Note that it is impossible to bound $p_T^\phi(z)/p_S^\phi(z)$ by a constant that is smaller than 1 so we write this condition as $\sup_{z \in \mathcal{Z}} p_T^\phi(z)/p_S^\phi(z) \leq 1 + \beta$ for some $\beta \geq 0$. Note that this is a relaxed condition compared with $p_T^\phi(z) \equiv p_S^\phi(z)$, which is a special case with $\beta = 0$.

[2] Per [109], there is a slight difference between JS-divergence and the original GAN objective [51]. We will use the term JS-divergence for the GAN objective.

Relaxing the exact matching condition to the more forgiving bounded density ratio condition makes it possible to obtain a perfect target domain classifier in many cases where the stricter condition does not, by requiring only that the (latent space) target domain support is contained in the source domain support, as shown in Figure 2.1(b). The following proposition states that our relaxed matching condition does not suffer from the previously-described problems concerning shifting label distributions (Proposition 2.3.1), and provides intuition regarding just how large β may need to be to admit a perfect target domain classifier.

Proposition 2.3.2. For every ρ_S, ρ_T, there exists a construction of (p_S, p_T, ϕ, h) such that $\mathcal{E}_S(\phi, h) = 0$, $\mathcal{E}_T(\phi, h) = 0$ and $\sup_{z \in \mathcal{Z}} p_T^\phi(z)/p_S^\phi(z) \leq \max\left\{\frac{\rho_T}{\rho_S}, \frac{1-\rho_T}{1-\rho_S}\right\}$.

Given this motivation, we propose relaxing from exact distribution matching to bounding the density ratio in the domain-adversarial learning objective (2.4). We call this *asymmetrically-relaxed distribution alignment* since we aim at upper bounding p_T^ϕ/p_S^ϕ (but not p_S^ϕ/p_T^ϕ). We now introduce a class of distances between distributions that can be minimized to achieve the relaxed alignment:

Definition 2.3.3 (β-admissible distances)**.** Given a family of distributions defined on the same space $\mathcal{Z}$, a distance metric D_β between distributions is called β-*admissible* if $D_\beta(p, q) = 0$ when $\sup_{z \in \mathcal{Z}} p(z)/q(z) \leq 1 + \beta$ and $D_\beta(p, q) > 0$ otherwise.

Our proposed approach is to *replace the typical distribution distance D in the domain-adversarial objective* (2.4) *with a β-admissible distance D_β* so that minimizing the new objective does not necessarily lead to a failure under label distribution shift. However, it is still premature to claim the justification of our approach due to the following issues: (i) We may not be able get a perfect source domain classifier with $\mathcal{E}_S(\phi, h) = 0$. This also indicates a trade-off in selecting β as (a) higher β will increase the upper bound ($\beta\mathcal{E}_S(\phi, h)$ according to (2.5)) on the third term in (2.2) (b) lower β will make a good target classifier impossible under label distribution shift. (ii) Minimizing $D_\beta(p_T^\phi, p_S^\phi)$ as part of an objective does not necessarily mean that we will obtain a solution with $D_\beta(p_T^\phi, p_S^\phi) = 0$. There may still be some proportion of samples from the target domain lying outside the support of source domain in the latent space $\mathcal{Z}$. In this case, the density ratio p_T^ϕ/p_S^ϕ is unbounded and (2.5) becomes vacuous. (iii) Even when we are able optimize the objective perfectly, i.e., $\mathcal{E}_S(\phi, h) = D_\beta(p_S^\phi, p_T^\phi) = 0$, with a proper choice of β such that there exists ϕ, h such that $\mathcal{E}_T(\phi, h) = 0$ holds simultaneously (e.g. Figure 2.1(b), Proposition 2.3.2), it is still not guaranteed that such ϕ, h is learned (e.g. Figure 2.2(a)), as the second term of (2.2) is unbounded and changes with ϕ. Put simply, the problem is that although there may exist alignments perfect for prediction, there also exist other alignments that satisfy the objective but predict poorly (on target data). To our knowledge this problem effects all domain-adversarial methods proposed in the literature, and how to theoretically guarantee that the desired alignment is learned remains an open question.

Next, we theoretically study the target classification error under asymmetrically-relaxed distribution alignment. Our analysis resolves the above issues by (i) working with imperfect source domain classifier and relaxed distribution alignment; and (ii) providing concrete assumptions under which a good target domain classifier can be learned.

2.4 Bounding the Target Domain Error

In a manner similar to (2.2), Ben-David et al. [12, 13] bound the target domain error by a sum of three terms: (i) the source domain error (ii) an $\mathcal{H}$-divergence between p_S^ϕ and p_T^ϕ (iii) the best possible classification error that can be achieved on the combination of p_S^ϕ and p_T^ϕ. We motivate our analysis by explaining why their results are insufficient to give a meaningful bound for domain-adversarial learning approaches. From a theoretical upper bound, we may desire to make claims in the following pattern:

Let $\mathcal{M}_\mathcal{A}$ be a set of models that satisfy a set of properties $\mathcal{A}$ (e.g. with low training error), and $\mathcal{B}$ be a set of assumptions on the data distributions (p_S, p_T, f). For any given model $M \in \mathcal{M}_\mathcal{A}$, its performance can be bounded by a certain quantity, i.e. $\mathcal{E}_T(M) \leq \epsilon_{\mathcal{A},\mathcal{B}}$.

Ideally, $\mathcal{A}$ should be *observable* on available data information (i.e. without knowing target labels), and assumptions $\mathcal{B}$ should be *model-independent* (independent of which model $M = (\phi, h)$ is learned among $\mathcal{M}_\mathcal{A}$). In the results of Ben-David et al. [12, 13], terms (i) and (ii) are observable so $\mathcal{A}$ can be set as achieving low quantities on these two terms. Since term (iii) is unobservable we may want to make assumptions on it. This term, however, is model-dependent when ϕ is learned jointly. To make a *model-independent* assumption on term (iii), we need to take the supremum over all $(\phi, h) \in \mathcal{M}_\mathcal{A}$, i.e., all possible models that achieve low values on (i) and (ii). This supremum can be vacuous without further assumptions as a cross-label mapping may also achieve low source error and distribution alignment (e.g. Figure 2.2(a) v.s. Figure 2.1(b)). Moreover, when $\mathcal{H}$ contains all possible binary classifiers, the $\mathcal{H}$-divergence is minimized only if the two distributions are the same, thus suffering the same problem as Proposition 2.3.1 and is therefore not suitable for motivating a learning objective.

To overcome these limitations, we propose a new theoretical bound on the target domain error which (a) treats the difference between p_S^ϕ and p_T^ϕ asymmetrically and (b) bounds the label consistency (second term in 2.2) by exploiting the Lipschitz-ness of ϕ as well as the separation and connectedness of data distributions. Our result can be interpreted as a combination of *observable model properties* and unobservable *model-independent assumptions* while being non-vacuous: it is able to guarantee correct classification for (some fraction of) data points from the target domain even where the source domain has zero density.

2.4.1 A general bound

We introduce our result with the following construction:

Construction 2.4.1. The following statements hold simultaneously:

1. (*Lipschitzness of representation mapping.*) ϕ is L-Lipschitz: $d_\mathcal{Z}(\phi(x_1), \phi(x_2)) \leq L d_\mathcal{X}(x_1, x_2)$ for any $x_1, x_2 \in \mathcal{X}$.
2. (*Imperfect asymmetrically-relaxed distribution alignment.*) For some $\beta \geq 0$, there exist a set $B \subset \mathcal{Z}$ such that $\frac{p_T^\phi(z)}{p_S^\phi(z)} \leq 1 + \beta$ holds for all $z \in B$ and $p_T^\phi(B) \geq 1 - \delta_1$.
3. (*Separation of source domain in the latent space.*) There exist two sets $C_0, C_1 \subset \mathcal{X}$ that satisfy:
   (a) $C_0 \cap C_1 = \emptyset$
   (b) $p_S(C_0 \cup C_1) \geq 1 - \delta_2$.

(c) For $i \in \{0, 1\}$, $f(x) = i$ for all $x \in C_i$.
(d) $\inf_{z_0 \in \phi(C_0), z_1 \in \phi(C_1)} d_{\mathcal{Z}}(z_0, z_1) \geq \Delta > 0$.

Note that this construction does not require any information about target domain labels so the statements [1-3] can be viewed as *observable properties* of ϕ. We now introduce our *model-independent* assumption:

Assumption 2.4.2. (*Connectedness from target domain to source domain.*) Given constants $(L, \beta, \Delta, \delta_1, \delta_2, \delta_3)$, assume that, for any $B_S, B_T \subset \mathcal{X}$ with $p_S(B_S) \geq 1 - \delta_2$ and $p_T(B_T) \geq 1 - \delta_1 - (1 + \beta)\delta_2$, there exists $C_T \subset B_T$ that satisfies the following conditions:

1. For any $x \in C_T$, there exists $x' \in C_T \cap B_S$ such that one can find a sequence of points $x_0, x_1, ..., x_m \in C_T$ with $x_0 = x$, $x_m = x'$, $f(x) = f(x')$ and $d_{\mathcal{X}}(x_{i-1}, x_i) < \frac{\Delta}{L}$ for all $i = 1, ..., m$.
2. $p_T(C_T) \geq 1 - \delta_3$.

We are ready to present our main result:

Theorem 2.4.3. Given a L-Lipschitz mapping $\phi \in \Phi$ and a binary classifier $h \in \mathcal{H}$, if ϕ satisfies the properties in Construction 2.4.1 with constants $(L, \beta, \Delta, \delta_1, \delta_2)$, and Assumption 2.4.2 holds with the same set of constants plus δ_3, then the target domain error can be bounded as

$$\mathcal{E}_T(\phi, h) \leq (1 + \beta)\mathcal{E}_S(\phi, h) + 3\delta_1 + 2(1 + \beta)\delta_2 + \delta_3 \,.$$

Notice that it is always possible to make Construction 2.4.1 by adjusting the constants L, β, Δ, δ_1, δ_2. Given these constants, Assumption 2.4.2 can always be satisfied by adjusting δ_3. So Theorem 2.4.3 is a general bound.

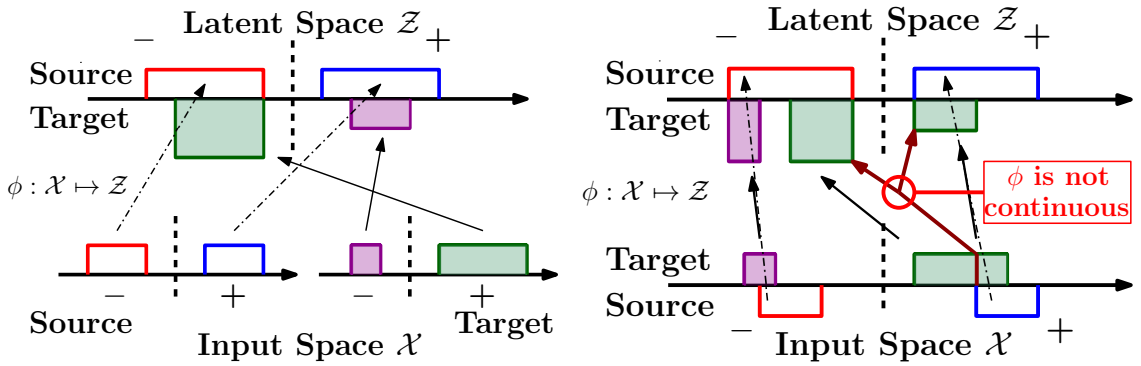
The key challenge in bounding $\mathcal{E}_T(\phi, h)$ is to bound the second term in (2.2) by identifying sufficient conditions that prevent cross-label mapping (e.g. Figure 2.2(a)). To resolve this challenge, we exploit the fact that if there exist a path from a target domain sample to a source domain sample in the input space $\mathcal{X}$ and all samples along the path are mapped into two separate regions in the latent space (due to distribution alignment), then these two connected samples cannot be mapped to different regions, as shown in Figure 2.2(b).

2.4.2 Example of a perfect target domain classifier

To interpret our result, we construct a simple situation where $\mathcal{E}_T(\phi, h) = 0$ is guaranteed when the domain adversarial objective with relaxed distribution alignment is minimized to zero, exploiting pure data-dependent assumptions:

Assumption 2.4.4. Assume the target support consists of disjoint clusters $\text{Supp}(p_T) = S_{T,0,1} \cup ...\cup S_{T,0,m_0} \cup S_{T,1,1} \cup ... \cup S_{T,1,m_1}$, where any cluster $S_{T,i,j}$ is connected and its labels are consistent: $f(x) = i$ for all $x \in S_{T,i,j}$. Moreover, each of these cluster overlaps with source distribution. That is, for any $i \in \{0, 1\}$ and $j \in \{1, ..., m_i\}$, $S_{T,i,j} \cap \text{Supp}(p_S) \neq \emptyset$.

Corollary 2.4.5. If Assumption 2.4.4 holds and there exists a continuous mapping ϕ such that (i) $\sup_{z \in \mathcal{Z}} p_T^{\phi}(z)/p_S^{\phi}(z) \leq 1 + \beta$ for some $\beta \geq 0$; (ii) for any pair $x_0, x_1 \in \text{Supp}(p_S)$ such that $f(x_0) = 0$ and $f(x_1) = 1$, we have $d_{\mathcal{Z}}(\phi(x_0), \phi(x_1)) \geq \Delta > 0$, then $\mathcal{E}_S(\phi, h) = 0$ indicates $\mathcal{E}_T(\phi, h) = 0$.



(a) Failure case (b) Failure impossible

Figure 2.2: (a) Label consistency is broken even if ϕ satisfies the relaxed distribution aligning requirement. (b) The main idea of our analysis: A continuous mapping cannot project a connected region into two regions separated by a margin. So label consistency is preserved for a region that is connected to the source domain.

Proof follows directly by observing that a construction of $\delta_1 = \delta_2 = \delta_3 = 0$ exists in Theorem 2.4.3. A simple example that satisfies Assumption 2.4.4 is Figure 2.2(b). For a real world example, consider the cat-dog classification problem. Say that source domain contains small-to-medium cats and dogs while target domain contains medium-to-large cats and dogs. The target domain consists of clusters (e.g. cats and dogs, or multiple sub-categories) and each of them overlaps with the source domain (the medium ones).

2.5 Asymmetrically-relaxed distances

So far, we have motivated the use of asymmetrically-relaxed distribution alignment which aims at bounding p_T^ϕ / p_S^ϕ by a constant instead of driving towards $p_S^\phi \equiv p_T^\phi$. More specifically, we propose to use a *β-admissible* (Definition 2.3.3) distance D_β in objective (2.4) to align the source and target encodings rather than the standard distances corresponding an adversarial domain classifier. In this section, we derive several *β-admissible* distance metrics that can be practically minimized with adversarial training. More specifically, we propose three types of distances (i) f-divergences; (ii) modified Wasserstein distance; (iii) reweighting distances; and demonstrate how to optimize them by adversarial training.

2.5.1 f-divergence

Given a convex and continuous function f which satisfies $f(1) = 0$, the f-divergence between two distributions p and q can be written as $D_f(p, q) = \int \mathrm{d}z p(z) f\left(\frac{q(z)}{p(z)}\right)$. According to Jensen's inequality $D_f(p, q) \geq f\left(\int \mathrm{d}z p(z) \frac{q(z)}{p(z)}\right) = 0$. Standard choices of f (see a list in Nowozin et al. [109]) are strictly convex thus $D_f(p, q) = 0$ if and only if $p \equiv q$ when f is strictly convex. To derive a β-adimissible variation for each standard choice of f, we linearize $f(u)$ where $u \geq \frac{1}{1+\beta}$.

If and only if $\frac{p(z)}{q(z)} \leq 1+\beta$ for all z, f becomes a linear function with respect to all $q(z)/p(z)$ and thus Jensen's inequality holds with equality.

Given a convex, continuous function $f : \mathbb{R}^+ \mapsto \mathbb{R}$ with $f(1) = 0$ and some $\beta \geq 0$, we introduce the partially linearized $\bar{f}_\beta$ as follows

$$\bar{f}_\beta(u) = \begin{cases} f(u) + C_{f,\beta} & \text{if } u \leq \frac{1}{1+\beta}\,, \\ f'(\frac{1}{1+\beta})u - f'(\frac{1}{1+\beta}) & \text{if } u > \frac{1}{1+\beta}\,. \end{cases}$$

where $C_{f,\beta} = -f(\frac{1}{1+\beta}) + f'(\frac{1}{1+\beta})\frac{1}{1+\beta} - f'(\frac{1}{1+\beta})$.

It can be shown that $\bar{f}_\beta$ is continuous, convex and $\bar{f}_\beta(1) = 0$. As we already explained, $D_{\bar{f}_\beta}(p,q) = 0$ if and only if $p(z)/q(z) \leq 1+\beta$ for all z. Hence is $D_{\bar{f}_\beta}$ is β-admissible.

Adversarial training According to Nowozin et al. [109], adversarial training [51] can be viewed as minimizing the dual form of f-divergences

$$D_f(p,q) = \sup_{T:\mathcal{Z}\mapsto \mathrm{dom}(f^*)} \mathbb{E}_{z\sim q}\left[T(z)\right] - \mathbb{E}_{z\sim p}\left[f^*(T(z))\right]$$

where f^* is the Fenchel Dual of f with $f^*(t) = \sup_{u\in\mathrm{dom}(f)}\{ut - f(u)\}$. Applying the same derivation for $\bar{f}_\beta$ we get[3]

$$D_{\bar{f}_\beta}(p,q) = \sup_{T:\mathcal{Z}\mapsto \mathrm{dom}(\bar{f}_\beta^*)} \mathbb{E}_{z\sim q}\left[T(z)\right] - \mathbb{E}_{z\sim p}\left[f^*(T(z))\right] \tag{2.6}$$

where $\mathrm{dom}(\bar{f}_\beta^*) = \mathrm{dom}(f^*) \cap \big(-\infty, f'(\frac{1}{1+\beta})\big]$.

Plugging in the corresponding f for JS-divergence gives

$$D_{\bar{f}_\beta}(p,q) = \\ \sup_{g:\mathcal{Z}\mapsto(0,1]} \mathbb{E}_{z\sim q}\left[\log\frac{g(z)}{2+\beta}\right] + \mathbb{E}_{z\sim p}\left[\log\left(1 - \frac{g(z)}{2+\beta}\right)\right] \tag{2.7}$$

where $g(z)$ can be parameterized by a neural network with sigmoid output as typically used in adversarial training.

2.5.2 Wasserstein distance

The idea behind modifying the Wasserstein distance is to model the optimal transport from p to the region where distributions have $1+\beta$ maximal density ratio with respect to q. We define the relaxed Wassertein distance as

$$W_\beta(p,q) = \inf_{\gamma\in\prod_\beta(p,q)} \mathbb{E}_{(z_1,z_2)\sim\gamma}\left[\|z_1 - z_2\|\right]\,,$$

where $\prod_\beta(p,q)$ is defined as the set of joint distributions γ over $\mathcal{Z}\times\mathcal{Z}$ such that $\forall z_1 \int \mathrm{d}z\gamma(z_1,z) = p(z_1)$ and $\forall z_2 \int \mathrm{d}z\gamma(z,z_2) \leq (1+\beta)q(z_2)$. W_β is β-admissible since no transportation is needed if p already lies in the qualified region with respect to q.

[3]We are omitting some additive constant term.

Adversarial training Following the derivation for the original Wasserstein distance, the dual form becomes

$$W_\beta(p,q) = \sup_g \mathbb{E}_{z\sim p}\left[g(z)\right] - (1+\beta)\mathbb{E}_{z\sim q}\left[g(z)\right] \tag{2.8}$$
$$\text{s.t. } \forall z \in \mathcal{Z}\,, g(z) \geq 0\,,$$
$$\forall z_1, z_2 \in \mathcal{Z}\,, g(z_1) - g(z_2) \leq \|z_1 - z_2\|\ ,$$

Optimization with adversarial training can be done by parameterizing g as a non-negative function (e.g. with soft-plus output $\log(1+e^x)$ or RELU output $\max(0,x)$) and following Arjovsky et al. [5], Gulrajani et al. [55] to enforce its Lipschitz continuity approximately.

2.5.3 Reweighting distance

Given any distance metric D, a generic way to make it β-admissible is to allow reweighting for one of the distances within a β-dependent range. The relaxed distance is then defined as the minimum achievable distance by such reweighting.

Given a distribution q over $\mathcal{Z}$ and a reweighting function $w : \mathcal{Z} \mapsto [0,\infty)$. The reweighted distribution q_w is defined as $q_w(z) = \frac{q(z)w(z)}{\int \mathrm{d}zq(z)w(z)}$. Define $\mathcal{W}_{\beta,q}$ to be a set of β-*qualified* reweighting with respect to q:

$$\mathcal{W}_{\beta,q} = \left\{ w : \mathcal{Z} \mapsto [0,1], \int \mathrm{d}zq(z)w(z) = \frac{1}{1+\beta} \right\}\ .$$

Then the relaxed distance can be defined as

$$D_\beta(p,q) = \min_{w\in\mathcal{W}_{\beta,q}} D(p,q_w)\,. \tag{2.9}$$

Such D_β is β-admissible since the set $\{q_w : w \in \mathcal{W}_{\beta,q}\}$ is exactly the set of p such that $\sup_{z\in\mathcal{Z}} p(z)/q(z) \leq 1+\beta$.

Adversarial training We propose an *implicit-reweighting-by-sorting* approach to optimize D_β without parameterizing the function w when D can be optimized by adversarial training. Adversarially trainable D shares a general form as

$$D(p,q) = \sup_{g\in\mathcal{G}} \mathbb{E}_{z\sim p}\left[f_1(g(z))\right] - \mathbb{E}_{z\sim q}\left[f_2(g(z))\right]\ ,$$

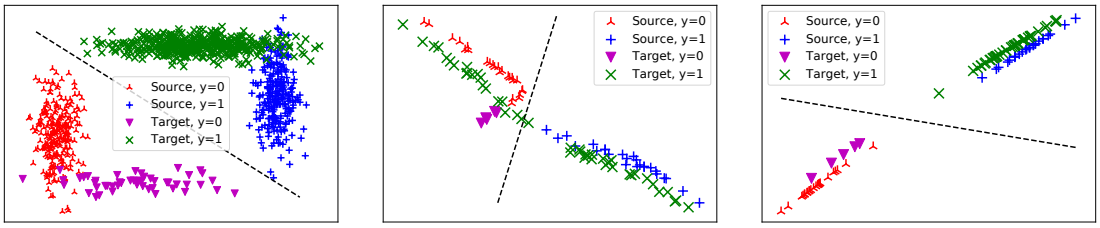
where f_1 and f_2 are monotonically increasing functions. According to (2.9), the relaxed distance can be written as

$$D_\beta(p,q) = \min_w \sup_{g\in\mathcal{G}} \mathbb{E}_{z\sim p}\left[f_1(g(z))\right] - \mathbb{E}_{z\sim q_w}\left[f_2(g(z))\right]\ ,$$
$$\text{s.t. } w : \mathcal{Z} \mapsto [0,1]\,, \int \mathrm{d}zq(z)w(z) = \frac{1}{1+\beta}\,. \tag{2.10}$$

One step of alternating minimization on D_β, could consist of fixing p, q, g and optimizing w. Then the problem becomes

$$\max_{w\in\mathcal{W}_{\beta,q}} \int \mathrm{d}zq(z)w(z)f_2(g(z))\,. \tag{2.11}$$

Observe that the optimal solution to (2.11) is to assign $w(z) = 1$ for the $\frac{1}{1+\beta}$ fraction of z from distribution q, where $f_2(g(z))$ take the largest values. Based on this observation, we propose to do the following sub-steps when optimizing (2.11) as an alternating minimization step: (i) Sample a minibatch of $z \sim q$; (ii) Sort these z in descending order according to $f_2(g(z))$; (iii) Assign $w(z) = 1$ to the first $\frac{1}{1+\beta}$ fraction of the list. Note that this optimization procedure is not justified in principle with mini-batch adversarial training but we found it to work well in our experiments.



(a) raw (synthetic) data (b) latent representations (DANN) (c) latent representations (ours)

Figure 2.3: Domain-adversarial training under label distribution shift on a synthetic dataset.

2.6 Experiments

To evaluate our approach, we implement Domain Adversarial Neural Networks (DANN), [46] replacing the JS-divergence (domain classifier) with our proposed β-admissible distances (Section 2.5). Our experiments address the following questions: (i) Does DANN suffer the limitation as anticipated (Section 2.3) when faced with label distribution shift? (ii) If so, do our β-admissible distances overcome these limitations? (iii) Absent shifting label distributions, is our approach comparable to DANN?

We implement adversarial training with different β-admissible distances (Section 2.5) and compare their performance with vanilla DANN. We name different implementations as follows. (a) SOURCE: source-only training. (b) DANN: JS-divergence (original DANN). (c) WDANN: original Wasserstein distance. (d) FDANN-β: β-admissible f-divergence, JS-version (2.7). (e) SDANN-β: reweighting JS-divergence (2.10), optimized by our proposed *implicit-reweighting-by-sorting*. (f) WDANN1-β: β-admissible Wasserstein distance (2.8) with soft-plus on critic output. (g) WDANN2-β: β-admissible Wasserstein distance (2.8) with RELU on critic output. (h) SWDANN-β: reweighting Wasserstein distance (2.10), optimized by *implicit-reweighting-by-sorting*. Adversarial training on Wasserstein distances follows Gulrajani et al. [55] but uses one-sided gradient-penalty. We always perform adversarial training with alternating minimization.

Synthetic datasets We create a mixture-of-Gaussians binary classification dataset where each domain contains two Gaussian distributions, one per label. For each label, the distributions in source and target domain have a small overlap, validating the assumptions in our analysis. We create a label distribution shift with balanced source data (50% 0's v.s. 50% 1's) and imbalanced

target data (10% 0's v.s. 90% 1's) as shown in Figure 2.3(a). Table 2.1 shows the target domain accuracy for different approaches. As expected, vanilla DANN fails under label distribution shift because a proportion of samples from the target inputs are mapped to regions of latent space corresponding to negative samples from the source domain (Figure 2.3(b)). In contrast, with our β-admissible distances, domain-adversarial networks are able to adapt successfully (Figure 2.3(c)), improving target accuracy from 89% (source-only) to 99% accuracy (with adaptation), except the cases where β is too small to admit a good target domain classifier (in this case we need $\beta \geq 0.9/0.5 - 1 = 0.8$). We also experiment with label-balanced target data (no label distribution shift). All approaches except source-only achieve an accuracy above 99%, so we do not present these results in a separate table.

Table 2.1: Classification accuracy on target domain with label distribution shift on a synthetic dataset.

METHOD	ACCURACY%		
SOURCE	89.4±1.1		
DANN	59.1±5.1	WDANN	50.8±32.1
β	0.5	2.0	4.0
FDANN-β	66.0± 41.6	**99.9± 0.0**	99.8±0.0
SDANN-β	**99.9± 0.1**	**99.9± 0.0**	**99.9±0.0**
WDANN1-β	45.7± 41.5	66.4± 41.1	**99.9±0.0**
WDANN2-β	97.6± 1.2	99.7± 0.2	99.5±0.3
SWDANN-β	79.0± 5.9	**99.9± 0.0**	**99.9±0.0**

Real datasets We experiment with the MNIST and USPS handwritten-digit datasets. For both directions (MNIST $\rightarrow$ USPS and USPS $\rightarrow$ MNIST), we experiment both with and without label distribution shift. The source domain is always class-balanced. To simulate label distribution shift, we sample target data from only half of the digits, e.g. [0-4] or [5-9]. Tables 2.2 and 2.3 show the target domain accuracy for different approaches with/without label distribution shift. As on synthetic datasets, we observe that DANN performs much worse than source-only training under label distribution shift. Compared to the original DANN, our approaches fare significantly better while achieving comparable performance absent label distribution shift.

2.6.1 Experiment Details

Synthetic datasets For source distribution, we sample class 0 from $\mathcal{N}([-1, -0.3], diag(0.1, 0.4))$ and class 1 from $\mathcal{N}([1, 0.3], diag(0.1, 0.4))$. For target distribution, we sample class 0 from $\mathcal{N}([-0.3, -1], diag(0.4, 0.1))$ and class 1 from $\mathcal{N}([0.3, 1], diag(0.4, 0.1))$. For label classifier, we use a fully-connect neural net with 3 hidden layers $(50, 50, 2)$ and the latent space is set as the last hidden layer. For domain classifier (critic) we use a fully-connect neural net with 2 hidden layers $(50, 50)$.

Table 2.2: Classification accuracy on target domain with/without label distribution shift on MNIST-USPS.

TARGET LABELS	[0-4] SHIFT	[5-9] SHIFT	[0-9] NO-SHIFT
SOURCE	74.3±1.0	59.5±3.0	66.7±2.1
DANN	50.0±1.9	28.2±2.8	78.5±1.6
FDANN-1	71.6±4.0	**67.5±2.3**	73.7±1.5
FDANN-2	74.3±2.5	61.9±2.9	72.6±0.9
FDANN-4	75.9±1.6	64.4±3.6	72.3±1.2
SDANN-1	71.6±3.7	49.1±6.3	81.0±1.3
SDANN-2	76.4±3.1	48.7±9.0	81.7±1.4
SDANN-4	**81.0±1.6**	60.8±7.5	**82.0±0.4**

Table 2.3: Classification accuracy on target domain with/without label distribution shift on USPS-MNIST.

TARGET LABELS	[0-4] SHIFT	[5-9] SHIFT	[0-9] NO-SHIFT
SOURCE	69.4±2.3	30.3±2.8	49.4±2.1
DANN	57.6±1.1	37.1±3.5	**81.9±6.7**
FDANN-1	80.4±2.0	40.1±3.2	75.4±4.5
FDANN-2	**86.6±4.9**	41.7±6.6	70.0±3.3
FDANN-4	77.6±6.8	34.7±7.1	58.5±2.2
SDANN-1	68.2±2.7	**45.4±7.1**	78.8±5.3
SDANN-2	78.6±3.6	36.1±5.2	77.4±5.7
SDANN-4	83.5±2.7	41.1±6.6	75.6±6.9

Image datasets For MNIST we subsample 2000 data points and for USPS we subsample 1800 data points. The subsampling process depends on the given label distribution (e.g. shift or no-shift). For label classifier, we use LeNet and the latent space is set as the last hidden layer. For domain classifier (critic) we use a fully-connect neural net with 2 hidden layers $(500, 500)$.

In all experiments, we use $\lambda = 1$ in the objective (2.4) and ADAM with learning rate 0.0001 and $\beta_1 = 0.5$ as the optimizer. We also apply a l2-regularization on the weights of ϕ and h with coefficient 0.001.

More discussion on synthetic experiments. The only unexcepted failure is WDANN1-2, which achieves only 20% accuracy in 2-out-of-5 runs. Looking in to the low accuracy runs we found that the l2-norm of the encoder weights is clearly higher than the successful runs. Large l2-norm of weights in ϕ likely results in a high Lipschitz constant L, which is undesirable according to our theory. We only implemented l2-regularization to encourage Lipschitz continuity of the encoder ϕ, which might be insufficient. How to enforce Lipschitz continuity of a neural network is still an open question. Trying more sophisticated approaches for Lipschitz continuity can a

future direction.

Choice of β. Since a good value of β may depend on the knowledge of target label distribution which is unknown, we experiment with different values of β. Empirically we did not find any clear pattern of correlation between value of β and performance as long as it is big enough to accommodate label distribution shift so we would leave it as an open question. In practice we suggest to use a moderate value such as 2 or 4, or estimate based on prior knowledge of target label distribution.

2.7 Related work

Our work makes distinct theoretical and algorithmic contributions to the domain adaptation literature. Concerning theory, we provide a risk bound that explains the behavior of domain-adversarial methods with model-independent assumptions on data distributions. Existing theories without assumptions of contained support [11, 12, 13, 24, 101] do not exhibit this property since (i) when applied to the input space, their results are not concerned with domain-adversarial learning as no latent space is introduced, (ii) when applied to the latent space, their unobservable constants/assumptions become ϕ-dependent, which is undesirable as explained in Section 2.4. Concerning algorithms, several prior works demonstrate empirical success of domain-adversarial approaches, [17, 46, 61, 125, 139, 140]. Among those, Cao et al. [21, 22] deal with the label distribution shift scenario through a heuristic reweighting scheme. However, their re-weighting presumes that they have a good classifier in the first place, creating a cyclic dependency.

2.8 Conclusions

We propose to use asymmetrically-relaxed distribution distances in domain-adversarial learning objectives, replacing standard ones which seek exact distribution matching in the latent space. While overcoming some limitations of the standard objectives under label distribution mismatch, we provide a theoretical guarantee for target domain performance under assumptions on data distributions. As our connectedness assumptions may not cover all cases where we expect domain adaptation to work in practice, (e.g. when the two domains are completely disjoint), providing analysis under other type of assumptions might of future interest.

2.9 Proofs

Derivation of (2.1).

$$\begin{aligned}
\mathcal{E}_U(\phi, h) &= \int \mathrm{d}x p_U(x) \left| h(\phi(x)) - f(x) \right| \\
&= \int \mathrm{d}x \int \mathrm{d}z p_U^\phi(z) \phi_U(x|z) \left| h(\phi(x)) - f(x) \right| \\
&= \int \mathrm{d}z p_U^\phi(z) \int \mathrm{d}x \phi_U(x|z) \left| h(z) - f(x) \right|
\end{aligned}$$

$$
\begin{aligned}
&= \int \mathrm{d}z p_U^\phi(z) \left| h(z) - \int \mathrm{d}x \phi_U(x|z) f(x) \right| \\
&\doteq \int \mathrm{d}z p_U^\phi(z) \left| h(z) - f_U^\phi(z) \right| \\
&\doteq \int \mathrm{d}z p_U^\phi(z) r_U(z; \phi, h)
\end{aligned}
$$

where we use the following fact: For any fixed z, $h(z) \in \{0, 1\}$, if $h(z) = 0$ then $|h(z) - f(x)| = f(x) - h(z)$ for all x. Similarly, when $h(z) = 1$, we have $|h(z) - f(x)| = h(z) - f(x)$ for all x. Thus we can move the integral over x inside the absolute operation. □

Proof of Proposition 2.3.1. First we have

$$
\rho_U = \int \mathrm{d}x p_U(x) f(x) = \int \mathrm{d}x \int \mathrm{d}z p_U^\phi(z) \phi_U(x|z) f(x) = \int \mathrm{d}z p_U^\phi(z) f_U^\phi(z) \,.
$$

When $\mathcal{E}_S(\phi, h) = 0$ we have

$$
\begin{aligned}
&\left| \int \mathrm{d}z p_S^\phi(z) h(z) - \rho_S \right| = \left| \int \mathrm{d}z p_S^\phi(z) h(z) - \int \mathrm{d}z p_S^\phi(z) f_S^\phi(z) \right| \\
&\leq \int \mathrm{d}z p_S^\phi(z) \left| h(z) - f_S^\phi(z) \right| = \mathcal{E}_S(\phi, h) = 0
\end{aligned}
$$

thus $\int \mathrm{d}z p_S^\phi(z) h(z) = \rho_S$.

Applying the fact that $p_S^\phi(z) = p_T^\phi(z)$ for all $z \in \mathcal{Z}$,

$$
\begin{aligned}
&\mathcal{E}_T(\phi, h) = \int \mathrm{d}z p_T^\phi(z) \left| h(z) - f_T^\phi(z) \right| \geq \left| \int \mathrm{d}z p_T^\phi(z) h(z) - \int \mathrm{d}z p_T^\phi(z) f_T^\phi(z) \right| \\
&= \left| \int \mathrm{d}z p_S^\phi(z) h(z) - \int \mathrm{d}z p_T^\phi(z) f_T^\phi(z) \right| = |\rho_S - \rho_T| \,,
\end{aligned}
$$

which concludes the proof. □

Proof of Proposition 2.3.2. Let p_S be the uniform distribution over $[0, 1]$ and p_T be the uniform distribution over $[2, 3]$. The labeling function f is set as $f(x) = 1$ iff $x \in [0, \rho_S] \cup [2, 2 + \rho_T]$ such that the definition of ρ_S and ρ_T is preserved. We construct the following mapping ϕ: For $x \in [0, 1]$ $\phi(x) = x$. For $x \in [2, 2 + \rho_T]$ $\phi(x) = (x - 2)\rho_S/\rho_T$. For $x \in [2 + \rho_T, 3]$ $\phi(x) = 1 - (3-x)(1-\rho_S)/(1-\rho_T)$. ϕ maps both source and target data into $[0, 1]$ with p_S^ϕ to be uniform over $[0, 1]$ and $p_T^\phi(z) = \rho_T/\rho_S$ when $z \in [0, \rho_S]$ and $p_T^\phi(z) = (1-\rho_T)/(1-\rho_S)$ when $z \in [\rho_S, 1]$. Since $p_S^\phi(z) = 1$ for all $z \in [0, 1]$ we can conclude that $\sup_{z \in \mathcal{Z}} p_T^\phi(z)/p_S^\phi(z) \leq \max\left\{ \frac{\rho_T}{\rho_S}, \frac{1-\rho_T}{1-\rho_S} \right\}$. □

Proof of Theorem 2.4.3. Instead of working with Assumption 2.4.2 we first extend Construction 2.4.1 with the following addition

Construction 2.9.1. (*Connectedness from target domain to source domain.*) Let $C_T \subset \mathcal{X}$ be a set of points in the raw data space that satisfy the following conditions:

1. $\phi(C_T) \subset \phi(C_0 \cup C_1)$.
2. For any $x \in C_T$, there exists $x' \in C_T \cap (C_0 \cup C_1)$ such that one can find a sequence of points $x_0, x_1, ..., x_m \in C_T$ with $x_0 = x$, $x_m = x'$, $f(x) = f(x')$ and $d_{\mathcal{X}}(x_{i-1}, x_i) < \frac{\Delta}{L}$ for all $i = 1, ..., m$.
3. $p_T(C_T) \geq 1 - \delta_3$.

We now proceed to prove bound based on Constructions 2.4.1 and 2.9.1. Later on we will show that Assumption 2.4.2 indicates the existence of Construction 2.9.1 so that the bound holds with a combination of Constructions 2.4.1 and Assumption 2.4.2.

The third term of (2.2) can be written as

$$
\begin{aligned}
&\int \mathrm{d}z p_S^\phi(z) \left(\frac{p_T^\phi(z)}{p_S^\phi(z)} - 1 \right) r_S(z; \phi, h) \\
&\leq \inf_{B \subseteq \mathcal{Z}} \int_B \mathrm{d}z p_S^\phi(z) \left(\frac{p_T^\phi(z)}{p_S^\phi(z)} - 1 \right) r_S(z; \phi, h) + \int_{B^c} \mathrm{d}z p_S^\phi(z) \left(\frac{p_T^\phi(z)}{p_S^\phi(z)} - 1 \right) r_S(z; \phi, h) \\
&\leq \inf_{B \subseteq \mathcal{Z}} \left(\sup_{z \in B} \frac{p_T^\phi(z)}{p_S^\phi(z)} - 1 \right) \int_B \mathrm{d}z p_S^\phi(z) r_S(z; \phi, h) + \int_{B^c} \mathrm{d}z p_T^\phi(z) r_S(z; \phi, h) \\
&\leq \inf_{B \subseteq \mathcal{Z}} \left(\sup_{z \in B} \frac{p_T^\phi(z)}{p_S^\phi(z)} - 1 \right) \mathcal{E}_S(\phi, h) + p_T^\phi(B^c) \\
&\leq \beta \mathcal{E}_S(\phi, h) + \delta_1 \,. \qquad (2.12)
\end{aligned}
$$

For the second term of (2.2), plugging in $r_U(z; \phi, h) = \left| h(z) - f_U^\phi(z) \right|$ gives

$$
\begin{aligned}
&\int \mathrm{d}z p_T^\phi(z) \left(r_T(z; \phi, h) - r_S(z; \phi, h) \right) \\
&= \int \mathrm{d}z p_T^\phi(z) \left(\left| h(z) - f_T^\phi(z) \right| - \left| h(z) - f_S^\phi(z) \right| \right) \\
&= \int \mathrm{d}z p_T^\phi(z) \left| f_T^\phi(z) - f_S^\phi(z) \right| \\
&= \int \mathrm{d}z p_T^\phi(z) \left| f_T^\phi(z) - f_S^\phi(z) \right| \left(\mathbb{1}\left\{ z \in \phi(C_0) \right\} + \mathbb{1}\left\{ z \in \phi(C_1) \right\} + \mathbb{1}\left\{ z \in (\phi(C_0) \cup \phi(C_1))^c \right\} \right) \\
&= \int \mathrm{d}z p_T^\phi(z) \left| f_T^\phi(z) - f_S^\phi(z) \right| \mathbb{1}\left\{ z \in \phi(C_0) \right\} + \int \mathrm{d}z p_T^\phi(z) \left| f_T^\phi(z) - f_S^\phi(z) \right| \mathbb{1}\left\{ z \in \phi(C_1) \right\} \\
&+ \int \mathrm{d}z p_T^\phi(z) \left| f_T^\phi(z) - f_S^\phi(z) \right| \mathbb{1}\left\{ z \in (\phi(C_0) \cup \phi(C_1))^c \right\} \qquad (2.13)
\end{aligned}
$$

Applying $\left| f_T^\phi(z) - f_S^\phi(z) \right| \leq f_T^\phi(z) + f_S^\phi(z)$ to the first part of (2.13) gives

$$
\begin{aligned}
&\int \mathrm{d}z p_T^\phi(z) \left| f_T^\phi(z) - f_S^\phi(z) \right| \mathbb{1}\left\{ z \in \phi(C_0) \right\} \\
&\leq \int \mathrm{d}z p_T^\phi(z) f_T^\phi(z) \mathbb{1}\left\{ z \in \phi(C_0) \right\} + \int \mathrm{d}z p_T^\phi(z) f_S^\phi(z) \mathbb{1}\left\{ z \in \phi(C_0) \right\}
\end{aligned}
$$

$$
\begin{aligned}
&= \int \mathrm{d}z p_T^\phi(z) \int \mathrm{d}x \phi_T(x|z) f(x) \mathbb{1}\left\{z \in \phi(C_0)\right\} + \int \mathrm{d}z p_T^\phi(z) f_S^\phi(z) \mathbb{1}\left\{z \in \phi(C_0)\right\} \\
&= \int \mathrm{d}x f(x) \int \mathrm{d}z p_T^\phi(z) \phi_T(x|z) \mathbb{1}\left\{z \in \phi(C_0)\right\} + \int \mathrm{d}z p_T^\phi(z) f_S^\phi(z) \mathbb{1}\left\{z \in \phi(C_0)\right\} \\
&= \int \mathrm{d}x f(x) p_T(x) \mathbb{1}\left\{\phi(x) \in \phi(C_0)\right\} + \int \mathrm{d}z p_T^\phi(z) f_S^\phi(z) \mathbb{1}\left\{z \in \phi(C_0)\right\} \\
&= \int \mathrm{d}x p_T(x) \mathbb{1}\left\{f(x) = 1, \phi(x) \in \phi(C_0)\right\} + \int \mathrm{d}z p_T^\phi(z) f_S^\phi(z) \mathbb{1}\left\{z \in \phi(C_0)\right\} \qquad (2.14)
\end{aligned}
$$

Similarly, applying $\left|f_T^\phi(z) - f_S^\phi(z)\right| = \left|(1 - f_T^\phi(z)) - (1 - f_S^\phi(z))\right| \leq (1 - f_T^\phi(z)) + (1 - f_S^\phi(z))$ to the second part of (2.13) gives

$$
\begin{aligned}
&\int \mathrm{d}z p_T^\phi(z) \left|f_T^\phi(z) - f_S^\phi(z)\right| \mathbb{1}\left\{z \in \phi(C_1)\right\} \\
&\leq \int \mathrm{d}z p_T^\phi(z)(1 - f_T^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} + \int \mathrm{d}z p_T^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \\
&= \int \mathrm{d}z p_T^\phi(z) \left(1 - \int \mathrm{d}x \phi_T(x|z) f(x)\right) \mathbb{1}\left\{z \in \phi(C_1)\right\} + \int \mathrm{d}z p_T^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \\
&= \int \mathrm{d}x (1 - f(x)) \int \mathrm{d}z p_T^\phi(z) \phi_T(x|z) \mathbb{1}\left\{z \in \phi(C_1)\right\} + \int \mathrm{d}z p_T^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \\
&= \int \mathrm{d}x (1 - f(x)) p_T(x) \mathbb{1}\left\{\phi(x) \in \phi(C_1)\right\} + \int \mathrm{d}z p_T^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \\
&= \int \mathrm{d}x p_T(x) \mathbb{1}\left\{f(x) = 0, \phi(x) \in \phi(C_1)\right\} + \int \mathrm{d}z p_T^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \qquad (2.15)
\end{aligned}
$$

Combining the second part of (2.14) and the second part of (2.15)

$$
\begin{aligned}
&\int \mathrm{d}z p_T^\phi(z) f_S^\phi(z) \mathbb{1}\left\{z \in \phi(C_0)\right\} + \int \mathrm{d}z p_T^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \\
&= \int \mathrm{d}z \frac{p_T^\phi(z)}{p_S^\phi(z)} p_S^\phi(z) f_S^\phi(z) \mathbb{1}\left\{z \in \phi(C_0)\right\} \left(\mathbb{1}\left\{z \in B\right\} + \mathbb{1}\left\{z \in B^c\right\}\right) \\
&+ \int \mathrm{d}z \frac{p_T^\phi(z)}{p_S^\phi(z)} p_S^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \left(\mathbb{1}\left\{z \in B\right\} + \mathbb{1}\left\{z \in B^c\right\}\right) \\
&\leq (1 + \beta) \int \mathrm{d}z p_S^\phi(z) f_S^\phi(z) \mathbb{1}\left\{z \in \phi(C_0)\right\} + (1 + \beta) \int \mathrm{d}z p_S^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\left\{z \in \phi(C_1)\right\} \\
&+ \int \mathrm{d}z p_T^\phi(z) \mathbb{1}\left\{z \in B^c\right\} \left(\mathbb{1}\left\{z \in \phi(C_0)\right\} + \mathbb{1}\left\{z \in \phi(C_1)\right\}\right) \\
&\leq (1 + \beta) \int \mathrm{d}x p_S(x) \mathbb{1}\left\{f(x) = 1, \phi(x) \in \phi(C_0)\right\} \\
&+ (1 + \beta) \int \mathrm{d}x p_S(x) \mathbb{1}\left\{f(x) = 0, \phi(x) \in \phi(C_1)\right\} + p_T(B^c) \\
&\leq (1 + \beta) \int \mathrm{d}x p_S(x) \left(\mathbb{1}\left\{f(x) = 1, \phi(x) \in \phi(C_0) \vee f(x) = 0, \phi(x) \in \phi(C_1)\right\}\right) + \delta_1 \qquad (2.16)
\end{aligned}
$$

For $i \in \{0,1\}$ if $x \in C_i$ then $f(x) = i$ and $\phi(x) \in C_i$. So if $f(x) = 1, \phi(x) \in \phi(C_0)$ or $f(x) = 0, \phi(x) \in \phi(C_1)$ holds we must have $x \notin C_0 \cup C_1$. Therefore, following (2.16) gives

$$\begin{aligned}
&\int \mathrm{d}z p_T^\phi(z) f_S^\phi(z) \mathbb{1}\{z \in \phi(C_0)\} + \int \mathrm{d}z p_T^\phi(z)(1 - f_S^\phi(z)) \mathbb{1}\{z \in \phi(C_1)\} \\
&\leq (1+\beta) \int \mathrm{d}x p_S(x) \mathbb{1}\{x \notin C_0 \cup C_1\} + \delta_1 \\
&= (1+\beta)(1 - p_S(C_0 \cup C_1)) + \delta_1 \\
&\leq (1+\beta)\delta_2 + \delta_1
\end{aligned} \tag{2.17}$$

Now looking at the first part of (2.14) and the first part of (2.15)

$$\begin{aligned}
&\int \mathrm{d}x p_T(x) \mathbb{1}\{f(x) = 1, \phi(x) \in \phi(C_0)\} + \int \mathrm{d}x p_T(x) \mathbb{1}\{f(x) = 0, \phi(x) \in \phi(C_1)\} \\
&= \int \mathrm{d}x p_T(x) \mathbb{1}\{f(x) = 1, \phi(x) \in \phi(C_0), x \in C_T\} \\
&+ \int \mathrm{d}x p_T(x) \mathbb{1}\{f(x) = 1, \phi(x) \in \phi(C_0), x \notin C_T\} \\
&+ \int \mathrm{d}x p_T(x) \mathbb{1}\{f(x) = 0, \phi(x) \in \phi(C_1), x \in C_T\} + \\
&\int \mathrm{d}x p_T(x) \mathbb{1}\{f(x) = 0, \phi(x) \in \phi(C_1), x \notin C_T\} \\
&\leq \int \mathrm{d}x p_T(x) \left(\mathbb{1}\{f(x) = 1, \phi(x) \in \phi(C_0), x \in C_T\} + \mathbb{1}\{f(x) = 0, \phi(x) \in \phi(C_1), x \in C_T\}\right) \\
&+ p_T(C_T^c) \\
&\leq \int \mathrm{d}x p_T(x) \mathbb{1}\{x \in C_T\} \mathbb{1}\{f(x) = 1, \phi(x) \in \phi(C_0) \vee f(x) = 0, \phi(x) \in \phi(C_1)\} + \delta_3 \,.
\end{aligned} \tag{2.18}$$

Next we show that the first part of (2.18) is 0. Recall that $\phi(C_T) \subset \phi(C_0 \cup C_1)$ and if $x \in C_T$ there exists $x' \in C_T \cap (C_0 \cup C_1)$ with a sequence of points $x_0, x_1, ..., x_m \in C_T$ such that $x_0 = x$, $x_m = x'$, $f(x) = f(x')$ and $d_{\mathcal{X}}(x_{i-1}, x_i) < \frac{\Delta}{L}$ for all $i = 1, ..., m$. So for $x \in C_T$ and $f(x) = i$, we pick such x'. Since ϕ is L-Lipschitz and $\phi(C_T) \subset \phi(C_0 \cup C_1)$ we have $\phi(x_0), \phi(x_1), ..., \phi(x_m) \in \phi(C_0 \cup C_1)$ and $d_{\mathcal{Z}}(\phi(x_{i-1}), \phi(x_i)) < \Delta$ for all $i = 1, ..., m$. Applying the fact that $\inf_{z_0 \in \phi(C_0), z_1 \in \phi(C_1)} d_{\mathcal{Z}}(z_0, z_1) \geq \Delta > 0$ we know that if $\phi(x) = \phi(x_0) \in \phi(C_j)$ for some $j \in \{0,1\}$ then $\phi(x') = \phi(x_m) \in \phi(C_j)$. From $x' \in C_0 \cup C_1$ and $f(x') = f(x) = i$ we have $\phi(x') \in \phi(C_i)$. Since $C_0 \cap C_1 = \emptyset$ we can conclude $i = j$ and thus $\phi(x) \in \phi(C_i)$ if $f(x) = i$ for any $x \in C_T$. Therefore, if $x \in C_T$, neither $f(x) = 1, \phi(x) \in \phi(C_0)$ nor $f(x) = 0, \phi(x) \in \phi(C_1)$ can hold. Hence the first part of (2.18) is 0.

So far by combining (2.17) and (2.18) we have shown that the sum of (2.14) and (2.15) (which are the first two parts of (2.13)) can be upper bounded by $\delta_1 + (1+\beta)\delta_2 + \delta_3$. For the third part of (2.13) we have

$$\int \mathrm{d}z p_T^\phi(z) \left| f_T^\phi(z) - f_S^\phi(z) \right| \mathbb{1}\{z \in (\phi(C_0) \cup \phi(C_1))^c\}$$

$$
\begin{aligned}
&\leq \int \mathrm{d}z p_T^{\phi}(z) \mathbb{1}\left\{z \in (\phi(C_0) \cup \phi(C_1))^c\right\} \\
&= \int \mathrm{d}z \frac{p_T^{\phi}(z)}{p_S^{\phi}(z)} p_S^{\phi}(z) \mathbb{1}\left\{z \in (\phi(C_0) \cup \phi(C_1))^c\right\} \left(\mathbb{1}\left\{z \in B\right\} + \mathbb{1}\left\{z \in B^c\right\}\right) \\
&\leq \int \mathrm{d}z \frac{p_T^{\phi}(z)}{p_S^{\phi}(z)} p_S^{\phi}(z) \mathbb{1}\left\{z \in (\phi(C_0) \cup \phi(C_1))^c\right\} \mathbb{1}\left\{z \in B\right\} + \int \mathrm{d}z p_T^{\phi}(z) \mathbb{1}\left\{z \in B^c\right\} \\
&\leq (1+\beta) \int \mathrm{d}z p_S^{\phi}(z) \mathbb{1}\left\{z \in (\phi(C_0) \cup \phi(C_1))^c\right\} + \delta_1 \\
&= (1+\beta) \left(1 - \int \mathrm{d}z p_S^{\phi}(z) \mathbb{1}\left\{z \in \phi(C_0) \cup \phi(C_1)\right\}\right) + \delta_1 \\
&= (1+\beta) \left(1 - \int \mathrm{d}x p_S(x) \mathbb{1}\left\{x \in \phi^{-1}\left(\phi(C_0) \cup \phi(C_1)\right)\right\}\right) + \delta_1 \\
&= (1+\beta) \left(1 - p_S\left(\phi^{-1}\left(\phi(C_0) \cup \phi(C_1)\right)\right)\right) + \delta_1 \\
&\leq (1+\beta) \left(1 - p_S\left(C_0 \cup C_1\right)\right) + \delta_1 \\
&\leq (1+\beta)\delta_2 + \delta_1 \,. \qquad (2.19)
\end{aligned}
$$

Putting (2.19) into (2.13) gives

$$
\int \mathrm{d}z p_T^{\phi}(z) \left(r_T(z;\phi,h) - r_S(z;\phi,h)\right) \leq 2\delta_1 + 2(1+\beta)\delta_2 + \delta_3 \,. \qquad (2.20)
$$

Plugging (2.12) and (2.20) into (2.2) gives the result of Theorem 2.4.3 under Constructions 2.4.1 and 2.9.1.

It remains to show that Assumption 2.4.2 implies the existence of a Construction 2.9.1. To prove this, we first write $\phi(C_T) \subset \phi(C_0 \cup C_1)$ as $C_T \subset \phi^{-1}(\phi(C_0 \cup C_1))$. By Construction 2.4.1 we have $p_S(C_0 \cup C_1) \geq 1 - \delta_2$. From (2.19) we have

$$
\begin{aligned}
p_T\left(\phi^{-1}(\phi(C_0 \cup C_1))\right) &= \int \mathrm{d}x p_T(x) \mathbb{1}\left\{x \in \phi^{-1}(\phi(C_0 \cup C_1))\right\} \\
&= \int \mathrm{d}z p_T^{\phi}(z) \mathbb{1}\left\{z \in \phi(C_0 \cup C_1)\right\} \geq (1+\beta)\delta_2 + \delta_1 \,.
\end{aligned}
$$

Setting $B_S = C_0 \cup C_1$ and $B_T = \phi^{-1}(\phi(C_0 \cup C_1)$ in Assumption 2.4.2 gives a construction of Construction 2.9.1, thus concluding the proof.

□

Proof of Corollary 2.4.5. Based on the statement of Corollary 2.4.5 it is obvious that Construction 2.4.1 can be made with $\delta_1 = 0$, $\delta_2 = 0$ and a finitely large L. (Here we implicitly assume that ϕ is bounded on $\mathcal{X}$). It remains to show that Assumption 2.4.2 holds with $\delta_3 = 0$. As $\delta_1 = \delta_2 = 0$, any B_S and B_T will be supersets of $\mathrm{Supp}(p_S)$ and $\mathrm{Supp}(p_T)$ respectively. So it sufficies to consider $B_S = \mathrm{Supp}(p_S)$ and $B_T = \mathrm{Supp}(p_T)$.

Now we verify that $C_T = \mathrm{Supp}(p_T)$ satisfies the requirements in Assumption 2.4.2. According to Assumption 2.4.4, for any $x \in \mathrm{Supp}(p_T)$, there must exist $S_{T,i,j}$ such that $x \in S_{T,i,j}$, $S_{T,i,j}$ is connected, $f(x') = i$ for all $x' \in S_{T,i,j}$ and $S_{T,i,j} \cap \mathrm{Supp}(p_S) \neq \emptyset$. Pick $x' \in S_{T,i,j} \cap \mathrm{Supp}(p_S)$.

Such x' satisfies $x' \in C_T \cap B_S$ with our choice of C_T and B_S. Since $S_{T,i,j}$ is connected we can find a sequence of points $x_0, ..., x_m \in S_{T,i,j}$ with $x_0 = 0$, $x_m = x'$ and $d_{\mathcal{X}}(x_{i-1}, x_i) < \epsilon$ for any $\epsilon > 0$. As $S_{T,i,j}$ is label consistent we have $f(x) = f(x')$. Picking $\epsilon = \frac{\Delta}{L}$ concludes the fact that $C_T = \mathrm{Supp}(p_T)$ satisfies the requirements in Assumption 2.4.2.

Since $p_T(\mathrm{Supp}(p_T)) = 1$ we have $\delta_3 = 0$. As a result, $\mathcal{E}_T(\phi, h) \leq (1+\beta)\mathcal{E}_S(\phi, h)$ holds according to Theorem 2.4.3, which concludes the proof of Corollary 2.4.5.

□

Derivation of (2.6). The Fenchel Dual of $\bar{f}_\beta(u)$ can be written as

$$\begin{aligned}
\bar{f}^*_\beta(t) &= \begin{cases} tf'^{-1}(t) - \bar{f}_\beta(f'^{-1}(t)) & \text{if } t \leq f'(\frac{1}{1+\beta})\,, \\ +\infty & \text{if } t > f'(\frac{1}{1+\beta})\,. \end{cases} \\
&= \begin{cases} tf'^{-1}(t) - f(f'^{-1}(t)) - C_{f,\beta} & \text{if } t \leq f'(\frac{1}{1+\beta})\,, \\ +\infty & \text{if } t > f'(\frac{1}{1+\beta})\,. \end{cases} \\
&= \begin{cases} f^*(t) - C_{f,\beta} & \text{if } t \leq f'(\frac{1}{1+\beta})\,, \\ +\infty & \text{if } t > f'(\frac{1}{1+\beta})\,. \end{cases},
\end{aligned}$$

where $C_{f,\beta} = -f(\frac{1}{1+\beta}) + f'(\frac{1}{1+\beta})\frac{1}{1+\beta} - f'(\frac{1}{1+\beta})$.

Therefore, the modified $\bar{f}_\beta$-divergence can be written as

$$D_{f,\beta}(p, q) = \sup_{T:\mathcal{Z}\mapsto \mathrm{dom}(f^*)\cap(-\infty, f'(\frac{1}{1+\beta})]} \mathbb{E}_{z\sim q}\left[T(z)\right] - \mathbb{E}_{z\sim p}\left[f^*(T(z))\right] + C_{f,\beta}\,.$$

□

Derivation of (2.7). According to Nowozin et al. [109], the GAN objecitve uses $f(u) = u\log u - (1+u)\log(1+u)$. Hence $f^*(t) = -\log(1-e^t)$, $f'(u) = \log\frac{u}{u+1}$ and $f'(\frac{1}{1+\beta}) = \log\frac{1}{2+\beta}$. So we need to parameterize $T : \mathcal{Z} \mapsto \big(-\infty, \log\frac{1}{2+\beta}\big]$. $T(z) = \log\frac{g(z)}{2+\beta}$ with $g(z) \in (0, 1]$ satisfies the range constraint for T. Plugging $T(z) = \log\frac{g(z)}{2+\beta}$ into (2.6) gives the result of (2.7).

□

Chapter 3

A Unified View of Label Shift Estimation

3.1 Overview

This chapter focuses on *label shift* [94, 116, 130], which aligns with the *anticausal* setting in which the labels y cause the features x [118]. Label shift arises in diagnostic problems because diseases cause symptoms. In this setting, an intervention on $p(y)$ induces the shift, but the process generating x given y is fixed ($p_S(x|y) = p_T(x|y)$). Under label shift, the optimal predictor may change, e.g., the probability that a patient suffers from a disease given their symptoms can increase under a pandemic. Contrast label shift with the better-known *covariate shift* assumption, which aligns with the assumption that x causes y, yielding the reverse implication that $p_S(y|x) = p_T(y|x)$.

Under label shift, our first task is to estimate the ratios $w(y) = p_T(y)/p_S(y)$ for all labels y. Two dominant approaches leverage off-the-shelf classifiers to estimate w: (i) *Black Box Shift Estimation* (BBSE) [94] and a variant called *Regularized Learning under Label Shift* (RLLS) [7]: moment-matching based estimators that leverage (possibly biased, uncalibrated, or inaccurate) predictions to estimate the shift; and (ii) Maximum Likelihood Label Shift (MLLS) [116]: an Expectation Maximization (EM) algorithm that assumes access to a classifier that outputs the true source distribution conditional probabilities $p_S(y|x)$.

Given a predictor $\widehat{f}$ with an invertible confusion matrix, BBSE and RLLS have known consistency results and finite-sample guarantees [7, 94]. However, MLLS, in combination with a calibration heuristic called Bias-Corrected Temperature Scaling (BCTS), outperforms them empirically [4].

In this work, we theoretically characterize MLLS, establishing conditions for consistency and bounding its finite-sample error. To start, we observe that given the true label conditional $p_S(y|x)$, MLLS is simply a Maximum Likelihood Estimation (MLE) problem and standard results apply. However, because we never know $p_S(y|x)$ exactly, MLLS is always applied with an estimated model $\widehat{f}$ and thus the procedure consists of MLE under model misspecification.

First, we prove that (i) *canonical calibration* (Definition 3.2.1) and (ii) an invertible confusion matrix (as required by BBSE) are *sufficient conditions* to ensure MLLS's consistency (Proposition 3.4.4 and Theorems 3.4.3). Recall that neural network classifiers tend to be uncalibrated absent post-hoc adjustments [56]. Second, we observe that confusion matrices can be

instruments for calibrating a classifier. Applying MLLS with this technique, BBSE and MLLS are distinguished only by their objective functions. Through extensive experiments, we show that they perform similarly, concluding that MLLS's superior performance (when applied with more granular calibration techniques) is not due to its objective but rather to the information lost by BBSE via confusion matrix calibration. Third, we analyze the finite-sample error of the MLLS estimator by decomposing its error into terms reflecting the miscalibration error and finite-sample error (Theorem 3.5.4). Our experiments relate MLLS's MSE to the granularity of the calibration.

In summary, we contribute the following: (i) Sufficient conditions for MLLS's consistency; (ii) Unification of MLLS and BBSE methods under a common framework, with BBSE corresponding to a particular choice of calibration method; (iii) Finite-sample error bounds for MLLS; (iv) Experiments that support our theoretical arguments.

3.2 Problem Setup

Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \{1, 2, \ldots, k\}$ the output space. Let p_S, p_T be the source and target distributions and their corresponding probability density (or mass) functions. We use $\mathbb{E}_S$ and $\mathbb{E}_T$ to denote expectations over the source and target distributions. In unsupervised domain adaptation, we possess labeled source data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ and unlabeled target data $\{x_{n+1}, x_{n+2}, \ldots, x_{n+m}\}$. We also assume access to a black-box predictor $\widehat{f} : \mathcal{X} \mapsto \Delta^{k-1}$, e.g., a model trained to approximate the true probability function f^*, where $f^*(x) := p_S(\cdot|x)$. Here and in the rest of this chapter, we use Δ^{k-1} to denote the standard k-dimensional probability simplex. For a vector v, we use v_y to access the element at index y.

Absent assumptions relating the source and target distributions, domain adaptation is underspecified [14]. We work with the *label shift* assumption, i.e., $p_S(x|y) = p_T(x|y)$, focusing on multiclass classification. Moreover, we assume non-zero support for all labels in the source distribution: for all $y \in \mathcal{Y}$, $p_S(y) \geq c > 0$ [7, 94]. Under label shift, three common goals are (i) detection—determining whether distribution shift has occurred; (ii) quantification—estimating the target label distribution; and (iii) correction—producing a predictor that minimizes error on the target distribution [94].

This work focuses on goal (ii), estimating importance weights $w(y) = p_T(y)/p_S(y)$ for all $y \in \mathcal{Y}$. Given w, we can update our classifiers on the fly, either by retraining in an importance-weighted ERM framework [7, 54, 94, 123]—a practice that may be problematic for overparameterized neural networks [20], or by applying an analytic correction [4, 116]. Within the ERM framework, the generalization result from Azizzadenesheli et al. [7] (Theorem 1) depends only on the error of the estimated weights, and hence any method that improves weight estimates tightens this bound.

There are multiple definitions of calibration in the multiclass setting. Guo et al. [56] study the calibration of the arg-max prediction, while Kumar et al. [76] study a notion of per-label calibration. We use canonical calibration [141] and the expected canonical calibration error on the source data defined as follows:

Definition 3.2.1 (Canonical calibration)**.** A prediction model $f : \mathcal{X} \mapsto \Delta^{k-1}$ is canonically calibrated on the source domain if for all $x \in \mathcal{X}$ and $j \in \mathcal{Y}$, $p_S(y = j|f(x)) = f_j(x)$.

Definition 3.2.2 (Expected canonical calibration error)**.** For a predictor f, the expected squared canonical calibration error on the source domain is $\mathcal{E}^2(f) = \mathbb{E}_S \left\| f - f_c \right\|^2$, where $f_c = p_S(y = \cdot | f(x))$.

Calibration methods typically work either by calibrating the model during training or by calibrating a trained classifier on held-out data, post-hoc. We refer the interested reader to Kumar et al. [76] and Guo et al. [56] for detailed studies on calibration. We focus on the latter category of methods. Our experiments follow Alexandari et al. [4], who leverage BCTS [1] to calibrate their models. BCTS extends temperature scaling [56] by incorporating per-class bias terms.

3.3 Related Work

Two families of solutions have been explored that leverage a blackbox predictor: BBSE [94], a moment matching method, uses the predictor $\widehat{f}$ to compute a confusion matrix $C_{\widehat{f}} := p_S(\widehat{y}, y) \in \mathbb{R}^{k \times k}$ on the source data. Depending on how $\widehat{y}$ is defined, there are two types of confusion matrix for a predictor $\widehat{f}$: (i) the *hard confusion matrix* $\widehat{y} = \arg\max \widehat{f}(x)$; and (ii) the *soft confusion matrix*, where $\widehat{y}$ is defined as a random prediction that follows the discrete distribution $\widehat{f}(x)$ over $\mathcal{Y}$. Both soft and hard confusion matrix can be estimated from labeled source data samples. The estimate $\widehat{w}$ is computed as $\widehat{w} := \widehat{C}_{\widehat{f}}^{-1} \widehat{\mu}$, where $\widehat{C}_{\widehat{f}}$ is the estimate of confusion matrix and $\widehat{\mu}$ is an estimate of $p_T(\widehat{y})$, computed by applying the predictor $\widehat{f}$ to the target data. In a related vein, RLLS [7] incorporates an additional regularization term of the form $\|w - 1\|$ and solves a constrained optimization problem to estimate the shift ratios w.

MLLS estimates w as if performing maximum likelihood estimation, but substitutes the predictor outputs for the true probabilities $p_S(y|x)$. Saerens et al. [116], who introduce this procedure, describe it as an application of EM. However, as observed in [4, 36], the likelihood objective is concave, and thus a variety of optimization algorithms may be applied to recover the MLLS estimate. Alexandari et al. [4] also showed that MLLS underperforms BBSE when applied naively, a phenomenon that we shed more light on in this work.

3.4 A Unified View

We now present a unified view that subsumes MLLS and BBSE and demonstrate how each is instantiated under this framework. We also establish identifiability and consistency conditions for MLLS, deferring a treatment of finite-sample issues to Section 3.5. For convenience, throughout Sections 3.4 and 3.5, we use the term *calibration* exclusively to refer to canonical calibration (Definition 3.2.1) on the source data.

3.4.1 A Unified Distribution Matching View

To start, we introduce a *generalized* distribution matching approach for estimating w. Under label shift, for any (possibly randomized) mapping from $\mathcal{X}$ to $\mathcal{Z}$, we have that $p_S(z|y) = p_T(z|y)$ since, $p_S(z|y) = p_T(z|y) = \int_{\mathcal{X}} p(z|x)p(x|y)dx$. Throughout the chapter, we use the notation

[1] Motivated by the strong empirical results in Alexandari et al. [4], we use BCTS in our experiments as a surrogate to canonical calibration.

$p(z|y)$ to represent either $p_S(z|y)$ or $p_T(z|y)$ (which are identical). We now define a family of distributions over $\mathcal{Z}$ parameterized by $w \in \mathcal{W}$ as

$$p_w(z) = \sum_{y=1}^{k} p(z|y) p_S(y) w_y = \sum_{y=1}^{k} p_S(z,y) w_y, \tag{3.1}$$

where $\mathcal{W} = \{w \mid \forall y\,, w_y \geq 0 \text{ and } \sum_{y=1}^{k} w_y p_S(y) = 1\}$. When $w = w^*$, we have that $p_w(z) = p_T(z)$. For fixed $p(z|x)$, $p_T(z)$ and $p_S(z,y)$ are known because $p_T(x)$ and $p_S(x,y)$ are known. So one potential strategy to estimate w^* is to find a weight vector w such that

$$\sum_{y=1}^{k} p_S(z,y) w_y = p_T(z) \quad \forall z \in \mathcal{Z}\,. \tag{3.2}$$

At least one such weight vector w must exist as w^* satisfies (3.2). We now characterize conditions under which the weight vector w satisfying (3.2) is unique:

Lemma 3.4.1 (Identifiability)**.** If the set of distributions $\{p(z|y) : y = 1, ..., k\}$ are linearly independent, then for any w that satisfies (3.2), we must have $w = w^*$. This condition is also necessary in general: if the linear independence does not hold then there exists a problem instance where we have $w, w^* \in \mathcal{W}$ satisfying (3.2) while $w \neq w^*$.

Lemma 3.4.1 follows from the fact that (3.2) is a linear system with at least one solution w^*. This solution is unique when $p_S(z,y)$ is of rank k. The linear independence condition in Lemma 3.4.1, in general, is sufficient for identifiability of discrete $\mathcal{Z}$. However, for continuous $\mathcal{Z}$, the linear dependence condition has the undesirable property of being sensitive to changes on sets of measure zero. By changing a collection of linearly dependent distributions on a set of measure zero, we can make them linearly independent. As a consequence, we impose a *stronger* notion of identifiability i.e., the set of distributions $\{p(z|y) : y = 1, ..., k\}$ are such that there does not exist $v \neq 0$ for which $\int_{\mathcal{Z}} |\sum_y p(z|y) v_y| dz = 0$. We refer this condition as *strict linear independence*.

In generalized distribution matching, one can set $p(z|x)$ to be the Dirac delta function at δ_x[2] such that $\mathcal{Z}$ is the same space as $\mathcal{X}$, which leads to solving (3.2) with z replaced by x. In practice where $\mathcal{X}$ is high-dimensional and/or continuous, approximating the solution to (3.2) from finite samples can be hard when choosing $z = x$. Our motivation for generalizing distribution matching from $\mathcal{X}$ to $\mathcal{Z}$ is that the solution to (3.2) can be better approximated using finite samples when $\mathcal{Z}$ is chosen carefully. Under this framework, the design of a label shift estimation algorithm can be decomposed into two parts: (i) the choice of $p(z|x)$ and (ii) how to approximate the solution to (3.2). Later on, we consider how these design choices may affect label shift estimation procedures in practice.

3.4.2 The Confusion Matrix Approach

If $\mathcal{Z}$ is a discrete space, one can first estimate $p_S(z,y) \in \mathbb{R}^{|\mathcal{Z}| \times k}$ and $p_T(z) \in \mathbb{R}$, and then subsequently attempt to solve (3.2). Confusion matrix approaches use $\mathcal{Z} = \mathcal{Y}$, and construct $p(z|x)$ using a black box predictor $\widehat{f}$. There are two common choices to construct the confusion matrix: (i) The soft confusion matrix approach: We set $p(z|x) := \widehat{f}(x) \in \Delta^{k-1}$. We then define a

[2]For simplicity we will use $z = x$ to denote that $p(z|x) = \delta_x$.

random variable $\widehat{y} \sim \widehat{f}(x)$ for each x. Then we construct $p_S(z, y) = p_S(\widehat{y}, y)$ and $p_T(z) = p_T(\widehat{y})$. (ii) The hard confusion matrix approach: Here we set $p(z|x) = \delta_{\arg\max \widehat{f}(x)}$. We then define a random variable $\widehat{y} = \arg\max \widehat{f}(x)$ for each x. Then again we have $p_S(z, y) = p_S(\widehat{y}, y)$ and $p_T(z) = p_T(\widehat{y})$.

Since $p_S(z, y)$ is a square matrix, the identifiability condition becomes the invertibility of the confusion matrix. Given an estimated confusion matrix, one can find w by inverting the confusion matrix (BBSE) or minimizing some distance between the vectors on the two sides of (3.2).

3.4.3 Maximum Likelihood Label Shift Estimation

When $\mathcal{Z}$ is a continuous space, the set of equations in (3.2) indexed by $\mathcal{Z}$ is intractable. In this case, one possibility is to find a weight vector $\widetilde{w}$ by minimizing the KL-divergence $\mathrm{KL}(p_T(z), p_w(z)) = \mathbb{E}_T\left[\log p_T(z)/p_w(z)\right]$, for p_w defined in (3.1). This is equivalent to maximizing the population log-likelihood: $\widetilde{w} := \arg\max_{w \in \mathcal{W}} \mathbb{E}_T\left[\log p_w(z)\right]$. One can further show that $\mathbb{E}_T\left[\log p_w(z)\right] = \mathbb{E}_T[\log \sum_{y=1}^k p_S(z, y) w_y] = \mathbb{E}_T[\log \sum_{y=1}^k p_S(y|z) p_S(z) w_y] = \mathbb{E}_T[\log \sum_{y=1}^k p_S(y|z) w_y] + \mathbb{E}_T\left[\log p_S(z)\right]$. Therefore we can equivalently define:

$$\widetilde{w} := \arg\max_{w \in \mathcal{W}} \mathbb{E}_T\Big[\log \sum\nolimits_{y=1}^k p_S(y|z) w_y\Big]. \tag{3.3}$$

This yields a straightforward convex optimization problem whose objective is bounded from below [4, 36]. Assuming access to labeled source data and unlabeled target data, one can maximize the empirical counterpart of the objective in (3.3), using either EM or an alternative iterative optimization scheme. Saerens et al. [116] derived an EM algorithm to maximize the objective (3.3) when $z = x$, assuming access to $p_S(y|x)$. Absent knowledge of the ground truth $p_S(y|x)$, we can plug in any approximate predictor f and optimize the following objective:

$$w_f := \arg\max_{w \in \mathcal{W}} \mathcal{L}(w, f) := \arg\max_{w \in \mathcal{W}} \mathbb{E}_T\left[\log f(x)^T w\right]. \tag{3.4}$$

In practice, f is fit from a finite sample drawn from $p_S(x, y)$ and standard machine learning methods often produce uncalibrated predictors. While BBSE and RLLS are provably consistent whenever the predictor f yields an invertible confusion matrix, to our knowledge, no prior works have established sufficient conditions to guarantee MLLS' consistency when f differs from $p_S(y|x)$.

It is intuitive that for some values of $f \neq p_S(y|x)$, MLLS will yield inconsistent estimates. Supplying empirical evidence, Alexandari et al. [4] show that MLLS performs poorly when f is a vanilla neural network predictor learned from data. However, Alexandari et al. [4] also show that in combination with a particular post-hoc calibration technique, MLLS achieves low error, significantly outperforming BBSE and RLLS. As the calibration error is not a distance metric between f and $p_S(y|x)$ (zero calibration error does not indicate $f = p_S(y|x)$), a calibrated predictor f may still be substantially different from $p_S(y|x)$. Some natural questions then arise:

1. *Why does calibration improve MLLS so dramatically?*
2. *Is calibration necessary or sufficient to ensure the consistency of MLLS?*

3. *What accounts for the comparative efficiency of MLLS over BBSE?* (Addressed in Section 3.5)

To address the first two questions, we make the following observations. Suppose we define z (for each x) with distribution $p(z|x) := \delta_{f(x)}$, for some calibrated predictor f. Then, because f is calibrated, it holds that $p_S(y|z) = f(x)$. Note that in general, the MLLS objective (3.4) can differ from (3.3). However, when $p(z|x) := \delta_{f(x)}$, the two objectives are identical. We can formalize this as follows:

Lemma 3.4.2. If f is calibrated, then the two objectives (3.3) and (3.4) are identical when $\mathcal{Z}$ is chosen as Δ^{k-1} and $p(z|x)$ is defined to be $\delta_{f(x)}$.

Lemma 3.4.2 follows from changing the variable of expectation in (3.4) from x to $f(x)$ and applying $f(x) = p_S(y|f(x))$ (definition of calibration). It shows that MLLS with a calibrated predictor on the input space $\mathcal{X}$ is in fact equivalent to performing distribution matching in the space $\mathcal{Z}$. Building on this observation, we now state our population-level consistency theorem for MLLS:

Theorem 3.4.3 (Population consistency of MLLS)**.** If a predictor $f : \mathcal{X} \mapsto \Delta^{k-1}$ is calibrated and the distributions $\{p(f(x)|y) : y = 1, \ldots, k\}$ are strictly linearly independent, then w^* is the unique maximizer of the MLLS objective (3.4).

Let $x_1, x_2, \ldots, x_m \overset{iid}{\sim} p_T(x)$. The finite sample objective for MLLS can be written as

$$\widehat{w}_f := \arg\max_{w \in \mathcal{W}} \frac{1}{m} \sum\nolimits_{i=1}^{m} \log f(x_i)^T w := \arg\max_{w \in \mathcal{W}} \mathcal{L}_m(w, f)\,. \tag{3.5}$$

The consistency of MLLS relies on the linear independence of the collection of distributions $\{p(f(x)|y) : y = 1, \ldots, k\}$. The following result develops several alternative equivalent characterizations of this linear independence condition.

Proposition 3.4.4. For a calibrated predictor f, the following statements are equivalent:

(1) $\{p(f(x)|y) : y = 1, \ldots, k\}$ are strictly linearly independent.

(2) $\mathbb{E}_S\left[f(x)f(x)^T\right]$ is invertible.

(3) The soft confusion matrix of f is invertible.

Proposition 3.4.4 shows that with a calibrated predictor, the invertibility condition as required by BBSE (or RLLS) is exactly the same as the linear independence condition required for MLLS's consistency.

3.4.4 MLLS with Confusion Matrix

So far, we have shown that MLLS with any calibrated predictor can be viewed as distribution matching in a latent space. Now we discuss a method to construct a predictor f to perform MLLS given any $p(z|x)$, e.g., those induced by confusion matrix approaches. Recall, we already have the maximum log-likelihood objective. It just remains to construct a calibrated predictor f from the confusion matrix.

This is straightforward when $p(z|x)$ is deterministic, i.e., $p(z|x) = \delta_{g(x)}$ for some function g: setting $f(x) = p_S(y|g(x))$ makes the objectives (3.3) and (3.4) to be the same. Recall that for the hard confusion matrix, the induced latent space is $p(z|x) = \delta_{\arg\max \widehat{f}(x)}$. So the corresponding

predictor in MLLS is $f(x) = p_S(y|\widehat{y}_x)$, where $\widehat{y}_x = \arg\max \widehat{f}(x)$. Then we obtain the MLLS objective for the hard confusion matrix:

$$\max_{w \in \mathcal{W}} \mathbb{E}_T \left[\log \sum\nolimits_{y=1}^{k} p_S(y|\widehat{y}_x) w_y \right] . \tag{3.6}$$

The confusion matrix $C_{\widehat{f}}$ and predictor $\widehat{f}$ directly give us $p_S(y|\widehat{y}_x)$. Given an input x, one can first get $\widehat{y}_x$ from $\widehat{f}$, then normalize the $\widehat{y}_x$-th row of $C_{\widehat{f}}$ as $p_S(y|\widehat{y}_x)$. We denote MLLS with hard confusion matrix calibration (3.6) by MLLS-CM.

When $p_S(z|x)$ is stochastic, we need to extend (3.4) to allow f to be a random predictor: $f(x) = p_S(y|z)$ for $z \sim p(z|x)$[3]. To incorporate the randomness of f, one only needs to change the expectation in (3.4) to be over both x and $f(x)$, then (3.4) becomes a rewrite of (3.3).

Proposition 3.4.5 indicates that constructing the confusion matrix is a calibration procedure. Thus, the predictor constructed with constructed using confusion matrix is calibrated and suitable for application with MLLS.

Proposition 3.4.5 (Vaicenavicius et al. [141])**.** For any function g, $f(x) = p_S(y|g(x))$ is a calibrated predictor.

We can now summarize the relationship between BBSE and MLLS: A label shift estimator involves two design choices: (i) designing the latent space $p(z|x)$ (which is equivalent to designing a calibrated predictor); and (ii) performing distribution matching in the new space $\mathcal{Z}$. In BBSE, we design a calibrated predictor via the confusion matrix and then perform distribution matching by directly solving linear equations. In general, MLLS does not specify how to obtain a calibrated predictor, but specifies KL minimization as the distribution matching procedure. One can apply the confusion matrix approach to obtain a calibrated predictor and then plug it into MLLS, which is the BBSE analog under MLLS, and is a special case of MLLS.

3.5 Finite-Sample Analysis

We now analyze the performance of MLLS estimator. Even when w^* is the unique optimizer of (3.4) for some calibrated predictor f, assuming convex optimization can be done perfectly, there are still two sources of error preventing us from exactly computing w^* in practice. First, we are optimizing a sample-based approximation (3.5) to the objective in expectation (3.4). We call this source of error *finite-sample error*. Second, the predictor f we use may not be perfectly calibrated on the source distribution as we only have access to samples from source data distribution $p_S(x, y)$. We call this source of error *miscalibration error*. We will first analyze how these two sources of errors affect the estimate of w^* separately and then give a general error bound that incorporates both.

Before presenting our analysis, we introduce some notation and regularity assumptions. For any predictor $f : \mathcal{X} \mapsto \Delta^{k-1}$, we define w_f and $\widehat{w}_f$ as in (3.4) and (3.5). If f satisfies the conditions in Theorem 3.4.3 (calibration and linear independence) then we have that $w_f = w^*$. Our goal is to bound $\|\widehat{w}_f - w^*\|$ for a given (possibly miscalibrated) predictor f. We now introduce a regularity condition:

[3]Here, by a random predictor we mean that the predictor outputs a random vector from Δ^{k-1}, not $\mathcal{Y}$.

Condition 3.5.1 (Regularity condition for a predictor f)**.** For any x within the support of $p_T(x)$, i.e. $p_T(x) > 0$, we have both $f(x)^T w_f \geq \tau$, $f(x)^T w^* \geq \tau$ for some universal constant $\tau > 0$.

Condition 3.5.1 is mild if f is calibrated since in this case $w_f = w^*$ is the maximizer of $\mathbb{E}_T \left[\log f(x)^T w\right]$, and the condition is satisfied if the expectation is finite. Since $f(x)^T w^*$ and $f(x)^T w_f$ are upper-bounded (they are the inner products of two vectors which sum to 1), they also must be lower-bounded away from 0 with arbitrarily high probability without any assumptions. For miscalibrated f, a similar justification holds for assumption that $f(x)^T w_f$ is lower bounded. Turning our attention to the assumption that $f(x)^T w^*$ is lower bounded, we note that it is sufficient if f is close (pointwise) to some calibrated predictor. This in turn is a reasonable assumption on the actual predictor we use for MLLS in practice as it is post-hoc calibrated on source data samples.

Define $\sigma_{f,w}$ to be the minimum eigenvalue of the Hessian $-\nabla_w^2 \mathcal{L}(w, f)$. To state our results compactly we use standard stochastic order notation (see, for instance, [142]). We first bound the estimation error introduced by only having finite samples from the target distribution in Lemma 3.5.2. Next, we bound the estimation error introduced by having a miscalibrated f in Lemma 3.5.3.

Lemma 3.5.2. For any predictor f that satisfies Condition 3.5.1, we have

$$\|w_f - \widehat{w}_f\| \leq \sigma_{f,w_f}^{-1} \mathcal{O}_p\left(m^{-1/2}\right).$$

Lemma 3.5.3. For any predictor f and any calibrated predictor f_c that satisfies Condition 3.5.1, we have

$$\|w_f - w^*\| \leq \sigma_{f,w^*}^{-1} \cdot C \cdot \mathbb{E}_T\left[\|f - f_c\|\right],$$

for some constant C.

If we set $f_c(x) = p_S(y|f(x))$, which is a calibrated predictor (Proposition 3.4.5), we can bound the error in terms of the calibration error of f on the source data [4]: $\|w_f - w^*\| \leq \sigma_{f,w^*}^{-1} \cdot C \cdot \mathcal{E}(f)$.

Note that since $p_S(y) > 0$ for all y, we can upper-bound the error in Lemma 3.5.3 with calibration error on the source data. We combine the two sources of error to bound the estimation error $\|\widehat{w}_f - w^*\|$:

Theorem 3.5.4. For any predictor f that satisfies Condition 3.5.1, we have

$$\|\widehat{w}_f - w^*\| \leq \sigma_{f,w_f}^{-1} \mathcal{O}_p\left(m^{-1/2}\right) + C \cdot \sigma_{f,w^*}^{-1} \mathcal{E}(f). \tag{3.7}$$

The estimation error of MLLS can be decomposed into (i) finite-sample error, which decays at a rate of $m^{-1/2}$; and (ii) the calibration error of the predictor that we use. The proof is a direct combination of Lemma 3.5.2 and Lemma 3.5.3 applied to the same f with the following error decomposition:

$$\|\widehat{w}_f - w^*\| \leq \underbrace{\|w_f - \widehat{w}_f\|}_{\text{finite-sample}} + \underbrace{\|w_f - w^*\|}_{\text{miscalibration}}.$$

[4] We present two upper bounds because the second is more interpretable while the first is tighter.

Theorem 3.5.4 shows that the estimation error depends inversely on the minimum eigenvalue of the Hessian at two different points w_f and w^*. One can unify these two eigenvalues as a single quantity σ_f, the minimum eigenvalue $\mathbb{E}_T\left[f(x)f(x)^T\right]$:

Proposition 3.5.5. For any $w \in \mathcal{W}$, we have $\sigma_{f,w} \geq p_{S,\min}\sigma_f$ where σ_f is the minimum eigenvalue of $\mathbb{E}_T\left[f(x)f(x)^T\right]$ and $p_{S,\min} = \min_{y\in\mathcal{Y}} p_S(y)$. Furthermore, if f satisfies Condition 3.5.1, we have $p_{S,\min}^2 \cdot \sigma_f \leq \sigma_{f,w} \leq \tau^{-2} \cdot \sigma_f$, for $w \in \{w_f, w^*\}$.

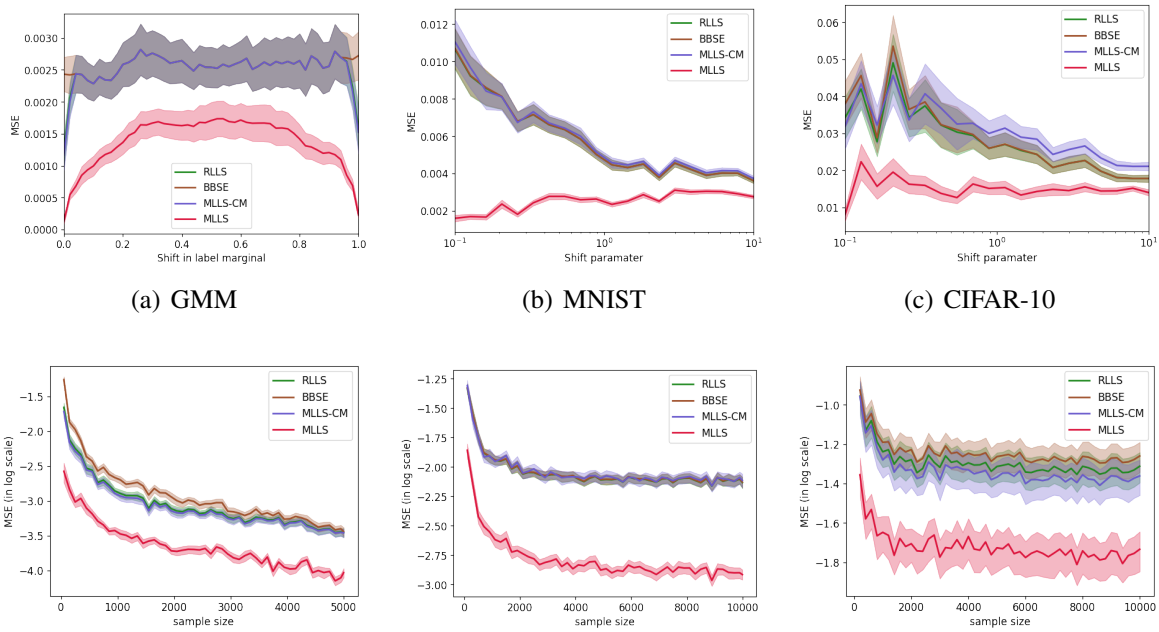
The estimation error bound explains the efficiency of MLLS. Informally, the error of MLLS depends inversely on the minimum eigenvalue of the Hessian of the likelihood σ_f. When we apply coarse calibration via the confusion matrix (in MLLS-CM), we only decrease the value of σ_f. Coarse calibration throws away information [75] and thus results in greater estimation error for MLLS. In Section 3.6, we empirically show that MLLS-CM's performance is similar to that of BBSE. Moreover, we show that the minimum eigenvalue of the Hessian obtained using confusion matrix calibration is smaller than the minimum eigenvalue obtained with more granular calibration. Our analysis and observations together suggest MLLS's superior performance than BBSE (or RLLS) is due to the granular calibration but not due to the difference in the optimization objective.

Finally, we want to highlight one minor point regarding applicability of our result. If f is calibrated, Theorem 3.5.4, together with Proposition 3.5.5, implies that MLLS is consistent if $\mathbb{E}_T\left[f(x)f(x)^T\right]$ is invertible. Compared to the consistency condition in Theorem 3.4.3 that $\mathbb{E}_S\left[f(x)f(x)^T\right]$ is invertible (together with Proposition 3.4.4), these two conditions are the same if the likelihood ratio $p_T(f(x))/p_S(f(x))$ is lower-bounded. This is true if all entries in w^* are non-zero. Even if w^* contains non-zero entries, the two conditions are still the same if there exists some $w_y^* > 0$ such that $p(f(x)|y)$ covers the full support of $p_S(f(x))$. In general however, the invertibility of $\mathbb{E}_T\left[f(x)f(x)^T\right]$ is a stronger requirement than the invertibility of $\mathbb{E}_S\left[f(x)f(x)^T\right]$. We leave further investigation of this gap for future work.

3.6 Experiments

We experimentally illustrate the performance of MLLS on synthetic data, MNIST [86], and CIFAR10 [74]. Following Lipton et al. [94], we experiment with *Dirichlet shift* simulations. On each run, we sample a target label distribution $p_T(y)$ from a Dirichlet with concentration parameter α. We then generate each target example by first sampling a label $y \sim p_T(y)$ and then sampling (with replacement) an example conditioned on that label . Note that smaller values of alpha correspond to more severe shift. In our experiments, the source label distribution is uniform.

First, we consider a mixture of two Gaussians with $\mu = 1$. With CIFAR10 and MNIST, we split the full training set into two subsets: train and valid, and use the provided test set as is. Then according to the label distribution, we randomly sample with replacement train, valid, and test set from each of their respective pool to form the source and target set. To learn the black box predictor on real datasets, we use the same architecture as Lipton et al. [94] for MNIST, and for CIFAR10 we use ResNet-18 [58] as in Azizzadenesheli et al. [7]. For simulated data, we use the true $p_S(y|x)$ as our predictor function. For each experiment, we sample 100 datasets for each shift parameter and evaluate the empirical MSE and variance of the estimated weights.



(a) GMM (b) MNIST (c) CIFAR-10

(d) GMM (e) MNIST (f) CIFAR-10

Figure 3.1: (**top**) MSE vs the degree of shift; For GMM, we control the shift in the label marginal for class 1 with a fixed target sample size of 1000. For multiclass problems—-MNIST and CIFAR-10, we control the Dirichlet shift parameter with a fixed sample size of 5000. (**bottom**) MSE (in log scale) vs target sample size; For GMM, we fix the label marginal for class 1 at 0.01 whereas for multiclass problems, MNIST and CIFAR-10, we fix the Dirichlet parameter to 0.1. In all plots, MLLS dominates other methods. All confusion matrix approaches perform similarly, indicating that the advantage of MLLS comes from the choice of calibration but not the way of performing distribution matching.

We consider three sets of experiments: (1) MSE vs degree of target shift; (2) MSE vs target sample sizes; and (3) MSE vs calibrated predictors on the source distribution. We refer to MLLS-CM as MLLS with hard confusion matrix calibration as in (3.6). In our experiments, we compare MLLS estimator with BBSE, RLLS, and MLLS-CM. For RLLS and BBSE, we use the publicly available code. To post-hoc calibration, we use BCTS [4] on the held-out validation set. Using the same validation set, we calculate the confusion matrix for BBSE, RLLS, and MLLS-CM.

We examine the performance of various estimators across all three datasets for various target dataset sizes and shift magnitudes (Figure 3.1). Across all shifts, MLLS (with BCTS-calibrated classifiers) *uniformly dominates* BBSE, RLLS, and MLLS-CM in terms of MSE (Figure 3.1). Observe for severe shifts, MLLS is comparatively dominant. As the available target data increased, all methods improve rapidly, with MLLS outperforming all other methods by a significant margin. Moreover, MLLS's advantages grow more pronounced under extreme shifts. Notice MLLS-CM is roughly equivalent to BBSE across all settings of dataset, target size, and shift magnitude. This concludes MLLS's superior performance is not because of differences in loss function used for distribution matching but due to differences in the granularity of the

predictions, caused by crude confusion matrix aggregation.

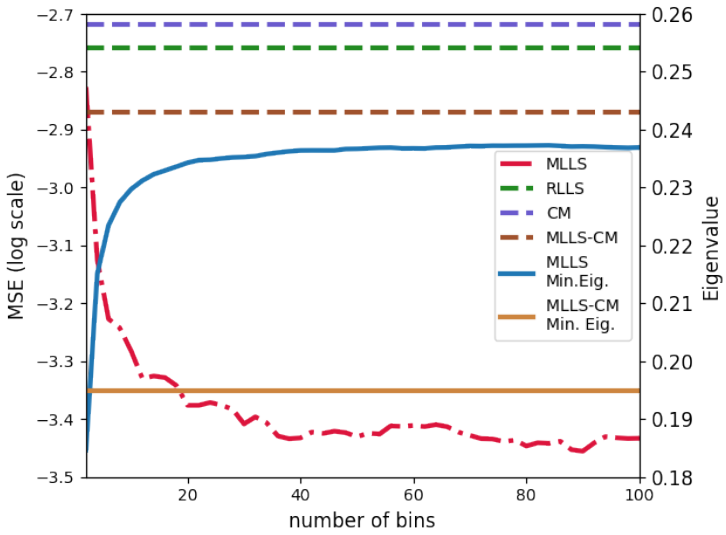


Figure 3.2: MSE (left-axis) with variation of minimum eigenvalue of the Hessian (right-axis) vs number of bins used for aggregation. With increase in number of bins, MSE decrease and the minimum eigenvalue increases.

Note that given a predictor f_1, we can partition our input space and produce another predictor f_2 that, for any data-point gives the expected output of f_1 on points belonging to that partition. If f_1 is calibrated, then f_2 will also be calibrated [141]. On synthetic data, we vary the granularity of calibration (for MLLS) by aggregating $p_S(y|x)$ over a variable number of equal-sized bins. With more bins, less information is lost due to calibration. Consequently, the minimum eigenvalue of the Hessian increases and the MSE decreases, supporting our theoretical bounds (Figure 3.2). We also verify that the confusion matrix calibration performs poorly (Figure 3.2). For MLLS-CM, the minimum eigenvalue of the Hessian is 0.195, significantly smaller than for the binned predictor for $\#\text{bin} \geq 4$. Thus, the poor performance of MLLS-CM is predicted by its looser upper bound per our analysis. Note that these experiments presume access to the true predictor $p_S(y|x)$ and thus the MSE strictly improves with the number of bins. In practice, with a fixed source dataset size, increasing the number of bins could lead to overfitting, worsening our calibration.

3.7 Conclusions

This chapter provides a unified framework relating techniques that use off-the-shelf predictors for label shift estimation. We argue that these methods all employ calibration, either explicitly or implicitly, differing only in the choice of calibration method and their optimization objective. Moreover, with our analysis we show that the choice of calibration method (and not the optimization objective for distribution matching) accounts for the advantage of MLLS with BCTS calibration over BBSE.

3.8 Proofs

Proof of Lemma 3.4.1. First we prove sufficiency. If there exists $w \neq w^*$ such that (3.2) holds, then we have $\sum_{y=1}^{k} p_S(z,y)(w_y - w_y^*) = 0$ for all $z \in \mathcal{Z}$. As $w - w^*$ is not the zero vector, $\{p_S(z,y), y = 1, ..., k\}$ are linearly dependent. Since $p_S(z,y) = p_S(y)p(z|y)$ and $p_S(y) > 0$ for all y (by assumption), we also have that $\{p(z|y), y = 1, ..., k\}$ are linearly dependent. By contradiction, we show that the linear independence is necessary.

To show necessity, assume $w_y^* = \frac{1}{kp_S(y)}$ for $y = 1, ..., k$. We know that w^* satisfies (3.2) by definition. If linear independence does not hold, then there exists a vector $v \in \mathbb{R}^k$ such that $v \neq 0$ and $\sum_{y=1}^{k} p_S(z,y)v_y = 0$ for all $z \in \mathcal{Z}$. Since the w^* we construct is not on the boundary of $\mathcal{W}$, we can scale v such that $w^* + \alpha v \in \mathcal{W}$ where $\alpha \geq 0$ and $v \neq 0$. Therefore, setting $w = w^* + \alpha v$ gives another solution for (3.2), which concludes the proof. □

Proof of Lemma 3.4.2.

$$\begin{aligned}
\mathbb{E}_T\left[\log f(x)^T w\right] &= \int p_T(x) \log[f(x)^T w] dx \\
&= \int\int p_T(x)p(z|x) \log[f(x)^T w] dx dz \\
&= \int\int p_T(x)p(z|x)\mathbb{1}\left\{f(x) = z\right\} \log[f(x)^T w] dx dz \\
&= \int\int p_T(x)p(z|x) \log[z^T w] dx dz \\
&= \int p_T(z) \log[z^T w] dz \\
&= \int p_T(z) \log\Big[\sum_{y=1}^{k} p_S(y|z)w\Big] dz\,,
\end{aligned}$$

where the final step uses the fact that f is calibrated.

□

Proof of Theorem 3.4.3. According to Lemma 3.4.2 we know that maximizing (3.4) is the same as maximizing (3.3) with $p(z|x) = \delta_{f(x)}$, thus also the same as minimizing the KL divergence between $p_T(z)$ and $p_w(z)$. Since $p_T(z) \equiv p_{w^*}(z)$ we know that w^* is a minimizer of the KL divergence such that the KL divergence is 0. We also have that $\mathrm{KL}(p_T(z), p_w(z)) = 0$ if and only if $p_T(z) \equiv p_w(z)$, so all maximizers of (3.4) should satisfy (3.2). According to Lemma 3.4.1, if the strict linear independence holds, then w^* is the unique solution of (3.2). Thus w^* is the unique maximizer of (3.4).

□

Proof of Proposition 3.4.4. We first show the equivalence of (1) and (2). If f is calibrated, we

have $p_S(f(x))f_y(x) = p_S(y)p(f(x)|y)$ for any x, y. Then for any vector $v \in \mathbb{R}^k$ we have

$$\sum_{y=1}^{k} v_y p(f(x)|y) = \sum_{y=1}^{k} \frac{v_y}{p_S(y)} p_S(y)p(f(x)|y) = \sum_{y=1}^{k} \frac{v_y}{p_S(y)} p_S(f(x))f_y(x) = p_S(f(x)) \sum_{y=1}^{k} \frac{v_y}{p_S(y)} f_y(x)\,. \tag{3.8}$$

On the other hand, we can have

$$\mathbb{E}_S\left[f(x)f(x)^T\right] = \int f(x)f(x)^T p_S(f(x))d(f(x))\,. \tag{3.9}$$

If $\{p(f(x)|y) \,:\, y = 1, \ldots, k\}$ are linearly dependent, then there exist $v \neq 0$ such that (3.8) is zero for any x. Consequently, there exists a non-zero vector u with $u_y = v_y/p_S(y)$ such that $u^T f(x) = 0$ for any x satisfying $p_S(f(x)) > 0$, which means $u^T \mathbb{E}_S\left[f(x)f(x)^T\right] u = 0$ and thus $\mathbb{E}_S\left[f(x)f(x)^T\right]$ is not invertible. On the other hand, if $\mathbb{E}_S\left[f(x)f(x)^T\right]$ is non-invertible, then there exist some $u \neq 0$ such that $u^T \mathbb{E}_S\left[f(x)f(x)^T\right] u = 0$. Further as $u^T \mathbb{E}_S\left[f(x)f(x)^T\right] u = \int u^T f(x)f(x)^T u \; p_S(x)dx = \int \left|f(x)^T u\right| p_S(x)dx$. As a result, the vector v with $v_y = p_S(y)u_y$ satisfies that (3.8) is zero for any x, which means $\{p(f(x)|y) \,:\, y = 1, \ldots, k\}$ are not strictly linearly independent.

Let C be the soft confusion matrix of f, then

$$\begin{aligned} C_{ij} = p_S(\widehat{y} = i, y = j) &= \int d(f(x))\; f_i(x)p(f(x)|y = j)p_S(y = j) \\ &= \int f_i(x)f_j(x)p_S(f(x))d(f(x))\,. \end{aligned}$$

Therefore, we have $C = \mathbb{E}_S\left[f(x)f(x)^T\right]$, which means (2) and (3) are equivalent.

□

Proof of Proposition 3.5.5. For any $v \in \mathbb{R}^k$, we have

$$v^T\left(-\nabla_w^2 \mathcal{L}(w, f)\right) v = \mathbb{E}_T\left[\frac{\left(v^T f(x)\right)^2}{\left(f(x)^T w\right)^2}\right] \in \left[\frac{1}{a^2}, \frac{1}{b^2}\right] \cdot v^T \mathbb{E}_T\left[f(x)f(x)^T\right] v\,,$$

where

$$a = \max_{x:p_S(x)>0} f(x)^T w \leq \frac{1}{p_{S,\min}}$$

and

$$b = \min_{x:p_S(x)>0} f(x)^T w \geq \tau$$

if f satisfies Condition 3.5.1 and $w \in \{w_f, w^*\}$. Therefore, we have

$$p_{S,\min}^2 \cdot \sigma_f \leq \sigma_{f,w} \leq \tau^{-2} \cdot \sigma_f$$

for $w \in \{w_f, w^*\}$.

□

3.8.1 Proof of Theorem 3.5.4

The gradient of the MLLS objective can be written as

$$\nabla_w \mathcal{L}(w, f) = \mathbb{E}_T \left[\frac{f(x)}{f(x)^T w} \right] , \tag{3.10}$$

and the Hessian is

$$\nabla_w^2 \mathcal{L}(w, f) = -\mathbb{E}_T \left[\frac{f(x) f(x)^T}{(f(x)^T w)^2} \right] . \tag{3.11}$$

We use $\lambda_{\min}(X)$ to denote the minimum eigenvalue of the matrix X.

Lemma 3.8.1 (Theorem 5.1.1 [137])**.** Let $X_1, X_2, \ldots, X_n$ be a finite sequence of identically distributed independent, random, symmetric matrices with common dimension k. Assume $0 \preceq X \preceq R \cdot I$ and $\mu_{\min} I \preceq \mathbb{E}[X] \preceq \mu_{\max} I$. With probability at least $1 - \delta$,

$$\lambda_{\min} \left(\frac{1}{n} \sum_{i=1}^{n} X_i \right) \geq \mu_{\min} - \sqrt{\frac{2R\mu_{\min} \log(\frac{k}{\delta})}{n}} . \tag{3.12}$$

Proof of Lemma 3.5.2. We present our proof in two steps. Step-1 is the non-probabilistic part, i.e., bounding the error $\|\widehat{w}_f - w_f\|$ in terms of the gradient difference $\|\nabla_w \mathcal{L}(w_f, f) - \nabla_w \mathcal{L}_m(w_f, f)\|$. This step uses Taylor's expansion upto second order terms for empirical log-likelihood around the true w^*. Step-2 involves deriving a concentration on the gradient difference using the Lipschitz property implied by Condition 3.5.1. Combining these two steps along with Lemma 3.15 concludes the proof. Now we detail each of these steps.

Step-1. We represent the empirical Negative Log-Likelihood (NLL) function with $\mathcal{L}_m$ by absorbing the negative sign to simplify notation. Using a Taylor expansion, we have

$$\mathcal{L}_m(\widehat{w}_f, f) = \mathcal{L}_m(w_f, f) + \langle \nabla_w \mathcal{L}_m(w_f, f), \widehat{w}_f - w_f \rangle + \frac{1}{2} (\widehat{w}_f - w_f)^T \nabla_w^2 \mathcal{L}_m(\widetilde{w}, f_c)(\widehat{w}_f - w_f) ,$$

where $\widetilde{w} \in [\widehat{w}_f, w_f]$. With the assumption $f^T w_f \geq \tau$, we have $\nabla_w^2 \mathcal{L}_m(\widetilde{w}, f) \geq \frac{\tau^2}{\min p_S(y)^2} \nabla_w^2 \mathcal{L}_m(w_f, f)$. Let $\kappa = \frac{\tau^2}{\min p_S(y)^2}$. Using this we get,

$$\mathcal{L}_m(\widehat{w}_f, f) \geq \mathcal{L}_m(w_f, f) + \langle \nabla_w \mathcal{L}_m(w_f, f), \widehat{w}_f - w_f \rangle + \frac{\kappa}{2} (\widehat{w}_f - w_f)^T \nabla_w^2 \mathcal{L}_m(w_f, f)(\widehat{w}_f - w_f)$$

$$\underbrace{\mathcal{L}_m(\widehat{w}_f, f) - \mathcal{L}_m(w_f, f)}_{\text{I}} - \langle \nabla_w \mathcal{L}_m(w_f, f), \widehat{w}_f - w_f \rangle \geq \frac{\kappa}{2} (\widehat{w}_f - w_f)^T \nabla_w^2 \mathcal{L}_m(w_f, f)(\widehat{w}_f - w_f) ,$$

where term-I is less than zero as $\widehat{w}_f$ is the minimizer of empirical NLL $\mathcal{L}_m(\widehat{w}_f, f)$. Ignoring term-I and re-arranging a few terms we get:

$$- \langle \nabla_w \mathcal{L}_m(w_f, f), \widehat{w}_f - w_f \rangle \geq \frac{\kappa}{2} (\widehat{w}_f - w_f)^T \nabla_w^2 \mathcal{L}_m(w_f, f)(\widehat{w}_f - w_f) ,$$

With first order optimality on w_f, $\langle \nabla_w \mathcal{L}(w_f, f), \widehat{w}_f - w_f \rangle \geq 0$. Plugging in this, we have,

$$\langle \nabla_w \mathcal{L}(w_f, f) - \nabla_w \mathcal{L}_m(w_f, f), \widehat{w}_f - w_f \rangle \geq \frac{\kappa}{2} (\widehat{w}_f - w_f)^T \nabla_w^2 \mathcal{L}_m(w_f, f)(\widehat{w}_f - w_f)\,,$$

Using Holder's inequality on the LHS we have,

$$\|\nabla_w \mathcal{L}(w_f, f) - \nabla_w \mathcal{L}_m(w_f, f)\| \, \|\widehat{w}_f - w_f\| \geq \frac{\kappa}{2} (\widehat{w}_f - w_f)^T \nabla_w^2 \mathcal{L}_m(w_f, f)(\widehat{w}_f - w_f)\,.$$

Let $\widehat{\sigma}_{f,w_f}$ be the minimum eigenvalue of $\nabla_w^2 \mathcal{L}_m(w^*, f_c)$. Using the fact that $(\widehat{w}_f - w_f)^T \nabla_w^2 \mathcal{L}_m(w_f, f)(\widehat{w}_f - w_f) \geq \widehat{\sigma}_{\min} \|\widehat{w}_f - w_f\|^2$, we get,

$$\|\nabla_w \mathcal{L}(w_f, f) - \nabla_w \mathcal{L}_m(w_f, f)\| \geq \frac{\kappa \widehat{\sigma}_{f,w_f}}{2} \|\widehat{w}_f - w_f\| \,. \tag{3.13}$$

Step-2. The empirical gradient is $\nabla_w \mathcal{L}_m(w_f, f) = \sum_{i=1}^m \frac{\nabla_w \mathcal{L}_1(x_i, w_f, f)}{m}$ where $\nabla \mathcal{L}_1(x_i, w_f, f) = \left[\frac{f_1(x_i)}{\langle f(x_i), w_f \rangle} \cdots \frac{f_l(x_i)}{\langle f(x_i), w_f \rangle} \cdots \frac{f_k(x_i)}{\langle f(x_i), w_f \rangle}\right]_{(k)}$. With the lower bound τ on $f^T w_f$, we can upper bound the gradient terms as

$$\|\nabla_w \mathcal{L}_1(x, w_f, f)\| \leq \frac{\|f\|}{\tau} \leq \frac{\|f\|_1}{\tau} \leq \frac{1}{\tau}\,.$$

As the gradient terms decompose and are independent, using Hoeffding's inequality we have with probability at least $1 - \frac{\delta}{2}$,

$$\|\nabla_w \mathcal{L}(w_f, f) - \nabla_w \mathcal{L}_m(w_f, f)\| \leq \frac{1}{2\tau} \sqrt{\frac{\log(4/\delta)}{m}}\,. \tag{3.14}$$

Let σ_{f,w_f} be the minimum eigenvalue of $\nabla_w^2 \mathcal{L}(w_f, f)$. Using lemma 3.8.1, with probability at least $1 - \frac{\delta}{2}$,

$$\frac{\widehat{\sigma}_{f,w_f}}{\sigma_{f,w_f}} \geq 1 - \tau \sqrt{\frac{\log(2k/\delta)}{m}}\,. \tag{3.15}$$

Plugging (3.14) and (3.15) in (3.13), and applying a union bound, we conclude that with probability at least $1 - \delta$,

$$\begin{aligned}
\|\widehat{w}_f - w_f\|_2 &\leq \frac{1}{\kappa\tau} \Big(\sigma_{f,w_f} - \sigma_{f,w_f} \tau \sqrt{\frac{\log(2k/\delta)}{m}}\Big)^{-1} \Big(\sqrt{\frac{\log(4/\delta)}{m}}\Big) \\
&\leq \frac{1}{\kappa\tau} \frac{1}{\sigma_{f,w_f}} \Big(1 + \tau \sqrt{\frac{\log(2k/\delta)}{m}}\Big) \sqrt{\frac{\log(4/\delta)}{m}}\,.
\end{aligned}$$

Neglecting the order m term and letting $c = \frac{1}{\kappa\tau}$, we have

$$\|\widehat{w}_f - w_f\| \leq \frac{c}{\sigma_{f,w_f}} \sqrt{\frac{\log(4/\delta)}{m}}\,.$$

□

Proof of Lemma 3.5.3. We present our proof in two steps. Note, all calculations are non-probabilistic. Step-1 involves bounding the error $\|w_f - w^*\|$ in terms of the gradient difference $\|\nabla_w \mathcal{L}(w^*, f_c) - \nabla_w \mathcal{L}(w^*, f)\|$. This step uses Taylor's expansion on $\mathcal{L}(w_f, f)$ upto the second orderth term for population log-likelihood around the true w^*. Step-2 involves deriving a bound on the gradient difference in terms of the difference $\|f - f_c\|$ using the Lipschitz property implied by Condition 3.5.1. Further, for a crude calibration choice of $f_c(x) = p_S(\cdot|x)$, the gradient difference can be bounded by miscalibration error. We now detail both of these steps.

Step-1. Similar to Lemma 3.5.2, we represent with $\mathcal{L}$ by absorbing the negative sign to simplify notation. Using the Taylor expansion, we have

$$\mathcal{L}(w_f, f) \geq \mathcal{L}(w^*, f) + \langle \nabla_w \mathcal{L}(w^*, f), w_f - w^* \rangle + \frac{1}{2}(w_f - w^*)^T \nabla_w^2 \mathcal{L}(\widetilde{w}, f)(w_f - w^*)\,,$$

where $\widetilde{w} \in [w_f, w^*]$. With the assumption $f^T w^* \geq \tau$, we have $\nabla_w^2 \mathcal{L}(\widetilde{w}, f) \geq \frac{\tau^2}{\min p_S(y)^2} \nabla_w^2 \mathcal{L}(w^*, f)$. Let $\kappa = \frac{\tau^2}{\min p_S(y)^2}$. Using this we get,

$$\mathcal{L}(w_f, f) \geq \mathcal{L}(w^*, f) + \langle \nabla_w \mathcal{L}(w^*, f), w_f - w^* \rangle + \frac{\kappa}{2}(w_f - w^*)^T \nabla_w^2 \mathcal{L}(w^*, f)(w_f - w^*)$$

$$\underbrace{\mathcal{L}(w_f, f) - \mathcal{L}(w^*, f)}_{\text{I}} \geq \langle \nabla_w \mathcal{L}(w_f, f), w_f - w^* \rangle + \frac{\kappa}{2}(w_f - w^*)^T \nabla_w^2 \mathcal{L}(w^*, f)(w_f - w^*)\,,$$

where term-I is less than zero as w_f is the minimizer of NLL $\mathcal{L}(w, f)$. Ignoring that term and re-arranging a few terms we get

$$-\langle \nabla_w \mathcal{L}(w^*, f), w_f - w^* \rangle \geq \frac{\kappa}{2}(w_f - w^*)^T \nabla_w^2 \mathcal{L}(w^*, f)(w_f - w^*)\,.$$

With first order optimality on w^*, $\langle \nabla_w \mathcal{L}(w^*, f_c), w_f - w^* \rangle \geq 0$. Using this we have:

$$\langle \nabla_w \mathcal{L}(w^*, f_c), w_f - w^* \rangle - \langle \nabla_w \mathcal{L}(w^*, f), w_f - w^* \rangle \geq \frac{\kappa}{2}(w_f - w^*)^T \nabla_w^2 \mathcal{L}(w^*, f)(w_f - w^*)\,,$$

$$\langle \nabla_w \mathcal{L}(w^*, f_c) - \nabla_w \mathcal{L}(w^*, f), w_f - w^* \rangle \geq \frac{\kappa}{2}(w_f - w^*)^T \nabla_w^2 \mathcal{L}(w^*, f)(w_f - w^*)\,.$$

As before, let $\sigma_{f,w}$ be the minimum eigenvalue of $\nabla_w^2 \mathcal{L}(w^*, f)$. Using the fact that $(w_f - w^*)^T \nabla_w^2 \mathcal{L}(w^*, f)(w_f - w^*) \geq \sigma_{f,w} \|w_f - w^*\|^2$, we get

$$\langle \nabla_w \mathcal{L}(w^*, f_c) - \nabla_w \mathcal{L}(w^*, f), w_f - w^* \rangle \geq \frac{\kappa \sigma_{f,w}}{2} \|w_f - w^*\|^2\,.$$

Using Holder's inequality on the LHS and re-arranging terms gives

$$\|\nabla_w \mathcal{L}(w^*, f_c) - \nabla_w \mathcal{L}(w^*, f)\| \geq \frac{\kappa \sigma_{f,w}}{2} \|w_f - w^*\|\,. \tag{3.16}$$

Step-2. By lower bound assumptions $f_c^T w^* \geq \tau$ and $f^T w^* \geq \tau$, we have

$$\|\nabla_w \mathcal{L}(w^*, f_c) - \nabla \mathcal{L}(w^*, f)\| \leq \mathbb{E}_T\left[\|\nabla \mathcal{L}_1(x, w^*, f_c) - \nabla \mathcal{L}_1(x, w^*, f)\|\right] \leq \frac{1}{\tau^2}\mathbb{E}_T\left[\|f_c(x) - f(x)\|\right]\,, \tag{3.17}$$

where the first inequality is implied by Jensen's inequality and the second is implied by the Lipschitz property of the gradient. Further, we have

$$\begin{aligned}\mathbb{E}_T\left[\|f_c(x) - f(x)\|\right] &= \mathbb{E}_S\left[\frac{p_T(x)}{p_S(x)}\|f_c(x) - f(x)\|\right] \\ &\leq \mathbb{E}_S\left[\max_y \frac{p_T(y)}{p_S(y)}\|f_c(x) - f(x)\|\right] \\ &\leq \max_y \frac{p_T(y)}{p_S(y)}\mathbb{E}_S\left[\|f_c(x) - f(x)\|\right]. \end{aligned} \tag{3.18}$$

Combining equations (3.16), (3.17), and (3.18), we have

$$\|w_f - w^*\| \leq \frac{2}{\kappa\sigma_{f,w}\tau^2}\max_y \frac{p_T(y)}{p_S(y)}\mathbb{E}_S\left[\|f_c(x) - f(x)\|\right]. \tag{3.19}$$

Further, if we set $f_c(x) = p_S(\cdot|f(x))$, which is a calibrated predictor according to Proposition 3.4.5, we can bound the error on the RHS in terms of the calibration error of f. Moreover, in the label shift estimation problem, we have the assumption that $p_S(y) \geq c > 0$ for all y. Hence, we have $\max_y p_T(y)/p_S(y) \leq 1/c$. Using Jensen's inequality, we get

$$\mathbb{E}_S\|f_c(x) - f(x)\| \leq \left(\mathbb{E}_S\|f_c(x) - f(x)\|^2\right)^{\frac{1}{2}} = \mathcal{E}(f). \tag{3.20}$$

Plugging (3.20) back in (3.19),we get the required upper bound. □

Chapter 4

Learning from Positive and Unlabeled Data

4.1 Overview

When deploying k-way classifiers in the wild, what can we do when confronted with data from a previously unseen class ($y = k + 1$)? Theory dictates that learning under distribution shift is impossible absent assumptions. And yet people appear to exhibit this capability routinely. Faced with new surprising symptoms, doctors can recognize the presence of a previously unseen ailment and attempt to estimate its prevalence. Similarly, naturalists can discover new species, estimate their range and population, and recognize them reliably going forward.

To begin making this problem tractable, we follow the label shift assumption [94, 116, 130], i.e., that while the class balance $p(y)$ can change, the class conditional distributions $p(x|y)$ do not. Moreover, we begin by focusing on the base case, where only one class has been seen previously, i.e., $k = 1$. Here, we possess (labeled) positive data from the source distribution, and (unlabeled) data from the target distribution, consisting of both positive and negative instances. This problem has been studied in the literature as *learning from positive and unlabeled data* [28, 88] and has typically been broken down into two subtasks: (i) Mixture Proportion Estimation (MPE) where we estimate α, the fraction of positives among the unlabeled examples; and (ii) classification where this estimate is incorporated into a scheme for learning a Positive-versus-Negative (PvN) binary classifier.

Traditionally, PU-learning have been motivated by settings involving large databases where unlabeled examples are abundant and a small fraction of the total positives have been extracted. For example, medical records might be annotated indicating the presence of certain diagnoses, but the unmarked passages are not necessarily negative. This setup has also been motivated by protein and gene identification [39]. Databases in molecular biology often contain lists of molecules known to exhibit some characteristic of interest. However, many other molecules may exhibit the desired characteristic, even if this remains unknown to science.

Many methods have been proposed for both MPE [9, 35, 39, 64, 65, 114, 120] and classification [33, 34, 72]. However, classical MPE methods break down in high-dimensional settings [114] or yield estimators whose accuracy depends on restrictive conditions [35, 120]. On the other hand, most recent proposals either lack theoretical coherence, rely on heroic assumptions, or depend precariously on tuning hyperparameters that are, by the very problem setting,

untunable to the best of our knowledge. For PU learning, Elkan and Noto [39] suggest training a classifier to distinguish positive from unlabeled data followed by a rescaling procedure. Du Plessis et al. [33] suggest an unbiased risk estimation framework for PU learning. However, these methods fail badly when applied with model classes capable of overfitting and thus implementations on high-dimensional datasets rely on extensive hyperparameter tuning and additional ad-hoc heuristics that do not transport effectively across datasets.

In this chapter, we propose (i) Best Bin Estimation (BBE), an effective technique for MPE that produces consistent estimates $\widehat{\alpha}$ under mild assumptions and admits finite-sample statistical guarantees ; and (ii) classification with the Conditional Value under Optimism (CVuO) objective, which discards the hardest-to-fit $\hat{\alpha}$ fraction of examples on each training epoch, removing the incentive to overfit to the unlabeled positive examples. Both methods are simple to implement, compatible with arbitrary hypothesis classes (including deep networks), and outperform existing methods in our experimental evaluation. Finally, we combine the two in an iterated Transform-Estimate-Discard (TED)^n framework that substantially improves both estimate MSE and classifier error.

We build on label shift methods [4, 7, 47, 94], that leverage black-box classifiers to reduce dimensionality, estimating the target label distribution as a functional of source and target push-forward distributions. While label shift methods rely on classifiers trained to separate previously seen classes, BBE is able to exploit a Positive-versus-Unlabeled (PvU) target classifier, which gives each input a score indicating how likely it is to be a positive sample. In particular, BBE identifies a threshold such that by estimating the ratio between the fractions of positive and unlabeled points receiving scores above the threshold, we obtain of the the mixture proportion α.

BBE works because in practice, for many datasets, PvU classifiers, even when uncalibrated, produce outputs with near monotonic calibration diagrams. Higher scores correspond to a higher proportion of positives, and when the positive data contains a separable sub-domain, i.e. a region of the input space where only the positive distribution has support, classifiers often exhibit a threshold above which the *top bin* contains mostly positive examples. We show that the existence of a (nearly) pure top bin is sufficient for BBE to produce a (nearly) consistent estimate $\hat{\alpha}$, whose finite sample convergence rates depend on the fraction of examples in the bin and whose bias depends on the *purity* of the bin. Crucially, we can estimate the threshold from data.We conduct experiments to establish the outperformance of BBE, CVuO, and (TED)^n over the previous methods.

4.2 Related Work

Research on MPE and PU learning date to [28, 30, 88] (see review by [10]). Elkan and Noto [39] first proposed to leverage a PvU classifier to estimate the mixture proportion. Du Plessis and Sugiyama [36] propose a different method for estimating the mixture coefficient based on Pearson divergence minimization. While they do not require a PvU classifier, they suffer the same shortcoming. Both methods require that the positive and negative examples have disjoint support. Our requirements are considerably milder. [16] observe that without assumptions on the underlying positive and negative distributions, the mixture proportion is not identifiable. Furthermore, [16] provide an *irreducibility* condition that identifies α and propose an estimator that

converges to the true α. While their estimator can converge arbitrarily slowly, Scott [120] showed faster convergence ($\mathcal{O}(1/\sqrt{n})$) under stronger conditions. Unfortunately, despite its appealing theoretical properties Blanchard et al. [16]'s estimator is computationally infeasible. Building on Blanchard et al. [16], Sanderson and Scott [117] and Scott [120] proposed estimating the mixture proportion from a ROC curve constructed for the PvU classifier. However, when the PvU classifier is not perfect, these methods are not clearly understood. Ramaswamy et al. [114] proposed the first computationally feasible algorithm for mixture proportion estimation with convergence guarantees to the true proportion. Their method KM, requires embedding distributions onto an RKHS. However, their estimator underperforms on high dimensional datasets and scales poorly with large datasets. Bekker and Davis [9] proposed TIcE, hoping to identify a positive subdomain in the input space using decision tree induction. This method also underperforms in high-dimensional settings.

In the most similar works, [65] and Ivanov [64] explore dimensionality reduction using a PvU classifier. Both methods estimate α through a procedure operating on the PvU classifier's output. However, neither methods has provided theoretical backing [64] concede that their method often fails and returns a zero estimate, requiring that they fall back to a different estimator. Moreover while both papers state that their method require the Bayes-optimal PvU classifier to identify α in the transformed space, we prove that even when hypothesis class is well specified for PvN learning, PvU training can fail to recover Bayes-optimal scoring function. By contrast, our estimator BBE is theoretically coherent under mild conditions and outperforms both of these methods empirically.

Given α, Elkan and Noto [39] propose a transformation via Bayes rule to obtain the PvN classifier. They also propose a weighted objective, with weights given by the PvU classifier. Other propose unbiased risk estimators [33, 34] which require the mixture proportion α. Du Plessis et al. [34] proposed an unbiased estimator with non-convex loss functions satisfying a specific symmetric condition, and subsequently Du Plessis et al. [33] generalized it to convex loss functions (denoted uPU in our experiments). in our experiments. Noting the problem of overfitting in modern overparameterized models, Kiryo et al. [72] propose a regularized extension that clips the loss on unlabeled data to zero. This is considered the current state-of-the-art in PU literature (denoted nnPU in our experiments). More recently, Ivanov [64] proposed Dedpul, which finetunes the PvU classifiers using several heuristics, Bayes rule, and Expectation Maximization (EM). Since their method only applies a post-processing procedure, they rely on a good domain discriminator classifier in the first place and several hyperparameters for their heuristics. Several classical methods attempt to learn weights that identify reliable negative examples [87, 92, 95, 96, 158]. While our loss may be similar in spirit, these earlier methods have not been successful with modern deep learning models.

4.3 Problem Setup

By $\|\cdot\|$ and $\langle\cdot,\cdot\rangle$, we denote the Euclidean norm and inner product, respectively. For a vector $v \in \mathbb{R}^d$, we use v_j to denote its j^{th} entry, and for an event E, we let $\mathbb{I}[E]$ denote the binary indicator of the event. By $|A|$, we denote the cardinality of set A. Let $\mathcal{X} \in \mathbb{R}^d$ be the input space and $\mathcal{Y} = \{-1, +1\}$ be the output space. Let $\mathrm{P} : \mathcal{X} \times \mathcal{Y} \to [0, 1]$ be the underlying joint

Algorithm 1 Best Bin Estimation (BBE)

1: INPUT: Validation positive (X_p) and unlabeled (X_u) samples. Blackbox model classifier $\hat{f} : \mathcal{X} \to [0, 1]$. Hyperparameter $0 < \delta < 1$, $\gamma > 1$.
2: $Z_p, Z_u = f(X_p), f(X_u)$.
3: $\hat{q}_u(z), \hat{q}_p(z) = \frac{\sum_{z_i \in Z_p} \mathbb{I}[z_i \geq z]}{n_p}, \frac{\sum_{z_i \in Z_u} \mathbb{I}[z_i \geq z]}{n_u}$ for all $z \in [0, 1]$.
4: Estimate $\hat{c} := \arg\min_{c \in [0,1]} \left(\frac{\hat{q}_u(c)}{\hat{q}_p(c)} + \frac{\gamma}{\hat{q}_p(c)} \left(\sqrt{\frac{\log(4/\delta)}{2n_u}} + \sqrt{\frac{\log(4/\delta)}{2n_p}} \right) \right)$.
5: OUTPUT: $\hat{\alpha} := \frac{\hat{q}_u(\hat{c})}{\hat{q}_p(\hat{c})}$

distribution and let p denote its corresponding density.

Let P_p and P_n be the class-conditional distributions for positive and negative class and $p_p(x) = p(x|y = +1)$ and $p_n(x) = p(x|y = -1)$ be the corresponding class-conditional densities. P_u denotes the distribution of the unlabeled data and p_u denotes its density. Let $\alpha \in [0, 1]$ be the fraction of positives among the unlabeled population, i.e., $\mathrm{P}_u = \alpha \mathrm{P}_p + (1 - \alpha)\mathrm{P}_n$. When learning from positive and unlabeled data, we obtain i.i.d. samples from the positive (class-conditional) distribution, which we denote as $X_p = \{x_1, x_2, \ldots, x_{n_p}\} \sim \mathrm{P}_p^{n_p}$ and i.i.d samples from unlabeled distribution as $X_u = \{x_{n_p+1}, x_{n_p+2}, \ldots, x_{n_p+n_u}\} \sim \mathrm{P}_u^{n_u}$.

MPE is the problem of estimating α. Absent assumptions on P_p, P_n and P_u, the mixture proportion α is not identifiable [16]. Indeed, if $\mathrm{P}_u = \alpha \mathrm{P}_p + (1 - \alpha)\mathrm{P}_n$, then any alternate decomposition of the form $\mathrm{P}_u = (\alpha - \beta)\mathrm{P}_p + (1 - \alpha + \beta)\mathrm{P}'_n$, for $\beta \in [0, \alpha)$ and $\mathrm{P}'_n = (1 - \alpha + \beta)^{-1}(\beta \mathrm{P}_p + (1 - \alpha)\mathrm{P}_n)$, is also valid. Since we do not observe samples from the distribution P_n, the parameter α is not identifiable. Blanchard et al. [16] formulate an *irreducibility* condition under which α is identifiable. Intuitively, the condition restricts P_n to ensure that it can not be a (non-trivial) mixture of P_p and any other distribution. While this irreducibility condition makes α identifiable, in the worst-case, the parameter α can be difficult to estimate and any estimator must suffer an arbitrarily slow rate of convergence [16]. In this work, we show mild conditions on the PvU classifier that make α identifiable and allows us to derive finite-sample convergence guarantees.

The classification task in PU learning is to learn a classifier $f : \mathcal{X} \to [0, 1]$ to approximate $p(y = +1|x)$. We assume that we are given a loss function $\ell : [0, 1] \times \mathcal{Y} \to \mathbb{R}$, such that $\ell(z, y)$ is the loss incurred by predicting z when the true label is y. For a classifier f and a sampled set $X = \{x_1, x_2, \ldots, x_n\}$, we let $\hat{L}^+(f; X) = \sum_{i=1}^n \ell(f(x_i), +1)/n$ denote the loss when predicting the samples as positive and $\hat{L}^-(f; X) = \sum_{i=1}^n \ell(f(x_i), -1)/n$ the loss when predicting the samples as negative. For a sample set X each with true label y, we define 0-1 error as $\hat{\mathcal{E}}^y(f; X) = \sum_{i=1}^n \mathbb{I}\left[y(f(x_i) - t) \leq 0\right]/n$ for some predefined threshold t . Unless stated otherwise, the threshold is assumed to be 0.5.

4.4 Mixture Proportion Estimation

In this section, we propose BBE, a method that leverages a blackbox classifier f to perform MPE and establish convergence guarantees. To begin, we assume access to a fixed classifier f. For intuition, you may think of f as a PvU classifer trained to classify some portion of the

positive v.s. unlabeled examples. In next sections, we discuss other ways to obtain a suitable classifier from PU data.

We now introduce some additional notation. Assume f transforms an input $x \in \mathcal{X}$ to $z \in [0,1]$, i.e., $z = f(x)$. For given probability density function p and a classifier f, define a function $q(z) = \int_{A_z} p(x)dx$, where $A_z = \{x \in \mathcal{X} : f(x) \geq z\}$ for all $z \in [0,1]$. Intuitively, $q(z)$ captures the cumulative density of points in a top bin, the proportion of input domain that is assigned a value larger than z by the classifier f in the transformed space. We now define an empirical estimator $\hat{w}q(z)$ given a set $X = \{x_1, x_2, \ldots, x_n\}$ sampled iid from $p(x)$. Let $Z = f(X)$. Define $\hat{q}(z) = \sum_{i=1}^{n} \mathbb{I}\left[z_i \geq z\right]/n$. For each pdf p_p, p_n and p_u, we define q_p, q_n and q_u respectively.

Without any assumptions on the underlying distribution and the classifier f, we aim to estimate $\alpha^* = \min_{c \in [0,1]} q_u(c)/q_p(c)$. We now explain our procedure. First, given a held-out dataset of positive (X_p) and unlabeled examples (X_u), we push all examples through the classifier f to obtain one-dimensional outputs $Z_p = f(X_p)$ and $Z_u = f(X_u)$. Next, with Z_p and Z_u, we estimate $\hat{q}_p$ and $\hat{q}_u$. Finally, we return the ratio $\hat{q}_u(\hat{c})/\hat{q}_p(\hat{c})$ at $\hat{c}$ that minimizes the upper confidence bound at a pre-specified level δ and a fixed parameter $\gamma > 1$. Our method is summarized in Algorithm 1. For theoretical guarantees, we multiply the confidence bound term with $\gamma > 1$ although $\gamma = 1$ works well in our experiments. We now show that the proposed BBE estimator comes with the following guarantee:

Theorem 4.4.1. Define $\alpha^* = \min_{c \in [0,1]} q_u(c)/q_p(c)$, $\mathcal{E}_n = \sqrt{\frac{\log(4/\delta)}{2n_p}} + \sqrt{\frac{\log(4/\delta)}{2n_u}}$. Define

$$\mathcal{C} = \left\{ c \in [0,1] \,:\, \frac{q_u(c)}{q_p(c)} \leq 1 \,, q_p(c) \geq 2\sqrt{\frac{\log 4/\delta}{2n_p}} \right\} \,,$$

$$c_0 = \arg\min_{c \in \mathcal{C}} \frac{q_u(c)}{q_p(c)} + \frac{2(\gamma+1)}{q_p(c)}\mathcal{E}_n \,,$$

and $\alpha_0 = q_u(c_0)/q_p(c_0)$. If $\mathcal{C} = \emptyset$ or $\alpha + \frac{2(\gamma+1)}{q_p(c_0)}\mathcal{E}_n \geq 1$ we take the trivial guarantee $\hat{\alpha} \in [0,1]$. Otherwise, we have

$$\hat{\alpha} \leq \alpha^* + (\alpha_0 - \alpha^*) + \frac{2(\gamma+1)}{q_p(c_0)}\mathcal{E}_n$$

and

$$\hat{\alpha} \geq \alpha^* - \frac{\alpha_0 - \alpha^*}{\gamma - 1} - \frac{2(\gamma+1)}{(\gamma-1)q_p(c_0)}\mathcal{E}_n \,.$$

We now present the condition under which $\hat{\alpha}$ is a consistent estimator of α:

Corollary 4.4.2. If there exist $c_f \in (0,1)$ such that $q_p(c_f) > 0$ and $q_n(c_f) = 0$, then $|\hat{\alpha} - \alpha| = \mathcal{O}_p\left(\min(n_p, n_u)^{-1/2}\right)$.

The condition in Corollary 4.4.2 means that the bin of samples x whose output $f(x) \in [c_f, 1]$ only contains positive samples. We refer to this condition as the *pure positive bin* property. In the more general case, $\hat{\alpha}$ is a consistent estimator of α^* and our bound in Theorem 4.4.1 captures the tradeoff due to the proportion of negative examples in the top bin (bias $|\alpha_0 - \alpha^*|$) versus the proportion of positives in the top bin (variance $q_p(c_0)^{-1}$).

Algorithm 2 PU learning with Conditional Value under Optimism (CVuO) objective

INPUT: Labeled positive training data (X_p) and unlabeled training samples (X_u). Mixture proportion estimate α.
Initialize a training model f_θ and an stochastic optimization algorithm $\mathcal{A}$.
$X_n := X_u$.
while training error $\hat{\mathcal{E}}^+(f_\theta; X_p) + \hat{\mathcal{E}}^-(f_\theta; X_n)$ is not converged **do**
    Rank samples $x_u \in X_u$ according to their loss values $\ell(f_\theta(x_u), -1)$.
    $X_n := X_{u,1-\alpha}$ where $X_{u,1-\alpha}$ denote the lowest ranked $1-\alpha$ fraction of samples.
    Shuffle (X_p, X_n) into B mini-batches. With (X_p^i, X_n^i) we denote i-th mini-batch.
    for $i = 1$ to B **do**
        Set the gradient $\nabla_\theta \left[\hat{L}^+(f_\theta; X_p^i) + \hat{L}^-(f_\theta; X_n^i)\right]$ and update θ with algorithm $\mathcal{A}$.
    end for
end while
OUTPUT: Trained classifier f_θ

4.5 Classification

Given positive and unlabeled data, we hope not only to identify α, but also to obtain a classifier that distinguishes effectively between positive and negative samples. In supervised learning with separable data (e.g., cleanly labeled image data), overparameterized models generalize well even after achieving near-zero training error. However, with PvU training over-parameterized models can memorize the unlabeled positives, assigning them confidently to the negative class, which can severely hurt generalization on PN data [157]. Moreover, while unbiased losses exist that estimate the PvN loss given PU data and the mixture proportion α, this unbiasedness only holds before the loss is optimized, and becomes ineffective with powerful deep learning models capable of memorization.

A variety of heuristics, including ad-hoc early stopping criteria, have been explored [64], where training proceeds until the loss on unseen PU data ceases to decrease. However, this approach leads to under-fitting. On the other hand by regularizing the loss function, nnPU Kiryo et al. [72] mitigates overfitting issues due to memorization. However, we observe that nnPU still leaves a substantial accuracy gap when compared to a model trained just on the positive and negative (from the unlabeled) data. This leads us to ask the following question: *can we improve performance over nnPU of a model just trained with PU data and make this gap smaller?* In an ideal scenario, if we could identify and remove all the positive points from the unlabeled data during training then we can hope to achieve improved performance over nnPU. Indeed, in practice, we observe that in the initial stages of PvU training, the model assigns relatively higher scores to positives than to negatives in the unlabeled data .

Inspired by this observation, we propose CVuO, a simple yet effective objective for PU learning. Below, we present our method assuming an access to the true MPE. Later (ref. Sec. 4.6), we combine BBE with CVuO optimization, yielding (TED)^n, an alternating optimization that significantly improves both the BBE estimates and the PvU classifier.

Given a training set of positives X_p and unlabeled X_u and the mixture proportion α, we begin by ranking the unlabeled data according the predicted probability (of being positive) by our

classifier. Then, in every epoch of training, we create a (temporary) set of provisionally negative samples X_n by removing α fraction of the unlabeled samples currently scored as most positive. Next, we update our classifier by minimize the loss on the positives X_p and provisional negatives X_n by treating them as negatives. We repeat this procedure until the training error on X_p and X_n converges. Likewise nnPU, note that this procedure does not need early stopping. Summary in Algorithm 2. The CVuO algorithm can be viewed as an alternating optimization algorithm that jointly optimizes a classifier f and a weighting function w on the unlabeled distribution:

$$\min_{f\in\mathcal{F},w} \int dx p_p(x) l(f(x),1) + \frac{1}{1-\alpha}\int dx p_u(x) w(x) l(f(x),0)\,,$$
$$\text{s.t. } w:\mathcal{X}\mapsto[0,1]\,,\ \int dx p_u(x) w(x) = 1-\alpha\,. \tag{4.1}$$

We now justify our loss function in the scenario when the support of positives and negatives is separable. We leave theoretic investigation on non-separable distributions for future work. We assume that the true alpha α is known and we have access to populations of positive and unlabeled data. We also assume that their exists a separator $f^*:\mathcal{X}\mapsto\{0,1\}$ that can perfectly separate the positive and negative distribution, i.e., $\int dx p_p(x)\mathbb{I}\left[f^*(x)\neq 1\right] + \int dx p_n(x)\mathbb{I}\left[f^*(x)\neq 0\right] = 0$. The following proposition shows that minimizing the objective (4.1) on separable positive and negative distributions gives a perfect classifier.

Proposition 4.5.1. For $\alpha\in(0,1)$, if there exists a classifier $f^*\in\mathcal{F}$ that can perfectly separate the positive and negative distributions, optimizing objective (4.1) with 0-1 loss leads to a classifier f that achieves 0 classification error on the unlabeled distribution.

4.6 Combining MPE and Classification

We are now ready to present our algorithm Transfer, Estimate and Discard (TED)^n that combines BBE and CVuO objective.

First, we observe the interaction between BBE and CVuO objective. If we have an accurate mixture proportion estimate, then it leads to improved classifier, in particular, we reject accurate number of prospective positive samples from unlabeled. Consequently, updating the classifier to minimize loss on positive versus retained unlabeled improves purity of top bin. This leads to an obvious alternating procedure where at each epoch, we first use BBE to estimate $\hat{\alpha}$ and then update the classifier with CVuO objective with $\hat{\alpha}$ as input. We repeat this until training error has not converged. Our method is summarized in Algorithm 3.

Note that we need to warm start with PvU (positive versus negative) training, since in the initial stages mixture proportion estimate is often close to 1 rejecting all the unlabeled examples. However, in experiments, we find that our procedure is not sensitive to the choice of number of warm start epochs and in a few cases with large datasets, we can even get away without warm start (i.e., $W=0$) without hurting the performance.

Finally, we discuss an important distinction with Dedpul which is also an alternating procedure. While in our algorithm, after updating mixture proportion estimate we retrain the classifier, Dedpul fixes the classifier, obtains output probabilities and then iteratively updates the mixture

Algorithm 3 Transform-Estimate-Discard (TED)^n

1: INPUT: Positive data (X_p) and unlabeled samples (X_u). Hyperparameter W, δ.
2: Initialize a training model f_θ and an stochastic optimization algorithm $\mathcal{A}$.
3: Randomly split positive and unlabeled data into training X_p^1, X_u^1 and hold-out set (X_p^2, X_u^2).
4: $X_n^1 := X_u^1$.
{// Warm start with domain discrimination training}
5: **for** $i = 1$ to W **do**
6: Shuffle (X_p^1, X_n^1) into B mini-batches. With $(X_p^{1^i}, X_n^{1^i})$ we denote i-th mini-batch.
7: **for** $i = 1$ to B **do**
8: Set the gradient $\nabla_\theta \left[\hat{L}^+(f_\theta; X_p^{1^i}) + \hat{L}^-(f_\theta; X_n^{1^i})\right]$ and update θ with algorithm $\mathcal{A}$.
9: **end for**
10: **end for**
11: **while** training error $\hat{\mathcal{E}}^+(f_\theta; X_p^1) + \hat{\mathcal{E}}^-(f_\theta; X_n^1)$ is not converged **do**
12: Estimate $\hat{w}\alpha$ using Algorithm 1 with (X_p^2, X_u^2) and f_θ as input.
13: Rank samples $x_u \in X_u^1$ according to their loss values $l(f_\theta(x_u), -1)$.
14: $X_n^1 := X_{u,1-\hat{\alpha}}^1$ where $X_{u,1-\hat{\alpha}}^1$ denote the lowest ranked $1 - \hat{\alpha}$ fraction of samples.
15: Train model f_θ for one epoch on (X_p^1, X_n^1) as in Lines 4-7.
16: **end while**
17: OUTPUT: Trained classifier f_θ

proportion estimate (prior) and output probabilities (posterior). Dedpul doesn't re-train the classifier.

4.7 Experiments

Having presented our MPE and classification algorithms, we now compare their performance with other methods empirically.

Datasets and Evaluation We simulate PU tasks on CIFAR-10 [74], MNIST [86], and IMDb sentiment analysis [99] datasets. We consider binarized versions of CIFAR-10 and MNIST. On CIFAR-10 dataset, we consider two classification problems: (i) binarized CIFAR, i.e., first 5 classes vs rest; (ii) Dog vs Cat in CIFAR. Similarly on MNIST, we consider: (i) binarized MNIST, i.e., digits 0-4 vs 5-9; (ii) MNIST17, i.e., digit 1 vs 7. IMDb dataset is binary. For MPE, we use a held out PU validation set. To evaluate PU classifiers, we calculate accuracy on held out positive versus negative dataset. For baselines that suffer from issues due to overfitting on unlabeled data, we report results with an *oracle early stopping* criterion. We report the accuracy averaged over 10 iterations of the best performing model. With nnPU and (TED)^n, we report average accuracy over 10 iterations of the final model.

Architectures For CIFAR datasets, we consider (fully connected) multilayer perceptrons (MLPs) with ReLU activations, all convolution nets [129], and ResNet18 [58]. For MNIST, we consider multilayer perceptrons (MLPs) with ReLU activations For the IMDb dataset, we fine-tune an off-the-shelf uncased BERT model [32, 147]. We did not tune hyperparameters or the optimization algorithm—instead we use the same benchmarked hyperparameters and optimization algorithm for each dataset. For our method, we use cross-entropy loss. For uPU and nnPU,

Dataset	Model	$(\text{TED})^n$	BBE*	Dedpul*	AlphaMax*	EN	KM2	TiCE
Binarized CIFAR	ResNet	**0.018**	0.072	0.075	0.125	0.175		
	All Conv	0.041	**0.038**	0.046	0.09	0.23	0.181	0.251
	FCN	0.184	0.175	**0.151**	0.3	0.355		
CIFAR Dog vs Cat	ResNet	**0.074**	0.120	0.113	0.17	0.205	0.11	0.203
	All Conv	**0.073**	0.093	0.098	0.19	0.274		
Binarized MNIST	FCN	**0.021**	0.028	0.027	0.09	0.067	0.102	0.247
MNIST17	FCN	**0.003**	0.008	0.006	0.075	0.065	0.03	0.117
IMDb	BERT	**0.008**	0.011	0.016	0.07	0.12	-	-

Table 4.1: Absolute estimation error when α is 0.5. "*" denote oracle early stopping as defined in Sec. 3.6. Results reported by aggregating absolute error over 10 epochs and 3 seeds.

we use Adam [71] with sigmoid loss.

Mixture Proportion Estimation First, we discuss results for MPE (Table 4.1). We compare our method with KM2, TiCE, Dedpul, AlphaMax and EN. For KM2 and TiCE, datasets are reduced to 100 dimensions with PCA. We use existing implementation for other methods. For our method, Dedpul and Alphamax, we use the same PvU classifier as input. On all datasets, classifier based estimators outperform KM and TiCE. With the same blackbox classifier, BBE performs similar or better than best alternate(s).

Classification with known MPE Now, we discuss results for classification with known α. We compare our method with uPU, nnPU, Dedpul and PvU training. Although, we solve both MPE and classification, some comparison methods do not. Ergo, we compare our classification algorithm with known MPE (Algorithm 2).

To begin, first we note that nnPU and PvU training with CVuO doesn't need early stopping. For all other methods, we report the best performance dictated by the aforementioned oracle stopping criterion. On all datasets, PvU training with CVuO leads to improved classification performance when compared with alternate approaches (Table 4.2).

Classification with unknown MPE Finally, we evaluate $(\text{TED})^n$, our alternating procedure for MPE and PU learning. Across many tasks, we observe substantial improvements over existing methods. Note that these improvements often are over an oracle early stopping baselines highlighting significance of our procedure.

4.8 Conclusions

In this chapter, we proposed two practical algorithms, BBE (for MPE) and CVuO optimization (for PU learning). Our methods out perform others empirically and BBE's mixture proportion estimates leverage black box classifiers to produce (nearly) consistent estimates with finite sample convergence guarantees whenever we possess a classifier with a (nearly) pure top bin. Moreover, $(\text{TED})^n$ combines our procedures in an iterative fashion, achieving further gains.

Dataset	Model	(TED)^n (unknown α)	CVuO (known α)	PvU* (known α)	Dedpul* (unknown α)	nnPU (known α)	uPU* (known α)
Binarized CIFAR	ResNet	**82.7**	82.6	78.3	78.4	76.8	75.8
	All Conv	76.8	**77.1**	74.1	76.9	72.1	71.3
	FCN	63.2	**65.9**	61.4	62.5	63.9	64.8
CIFAR Dog vs Cat	ResNet	**76.1**	74.0	71.6	70.9	72.6	69.5
	All Conv	**72.2**	71.0	70.1	70.5	68.4	65.2
Binarized MNIST	FCN	95.9	**96.4**	94.5	95.2	95.9	95.0
MNIST17	FCN	**98.6**	**98.6**	93.7	98.1	98.2	98.4
IMDb	BERT	**87.6**	87.4	86.1	87.3	86.2	85.9

Table 4.2: Accuracy for PvN classification with PU learning. "*" denote oracle early stopping as defined in Sec. 3.6. Results reported by aggregating aggregating over 10 epochs and 3 seeds.

4.9 Proofs

Proof of Theorem 4.4.1. According to DKW inequality [38], with high probability, the empirical pdfs $\hat{q}_u$ and $\hat{q}_p$ are bounded around q_u and q_p at any $c \in [0,1]$:

$$\mathbb{P}\left(\forall c \in [0,1], |\hat{q}_u(c) - q_u(c)| \leq \sqrt{\frac{\log(2/\delta)}{2n_u}}\right) \geq 1-\delta$$

and

$$\mathbb{P}\left(\forall c \in [0,1], |\hat{q}_p(c) - q_p(c)| \leq \sqrt{\frac{\log(2/\delta)}{2n_p}}\right) \geq 1-\delta\,.$$

For the remaining of the proof, we assume that $|\hat{q}_u(c) - q_u(c)| \leq \sqrt{\frac{\log(4/\delta)}{2n_u}}$ and $|\hat{q}_p(c) - q_p(c)| \leq \sqrt{\frac{\log(4/\delta)}{2n_p}}$ hold for all $c \in [0,1]$, which happens with probability $1-\delta$.

First, we give a general bound on the difference between $\hat{q}_u(c)/\hat{q}_p(c)$ and $q_u(c)/q_p(c)$:

$$\begin{aligned}\left|\frac{\hat{q}_u(c)}{\hat{q}_p(c)} - \frac{q_u(c)}{q_p(c)}\right| &= \frac{1}{\hat{q}_p(c)\cdot q_p(c)}\left|\hat{q}_u(c)\cdot q_p(c) - q_p(c)\cdot q_u(c) + q_p(c)\cdot q_u(c) - \hat{q}_p(c)\cdot q_u(c)\right| \\ &\leq \frac{1}{\hat{q}_p(c)}\left|\hat{q}_u(c) - q_u(c)\right| + \frac{q_u(c)}{\hat{q}_p(c)\cdot q_p(c)}\left|\hat{q}_p(c) - q_p(c)\right|\,. \end{aligned} \tag{4.2}$$

Plugging the high probability bounds, we have

$$\left|\frac{\hat{q}_u(c)}{\hat{q}_p(c)} - \frac{q_u(c)}{q_p(c)}\right| \leq \frac{1}{\hat{q}_p(c)}\left(\sqrt{\frac{\log(4/\delta)}{2n_u}} + \frac{q_u(c)}{q_p(c)}\sqrt{\frac{\log(4/\delta)}{2n_p}}\right)\,. \tag{4.3}$$

Recall that $\hat{c} := \arg\min_{c\in[0,1]} \frac{\hat{q}_u(c)}{\hat{q}_p(c)} + \frac{\gamma}{\hat{q}_p(c)}\mathcal{E}_n$ and $q_u(c_0)/q_p(c_0) \leq 1$. Applying (4.3) to c_0 gives

$$\frac{\hat{q}_u(\hat{c})}{\hat{q}_p(\hat{c})} + \frac{\gamma}{\hat{q}_p(\hat{c})}\mathcal{E}_n \leq \frac{\hat{q}_u(c_0)}{\hat{q}_p(c_0)} + \frac{\gamma}{\hat{q}_p(c_0)}\mathcal{E}_n \leq \alpha_0 + \frac{1+\gamma}{\hat{q}_p(c_0)}\mathcal{E}_n\,.$$

Since $q_p(c_0) \geq 2\sqrt{\frac{\log 4/\delta}{2n_p}}$ and $\hat{q}_p(c_0) \geq q_p(c_0) - \sqrt{\frac{\log 4/\delta}{2n_p}}$ we have

$$\frac{\hat{q}_u(\hat{c})}{\hat{q}_p(\hat{c})} + \frac{\gamma}{\hat{q}_p(\hat{c})}\mathcal{E}_n \leq \alpha_0 + \frac{2(1+\gamma)}{q_p(c_0)}\mathcal{E}_n\,, \tag{4.4}$$

which gives the upper bound $\hat{\alpha} \leq \alpha_0 + \frac{2(1+\gamma)}{q_p(c_0)}\mathcal{E}_n$.

We now derive the lower bound for $\hat{\alpha}$. Applying (4.3) to $\hat{c}$ gives

$$\begin{aligned}\frac{\hat{q}_u(\hat{c})}{\hat{q}_p(\hat{c})} &\geq \frac{q_u(\hat{c})}{q_p(\hat{c})}\left(1 - \frac{1}{\hat{q}_p(\hat{c})}\sqrt{\frac{\log 4/\delta}{2n_p}}\right) - \frac{1}{\hat{q}_p(\hat{c})}\sqrt{\frac{\log 4/\delta}{2n_u}} \\ &\geq \alpha^* - \frac{1}{\hat{q}_p(\hat{c})}\mathcal{E}_n\,,\end{aligned} \tag{4.5}$$

where we use the fact that $\alpha^* \leq 1$ and

$$\frac{1}{\hat{q}_p(\hat{c})}\sqrt{\frac{\log 4/\delta}{2n_p}} \leq \frac{1}{\hat{q}_p(\hat{c})}\mathcal{E}_n \leq \alpha_0 + \frac{2(1+\gamma)}{q_p(c_0)}\mathcal{E}_n \leq 1\,.$$

Plugging (4.5) into (4.4) gives

$$\frac{\gamma - 1}{\hat{q}_p(\hat{c})}\mathcal{E}_n \leq \alpha_0 - \alpha^* + \frac{2(1+\gamma)}{q_p(c_0)}\mathcal{E}_n\,.$$

Plugging this upper bound on $\frac{1}{\hat{q}_p(\hat{c})}\mathcal{E}_n$ back into (4.5) concludes the proof.

□

Proof of Corollary 4.4.2. We only consider large enough n_p such that $c_f \in \mathcal{C}$. According to the condition we know that $\alpha^* = \alpha = q_u(c_f)/q_p(c_f)$. According to the definition of c_0 we have

$$\alpha_0 + \frac{2(1+\gamma)}{q_p(c_0)}\mathcal{E}_n \leq \alpha + \frac{2(1+\gamma)}{q_p(c_f)}\mathcal{E}_n\,.$$

Thus $\alpha_0 - \alpha + \frac{2(1+\gamma)}{q_p(c_0)}\mathcal{E}_n \leq \frac{2(1+\gamma)}{q_p(c_f)}\mathcal{E}_n$ which concludes the proof.

□

Proof of Proposition 4.5.1. First we observe that having $w(x) = 1 - f^*(x)$ leads to the objective value being minimized to 0 as well as a perfect classifier f. This is because

$$\frac{1}{1-\alpha}\int dx p_u(x)(1 - f^*(x))l(f(x), 0) = \int dx p_n(x) l(f(x), 0)$$

thus the objective becomes classifying positive v.s. negative, which leads to a perfect classifier if $\mathcal{F}$ contains one. Now we show that for any f such that the classification error is non-zero then the objective (4.1) must be greater than zero no matter what w is. Suppose f satisfies

$$\int dx p_p(x) l(f(x), 1) + \int dx p_n(x) l(f(x), 0) > 0 \,.$$

We know that either $\int dx p_p(x) l(f(x), 1) > 0$ or $\int dx p_n(x) l(f(x), 0) > 0$ will hold.

If $\int dx p_p(x) l(f(x), 1) > 0$ we know that (4.1) must be positive. If $\int dx p_p(x) l(f(x), 1) = 0$ and $\int dx p_n(x) l(f(x), 0) > 0$ we have $l(f(x), 0) = 1$ almost everywhere in $p_p(x)$ thus

$$\begin{aligned}
&\frac{1}{1-\alpha} \int dx p_u(x) w(x) l(f(x), 0) \\
&= \frac{\alpha}{1-\alpha} \int dx p_p(x) w(x) l(f(x), 0) + \int dx p_n(x) w(x) l(f(x), 0) \\
&= \frac{\alpha}{1-\alpha} \int dx p_p(x) w(x) + \int dx p_n(x) w(x) l(f(x), 0) \,.
\end{aligned}$$

If $\int dx p_p(x) w(x) > 0$ we know that (4.1) must be positive. If $\int dx p_p(x) w(x) = 0$, since we know that

$$\int dx p_u(x) w(x) = \alpha \int dx p_p(x) w(x) + (1-\alpha) \int dx p_n(x) w(x) = 1 - \alpha$$

we have $\int dx p_n(x) w(x) = 1$ which means $w(x) = 1$ almost everywhere in $p_n(x)$. This leads to the fact that $\int dx p_n(x) l(f(x), 0) > 0$ indicates $\int dx p_n(x) w(x) l(f(x), 0) > 0$, which concludes the proof.

□

Part II

Offline Policy Optimization

Chapter 5

An Empirical Study on Behavior Regularized Offline Reinforcement Learning

5.1 Overview

Offline reinforcement learning (RL) describes the setting in which a learner only has access to a fixed dataset of experience. In contrast to online RL, additional interactions with the environment during learning are not permitted. This setting is of particular interest for applications in which deploying a policy is costly or there is a safety concern with updating the policy online [91]. For example, for recommendation systems [25, 90] or health applications [104], deploying a new policy may only be done after extensive testing and evaluation. In these cases, the offline dataset is often very large, potentially encompassing years of logged experience. Nevertheless, the inability to interact with the environment directly poses a challenge to modern RL algorithms.

Issues with RL algorithms in the offline setting typically arise in cases where state and action spaces are large or continuous, necessitating the use of function approximation. While off-policy (deep) RL algorithms such as DQN [102], DDPG [93], and SAC [57] may be run directly on offline datasets to learn a policy, the performance of these algorithms has been shown to be sensitive to the experience dataset distribution, even in the online setting when using a replay buffer [41, 143]. Moreover, Fujimoto et al. [42] and Kumar et al. [77] empirically confirm that in the offline setting, DDPG fails to learn a good policy, even when the dataset is collected by a single behavior policy, with or without noise added to the behavior policy. These striking failure cases are hypothesized to be caused by erroneous generalization of the state-action value function (Q-value function) learned with function approximators, as suggested by Baird [8], Sutton [133], Tsitsiklis and Van Roy [138], Van Hasselt et al. [143].

To remedy this issue, several types of approaches have been proposed recently. One common paradigm is to regularize the learned policy towards the behavior policy, based on the intuition that unseen state-action pairs are more likely to receive overestimated Q-values [42, 66, 77, 82]. A number of works have also leveraged ensembles of target Q-values to stabilize and temper the optimism in the learned Q-function [3, 42]. Yet another class of approaches proposes to learn the

value function for the behavior policy first and then perform policy improvement via advantage weighted regression [111, 146], although this is known to be approximately equivalent to the first paradigm above of regularizing to the behavior policy [112]. These various proposed remedies have been shown to improve upon DQN or DDPG at performing policy improvement based on offline data.

Still, each proposal makes several modifications to the components of baseline off-policy RL algorithms, and each modification may be implemented in many ways. So a natural question to ask is, which of the design choices in these offline RL algorithms are necessary to achieve good performance? For example, to estimate the target Q-value when minimizing the Bellman error, Fujimoto et al. [42] uses a soft combination of two target Q-values, which is different from TD3 [43], where the minimum of two target Q-values is used. This soft combination is maintained by Kumar et al. [77], while further increasing the number of Q-networks from two to four. As another example, when regularizing towards the behavior policy, Jaques et al. [66] uses a Kullback-Leibler (KL) divergence with a fixed regularization weight while Kumar et al. [77] proposes to use Maximum Mean Discrepancy (MMD) with an adaptively trained regularization weight. Are these design choices crucial to success in offline settings? Or are they simply the result of multiple, human-directed iterations of research?

In this chapter, we aim at evaluating the importance of different algorithmic building components as well as comparing different design choices in offline RL approaches. We focus on behavior regularized approaches applied to continuous action domains, encompassing many of the recently demonstrated successes. We introduce *behavior regularized actor critic* (BRAC), a general algorithmic framework which covers existing approaches while enabling us to compare the performance of different variants in a modular way. We find that many simple variants of the behavior regularized approach can yield good performance, while previously suggested sophisticated techniques such as weighted Q-ensembles and adaptive regularization weights are not crucial. Experimental ablations reveal further insights into how different design choices affect the performance and robustness of the behavior regularized approach in the offline RL setting (summarized in Table 5.1).

5.2 Background

5.2.1 Markov Decision Processes

We consider the standard fully-observed Markov Decision Process (MDP) setting [113]. An MDP can be represented as $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(\cdot|s,a)$ is the transition probability distribution function, $R(s,a)$ is the reward function and γ is the discount factor. The goal is to find a policy $\pi(\cdot|s)$ that maximizes the cumulative discounted reward starting from any state $s \in \mathcal{S}$. Let $P^\pi(\cdot|s)$ denote the induced transition distribution for policy π. For later convenience, we also introduce the notion of multi-step transition distributions as P_t^π, where $P_t^\pi(\cdot|s)$ denotes the distribution over the state space after rolling out P^π for t steps starting from state s. For example, $P_0^\pi(\cdot|s)$ is the Dirac delta function at s and $P_1^\pi(\cdot|s) = P^\pi(\cdot|s)$. We use $R^\pi(s)$ to denote the expected reward at state s when following policy π, i.e. $R^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}[R(s,a)]$. The state value function (a.k.a. value function) is defined

by $V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim P_t^\pi(s)} [R^\pi(s_t)]$. The action-value function (a.k.a. Q-function) can be written as $Q^\pi(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)} [V^\pi(s')]$. The optimal policy is defined as the policy π^* that maximizes $V^{\pi^*}(s)$ at all states $s \in \mathcal{S}$. In the commonly used actor critic paradigm, one optimizes a policy $\pi_\theta(\cdot|s)$ by alternatively learning a Q-value function Q_ψ to minimize Bellman errors over single step transitions (s, a, r, s'),

$$\mathbb{E}_{a' \sim \pi_\theta(\cdot|s')} \left[\left(r + \gamma \bar{Q}(s', a') - Q_\psi(s,a) \right)^2 \right], \tag{5.1}$$

where $\bar{Q}$ denotes a target Q function; e.g., it is common to use a slowly-updated target parameter set ψ' to determine the target Q function as $Q_{\psi'}(s', a')$. Then, the policy is updated to maximize the Q-values, $\mathbb{E}_{a \sim \pi(\cdot|s)} [Q_\psi(s,a)]$.

5.2.2 Offline Reinforcement Learning

Offline RL (also known as batch RL [81]) considers the problem of learning a policy π from a fixed dataset $\mathcal{D}$ consisting of single-step transitions (s, a, r, s'). Slightly abusing the notion of "behavior", we define the behavior policy $\pi_b(a|s)$ as the conditional distribution $p(a|s)$ observed in the dataset distribution $\mathcal{D}$. Under this definition, such a behavior policy π_b is always well-defined even if the dataset was collected by multiple, distinct behavior policies. Because direct access to π_b is typically not available, it is common in previous work to approximate this behavior policy via a max-likelihood optimization over $\mathcal{D}$:

$$\hat{\pi}_b := \arg\max_{\hat{\pi}} \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} \left[\log \hat{\pi}(a|s) \right]. \tag{5.2}$$

We denote the learned policy as $\hat{\pi}_b$ and refer to it as the "cloned policy" to distinguish it from the true behavior policy.

In this work, we focus on the offline RL problem for complex continuous domains. We briefly review two recently proposed approaches, BEAR [77] and BCQ [42].

BEAR Motivated by the hypothesis that deep RL algorithms generalize poorly to actions outside the support of the behavior policy, Kumar et al. [77] propose BEAR, which learns a policy to maximize Q-values while penalizing it for diverging from the behavior policy support. BEAR measures divergence from the behavior policy using a sample-based estimate of kernel MMD [53]:

$$\begin{aligned} \mathrm{MMD}_k^2(\pi(\cdot|s), \pi_b(\cdot|s)) = & \mathop{\mathbb{E}}_{x, x' \sim \pi(\cdot|s)} [K(x, x')] \\ & - 2\mathbb{E}_{x \sim \pi(\cdot|s) y \sim \pi_b(\cdot|s)} [K(x,y)] + \mathop{\mathbb{E}}_{y, y' \sim \pi_b(\cdot|s)} [K(y, y')] , \end{aligned} \tag{5.3}$$

where K is a kernel function. Furthermore, to avoid overestimation in the Q-values, the target Q-value function $\bar{Q}$ is calculated as,

$$\bar{Q}(s', a') := \frac{3}{4} \min_{j=1,\ldots,k} Q_{\psi'_j}(s', a') + \frac{1}{4} \max_{j=1,\ldots,k} Q_{\psi'_j}(s', a'), \tag{5.4}$$

where ψ'_j denotes a soft-updated ensemble of target Q functions. BEAR uses size $k = 4$ ensembles. BEAR also penalizes target Q-values by an ensemble variance term. However, their empirical results show that there is no clear benefit to doing so, thus we omit this term.

BCQ BCQ enforces π to be close to π_b via a specific parameterization of π:

$$\pi_\theta(a|s) := \underset{a_i+\xi_\theta(s,a_i)}{\arg\max} Q_\psi(s, a_i + \xi_\theta(s, a_i)) \quad \text{for } a_i \sim \pi_b(a|s),\ i = 1, \ldots, N, \tag{5.5}$$

where ξ_θ is a function approximator with bounded output in $[-\Phi, \Phi]$ where Φ is a hyperparameter. N is an additional hyperparameter used during evaluation to compute π_θ and during training for Q-value updates. The target Q-value function $\bar{Q}$ is calculated as in Equation 5.4 but with $k = 2$.

5.3 Behavior Regularized Actor Critic

Encouraging the learned policy to be close to the behavior policy is a common theme in previous approaches to offline RL. To evaluate the effect of different behavior policy regularizers, we introduce *behavior regularized actor critic* (BRAC), an algorithmic framework which generalizes existing approaches while providing more implementation options.

We begin by describing two common alternatives for adding a policy regularizer to the objective – through a penalty in the value function or as a penalty solely on the policy – and then elaborate on the learning procedures for these regularizations. We then expand on some specific choices of regularizers and how they can be used to reproduce previously proposed offline RL algorithms.

Value penalty (vp) We begin by introducing *value penalty*, a mechanism for regularizing a learned policy to the behavior policy through a penalty in the value function. Specifically, we define the penalized value function as

$$V_D^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{s_t \sim P_t^\pi(s)} \left[R^\pi(s_t) - \alpha D\left(\pi, \pi_b; s_t\right) \right], \tag{5.6}$$

where D is a divergence between distributions over actions (e.g., MMD or KL divergence)[1]. We then learn a policy π to maximize V_D^π at all states. This standard paradigm is commonly used in the literature; for example, in SAC [57], Trust-PCL [105], and reg-MPI [49].

Geist et al. [49] show that when D is strongly convex in its first input π, there is a unique regularized optimal policy π_D^* that maximizes the penalized value function (5.6) at all states. The following proposition gives a lower bound on the performance of the regularized optimal policy π_D^* in terms of the behavior policy π_b and the unregularized optimal policy π^*.

Proposition 5.3.1. Let V^π be the value function for policy π under the original (unpenalized) value definition. For any $s \in \mathcal{S}$ we have

$$V^{\pi_D^*}(s) \geq \max\left\{ V^{\pi_b}(s), V^{\pi^*}(s) - \frac{\alpha}{1-\gamma} D_{\max} \right\}, \tag{5.7}$$

where $D_{\max} = \max_s D\left(\pi^*, \pi_b; s\right)$.

[1] We use $D\left(\pi, \pi_b; s_t\right)$ to denote $D\left(\pi(\cdot|s_t), \pi_b(\cdot|s_t)\right)$.

Proof. D is a divergence, hence non-negative, so we have $V^{\pi^*_D}(s) \geq V_D^{\pi^*_D}(s)$. Thus, it suffices to show that the RHS of (5.7) is a lower bound for $V_D^{\pi^*_D}(s)$, which can be seen by plugging π_b and π^* into (5.6) respectively then applying the fact that π^*_D is the maximizer of (5.6). □

Remark 5.3.2. The lower bound in (5.7) goes to $V^{\pi^*}(s)$ as $\alpha \to 0$ or $\pi_b \to \pi^*$, and goes to $V^{\pi_b}(s)$ as $\alpha \to +\infty$.

Proposition 5.3.1 characterizes the performance of the optimal regularized policy in terms of the regularization weight α and the quality of the optimal policy, assuming a sufficient ability to learn V_D^π and π^*_D[2]. In practical offline settings, one must consider the error introduced into the calculation of V_D^π (and thus π^*_D) due to finite samples. That is, in practice π^*_D is learned via a value function $\hat{V}_D^\pi$ on the *empirical* MDP (induced by the finite samples observed in the offline dataset $\mathcal{D}$). Confidence bounds on the true V_D^π may be derived from confidence bounds on the true P and R, which are inversely correlated with the number of samples [110]. Conceptually, the regularization coefficient α should therefore be chosen to trade-off with these confidence intervals; i.e., when the confidence bounds are large due to a limited finite dataset, it is better to stay close to the behavior policy to avoid drastic degradation in quality of the policy on the true MDP. As we focus on an empirical study of behavior regularized methods, a theoretical and more specific characterization of the finite sample analysis with respect to α is outside the scope of this work, although the curious reader may look to Laroche et al. [83] for one possible approach to this question.

Policy regularization (pr) The second way to add the regularizer is to only regularize the policy during policy optimization. That is, we first learn *unregularized* state and Q-value functions V^π, Q^π. We then apply regularization during policy optimization. That is, at each state s we aim to learn π to maximize $\mathbb{E}_{a\sim\pi(\cdot|s)}[Q^\pi(s,a) - \alpha D(\pi, \pi_b; s)]$, for an appropriate regularization $\alpha \geq 0$. We call this variant of behavior regularized learning *policy regularization* (pr).

Policy regularization is similar to the regularization employed in many practical algorithms, such as A3C [103]. Although this form of regularization enjoys much practical success, it is important to note that in contrast to value penalty whose convergence properties are well-understood [49, 107], the convergence of policy regularization is *not* well-characterized. In fact, in certain settings it is known that value iteration with policy regularization is non-convergent [6]. Still, because it is a common choice in practice, we keep it as an option in our empirical study.

Learning with a regularizer We now describe how these two regularization approaches are performed in BRAC. Following the typical actor critic framework, we learn a Q-value via the objective,

$$\min_{Q_\psi} \mathbb{E}_{\substack{(s,a,r,s')\sim\mathcal{D} \\ a'\sim\pi_\theta(\cdot|s')}}[\delta(s,a,r,s',a')^2], \tag{5.8}$$

where

$$\delta(s,a,r,s',a') = -Q_\psi(s,a) + r + \gamma\big(\bar{Q}(s',a') - \alpha_Q \hat{D}\left(\pi_\theta, \pi_b; s'\right)\big),$$

[2]For characterization of the convergence of value and policy-based instantiations of the value penalty regularization, see Geist et al. [49], Laroche et al. [83], Neu et al. [107], Shani et al. [121].

$\bar{Q}$ denotes a target Q function (calculated using e.g., an ensemble, slowly updated target networks, etc.), and $\hat{D}$ denotes a sample-based estimate of the divergence function D.

The policy learning objective can be written as,

$$\max_{\pi_\theta} \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\, \mathbb{E}_{a''\sim\pi_\theta(\cdot|s)}\left[Q_\psi(s,a'')\right] - \alpha_\pi \hat{D}\left(\pi_\theta, \pi_b; s\right)\Big] . \tag{5.9}$$

Accordingly, BRAC performs alternating gradient updates based on (5.8) and (5.9). Note that for value penalty $\alpha_Q = \alpha_\pi \geq 0$, whereas in policy regularization $\alpha_Q = 0$ and $\alpha_\pi \geq 0$.

In addition to the choice of value penalty or policy regularization, the choice of D and how to perform sample estimation of $\hat{D}$ are key design choices of BRAC, and we elaborate on a few options that we will evaluate in our empirical study:

Kernel MMD We can compute a sample based estimate of kernel MMD (Eq 5.3) by drawing samples from both π_θ and π_b. Because we do not have access to multiple samples from π_b, this requires a pre-estimated cloned policy $\hat{\pi}_b$.

KL Divergence With KL Divergence, the behavior regularizer can be written as

$$D_{\mathrm{KL}}\left(\pi_\theta, \pi_b; s\right) = \mathbb{E}_{a\sim\pi_\theta(\cdot|s)}\left[\log \pi_\theta(a|s) - \log \pi_b(a|s)\right] .$$

Directly estimating D_{KL} via samples requires having access to the density of both π_θ and π_b; as in MMD, the cloned $\hat{\pi}_b$ can be used in place of π_b. Alternatively, we can avoid estimating π_b explicitly, by using the dual form of the KL-divergence. Specifically, any f-divergence [26] has a dual form [109] given by,

$$\begin{aligned} D_f(p,q) &= \mathbb{E}_{x\sim p}\left[f\left(q(x)/p(x)\right)\right] \\ &= \max_{g:\mathcal{X}\mapsto \mathrm{dom}(f^*)} \mathbb{E}_{x\sim q}\left[g(x)\right] - \mathbb{E}_{x\sim p}\left[f^*(g(x))\right] , \end{aligned}$$

where f^* is the Fenchel dual of f. In this case, one no longer needs to estimate a cloned policy $\hat{\pi}_b$ but instead needs to learn a discriminator function g with minimax optimization as in [109]. This sample based dual estimation can be applied to any f-divergence. In the case of a KL-divergence, $f(x) = -\log x$ and $f^*(t) = -\log(-t) - 1$.

Wasserstein One may also use the Wassertein distance as the divergence D. For sample-based estimation, one may use its dual form,

$$W(p,q) = \sup_{g:||g||_L\leq 1} \mathbb{E}_{x\sim p}\left[g(x)\right] - \mathbb{E}_{x\sim q}\left[g(x)\right]$$

and maintain a discriminator g as in [55].

Now we discuss how existing approaches can be instantiated under the framework of BRAC.

BEAR To re-create BEAR with BRAC, one uses policy regularization with the sample-based kernel MMD for $\hat{D}$ and uses a min-max ensemble estimate for $\bar{Q}$ (Equation 5.4). Furthermore, BEAR adaptively trains the regularization weight α as a Lagrange multiplier: it sets a threshold $\epsilon > 0$ for the kernel MMD distance and increases α if the current average divergence is above the threshold and decreases α if below the threshold.

BCQ The BCQ algorithm does not use any regularizers (i.e. $\alpha = 0$ for both value and policy objectives). Still, the algorithm may be realized by BRAC if one restricts the policy optimization in Equation 5.9 to be over parameterized policies based on Equation 5.5.

KL-Control There has been a rich set of work which investigates regularizing the learned policy through KL-divergence with respect to another policy, e.g. [1, 68, 105, 119]. Notably, Jaques et al. [66] apply this idea to offline RL in discrete action domains by introducing a KL value penalty in the Q-value definition. BRAC can realize this algorithm as well.

To summarize, one can instantiate the behavior regularized actor critic framework with different design choices, including how to estimate the target Q value, which divergence to use, whether to learn α adaptively, whether to use a value penalty in the Q function objective (5.8) or just use policy regularization in (5.9) and so on. In the next section, we empirically evaluate a set of these different design choices to provide insights into what actually matters when approaching the offline RL problem.

5.4 Experiments

Table 5.1: Summary of our empirical findings on continuous control tasks we evaluate.

Design Choice	**Observation**
Fixed v.s. adaptive training of α.	Fixed α usually performs better than adaptive training (Figure 5.1).
Target Q-value construction.	Taking the min of two target Q-functions is sufficient (Figures 5.2 & 5.3).
Value penalty v.s. policy regularization.	Value penalty performs better (Figures 5.4 & 5.8 (Appendix)).
Divergence for regularization.	MMD, KL, KL-dual and Wasserstein perform similarly well (Figure 5.5).
KL-control v.s. BEAR, BCQ, SAC, or BC.	KL-control typically outperforms the other methods (Figure 5.6).

The BRAC framework encompasses several previously proposed methods depending on specific design choices (e.g., whether to use value penalty or policy regularization, how to compute the target Q-value, and how to impose the behavior regularization). For a practitioner, key questions are: How should these design choices be made? Which variations among these different algorithms actually matter? To answer these questions, we perform a systematic evaluation of BRAC under different design choices.

Following Kumar et al. [77], we evaluate performance on four Mujoco [136] continuous control environments in OpenAI Gym [18]: Ant-v2, HalfCheetah-v2, Hopper-v2, and Walker2d-v2. In many real-world applications of RL, one has logged data from sub-optimal policies (e.g., robotic control and recommendation systems). To simulate this scenario, we collect the offline dataset with a sub-optimal policy perturbed by additional noise. We evaluate offline RL algorithms by training on these fixed datasets and evaluating the learned policies on the real environments. Currently, BEAR [77] provides state-of-the-art performance on these tasks, so to understand the effect of variations under our BRAC framework, we start by implementing

BEAR in BRAC and run a series of comparisons by varying different design choices: adaptive vs. fixed regularization, different ensembles for estimating target Q-values, value penalty vs. policy regularization and divergence choice for the regularizer. We then evaluate BCQ, which has a different design in the BRAC framework, and compare it to other BRAC variants as well as several baseline algorithms. We briefly summarize our findings in Table 5.1.

5.4.1 Fixed v.s. adaptive regularization weights

In BEAR, regularization is controlled by a threshold ϵ, which is used for adaptively training the Lagrangian multiplier α, whereas typically (e.g., in KL-control) one uses a fixed α. In our initial experiments with BEAR, we found that when using the recommended value of ϵ, the learned value of α consistently increased during training, implying that the MMD constraint between π_θ and π_b was almost never satisfied. This suggests that BEAR is effectively performing policy regularization with a large α rather than constrained optimization. This led us to question if adaptively training α is better than using a fixed α. To investigate this question, we evaluate the performance of both approaches (with appropriate hyperparameter tuning for each, over either α or ϵ) in Figure 5.1. On most datasets, both approaches learn a policy that is much better than the *partially trained policy*[3], although we do observe a consistent modest advantage when using a fixed α. Because using a fixed α is simpler and performs better than adaptive training, we use this approach in subsequent experiments.

5.4.2 Ensemble for target Q-values

Another important design choice in BRAC is how to compute the target Q-value, and specifically, whether one should use the sophisticated ensemble strategies employed by BEAR and BCQ. Both BEAR and BCQ use a weighted mixture of the minimum and maximum among multiple learned Q-functions (compared to TD3 which simply uses the minimum of two). BEAR further increases the number of Q-functions from 2 to 4. To investigate these design choices, we first experiment with different number of Q-functions $k = \{1, 2, 4\}$. Results are shown in Figure 5.2. Fujimoto et al. [43] show that using two Q-functions provides significant improvements in online RL; similarly, we find that using $k = 1$ sometimes fails to learn a good policy (e.g., in Walker2d) in the offline setting. Using $k = 4$ has a small advantage compared to $k = 2$ except in Hopper. Both $k = 2$ and $k = 4$ significantly improve over the partially trained policy baseline. In general, increasing the value of k in the ensemble will lead to more stable and/or better performance, but requires more computation cost. On these domains we found that $k = 4$ only gives marginal improvement over $k = 2$, so we use $k = 2$ in our remaining experiments.

Regarding whether using a weighed mixture of Q-values or the minimum, we compare these two options under $k = 2$. Results are shown in Figure 5.3. We find that taking the minimum performs slightly better than taking a mixture except in Hopper, and both successfully outperform the partially trained policy in all cases. Due to the simplicity and strong performance of taking the minimum of two Q-functions, we use this approach in subsequent experiments.

[3]The partially trained policy is the policy used to collect data *without* injected noise. The true *behavior policy* and *behavior cloning* will usually result in worse performance due to injected noise when collecting the data.

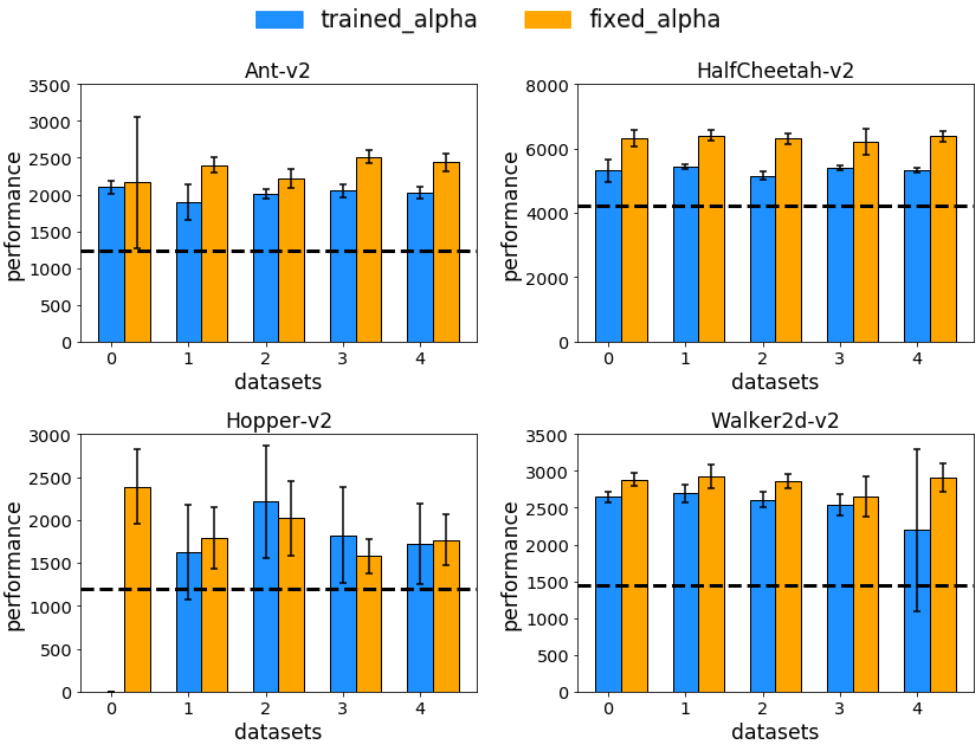


Figure 5.1: Comparing fixed α with adaptively trained α. Black dashed lines are the performance of the partially trained policies (distinct from the behavior policies which have injected noise). We report the mean over the last 10 evaluation points (during training) averaged over 5 different random seeds. Each evaluation point is the return averaged over 20 episodes. We omit the bar if the performance is negative.

5.4.3 Value penalty or policy regularization

So far, we have evaluated variations in regularization weights and ensemble of Q-values. We found that the technical complexity introduced in recent works is not always necessary to achieve state-of-the-art performance. With these simplifications, we now evaluate a major variation of design choices in BRAC — using *value penalty* or *policy regularization*. We follow our simplified version of BEAR: MMD policy regularization, fixed α, and computation of target Q-values based on the minimum of a $k = 2$ ensemble. We compare this instantiation of BRAC to its value penalty version, with results shown in Figure 5.4. While both variants outperform the partially trained policy, we find that value penalty performs slightly better than policy regularization in most cases. We consistently observed this advantage with other divergence choices (see Appendix Figure 5.8 for a full comparison). The benefit of value penalty over policy regularization may be partially explained by the stronger theoretical foundations possessed by value penalty [49, 107].

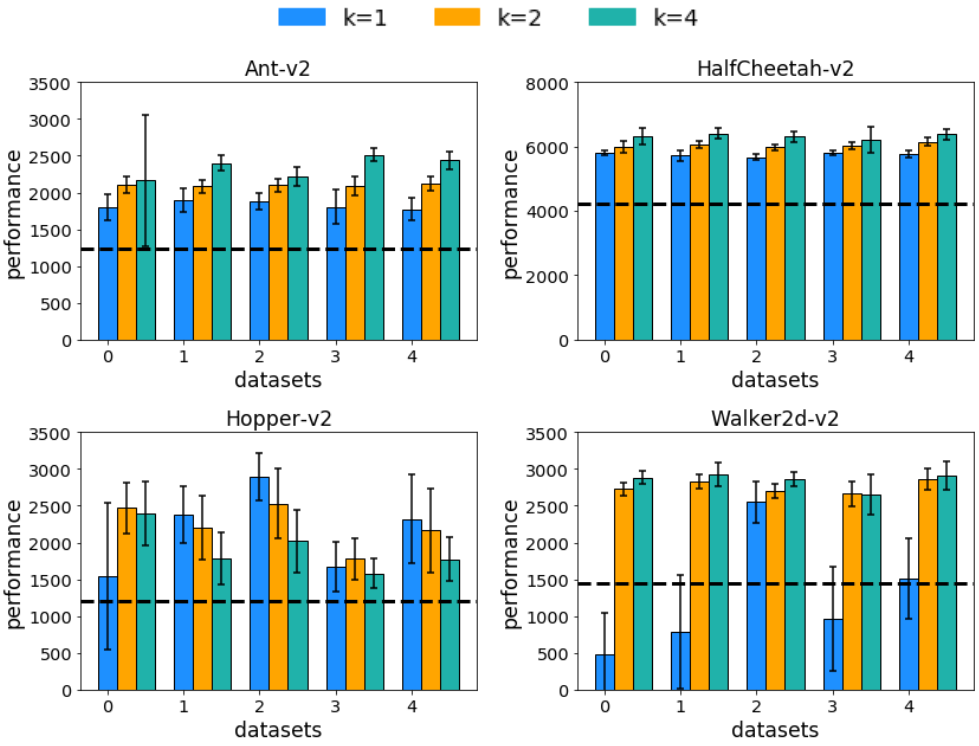


Figure 5.2: Comparing different number of Q-functions for target Q-value ensemble. We use a weighted mixture to compute the target value for all of these variants. As expected, we find that using an ensemble ($k > 1$) is better than using a single Q-function.

5.4.4 Divergences for regularization

We evaluated four choices of divergences used as the regularizer D: (a) MMD (as in BEAR), (b) KL in the primal form with estimated behavior policy (as in KL-control), and (c) KL and (d) Wasserstein in their dual forms without estimating a behavior policy. As shown in Figure 5.5, we do not find any specific divergence performing consistently better or worse than the others. All variants are able to learn a policy that significantly improves over the behavior policy in all cases.

In contrast, Kumar et al. [77] argue that sampled MMD is superior to KL based on the idea that it is better to regularize the support of the learned policy distribution to be within the support of the behavior policy rather than forcing the two distributions to be similar. While conceptually reasonable, we do not find support for that argument in our experiments: (i) we find that KL and Wassertein can perform similarly well to MMD even though they are not designed for support matching; (ii) we briefly tried divergences that are explicitly designed for support matching (the relaxed KL and relaxed Wasserstein distances proposed by [149]), but did not observe a clear benefit to the additional complexity. We conjecture that this is because even if one uses noisy or multiple behavior policies to collect data, the noise is reflected more in the diversity of states rather than the diversity of actions on a single state (due to the nature of environment dynamics). However, we expect this support matching vs. distribution matching distinction may matter in other scenarios such as smaller state spaces or contextual bandits, which is a potential direction

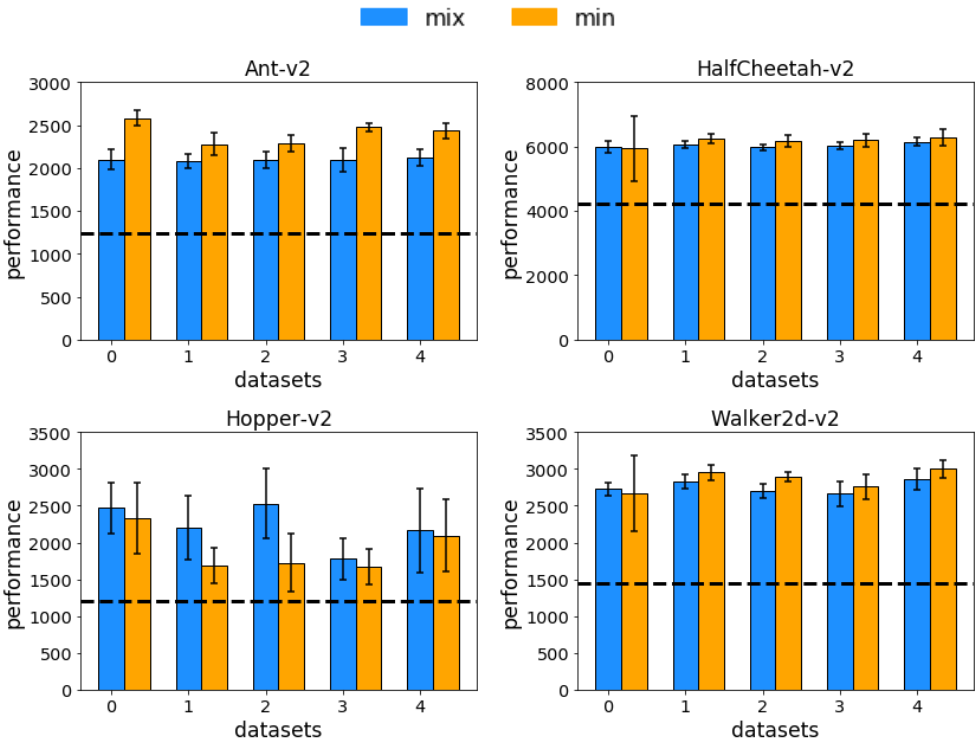


Figure 5.3: Comparing taking the minimum v.s. a weighted mixture in Q-value ensemble. We find that simply taking the minimum is usually slightly better, except in Hopper-v2.

for future work.

5.4.5 Comparison to BCQ and other baselines

We now compare one of our best performing algorithms so far, kl_vp (value penalty with KL divergence in the primal form), to BCQ, BEAR, and two other baselines: vanilla SAC (which uses adaptive entropy regularization) and behavior cloning. Figure 5.6 shows the comparison. We find that vanilla SAC only works in the HalfCheetah environment and fails in the other three environments. Behavior cloning never learns a better policy than the partially trained policy used to collect the data. Although BCQ consistently learns a policy that is better than the partially trained policy, its performance is always clearly worse than kl_vp (and other variants whose performance is similar to kl_vp, according to our previous experiments). We conclude that BCQ is less favorable than explicitly using a divergence for behavior regularization (BEAR and kl_vp). Although, tuning additional hyperparameters beyond Φ for BCQ may improve performance.

5.4.6 Hyperparameter Sensitivity

In our experiments, we find that many simple algorithmic designs achieve good performance under the framework of BRAC. For example, all of the 4 divergences we tried perform similarly well when used for regularization. In these experiments, we allowed for appropriate hyperparameter tuning over policy learning rate and regularization weight, as we initially found that not

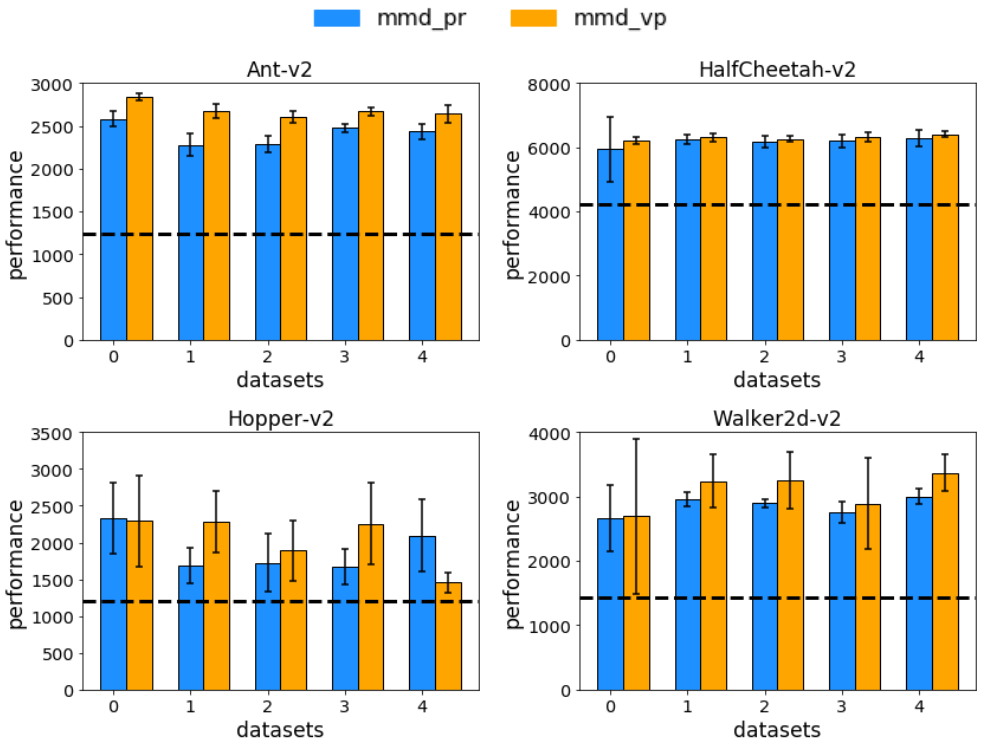


Figure 5.4: Comparing policy regularization (pr) v.s. value penalty (vp) with MMD. The use of value penalty is usually slightly better.

doing so can lead to premature and incorrect conclusions.[4] However, some design choices may be more robust to hyperparameters than others. To investigate this, we also analyzed the sensitivity to hyperparameters for all algorithmic variants (Appendix Figures 5.9 and 5.10). To summarize, we found that (i) MMD and KL Divergence are similar in terms of sensitivity to hyperparameters, (ii) using the dual form of divergences (e.g. KL dual, Wasserstein) appears to be more sensitive to hyperparameters, possibly because of the more complex training procedure (optimizing a min-imax objective), and (iii) value penalty is slightly more sensitive to hyperparameters than policy regularization despite its more favorable performance under the best hyperparameters.

Although we utilized hyperparameter searches in our results, in pure offline RL settings, testing on the real environment is infeasible. Thus, a natural question is how to select the best hyperparameter or the best learned policy among many without direct testing. As a preliminary attempt, we evaluated whether the Q-values learned during training can be used as a proxy for hyperparameter selection. Specifically, we look at the correlation between the average learned Q-values (in mini-batches) and the true performance. Figure 5.7 shows sampled visualizations of these Q-values. We find that the learned Q-values are not a good indicator of the performance, even when they are within a reasonable range (i.e., not diverging during training). A more formal direction for doing hyperparameter selection is to do off-policy evaluation. However, off-policy evaluation is an open research problem with limited success on complex continuous control tasks

[4] For example, taking the optimal hyperparameters from one design choice and then applying them to a different design choice (e.g., MMD vs KL divergence) can lead to incorrect conclusions (specifically, that using KL is worse than using MMD, only because one transferred the hyperparameters used for MMD to KL).

(see Irpan et al. [63], Liu et al. [97], Nachum et al. [106] for recent attempts), we leave hyperparameter selection as future work and encourage more researchers to investigate this direction.

5.5 Conclusions

In this chapter, we introduced *behavior regularized actor critic* (BRAC), an algorithmic framework, which generalizes existing approaches to solve the offline RL problem by regularizing to the behavior policy. In our experiments, we showed that many sophisticated training techniques, such as weighted target Q-value ensembles and adaptive regularization coefficients are not necessary in order to achieve state-of-the-art performance. We found that the use of value penalty is slightly better than policy regularization, while many possible divergences (KL, MMD, Wasserstein) can achieve similar performance. Perhaps the most important differentiator in these offline settings is whether proper hyperparameters are used. Although some variants of BRAC are more robust to hyperparameters than others, every variant relies on a suitable set of hyperparameters to train well. Finding these hyperparameters without interacting with the environment is a challenging open problem (i.e., off-policy evaluation).

5.6 Additional Experiment Results

5.6.1 Additional experiment details

Dataset collection Following Kumar et al. [77], we evaluate performance on four Mujoco [136] continuous control environments in OpenAI Gym [18]: Ant-v2, HalfCheetah-v2, Hopper-v2, and Walker2d-v2. In many real-world applications of RL, one has logged data from sub-optimal policies (e.g., robotic control and recommendation systems). To simulate this scenario, we collect the offline dataset with a sub-optimal policy perturbed by additional noise. To obtain a partially trained policy, we train a policy with SAC and online interactions until the policy performance achieves a performance threshold ($1000, 4000, 1000, 1000$ for Ant-v2, HalfCheetah-v2, Hopper-v2, Walker2d-v2, respectively, similar to the protocol established by Kumar et al. [77]). Then, we perturb the partially trained policy with noise (Gaussian noise or ϵ-greedy at different levels) to simulate different exploration strategies resulting in five noisy behavior policies. We collect 1 million transitions according to each behavior policy resulting in five datasets for each environment.

For each environment, we collect five datasets: {no-noise, eps-0.1, eps-0.3, gauss-0.1, gauss-0.3} using a partially trained policy π. Each dataset contains 1 million transitions. Different datasets are collected with different injected noise, corresponding to different levels and strategies of exploration. The specific noise configurations are shown below:

- **no-noise** : The dataset is collected by purely executing the partially trained policy π without adding noise.
- **eps-0.1**: We make an epsilon greedy policy π' with 0.1 probability. That is, at each step, π' has 0.1 probability to take a uniformly random action, otherwise takes the action sampled from π. The final dataset is a mixture of three parts: 40% transitions are collected by π',

40% transitions are collected by purely executing π, the remaining 20% are collected by a random walk policy which takes a uniformly random action at every step. This mixture is motivated by that one may only want to perform exploration in only a portion of episodes when deploying a policy.

- **eps-0.3**: π' is an epsilon greedy policy with 0.3 probability to take a random action. We do the same mixture as in eps-0.1.
- **gauss-0.1**: π' is taken as adding an independent $\mathcal{N}(0, 0.1^2)$ Gaussian noise to each action sampled from π. We do the same mixture as in eps-0.1.
- **gauss-0.3**: π' is taken as adding an independent $\mathcal{N}(0, 0.3^2)$ Gaussian noise to each action sampled from π. We do the same mixture as in eps-0.1.

Hyperparameter search In preliminary experiments, we found that policy learning rate and regularization strength have a significant effect on performance. As a result, for each variant of BRAC and each environment, we do a grid search over policy learning rate and regularization strength. For policy learning rate, we search over six values, ranging from $3 \cdot 10^{-6}$ to 0.001. The regularization strength is controlled differently in different algorithms. In the simplest case, the regularization weight α is fixed; in BEAR the regularization weight is adaptively trained with dual gradient ascent based on a divergence constraint ϵ that is tuned as a hyperparameter; in BCQ the corresponding tuning is for the perturbation range Φ. For each of these options, we search over five values. For existing algorithms such as BEAR and BCQ, the reported hyperparameters in their papers [42, 77] are included in this search range, and we further empirically confirmed that our implementations of these previously proposed algorithms match the results of the originally published implementations.

Mor specifically, for policy learning rate, we search over six values: $\{3 \cdot 10^6, 1 \cdot 10^5, 3 \cdot 10^5, 0.0001, 0.0003, 0.001\}$. The regularization strength is controlled differently in different algorithms:

- In BCQ, we search for the perturbation range $\Phi \in \{0.005, 0.015, 0.05, 0.15, 0.5\}$. 0.05 is the reported value by its paper [42].
- In BEAR the regularization weight α is adaptively trained with dual gradient ascent based on a divergence constraint ϵ that is tuned as a hyperparameter.
  We search for $\epsilon \in \{0.015, 0.05, 0.15, 0.5, 1.5\}$. 0.05 is the reported value by its paper [77].
- When MMD is used with a fixed α, we search for $\alpha \in \{3, 10, 30, 100, 300\}$.
- When KL divergence is used with a fixed α (both KL and KL_dual), we search for $\alpha \in \{0.1, 0.3, 1.0, 3.0, 10.0\}$.
- When Wasserstein distance is used with a fixed α, we search for $\alpha \in \{0.3, 1.0, 3.0, 10.0, 30.0\}$.

in BEAR the regularization weight is adaptively trained with dual gradient ascent based on a divergence constraint ϵ that is tuned as a hyperparameter;

In the simplest case, the regularization weight α is fixed; in BCQ the corresponding tuning is for the perturbation range Φ. For each of these options, we search over five values (see Appendix for details). For existing algorithms such as BEAR and BCQ, the reported hyperparameters in their papers [42, 77] are included in this search range, We select the best hyperparameters

according to the average performance over all five datasets.

Implementation details All experiments are implemented with Tensorflow and executed on CPUs. For all function approximators, we use fully connected neural networks with RELU activations. For policy networks, we use $\tanh(\text{Gaussian})$ on outputs following BEAR [77], except for BCQ where we follow their open sourced implementation. For BEAR and BCQ we follow the network sizes as in their papers. For other variants of BRAC, we shrink the policy networks from $(400, 300)$ to $(200, 200)$ and Q-networks from $(400, 300)$ to $(300, 300)$ for saving computation time without losing performance. Q-function learning rate is always 0.001. As in other deep RL algorithms, we maintain source and target Q-functions with an update rate 0.005 per iteration. For MMD we use Laplacian kernels with bandwidth reported by [42]. For divergences in the dual form (both KL_dual and Wasserstein), we training a $(300, 300)$ fully connected network as the critic in the minimax objective. Gradient penalty (one sided version of the penalty in [55] with coefficient 5.0) is applied to both KL and Wasserstein dual training. In each training iteration, the dual critic is updated for 3 steps (which we find better than only 1 step) with learning rate 0.0001. We use Adam for all optimizers. Each agent is trained for 0.5 million steps with batch size 256 (except for BCQ we use 100 according their open sourced implementation). At test time we follow [77] and [42] by sampling 10 actions from π_θ at each step and take the one with highest learned Q-value.

5.6.2 Value penalty v.s. policy regularization

See Figure 5.8.

5.6.3 Full performance results under different hyperparameters

See Figure 5.9 and 5.10.

5.6.4 Additional training curves

See FIgure 5.11, 5.12 and 5.13.

5.6.5 Full performance results under the best hyperparameters

See Table 5.2, 5.3, 5.4 and 5.5.

Environment: Ant-v2			**Partially trained policy: 1241**		
dataset	no-noise	eps-0.1	eps-0.3	gauss-0.1	gauss-0.3
SAC	0	-1109	-911	-1071	-1498
BC	1235	1300	1278	1203	1240
BCQ	1921	1864	1504	1731	1887
BEAR	2100	1897	2008	2054	2018
MMD_vp	**2839**	**2672**	**2602**	**2667**	2640
KL_vp	2514	2530	2484	2615	**2661**
KL_dual_vp	2626	2334	2256	2404	2433
W_vp	2646	2417	2409	2474	2487
MMD_pr	2583	2280	2285	2477	2435
KL_pr	2241	2247	2181	2263	2233
KL_dual_pr	2218	1984	2144	2215	2201
W_pr	2241	2186	2284	2365	2344

Table 5.2: Evaluation results with tuned hyperparameters. 0 performance means overflow encountered during training due to diverging Q-functions.

Environment: HalfCheetah-v2			**Partially trained policy: 4206**		
dataset	no-noise	eps-0.1	eps-0.3	gauss-0.1	gauss-0.3
SAC	5093	6174	5978	6082	6090
BC	4465	3206	3751	4084	4033
BCQ	5064	5693	5588	5614	5837
BEAR	5325	5435	5149	5394	5329
MMD_vp	**6207**	**6307**	**6263**	**6323**	**6400**
KL_vp	6104	6212	6104	6219	6206
KL_dual_vp	6209	6087	6359	5972	6340
W_vp	5957	6014	6001	5939	6025
MMD_pr	5936	6242	6166	6200	6294
KL_pr	6032	6116	6035	5969	6219
KL_dual_pr	5944	6183	6207	5789	6050
W_pr	5897	5923	5970	5894	6031

Table 5.3: Evaluation results with tuned hyperparameters.

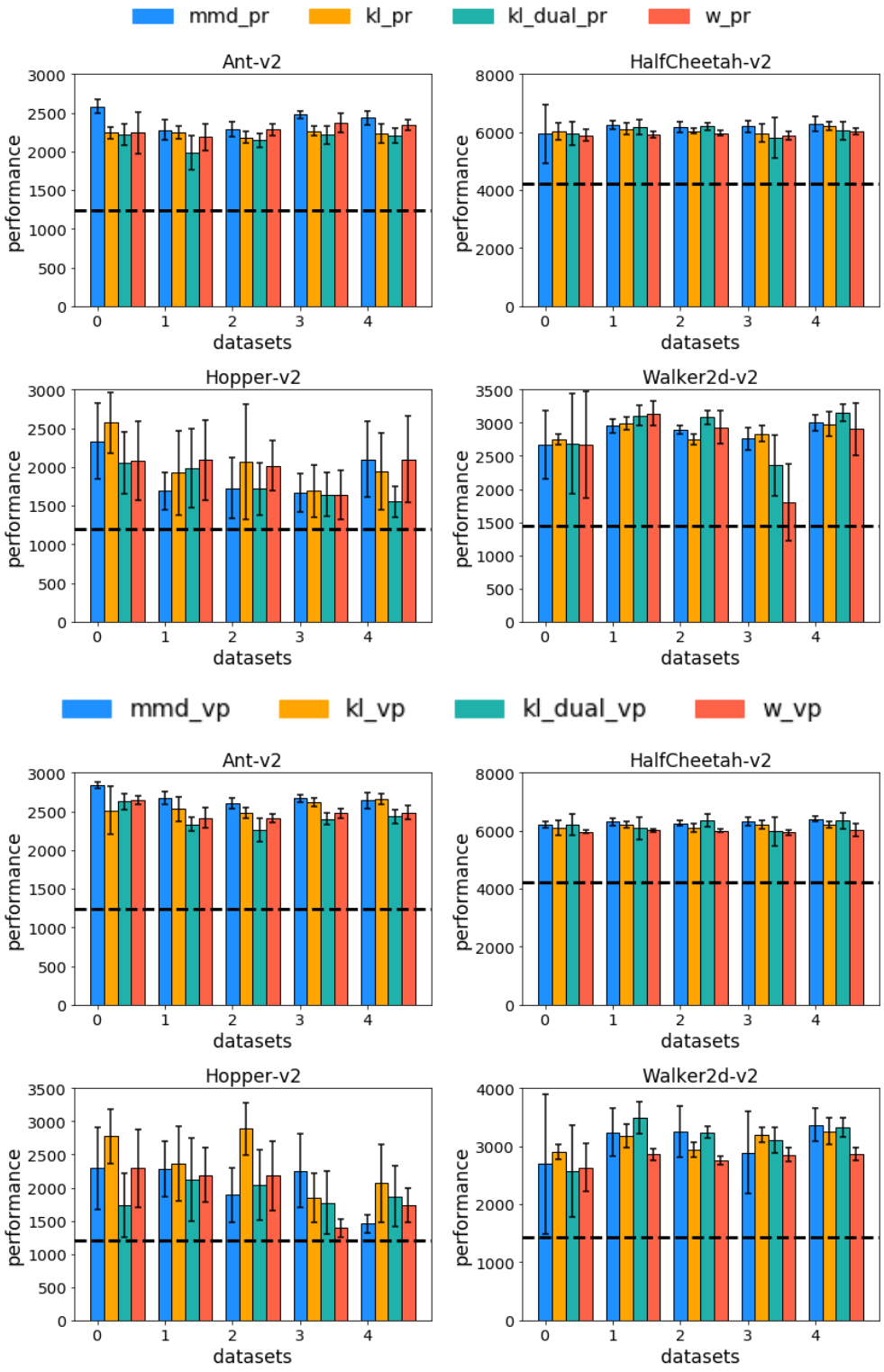


Figure 5.5: Comparing different divergences under both policy regularization (top row) and value penalty (bottom row). All variants yield similar performance, which is significantly better than the partially trained policy.

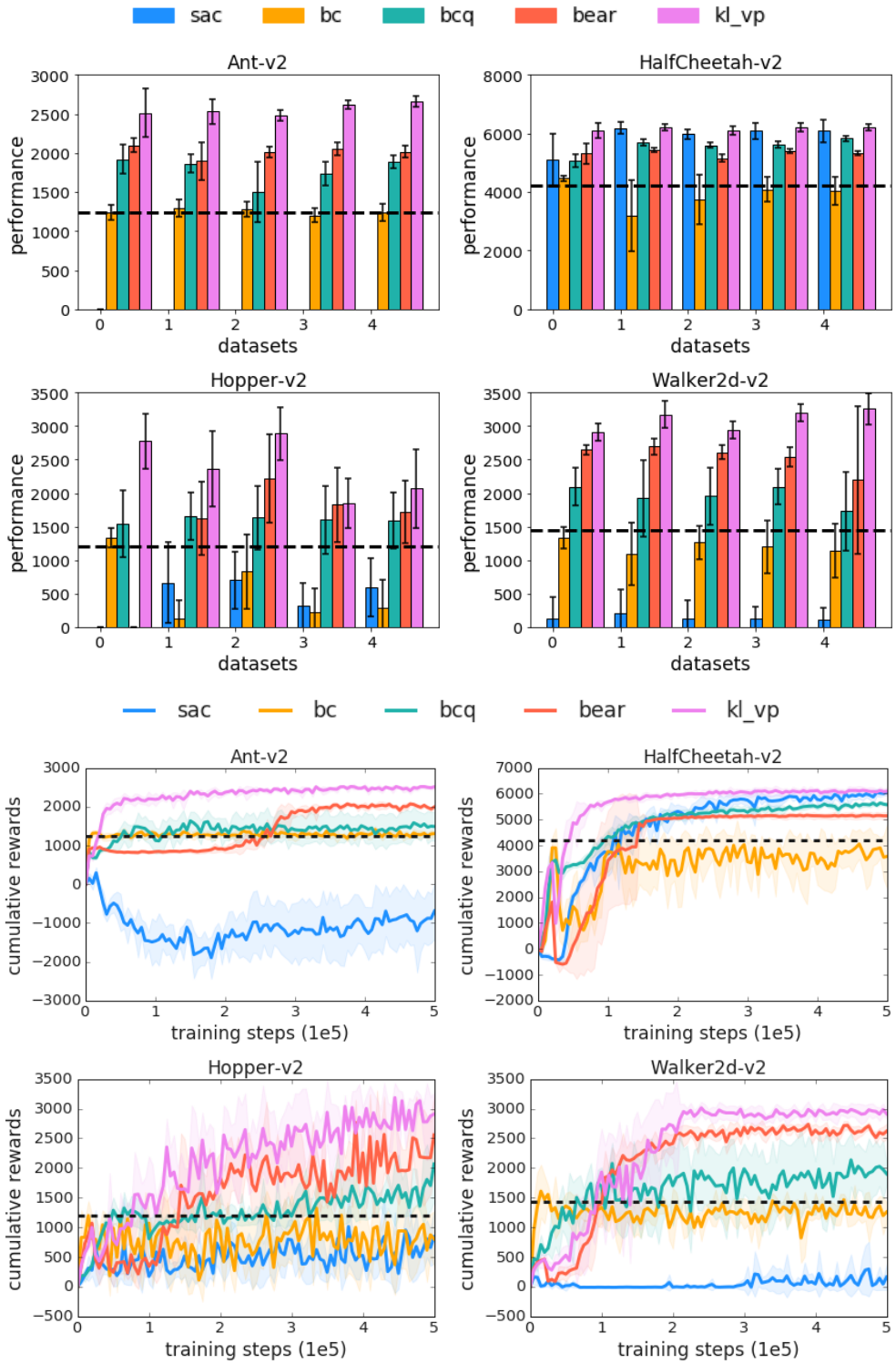


Figure 5.6: Comparing value penalty with KL divergence (kl_vp) to vanilla SAC, behavior cloning (bc), BCQ and BEAR. Bottom row shows sampled training curves with 1 out of the 5 datasets. See Appendix for training curves on all datasets.

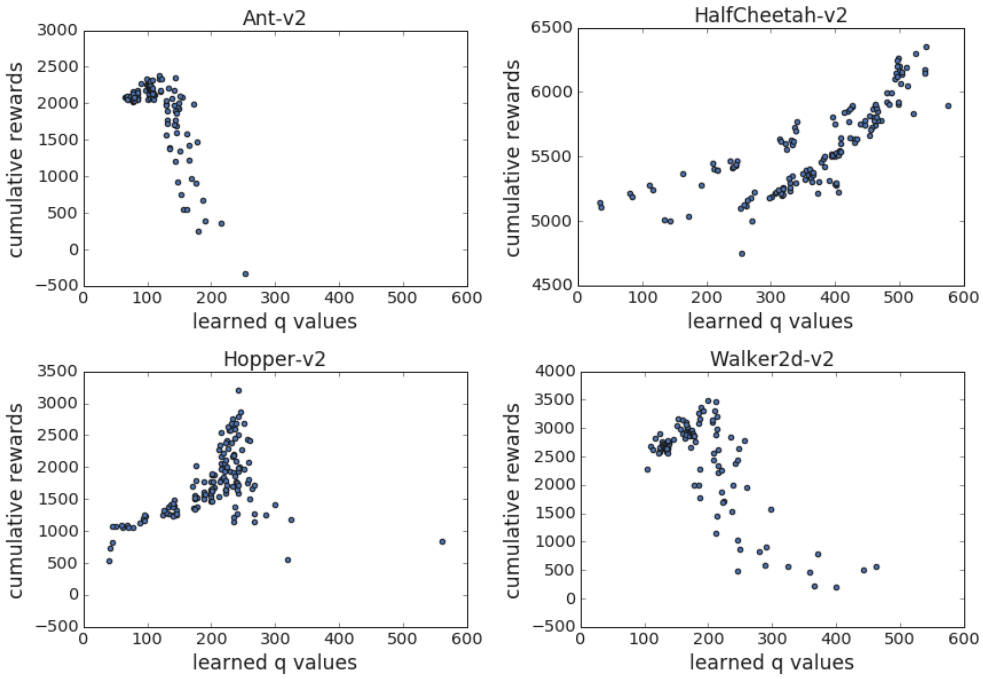


Figure 5.7: Correlation between learned Q-values and performance. x-axis is the average of learned $Q_\psi(s, a)$ over the last 500 training batches. y-axis is the average performance over the last 10 evaluation points. Each plot corresponds to a (environment, algorithm, dataset) tuple. Different points in each plot correspond to different hyperparameters and different random seeds.

Environment: Hopper-v2			**Partially trained policy: 1202**		
dataset	no-noise	eps-0.1	eps-0.3	gauss-0.1	gauss-0.3
SAC	0.2655	661.7	701	311.2	592.6
BC	1330	129.4	828.3	221.1	284.6
BCQ	1543	1652	1632	1599	1590
BEAR	0	1620	2213	1825	1720
MMD_vp	2291	2282	1892	**2255**	1458
KL_vp	**2774**	**2360**	**2892**	1851	2066
KL_dual_vp	1735	2121	2043	1770	1872
W_vp	2292	2187	2178	1390	1739
MMD_pr	2334	1688	1725	1666	**2097**
KL_pr	2574	1925	2064	1688	1947
KL_dual_pr	2053	1985	1719	1641	1551
W_pr	2080	2089	2015	1635	**2097**

Table 5.4: Evaluation results with tuned hyperparameters.

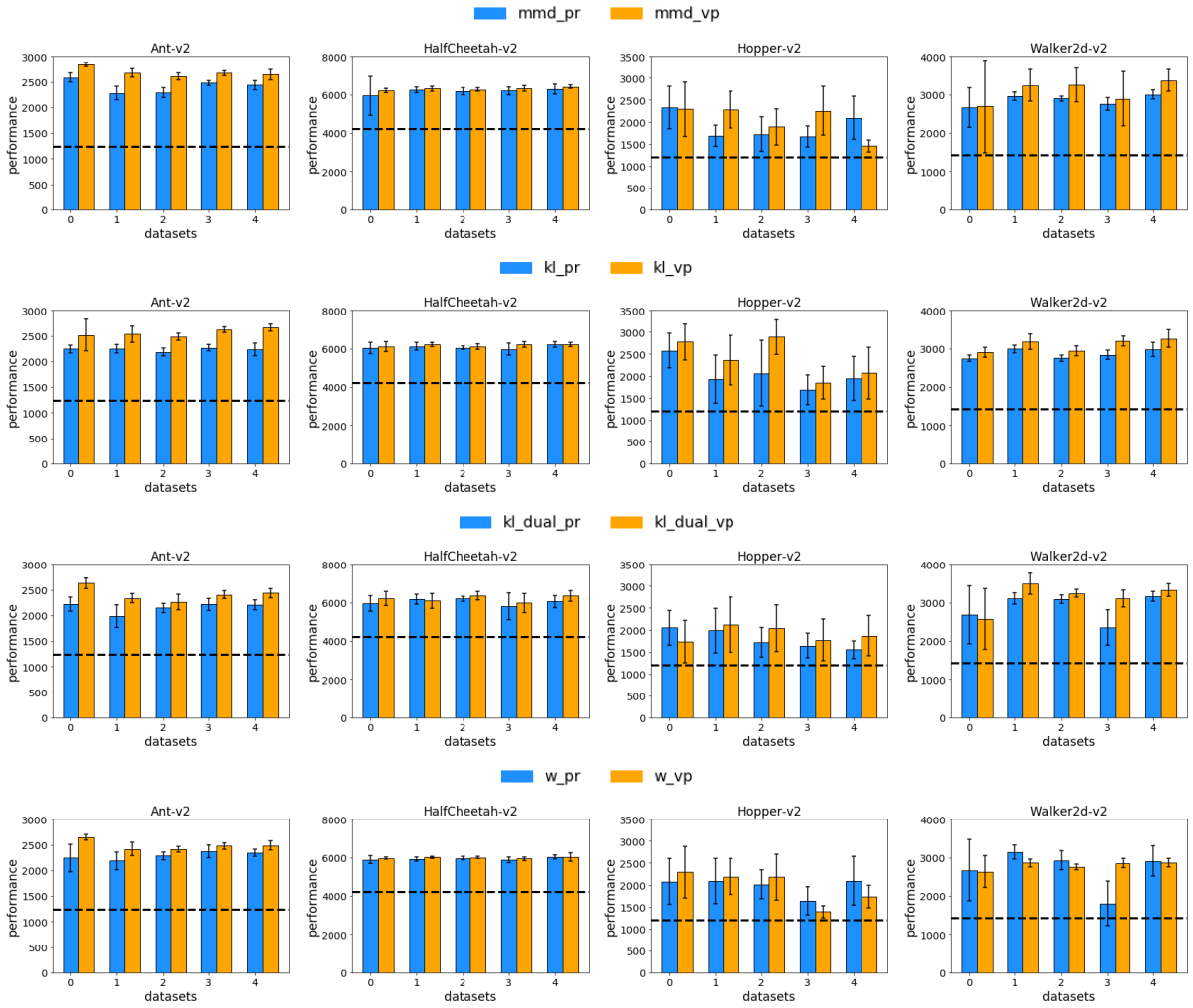


Figure 5.8: Comparing policy regularization (pr) v.s. value penalty (vp) with all four divergences. The use of value penalty is usually slightly better.

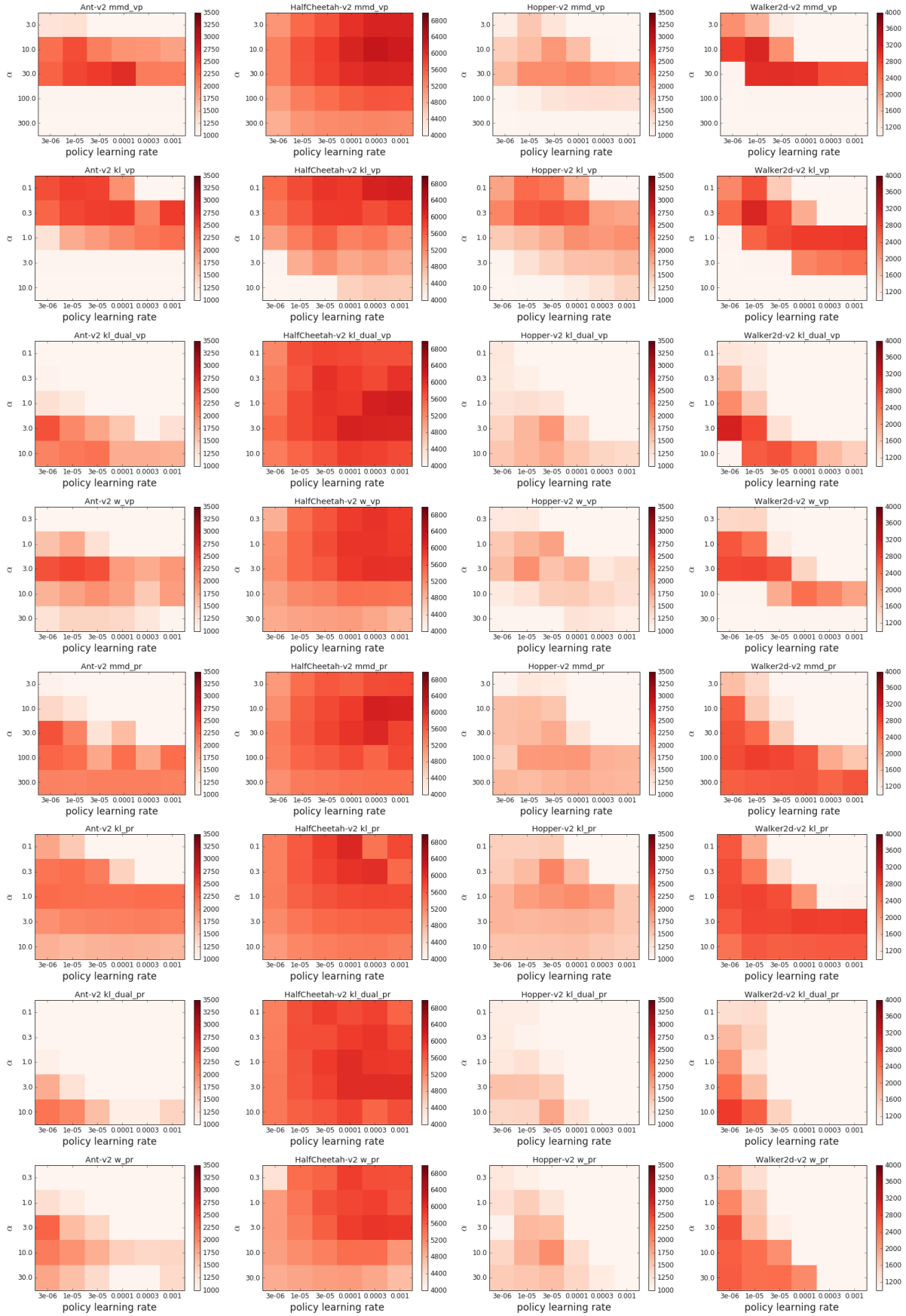


Figure 5.9: Visualization of performance under different hyperparameters. The performance is averaged over all five datasets.

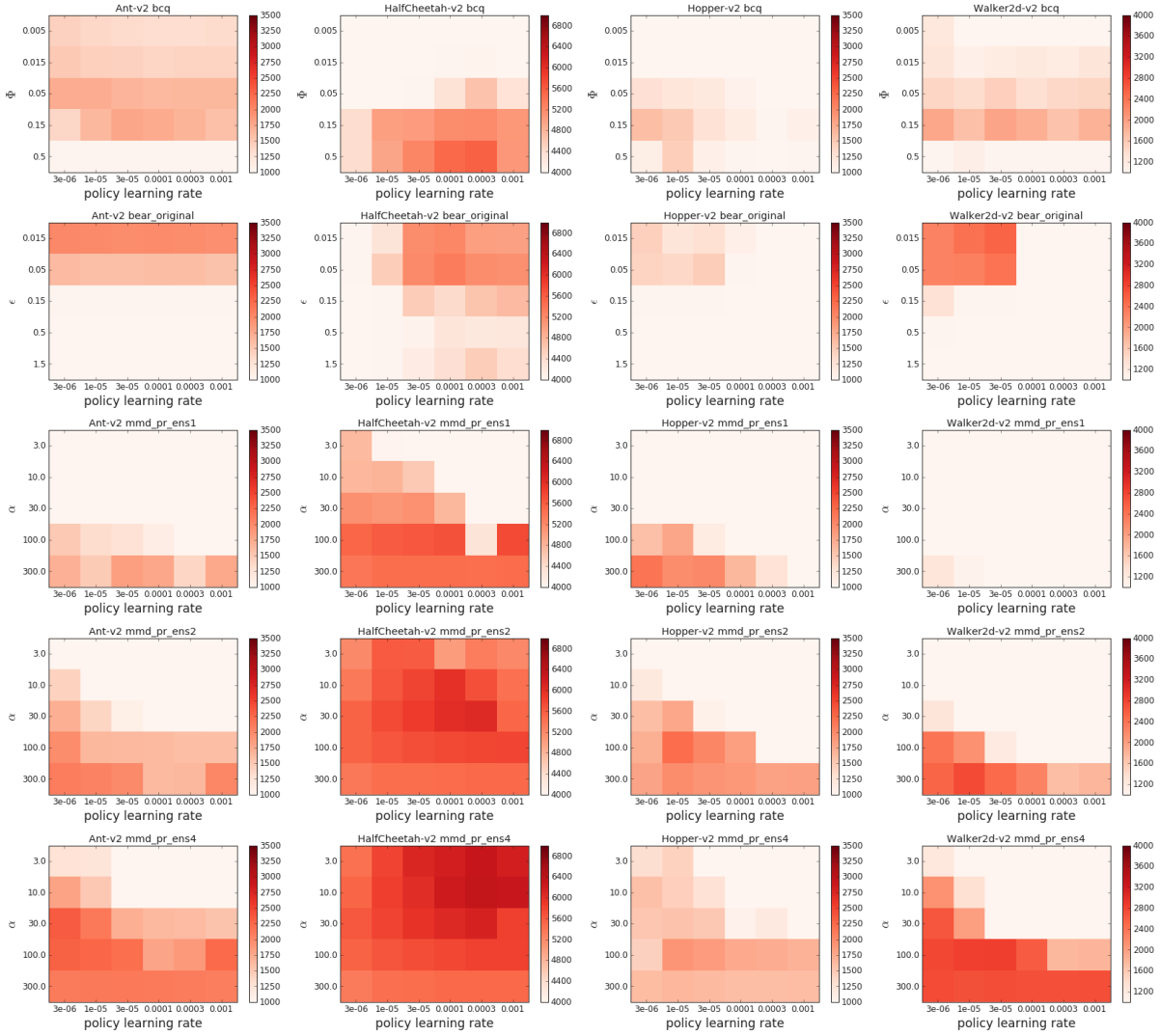


Figure 5.10: Visualization of performance under different hyperparameters.

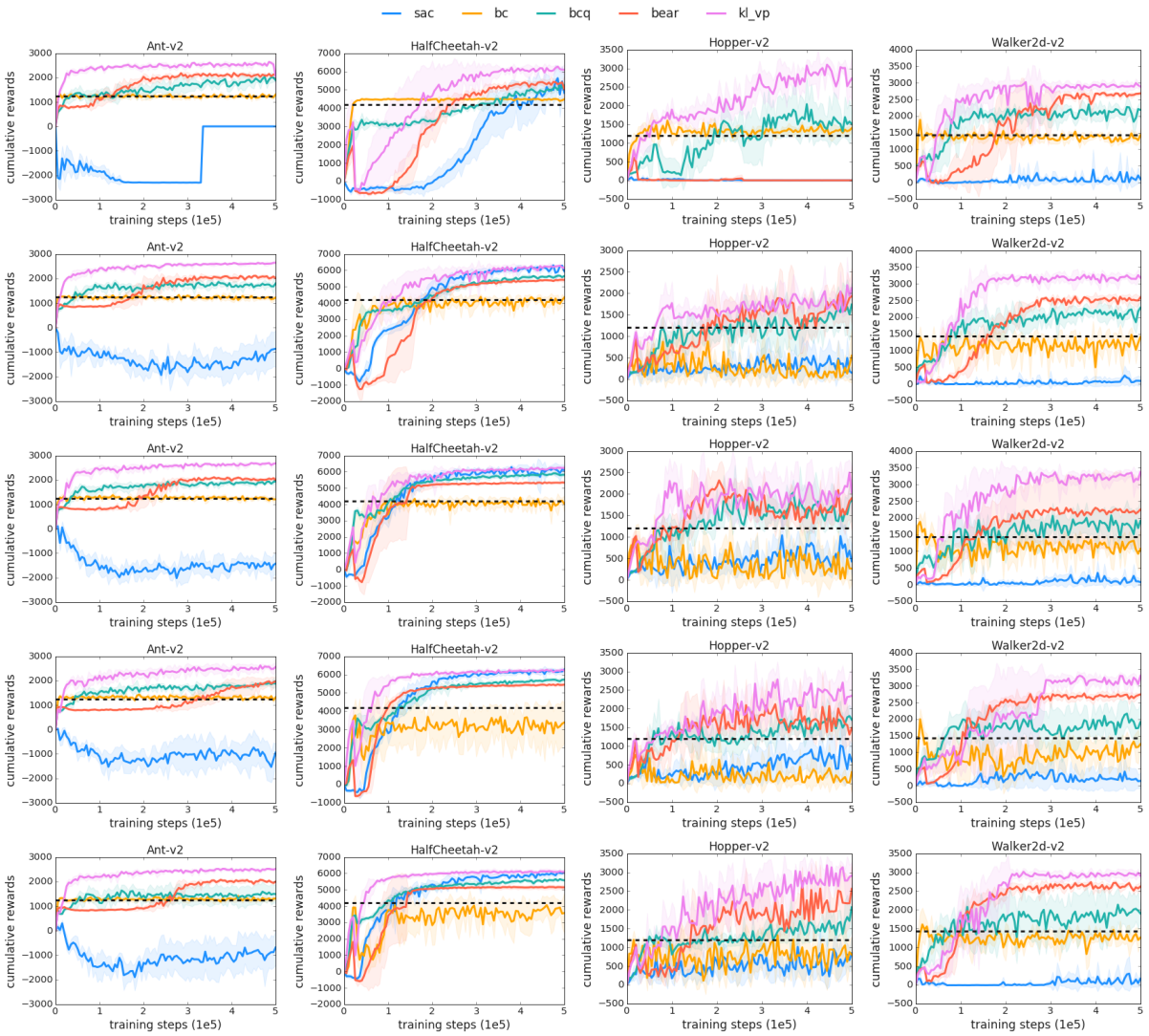


Figure 5.11: Training curves on all five datasets when comparing kl_vp to other baselines.

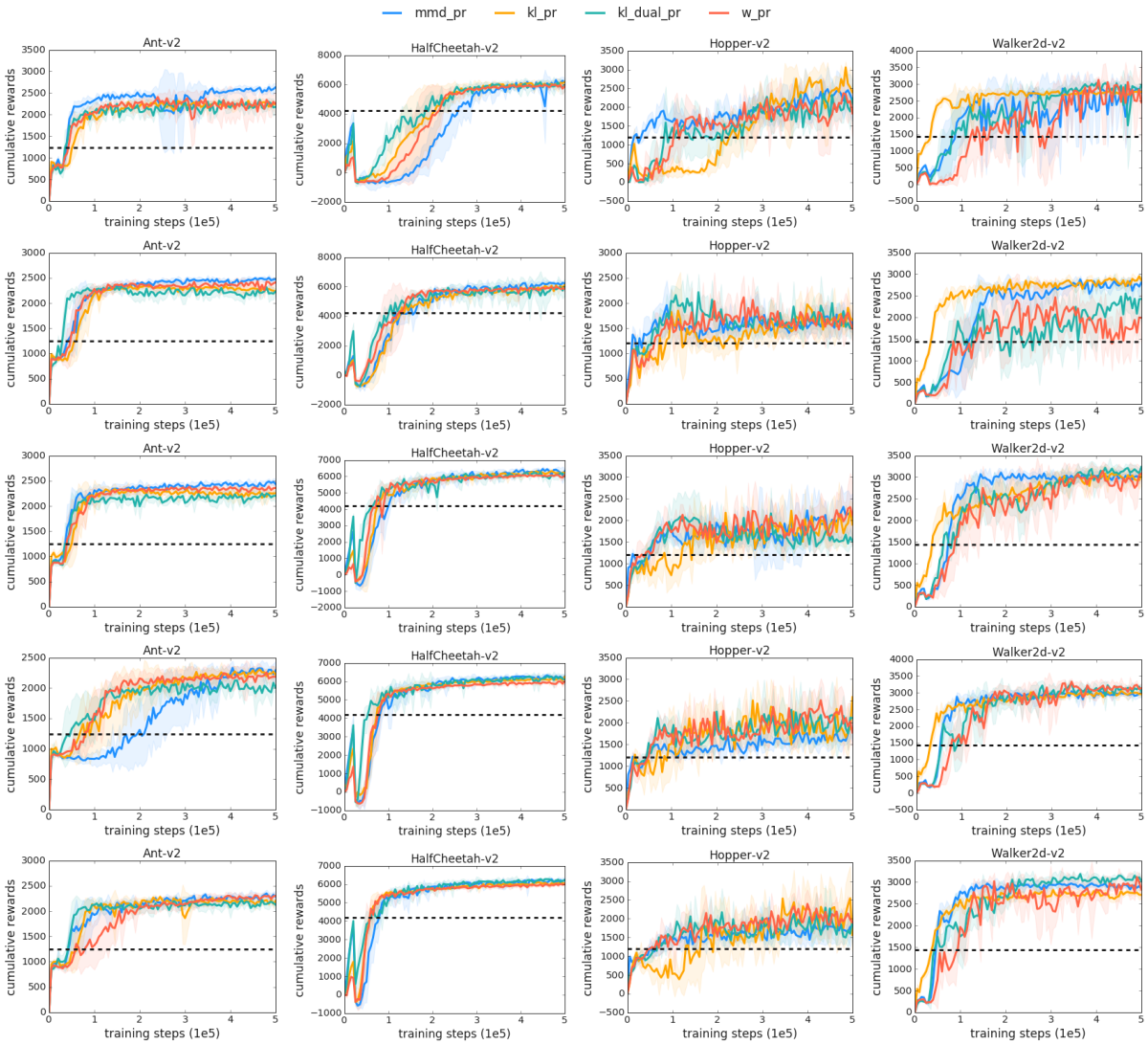


Figure 5.12: Training curves when comparing different divergences with policy regularization. All divergences perform similarly.

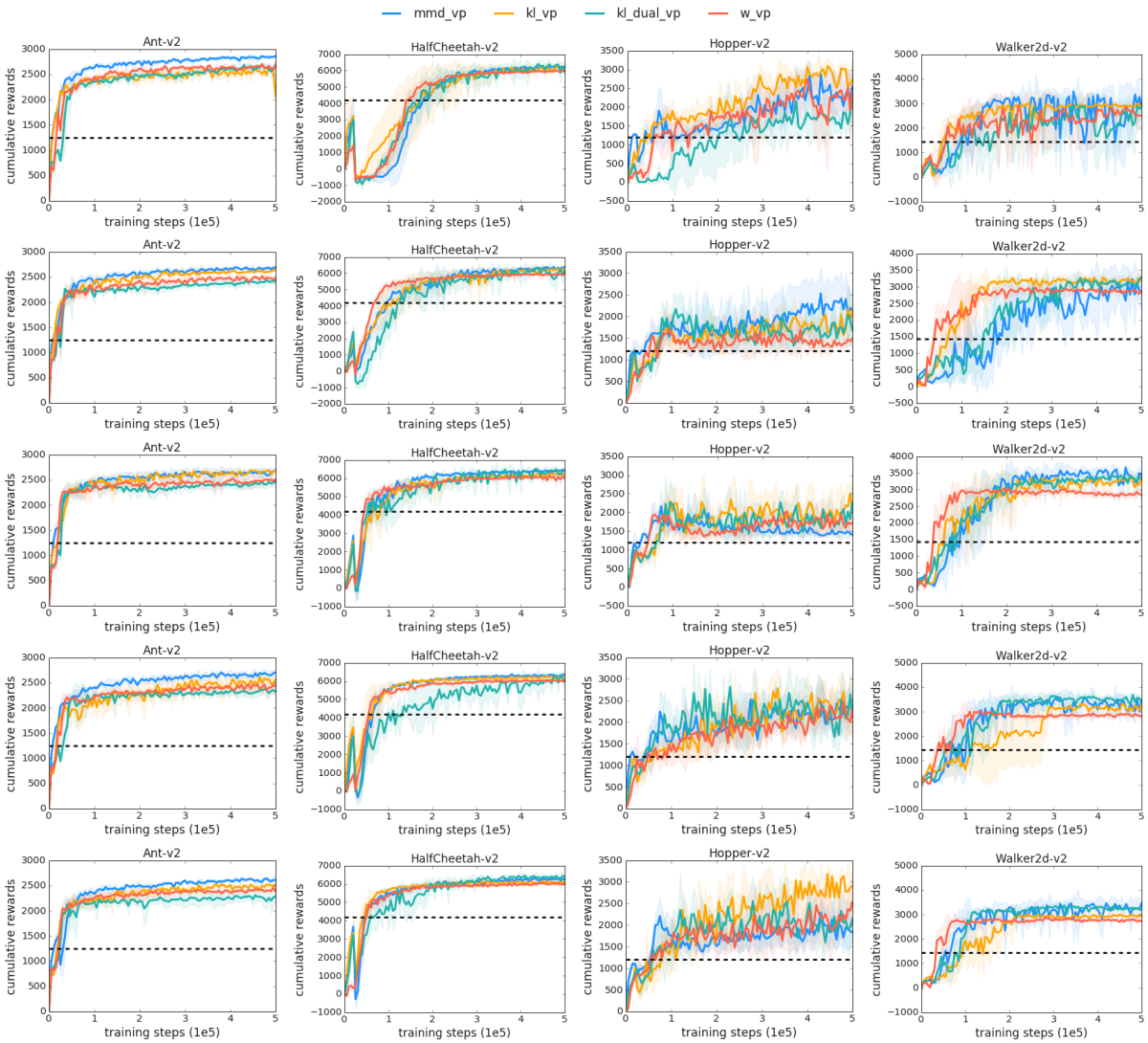


Figure 5.13: Training curves when comparing different divergences with value penalty. All divergences perform similarly.

Environment: Walker-v2			**Partially trained policy: 1439**		
dataset	no-noise	eps-0.1	eps-0.3	gauss-0.1	gauss-0.3
SAC	131.7	213.5	127.1	119.3	109.3
BC	1334	1092	1263	1199	1137
BCQ	2095	1921	1953	2094	1734
BEAR	2646	2695	2608	2539	2194
MMD_vp	2694	3241	**3255**	2893	**3368**
KL_vp	**2907**	3175	2942	**3193**	3261
KL_dual_vp	2575	**3490**	3236	3103	3333
W_vp	2635	2863	2758	2856	2862
MMD_pr	2670	2957	2897	2759	3004
KL_pr	2744	2990	2747	2837	2981
KL_dual_pr	2682	3109	3080	2357	3155
W_pr	2667	3140	2928	1804	2907

Table 5.5: Evaluation results with tuned hyperparameters.

Chapter 6

On the Optimality of Offline Policy Optimization Algorithms

6.1 Overview

We consider the problem of *offline (or batch) policy optimization*, where a learner must infer a behavior policy given only access to a fixed dataset of previously collected experience, with no further environment interaction available. Interest in this problem has grown recently, as effective solutions hold the promise of extracting powerful decision making strategies from years of logged experience, with important applications to many practical problems [25, 66, 89, 131, 134].

Despite the prevalence and importance of batch policy optimization, the theoretical understanding of this problem has, until recently, been rather limited. A fundamental challenge in batch policy optimization is the insufficient coverage of the dataset. In online reinforcement learning (RL), the learner is allowed to continually explore the environment to collect useful information for the learning tasks. By contrast, in the batch setting, the learner has to evaluate and optimize over various candidate policies based only on experience that has been collected a priori. The distribution mismatch between the logged experience and agent-environment interaction with a learned policy can cause erroneous value overestimation, which leads to the failure of standard policy optimization methods [44]. To overcome this problem, recent studies propose to use the *pessimistic principle*, by either learning a pessimistic value function [66, 77, 78, 134, 148] or pessimistic surrogate [19], or planning with a pessimistic model [70, 155]. However, it still remains unclear how to maximally exploit the logged experience without further exploration.

In this chapter, we investigate batch policy optimization with finite-armed stochastic bandits, and make three contributions toward better understanding the statistical limits of this problem. *First*, we prove a minimax lower bound of $\Omega(1/\sqrt{\min_i n_i})$ on the simple regret for batch policy optimization with stochastic bandits, where n_i is the number of times arm i was chosen in the dataset. We then introduce the notion of a confidence-adjusted index algorithm that unifies both the optimistic and pessimistic principles in a single algorithmic framework. Our analysis suggests that any index algorithm with an appropriate adjustment, whether pessimistic or optimistic, is minimax optimal.

Second, we analyze the instance-dependent regret of batch policy optimization algorithms. Perhaps surprisingly, our main result shows that instance-dependent optimality, which is commonly used in the literature of minimizing cumulative regret of stochastic bandits, does not exist in the batch setting. Together with our first contribution, this finding challenges recent theoretical findings in batch RL that claim pessimistic algorithms are an optimal choice [e.g., 19, 67]. In fact, our analysis suggests that for any algorithm that performs optimally in some environment, there must always exist another environment where the algorithm suffers arbitrarily larger regret than an optimal strategy there. Therefore, any reasonable algorithm is equally optimal, or not optimal, depending on the exact problem instance the algorithm is facing. In this sense, for batch policy optimization, there remains a lack of a well-defined optimality criterion that can be used to choose between algorithms.

Third, we provide a characterization of the pessimistic algorithm by introducing a weighted-minimax objective. In particular, the pessimistic algorithm can be considered to be optimal in the sense that it achieves a regret that is comparable to the inherent difficulty of optimal value prediction on an instance-by-instance basis. Overall, the theoretical study we provide consolidates recent research findings on the impact of being pessimistic in batch policy optimization [19, 67, 70, 78, 98, 155].

The remainder of the chapter is organized as follows. After defining the problem setup in Sections 6.2, we present the three main contributions in Sections 6.3 to 6.5 as aforementioned. Section 6.6 discusses the related works.

6.2 Problem setup

To simplify the exposition, we express our results for batch policy optimization in the setting of stochastic finite-armed bandits. In particular, assume the action space consists of $k > 0$ arms, where the available data takes the form of $n_i > 0$ real-valued observations $X_{i,1}, \ldots, X_{i,n_i}$ for each arm $i \in [k] := \{1, \ldots, k\}$. This data represents the outcomes of n_i pulls of each arm i. We assume further that the data for each arm i is *i.i.d.* with $X_{i,j} \sim P_i$ such that P_i is the reward distribution for arm i. Let $\mu_i = \int x P_i(dx)$ denote the mean reward that results from pulling arm i. All observations in the data set $X = (X_{i,j})_{i\in[k], j\in[n_i]}$ are assumed to be independent.

We consider the problem of designing an algorithm that takes the counts $(n_i)_{i\in[k]}$ and observations $X \in \times_{i\in[k]} \mathbb{R}^{n_i}$ as inputs and returns the index of a single arm in $[k]$, where the goal is to select an arm with the highest mean reward. Let $\mathcal{A}(X) \in [k]$ be the output of algorithm $\mathcal{A}$, Then the (simple) regret of $\mathcal{A}$ is defined as

$$\mathcal{R}(\mathcal{A}, \theta) = \mu^* - \mathbb{E}_{X\sim\theta}\left[\mu_{\mathcal{A}(X)}\right] ,$$

where $\mu^* = \max_i \mu_i$ is the maximum reward. Here, the expectation $\mathbb{E}\left[X \sim \theta\right]$ considers the randomness of the data X generated from problem instance θ, and also any randomness in the algorithm $\mathcal{A}$, which together induce the distribution of the random choice $\mathcal{A}(X)$. Note that this definition of regret depends both on the algorithm $\mathcal{A}$ and the problem instance $\theta = ((n_i)_{i\in[k]}, (P_i)_{i\in[k]})$. When θ is fixed, we will use $\mathcal{R}(\mathcal{A})$ to reduce clutter.

For convenience, we also let $n = \sum_i n_i$ and $n_{\min}$ denote the total number of observations and the minimum number of observations in the data. The optimal arm is a^* and the suboptimality

gap is $\Delta_i = \mu^* - \mu_i$. The largest and smallest non-zero gaps are $\Delta_{\max} = \max_i \Delta_i$ and $\Delta_{\min} = \min_{i:\Delta_i>0} \Delta_i$. In what follows, we assume that the distributions P_i are 1-subgaussian with means in the unit interval $[0, 1]$. We denote the set of these distributions by $\mathcal{P}$. The set of all instances where the distributions satisfy these properties is denoted by Θ. The set of instances with $\mathbf{n} = (n_i)_{i\in[k]}$ fixed is denoted by $\Theta_{\mathbf{n}}$. Thus, $\Theta = \cup_{\mathbf{n}}\Theta_{\mathbf{n}}$. Finally, we define $|\mathbf{n}| = \sum_i n_i$ for $\mathbf{n} = (n_i)_{i\in[k]}$.

6.3 Minimax Analysis

In this section, we introduce the notion of a *confidence-adjusted index algorithm*, and prove that a broad range of such algorithms are minimax optimal up to a logarithmic factor. A confidence-adjusted index algorithm is one that calculates an index for each arm based on the data for that arm only, then chooses an arm that maximizes the index. We consider index algorithms where the index of arm $i \in [k]$ is defined as the sum of the sample mean of this arm, $\hat{\mu}_i = \frac{1}{n_i}\sum_{j=1}^{n_i} X_{i,j}$ plus a bias term of the form $\alpha/\sqrt{n_i}$ with $\alpha \in \mathbb{R}$. That is, given the input data X, the algorithm selects an arm according to

$$\arg\max_{i\in[k]} \ \hat{\mu}_i + \frac{\alpha}{\sqrt{n_i}}\,. \tag{6.1}$$

The reason we call these confidence-adjusted is because for a given confidence level $\delta > 0$, by Hoeffding's inequality, it follows that

$$\mu_i \in \left[\hat{\mu}_i - \frac{\beta_\delta}{\sqrt{n_i}},\ \hat{\mu}_i + \frac{\beta_\delta}{\sqrt{n_i}}\right] \tag{6.2}$$

with probability at least $1 - \delta$ for all arms with

$$\beta_\delta = \sqrt{2\log\left(\frac{k}{\delta}\right)}\,.$$

Thus, the family of confidence-adjusted index algorithms consists of all algorithms that follow this strategy, where each particular algorithm is defined by a (data independent) choice of α. For example, an algorithm specified by $\alpha = -\beta_\delta$ chooses the arm with highest lower-confidence bound (highest LCB value), while an algorithm specified by $\alpha = \beta_\delta$ chooses the arm with the highest upper-confidence bound (highest UCB value). Note that $\alpha = 0$ corresponds to what is known as the *greedy* (sample mean maximizing) choice.

Readers familiar with the literature on batch policy optimization will recognize that $\alpha = -\beta_\delta$ implements what is known as the pessimistic algorithm [19, 67, 70, 153], or distributionally robust choice, or risk-adverse strategy. It is therefore natural to question the utility of considering batch policy optimization algorithms that *maximize* UCB values (i.e., implement optimism in the presence of uncertainty, or risk-seeking behavior, even when there is no opportunity for exploration). However, our first main result is that for batch policy optimization a risk-seeking (or greedy) algorithm cannot be distinguished from the more commonly proposed pessimistic approach in terms of minimax regret.

To establish this finding, we first provide a lower bound on the minimax regret:

Theorem 6.3.1. Fix $\mathbf{n} = (n_i)_{i\in[k]}$ with $n_1 \leq \cdots \leq n_k$. Then, there exists a universal constant $c > 0$ such that

$$\inf_{\mathcal{A}} \sup_{\theta\in\Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A},\theta) \geq c \max_{m\in[k]} \sqrt{\frac{\max(1,\log(m))}{n_m}}\,.$$

The assumption of increasing counts, $n_1 \leq \cdots \leq n_k$, is only needed to simplify the statement; the arm indices can always be re-ordered without loss of generality. The proof follows by arguing that the minimax regret is lower bounded by the Bayesian regret of the Bayesian optimal policy for any prior. Then, with a judicious choice of prior, the Bayesian optimal policy has a simple form. Intuitively, the available data permits estimation of the mean of action a with accuracy $O(\sqrt{1/n_a})$. The additional logarithmic factor appears when $n_1, \ldots, n_m$ are relatively close, in which case the lower bound is demonstrating the necessity of a union bound that appears in the upper bound that follows.

Next we show that a wide range of confidence-adjusted index algorithms are nearly minimax optimal when their confidence parameter is properly chosen:

Theorem 6.3.2. Fix $\mathbf{n} = (n_i)_{i\in[k]}$. Let δ be the solution of $\delta = \sqrt{32\log(k/\delta)/\min_i n_i}$, and $\mathcal{I}$ be the confidence-adjusted index algorithm with parameter α. Then, for any $\alpha \in [-\beta_\delta, \beta_\delta]$, we have

$$\sup_{\theta\in\Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{I}(\alpha),\theta) \leq 12\sqrt{\frac{\log(k/\delta)}{\min_i n_i}}\,.$$

Remark 6.3.3. Theorem 6.3.2 also holds for algorithms that use different $\alpha_i \in [-\beta_\delta, \beta_\delta]$ for different arms.

Perhaps a little unexpectedly, we see that *regardless* of optimism vs. pessimism, index algorithms with the right amount of adjustment, or *even no adjustment*, are minimax optimal, up to an order $\sqrt{\log(kn)}$ factor. We note that although these algorithms have the same worst case performance, they can behave very differently indeed on individual instances, as we show in the next section.

In effect, what these two results tell us is that minimax optimality is too weak as a criterion to distinguish between pessimistic versus optimistic (or greedy) algorithms when considering the "fixed count" setting of batch policy optimization. This leads us to ask whether more refined optimality criteria are able to provide nontrivial guidance in the selection of batch policy optimization methods. One such criterion, considered next, is known as instance-optimality in the literature of cumulative regret minimization for stochastic bandits.

6.4 Instance-Dependent Analysis

To better distinguish between algorithms we require a much more refined notion of performance that goes beyond merely considering worst-case behavior over all problem instances. Even if two algorithms have the same worst case performance, they can behave very differently on individual instances. Therefore, we consider the instance dependent performance of confidence-adjusted index algorithms.

6.4.1 Instance-dependent Upper Bound

Our next result provides a regret upper bound for a general form of index algorithm. All upper bounds in this section hold for any $\theta \in \Theta_{\mathbf{n}}$ unless otherwise specified, and we use $\mathcal{R}(\mathcal{A})$ instead of $\mathcal{R}(\mathcal{A}, \theta)$ to simplify the notation.

Theorem 6.4.1. Consider a general form of index algorithm, $\mathcal{A}(X) = \arg\max_i \hat{\mu}_i + b_i$, where b_i denotes the bias for arm $i \in [k]$ specified by the algorithm. For $2 \leq i \leq k$ and $\eta \in \mathbb{R}$, define

$$g_i(\eta) = \sum_{j \geq i} e^{-\frac{n_j}{2}(\eta - \mu_j - b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \eta)_+^2}$$

and $g_i^* = \min_\eta g_i(\eta)$. Assuming $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k$, for the index algorithms (6.1) we have

$$\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq \min\{1, g_i^*\} \tag{6.3}$$

and

$$\mathcal{R}(\mathcal{A}) \leq \sum_{2 \leq i \leq k} \Delta_i \left(\min\{1, g_i^*\} - \min\{1, g_{i+1}^*\}\right) \tag{6.4}$$

where we define $g_{k+1}^* = 0$.

The assumption $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_k$ is only required to express the statement simply; the indices can be reordered without loss of generality. The expression in (6.3) is a bit difficult to work with, so to make the subsequent analysis simpler we provide a looser but more interpretable bound for general index algorithms as follows.

Corollary 6.4.2. Following the setting of Theorem 6.4.1, consider any index algorithm and any $\delta \in (0, 1)$. Define $U_i = \mu_i + b_i + \beta_\delta/\sqrt{n_i}$ and $L_i = \mu_i + b_i - \beta_\delta/\sqrt{n_i}$. Let $h = \max\{i \in [k] : \max_{j<i} L_j < \max_{j' \geq i} U_{j'}\}$. Then we have

$$\begin{aligned}\mathcal{R}(\mathcal{A}) &\leq \Delta_h + \frac{\delta}{k}\Delta_{\max} \\ &+ \frac{\delta}{k}\sum_{i>h}(\Delta_i - \Delta_{i-1})\sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j\right)^2}.\end{aligned}$$

Remark 6.4.3. The upper bound in Corollary 6.4.2 can be further relaxed as $\mathcal{R}(\mathcal{A}) \leq \Delta_h + \delta\Delta_{\max}$.

Remark 6.4.4. The minimax regret upper bound (Theorem 6.3.2) can be recovered a result of Corollary 6.4.2 (see supplement).

Corollary 6.4.2 highlights an inherent optimization property of index algorithms: they work by designing an additive adjustment for each arm, such that all of the bad arms ($i > h$) can be eliminated efficiently, i.e., it is desirable to make h as small as possible. We note that although one can directly plug in the specific choices of $\{b_i\}_{i \in [k]}$ to get instance-dependent upper bounds for different algorithms, it is not clear how their performance compares to one another. Therefore, we provide simpler relaxed upper bounds for the three specific cases, greedy, LCB and UCB, to allow us to better differentiate their performance across different problem instances (see supplement for details).

Corollary 6.4.5 (Regret Upper bound for Greedy)**.** Following the setting of Theorem 6.4.1, for any $0 < \delta < 1$, the regret of greedy ($\alpha = 0$) on any problem instance is upper bounded by

$$\mathcal{R}(\mathcal{A}) \leq \min_{i \in [k]} \left(\Delta_i + \sqrt{\frac{2}{n_i} \log \frac{k}{\delta}} + \max_{j > i} \sqrt{\frac{2}{n_j} \log \frac{k}{\delta}} \right) + \delta \,.$$

Corollary 6.4.6 (Regret Upper bound for LCB)**.** Following the setting of Theorem 6.4.1, for any $0 < \delta < 1$, the regret of LCB ($\alpha = -\beta_\delta$) on any problem instance is upper bounded by

$$\mathcal{R}(\mathcal{A}) \leq \min_{i \in [k]} \Delta_i + \sqrt{\frac{8}{n_i} \log \frac{k}{\delta}} + \delta \,.$$

Corollary 6.4.7 (Regret Upper bound for UCB)**.** Following the setting of Theorem 6.4.1, for any $0 < \delta < 1$, the regret of UCB ($\alpha = \beta_\delta$) on any problem instance is upper bounded by

$$\mathcal{R}(\mathcal{A}) \leq \min_{i \in [k]} \left(\Delta_i + \max_{j > i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \right) + \delta \,.$$

Remark 6.4.8. The results in these corollaries sacrifice the tightness of instance-dependence to obtain cleaner bounds for the different algorithms. The tightest instance dependent bounds can be derived from Theorem 6.4.1 by optimizing η.

Discussion. The regret upper bounds presented above suggest that although they are all nearly minimax optimal, UCB, LCB and greedy exhibit distinct behavior on individual instances. Each will eventually select the best arm with high probability when n_i gets large for *all* $i \in [k]$, but their performance can be very different when n_i gets large for only a *subset* of arms $S \subset [k]$. For example, LCB performs well whenever S contains a good arm (i.e., with small Δ_i and large n_i). UCB performs well when there is a good arm i such that all worse arms are in S (n_j large for all $j > i$). For the greedy algorithm, the regret upper bound is small only when there is a good arm i where n_j is large for all $j \geq i$, in which situation both LCB and UCB perform well.

Clearly there are instances where LCB performs much better than UCB and vice versa. Consider an environment where there are two groups of arms: one with higher rewards and another with lower rewards. The behavior policy plays a subset of the arms $S \subset [k]$ a large number of times and ignores the rest. If S contains at least one good arm but no bad arm, LCB will select a good played arm (with high probability) while UCB will select a bad unplayed arm. If S consists of all bad arms, then LCB will select a bad arm by being pessimistic about the unobserved good arms while UCB is guaranteed to select a good arm by being optimistic.

This example actually raises a potential reason to favor LCB, since the condition for UCB to outperform LCB is stricter: requiring the behavior policy to play all bad arms while ignoring all good arms. To formalize this, we compare the upper bounds for the two algorithms by taking the n_i for a subset of arms $i \in S \subset [k]$ to infinity. For $\mathcal{A} \in \{\text{greedy}, \text{LCB}, \text{UCB}\}$, let $\hat{\mathcal{R}}_S(\mathcal{A})$ be the regret upper bounds with $\{n_i\}_{i \in S} \to \infty$ and $\{n_i\}_{i \notin S} = 1$ while fixing $\mu_1, ..., \mu_k$ in Corollary 6.4.5, 6.4.6, and 6.4.7 respectively. Then LCB dominates the three algorithms with high probability under a uniform prior for S:

Proposition 6.4.9. Suppose $\mu_1 > \mu_2 > ... > \mu_k$ and $S \subset [k]$ is uniformly sampled from all subsets with size $m < k$, then

$$\mathbb{P}\left(\hat{\mathcal{R}}_S(\text{LCB}) < \hat{\mathcal{R}}_S(\text{UCB})\right) \geq 1 - \frac{(k-m)!m!}{k!}.$$

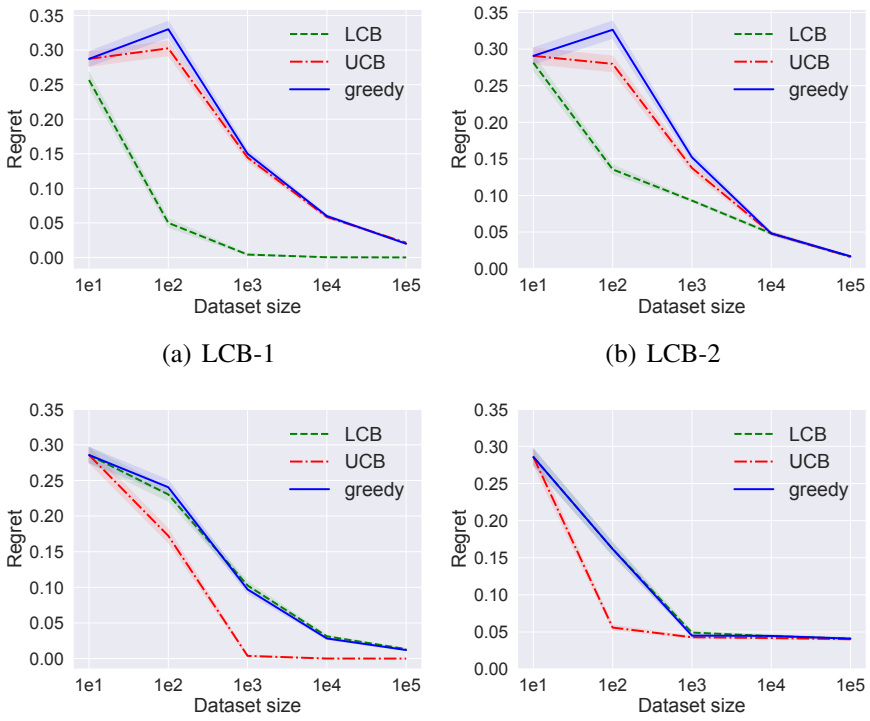
This lower bound is $1/2$ when $k = 2$ and approaches 1 when k increases for any $0 < m < k$ since it is always lower bounded by $1 - 1/k$. The same argument applies when comparing LCB to greedy.

To summarize, when comparing different algorithms by their upper bounds, we have the following observations: (i) These algorithms behave differently on different instances, and none of them outperforms the others on all instances. (ii) Both scenarios where LCB is better and scenarios where UCB is better exist. (iii) LCB is more favorable when k is not too small because it is the best option among these algorithms on most of the instances.

Simulation results. Since our discussion is based on comparing only the upper bounds (instead of the exact regret) for different algorithms, it is a question that whether these statements still hold in terms of their actual performance. To answer this question, we verify these statements through experiments on synthetic problems. The details of these synthetic experiments can be found in the supplementary material.

We first verify that there exist instances where LCB is the best among the three algorithms as well as instances where UCB is the best. For LCB to perform well, we construct two ϵ-greedy behavior policies on a 100-arm bandit where the best arm or a near-optimal arm is selected to be played with a high frequency while the other arms are uniformly played with a low frequency. Figure 6.1(a) and 6.1(b) show that LCB outperforms UCB and greedy on these two instances, verifying our observation from the upper bound (Corollary 6.4.6) that LCB only requires a good behavior policy while UCB and greedy require bad arms to be eliminated (which is not the case for ϵ-greedy policies). For UCB to outperform LCB, we set the behavior policy to play a set of near-optimal arms with only a small number of times and play the rest of the arms uniformly. Figure 6.1(c) and 6.1(d) show that UCB outperforms LCB and greedy on these two instances, verifying our observation from the upper bound (Corollary 6.4.7) that UCB only requires all worse arms to be identified.

We now verify the statement that LCB is the best option on most of the instances when k is not too small. We verify this statement in two aspects: First, we show that when $k = 2$, LCB and UCB have an equal chance to be the better algorithm. More specifically, we fix $n_1 > n_2$ (note that if $n_1 = n_2$ all index algorithms are the same as greedy) and vary $\mu_1 - \mu_2$ from -1 to 1. Intuitively, when $|\mu_1 - \mu_2|$ is large, the problem is relatively easy for all algorithms. For $\mu_1 - \mu_2$ in the medium range, as it becomes larger, the good arm is tried more often, thus the problem becomes easier for LCB and harder for UCB. Figure 6.2(a) and 6.2(b) confirm this and show that both LCB and UCB are the best option on half of the instances. Second, we show that as k grows, LCB quickly becomes the more favorable algorithm, outperforming UCB and greedy on an increasing fraction of instances. More specifically, we vary k and sample a set of instances from the prior distribution introduced in Proposition 6.4.9 with $|S| = k/2$ and $|S| = k/4$. Figure 6.2(c) and 6.2(d) shows that the fraction of instances where LCB is the best quickly approaches 1 as k increases.



(a) LCB-1 (b) LCB-2

(c) UCB-1 (d) UCB-2

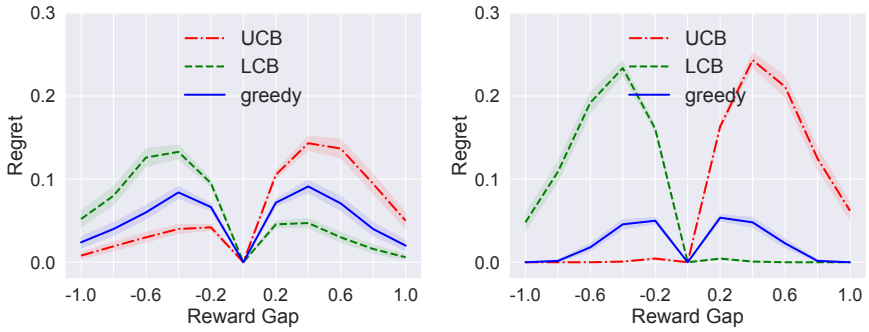
Figure 6.1: Comparing UCB, LCB and greedy on synthetic problems (with $k = 100$). (a) and (b): Problem instances where LCB has the best performance. The data set is generated by a behavior policy that pulls an arm i with *high* frequency and the other arms uniformly. In (a) i is the best arm while in (b) i is the 10th-best arm. (c) and (d): Problem instances where UCB has the best performance. The data set is generated by a behavior policy that pulls a set of good arms $\{j : j \leq i\}$ with very *small* frequency and the other arms uniformly. In (c) we use $i = 1$ while in (d) we use $i = 10$. Experiment details are provided in the supplementary material.

6.4.2 Instance-dependent Lower Bound

We have established that, despite all being minimax optimal, index algorithms with different adjustment can exhibit very different performance on specific problem instances. One might therefore wonder if instance optimal algorithms exist for batch policy optimization with finite-armed stochastic bandits. To answer this question, we next show that there is no instance optimal algorithm in the batch optimization setting for stochastic bandits, which is a very different outcome from the setting of cumulative regret minimization for online stochastic bandits.

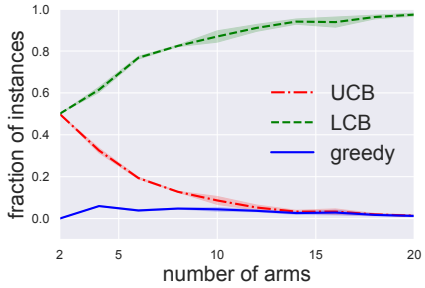
For cumulative regret minimization, Lai and Robbins [79] introduced an asymptotic notion of instance optimality [85]. The idea is to first remove algorithms that are insufficiently adaptive, then define a yardstick (or benchmark) for each instance as the best (normalized) asymptotic performance that can be achieved with the remaining adaptive algorithms. An algorithm that meets this benchmark over all instances is then considered to be an instance optimal algorithm.

When adapting this notion of instance optimality to the batch setting there are two decisions that need to be made: what is an appropriate notion of "sufficient adaptivity" and whether, of

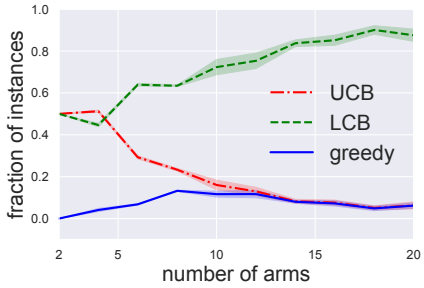


(a) $k = 2, n_1/n_2 = 2$

(b) $k = 2, n_1/n_2 = 10$



(c) $|S| = k/2$



(d) $|S| = k/4$

Figure 6.2: Comparing UCB, LCB and greedy on synthetic problems. (a) and (b): A set of two-armed bandit instances where both LCB and UCB dominate half of the instances. (c) and (d): For each k, we first sample 100 vectors $\vec{\mu} = [\mu_1, ..., \mu_k]$ and for each $\vec{\mu}$ we uniformly sample 100 (if exist) subsets $S \subset k, |S| = m$ ($m = k/2$ in (c) and $m = k/4$ in (d)), to generate up to 10k instances. We then count the fraction of instances where each algorithm performs better than the other two algorithms among the randomly sampled set of instances. Experiment details are provided in the supplementary material.

course, a similar asymptotic notion is sought or optimality can be adapted to the finite sample setting. Here, we consider the asymptotic case, as one usually expects this to be easier.

We consider the 2-armed bandit case ($k = 2$) with Gaussian reward distributions $\mathcal{N}(\mu_1, 1)$ and $\mathcal{N}(\mu_2, 1)$ for each arm respectively. Recall that, in this setting, fixing $\mathbf{n} = (n_1, n_2)$ each instance $\theta \in \Theta_{\mathbf{n}}$ is defined by (μ_1, μ_2). We assume that algorithms only make decisions based on the sufficient statistic — empirical means for each arm, which in this case reduces to $X = (X_1, X_2, \mathbf{n})$ with $X_i \sim \mathcal{N}(\mu_i, 1/n_i)$.

To introduce an asymptotic notion, we further denote $n = n_1 + n_2$, $\pi_1 = n_1/n$, and $\pi_2 = n_2/n = 1 - \pi_1$. Assume $\pi_1, \pi_2 > 0$; then each $\mathbf{n}$ can be uniquely defined by (n, π_1) for $\pi_1 \in (0, 1)$. We also ignore the fact that n_1 and n_2 should be integers since we assume the algorithms can only make decisions based on the sufficient statistic $X_i \sim \mathcal{N}(\mu_i, 1/n_i)$, which is well defined even when n_i is not an integer.

Definition 6.4.10 (Minimax Optimality)**.** Given a constant $c \geq 1$, an algorithm is said to be minimax optimal if its worst case regret is bounded by the minimax value of the problem up to a

multiplicative factor c. We define the set of minimax optimal algorithms as

$$\mathcal{M}_{\mathbf{n},c} = \left\{ \mathcal{A} : \sup_{\theta \in \Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}, \theta) \leq c \cdot \inf_{\mathcal{A}'} \sup_{\theta \in \Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}', \theta) \right\} .$$

Definition 6.4.11 (Instance-dependent Lower Bound)**.** Given a set of algorithms $\mathcal{M}$, for each $\theta \in \Theta_{\mathbf{n}}$, we define the instance-dependent lower bound as $\mathcal{R}^*_{\mathcal{M}}(\theta) = \inf_{\mathcal{A} \in \mathcal{M}} \mathcal{R}(\mathcal{A}, \theta)$.

The following theorem states the non-existence of instance optimal algorithms up to a constant multiplicative factor.

Theorem 6.4.12. Let c_0 be the constant in minimax lower bound such that $\inf_{\mathcal{A}} \sup_{\theta \in \Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}, \theta) \geq c_0/\sqrt{n_{\min}}$. Then for any $c > 2/c_0$ and any algorithm $\mathcal{A}$, we have

$$\sup_{\theta \in \Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A}, \theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} \geq \frac{n_{\min}}{n_{\min} + 4} e^{\frac{\beta^2}{4} + \frac{\beta}{4}\sqrt{n_{\min}}}$$

where $\beta = cc_0 - 2$.

Corollary 6.4.13. There is no algorithm that is instance optimal up to a constant multiplicative factor. That is, fixing $\pi_1 \in (0, 1)$, given any $c > 2/c_0$ and for any algorithm $\mathcal{A}$, we have

$$\limsup_{n \to \infty} \sup_{\theta \in \Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A}, \theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} = +\infty .$$

The proof of Theorem 6.4.12 follows by constructing two competing instances where the performance of any single algorithm cannot simultaneously match the performance of the adapted algorithm on each specific instance.

Step 1, define the algorithm $\mathcal{A}_\beta$ as

$$\mathcal{A}_\beta(X) = \begin{cases} 1 & \text{if } X_1 - X_2 \geq \frac{\beta}{\sqrt{n_{\min}}} \\ 2 & \text{otherwise} \end{cases} .$$

For any β within a certain range, it can be shown that $\mathcal{A}_\beta \in \mathcal{M}_{\mathbf{n},c}$, hence $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta) \leq \mathcal{R}(\mathcal{A}_\beta, \theta)$.

Step 2, construct two problem instances as follows. Fix a $\lambda \in \mathbb{R}$ and $\eta > 0$, and define

$$\theta_1 = (\mu_1, \mu_2) = (\lambda + \frac{\eta}{n_1}, \lambda - \frac{\eta}{n_2}) ,$$

$$\theta_2 = (\mu_1', \mu_2') = (\lambda - \frac{\eta}{n_1}, \lambda + \frac{\eta}{n_2}) .$$

Since we have $X_1 - X_2 \sim \mathcal{N}(\Delta, \sigma^2)$ on instance θ_1 and $X_1 - X_2 \sim \mathcal{N}(-\Delta, \sigma^2)$ on instance θ_2, where $\Delta = (\frac{1}{n_1} + \frac{1}{n_2})\eta$ and $\sigma^2 = \frac{1}{n_1} + \frac{1}{n_2}$, the regret of $\mathcal{A}_\beta$ on both instances can be computed using the CDF of Gaussian distributions. Note that $\mathcal{R}(\mathcal{A}_{-\beta}, \theta_1) = \mathcal{R}(\mathcal{A}_\beta, \theta_2)$. We now chose a $\beta_1 < 0$ for θ_1 to upper bound $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_1)$ by $\mathcal{R}(\mathcal{A}_{\beta_1}, \theta_1)$ and use $\beta_2 = -\beta_1 > 0$ to upper bound $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_2)$ by $\mathcal{R}(\mathcal{A}_{\beta_2}, \theta_1)$.

Then applying the Neyman-Pearson Lemma [108] to this scenario gives that $\mathcal{A}_0$ is the optimal algorithm in terms of balancing the regret on θ_1 and θ_2:

$$\mathcal{R}(\mathcal{A}_0, \theta_1) = \mathcal{R}(\mathcal{A}_0, \theta_2) = \min_{\mathcal{A}} \max\{\mathcal{R}(\mathcal{A}, \theta_1), \mathcal{R}(\mathcal{A}, \theta_2)\} .$$

Step 3, combining the above results gives

$$\begin{aligned}\sup_{\theta\in\Theta_{\mathbf{n}}}\frac{\mathcal{R}(\mathcal{A},\theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} &\geq \max\left\{\frac{\mathcal{R}(\mathcal{A},\theta_1)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_1)},\frac{\mathcal{R}(\mathcal{A},\theta_2)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_2)}\right\}\\ &\geq \max\left\{\frac{\mathcal{R}(\mathcal{A},\theta_1)}{\mathcal{R}(\mathcal{A}_{\beta_1},\theta_1)},\frac{\mathcal{R}(\mathcal{A},\theta_2)}{\mathcal{R}(\mathcal{A}_{\beta_2},\theta_2)}\right\}\\ &= \frac{\max\left\{\mathcal{R}(\mathcal{A},\theta_1),\mathcal{R}(\mathcal{A},\theta_2)\right\}}{\mathcal{R}(\mathcal{A}_{\beta_1},\theta_1)}\\ &\geq \frac{\mathcal{R}(\mathcal{A}_0,\theta_1)}{\mathcal{R}(\mathcal{A}_{\beta_1},\theta_1)}.\end{aligned}$$

Note that both the regret $\mathcal{R}(\mathcal{A}_0,\theta_1)$ and $\mathcal{R}(\mathcal{A}_{\beta_1},\theta_1)$ can be exact expressed as CDFs of Gaussian distributions: $\mathcal{R}(\mathcal{A}_0,\theta_1)=\Phi\left(-\Delta/\sigma\right)$ and $\mathcal{R}(\mathcal{A}_{\beta_1},\theta_1)=\Phi\left(-\beta/(\sigma\sqrt{n_{\min}})-\Delta/\sigma\right)$ where Φ is the CDF of the standard normal distribution.

Now we can conclude the proof by picking $\lambda=1/2$ and $\eta=n_{\min}/2$ such that $\theta_1,\theta_2\in[0,1]^2$. Then the result in Theorem 6.4.12 can then be proved by applying an approximation of Φ and setting $\beta_1=-\beta_2=2-cc_0$ such that both β_1 and β_2 are within the range that makes $\mathcal{A}_\beta\in\mathcal{M}_{\mathbf{n},c}$.

To summarize, in batch problems, unlike in online learning, there is no universally adaptive algorithm, and in fact all the confidence-adjusted algorithms have a niche where they outperform the others. Thus, any reasonable algorithm is equally optimal, or not optimal, depending on whether the minimax or instance optimality is considered. In this sense, there remains a lack of a well-defined optimality criterion that can be used to choose between algorithms for batch policy optimization.

6.5 A Characterization of Pessimism

It is known that the pessimistic algorithm, maximizing a lower confidence bound on the value, satisfies many desirable properties: it is consistent with rational decision making using preferences that satisfy uncertainty aversion and certainty-independence [50], it avoids the optimizer's curse [127], it allows for optimal inference in an asymptotic sense [80], and in a certain sense it is the unique strategy that achieves these properties [132, 144]. However, a pure statistical decision theoretic justification (in the sense of Berger [15]) is still lacking.

The instance-dependent lower bound presented above attempts to characterize the optimal performance of an algorithm on an instance-by-instance basis. In particular, one can interpret the objective $\mathcal{R}(\mathcal{A},\theta)/\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$ defined in Theorem 6.4.12 as weighting each instance θ by $1/\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$, where this can be interpreted as a measure of instance difficulty. It is natural to consider an algorithm to be optimal if it can perform well relative to this weighted criteria. However, given that the performance of an algorithm can be arbitrarily different across instances, no such optimal algorithm can exist under this criterion. The question we address here is whether other measures of instance difficulty might be used to distinguish some algorithms as naturally advantageous over others.

In a recent study, Jin et al. [67] show that the pessimistic algorithm is minimax optimal when weighting each instance by the variance induced by the optimal policy. In another recent

paper, Buckman et al. [19] point out that the pessimistic choice has the property that its regret improves whenever the optimal choice's value is easier to predict. In particular, with our notation, their most relevant result (Theorem 3) implies the following: if b_i defines an interval such that $\mu_i \in [\hat{\mu}_i - b_i, \hat{\mu}_i + b_i]$ for all $i \in [k]$, then for $i' = \arg\max_i \hat{\mu}_i - b_i$ one obtains [1]

$$\mu^* - \mu_{i'} \leq 2b_{a^*} \,. \tag{6.5}$$

If we (liberally) interpret b_{a^*} as a measure of how hard it is to predict the value of the optimal choice, this inequality suggests that the pessimistic choice could be justified as the choice that makes the regret comparable to the error of predicting the optimal value.

To make this intuition precise, consider the same problem setup as discussed in Section 6.2. Suppose that the reward distribution for each arm $i \in [k]$ is a Gaussian with unit variance. Consider the problem of estimating the optimal value μ^* where the optimal arm a^* is also provided to the estimator. We define the set of minimax optimal estimators.

Definition 6.5.1 (Minimax Estimator)**.** For fixed $\mathbf{n} = (n_i)_{i\in[k]}$, an estimator is said to be minimax optimal if its worst case error is bounded by the minimax estimate error of the problem up to some constant. We define the set of minimax optimal estimators as

$$\mathcal{V}^*_{\mathbf{n}} = \left\{ \nu : \sup_{\theta\in\Theta_{\mathbf{n}}} \mathbb{E}_\theta \left[|\mu^* - \nu|\right] \leq c \inf_{\nu'\in\mathcal{V}} \sup_{\theta\in\Theta_{\mathbf{n}}} \mathbb{E}_\theta \left[|\mu^* - \nu'|\right] \right\}$$

where c is a universal constant, and $\mathcal{V}$ is the set of all possible estimators.

Now consider using this optimal value estimation problem as a measure of how difficult a problem instance is, and then use this to weight each problem instance as in the definition of instance-dependent lower bound. In particular, let

$$\mathcal{E}^*(\theta) = \inf_{\nu\in\mathcal{V}^*_{\mathbf{n}}} \mathbb{E}_\theta \left[|\mu^* - \nu|\right]$$

be the inherent difficulty of estimating the optimal value μ^* on problem instance θ. The previous result (6.5) suggests (but does not prove) that $\sup_\theta \frac{\mathcal{R}(\text{LCB},\theta)}{\mathcal{E}^*(\theta)} < +\infty$. We now show that not only does this hold, but up to a constant factor, the LCB algorithm is nearly weighted minimax optimal with the weighting given by $\mathcal{E}^*(\theta)$.

Proposition 6.5.2. For any $\mathbf{n} = (n_i)_{i\in[k]}$,

$$\sup_{\theta\in\Theta_{\mathbf{n}}} \frac{\mathcal{R}(\text{LCB},\theta)}{\mathcal{E}^*(\theta)} < c\sqrt{\log|\mathbf{n}|}\,,$$

where c is some universal constant.

Proposition 6.5.3. There exists a sequence $\{\mathbf{n}_j\}$ such that

$$\limsup_{j\to\infty} \sup_{\theta\in\Theta_{\mathbf{n}_j}} \frac{\mathcal{R}(\text{UCB},\theta)}{\sqrt{\log|\mathbf{n}_j|}\cdot\mathcal{E}^*(\theta)} = +\infty$$

[1] This inequality follows directly from the definitions: $\mu^* - \mu_{i'} \leq \mu^* - (\hat{\mu}_{i'} - b_{i'}) \leq \mu^* - (\hat{\mu}_{a^*} - b_{a^*}) \leq 2b_{a^*}$ and we believe this was known as a folklore result, although we are not able to point to a previous paper that includes this inequality. The logic of this inequality is the same as that used in proving regret bounds for UCB policies [79, 85]. It is also clear that the result holds for any data-driven stochastic optimization problem regardless of the structure of the problem. Theorem 3 of Buckman et al. [19] with this notation states that $\mu^* - \mu_{i'} \leq \min_i \mu^* - \mu_i + 2b_i$.

$$\limsup_{j\to\infty} \sup_{\theta\in\Theta_{\mathbf{n}_j}} \frac{\mathcal{R}(\text{greedy},\theta)}{\sqrt{\log|\mathbf{n}_j|\cdot\mathcal{E}^*(\theta)}} = +\infty$$

That is, the pessimistic algorithm can be justified by weighting each instance using the difficulty of predicting the optimal value. We note that this result does not contradict the no-instance-optimality property of batch policy optimization with stochastic bandits (Corollary 6.4.13). In fact, it only provides a characterization of pessimism: the pessimistic choice is beneficial when the batch dataset contains enough information that is good for predicting the optimal value.

6.6 Related work

In the context of offline bandit and RL, a number of approaches based on the pessimistic principle have been proposed and demonstrate great success in practical problems [19, 66, 70, 77, 78, 126, 134, 148, 155]. We refer interested readers to the survey by Levine et al. [89] for recent developments on this topic. To implement the pessimistic principle, the distributional robust optimization (DRO) becomes one powerful tool in bandit [40, 69] and RL [23, 27, 31, 151, 152, 154].

In terms of theoretical perspective, the statistical properties of general DRO, *e.g.*, the consistency and asymptotic expansion of DRO, is analyzed in [37]. Liu et al. [98] provides regret analysis for a pessimistic algorithm based on stationary distribution estimation in offline RL with insufficient data coverage. Jin et al. [67] and Kidambi et al. [70] recently prove that the pessimistic algorithm is nearly minimax optimal for batch policy optimization. However, the theoretical justification of the benefits of pessimitic principle vs. alternatives are missing in offline RL.

Decision theory motivates DRO with an axiomatic characterization of min-max (or distributionally robust) utility: Preferences of decision makers who face an uncertain decision problem and whose preference relationships over their choices satisfy certain axioms follow an ordering given by assigning max-min utility to these preferences [50]. Thus, if we believe that the preferences of the user follow the axioms stated in the above work, one must use a distributionally optimal (pessimistic) choice. On the other hand, Smith and Winkler [128] raise the "optimizer's curse" due to statistical effect, which describes the phenomena that the resulting decision policy may disappoint on unseen out-of-sample data, *i.e.*, the actual value of the candidate decision is below the predicted value. Sutter et al. [132], Van Parys et al. [144] justify the optimality of DRO in combating with such an overfitting issue to avoid the optimizer's curse. Moreover, Delage et al. [29] demonstrate the benefits of randomized policy from DRO in the face of uncertainty comparing with deterministic policy. While reassuring, these still leave open the question whether there is a justification for the pessimistic choice dictated by some alternate logic, or perhaps a more direct logic reasoning in terms of regret in decision problem itself [85].

Our theoretical analysis answer this question, and provides a complete and direct justification for all confidence-based index algorithms. Specifically, we show all confidence-based index algorithms are nearly minimax optimal in terms of regret. More importantly, our instance-dependent analysis show that for any algorithm one can always find some problem instance where

the algorithm will suffer arbitrarily large regret. Therefore, one cannot directly compare the performance of two algorithms without specifying the problem instance. Buckman et al. [19] state that for the pessimistic choice to be a good one, it suffices to have data that makes predicting the value of the optimal policy feasible. We provide a formal analysis to support this intuition: the pessimistic algorithm is nearly minimax optimal when weighting individual instance by its inherent difficulty of estimating the optimal value. This weighted criterion can be used to distinguish pessimistic algorithm from other confidence-adjusted index algorithms.

6.7 Conclusions

In this chapter we study the statistical limits of batch policy optimization with finite-armed bandits. We introduce a family of confidence-adjusted index algorithms that provides a general analysis framework to unify the commonly used optimistic and pessimistic principles. For this family, we show that any index algorithm with an appropriate adjustment is nearly minimax optimal. Our analysis also reveals another important finding, that for any algorithm that performs optimally in some environment, there exists another environment where the same algorithm can suffer arbitrarily large regret. Therefore, the instance-dependent optimality cannot be achieved by any algorithm. To distinguish the algorithms in offline setting, we introduce a weighted minimax objective and justify the pessimistic algorithm is nearly optimal under this criterion.

6.8 Additional Experiment Details

Figure 6.1 The reward distribution for each arm $i \in [100]$ is a Gaussian with unit variance. The mean rewards μ_i are uniformly spread over $[0, 1]$. In particular, we have $\mu_1 \geq \ldots \mu_{100}$, $\mu_i - \mu_{i+1} = 0.01$ for $1 \leq i < 99$, and $\mu_1 = 1$. When generating the data set, we split the arms into two sets S_1 and $S_2 = [k] \setminus S_2$. For each arm $i \in \mathcal{S}_1$, we collect πn data; for each arm $i \in S_2$, we collect $n(1 - \pi|S_1|)/|S_2|$ data, where n is the total sample size, and $0 \leq \pi \leq 1/|\mathcal{S}_1|$ is a parameter to be chosen to generate different data sets. We consider four data sets: ***LCB***-1 ($S_1 = \{1\}$, $\pi = 0.3$); ***LCB***-2 ($S_1 = \{10\}$, $\pi = 0.3$); ***UCB***-1 ($S_1 = \{1\}$, $\pi = 1e^{-4}$); ***UCB***-2 ($S_1 = \{1, \ldots, 10\}$, $\pi = 1e^{-4}$). For each instance, we run each algorithm 500 times and use the average performance to approximate the expected simple regret. Error bars are the standard deviation of the simple regret over the 500 runs.

Figure 6.2(a) and 6.2(b) The reward distribution for each arm $i \in [2]$ is a Gaussian with unit variance. We fix $\mu_1 = 0$ and vary μ_2 accordingly. In Figure 6.2(a), $n_1 = 10, n_2 = 5$. In Figure 6.2(b), $n_1 = 100, n_2 = 10$. For each instance, we run each algorithm 100 times and use the average performance to approximate the expected simple regret. Error bars are the standard deviation of the simple regret over the 100 runs.

Figure 6.2(c) and 6.2(d) For each k, we first sample 100 vectors $\vec{\mu} = [\mu_1, ..., \mu_k]$ in the following way: We generate $\vec{\mu}_0$ with $\vec{\mu}_{0,i} = \frac{i-1}{2(k-1)} + \frac{1}{4}$ such that all reward means are evenly distributed with in $[\frac{1}{4}, \frac{3}{4}]$. We then add independent Gaussian noise with standard deviation 0.05 to each $\vec{\mu}_{0,i}$

to get a sampled $\vec{\mu}$. Generating 100 noise vectors with size k gives 100 samples of $\vec{\mu}$. For each $\vec{\mu}$ we uniformly sample 100 (if exist) subsets $S \subset k, |S| = m$ ($m = k/2$ in (c) and $m = k/4$ in (d)), to generate up to 10k instances. We set $n_i = 100$ for $i \in S$ and $n_i = 1$ for $i \notin S$. For each instance, we run each algorithm 100 times and use the average performance to approximate the expected simple regret. We then select the algorithm with the best average performance for each instance and count the fraction of instances where each algorithm performs the best. Experiment details are provided in the supplementary material. Error bars are representing the standard deviation of the reported fraction over 5 different runs of the whole procedure.

6.9 Proofs

6.9.1 Proof of Minimax Results

Proof of Theorem 6.3.1. Let $m \geq 2$ and $\mu^1, \ldots, \mu^m$ be a collection of vectors in $\mathbb{R}^k$ with $\mu_a^b = \Delta \mathbb{I}\{a = b\}$ where $\Delta > 0$ is a constant to be chosen later. Next, let θ_b be the environment in $\Theta_{\mathbf{n}}$ with P_a a Gaussian distribution with mean μ_a^b and unit variance. Let B be a random variable uniformly distributed on $[m]$ where $m \in [k]$. The Bayesian regret of an algorithm $\mathcal{A}$ is

$$\mathcal{BR}^* = \inf_{\mathcal{A}} \mathbb{E}\left[\mathcal{R}(\mathcal{A}, \theta_B)\right] = \Delta \mathbb{E}\left[\mathbb{I}\{A \neq B\}\right] ,$$

where $A \in [k]$ is the $\sigma(X)$-measurable random variable representing the decision of the Bayesian optimal policy, which is $A = \arg\max_{b \in [k]} \mathbb{P}\{B = b | X\}$. By Bayes' law and the choice of uniform prior,

$$\begin{aligned}
\mathbb{P}\{B = b | X\} &\propto \exp\left(-\frac{1}{2}\sum_{a=1}^{k} n_a(\hat{\mu}_a - \mu_a^b)^2\right) \\
&= \exp\left(-\frac{1}{2}\sum_{a=1}^{k} n_a(\hat{\mu}_a - \Delta\mathbb{I}\{a = b\})^2\right) .
\end{aligned}$$

Therefore, the Bayesian optimal policy chooses

$$A = \arg\min_{b \in [k]} n_b(\Delta/2 - \hat{\mu}_b) .$$

On the other hand,

$$\mathcal{BR}^* = \Delta \mathbb{P}\{A \neq B\} = \frac{\Delta}{k}\sum_{b=1}^{k} \mathbb{P}_b(A \neq b) ,$$

where $\mathbb{P}_b = \mathbb{P}\{\cdot | B = b\}$. Let $b \in [m]$ be arbitrary. Then,

$$\begin{aligned}
&\mathbb{P}_b\{A \neq b\} \\
&\geq \mathbb{P}_b\left\{\hat{\mu}_b \leq \Delta \text{ and } \max_{a \in [m] \setminus \{b\}} \hat{\mu}_a \geq \frac{\Delta}{2}\left(1 + \frac{n_b}{n_a}\right)\right\}
\end{aligned}$$

$$\geq \frac{1}{2}\left(1 - \prod_{a\in[m]\setminus\{b\}}\left(1 - \mathbb{P}_b\left\{\hat{\mu}_a \geq \frac{\Delta}{2}\left(1+\frac{n_b}{n_a}\right)\right\}\right)\right)$$
$$\geq \frac{1}{2}\left(1 - \prod_{a>b}\left(1 - \mathbb{P}_b\left\{\hat{\mu}_a \geq \Delta\right\}\right)\right),$$

where in the second inequality we used independence and the fact that the law of $\hat{\mu}_b$ under $\mathbb{P}_b$ is Gaussian with mean Δ and variance $1/n_b$. The first inequality follows because

$$\left\{\hat{\mu}_b \leq \Delta \text{ and } \max_{a\neq b}\hat{\mu}_a \geq \frac{\Delta}{2}\left(1+\frac{n_b}{n_a}\right)\right\} \subset \{A \neq b\}.$$

Let $b < a \leq m$ and

$$\delta_a(\Delta) = \frac{1}{\Delta\sqrt{n_a} + \sqrt{4 + n_a\Delta^2}}\sqrt{\frac{2}{\pi}}\exp\left(-\frac{n_a\Delta^2}{2}\right).$$

Since for $a \neq b$, $\hat{\mu}_a$ has law $\mathcal{N}(0, 1/n_a)$ under $\mathbb{P}_b$, by standard Gaussian tail inequalities [2, §26],

$$\mathbb{P}_b\{\hat{\mu}_a \geq \Delta\} = \mathbb{P}_b\{\hat{\mu}_a\sqrt{n_a} \geq \Delta\sqrt{n_a}\} \geq \delta_a(\Delta) \geq \delta_m(\Delta),$$

where the last inequality follows from our assumption that $n_1 \leq \cdots \leq n_k$. Therefore, choosing Δ so that $\delta_m(\Delta) = 1/(2m)$,

$$\begin{aligned}
\mathcal{BR}^* &\geq \frac{\Delta}{2m}\sum_{b\in[m]}\left(1 - (1-\delta_m(\Delta))^{m-b}\right) \\
&\geq \frac{\Delta}{2m}\sum_{b\in[m]}\left(1 - \left(1 - \frac{1}{2m}\right)^{m-b}\right) \\
&\geq \frac{\Delta}{2m}\sum_{b\leq m/2}\left(1 - \left(1 - \frac{1}{2m}\right)^{m/2}\right) \\
&\geq \frac{\Delta(m-1)}{20m} \geq \frac{\Delta}{40}.
\end{aligned}$$

A calculation shows there exists a universal constant $c > 0$ such that

$$\Delta \geq c\sqrt{\frac{\log(m)}{n_m}},$$

which shows there exists a (different) universal constant $c > 0$ such that

$$\inf_{\mathcal{A}}\sup_{\theta}\mathcal{R}(\mathcal{A},\theta) \geq \mathcal{BR}^* \geq \max_{m\geq 2} c\sqrt{\frac{\log(m)}{n_m}}.$$

The argument above relies on the assumption that $m \geq 2$. A minor modification is needed to handle the case where n_1 is much smaller than n_2. Let B be uniformly distributed on $\{1, 2\}$ and let $\theta_1, \theta_2 \in \Theta_{\mathbf{n}}$ be defined as above, but with $\mu^1 = (\Delta, 0)$ and $\mu^2 = (-\Delta, 0)$ for some constant $\Delta > 0$ to be tuned momentarily. As before, the Bayesian optimal policy has a simple closed form solution, which is

$$A = \begin{cases} 1 & \text{if } \hat{\mu}_1 \geq 0 \\ 2 & \text{otherwise}\,. \end{cases}$$

The Bayesian regret of this policy satisfies

$$\begin{aligned} \mathcal{BR}^* &= \frac{1}{2}\mathcal{R}(\mathcal{A}, \theta_1) + \frac{1}{2}\mathcal{R}(\mathcal{A}, \theta_2) \geq \frac{1}{2}\mathcal{R}(\mathcal{A}, \theta_1) \\ &\geq \frac{1}{2}\mathbb{P}_1\{A = 2\} \geq \frac{\Delta}{2}\mathbb{P}_1\{\hat{\mu}_1 < 0\} \\ &\geq \sqrt{\frac{2}{\pi}} \frac{\Delta}{2\Delta\sqrt{n_1} + 2\sqrt{4 + n_1\Delta^2}} \exp\left(-\frac{n_1\Delta^2}{2}\right) \\ &\geq \frac{1}{13}\sqrt{\frac{1}{n_1}}\,, \end{aligned}$$

where the final inequality follows by tuning Δ. □

Proof of Theorem 6.3.2. Let $i' = \arg\max_i \tilde{\mu}_i$. Then, given that (6.2) is true for all arms, which is with probability at least $1 - \delta$, we have

$$\begin{aligned} \mu^* - \mu_{i'} &= \mu^* - \tilde{\mu}_{a^*} + \tilde{\mu}_{a^*} - \tilde{\mu}_{i'} + \tilde{\mu}_{i'} - \mu_{i'} \\ &\leq \mu^* - \tilde{\mu}_{a^*} + \tilde{\mu}_{i'} - \mu_{i'} \\ &\leq \mu^* - \hat{\mu}_{a^*} + \hat{\mu}_{i'} - \mu_{i'} + 2\sqrt{\frac{2\log(k/\delta)}{\min_i n_i}} \\ &\leq \sqrt{\frac{32\log(k/\delta)}{\min_i n_i}}\,, \end{aligned}$$

where the first two inequalities follow from the definition of the index algorithm, and the last follows from (6.2). Using the tower rule gives the desired result. □

6.9.2 Proof of Instance-dependent Results

Proof of Theorem 6.4.1. Assuming $\mu_1 \geq \mu_2 \geq ... \geq \mu_k$, if we have $\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq b_i$, then we can write

$$\begin{aligned} \mathcal{R}(\mathcal{A}) &= \sum_{2 \leq i \leq k} \Delta_i \mathbb{P}\left(\mathcal{A}(X) = i\right) \\ &= \sum_{2 \leq i \leq k} \Delta_i \left(\mathbb{P}\left(\mathcal{A}(X) \geq i\right) - \mathbb{P}\left(\mathcal{A}(X) \geq i + 1\right)\right) \end{aligned}$$

$$
\begin{aligned}
&= \sum_{2\le i\le k} (\Delta_i - \Delta_{i-1})\, \mathbb{P}\left(\mathcal{A}(X) \ge i\right) \\
&\le \sum_{2\le i\le k} (\Delta_i - \Delta_{i-1})\, b_i \\
&= \sum_{2\le i\le k} \Delta_i (b_i - b_{i+1})\,.
\end{aligned}
$$

To upper bound $\mathbb{P}\left(\mathcal{A}(X) \ge i\right)$, let I_i be the index used by algorithm $\mathcal{A}$, i.e., $\mathcal{A}(X) = \arg\max_i I_i$. Then

$$
\mathbb{P}\left(\mathcal{A}(X) \ge i\right) \le \mathbb{P}\left(\max_{j\ge i} I_j \ge \max_{j<i} I_j\right)\,.
$$

Hence we can further write

$$
\begin{aligned}
\mathbb{P}\left(\mathcal{A}(X) \ge i\right) &\le \mathbb{P}\left(\max_{j\ge i} I_j \ge \max_{j<i} I_i, \max_{j<i} I_j \ge \eta\right) \\
&\quad + \mathbb{P}\left(\max_{j\ge i} I_j \ge \max_{j<i} I_i, \max_{j<i} I_j < \eta\right) \\
&\le \mathbb{P}\left(\max_{j\ge i} I_j \ge \eta\right) + \mathbb{P}\left(\max_{j<i} I_j < \eta\right)\,. \qquad (6.6)
\end{aligned}
$$

Next we optimize the choice of η according to the specific choice of the index. For this let $I_i = \hat{\mu}_i + b_i$.

Continuing with (6.6), for the first term, by the union bound we have

$$
\mathbb{P}\left(\max_{j\ge i} I_j \ge \eta\right) \le \sum_{j\ge i} \mathbb{P}\left(I_j \ge \eta\right)\,.
$$

For each $j \ge i$, by Hoeffding's inequality we have

$$
\mathbb{P}\left(I_j \ge \eta\right) \le e^{-\frac{n_j}{2}(\eta-\mu_j-b_j)_+^2}\,.
$$

For the second term in (6.6), we have $\mathbb{P}\left(\max_{j<i} I_j < \eta\right) \le \mathbb{P}\left(I_j < \eta\right)$ for each $j < i$.

By Hoeffding's inequality we have

$$
\mathbb{P}\left(I_j < \eta\right) \le e^{-\frac{n_j}{2}(\mu_j+b_j-\eta)_+^2}\,,
$$

and thus

$$
\mathbb{P}\left(\max_{j<i} I_j < \eta\right) \le \min_{j<i} e^{-\frac{n_j}{2}(\mu_j+b_j-\eta)_+^2}\,.
$$

Define

$$
g_i(\eta) = \sum_{j\ge i} e^{-\frac{n_j}{2}(\eta-\mu_j-b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j+b_j-\eta)_+^2}
$$

and $g_i^* = \min_\eta g_i(\eta)$. Then we have

$$\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq \min\{1, g_i^*\}\,.$$

Putting everything together, we bound the expected regret as

$$\mathcal{R}(\mathcal{A}) \leq \sum_{2 \leq i \leq k} \Delta_i \left(\min\{1, g_i^*\} - \min\{1, g_{i+1}^*\}\right)$$

where we define $g_{k+1}^* = 0$.

□

Proof of Remark 6.4.4. Recall the definition of $g_i(\eta)$:

$$g_i(\eta) = \sum_{j \geq i} e^{-\frac{n_j}{2}(\eta - \mu_j - b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \eta)_+^2}\,.$$

Let $\eta = \mu_1 - 2\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}$. Then, for the second term of $g_i(\eta)$,

$$\min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \eta)_+^2} \leq e^{-\frac{n_1}{2}\left(2\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}} - \sqrt{\frac{2}{n_1} \log \frac{k}{\delta}}\right)_+^2} \leq \frac{\delta}{k}\,.$$

For the first term,

$$\sum_{j \geq i} e^{-\frac{n_j}{2}(\eta - \mu_j - b_j)_+^2} = \sum_{j \geq i} e^{-\frac{n_j}{2}\left(\mu_1 - 2\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}} - \mu_j - b_j\right)_+^2} \leq \sum_{j \geq i} e^{-\frac{n_{\min}}{2}\left(\Delta_j - 3\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}\right)_+^2}\,.$$

Thus,

$$g_i^* \leq \sum_{j \geq i} e^{-\frac{n_{\min}}{2}\left(\Delta_j - 3\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}\right)_+^2} + \frac{\delta}{k}\,.$$

For arm i such that $\Delta_i \geq 4\sqrt{\frac{2}{n_{\min}} \log \frac{k}{\delta}}$, by Theorem 6.4.1 we have $P(\mathcal{A}(X) \geq i) \leq g_i^* \leq \delta$. The result then follows by the tower rule. □

Proof of Corollary 6.4.2. For each i, let $\eta_i = \max_{j<i} L_j$. Then,

$$g_i(\eta_i) = \sum_{j \geq i} e^{-\frac{n_j}{2}(\max_{j<i} L_j - \mu_j - b_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \max_{j<i} L_j)_+^2}\,.$$

Let $s = \arg\max_{j<i} L_j$. For the second term we have,

$$\min_{j<i} e^{-\frac{n_j}{2}(\mu_j + b_j - \max_{j<i} L_j)_+^2} \leq e^{-\frac{n_s}{2}(\mu_s + b_s - L_s)_+^2} \leq \frac{\delta}{k}\,.$$

Next we consider the first term. Recall that $h = \max\{i \in [k] : \max_{j<i} L_j < \max_{j' \geq i} U_{j'}\}$. Then for any $i > h$, we have $\max_{j<i} L_j \geq U_{j'}$ for all $j' \geq i$. Therefore,

$$\sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - \mu_j - b_j\right)_+^2} = \sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j + \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2} \leq \frac{\delta}{k} \sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j\right)^2}.$$

Note that for $i \leq h$, $\Delta_i \leq \Delta_h$. Thus we have,

$$\begin{aligned}
\mathcal{R}(\mathcal{A}) &\leq \Delta_h + \sum_{i>h} (\Delta_i - \Delta_{i-1}) \mathbb{P}(\mathcal{A}(X) \geq i) \\
&\leq \Delta_h + \frac{\delta}{k}\Delta_{\max} + \frac{\delta}{k} \sum_{i>h} (\Delta_i - \Delta_{i-1}) \sum_{j \geq i} e^{-\frac{n_j}{2}\left(\max_{j'<i} L_{j'} - U_j\right)^2},
\end{aligned}$$

which concludes the proof. □

Proof of Corollary 6.4.5. Considering the greedy algorithm, for each $i \geq 2$,

$$g_i(\eta) = \sum_{j \geq i} e^{-\frac{n_j}{2}(\eta - \mu_j)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}(\mu_j - \eta)_+^2}.$$

Define $h_i = \arg\max_{j<i} \mu_j - \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}$ and $\eta_i = \mu_{h_i} - \sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}}$. Then we have $e^{-\frac{n_{h_i}}{2}\left(\mu_{h_i} - \eta_i\right)_+^2} = \delta/k$. Then for $j \geq i$ we have

$$e^{-\frac{n_j}{2}(\eta_i - \mu_j)_+^2} = e^{-\frac{n_j}{2}\left(\mu_{h_i} - \mu_j - \sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}}\right)_+^2}.$$

When $\mu_{h_i} - \mu_j \geq \sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}} + \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}$ we have $e^{-\frac{n_j}{2}(\eta_i - \mu_j)_+^2} \leq \delta/k$.

Define

$$U_i = \mathbb{1}\left\{\forall j \geq i, \mu_{h_i} - \mu_j \geq \sqrt{\frac{2}{n_{h_i}}\log\frac{k}{\delta}} + \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right\}.$$

Then we have $g_i^* U_i \leq \frac{k-i+2}{k}\delta \leq \delta$. According to Theorem 6.4.1 we have $\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq \min\{1, g_i^*\}$, so for any i such that $\mathbb{P}\left(\mathcal{A}(X) \geq i\right) > \delta$, we must have $U_i = 0$, which is equivalent to

$$\max_{j<i} \mu_j - \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}} < \max_{j \geq i} \mu_j + \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}. \tag{6.7}$$

Let $\hat{i}$ be the largest index i that satisfies (6.7). Then we have $\mathbb{P}\left(\mathcal{A}(X) \geq \hat{i} + 1\right) \leq \delta$. Therefore, we have $\mathbb{P}\left(\mu^* - \mu_{\mathcal{A}(X)} \leq \Delta_{\hat{i}}\right) \geq 1 - \delta$, and it remains to upper bound $\Delta_{\hat{i}}$.

For any $i \in [k]$, if $\hat{i} \leq i$ then $\Delta_{\hat{i}} \leq \Delta_i$. If $\hat{i} > i$ we have

$$\max_{j<\hat{i}} \mu_j - \sqrt{\frac{2}{n_j}\log\frac{k}{\delta}} \geq \mu_i - \sqrt{\frac{2}{n_i}\log\frac{k}{\delta}}$$

and

$$\max_{j\geq\hat{i}}\mu_j+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\leq\mu_{\hat{i}}+\max_{j>i}\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\,.$$

Applying (6.7) gives

$$\Delta_{\hat{i}}-\Delta_i=\mu_i-\mu_{\hat{i}}\leq\sqrt{\frac{2}{n_i}\log\frac{k}{\delta}}+\max_{j>i}\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\,,$$

so

$$\Delta_{\hat{i}}\leq\Delta_i+\sqrt{\frac{2}{n_i}\log\frac{k}{\delta}}+\max_{j>i}\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}$$

holds for any $i\in[k]$, concluding the proof. □

Proof of Corollary 6.4.6. Let $\eta=\max_i\mu_i-\sqrt{\frac{8}{n_i}\log\frac{k}{\delta}}$. Considering the LCB algorithm, for each $i\geq 2$, we have

$$g_i(\eta)=\sum_{j\geq i}e^{-\frac{n_j}{2}\left(\eta-\mu_j+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}+\min_{j<i}e^{-\frac{n_j}{2}\left(\mu_j-\eta-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}.$$

Define $h_i=\arg\max_{j<i}\mu_j-\sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}$ and $\eta_i=\mu_{h_i}-\sqrt{\frac{8}{n_{h_i}}\log\frac{k}{\delta}}$. Then we have

$$e^{-\frac{n_{h_i}}{2}\left(\mu_{h_i}-\eta_i-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}=\delta/k\,.$$

Now, consider $j\geq i$. Then,

$$e^{-\frac{n_j}{2}\left(\eta_i-\mu_j+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}\leq\frac{\delta}{k}$$

whenever $\eta_i-\mu_j\geq 0$, i.e. $\mu_{h_i}-\sqrt{\frac{8}{n_{h_i}}\log\frac{k}{\delta}}\geq\mu_j$.

Define

$$U_i=\mathbb{1}\left\{\forall j\geq i,\mu_{h_i}-\sqrt{\frac{8}{n_{h_i}}\log\frac{k}{\delta}}\geq\mu_j\right\}.$$

Then we have $g_i^*U_i\leq\frac{k-i+2}{k}\delta\leq\delta$. According to Theorem 6.4.1 we have $\mathbb{P}\left(\mathcal{A}(X)\geq i\right)\leq\min\{1,g_i^*\}$, so for any i such that $\mathbb{P}\left(\mathcal{A}(X)\geq i\right)>\delta$, we must have $U_i=0$, which is equivalent to that there exists some $s\geq i$ such that

$$\mu_s>\max_{j<i}\mu_j-\sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}\,. \tag{6.8}$$

Let $\hat{i}$ be the largest index i that satisfies (6.8). Then we have $\mathbb{P}\left(\mathcal{A}(X) \geq \hat{i}+1\right) \leq \delta$ and thus $\mathbb{P}\left(\mu^* - \mu_{\mathcal{A}(X)} \leq \Delta_{\hat{i}}\right) \geq 1-\delta$. It remains to upper bound $\Delta_{\hat{i}}$.

For any $i \in [k]$, if $\hat{i} \leq i$ then $\Delta_{\hat{i}} \leq \Delta_i$. If $\hat{i} > i$ we have

$$\mu_{\hat{i}} > \max_{j<\hat{i}} \mu_j - \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}} \geq \mu_i - \sqrt{\frac{8}{n_i}\log\frac{k}{\delta}}\,.$$

Therefore,

$$\Delta_{\hat{i}} = \Delta_i + \mu_i - \mu_{\hat{i}} \leq \Delta_i + \sqrt{\frac{8}{n_i}\log\frac{k}{\delta}}\,,$$

which concludes the proof. □

Proof of Corollary 6.4.7. Consider now the UCB algorithm. Then, for each $i \geq 2$,

$$g_i(\eta) = \sum_{j\geq i} e^{-\frac{n_j}{2}\left(\eta-\mu_j-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2} + \min_{j<i} e^{-\frac{n_j}{2}\left(\mu_j-\eta+\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2}.$$

Pick $\eta = \mu_1$ then the second term in $g_i(\eta)$ becomes δ/k. For j such that $\Delta_j \geq \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}$ we have

$$e^{-\frac{n_j}{2}\left(\eta-\mu_j-\sqrt{\frac{2}{n_j}\log\frac{k}{\delta}}\right)_+^2} \leq \frac{\delta}{k}\,.$$

Define

$$U_i = \mathbb{1}\left\{\forall j \geq i, \Delta_j \geq \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}\right\}.$$

Then we have $g_i^* U_i \leq \frac{k-i+2}{k}\delta \leq \delta$. According to Theorem 6.4.1 we have $\mathbb{P}\left(\mathcal{A}(X) \geq i\right) \leq \min\{1, g_i^*\}$, so for any i such that $\mathbb{P}\left(\mathcal{A}(X) \geq i\right) > \delta$, we must have $U_i = 0$, which is equivalent to

$$\max_{j\geq i} \mu_j + \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}} > \mu_1\,. \tag{6.9}$$

Let $\hat{i}$ be the largest index i that satisfies (6.9). Then we have $\mathbb{P}\left(\mathcal{A}(X) \geq \hat{i}+1\right) \leq \delta$. Therefore, we have $\mathbb{P}\left(\mu^* - \mu_{\mathcal{A}(X)} \leq \Delta_{\hat{i}}\right) \geq 1-\delta$. It remains to upper bound $\Delta_{\hat{i}}$.

For any $i \in [k]$, if $\hat{i} \leq i$ then $\Delta_{\hat{i}} \leq \Delta_i$. If $\hat{i} > i$, we have

$$\max_{j\geq \hat{i}} \mu_j + \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}} \leq \mu_{\hat{i}} + \max_{j>i} \sqrt{\frac{8}{n_j}\log\frac{k}{\delta}}\,.$$

Applying (6.9) gives

$$\Delta_{\hat{i}} = \mu_1 - \mu_{\hat{i}} \leq \max_{j>i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \,.$$

Therefore,

$$\Delta_{\hat{i}} \leq \max \left\{ \Delta_i, \max_{j>i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \right\} \leq \Delta_i + \max_{j>i} \sqrt{\frac{8}{n_j} \log \frac{k}{\delta}} \,.$$

for any $i \in [k]$, which concludes the proof. □

Proof of Proposition 6.4.9. Fixing $S \subset [k]$, we take $\{n_i\}_{i \in S} \to \infty$ and $\{n_i\}_{i \notin S} = 1$. The upper bound for LCB in Corollary 6.4.6 can be written as

$$\begin{aligned} \hat{\mathcal{R}}_S(\text{LCB}) &= \min \left\{ \min_{i \in S} \Delta_i, \min_{i \notin S} \left(\Delta_i + \sqrt{8 \log \frac{k}{\delta}} \right) \right\} + \delta \\ &= \min_{i \in S} \Delta_i + \delta \\ &= \Delta_{\min\{i \in [k] : i \in S\}} + \delta \,. \end{aligned}$$

Similarly, we have

$$\hat{\mathcal{R}}_S(\text{UCB}) = \min_{i \in [k]} \left(\Delta_i + \max_{j>i, j \notin S} \sqrt{8 \log \frac{k}{\delta}} \right) + \delta$$

and

$$\hat{\mathcal{R}}_S(\text{greedy}) \geq \min_{i \in [k]} \left(\Delta_i + \max_{j>i, j \notin S} \sqrt{2 \log \frac{k}{\delta}} \right) + \delta \,.$$

Note that for $\delta \in (0,1)$, $\sqrt{2 \log \frac{k}{\delta}} > 1 \geq \Delta_{\max}$. So we can further lower bound $\hat{\mathcal{R}}_S(\text{UCB})$ and $\hat{\mathcal{R}}_S(\text{greedy})$ by $\Delta_h + \delta$ where $h = \min\{i \in [k] : \forall j > i, j \in S\}$. Let $m = |S|$. Notice that unless $S = \{k-m+1, ..., k\}$, we always have $\min\{i \in [k] : i \in S\} < \min\{i \in [k] : \forall j > i, j \in S\}$. So we have $\hat{\mathcal{R}}_S(\text{LCB}) < \hat{\mathcal{R}}_S(\text{UCB})$ (or $\hat{\mathcal{R}}_S(\text{greedy})$) whenever $S \neq \{k-m+1, ..., k\}$. Under the uniform distribution over all possible subsets for S, the event $S = \{k-m+1, ..., k\}$ happens with probability $\binom{k}{m}^{-1}$, which concludes the proof.

□

6.9.3 Instance-dependent Lower Bounds

Proof of Theorem 6.4.12. We first derive an upper bound for $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$. Assuming $X = (X_1, X_2, \mathbf{n})$ with $X_i \sim \mathcal{N}(\mu_i, 1/n_i)$, for any $\beta \in \mathbb{R}$, we define algorithm $\mathcal{A}_\beta$ as

$$\mathcal{A}_\beta(X) = \begin{cases} 1, & \text{if } X_1 - X_2 \geq \frac{\beta}{\sqrt{n_{\min}}} \,; \\ 2, & \text{otherwise} \,. \end{cases}$$

We now analyze the regret for $\mathcal{A}_\beta$. By Hoeffding's inequality we have the following instance-dependent regret upper bound:

Proposition 6.9.1. Consider any $\beta \in \mathbb{R}$ and $\theta \in \Theta_{\mathbf{n}}$. Let $\Delta = |\mu_1 - \mu_2|$. If $\mu_1 \geq \mu_2$ then

$$\mathcal{R}(\mathcal{A}_\beta, \theta) \leq \mathbb{1}\left\{\Delta \leq \frac{\beta}{\sqrt{n_{\min}}}\right\} \frac{\beta}{\sqrt{n_{\min}}} + \mathbb{1}\left\{\Delta > \frac{\beta}{\sqrt{n_{\min}}}\right\} e^{-\frac{n_{\min}}{4}\left(\Delta - \frac{\beta}{\sqrt{n_{\min}}}\right)_+^2}.$$

Furthermore, if $\mu_1 < \mu_2$, we have

$$\mathcal{R}(\mathcal{A}_\beta, \theta) \leq \mathbb{1}\left\{\Delta \leq \frac{-\beta}{\sqrt{n_{\min}}}\right\} \frac{-\beta}{\sqrt{n_{\min}}} + \mathbb{1}\left\{\Delta > \frac{-\beta}{\sqrt{n_{\min}}}\right\} e^{-\frac{n_{\min}}{4}\left(\Delta + \frac{\beta}{\sqrt{n_{\min}}}\right)_+^2}.$$

Maximizing over Δ gives our worst case regret guarantee:

Proposition 6.9.2. For any $\beta \in \mathbb{R}$,

$$\sup_{\theta \in \Theta_{\mathbf{n}}} \mathcal{R}(\mathcal{A}_\beta, \theta) \leq \frac{|\beta| + 2}{\sqrt{n_{\min}}}.$$

$\mathcal{A}_\beta(X)$ is minimax optimal for a specific range of β:

Proposition 6.9.3. If $|\beta| \leq cc_0 - 2$ then $\mathcal{A}_\beta \in \mathcal{M}_{\mathbf{n},c}$.

Given $\theta \in \Theta_{\mathbf{n}}$, to upper bound $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)$, we pick β such that $\mathcal{A}_\beta \in \mathcal{M}_{\mathbf{n},c}$ and $\mathcal{A}_\beta$ performs well on θ. For θ where $\mu_1 \geq \mu_2$, we set $\beta = 2 - cc_0$ thus $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta) \leq \mathcal{R}(\mathcal{A}_{2-cc_0}, \theta)$. For θ where $\mu_1 < \mu_2$, we set $\beta = cc_0 - 2$ thus $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta) \leq \mathcal{R}(\mathcal{A}_{cc_0-2}, \theta)$.

We now construct two instances $\theta_1, \theta_2 \in \Theta_{\mathbf{n}}$ and show that no algorithm can achieve regret close to $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}$ on both instances. Fixing some $\lambda \in \mathbb{R}$ and $\eta > 0$, we define

$$\theta_1 = (\mu_1, \mu_2) = (\lambda + \frac{\eta}{n_1}, \lambda - \frac{\eta}{n_2})$$

and

$$\theta_2 = (\mu_1', \mu_2') = (\lambda - \frac{\eta}{n_1}, \lambda + \frac{\eta}{n_2}).$$

On instance θ_1 we have $X_1 - X_2 \sim \mathcal{N}((\frac{1}{n_1} + \frac{1}{n_2})\eta, \frac{1}{n_1} + \frac{1}{n_2})$ while on instance θ_2 we have $X_1 - X_2 \sim \mathcal{N}(-(\frac{1}{n_1} + \frac{1}{n_2})\eta, \frac{1}{n_1} + \frac{1}{n_2})$. Let Φ be the CDF of the standard normal distribution $\mathcal{N}(0,1)$, $\Delta = (\frac{1}{n_1} + \frac{1}{n_2})\eta$, and $\sigma^2 = \frac{1}{n_1} + \frac{1}{n_2}$. Then we have

$$\begin{aligned}
\mathcal{R}(\mathcal{A}_\beta, \theta_1) &= \Delta \mathbb{P}_{\theta_1}(\mathcal{A}_\beta = 2) \\
&= \Delta \mathbb{P}_{\theta_1}\left(X_1 - X_2 < \frac{\beta}{\sqrt{n_{\min}}}\right) \\
&= \Delta \Phi\left(\frac{\beta - \Delta\sqrt{n_{\min}}}{\sigma\sqrt{n_{\min}}}\right),
\end{aligned}$$

and

$$\mathcal{R}(\mathcal{A}_{-\beta}, \theta_2) = \Delta \mathbb{P}_{\theta_2}(\mathcal{A}_{-\beta} = 1)$$

$$= \Delta \mathbb{P}_{\theta_2} \left(X_1 - X_2 \geq -\frac{\beta}{\sqrt{n_{\min}}} \right)$$
$$= \Delta \Phi \left(\frac{\beta - \Delta\sqrt{n_{\min}}}{\sigma\sqrt{n_{\min}}} \right) .$$

It follows that our upper bound on $\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}$ is the same for both instances, i.e., $\mathcal{R}(\mathcal{A}_{2-cc_0}, \theta_1) = \mathcal{R}(\mathcal{A}_{cc_0-2}, \theta_2)$. Next we show that the greedy algorithm $\mathcal{A}_0$ is optimal in terms of minimizing the worse regret between θ_1 and θ_2.

Lemma 6.9.4. Let $\mathcal{A}_0$ be the greedy algorithm where $\mathcal{A}_0(X) = 1$ if $X_1 \geq X_2$ and $\mathcal{A}_0(X) = 2$ otherwise. Then we have

$$\mathcal{R}(\mathcal{A}_0, \theta_1) = \mathcal{R}(\mathcal{A}_0, \theta_2) = \min_{\mathcal{A}} \max\{\mathcal{R}(\mathcal{A}, \theta_1), \mathcal{R}(\mathcal{A}, \theta_2)\} .$$

Proof of Lemma 6.9.4. The first step is to show that by applying the Neyman-Pearson Lemma, thresholding algorithms on $X_1 - X_2$ perform the most powerful hypothesis tests between θ_1 and θ_2.

Let f_θ be the probability density function for the observation (X_1, X_2) under instance θ. Then, the likelihood ratio function can be written as

$$\frac{f_{\theta_1}(X_1, X_2)}{f_{\theta_2}(X_1, X_2)} = \frac{e^{-\frac{n_1}{2}(X_1 - \lambda - \eta/n_1)^2 - \frac{n_2}{2}(X_2 - \lambda + \eta/n_2)^2}}{e^{-\frac{n_1}{2}(X_1 - \lambda + \eta/n_1)^2 - \frac{n_2}{2}(X_2 - \lambda - \eta/n_2)^2}} = e^{2\eta(X_1 - X_2)} .$$

Applying the Neyman-Pearson Lemma to our scenario gives the following statement:

Proposition 6.9.5 (Neyman-Pearson Lemma)**.** For any $\gamma > 0$ let $\mathcal{A}^\gamma$ be the algorithm where $\mathcal{A}^\gamma(X) = 1$ if $\frac{f_{\theta_1}(X_1, X_2)}{f_{\theta_2}(X_1, X_2)} \geq \gamma$ and $\mathcal{A}^\gamma(X) = 2$ otherwise. Let $\alpha = \mathbb{P}_{\theta_1}(\mathcal{A}^\gamma(X) = 2)$. Then for any algorithm $\mathcal{A}'$ such that $\mathbb{P}_{\theta_1}(\mathcal{A}'(X) = 2) = \alpha$, we have $\mathbb{P}_{\theta_2}(\mathcal{A}'(X) = 1) \geq \mathbb{P}_{\theta_2}(\mathcal{A}^\gamma(X) = 1)$.

Note that $\frac{f_{\theta_1}(X_1, X_2)}{f_{\theta_2}(X_1, X_2)} \geq \gamma$ is equivalent to $X_1 - X_2 \geq (2\eta)^{-1} \log \gamma$. Returning to the proof of Lemma 6.9.4, consider an arbitrary algorithm $\mathcal{A}'$ and let $\alpha = \mathcal{R}(\mathcal{A}', \theta_1)/\Delta = \mathbb{P}_{\theta_1}(\mathcal{A}'(X) = 2)$. Let γ be the threshold that satisfies $\mathbb{P}_{\theta_1}(\mathcal{A}^\gamma(X) = 2) = \alpha$. This exists because X_1, X_2 follow a continuous distribution. According to Proposition 6.9.5 we have $\mathbb{P}_{\theta_2}(\mathcal{A}'(X) = 1) \geq \mathbb{P}_{\theta_2}(\mathcal{A}^\gamma(X) = 1)$. Therefore, we have shown that $\mathcal{R}(\mathcal{A}^\gamma, \theta_1) = \mathcal{R}(\mathcal{A}', \theta_1)$ and $\mathcal{R}(\mathcal{A}^\gamma, \theta_2) \leq \mathcal{R}(\mathcal{A}', \theta_2)$, which means that for any algorithm $\mathcal{A}'$ there exists some γ such that

$$\max\{\mathcal{R}(\mathcal{A}^\gamma, \theta_1), \mathcal{R}(\mathcal{A}^\gamma, \theta_2)\} \leq \max\{\mathcal{R}(\mathcal{A}', \theta_1), \mathcal{R}(\mathcal{A}', \theta_2)\} .$$

It remains to show that $\gamma = 1$ is the minimizer of $\max\{\mathcal{R}(\mathcal{A}^\gamma, \theta_1), \mathcal{R}(\mathcal{A}^\gamma, \theta_2)\}$. This comes from the fact that $\mathcal{R}(\mathcal{A}^\gamma, \theta_1)$ is a monotonically increasing function of γ while $\mathcal{R}(\mathcal{A}^\gamma, \theta_2)$ is a monotonically decreasing function of γ and $\gamma = 1$ makes $\mathcal{R}(\mathcal{A}^\gamma, \theta_1) = \mathcal{R}(\mathcal{A}^\gamma, \theta_2)$, which means that $\gamma = 1$ is the minimizer. □

We now continue with the proof of Theorem 6.4.12. Applying Lemma 6.9.4 gives

$$\begin{aligned}
\sup_{\theta\in\Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A},\theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} &\geq \max\left\{\frac{\mathcal{R}(\mathcal{A},\theta_1)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_1)}, \frac{\mathcal{R}(\mathcal{A},\theta_2)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta_2)}\right\} \\
&\geq \max\left\{\frac{\mathcal{R}(\mathcal{A},\theta_1)}{\mathcal{R}(\mathcal{A}_{2-cc_0},\theta_1)}, \frac{\mathcal{R}(\mathcal{A},\theta_2)}{\mathcal{R}(\mathcal{A}_{cc_0-2},\theta_2)}\right\} \\
&= \frac{\max\left\{\mathcal{R}(\mathcal{A},\theta_1), \mathcal{R}(\mathcal{A},\theta_2)\right\}}{\mathcal{R}(\mathcal{A}_{2-cc_0},\theta_1)} \\
&\geq \frac{\mathcal{R}(\mathcal{A}_0,\theta_1)}{\mathcal{R}(\mathcal{A}_{2-cc_0},\theta_1)} \\
&= \frac{\Phi\left(-\frac{\Delta}{\sigma}\right)}{\Phi\left(-\frac{cc_0-2}{\sigma\sqrt{n_{\min}}} - \frac{\Delta}{\sigma}\right)}. \qquad (6.10)
\end{aligned}$$

Now we apply the fact that for $x > 0$, $\frac{x}{1+x^2}\phi(x) < \Phi(-x) < \frac{1}{x}\phi(x)$ to lower bound (6.10), where ϕ is the probability density function of the standard normal distribution. Choosing $\beta = cc_0 - 2$, we have

$$\frac{\Phi\left(-\frac{\Delta}{\sigma}\right)}{\Phi\left(-\frac{\beta}{\sigma\sqrt{n_{\min}}} - \frac{\Delta}{\sigma}\right)} \geq \frac{\beta + \Delta\sqrt{n_{\min}}}{\sigma\sqrt{n_{\min}}} \frac{\Delta/\sigma}{1+(\Delta/\sigma)^2} e^{\frac{1}{2}\left(\frac{\beta^2}{\sigma^2 n_{\min}} + \frac{\beta\Delta}{\sigma^2\sqrt{n_{\min}}}\right)} \geq \frac{\eta^2}{n_{\min}+\eta^2} e^{\frac{\beta^2}{4} + \frac{\beta\eta}{2\sqrt{n_{\min}}}}.$$

Picking $\lambda = 1/2$ and $\eta = n_{\min}/2$ such that $\theta_1, \theta_2 \in [0,1]^2$, we have

$$\sup_{\theta\in\Theta_{\mathbf{n}}} \frac{\mathcal{R}(\mathcal{A},\theta)}{\mathcal{R}^*_{\mathcal{M}_{\mathbf{n},c}}(\theta)} \geq \frac{n_{\min}}{n_{\min}+4} e^{\frac{\beta^2}{4} + \frac{\beta}{4}\sqrt{n_{\min}}},$$

which concludes the proof. □

6.9.4 Proof for Section 6.5

For any θ, let μ_1 and n_1 be the reward mean and sample count for the optimal arm. We first prove that $\mathcal{E}^*(\theta)$ is at the order of $1/\sqrt{n_1}$ for any θ.

Proposition 6.9.6. There exist universal constants c_0 and c_1 such that, for any $\theta \in \Theta_{\mathbf{n}}$, $c_0/\sqrt{n_1} \leq \mathcal{E}^*(\theta) \leq c_1/\sqrt{n_1}$.

Proof of Proposition 6.9.6. For any constant $c > 0$, define $\theta' \in \Theta$ such that the only difference between θ' and θ is the mean for the optimal arm: θ' has $\mu_1' = \mu_1 + \frac{4c}{\sqrt{n_1}}$.

For any algorithm such that $\mathbb{E}_{\theta'}\left[|\mu_1' - \nu|\right] \leq \frac{c}{\sqrt{n_1}}$, we have $\mathbb{P}_{\theta'}\left(\nu \geq \mu_1 + \frac{2c}{\sqrt{n_1}}\right) \geq \frac{1}{2}$ by Markov inequality. Applying the fact that, when p and q are two Bernoulli distributions with parameter p and q respectively, if $p \geq 1/2$ we have $\mathrm{KL}(p,q) \geq \frac{1}{2}\log\frac{1}{4q}$. Then we have

$$\mathbb{P}_\theta\left(\nu \geq \mu_1 + \frac{2c}{\sqrt{n_1}}\right) \geq \frac{1}{4} e^{-\mathrm{KL}(\theta,\theta')} = \frac{1}{4} e^{-4c^2}.$$

Therefore, we have

$$\mathbb{E}_\theta\left[|\mu_1-\nu|\right] \geq \frac{2c}{\sqrt{n_1}}\mathbb{P}_\theta\left(\nu \geq \mu_1 + \frac{2c}{\sqrt{n_1}}\right) \geq \frac{ce^{-4c^2}}{2\sqrt{n_1}}\,.$$

Now we apply the fact that the empirical mean estimator $\nu = \hat{\mu}_1$ has $\mathbb{E}_\theta\left[|\mu_1-\nu|\right] \leq \frac{1}{\sqrt{n_1}}$ for any θ. We know that $\inf_\nu \sup_\theta \mathbb{E}_\theta\left[|\mu_1-\nu|\right] \leq \frac{1}{\sqrt{n_1}}$. Let c_2 be the constant in the definition of $\mathcal{V}^*_{\mathbf{n}}$, then $\frac{c_2 e^{-4c_2^2}}{2\sqrt{n_1}}$ is a lower bound on $\mathcal{E}^*(\theta)$ for any θ due to the fact that relaxing the constraint on the minimax optimality gives a lower instance dependent regret lower bound. Since the minimax value is also an upper bound on $\mathcal{E}^*(\theta)$ we know that, there exist universal constants c_0 and c_1 such that, for any $\theta \in \Theta_{\mathbf{n}}$, $c_0/\sqrt{n_1} \leq \mathcal{E}^*(\theta) \leq c_1/\sqrt{n_1}$.

□

Proof of Proposition 6.5.2. Picking $\delta = \frac{1}{\sqrt{|\mathbf{n}|}}$ for the LCB algorithm, according to Corollary 6.4.6 gives that there exists a universal constant c (which may contain the term $\log k$) such that $\mathcal{R}(\text{LCB}, \theta) \leq \frac{c\sqrt{\log|\mathbf{n}|}}{\sqrt{n_1}}$. Applying Proposition 6.9.6 concludes the proof.

□

Proof of Proposition 6.5.3. Consider a sequence of counts $\mathbf{n}_1, \mathbf{n}_2, ...$ with $n_2 = 1$ and $n_1 = 2, 3, ..., +\infty$. Fix $\mu_1 = \mu_2 + 0.1$ and let $\Delta = \mu_1 - \mu_2$. For the UCB algorithm, we have

$$\begin{aligned}
\mathcal{R}(\text{UCB}, \theta) &= \Delta\mathbb{P}_\theta\left(\hat{\mu}_2 + \frac{\beta_\delta}{\sqrt{n_2}} \geq \hat{\mu}_1 + \frac{\beta_\delta}{\sqrt{n_1}}\right) \\
&= 0.1\mathbb{P}_\theta\left(\hat{\mu}_1 - \hat{\mu}_2 \leq \frac{\beta_\delta}{\sqrt{n_2}} - \frac{\beta_\delta}{\sqrt{n_1}}\right) \\
&\geq 0.1\mathbb{P}_\theta\left(\hat{\mu}_1 - \hat{\mu}_2 \leq \left(1 - \frac{1}{\sqrt{2}}\right)\beta_\delta\right) \\
&\geq 0.1\mathbb{P}_\theta\left(\hat{\mu}_1 - \hat{\mu}_2 \leq \left(1 - \frac{1}{\sqrt{2}}\right)\right) \\
&\geq 0.1\mathbb{P}_\theta\left(\hat{\mu}_1 - \hat{\mu}_2 \leq \Delta\right) \\
&= 0.05
\end{aligned}$$

where we applied the fact that $\beta_\delta \geq 1$ for any $\delta \in (0, 1)$ and the random variable $\hat{\mu}_1 - \hat{\mu}_2$ follows a Gaussian distribution with mean Δ. Applying Proposition 6.9.6 gives

$$\limsup_{j\to\infty} \sup_{\theta\in\Theta_{\mathbf{n}_j}} \frac{\mathcal{R}(\text{UCB}, \theta)}{\sqrt{\log|\mathbf{n}_j|}\cdot\mathcal{E}^*(\theta)} \geq \limsup_{j\to\infty} \frac{0.05\sqrt{j+1}}{c_1\sqrt{\log(j+2)}} = +\infty$$

For the greedy algorithm, we have

$$\mathcal{R}(\text{greedy}, \theta) = 0.1\mathbb{P}_\theta\left(\hat{\mu}_1 - \hat{\mu}_2 \leq 0\right)\,.$$

The random variable $\hat{\mu}_1 - \hat{\mu}_2$ follows a Gaussian distribution with mean $\Delta > 0$ and variance $\frac{1}{n_1} + \frac{1}{n_2} \geq 1$. Since shrinking the variance of $\hat{\mu}_1 - \hat{\mu}_2$ will lower the probability $\mathbb{P}_\theta\left(\hat{\mu}_1 - \hat{\mu}_2 \leq 0\right)$,

we have $\mathcal{R}(\text{greedy}, \theta) \geq 0.1\Phi(-0.1)$ where Φ is the CDF for the standard normal distribution. Now using a similar statement as for the UCB algorithm gives the result.

□

Chapter 7

Future Directions

For unsupervised domain adaptation, an immediate future problem is to extend the PU learning setting to allow multiple classes in the source distribution. A more general future direction is to explore other types of distribution shift that (i) do not assume contained support, (ii) go beyond label shift, and (iii) are more tractable than general covariate shift.

For offline policy optimization, the question of how to implement the pessimistic principle in reinforcement learning is still largely under-explored. For example, following our work, Kostrikov et al. [73], Kumar et al. [78] investigate how to better regularize the learned value and/or policy function towards the behavior policy. One may notice that there is a distinction between regularizing towards the behavior policy (e.g. in Chapter 5) and optimizing the lower confidence bound of value predictions (e.g. in Chapter 6). This is because, in behavior-regularized algorithms, the input (action) distribution is used as a proxy for the uncertainty in value prediction. So a natural question to ask is whether we can incorporate other off-the-shelf uncertainty quantification techniques for deep learning (e.g., Bayesian neural networks, ensembles, or simply parameterize distributional outputs such as using softmax, etc) into the development of offline RL algorithms. However, there is an additional complication in the reinforcement learning scenario: Most of these uncertainty quantification techniques are built for the scenario of learning to predict a fixed target. In reinforcement learning, successful algorithms such as DQN, DDPG and their variants are training value functions to predict a moving target, while slowing down the prediction-target movement may hurt the performance of these algorithms. This distinction makes many of the uncertainty quantification techniques less applicable in the reinforcement learning scenario. So one future direction can be to investigate how to develop reinforcement learning algorithms that can be decomposed into fixed-target prediction problems such that off-the-shelf uncertainty quantification techniques become more applicable.

Bibliography

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018. 5.3

[2] Milton Abramowitz, Irene A Stegun, and Robert H Romer. Handbook of mathematical functions with formulas, graphs, and mathematical tables, 1988. 6.9.1

[3] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. *arXiv preprint arXiv:1907.04543*, 2019. 5.1

[4] Amr Alexandari, Anshul Kundaje, and Avanti Shrikumar. Adapting to label shift with bias-corrected calibration. In *arXiv preprint arXiv:1901.06852*, 2019. 1.2.1, 3.1, 3.2, 3.2, 3.3, 1, 3.4.3, 3.4.3, 3.6, 4.1

[5] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 2.5.2

[6] Kavosh Asadi and Michael L Littman. An alternative softmax operator for reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 243–252. JMLR. org, 2017. 5.3

[7] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. In *International Conference on Learning Representations (ICLR)*, 2019. 3.1, 3.2, 3.3, 3.6, 4.1

[8] Leemon Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995. 5.1

[9] Jessa Bekker and Jesse Davis. Estimating the class prior in positive and unlabeled data through decision tree induction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 4.1, 4.2

[10] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: a survey. *Mach. Learn.*, 2020. 4.2

[11] Shai Ben-David and Ruth Urner. Domain adaptation–can quantity compensate for quality? *Annals of Mathematics and Artificial Intelligence*, 70(3):185–202, 2014. 2.7

[12] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007. 2.4, 2.7

[13] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jen-

nifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010. 2.1, 2.4, 2.7

[14] Shai Ben-David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *International Conference on Artificial Intelligence and Statistics*, pages 129–136, 2010. 1.2.1, 2.1, 3.2

[15] James O Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer New York, New York, NY, January 1985. URL `http://link.springer.com/10.1007/978-1-4757-4286-2`. 6.5

[16] Gilles Blanchard, Gyemin Lee, and Clayton Scott. Semi-supervised novelty detection. *The Journal of Machine Learning Research*, 11:2973–3009, 2010. 4.2, 4.3

[17] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016. 2.7

[18] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 5.4, 5.6.1

[19] Jacob Buckman, Carles Gelada, and Marc G Bellemare. The importance of pessimism in Fixed-Dataset policy optimization. September 2020. URL `http://arxiv.org/abs/2009.06799`. 6.1, 6.3, 6.5, 1, 6.6

[20] Jonathon Byrd and Zachary C Lipton. What is the effect of importance weighting in deep learning? In *International Conference on Machine Learning (ICML)*, 2019. 3.2

[21] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Michael I Jordan. Partial transfer learning with selective adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2724–2732, 2018. 2.7

[22] Zhangjie Cao, Lijia Ma, Mingsheng Long, and Jianmin Wang. Partial adversarial domain adaptation. In *European Conference on Computer Vision*, pages 139–155. Springer, 2018. 2.7

[23] Zhi Chen, Pengqian Yu, and William B Haskell. Distributionally robust optimization for sequential decision-making. *Optimization*, 68(12):2397–2426, 2019. 6.6

[24] Corinna Cortes and Mehryar Mohri. Domain adaptation in regression. In *International Conference on Algorithmic Learning Theory*, pages 308–323. Springer, 2011. 2.7

[25] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198. ACM, 2016. 5.1, 6.1

[26] Imre Csiszár. Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten. *Magyer Tud. Akad. Mat. Kutato Int. Koezl.*, 8: 85–108, 1964. 5.3

[27] Bo Dai, Ofir Nachum, Yinlam Chow, Lihong Li, Csaba Szepesvári, and Dale Schuurmans. Coindice: Off-policy confidence interval estimation. *arXiv preprint arXiv:2010.11652*, 2020. 6.6

[28] Francesco De Comité, François Denis, Rémi Gilleron, and Fabien Letouzey. Positive and unlabeled examples help learning. In *International Conference on Algorithmic Learning Theory*, pages 219–230. Springer, 1999. 4.1, 4.2

[29] Erick Delage, Daniel Kuhn, and Wolfram Wiesemann. "dice"-sion–making under uncertainty: When can a random decision reduce risk? *Management Science*, 65(7):3282–3301, July 2019. 6.6

[30] François Denis. Pac learning from positive statistical queries. In *International Conference on Algorithmic Learning Theory*, pages 112–126. Springer, 1998. 4.2

[31] Esther Derman and Shie Mannor. Distributional robustness and regularization in reinforcement learning. March 2020. URL `http://arxiv.org/abs/2003.02894`. 6.6

[32] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1.1, 4.7

[33] Marthinus Du Plessis, Gang Niu, and Masashi Sugiyama. Convex formulation for learning from positive and unlabeled data. In *International conference on machine learning*, pages 1386–1394, 2015. 4.1, 4.2

[34] Marthinus C Du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. *Advances in neural information processing systems*, 27:703–711, 2014. 4.1, 4.2

[35] Marthinus Christoffel Du Plessis and Masashi Sugiyama. Class prior estimation from positive and unlabeled data. *IEICE TRANSACTIONS on Information and Systems*, 97(5): 1358–1362, 2014. 4.1

[36] Marthinus Christoffel Du Plessis and Masashi Sugiyama. Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, 50: 110–119, 2014. 3.3, 3.4.3, 4.2

[37] John Duchi, Peter Glynn, and Hongseok Namkoong. Statistics of robust optimization: A generalized empirical likelihood approach. October 2016. URL `https://arxiv.org/abs/1610.03425v3`. 6.6

[38] Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, pages 642–669, 1956. 4.9

[39] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008. 4.1, 4.2

[40] Louis Faury, Ugo Tanielian, Flavian Vasile, Elena Smirnova, and Elvis Dohmatob. Distributionally robust counterfactual risk minimization. June 2019. URL `http://arxiv.org/abs/1906.06211`. 6.6

[41] Justin Fu, Aviral Kumar, Matthew Soh, and Sergey Levine. Diagnosing bottlenecks in deep q-learning algorithms. *arXiv preprint arXiv:1902.10250*, 2019. 5.1

[42] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018. 5.1, 5.2.2, 5.6.1

[43] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018. 5.1, 5.4.2

[44] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019. 6.1

[45] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982. 1.1

[46] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016. 1.2.1, 2.1, 2.2, 2.6, 2.7

[47] Saurabh Garg, Yifan Wu, Sivaraman Balakrishnan, and Zachary Lipton. A unified view of label shift estimation. In *Advances in Neural Information Processing Systems*, volume 33, pages 3290–3300, 2020. 1.2.1, 4.1

[48] Saurabh Garg, Yifan Wu, Alexander Smola, Sivaraman Balakrishnan, and Zachary Lipton. Mixture proportion estimation and pu learning: A modern approach. In *To appear*, 2021. 1.2.1

[49] Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. A theory of regularized markov decision processes. In *International Conference on Machine Learning*, pages 2160–2169, 2019. 5.3, 5.3, 2, 5.4.3

[50] Itzhak Gilboa and David Schmeidler. Maxmin expected utility with non-unique prior. *Journal of Mathematical Economics*, 18(2):141–153, 1989. 6.5, 6.6

[51] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 2, 2.5.1

[52] Todd L Graves and Tze Leung Lai. Asymptotically efficient adaptive choice of control laws incontrolled markov chains. *SIAM journal on control and optimization*, 35(3):715–743, 1997. 1.2.2

[53] Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alexander J Smola. A kernel approach to comparing distributions. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 1637. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007. 5.2.2

[54] Arthur Gretton, Alexander J Smola, Jiayuan Huang, Marcel Schmittfull, Karsten M Borgwardt, and Bernhard Schölkopf. Covariate shift by kernel mean matching. *Journal of Machine Learning Research*, 2009. 1.2.1, 2.1, 3.2

[55] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information*

Processing Systems, pages 5767–5777, 2017. 2.5.2, 2.6, 5.3, 5.6.1

[56] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017. 3.1, 3.2, 3.2

[57] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018. 5.1, 5.3

[58] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 3.6, 4.7

[59] James J Heckman. Sample selection bias as a specification error (with an application to the estimation of labor supply functions), 1977. 1.2.1, 2.1

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1.1

[61] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017. 2.7

[62] Jiayuan Huang, Arthur Gretton, Karsten M Borgwardt, Bernhard Schölkopf, and Alex J Smola. Correcting sample selection bias by unlabeled data. In *Advances in neural information processing systems*, pages 601–608, 2007. 1.2.1, 2.1, 2.2

[63] Alex Irpan, Kanishka Rao, Konstantinos Bousmalis, Chris Harris, Julian Ibarz, and Sergey Levine. Off-policy evaluation via off-policy classification. *arXiv preprint arXiv:1906.01624*, 2019. 5.4.6

[64] Dmitry Ivanov. DEDPUL: Difference-of-estimated-densities-based positive-unlabeled learning. *arXiv preprint arXiv:1902.06965*, 2019. 4.1, 4.2, 4.5

[65] Shantanu Jain, Martha White, Michael W Trosset, and Predrag Radivojac. Nonparametric semi-supervised learning of class proportions. *arXiv preprint arXiv:1601.01944*, 2016. 4.1, 4.2

[66] Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019. 5.1, 5.3, 6.1, 6.6

[67] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline RL? 2020. 6.1, 6.3, 6.5, 6.6

[68] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pages 1531–1538, 2002. 5.3

[69] Nikos Karampatziakis, John Langford, and Paul Mineiro. Empirical likelihood for contextual bandits. *arXiv preprint arXiv:1906.03323*, 2019. 6.6

[70] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In *NeurIPS*, 2020. 6.1, 6.3, 6.6

[71] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4.7

[72] Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Advances in neural information processing systems*, pages 1675–1685, 2017. 4.1, 4.2, 4.5

[73] Ilya Kostrikov, Rob Fergus, Jonathan Tompson, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5774–5783. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/kostrikov21a.html`. 7

[74] Alex Krizhevsky and Geoffrey Hinton. Learning Multiple Layers of Features from Tiny Images. Technical report, Citeseer, 2009. 3.6, 4.7

[75] Volodymyr Kuleshov and Percy S Liang. Calibrated structured prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015. 3.5

[76] Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. 3.2, 3.2

[77] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019. 5.1, 5.2.2, 5.4, 5.4.4, 5.6.1, 6.1, 6.6

[78] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020. 6.1, 6.6, 7

[79] Tze Leung Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985. 1.2.2, 6.4.2, 1

[80] Henry Lam. Recovering best statistical guarantees via the empirical Divergence-Based distributionally robust optimization. *Oper. Res.*, 67(4):1090–1105, July 2019. URL `https://doi.org/10.1287/opre.2018.1786`. 6.5

[81] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning*, pages 45–73. Springer, 2012. 5.2.2

[82] Romain Laroche and Paul Trichelair. Safe policy improvement with baseline bootstrapping. *arXiv preprint arXiv:1712.06924*, 2017. 5.1

[83] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661, 2019. 5.3, 2

[84] Tor Lattimore. Improved regret for zeroth-order adversarial bandit convex optimisation. *Mathematical Statistics and Learning*, 2(3):311–334, 2020. 1.2.2

[85] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020. 6.4.2, 1, 6.6

[86] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86, 1998. 3.6, 4.7

[87] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, pages 448–455, 2003. 4.2

[88] Fabien Letouzey, François Denis, and Rémi Gilleron. Learning from positive and unlabeled examples. In *International Conference on Algorithmic Learning Theory*, pages 71–85. Springer, 2000. 4.1, 4.2

[89] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. 2020. 1.2.2, 6.1, 6.6

[90] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 297–306. ACM, 2011. 5.1

[91] Lihong Li, Rémi Munos, and Csaba Szepesvári. Toward minimax off-policy value estimation. 2015. 5.1

[92] Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*. Citeseer, 2003. 4.2

[93] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 5.1

[94] Zachary C Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018. 1.2.1, 2.1, 3.1, 3.2, 3.3, 3.6, 4.1

[95] Bing Liu, Wee Sun Lee, Philip S Yu, and Xiaoli Li. Partially supervised classification of text documents. In *ICML*, volume 2, pages 387–394, 2002. 4.2

[96] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. Building text classifiers using positive and unlabeled examples. In *Third IEEE International Conference on Data Mining*, pages 179–186. IEEE, 2003. 4.2

[97] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366, 2018. 5.4.6

[98] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Provably good batch reinforcement learning without great exploration. *arXiv preprint arXiv:2007.08202*, 2020. 6.1, 6.6

[99] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011. 4.7

[100] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017. 1.1

[101] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learn-

ing bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009. 2.7

[102] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 5.1

[103] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016. 5.3

[104] Susan A Murphy, Mark J van der Laan, James M Robins, and Conduct Problems Prevention Research Group. Marginal mean models for dynamic regimes. *Journal of the American Statistical Association*, 96(456):1410–1423, 2001. 5.1

[105] Ofir Nachum, Mohammad Norouzi, Kelvin Xu, and Dale Schuurmans. Trust-pcl: An off-policy trust region method for continuous control. *arXiv preprint arXiv:1707.01891*, 2017. 5.3, 5.3

[106] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. *arXiv preprint arXiv:1906.04733*, 2019. 5.4.6

[107] Gergely Neu, Anders Jonsson, and Vicenç Gómez. A unified view of entropy-regularized markov decision processes. *arXiv preprint arXiv:1705.07798*, 2017. 5.3, 2, 5.4.3

[108] Jerzy Neyman and Egon Sharpe Pearson. Ix. on the problem of the most efficient tests of statistical hypotheses. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 231(694-706):289–337, 1933. 6.4.2

[109] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-gan: Training generative neural samplers using variational divergence minimization. In *Advances in Neural Information Processing Systems*, pages 271–279, 2016. 2, 2.5.1, 2.9, 5.3

[110] Ian Osband, Daniel Russo, Z Wen, and B Van Roy. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 2017. 5.3

[111] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019. 5.1

[112] Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 5.1

[113] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990. 5.2.1

[114] Harish Ramaswamy, Clayton Scott, and Ambuj Tewari. Mixture proportion estimation via kernel embeddings of distributions. In *International conference on machine learning*, pages 2052–2060, 2016. 4.1, 4.2

[115] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019. 1.1

[116] Marco Saerens, Patrice Latinne, and Christine Decaestecker. Adjusting the outputs of a classifier to new a priori probabilities: a simple procedure. *Neural computation*, 14(1): 21–41, 2002. 1.2.1, 2.1, 3.1, 3.2, 3.3, 3.4.3, 4.1

[117] Tyler Sanderson and Clayton Scott. Class proportion estimation with application to multiclass anomaly rejection. In *Artificial Intelligence and Statistics*, pages 850–858, 2014. 4.2

[118] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On Causal and Anticausal Learning. In *International Conference on Machine Learning (ICML)*, 2012. 3.1

[119] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015. 5.3

[120] Clayton Scott. A rate of convergence for mixture proportion estimation, with application to learning from noisy labels. In *Artificial Intelligence and Statistics*, pages 838–846, 2015. 4.1, 4.2

[121] Lior Shani, Yonathan Efroni, and Shie Mannor. Adaptive trust region policy optimization: Global convergence and faster rates for regularized mdps. *arXiv preprint arXiv:1909.02769*, 2019. 2

[122] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 2.2

[123] Hidetoshi Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000. 1.2.1, 2.1, 3.2

[124] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019. 1.1

[125] Rui Shu, Hung H Bui, Hirokazu Narui, and Stefano Ermon. A dirt-t approach to unsupervised domain adaptation. *arXiv preprint arXiv:1802.08735*, 2018. 2.7

[126] Noah Y Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, and Martin Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020. 6.6

[127] James E Smith and Robert L Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Manage. Sci.*, 52(3):311–322, March 2006. URL `https://doi.org/10.1287/mnsc.1050.0451`. 6.5

[128] James E Smith and Robert L Winkler. The optimizers curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006. 6.6

[129] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014. 4.7

[130] Amos Storkey. When Training and Test Sets Are Different: Characterizing Learning Transfer. *Dataset Shift in Machine Learning*, 2009. 3.1, 4.1

[131] Alexander L. Strehl, John Langford, Lihong Li, and Sham M. Kakade. Learning from logged implicit exploration data. In *Advances in Neural Information Processing Systems 23*, pages 2217–2225, 2011. 6.1

[132] Tobias Sutter, Bart P G Van Parys, and Daniel Kuhn. A general framework for optimal Data-Driven optimization. October 2020. URL `http://arxiv.org/abs/2010.06606`. 6.5, 6.6

[133] Richard S Sutton. On the virtues of linear learning and trajectory distributions. In *Proceedings of the Workshop on Value Function Approximation, Machine Learning Conference*, page 85, 1995. 5.1

[134] Adith Swaminathan and Thorsten Joachims. Batch learning from logged bandit feedback through counterfactual risk minimization. *The Journal of Machine Learning Research*, 16 (1):1731–1755, 2015. 6.1, 6.6

[135] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1.1

[136] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. 5.4, 5.6.1

[137] Joel A Tropp et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 2015. 3.8.1

[138] John N Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. In *Advances in neural information processing systems*, pages 1075–1081, 1997. 5.1

[139] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. 2.1, 2.7

[140] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014. 2.7

[141] Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas B Schön. Evaluating model calibration in classification. In *International Conference on Machine Learning (ICML)*, 2019. 3.2, 3.4.5, 3.6

[142] Aad W van der Vaart and Jon A Wellner. Weak convergence. In *Weak convergence and empirical processes*. Springer, 1996. 3.5

[143] Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018. 5.1

[144] Bart P G Van Parys, Peyman Mohajerin Esfahani, and Daniel Kuhn. From data to decisions: Distributionally robust optimization is optimal. April 2017. URL `http://arxiv.org/abs/1704.04118`. 6.5, 6.6

[145] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 1.1

[146] Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al. Exponentially weighted imitation learning for batched historical data. In *Advances in Neural Information Processing Systems*, pages 6288–6297, 2018. 5.1

[147] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rmi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, 2020. 4.7

[148] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. 1.2.2, 6.1, 6.6

[149] Yifan Wu, Ezra Winston, Divyansh Kaushik, and Zachary Lipton. Domain adaptation with asymmetrically-relaxed distribution alignment. In *International Conference on Machine Learning*, pages 6872–6881, 2019. 1.2.1, 5.4.4

[150] Chenjun Xiao*, Yifan Wu*, Bo Dai, Tor Lattimore, Jincheng Mei, Lihong Li, Csaba Szepesvari, and Dale Schuurmans. On the optimality of batch policy optimization algorithms. In *International Conference on Machine Learning*, pages 11362–11371. PMLR, 2021. 1.2.2

[151] Huan Xu and Shie Mannor. Distributionally robust markov decision processes. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems-Volume 2*, pages 2505–2513, 2010. 6.6

[152] Insoon Yang. A convex optimization approach to distributionally robust markov decision processes with wasserstein distance. *IEEE control systems letters*, 1(1):164–169, 2017. 6.6

[153] Ming Yin, Yu Bai, and Yu-Xiang Wang. Near-optimal offline reinforcement learning via double variance reduction. *arXiv preprint arXiv:2102.01748*, 2021. 6.3

[154] Pengqian Yu and Huan Xu. Distributionally robust counterpart in markov decision processes. *IEEE Transactions on Automatic Control*, 61(9):2538–2543, 2015. 6.6

[155] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. 2020. 6.1, 6.6

[156] Yaoliang Yu and Csaba Szepesvári. Analysis of kernel mean matching under covariate shift. *arXiv preprint arXiv:1206.4650*, 2012. 1.2.1, 2.1

[157] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016. 4.5

[158] Dell Zhang and Wee Sun Lee. A simple probabilistic approach to learning from positive and unlabeled examples. In *Proceedings of the 5th annual UK workshop on computational intelligence (UKCI)*, pages 83–87, 2005. 4.2

[159] Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827, 2013. 1.2.1, 2.1