Part I

Optimal Data Distributions in Machine Learning

Chapter 1

Introduction

A basic assumption in learning theory is that the training and test sets are drawn from the same probability distribution. However, in many practical situations, this assumption about the training and test distributions does not hold. To illustrate this, take, for example, a recommender system. In this case, the system is trained with data gathered over a long period of time during which users rate certain items. Nevertheless, the system will be used and tested not only on old items that the user has not rated yet, but also on new items and new users. Due to changes in opinions, moods, trends, etc., with time, there is no guarantee that the distribution of the test data will be the same as that of the training data. Rather, a realistic assumption is to model this situation as one in which training and test distributions may differ. Other examples where this is the case have been reported in natural language processing [40] and speech recognition [15]. These systems are commonly trained by gathering speech samples from only a few individuals due to resource constraints. However, the system is tested later on the general population and hence training and test distributions are likely to differ. In this case it is not the effect of time that makes the two distributions differ, but rather the effect of having a biased sample. Other examples of differing distributions are commonly found in applications that involve experimental set-ups. Some of these set-ups can involve conditions such as lighting, temperature, etc., which vary from experiment to experiment, making the training and test distributions differ, as in [6].

The problem described above is referred to as dataset shift, and sometimes subdivided into covariate shift and sample selection bias, as described in [54]. Covariate shift occurs when the training distribution P_R of the input variable x is different from the test distribution P_S of the same variable x. Sample selection bias occurs when the sample used for training, is not representative of the overall distribution, due to some bias (intended or unintended) in the sampling process. This can be modeled as having $P_R(x) \neq P_S(x)$, but it also can be modeled with an additional random variable s called the sample selection variable. The selection variable indicates if a sample is included or not in the

training or test sets. In this case, if the overall distribution from which data is sampled is P, then $P_R(x) = P(x|s = 1)$.

There are various methods that have been devised to correct for this problem, and is part of the ongoing work on domain adaptation and transfer learning. Although adjustments to the theory become necessary, and numerous methods that will be described shortly have been devised to correct the problem of mismatched training and test distributions, the fact that the theory requires a matched distribution assumption to go through does not necessarily mean that matched distributions will lead to better performance; just that they lead to theoretically more predictable performance. However, the question of whether they do lead to better performance has not been addressed in the case of supervised learning, perhaps because of an intuitive expectation that the answer would be yes. Hence, in this part of the thesis the work is aimed to answer three fundamental questions in this learning scenario:

1. Is it better, in terms of out-of-sample performance, to have the training distribution P_R equal to the test distribution P_S?
2. If so, is it advantageous to apply weights to the training points to achieve this?
3. What is the algorithmic way to achieve it?

Answering these three questions led to the results reported in this part of the thesis.

1.1 Overview

The seemingly obvious answer to the first question is much more interesting than expected and is the topic of Chapter 2. In that chapter, we first show the simulation setup that led us to conceive the idea that using mismatched training and test distributions could lead to better performance in the supervised learning setting. We present both empirical and analytic results of this evidence.

We then introduce in Chapter 3 the formal notion of the *dual distribution*, which is the optimal training distribution to draw samples from, for the learning algorithm. We then formulate the optimization problem that allows us to find this dual distribution, and describe how to solve for it in the general case. We also analyze various properties and parameters that affect the dual distribution.

Chapter 4 describes the various effects that come into play when weights are used to change the original training distribution. On the one hand, training with data sampled from the dual distribution will improve performance, and so using weights that make the training distribution look like the dual distribution should be advantageous. On the other hand, weighting samples rather than sampling from the desired distribution are not equivalent. The former can have a negative effect, in terms of

an increase in the variance of our error estimates, which can also be viewed as an effective sample size reduction. Each learning scenario yields a different bottom line performance after adding up these effects, so that it is sometimes beneficial to use weights while other times it is not. Hence, we introduce an algorithm that determines when weighting is beneficial in a given practical scenario.

Chapter 5 introduces a class of algorithms that can be used to match the training distribution to any desired distribution, for example, the dual distribution. The algorithms we introduce have the advantage of selecting only the desired coordinates along which matching is desired. They are also efficient so they can be used in very large datasets. The efficiency issue was a constraint that we took into account since we conceived the method initially for recommender systems, which have very large datasets composed of ratings for thousands or millions of items, by thousands or millions of users.

1.2 Literature overview

As discussed, the problem of dataset shift has led to a substantial amount of work aimed at correcting the problem. All of the work assumes that the answer to the first question we pose is affirmative, and hence try to make $P_R = P_S$. The numerous methods can be roughly divided into four types [48].

The first type is referred to as instance weighting for covariate shift, in which weights are given to points in the training set, such that the two distributions become effectively matched. Some of these methods include discriminative approaches as in [13, 14]. To do this, these methods train a classifier that can distinguish between samples coming from the training distribution and samples coming from the test distributions. Other methods make assumptions regarding the source of the bias and explicitly model a selection bias variable [71]. Others try to match the two distributions in some Reproducing Kernel Hilbert Space as Kernel Mean Matching [38], while others use parametric models for the ratio of test to train densities, using the Kullback-Liebler divergence as in KLIEP (Kullback-Liebler importance estimation procedure) [67], or least squares deviation as in LSIF (least squares importance fitting) [42], among others. Additional approaches are given in [57, 27, 55, 64]. A detailed description of these methods is given in Chapter 5. All these methods rely on finding weights, which is not trivial as the actual distributions are not known. Furthermore, the addition of weights reduces the effective sample size of the training set, hurting the out-of-sample performance [61]. Another issue that comes up regards cross-validation, as it becomes necessary to match the distribution of the validation set to the test set. As some of the methods find weights that are only meaningful with respect to the rest of the sample, aggregating weights for different sets in K-fold type validation methods is no longer trivial. This issue is addressed in methods like importance weighting cross-validation [67]. On the theoretical side, learning bounds for the instance weighting setting are shown in [25, 72]. Further theoretical results in a more general setting of learning from different

domains are given in [10].

The second type of methods use self-labeling or co-training techniques so that samples from the test set, which are unlabeled, are introduced in the training set in order to match the distributions, and are labeled using the labeled data. A final model is then re-estimated with these new points. Some of these methods are described in [18, 46, 32]. A third approach is to change the feature representation, so that features are selected, discarded, or transformed in an effort to make training and test distributions similar. This idea is explored in various methods, including [16, 15, 11, 52], among many others. Finally, cluster based methods rely on the assumption that the decision boundaries have low density probabilities [34], and hence try to label new data in regions that are under-represented in the training set through clustering, as proposed in [17, 51]. For a more detailed review on these and other methods, refer to [48] and [64].

1.3 The learning setup

Before we answer the questions presented, we introduce the notation that will be used throughout the thesis and that describes the learning problem. Let $R = \{x_i, y_i\}_{i=1}^N$ be the training set, with $x_i \in \mathcal{X}$, and $y_i \in \mathcal{Y}$. $\mathcal{X}$ is known as the input space, and $\mathcal{Y}$ as the output space. We assume x_i are iid $\sim P_R$, where P_R is the training distribution. The objective of the learning algorithm is to find a hypothesis $h \in \mathcal{H}$ that is closest to the target function f, where $f : \mathcal{X} \to \mathcal{Y}$. $\mathcal{H}$ is known as the hypothesis set, where each $h \in \mathcal{H}$ is $h : \mathcal{X} \to \mathcal{Y}$. The notion of closeness to the target function is determined by a chosen loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. The returned hypothesis by the learning algorithm is denoted by g. Finally, it is conventional to model the noisy data using a stochastic noise process ϵ, where ϵ_i is the corresponding realization for x_i, so that $y_i = f(x_i) + \epsilon_i$. In this framework, learning consists of solving the following optimization problem:

$$g = \arg\min_h \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i), y_i). \tag{1.1}$$

In parametric learning, as the name suggests, the hypothesis set $\mathcal{H}$ is parametrized by $\theta \in \mathcal{Z}^M$, where $\mathcal{Z}^M$ is the M-dimensional space where the parameters live. That is $\mathcal{H} = \{h(\cdot;\theta)|\theta \in \mathcal{Z}^M\}$. The learning algorithm outputs an optimal parameter $\theta^\star \in \mathcal{Z}^M$ given by

$$\theta^\star = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} \ell(h(x_i;\theta), y_i), \tag{1.2}$$

and

$$g(x) = h(x;\theta^\star). \tag{1.3}$$

Now, let $x \sim P_S$ where P_S denotes the test distribution. In the usual learning setting $P_R = P_S$, but here, we consider precisely the scenario where $P_R \neq P_S$. Finally, for a point $x \sim P_S$, the out-of-sample error E_{out} is given by

$$E_{\text{out}}(x, R, f, \epsilon) = \ell(g(x), y). \tag{1.4}$$

The out-of-ample error E_{out} at x, depends not only on the point itself, but also on the dataset R, which in turn determines which $g \in \mathcal{H}$ is returned. Finally, the target function f and the noise process ϵ also affect this error (y depends on f and ϵ). The overall out-of-sample error is the expected value of the pointwise error,

$$E_{\text{out}} = \mathbb{E}_x[E_{\text{out}}(x, R, f, \epsilon)]. \tag{1.5}$$

Here $\mathbb{E}_x[\cdot]$ denotes the expected value of the expression with respect to variable x.