October 15, 2007
RT0756
Computer Science 12 pages

Research Report

Direct density ratio estimation for large-scale covariate shift adaptation

Yuta Tsuboi, Hisashi Kashima, Shohei Hido, Steffen Bickel, and Masashi Sugiyama

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation

Yuta Tsuboi Hisashi Kashima Shohei Hido* Steffen Bickel† Masashi Sugiyama‡

Abstract

Covariate shift is a situation in supervised learning where training and test *inputs* follow different distributions even though the functional relation remains unchanged. A common approach to compensating for the bias caused by covariate shift is to reweight the training samples according to *importance*, which is the ratio of test and training densities. We propose a novel method that allows us to directly estimate the importance from samples without going through the hard task of density estimation. An advantage of the proposed method is that the computation time is nearly independent of the number of test input samples, which is highly beneficial in recent applications with large numbers of unlabeled samples. We demonstrate through experiments that the proposed method is computationally more efficient than existing approaches with comparable accuracy.

Keywords

Covariate Shift, Importance Weight, Massive Unlabeled Data, KLIEP

1 Introduction

An assumption that is commonly imposed—either explicitly or implicitly—in virtually all supervised learning methods is that the training and test samples follow the *same* probability distribution. However, this fundamental assumption is often violated in practice, causing standard machine learning methods not to work as expected. In this paper, we address supervised learning problems in the absence of this fundamental assumption.

If the training and test distributions share nothing in common, we may not be able to learn anything about the test distribution from the training samples. For a meaningful discussion, the training and test distributions should be related to each other in some sense. A situation where the input distribution $p(\boldsymbol{x})$ is different in the training and test phases but the conditional distribution of output values, $p(y|\boldsymbol{x})$, remains unchanged is called *covariate shift* [19]. In many real-world applications such as robot control [25, 18], bioinformatics [1, 5], spam filtering [4], natural language processing [14], brain-computer interfacing [28, 22], or econometrics [12], covariate shift is likely. Covariate shift is also naturally induced in selective sampling or active learning scenarios [8, 7, 27, 15, 21]. For this reason, learning under covariate shift is receiving a lot of attention these days in the machine learning community (such as in the NIPS2006 workshop [6] and the ECML2006 workshop [2]).

Under covariate shift, standard learning methods such as maximum likelihood estimation are no longer *consistent*, i.e., they do not produce the optimal solution even when the number of training samples tends to be infinity. Thus, there exists an estimation bias induced by covariate shift. It has been shown that the bias can be asymptotically canceled by weighting the log likelihood terms according to the *importance* [9, 19, 29]:

$$w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})},$$

where $p_{\mathrm{te}}(\boldsymbol{x})$ and $p_{\mathrm{tr}}(\boldsymbol{x})$ are the test and training input densities. Since the importance is usually unknown in reality, the central issue of practical covariate shift adaptation is how to accurately estimate the importance[1].

A naive approach to importance estimation is to first estimate the training and test densities separately

*Tokyo Research Laboratory, IBM Research.

†Max Planck Institute for Computer Science

‡Department of Computer Science, Tokyo Institute of Technology.

[1] Covariate shift matters in parameter learning only when the model used for function learning is *misspecified* (i.e., the model is so simple that the true learning target function cannot be expressed) [19]. When the model is correctly (or overly) specified, the ordinary maximum likelihood estimation is still consistent. On this basis, there is a criticism that importance weighting is not needed, but just the use of a sufficiently complex model can settle the problem. However, overly complex models result in large estimation variances, and so in practice we need to choose a complex enough but not overly complex model. To choose such an appropriate model, we usually use a model selection technique such as cross validation (CV). However, the ordinary CV score is biased due to covariate shift and we still need to importance-weight the CV score (or any other model selection criteria) for unbiasedness [19, 29, 23, 22]. For this reason, estimating the importance is indispensable when covariate shift occurs.

from the training and test input samples, and then estimate the importance by taking the ratio of the estimated densities. However, density estimation is known to be a hard problem particularly in high dimensional cases [10]. Therefore, this naive approach is usually ineffective—directly estimating the importance *without* estimating the densities is more promising. Therefore, several methods that allow us to directly obtain importance estimates without going through density estimation have been proposed recently, such as *kernel mean matching* (KMM) [13], the logistic regression based method (LogReg) [3], and the *Kullback-Leibler Importance Estimation Procedure* (KLIEP) [24].

KMM is based on a special property of *universal reproducing kernel Hilbert spaces* (Gaussian reproducing kernel Hilbert spaces are typical examples) [20] and KMM allows us to directly obtain the importance estimates at the training input points. Since the KMM optimization problem is formulated as a convex quadratic programming problem, it leads to a unique global solution. KMM has been shown to work well, as long as the kernel parameters such as the Gaussian width are chosen appropriately. However, to the best of our knowledge, there is no reliable method to determine the Gaussian width and the regularization parameter in the KMM algorithm[2]. Therefore, the lack of model selection procedures is a critical limitation of KMM in practical applications.

LogReg builds a probabilistic classifier that separates training input samples from test input samples, and the importance can be directly estimated by LogReg. The maximum likelihood estimation of the LogReg can be formulated as a convex optimization problem, so the unique global optimal solution can be obtained. In addition, since LogReg only solves a standard supervised classification problem, the tuning parameters such as the kernel width and the regularization parameter can be optimized by the standard CV procedure. This is a very useful property in practice.

KLIEP tries to match an importance-based estimation of the test input distribution to the true test input distribution in terms of the Kullback-Leibler divergence. KLIEP solves this matching problem in a non-parametric fashion. The training and test input distributions are not parameterized, but only the importance is parameterized. The KLIEP optimization problem is convex and therefore a unique global optimal solution can be obtained. Furthermore, the global solution tends to be sparse, so it is computationally efficient in the test phase. Since KLIEP is based on the minimization of the Kullback-Leibler divergence, the model selection of KLIEP, such as the choice of the kernel width and the regularization parameter, can be carried out naturally through the *likelihood CV* procedure [10], so no open tuning parameter remains.

As reviewed above, LogReg and KLIEP seem to have advantages over KMM, since they are equipped with built-in model selection procedures. On the other hand, from the viewpoint of scalability, all three of the methods have limitations—in recent applications such as spam filtering [4] and information retrieval [11], the number of test (unlabeled) samples are enormous. The purpose of this paper is to develop a computationally efficient covariate shift adaptation method that can deal with large sets of unlabeled data.

Our new method is primarily based on KLIEP. The key difference is that the original KLIEP uses a linearly parameterized function for modeling the importance, while we adopt a *log-linear* model. By definition, the log-linear model only takes non-negative values. This allows us to reformulate the KLIEP optimization problem as an *unconstrained* convex problem. Then we develop a new scalable estimation procedure whose computation time is nearly independent of the number of test samples. More precisely, we need to scan a large number of test samples only once to compute a summary statistic in the beginning (this pre-computation can be carried out in linear time and constant storage space). The main optimization procedure does not use the test samples themselves, but only uses the summary statistic. Therefore, the computation time of the main optimization procedure is independent of the number of test samples.

The experiments show that the proposed method is computationally much more efficient than the existing approaches. Therefore the range of application of covariate shift adaptation can be greatly enlarged towards large-scale problems. As regards estimation accuracy, we experimentally show that the performance of the proposed method is comparable to the best existing methods for small and middle sized problems (since the existing methods cannot be applied to large-scale problems due to the computational costs). Thus the proposed method can be a useful alternative to the existing covariate shift adaptation methods.

[2]Intuitively, it seems possible to optimize the kernel width and the regularization parameter simply by using CV for the performance of subsequent learning algorithms. However, this is highly unreliable since the ordinary CV score is biased under covariate shift. For unbiased estimation of the prediction performance of subsequent learning algorithms, the CV procedure itself needs to be importance-weighted [29, 22]. Since the importance weight has to have been fixed when model selection is carried out using the importance weighted CV, it cannot be used for model selection of importance estimation algorithms. Note that once the importance weight has been fixed, the importance-weighted CV can be used for model selection of subsequent learning algorithms.

2 Problem Formulation

In this section, we formulate the supervised learning problem under covariate shift and briefly review existing techniques for covariate shift adaptation.

2.1 Supervised learning under covariate shift. Let $\boldsymbol{x} \in \boldsymbol{X} \subset \Re^d$ be an input variable and $y \in Y$ be an output variable. Y is a real space in regression cases or a set of categories in classification cases. In standard supervised learning frameworks, it is assumed that $\boldsymbol{x}$ is independently drawn from an input distribution and y is independently drawn from a conditional distribution both in training and test phases. In contrast, here we consider a situation called *covariate shift* [19], i.e., the input distribution differs in the training and test phases, but the conditional distribution remains unchanged.

Suppose we have independent and identically distributed (i.i.d.) training input samples $D_{\mathrm{tr}} = \{\boldsymbol{x}_i\}_{i=1}^{N_{\mathrm{tr}}}$ from a distribution with strictly positive density $p_{\mathrm{tr}}(\boldsymbol{x})$, and test input samples $D_{\mathrm{te}} = \{\boldsymbol{x}_i\}_{i=1}^{N_{\mathrm{te}}}$ from a distribution with density $p_{\mathrm{te}}(\boldsymbol{x})$. In addition to the input samples, suppose we have training output samples $\{y_i\}_{i=1}^{N_{\mathrm{tr}}}$ drawn from the conditional distribution with conditional density $p(y|\boldsymbol{x} = \boldsymbol{x}_i)$, respectively. Typically, the number N_{tr} of training samples is rather small due to the high labeling cost, while the number N_{te} of test input samples is very large since they are often easily available. We denote training sample pairs of input and output as $Z_{\mathrm{tr}} = \{z_i \mid z_i = (\boldsymbol{x}_i, y_i)\}_{i=1}^{N_{\mathrm{tr}}}$.

We use the following linear model:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle, \tag{2.1}$$

where $\boldsymbol{\theta}$ is the parameter vector, $\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{X} \to \Re^t$ is a basis function of $\boldsymbol{x}$, and $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ denotes the Euclidean inner product between vector $\boldsymbol{u}$ and $\boldsymbol{v}$: $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{l=1}^{d} u_l v_l$. Note that this model can contain a bias parameter by just including a constant basis function in $\boldsymbol{\phi}(\boldsymbol{x})$. Throughout the paper, we suppose that this linear model is not generally specified correctly, i.e., the true input-output function is not necessarily included in the above linear model. Since we do not know the true function class in practice, dealing with misspecified models is quite realistic.

The goal of supervised learning is to learn the parameter $\boldsymbol{\theta}$ so that the output values for the test inputs can be accurately predicted. Thus our error metric (which is usually called the *generalization error*) is given by

$$\iint \mathrm{Loss}(\boldsymbol{x}, y, f_{\boldsymbol{\theta}}(\boldsymbol{x})) p_{\mathrm{te}}(\boldsymbol{x}) p(y|\boldsymbol{x}) d\boldsymbol{x} dy, \tag{2.2}$$

where $\mathrm{Loss}(\boldsymbol{x}, y, f_{\boldsymbol{\theta}}(\boldsymbol{x})) : \boldsymbol{X} \times Y \times Y \to \Re$ is a loss function, such as the squared loss in a regression case or the zero-one loss in a classification case.

In supervised learning under covariate shift, the following quantity called the *test domain importance* plays an important role:

$$w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}. \tag{2.3}$$

The importance can be used for adjusting the difference between the training and test input distributions: for any function $A(\boldsymbol{x})$,

$$\int A(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} = \int A(\boldsymbol{x}) w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x}. \tag{2.4}$$

2.2 Parameter learning under covariate shift. Here we review two typical parameter learning methods under covariate shift: one is *importance weighted least squares* (IWLS) for regression and the other is *importance weighted logistic regression* (IWLR) for classification.

IWLS: A standard learning method in regression scenarios would be ordinary least squares (LS):

$$\hat{\boldsymbol{\theta}}_{\mathrm{LS}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x}, y) \in Z_{\mathrm{tr}}} (f_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2 \right].$$

LS is known to be consistent under a usual setting. However, it is no longer consistent for misspecified models under covariate shift. Instead, IWLS is consistent [19]:

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x}, y) \in Z_{\mathrm{tr}}} w(\boldsymbol{x}) (f_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2 + \lambda \|\boldsymbol{\theta}\|^2 \right], \tag{2.5}$$

where the importance $w(\boldsymbol{x})$ is used as weights. Here we also added a penalty term $\lambda \|\boldsymbol{\theta}\|^2$ for regularization, where λ is a regularization parameter.

For the linear model (2.1), the above optimization problem is convex and the unique global solution $\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}}$ can be computed in a closed-form as

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}} = (\boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\top \boldsymbol{W} \boldsymbol{y},$$

where $\boldsymbol{I}$ is the identity matrix,

$$\Phi_{i,l} = \phi_l(\boldsymbol{x}_i), \quad \boldsymbol{y} = (y_1, y_2, \ldots, y_{N_{\mathrm{tr}}})^\top, \text{ and}$$
$$\boldsymbol{W} = \mathrm{diag}(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{N_{\mathrm{tr}}}).$$

IWLR: For simplicity, we focus on the two-class case, i.e. $Y = \{-1, 1\}$; we note that it is straightforward to extend all of the discussions in this paper to multi-class cases.

Let us model the posterior probability of class y given $\boldsymbol{x}$ using a parametric model $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ as

$$p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \frac{\exp(y f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{1+\exp(y f_{\boldsymbol{\theta}}(\boldsymbol{x}))}. \tag{2.6}$$

Then a test input sample $\boldsymbol{x}$ is classified by choosing the most probable class:

$$\hat{y} = \operatorname*{argmax}_y p_{\boldsymbol{\theta}}(y|\boldsymbol{x}). \tag{2.7}$$

A standard learning method for the above probabilistic classification scenarios would be ordinary logistic regression (LR):

$$\hat{\boldsymbol{\theta}}_{\mathrm{LR}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} \left(\log\left(1+\exp\left(y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right) - y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) \right].$$

Similar to the case of LS, LR is consistent under a usual setting, but is no longer consistent for misspecified models under covariate shift. Instead, IWLR is consistent:

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLR}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} w(\boldsymbol{x}) \left(\log\left(1+\exp\left(y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right) - y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) + \lambda\|\boldsymbol{\theta}\|^2 \right]. \tag{2.8}$$

Here we also added a penalty term $\lambda\|\boldsymbol{\theta}\|^2$ for regularization, where λ is a regularization parameter.

This optimization problem is known to be convex and a unique optimal solution can be computed using standard non-linear optimization techniques such as a gradient ascent method or some variants of the Newton method. The gradient of the above objective function is given by

$$\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} w(\boldsymbol{x}) \left(y p_\theta(y|\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x}) - y\boldsymbol{\phi}(\boldsymbol{x})\right) + 2\lambda\boldsymbol{\theta}. \tag{2.9}$$

2.3 Model selection under covariate shift. In the above learning methods, the choice of model parameters such as the basis functions $\boldsymbol{\phi}$ and the regularization parameter λ heavily affects the prediction performance. This problem is called model selection and is one of the key concerns in machine learning.

A popular model selection method in the machine learning community would be cross validation (CV). The performance of CV is guaranteed in the sense that it gives an unbiased estimate of the generalization error. However, this useful theoretical property is no longer true under covariate shift [29]. To cope with this problem, a variant of CV called *importance weighted CV* (IWCV) has been proposed for model selection under covariate shift [22]. It has been proved that IWCV gives an unbiased estimate of the generalization error even under the covariate shift.

Here, let us briefly describe the IWCV procedure. We first divide the training samples $\{z_i\}_{i=1}^{N_{\mathrm{tr}}}$ into R disjoint subsets $\{Z^r\}_{r=1}^{R}$. Then we learn a function $f_{\boldsymbol{\theta}}^r(\boldsymbol{x})$ from $\{Z^j\}_{j\neq r}$ by IWLS/IWLR and compute its mean test error for the remaining samples Z^r:

$$\frac{1}{|Z^r|}\sum_{(\boldsymbol{x},y)\in Z^r} w(\boldsymbol{x})\left(f_{\boldsymbol{\theta}}^r(\boldsymbol{x})-y\right)^2, \quad \text{(regression)}$$

$$\frac{1}{|Z^r|}\sum_{(\boldsymbol{x},y)\in Z^r} w(\boldsymbol{x}) I(\hat{y}=y), \quad \text{(classification)}$$

where $I(\cdot)$ denotes the indicator function. We repeat this procedure for $r=1,2,\ldots,R$ and choose the model such that the average of the above mean test error is minimized.

3 Importance Estimation

As we have seen in the previous section, the importance $w(\boldsymbol{x})$ plays a central role in covariate shift adaptation. However, the importance is unknown in practice so we need to estimate it from samples.

Direct importance estimation methods that do not involve density estimation steps have been developed recently [13, 3, 24]. Here we review one of those direct methods called the *Kullback-Leibler Importance Estimation Procedure* (KLIEP) [24]. Other methods will be reviewed in Section 6.

3.1 KLIEP. Let us model $w(\boldsymbol{x})$ with the following linear model:

$$\hat{w}(\boldsymbol{x}) = \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle, \tag{3.10}$$

where $\boldsymbol{\alpha} \in \Re^b$ is a model parameter vector. Since the importance should be non-negative by definition, we suppose that both $\boldsymbol{\alpha}$ and $\boldsymbol{\psi}(\boldsymbol{x})$ are non-negative.

Using the importance estimation $\hat{w}(\boldsymbol{x})$, we can estimate the test input density $p_{\mathrm{te}}(\boldsymbol{x})$ by

$$\hat{p}_{\mathrm{te}}(\boldsymbol{x}) = p_{\mathrm{tr}}(\boldsymbol{x})\hat{w}(\boldsymbol{x}). \tag{3.11}$$

Now we learn the parameter $\boldsymbol{\alpha}$ so that the Kullback-Leibler divergence from $p_{\mathrm{te}}(\boldsymbol{x})$ to $\hat{p}_{\mathrm{te}}(\boldsymbol{x})$ is minimized:

$$\begin{aligned} KL[p_{\mathrm{te}}(\boldsymbol{x})||\hat{p}_{\mathrm{te}}(\boldsymbol{x})] &= \int_D p_{\mathrm{te}}(\boldsymbol{x}) \log \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})\hat{w}(\boldsymbol{x})} d\boldsymbol{x} \\ &= \int_D p_{\mathrm{te}}(\boldsymbol{x}) \log \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} d\boldsymbol{x} - \int_D p_{\mathrm{te}}(\boldsymbol{x}) \log \hat{w}(\boldsymbol{x}) d\boldsymbol{x}. \end{aligned} \tag{3.12}$$

Since the first term in Eq.(3.12) is independent of $\boldsymbol{\alpha}$, we ignore it and focus on the second term, which we denote by J_{KLIEP}:

$$J_{\mathrm{KLIEP}} = \int_D p_{\mathrm{te}}(\boldsymbol{x}) \log \hat{w}(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \log \hat{w}(\boldsymbol{x}), \tag{3.13}$$

where an empirical approximation based on the test input samples is used. This is the objective function to be maximized. The value of $\hat{w}(\boldsymbol{x})$ should be properly normalized since it is a probability density function:

$$1 = \int_D \hat{p}_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} = \int_D p_{\mathrm{tr}}(\boldsymbol{x}) \hat{w}(\boldsymbol{x}) d\boldsymbol{x} \approx \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \hat{w}(\boldsymbol{x}), \tag{3.14}$$

where the empirical approximation based on the training samples is used.

Then the resulting optimization problem is expressed as

$$\begin{aligned} &\underset{\boldsymbol{\alpha}}{\text{maximize}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \log \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle \\ &\quad \text{subject to} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle = N_{\mathrm{tr}} \text{ and } \boldsymbol{\alpha} \geq \mathbf{0}, \end{aligned}$$

which is convex. Thus the global solution can be obtained by iteratively performing gradient ascent and feasibility satisfaction.

3.2 Model selection by likelihood CV. The performance of KLIEP depends on the choice of the basis functions $\psi(\boldsymbol{x})$ (and possibly an additional regularization parameter). Since KLIEP is based on the maximization of the score J_{KLIEP}, it would be natural to select the model such that J_{KLIEP} is maximized. The expectation over $p_{\mathrm{te}}(\boldsymbol{x})$ involved in J_{KLIEP} can be numerically approximated by *likelihood CV* (LCV) [10] as follows: First, divide the test samples D_{te} into R disjoint subsets $\{D_{\mathrm{te}}^r\}_{r=1}^R$. Then, obtain an importance estimate $\hat{w}^r(\boldsymbol{x})$ from $\{D_{\mathrm{te}}^t\}_{t \neq r}^R$ and approximate the score J_{KLIEP} using D_{te}^r as

$$\hat{J}_{\mathrm{KLIEP}}^r = \frac{1}{|D_{\mathrm{te}}^r|} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}^r} \hat{w}^r(\boldsymbol{x}). \tag{3.15}$$

This procedure is repeated for $r = 1, 2, \ldots, R$ and choose the model such that the average of $\hat{J}_{\mathrm{KLIEP}}^r$ for all r is maximized.

One of the potential general limitations of CV is that it is not reliable in small sample cases, since data splitting by CV further reduces the sample size. A key advantage of the LCV procedure is that, not the training samples, but the test input samples are cross-validated. This contributes greatly to improving the model selection accuracy, since the number of training samples is typically limited while there are lots of test input samples available.

As basis functions, it is suggested to use Gaussian kernels centered at a subset of the test input points D_{te} [24]:

$$K_s(\boldsymbol{x}, \boldsymbol{x}_l) = \exp\left\{ -\frac{\|\boldsymbol{x} - \boldsymbol{x}_l\|^2}{2s^2} \right\}, \tag{3.16}$$

where $\boldsymbol{x}_l \in D_{\mathrm{te}}$ is a template test sample and s is the kernel width. This is a heuristic to allocate many kernels at high test input density regions since many kernels may be needed in the region where the output of the target function is large. In the original paper, the number of Gaussian centers was fixed at $N_{\mathrm{te}}/10$ for computational efficiency and the kernel width s was chosen by LCV.

4 KLIEP for Log-linear Models

As shown above, KLIEP has its own model selection procedure and has been shown to work well in importance estimation [24]. However, it has a weakness in computation time. In each step of gradient ascent, the summation over all test input samples needs to be computed, which is prohibitively slow in large-scale problems. The main contribution of this paper is to extend KLIEP so that it can deal with large sets of test input data.

4.1 LL-KLIEP. In the original KLIEP, a linearly parameterized model (3.10) is used for modeling the importance function. Here, we propose using a (normalized) log-linear model[3] for modeling the importance $w(\boldsymbol{x})$ as

$$\hat{w}(\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}') \rangle)}, \tag{4.17}$$

where the denominator guarantees the normalization constraint (3.14). By definition, the log-linear model takes only non-negative values. Therefore, we no longer

[3] The log-linear model can have numerical problems since it contains an exponential function. To cope with this problem, we do not directly compute the value of $\hat{w}(\boldsymbol{x})$, but we compute it in the exponential of the logarithmic domain, i.e.,

$$\exp(\log \hat{w}(\boldsymbol{x})) = \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \log \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)).$$

To further stabilize the computation, we compute the logarithmic sum of the exponential functions as

$$\log(\exp(a) + \exp(b)) = \log(1 + \exp(b - a)),$$

where we pick the smaller exponent as b.

need the non-negative constraint for the parameter (and the basis functions). Then the optimization problem becomes *unconstrained*:

$$\underset{\boldsymbol{\alpha}}{\text{maximize}}\, J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha}),$$

where

$$J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha}) = \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \log \hat{w}(\boldsymbol{x})$$

$$= \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \log \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle). \tag{4.18}$$

Below, we refer to this method as *LL-KLIEP* (log-linear KLIEP). In practice, we may add a penalty term for regularization:

$$\jmath(\boldsymbol{\alpha}) = J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha}) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}, \tag{4.19}$$

where σ^2 is a regularization parameter.

An advantage of LL-KLIEP over the original KLIEP is its computational efficiency. The gradient of $\jmath(\boldsymbol{\alpha})$ can be computed as

$$\begin{aligned} \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} =& \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \boldsymbol{\psi}(\boldsymbol{x}) \\ & - \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\sum_{\boldsymbol{x}' \in D_{\mathrm{te}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}') \rangle)} \boldsymbol{\psi}(\boldsymbol{x}) - \frac{\boldsymbol{\alpha}}{\sigma^2} \\ =& F - \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \hat{w}(x) \boldsymbol{\psi}(\boldsymbol{x}) - \frac{\boldsymbol{\alpha}}{\sigma^2}, \end{aligned} \tag{4.20}$$

where

$$F = \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \boldsymbol{\psi}(\boldsymbol{x}).$$

This means that once we pre-compute the value of F, we do not need to use the test samples when we compute the gradient. This contributes greatly to reducing the computation time when the number of test samples is large. In addition, we do not need to store all of the test samples in memory since we only need the value of F. The required storage capacity is only $\Omega(cN_{\mathrm{tr}})$, where c is the average number of non-zero basis entries.

Although the above optimization procedure may be more efficient than original KLIEP, there still exists a potential weakness: we still need to use all test samples when computing the values of $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ or $\jmath(\boldsymbol{\alpha})$. The value of $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ is needed we choose a model by LCV, and the value of $\jmath(\boldsymbol{\alpha})$ is often utilized in line search or in the stopping criterion.

4.2 LL-KLIEP(LS). Here, we introduce another optimization technique for LL-KLIEP that enables us to overcome the above weakness. Our basic idea is to encourage the derivative of the convex objective function to be zero. We use a squared norm to measure the 'magnitude' of the derivative (4.20):

$$\jmath_{\mathrm{LS}}(\boldsymbol{\alpha}) = \frac{1}{2} \left\| \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \right\|^2. \tag{4.21}$$

The partial derivative of Eq.(4.21) with respect to $\boldsymbol{\alpha}$ is expressed as

$$\frac{\partial \jmath_{\mathrm{LS}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{\partial^2 \jmath(\boldsymbol{\alpha})}{\partial^2 \boldsymbol{\alpha}} \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}. \tag{4.22}$$

This means that the computational complexity of the above derivative is $O(b^2 N_{\mathrm{tr}})$, which is independent of N_{te}. Also, the required storage space is independent of N_{te}: $\Omega(b^2 + cN_{\mathrm{tr}})$. We refer to this approach as LL-KLIEP(LS1) below.

The computation time and storage space of LL-KLIEP(LS1) are quadratic functions of the number of parameters b, which could be a bottleneck in high dimensional problems. To cope with this problem, we make use of the *representer theorem* [26]. Our idea is to represent the parameter $\boldsymbol{\alpha}$ as a linear combination of the input samples:

$$\boldsymbol{\alpha} = \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x}) \beta_{\boldsymbol{x}},$$

where $\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in D_{\mathrm{tr}}}$ is a data-wise parameter. Then Eq.(4.21) can be rewritten as

$$\jmath_{\mathrm{LS}}(\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in D_{\mathrm{tr}}}) = \frac{1}{2} \left\| F - \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x}) \omega(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{\boldsymbol{\psi}(\boldsymbol{x}) \beta_{\boldsymbol{x}}}{\sigma^2} \right\|^2, \tag{4.23}$$

where

$$\omega(\boldsymbol{x}) = \frac{\exp(\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} K(\boldsymbol{x}, \boldsymbol{x}') \beta_{\boldsymbol{x}'})}{\sum_{\boldsymbol{x}'' \in D_{\mathrm{tr}}} \exp(\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} K(\boldsymbol{x}'', \boldsymbol{x}') \beta_{\boldsymbol{x}'})}, \tag{4.24}$$

$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{x}') \rangle.$$

The partial derivative of Eq.(4.23) with respect to $\beta_{\boldsymbol{x}}$ is:

$$\begin{aligned} \frac{\partial \jmath_{\mathrm{LS}}(\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in D_{\mathrm{tr}}})}{\partial \beta_{\boldsymbol{x}}} = & \\ \left\langle F - \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x}') \left(\omega(\boldsymbol{x}') - \frac{\beta_{\boldsymbol{x}'}}{\sigma^2} \right), \right. & \\ \left. \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \omega(\boldsymbol{x}') \boldsymbol{\psi}(\boldsymbol{x}') \langle \boldsymbol{\varphi}(\boldsymbol{x}'), \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \frac{\boldsymbol{\psi}(\boldsymbol{x})}{\sigma^2} \right\rangle, & \end{aligned} \tag{4.25}$$

Table 1: Computational complexity and space requirements.

	Computational complexity			Space requirement	
	Pre. Comp. (once)	Objective	Derivative	Objective	Derivative
KLIEP	0	$O(bN_{\mathrm{tr}} + bN_{\mathrm{te}})$	$O(bN_{\mathrm{tr}} + bN_{\mathrm{te}})$	$\Omega(cN_{\mathrm{tr}} + cN_{\mathrm{te}})$	$\Omega(cN_{\mathrm{tr}} + cN_{\mathrm{te}})$
LL-KLIEP	$O(bN_{\mathrm{te}})$	$O(bN_{\mathrm{tr}} + bN_{\mathrm{te}})$	$O(bN_{\mathrm{tr}})$	$\Omega(cN_{\mathrm{tr}} + cN_{\mathrm{te}})$	$\Omega(cN_{\mathrm{tr}})$
LL-KLIEP(LS1)	$O(bN_{\mathrm{te}})$	$O(bN_{\mathrm{tr}})$	$O(b^2N_{\mathrm{tr}})$	$\Omega(cN_{\mathrm{tr}})$	$\Omega(b^2 + cN_{\mathrm{tr}})$
LL-KLIEP(LS2)	$O(bN_{\mathrm{te}})$	$O(bN_{\mathrm{tr}}^2)$	$O(bN_{\mathrm{tr}}^2)$	$\Omega(cN_{\mathrm{tr}})$	$\Omega(N_{\mathrm{tr}}^2 + cN_{\mathrm{tr}})$

where $\boldsymbol{\varphi}(\boldsymbol{x}) = \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \omega(\boldsymbol{x}')\boldsymbol{\psi}(\boldsymbol{x}') - \boldsymbol{\psi}(\boldsymbol{x})$. We refer to this approach as LL-KLIEP(LS2).

The computation of LL-KLIEP(LS2) requires $O(bN_{\mathrm{tr}}^2)$ time and $\Omega(N_{\mathrm{tr}}^2 + cN_{\mathrm{tr}})$ space. The computation time is linear with respect to the number of parameters b and the storage space is independent of b. This is a significant improvement over the direct computation of the partial derivative in Eq.(4.22).

For LL-KLIEP(LS), LCV can also be computed very efficiently. In each validation set using D_{te}^r, we can compute the validation error as

$$\hat{J}^r_{\text{LL-KLIEP(LS)}} = \left\| F^r - \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \hat{w}^r(\boldsymbol{x})\boldsymbol{\psi}(\boldsymbol{x}) \right\|^2,$$

where

$$F^r = \frac{1}{|D_{\mathrm{te}}^r|} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}^r} \boldsymbol{\psi}(\boldsymbol{x}).$$

Note that, once the mean basis vectors F^r are calculated for all R disjoint subsets of D_{te}, $\hat{J}^r_{\text{LL-KLIEP(LS)}}$ can be evaluated independently of the size of the test data D_{te}^r.

The computational complexity and storage space of each method are summarized in Table 1.

5 Illustrative Examples

In this section, we illustrate the behavior of the proposed LL-KLIEP and show how it can be applied in covariate shift adaptation.

5.1 Regression under covariate shift. Let us consider an illustrative regression problem of learning

$$f(x) = \mathrm{sinc}(x).$$

Let the training and test input densities be $p_{\mathrm{tr}}(x) = \mathcal{N}(x; 1, 1^2)$ and $p_{\mathrm{te}}(x) = \mathcal{N}(x; 1, 0.5^2)$, where $\mathcal{N}(x; \mu, \sigma^2)$ denotes the Gaussian density with mean μ and variance σ^2. We create the training output value $\{y_i\}_{i=1}^{N_{\mathrm{tr}}}$ as $y_i = f(x_i) + \epsilon_i$, where the noise $\{\epsilon_i\}_{i=1}^{N_t r}$ has density $\mathcal{N}(\epsilon; 0, 0.25^2)$. Let the number of training samples be $N_{\mathrm{tr}} = 200$ and the number of test samples be $N_{\mathrm{te}} = 1000$. These settings imply that we are considering an extrapolation problem (see Figure 1(a)).

We used 100 Gaussian basis functions centered at randomly chosen test input samples. Figure 1(b) shows the true importance $w(x)$ and an estimated importance $\hat{w}(x)$ by using LL-KLIEP, where the hyper-parameters such as the Gaussian width and the regularization parameter are selected by LCV. We also tested LL-KLIEP(LS1) and LL-KLIEP(LS2), but we omit their graphs since their solutions are are almost identical to the solution of LL-KLIEP.

Table 2: Specifications of illustrative classification data.

		Training $p_{\mathrm{tr}}(\boldsymbol{x}, y)$		Test $p_{\mathrm{te}}(\boldsymbol{x}, y)$	
		$y = 0$	$y = 1$	$y = 0$	$y = 1$
Fig. 2(a)	μ	(-1,-1)	(3,-1)	(0,3.5)	(4,2.5)
	Σ	$\begin{pmatrix} 0.25 & 0 \\ 0 & 4 \end{pmatrix}$		$\begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$	
Fig. 2(b)	μ	(-1,0)	(4,2)	(0,2)	(3,1)
	Σ	$\begin{pmatrix} 0.75 & 0 \\ 0 & 1.5 \end{pmatrix}$		$\begin{pmatrix} 0.75 & 0.5 \\ 0.01 & 0.1 \end{pmatrix}$	

Figure 1(c) depicts the values of the true $J_{\mathrm{LL-KLIEP}}$ (see Eq.(4.18)) and its estimate by 5-fold LCV. The means, the 25 percentiles, and the 75 percentiles over each validation are plotted as functions of the kernel width s for $\sigma = 1$. We also plot the normalized mean squared error of the estimated importance:

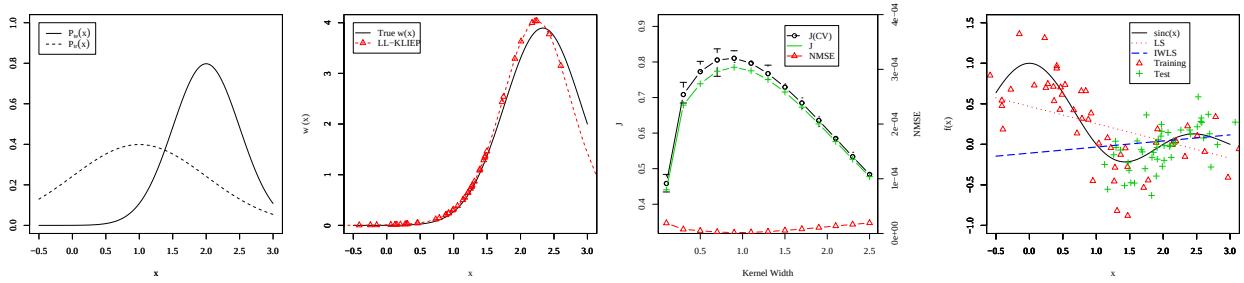
(5.26)

$$\mathrm{NMSE} = \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \left(\frac{\hat{w}(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \hat{w}(\boldsymbol{x}')} - \frac{w(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} w(\boldsymbol{x}')} \right)^2.$$

The graph shows that LCV gives a very good estimate of $J_{\mathrm{LL-KLIEP}}$ and also NMSE.

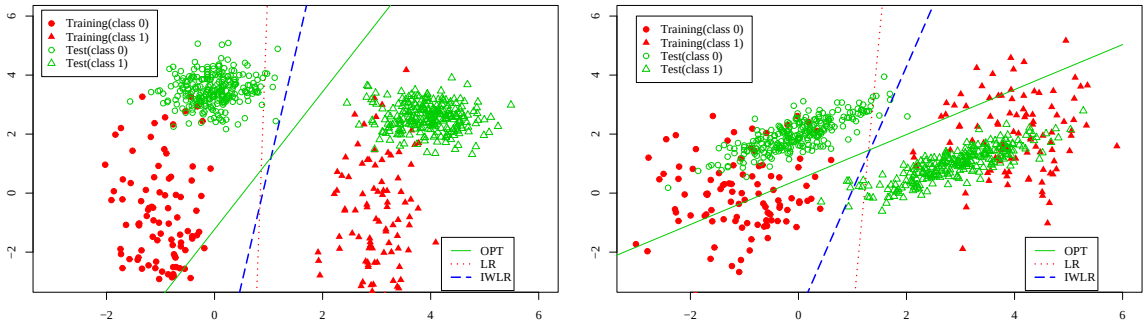
Figure 1(d) shows the true learning target function and functions learned by ordinary LS and IWLS with a linear basis function, such as $\boldsymbol{\phi}(x) = (1, x)^\top$ (section 2.2). The regularization parameter λ was selected by CV for LS and IWCV for IWLS (section 2.3). The results show that the learned function using IWLS goes reasonably well through the test samples, while that of ordinary LS overfits the training samples. Note that the output of the test samples are not used to obtain the learned functions.

5.2 Classification under covariate shift. Next, let us consider two illustrative binary classification problems, where two-dimensional samples were generated from Gaussian distributions (see Table 2 and Figure 2).



(a) Training and test densities (b) Estimated importance (c) Model selection curve (d) True and learned functions

Figure 1: Regression under covariate shift.



(a) $p_{\mathrm{te}}(\boldsymbol{x})$ is linearly shifted from $p_{\mathrm{tr}}(\boldsymbol{x})$. (b) $p_{\mathrm{te}}(\boldsymbol{x})$ is rotated from $p_{\mathrm{tr}}(\boldsymbol{x})$.

Figure 2: Classification examples under covariate shift.

These data sets correspond to a 'linear shift' and a 'non-linear shift' (rotation).

Let the number of the training samples be $N_{\mathrm{tr}} = 200$ and that of the test samples be $N_{\mathrm{te}} = 1000$ (only 500 test samples are plotted for clarity). We used LR/IWLR for the training classifiers (see section 2.2), and employed CV/IWCV for the regularization parameter tuning (see section 2.3). We used a linear basis function for IWLR: $\boldsymbol{\phi}(\boldsymbol{x}) = (1, \boldsymbol{x}^\top)^\top$.

Figure 2 shows the decision boundaries obtained by LR+CV and IWLR+IWCV. For references, we also show 'OPT', which is the optimal decision boundary obtained using the test input-output samples. For the data set depicted in Figure 2(a), the correct classification rate of LR+CV is 99.1% while that of IWLR+IWCV is 100%. For the data set depicted in Figure 2(b), the correct classification rate of LR+CV is 97.2% while that of IWLR+IWCV is 99.1%. Thus, for both cases, the prediction performance is improved by importance weighting.

6 Discussion

In this section, we compare the proposed LL-KLIEP with existing importance estimation approaches.

6.1 Kernel density estimator. The kernel density estimator (KDE) is a non-parametric technique to estimate a density $p(\boldsymbol{x})$ from samples $\{\boldsymbol{x}_l\}_{l=1}^N$. For the Gaussian kernel, KDE is expressed as

$$\hat{p}(\boldsymbol{x}) = \frac{1}{(2\pi s^2)^{d/2} N} \sum_{l=1}^{N} K_s(\boldsymbol{x}, \boldsymbol{x}_l), \tag{6.27}$$

where $K_s(\boldsymbol{x}, \boldsymbol{x}')$ is the Gaussian kernel (3.16). The performance of KDE depends on the choice of the kernel width s, which can be optimized by LCV [10]. Note that LCV corresponds to choosing s such that the Kullback-Leibler divergence from $p(\boldsymbol{x})$ to $\hat{p}(\boldsymbol{x})$ is minimized.

KDE can be used for importance estimation by first obtaining $\hat{p}_{\mathrm{tr}}(\boldsymbol{x})$ and $\hat{p}_{\mathrm{te}}(\boldsymbol{x})$ separately from $\{\boldsymbol{x}_i\}_{i=1}^{N_{\mathrm{tr}}}$ and $\{\boldsymbol{x}_i\}_{i=1}^{N_{\mathrm{te}}}$ and then estimating the importance as $\hat{w}(\boldsymbol{x}) = \hat{p}_{\mathrm{te}}(\boldsymbol{x})/\hat{p}_{\mathrm{tr}}(\boldsymbol{x})$. A potential limitation of this approach is that KDE suffers from the *curse of dimensionality* [10], since the number of samples needed to maintain the same approximation quality grows exponentially as the dimension of the input space increases. This is particularly critical when estimating $p_{\mathrm{tr}}(\boldsymbol{x})$ since the number of training input samples is typically limited. In addition, model selection by LCV is unreliable in such cases, since data splitting in the CV procedure further reduces the sample size. Therefore, in high-dimensional cases LL-KLIEP may be more reliable than the KDE-based approach.

6.2 Kernel mean matching. The kernel mean matching (KMM) method avoids density estimation and directly gives an estimate of the importance at the train-

ing input points [13]. The basic idea of KMM is to find $w(\boldsymbol{x})$ such that the maximum difference of the means of nonlinearly transformed samples drawn from $p_{\mathrm{te}}(\boldsymbol{x})$ and $\hat{w}(\boldsymbol{x})p_{\mathrm{tr}}(\boldsymbol{x})$ is minimized in some feature space $\mathcal{F}$:

$$\min_{w(\boldsymbol{x})} \sup_{f:f\in\mathcal{F},\|f\|_{\mathcal{F}}=1} \|\mathrm{E}_{\mathrm{te}}[f(\boldsymbol{x})] - \mathrm{E}_{\mathrm{tr}}[w(\boldsymbol{x})f(\boldsymbol{x})]\|_{\mathcal{F}}^2$$
$$\text{subject to } \mathrm{E}_{\mathrm{tr}}[w(\boldsymbol{x})] = 1 \text{ and } w(\boldsymbol{x}) \geq 0.$$

It has been shown [13] that the solution of this problem agrees with the true importance if $\mathcal{F}$ is a *universal reproducing kernel Hilbert space* [20]. The Gaussian kernel (3.16) is known to induce a universal reproducing kernel Hilbert space and an empirical version of the above problem is expressed by the following quadratic program.

$$\min_{\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}} \left[\frac{1}{2}\sum_{\boldsymbol{x},\boldsymbol{x}'\in D_{\mathrm{tr}}} w(\boldsymbol{x})w(\boldsymbol{x}')K_\sigma(\boldsymbol{x},\boldsymbol{x}') - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x})\kappa(\boldsymbol{x})\right]$$
$$\text{subject to } \left|\sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x}) - N_{\mathrm{tr}}\right| \leq N_{\mathrm{tr}}\epsilon, \text{ and}$$
$$0 \leq w(\boldsymbol{x}) \leq B \text{ for all } \boldsymbol{x} \in D_{\mathrm{tr}},$$

where

$$\kappa(\boldsymbol{x}) = \frac{N_{\mathrm{tr}}}{N_{\mathrm{te}}} \sum_{\boldsymbol{x}'\in D_{\mathrm{te}}} K_\sigma(\boldsymbol{x},\boldsymbol{x}').$$

B (≥ 0) and ϵ (≥ 0) are tuning parameters. The solution $\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}$ is an estimate of the importance at the training input points.

Since KMM does not require density estimates, it is expected to work well even in high dimensional cases. However, the performance is dependent on the tuning parameters B, ϵ, and σ and they cannot be optimized easily, e.g., by CV, since estimates of the importance are available only at the training input points. Thus, an out-of-sample extension is needed to apply KMM in the CV framework, but this currently seems to be an open research issue.

Here, we show that LL-KLIEP(LS2) (see Eq.(4.23)) has a tight connection to KMM. Up to irrelevant constants, Eq.(4.23) without a regularizer can be expressed as

$$\frac{1}{2}\sum_{\boldsymbol{x},\boldsymbol{x}'\in D_{\mathrm{tr}}} w(\boldsymbol{x})w(\boldsymbol{x}')K_\sigma(\boldsymbol{x},\boldsymbol{x}') - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x})\kappa(\boldsymbol{x}),$$

which is exactly the same form as the objective function of KMM. Thus, KMM and LL-KLIEP(LS2) share a common objective function, although they are derived from very different frameworks.

However, KMM and LL-KLIEP(LS2) still have a significant difference—KMM directly optimizes the importance values $\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}$, while LL-KLIEP(LS2) optimizes the parameter $\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}$ in the importance model (4.24). Thus, LL-KLIEP(LS2) learns the entire importance function and therefore it allows us to *interpolate* the value of the importance function at any input point. This interpolation property is a significant advantage over KMM since it allows us to use LCV for model selection. Therefore, LL-KLIEP(LS2) may be regarded as an extension of KMM.

6.3 Logistic regression discriminating training and test input data. Another method to directly estimate the importance weights is to use a probabilistic classifier. Let us assign a selector variable $\delta = -1$ to the training input samples and $\delta = 1$ to the test input samples. This means that the training and test input densities are written as

$$p_{\mathrm{tr}}(\boldsymbol{x}) = p(\boldsymbol{x}|\delta = -1), \quad p_{\mathrm{te}}(\boldsymbol{x}) = p(\boldsymbol{x}|\delta = 1).$$

A simple calculation shows that the importance can be expressed in terms of δ as [3]:

$$w(\boldsymbol{x}) = \frac{p(\delta=-1)}{p(\delta=1)}\frac{p(\delta=1|\boldsymbol{x})}{p(\delta=-1|\boldsymbol{x})}. \tag{6.28}$$

The probability ratio $p(\delta = -1)/p(\delta = 1)$ may be simply estimated using the ratio of the numbers of training and test input samples. The conditional probability $p(\delta|\boldsymbol{x})$ may be learned by discriminating between the test input samples and the training input samples using LR, where δ plays the role of a class variable (cf. Eq.(2.6)). Let us train the LR model by regularized maximum likelihood estimation. The objective function to be maximized is given by

$$\mathrm{LR}(\boldsymbol{\alpha}) = \sum_{\boldsymbol{x}\in D_{\mathrm{te}}\cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}}\langle\boldsymbol{\alpha},\boldsymbol{\psi}(\boldsymbol{x})\rangle$$
$$- \sum_{\boldsymbol{x}\in D_{\mathrm{te}}\cup D_{\mathrm{tr}}} \log(1+\exp(\delta_{\boldsymbol{x}}\langle\boldsymbol{\alpha},\boldsymbol{\psi}(\boldsymbol{x})\rangle)) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}, \tag{6.29}$$

where the first term is the main likelihood term, the second term is a normalizer, , and the third term is a regularizer. Since this is a convex optimization problem, the global solution can be obtained by standard nonlinear optimization methods. The gradient of the objective function is given as

$$\frac{\partial \mathrm{LR}(\boldsymbol{\alpha})}{\partial\boldsymbol{\alpha}} = \sum_{\boldsymbol{x}\in D_{\mathrm{te}}\cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}}\boldsymbol{\psi}(\boldsymbol{x}) - \sum_{\boldsymbol{x}\in D_{\mathrm{te}}\cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}}p_{\boldsymbol{\alpha}}(\delta_{\boldsymbol{x}}|\boldsymbol{x})\boldsymbol{\psi}(\boldsymbol{x}) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}. \tag{6.30}$$

Then the importance estimate is given by

$$\hat{w}(\boldsymbol{x}) = \frac{N_{\mathrm{tr}}}{N_{\mathrm{te}}}\exp(\langle\boldsymbol{\alpha},\boldsymbol{\psi}(\boldsymbol{x})\rangle). \tag{6.31}$$

We refer to this approach as *LogReg*.

Eq.(6.31) shows that the function model of the importance in LogReg is actually the same as that of LL-KLIEP except for a scaling factor (cf. Eq.(4.17)). However, the optimization criteria of LL-KLIEP and LogReg are different—in LL-KLIEP, the summation is taken only over the training or test input samples but not both, while the summation in LogReg is over both the training and test input samples. This difference is significant since LogReg does not allow us to use the computational trick we proposed in Section 4.2. Thus LL-KLIEP has the advantage in computation time and storage space consumption over LogReg.

7 Experiments

In this section, we experimentally compare the performance of LL-KLIEP with existing methods.

7.1 Toy experiments. Let $p_{\mathrm{tr}}(\boldsymbol{x}) = \mathcal{N}(\mathbf{0}_d, \boldsymbol{I}_d)$ and $p_{\mathrm{te}}(\boldsymbol{x}) = \mathcal{N}((1, 0, \ldots, 0)^\top, 0.75^2 \boldsymbol{I}_d)$. The task is to estimate the importance at the training input points:

$$w(\boldsymbol{x}) = p_{\mathrm{te}}(\boldsymbol{x})/p_{\mathrm{tr}}(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in D_{\mathrm{tr}}.$$

We compared LL-KLIEP, LL-KLIEP(LS1), LL-KLIEP(LS2), KLIEP, KDE, KMM, and LogReg. For LL-KLIEP, LL-KLIEP(LS1), and LL-KLIEP(LS2), we used 5-fold LCV to choose the regularization parameter σ and the kernel width s. For KLIEP, we use 5-fold LCV to choose the kernel width s. For KDE, we used 5-fold LCV to choose the kernel widths for the training and test densities. For KMM, we used $B = 1000$ and $\epsilon = (\sqrt{N_{\mathrm{tr}}} - 1)/\sqrt{N_{\mathrm{tr}}}$ following the suggestion in the original KMM paper [13]. We tested two different values of the kernel width ($s = 0.1$ and $s = 1.0$) for KMM since there is no reliable method to determine the kernel width. For LogReg, we used 5-fold CV to choose the regularization parameter σ and the kernel width s.

We fixed the number of test and training input samples at $N_{\mathrm{te}} = 1000$ and $N_{\mathrm{tr}} = 100$, and varied the input dimension $d = 2, 4, \ldots, 20$. We ran the simulation 100 times for each d, and evaluated the estimation accuracy of $\{w(\boldsymbol{x})\}_{\boldsymbol{x} \in D_{\mathrm{tr}}}$ by the mean NMSE (see Eq.(5.26)).

The mean NMSE over 100 trials is plotted in Figure 3. We omitted the graphs of LL-KLIEP(LS1) and LL-KLIEP(LS2) since they are almost identical to the solution of LL-KLIEP. Figure 3 shows that the error of KDE sharply increases as the input dimension grows, while LL-KLIEP, KLIEP, and LogReg tend to give much smaller errors than KDE. This would be the fruit of directly estimating the importance without going through density estimation. The results of LL-KLIEP and LogReg are slightly better than KLIEP,

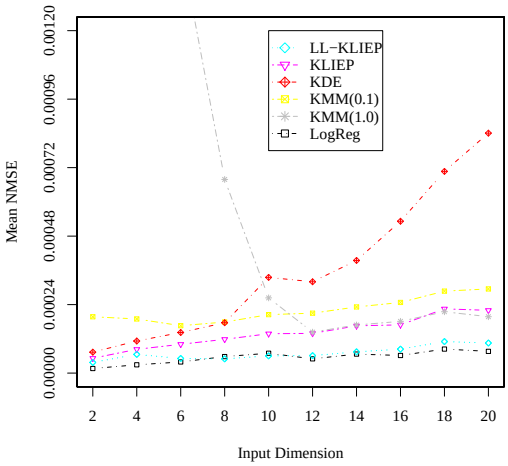
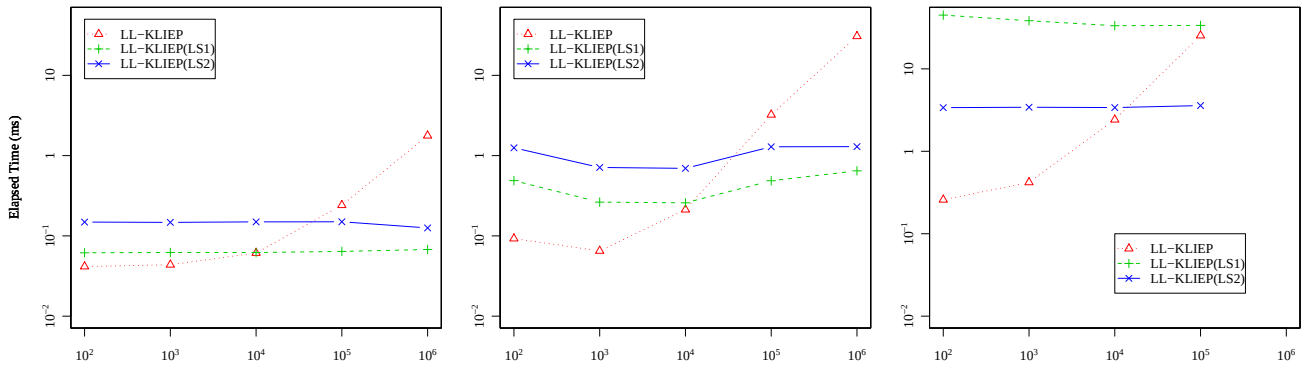


Figure 3: Mean NMSE over 100 trials. 'KMM(s)' denotes KMM with kernel width s.

perhaps because the original KLIEP does not contain a regularizer; we believe that the performance of KLIEP could be improved by adding a regularizer as used in LL-KLIEP and LogReg. KMM also works reasonably well, as long as that the kernel width s is chosen appropriately. However, the performance of KMM is highly dependent on s and determining its appropriate value may be difficult. Overall, the accuracy of LL-KLIEP is comparable to the best existing approaches.

Next, we compare the computational cost of LL-KLIEP, LL-KLIEP(LS1), and LL-KLIEP(LS2). We fixed the number of training input points at $N_{\mathrm{tr}} = 100$ and changed the input dimension $d = 10, 100, 1000$ and the number of test samples $N_{\mathrm{te}} = 10^2, 10^3, \ldots, 10^6$. In this experiment, we used linear basis function so that the number of bases is equivalent to the input dimension. We repeated the experiments 100 times for each N_{te} and d on the PC server with an Intel® Xeon® 3.6GHz ×2. All of them are implemented on R (http://www.r-project.org).

Figure 4 shows the average elapsed times for LL-KLIEP, LL-KLIEP(LS1), and LL-KLIEP(LS2). Note that these results include the pre-computation times for the test samples. When $d = 1000$, the result of $N_{\mathrm{te}} = 10^6$ was excluded for LL-KLIEP and LL-KLIEP(LS2) because of the large memory requirements. The results show that the computational cost of LL-KLIEP increases as the amount of test data N_{te} grows, but the computational cost of LL-KLIEP(LS) is nearly independent of the number of test data N_{te}. This is in good agreement with our theoretical analysis in Section 4.2. As we expected, LL-KLIEP is faster than LL-KLIEP(LS) when the number of test samples is small, LL-KLIEP(LS1) is faster than LL-KLIEP(LS2) for lower dimensional data, and LL-KLIEP(LS2) is advantageous for high dimensional problem.



(a) $d = 10, N_{\mathrm{tr}} = 100$ (b) $d = 100, N_{\mathrm{tr}} = 100$ (c) $d = 1000, N_{\mathrm{tr}} = 100$

Figure 4: Average computation time over 100 trials. The x (horizontal) axis represents the number of test samples (N_{te}), and the y (vertical) axis represents the elapsed time (millisecond), respectively.

7.2 Covariate shift adaptation with regression and classification benchmark data sets. Finally, we tested the importance estimation methods for covariate shift adaptation in regression and classification benchmark problems (see Table 7.2).

Each data set consists of input/output samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$. We normalized all of the input samples $\{\boldsymbol{x}_i\}_{i=1}^{N}$ into $[0,1]^d$ and chose the test samples $Z_{\mathrm{te}} = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{N_{\mathrm{te}}}$ from the pool $\{\boldsymbol{x}_i\}_{i=1}^{N}$ with probability $\min(1, 4(x_i^{(c)})^2)$, where $x_i^{(c)}$ was the c-th element of $\boldsymbol{x}_i$ and i was randomly determined and fixed in each trial. We chose the training samples $Z_{\mathrm{tr}} = \{(\boldsymbol{x}_j, y_j)\}_{j=1}^{N_{\mathrm{tr}}}$ uniformly from the rest. In this experiment, the test input density tends to be lower than the training input density when $x_i^{(c)}$ is small. We set the number of samples at $N_{\mathrm{tr}} = 100$ and $N_{\mathrm{te}} = 500$ for all of the data sets.

We used IWLS for regression problems and IWRL for classification problems. Their basis functions are linear: $\boldsymbol{\phi}(\boldsymbol{x}) = (1, \boldsymbol{x}^\top)^\top$.

We ran the experiments 100 times for each data set and evaluated the *mean test error*:

$$\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{te}}} (f_\theta(\boldsymbol{x}) - y)^2, \quad \text{(regression)} \tag{7.32}$$

$$\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{te}}} I(\mathrm{sign}(f_\theta(\boldsymbol{x})) = y), \quad \text{(classification)} \tag{7.33}$$

where $f_\theta(\boldsymbol{x})$ is defined in Eq.(2.1). We compared the importance estimation methods used in Section 7.1.

The results are summarized in Table 7.2, where "Uniform" denotes uniform weights, i.e., no importance weight is used. The table shows that LL-KLIEP compares favorably with Uniform, implying that the importance weighted methods are useful for improving the prediction performance under covariate shift. KDE tends to be worse than Uniform, which may be due to the high dimensionality. The direct importance estimation methods LL-KLIEP, KLIEP, and LogReg tend to outperform Uniform. KMM also works well given that the kernel width is chosen appropriately—but choosing the appropriate kernel width is difficult in practice without prior knowledge.

Over all, we found that LL-KLIEP is highly scalable to large sets of test data while the accuracy is comparable to the best existing methods.

8 Conclusions

In this paper, we addressed the problem of estimating the importance for covariate shift adaptation. We proposed a scalable direct importance estimation method called *LL-KLIEP*. The computation time of LL-KLIEP is nearly independent of the amount of test data, which is a significant advantage over existing approahes when we deal with large numbers of test samples. Our experiments highlighted this advantage and we experimentally confirmed that the accuracy of the proposed method is comparable to the best existing methods. Therefore the proposed method is a promising method for large-scale covariate shift adaptation.

9 Acknowledgments

We thank the members of the T-PRIMAL group for many helpful discussions related to this work.

References

[1] P. Baldi and S. Brunak. *Bioinformatics: The Machine Learning Approach*. MIT Press, Cambridge, 1998.

[2] S. Bickel. ECML 2006 discovery challenge workshop, 2006.

[3] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81 – 88. ACM Press, 2007.

Table 3: Mean test error averaged over 100 trials. The numbers in the brackets are the standard deviation. All the error values are normalized by that of Uniform (no importance weighting). The first 5 are regression data sets taken from DELVE [16] and the next is a classification data taken from IDA [17]. '# of better' is the number of data sets for which each method performed better than uniform. 'KMM(s)' denotes KMM with kernel width s.

Data	Dim	Uniform	LL-KLIEP	KLIEP	KDE	KMM(0.1)	KMM(1.0)	LogReg
kin-8fh	8	1.000(0.013)	**0.998**(0.013)	**0.998**(0.013)	1.102(0.023)	1.000(0.013)	2.436(0.202)	**0.998**(0.013)
kin-8fm	8	1.000(0.009)	**0.998**(0.009)	**0.999**(0.009)	1.0848(0.013)	**0.999**(0.009)	2.675(0.173)	**0.998**(0.009)
kin-8nh	8	1.000(0.115)	**0.999**(0.114)	1.000(0.115)	1.053(0.155)	1.001(0.116)	1.8416(1.400)	**0.999**(0.114)
kin-8nm	8	1.000(0.100)	**0.998**(0.097)	**0.998**(0.099)	1.047(0.135)	**0.998**(0.100)	2.116(2.515)	**0.997**(0.098)
abalone	7	1.000(1.747)	**0.998**(1.709)	**0.992**(1.645)	1.023(1.912)	1.902(18.337)	3.593(57.134)	**0.988**(1.591)
twonorm	20	1.000(0.021)	**0.994**(0.021)	**0.997**(0.022)	1.001(0.021)	1.001(0.021)	1.002(0.020)	**0.998**(0.021)
# of better		–	6	5	0	2	0	6

[4] S. Bickel and T. Scheffer. Dirichlet-enhanced spam filtering based on biased samples. In *Advances in Neural Information Processing Systems*. MIT Press, Cambridge, MA, 2007.

[5] K. M. Borgwardt, A. Gretton, M. J. Rasch, H.-P. Kriegel, B. Schölkopf, and A. J. Smola. Integrating structured biological data by kernel maximum mean discrepancy. *Bioinformatics*, 22(14):e49–e57, 2006.

[6] J. Q. Candela, N. Lawrence, A. Schwaighofer, and M. Sugiyama. NIPS 2006 workshop on learning when test and training inputs have different distributions, 2006.

[7] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[8] V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.

[9] G. S. Fishman. *Monte Carlo: Concepts, Algorithms, and Applications*. Springer-Verlag, Berlin, 1996.

[10] W. Härdle, M. Müller, S. Sperlich, and A. Werwatz. *Nonparametric and Semiparametric Models*. Springer Series in Statistics. Springer, Berlin, 2004.

[11] D. Hawking, E. Voorhees, N. Craswell, and P. Bailey. Overview of the trec-8 web track. In *Proceedings of the Eighth Text REtrieval Conference*, pages 131–150.

[12] J. J. Heckman. Sample selection bias as a specification error. *Econometrica*, 47(1):153–162, 1979.

[13] J. Huang, A. J. Smola, A. Gretton, K. M. Borgwardt, and B. Schölkopf. Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems*, pages 601–608, Cambridge, MA, 2007. MIT Press.

[14] J. Jiang and C. Zhai. Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 264–271, 2007.

[15] T. Kanamori and H. Shimodaira. Active learning algorithm using the maximum weighted log-likelihood estimator. *Journal of Statistical Planning and Inference*, 116(1):149–162, 2003.

[16] C. E. Rasmussen, R. M. Neal, G. E. Hinton, D. V. Camp, M. Revow, Z. Ghahramani, R. Kustra, and R. Tibshirani. *The delve manual*, 1996.

[17] G. Rätsch, T. Onoda, and K.-R. Müller. Soft margins for adaboost. *Machine Learning*, pages 287–320, 2001.

[18] C. R. Shelton. *Importance Sampling for Reinforcement Learning with Multiple Objectives*. PhD thesis, Massachusetts Institute of Technology, 2001.

[19] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227–244, 2000.

[20] I. Steinwart. On the influence of the kernel on the consistency of support vector machines. *Journal of Machine Learning Research*, 2:67–93, 2001.

[21] M. Sugiyama. Active learning in approximately linear regression based on conditional expectation of generalization error. *Journal of Machine Learning Research*, 7:141–166, Jan. 2006.

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

[23] M. Sugiyama and K.-R. Müller. Input-dependent estimation of generalization error under covariate shift. *Statistics & Decisions*, 23(4):249–279, 2005.

[24] M. Sugiyama, S. Nakajima, H. Kashima, P. von Bünau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems*, Cambridge, MA, 2008. MIT Press.

[25] R. S. Sutton and G. A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[26] G. Wahba. *Spline Model for Observational Data*. Society for Industrial and Applied Mathematics, Philadelphia and Pennsylvania, 1990.

[27] D. P. Wiens. Robust weights and designs for biased regression models: Least squares and generalized M-estimation. *Journal of Statistical Planning and Inference*, 83(2):395–412, 2000.

[28] J. R. Wolpaw, N. Birbaumer, D. J. McFarland, G. Pfurtscheller, and T. M. Vaughan. Brain-computer interfaces for communication and control. *Clinical Neurophysiology*, 113(6):767–791, 2002.

[29] B. Zadrozny. Learning and evaluating classifiers under sample selection bias. In *Proceedings of the 21st International Conference on Machine Learning*, New York, NY, 2004. ACM Press.