Journal of Information Processing Vol. 17 138–155 (Apr. 2009)

Regular Paper

Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation

Yuta Tsuboi,[†1] Hisashi Kashima,[†1] Shohei Hido,[†1]
Steffen Bickel[†2] and Masashi Sugiyama[†3]

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 loss function according to the *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. We also describe a promising result for large-scale covariate shift adaptation in a natural language processing task.

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* [21]. In many real-world applications such as robot control [15),20),27)], bioinformatics [1),6)], spam filtering [3)], natural language processing [17)], brain-computer interfacing [24),31)], or econometrics [14)], covariate shift is likely. Covariate shift is also naturally induced in selective sampling or active learning scenarios [8),10),18),23),30)]. 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 [7)] 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* [11),21),32)]:

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

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 from the training and test input samples, and then estimate the importance by taking the ratio of the estimated densities. However,

†1 Tokyo Research Laboratory, IBM Research
†2 Department of Computer Science, University of Potsdam, Germany
†3 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) [21)]. 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 [21),24),25),32)]. For this reason, estimating the importance is indispensable when covariate shift occurs.

density estimation is known to be a hard problem particularly in high dimensional cases [12]. 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) [16], the *logistic regression* based method (LogReg) [4], and the *Kullback-Leibler Importance Estimation Procedure* (KLIEP) [26].

KMM is based on a special property of *universal reproducing kernel Hilbert spaces* (Gaussian reproducing kernel Hilbert spaces are typical examples) [22], 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 always leads to the 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 [⋆1]. 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 cross-validation (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 [12], 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 [3] and information retrieval [13], the number of test (unlabeled) samples is enormous, especially on the Web. In these text processing applications, the distribution of training and test inputs can be changed between domains because of differences in vocabulary and writing style. The purpose of this paper is to develop a computationally efficient covariate shift adaptation method that can deal with a large number of unlabeled data points.

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 precomputation 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

⋆1 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 [24),32]. 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.

more efficient than the existing approaches. Therefore the range of application of covariate shift adaptation can be greatly enlarged towards large-scale problems. As for 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.

The rest of this paper is organized as follows. Section 2 formulates the supervised learning problem under covariate shift and review covariate shift adaptation techniques. Section 3 reviews an existing direct importance estimation method, KLIEP. Section 4 proposes a new direct importance estimation method that is scalable to large test data sets. Section 5 illustrates how the proposed method works in covariate shift adaptation using simple regression and classification data sets. Section 6 discusses the relation of the proposed method to existing approaches. Section 7 reports experimental results comparing the computation time and estimation accuracy of the proposed and existing methods. Finally, Section 8 gives conclusions.

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* [21], 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:

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \langle \boldsymbol{\theta}, \boldsymbol{\phi}(\boldsymbol{x}) \rangle, \tag{1}$$

where $\boldsymbol{\theta}$ is the parameter vector, $\boldsymbol{\phi}(\boldsymbol{x}) : \boldsymbol{X} \to \Re^h$ is a basis function of $\boldsymbol{x}$, and $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ denotes the Euclidean inner product between vector $\boldsymbol{u}$ and $\boldsymbol{v}$: $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{l=1}^{h} u_l v_l$. 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

$$\iint \mathrm{Loss}(\boldsymbol{x}, y, f_{\boldsymbol{\theta}}(\boldsymbol{x})) p_{\mathrm{te}}(\boldsymbol{x}) p(y|\boldsymbol{x}) d\boldsymbol{x} dy, \tag{2}$$

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:

$$w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}. \tag{3}$$

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

$$\int A(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} = \int A(\boldsymbol{x}) w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x}. \tag{4}$$

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.

2.2.1 IWLS

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

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

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 [21]:

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} w(\boldsymbol{x}) \left(f_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\right)^2 + \lambda\|\boldsymbol{\theta}\|^2 \right], \tag{5}$$

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 (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,

$\Phi_{i,l} = \phi_l(\boldsymbol{x}^{(i)})$, $\boldsymbol{y} = (y^{(1)}, y^{(2)}, \ldots, y^{(N_{\mathrm{tr}})})^\top$, and
$\boldsymbol{W} = \mathrm{diag}(\boldsymbol{w}^{(1)}, \boldsymbol{w}^{(2)}, \ldots, \boldsymbol{w}^{(N_{\mathrm{tr}})})$.

2.2.2 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

$$p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \frac{\exp(y f_{\boldsymbol{\theta}}(\boldsymbol{x}))}{1 + \exp(y f_{\boldsymbol{\theta}}(\boldsymbol{x}))}. \tag{6}$$

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

$$\hat{y} = \operatorname*{argmax}_{y} p_{\boldsymbol{\theta}}(y|\boldsymbol{x}). \tag{7}$$

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

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

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:

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLR}} \equiv \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{(\boldsymbol{x},y)\in Z_{\mathrm{tr}}} w(\boldsymbol{x}) \left(\log\left(1 + \exp\left(y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right) - y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) + \lambda\|\boldsymbol{\theta}\|^2 \right]. \tag{8}$$

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 descent method or some variants of the Newton method. The gradient of the above objective function is given by

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

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 [32]. To cope with this problem, a variant of CV called *importance weighted CV* (IWCV) has been proposed for model selection under covariate shift [24]. It has been proved that IWCV gives an unbiased estimate of the generalization error even under 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 [4),16),26)]. Here we review one of those direct methods called the *Kullback-Leibler Importance Estimation Procedure* (KLIEP) [26]. Other methods will be reviewed in Section 6.

3.1 KLIEP

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

$$\hat{w}(\boldsymbol{x})=\langle\boldsymbol{\alpha},\boldsymbol{\psi}(\boldsymbol{x})\rangle, \tag{9}$$

where $\boldsymbol{\alpha}\in\Re^b$ is a model parameter vector and $\boldsymbol{\psi}(\boldsymbol{x})\in\Re^b$ is a basis function. 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

$$\hat{p}_{\mathrm{te}}(\boldsymbol{x})=p_{\mathrm{tr}}(\boldsymbol{x})\hat{w}(\boldsymbol{x}). \tag{10}$$

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:

$$\begin{aligned}\mathrm{KL}[p_{\mathrm{te}}(\boldsymbol{x})||\hat{p}_{\mathrm{te}}(\boldsymbol{x})] &= \int_D p_{\mathrm{te}}(\boldsymbol{x})\log\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})\hat{w}(\boldsymbol{x})}d\boldsymbol{x}\\ &= \int_D p_{\mathrm{te}}(\boldsymbol{x})\log\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}d\boldsymbol{x}-\int_D p_{\mathrm{te}}(\boldsymbol{x})\log\hat{w}(\boldsymbol{x})d\boldsymbol{x}. \end{aligned}\tag{11}$$

Since the first term in Eq. (11) is independent of $\boldsymbol{\alpha}$, we ignore it and focus on the second term, which we denote by J_{KLIEP}:

$$J_{\mathrm{KLIEP}}=\int_D p_{\mathrm{te}}(\boldsymbol{x})\log\hat{w}(\boldsymbol{x})d\boldsymbol{x}\approx\frac{1}{N_{\mathrm{te}}}\sum_{\boldsymbol{x}\in D_{\mathrm{te}}}\log\hat{w}(\boldsymbol{x}), \tag{12}$$

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:

$$1=\int_D\hat{p}_{\mathrm{te}}(\boldsymbol{x})d\boldsymbol{x}=\int_D p_{\mathrm{tr}}(\boldsymbol{x})\hat{w}(\boldsymbol{x})d\boldsymbol{x}\approx\frac{1}{N_{\mathrm{tr}}}\sum_{\boldsymbol{x}\in D_{\mathrm{tr}}}\hat{w}(\boldsymbol{x}), \tag{13}$$

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

Then the resulting optimization problem is expressed as

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

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) [12] 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

$$\hat{J}_{\mathrm{KLIEP}}^r = \frac{1}{|D_{\mathrm{te}}^r|}\sum_{\boldsymbol{x}\in D_{\mathrm{te}}^r}\hat{w}^r(\boldsymbol{x}). \tag{14}$$

This procedure is repeated for $r = 1, 2, \ldots, R$ for each model 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} [26]:

$$K_s(\boldsymbol{x}, \boldsymbol{x}_l) = \exp\left\{-\frac{\|\boldsymbol{x}-\boldsymbol{x}_l\|^2}{2s^2}\right\}, \tag{15}$$

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 [26]. 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 (9) is used for modeling the importance function. Here, we propose using a (normalized) log-linear model for modeling the importance $w(\boldsymbol{x})$ as

$$\hat{w}(\boldsymbol{x}) = \frac{\exp(\langle\boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x})\rangle)}{\frac{1}{N_{\mathrm{tr}}}\sum_{\boldsymbol{x}'\in D_{\mathrm{tr}}}\exp(\langle\boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}')\rangle)}, \tag{16}$$

where the denominator guarantees the normalization constraint (13) [*1]. By definition, the log-linear model takes only non-negative values. Therefore, we no longer 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

$$\begin{aligned} J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha}) &= \frac{1}{N_{\mathrm{te}}}\sum_{\boldsymbol{x}\in D_{\mathrm{te}}}\log\hat{w}(\boldsymbol{x}) \\ &= \frac{1}{N_{\mathrm{te}}}\sum_{\boldsymbol{x}\in D_{\mathrm{te}}}\langle\boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x})\rangle - \log\frac{1}{N_{\mathrm{tr}}}\sum_{\boldsymbol{x}\in D_{\mathrm{tr}}}\exp(\langle\boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x})\rangle). \end{aligned} \tag{17}$$

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

$$\jmath(\boldsymbol{\alpha}) = J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha}) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}, \tag{18}$$

where σ^2 is a regularization parameter.

An advantage of LL-KLIEP over the original KLIEP is its computational effi-

[*1] 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.

ciency. The gradient of $\jmath(\boldsymbol{\alpha})$ can be computed as

$$\frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \boldsymbol{\psi}(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}') \rangle)} \boldsymbol{\psi}(\boldsymbol{x}) - \frac{\boldsymbol{\alpha}}{\sigma^2}$$
$$= F - \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \hat{w}(x) \boldsymbol{\psi}(\boldsymbol{x}) - \frac{\boldsymbol{\alpha}}{\sigma^2}, \tag{19}$$

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.

As the model selection of KLIEP, LCV can be used to find the optimal hyper-parameters of LL-KLIEP. Since $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ is evaluated using both training and test samples, both test and training samples can be divided into R disjoint subset $\{D_{\mathrm{te}}^r\}_{r=1}^R$ and $\{D_{\mathrm{tr}}^r\}_{r=1}^R$ in the LCV procedure. After the estimation of $\hat{w}^r(\boldsymbol{x})$, $J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})$ can be approximated as

$$\hat{J}_{\mathrm{LL-KLIEP}}^r(\boldsymbol{\alpha}) = \frac{1}{|D_{\mathrm{te}}^r|} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}^r} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \log \frac{1}{|D_{\mathrm{tr}}^r|} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}^r} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle).$$

Although the proposed optimization procedure may be more efficient than original KLIEP, there still exists a potential weakness: we still need to use all the 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 when 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 (19):

$$\jmath_{\mathrm{LS}}(\boldsymbol{\alpha}) = \frac{1}{2} \left\| \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \right\|^2. \tag{20}$$

The partial derivative of Eq. (20) with respect to $\boldsymbol{\alpha}$ is expressed as

$$\frac{\partial \jmath_{\mathrm{LS}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{\partial^2 \jmath(\boldsymbol{\alpha})}{\partial^2 \boldsymbol{\alpha}} \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}. \tag{21}$$

Note that the first component of Eq. (21) is the Hessian matrix of $\jmath(\boldsymbol{\alpha})$:

$$\frac{\partial^2 \jmath(\boldsymbol{\alpha})}{\partial^2 \boldsymbol{\alpha}} = \left(\sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{1}{N_{\mathrm{tr}}} w(\boldsymbol{x}) \left(\overline{\boldsymbol{\psi}}(\boldsymbol{x}) - \boldsymbol{\psi}(\boldsymbol{x}) \right) \boldsymbol{\psi}(\boldsymbol{x})^T - \frac{I}{\sigma^2} \right),$$

where

$$\overline{\boldsymbol{\psi}}(\boldsymbol{x}) = \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{1}{N_{\mathrm{tr}}} w(\boldsymbol{x}) \boldsymbol{\psi}(\boldsymbol{x}),$$

$\boldsymbol{\psi}(\boldsymbol{x})^T$ is the transpose of $\boldsymbol{\psi}(\boldsymbol{x})$, and I is the identity matrix. When we explicitly compute the Hessian matrix of $\jmath(\boldsymbol{\alpha})$, the computational complexity of the 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-a)* below.

The computation time and storage space of LL-KLIEP(LS1-a) are quadratic functions of the number of parameters b, which could be a bottleneck in high dimensional problems. To cope with this problem, we propose two approaches.

One approach for high dimensional problems is directly computing the product between the Hessian matrix and the gradient vector of $\jmath(\boldsymbol{\alpha})$ without storing the Hessian matrix:

$$\frac{\partial \jmath_{\mathrm{LS}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \left(\sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{1}{N_{\mathrm{tr}}} w(\boldsymbol{x}) \left(\overline{\boldsymbol{\psi}}(\boldsymbol{x}) - \boldsymbol{\psi}(\boldsymbol{x}) \right) \langle \boldsymbol{\psi}(\boldsymbol{x}), G \rangle - \frac{G}{\sigma^2} \right), \tag{22}$$

where $G = \frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}$. Since the inner product $\langle \boldsymbol{\psi}(\boldsymbol{x}), G \rangle$ requires $O(b)$ time, we can compute Eq. (22) in total with $O(bN_{\mathrm{tr}})$ computation time and $\Omega(cN_{\mathrm{tr}})$ space. We refer this approach as *LL-KLIEP(LS1-b)*, which is still independent of N_{te} and suitable for high dimensional problems compared with LL-KLIEP(LS1-a).

In the other approach for high dimensional problems, we make use of the *representer theorem* [29]. 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. (20) can be rewritten as

$$\jmath_{\mathrm{LS}}(\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in D_{\mathrm{tr}}}) = \frac{1}{2} \left\| F - \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x}) \omega(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \frac{\boldsymbol{\psi}(\boldsymbol{x}) \beta_{\boldsymbol{x}}}{\sigma^2} \right\|^2, \tag{23}$$

where

$$\omega(\boldsymbol{x}) = \frac{\exp(\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} K(\boldsymbol{x}, \boldsymbol{x}') \beta_{\boldsymbol{x}'})}{\sum_{\boldsymbol{x}'' \in D_{\mathrm{tr}}} \exp(\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} K(\boldsymbol{x}'', \boldsymbol{x}') \beta_{\boldsymbol{x}'})}, \tag{24}$$
$$K(\boldsymbol{x}, \boldsymbol{x}') = \langle \boldsymbol{\psi}(\boldsymbol{x}), \boldsymbol{\psi}(\boldsymbol{x}') \rangle .$$

The partial derivative of Eq. (23) with respect to $\beta_{\boldsymbol{x}}$ is:

$$\frac{\partial \jmath_{\mathrm{LS}}(\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in D_{\mathrm{tr}}})}{\partial \beta_{\boldsymbol{x}}} = \left\langle F - \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \boldsymbol{\psi}(\boldsymbol{x}') \left(\omega(\boldsymbol{x}') - \frac{\beta_{\boldsymbol{x}'}}{\sigma^2} \right), \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \omega(\boldsymbol{x}') \boldsymbol{\psi}(\boldsymbol{x}') \langle \boldsymbol{\varphi}(\boldsymbol{x}'), \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \frac{\boldsymbol{\psi}(\boldsymbol{x})}{\sigma^2} \right\rangle, \tag{25}$$

where $\boldsymbol{\varphi}(\boldsymbol{x}) = \sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \omega(\boldsymbol{x}') \boldsymbol{\psi}(\boldsymbol{x}') - \boldsymbol{\psi}(\boldsymbol{x})$. By the change of variables, it is not required to calculate the partial derivative with respect to $\boldsymbol{\alpha}$ so that we can avoid the computation of the Hessian matrix of $\jmath(\boldsymbol{\alpha})$. 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 also an improvement over the direct computation of the partial derivative in Eq. (21).

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

Table 1 Computational complexity and space requirements. N_{tr} is the number of training samples, N_{te} is the number of test samples, b is the number of parameters, and c is the average number of non-zero basis entries. "Precomp." denotes the computational complexity of once-off precomputation.

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

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

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**. In terms of the complexity analysis, LL-KLIEP(LS1-b) is the best solution for the large amount of test inputs. We verified the analysis by the computational experiments in Section 7.1.

5. Illustrative Examples

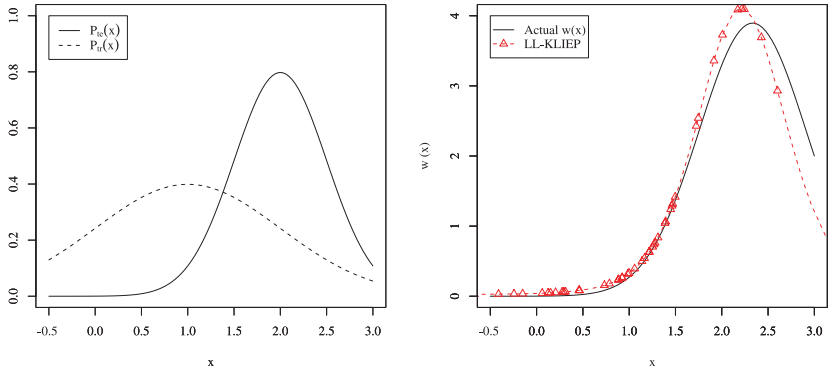
In this section, we illustrate the behavior of the proposed LL-KLIEP and show how it can be applied in covariate shift adaptation.

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 $p_{\mathrm{te}}(x) = \mathcal{N}(x; 2, 0.5^2)$, 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 $y^{(i)} = f(x^{(i)}) + \epsilon^{(i)}$,



(a) Training and test densities (b) Estimated importance

Fig. 1 Importance estimation.

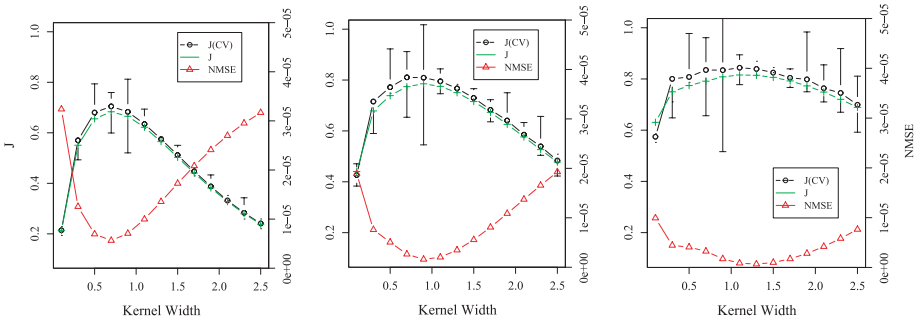
where the noise $\{\epsilon^{(i)}\}_{i=1}^{N_{\mathrm{tr}}}$ has density $\mathcal{N}(\epsilon; 0, 0.25^2)$. Let the number of training samples be $N_{\mathrm{tr}} = 200$ and the number of test samples be $N_{\mathrm{te}} = 1{,}000$. These settings imply that we are considering an extrapolation problem (see **Fig. 1** (a)).

We used 100 Gaussian basis functions centered at randomly chosen test input samples. Figure 1 (b) shows the actual 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-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2), but we omit their graphs since their solutions are almost identical to the solution of LL-KLIEP.

Figure 2 depicts the values of the true $J_{\mathrm{LL-KLIEP}}$ (see Eq. (17)) 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 the different $\sigma = 0.5, 1, 2$. We also plot the normalized mean squared error of the estimated importance:

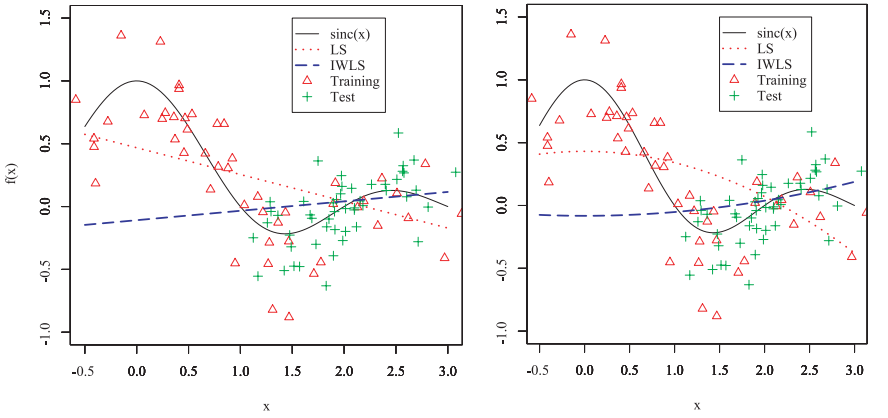
$$\mathrm{NMSE} = \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \left(\frac{\hat{w}(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} \hat{w}(\boldsymbol{x}')} - \frac{w(\boldsymbol{x})}{\sum_{\boldsymbol{x}' \in D_{\mathrm{tr}}} w(\boldsymbol{x}')} \right)^2. \tag{26}$$

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



(a) 5CV ($\sigma = 0.5$) (b) 5CV ($\sigma = 1$) (c) 5CV ($\sigma = 2$)

Fig. 2 Model selection curve.



(a) Polynomials of order 1 (b) Polynomials of order 2

Fig. 3 Regression under covariate shift (True and learned functions).

NMSE. Figure 2 also shows that σ value affects the importance estimation in terms of NMSE.

Figure 3 shows the true learning target function and functions learned by ordinary LS and IWLS with a linear basis function (Fig. 3 (a)), i.e., $\boldsymbol{\phi}(x) = (1, x)^\top$, and a quadratic basis function (Fig. 3 (b)), i.e., $\boldsymbol{\phi}(x) = (1, x, x^2)^\top$ (Section 2.2). The regularization parameter λ was selected by CV for LS and IWCV for IWLS

Table 2 Specifications of illustrative classification data.

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

(Sections 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

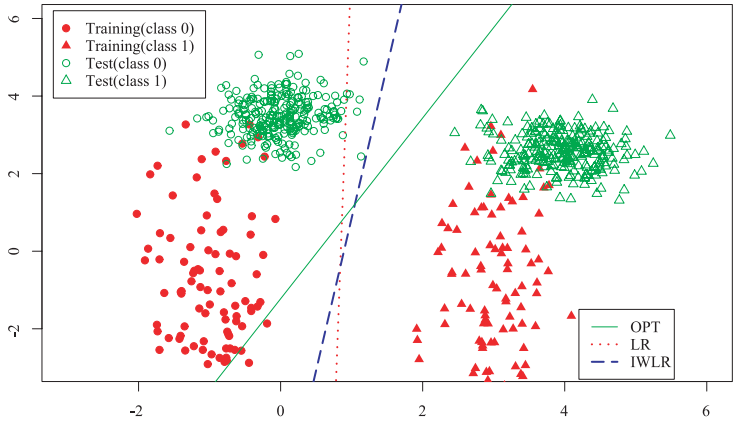
Next, let us consider two illustrative binary classification problems, where two-dimensional samples were generated from Gaussian distributions (see **Table 2** and **Fig. 4**). These data sets correspond to a 'linear shift' and a 'non-linear shift' (rotation).

Let the number of the training samples be $N_{\rm tr} = 200$ and that of the test samples be $N_{\rm te} = 1{,}000$ (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 LR/IWLR: $\boldsymbol{\phi}(\boldsymbol{x}) = (1, \boldsymbol{x}^\top)^\top$.

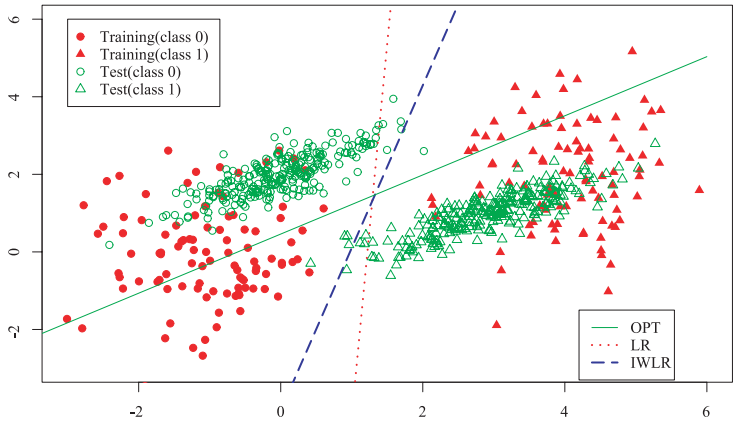
Figure 4 shows the decision boundaries obtained by LR+CV and IWLR+IWCV. For references, we also show 'OPT', which is the optimal decision boundary obtained using the test input-output samples. For the data set depicted in Fig. 4 (a), the correct classification rate of LR+CV is 99.1% while that of IWLR+IWCV is 100%. For the data set depicted in Fig. 4 (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. Related Work and Discussion

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



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



(b) $p_{\rm te}(\boldsymbol{x})$ is rotated from $p_{\rm tr}(\boldsymbol{x})$.

Fig. 4 Classification examples under covariate shift.

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

$$\hat{p}(\boldsymbol{x}) = \frac{1}{(2\pi s^2)^{d/2} N} \sum_{l=1}^{N} K_s(\boldsymbol{x}, \boldsymbol{x}_l), \tag{27}$$

where $K_s(\boldsymbol{x}, \boldsymbol{x}')$ is the Gaussian kernel (15). The performance of KDE depends on the choice of the kernel width s, which can be optimized by LCV [12]. 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* [12], 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 training input points [16].

The basic idea of KMM is to find $w(\boldsymbol{x})$ such that the mean discrepancy between nonlinearly transformed samples drawn from $p_{\mathrm{te}}(\boldsymbol{x})$ and $p_{\mathrm{tr}}(\boldsymbol{x})$ is minimized in a *universal reproducing kernel Hilbert space* [22]. The Gaussian kernel (15) is an example of kernels that induce universal reproducing kernel Hilbert spaces and it has been shown that the solution of the following optimization problem agrees with the true importance:

$$\min_{w(\boldsymbol{x})} \left\| \int K_s(\boldsymbol{x}, \cdot) p_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} - \int K_s(\boldsymbol{x}, \cdot) w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x} \right\|_{\mathcal{F}}^2$$
$$\text{subject to } \int w(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x} = 1 \text{ and } w(\boldsymbol{x}) \geq 0,$$

where $\|\cdot\|_{\mathcal{F}}$ denotes the norm in the Gaussian reproducing kernel Hilbert space and $K_s(\boldsymbol{x}, \boldsymbol{x}')$ is the Gaussian kernel (15).

An empirical version of the above problem is reduced to the following quadratic program:

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

where

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

B (≥ 0), ϵ (≥ 0), and s (≥ 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 s 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. (23)) has a tight connection to KMM. Up to irrelevant constants, Eq. (23) without a regularizer can be expressed as

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

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 (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 inputs and $\delta = 1$ to the test inputs. 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[4]:

$$w(\boldsymbol{x}) = \frac{p(\delta = -1)}{p(\delta = 1)} \frac{p(\delta = 1|\boldsymbol{x})}{p(\delta = -1|\boldsymbol{x})}. \tag{28}$$

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. (6)). Let us train the LR model by regularized maximum likelihood estimation. The objective function to be maximized is given by

$$\mathrm{LR}(\boldsymbol{\alpha}) = \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \log(1 + \exp(\delta_{\boldsymbol{x}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}, \tag{29}$$

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 non-linear optimization methods. The gradient of the objective function is given as

$$\frac{\partial \mathrm{LR}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}} \boldsymbol{\psi}(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in D_{\mathrm{te}} \cup D_{\mathrm{tr}}} \delta_{\boldsymbol{x}} p_{\boldsymbol{\alpha}}(\delta_{\boldsymbol{x}}|\boldsymbol{x}) \boldsymbol{\psi}(\boldsymbol{x}) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}. \tag{30}$$

Then the importance estimate is given by

$$\hat{w}(\boldsymbol{x}) = \frac{N_{\mathrm{tr}}}{N_{\mathrm{te}}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle). \tag{31}$$

We refer to this approach as *LogReg*.

Equation (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. (16)). 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.

Bickel, et al. [4] proposed simultaneous optimization of both importance estimator and classifier. Although their method can perform better than our two stage method which solves importance estimation and classifier's parameter estimation separately, it has a weakness in model selection. Since the hyper-parameter of their method is supposed to be tuned for test samples, CV is not applicable if no labeled test sample is available. On the other hand, our method can select hyper-parameters for both importance estimation by LCV and classification by IWCV.

The characteristics of the proposed and related methods are summarized in **Table 3**.

Table 3 Relation between the proposed and related methods.

	Model selection	Direct importance model	Optimization
KDE	Available	Not Available	Analytic
KMM	Not Available	Non-parametric	Constraint quadratic program
LogReg	Available	Log-linear	Unconstraint non-linear
KLIEP	Available	Linear	Constraint non-linear
LL-KLIEP	Available	Log-linear	Unconstraint non-linear

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:

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

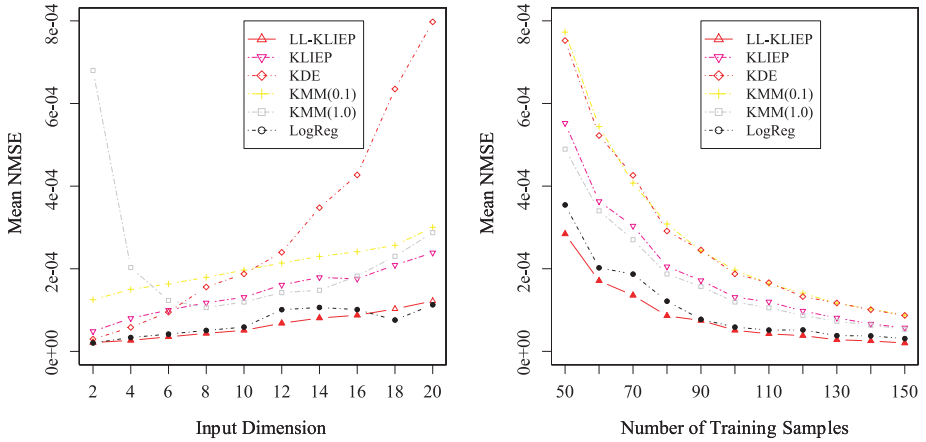
We compared KLIEP, KDE, KMM, LogReg, LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2). For LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2), we used 5-fold LCV to choose the regularization parameter σ and the kernel width s. For KLIEP, we use 5-fold LCV to choose the kernel width s. For KDE, we used 5-fold LCV to choose the kernel widths for the training and test densities. For KMM, we used $B = 1{,}000$ and $\epsilon = (\sqrt{N_{\mathrm{tr}}} - 1)/\sqrt{N_{\mathrm{tr}}}$ following the suggestion in the original KMM paper [16]. 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 input samples at $N_{\mathrm{te}} = 1{,}000$ and considered the following setting for the number of training input samples N_{tr} and the input dimension d:

(1) $N_{\mathrm{tr}} = 100$ and $d = 2, 4, \ldots, 20$.
(2) $d = 10$ and $N_{\mathrm{tr}} = 50, \ldots, 150$.

We ran the simulation 100 times for each d and N_{tr}, and evaluated the estimation accuracy of $\{w(\boldsymbol{x})\}_{\boldsymbol{x} \in D_{\mathrm{tr}}}$ by the mean NMSE (see Eq. (26)).

The mean NMSE over 100 trials is plotted in **Fig. 5**. The filled plot markers indicate the best method and comparable ones based on the *Wilcoxon* signed rank test at the significance level 1% in terms of the NMSE. We omitted the graphs of LL-KLIEP(LS1-a), LL-KLIEP(LS1-b) and LL-KLIEP(LS2) since they are almost identical to the result of LL-KLIEP. Figure 5 (a) 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. Figure 5 (b) shows that the errors of all methods tend to decrease as the number of training samples grows. Again, LL-KLIEP and LogReg are shown to work well. These 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, perhaps

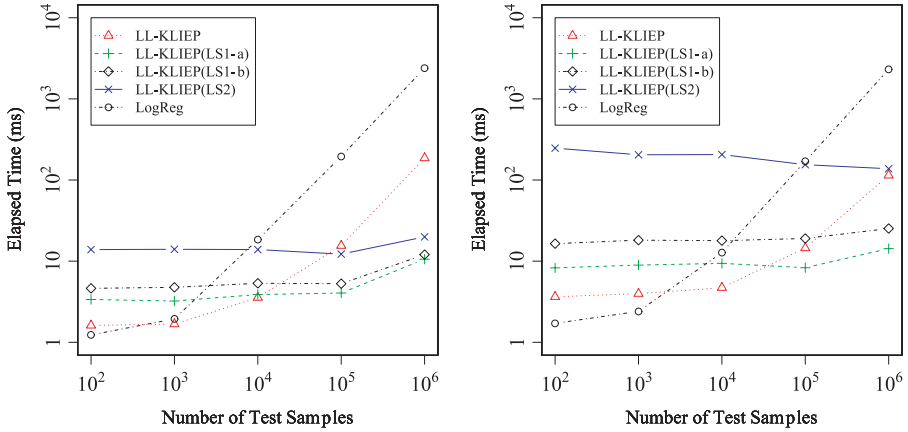


(a) Varying input dimension (b) Varying training sample size

Fig. 5 Mean NMSE over 100 trials. The filled plot markers indicate the best method and comparable ones based on the *Wilcoxon* signed rank test at the significance level 1% in terms of the NMSE. 'KMM(s)' denotes KMM with kernel width s.

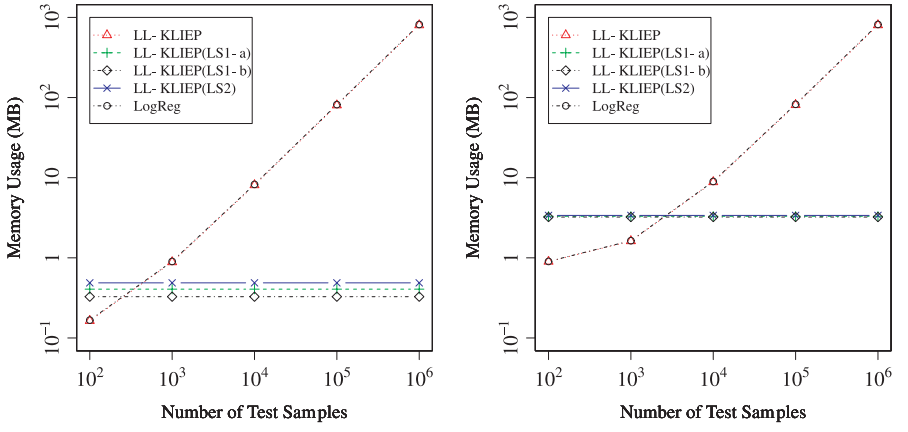
because the original KLIEP does not contain a regularizer; we believe that the performance of KLIEP could be improved by adding a regularizer as used in LL-KLIEP and LogReg. KMM also works reasonably well, as long as the kernel width s is chosen appropriately. However, the performance of KMM is highly 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 compared the computational cost of LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), LL-KLIEP(LS2), and LogReg, which have good accuracy in the previous experiments. We investigated the entire computation time of all of them including cross-validation and the precomputation times for the test samples. Note that the Gaussian width s and the regularization parameter σ are chosen over the 5×5 equidistant grid in this experiment for all the methods. We fixed the input dimension at $d = 10$ and changed the number of training input points $N_{\mathrm{tr}} = 10^2, 10^3$ and the number of test samples $N_{\mathrm{te}} = 10^2, 10^3, \ldots, 10^6$. We repeated the experiments 100 times for each N_{tr} and N_{te} on the PC server with an Intel® Xeon® 2.66 GHz. All of them are implemented on R (http://www.r-



(a) $d = 10, N_{\mathrm{tr}} = 100$ (b) $d = 10, N_{\mathrm{tr}} = 1{,}000$

Fig. 6 Average computation time over 100 trials. The horizontal axis represents the number of test samples (N_{te}), and the vertical axis represents the elapsed time (millisecond), respectively.



(a) $d = 10, N_{\mathrm{tr}} = 100$ (b) $d = 10, N_{\mathrm{tr}} = 1{,}000$

Fig. 7 Memory usage. The horizontal axis represents the number of test samples (N_{te}), and the vertical axis represents the memory usage (MB), respectively.

project.org) and *conjugate gradient* method was used to optimize their objective functions.

Figure 6 shows the average elapsed times for LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), LL-KLIEP(LS2), and LogReg. The results show that the computational cost of LL-KLIEP and LogReg 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 samples N_{te}. This is in good agreement with our theoretical analysis in Section 4.2. Thus the cost of dealing with a large amount of test data in each optimization step is much higher than that at one time precomputation.

We also compared the memory usage of LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), LL-KLIEP(LS2), and LogReg. We used the same implementation and computational environment as the previous experiments. **Figure 7** shows the memory usage of each method. The results show that the space requirement of LL-KLIEP and LogReg increases as the amount of test data N_{te} grows, but that of LL-KLIEP(LS) is independent of the number of test data N_{te}.

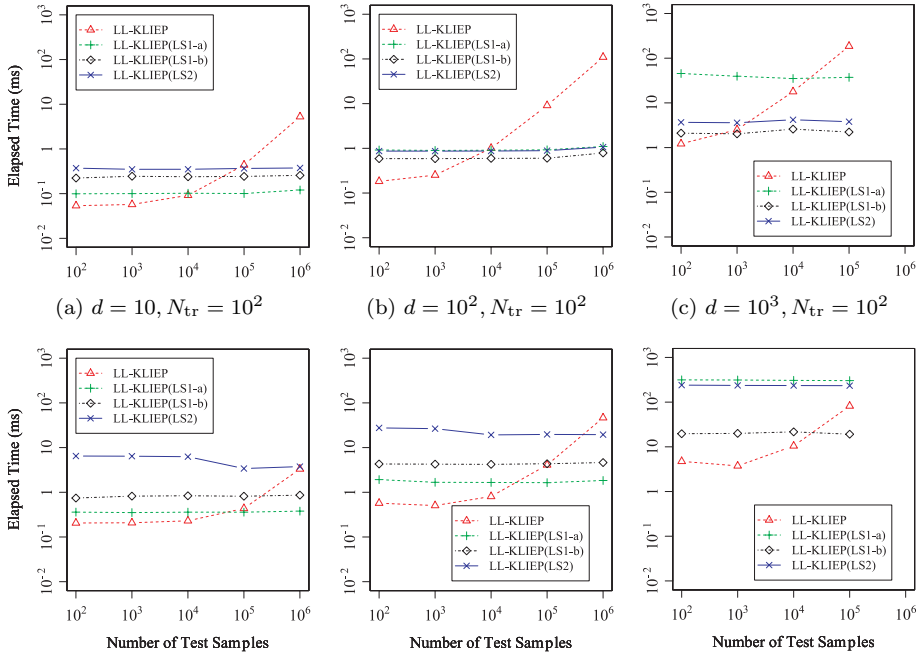
In addition, we compared the computational cost of LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2) in detail. We examined the combination of the following setting for the number of test samples N_{te}, the number of training inputs N_{tr}, and the input dimension d:

- $N_{\mathrm{te}} = 10^2, 10^3, \ldots, 10^6$
- $N_{\mathrm{tr}} = 10^2, 10^3$
- $d = 10^2, 10^3, 10^4$.

In this experiment, we used a linear basis function so that the number of bases is equivalent to the input dimension. Since the computational time of cross-validation is conceptually a scalar multiple of that of each optimization step, we compared the computational time including the precomputation times for the test inputs after the model parameters are fixed. We repeated the experiments 100 times for each N_{te}, N_{tr}, and d using the same implementation and computational environment as the previous experiments.

Figure 8 shows the average elapsed times for LL-KLIEP, LL-KLIEP(LS1-a), LL-KLIEP(LS1-b), and LL-KLIEP(LS2). When $d = 10^3$, the result of $N_{\mathrm{te}} = 10^6$ was excluded because of the large memory requirements. As we expected,

(1) LL-KLIEP is faster than LL-KLIEP(LS) when the number of test samples



(a) $d = 10, N_{\mathrm{tr}} = 10^2$ (b) $d = 10^2, N_{\mathrm{tr}} = 10^2$ (c) $d = 10^3, N_{\mathrm{tr}} = 10^2$

(d) $d = 10, N_{\mathrm{tr}} = 10^3$ (e) $d = 10^2, N_{\mathrm{tr}} = 10^3$ (f) $d = 10^3, N_{\mathrm{tr}} = 10^3$

Fig. 8 Average computation time over 100 trials. The horizontal axis represents the number of test samples (N_{te}), and the vertical axis represents the elapsed time (millisecond), respectively.

is small,

(2) LL-KLIEP(LS1-a) is faster than LL-KLIEP(LS2) for lower dimensional data,

(3) both LL-KLIEP(LS1-b) and LL-KLIEP(LS2) are advantageous for high dimensional problems, and

(4) the computational cost of LL-KLIEP(LS1-b) increases for the larger amount of training inputs.

Since LL-KLIEP(LS1-b) outperformed LL-KLIEP(LS2) in all the settings, we conclude LL-KLIEP(LS1-b) is more suitable for high dimensional problems. However, LL-KLIEP(LS1-a) is faster than LL-KLIEP(LS1-b) for lower dimensional data against the complexity analysis. One reason for this result might be that the computation of LL-KLIEP(LS1-b) is relatively complex in the iteration of the training inputs compared with LL-KLIEP(LS1-a). Therefore, LL-KLIEP(LS1-a) runs faster than LL-KLIEP(LS1-b) if the number of dimensions is small enough not to ignore this overhead.

7.2 Natural Language Processing Task

In this section, we show the experimental result that LL-KLIEP is applied to the domain adaptation task of a natural language processing (NLP).

Since statistical NLP systems tend to perform worse in different domains because of differences in vocabulary and writing style, the domain adaptation of NLP system is one of the important issues in NLP [5),9),17)].

In this experiment, we conducted domain adaptation experiments for the Japanese word segmentation task. It is not trivial to detect word boundaries for non-segmented languages such as Japanese or Chinese. In the word segmentation task, $\boldsymbol{x}=(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T) \in \boldsymbol{X}$ represents a given sequence of character boundaries of a sentence and $\boldsymbol{y}=(y_1, y_2, \cdots, y_T) \in \boldsymbol{Y}$ is a sequence of the corresponding labels, which specify whether the current position is a word boundary. It is reasonable to consider the domain adaptation task of word segmentation systems as a covariate shift adaptation problem since word segmentation policy ($p(\boldsymbol{y}|\boldsymbol{x})$) is rarely changed between domains in the same language, but the distribution of characters ($p(\boldsymbol{x})$) tends to be changed between domains.

In the experiments, we used the same data and feature set as Tsuboi et al. [28)] which aims to adapt a word segmentation system from a daily conversation domain to a medical domain. One of the characteristics of NLP tasks is high dimensionality. The total number of distinct features was about $d = 300\,K$ in this data. In this experiment, we used labeled (i.e., word segmented) $N_{\mathrm{tr}} = 13{,}000$ sentences for the source domain and unlabeled (i.e., unsegmented) $N_{\mathrm{te}} = 53{,}834$ sentences for the target domain. In addition, we have 1,000 labeled target domain sentences for the evaluation of the domain adaptation task.

As a word segmentation model, we employed *Conditional Random Fields* (CRFs), which are the generalization of the LR (6) for structured prediction [19)]. CRFs model the conditional probability $p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$ of output structure $\boldsymbol{y}$ given an input $\boldsymbol{x}$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle)}{\sum_{\boldsymbol{y} \in \boldsymbol{Y}} \exp(\langle \boldsymbol{\theta}, \boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) \rangle)}, \tag{32}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}, \boldsymbol{y}) : \boldsymbol{X} \times \boldsymbol{Y} \to \Re^h$ denote a basis function mapping from a pair of $\boldsymbol{x}$ and $\boldsymbol{y}$ to an h dimensional vector. Although conventional CRF learning algorithms minimize the regularized negative log-likelihood, we implemented an importance weighted CRF training algorithm (IWCRF) in the same way as IWLR:

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

To estimate the importance of each sentence, we used the source domain data as training data D_{tr} and the unlabeled target domain data as test data D_{te}. We defined the basis function of the importance model $\hat{w}(\boldsymbol{x})$ as the average value of features for CRFs in a sentence:

$$\boldsymbol{\psi}(\boldsymbol{x}) = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{x}_t. \tag{33}$$

The performance measure in the experiments is the standard F measure score, $\mathrm{F} = 2\mathrm{RP}/(\mathrm{R} + \mathrm{P})$ where

$$\mathrm{R} = \frac{\#\text{ of correct words}}{\#\text{ of words in test data}} \times 100$$
$$\mathrm{P} = \frac{\#\text{ of correct words}}{\#\text{ of words in system output}} \times 100.$$

In addition, we tuned the hyper-parameter of IWCRF based on an *importance weighted F measure score*, IWF, in which the number of correct words is weighted by the importance of the sentence that these words belong to:

$$\mathrm{IWF}(D) = \frac{2 \times \mathrm{IWR}(D) \times \mathrm{IWP}(D)}{\mathrm{IWR}(D) + \mathrm{IWP}(D)}$$

for the validation set D where

$$\mathrm{IWR}(D) = \frac{\sum_{(\boldsymbol{x}, \boldsymbol{y}) \in D} w(\boldsymbol{x}) \sum_{v_t \in \boldsymbol{y}} [\hat{v}_t = v_t]}{\sum_{(\boldsymbol{x}, \boldsymbol{y}) \in D} \sum_{v_t \in \boldsymbol{y}} w(\boldsymbol{x})} \times 100$$

Table 4 Word segmentation performance in the target domain. "CRF + 1,000" stands for the performance of a CRF additionally using 1,000 manual word segmentations of the target domain.

	F	R	P
CRF	92.30	90.58	94.08
IWCRF (LL-KLIEP)	**94.46**	**94.32**	94.59
IWCRF (LogReg)	93.68	94.30	93.07
CRF + 1,000	94.43	93.49	**95.39**

$$\mathrm{IWP}(D) = \frac{\sum_{(\boldsymbol{x}, \boldsymbol{y}) \in D} w(\boldsymbol{x}) \sum_{v_t \in \boldsymbol{y}} [\hat{v}_t = v_t]}{\sum_{(\boldsymbol{x}, \boldsymbol{y}) \in D} \sum_{\hat{v}_t \in \hat{\boldsymbol{y}}} w(\boldsymbol{x})} \times 100, \text{ and}$$

v_t denotes a t-th word in a sentence $(\boldsymbol{x}, \boldsymbol{y})$ and $\hat{v}_t$ denotes a t-th word of a system prediction $\hat{\boldsymbol{y}} = \operatorname{argmax}_{\boldsymbol{y}} p_{\boldsymbol{\theta}}(\boldsymbol{y}|\boldsymbol{x})$. We used 1/10 of training data D_{tr} as the validation set.

Then, we compared the target domain performance between CRF, IWCRF, and CRF which was trained additionally using 1,000 manual word segmentations, described in Tsuboi, et al. [28], denoted as "CRF + 1,000". For importance estimation, we compared LL-KLIEP and LogReg for which we employed 5-fold CV to find the optimal hyper-parameter σ.

Table 4 shows the result of the performance of each method. Surprisingly, the F score of IWCRF (LL-KLIEP) outperformed not only that of CRF, but also that of "CRF + 1,000", so the benefit of the importance weighting is worth the manual labeling of 1,000 words. Most notably, the empirical result shows that the covariate shift adaptation technique improves the coverage (R) in the target domain. Comparing with LogReg, LL-KLIEP results in higher final system performance. Since it is easy to obtain large amounts of unlabeled text data in NLP tasks, we believe the domain adaptation of NLP tasks is one of the promising applications of the proposed method.

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

approaches when we deal with a large number 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. Finally, a natural language processing experiment shows a promising result of the proposed method for large-scale covariate shift adaptation.

References

1) Baldi, P. and Brunak, S.: *Bioinformatics: The Machine Learning Approach*, MIT Press, Cambridge (1998).
2) Bickel, S.: ECML 2006 Discovery Challenge Workshop (2006).
3) Bickel, S. and Scheffer, T.: Dirichlet-Enhanced Spam Filtering based on Biased Samples, *Advances in Neural Information Processing Systems*, MIT Press, Cambridge, MA, pp.161–168 (2007).
4) Bickel, S., Brückner, M. and Scheffer, T.: Discriminative Learning for Differing Training and Test Distributions, *Proc. 24th international conference on Machine learning*, pp.81–88, ACM Press (2007).
5) Blitzer, J., McDonald, R. and Pereira, F.: Domain adaptation with structural correspondence learning, *Proc. Conference on Empirical Methods in Natural Language Processing*, pp.120–128 (2006).
6) Borgwardt, K.M., Gretton, A., Rasch, M.J., Kriegel, H.-P., Schölkopf, B. and Smola, A.J.: Integrating Structured Biological Data by Kernel Maximum Mean Discrepancy, *Bioinformatics*, Vol.22, No.14, pp.e49–e57 (2006).
7) Candela, J.Q., Lawrence, N., Schwaighofer, A. and Sugiyama, M.: *NIPS 2006 Workshop on Learning When Test and Training Inputs Have Different Distributions* (2006).
8) Cohn, D.A., Ghahramani, Z. and Jordan, M.I.: Active Learning with Statistical Models, *Journal of Artificial Intelligence Research*, Vol.4, pp.129–145 (1996).
9) Daumé III, H.: Frustratingly easy domain adaptation, *Proc. 45th Annual Meeting of the Association for Computational Linguistics*, pp.256–263 (2007).
10) Fedorov, V.V.: *Theory of Optimal Experiments*, Academic Press, New York (1972).
11) Fishman, G.S.: *Monte Carlo: Concepts, Algorithms, and Applications*, Springer-Verlag, Berlin (1996).
12) Härdle, W., Müller, M., Sperlich, S. and Werwatz, A.: *Nonparametric and Semiparametric Models*, Springer, Berlin (2004).
13) Hawking, D., Voorhees, E., Craswell, N. and Bailey, P.: Overview of the TREC-8 Web Track, *Proc. 8th Text REtrieval Conference*, pp.131–150 (1999).
14) Heckman, J.J.: Sample Selection Bias as a Specification Error, *Econometrica*, Vol.47, No.1, pp.153–162 (1979).
15) Hirotaka, H., Akiyama, T., Sugiyama, M. and Peters, J.: Adaptive Importance Sampling with Automatic Model Selection in Value Function Approximation, *Proc. 23rd AAAI Conference on Artificial Intelligence*, Chicago, USA, pp.1351–1356 (2008).
16) Huang, J., Smola, A.J., Gretton, A., Borgwardt, K.M. and Schölkopf, B.: Correcting Sample Selection Bias by Unlabeled Data, *Advances in Neural Information Processing Systems*, Cambridge, MA, pp.601–608, MIT Press (2007).
17) Jiang, J. and Zhai, C.: Instance Weighting for Domain Adaptation in NLP, *Proc. 45th Annual Meeting of the Association for Computational Linguistics*, pp.264–271 (2007).
18) Kanamori, T. and Shimodaira, H.: Active Learning Algorithm Using the Maximum Weighted Log-Likelihood Estimator, *Journal of Statistical Planning and Inference*, Vol.116, No.1, pp.149–162 (2003).
19) Lafferty, J., McCallum, A. and Pereira, F.: Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data, *Proc. 18th International Conference on Machine Learning*, pp.282–289 (2001).
20) Shelton, C.R.: Importance Sampling for Reinforcement Learning with Multiple Objectives, Ph.D. Thesis, Massachusetts Institute of Technology (2001).
21) Shimodaira, H.: Improving Predictive Inference under Covariate Shift by Weighting the Log-Likelihood Function, *Journal of Statistical Planning and Inference*, Vol.90, No.2, pp.227–244 (2000).
22) Steinwart, I.: On the Influence of the Kernel on the Consistency of Support Vector Machines, *Journal of Machine Learning Research*, Vol.2, pp.67–93 (2001).
23) Sugiyama, M.: Active Learning in Approximately Linear Regression Based on Conditional Expectation of Generalization Error, *Journal of Machine Learning Research*, Vol.7, pp.141–166 (2006).
24) Sugiyama, M., Krauledat, M. and Müller, K.-R.: Covariate Shift Adaptation by Importance Weighted Cross Validation, *Journal of Machine Learning Research*, Vol.8, pp.985–1005 (2007).
25) Sugiyama, M. and Müller, K.-R.: Input-Dependent Estimation of Generalization Error under Covariate Shift, *Statistics & Decisions*, Vol.23, No.4, pp.249–279 (2005).
26) Sugiyama, M., Nakajima, S., Kashima, H., von Bünau, P. and Kawanabe, M.: Direct Importance Estimation with Model Selection and Its Application to Covariate Shift Adaptation, *Advances in Neural Information Processing Systems*, Cambridge, MA, MIT Press (2008).
27) Sutton, R.S. and Barto, G.A.: *Reinforcement Learning: An Introduction*, MIT Press, Cambridge, MA (1998).
28) Tsuboi, Y., Kashima, H., Mori, S., Oda, H. and Matsumoto, Y.: Training Conditional Random Fields Using Incomplete Annotations, *Proc. 22nd International Conference on Computational Linguistics*, pp.897–904 (2008).
29) Wahba, G.: *Spline Model for Observational Data*, Society for Industrial and Ap-

plied Mathematics, Philadelphia and Pennsylvania (1990).
30) Wiens, D.P.: Robust Weights and Designs for Biased Regression Models: Least Squares and Generalized M-Estimation, *Journal of Statistical Planning and Inference*, Vol.83, No.2, pp.395–412 (2000).
31) Wolpaw, J.R., Birbaumer, N., McFarland, D.J., Pfurtscheller, G. and Vaughan, T.M.: Brain-Computer Interfaces for Communication and Control, *Clinical Neurophysiology*, Vol.113, No.6, pp.767–791 (2002).
32) Zadrozny, B.: Learning and Evaluating Classifiers under Sample Selection Bias, *Proc. 21st International Conference on Machine Learning*, New York, NY, ACM Press (2004).

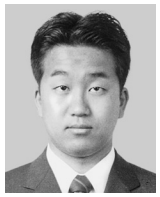
(Received June 2, 2008)
(Accepted January 7, 2009)
(Released April 8, 2009)

Yuta Tsuboi was born in 1976. He received his M.E. from Nara Institute of Science and Technology in 2002, and has been engaged in research on text mining at IBM Research from 2002. His research interests include natural language processing, machine learning and data mining. He is a member of IPSJ.

Hisashi Kashima was born in 1975. He is a researcher of Tokyo Research Laboratory of IBM Research since 1999. He received his M.E. and Ph.D. degrees from Kyoto University in 1999 and 2007, respectively. His research interests include machine learning and its application to bioinformatics, industrial analytics, and business intelligence.

Shohei Hido was born in 1981. He received B. Eng. and M. Informatics from Kyoto University in 2004 and 2006 respectively. He has been a researcher of Tokyo Research Laboratory of IBM Research since 2006. His research interests include machine learning, data mining and their applications to industry.

Steffen Bickel was born in 1974. He received his joint masters degree in computer science and business administration from University of Darmstadt, Germany in 2002. He is Ph.D. student at Potsdam University, Germany. His research interests include machine learning and information retrieval.



Masashi Sugiyama was born in 1974 in Osaka, Japan. He received the B. Eng., M. Eng., and D. Eng. degrees from Department of Computer Science, Tokyo Institute of Technology, Tokyo, Japan, in 1997, 1999, and 2001, respectively. In 2001, he was appointed as a Research associate in the same institute, and from 2003, he is an Associate professor. His research interests include machine learning and signal/image processing.