Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation

Yuta Tsuboi (*), Hisashi Kashima, Shohei Hido,
IBM Research, Tokyo Research Laboratory
** Nara Institute of Science and Technology*
Steffen Bickel,
Max Planck Institute for Computer Science
Masashi Sugiyama
Tokyo Institute of Technology, Department of Computer Science

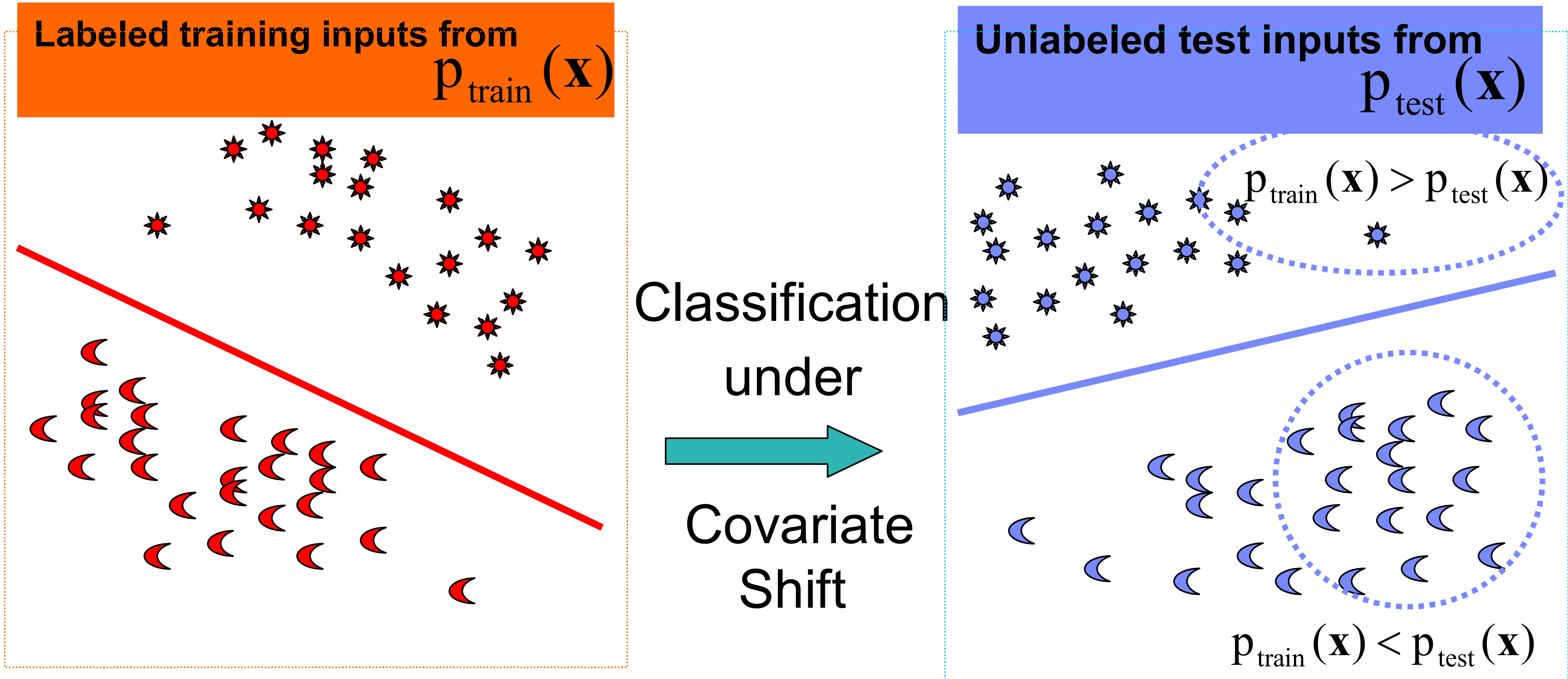
Abstract

- **Problem**: *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.
- **Contributions**:
  - **LL-KLIEP**: KLIEP (Sugiyama, *et. al.* 2007) for Log-linear models
    - Natural modeling for density ratio functions
  - **LL-KLIEP(LS)**: Another optimization technique for LL-KLIEP
    - the computation time is nearly independent of the number of test input samples, and
    - the memory requirement is independent of the number of test input samples
    - which is beneficial in applications with large numbers of unlabeled samples.

Covariate shift situation

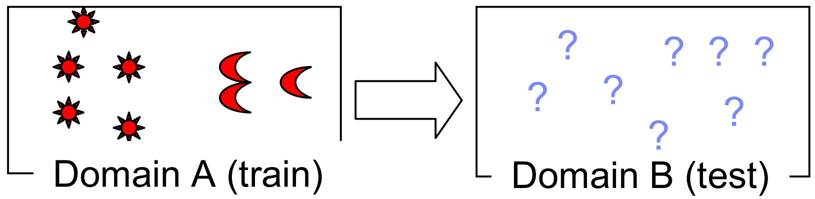
Training and test *inputs* x follow different distributions

- **Input distribution changes:** $p_{\text{train}}(\mathbf{x}) \neq p_{\text{test}}(\mathbf{x})$
- **Functional relation remains unchanged:** $p_{\text{train}}(y \mid \mathbf{x}) = p_{\text{test}}(y \mid \mathbf{x})$



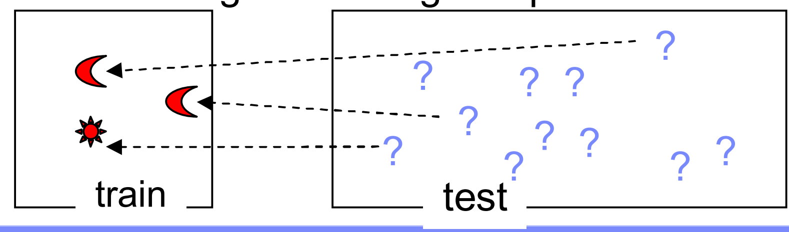
Examples of covariate shift situation
Domain Adaptation & Selective Sampling (Active Learning)

- **Domain adaptation of statistical classifiers**
  - The data distribution in the test domain is different from that in the training domain. (Note: the functional relation can also be changed)
    - E.g.: Spam filters can be trained on public collections of spam, but are applied to an individual person's inbox. (Personalization)



- **Selective sampling (active learning) of statistical classifiers**
  - The learning algorithm can actively query the teacher for labels.
  - Since the learner chooses
    the examples by design,
    the data distribution of
    the labeled training examples is
    different from that of a sample pool.

Selecting & Labeling samples



A common approach for covariate shift situation
Weighting the training examples by importance.

- **Density ratio (importance):** $w(\mathbf{x}) = \dfrac{p_{\text{test}}(\mathbf{x})}{p_{\text{train}}(\mathbf{x})}.$

- **Example: Importance Weighted Logistic Regression (IWLR)**
  - Weighted Log-likelihood for Logistic Regression (LR)

$$\int\int p_{\text{test}}(\mathbf{x})\,p(y\,|\,\mathbf{x})\,p_{\boldsymbol{\theta}}(y\,|\,\mathbf{x})\,d\mathbf{x}dy$$

LR model

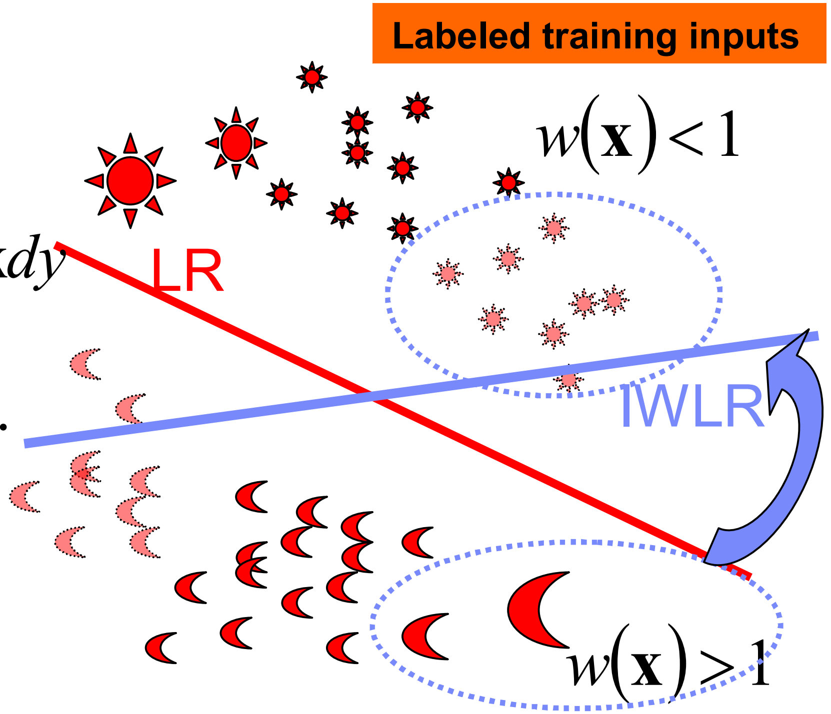
$$= \int\int p_{\text{train}}(\mathbf{x})\frac{p_{\text{test}}(\mathbf{x})}{p_{\text{train}}(\mathbf{x})}p(y\,|\,\mathbf{x})\log p_{\boldsymbol{\theta}}(y\,|\,\mathbf{x})\,d\mathbf{x}dy$$

$$\approx \sum_{(\mathbf{x},y)\in Z_{\text{train}}} w(\mathbf{x})\left(y\,f_{\boldsymbol{\theta}}(\mathbf{x}) - \log\left(1+\exp\left(y\,f_{\boldsymbol{\theta}}(\mathbf{x})\right)\right)\right).$$

Training data

Density ratio (Importance)

Log-likelihood



We need to estimate the density ratio from samples.

Importance Estimation

- **Problem setting:** i.i.d. training and test samples are given

Training inputs: $D_{tr} = \{x_i\}_{i=1}^{N_{tr}}$ from $\mathrm{P}_{train}(\mathbf{x})$

Test inputs: $D_{te} = \{x_i\}_{i=1}^{N_{te}}$ from $\mathrm{P}_{test}(\mathbf{x})$

- Naïve approach: estimate $\mathrm{P}_{train}(\mathbf{x})$ and $\mathrm{P}_{test}(\mathbf{x})$ separately, and take the ratio of the density estimates
- However, density $\mathrm{P}(\mathbf{x})$ estimation is the hard problem particularly in high dimensional cases.

Modeling Density Ratio by Log-linear Model

- We use a **log-linear model**:

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

α : model parameter

$\psi(\mathbf{x})$: basis function

- Log-linear model
  - $\hat{w}(\mathbf{x})$ takes only non-negative values.

→ **natural modeling for ratio** （α and $\psi(\mathbf{x})$ can be an arbitrary value）

- **Test density** is approximated by

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

$$p_{test}(\mathbf{x}) = \cancel{p_{train}(\mathbf{x})} \frac{p_{test}(\mathbf{x})}{\cancel{p_{train}(\mathbf{x})}}$$

- **Learn** α so that $\hat{p}_{test}(\mathbf{x})$ approximates $p_{test}(\mathbf{x})$ well.

→ The denominator guarantees $\hat{p}_{test}(\mathbf{x})$ be a probability density function

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

Training data set

Kullback—Leibler (KL) Divergence

- Minimize KL divergence between $\mathrm{p}_{\mathrm{test}}(\mathbf{x})$ and $\hat{\mathrm{p}}_{\mathrm{test}}(\mathbf{x})$:

$$\arg\min_{\boldsymbol{\alpha}} \mathrm{KL}\left[\mathrm{p}_{\mathrm{test}}(\mathbf{x}) \,\|\, \hat{\mathrm{p}}_{\mathrm{test}}(\mathbf{x})\right]$$

$$\hat{\mathrm{p}}_{\mathrm{test}}(\mathbf{x}) = \mathrm{p}_{\mathrm{train}}(\mathbf{x})\hat{w}(\mathbf{x})$$

$$\mathrm{KL}\left[\mathrm{p}_{\mathrm{test}}(\mathbf{x}) \,\|\, \hat{\mathrm{p}}_{\mathrm{test}}(\mathbf{x})\right]$$

$$= \int \mathrm{p}_{\mathrm{test}}(\mathbf{x}) \log \frac{\mathrm{p}_{\mathrm{test}}(\mathbf{x})}{\hat{\mathrm{p}}_{\mathrm{train}}(\mathbf{x})\hat{w}(\mathbf{x})} d\mathbf{x}$$

$$= \underbrace{\int \mathrm{p}_{\mathrm{test}}(\mathbf{x}) \log \frac{\mathrm{p}_{\mathrm{test}}(\mathbf{x})}{\mathrm{p}_{\mathrm{train}}(\mathbf{x})} d\mathbf{x}}_{\text{constant}} - \underbrace{\int \mathrm{p}_{\mathrm{test}}(\mathbf{x}) \log \hat{w}(\mathbf{x}) d\mathbf{x}}_{\text{relevant}}$$

Learning Importance Function

- Thus,

$$\underset{\boldsymbol{\alpha}}{\arg\min}\, \mathrm{KL}\left[\mathrm{p}_{\text{test}}(\mathbf{x}) \,\|\, \hat{\mathrm{p}}_{\text{test}}(\mathbf{x})\right]$$

$$\Leftrightarrow \underset{\boldsymbol{\alpha}}{\arg\max} \int \mathrm{p}_{\text{test}}(\mathbf{x}) \log \hat{w}(\mathbf{x})\, d\mathbf{x}$$

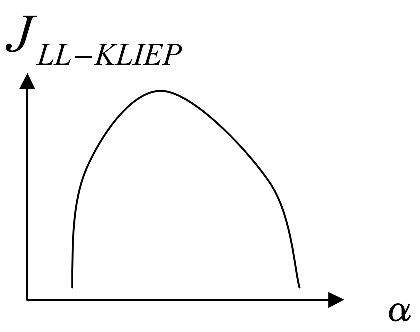
Objective function

- Empirical approximation of objective function (*LL-KLIEP*)

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

Test data set

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



- Unconstraint convex optimization:
  - standard gradient ascent method can be used.
  - unique global solution is available.

Kullback-Leibler Importance Estimation Procedure (KLIEP) for Log-linear Models: LL-KLIEP

- Regularized version of LL-KLIEP

$$\jmath(\boldsymbol{\alpha}) = \frac{1}{N_{te}} \sum_{\boldsymbol{x} \in D_{te}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle - \log \frac{1}{N_{tr}} \sum_{\boldsymbol{x} \in D_{tr}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}$$

regularizer

- Gradient of the objective function

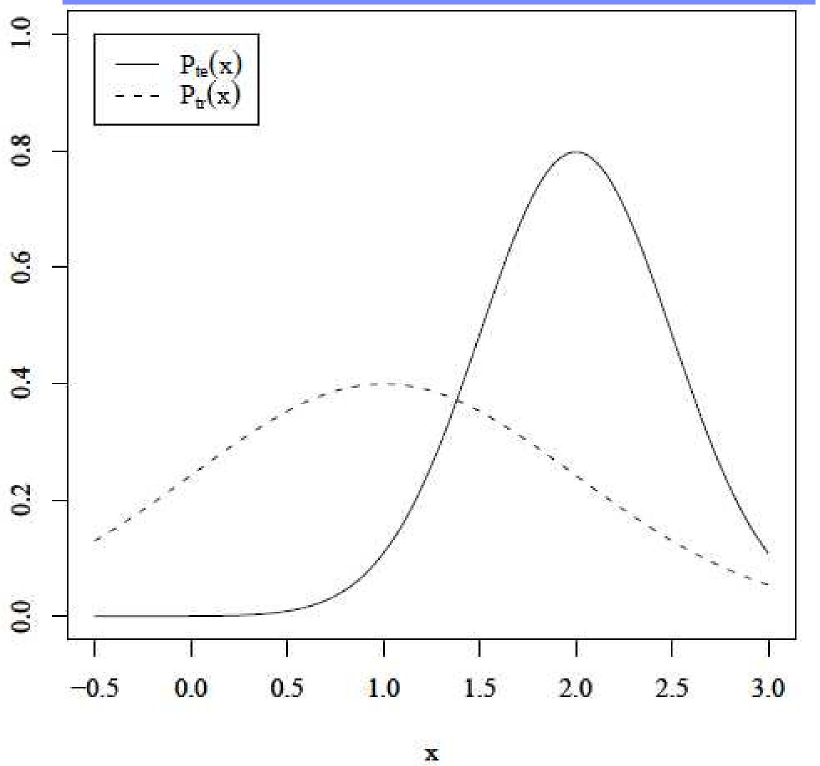
$$\frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{1}{N_{te}} \sum_{\boldsymbol{x} \in D_{te}} \boldsymbol{\psi}(\boldsymbol{x}) - \sum_{\boldsymbol{x} \in D_{tr}} \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\sum_{\boldsymbol{x}'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

Samples were generated from two Gaussian distributions.
We used 100 Gaussian basis functions (Gaussian kernels) centered at randomly chosen test input samples.

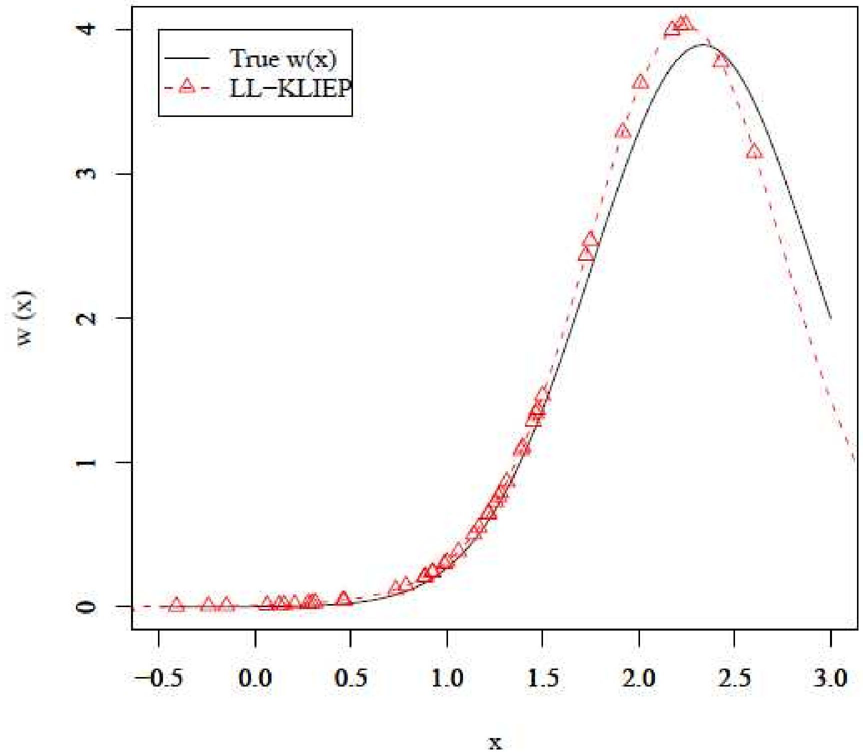
$$\hat{w}(\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\frac{1}{N_{tr}} \sum_{\boldsymbol{x}' \in D_{tr}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}') \rangle)}$$

$$\psi_l(\mathbf{x}) = \exp\left(- \frac{\left\| \mathbf{x} - \mathbf{x}_l^{\text{test}} \right\|^2}{2s^2} \right)$$

Training and Test Densities



Estimated Importance



Model selection of KLIEP/LL-KLIEP
Likelihood Cross Validation (LCV)

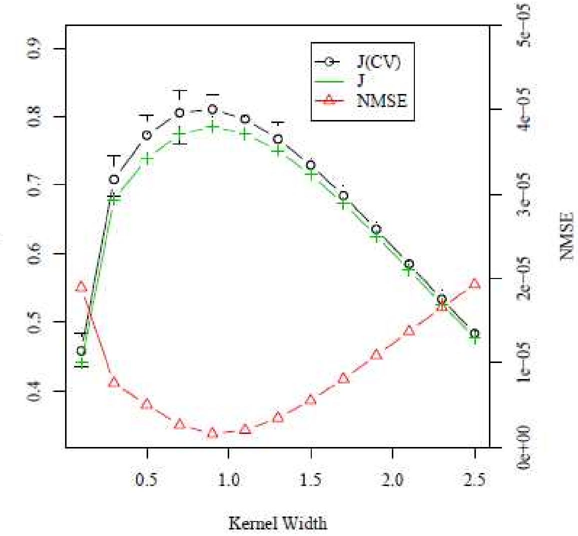
- The performance of KLIEP depends on the choice of the basis functions $\boldsymbol{\psi}(\mathbf{x})$

  → How to choose hyper parameters, e.g., the kernel width s for Gaussian kernels:
  $$K_s(\boldsymbol{x}, \boldsymbol{x}_l) = \exp\left\{-\frac{\|\boldsymbol{x} - \boldsymbol{x}_l\|^2}{2s^2}\right\},$$

- However, the correct value of importance for each **x** is not available for unknown distributions $\mathrm{p}_{\mathrm{train}}(\mathbf{x})$ and $\mathrm{p}_{\mathrm{test}}(\mathbf{x})$

  → unsupervised learning setting

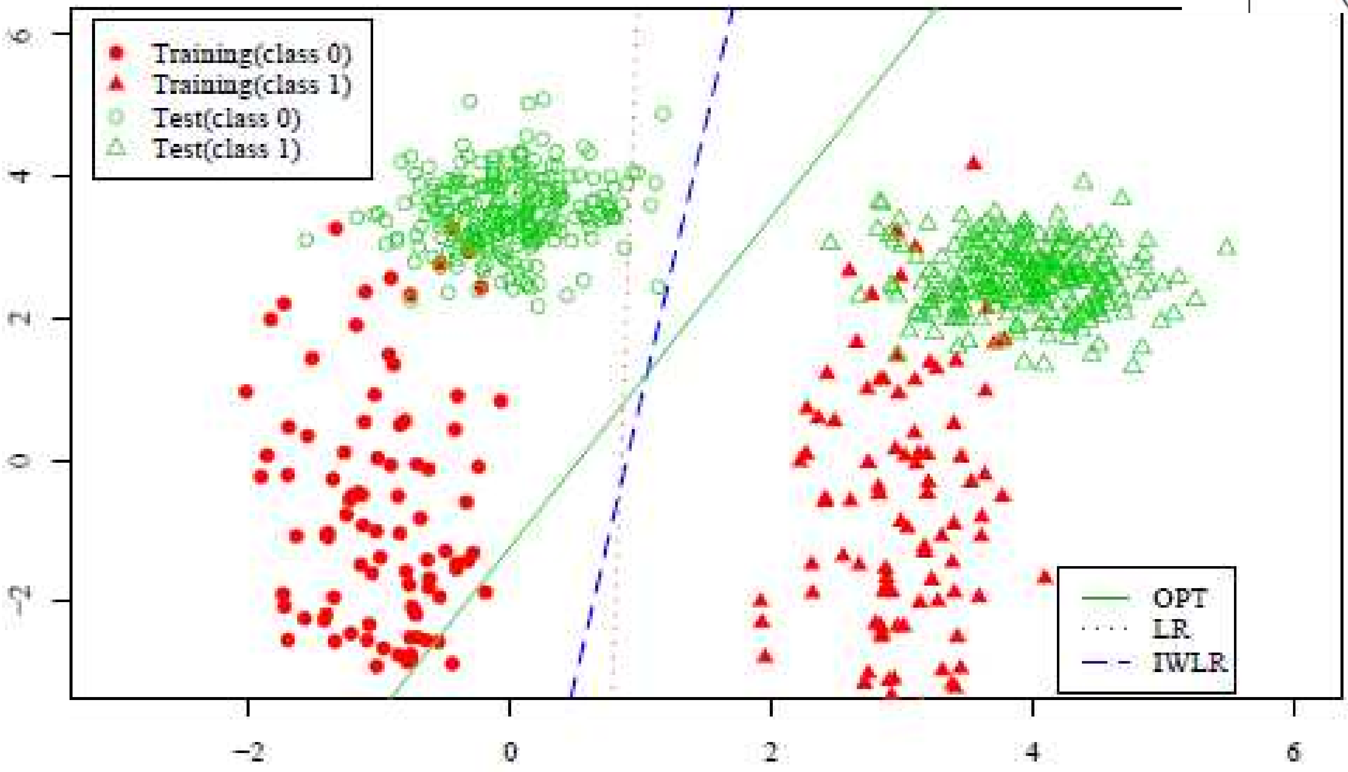
- LCV: Select the model such that maximized $J(\boldsymbol{\alpha})$
  1. Divide test samples into R disjoint subsets: $\{D^r_{\mathrm{te}}\}_{r=1}^R$
  2. Learn importance: $\hat{w}^r(\boldsymbol{x})$ from $\{D^t_{\mathrm{te}}\}_{t \neq r}^R$
  3. Evaluate a model by likelihood: $$\frac{1}{|D^r_{\mathrm{te}}|} \sum_{\boldsymbol{x} \in D^r_{\mathrm{te}}} \hat{w}^r(\boldsymbol{x})$$



Classification example under Covariate shift 2-dimensional samples were generated from Gaussian distributions

- We used Importance Weighted Logistic Regression (IWLR)

	Training $p_{tr}(\boldsymbol{x}, y)$		Test $p_{te}(\boldsymbol{x}, y)$	
	$y = 0$	$y = 1$	$y = 0$	$y = 1$
μ	(-1,-1)	(3,-1)	(0,3.5)	(4,2.5)
Σ	$\begin{pmatrix} 0.25 & 0 \\ 0 & 4 \end{pmatrix}$		$\begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$	



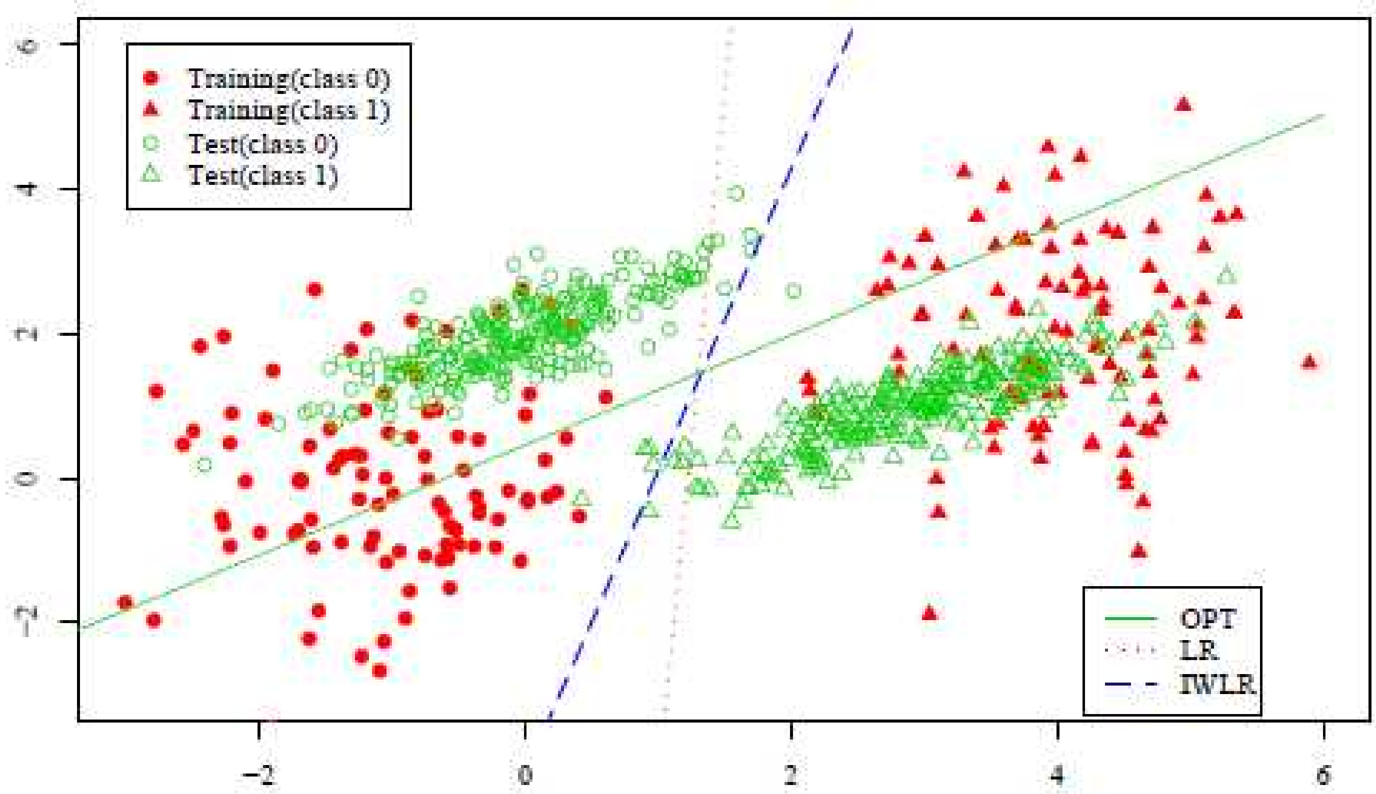
(a) $p_{te}(\boldsymbol{x})$ is linearly shifted from $p_{tr}(\boldsymbol{x})$.

Correct classification rate of **LR** is **99.1%** while that of **IWLR** is **100%.**

Classification example under Covariate shift 2-dimensional samples were generated from Gaussian distributions

- We used Importance Weighted Logistic Regression (IWLR).

μ	(-1,0)	(4,2)	(0,2)	(3,1)
Σ	$\begin{pmatrix} 0.75 & 0 \\ 0 & 1.5 \end{pmatrix}$		$\begin{pmatrix} 0.75 & 0.5 \\ 0.01 & 0.1 \end{pmatrix}$	



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

Correct classification rate of **LR** is **97.2%** while that of IWLR is **99.1%.**

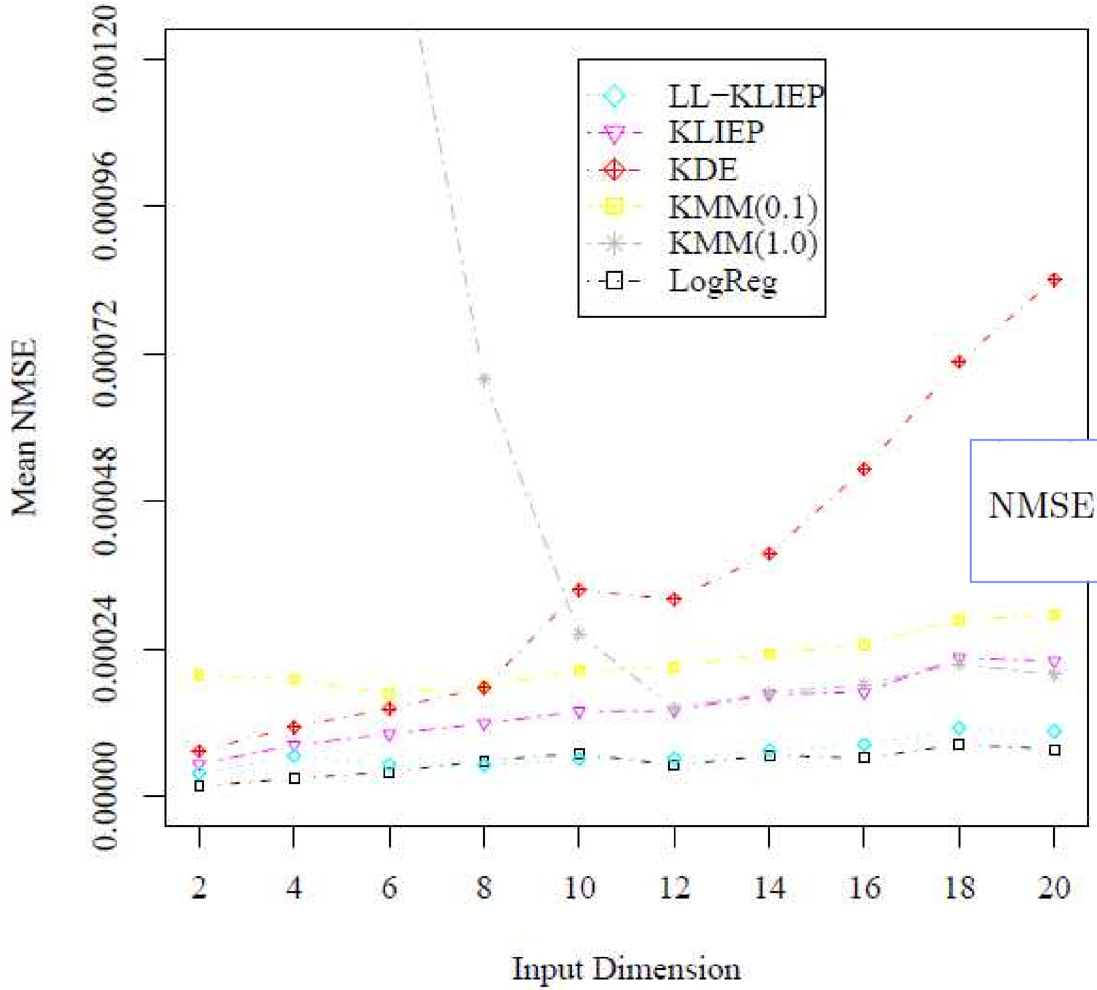
Related Work of Density Ratio Estimation

$$w(\mathbf{x}) = \frac{p_{\text{test}}(\mathbf{x})}{p_{\text{train}}(\mathbf{x})}.$$

- **Kernel density estimator (KDE)**
  - Separately estimate training and test input densities.
  - Gaussian kernel width is chosen by likelihood cross-validation.
- **Kernel Mean Matching (KMM)** (Huang *et al.*, NIPS2006)
  - Direct importance estimation method in universal reproducing kernel Hilbert spaces (RKHS)
  - There is <u>no model selection method</u> for kernel width.
- **Logistic regression (LogReg)** (Beckel *et al.*, ICML2007)
  - Classifier discriminating training and test input data.
  - Gaussian kernel width is chosen by likelihood cross-validation.
- **Kullback-Leibler Importance Estimation Procedure (KLIEP)** (Sugiyama *et al.*, NIPS2007)
  - Direct importance estimation method using KL Divergence.
  - Gaussian kernel width is chosen by likelihood cross-validation.

Experiments
varying input dimension

$$p_{tr}(x) = \mathcal{N}(0_d, I_d)$$
$$p_{te}(x) = \mathcal{N}((1,0,\ldots,0)^\top, 0.75^2 I_d)$$



Mean NMSE over 100 trials.

KMM (s) denotes KMM with kernel width s

NMSE:
Normalized Mean Squared Error

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

KDE: Suffers from the curse of dimensionality

KMM: Performance depends on kernel width

KLIEP, LogReg, and LL-KLIEP: Model selection by LCV works well

Disadvantage: LL-KLIEP and previous methods require to use all test inputs in their optimization procedure.

- We need to iterate over all test inputs when computing the values of the objective function:

$$\jmath(\boldsymbol{\alpha}) = \frac{1}{N_{te}} \sum_{\boldsymbol{x} \in D_{te}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle$$

Evaluation over Test data set

$$- \log \frac{1}{N_{tr}} \sum_{\boldsymbol{x} \in D_{tr}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle) - \frac{||\boldsymbol{\alpha}||^2}{2\sigma^2}$$

Evaluation over Training data set

- However, the derivative of the objective function requires the evaluation of all test samples <u>once</u>.

$$\frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = F - \frac{1}{N_{\mathrm{tr}}} \sum_{\boldsymbol{x} \in D_{\mathrm{tr}}} \hat{w}(x) \boldsymbol{\psi}(\boldsymbol{x}) - \frac{\boldsymbol{\alpha}}{\sigma^2}$$

$F = \frac{1}{N_{\mathrm{te}}} \sum_{\boldsymbol{x} \in D_{\mathrm{te}}} \psi(x)$: **Independent from** α **→ Pre-computing the value**

An optimization technique w/o the objective function evaluation
LL-KLIEP(LS1)

- Idea: the derivative of the convex objective function to be zero at the optimum point.
  → Minimizing a squared norm to measure the 'magnitude' of the derivative:

Objective function for LL-KLIEP(LS1)
$$\jmath_{\mathrm{LS}}(\boldsymbol{\alpha}) = \frac{1}{2}\left\|\frac{\partial \jmath(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}\right\|^2.$$

- The partial derivative of LL-KLIEP(LS1):

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

- Computation time & memory size are independent of N_{te}.
  - However, the derivative is a quadratic function of the number of parameters, which could be a bottleneck in high dimensional problems.

LL-KLIEP(LS2)

- Idea: representing the parameter α as a linear combination of the training inputs (representer theorem (Wahba 1990)):

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

where $\{\beta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in D_{\mathrm{tr}}}$ is a data-wise parameter.

Objective function for LL-KLIEP(LS2)

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

where

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

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

  – Now, the computation time is linear w.r.t. the number of parameters (quadratic w.r.t. the number of the training inputs).

LL-KLIEP (LS): No iteration and no storage for N_{te} in optimization → Well-suited to the applications with the large amount of test samples

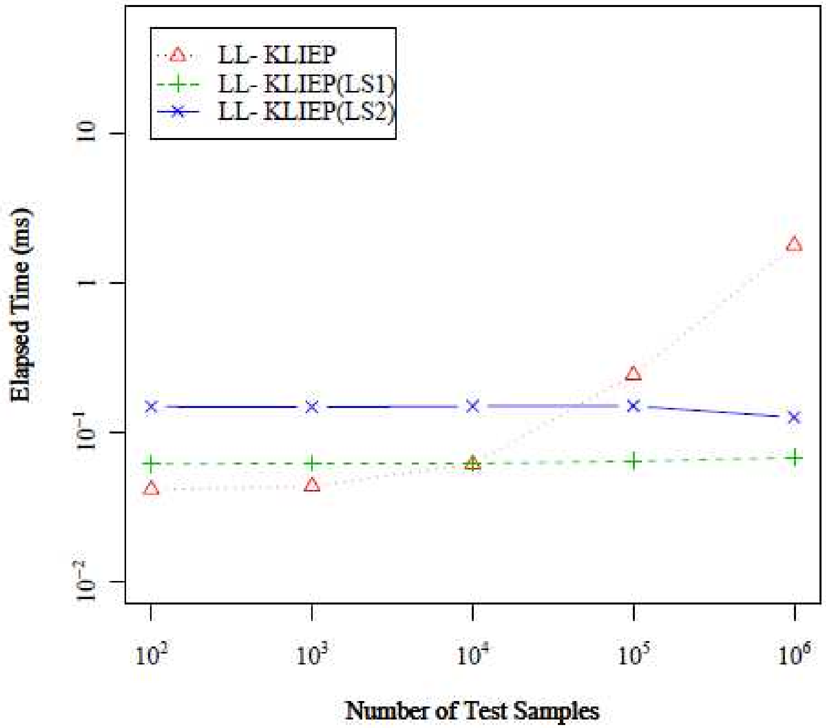
Computational complexity and space requirements. N_{tr} is the number of training samples, N_{te} is the number of test samples, b is the number of parameters, and c is the average number of non-zero basis entries.

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

- LL-KLIEP (LS1) : For lower-dimensional and large-scale training data.
- LL-KLIEP (LS2): For higher-dimensional and moderate-size training data.

Average computation time (including Pre-comp.) over 100 trials
We varied the number of test inputs, and fixed the number of training inputs.

- we used linear basis function so that the number of bases is equivalent to the input dimension.
  - > d: input dimension = #parameter,
    N_{tr}: The number of training inputs, N_{te}: The number of test inputs



(a) $d = 10, N_{tr} = 100$

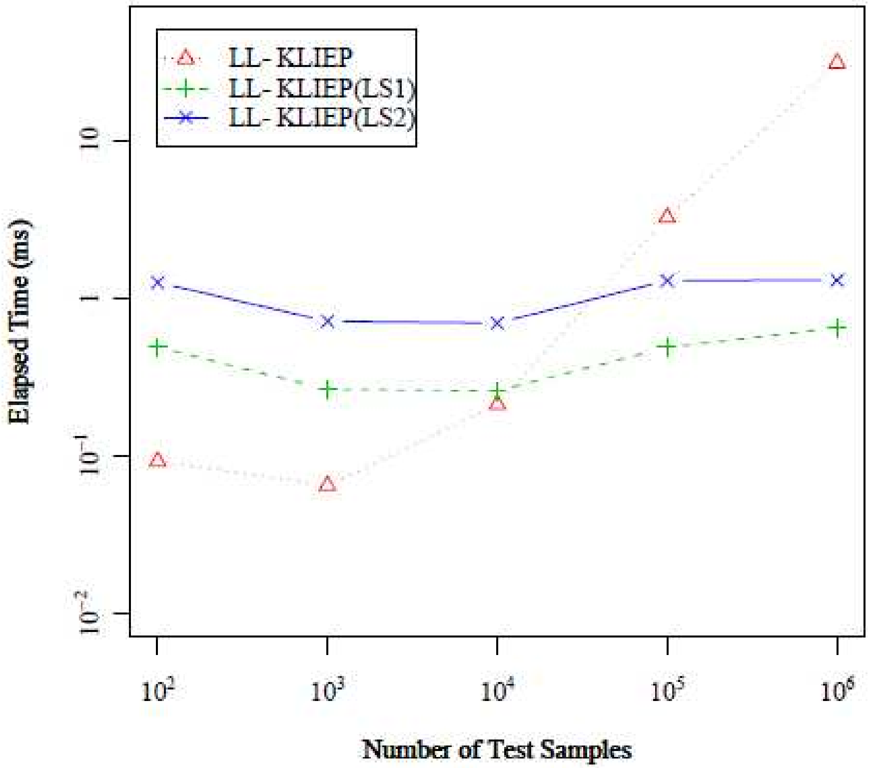
Lower-dimensional data

The computation time of LL-KLIEP(LS) is independent from the number of test inputs.

Average computation time (including Pre-comp.) over 100 trials
We varied the number of test inputs, and fixed the number of training inputs.

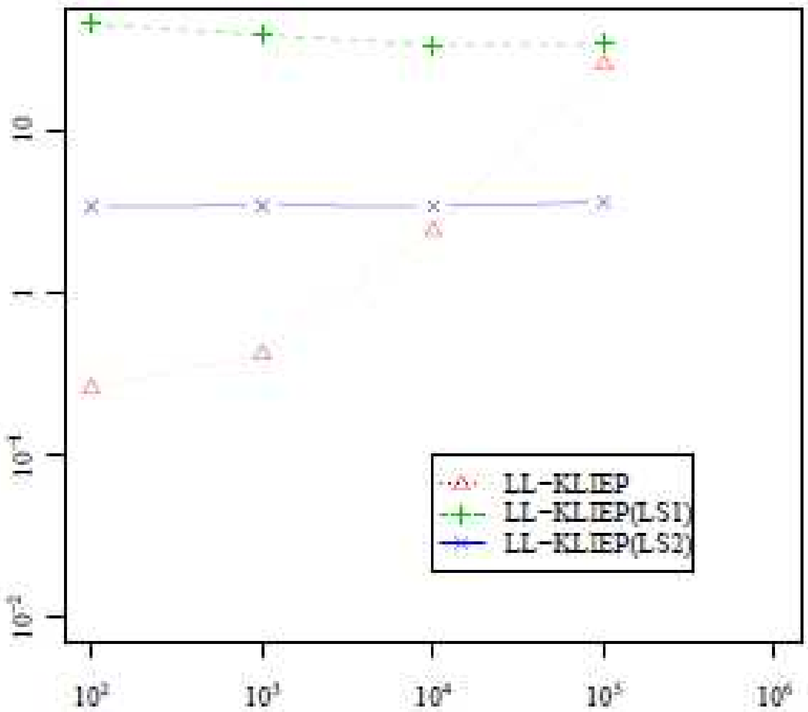
- d: input dimension = #parameter,
  N_{tr}: The number of training inputs, N_{te}: The number of test inputs

Moderate-dimensional data



(b) $d = 100, N_{tr} = 100$

Higher-dimensional data



(c) $d = 1000, N_{tr} = 100$

Related work: Kernel Mean Matching (KMM)

LL-KLIEP (LS2) without a regularizer has the same form as the objective function of KMM.

- Moment matching method:

Objective function for KMM

$$\min_{\{w(\boldsymbol{x})\}_{\boldsymbol{x}\in D_{\mathrm{tr}}}} \left[\frac{1}{2}\sum_{\boldsymbol{x},\boldsymbol{x}'\in D_{\mathrm{tr}}} w(\boldsymbol{x})w(\boldsymbol{x}')K_s(\boldsymbol{x},\boldsymbol{x}') - \sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x})\kappa(\boldsymbol{x})\right]$$

$$\text{subject to } \left|\sum_{\boldsymbol{x}\in D_{\mathrm{tr}}} w(\boldsymbol{x}) - N_{\mathrm{tr}}\right| \leq N_{\mathrm{tr}}\epsilon, \text{ and}$$

$$0 \leq w(\boldsymbol{x}) \leq B \text{ for all } \boldsymbol{x} \in D_{\mathrm{tr}},$$

where

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

Disadvantage of KMM.
The estimates of w(**x**) are only available for training samples→ Cannot optimize hyper parameters by CV

- The objective function of LL-KLIEP (LS2):

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

Related work: Logistic regression (LogReg) Classifier discriminating training and test input data

- Selector variable δ = -1 to the training input samples and δ = 1 to the test input samples:

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

- Importance can be $$w(x) = \frac{p(\delta = -1)}{p(\delta = 1)} \frac{p(\delta = 1|x)}{p(\delta = -1|x)}.$$

- The conditional probability p(δ j**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.

$$\hat{w}(x) = \underbrace{\frac{N_{\mathrm{tr}}}{N_{\mathrm{te}}}}_{\text{Empirical estimation}} \exp(\langle \alpha, \psi(x) \rangle)$$

- Objective function: regularized maximum likelihood estimation
- Disadvantage: summation over both training and test samples in CV.