# Direct Density Ratio Estimation for Large-scale Covariate Shift Adaptation

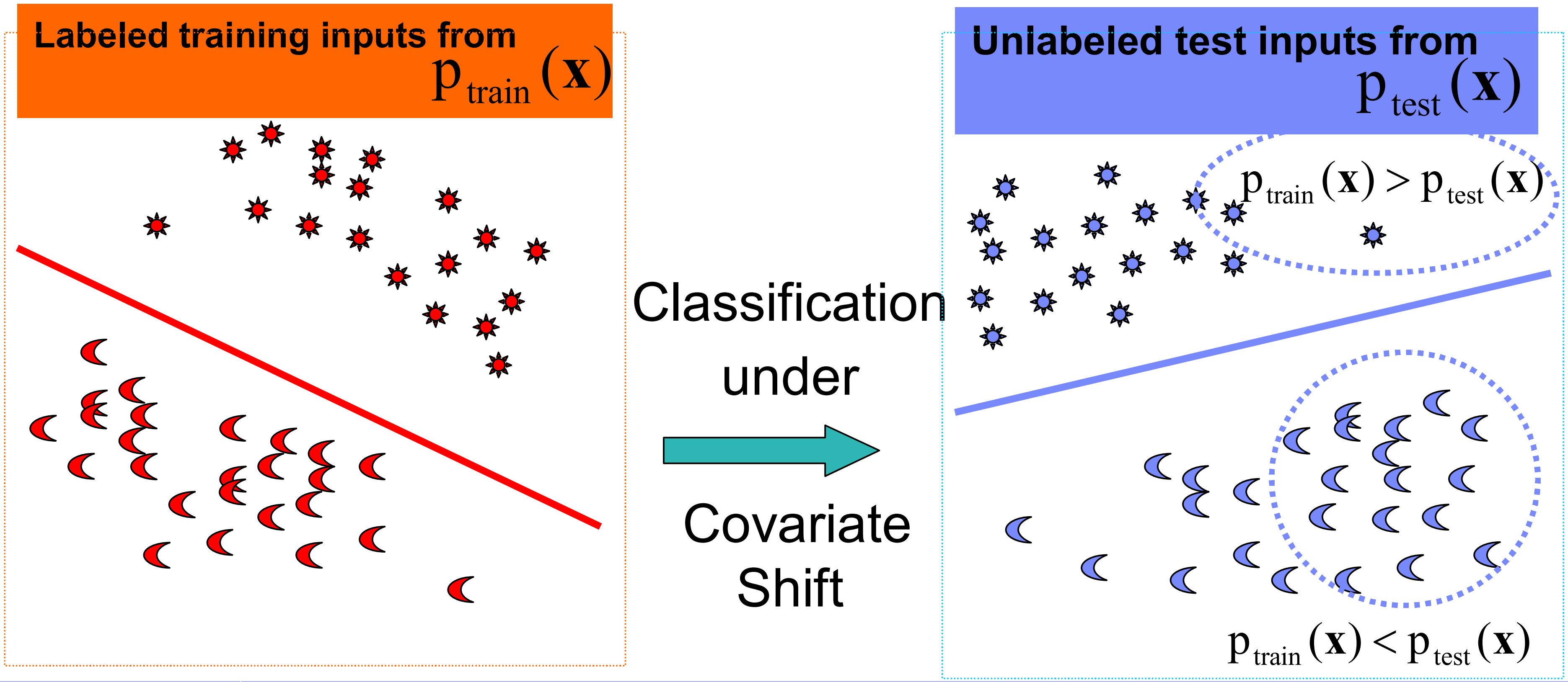
**Yuta Tsuboi (*), Hisashi Kashima, Shohei Hido,**
*IBM Research, Tokyo Research Laboratory*
* *Nara Institute of Science and Technology*
**Steffen Bickel,**
*Max Planck Institute for Computer Science*
**Masashi Sugiyama**
*Tokyo Institute of Technology, Department of Computer Science*

# Table of Contents

# Covariate shift situation

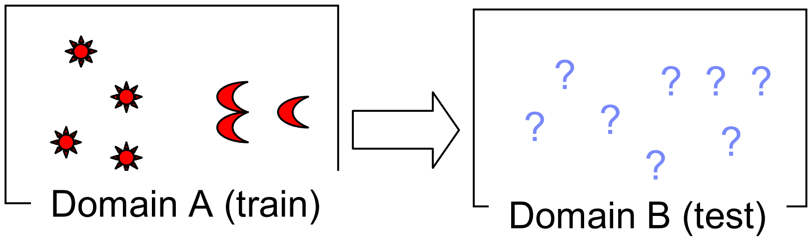
## Training and test *inputs* x follow different distributions

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

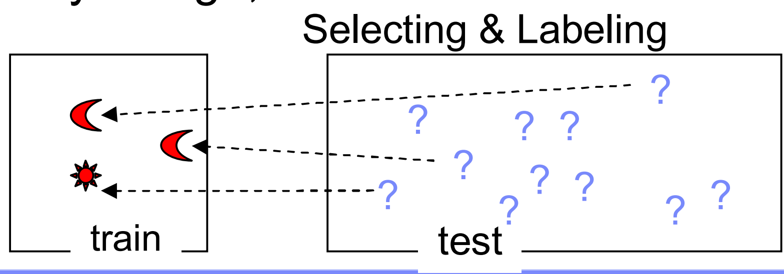


# Examples of covariate shift situation
# Domain Adaptation & Selective Sampling (Active Learning)

- **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.



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

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

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

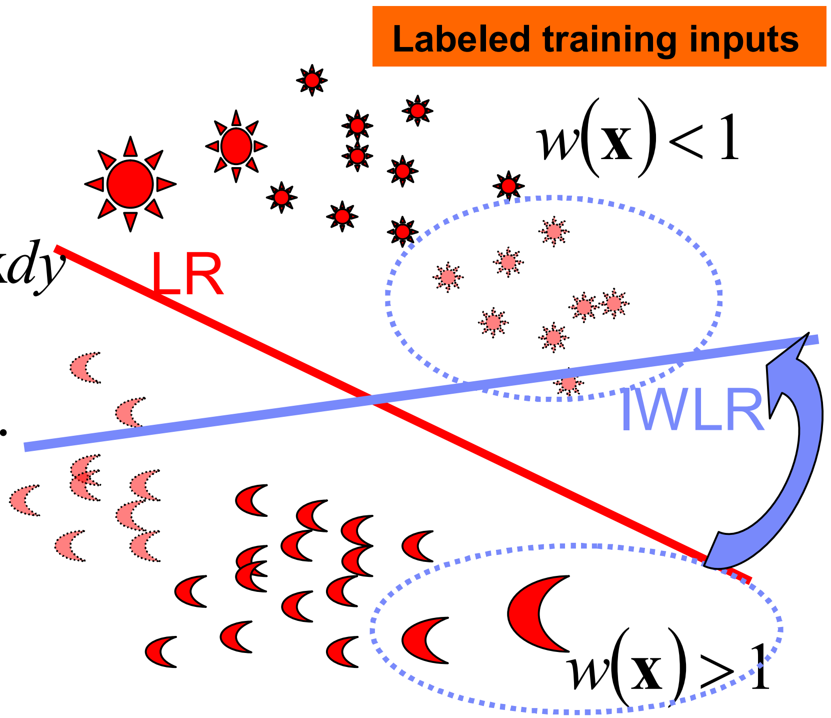
$$\iint p_{test}(\mathbf{x})\, p(y \mid \mathbf{x})\, p_{\theta}(y \mid \mathbf{x})\, d\mathbf{x}dy$$

LR model

$$= \iint p_{train}(\mathbf{x}) \frac{p_{test}(\mathbf{x})}{p_{train}(\mathbf{x})} p(y \mid \mathbf{x}) \log p_{\theta}(y \mid \mathbf{x})\, d\mathbf{x}dy$$

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

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

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

- Naïve approach: estimate $\mathrm{P}_{\mathrm{train}}(\mathbf{x})$ and $\mathrm{P}_{\mathrm{test}}(\mathbf{x})$ 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.

# Table of Contents

1. Motivation
   - Covariate shift situation
   - Direct density estimation methods
2. Proposed Method
   - LL-KLIEP: Log-linear model for KLIEP
     - Log-linear model: natural modeling for density ratio function
     - Standard optimization techniques to learn a density ratio function
   - LL-KLIEP(LS): Another optimization technique for LL-KLIEP
     - For applications with large numbers of unlabeled test inputs

## Modeling Density ratio by Log-linear Model

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

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

Training data set

$\alpha$ : model parameter

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

< , > : inner product

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

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

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

- **Test density** is approximated by

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

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

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

## Kullback—Leibler (KL) Divergence

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

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

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

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

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

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

constant

relevant

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

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

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

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

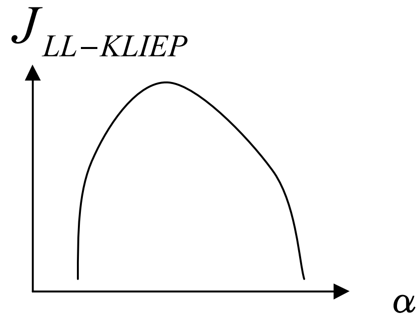
$$J_{LL-KLIEP}(\boldsymbol{\alpha}) = \frac{1}{N_{te}} \sum_{\boldsymbol{x} \in D_{te}} \log \hat{w}(\boldsymbol{x})$$

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

Test data set

Training data set

Objective function




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

# Mean Matching via LL-KLIEP

- Gradient of the objective function

$$\frac{\partial J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{1}{N_{\mathrm{test}}} \sum_{\mathbf{x} \in D_{\mathrm{test}}} \psi(\mathbf{x}) - \frac{1}{N_{\mathrm{train}}} \sum_{\mathbf{x} \in D_{\mathrm{train}}} w(\mathbf{x}) \psi(\mathbf{x})$$

The mean of Test data

The mean of Weighted Training data

At the optimum,
the mean $\psi(\mathbf{x})$ of test inputs = the mean $w(\mathbf{x})\psi(\mathbf{x})$ of training inputs.

→ Finding w(**x**) matching the mean of two distributions.

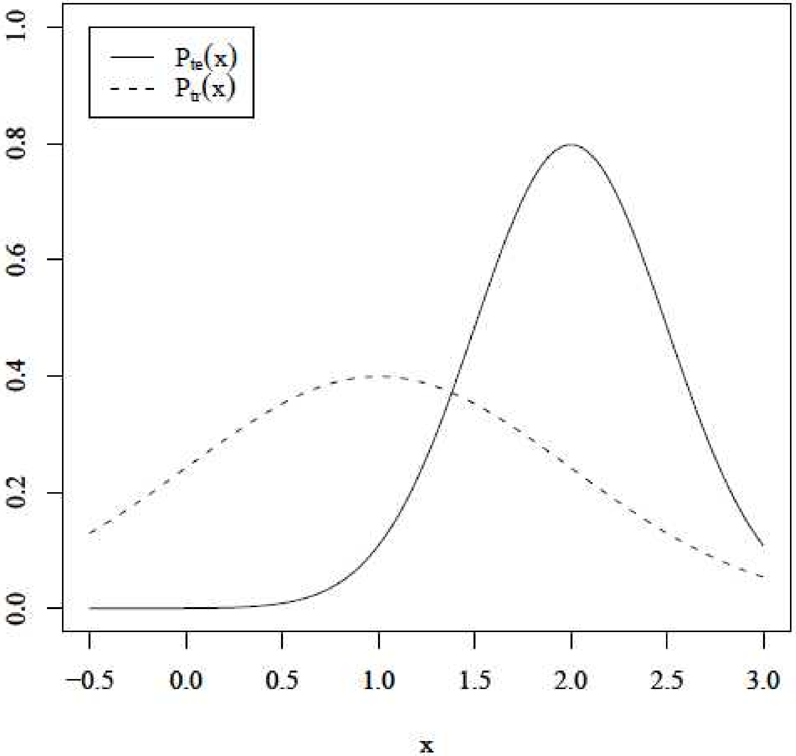
# Samples were generated from two Gaussian distributions.

## We used 100 Gaussian basis functions (Gaussian kernels) centered at randomly chosen test input samples.

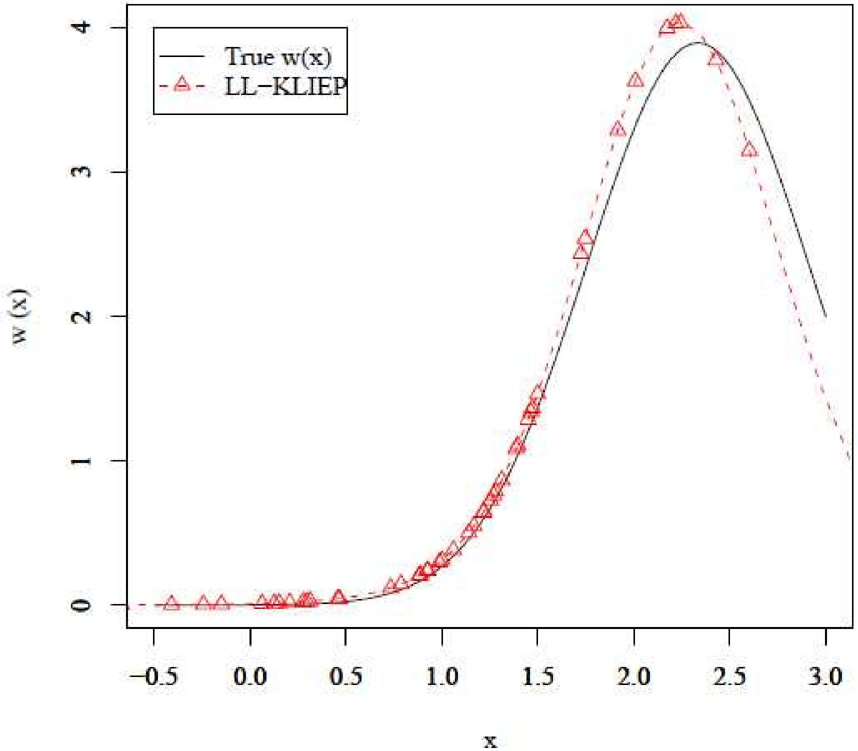
$$\hat{w}(\boldsymbol{x}) = \frac{\exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}) \rangle)}{\frac{1}{N_{tr}} \sum_{\boldsymbol{x}' \in D_{tr}} \exp(\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\boldsymbol{x}') \rangle)}$$

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

**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 $\psi(\mathbf{x})$

→ How to choose hyper parameters, e.g., the kernel width s for Gaussian kernels:

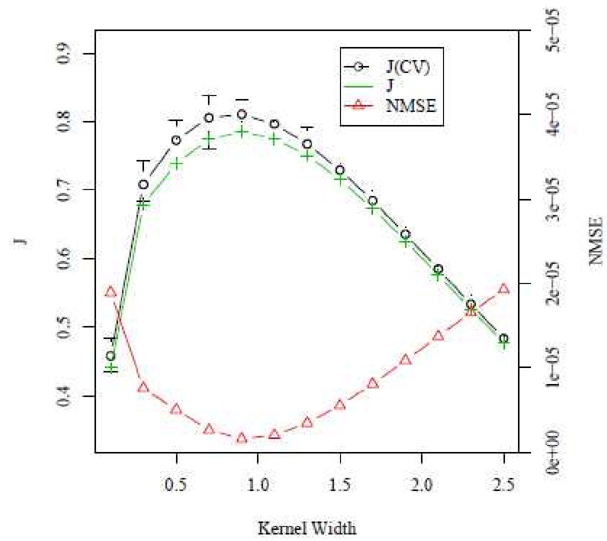
$$K_s(\boldsymbol{x}, \boldsymbol{x}_l) = \exp\left\{-\frac{\|\boldsymbol{x}-\boldsymbol{x}_l\|^2}{2s^2}\right\},$$

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

→ unsupervised learning setting

- LCV: Select the model such that maximized $J(\boldsymbol{\alpha})$

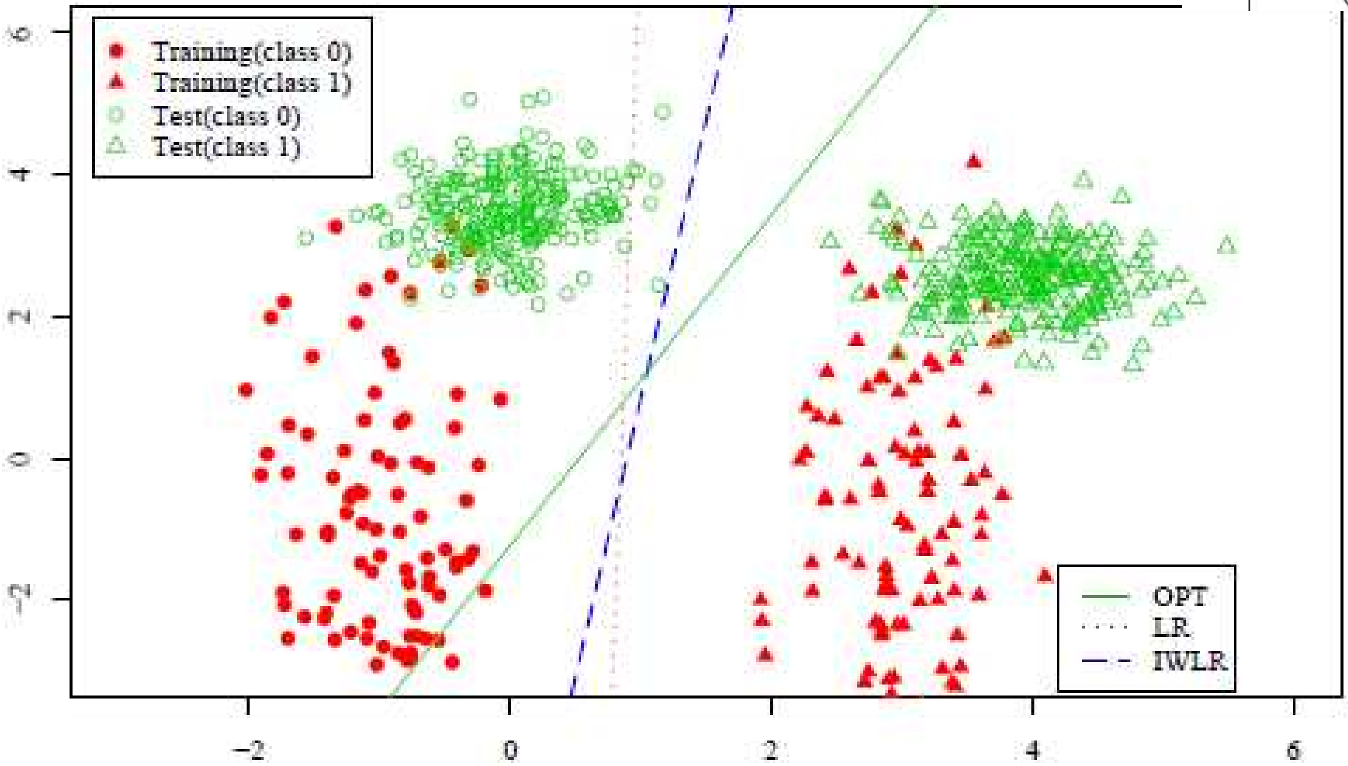
1. Divide test samples into R disjoint subsets: $\{D^r_{\text{te}}\}_{r=1}^R$
2. Learn importance: $\hat{w}^r(\boldsymbol{x})$ from $\{D^t_{\text{te}}\}_{t\neq r}^R$
3. Evaluate a model by likelihood: $\frac{1}{|D^r_{\text{te}}|}\sum_{\boldsymbol{x}\in D^r_{\text{te}}} \hat{w}^r(\boldsymbol{x})$



# 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 $p_{te}(\boldsymbol{x}, y)$ | |
|---|---|---|---|---|
| | $y=0$ | $y=1$ | $y=0$ | $y=1$ |
| $\mu$ | (-1,-1) | (3,-1) | (0,3.5) | (4,2.5) |
| $\Sigma$ | $\begin{pmatrix} 0.25 & 0 \\ 0 & 4 \end{pmatrix}$ | | $\begin{pmatrix} 0.25 & 0 \\ 0 & 0.25 \end{pmatrix}$ | |



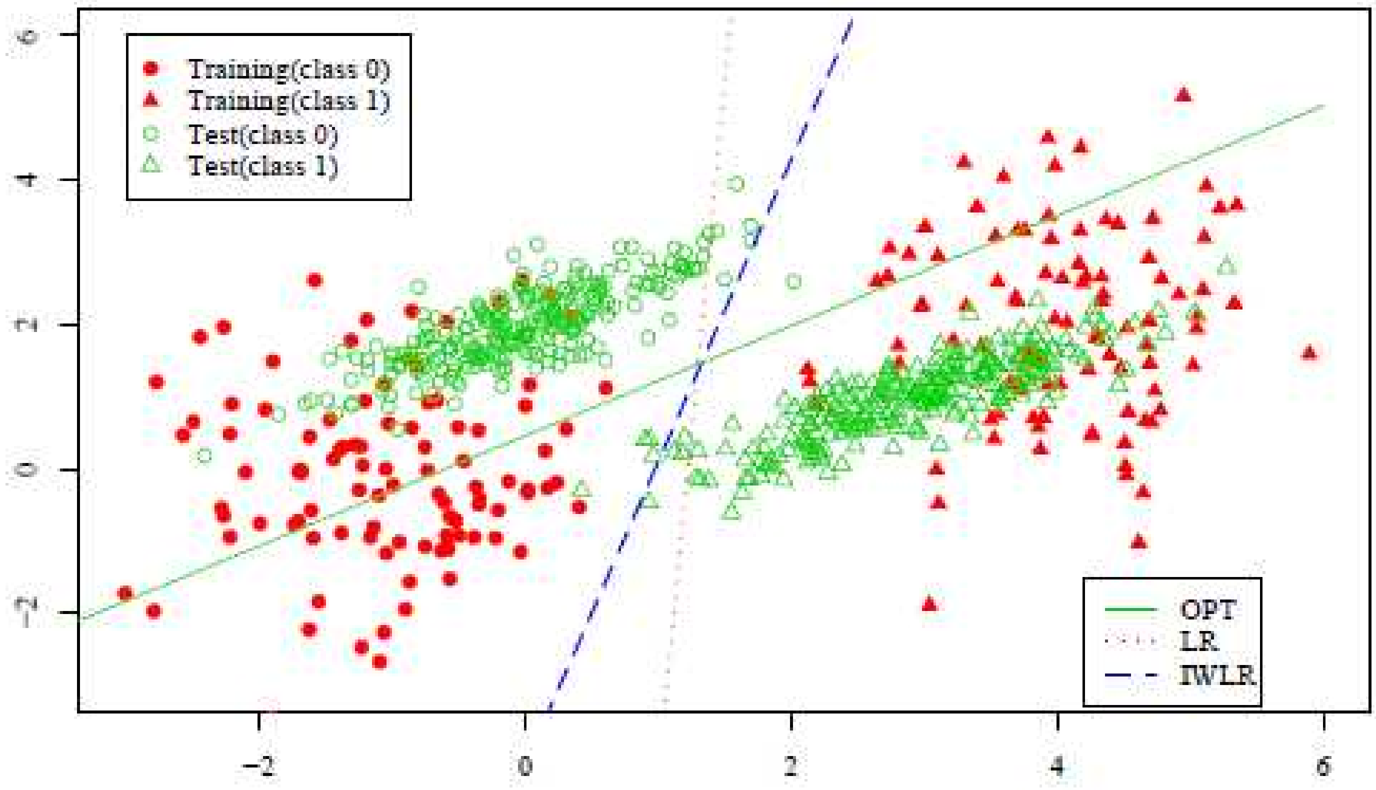

(a) $p_{te}(\boldsymbol{x})$ is linearly shifted from $p_{tr}(\boldsymbol{x})$.

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).

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




(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%.**

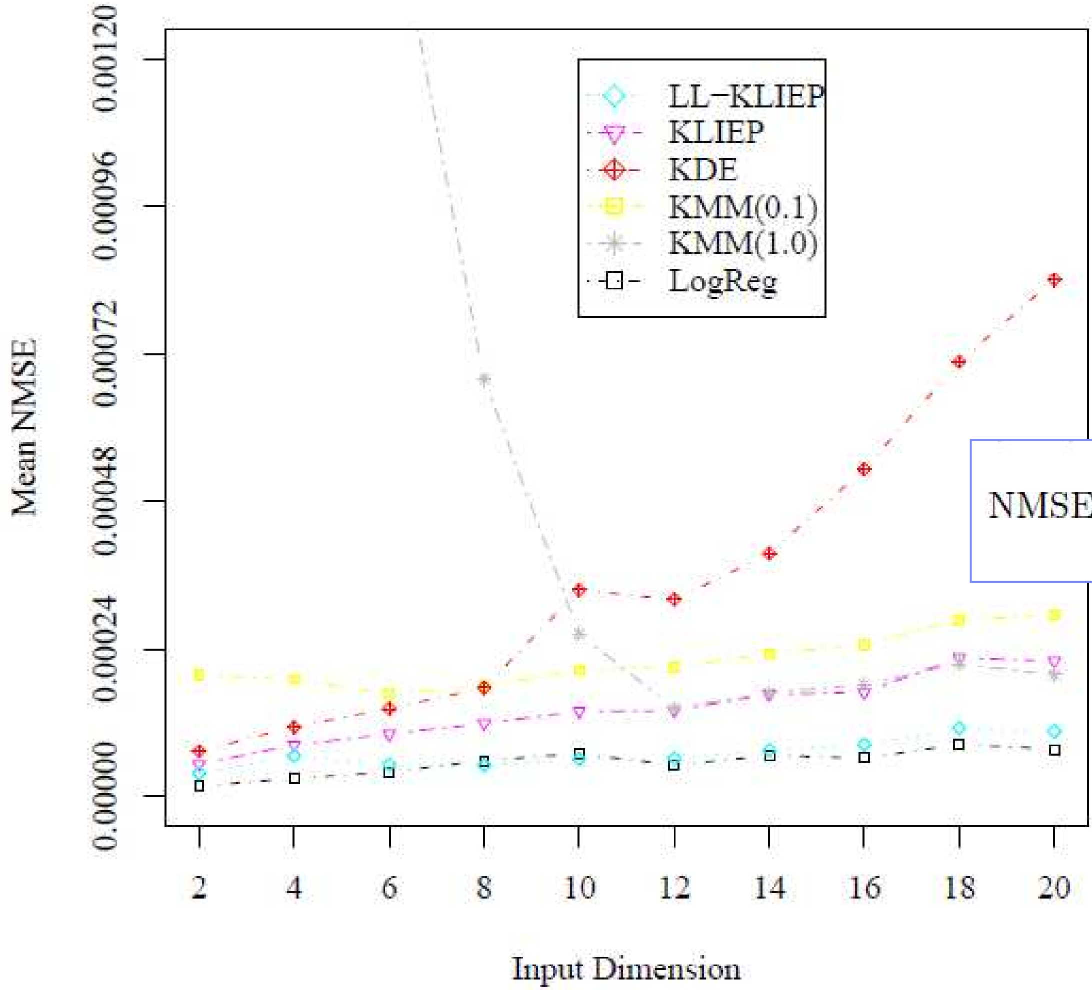
# Related Work of Density Ratio Estimation

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

- **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

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




**Mean NMSE over 100 trials.**

**KMM (s) denotes KMM with kernel width s**

**NMSE**:
Normalized Mean Squared Error

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

**KDE**: Suffers from the curse of dimensionality

**KMM**: Performance depends on kernel width

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

# Table of Contents



# 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**:

$$J_{\text{LL-KLIEP}}(\boldsymbol{\alpha}) = \frac{1}{N_{\text{test}}} \sum_{\mathbf{x} \in D_{\text{test}}} \langle \boldsymbol{\alpha}, \psi(\mathbf{x}) \rangle - \log \frac{1}{N_{\text{train}}} \sum_{\mathbf{x} \in D_{\text{train}}} \log \langle \boldsymbol{\alpha}, \psi(\mathbf{x}) \rangle$$

**Evaluation over Test data set**

Evaluation over Training data set

- However, the **gradient** of the objective function requires the evaluation of all test samples once.

$$\frac{\partial J_{\text{LL-KLIEP}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{1}{N_{\text{test}}} \sum_{\mathbf{x} \in D_{\text{test}}} \psi(\mathbf{x}) - \frac{1}{N_{\text{train}}} \sum_{\mathbf{x} \in D_{\text{train}}} w(\mathbf{x}) \psi(\mathbf{x})$$

**Independent from $\alpha$ → 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)

$$J_{\mathrm{LL-KLIEP(LS1)}} = \frac{1}{2}\left\| \frac{\partial J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} \right\|^2$$

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

The partial derivative of LL-KLIEP(LS1)

$$\frac{J_{\mathrm{LL-KLIEP(LS1)}}(\boldsymbol{\alpha})}{\boldsymbol{\alpha}} = \frac{\partial^2 J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})}{\partial^2 \boldsymbol{\alpha}} \frac{\partial J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}}$$

## LL-KLIEP(LS) for the high-dimensional data
# LL-KLIEP(LS2)

- Idea: representing the parameter $\alpha$ 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.

  – Now, the computation time is linear w.r.t. the number of parameters, $\alpha$ (quadratic w.r.t. the number of the training inputs, $N_{train}$).

## 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

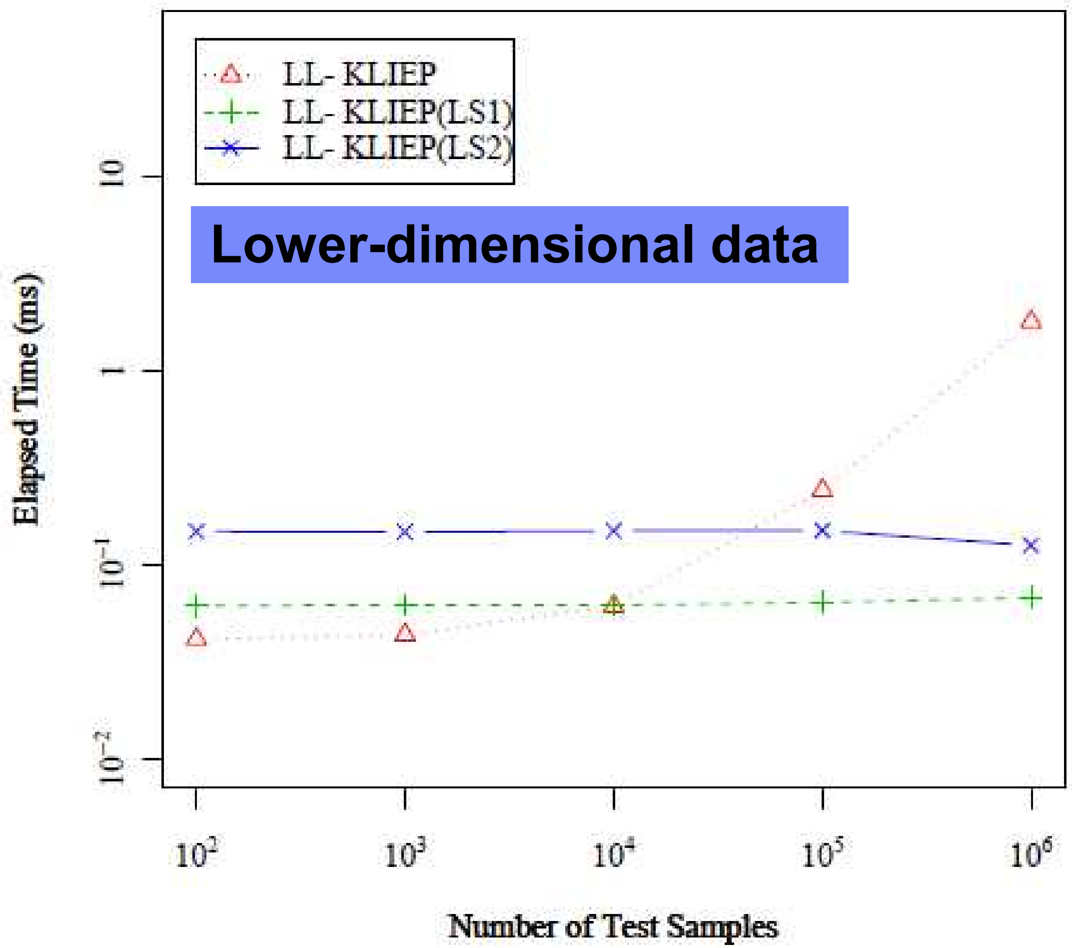
Computational complexity and space requirements. $N_{\text{tr}}$ is the number of training samples, $N_{\text{te}}$ is the number of test samples, $b$ is the number of parameters, and $c$ is the average number of non-zero basis entries.

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

- 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.




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

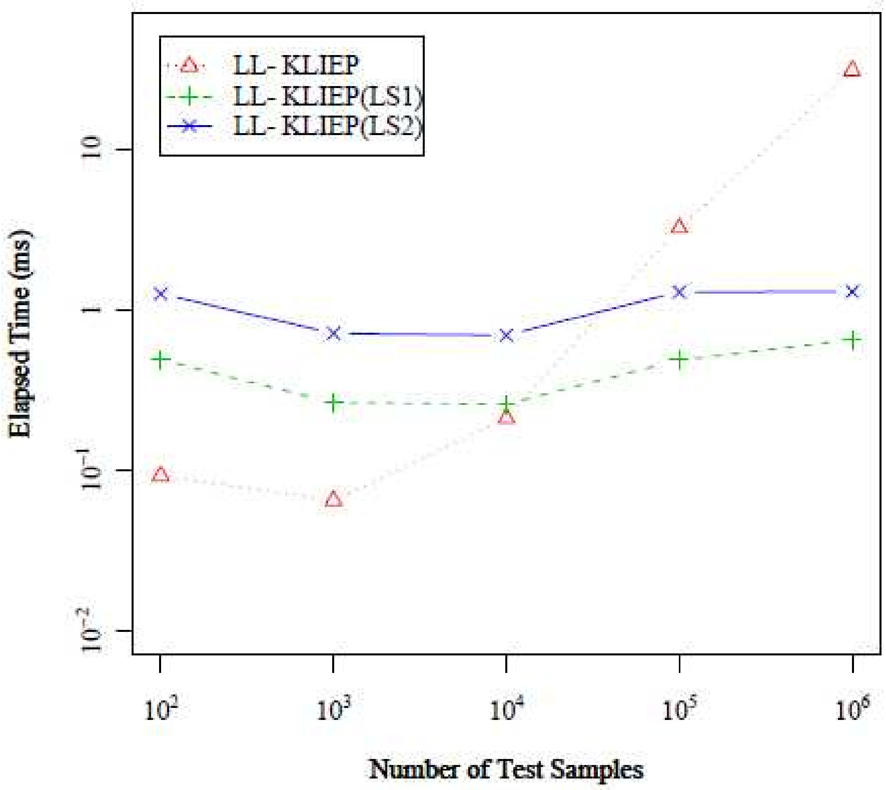
- we used linear basis function so that the number of bases is equivalent to the input dimension.
- d: input dimension = #parameter,
  $N_{tr}$: The number of training inputs,
  $N_{te}$: The number of test inputs

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.

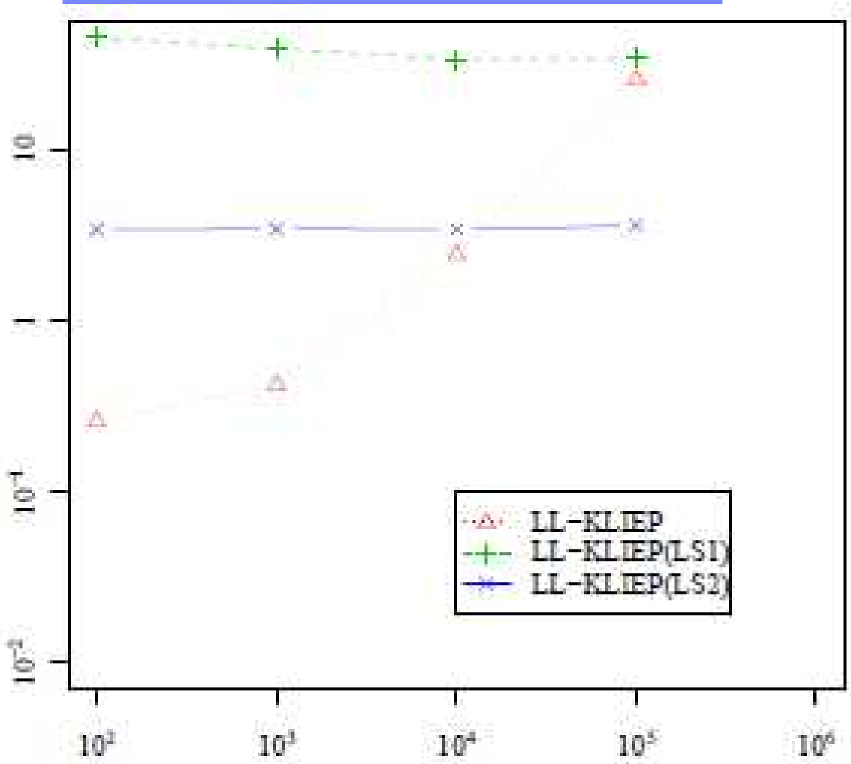
- d: input dimension = #parameter,
  $N_{tr}$: The number of training inputs, $N_{te}$: The number of test inputs

**Moderate-dimensional data**




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

**Higher-dimensional data**




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

## Conclusion

- We proposed a density ratio estimation method called *LL-KLIEP*.
- We also proposed a scalable optimization technique for LL-KLIEP, in which all the test inputs are iterated once.
  - The computation time is nearly independent of the amount of test data
  - The memory requirement is independent of the amount of test data.

# Thank you!

# KLIEP/LL-KLIEP objective functions

- **KLIEP has a log form in the evaluation of test inputs.**

$$J_{\mathrm{KLIEP}}(\boldsymbol{\alpha}) = \frac{1}{N_{\mathrm{test}}} \sum_{\mathbf{x} \in D_{\mathrm{test}}} \log\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\mathbf{x}) \rangle,$$

$$\text{subject to } \frac{1}{N_{\mathrm{train}}} \sum_{\mathbf{x} \in D_{\mathrm{train}}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\mathbf{x}) \rangle = 1, \text{ and } \boldsymbol{\alpha} \geq 0$$

$$\frac{\partial J_{\mathrm{KLIEP}}(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = \frac{1}{N_{\mathrm{test}}} \sum_{\mathbf{x} \in D_{\mathrm{test}}} \frac{\boldsymbol{\psi}(\mathbf{x})}{\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\mathbf{x}) \rangle}$$

- **LL-KLIEP has a linear form in the evaluation of test inputs.**

$$J_{\mathrm{LL-KLIEP}}(\boldsymbol{\alpha}) = \frac{1}{N_{\mathrm{test}}} \sum_{\mathbf{x} \in D_{\mathrm{test}}} \langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\mathbf{x}) \rangle - \log \frac{1}{N_{\mathrm{train}}} \sum_{\mathbf{x} \in D_{\mathrm{train}}} \exp\langle \boldsymbol{\alpha}, \boldsymbol{\psi}(\mathbf{x}) \rangle$$

# 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

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

$$\frac{1}{N_{\text{train}}} w(\mathbf{x})$$

At the optimum,
the mean $\psi(\mathbf{x})$ of test inputs = the mean $w(\mathbf{x})\psi(\mathbf{x})$ of training inputs.

## LL-KLIEP(LS) for the high-dimensional data
# LL-KLIEP(LS2)

- Idea: representing the parameter $\alpha$ 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).

## 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 $\delta$ = -1 to the training input samples and $\delta$ = 1 to the test input samples:

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

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

- The conditional probability p( $\delta$ j**x**) may be learned by discriminating between the test input samples and the training input samples using LR, where $\delta$ plays the role of a class variable.

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

Empirical estimation

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

# Related work:
# Kernel density estimator (KDE)

- Estimating $p_{train}(\mathbf{x})$ and $p_{test}(\mathbf{x})$ separately.
- KDE: non-parametric density estimator

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

- KDE suffers from the curse of dimensionality

# An example of supervised learning under covariate shift
# Importance weighted logistic regression (IWLR)

- Logistic Regression (LR) : binary case

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

- LR classifier $\hat{y} = \operatorname*{argmax}_{y} p_{\boldsymbol{\theta}}(y|\boldsymbol{x})$

- Training LR:
  Density ratio is used as weights in the log-likelihood function

$$\sum_{(\boldsymbol{x},y)\in Z_{tr}} w(\boldsymbol{x}) \left(\log\left(1+\exp\left(y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right) - y f_{\boldsymbol{\theta}}(\boldsymbol{x})\right) + \lambda\|\boldsymbol{\theta}\|^2$$

Training data

Density ratio

Log-likelihood

Regularizer
$\lambda$ : a hyper parameter