Divergence Estimation via Density-Ratio Estimation

Makoto Yamada

myamada@i.Kyoto-u.ac.jp

Research Motivation

- Measuring similarity between data sets is important for various problems.
  - Similarity between documents (ranking).
  - Similarity between access logs (illegal access detection)
- Divergence is useful for measuring (dis)similarity.
  - Kullback-Leibler divergence

Ratio of densities

$$\mathrm{KL} = \int p'(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}$$

  - Mutual information

$$\mathrm{MI} = \iint p(\boldsymbol{x}, \boldsymbol{y}) \log \frac{p(\boldsymbol{x}, \boldsymbol{y})}{p(\boldsymbol{x})p(\boldsymbol{y})} \mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y}$$

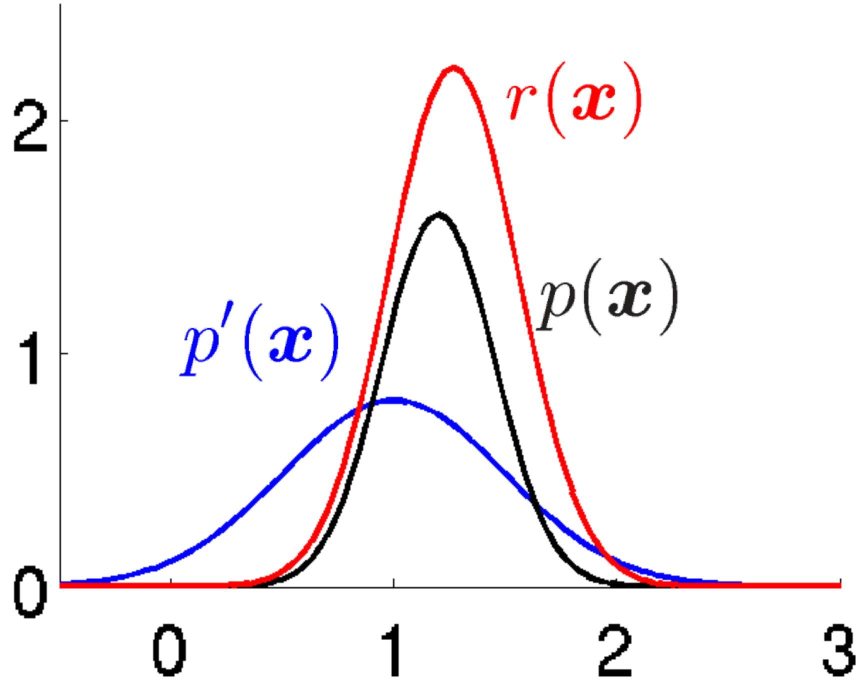
Density Ratio

- The ratio of probability densities:

$$r(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})}$$

$\boldsymbol{x} \in \mathbb{R}^d$

We mainly focus on continuous variable.

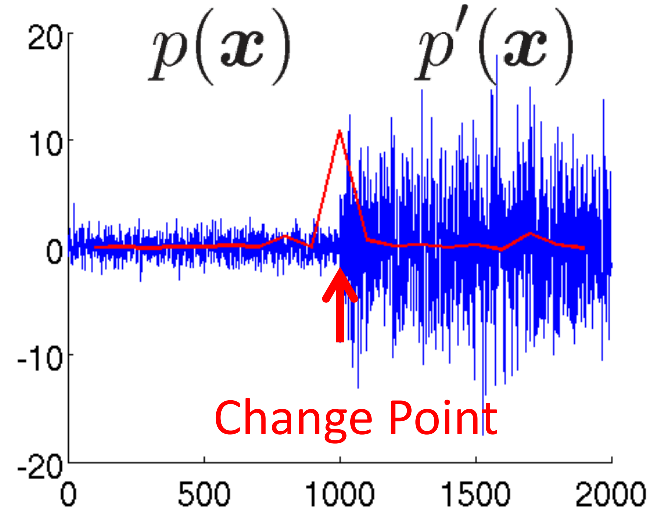


e.g., Kullback-Leibler divergence:

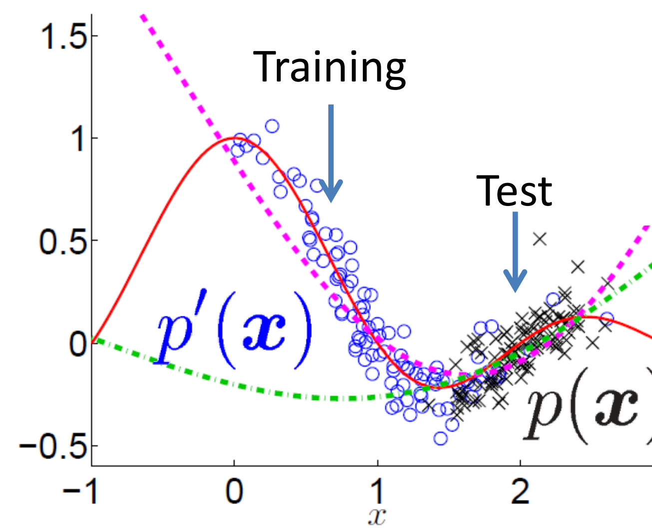
$$\mathrm{KL} = \int p'(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})} \mathrm{d}\boldsymbol{x}$$

Density-Ratio Applications

- Change point detection
- Transfer learning
  - Speaker identification
  - Human pose estimation
  - Action recognition
- Dimensionality reduction
- Outlier detection



$$\frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})}$$



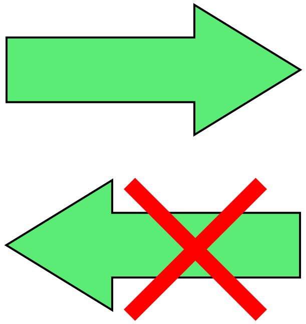
Direct Density-Ratio Estimation

Sugiyama, Suzuki, & Kanamori (2012)

- Naïve approach: Separately estimate $p(\boldsymbol{x}), p'(\boldsymbol{x})$ and take their ratio ⇒ poor
- Vapnik said: When solving a problem of interest, one should not solve a more general problems as an intermediate step (Vapnik principle)

Knowing densities

$$p(\boldsymbol{x}), p'(\boldsymbol{x})$$



Knowing ratio

$$r(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})}$$

⇒Estimating densities is more general than estimating a density ratio

- Following the Vapnik principle, methods which directly estimate the density ratio without density estimation were proposed.

Density-Ratio after 2016

- We developed several density-ratio based approaches (mostly kernel based approaches) by 2012.
- After 2016
  - Generative Adversarial Networks (GAN)
    - Generative Adversarial Nets from a Density Ratio Estimation Perspective (arXiv)
    - Learning in implicit generative models (arXiv)
  - Approximate Bayesian Computation (ABC)
    - Likelihood-free inference by ratio estimation
  - Mutual Information estimation
    - MINE: Mutual Information Neural Estimation

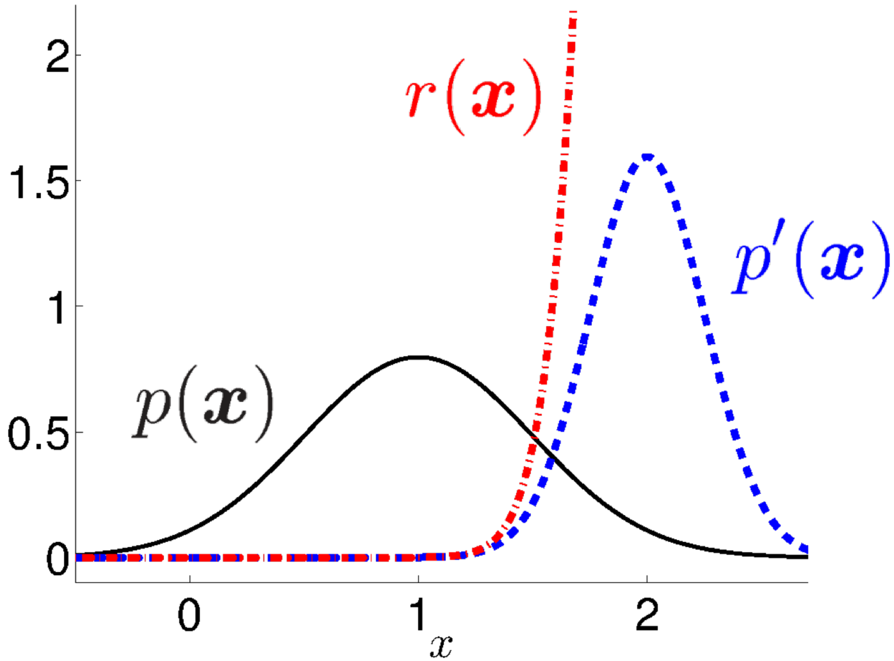
Research Motivation

- Density ratio

$$r(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})}$$

can diverge to infinity under a rather simple setting.

Cortes et al. (NIPS 2010)



$$p(\boldsymbol{x}) = N(1, 0.5^2)$$
$$p'(\boldsymbol{x}) = N(2, 0.25^2)$$

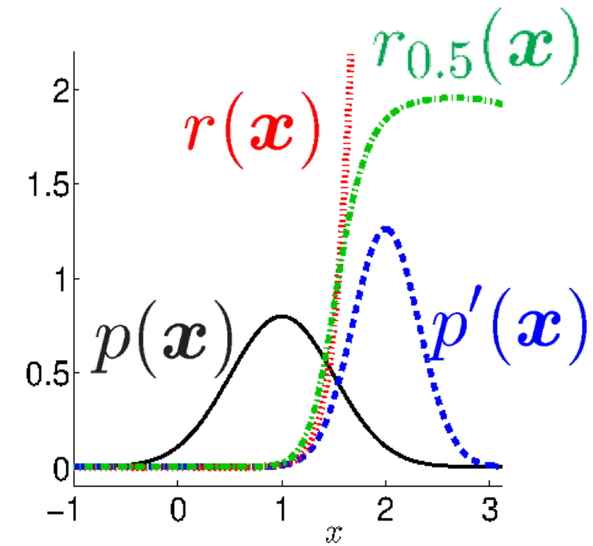
Relative Density-Ratio

Yamada et al. (NIPS 2011)

- Relative density-ratio:

$$r_\alpha(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{\alpha p(\boldsymbol{x}) + (1-\alpha)p'(\boldsymbol{x})}$$

$$0 \leq \alpha \leq 1$$



Relative density-ratio is bounded above by $1/\alpha$!

- Relative Pearson divergence:

$$\mathrm{PE}_\alpha = \frac{1}{2}\int (r_\alpha(\boldsymbol{x}) - 1)^2 q_\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$

$$q_\alpha(\boldsymbol{x}) = \alpha p(\boldsymbol{x}) + (1-\alpha)p'(\boldsymbol{x})$$

$$\mathrm{PE}_\alpha = 0 \iff p(\boldsymbol{x}) = p'(\boldsymbol{x})$$

Relative unconstrained Least-Squares Importance Fitting (RuLSIF)

- Data: $\{\boldsymbol{x}_i\}_{i=1}^{n} \overset{i.i.d.}{\sim} p(\boldsymbol{x}) \quad \{\boldsymbol{x}'_i\}_{i=1}^{n'} \overset{i.i.d.}{\sim} p'(\boldsymbol{x})$
- Key idea: Fit a density-ratio model $r(\boldsymbol{x};\boldsymbol{\theta})$ to the true density-ratio $r_\alpha(\boldsymbol{x})$ under squared-loss.

$$q_\alpha(\boldsymbol{x}) = \alpha p(\boldsymbol{x}) + (1-\alpha)p'(\boldsymbol{x})$$

$$J_0(\boldsymbol{\theta}) = \frac{1}{2}\int \left(r(\boldsymbol{x};\boldsymbol{\theta}) - r_\alpha(\boldsymbol{x})\right)^2 q_\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$

$$= \frac{1}{2}\int r^2(\boldsymbol{x};\boldsymbol{\theta})q_\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \int r(\boldsymbol{x};\boldsymbol{\theta})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + C$$

Constant

$$\widehat{J}(\boldsymbol{\theta}) = \frac{\alpha}{2n}\sum_{i=1}^{n} r^2(\boldsymbol{x}_i;\boldsymbol{\theta}) + \frac{(1-\alpha)}{2n'}\sum_{i=1}^{n'} r^2(\boldsymbol{x}'_i;\boldsymbol{\theta}) - \frac{1}{n}\sum_{i=1}^{n} r(\boldsymbol{x}_i;\boldsymbol{\theta})$$

RuLSIF: Model

- Kernel model:

$$r(\boldsymbol{x};\boldsymbol{\theta}) = \sum_{\ell=1}^{n} \theta_\ell K(\boldsymbol{x},\boldsymbol{x}_\ell) = \boldsymbol{\theta}^\top \boldsymbol{k}(\boldsymbol{x})$$

$$K(\boldsymbol{x},\boldsymbol{x}') = \exp\left(-\frac{\|\,\boldsymbol{x}-\boldsymbol{x}'\,\|^2}{2\sigma^2}\right) \quad \sigma^2 > 0$$

$$\boldsymbol{\theta} = [\theta_1,\ldots,\theta_n]^\top \;\; \boldsymbol{k}(\boldsymbol{x}) = [K(\boldsymbol{x},\boldsymbol{x}_1),\ldots,K(\boldsymbol{x},\boldsymbol{x}_n)]^\top$$

- Cost function with kernel model:

$$\widehat{J}(\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{\theta}^\top \widehat{\boldsymbol{H}}\boldsymbol{\theta} - \boldsymbol{\theta}^\top \widehat{\boldsymbol{h}}$$

$$\widehat{\boldsymbol{H}} = \frac{\alpha}{n}\sum_{i=1}^{n} \boldsymbol{k}(\boldsymbol{x}_i)\boldsymbol{k}(\boldsymbol{x}_i)^\top + \frac{1-\alpha}{n'}\sum_{i=1}^{n'} \boldsymbol{k}(\boldsymbol{x}'_i)\boldsymbol{k}(\boldsymbol{x}'_i)^\top \qquad \widehat{\boldsymbol{h}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{k}(\boldsymbol{x}_i)$$

RuLSIF: Solution

- Optimization problem:

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^n}{\mathrm{argmin}} \left[\frac{1}{2} \boldsymbol{\theta}^\top \widehat{\boldsymbol{H}} \boldsymbol{\theta} - \widehat{\boldsymbol{h}}^\top \boldsymbol{\theta} + \boxed{\frac{\lambda}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta}} \right]$$

Regularizer $\lambda > 0$

- Solution (analytically obtained):

$$\widehat{\boldsymbol{\theta}} = (\widehat{\boldsymbol{H}} + \lambda \boldsymbol{I}_n)^{-1} \widehat{\boldsymbol{h}}$$

$$\widehat{r}_\alpha(\boldsymbol{x}) = \widehat{\boldsymbol{\theta}}^\top \boldsymbol{k}(\boldsymbol{x})$$

  - Cross-validation is possible.
  - If $\alpha = 0$, RuLSIF is equivalent to uLSIF. Kanamori et al. (JMLR 2009)

$$r_0(\boldsymbol{x}) = \frac{p(\boldsymbol{x})}{p'(\boldsymbol{x})}$$

Relative PE Divergence Estimators

- Relative PE divergence:

$$\mathrm{PE}_\alpha = \frac{1}{2}\int (r_\alpha(\boldsymbol{x}) - 1)^2 q_\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$

$$= -\frac{1}{2}\int r_\alpha^2(\boldsymbol{x})q_\alpha(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + \int r_\alpha(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \frac{1}{2} \quad \text{(A)}$$

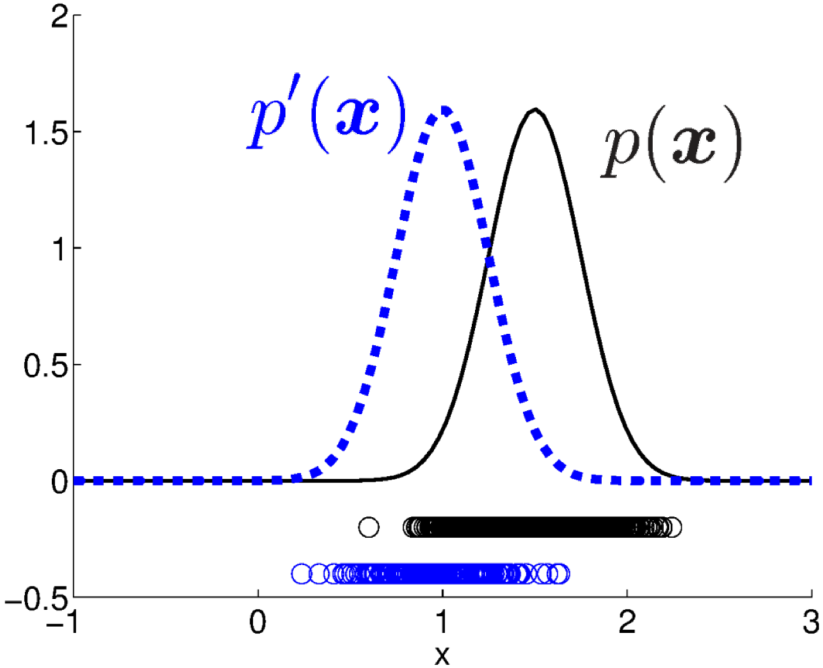
$$= \frac{1}{2}\int r_\alpha(\boldsymbol{x})p(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - \frac{1}{2} \quad \text{(B)}$$

- Relative PE divergence estimators:

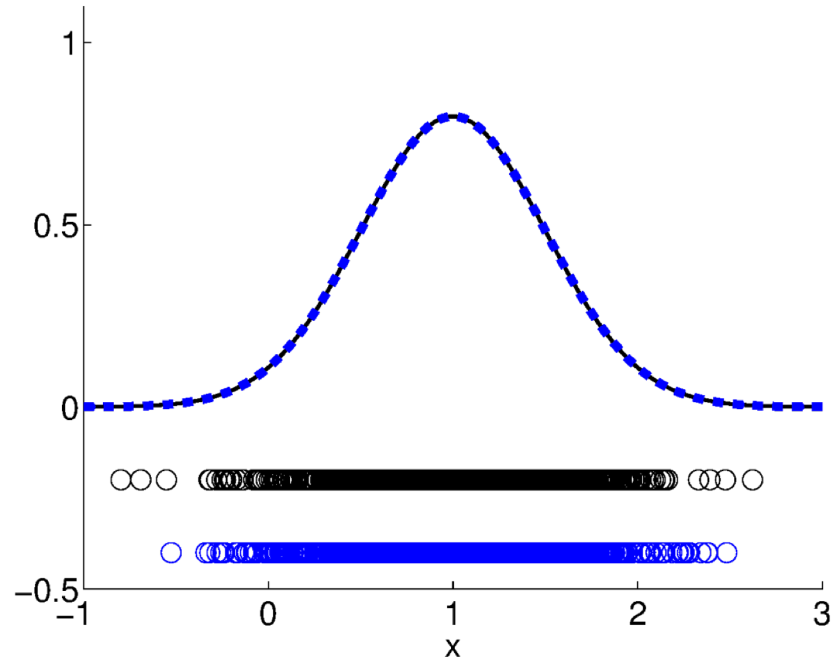
(A) $$\widehat{\mathrm{PE}}_\alpha := -\frac{\alpha}{2n}\sum_{i=1}^{n}\widehat{r}_\alpha(\boldsymbol{x}_i)^2 - \frac{(1-\alpha)}{2n'}\sum_{i=1}^{n'}\widehat{r}_\alpha(\boldsymbol{x}'_i)^2 + \frac{1}{n}\sum_{i=1}^{n}\widehat{r}_\alpha(\boldsymbol{x}_i) - \frac{1}{2}$$

(B) $$\widetilde{\mathrm{PE}}_\alpha := \frac{1}{2n}\sum_{i=1}^{n}\widehat{r}_\alpha(\boldsymbol{x}_i) - \frac{1}{2}$$

Toy Experiments: RuLSIF



$\widehat{\mathrm{PE}}_{0.5}$: 0.2865



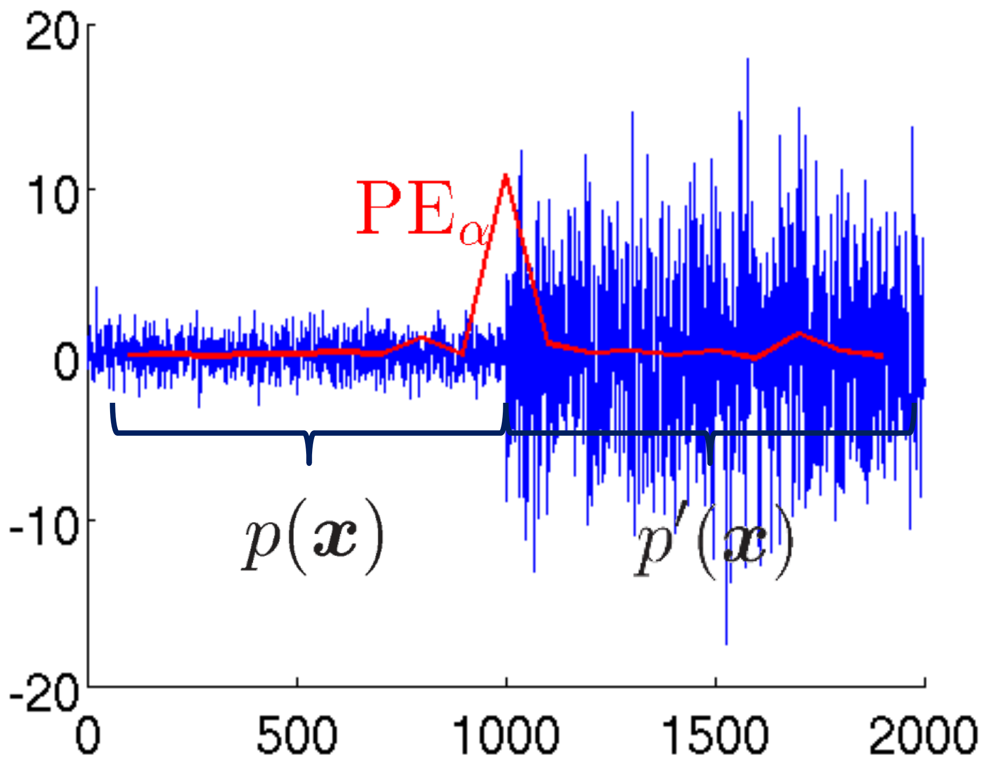
$\widehat{\mathrm{PE}}_{0.5}$:-0.0001

Change Point Detection

Liu, Yamada, Collier & Sugiyama (Neural Networks to appear)

- Change-point detection based on PE:

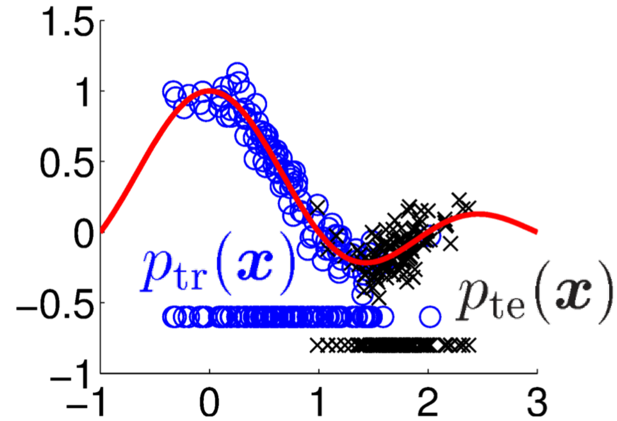
$$\begin{cases} \mathrm{PE}_\alpha < \tau & \text{(No abrupt change)} \\ \mathrm{PE}_\alpha \geq \tau & \text{(Abrupt change)} \end{cases}$$



Covariate Shift Adaptation (Transfer Learning)

Shimodaira (JSPI, 2000)

- Key idea: Reduce generalization error in test data set (Not in training dataset)!



$p_{\mathrm{tr}}(\boldsymbol{x}) \neq p_{\mathrm{te}}(\boldsymbol{x})$

- Covariate shift adaptation setup
  - Training data: $\{(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{i.i.d.}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})$
  - Test data: $\{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n_{\mathrm{te}}} \overset{i.i.d.}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$
  - Key idea: Learning a function so that error in test data is minimized under the assumption: $p_{\mathrm{tr}}(\boldsymbol{y}|\boldsymbol{x}) = p_{\mathrm{te}}(\boldsymbol{y}|\boldsymbol{x})$

$$J(\boldsymbol{w}) = \iint \mathrm{loss}(\boldsymbol{y}, \boldsymbol{f}_{\boldsymbol{w}}(\boldsymbol{x}))p_{\mathrm{te}}(\boldsymbol{x}, \boldsymbol{y})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y}$$

$$= \iint \mathrm{loss}(\boldsymbol{y}, \boldsymbol{f}_{\boldsymbol{w}}(\boldsymbol{x}))\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})\mathrm{d}\boldsymbol{x}\mathrm{d}\boldsymbol{y}$$

Exponentially-flattened IW (EIW) empirical error minimization

Shimodaira (JSPI 2000)

- Flatten the importance weight by $0 \leq \tau \leq 1$

$$\min_{f \in \mathcal{F}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\tau} \mathrm{loss}(y_i^{\mathrm{tr}}, f(\boldsymbol{x}_i^{\mathrm{tr}})) \right]$$

$\tau = 0$ → empirical error minimization.

$0 < \tau < 1$ → Intermediate

$\tau = 1$ → IW empirical error minimization

Setting τ to $0 < \tau < 1$ is practically useful for stabilizing the covariate shift adaptation, even though it cannot give an unbiased model under covariate shift.

It still needs importance weight estimation ☹

Relative importance-weighted (RIW) empirical error minimization

Yamada et al. (NIPS 2011)

- Use relative importance weight (RIW):

$$r_\alpha(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{(1-\alpha)p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}}) + \alpha p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} < \frac{1}{1-\alpha} \Longleftrightarrow \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})}\right)^{\tau}$$

  $0 \leq \alpha \leq 1$ controls the adaptiveness to the test distribution.

- RIW-empirical error minimization:

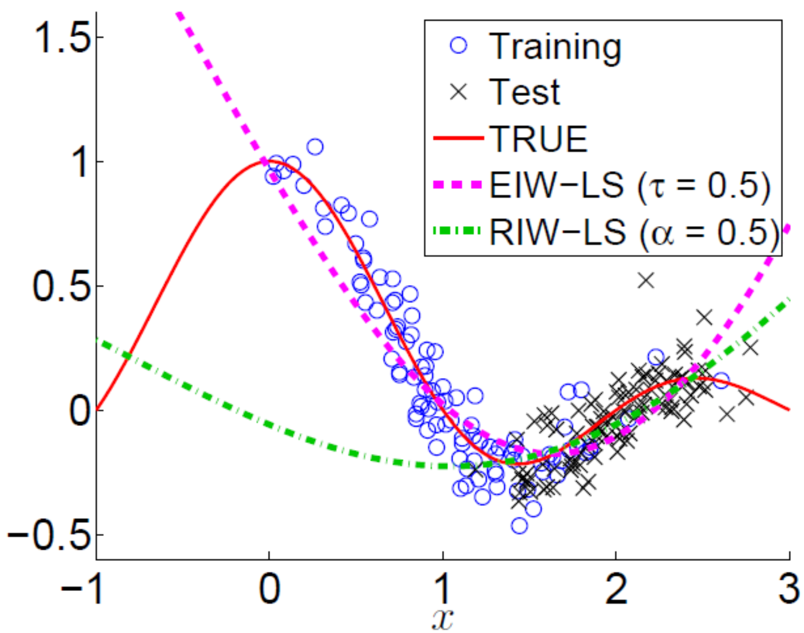
$$\min_{f \in \mathcal{F}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{(1-\alpha)p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}}) + \alpha p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right) \mathrm{loss}(y_i^{\mathrm{tr}}, f(\boldsymbol{x}_i^{\mathrm{tr}})) \right]$$

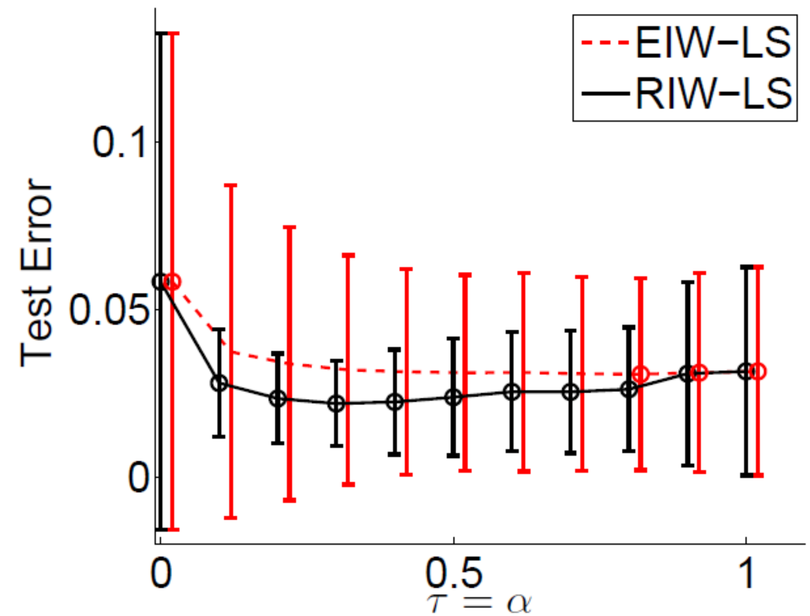
  $\alpha = 0.5$ works well in practice.

Toy Example

Yamada et al. (NIPS 2011)

- Predicted output by IWKR (IWKR = RIW-LS)





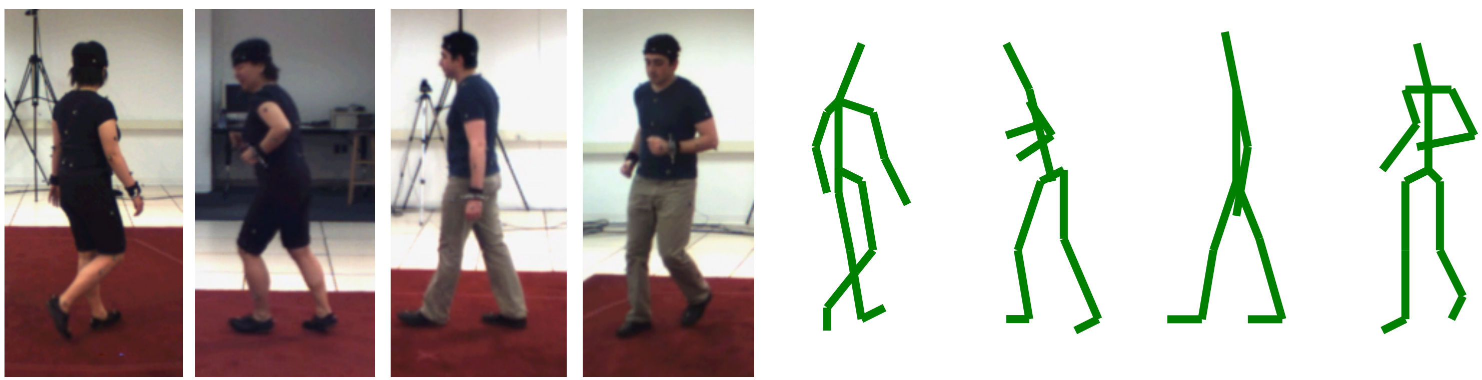
RIW method gives smaller error and variance☺

Application1: 3D Pose Estimation

Yamada, Sigal, & Raptis (ECCV 2012)

- **Given:** large database of image-pose pairs

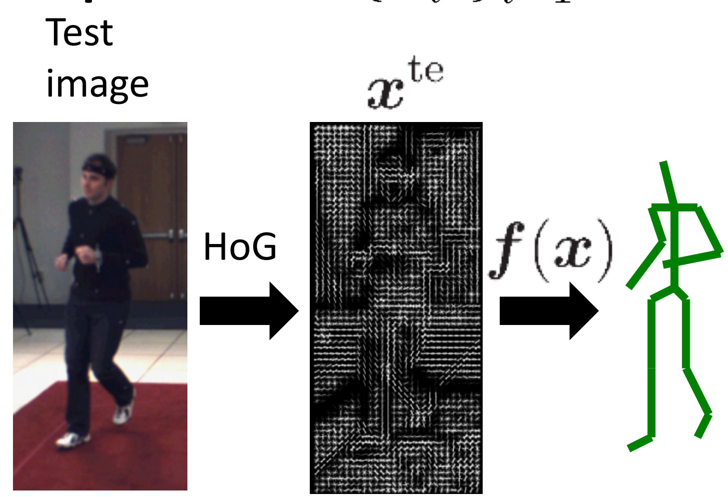
$$\{(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{y}_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \stackrel{i.i.d.}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, \boldsymbol{y})$$



- **Inference:** Predict human pose of $\{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n_{\mathrm{te}}} \stackrel{i.i.d.}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$

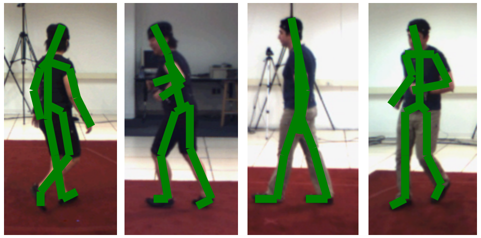
$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{e}$$

e.g., Kernel Regression



HumanEva-I Experiments: Settings

Sigal et al. (TR 2006)



- Experimental Settings:
  - **Selection bias (C1-3):** All camera data is used for training and testing.
  - **Subject transfer (C1):** Test subject is not included in training phase.
- Error metric:

$$Error_{pose}(\widehat{\boldsymbol{y}}, \boldsymbol{y}^*) = \frac{1}{20}\sum_{m=1}^{20} \|\widehat{\boldsymbol{y}}^{(m)} - \boldsymbol{y}^{*(m)}\|$$

$\boldsymbol{y}^* \in \mathbb{R}^{60}$: True pose

Application2: Human Activity Recognition

Yamada et al.(NIPS 2011)

- Human Activity Recognition by accelerometer
  - Walk, run, bicycle riding, and train riding classification by accelerometer sensor in iPod touch
  - Training: 20 subjects data set
  - Test: A new subject not included in the training set

Task	KLR ($\alpha = 0,\ \tau = 0$)	RIW-KLR ($\alpha = 0.5$)	EIW-KLR ($\tau = 0.5$)	IW-KLR ($\alpha = 1,\ \tau = 1$)
Walks vs. run	0.803 (0.082)	**0.889(0.035)**	**0.882(0.039)**	**0.882 (0.035)**
Walks vs. bicycle	0.880 (0.025)	**0.892(0.035)**	0.867 (0.054)	0.854 (0.070)
Walks vs. train	0.985 (0.017)	**0.992(0.008)**	0.989 (0.011)	0.983 (0.021)

Relative importance weight performs well☺

Conclusion

- Relative Density-Ratio

  Relative Density-ratio is promising for various types of applications.

  - Change-point detection
  - Transfer learning