Deep Learning: Theory, Applications, and Implications (DL2024 Tokyo) Mar. 18, 2024

Importance-Weighting Approach to Distribution Shift Adaptation

Masashi Sugiyama

RIKEN Center for Advanced Intelligence Project/
The University of Tokyo, Japan

http://www.ms.k.u-tokyo.ac.jp/sugi/

Learning under Distribution Shifts

Given:

- Training data $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$

$\boldsymbol{x}$: Input

y : Output

Goal:

- Learn predictor $y = f(\boldsymbol{x})$ minimizing the test risk (with some additional data from the test domain).

$$\min_f R(f) \qquad R(f) = \mathbb{E}_{p_{\mathrm{te}}(\boldsymbol{x}, y)}[\ell(f(\boldsymbol{x}), y)]$$

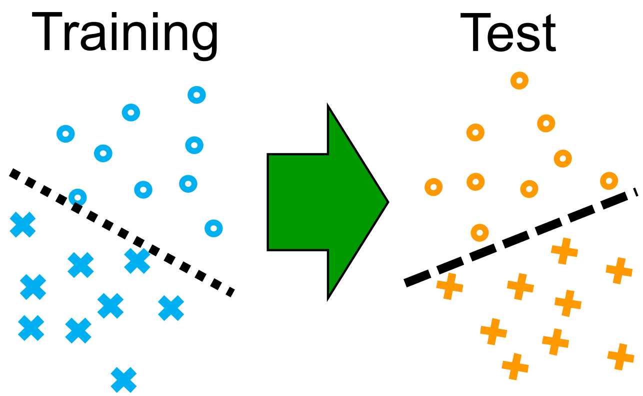
ℓ : loss

Challenge:

- Overcome changing distributions!

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

  - Non-stationary of the environments.
  - Sample selection bias due to privacy concerns.



NIPS Workshop on Learning when Test and Training Inputs Have Different Distributions, Whistler 2006

Workshop

Learning when test and training inputs have different distributions

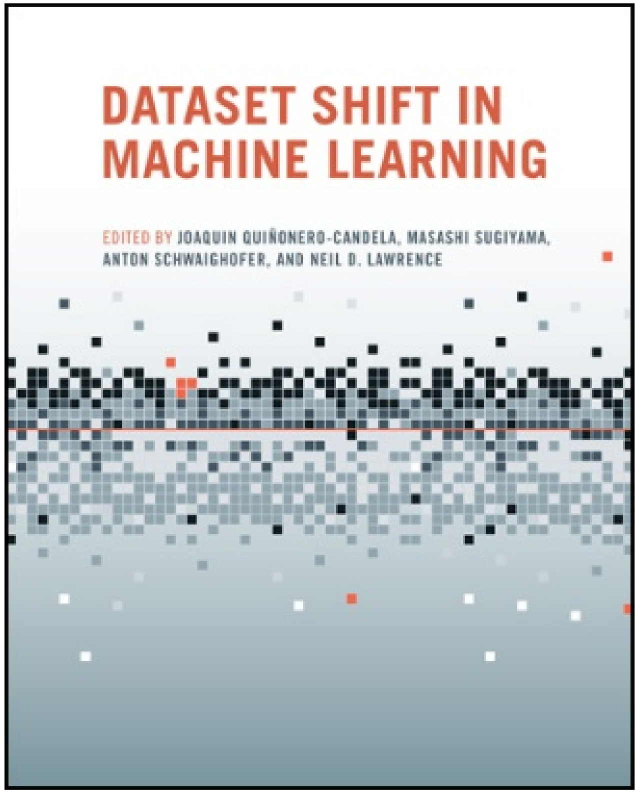
Joaquin Quiñonero Candela · Masashi Sugiyama · Anton Schwaighofer · Neil D Lawrence

Sat Dec 09 05:00 PM -- 05:00 PM (JST) @ Nordic

Event URL: http://ida.first.fraunhofer.de/projects/different06/ »

Many machine learning algorithms assume that the training and the test data are drawn from the same distribution. Indeed many of the proofs of statistical consistency, etc., rely on this assumption. However, in practice we are very often faced with the situation where the training and the test data both follow the same conditional distribution, p(y|x), but the input distributions, p(x), differ. For example, principles of experimental design dictate that training data is acquired in a specific manner that bears little resemblance to the way the test inputs may later be generated. The aim of this workshop will be to try and shed light on the kind of situations where explicitly addressing the difference in the input distributions is beneficial, and on what the most sensible ways of doing this are.

Quiñonero-Candela, Sugiyama, Schwaighofer & Lawrence (Eds.), Dataset Shift in Machine Learning, MIT Press, 2009.



- "Distribution Shift" workshops at NeurIPS2021, 2022, and 2023.

NeurIPS 2023 Workshop on Distribution Shifts (DistShift)

New Frontiers with Foundation Models

December 2023, New Orleans, USA

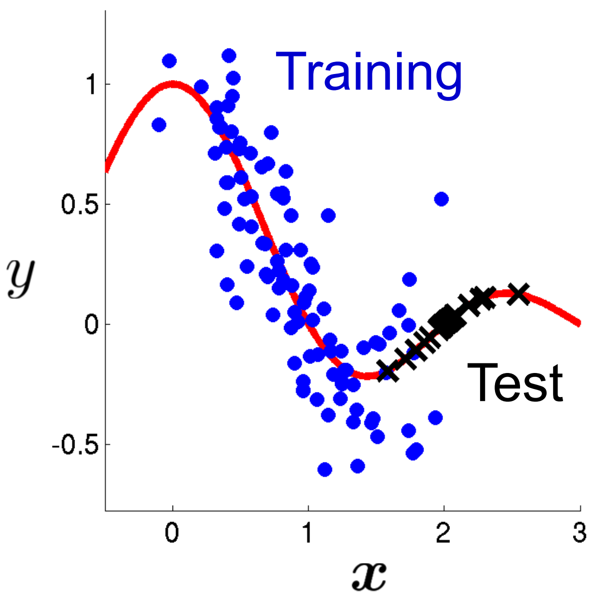
New Orleans Convention Center

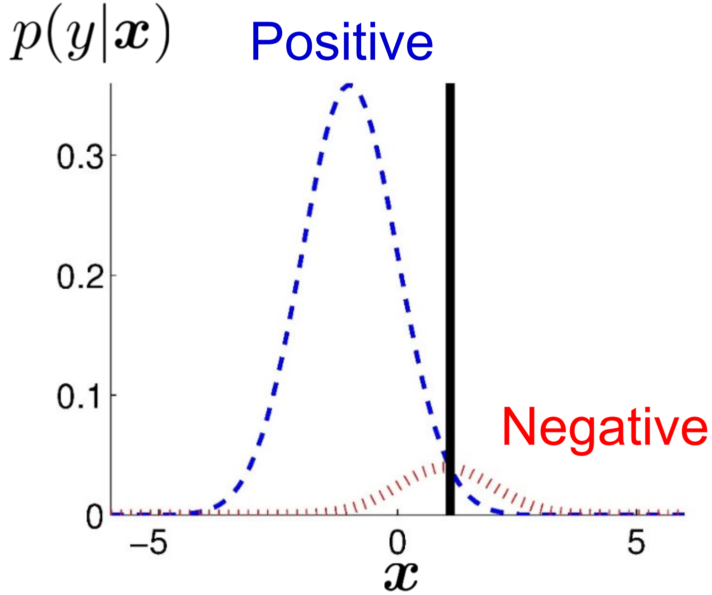
Contents

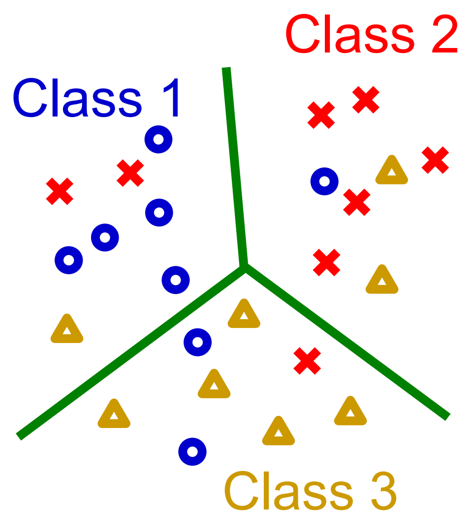
Types of Distribution Shifts

$\boldsymbol{x}$: Input y : Output

- Joint shift: $p_{\mathrm{tr}}(\boldsymbol{x}, y) \neq p_{\mathrm{te}}(\boldsymbol{x}, y)$
- Covariate shift: $p_{\mathrm{tr}}(\boldsymbol{x}) \neq p_{\mathrm{te}}(\boldsymbol{x})$
- Label shift: $p_{\mathrm{tr}}(y) \neq p_{\mathrm{te}}(y)$
- Output noise: $p_{\mathrm{tr}}(y|\boldsymbol{x}) \neq p_{\mathrm{te}}(y|\boldsymbol{x})$
- Class-conditional shift: $p_{\mathrm{tr}}(\boldsymbol{x}|y) \neq p_{\mathrm{te}}(\boldsymbol{x}|y)$







Covariate Shift Adaptation

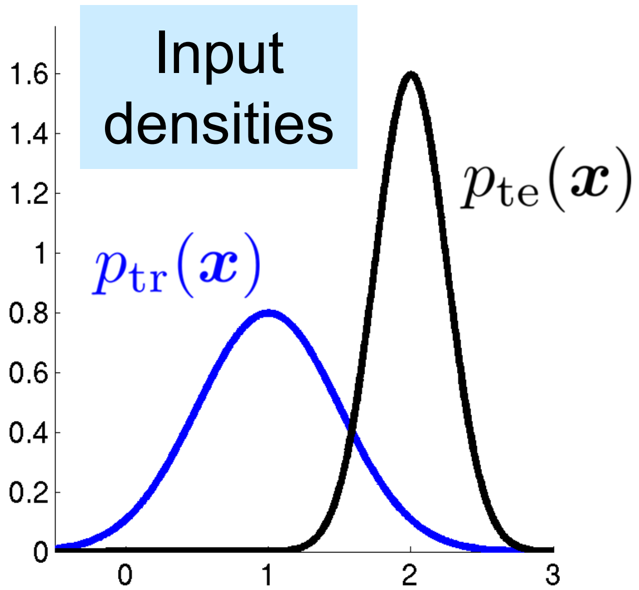
Shimodaira (JSPI2000)

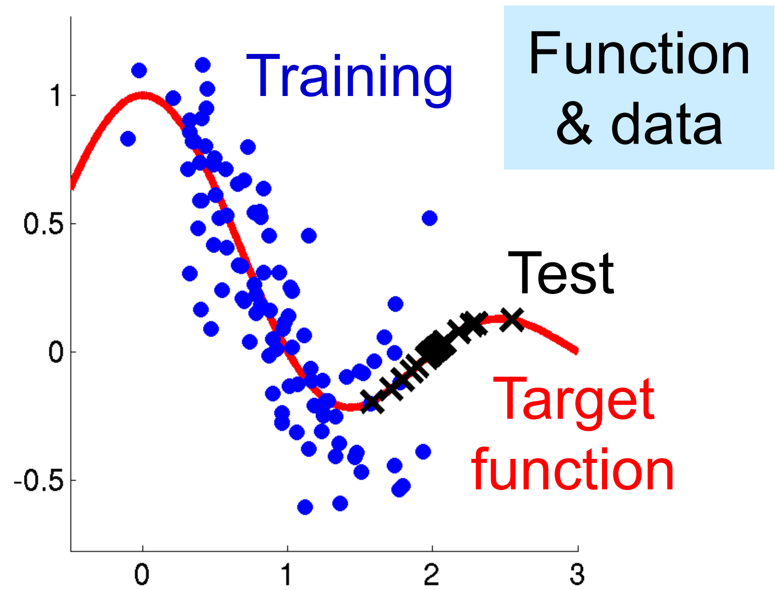
- Training and test input distributions are different,

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

but the output-given-input distribution is unchanged:

$$p_{\mathrm{tr}}(y|\boldsymbol{x}) = p_{\mathrm{te}}(y|\boldsymbol{x}) = p(y|\boldsymbol{x})$$





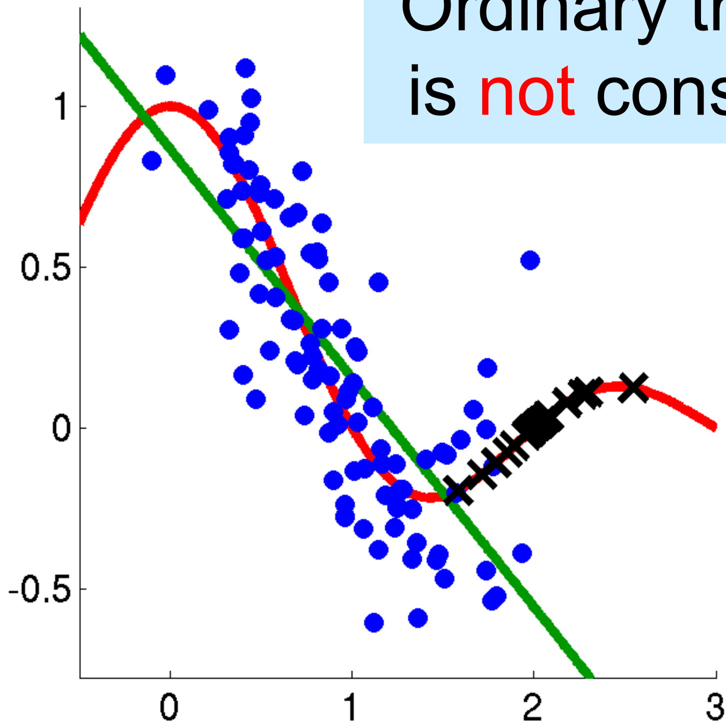
- Given:
  - Labeled training data: $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
  - Unlabeled test data: $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$

Importance-Weighted Training

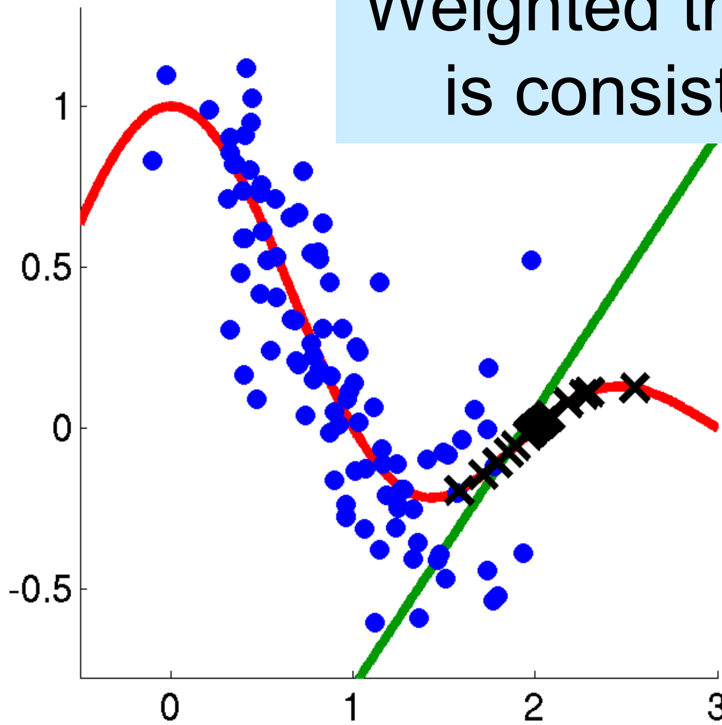
$$\underset{f}{\operatorname{argmin}} \left[\sum_{i=1}^{n_{\mathrm{tr}}} \ell(f(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}}) \right]$$

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

$$\underset{f}{\operatorname{argmin}} \left[\sum_{i=1}^{n_{\mathrm{tr}}} \underbrace{\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})}}_{\text{Importance}} \ell(f(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}}) \right]$$



Ordinary training is not consistent



Weighted training is consistent

- How do we estimate the importance?

Direct Importance Estimation

- **Given**: training and test input data

$$\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}) \qquad \{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$$

- **Goal:** estimate the density ratio $\dfrac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$

- **Kernel mean matching:** Huang+ (NeurIPS2006)
  - Match the means of $p_{\mathrm{te}}(\boldsymbol{x})$ and $r(\boldsymbol{x})p_{\mathrm{tr}}(\boldsymbol{x})$ in RKHS $\mathcal{H}$.

$$\min_{r\in\mathcal{H}} \left\| \mathbb{E}_{p_{\mathrm{te}}(\boldsymbol{x})}[K(\boldsymbol{x},\cdot)] - \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x})}[K(\boldsymbol{x},\cdot)r(\boldsymbol{x})] \right\|_{\mathcal{H}}^2$$

$K(\cdot,\cdot)$: Characteristic kernel

- **Least-squares importance fitting (LSIF)**:
  - Fit a model $r(\boldsymbol{x})$ to $\dfrac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ by least squares: Kanamori, Hido & Sugiyama (NeurIPS2008)

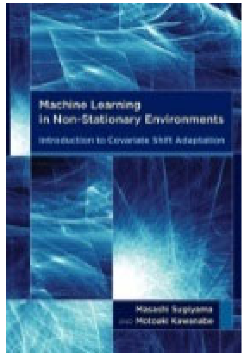
$$\operatorname*{argmin}_r \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x})}\left[\left(r(\boldsymbol{x}) - \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}\right)^2\right] = \operatorname*{argmin}_r \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x})}[r(\boldsymbol{x})^2] - 2\mathbb{E}_{p_{\mathrm{te}}(\boldsymbol{x})}[r(\boldsymbol{x})]$$

  - Extendable to Bregman divergences: Sugiyama, Suzuki & Kanamori (AISM2012)

$$\operatorname*{argmin}_r \mathrm{Breg}_\psi\left(\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} \middle\| r\right) = \operatorname*{argmin}_r \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x})}[\partial\psi(r(\boldsymbol{x}))r(x) - \psi(r(\boldsymbol{x}))] - \mathbb{E}_{p_{\mathrm{te}}(\boldsymbol{x})}[\partial\psi(r(\boldsymbol{x}))]$$

Joint Importance-Predictor Estimation

Sugiyama & Kawanabe, Machine Learning in Non-Stationary Environments, MIT Press, 2012



- Classical approaches are two steps:
  1. Importance weight estimation (e.g., LSIF):

$$\widehat{r} = \operatorname*{argmin}_{r} \widehat{J}_1(r) \qquad J_1(r) = \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x})}\left[\left(r(\boldsymbol{x}) - \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}\right)^2\right]$$

  2. Importance-weighted predictor training:

$$\widehat{f} = \operatorname*{argmin}_{f} \widehat{J}_2^{\ell}(f, \widehat{r}) \qquad J_2^{\ell}(f, r) = \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x}, y)}\left[r(\boldsymbol{x})\ell(f(\boldsymbol{x}), y)\right]$$

- For $\ell_{\mathrm{te}} \leq 1, \ell_{\mathrm{tr}} \geq \ell_{\mathrm{te}}, r \geq 0$, the test risk is bounded as

$$\frac{1}{2} R_{\ell_{\mathrm{te}}}(f)^2 \leq J_{\ell_{\mathrm{tr}}}(f, r) \qquad J_{\ell}(f, r) = J_1(r) + J_2^{\ell}(f, r)$$

$$R_{\ell}(f) = \mathbb{E}_{p_{\mathrm{te}}(\boldsymbol{x}, y)}\left[\ell(f(\boldsymbol{x}), y)\right]$$

Zhang, Yamane, Lu & Sugiyama (ACML2020, SNCS2021)

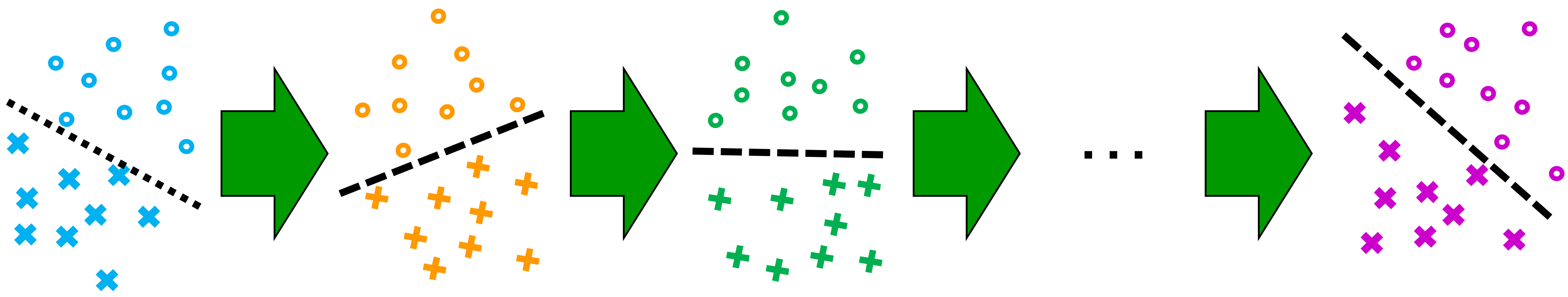
  - Joint upper-bound minimization: $\widehat{f} = \operatorname*{argmin}_{f} \min_{r \geq 0} \widehat{J}_{\ell_{\mathrm{tr}}}(f, r)$

Contents

Continuous Distribution Shifts

- So far, we focused on a fixed test domain:
  - We trained a predictor to match the test domain.
- However, test domains can change over time.

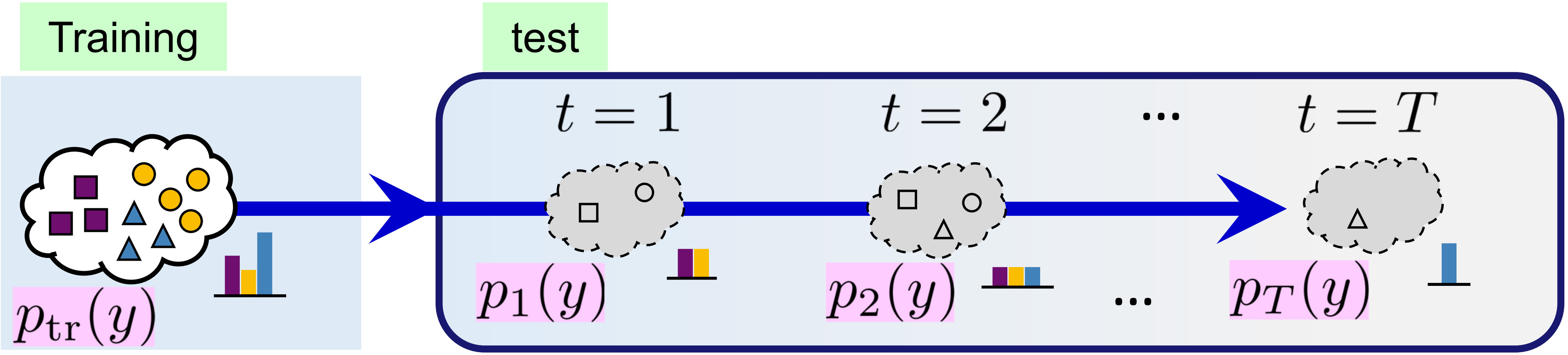
Training	Test 1	Test 2	…	Test T
$p_{\mathrm{tr}}(\boldsymbol{x}, y)$	$p_1(\boldsymbol{x}, y)$	$p_2(\boldsymbol{x}, y)$		$p_T(\boldsymbol{x}, y)$



- Goal: Obtain predictor $\widehat{f}_t$ that works well for $p_t(\boldsymbol{x}, y)$.

$$R_t(f) = \mathbb{E}_{p_t(\boldsymbol{x},y)}[\ell(f(\boldsymbol{x}), y)] \quad t = 1, \ldots, T$$

Continuous Label Shift



- Class-priors $p_t(y)$ change arbitrarily over time, but class-conditionals stay unchanged:

$$p_t(\boldsymbol{x}|y) = p_{\mathrm{tr}}(\boldsymbol{x}|y) \quad t = 1, \ldots, T$$

- Given:
  - (Large) labeled training data: $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
  - (Small) unlabeled test data: $\{\boldsymbol{x}_i^{(t)}\}_{i=1}^{n_t} \overset{\mathrm{i.i.d.}}{\sim} p_t(\boldsymbol{x})$

$$t = 1, \ldots, T$$

ATLAS (Adapting To LAbel Shift)

Bai, Zhang, Zhao, Sugiyama & Zhou (NeurIPS2022)

$$\min_f \sum_{i=1}^{n_{\mathrm{tr}}} \frac{\widehat{p}_t(y_i^{\mathrm{tr}})}{\widehat{p}_{\mathrm{tr}}(y_i^{\mathrm{tr}})} \ell(f(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}})$$

- Batch importance weighing requires retraining in each time step.
- Can we make it computationally more efficient?
  - Online learning!
- We use online convex optimization, assuming (Hazan (2016))
  - convex loss ℓ (e.g., logistic),
  - linear model $f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}, \quad \boldsymbol{w} \in \mathcal{W}$.

$$\boldsymbol{w}_t = \Pi_{\mathcal{W}} \left[\boldsymbol{w}_{t-1} - \eta \nabla \widehat{R}_{t-1}(\boldsymbol{w}_{t-1}) \right]$$

$\Pi_{\mathcal{W}}$: projection

$\eta > 0$: step size

- We use black box shift estimation for class priors.

Lipton+ (ICML2018)

Choice of Step Size η

$$\boldsymbol{w}_t = \Pi_{\mathcal{W}}\left[\boldsymbol{w}_{t-1} - \eta \nabla \widehat{R}_{t-1}(\boldsymbol{w}_{t-1})\right]$$

- If the speed of distribution shift is
  - slow, η should be small to keep the previous classifier.
  - fast, η should be large to quickly update the classifier.
- How do we choose η in practice?
  - Ensemble learning! Zhao+ (NeurIPS2020)
- For $0 < \eta_1 < \cdots < \eta_M$, we run M learners:

$$\boldsymbol{w}_t^{(m)} = \Pi_{\mathcal{W}}\left[\boldsymbol{w}_{t-1}^{(m)} - \eta_m \nabla \widehat{R}_{t-1}(\boldsymbol{w}_{t-1}^{(m)})\right] \qquad \widehat{R}_t(\boldsymbol{w}) = \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \frac{\widehat{p}_t(y_i^{\mathrm{tr}})}{\widehat{p}_{\mathrm{tr}}(y_i^{\mathrm{tr}})} \ell(\boldsymbol{w}^\top \boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})$$

- Final output is the weighted average (cf. Hedge):

Freund+ (JCSS1997)

$$\boldsymbol{w}_t = \sum_{m=1}^{M} p_t^{(m)} \boldsymbol{w}_t^{(m)} \qquad p_t^{(m)} \propto \exp\left(-\varepsilon \sum_{s=1}^{t-1} \widehat{R}_s(\boldsymbol{w}_s^{(m)})\right) \qquad \varepsilon = \Theta\left(\sqrt{\frac{\ln M}{T}}\right)$$

Theoretical Analysis

- **Shift intensity**: $V_T = \sum_{t=2}^{T}\sum_{y=1}^{c} |p_t(y) - p_{t-1}(y)|$ $\geq \Theta(T^{-\frac{1}{2}})$
- When V_T is known:
  - Dynamic regret is minimized with step size $\eta = \Theta(V_T^{\frac{1}{3}} T^{-\frac{1}{3}})$:

$$\mathbb{E}\left[\underbrace{\sum_{t=1}^{T} R_t(\boldsymbol{w}_t)}_{\text{Risk of our model}} - \underbrace{\sum_{t=1}^{T} \min_{\boldsymbol{w}\in\mathcal{W}} R_t(\boldsymbol{w})}_{\text{Risk of the best model at each iteration}}\right] = \mathcal{O}(V_T^{\frac{1}{3}} T^{\frac{2}{3}})$$

$$R_t(\boldsymbol{w}) = \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x}|y)p_t(y)}[\ell(\boldsymbol{w}^\top \boldsymbol{x}, y)]$$

- Even when V_T is unknown:
  - ATLAS still achieves the same dynamic regret!
    - Number of learners: $M = 1 + \lceil \frac{1}{2}\log_2(1+2T)\rceil$
    - Step size: $\eta_m = 2^{m-1} Z/\sqrt{T}, \quad m = 1, \ldots, M$

ATLAS with Hints

- If we have some hints, can we perform better?
  - Hint function: $H_t(\boldsymbol{w})$ $\approx R_t(\boldsymbol{w}) = \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x}|y)p_t(y)}[\ell(\boldsymbol{w}^\top \boldsymbol{x}, y)]$

- ATLAS-ADA:
  - $\boldsymbol{w}_{t-\frac{1}{2}} = \Pi_{\mathcal{W}}\left[\boldsymbol{w}_{t-1} - \eta \nabla \widehat{R}_{t-1}(\boldsymbol{w}_{t-1})\right]$ ← Same as ATLAS
  - $\boldsymbol{w}_t = \operatorname*{argmin}_{\boldsymbol{w}\in\mathcal{W}} \left[\eta H_t(\boldsymbol{w}) + \frac{1}{2}\|\boldsymbol{w} - \boldsymbol{w}_{t-\frac{1}{2}}\|^2\right]$ ← Use the hint to match the next loss

- Dynamic regret: $\mathbb{E}\left[\sum_{t=1}^{T} R_t(\boldsymbol{w}_t) - \sum_{t=1}^{T} \min_{\boldsymbol{w}\in\mathcal{W}} R_t(\boldsymbol{w})\right] = \mathcal{O}\left(V_T^{\frac{1}{3}} G_T^{\frac{1}{3}} T^{\frac{1}{3}}\right)$
  - Reusability: $G_T = \sum_{t=1}^{T} \mathbb{E}\left[\sup_{\boldsymbol{w}\in\mathcal{W}} \|\nabla \widehat{R}_t(\boldsymbol{w}) - \nabla H_t(\boldsymbol{w})\|^2\right]$ $\leq \mathcal{O}(T)$
  - ATLAS-ADA is better and safe! $\mathcal{O}(V_T^{\frac{1}{3}} T^{\frac{2}{3}})$

Continuous Covariate Shift

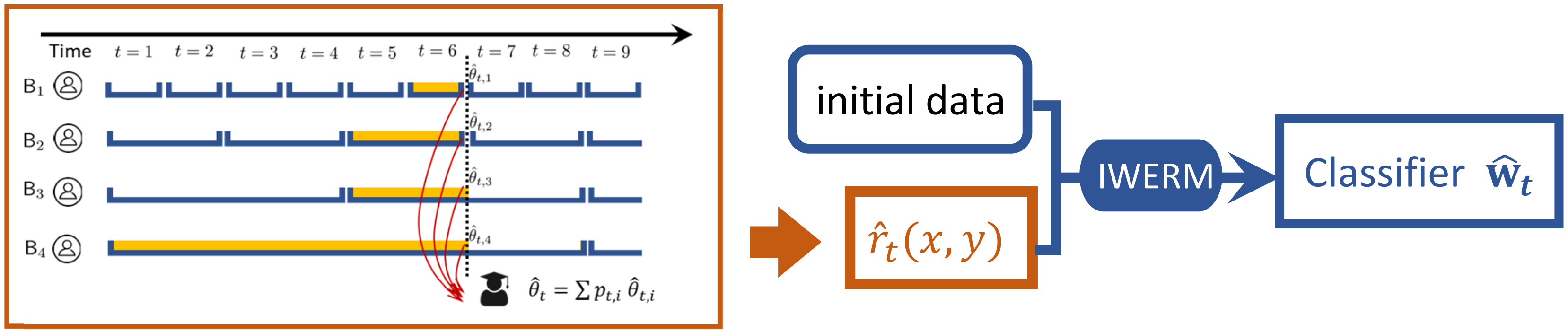
Zhang, Zhang, Zhao & Sugiyama (NeurIPS2023)

- Input density $p_t(\boldsymbol{x})$ changes arbitrarily over time, but output-given-input is unchanged: $p_{\mathrm{tr}}(y|\boldsymbol{x}) = p_t(y|\boldsymbol{x})$

$t = 1, \ldots, T$

- Given:
  - (Large) labeled training data: $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
  - (Small) unlabeled test data: $\{\boldsymbol{x}_i^{(t)}\}_{i=1}^{n_t} \overset{\mathrm{i.i.d.}}{\sim} p_t(\boldsymbol{x})$

- We use online density ratio estimation:



Contents

1. Importance Weighting
2. Continuous Shifts
3. Ongoing Challenges

Joint Distribution Shifts

- Many distribution shift works considers a specific shift type (e.g., covariate shift).

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

$$p_{\mathrm{tr}}(y|\boldsymbol{x}) = p_{\mathrm{te}}(y|\boldsymbol{x}) = p(y|\boldsymbol{x})$$

  - However, identification of the shift type is challenging.

- Let's consider joint shift:

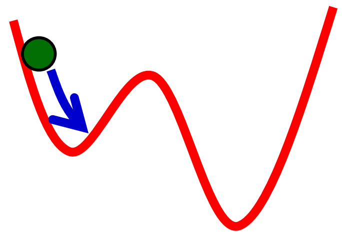
$$p_{\mathrm{tr}}(\boldsymbol{x}, y) \neq p_{\mathrm{te}}(\boldsymbol{x}, y)$$

Mini-Batch-Wise Loss Matching

Fang, Lu, Niu & Sugiyama (NeurIPS2020)

- **Given**:
  - (Large) labeled training data: $\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$
  - (Small) labeled test data: $\{(\boldsymbol{x}_j^{\mathrm{te}}, y_j^{\mathrm{te}})\}_{j=1}^{n_{\mathrm{te}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x}, y)$

- We try to learn the importance weight dynamically in the mini-batch-wise manner.



$$f \leftarrow f - \eta \nabla \widehat{R}(f)$$

$\eta > 0$: step size

- For each mini-batch $\{(\tilde{\boldsymbol{x}}_i^{\mathrm{tr}}, \tilde{y}_i^{\mathrm{tr}})\}_{i=1}^{\tilde{n}_{\mathrm{tr}}}, \{(\tilde{\boldsymbol{x}}_j^{\mathrm{te}}, \tilde{y}_j^{\mathrm{te}})\}_{j=1}^{\tilde{n}_{\mathrm{te}}}$, importance weights are estimated by kernel mean matching for loss values:

Huang+ (NeurIPS2006)

$$\frac{1}{\tilde{n}_{\mathrm{tr}}} \sum_{i=1}^{\tilde{n}_{\mathrm{tr}}} r_i \ell(f(\tilde{\boldsymbol{x}}_i^{\mathrm{tr}}), \tilde{y}_i^{\mathrm{tr}}) \approx \frac{1}{\tilde{n}_{\mathrm{te}}} \sum_{j=1}^{\tilde{n}_{\mathrm{te}}} \ell(f(\tilde{\boldsymbol{x}}_j^{\mathrm{te}}), \tilde{y}_j^{\mathrm{te}})$$

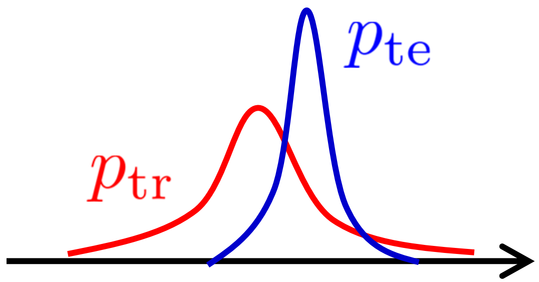
$$r_i \approx \frac{p_{\mathrm{te}}(\tilde{\boldsymbol{x}}_i^{\mathrm{tr}}, \tilde{y}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\tilde{\boldsymbol{x}}_i^{\mathrm{tr}}, \tilde{y}_i^{\mathrm{tr}})}$$

Out-Of-Domain Extension

- **Limitation of importance weighting**:
  - The training domain must cover the test domain.

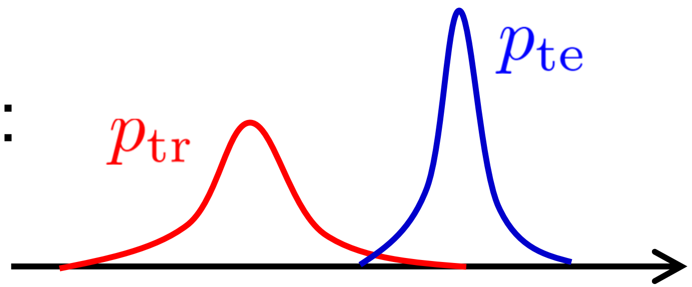
$$\frac{p_{\mathrm{te}}(\boldsymbol{x}, y)}{p_{\mathrm{tr}}(\boldsymbol{x}, y)} < \infty$$

- What if the test domain sticks out from the training domain?



- **Out-of-domain extension**: Fang, Lu, Niu & Sugiyama (NeurIPS2023)
  - Split training data into in-/out-domains by outlier detection (e.g., 1-class SVM):

$$\{(\boldsymbol{x}_j^{\mathrm{te_{in}}}, y_j^{\mathrm{te_{in}}})\}_{j=1}^{n_{\mathrm{te_{in}}}} \qquad \{(\boldsymbol{x}_j^{\mathrm{te_{out}}}, y_j^{\mathrm{te_{out}}})\}_{j=1}^{n_{\mathrm{te_{out}}}}$$

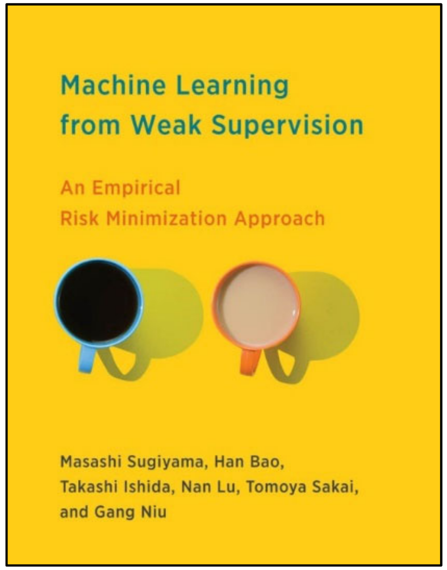


  - Compute the loss separately:

$$\frac{n_{\mathrm{te_{in}}}}{n_{\mathrm{tr}} n_{\mathrm{te}}} \sum_{i=1}^{n_{\mathrm{tr}}} \frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})} \ell(f(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}}) + \frac{1}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te_{out}}}} \ell(f(\boldsymbol{x}_j^{\mathrm{te_{out}}}), y_j^{\mathrm{te_{out}}})$$

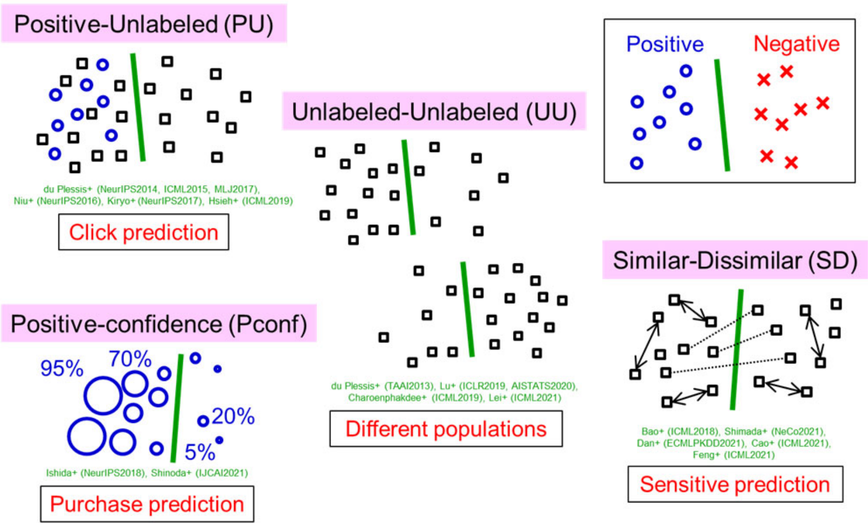
Ongoing Challenges

- For joint shift adaptation, requiring labeled test data is too strong.
- Can we use weakly supervised learning?
  - Unbiased risk estimation from weak supervision.
  - Any loss, classifier, and optimizer can be used.



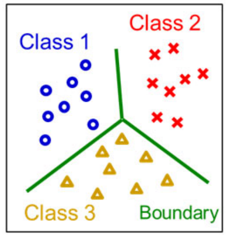
Sugiyama, Bao, Ishida, Lu, Sakai & Niu (MIT Press, 2022)

Weakly Supervised Classification (Binary)



Weakly Supervised Classification (Multiclass)

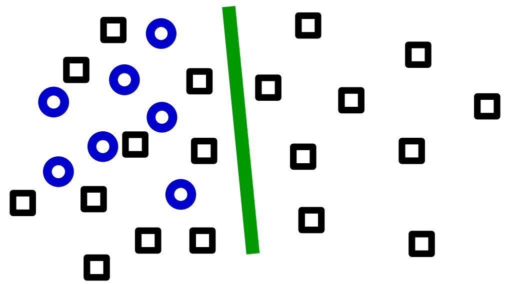
- Multi-class weak-labels:
  - Complementary labels: Specify a class that a pattern does not belong to ("not 1").
    Ishida et al. (NIPS2017, ICML2019), Chou et al. (ICML2020)
  - Partial labels: Specify a subset of classes that contains the correct one ("1 or 2").
    Feng et al. (ICML2020, NeurIPS2020), Lv et al. (ICML2020)
  - Single-class confidence: One-class data with full confidence ("1 with 60%, 2 with 30%, and 3 with 10%")
    Cao et al. (arXiv2021)



Positive-Unlabeled (PU) Classification

- **Given:** PU samples (no N samples)

$$\{\boldsymbol{x}_i^{\mathrm{P}}\}_{i=1}^{n_{\mathrm{P}}} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x}|y=+1) \qquad \{\boldsymbol{x}_j^{\mathrm{U}}\}_{j=1}^{n_{\mathrm{U}}} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x})$$



Unlabeled (mixture of positives and negatives)

- **Goal:** Obtain a risk-minimizing classifier

$$\min_{f\in\mathcal{F}} R(f) \qquad R(f) = \mathbb{E}_{p(\boldsymbol{x},y)}\left[\ell\Big(y, f(\boldsymbol{x})\Big)\right]$$

$\mathbb{E}$: expectation $\quad \ell$: loss $\quad y = \{+1, -1\}$

- **Unbiased risk estimator:** du Plessis+ (NeurIPS2014, ICML2015)

$$\widehat{R}_{\mathrm{PU}}(f) = \frac{\pi}{n_{\mathrm{P}}}\sum_{i=1}^{n_{\mathrm{P}}}\ell\Big(+1, f(\boldsymbol{x}_i^{\mathrm{P}})\Big) + \frac{1}{n_{\mathrm{U}}}\sum_{j=1}^{n_{\mathrm{U}}}\ell\Big(-1, f(\boldsymbol{x}_j^{\mathrm{U}})\Big) - \frac{\pi}{n_{\mathrm{P}}}\sum_{i=1}^{n_{\mathrm{P}}}\ell\Big(-1, f(\boldsymbol{x}_i^{\mathrm{P}})\Big)$$

$\pi = p(y=+1)$

  - **Optimal convergence rate:** Niu, du Plessis, Sakai, Ma & Sugiyama (NIPS2016)

$$R(\widehat{f}_{\mathrm{PU}}) - R(f^*) \le C(\delta)\left(\frac{2\pi}{\sqrt{n_{\mathrm{P}}}} + \frac{1}{\sqrt{n_{\mathrm{U}}}}\right)$$

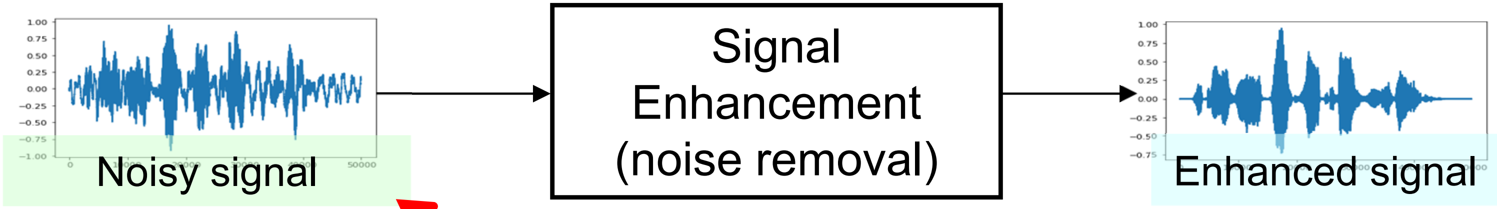
with probability $1-\delta$

$\widehat{f}_{\mathrm{PU}} = \mathrm{argmin}_{f\in\mathcal{F}}\, \widehat{R}_{\mathrm{PU}}(f)$

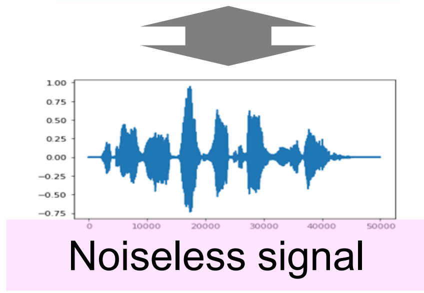
$f^* = \mathrm{argmin}_{f\in\mathcal{F}}\, R(f)$

Signal Enhancement by PU Classification

Ito & Sugiyama (ICASSP2023, Best Paper Award)



Parallel training data



- **Existing method:** Use noisy/noiseless parallel training data
  - In practice, use synthetic data
    → Do not generalize well in reality.

- **Proposed method:** Use non-parallel noisy signal and noise.

	Methods	SI-SNRi [dB]
Non-parallel	Proposed	14.62 (0.20)
Non-parallel	MixIT Wisdom+ (NeurIPS2020)	12.19 (4.50)
Parallel	Supervised	15.86 (1.28)