Towards More Robust and Reliable Machine Learning

IEEE WCCI 2024

Masashi Sugiyama

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

RIKEN

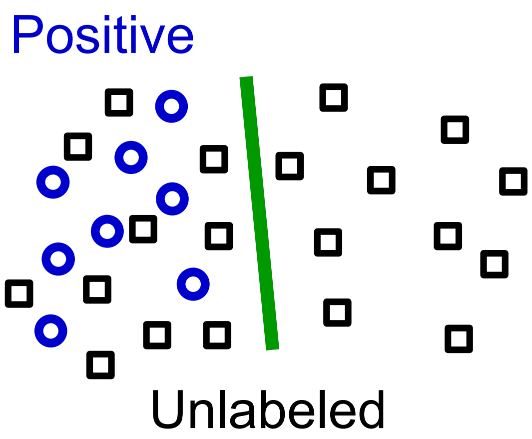
AIP

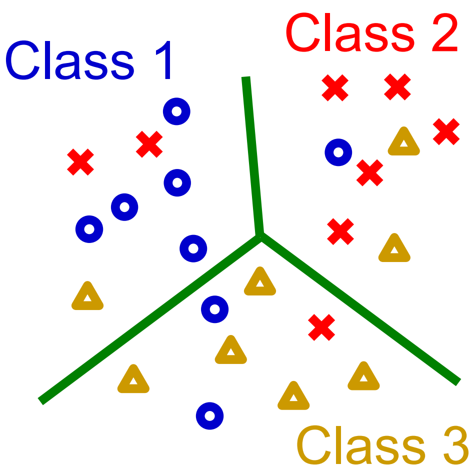
東京大学
THE UNIVERSITY OF TOKYO

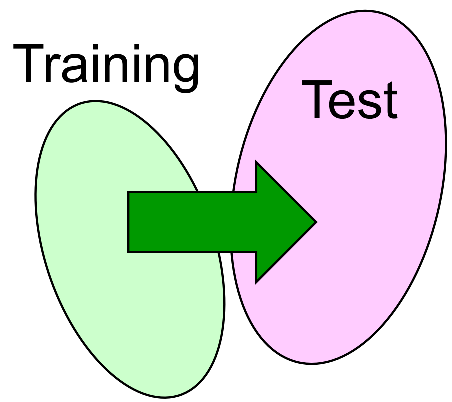
http://www.ms.k.u-tokyo.ac.jp/sugi/

Reliable Machine Learning

- Reliability of machine learning systems can be degraded by various factors:
  - Insufficient information: weak supervision.
  - Label noise: human error, sensor error.
  - Data bias: changing environments, privacy.
- Improving the reliability is an urgent challenge!





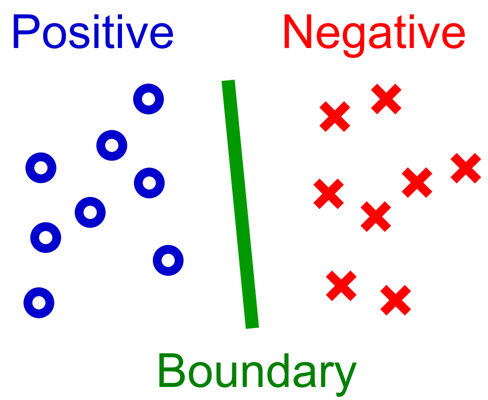


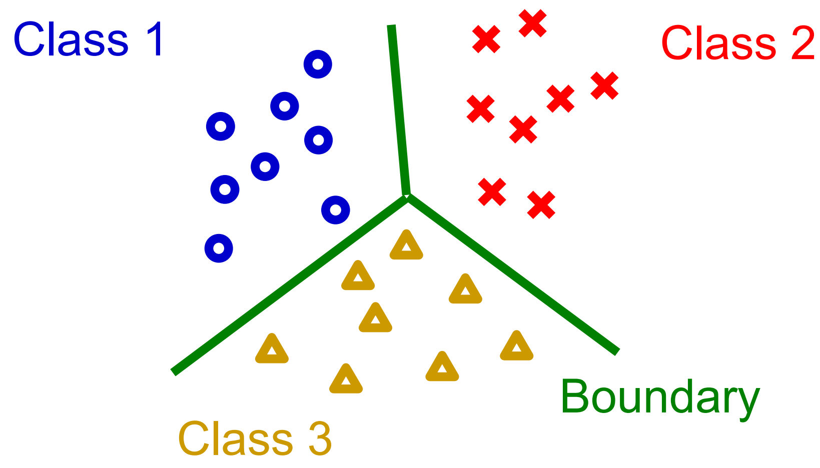
Contents

1. Weakly Supervised Learning
2. Noisy-Label Learning
3. Transfer Learning
4. Towards More Reliable Learning

Weakly Supervised Classification

- Supervised classification from big labeled data is successful: speech, image, language, …





- However, there are many applications where big labeled data is not available:
  - Medicine, disaster, robot, brain, …

- We want to utilize "weak" supervision that can be collected easily!

Positive-Unlabeled (PU) Classification

Li+ (IJCAI2003)

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

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

- **Goal:** Obtain a classifier minimizing the PN risk.

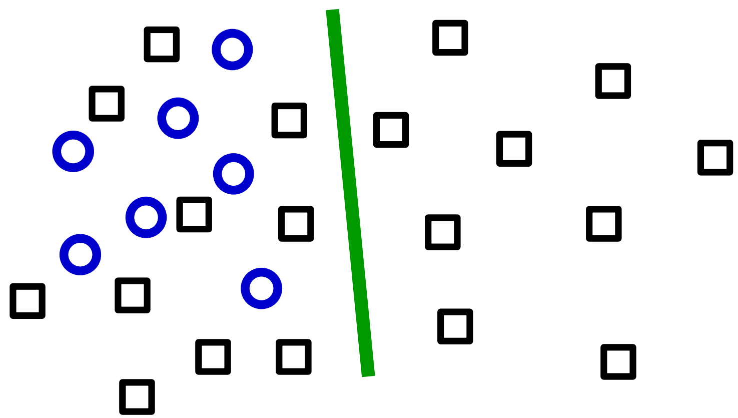
$$\min_f R(f) \qquad R(f) = \mathbb{E}_{p(\boldsymbol{x},y)}\left[\ell\Big(y, f(\boldsymbol{x})\Big)\right]$$

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

Example: Ad click prediction

- Clicked ad: User likes it → P
- Unclicked ad: User dislikes it or User likes it but doesn't have time to click it → U (=P or N)

Positive [Negative]



Unlabeled (mixture of positives and negatives)

PU Unbiased Risk Estimation

du Plessis+ (NeurIPS2014, ICML2015)

- Decompose the risk:

$$R(f) = \underbrace{\pi \mathbb{E}_{p(\boldsymbol{x}|y=+1)}\Big[\ell\Big(+1, f(\boldsymbol{x})\Big)\Big]}_{\text{Risk for P data}} + \underbrace{(1-\pi)\mathbb{E}_{p(\boldsymbol{x}|y=-1)}\Big[\ell\Big(-1, f(\boldsymbol{x})\Big)\Big]}_{\text{Risk for N data } R^{-}(f)}$$

$\pi = p(y=+1)$: Class prior (assumed known) →

Scott+ (AISTATS2009)
Ramaswamy+ (ICML2016)
du Plessis+ (MLJ2017)
Yao+ (ICLR2022)

- Without N data, $R^{-}(f)$ can not be estimated directly:
  - Eliminate the expectation over N data as

$$R^{-}(f) = \mathbb{E}_{p(\boldsymbol{x})}\Big[\ell\Big(-1, f(\boldsymbol{x})\Big)\Big] - \pi \mathbb{E}_{p(\boldsymbol{x}|y=+1)}\Big[\ell\Big(-1, f(\boldsymbol{x})\Big)\Big]$$

$$p(\boldsymbol{x}) = \pi p(\boldsymbol{x}|y=+1) + (1-\pi)p(\boldsymbol{x}|y=-1)$$

- Unbiased risk estimator:

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

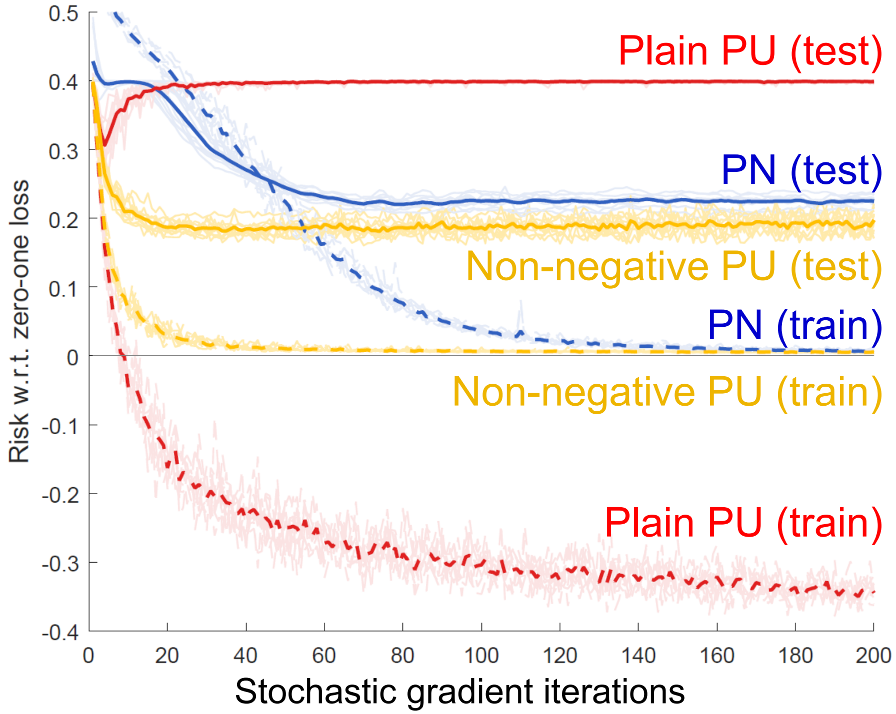
Non-Negative Risk Correction

Kiryo+ (NeurIPS2017) , Lu+ (AISTATS2020)

$$R(f) = \underbrace{\pi \mathbb{E}_{p(\boldsymbol{x}|y=+1)}\Big[\ell\Big(+1, f(\boldsymbol{x})\Big)\Big]}_{\text{Risk for P data}} + \underbrace{(1-\pi)\mathbb{E}_{p(\boldsymbol{x}|y=-1)}\Big[\ell\Big(-1, f(\boldsymbol{x})\Big)\Big]}_{\text{Risk for N data } R^{-}(f)}$$

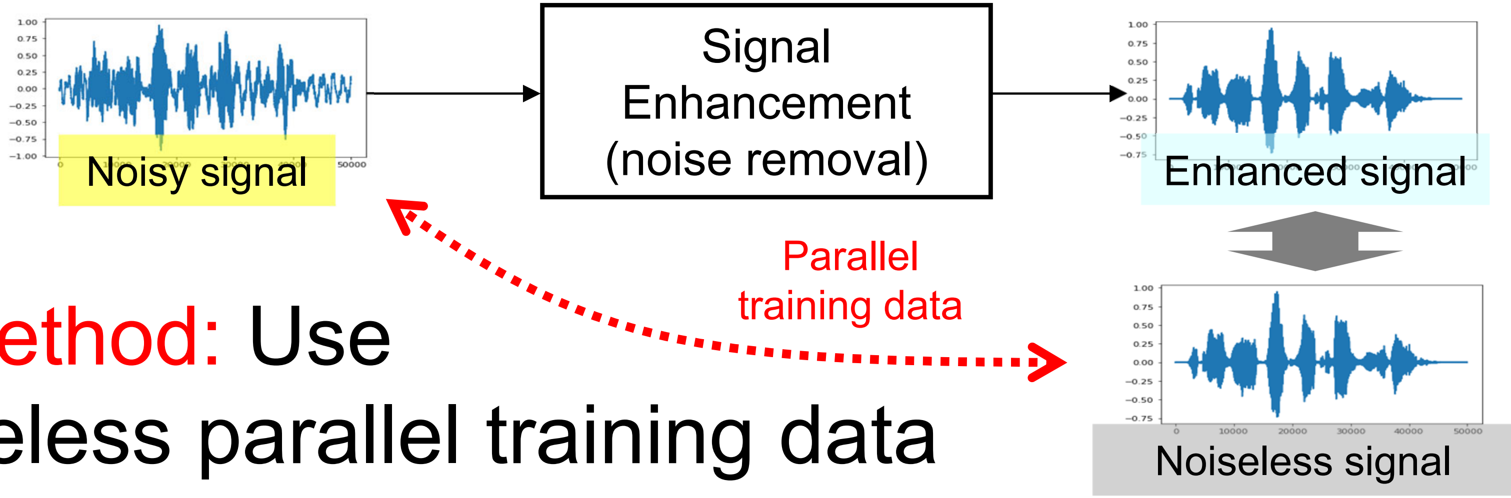
- Risk for N data: $R^{-}(f) = \mathbb{E}_{p(\boldsymbol{x})}\Big[\ell\Big(-1, f(\boldsymbol{x})\Big)\Big] - \pi \mathbb{E}_{p(\boldsymbol{x}|y=+1)}\Big[\ell\Big(-1, f(\boldsymbol{x})\Big)\Big]$
- Empirical estimate: $\widehat{R}^{-}_{\mathrm{PU}}(f) = \frac{1}{n_{\mathrm{U}}}\sum_{i=1}^{n_{\mathrm{U}}} \ell\Big(-1, f(\boldsymbol{x}_i^{\mathrm{U}})\Big) - \frac{\pi}{n_{\mathrm{P}}}\sum_{i=1}^{n_{\mathrm{P}}} \ell\Big(-1, f(\boldsymbol{x}_i^{\mathrm{P}})\Big)$
- When loss is non-negative:
  - True $R^{-}(f)$ is non-negative.
  - But empirical estimate can be negative!
- Non-negative correction:

$$\widetilde{R}_{\mathrm{PU}}(f) = \frac{\pi}{n_{\mathrm{P}}}\sum_{i=1}^{n_{\mathrm{P}}} \ell\Big(f(\boldsymbol{x}_i^{\mathrm{P}})\Big) + \max\Big\{0, \ \widehat{R}^{-}_{\mathrm{PU}}(f)\Big\}$$



Signal Enhancement by PU Classification

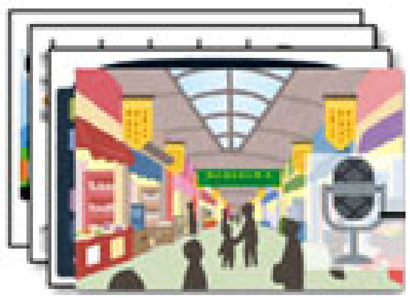
Ito+ (ICASSP2023, Best Paper Award)



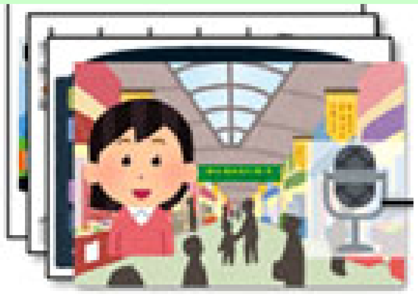
- **Existing method:** Use noisy/noiseless parallel training data
  - In practice, use synthetic data
    → Do not generalize well in reality.

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

Pure noise (positive)



Noise + Speech (unlabeled)

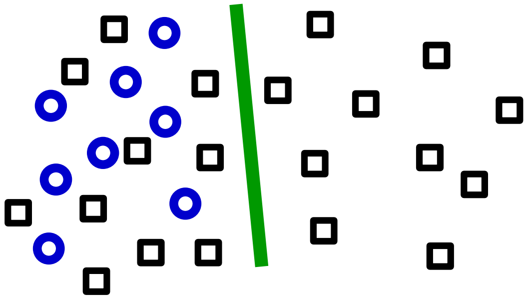


	Methods	SI-SNRi [dB]
Non-parallel	Proposed	14.62 (0.20)
Non-parallel	MixIT Wisdom+ (NeurIPS2020)	12.19 (4.50)
Parallel	Supervised	15.86 (1.28)

Various Extensions (Binary)

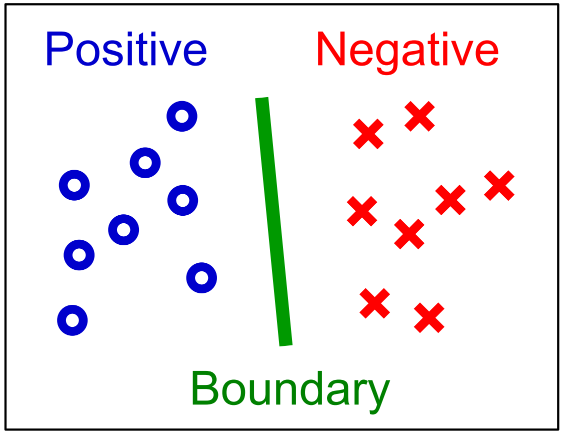
- Similar unbiased risk estimation is possible!

Positive-Unlabeled (PU)

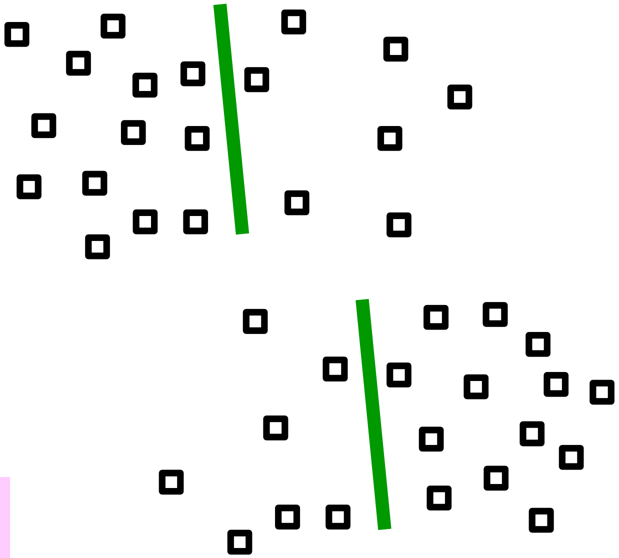


du Plessis+ (NeurIPS2014, ICML2015, MLJ2017), Niu+ (NeurIPS2016), Kiryo+ (NeurIPS2017), Hsieh+ (ICML2019)

Click prediction



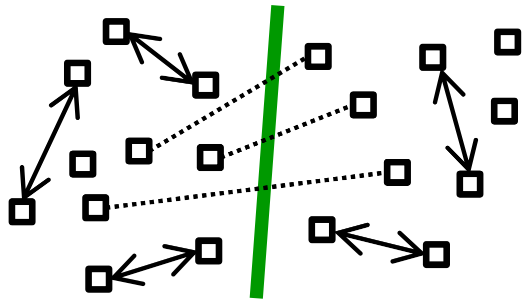
Unlabeled-Unlabeled (UU)



du Plessis+ (TAAI2013), Lu+ (ICLR2019, AISTATS2020), Charoenphakdee+ (ICML2019), Lei+ (ICML2021)

Different populations

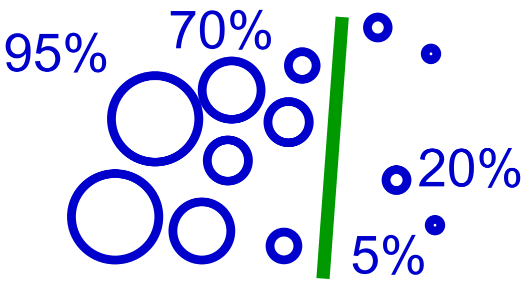
Similar-Dissimilar (SD)



Bao+ (ICML2018), Shimada+ (NeCo2021), Dan+ (ECMLPKDD2021), Cao+ (ICML2021), Feng+ (ICML2021)

Sensitive prediction

Positive-confidence (Pconf)

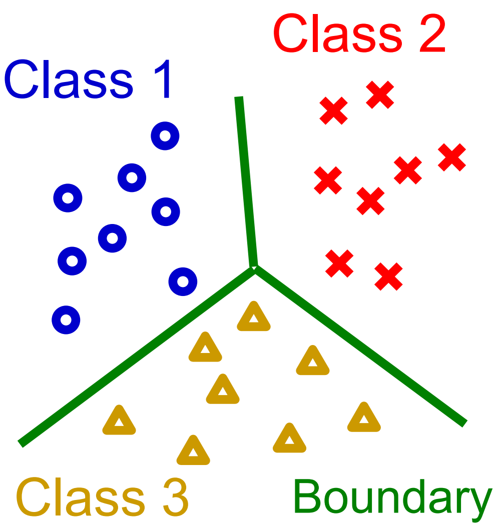


Ishida+ (NeurIPS2018), Shinoda+ (IJCAI2021)

Purchase prediction

Various Extensions (Multiclass)

- Labeling patterns in multi-class problems is even more painful.

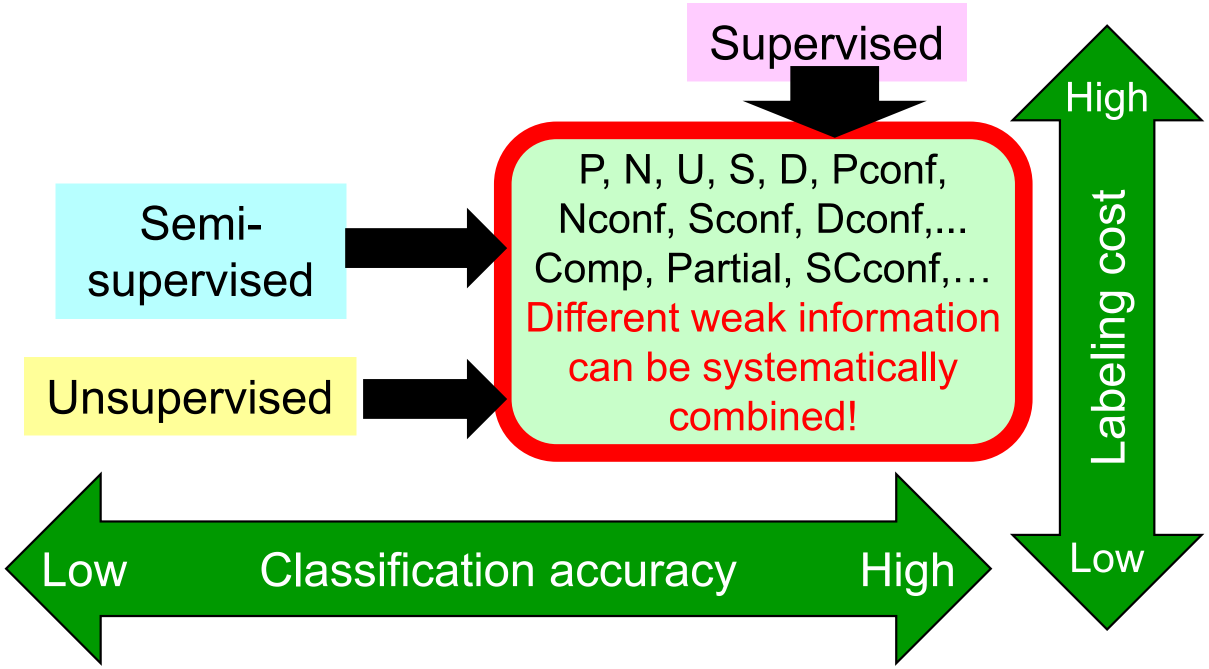


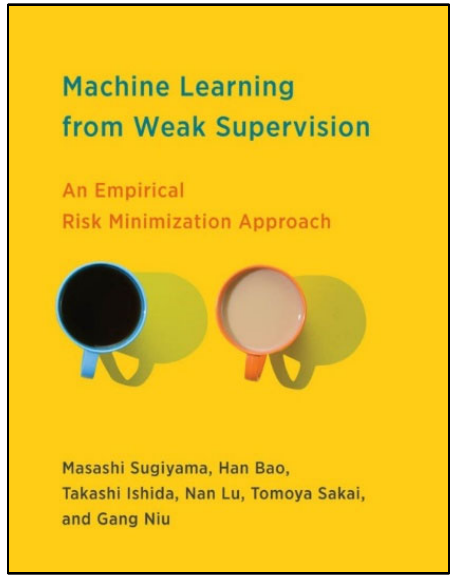
- Multi-class weak-labels:
  - Complementary label: Ishida+ (NeurIPS2017, ICML2019), Chou+ (ICML2020)
    Specifies a class that a pattern does not belong to ("not 1").
  - Partial label: Specifies a subset of classes that contains the correct one ("1 or 2"). Feng+ (ICML2020, NeurIPS2020), Lv+ (ICML2020)
  - Single-class confidence: Cao+ (arXiv2021)
    One-class data with full confidence ("1 with 60%, 2 with 30%, and 3 with 10%")

- Similar unbiased risk estimation is possible!

Summary: Weakly Supervised Learning (WSL)

- Empirical risk minimization framework for WSL:
  - Any loss, classifier, and optimizer can be used.





Sugiyama+
(MIT Press, 2022)

- Recent progress:
  - Unified frameworks, new problems, new algorithms,…

    Chiang+ (arXiv2023), Chen+ (ICML2024) Wang+ (NeurIPS2023) Wang+ (ICML2024)
  - Imitation learning, large language models,…

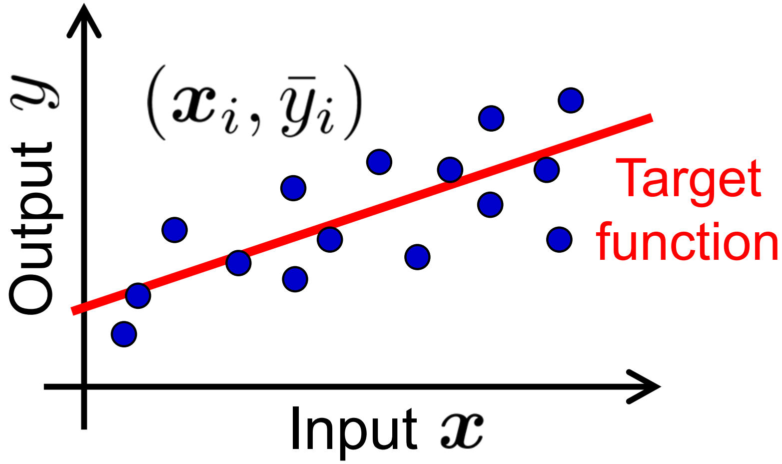
    Cai+ (NeurIPS2023) Zhang+ (ICML2024)

Contents

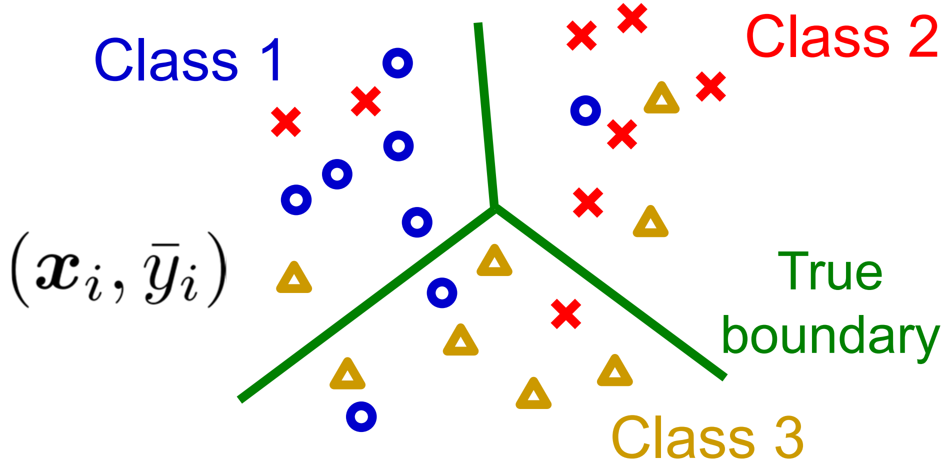
1. Weakly Supervised Learning
2. Noisy-Label Learning
3. Transfer Learning
4. Towards More Reliable Learning

Supervised Learning with Noisy Output

Regression (additive noise)



Classification (label flipping noise)



$$\min_{\boldsymbol{g}} \sum_{i=1}^{n} \ell\Big(\bar{y}_i, \boldsymbol{g}(\boldsymbol{x}_i)\Big)$$

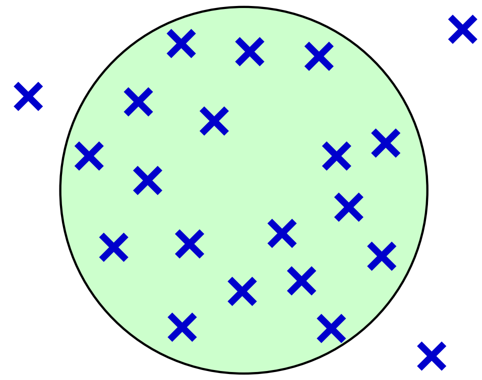
ℓ : loss

$\bar{y}$: noisy output

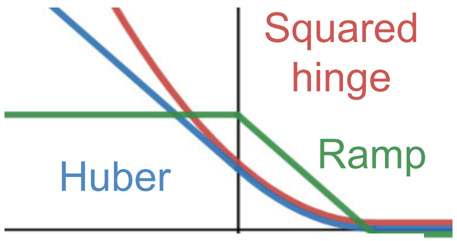
$\boldsymbol{g}$: probabilistic classifier

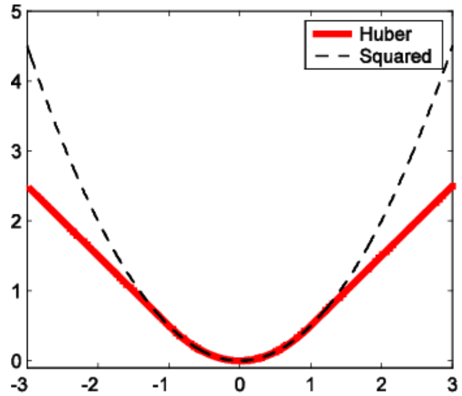
- Hasn't such a classic problem been solved?
  - Regression: Yes, noisy big data yield consistency.
  - Classification: Specific noise reduction mechanism is needed to achieve consistency!

Classical Approaches

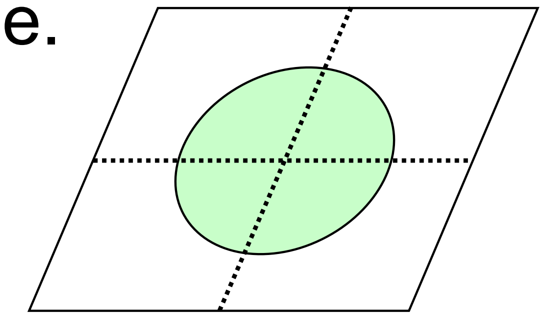


- **Unsupervised outlier removal:**
  - Substantially more difficult than classification.
- **Robust loss:**
  - Works well for regression, but limited effectiveness for classification.

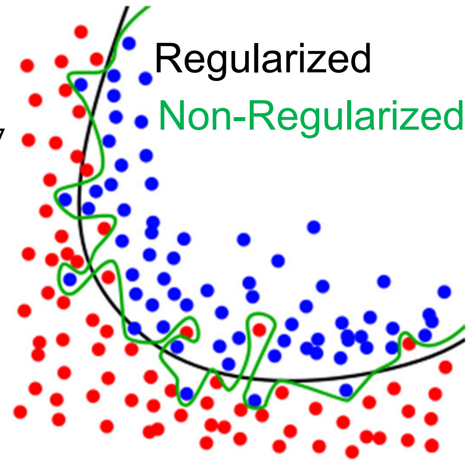




- **Regularization:**
  - Effective in suppressing overfitting, but too smooth for strong noise.
- **Need new approaches!**



ℓ_2-regularization



https://en.wikipedia.org/wiki/Overfitting

Correction with Noise Transition

- Noise transition matrix $\boldsymbol{T}$:
  - Clean-to-noisy flipping probability.

$\boldsymbol{T} =$

1	0	0
0.1	0.8	0.1
0.5	0.5	0

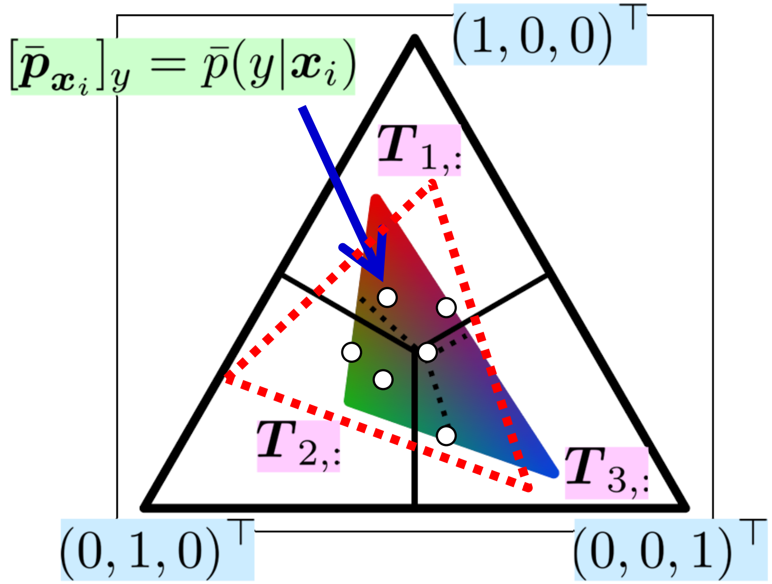
- Major approaches: Patrini+ (CVPR2017)
  - Classifier adjustment by $\boldsymbol{T}^\top$ to simulate noise.
  - Loss correction by $\boldsymbol{T}^{-1}$ to eliminate noise.

- We want to estimate $\boldsymbol{T}$ only from noisy data:
  - Use human cognition as a “mask” for $\boldsymbol{T}$. Han+ (NeurIPS2018)
  - Reduce estimation error of $\boldsymbol{T}$. Xia+ (NeurIPS2019) Yao+ (NeurIPS2020)
  - Learn $\boldsymbol{T}$ and classifier simultaneously. Zhang+ (ICML2021)
  - Estimate $\boldsymbol{T}$ under weaker conditions. Li+ (ICML2021)

Volume Minimization

Li+ (ICML2021)

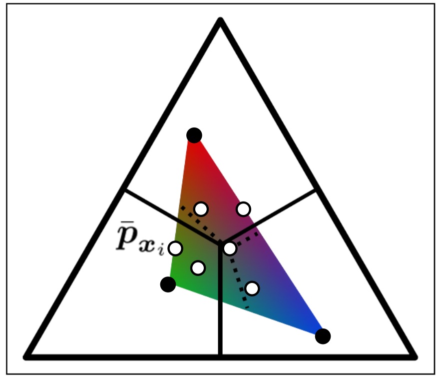
- Noise transition matrix $\boldsymbol{T}$ forms a simplex.
- Noisy training data $\{(\boldsymbol{x}_i, \bar{y}_i)\}_{i=1}^{n}$ can be mapped in the simplex.



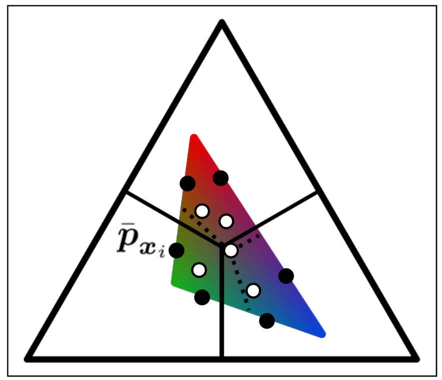
- Find a minimum volume simplex that contains all training data:

$$\min_{\boldsymbol{T}', \boldsymbol{g}} \sum_{i=1}^{n} \ell(\bar{y}_i, \boldsymbol{T}'^\top \boldsymbol{g}(\boldsymbol{x}_i)) + \lambda \log \det(\boldsymbol{T}')$$

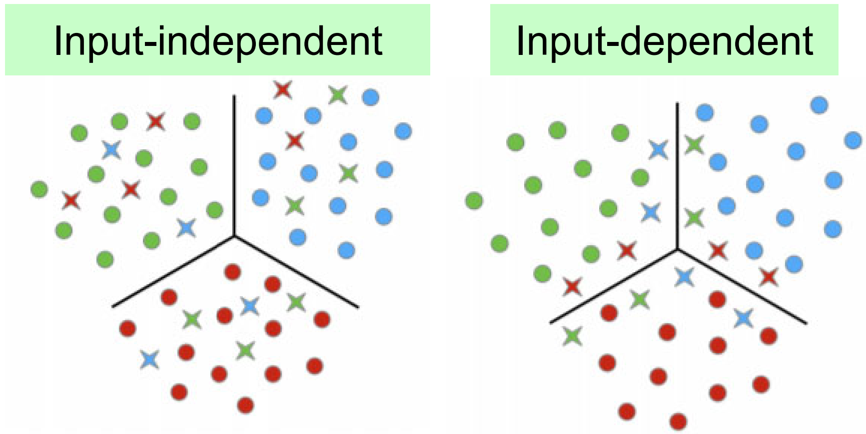
$\lambda > 0$



  - With noiseless labels, we can find the true $\boldsymbol{T}$.
  - Even without noiseless labels, "sufficiently scattered" training data allow identification of the true $\boldsymbol{T}$!



Beyond Input-Independent Noise



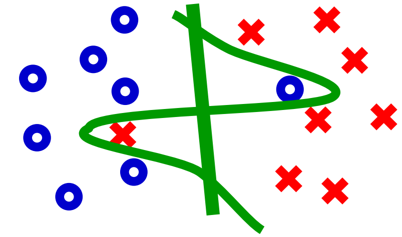
- Real-world noise may be input-dependent:
  - E.g., noise level is high near the boundary.
- Modeling input-dependent noise: $T_{y,\bar{y}}(\boldsymbol{x}) = \bar{p}(\bar{y}|y, \boldsymbol{x})$
  - Extremely challenging to estimate the noise transition matrix function!
- Exploring heuristic solutions:
  - Parts-based estimation. Xia+ (NeurIPS2020)
  - Use of additional confidence scores. Berthon+ (ICML2021)
  - Manifold regularization. Cheng+ (CVPR2022)

Co-teaching

Memorization of neural nets:

Arpit+ (ICML2017)
Zhang+ (ICLR2017)

- Stochastic gradient descent fits clean data faster.
- However, naïve early stopping does not work well.



"Co-teaching" between two neural nets:

- Teach small-loss data each other.

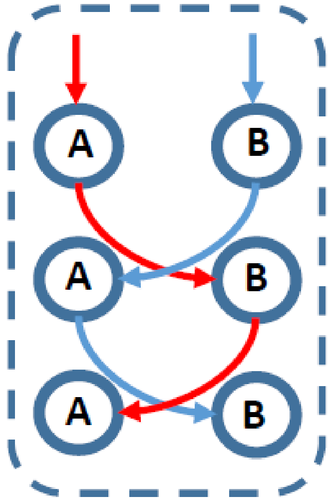
Han+ (NeurIPS2018)

- Teach only disagreed data.

Yu+ (ICML2019)

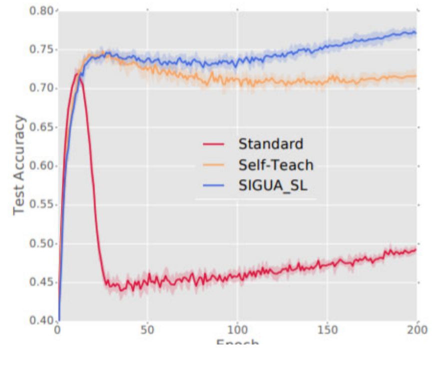
- Gradient ascent for large-loss data.

Han+ (ICML2020)



No theory but very robust in experiments:

- Works well even if 50% random label flipping!



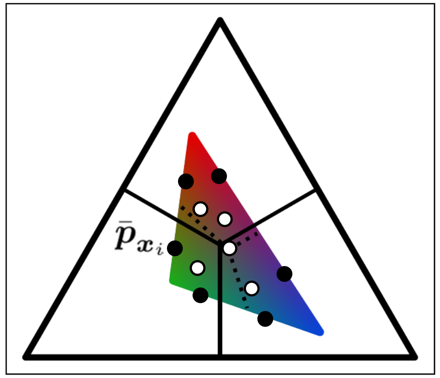
Summary: Noisy-Label Learning

- Explicit treatment of label noise is necessary:
  - Loss correction by noise transition is promising.

$$T_{y,\bar{y}} = \bar{p}(\bar{y}|y)$$

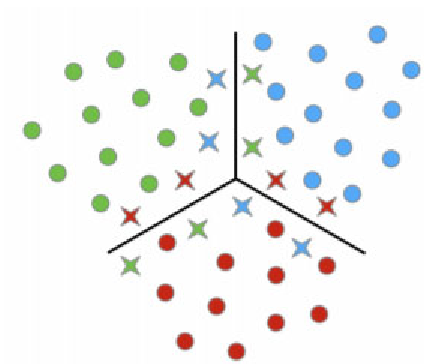
- However, noise transition is generally non-identifiable:

$$\boldsymbol{T}^\top \boldsymbol{p} = \boldsymbol{T}_2^\top (\boldsymbol{T}_1^\top \boldsymbol{p}) \qquad \boldsymbol{T} = \boldsymbol{T}_1 \boldsymbol{T}_2$$



  - Recent development allows consistent estimation under mild assumptions.

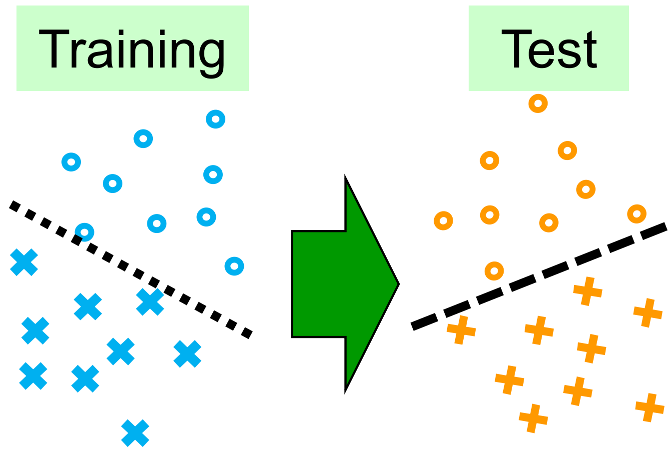
- Real-world noise is often input-dependent:
  - Heuristic solutions have been developed.
  - Further theoretical development is needed.

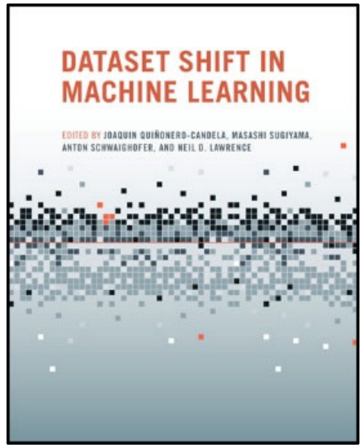


Contents

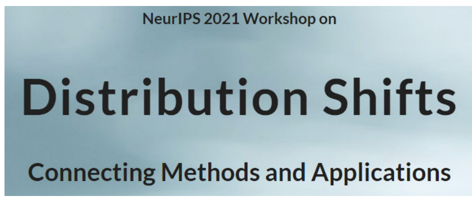
Transfer Learning

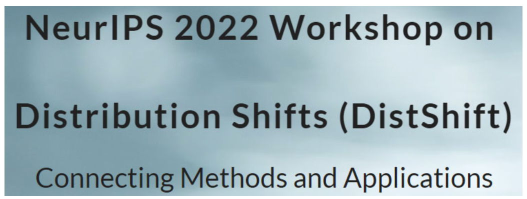
- Training/test data often follow **different distributions:**
  - Changing environments,
  - Sample selection bias (privacy).
- **Transfer learning:**
  - Train a test-domain predictor using training data from different domains.





Quiñonero-Candela+
(MIT Press 2009)





Basics: Importance-Weighted Training

- **Covariate shift:** Only input distributions change.

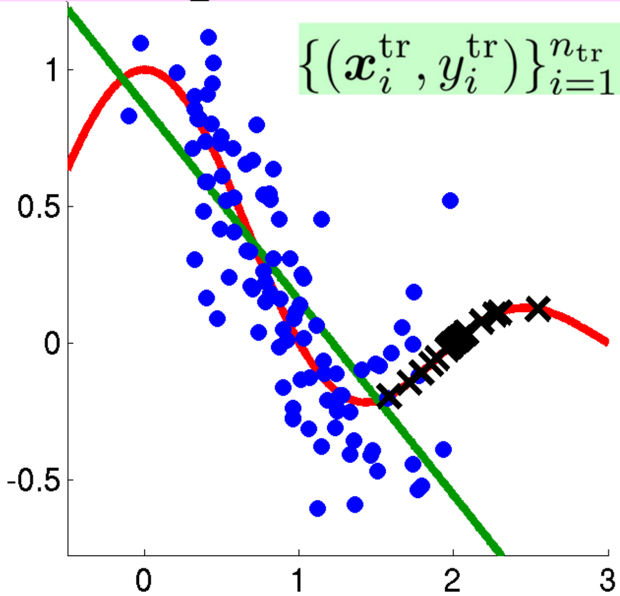
Shimodaira (JSPI2000)

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

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

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

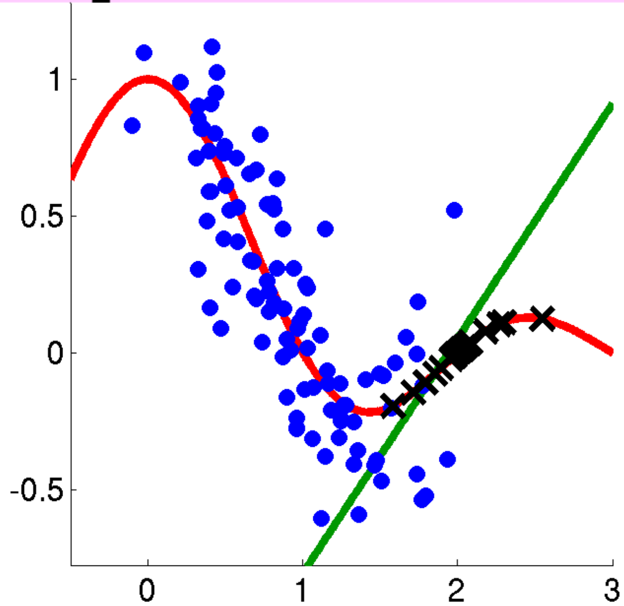
$$\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{tr}}(\boldsymbol{x}, y)$$



Ordinary training is not consistent

Importance

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



Importance-weighted training is consistent

Direct Importance Estimation

- Goal: Estimate $\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ from training and test input data

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

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

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

$K(\boldsymbol{x},\cdot)$: kernel

- Least-squares importance fitting (LSIF):
  - Fit a model $r(\boldsymbol{x})$ to $\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ by least squares: Kanamori+ (NeurIPS2008)

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

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

- They do not estimate $p_{\mathrm{tr}}(\boldsymbol{x}), p_{\mathrm{te}}(\boldsymbol{x})$, but $\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ directly!

Classical Two-Step Adaptation

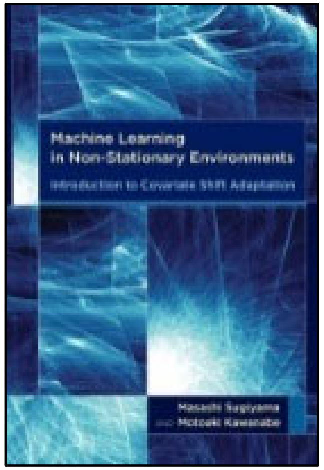
1. Importance weight estimation
   (e.g., least-squares importance fitting):

Kanamori+ (JMLR2009)

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

2. Weighted predictor training:

$$\widehat{f} = \underset{f}{\mathrm{argmin}}\, \widehat{\mathbb{E}}_{p_{\mathrm{tr}}(\boldsymbol{x},y)} [\widehat{r}(\boldsymbol{x}) \ell(f(\boldsymbol{x}), y)]$$



Sugiyama+ (MIT Press 2012)

- However, estimation error in Step 1 is not taken into account in Step 2.
  - We want to integrate these two steps!

Joint Weight-Predictor Optimization

Zhang+ (ACML2020, SNCS2021)

- Given: Labeled training data and unlabeled test data

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

- Joint minimization of a risk upper bound:

$$\min_{r \geq 0, f} J_{\ell'}(r, f) \qquad \frac{1}{2} R_\ell(f)^2 \leq J_{\ell'}(r, f) \qquad \ell \leq 1, \ell' \geq \ell$$

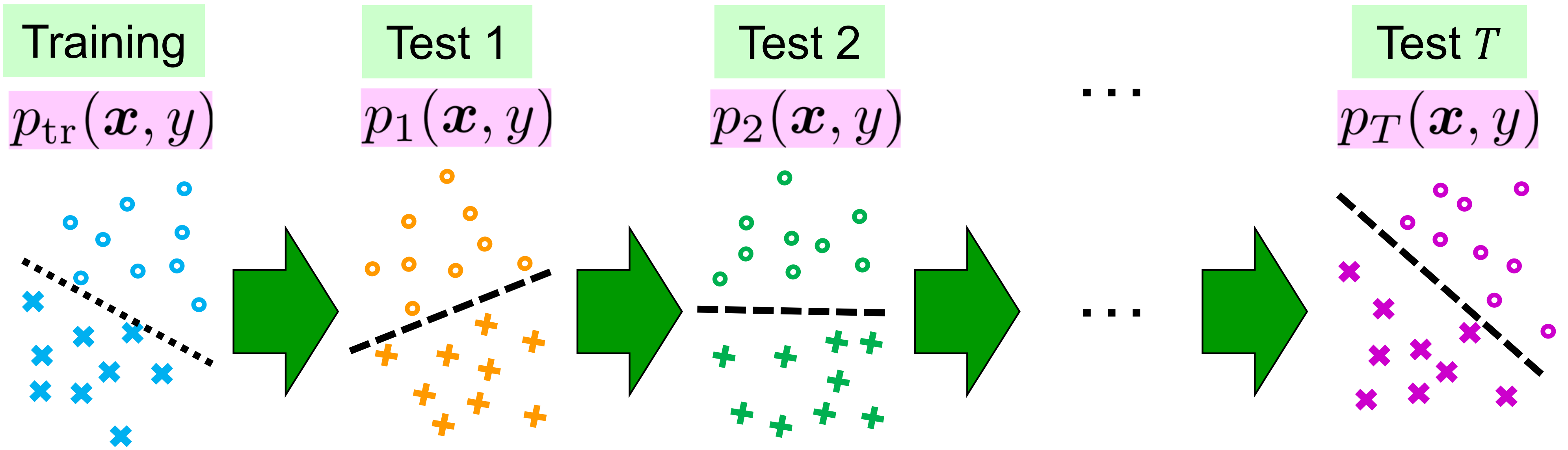
$$R_\ell(f) = \mathbb{E}_{p_{\mathrm{te}}(\boldsymbol{x}, y)}[\ell(f(\boldsymbol{x}), y)]$$

$$J_{\ell'}(r, f) = \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x})}\left[\left(r(\boldsymbol{x}) - \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}\right)^2\right] \quad \leftarrow \text{1st step}$$

$$+ \left(\mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x}, y)}[r(\boldsymbol{x})\ell'(f(\boldsymbol{x}), y)]\right)^2 \quad \leftarrow \text{2nd step}$$

  - Classic approach corresponds to 2-step minimization.

Extensions to Sequential Shifts



- Sequential label shift: Bai+ (NeurIPS2022)
  - Only class-prior $p_t(y)$ changes.
- Sequential covariate shift: Zhang+ (NeurIPS2023)
  - Only input density $p_t(\boldsymbol{x})$ changes.
- Without knowing the shift intensity, we can achieve the same dynamic regret as the case with known shift intensity.

$$\mathbb{E}\left[\sum_{t=1}^{T} R_t(f_t) - \sum_{t=1}^{T} \min_{f\in\mathcal{F}} R_t(f)\right]$$

Contents

Joint Shift

- Many distribution shift works focus on a particular shift type (e.g., covariate shift):

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

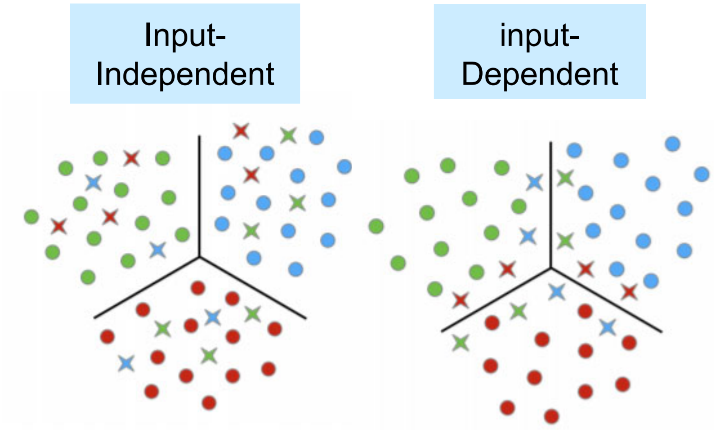
  - However, identification of the shift type is challenging.

- Label noise is also a type of distribution shift:

$$p_{\mathrm{tr}}(\bar{y}|\boldsymbol{x}) = \sum_{y} \underbrace{p(\bar{y}|y,\boldsymbol{x})}_{\text{Noise transition}} p_{\mathrm{te}}(y|\boldsymbol{x})$$

$\bar{y}$: Noisy class label

  - Nice theory for input-independent noise: $p(\bar{y}|y,\boldsymbol{x}) = p(\bar{y}|y)$
  - But input-dependent noise is hard.



- Let's consider joint shift:

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

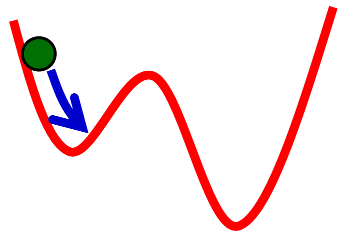
Mini-Batch-Wise Loss Matching

- Given:

Fang+ (NeurIPS2020)

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

- 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)

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

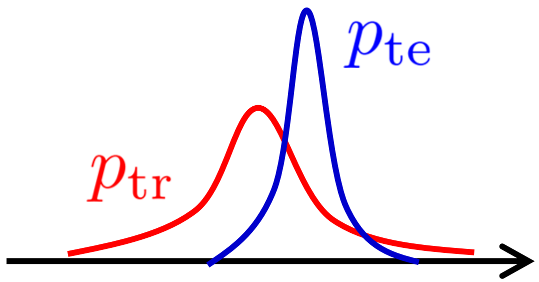
$$r_i \approx \frac{p_{\mathrm{te}}(\tilde{\boldsymbol{x}}_i^{\mathrm{tr}}, \tilde{y}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\tilde{\boldsymbol{x}}_i^{\mathrm{tr}}, \tilde{y}_i^{\mathrm{tr}})}$$

Out-of-Domain Extension

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

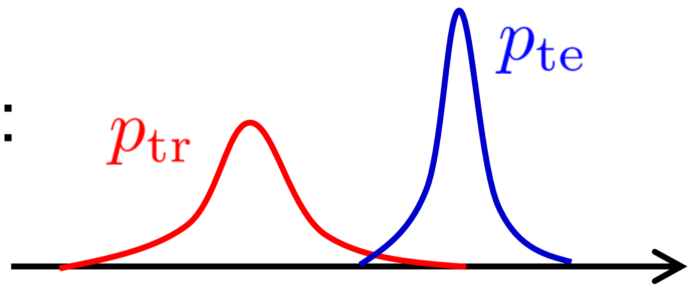
$$\frac{p_{\mathrm{te}}(\boldsymbol{x},y)}{p_{\mathrm{tr}}(\boldsymbol{x},y)} < \infty$$

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



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

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

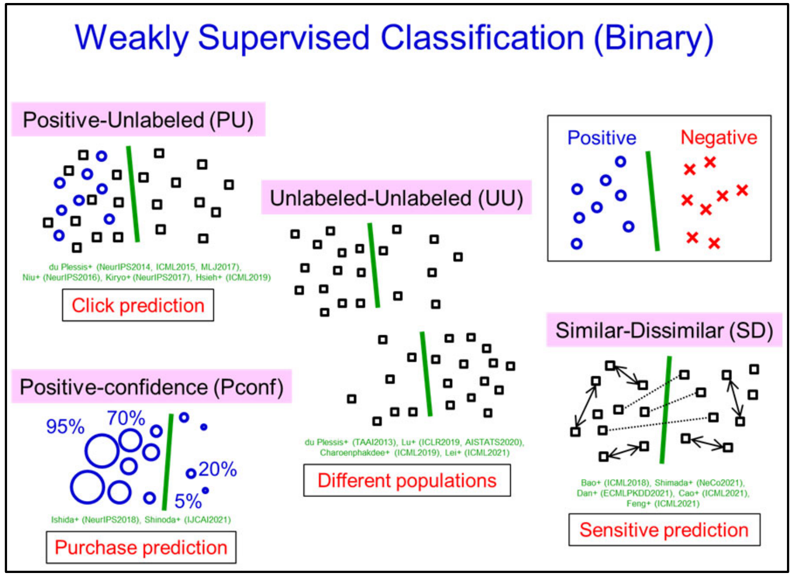


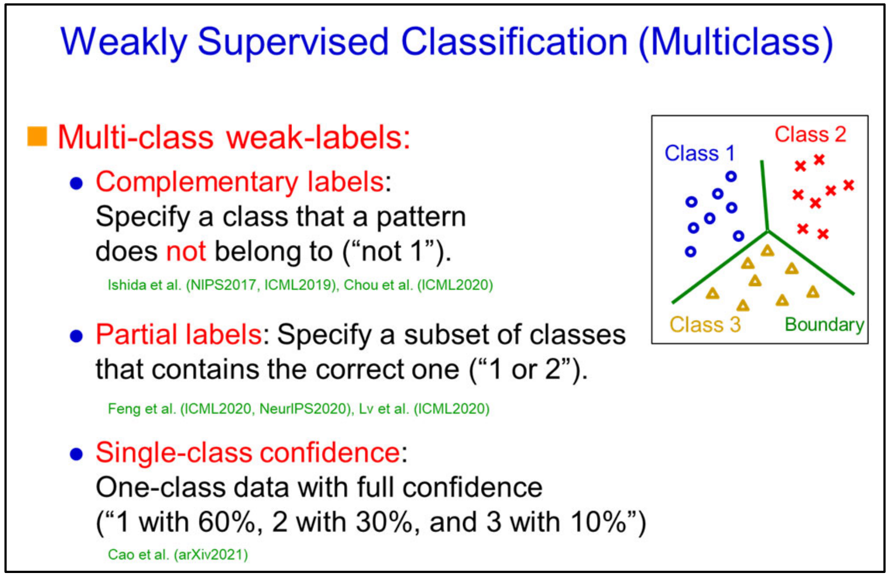
  - Compute the loss separately:

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

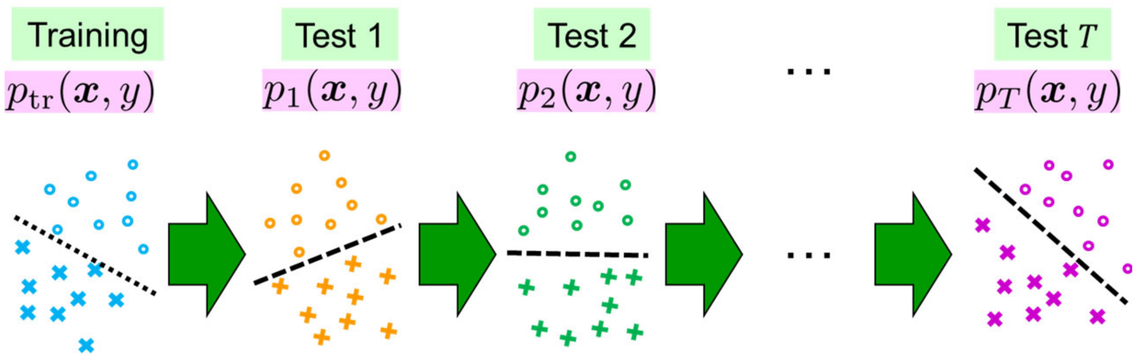
Ongoing Challenges

- For joint shift adaptation, requiring labeled test data is too strong.
  - Can we use weakly supervised learning?





- Can we handle sequential joint shift?



Towards Machine Learning with (Almost) No Assumptions

- So far:
  - Develop an algorithm with guarantee under some assumption.
  - If the assumption is correct, it works well with guarantee (but if not, there is no guarantee).
- In practice:
  - We don't know whether the assumption holds or not.
  - We try it and if we are lucky, it works well (if we are unlucky, we suffer…).
- Future challenge:
  - Develop an algorithm with minimum guarantee under (almost) no assumptions.
  - This is the first method we should use in practice.

Thank you!