Robust Machine Learning for Reliable Deployment

Masashi Sugiyama

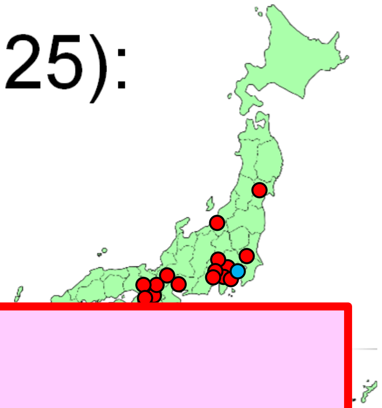
RIKEN Center for Advanced Intelligence Project/
The University of Tokyo

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

Slides:
https://drive.google.com/file/d/1Z0DjWKm5nYmYdhiJfpsWT3VSf8Liu6BI/

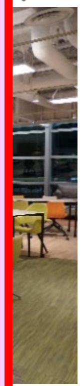
RIKEN Center for Advanced Intelligence Project (AIP)

- 10-year national project in Japan (2016-2025):
  - Develop next-generation AI technology (learning and optimization theory, etc.)

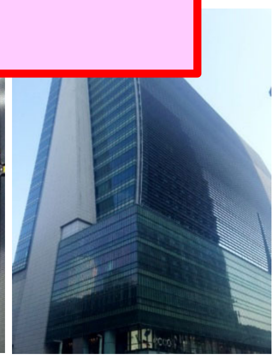


Imperfect Information Learning Team:

Develop novel ML theories and algorithms that enable accurate learning from limited information.



(150+ researchers, 200+ students, 150+ interns, 300+ visiting scientists, 40+ industry projects)

Imperfect Information Learning Team

- Members:
  - Gang Niu (Research Scientist): Learning theory
  - Voot Tangkaratt (Postdoc): Reinforcement learning
  - Shuo Chen (Postdoc): Metric learning
  - Jingfeng Zhang (Postdoc): Adversarial learning
  - Jiaqi Lyu (Postdoc): Weakly supervised learning
  - Many great Visiting Scientists,
    Junior Research Associates, Part-Timers,
    and Interns over the world!





Today's Topic: Robust Machine Learning

- In real-world applications, it becomes increasingly important to consider robustness against various factors:
  - Data bias: changing environments, privacy.
  - Insufficient information: weak supervision.
  - Label noise: human error, sensor error.
  - Attack: adversarial noise, distribution shift.

- In this talk, I will give an overview of our recent advances in robust machine learning.

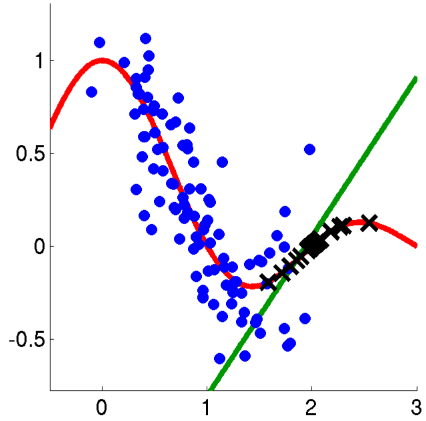
http://www.ms.k.u-tokyo.ac.jp/sugi/publications.html

Contents

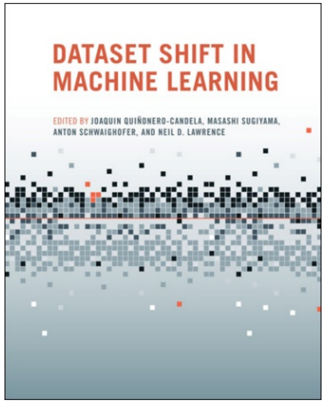
1. Transfer learning
2. Weakly supervised classification
3. Future outlook

Transfer Learning

- Training and test data often have different distributions, due to
  - changing environments,
  - sample selection bias (privacy).



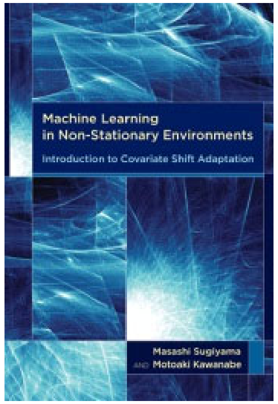
- Transfer learning (domain adaptation):
  - Train a test-domain predictor using training data from different domains.



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

(Edited volume from NIPS2006 Workshop on Learning When Test and Training Inputs Have Different Distributions)

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



Problem Setup

- **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

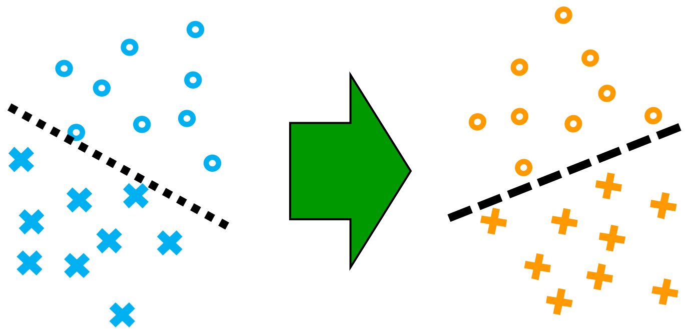
- **Goal:**
  - Train a predictor $y = f(\boldsymbol{x})$ that works well in the test domain (with some additional data from the test domain).

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

ℓ : loss function

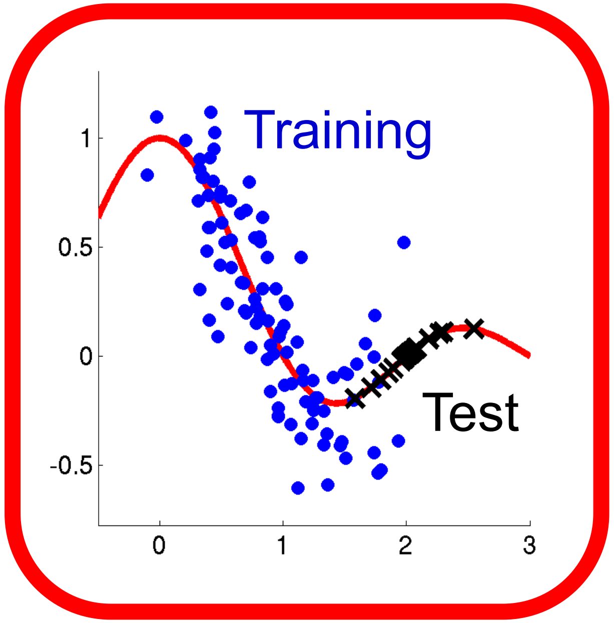
- **Challenge:**
  - Overcome changing distributions!

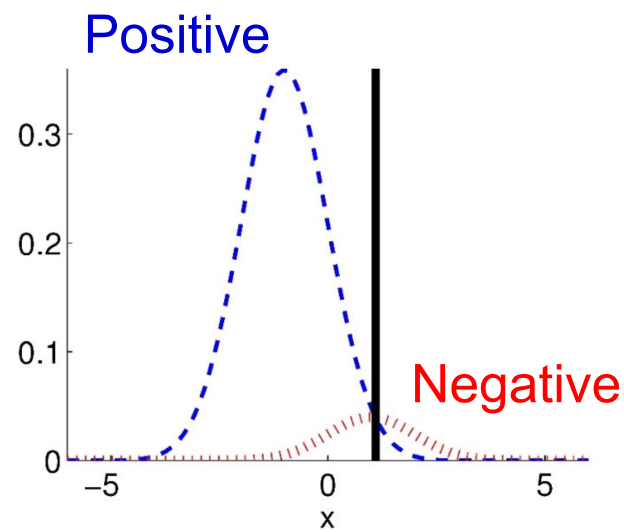
$$p_{\mathrm{tr}}(\boldsymbol{x}, y) \neq p_{\mathrm{te}}(\boldsymbol{x}, y)$$

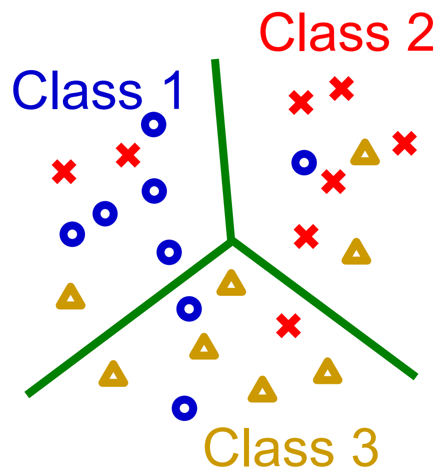


Various Scenarios

- Full-distribution shift: $p_{\mathrm{tr}}(\boldsymbol{x}, y) \neq p_{\mathrm{te}}(\boldsymbol{x}, y)$
- Covariate shift: $p_{\mathrm{tr}}(\boldsymbol{x}) \neq p_{\mathrm{te}}(\boldsymbol{x})$
- Class-prior/target 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: $p_{\mathrm{tr}}(\boldsymbol{x}|y) \neq p_{\mathrm{te}}(\boldsymbol{x}|y)$







Regression under Covariate Shift

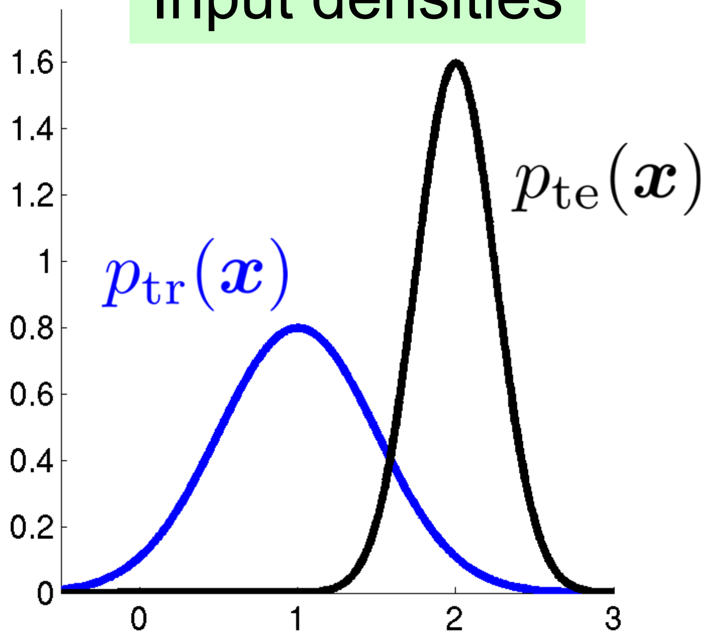
- **Covariate shift:** Shimodaira (JSPI2000)
  - Training and test input distributions are different:

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

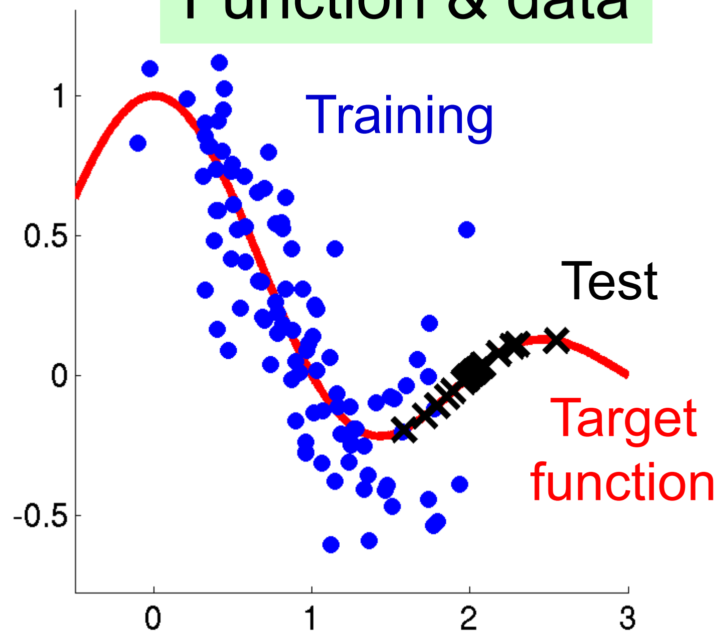
  - But the output-given-input distribution remains unchanged:

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

Input densities



Function & data

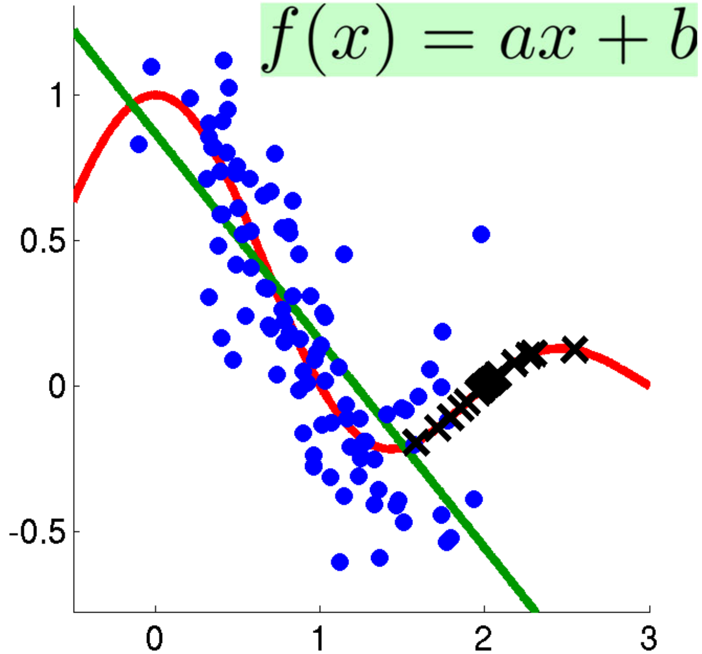


Empirical Risk Minimization (ERM)

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

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

- Generally, ERM is consistent:
  - Learned function converges to the optimal solution when $n_{\mathrm{tr}} \to \infty$.



- However, covariate shift makes ERM inconsistent:

$$\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \ell(f(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}}) \overset{n_{\mathrm{tr}} \to \infty}{\longrightarrow} \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x},y)}[\ell(f(\boldsymbol{x}), y)] \neq R(f)$$

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

Importance-Weighted ERM (IWERM)

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

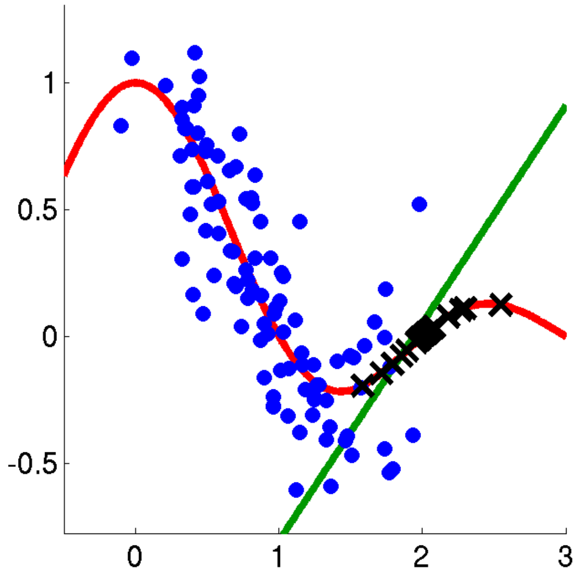
- IWERM is consistent even under covariate shift.

$$\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \ell(f(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}})$$

$$\overset{n_{\mathrm{tr}}\to\infty}{\longrightarrow} \mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x},y)} \left[\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} \ell(f(\boldsymbol{x}), y) \right]$$

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

$$f(x) = ax + b$$

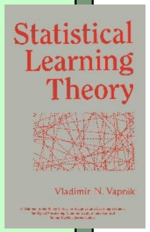


- How can we know the importance weight?

Importance Weight Estimation

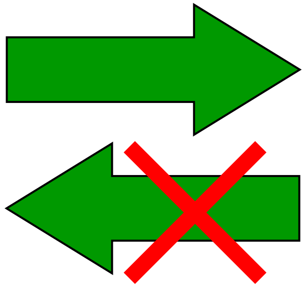
Vapnik's principle: Vapnik (Wiley, 1998)

When solving a problem of interest,
one should not solve a more general problem
as an intermediate step



Knowing densities

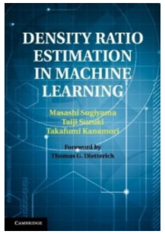
$$p_{\mathrm{te}}(\boldsymbol{x}), p_{\mathrm{tr}}(\boldsymbol{x})$$



Knowing ratio

$$r^*(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$$

- Estimating the density ratio is substantially easier than estimating both the densities!
- Various direct density-ratio estimators were developed.



Sugiyama, Suzuki & Kanamori,
Density Ratio Estimation
in Machine Learning
(Cambridge University Press, 2012)

Least-Squares Importance Fitting (LSIF)

Kanamori et al. (JMLR2009)

- Given training and test input data:

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

- Directly fit a model r to $r^*(\boldsymbol{x}) = \dfrac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$ by LS:

$$\min_r Q(r) \qquad Q(r) = \int \Big(r(\boldsymbol{x}) - r^*(\boldsymbol{x})\Big)^2 p_{\mathrm{tr}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x}$$

  - Empirical approximation:

$$Q(r) = \int r(\boldsymbol{x})^2 p_{\mathrm{tr}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} - 2\int r(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} + C$$

$$\approx \frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}} r(\boldsymbol{x}_i^{\mathrm{tr}})^2 - \frac{2}{n_{\mathrm{te}}}\sum_{j=1}^{n_{\mathrm{te}}} r(\boldsymbol{x}_j^{\mathrm{te}}) + C$$

From Two-Step Adaptation to One-Step Adaptation

- The classical approaches are two steps:
  1. Weight estimation (e.g., LSIF):

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

  2. Weighted predictor training (e.g., IWERM):

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

- Can we integrate these two steps?

Contents

1. Transfer learning
   A) Joint upper-bound minimization
   B) Dynamic importance weighting
2. Weakly supervised classification
3. Future outlook

Joint Upper-Bound Minimization

Zhang et al. (ACML2020, SNCS2021)

- Suppose we are 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: $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x})$
- Goal: We want to minimize the test risk.

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

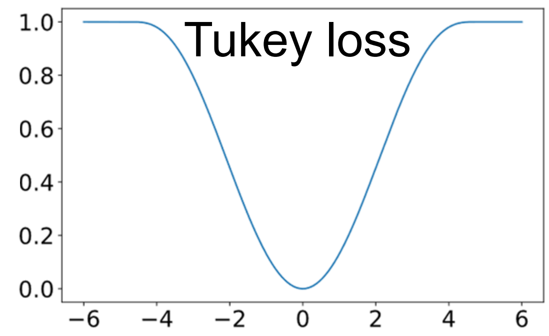
ℓ : evaluation loss

- We use two losses $\ell(\leq 1), \ell'(\geq \ell)$.

ℓ' : surrogate loss

For example:

- ℓ : 0/1, ℓ' : hinge or softmax cross-entropy (classification)
- ℓ : Tukey, ℓ': squared (regression)



Risk Upper-Bounding (cont.)

Zhang et al. (ACML2020, SNCS2021)

- For $\ell \leq 1, \ell' \geq \ell, r \geq 0$,
  the test risk is upper-bounded as

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

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

$$J_{\ell'}(r, f) = (\mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x},y)}[r(\boldsymbol{x})\ell'(f(\boldsymbol{x}), y)])^2 \quad \leftarrow \text{IWERM}$$

$$+\mathbb{E}_{p_{\mathrm{tr}}(\boldsymbol{x})}[(r(\boldsymbol{x}) - r^*(\boldsymbol{x}))^2] \quad \leftarrow \text{LSIF}$$

- In terms of this upper-bound minimization,
  2-step (LSIF followed by IWERM) is not optimal:
- Let's directly minimize the upper bound w.r.t. r, f !

Theoretical Analysis

- Under some mild conditions, the test risk of the empirical solution $\widehat{f} = \mathop{\mathrm{argmin}}_{f} \min_{r} \widehat{J}_{\ell'}(r, f)$ is upper-bounded as

$$R_\ell(\widehat{f}) \le \sqrt{2} \min_{f} R_{\ell'}(f) + \mathcal{O}_p(n_{\mathrm{tr}}^{-1/4} + n_{\mathrm{te}}^{-1/4})$$

$$\widehat{J}_{\ell'}(r, f) = \left(\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} r(\boldsymbol{x}_i^{\mathrm{tr}}) \ell'(f(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}}) \right)^2 + \left(\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} r(\boldsymbol{x}_i^{\mathrm{tr}})^2 - \frac{2}{n_{\mathrm{te}}} \sum_{j=1}^{n_{\mathrm{te}}} r(\boldsymbol{x}_j^{\mathrm{tr}}) + C \right)$$

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

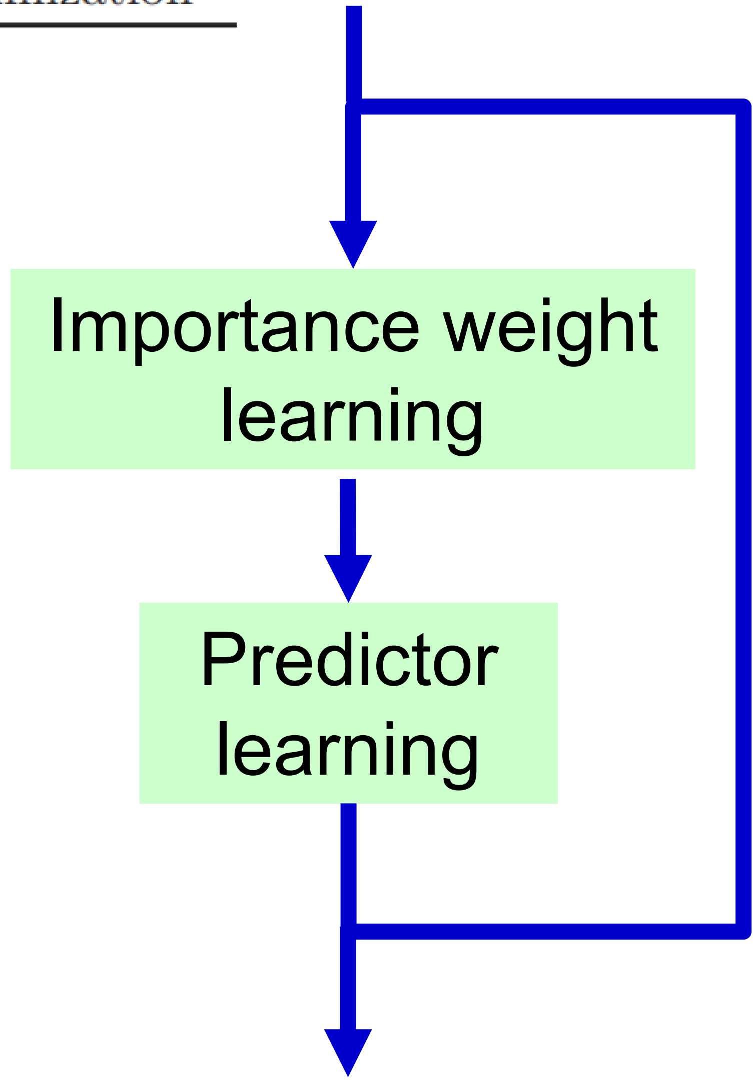
$$R_\ell(\widehat{f}) = \mathbb{E}_{p_{\mathrm{te}}(\boldsymbol{x}, y)}[\ell(\widehat{f}(\boldsymbol{x}), y)]$$

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

Practical Implementation

Algorithm 2 Gradient-based Alternating Minimization

$\mathcal{Z}^{\mathrm{tr}}, \mathcal{X}^{\mathrm{te}} \leftarrow \{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}, \{\boldsymbol{x}_i^{\mathrm{te}}\}_{i=1}^{n_{\mathrm{te}}}$
$\mathcal{A} \leftarrow$ a gradient-based optimizer
$\boldsymbol{f} \leftarrow$ an arbitrary classifier
for round $= 0, 1, \ldots, \mathrm{numOfRounds} - 1$ **do**
  for epoch $= 0, 1, \ldots, \mathrm{numOfEpochsForG} - 1$ **do**
    for $i = 0, 1, \ldots, \mathrm{numOfMiniBatches} - 1$ **do**
      $\mathcal{Z}_i^{\mathrm{tr}}, \mathcal{X}_i^{\mathrm{te}} \leftarrow \mathrm{sampleMiniBatch}(\mathcal{Z}^{\mathrm{tr}}, \mathcal{X}^{\mathrm{te}})$
      $g \leftarrow \mathcal{A}(g, \nabla_g \widehat{J}_{\mathrm{UB}}(\boldsymbol{f}, g; \mathcal{Z}_i^{\mathrm{tr}} \cup \mathcal{X}_i^{\mathrm{te}}))$
    end for
  end for
  for epoch $= 0, 1, \ldots, \mathrm{numOfEpochsForF} - 1$ **do**
    for $i = 0, 1, \ldots, \mathrm{numOfMiniBatches} - 1$ **do**
      $\mathcal{Z}_i^{\mathrm{tr}} \leftarrow \mathrm{sampleMiniBatch}(\mathcal{Z}^{\mathrm{tr}})$
      $w_j \leftarrow \max(g(\boldsymbol{x}_j), 0), \ \forall (\boldsymbol{x}_j, \cdot) \in \mathcal{Z}_i^{\mathrm{tr}}$
      $w_j \leftarrow w_j / \sum_j w_j, \ \forall j$
      $L_i \leftarrow \sum_{(\boldsymbol{x}_j, y_j) \in \mathcal{Z}_i^{\mathrm{tr}}} w_j \ell_{\mathrm{UB}}(\boldsymbol{f}(\boldsymbol{x}_j), y_j)$
      $\boldsymbol{f} \leftarrow \mathcal{A}(\boldsymbol{f}, \nabla_{\boldsymbol{f}} L_i)$
    end for
  end for
end for



Experimental Evaluation

Table 3 Mean test classification accuracy averaged over 5 trials on image datasets with neural networks. The numbers in the brackets are the standard deviations. For each dataset, the best method and comparable ones based on the *paired t-test* at the significance level 5% are described in bold face.

Dataset	Shift Level (a, b)	ERM	EIWERM	RIWERM	one-step
Fashion-MNIST	(2, 4)	81.71(0.17)	84.02(0.18)	84.12(0.06)	**85.07(0.08)**
	(2, 5)	72.52(0.54)	76.68(0.27)	77.43(0.29)	**78.83(0.20)**
	(2, 6)	60.10(0.34)	65.73(0.34)	66.73(0.55)	**69.23(0.25)**
Kuzushiji-MNIST	(2, 4)	77.09(0.18)	80.92(0.32)	81.17(0.24)	**82.45(0.12)**
	(2, 5)	65.06(0.26)	71.02(0.50)	72.16(0.19)	**74.03(0.16)**
	(2, 6)	51.24(0.30)	58.78(0.38)	60.14(0.93)	**62.70(0.55)**

Shimodaira (JSPI2000)

Yamada et al. (NIPS2011, NeCo2013)

Contents

1. Transfer learning
   A) Joint upper-bound minimization
   B) Dynamic importance weighting
2. Weakly supervised classification
3. Future outlook

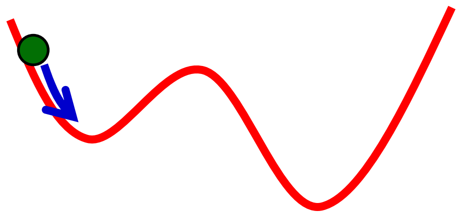
Dynamic Importance Weighting

Fang et al. (NeurIPS2020)

- Deep learning adopts stochastic optimization:

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

$\eta > 0$: Learning rate



- Let's learn
  - Importance weight r
  - predictor f

dynamically in the mini-batch-wise manner.

Mini-Batch-Wise Loss Matching

- Suppose we are 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: $\{(\boldsymbol{x}_j^{\mathrm{te}}, y_j^{\mathrm{te}})\}_{j=1}^{n_{\mathrm{te}}} \overset{\mathrm{i.i.d.}}{\sim} p_{\mathrm{te}}(\boldsymbol{x}, y)$

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

Huang, et al. (NeurIPS2007)

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

- No covariate shift assumption is needed!

Practical Implementation

Algorithm 1 Dynamic importance weighting (in a mini-batch).

Require: a training mini-batch $\mathcal{S}^{\mathrm{tr}}$, a validation mini-batch $\mathcal{S}^{\mathrm{v}}$, the current model $\boldsymbol{f}_{\theta_t}$

1: `forward` the input parts of $\mathcal{S}^{\mathrm{tr}}$ & $\mathcal{S}^{\mathrm{v}}$
2: `compute` the loss values as $\mathcal{L}^{\mathrm{tr}}$ & $\mathcal{L}^{\mathrm{v}}$
3: `match` $\mathcal{L}^{\mathrm{tr}}$ & $\mathcal{L}^{\mathrm{v}}$ to obtain $\mathcal{W}$
4: `weight` the empirical risk $\widehat{R}(\boldsymbol{f}_\theta)$ by $\mathcal{W}$
5: `backward` $\widehat{R}(\boldsymbol{f}_\theta)$ and update θ

Experimental Evaluation

Table 4: Mean accuracy (standard deviation) in percentage on Fashion-MNIST (F-MNIST for short), CIFAR-10/100 under label noise (5 trials). Best and comparable methods (paired t-test at significance level 5%) are highlighted in bold. p/s is short for pair/symmetric flip.

	Noise	Clean	Uniform	Random	IW	Reweight	DIW
F-MNIST	0.3 p	71.05 (1.03)	76.89 (1.06)	84.62 (0.68)	82.69 (0.38)	**88.74 (0.19)**	88.19 (0.43)
	0.4 s	73.55 (0.80)	77.13 (2.21)	84.58 (0.76)	80.54 (0.66)	85.94 (0.51)	**88.29 (0.18)**
	0.5 s	73.55 (0.80)	73.70 (1.83)	82.49 (1.29)	78.90 (0.97)	84.05 (0.51)	**87.67 (0.57)**
CIFAR-10	0.3 p	45.62 (1.66)	77.75 (3.27)	83.20 (0.62)	45.02 (2.25)	82.44 (1.00)	**84.44 (0.70)**
	0.4 s	45.61 (1.89)	69.59 (1.83)	76.90 (0.43)	44.31 (2.14)	76.69 (0.57)	**80.40 (0.69)**
	0.5 s	46.35 (1.24)	65.23 (1.11)	71.56 (1.31)	42.84 (2.35)	72.62 (0.74)	**76.26 (0.73)**
CIFAR-100	0.3 p	10.82 (0.44)	50.20 (0.53)	48.65 (1.16)	10.85 (0.59)	48.48 (1.52)	**53.94 (0.29)**
	0.4 s	10.82 (0.44)	46.34 (0.88)	42.17 (1.05)	10.61 (0.53)	42.15 (0.96)	**53.66 (0.28)**
	0.5 s	10.82 (0.44)	41.35 (0.59)	34.99 (1.19)	10.58 (0.17)	36.17 (1.74)	**49.13 (0.98)**

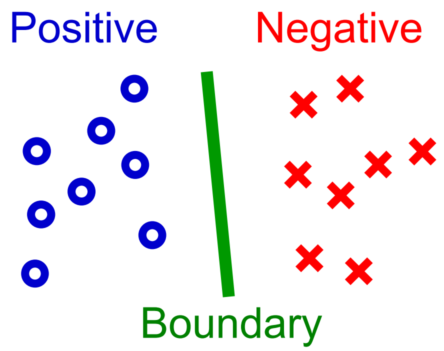
Contents

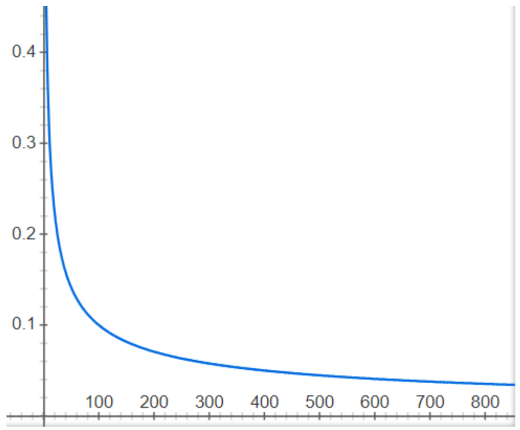
1. Transfer learning
2. Weakly supervised classification
3. Future outlook

ML from Limited Data

- ML from big labeled data is successful.
  - Speech, image, language, advertisement,…
  - Estimation error of the boundary decreases in order $1/\sqrt{n}$.

n : Number of labeled samples



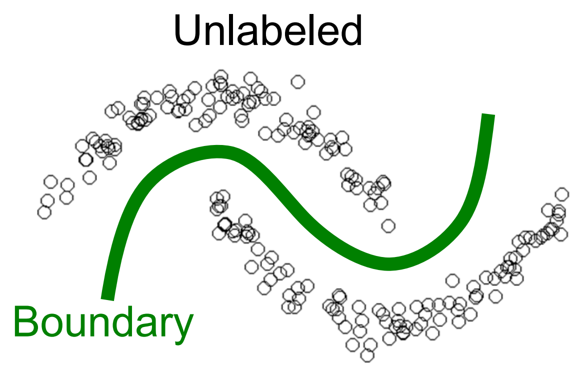


- However, there are various applications where big labeled data is not available.
  - Medicine, disaster, robots, brain, …

Alternatives to Supervised Classification

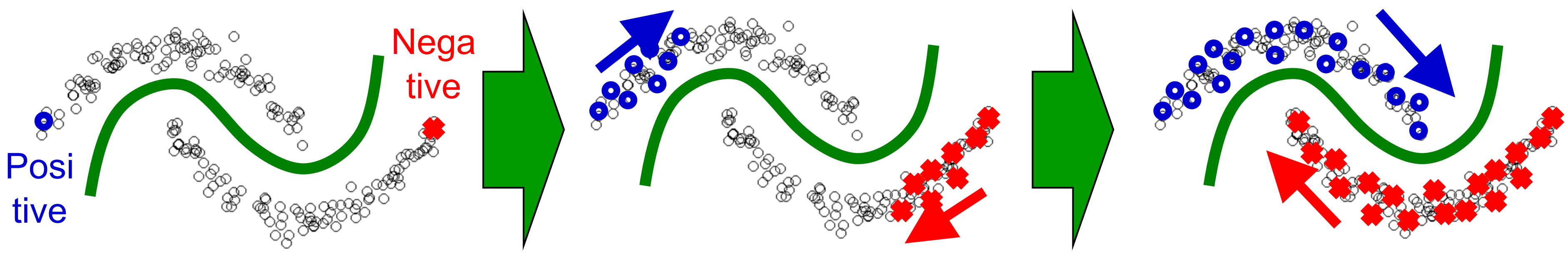
Unsupervised classification:

- No label is used.
- Essentially clustering.
- No guarantee for prediction.



Semi-supervised classification:

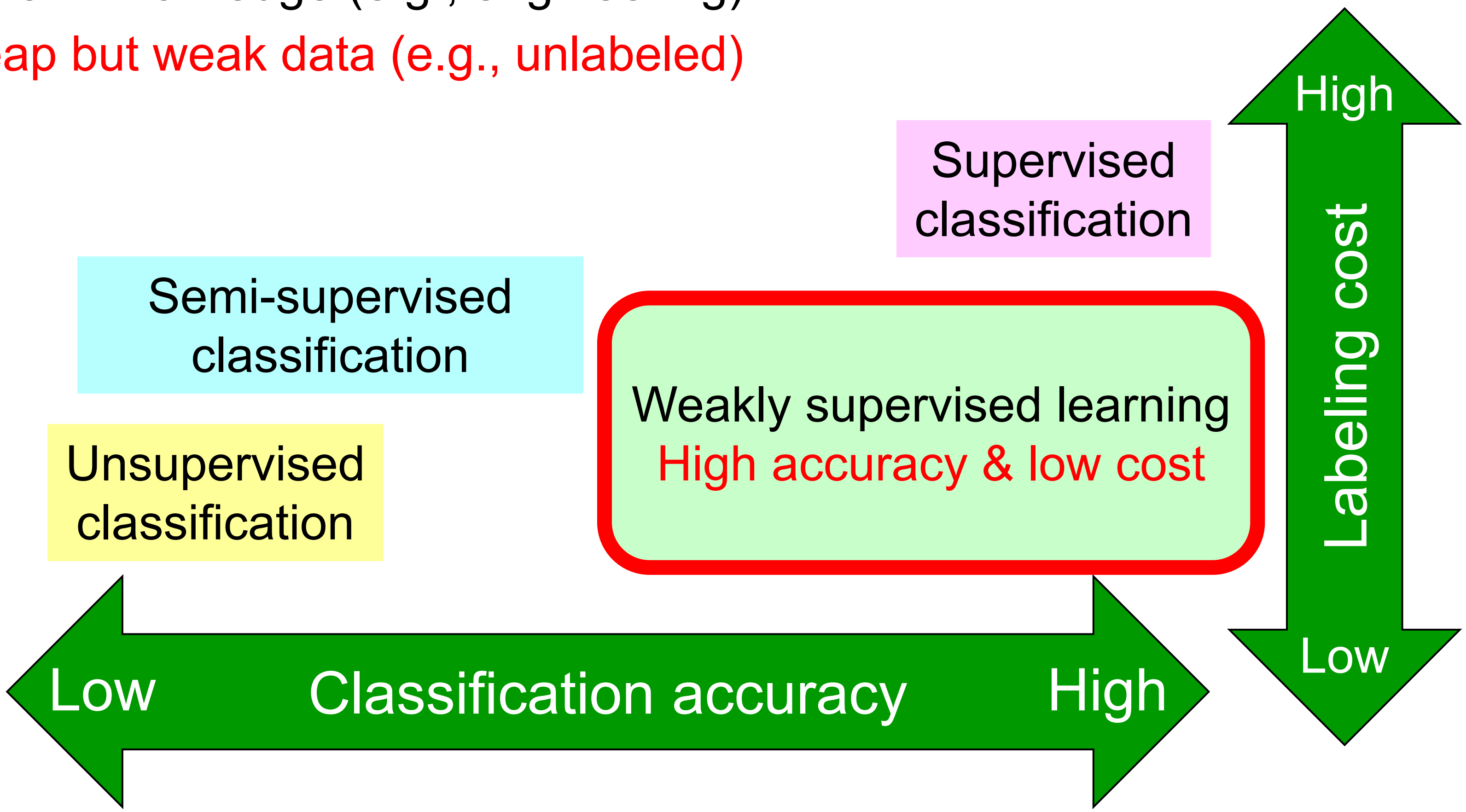
- Additionally use a small amount of labeled data.
- Propagate labels along clusters.
- No guarantee for prediction.



Weakly Supervised Learning

Coping with labeling cost:

- Improve data collection (e.g., crowdsourcing)
- Use a simulator to generate pseudo data (e.g., physics, chemistry, robotics, etc.)
- Use domain knowledge (e.g., engineering)
- Use cheap but weak data (e.g., unlabeled)



Contents

1. Transfer learning
2. Weakly supervised classification
   - A) Positive-unlabeled classification
   - B) Extensions
3. Future outlook

Positive-Unlabeled Classification

- Given: Positive and unlabeled samples

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

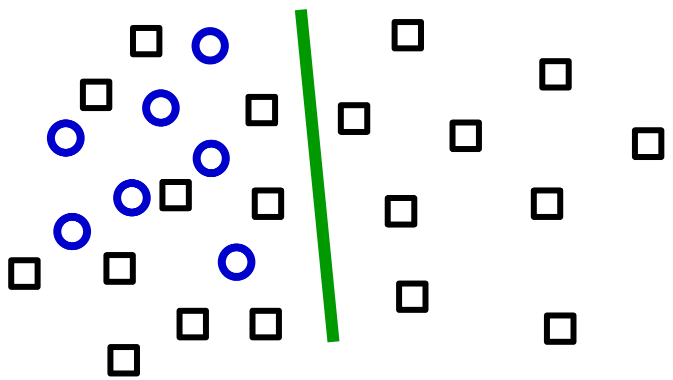
$$\{\boldsymbol{x}_j^{\mathrm{U}}\}_{j=1}^{n_{\mathrm{U}}} \overset{\text{i.i.d.}}{\sim} p(\boldsymbol{x})$$

- Goal: Obtain a PN classifier

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



Unlabeled (mixture of positives and negatives)

PN Risk Decomposition

- Risk of classifier f :

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

ℓ : loss function

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

$\pi = p(y = +1)$: Class-prior probability
(assumed known; can be estimated)

Scott & Blanchard (AISTATS2009)
Blanchard et al. (JMLR2010)
du Plessis et al. (IEICE2014, MLJ2017)
Ramaswamy et al. (ICML2016)
Yao et al. (arXiv2020)

- Since we do not have N data in the PU setting, the risk cannot be directly estimated.

PU Risk Estimation

du Plessis et al. (ICML2015)

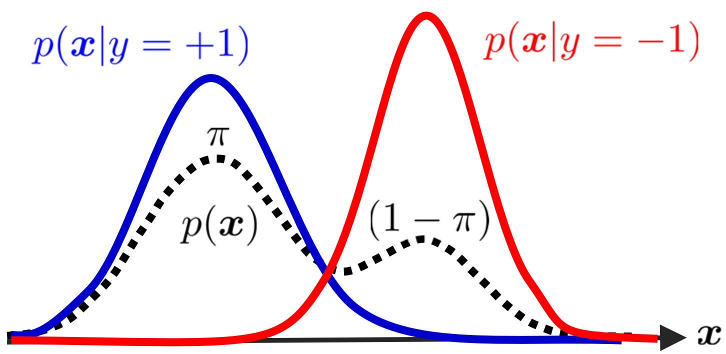
$$R(f) = \pi \mathbb{E}_{p(\boldsymbol{x}|y=+1)}\Big[\ell\Big(f(\boldsymbol{x})\Big)\Big] + (1-\pi)\mathbb{E}_{p(\boldsymbol{x}|y=-1)}\Big[\ell\Big(-f(\boldsymbol{x})\Big)\Big]$$

- U-density is a mixture of P- and N-densities:

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

- This allows us to eliminate the N-density:

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



PU Empirical Risk Minimization

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

- Replacing expectations by sample averages gives an empirical risk:

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

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

- Optimal convergence rate is attained:

Niu et al. (NIPS2016)

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

with probability $1-\delta$

$$\widehat{f}_{\mathrm{PU}} = \mathrm{argmin}_f\, \widehat{R}_{\mathrm{PU}}(f)$$

$$f^* = \mathrm{argmin}_f\, R(f)$$

$n_{\mathrm{P}}, n_{\mathrm{U}}$: # of P, U samples

Theoretical Comparison with PN

Niu et al. (NIPS2016)

- Estimation error bounds for PU and PN:

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

$$R(\widehat{f}_{\mathrm{PN}}) - R(f^*) \leq C(\delta)\left(\frac{\pi}{\sqrt{n_{\mathrm{P}}}} + \frac{1-\pi}{\sqrt{n_{\mathrm{N}}}}\right)$$

$$\widehat{f}_{\mathrm{PN}} = \operatorname*{argmin}_f \widehat{R}_{\mathrm{PN}}(f)$$

with probability $1-\delta$

$$\widehat{R}_{\mathrm{PN}}(f) = \frac{1}{n}\sum_{i=1}^{n} \ell\Big(y_i f(\boldsymbol{x}_i)\Big)$$

$n_{\mathrm{P}}, n_{\mathrm{N}}, n_{\mathrm{U}}$: # of P, N, U samples

- Comparison: PU bound is smaller than PN if

$$\frac{\pi}{\sqrt{n_{\mathrm{P}}}} + \frac{1}{\sqrt{n_{\mathrm{U}}}} < \frac{1-\pi}{\sqrt{n_{\mathrm{N}}}}$$

  - PU can be better than PN, provided many PU data!

Further Correction

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

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

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

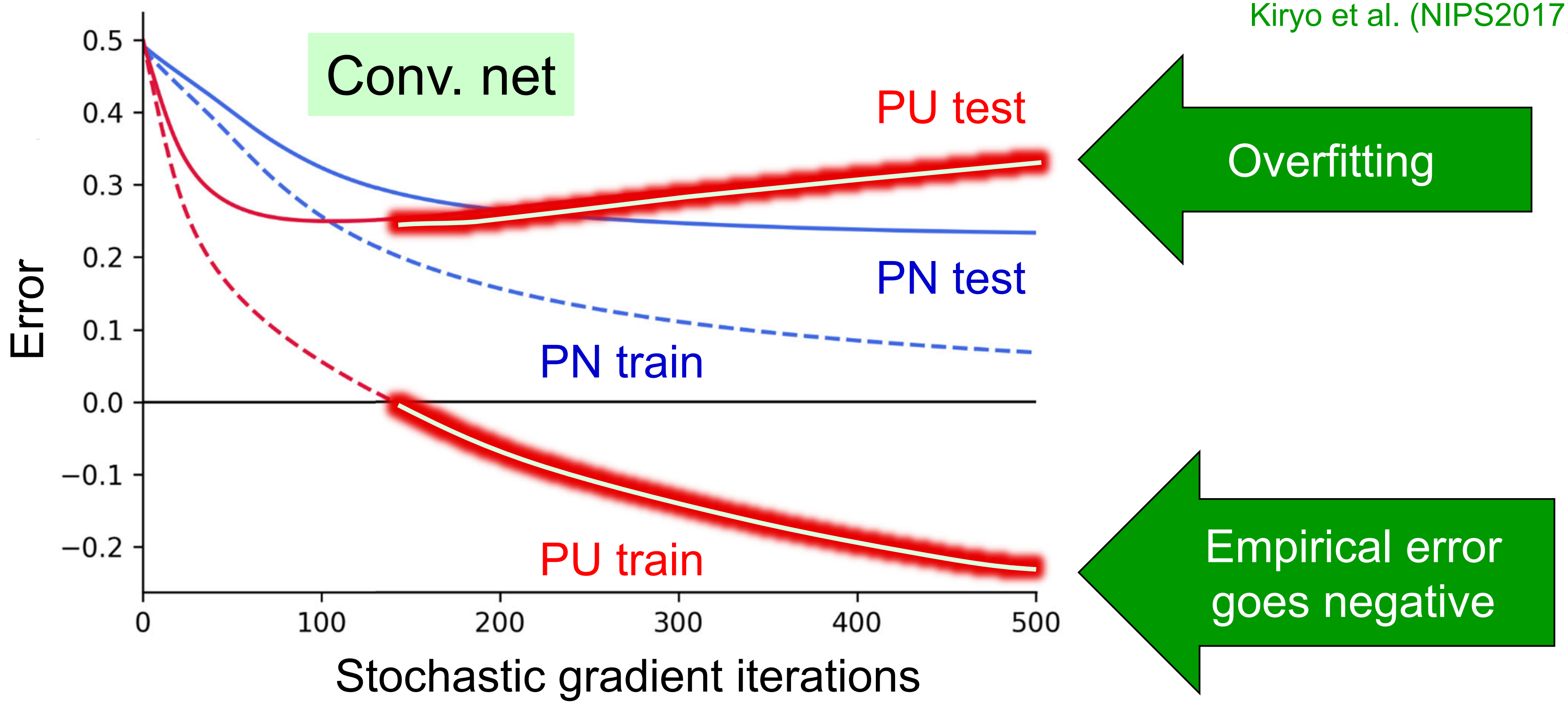
  - If $\ell(m) \geq 0, \ \forall m$ $\quad R^-(f) \geq 0$
  - However, its PU empirical approximation can be negative due to "difference of approximations".

$$\widehat{R}^-_{\mathrm{PU}}(f) = \frac{1}{n_{\mathrm{U}}}\sum_{i=1}^{n_{\mathrm{U}}} \ell\Big(-f(\boldsymbol{x}_i^{\mathrm{U}})\Big) - \frac{\pi}{n_{\mathrm{P}}}\sum_{i=1}^{n_{\mathrm{P}}} \ell\Big(-f(\boldsymbol{x}_i^{\mathrm{P}})\Big) \not\geq 0$$

  - This problem is more critical for flexible models such as deep neural networks.

Non-Negative PU Classification

Kiryo et al. (NIPS2017)



- We constrain the sample approximation term to be non-negative through back-prop training:

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

- This risk estimator is biased. Is it really good?

Theoretical Analysis

Kiryo et al. (NIPS2017)

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

- $\widetilde{R}_{\mathrm{PU}}(f)$ is still consistent and its bias decreases exponentially: $\mathcal{O}(e^{-n_{\mathrm{P}}-n_{\mathrm{U}}})$

  $n_{\mathrm{P}}, n_{\mathrm{U}}$: # of P, U samples

  - In practice, we can ignore the bias of $\widetilde{R}_{\mathrm{PU}}(f)$!

- Mean-squared error of $\widetilde{R}_{\mathrm{PU}}(f)$ is not more than the original one:
  - In practice, $\widetilde{R}_{\mathrm{PU}}(f)$ is more reliable!

- Risk of $\operatorname{argmin}_f \widetilde{R}_{\mathrm{PU}}(f)$ for linear models attains the optimal convergence rate: $\mathcal{O}_p\left(\frac{1}{\sqrt{n_{\mathrm{P}}}}+\frac{1}{\sqrt{n_{\mathrm{U}}}}\right)$
  - Learned function is still optimal.

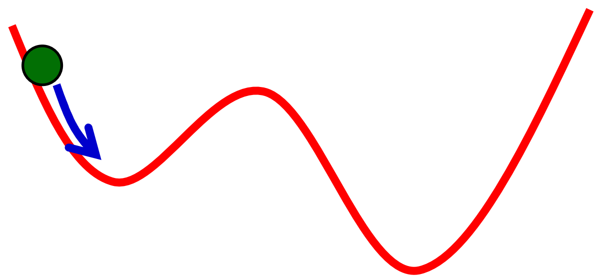
Practical Implementation for Deep Learning

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

- Use mini-batch stochastic gradient optimization:
  - If $\widehat{R}_{\mathrm{PU}}^{-}(f) \geq 0$, perform gradient descent as usual.
  - If $\widehat{R}_{\mathrm{PU}}^{-}(f) < 0$, perform gradient ascent:

For poor mini-batch data,

- Step back the gradient to avoid converging to a poor local optimum
- and recompute the gradient with a new mini-batch.

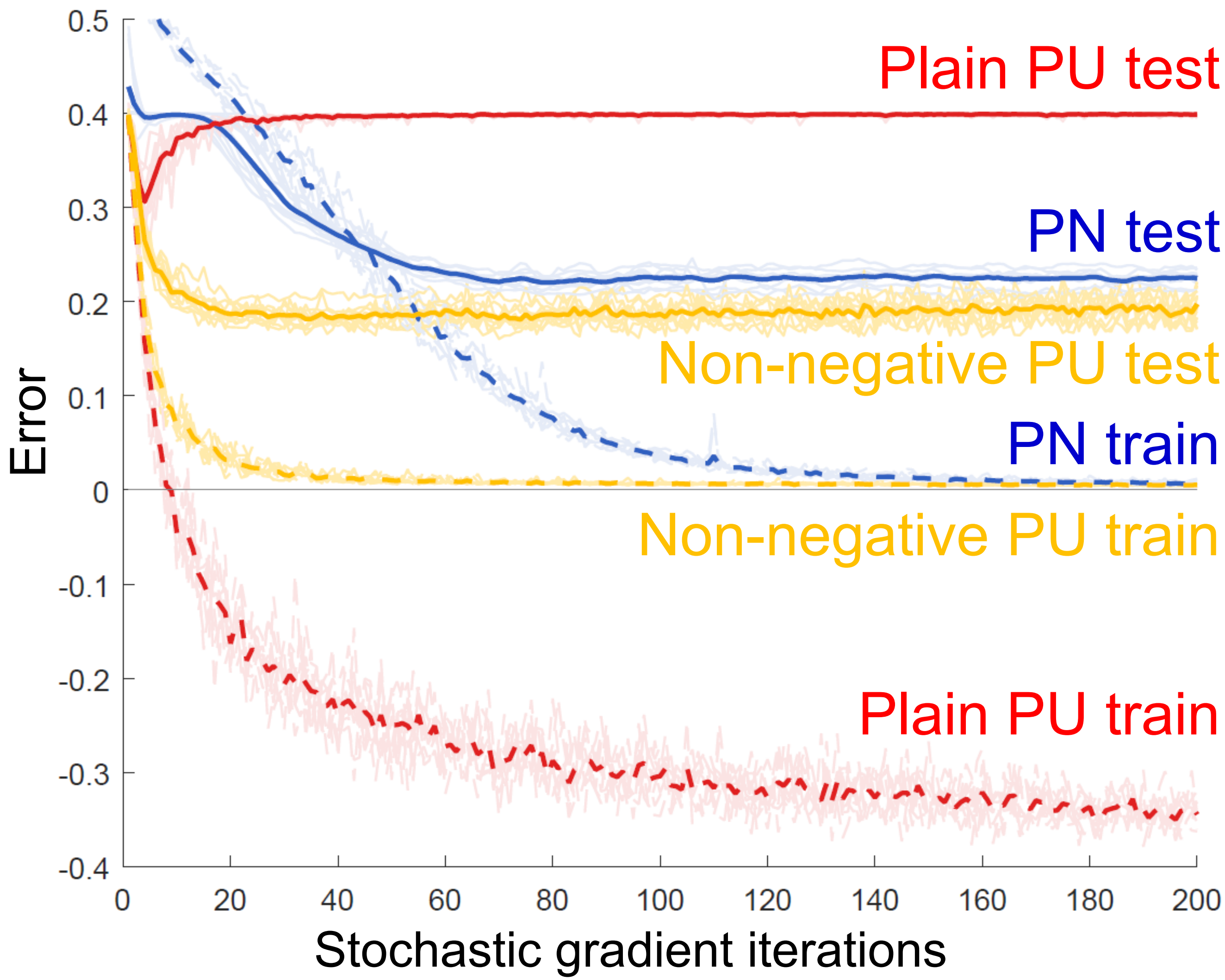


Experiments

- With a large number of unlabeled data, non-negative PU can even outperform PN!

- Binary CIFAR-10:
  Positive (airplane, automobile, ship, truck)
  Negative (bird, cat, deer, dog, frog, horse)
- 13-layer CNN with ReLU

$n_{\mathrm{P}} = 1000$
$n_{\mathrm{U}} = 50000$
$\pi = 0.4$



Summary

- Risk-rewriting: Rewrite the classification risk only in terms of weak data.

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

  - Standard empirical risk minimization formulation.
  - Optimal convergence guarantee.
  - Compatible with any loss, regularization, model, and optimizer.
  - Applicable to various weak data (shown next).
- Non-negative risk correction: Utilize intrinsic non-negativity to mitigate overfitting.
  - Non-negativity of loss, convexity, etc.
  - Applicable to various weak data. Lu et al. (ICLR2019)
  - Applicable to noisy-label learning. Han et al. (ICML2020)

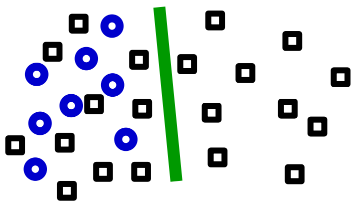
Contents

1. Transfer learning
2. Weakly supervised classification
   - A) Positive-unlabeled classification
   - B) Extensions
3. Future outlook

Various Binary Weak Labels

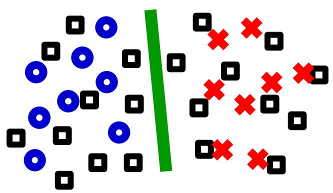
- Various weakly supervised classification problems can be solved by risk-rewriting systematically!

Positive-Unlabeled (PU)
(ex: click prediction)



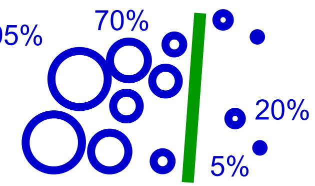
du Plessis et al. (NIPS2014, ICML2015, MLJ2017)
Niu et al. (NIPS2016),
Kiryo et al. (NIPS2017)
Hsieh et al. (ICML2019)

Semi-Supervised (PU+PN)
(first theoretically guaranteed method)



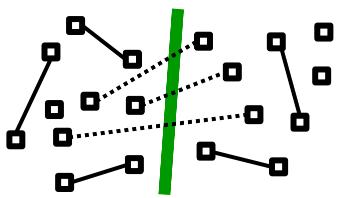
Sakai et al. (ICML2017, ML2018)

Positive-confidence (Pconf)
(ex: purchase prediction)



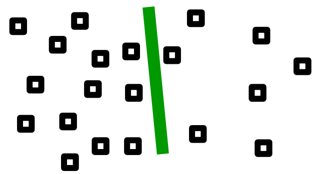
Ishida et al. (NeurIPS2018)
Shinoda et al. (IJCAI2021)

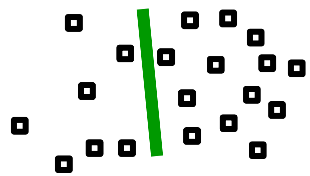
Similar-Dissimilar (SD)
(delicate information)



Bao et al. (ICML2018)
Shimada et al. (NeCo2021)
Dan et al. (ECMLPKDD2021)
Cao et al. (ICML2021)
Feng et al. (ICML2021)

Unlabeled-Unlabeled (UU)
(learning from different populations)

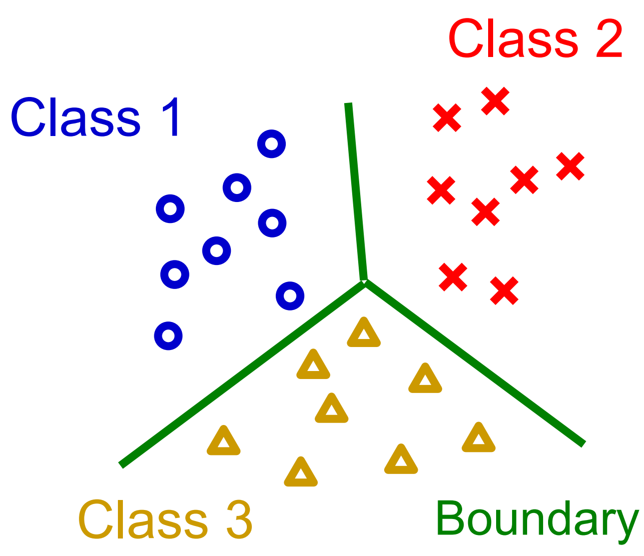




du Plessis et al.,(TAAI2013)
Lu et al. (ICLR2019, AISTATS2020)
Charoenphakdee et al. (ICML2019)
Lei et al. (ICML2021)

Multiclass Methods

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



- Risk rewriting is still possible in multi-class problems!
- Multi-class weak-labels:
  - Complementary labels: Specify a class that a pattern does not belong to ("not 1").

$1/\sqrt{n}$

Ishida et al. (NIPS2017, ICML2019), Chou et al. (ICML2020)

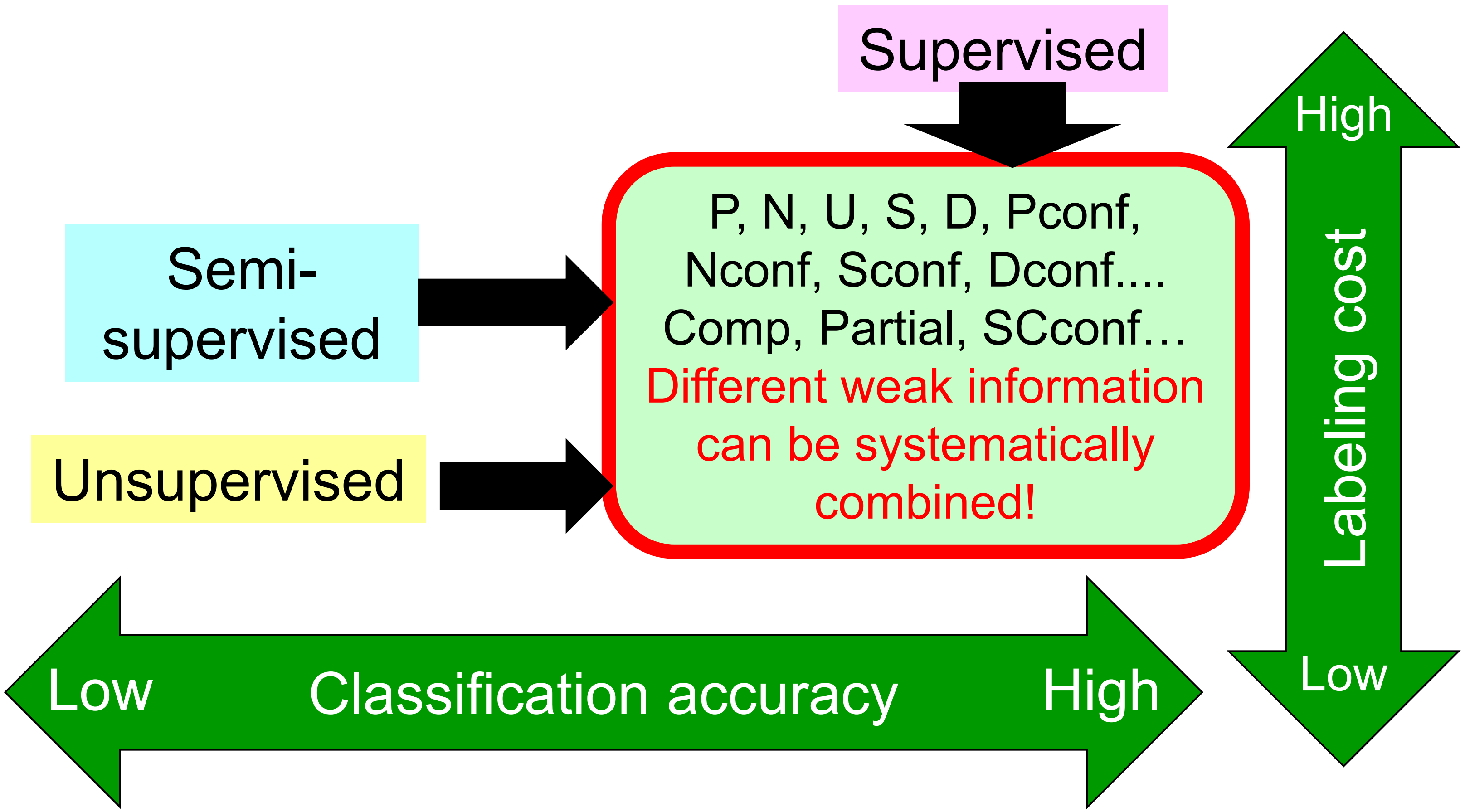
  - Partial labels: Specify a subset of classes that contains the correct one ("1 or 2").

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)

Summary: Empirical Risk Minimization Framework for Weakly Supervised Learning



Sugiyama, Bao, Ishida, Lu, Sakai & Niu,
Machine Learning from Weak Supervision,
MIT Press, in Press.

Contents

1. Transfer learning
2. Weakly supervised classification
3. Future outlook

Challenges in Reliable Machine Learning

- Reliability for expectable situations:
  - Model the corruption process explicitly and correct the solution.
    - How to handle modeling error?
- Reliability for unexpected situations:
  - Consider worst-case robustness (“min-max”).
    - How to make it less conservative?
  - Include human support (“rejection”).
    - How to handle real-time applications?
- Exploring somewhere in the middle would be practically more useful:
  - Use partial knowledge of the corruption process.

Axes of ML Research

Learning Method

Noise-robust
Adversarial
Transfer
Reinforcement
Weakly supervised
Semi-supervised
Unsupervised
Supervised

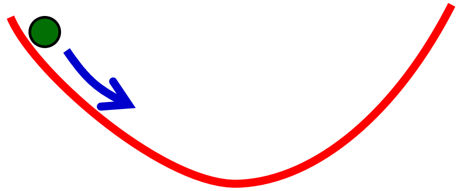
- Decomposing ML research into conceptually orthogonal topics:
  - Model
  - Learning method
  - Regularizer
  - Optimizer
  - …

Model

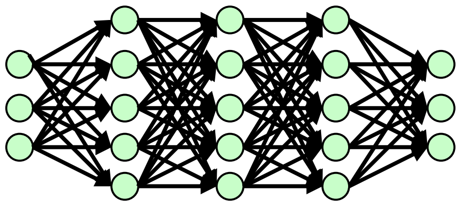
Linear | Additive | Kernel | Deep

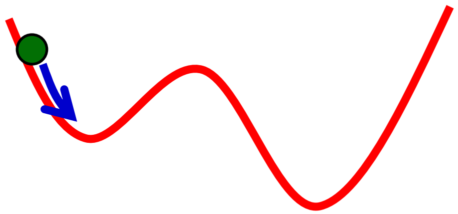
Theory ⟷ Application

Technological Breakthroughs



- Classical convex learning methods allow us to analyze the global solution.
- Since optimization in deep learning is complex, stochastic gradient descent is used.





- Thanks to the "gradual learning" nature, we can utilized intermediate learning results:
  - Strengthening supervision for weakly supervised learning.
  - Dynamic importance weighting for transfer learning.
  - Dynamic noise transition estimation for noise-robust learning.
  - Co-teaching for noise-robust learning.

Internship at RIKEN-AIP

- Many of the results introduced todays were first-authored by internship students.
- We wanted to invite interns also from EPFL, if their supervisors allow.
  - However, due to travel restrictions by COVID-19, our internship program has been suspended.
- The only possibility now is an informal internship:
  - Only remote collaboration.
  - No allowance, no official affiliation (too bad…).
- But if you are still interested, please let us know through your supervisor!

Thank you very much!