When Deep Learning Meets Weakly-Supervised Learning

Gang Niu

Project Assistant Professor, Univ. of Tokyo
Visiting Scientist, RIKEN AIP

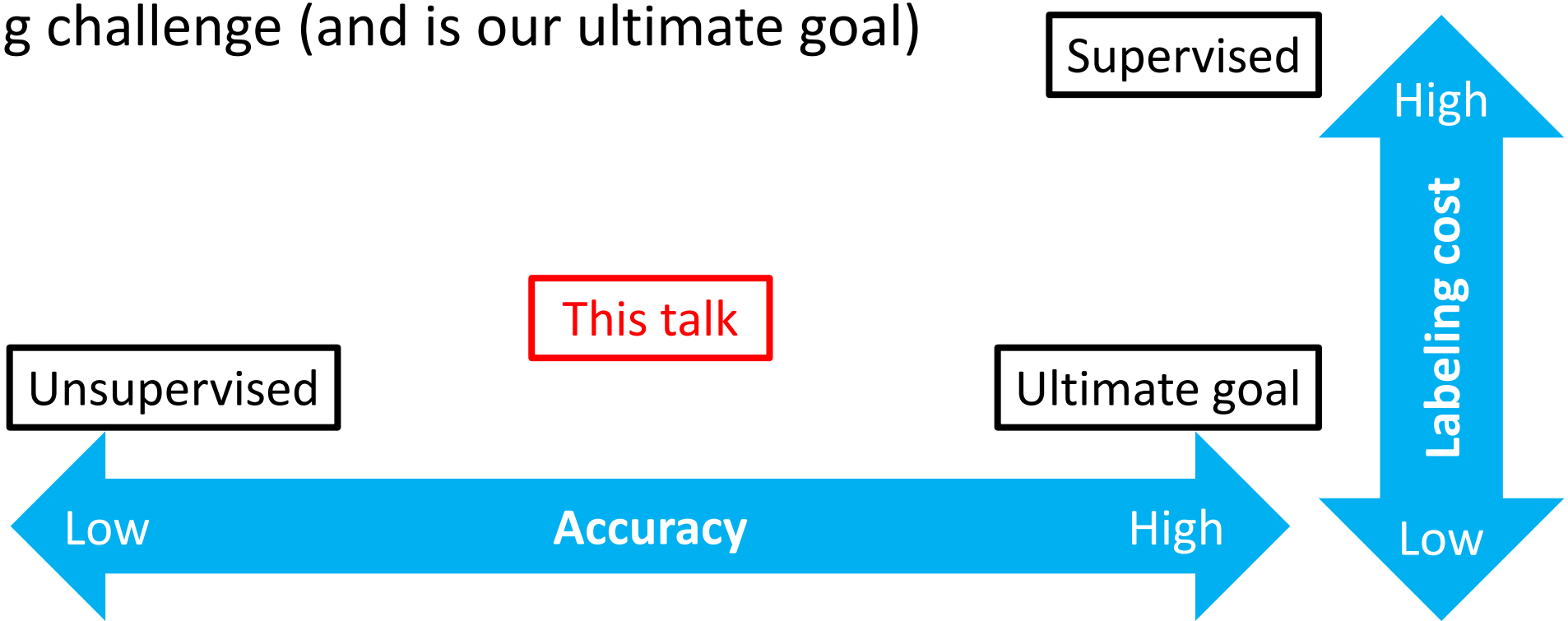
Joint work with Masashi Sugiyama,
Marthinus C. du Plessis, and many students

東京大学
THE UNIVERSITY OF TOKYO

RIKEN

About this talk

- Fully supervised deep learning from big data is successful
- Nevertheless, massive labeled data is not always available
  - Medicine, manufacturing, disaster, infrastructure …
- Achieving high **accuracy** with low **labeling costs** is always a big challenge (and is our ultimate goal)



About the title

- There are various definitions of **weakly-supervised learning**
- Here, we mean (binary/multi-class) **classification**, such that
  1. The focus is still inductive learning but not transductive
  2. The performance measure is still the classification error
  3. Not all training data are equipped with (ordinary) labels
- Two types of weakly-supervised learning
  - **Semi-supervised learning** (Chapelle+, *Semi-Supervised Learning, 2006*) where we have a small set of fully labeled training data
  - Other learning problems where no such set is available

Semi-supervised learning

- Most popular form of learning objectives to be minimized:

  Empirical risk (labeled data) + **Regularization** (unlabeled data)

  - Empirical risk is defined exactly same as in supervised learning
  - Regularization is based on the local smoothness or robustness

Explicit regularization in objective function

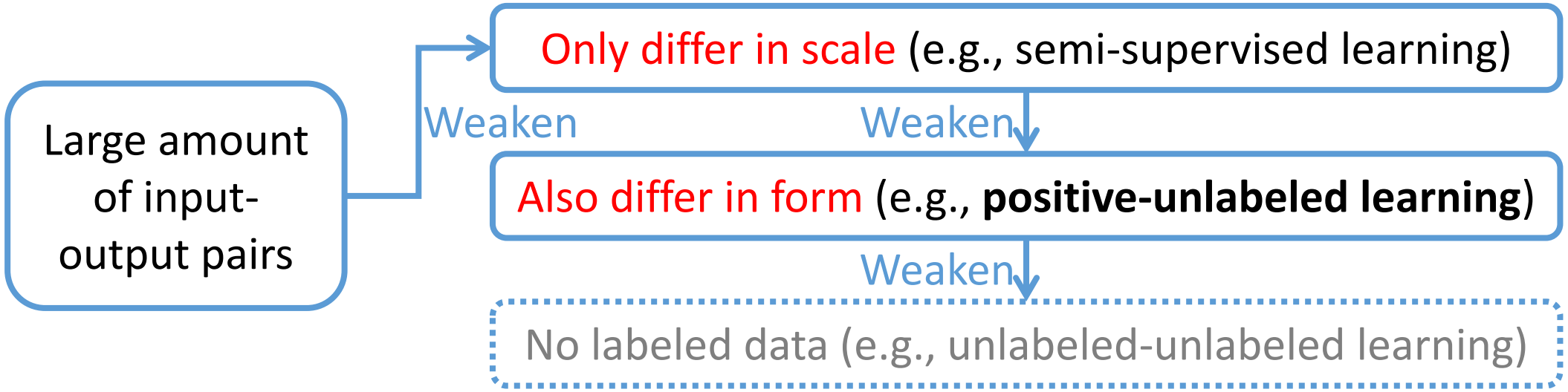
- Manifold regularization (Belkin+, *JMLR 2006*)
- Virtual adversarial training (Miyato+, *ICLR'16*)

Implicit regularization in training algorithm

- Temporal ensembling (Laine & Aila, *ICLR'17*)
- Mean teacher (Tarvainen & Valpola, *NIPS'17*)

Other weakly-supervised learning problems

- Characteristic of labeled data for training



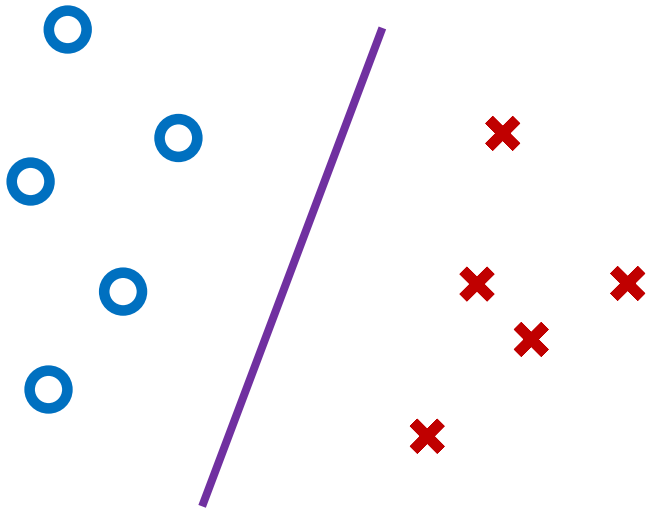
- Hence, we need to rewrite the **true risk**, if we want to follow ERM
- Fundamental questions:
  1. How to design **unbiased risk estimators**?
  2. When **deep learning** is involved, is this still the right way to go?

Question 1

How to design unbiased risk estimators?

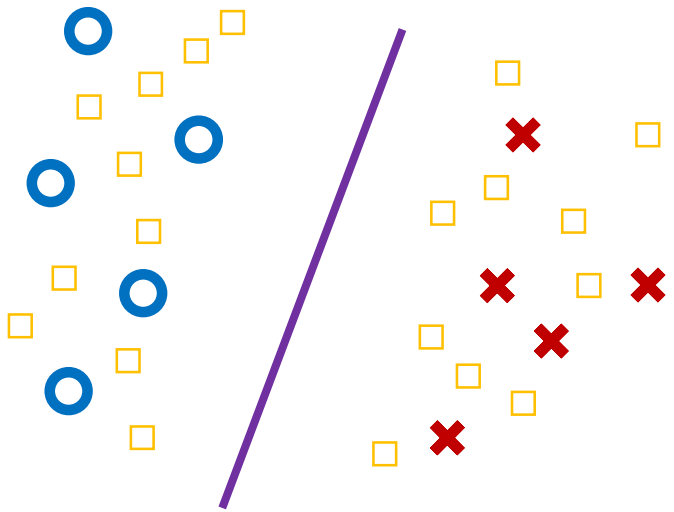
Problem settings in a nutshell

PN learning
(i.e., supervised learning)



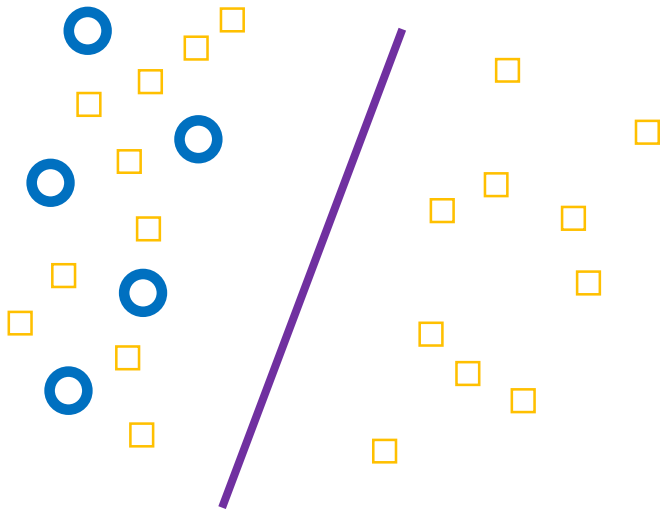
P & N data are
available for training

PNU learning
(i.e., semi-supervised learning)



P, N & U data are
available for training

PU learning



P & U data are
available for training

: positive data : negative data : unlabeled data

Notation

Random variable	Input $X \in \mathbb{R}^d$	Output $Y \in \{\pm 1\}$
Density	Underlying joint density $p(x, y)$	
	$p(x)$	$p_{\mathrm{p}}(x) = p(x \mid Y = +1)$	$p_{\mathrm{n}}(x) = p(x \mid Y = -1)$
Class-prior probability $\pi_{\mathrm{p}} = p(Y = +1)$ Assumed known; can be estimated Ramaswamy+ (*ICML'16*); Jain+ (*NIPS'16*); du Plessis+ (*MLJ 2017*)		
Expectation	$\mathbb{E}_{\mathrm{p}}[\cdot] = \mathbb{E}_{X \sim p_{\mathrm{p}}}[\cdot]$	$\mathbb{E}_{\mathrm{n}}[\cdot] = \mathbb{E}_{X \sim p_{\mathrm{n}}}[\cdot]$
Dataset	$\mathcal{X}_{\mathrm{p}} = \{x_i^{\mathrm{p}}\}_{i=1}^{n_{\mathrm{p}}} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{p}}(x)$	$\mathcal{X}_{\mathrm{n}} = \{x_i^{\mathrm{n}}\}_{i=1}^{n_{\mathrm{n}}} \overset{\text{i.i.d.}}{\sim} p_{\mathrm{n}}(x)$	$\mathcal{X}_{\mathrm{u}} = \{x_i^{\mathrm{u}}\}_{i=1}^{n_{\mathrm{u}}} \overset{\text{i.i.d.}}{\sim} p(x)$

Empirical risk estimator in PN learning

- Let g be a **decision function** & ℓ be a **loss function**
- The **risk** of g is

$$R(g) = \mathbb{E}_{(X,Y)\sim p(x,y)}\left[\ell\left(Yg(X)\right)\right]$$
$$= \pi_{\mathrm{p}}\mathbb{E}_{\mathrm{p}}\left[\ell\left(g(X)\right)\right] + \pi_{\mathrm{n}}\mathbb{E}_{\mathrm{n}}\left[\ell\left(-g(X)\right)\right]$$

  where $\pi_{\mathrm{n}} = 1 - \pi_{\mathrm{p}}$

- The risk can be approximated directly by

$$\hat{R}_{\mathrm{pn}}(g) = \frac{\pi_{\mathrm{p}}}{n_{\mathrm{p}}}\sum_{x\in\mathcal{X}_{\mathrm{p}}}\ell\left(g(x)\right) + \frac{\pi_{\mathrm{n}}}{n_{\mathrm{n}}}\sum_{x\in\mathcal{X}_{\mathrm{n}}}\ell\left(-g(x)\right)$$

- This doesn't work for PU learning!

Empirical risk estimator in PU learning (du Plessis+, *ICML'15*)

- Key observations
  - $\pi_{\mathrm{n}} p_{\mathrm{n}}(x) = p(x) - \pi_{\mathrm{p}} p_{\mathrm{p}}(x)$
  - $\pi_{\mathrm{n}} \mathbb{E}_{\mathrm{n}}\left[\ell\left(-g(X)\right)\right] = \mathbb{E}_X\left[\ell\left(-g(X)\right)\right] - \pi_{\mathrm{p}} \mathbb{E}_{\mathrm{p}}\left[\ell\left(-g(X)\right)\right]$
- Thus the risk can be expressed as

$$R(g) = \pi_{\mathrm{p}} \mathbb{E}_{\mathrm{p}}\left[\ell\left(g(X)\right) - \ell\left(-g(X)\right)\right] + \mathbb{E}_X\left[\ell\left(-g(X)\right)\right]$$

- This can be approximated indirectly by

$$\hat{R}_{\mathrm{pu}}(g) = \frac{\pi_{\mathrm{p}}}{n_{\mathrm{p}}} \sum_{x \in \mathcal{X}_{\mathrm{p}}} \left[\ell\left(g(x)\right) - \ell\left(-g(x)\right)\right] + \frac{1}{n_{\mathrm{u}}} \sum_{x \in \mathcal{X}_{\mathrm{u}}} \ell\left(-g(x)\right)$$

- Simple in retrospect!

Non-convex special case (du Plessis+, *NIPS'14*)

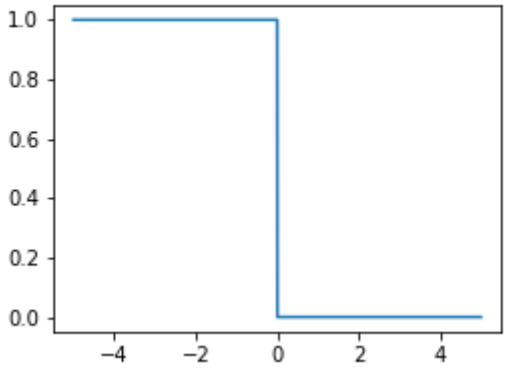
- If $\ell(t) + \ell(-t) = 1$

$$R(g) = 2\pi_{\mathrm{p}}\mathbb{E}_{\mathrm{p}}\left[\ell\left(g(X)\right)\right] + \mathbb{E}_X\left[\ell\left(-g(X)\right)\right] - \pi_{\mathrm{p}}$$

  - Non-convex in g

- Examples

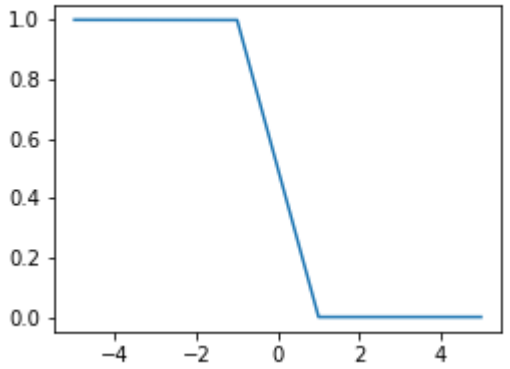
Zero-one loss



$(1 - \mathrm{sign}(t))/2$

Fit evaluation & validation

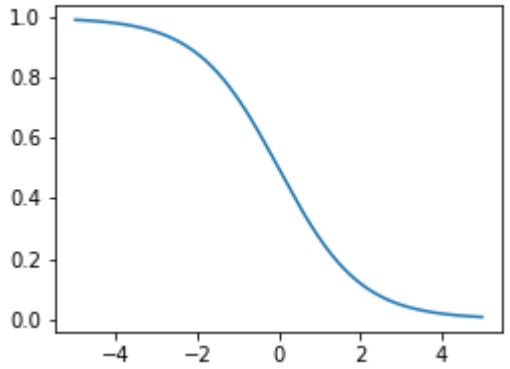
(scaled) **Ramp** loss



$\max\{0, \min\{1, (1-t)/2\}\}$

Fit CCCP solver

Sigmoid loss



$1/(1 + \exp(t))$

Fit SGD solver

Convex special case (du Plessis+, *ICML'15*)

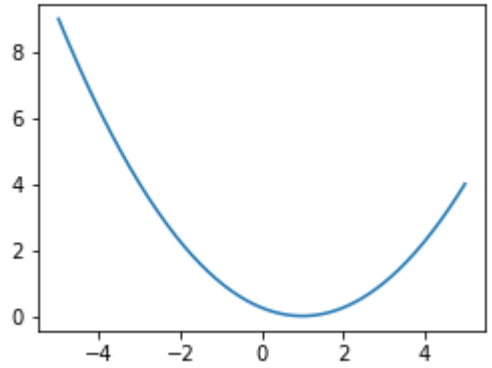
- If $\ell(t) - \ell(-t) = -t$ (Natarajan+, *NIPS'13*; Patrini+, *ICML'16*)

$$R(g) = \pi_{\mathrm{p}} \mathbb{E}_{\mathrm{p}}[-g(X)] + \mathbb{E}_{X}[\ell(-g(X))]$$

  - Convex in g, and convex in θ if $g(x;\theta)$ is linear in θ

- Examples

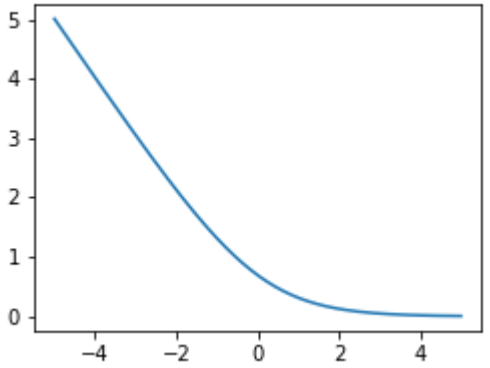
(scaled) **Squared** loss



$(t-1)^2/4$

Analytic solution

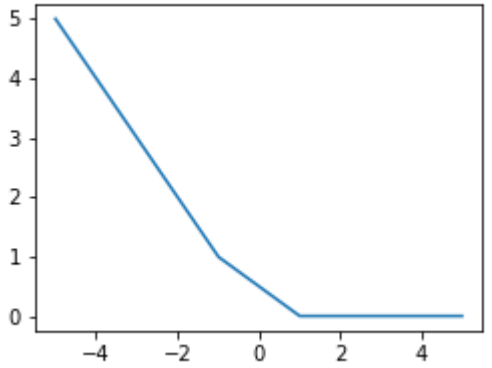
Logistic loss



$\ln(1+\exp(-t))$

Fit SGD solver

Double hinge loss



$\max\{0, (1-t)/2, -t\}$

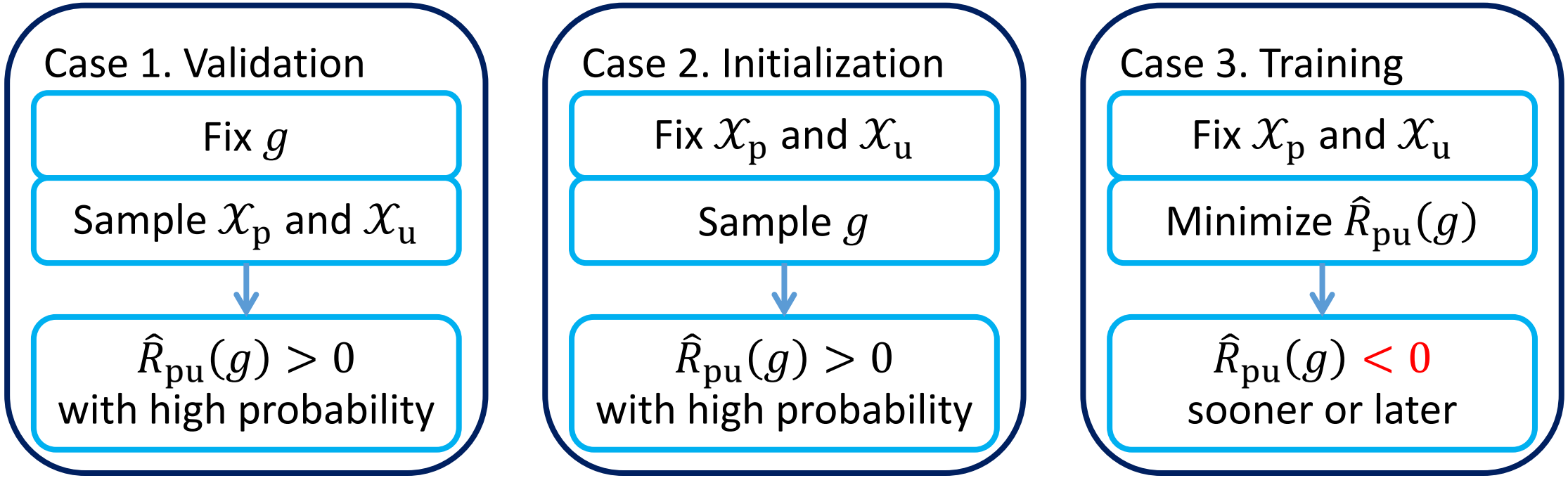
Fit QP solver

Question 2

When deep learning is involved,
is this still the right way to go?

Thought experiment

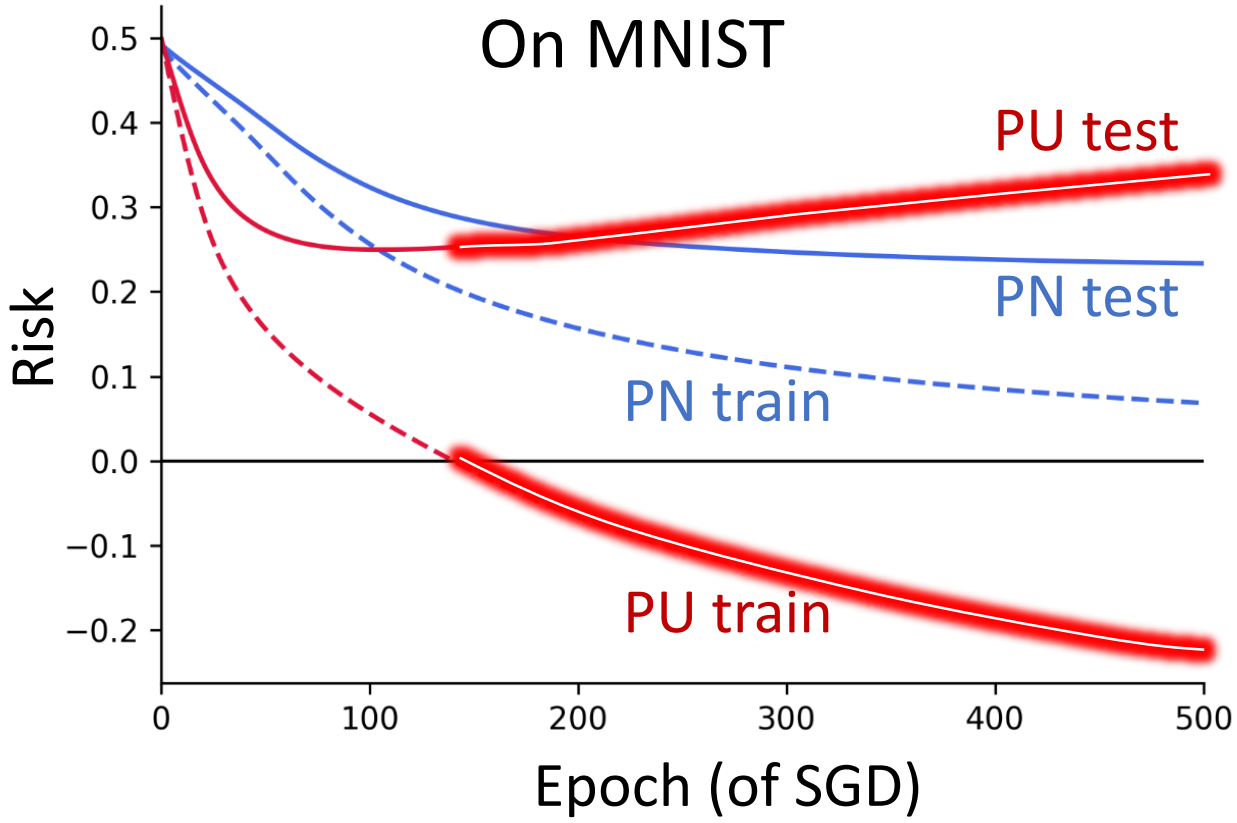
- Assume g is fairly flexible (such as **deep NNs**) and $\forall g, R(g) > 0$
- Consider when deep learning meets weakly-supervised learning:



- $\hat{R}_{\mathrm{pu}}(g) < 0$ during training must be **overfitting** since $\forall g, R(g) > 0$

Real experiment

- $\hat{R}_{\mathrm{pu}}(g)$ is nice for training **linear-in-parameter models**
- It cannot be used for training even the shallowest **MLP**



Overfitting

Emp. risk < 0

- P = {even digits}
  N = {odd digits}
- $\pi_{\mathrm{p}} = 0.49$
  $n_{\mathrm{p}} = 100$
  $n_{\mathrm{n}} = 50$
  $n_{\mathrm{u}} = 59{,}900$
- We can observe
  P is too limited
  so U cannot help

Non-negative risk estimator (Kiryo+, *NIPS'17*)

- Rescue with neither changing model nor labeling more data
- Recall $\pi_{\mathrm{n}}\mathbb{E}_{\mathrm{n}}\left[\ell\left(-g(X)\right)\right] = \mathbb{E}_X\left[\ell\left(-g(X)\right)\right] - \pi_{\mathrm{p}}\mathbb{E}_{\mathrm{p}}\left[\ell\left(-g(X)\right)\right]$
  - Approximate left-hand-side ➔ $\hat{R}_{\mathrm{pn}}(g) \geq 0$
  - Approximate right-hand-side ➔ $\hat{R}_{\mathrm{pu}}(g) \not\geq 0$
- Force it to be non-negative!

$$\tilde{R}_{\mathrm{pu}}(g) = \frac{\pi_{\mathrm{p}}}{n_{\mathrm{p}}}\sum_{x\in\mathcal{X}_{\mathrm{p}}}\left[\ell\left(g(x)\right)\right] + \max\left\{0, \frac{1}{n_{\mathrm{u}}}\sum_{x\in\mathcal{X}_{\mathrm{u}}}\ell\left(-g(x)\right) - \frac{\pi_{\mathrm{p}}}{n_{\mathrm{p}}}\sum_{x\in\mathcal{X}_{\mathrm{p}}}\left[\ell\left(-g(x)\right)\right]\right\}$$

- Minimizing $\tilde{R}_{\mathrm{pu}}(g)$ is no longer **embarrassingly parallel**

Large-scale learning algorithm (Kiryo+, *NIPS'17*)

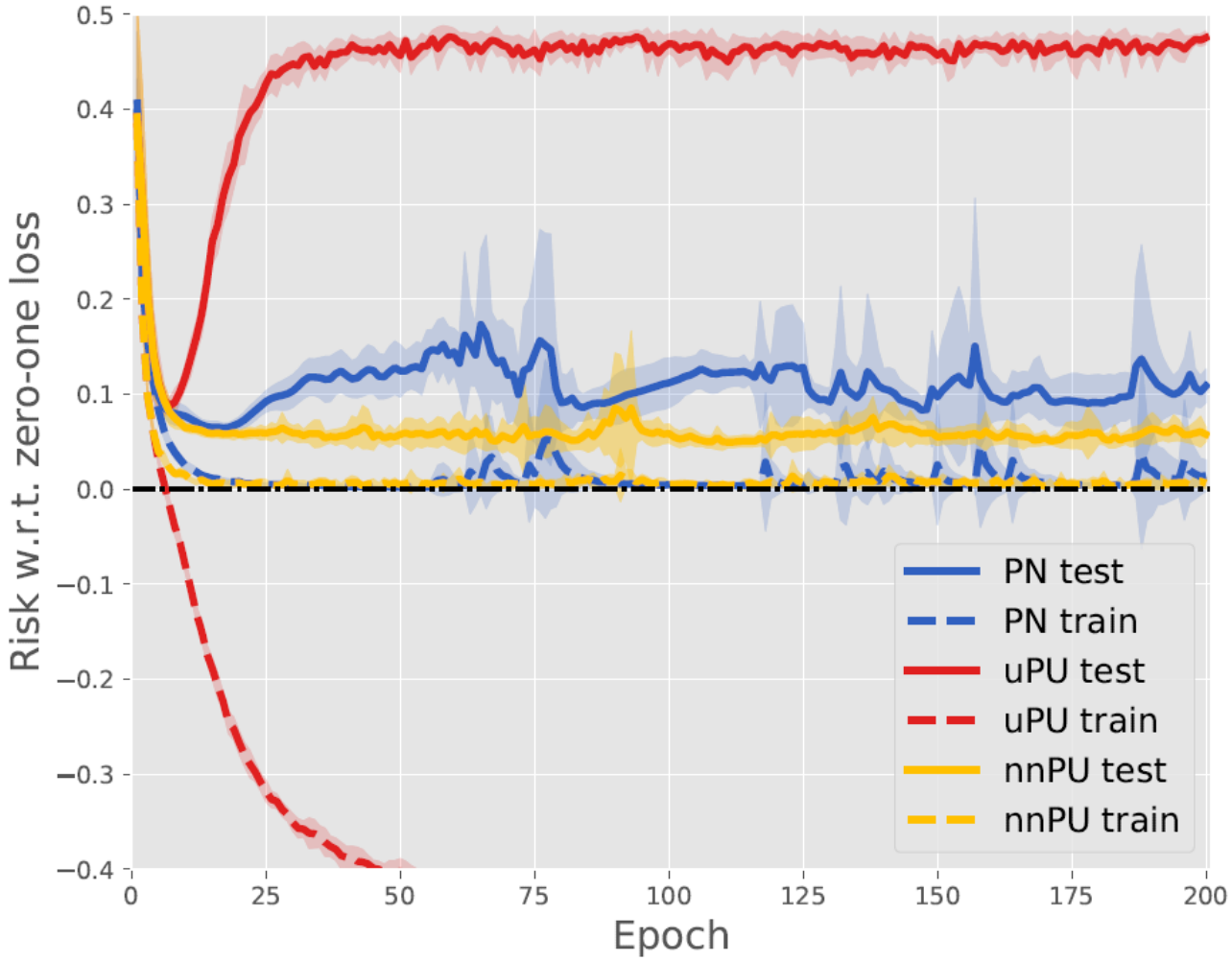
- Safe to minimize $\tilde{R}_{\mathrm{pu}}(g)$ averaged over mini-batches

$$\max\left\{0, \frac{1}{n_{\mathrm{u}}}\sum_{x\in\mathcal{X}_{\mathrm{u}}}\ell\left(-g(x)\right) - \frac{\pi_{\mathrm{p}}}{n_{\mathrm{p}}}\sum_{x\in\mathcal{X}_{\mathrm{p}}}\left[\ell\left(-g(x)\right)\right]\right\}$$

$$\leq \frac{1}{N}\sum_{i=1}^{N}\max\left\{0, \underbrace{\frac{1}{n_{\mathrm{u}}/N}\sum_{x\in\mathcal{X}_{\mathrm{u}}^{i}}\ell\left(-g(x)\right) - \frac{\pi_{\mathrm{p}}}{n_{\mathrm{p}}/N}\sum_{x\in\mathcal{X}_{\mathrm{p}}^{i}}\left[\ell\left(-g(x)\right)\right]}_{\text{Denote by } \Delta}\right\}$$

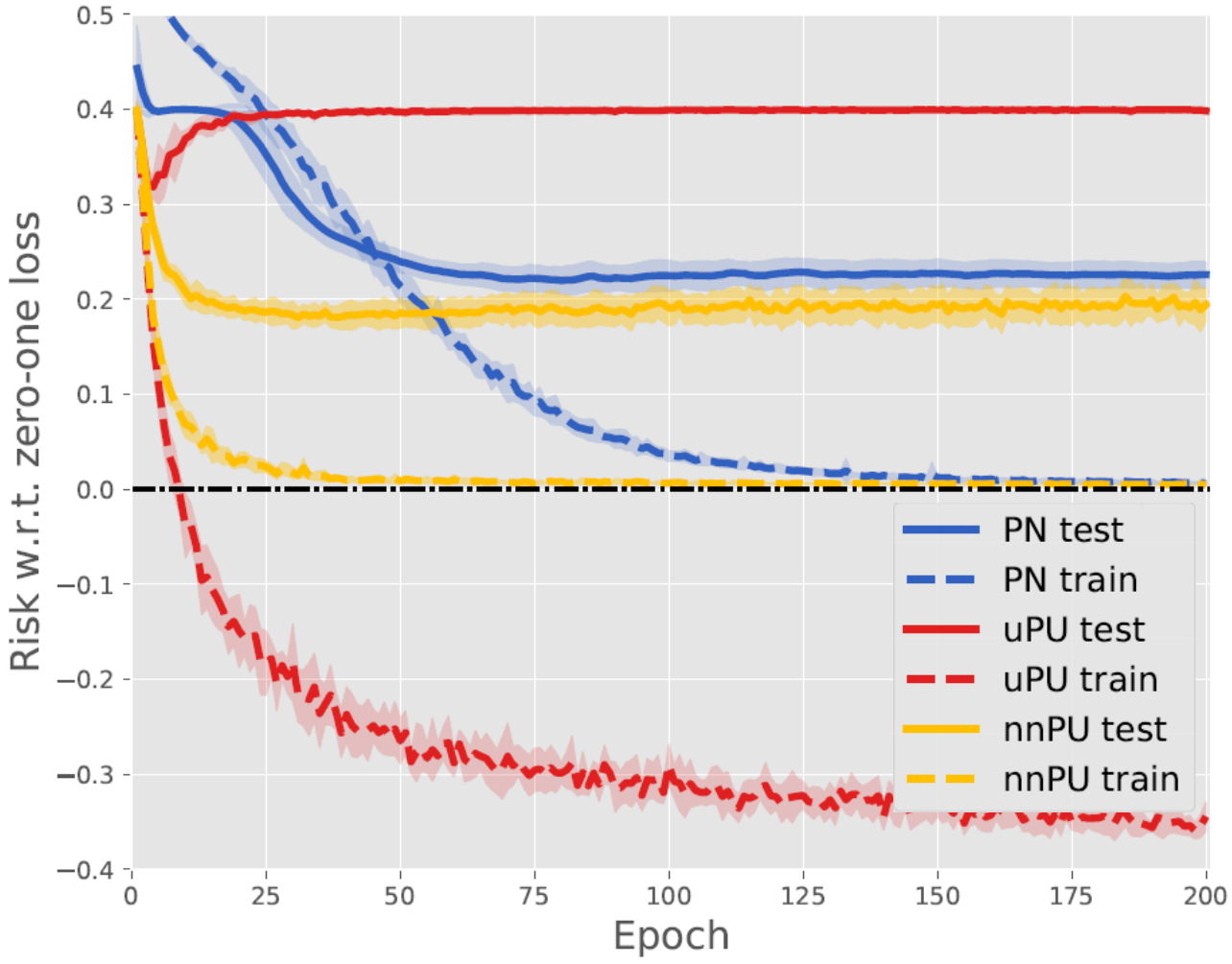
- Given i-th mini-batch $(\mathcal{X}_{\mathrm{p}}^{i}, \mathcal{X}_{\mathrm{u}}^{i})$
  - Gradient decent according to $\hat{R}_{\mathrm{pu}}$ if $\Delta \geq 0$ ← Fit this mini-batch
  - Gradient ascent according to Δ otherwise ← Correct overfitting
- Updates are done by external **SGD**-like algorithms

Experiments on MNIST



- P = {even digits, i.e., 0, 2, 4, 6 & 8}
  N = {odd digits, i.e., 1, 3, 5, 7 & 9}
- $\pi_{\mathrm{p}} = 0.49$
  $n_{\mathrm{p}} = 1{,}000$
  $n_{\mathrm{n}} = \left(\pi_{\mathrm{n}}/2\pi_{\mathrm{p}}\right)^2 n_{\mathrm{p}}$
  $n_{\mathrm{u}} = 60{,}000$
- Model: 6-layer MLP with **ReLU** (Nair & Hinton, *ICML'10*) & **Batch Normalization** (Ioffe & Szegedy, *ICML'15*)

Experiments on CIFAR-10



- P = {airplane, automobile, ship & truck}
  N = {bird, cat, deer, dog, frog & horse}
- $\pi_{\mathrm{p}} = 0.40$
  $n_{\mathrm{p}} = 1{,}000$
  $n_{\mathrm{n}} = \left(\pi_{\mathrm{n}}/2\pi_{\mathrm{p}}\right)^2 n_{\mathrm{p}}$
  $n_{\mathrm{u}} = 50{,}000$
- Model: 13-layer CNN which is known as **All Convolutional Net** (Springenberg+, *ICLR'15*)

When deep learning meets weakly-supervised learning

PU classification is not a special case!

Problems that suffer (similarly to PU classification)

PU learning while R of ERM is replaced with some other criteria

- **AUC** maximization (Sakai+, *MLJ to appear*)
- **SMI** estimation & maximization (Sakai+, *arXiv 2018*) for dimensionality reduction & independence test

Learning binary classifiers from two datasets (neither PN nor PU)

- Two U having different class priors (du Plessis+, *TAAI'13*; Menon+, *ICML'15*)
- Pairwise similarity dataset & U (Bao+, *arXiv 2018*)

Learning multi-class classifiers from extremely **noisy labels**

- A **complementary label** specifies which class x_i is not from (Ishida+, *NIPS'17*)