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
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)

![Diagram showing supervised and unsupervised learning with high and low accuracy and labeling costs.](image-url)
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
  
  - Large amount of input-output pairs

  - Only differ in scale (e.g., semi-supervised learning)

  - Also differ in form (e.g., positive-unlabeled learning)

  - No labeled data (e.g., unlabeled-unlabeled learning)

- 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

<table>
<thead>
<tr>
<th>Random variable</th>
<th>Input $X \in \mathbb{R}^d$</th>
<th>Output $Y \in {\pm 1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Density</strong></td>
<td><strong>Underlying joint density</strong> $p(x, y)$</td>
<td></td>
</tr>
<tr>
<td>$p(x)$</td>
<td>$p_p(x) = p(x</td>
<td>Y = +1)$</td>
</tr>
<tr>
<td><strong>Class-prior probability</strong> $\pi_p = p(Y = +1)$</td>
<td>Assumed known; can be estimated Ramaswamy+ (ICML'16); Jain+ (NIPS'16); du Plessis+ (MLJ 2017)</td>
<td></td>
</tr>
<tr>
<td><strong>Expectation</strong></td>
<td>$\mathbb{E}<em>p[\cdot] = \mathbb{E}</em>{X \sim p}[\cdot]$</td>
<td>$\mathbb{E}<em>n[\cdot] = \mathbb{E}</em>{X \sim p_n}[\cdot]$</td>
</tr>
<tr>
<td><strong>Dataset</strong></td>
<td>$\mathcal{X}<em>p = {x_i^p}</em>{i=1}^{n_p} \text{ i.i.d.} \sim p_p(x)$</td>
<td>$\mathcal{X}<em>n = {x_i^n}</em>{i=1}^{n_n} \text{ i.i.d.} \sim p_n(x)$</td>
</tr>
</tbody>
</table>
Empirical risk estimator in PN learning

- Let $g$ be a decision function & $\ell$ be a loss function

- The risk of $g$ is

$$R(g) = \mathbb{E}_{(X,Y) \sim p(x,y)}[\ell(Yg(X))]$$

$$= \pi_p \mathbb{E}_p[\ell(g(X))] + \pi_n \mathbb{E}_n[\ell(-g(X))]$$

where $\pi_n = 1 - \pi_p$

- The risk can be approximated directly by

$$\hat{R}_{pn}(g) = \frac{\pi_p}{n_p} \sum_{x \in X_p} \ell(g(x)) + \frac{\pi_n}{n_n} \sum_{x \in X_n} \ell(-g(x))$$

- This doesn’t work for PU learning!
Empirical risk estimator in PU learning (du Plessis+, ICML’15)

- **Key observations**
  - $\pi_n p_n(x) = p(x) - \pi_p p_p(x)$
  - $\pi_n \mathbb{E}_n[\ell(-g(X))] = \mathbb{E}_x[\ell(-g(X))] - \pi_p \mathbb{E}_p[\ell(-g(X))]$

- Thus the risk can be expressed as
  \[ R(g) = \pi_p \mathbb{E}_p[\ell(g(X)) - \ell(-g(X))] + \mathbb{E}_x[\ell(-g(X))] \]

- This can be approximated **indirectly** by
  \[ \hat{R}_{pu}(g) = \frac{\pi_p}{n_p} \sum_{x \in X_p}[\ell(g(x)) - \ell(-g(x))] + \frac{1}{n_u} \sum_{x \in X_u}[\ell(-g(x))] \]

- Simple in retrospect!
Non-convex special case (du Plessis+, NIPS’14)

- If $\ell(t) + \ell(-t) = 1$
  $$R(g) = 2\pi_p \mathbb{E}_p[\ell(g(X))] + \mathbb{E}_X[\ell(-g(X))] - \pi_p$$
  - Non-convex in $g$

- Examples
  
  **Zero-one loss**
  
  $$(1 - \text{sign}(t))/2$$
  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)
  \[ R(g) = \pi_p \mathbb{E}_p[-g(X)] + \mathbb{E}_X[\ell(-g(X))] \]
  - Convex in $g$, and convex in $\theta$ if $g(x; \theta)$ is linear in $\theta$

- 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:
  
  **Case 1. Validation**
  
  Fix $g$
  
  Sample $\mathcal{X}_p$ and $\mathcal{X}_u$
  
  $\hat{R}_{pu}(g) > 0$ with high probability

  **Case 2. Initialization**
  
  Fix $\mathcal{X}_p$ and $\mathcal{X}_u$
  
  Sample $g$
  
  $\hat{R}_{pu}(g) > 0$ with high probability

  **Case 3. Training**
  
  Fix $\mathcal{X}_p$ and $\mathcal{X}_u$
  
  Minimize $\hat{R}_{pu}(g)$
  
  $\hat{R}_{pu}(g) < 0$ sooner or later

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

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

On MNIST

- $P = \{\text{even digits}\}$
- $N = \{\text{odd digits}\}$
- $\pi_p = 0.49$
- $n_p = 100$
- $n_n = 50$
- $n_u = 59,900$

- We can observe $P$ is too limited so $U$ cannot help

Emp. risk $< 0$

Overfitting
Non-negative risk estimator \textit{(Kiryo+, NIPS'17)}

- Rescue with neither changing model nor labeling more data

- Recall \( \pi_n \mathbb{E}_n[\ell(-g(X))] = \mathbb{E}_x[\ell(-g(X))] - \pi_p \mathbb{E}_p[\ell(-g(X))] \)
  - Approximate left-hand-side \( \Rightarrow \hat{R}_{pn}(g) \geq 0 \)
  - Approximate right-hand-side \( \Rightarrow \hat{R}_{pu}(g) \geq 0 \)

- Force it to be non-negative!
  \[
  \hat{R}_{pu}(g) = \frac{\pi_p}{n_p} \sum_{x \in x_p} \ell(g(x)) + \max \left\{ 0, \frac{1}{n_u} \sum_{x \in x_u} \ell(-g(x)) - \frac{\pi_p}{n_p} \sum_{x \in x_p} \ell(-g(x)) \right\}
  \]

- Minimizing \( \hat{R}_{pu}(g) \) is \textit{no longer embarrassingly parallel}
Large-scale learning algorithm  (Kiryo+, NIPS’17)

- Safe to minimize $\tilde{R}_{pu}(g)$ averaged over mini-batches
  \[
  \max \left\{ 0, \frac{1}{n_u} \sum_{x \in X_u} \ell(-g(x)) - \frac{\pi_p}{n_p} \sum_{x \in X_p} \ell(-g(x)) \right\} 
  \]
  \[
  \leq \frac{1}{N} \sum_{i=1}^{N} \max \left\{ 0, \frac{1}{n_u/N} \sum_{x \in X_u^i} \ell(-g(x)) - \frac{\pi_p}{n_p/N} \sum_{x \in X_p^i} \ell(-g(x)) \right\}
  \]
  Denote by $\Delta$

- Given $i$-th mini-batch $(X_p^i, X_u^i)$
  - Gradient decent according to $\tilde{R}_{pu}$ if $\Delta \geq 0$  ➡️ Fit this mini-batch
  - Gradient ascent according to $\Delta$ otherwise  ➡️ Correct overfitting

- Updates are done by external SGD-like algorithms
Experiments on MNIST

- \( P = \{ \text{even digits, i.e.,} \ 0, 2, 4, 6 & 8 \} \)
- \( N = \{ \text{odd digits, i.e.,} \ 1, 3, 5, 7 & 9 \} \)
- \( \pi_p = 0.49 \)
- \( n_p = 1,000 \)
- \( n_n = \left( \frac{\pi_n}{2\pi_p} \right)^2 n_p \)
- \( n_u = 60,000 \)
- Model: 6-layer MLP with ReLU (Nair & Hinton, ICML’10) & Batch Normalization (Ioffe & Szegedy, ICML’15)
Experiments on CIFAR-10

- \( P = \{ \text{airplane, automobile, ship \\ & truck} \} \)
- \( N = \{ \text{bird, cat, deer, dog, frog \\ & horse} \} \)
- \( \pi_p = 0.40 \)
- \( n_p = 1,000 \)
- \( n_n = \left(\frac{\pi_n}{2\pi_p}\right)^2 n_p \)
- \( n_u = 50,000 \)
- 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*)