Recent Advances on Positive-Unlabeled (PU) Learning

Gang Niu

Project Assistant Professor, UTokyo
Visiting Scientist, RIKEN AIP

Collaborated with Masashi Sugiyama, Marthinus C. du Plessis, Tomoya Sakai, Ryuichi Kiryo & Yao Ma
What is this talk about?

- 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 labeled data, accuracy, and labeling cost with supervised and unsupervised learning]

**Supervised**

Unsupervised

This talk

Ultimate goal

High Accuracy

Low Accuracy

Low Labeling cost

High Labeling cost
Overview

- Focus on binary classification – the most studied learning problem
- In PU learning, a binary classifier is trained from only P & U data
- Unbiased PU learning
  - State-of-the-art approach to training linear-in-parameter models
  - \textbf{Unbiased empirical risk estimators} are minimized
- Non-negative PU learning
  - State-of-the-art approach to training deep models
  - \textbf{Non-negative empirical risk estimators} are minimized
- PNU learning
  - \textbf{Convex combinations} of the objectives of PU & PN learning
Outline

- Introduction

- Unbiased PU Learning (du Plessis, Niu & Sugiyama, *NIPS’14 & ICML’15*)

- Non-negative PU Learning (Kiryo, Niu, du Plessis & Sugiyama, *NIPS’17*)
  (for oral presentation; there are 40 orals/678 acceptance/3240 submissions)

- PNU Learning (Sakai, du Plessis, Niu & Sugiyama, *ICML’17*)

- Theoretical Analyses (Niu, du Plessis, Sakai, Ma & Sugiyama, *NIPS’16*)
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
Motivation of PU learning

- PN learning (whether deep or not) is data demanding
- Then, is PNU learning the most natural choice?
  - Certainly, if the two classes are symmetric  
    ➔ Which is P & which is N does not matter
  - Not really, if there is intrinsic difference in them  
    ➔ Which is P & which is N matters
- PU learning is preferred due to the following reasons
  I. N data are too expensive
  II. N data are too diverse
  III. “N data” are impure
Case I: N data are too expensive

- Some data-collecting activity is prohibited by law
  - Suppose I am a market researcher at Apple
  - Let Samsung be the imaginary enemy
  - Plan A: Hack the data center of Samsung
  - Plan B: Send corporate spies to Samsung

- Some activity is costly & risky
  - Some clinical trials involve healthy subjects with no pre-existing medical conditions → Extreme financial incentives
  - While others pertain to patients with specific health conditions who are willing to try an experimental treatment
    (https://en.wikipedia.org/wiki/Clinical_trial)
Case II: N data are too diverse

- In the first running example
  - **Android** is neither the only competitor
  - Nor is Samsung the only Android vendor

- In the second running example
  - Too difficult to sample healthy subjects without **selection bias**

![Table Image](http://www.idc.com/promo/smartphone-market-share/vendor)
Case III: “N data” are impure

- Wait, and rethink the definition of $N$
- For the sake of smartphone advertising
  - A customer is $N = S/he$ hates iPhone and would never buy it in the whole life
  - $N = \{\text{non-potential user}\} \in \{\text{not existing user}\} = U$
- For the purpose of drug testing
  - Many diseases are due to chromosomal abnormality
  - $P = \{\text{specific genetic disorder}\} \ni \{\text{specific medical condition}\}$
  - $N = \{\text{not possess this disorder}\} \in \{\text{not observe this condition}\} = U$
Previous work in PU learning

- Binary classification (applied to retrieval & novelty/outlier detection)
  - Statistical query model Denis (ALT’98); De Comité+ (ALT’99); Letouzey+ (ALT’00)
  - Linear(-in-parameter) model
    Liu+ (ICML’02); Li & Liu (IJCAI’03); Lee & Liu (ICML’03); Liu+ (ICDM’03);
    Elkan & Noto (KDD’08); The first that might be unbiased
    Ward+ (Biometrics 2009); Scott & Blanchard (AISTATS’09); Blanchard+ (JMLR 2010);
    du Plessis+ (NIPS’14) The first that must be unbiased

- Other applications
  - Matrix completion Hsieh+ (ICML’15)
  - Sequential data Li+ (SDM’09); Nguyen+ (IJCAI’11)
Outline

- Introduction
- Unbiased PU Learning
- Non-negative PU Learning
- PNU Learning
- Theoretical Analyses

- A “cookbook” procedure of ERM
  1. Choose a loss, so that the (expected) **learning objective** (which is known as the **risk**) can be defined
  2. Choose a **model**, so that the risk can be minimized over this model family (rather than all measurable functions)
  3. Approximate the risk by an **empirical risk estimator**
  4. Minimize the empirical risk by an **optimization algorithm**

- Nice to have **independent** learning obj., model & opt. alg. Arguably a key reason for the great success of deep learning

- 3. – Straightforward for PN; non-trivial for PU
Notation

- $X \in \mathbb{R}^d, Y \in \{\pm 1\}$: Input & output RVs
- $p(x, y)$: Underlying joint density
- $p_p(x) = p(x|Y = +1)$: P marginal
- $p_n(x) = p(x|Y = -1)$: N marginal
- $\pi_p = p(Y = +1)$: Class-prior probability – Assumed known; can be estimated Ramaswamy+ (ICML’16); Jain+ (NIPS’16); du Plessis+ (MLJ 2017)
- $\mathbb{E}_p[\cdot] = \mathbb{E}_{X \sim p_p}[\cdot], \mathbb{E}_n[\cdot] = \mathbb{E}_{X \sim p_n}[\cdot]$: Expectation over P/N marginal
- $X_p = \{x_i^p\}_{i=1}^{n_p}$ i.i.d.: P data
- $X_n = \{x_i^n\}_{i=1}^{n_n}$ i.i.d.: N data
- $X_u = \{x_i^u\}_{i=1}^{n_u}$ i.i.d.: U data
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

  \[
  \hat{R}_{pn}(g) = \frac{\pi_p}{n_p} \sum_{x \in \mathcal{X}_p} \ell(g(x)) + \frac{\pi_n}{n_n} \sum_{x \in \mathcal{X}_n} \ell(-g(x))
  \]

- 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 $\ell(t) + \ell(-t) = 1$
  $$R(g) = 2\pi_p E_p[\ell(g(X))] + E_X[\ell(-g(X))] - \pi_p$$
  
  - Non-convex in $g$

- Examples

**Zero-one** loss

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

Fit evaluation & validation

**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
  $$\frac{(t - 1)^2}{4}$$
  Analytic solution

  **Logistic** loss
  $$\ln(1 + \exp(-t))$$
  Fit SGD solver

  **Double hinge** loss
  $$\max\{0, (1 - t)/2, -t\}$$
  Fit QP solver
Remarks

- $\hat{R}_{pu}(g)$ is unbiased & consistent, similarly to $\hat{R}_{pn}(g)$
  - Why is unbiasedness important? Easy to be consistent

- Biased SVM (Liu+, ICDM’03) – best method prior to Elkan & Noto (KDD’08)
  \[
  \frac{1}{2} \|w\|^2 + C_p \sum_{x \in X_p} \ell_H(w \cdot x + b) + C_u \sum_{x \in X_u} \ell_H(-w \cdot x - b)
  \]
  - Guess/search $C_p$ & $C_u$
  - No learning guarantee

- SMO solver speeding up double hinge loss (Sansone+, arXiv 2016)

Note that theoretical & experimental results omitted for simplicity
Outline

- Introduction
- Unbiased PU Learning
- Non-negative PU Learning
- PNU Learning
- Theoretical Analyses
Revisit ERM

2. Choose a model \( \mathcal{G} \)
   - **Approximation error** \( \inf_{g \in \mathcal{G}} R(g) - \inf_g R(g) \)
     Smaller for more flexible model

3. Approximate \( R(g) \) by \( \hat{R}(g) \)
   - **Estimation error** \( R(\hat{g}) - \inf_{g \in \mathcal{G}} R(g) \) \( \iff \hat{g} = \arg \inf_{g \in \mathcal{G}} \hat{R}(g) \)
     Smaller for less flexible model, or bigger training data
     Converge to zero, if learning is consistent

4. Minimize \( \hat{R}(g) \) by an optimization algorithm which returns \( \hat{g}' \)
   - **Optimization error** \( \hat{R}(\hat{g}') - \hat{R}(\hat{g}) \)
Motivation

- \( \hat{R}_{pu}(g) \) is nice for training linear-in-parameter models
- However, it **cannot** be used for training deep networks

On MNIST

- \( P = \{ \text{even digits} \} \)
- \( N = \{ \text{odd digits} \} \)
- \( n_p = 100 \)
- \( n_n = 50 \)
- \( n_u = 59,900 \)
- \( P \) too limited
- \( U \) cannot help

![Graph showing risk over epochs for PU and PN training and testing](image)

- PU test
- PN test
- PU train
- PN train

Emp. risk < 0

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

- 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 \( \Rightarrow \hat{R}_{pu}(g) \nless 0 \)
- 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 no longer embarrassingly parallel
Let \((\mathcal{X}_p^1, \mathcal{X}_u^1), \ldots, (\mathcal{X}_p^N, \mathcal{X}_u^N)\) be mini-batches

\[
\max \left\{ 0, \frac{1}{n_u} \sum_{x \in \mathcal{X}_u} \ell(-g(x)) - \frac{\pi_p}{n_p} \sum_{x \in \mathcal{X}_p} \ell(-g(x)) \right\}
\]

\[
= \max \left\{ 0, \sum_{i=1}^N \frac{1}{n_u} \sum_{x \in \mathcal{X}_u^i} \ell(-g(x)) - \sum_{i=1}^N \frac{\pi_p}{n_p} \sum_{x \in \mathcal{X}_p^i} \ell(-g(x)) \right\}
\]

\[
\leq \frac{1}{N} \sum_{i=1}^N \max \left\{ 0, \frac{1}{n_u/N} \sum_{x \in \mathcal{X}_u^i} \ell(-g(x)) - \frac{\pi_p}{n_p/N} \sum_{x \in \mathcal{X}_p^i} \ell(-g(x)) \right\}
\]

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

You can choose whatever stochastic opt. alg. you like

- **Adam** (Kingma+, *ICLR’15*) or **AdaGrad** (Duchi+, *JMLR 2011*)
Experiments on MNIST

- $P = \{\text{even digits, i.e., } 0, 2, 4, 6 \& 8\}$
- $N = \{\text{odd digits, i.e., } 1, 3, 5, 7 \& 9\}$
- $\pi_p = 0.49$
- $n_p = 1,000$
- $n_n = \left(\pi_n/2\pi_p\right)^2 n_p$
- $n_u = 60,000$

- Model: 6-layer MLP with ReLU (Nair & Hinton, ICML’10)
  & BN (Ioffe & Szegedy, ICML’15)
Experiments on CIFAR10

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

- Model: 13-layer CNN – All Convolutional Net (Springenberg+, ICLR’15)
Remarks

- $\tilde{R}_{pu}(g)$ is not a shrinkage estimator
  - $\tilde{R}_{pu}(g) = \hat{R}_{pu}(g)$ if $\hat{R}_{pu}(g) \geq 0$ \(\Leftarrow\) Identical for simple models
  - $\tilde{R}_{pu}(g) > \hat{R}_{pu}(g)$ if $\hat{R}_{pu}(g) < 0$ \(\Leftarrow\) Different for complex models

- $\tilde{R}_{pu}(g)$ is biased but still consistent; bias is in $\mathcal{O}\left(\exp\left(-\frac{1}{1/n_p+1/n_u}\right)\right)$

- Mean squared error is reduced for certain losses

- Learning is consistent; estimation error bound is in $\mathcal{O}_p\left(\frac{1}{\sqrt{n_p}} + \frac{1}{\sqrt{n_u}}\right)$
  for linear-in-parameter models
Outline

- Introduction
- Unbiased PU Learning
- Non-negative PU Learning
- PNU Learning
- Theoretical Analyses
Motivation

- SSL usually needs additional **distributional assumptions**
  - Cluster (Chapelle+, *NIPS’02*), manifold (Belkin+, *JMLR 2006*) ...
  - U data are used in **regularization**
    ➔ They will bias the classifier (learned following ERM)

- Violated assumption ➔ U data become **harmful & hurt**

- PU learning has **no** additional distributional assumption
  - U data are used in **risk evaluation**
    ➔ They will **not** bias the classifier
$\text{Illustration}$

$\text{PNU learning} = \text{PN learning} + \text{PU learning}$

$\text{PNU learning} = \text{PN learning} + \text{NU learning}$
Unbiased risk estimators \cite{Sakai+17}

- Given $\gamma \in [0,1]$, PN+PU learning is to minimize
  \[ \hat{R}_{pn,pu}^\gamma(g) = (1 - \gamma)\hat{R}_{pn}(g) + \gamma \hat{R}_{pu}(g) \]

- Analogously, PN+NU learning is to minimize
  \[ \hat{R}_{pn,nu}^\gamma(g) = (1 - \gamma)\hat{R}_{pn}(g) + \gamma \hat{R}_{nu}(g) \]

  where \( \hat{R}_{nu}(g) = \frac{n_n}{n_n} \sum_{x \in X_n} [\ell(-g(x)) - \ell(g(x))] + \frac{1}{n_u} \sum_{x \in X_u} \ell(g(x)) \)

- Given $\eta \in [-1,1]$, we define PNU learning as to minimize
  \[ \hat{R}_{pnu}^\eta(g) = \begin{cases} 
  \hat{R}_{pn,pu}^\eta(g), & \text{if } \eta \geq 0 \\
  \hat{R}_{pn,nu}^{-\eta}(g), & \text{if } \eta < 0 
  \end{cases} \]
  - NU learning, if $\eta = -1$
  - PN learning, if $\eta = 0$
  - PU learning, if $\eta = +1$
Why not PU+NU?

- Motivated by theoretical comparisons \((\text{Niu+}, NIPS'16)\)
- PN learning can never be the worst among PN, PU & NU learning
  - If \(n_u\) is sufficiently large (compared with \(n_p\) & \(n_n\))
    \(\Rightarrow\) PN learning is the second best
  - Otherwise \(\Rightarrow\) PN learning is the best
- PU+NU learning is not the best possible combination
  PN+PU & PN+NU learning are the best combinations

Note that theoretical & experimental results omitted for simplicity
Outline

- Introduction
- Unbiased PU Learning
- Non-negative PU Learning
- PNU Learning
- Theoretical Analyses
To be short and intuitive

- Based on upper bounds on estimation errors
- Find simple conditions, such that
  - PU learning is likely to outperform PN learning
    \[ \frac{\pi_p}{\sqrt{n_p}} + \frac{1}{\sqrt{n_u}} < \frac{\pi_n}{\sqrt{n_n}} \]
  - NU learning is likely to outperform PN learning
    \[ \frac{\pi_n}{\sqrt{n_n}} + \frac{1}{\sqrt{n_u}} < \frac{\pi_p}{\sqrt{n_p}} \]
- Either PU or NU learning (depending on \(\pi_p, \pi_n, n_p \& n_n\)) given infinite U data will improve on PN learning
Simplified estimation error bounds \(\text{ (Niu+}, \text{ NIPS’16)\)}

- Let \(g^*/\hat{g}_{\text{pn}}, \hat{g}_{\text{pu}} \text{ & } \hat{g}_{\text{nu}}\) be true/empirical risk minimizers

- Assume Rademacher complexities (Mohri+, Foundations of ML, 2012) of the function class \(\mathcal{G}\) (a.k.a. model family) are in \(\mathcal{O}(1/\sqrt{n})\)

- Bounds below hold separately with probability at least \(1 - \delta\)

\[
\begin{align*}
R(\hat{g}_{\text{pn}}) - R(g^*) &\leq C(\delta) \cdot \left(\frac{\pi_p}{\sqrt{n_p}} + \frac{\pi_n}{\sqrt{n_n}}\right) \\
R(\hat{g}_{\text{pu}}) - R(g^*) &\leq C(\delta) \cdot \left(2\frac{\pi_p}{\sqrt{n_p}} + \frac{1}{\sqrt{n_u}}\right) \\
R(\hat{g}_{\text{nu}}) - R(g^*) &\leq C(\delta) \cdot \left(\frac{1}{\sqrt{n_u}} + 2\frac{\pi_n}{\sqrt{n_n}}\right)
\end{align*}
\]

where \(C(\delta)\) is a function of \(\delta\)

- Originally for du Plessis+ (NIPS’14) but can be extended to du Plessis+ (ICML’15)
Finite-sample comparisons  (Niu+, NIPS’16)

- Tighter PU bound than PN bound, if $\alpha_{pu,pn} = \frac{\pi_p/\sqrt{n_p} + 1/\sqrt{n_u}}{\pi_n/\sqrt{n_n}} < 1$

- Tighter NU bound than PN bound, if $\alpha_{nu,pn} = \frac{\pi_n/\sqrt{n_n} + 1/\sqrt{n_u}}{\pi_p/\sqrt{n_p}} < 1$

- $\alpha_{pu,pn}$ is monotonically decreasing in $n_p$ & $n_u$
  - More U data $\Rightarrow$ PU improves
  - More P data $\Rightarrow$ PN improves too, but PU improves more!

- Many other implications discussed as well
  - When $n_p$, $n_n$ & $n_u$ are proportional
  - When $n_p/n_n \approx \pi_p/\pi_n$ is further imposed
Asymptotic comparisons  \((\text{Niu}+, NIPS'16)\)

- In practice, we may find PU worse than PN and \(\alpha_{pu,pn} > 1\)
  - Give up PU? Collect more U data (in order to improve PU)?

- Assume \(n_p, n_n < \infty \) & \(n_u \to \infty\), or \(O(n_p) = O(n_n) < O(n_u)\)
  - \(\alpha_{pu,pn}^* \) & \(\alpha_{nu,pn}^*\) exist as the limits of \(\alpha_{pu,pn}\) & \(\alpha_{nu,pn}\)

- Then, \(\alpha_{pu,pn}^* \cdot \alpha_{nu,pn}^* = 1\)
  - Either \(\alpha_{pu,pn}^* < 1\)  \(\leftarrow\) Limit of PU will improve on that of PN
  - Or \(\alpha_{nu,pn}^* < 1\)  \(\leftarrow\) Limit of NU will improve on that of PN
  - Exception: \(n_p/n_n \to \pi_p^2/\pi_n^2\)  \(\leftarrow\) All 3 limits are equally good
Take-home message

- Unbiased PU learning and **simple** models given **small** data
  - **Logistic** loss or **double hinge** loss for convexity

- Non-negative PU learning and **complex** models given **big** data
  - **Sigmoid** loss for a similar shape to **zero-one** loss

- When PU doesn’t work ⇒ Label some N to see whether PN works
  - PN works and $\pi_p / \sqrt{n_p} > \pi_n / \sqrt{n_n}$ ⇒ Label more P (expensive) or switch to NU/PNU if you want
  - PN works and $\pi_p / \sqrt{n_p} < \pi_n / \sqrt{n_n}$ ⇒ Collect more U (cheap)
  - PN doesn’t work as well ⇒ Find a more suitable model
Beyond this talk …

- We are **two-sample** PU (Ward, Hastie, Barry, Elith & Leathwick, *Biometrics* 2009)
  - Fairly different from **one-sample** PU (Elkan & Noto, *KDD*’08)
  - According to Menon, Van Rooyen, Ong, Williamson (*ICML*’15)
    ∈ **mutually contaminated** learning (Scott, *AISTATS*’15)
    ∉ **class-conditional noise** learning (Natarajan, Dhillon, Ravikumar & Tewari, *NIPS*’13)

- Following work in PU learning
  - Multi-label ranking (Kanehira & Harada, *CVPR*’16) ← Fancy application
  - AUC maximization (PU & SSL) (Sakai, Niu & Sugiyama, *MLJ*, to appear)
  - Multi-instance binary classification (Bao, Sakai, Sato & Sugiyama, *arXiv 2017*)
  - Multi-class classification (Konno, *UTokyo bachelor thesis 2017*)