Predictive Adversarial Learning from Positive and Unlabeled Data

Wenpeng Hu[1,*], **Ran Le**[2,*], **Bing Liu**[3,†], **Feng Ji**[4], **Jinwen Ma**[1], **Dongyan Zhao**[2], **Rui Yan**[2,‡]
[1] Department of Information Science, School of Mathematical Sciences, Peking University
[2] Wangxuan Institute of Computer Technology, Peking University
[3] Department of Computer Science, University of Illinois at Chicago
[4] Alibaba Group
{wenpeng.hu, leran, jwma, zhaody, ruiyan}@pku.edu.cn, liub@uic.edu, zhongxiu.jf@alibaba-inc.com

Abstract

This paper studies learning from positive and unlabeled examples, known as *PU learning*. It proposes a novel PU learning method called *Predictive Adversarial Networks* (PAN) based on GAN (Generative Adversarial Networks). GAN learns a generator to generate data (e.g., images) to fool a discriminator which tries to determine whether the generated data belong to a (positive) training class. PU learning can be casted as trying to identify (not generate) likely positive instances from the unlabeled set to fool a discriminator that determines whether the identified likely positive instances from the unlabeled set are indeed positive. However, directly applying GAN is problematic because GAN focuses on only the positive data. The resulting PU learning method will have high precision but low recall. We propose a new objective function based on KL-divergence. Evaluation using both image and text data shows that PAN outperforms state-of-the-art PU learning methods and also a direct adaptation of GAN for PU learning.

1 Introduction

Positive-unlabeled learning (or PU learning) learns a binary classifier from only Positive ($\mathcal{P}$) and Unlabeled ($\mathcal{U}$) examples with no labeled negative examples (Liu et al. 2002, 2003; Denis, Gilleron, and Letouzey 2005). PU learning has many applications in text analysis, bio-medicine, recommendation, remote sensing, matrix completion, etc (Lee and Liu 2003; Li and Liu 2003; Li, Guo, and Elkan 2010; Hsieh, Natarajan, and Dhillon 2015). A comprehensive survey of the area can be found in (Bekker and Davis 2020).

In this paper, we propose a novel adversarial PU learning method inspired by GAN (generative adversary networks) (Goodfellow et al. 2014). GAN aims to generate data of a particular training class, which is like the positive class $\mathcal{P}$ in PU learning. GAN works by generating likely positive data using a generator $G(\cdot)$ to fool a discriminator $D(\cdot)$, which determines whether the generated data indeed belong to the training (positive) class. For PU learning, this is like choosing likely positive instances from the unlabeled set $\mathcal{U}$ (achieved by a classifier) also to fool a discriminator $D(\cdot)$. Thus, we can simply replace GAN's generator with a classifier $C(\cdot)$ to produce a PU learner.[1] Note that the classifier is also a discriminator, but we use the term *classifier* here to distinguish it from the original *discriminator* of GAN. Note also both $C(\cdot)$ and $D(\cdot)$ are neural networks.[2]

However, direct adaptation of GAN for PU learning (which is one of our baselines, called *a-GAN*) is problematic. The reason is that GAN focuses on only high positive precision (e.g., generating high quality positive images), but PU learning needs to consider the overall (both positive and negative) performance of $C(\cdot)$. We thus need a different objective function. This paper proposes such an objective function based on Kullback-Leibler (KL) divergence, which also has a new adversarial training method. The proposed PU learning technique is called *PAN* (*Predictive Adversary Networks*) due to the use of the classifier to replace the generator in GAN. KL-divergence in PAN measures whether $C(\cdot)$ can produce similar predictions to those of $D(\cdot)$ for all examples in $\mathcal{U}$. If $C(\cdot)$ gives similar predictions, it means that the examples obtaining high probabilities from $C(\cdot)$ also get high probabilities from $D(\cdot)$, achieving the goal of fooling $D(\cdot)$, which will give us a good final PU classifier $C(\cdot)$.

A major advantage of PAN is that it does not need the input of class prior probability, which many state-of-the-art systems need. In practice, the class prior is unknown, although there are methods to estimate it (see Sec. 2). We will see when the class prior probability estimate is off, the existing methods can perform quite poorly (Sec. 5.2).

In (Hou et al. 2018) and (Chiaroni et al. 2018), the authors employed GAN to generate positive and/or negative data and then use a separate learner to learn the final PU classifier using the generated data, but they are not adaptations of GAN like PAN and their generators generate only images. PAN can be applied to any data as it has no generator.

[*] Equal contribution

[†] The work was partly done when Bing Liu was at Peking University on leave of absence from University of Illinois at Chicago.

[‡] Corresponding author: Rui Yan (ruiyan@pku.edu.cn)

[1] PU learning is analogous to GAN because if we put all the data (e.g., images) that can be generated by GAN's generator in a set, the set should be regarded as unlabeled as it contains both good (positive) and bad (negative) images. Then what the generator does is like selecting good images to fool the discriminator, which is exactly what a PU classifier does with a set of *given* unlabeled data.

[2] Their exact architectures are given in the experiment section (see Training Details in Sec. 5.1) since $C(\cdot)$ and $D(\cdot)$ have different architectures for text and images.

We evaluate PAN using both text and image data and show that it outperforms start-of-the-art PU learning methods even when we give them the perfect class prior probabilities.

2 Related Work

PU learning has been studied for the past two decades. The term *PU learning* was perhaps first used in (Li and Liu 2005). Early theoretical results were reported in (Liu et al. 2002; Denis, Gilleron, and Letouzey 2005). Liu et al. (2002) studied the sample complexity of the problem, and Denis, Gilleron, and Letouzey (2005) investigated it under the statistical query model and the PAC learning framework. Elkan and Noto (2008) showed that if ranking is the goal, then PU learning is equivalent to the conventional binary learning. Due to many applications, there has been a recent surge of interest in PU learning (Du Plessis, Niu, and Sugiyama 2014; du Plessis, Niu, and Sugiyama 2015; Chang et al. 2016; Niu et al. 2016; Sakai et al. 2017; Chiaroni et al. 2018; Sakai, Niu, and Sugiyama 2018; Shi et al. 2018; Sansone, De Natale, and Zhou 2018; Kato, Teshima, and Honda 2019; Hsieh, Niu, and Sugiyama 2019; Sakai, Niu, and Sugiyama 2020). See (Bekker and Davis 2020) for a comprehensive survey of the subject.

Early PU learning algorithms mainly employed 2 heuristic steps (Liu et al. 2002; Li and Liu 2003; Yu, Han, and Chang 2002). Step 1 finds some *reliable negative examples* (RN) from the unlabeled set. Step 2 uses the positive set, the RN set, and the remaining unlabeled set to build the final classifier. Liu et al. (2003) proposed a more principled method called Biased-SVM based on constrained optimization, which regards the unlabeled data as having noisy labels. The same idea was also adopted in (Shi et al. 2018). Lee and Liu (2003) and Elkan and Noto (2008) re-weighted training examples. du Plessis, Niu, and Sugiyama (2015) and Kiryo et al. (2017) used unbiased risk estimators. Kato, Teshima, and Honda (2019) and Hsieh, Niu, and Sugiyama (2019) dealt with sample selection bias. However, none of these existing papers explored adversarial learning like PAN.

Hou et al. (2018) used GAN to generate positive and negative examples to build a classifier. Chiaroni et al. (2018) used GAN to generate only negative training examples. Both papers are for image classification. Generating text and other forms of data using GAN is more challenging. PAN does not generate data and thus can be applied to any form of data. Our formulation and objective function are also quite different.

Weighted adversarial nets (WAN) (Chen et al. 2018; Zhang et al. 2018) is related and similar to our baseline a-GAN as a-GAN also weights the discriminator. But PAN differs significantly because although WAN weights the discriminator, its adversarial training is the same as GAN, similar to our a-GAN. DAN posted on arXiv (Liu, Chen, and Wu 2019) is also quite similar to our baseline a-GAN as we can see from their Eq. 5 and our Eq. 2. But PAN takes an entirely different approach as we will see in the next two sections.

Other related works include leveraging biased negative examples (Sakai et al. 2017), studying the random assumption of PU learning (Bekker and Davis 2020), scalable PU learning (Sansone, De Natale, and Zhou 2018), using traditional margin-based methods (Xu et al. 2017; Gong et al. 2018), and leveraging the reliable supervision provided by the model itself (Chen et al. 2020). More related work can be found in the survey (Bekker and Davis 2020).

A key weakness of many systems is that they need the class prior probability (du Plessis, Niu, and Sugiyama 2015; Kiryo et al. 2017; Hou et al. 2018; Xu et al. 2017; Chiaroni et al. 2018; Kato, Teshima, and Honda 2019), which is hard for the user to provide. There are methods that estimate the prior (Ramaswamy, Scott, and Tewari 2016; Jain, White, and Radivojac 2016; du Plessis, Niu, and Sugiyama 2017), but we show if the estimate is off, the results can be quite poor. Zhang, Hou, and Zhang (2020) proposed a PU learning method that can estimate the class prior in an implicit way, but we could not compare with it as its code is not available.

3 Background

GAN is an *adversarial* learner that learns a generator to generate data instances (e.g., images) similar to those in the real/training data. It has two networks, a *generator* $G(\cdot)$ and a *discriminator* $D(\cdot)$. $G(\cdot)$ tries to generate new data instances that can approximate the real data to fool the discriminator. $D(\cdot)$ tries to discriminate the generated data from the real data (Gao et al. 2019a,b; Chan et al. 2019). Through an iterative and adversarial process, a good generator $G(\cdot)$ is learned that can generate new data instances that $D(\cdot)$ has hard time to distinguish from the real data. GAN is formulated as a minmax game as follows:

$$\begin{aligned}&\min_G \max_D \ \mathrm{V}(D,G)\\ &= \mathbb{E}_{\mathbf{x}\sim P_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim P_z(\mathbf{z})}[\log(1-D(G(\mathbf{z})))]\end{aligned} \tag{1}$$

where P_{data} is the data generating distribution of the real data, and P_z is the data generating distribution of the generator.

3.1 Direct Adaptation of GAN for PU Learning

We now propose a direct adaptation of GAN for PU learning, called *a-GAN*, The real data in GAN is our labeled positive data $\mathcal{P}$. As illustrated in Figure 1, the discriminator $D(\cdot)$ in a-GAN still plays the same role as that in GAN, but the generator in a-GAN is replaced with another discriminator, which we call the *classifier* $C(\cdot)$. The goal of $C(\cdot)$ is to identify likely positives in the unlabeled set $\mathcal{U}$ to give to the discriminator for it to decide whether these are real positive data. The following equation shows this approach:

$$\begin{aligned}\min_C \max_D \ \mathrm{V}(D,C) &= \mathbb{E}_{\mathbf{x}^p\sim P^p(\mathbf{x}^p)}[\log D(\mathbf{x}^p)]\\ &+\mathbb{E}_{\mathbf{x}^s=\arg_{\mathbf{x}^u\sim P^u(\mathbf{x}^u)} C(\mathbf{x}^u)=1}[\log(1-D(\mathbf{x}^s))]\end{aligned} \tag{2}$$

where $\mathbf{x}_s$ is an example judged as a likely positive example from $\mathcal{U}$ by $C(\cdot)$. P^p and P^u are the data generating distributions of the known positive data and the unlabeled data, respectively. The known positive examples are randomly sampled from the positive population. The hidden positives in $\mathcal{U}$ is also randomly sampled from the same positive population.

Due to the discreteness of the last term in the equation, we use the Policy Gradient method (Sutton and Barto 2018) from reinforcement learning to train it, where the last term is regarded as the reward for optimizing $C(\cdot)$.

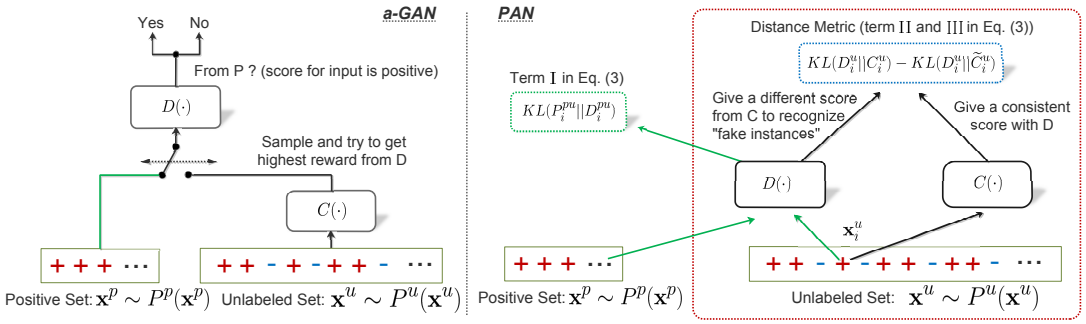

Figure 1: An illustration of the objective functions of a-GAN (left) and PAN (right) as a comparison of the two models.

a-GAN does reasonably well (see Sec. 5.2). However, since it focuses on the positive data only like GAN, its learning is unbalanced, which causes some confusion with the separation of positive and negative data. Next, we present the proposed method PAN, which balances positive and negative and is able to produce a better separation for them.

4 Proposed PAN

PAN adopts the same adversarial learning idea to build a PU classifier $C(\cdot)$. However, as the right part of Figure 1 shows, instead of using $D(\cdot)$ to directly discriminate the known positive data and the selected positive data by $C(\cdot)$, we propose to use the adversarial learning idea on the probability distributions of $D(\cdot)$ and $C(\cdot)$ on each example (or instance). Specifically, in the part surrounded by the red dash-lined box in Figure 1, $D(\cdot)$ and $C(\cdot)$ produce a score for each input example $\mathbf{x}_i^u$ from the unlabeled set $\mathcal{U}$ with different optimization objectives. $D(\cdot)$ tries to give $\mathbf{x}_i^u$ the opposite prediction score to that of $C(\cdot)$ in order to identify it as a "fake" example; $C(\cdot)$ tries to give $\mathbf{x}_i^u$ a similar score to that of $D(\cdot)$ to fool $D(\cdot)$. The adversarial learning is performed through a distance metric. $D(\cdot)$ tries to enlarge the distance with $D(\cdot)$ but $C(\cdot)$ tries to shrink the distance, which is applied to each example in $\mathcal{U}$ (no sampling is used). We choose KL-divergence as the metric, which minimizes the information loss between two probability distributions as it has been shown to be able to learn and suit complex distributions (Goodfellow et al. 2014). The green links in Figure 1 show the procedure of optimizing the known positive data. Note that the unlabeled data are regarded as the negative data in PAN to endow $D(\cdot)$ the ability to recognize negative examples to some extent.

4.1 Predictive Adversary Networks (PAN)

Unlike GAN, which only generates positive examples that are hard to distinguish by the discriminator $D(\cdot)$, we also want the remaining unlabeled examples to be easily distinguishable as possible negatives by the discriminator. To this end, $C(\cdot)$ tries to separate positive and negative examples in $\mathcal{U}$ with a large margin. That is, $C(\cdot)$ not only gives high probabilities to examples that $D(\cdot)$ has difficulty to distinguish (meaning $D(\cdot)$ also gives high probabilities to those examples) but also low probabilities to examples that are easy to distinguish by $D(\cdot)$ (meaning $D(\cdot)$ also gives low probabilities to those examples because of the easy separation). Note, when we say $C(\cdot)$ or $D(\cdot)$ gives high/low probability, we mean the probability of being positive. We propose to achieve our goal by controlling the distance (similarity) between the predictions of $C(\cdot)$ and $D(\cdot)$ on $\mathcal{U}$. We use the sum of KL-divergences on the predictions of all examples (or instances) in $\mathcal{U}$ as the distance. In detail, PAN assumes the output of $D(\cdot)$ (respectively, $C(\cdot)$) on ith instance as a discrete distribution over binary outcomes (or classes) of *positive* and *negative*. For example, if $D(\cdot)$ (likewise, $C(\cdot)$) gives an instance the probability of 0.3. It means that for the positive outcome or class, the probability is 0.3, and for the negative outcome, the probability is 0.7. We use D_i and C_i to denote the two distributions and KL-divergence is employed to measure their distance. Superscripts ^{pu} and ^{u} denote the corresponding datasets. PAN's objective is defined as follows:

$$\min_C \max_D \ \mathrm{V}(D,C) = \underbrace{-\sum_{i=1}^{n} KL(P_i^{pu}||D_i^{pu})}_{\text{I}} + \lambda \Big(\underbrace{\sum_{i=1}^{n_u} KL(D_i^u||C_i^u)}_{\text{II}} - \underbrace{\sum_{i=1}^{n_u} KL(D_i^u||\widetilde{C}_i^u)}_{\text{III}}\Big) \tag{3}$$

where P_i^{pu} is the probability distribution of positive and unlabeled of the ith instance (we treat *unlabeled* as negative, which is an issue to be addressed shortly) in the given PU data X^{pu} (including both positive X^p and unlabeled X^u data), and n and n_u are the total numbers of training examples in X^{pu} and X^u respectively. $\widetilde{C}_i^u$ denotes the opposite distribution of C_i^u, i.e., $1 - C_i^u$ (with a slight abuse of notation). λ is a hyper-parameter for balancing the distances.

We marked three terms in Eq. 3. The adversarial learning of Eq. 3 works as follows: The first term marked by I is for minimizing the sum of the divergences between D_i^{pu} and P_i^{pu} (notice the minus sign in front). It aims to achieve the goal of helping $D(\cdot)$ recognize positive instances (it is necessary especially at the beginning of training). When optimizing $C(\cdot)$, the term marked by II minimizes the sum of the KL-divergences from C_i^u to D_i^u, indicating that $C(\cdot)$ tries to give the same probability to the input $\mathbf{x}_i^u$ as $D(\cdot)$. In this case, the instances/examples getting high probabilities (chosen by $C(\cdot)$ as positive) can also get high probabilities from $D(\cdot)$. This achieves the goal of fooling $D(\cdot)$ by $C(\cdot)$. When optimizing $D(\cdot)$, the term marked II maximizes the sum of the KL-divergences between D_i^u and C_i^u, meaning that $D(\cdot)$ tries to give low probabilities to the instances that get high probabilities from $C(\cdot)$ in order to detect 'fake' positive examples, and vice versa.

Using the first two terms can already perform the function

of PAN. An advantage of PAN is that it can optimize both positive and negative examples in the unlabeled set. However, we show that the term marked by II produces asymmetric gradients for positive and negative examples for both $D(\cdot)$ and $C(\cdot)$ in *Sec.* 4.2. That means the term marked II can cause unbalanced training between positive and negative examples and lead to high precision and low recall. To this end, we propose the term marked III which can eliminate the concern (see Sec. 4.2). Ablation study also shows the effectiveness of term III in *Sec.* 5.3. With all three terms, we build an adversarial learning approach for PU learning through the minimizing and maximizing operations discussed above, i.e., a minmax game between $D(\cdot)$ and $C(\cdot)$.

4.2 Asymmetry of $KL(D_i||C_i)$ for Positive and Negative Data

In this section, we use the gradient asymmetry for positive and negative of KL-divergence to show the need for the term marked III in Eq. 3. The term marked II in Eq. 3 will produce asymmetric gradients for positive and negative examples for both $D(\cdot)$ and $C(\cdot)$ due to asymmetry of $KL(D_i||C_i)$ for positive and negative data (explained below). If we don't have term III, the gradient of $D(\cdot)$ is:

$$\begin{aligned}\nabla_D V(D,C) &= \nabla_D[-\sum_{i=1}^{n} KL(P_i^{pu}||D_i^{pu}) + \lambda(\sum_{i=1}^{n_u} KL(D_i^u||C_i^u)] \\ &= \underbrace{\sum_{i=1}^{n_p} \frac{1}{D(x_i^p)} - \sum_{i=1}^{n_u} \frac{1}{1-D(x_i^u)}}_{(a)} + \underbrace{\sum_{i=1}^{n_u} \log \frac{D(x_i^u)(1-C(x_i^u))}{(1-D(x_i^u))C(x_i^u)}}_{(b)}\end{aligned} \tag{4}$$

where n_p and n_u are the sizes of positive and unlabeled set respectively. Term (a) is symmetric for positive and unlabeled data as they can obtain gradients with the same scale for the corresponding position, e.g., $D(x_i^p) + D(x_j^u) = 1$. But it is asymmetric for positive and negative data as positive data exist in the unlabeled set. That causes the positive being over optimized toward negative. Unfortunately, term (b) is also asymmetric for positive and negative data. We can see that the zero point of gradient for term (b) is:

$$\log \frac{D(x_i^u)(1-C(x_i^u))}{(1-D(x_i^u))C(x_i^u)} = 0 \Rightarrow\ D(x_i^u) = C(x_i^u) \tag{5}$$

which means that the zero point moves according to $C(x_i^u)$. In the worst case, if $C(\cdot)$ overfits to give a small probability to instances in the unlabeled set, $D(\cdot)$ is not easy to escape from overfitting. That will cause high precision and low recall.

The asymmetric phenomenon also occurs in Eq. 3 without term III as the gradient for $C(\cdot)$ is:

$$\begin{aligned}\nabla_C V(D,C) &= \nabla_C[-\sum_{i=1}^{n} KL(P_i^{pu}||D_i^{pu}) + \lambda(\sum_{i=1}^{n_u} KL(D_i^u||C_i^u)] \\ &= \underbrace{\sum_{i=1}^{n_u} \log \frac{C(x_i^u) - D(x_i^u)}{(1-C(x_i^u))C(x_i^u)}}_{(c)}\end{aligned} \tag{6}$$

Clearly, it is asymmetric for positive and negative data, as positives have a different gradient scale compared to negatives. And that can cause the unbalanced training problem. In this case, we propose to use the flipped distribution of C_i^u, denoted by $\widetilde{C}_i^u$, to address the problem (please refer to term III in Eq. 3). After adding the term marked III, the asymmetric gradient problem caused by the term marked II is eliminated. The gradient for $D(\cdot)$ now is $\sum_{i=1}^{n_u} \log \frac{(1-C(x_i^u))}{C(x_i^u)}$ which can be regarded as a constant when optimizing $D(\cdot)$. And the gradient for $C(\cdot)$ now is $\sum_{i=1}^{n_u} \log \frac{2D(x_i^u)-1}{(1-C(x_i^u))C(x_i^u)}$ which is symmetric between positive and negative.

4.3 Simplification of Equation 3

To facilitate the optimization of the objective function Eq. 3, we use D to denote $D(\mathbf{x}^{pu})$ and C to denote $C(\mathbf{x}^{pu})$ and simplify Eq. 3 to (see *Appendix A.1* for derivations):

$$\begin{aligned}&\min_C \max_D\ \mathrm{V}(D,C) \\ &= \underbrace{\mathbb{E}_{\mathbf{x}^p \sim P^p(\mathbf{x}^p)}[\log D(\mathbf{x}^p)] + \mathbb{E}_{\mathbf{x}^u \sim P^u(\mathbf{x}^u)}[\log(1-D(\mathbf{x}^u))]}_{\mathrm{IV}:\, -H(P^L, D(\mathbf{x}^{pu}))} \\ &+ \lambda \cdot \underbrace{\mathbb{E}_{\mathbf{x}^u \sim P^u(\mathbf{x}^u)}[\underbrace{(\log(1-C(\mathbf{x}^u)) - \log(C(\mathbf{x}^u)))}_{\mathrm{V}}(2D(\mathbf{x}^u)-1)]}_{\mathrm{VI}}\end{aligned} \tag{7}$$

where P^p denotes the distribution of the positive data. As we marked in Eq. 7, term IV is the cross entropy loss between $D(\mathbf{x}^{pu})$ and the ground-truth label distribution P^{pu} of the PU data, denoted by $H(P^{pu}, D(\mathbf{x}^{pu}))$. About the term marked VI, we elaborate with the following two points:

(1). Term VI can be viewed approximately as a policy gradient reinforcement learning algorithm for training $C(\cdot)$ but with no sampling operation, if we regard $D(\mathbf{x}^u)$ as the reward and term V as the policy. If $D(\cdot)$ outputs a high 'reward' that exceeds 0.5, meaning that $D(\cdot)$ judges the current input as a positive instance with high probability, Eq. 7 will maximize the probability of $C(\cdot)$ over the current input to fit the distribution of $D(\cdot)$. However, if $D(\cdot)$ outputs a low 'reward' below 0.5, minimizing Eq. 7 is equivalent to minimizing the probability of $C(\cdot)$ over the current input. As a consequence, the distribution of $C(\cdot)$ is made closer to $D(\cdot)$.

(2). The term marked V in Eq. 7 is a comparison game between the likelihood $\log(C(\mathbf{x}^u))$ of choosing an example or the likelihood $\log(1-C(\mathbf{x}^u))$ of not choosing an example x^u. If the choosing probability is greater than the not choosing probability, the value of term V is less than 0. Then to optimize $D(\cdot)$, maximizing Eq. 7 is equivalent to minimizing the probability of $D(\mathbf{x}^u)$. On the contrary, maximizing Eq. 7 is equivalent to maximizing the probability of $D(\mathbf{x}^u)$. Clearly, this is an adversarial learning method: for the case that term V is less than 0, $D(\cdot)$ tries to distinguish examples selected by $C(\cdot)$ (to give low probabilities to these examples). For the case that term VII is greater than 0, $D(\cdot)$ tries to give high probabilities to examples not selected by $C(\cdot)$, which helps the system move away from local training optimal.

Analysis of the Learned Classifier ($C(\cdot)$): Although the proposed PAN is quite different from the original GAN, its running follows a similar adversarial procedure. The behaviors of optimal $C(\cdot)$ and $D(\cdot)$ are that $C(\cdot)$ gives the same prediction as $D(\cdot)$ while $D(\cdot)$ cannot move away from $C(\cdot)$, which means $D(\cdot)$ also give positive (or negative) scores to examples that get positive (or negative) scores from $C(\cdot)$.[3]

[3]Positive (or negative) score means the score is $>$ (or $<$) 0.5.

Algorithm 1 PAN training by the minibatch stochastic gradient descent method.

Input: given positive training data X^p and unlabeled training data X^u;
Initialization: Randomly initialize $D(\cdot)$ and $C(\cdot)$;
for number of training steps **do**
  // Training $D(\cdot)$ k steps, we set $k = 1$.
  for k steps **do**
    • Sample a mini-batch of m positive examples $\{\mathbf{x}_1^p, \ldots, \mathbf{x}_m^p\}$ from X^p;
    • Sample a mini-batch of m unlabeled examples $\{\mathbf{x}_1^u, \ldots, \mathbf{x}_m^u\}$ from X^u;
    • Update $D(\cdot)$ by ascending its stochastic gradient:

$$\nabla_{\theta_d} \sum_{i=1}^{m} [\log D(\mathbf{x}_i^p) + \log(1 - D(\mathbf{x}_i^u)) + \lambda(\log(1 - C(\mathbf{x}_i^u)) - \log C(\mathbf{x}_i^u))(2D(\mathbf{x}_i^u) - 1)]$$

  end for
  • Sample a mini-batch of m unlabeled examples $\{\mathbf{x}_1^u, \ldots, \mathbf{x}_m^u\}$ from X^u;
  • Update $C(\cdot)$ by descending its stochastic gradient:

$$\nabla_{\theta_c} \sum_{i=1}^{m} [\lambda(\log(1 - C(\mathbf{x}_i^u)) - \log C(\mathbf{x}_i^u))(2D(\mathbf{x}_i^u) - 1)]$$

end for

The reason that $D(\cdot)$ cannot move away from $C(\cdot)$ is because that will incur errors for $D(\cdot)$ on the known positive data.

4.4 Training Algorithm of PAN

Algorithm 1 gives the training algorithm of PAN using stochastic gradient descent (SGD) for conciseness, but our method is not limited to using SGD. In this work, we use Adam for optimization. The algorithm alternately trains the discriminator $D(\cdot)$ and the classifier $C(\cdot)$. In each step or iteration, the lines between 5 to 10 (not including 10) are for training $D(\cdot)$ and the lines after 10 are for training $C(\cdot)$. The details of the algorithm are self-explanatory.

5 Experiments

We now evaluate the proposed technique PAN and compare it with state-of-the-art baselines. Three text and two image classification datasets are used in our experiments.
(1). **YELP**: a collection of online reviews from Yelp. Each review is labeled with a star rating ranging from 1 to 5. The dataset is extracted from the Yelp Dataset Challenge 2015.
(2). **RT**: a collection of online reviews from rotten tomatoes with sentiment labels *good* and *bad*.
(3). **IMDB**: another collection of online review for binary sentiment classification.
(4). **20News**: a collection of about 20,000 newsgroup posts, partitioned evenly across 20 different news topics.
(5). **MNIST**: a collection of 70,000 images of handwritten digits from 0 to 9.
(6). **CIFAR10**: a collection of 60000 32x32 colour images of 10 classes, with 6000 images per class. See [4] for all datasets.

[4]YELP: http://www.yelp.com/dataset_challenge; RT: http://www.cs.cornell.edu/ people/pabo/movie-review-data/; IMDB: https://www.imdb.com/interfaces/; 20NEWS: http://qwone.com/~jason/20Newsgroups/; MNIST: http://yann.lecun.com/exdb/mnist/; CIFAR10: http://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz .

5.1 Experiment Settings

Data Preparation: Since the five datasets are for traditional supervised learning with class labels, we need to prepare positive $\mathcal{P}$ and unlabeled $\mathcal{U}$ data for PU learning. We use two steps, after which we obtain the training and testing data for each dataset (on the left of the dataset name in Table 1).

Step 1 - *Constructing positive and negative data*. As not all datasets have 2 classes, we need to make each of them a two-class (positive and negative) dataset. RT and IMDB are already two-class datasets. For YELP, which has 5 classes, we remove the reviews with the class label of 3-stars, and split the remaining classes into two: one as the positive data (4 or 5 stars) and the other as the negative data (1 or 2 stars) (this is commonly done for sentiment classification (Liu 2012)). Following the baseline (Kiryo et al. 2017), for 20News, topics 'alt.', 'comp.', 'misc.', and 'rec.' form the positive data, and topics 'sci.', 'soc.' and 'talk.' form the negative data. For MNIST, all images labeled with even numbers form the positive data and all images labeled with odd numbers form the negative data. For CIFAR10, classes airplane, automobile, ship and truck are used as the positive data and the rest as the negative data.

Step 2 - *Creating PU learning datasets.* After step 1, we get positive and negative training data for each dataset. We then build the PU learning training dataset, which includes positive and unlabeled data as follows. For each dataset (except CIFAR10), we randomly select 10% (5% for CIFAR10 for diversity) of the positive data as the known positive data $\mathcal{P}$. We will show more results by varying the ratio in *Sec.* 5.3. The unlabeled data $\mathcal{U}$ consists of the negative data and the remaining positive data in the dataset.[5]

Baselines: We use our a-GAN and five state-of-the art representative approaches as the baselines.

a-GAN is the direct adaptation of GAN given in Sec. 3.

UPU (du Plessis, Niu, and Sugiyama 2015) is proposed as a general unbiased estimator for PU leaning that is convex for loss functions meeting certain linear-odd conditions.

NNPU (Kiryo et al. 2017) is a non-negative risk estimator for PU learning. It is more robust against overfitting, and is able to use flexible models even given limited $\mathcal{P}$ data. Note that NNPU has two versions, the linear and the MLP versions. We give the results of the MLP version as it does better.

GenPU (Hou et al. 2018) uses the GAN framework and an array of generators and discriminators to generate both positive and negative data for PU learning.

PMPU (Gong et al. 2018) is a traditional SVM based PU learning method.

NNPUSB (Kato, Teshima, and Honda 2019) is a recent algorithm that extends NNPU with an additional mechanism for handling sample selection bias.

[5]PU learning has two data sampling settings. We use the one-pass (Niu et al. 2016) or single-training-set (Elkan and Noto 2008) setting, not case-controlled or two-pass (Ward et al. 2009) setting.

Training			Testing		Dataset	a-GAN		UPU		NNPU		NNPUSB		PAN	
P-Label	Unlabel														
Pos	Pos	Neg	Pos	Neg		F	Acc	F	Acc	F	Acc	F	Acc	F	Acc
26,000	234,000	260,000	20,000	20,000	YELP	83.7	83.3	79.7	79.3	80.7	81.1	81.9	81.8	83.5	83.6
426	3,839	4,264	1086	1047	RT	66.1	58.0	50.2	56.5	62.4	58.6	66.6	59.6	66.6	64.1
1,250	11,250	12,500	12,500	12,500	IMDB	73.0	70.6	70.4	69.9	76.2	74.6	74.2	71.9	77.1	78.8
800	7,144	6,056	1,800	1,800	20News	63.5	68.7	59.1	53.1	78.5	78.1	75.9	75.6	81.1	81.0
3,000	29,492	30,508	4,926	5,074	MNIST	94.7	95.0	94.2	94.3	95.4	95.4	95.6	95.6	96.5	96.4
1,000	20,000	30,000	4,000	6,000	CIFAR10	76.6	83.0	86.2	89.0	86.1	88.8	86.6	88.6	87.2	89.7
-	-	-	-	-	Average	76.3	76.4	73.3	73.7	79.9	79.4	80.1	78.9	**82.0**	**82.3**

Table 1: Dataset details and experiment results: On the left of Dataset - training and testing data for each dataset. On the right - F-score (F) and accuracy (Acc) of PAN and baselines for the dataset

Dataset	***GenPU***	***PMPU***	***PAN***
MNIST	70.43	95.74	**96.42**
CIFAR10	66.25	81.34	**89.70**

Table 2: Comparison of PAN, GenPU and PMPU in Accuracy

Note that both UPU and NNPU need the input of the class prior probability, which is often not available in practice. In our experiments, we give them the correct class priors. Even with this favorable condition, they are still weaker than PAN. PAN does not need the class prior probability input. For UPU and NNPU, we use the open source code from the authors and a third party[6], respectively. For NNPUSB, we use the original code provided by its authors. Note also we use the same network as these baselines, including architecture, number of parameters, and the optimization method. We also give them exactly the same positive and unlabeled training data and the test data. For GenPU, we again use the code provided by the authors. For PMPU's results, since there is no source code available, we used the best results reported in its paper.

Training Details: For a fair comparison, **PAN** uses the same architecture for classifier $C(\cdot)$ as NNPU. For text, a 2-layer convolutional network (CNN), with 5 * 100 and 3 * 100 convolutions for layers 1 and 2 respectively, and 100 filters for each layer, is used as the classifier $C(\cdot)$ and discriminator $D(\cdot)$. The word embeddings are also trained by the system. For MNIST, the classifier is a 3-layer MLP (with 2 hidden layers, more specifically, d-512-256-1) as it is fairly simple. The classifier for the CIFAR10 dataset was an all convolutional net: (32 $\times$ 32 $\times$ 3)-[C(3 $\times$ 3, 96)] - C(3 $\times$ 3, 96, 2) - [C(1 $\times$ 1, 192)] - C(1 $\times$ 1, 10) - 1000 - 1000 - 1, where each input is a 32 $\times$ 32 RGB image, C(3$\times$3, 96) means 96 channels of 3$\times$3 convolutions followed by ReLU, C(3 $\times$ 3, 96, 2) means a similar layer but with stride 2, etc. We set λ in Eq. 3 and Eq. 7 to 0.0001, please see more details in *Sec.* 5.3. We also balance the impact of positive and unlabeled data for term I in Eq. 3 in training; otherwise the positive examples will be dominated by the unlabeled data. We use 1:1 ratio of positive data and unlabeled data in each mini-batch in training. The network parameters are updated using the Adam algorithm with learning rate 0.0001. For **a-GAN**, it needs pre-training of $D(\cdot)$. We use the original positive and unlabeled (regarded as negative) data to pre-train $D(\cdot)$ in order to give it the ability to classify positive and unlabeled data. We pre-train $D(\cdot)$ 3 epochs for each dataset.

[6]https://github.com/GarrettLee/nnpu_tf

5.2 Results and Analysis

The accuracy and F-score results of a-GAN, UPU, NNPU, NNPUSB and PAN are given in Table 1 (on the right side of dataset names) and the results for GenPU and PMPU are given Table 2. Since different epochs give different results, for a fair comparison, we give the average of both the best F-score (F) and best accuracy (Acc) for each system on each dataset over 200 epochs (all systems converged before 200 epochs) over 5 runs. Note that we also obtained the test accuracy, precision and recall of each epoch of each method and plotted them in Figures 2-7. However, due to the space limit, we have to put the figures in *Appendix A.3*.

The F-score for each dataset in Table 1 is measured on the positive data/class as in PU learning the user is normally interested in identifying only the positive data. The last row in the table gives the average result for each column. From the results on the right side of dataset names in Table 1, we can observe that PAN outperforms all baselines on all datasets (the last row gives the average of each algorithm for all datasets). Among the baselines, NNPU and NNPUSB are the strongest and their results are very similar. PAN outperforms them markedly. Given that PAN does not need class prior probability input, this is even more significant. The direct adaptation of GAN a-GAN is weaker than both NNPU and PAN, but is better than UPU. Although NNPUSB extends NNPU, it did not do better than NNPU. The reason could be that our data do not have sample selection bias, which NNPUSB tries to address.

Table 2 shows the results of GenPU and PMPU. Since GenPU's data generator cannot generate text, there is no result for the text datasets. GenPU's results are dramatically worse. Hou et al. (2018) noted that GenPU does well with few classes as the positive and negative, e.g., 1 class as positive and 1 class as negative. However, in our case, both positive and negative consist of many classes, which make GenPU suffer. PMPU is markedly poorer than PAN too.

Varying positive data ratio to test NNPU's sensitivity to class prior probability: We use MNIST and CIFAR10 as representatives to study this issue. For each dataset, we randomly select 1 or 2 classes in the original data to form the positive set, and the rest to form the negative set to generate 2 PU learning datasets as discussed above. Since both MNIST

Dataset	1 class as positive (1:9) - results given as F / Acc				2 classes as positive (2:8) - results given as F / Acc			
	PAN	NNPU			PAN	NNPU		
		1:9 (correct)	3:7 (wrong)	4:6 (wrong)		2:8 (correct)	4:6 (wrong)	5:5 (wrong)
MNIST	97.88 / 99.19	97.82 / 99.16	91.16 / 96.29	82.13 / 91.10	95.59 / 98.42	95.55 / 98.32	77.50 / 88.90	66.94 / 81.41
CIFAR10	51.94 / 84.73	51.46 / 84.65	48.55 / 82.79	41.99 / 76.64	59.42 / 78.35	56.45 / 78.27	57.67 / 77.61	54.62 / 73.29

Table 3: Varying the positive data and the class prior probability for NNPU.

Model	***YELP***	***20News***	***MNIST***	***CIFAR10***
PAN without term III	80.67	72.63	96.30	89.38
PAN full model	83.56	81.00	96.42	89.70

Table 4: Accuracy (%) on different datasets for PAN with or without term III.

Model	***YELP***	***20News***	***MNIST***	***CIFAR10***
PAN without term III	81.86	73.59	96.27	86.68
PAN full model	83.45	81.06	96.51	87.22

Table 5: F-score comparison on different datasets with or without term III.

and CIFAR have 10 classes, for the 1-class positive PU data, the class prior probability is 10% for positive and 90% for negative (or 1:9 for short). For the 2-class positive PU data, it is 20% for positive and 80% for negative (or 2:8). The results of NNPU and PAN are given in Table 3. We see similar improvements from PAN over NNPU with the exact class prior given to NNPU (1:9 or 2:8). For the 1:9 (respectively, 2:8) case for both MNIST and CIFAR10, if we change the class prior from the correct 1:9 (2:8) to the wrong 2:8 (3:7), NNPU's result drops are small (not in Table 3). So NNPU has some robustness. However, if we change to the wrong 3:7 or 4:6 (for the correct 1:9), and 4:6 or 5:5 (for the correct 2:8), the drops are dramatic for MNIST. For CIFAR10, they are smaller, even a small increase in F for the wrong 4:6 (correct 2:8), likely an outlier as this data is hard, but still poorer than PAN. We conclude although the class prior can be estimated, if the estimate is off, the results can be quite poor.

5.3 Additional Experiment Results

Here we discuss additional results and detailed analysis of PAN in terms of robustness, varied positive ratio and the selection of hyper-parameter λ in Eq. 3 (also 7).

Varying Known Positive Data Ratio. We show and analyse the performance of PAN with varied ratios of known positive examples from 1% to 30%, and show the accuracy and F-score of PAN and the baseline NNPU on MNIST and CIFAR10 datasets. Note that NNPU is the strongest baseline. The results are reported in Tables 7 and 8, which show that PAN can do well with different proportions of known positive data and even with extremely few known positive examples. Note that the margin between PAN and NNPU goes larger with the decrease of the ratio of the known positive examples, which indicates PAN is more effective than NNPU. The margin is smaller as the known positive ratio increases. That is because if we have enough positive data, the accuracy of PU learning methods will approach the performance of supervised binary classification. In this case, the limitation of getting good results is no longer the PU learning method, but its underlying classification method.

Model	Varying λ - results are Acc			
	0.01	0.001	0.0001	0.00001
PAN	82.70	88.90	90.30	88.23

Table 6: Sensitivity of λ on MNIST.

Model	MNIST - results given as F / Acc			
	1%	10%	20%	30%
NNPU	88.34/88.51	95.60/96.51	96.89/96.96	97.51/97.57
PAN	90.45/90.30	96.51/96.42	97.38/97.43	97.90/97.95

Table 7: Varying the ratios of known positive data on MNIST.

Model	CIFAR10 - results given as F / Acc			
	1%	10%	20%	30%
NNPU	81.41/84.22	87.84/90.14	89.05/91.04	90.01/91.66
PAN	82.70/86.10	88.37/90.77	89.74/91.85	90.65/92.49

Table 8: Varying ratios of known positive data on CIFAR10.

Hyper-parameter Selection. λ is the hyper-parameter that balances the KL-divergences. Here, we show that λ should be a small value but it is not too sensitive when it is around 0.0001 (see Table 6). In our case, we set it to 0.0001.

Ablation Study for Term III in Eq. 3. Tables 4 and 5 show PAN's ablation results in accuracy and F-score with or without term III respectively. From the two tables, we can see that adding term III improves the performance of PAN.

6 Conclusions

This paper proposed a new GAN style PU learning method, called PAN. PAN is significantly different from GAN as PAN does not use a generator but a classifier in its place. The objective function of PAN is also entirely different. It tries to solve the problem that GAN overly focuses on the positive class (real data). The new objective function is based on KL-divergence, which also has a new adversarial training method. PAN thus represents a new way of doing PU learning. Empirical evaluation using both text and image datasets showed that PAN outperformed the state-of-the-art baselines. Also importantly, PAN obtained the better results without using any class prior probability information, which most state-of-the-art baselines need. Our future work will focus on further improving PAN's accuracy.

Acknowledgments

This work was supported in part by the National Key R&D Program of China (No2020AAA0106600), NSFC 61876196, Beijing Academy of Artificial Intelligence, and an Alibaba Innovation Research (AIR) grant.

References

Bekker, J.; and Davis, J. 2020. Learning from positive and unlabeled data: a survey. *Mach. Learn.* 109(4): 719–760.

Chan, Z.; Li, J.; Yang, X.; Chen, X.; Hu, W.; Zhao, D.; and Yan, R. 2019. Modeling personalization in continuous space for response generation via augmented wasserstein autoencoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1931–1940.

Chang, S.; Zhang, Y.; Tang, J.; Yin, D.; Chang, Y.; and Hasegawa-Johnson, mark Huang, T. 2016. Positive-unlabeled learning in streaming networks. In *KDD-2016*.

Chen, Q.; Liu, Y.; Wang, Z.; Wassell, I.; and Chetty, K. 2018. Re-weighted adversarial adaptation network for unsupervised domain adaptation. In *CVPR*.

Chen, X.; Chen, W.; Chen, T.; Yuan, Y.; Gong, C.; Chen, K.; and Wang, Z. 2020. Self-pu: Self boosted and calibrated positive-unlabeled training. In *International Conference on Machine Learning*, 1510–1519. PMLR.

Chiaroni, F.; Rahal, M.-C.; Hueber, N.; and Dufaux, F. 2018. Learning with a generative adversarial network from a positive unlabeled dataset for image classification. In *ICIP*.

Denis, F.; Gilleron, R.; and Letouzey, F. 2005. Learning from positive and unlabeled examples. *Theoretical Computer Science* 348(1): 70–83.

Du Plessis, M. C.; Niu, G.; and Sugiyama, M. 2014. Analysis of learning from positive and unlabeled data. In *Advances in neural information processing systems*, 703–711.

du Plessis, M. C.; Niu, G.; and Sugiyama, M. 2015. Convex formulation for learning from positive and unlabeled data. In *ICML*.

du Plessis, M. C.; Niu, G.; and Sugiyama, M. 2017. Class-prior estimation for learning from positive and unlabeled data. *Machine Learning* .

Elkan, C.; and Noto, K. 2008. Learning classifiers from only positive and unlabeled data. In *KDD-2008*.

Gao, S.; Chen, X.; Li, P.; Ren, Z.; Bing, L.; Zhao, D.; and Yan, R. 2019a. Abstractive Text Summarization by Incorporating Reader Comments. In *AAAI*.

Gao, S.; Ren, Z.; Zhao, Y.; Zhao, D.; Yin, D.; and Yan, R. 2019b. Product-Aware Answer Generation in E-Commerce Question-Answering. In *WSDM*.

Gong, T.; Wang, Guangtao Ye, J.; Xu, Z.; and Lin, M. 2018. Margin Based PU Learning. In *AAAI-18*.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *NIPS-2014*.

Hou, M.; Chaib-Draa, B.; Li, C.; and Zhao, Q. 2018. Generative adversarial positive-unlabelled learning. *IJCAI* .

Hsieh, C.-J.; Natarajan, N.; and Dhillon, I. S. 2015. PU Learning for Matrix Completion. In *ICML*, 2445–2453.

Hsieh, Y.-G.; Niu, G.; and Sugiyama, M. 2019. Classification from Positive, Unlabeled and Biased Negative Data. In *ICML-2019*.

Jain, S.; White, M.; and Radivojac, P. 2016. Estimating the class prior and posterior from noisy positives and unlabeled data. In *NIPS-2016*.

Kato, M.; Teshima, T.; and Honda, J. 2019. Learning from Positive and Unlabeled Data with a Selection Bias. In *ICLR*.

Kiryo, R.; Niu, G.; du Plessis, M. C.; and Sugiyama, M. 2017. Positive-unlabeled learning with non-negative risk estimator. In *NIPS*.

Lee, W. S.; and Liu, B. 2003. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML*, volume 3, 448–455.

Li, W.; Guo, Q.; and Elkan, C. 2010. A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *IEEE TGRS* .

Li, X.-L.; and Liu, B. 2003. Learning to classify texts using positive & unlabeled data. In *IJCAI*.

Li, X.-L.; and Liu, B. 2005. Learning from positive and unlabeled examples with different data distributions. In *European conference on machine learning*, 218–229. Springer.

Liu, B. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5(1): 1–167.

Liu, B.; Dai, Y.; Li, X.; Lee, W. S.; and Yu, P. S. 2003. Building text classifiers using positive and unlabeled examples. In *ICDM*.

Liu, B.; Lee, W. S.; Yu, P. S.; and Li, X. 2002. Partially supervised classification of text documents. In *ICML*, volume 2, 387–394.

Liu, F.; Chen, H.; and Wu, H. 2019. Discriminative adversarial networks for positive-unlabeled learning. *arXiv preprint arXiv:1906.00642* .

Niu, G.; du Plessis, M. C.; Sakai, T.; Ma, Y.; and Sugiyama, M. 2016. Theoretical comparisons of positive-unlabeled learning against positive-negative learning. In *NIPS*, 1199–1207.

Ramaswamy, H. G.; Scott, C.; and Tewari, A. 2016. Mixture Proportion Estimation via Kernel Embedding of Distributions. In *ICML-2016*.

Sakai, T.; du Plessis, M. C.; Niu, G.; and Sugiyama, M. 2017. Semi-supervised classification based on classification from positive and unlabeled data. In *ICML*.

Sakai, T.; Niu, G.; and Sugiyama, M. 2018. Semi-supervised AUC optimization based on positive-unlabeled learning. *Machine Learning* 107(4): 767–794.

Sakai, T.; Niu, G.; and Sugiyama, M. 2020. Information-Theoretic Representation Learning for Positive-Unlabeled Classification. *Neural Computation* 33(1): 244–268.

Sansone, E.; De Natale, F. G.; and Zhou, Z.-H. 2018. Efficient training for positive unlabeled learning. *IEEE PAMI* .

Shi, H.; Pan, S.; Yang, J.; and Gong, C. 2018. Positive and Unlabeled Learning via Loss Decomposition and Centroid Estimation. In *IJCAI*.

Sutton, R.; and Barto, A. 2018. *Reinforcement learning: An introduction*. MIT Press.

Ward, G.; Hastie, T.; Barry, S.; Elith, J.; and Leathwick, J. R. 2009. Presence-only data and the EM algorithm. *Biometrics* 65(2): 554–563.

Xu, Y.; Xu, C.; Chao, X.; and Tao, D. 2017. Multi-Positive and Unlabeled Learning. In *IJCAI-17*.

Yu, H.; Han, J.; and Chang, K. C.-C. 2002. PEBL: positive example based learning for web page classification using SVM. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 239–248.

Zhang, C.; Hou, Y.; and Zhang, Y. 2020. Learning from Positive and Unlabeled Data without Explicit Estimation of Class Prior. In *AAAI*, 6762–6769.

Zhang, J.; Ding, Z.; Li, W.; and Ogunbona, P. 2018. Importance weighted adversarial nets for partial domain adaptation. In *CVPR*.