Generative Adversarial Positive-Unlabeled Learning

Ming Hou, Brahim Chaib-draa, Chao Li, Qibin Zhao

1 Tensor Learning Unit, Center for Advanced Intelligence Project, RIKEN, Japan
2 Department of Computer Science and Software Engineering, Laval University, Canada
3 Causal Inference Team, Center for Advanced Intelligence Project, RIKEN, Japan
4 School of Automation, Guangdong University of Technology, China

ming.hou@riken.jp, brahim.chaib-draa@ift.ulaval.ca, chao.li.hf@riken.jp, qibin.zhao@riken.jp

Abstract

In this work, we consider the task of classifying binary positive-unlabeled (PU) data. Existing discriminative learning based PU models attempt to seek an optimal reweighting strategy for unlabeled (U) data, so that a decent decision boundary can be found. However, given limited positive (P) data, the conventional PU models tend to suffer from overfitting when adapted to very flexible deep neural networks. In contrast, we are the first to innovate a totally new paradigm to attack the binary PU task, from the perspective of generative learning by leveraging the powerful generative adversarial networks (GAN). Our generative positive-unlabeled (GenPU) framework incorporates an array of discriminators and generators that are endowed with different roles in simultaneously producing positive and negative realistic samples. We also provide theoretical analysis to justify that, at equilibrium, GenPU is capable of recovering both positive and negative data distributions. Moreover, we show GenPU is generalizable and closely related to the semi-supervised classification. Given rather limited P data, experiments on both synthetic and real-world dataset demonstrate the effectiveness of our proposed framework. With infinite realistic and diverse samples generated from GenPU, a very flexible classifier can then be trained using deep neural networks.

1 Introduction

Positive-unlabeled (PU) classification [Denis et al., 2005] has gained great popularity in dealing with limited partially labeled data and succeeded in a broad range of applications such as automatic label identification. Yet, PU can be used for the detection of outliers in an unlabeled dataset with knowledge only from a collection of inlier data [Hido et al., 2008]. PU also finds its usefulness in ‘one-vs-rest’ classification task such as land-cover classification (urban vs non-urban) where non-urban data are too diverse to be labeled than urban data [Li et al., 2011].

The most commonly used PU approaches for binary classification can typically be categorized, in terms of the way of handling U data, into two types [Kiryo et al., 2017]. One type such as [Liu et al., 2002; Li and Liu, 2003] attempts to recognize negative samples in the U data and then feed them to classical positive-negative (PN) models. However, these approaches depend heavily on the heuristic strategies and often yield a poor solution. The other type, including [Liu et al., 2003; Lee and Liu, 2003], offers a better solution by treating U data to be N data with a decayed weight. Nevertheless, finding an optimal weight turns out to be quite costly. Most importantly, the classifiers trained based on above approaches suffer from a systematic estimation bias [Du Plessis et al., 2015; Kiryo et al., 2017].

Seeking for unbiased PU classifier, [Du Plessis et al., 2014] investigated the strategy of viewing U data as a weighted mixture of P and N data [Elkan and Noto, 2008], and introduced an unbiased risk estimator by exploiting some non-convex symmetric losses, i.e., the ramp loss. Although cancelling the bias, the non-convex loss is undesirable for PU due to the difficulty of non-convex optimization. To this end, [Du Plessis et al., 2015] proposed a more general risk estimator which is always unbiased and convex if the convex loss satisfies a linear-odd condition [Patrini et al., 2016]. Theoretically, these authors argue that the estimator yields globally optimal solution, with more appealing learning properties than the non-convex counterpart. More recently, [Kiryo et al., 2017] observed that the aforementioned unbiased risk estimators can go negative without bounding from the below, leading to serious overfitting when the classifier becomes too flexible. To fix this, they presented a non-negative biased risk estimator yet with favorable theoretical guarantees in terms of consistency, mean-squared-error reduction and estimation error. The proposed estimator is shown to be more robust against overfitting than previous unbiased ones. However, given limited P data, the overfitting issue still exists especially when very flexible deep neural network is applied.

Generative models, on the other hand, have the advantage in expressing complex data distribution. Apart from distribution density estimation, generative models are often applied...
to learn a function that is able to create more samples from the approximate distribution. Lately, a large body of successful deep generative models have emerged, especially generative adversarial networks (GAN) [Goodfellow et al., 2014; Salimans et al., 2016]. GAN intends to solve the task of generative modeling by making two agents play a game against each other. One agent named generator synthesizes fake data from random noise; the other agent, termed as discriminator, examines both real and fake data and determines whether it is real or not. Both agents keep evolving over time and get better and better at their jobs. Eventually, the generator is forced to create synthetic data which is as realistic as possible to those from the training dataset.

Inspired by the tremendous success and expressive power of GAN, we tackle the binary PU classification task by resorting to generative modeling, and propose our generative positive-unlabeled (GenPU) learning framework. Building upon GAN, our GenPU model includes an array of generators and discriminators as agents in the game. These agents are devised to play different parts in simultaneously generating positive and negative real-like samples, and thereafter a standard PN classifier can be trained on those synthetic samples. Given a small portion of labeled P data as seeds, GenPU is able to capture the underlying P and N data distributions, with the capability to create infinite diverse P and N samples. In this way, the overfitting problem of conventional PU can be greatly mitigated. Furthermore, our GenPU is generalizable in the sense that it can be established by switching to different underlying GAN variants with distance measurements (i.e., Wasserstein GAN [Arjovsky et al., 2017]) other than Jensen-Shannon divergence (JSD). As long as those variants are sophisticated to produce high-quality diverse samples, the optimal accuracy could be achieved by training a very deep neural networks.

Our main contribution (i) we are the first (to our knowledge) to invent a totally new paradigm to effectively solve the PU task through deep generative models; (ii) we provide theoretical analysis to prove that, at equilibrium, our model is capable of learning both positive and negative data distributions; (iii) we experimentally show the effectiveness of the proposed model given limited P data on both synthetic and real-world dataset; (iv) our method can be easily extended to solve the semi-supervised classification, and also opens a door to new solutions of many other weakly supervised learning tasks from the aspect of generative learning.

2 Preliminaries
2.1 Positive-Unlabeled (PU) Classification

Given as input \(d\)-dimensional random variable \(x \in \mathbb{R}^d\) and scalar random variable \(y \in \{\pm 1\}\) as class label, and let \(p(x, y)\) be the joint density, the class-conditional densities are:

\[
p_p(x) = p(x|y = 1) \quad n_n(x) = p(x|y = -1),
\]

while \(p(x)\) refers to as the unlabeled marginal density. The standard PU classification task [Ward et al., 2009] consists of a positive dataset \(\mathcal{X}_p\) and an unlabeled dataset \(\mathcal{X}_u\) with i.i.d samples drawn from \(p_p(x)\) and \(p(x)\), respectively:

\[
\mathcal{X}_p = \{x^i_p\}_{i=1}^{n_p} \sim p_p(x) \quad \mathcal{X}_u = \{x^i_u\}_{i=1}^{n_u} \sim p(x).
\]

Due to the fact that the unlabeled data can be regarded as a mixture of both positive and negative samples, the marginal density turns out to be

\[
p(x) = \pi_p p(x|y = 1) + \pi_n p(x|y = -1),
\]

where \(\pi_p = p(y = 1)\) and \(\pi_n = 1 - \pi_p\) are denoted as class-prior probability, which is usually unknown in advance and can be estimated from the given data [Jain et al., 2016]. The objective of PU task is to train a classifier on \(\mathcal{X}_p\) and \(\mathcal{X}_u\) so as to classify the new unseen pattern \(x^\text{new}\).

In contrast to PU classification, positive-negative (PN) classification assumes all negative samples, \(\mathcal{X}_n = \{x^i_n\}_{i=1}^{n_n} \sim p_n(x),\) are labeled, so that the classifier can be trained in an ordinary supervised learning fashion.

2.2 Generative Adversarial Networks (GAN)

GAN, originated in [Goodfellow et al., 2014], is one of the most recent successful generative models that is equipped with the power of producing distributional outputs. GAN obtains this capability through an adversarial competition between a generator \(G\) and a discriminator \(D\) that involves optimizing the following minimax objective function:

\[
\min_G \max_D V(G, D) = \min_G \max_D \mathbb{E}_{x \sim p_p(x)} \log(D(x)) + \mathbb{E}_{z \sim p_z(z)} \log(1 - D(G(z))),
\]

where \(p_p(x)\) represents true data distribution; \(p_z(z)\) is typically a simple prior distribution (e.g., \(N(0, 1)\)) for latent code \(z\), while a generator distribution \(p_G(x)\) associated with \(G\) is induced by the transformation \(G(z)\): \(z \rightarrow x\).

To find the optimal solution, [Goodfellow et al., 2014] employed simultaneous stochastic gradient descent (SGD) for alternately updating \(D\) and \(G\). The authors argued that, given the optimal \(D\), minimizing \(G\) is equivalent to minimizing the distribution distance between \(p_p(x)\) and \(p_G(x)\). At convergence, GAN has \(p_p(x) = p_G(x)\).

3 Generative PU Classification
3.1 Notations

Throughout the paper, \(p_p(x), p_n(x), p(x)\) denote the positive data distribution, the negative data distribution and the entire data distribution, respectively. \(D_p, D_u, D_n\) are referred to as the positive, unlabeled and negative discriminators, while \(G_p, G_n\) stand for positive and negative generators, targeting to produce real-like positive and negative samples. Correspondingly, \(p_G(x), p_p(x)\) describe the positive and negative distributions induced by the generator functions \(G_p(z)\) and \(G_n(z)\).

3.2 Proposed GenPU Model

We build our GenPU model upon GAN by leveraging its potentiality in producing realistic data, with the goal of identification of both positive and negative distributions from P and U data. Then, a decision boundary can be made by training standard PN classifier on the generated samples. Figure 1 illustrates the architecture of the proposed framework.
In brief, GenPU framework is an analogy to a minimax game comprising of two generators \( \{G_p, G_n\} \) and three discriminators \( \{D_p, D_u, D_n\} \). Guided by the adversarial supervision of \( \{D_p, D_u, D_n\} \), \( \{G_p, G_n\} \) are tasked with synthesizing positive and negative samples that are indistinguishable with the real ones drawn from \( \{p_p(x), p_u(x)\} \), respectively. As being their competitive opponents, \( \{D_p, D_u, D_n\} \) are devised to play distinct roles in instructing the learning process of \( \{G_p, G_n\} \).

More formally, the overall GenPU objective function can be decomposed, in views of \( G_p \) and \( G_n \), as follows:

\[
\Psi(G_p, G_n, D_p, D_u, D_n) = \pi_p \Phi_{G_p, D_p, D_u} + \pi_n \Phi_{G_n, D_u, D_n},
\]

where \( \pi_p \) and \( \pi_n \) corresponding to \( G_p \) and \( G_n \) are the priors for positive class and negative class, satisfying \( \pi_p + \pi_n = 1 \). Here, we assume \( \pi_p \) and \( \pi_n \) are predetermined and fixed.

The first term linked with \( G_p \) in (3) can be further split into two standard GAN components \( GAN_{G_p, D_p} \) and \( GAN_{G_n, D_u} \):

\[
\Phi_{G_p, D_p, D_u} = \lambda_p \min_{D_p} \max_{G_p} \mathcal{V}_{G_p, D_p}(G, D) + \lambda_u \min_{D_u} \max_{G_p} \mathcal{V}_{G_p, D_u}(G, D),
\]

where \( \lambda_p \) and \( \lambda_u \) are the weights balancing the relative importance of effects between \( D_p \) and \( D_u \). In particular, the value functions of \( GAN_{G_p, D_p} \) and \( GAN_{G_n, D_u} \) are

\[
\mathcal{V}_{G_p, D_p}(G, D) = E_{x \sim p_p(x)} \log(D_p(x)) + E_{z \sim p_z} \log(1 - D_p(G_p(z)))
\]

and

\[
\mathcal{V}_{G_p, D_u}(G, D) = E_{x \sim p_p(x)} \log(D_u(x)) + E_{z \sim p_z} \log(1 - D_u(G_p(z))).
\]

On the other hand, the second term linked with \( G_n \) in (3) can also be split into GAN components, namely \( GAN_{G_n, D_u} \) and \( GAN_{G_n, D_n} \):

\[
\Phi_{G_n, D_u, D_n} = \lambda_u \min_{D_u} \max_{G_n} \mathcal{V}_{G_n, D_u}(G, D) + \lambda_n \max_{D_n} \min_{G_n} \mathcal{V}_{G_n, D_n}(G, D),
\]

where\( \lambda_u \) and \( \lambda_n \) control the trade-off between \( D_u \) and \( D_n \). \( GAN_{G_n, D_u} \) also takes the form of the standard GAN with the value function

\[
\mathcal{V}_{G_n, D_u}(G, D) = E_{x \sim p_p(x)} \log(D_u(x)) + E_{z \sim p_z} \log(1 - D_u(G_n(z))).
\]

The value function of \( GAN_{G_n, D_n} \) is given by

\[
\mathcal{V}_{G_n, D_n}(G, D) = E_{x \sim p_p(x)} \log(D_n(x)) + E_{z \sim p_z} \log(1 - D_n(G_n(z))).
\]

In contrast to the ‘zero-sum’ loss applied elsewhere, the optimization of \( GAN_{G_n, D_u} \) is given by first maximizing (9) to obtain the optimal \( D_n^* \) as

\[
D_n^* = \arg \max_{D_n} \mathcal{V}_{G_n, D_n}(G, D_n^*).
\]

Such modification makes \( GAN_{G_n, D_u} \) different from the standard GAN, and this is reflected by the second term of (7).

Intuitively, (5)-(6) indicate \( G_p \) co-supervised under both \( D_p \) and \( D_u \), endeavours to minimize the distance between the induced distribution \( p_{2p}(x) \) and positive data distribution \( p_p(x) \), while striving to stay around within the whole data distribution \( p(x) \). In fact, \( G_p \) tries to deceive both discriminators by simultaneously maximizing \( D_p \)’s and \( D_u \)’s outputs on fake positive samples. As a result, the loss terms in (5) and (6) jointly guide \( p_{2p}(x) \) gradually moves towards and finally settles to \( p_p(x) \) or \( p_u(x) \).

Equations (8)-(11) suggest \( G_n \), when facing both \( D_u \) and \( D_n \), struggles to make the induced \( p_{2n}(x) \) stay away from \( p_p(x) \), and also makes its effort to force \( p_{2n}(x) \) to lie within \( p(x) \). To achieve this, the objective in (11) favors \( G_n \) to produce negative examples; this in turn helps \( D_n \) to maximize the objective in (10) to separate positive training samples from fake negative samples rather than confusing \( D_n \). Notice that, in the value function (11), \( G_n \) is designed to minimize \( D_n \)’s output instead of maximizing it when feeding \( D_n \) with fake negative samples. Consequently, \( D_n \) will send uniformly negative feedback to \( G_n \). In this way, the gradient information derived from negative feedback decreases \( p_{2n}(x) \) where the positive data region \( p_p(x) \) is large. In the meantime, the gradient signals from \( D_n \) increase \( p_{2n}(x) \) outside the positive region but still restricting \( p_{2n}(x) \) in the true data distribution \( p(x) \). This crucial effect will eventually push \( p_{2n}(x) \) away from \( p_p(x) \) but towards \( p_u(x) \).

Figure 1: Our GenPU framework. \( D_p \) receives as inputs the real positive examples from \( X_p \) and the synthetic positive examples from \( G_p; D_n \) receives as inputs the real positive examples from \( X_p \) and the synthetic negative examples from \( G_n; D_u \) receives as inputs real unlabelled examples from \( X_u \), synthetic positive examples from \( G_p \), as well as synthetic negative examples from \( G_n \) at the same time. Associated with different loss functions, \( G_p \) and \( G_n \) are designated to generate positive and negative examples, respectively.

Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18)
3.3 Theoretical Analysis

Theoretically, suppose all the \( \{G_p, G_n\} \) and \( \{D_p, D_u, D_n\} \) have enough capacity. Then the following results show that, at Nash equilibrium point of (3), the minimal JSD between the distributions induced by \( \{G_p, G_n\} \) and the data distributions \( \{p_p(x), p_n(x)\} \) are achieved, respectively, i.e., \( p_{gp}(x) = p_p(x) \) and \( p_{gn}(x) = p_n(x) \). Meanwhile, the JSD between the distribution induced by \( G_p \) and data distribution \( p_p(x) \) is maximized, i.e., \( p_{gp}(x) \) almost never overlaps with \( p_p(x) \).

**Proposition 1.** Given fixed generators \( G_p, G_n \) and known class prior \( \pi_p \), the optimal discriminators \( D_p, D_u \) and \( D_n \) for the objective in equation (3) have the following forms:

\[
D_p^*(x) = \frac{p_p(x)}{p_p(x) + p_{gp}(x)},
\]

\[
D_u^*(x) = \frac{p(x)}{p(x) + \pi_p p_{gp}(x) + \pi_n p_{gn}(x)}
\]

and

\[
D_n^*(x) = \frac{p_p(x)}{p_p(x) + p_{gn}(x)}.
\]

**Proof.** Assume that all the discriminators \( D_p, D_u \) and \( D_n \) can be optimized in functional space. Differentiating the objective \( \mathcal{V}(G, D) \) in (3) w.r.t. \( D_p, D_u \) and \( D_n \) and equating the functional derivatives to zero, we can obtain the optimal \( D_p^*, D_u^* \) and \( D_n^* \) as described above. \( \square \)

**Theorem 2.** Suppose the data distribution \( p(x) \) in the standard PU learning setting takes form of \( p(x) = \pi_p p_p(x) + \pi_n p_n(x) \), where \( p_p(x) \) and \( p_n(x) \) are well-separated. Given the optimal \( D_p^*, D_u^* \) and \( D_n^* \), the minimax optimization problem with the objective function in (3) obtains its optimal solution if

\[ p_{gp}(x) = p_p(x) \quad \text{and} \quad p_{gn}(x) = p_n(x), \]

with the objective value of \(-\pi_p \lambda_p + \lambda_u \log(4)\).

**Proof.** Substituting the optimal \( D_p^*, D_u^* \) and \( D_n^* \) into (3), the objective can be rewritten as follows:

\[
\mathcal{V}(G, D^*) = \pi_p \cdot \left\{ \lambda_p \cdot \mathbb{E}_{x \sim p_p(x)} \log \left( \frac{p_p(x)}{p_p(x) + p_{gp}(x)} \right) \right. \\
+ \lambda_u \cdot \mathbb{E}_{x \sim p_p(x)} \log \left( \frac{p(x)}{p(x) + \pi_p p_{gp}(x) + \pi_n p_{gn}(x)} \right) \\
\left. + \pi_n \cdot \mathbb{E}_{x \sim p_p(x)} \log \left( \frac{p(x)}{p(x) + \pi_p p_{gp}(x) + \pi_n p_{gn}(x)} \right) \right\}
\]

Combining the intermediate terms associated with \( \lambda_u \) using the fact \( \pi_p + \pi_n = 1 \), we reorganize (13) and arrive at

\[
G^* = \arg \min_G \mathcal{V}(G, D^*)
\]

\[
= \arg \min_G \pi_p \cdot \lambda_p \cdot \left[ 2 \cdot \text{JSD}(p_p || p_{gp}) - \log(4) \right] \\
+ \lambda_u \cdot \left[ 2 \cdot \text{JSD}(p_p || p_{gp} + \pi_n p_{gn}) - \log(4) \right] \\
- \pi_n \cdot \lambda_n \cdot \left[ 2 \cdot \text{JSD}(p_p || p_{gn}) - \log(4) \right],
\]

which peaks its minimum if

\[ p_{gp}(x) = p_p(x),\]

\[ \pi_p p_{gp}(x) + \pi_n p_{gn}(x) = p(x) \]

and for almost every \( x \) except for those in a zero measure set

\[ p_p(x) > 0 \Rightarrow p_{gn}(x) = 0, \quad p_{gn}(x) > 0 \Rightarrow p_p(x) = 0. \]

The solution to \( G = \{G_p, G_n\} \) must jointly satisfy the conditions described in (15)-(17), which implies (12) and leads to the minimum objective value of \(-\pi_p \lambda_p + \lambda_u \log(4)\). \( \square \)

The theorem reveals that approaching to Nash equilibrium is equivalent to jointly minimizing JSD \( p_p || p_{gp} + \pi_n p_{gn} \) and JSD \( p_p || p_{gp} \) and maximizing JSD \( p_p || p_{gn} \) at the same time, thus exactly capturing \( p_p \) and \( p_n \).

3.4 Connection to Semi-Supervised Classification

The goal of semi-supervised classification is to learn a classifier from positive, negative and unlabeled data. In such context, besides training sets \( X_p \) and \( X_u \), a partially labeled negative set \( X_n \) is also available, with samples drawn from negative data distribution \( p_n(x) \).

In fact, the very same architecture of GenPU can be applied to the semi-supervised classification task by just adapting the standard GAN value function to \( G_n \), then the total value function turns out to be

\[
\mathcal{V}(G, D) = \\
\pi_p \left\{ \lambda_p \left[ \mathbb{E}_{x \sim p_p(x)} \log \left( \frac{p_p(x)}{p_p(x) + p_{gp}(x)} \right) \right] \\
+ \mathbb{E}_{x \sim p_p(x)} \log \left( \frac{\pi_p p_{gp}(x) + \pi_n p_{gn}(x)}{p(x)} \right) \right\} \\
+ \pi_n \left\{ \lambda_n \left[ \mathbb{E}_{x \sim p_p(x)} \log \left( \frac{p(x)}{p(x) + \pi_p p_{gp}(x) + \pi_n p_{gn}(x)} \right) \right] \\
+ \mathbb{E}_{x \sim p_p(x)} \log \left( \frac{\pi_n p_{gn}(x)}{p(x) + \pi_p p_{gp}(x) + \pi_n p_{gn}(x)} \right) \right\}
\]

with above formulation, \( D_n \) discriminates the negative training samples from the synthetic negative samples produced by \( G_n \). Now \( G_n \) intends to fool \( D_n \) and \( D_u \) simultaneously by outputting realistic examples, just like \( G_p \) does for \( D_p \) and \( D_u \). Being attracted by both \( p(x) \) and \( p_n(x) \), the induced distribution \( p_{gn}(x) \) gradually approaches \( p_n(x) \) and finally recovers the true distribution \( p_n(x) \) of \( p(x) \). Theoretically, it is not hard to show the optimal \( G_p \) and \( G_n \) at convergence, give rise to \( p_{gp}(x) = p_p(x) \) and \( p_{gn}(x) = p_n(x) \).

4 Experimental Results

We show the efficacy of our framework by conducting experiments on synthetic and real-world images datasets. For real
data, the approaches including oracle PN, unbiased PU (UPU) [Du Plessis et al., 2015], non-negative PU (NNPU) [Kiryo et al., 2017] \(^1\) are selected for comparison. Specifically, the Oracle PN means all the training labels are available for all the P and N data, whose performance is just used as a reference for other approaches. For all the methods, the true class-prior \(\pi_p\) is assumed to be known in advance. Regarding the weights of GenPU, for simplicity, we set \(\lambda_u = 1\) and freely tune \(\lambda_p\) and \(\lambda_n\) on the validation set via grid search over the range like \([... 0.01, 0.02, ... 0.1, 0.2, ..., 1, 2, ...]\).

4.1 Synthetic Simulation

We begin our test with a toy example to visualize the learning behaviors of our GenPU. The training samples are synthesized using concentric circles, Gaussian mixtures and half moons functions with Gaussian noises added to the data (standard deviation is 0.1414). The training set contains 5000 positive and 5000 negative samples, which are then partitioned into 500 positively labelled and 9500 unlabelled samples. We establish the generators with two fully connected hidden layers and the discriminators with one hidden layer. There are 128 ReLU units contained in all hidden layers. The dimensionality of the input latent code is set to 256. Figure 2 depicts the evolution of positive and negative samples produced by GenPU through time. As expected, in all the scenarios, the induced generator distributions successfully converge to the respective true data distributions given limited P data. Notice that the Gaussian mixtures cases demonstrate the capability of our GenPU to learn a distribution with multiple submodes.

4.2 MNIST and USPS Dataset

Next, the evaluation is carried out on MNIST [LeCun et al., 1998] and USPS [LeCun et al., 1990] datasets. For MNIST, we each time select a pair of digits to construct the P and N samples, which is in contrast to that of NNPU drops significantly from 0.969 to 0.843. Spectacularly, GenPU still remains highly accurate even if only one labeled sample is provided whereas NNPU fails in this situation.

Figure 3 reports the training and test errors of the classifiers for distinct settings of \(N_l\). When \(N_l = 100\), UPU suffers from a serious overfitting to training data, whilst both NNPU and GenPU perform fairly well. As \(N_l\) goes small (i.e., 5), NNPU also starts to overfit. It should be men-

---

\(^1\)The software codes for UPU and NNPU are downloaded from https://github.com/kiryor/nnPUlearning

<table>
<thead>
<tr>
<th>Operation</th>
<th>Feature Maps</th>
<th>Nonlinearity</th>
</tr>
</thead>
<tbody>
<tr>
<td>(G_p(z)), (G_n(z)) : (z \sim \mathcal{N}(0, I))</td>
<td>100</td>
<td>leaky relu</td>
</tr>
<tr>
<td>fully connected</td>
<td>256</td>
<td>leaky relu</td>
</tr>
<tr>
<td>fully connected</td>
<td>256</td>
<td>leaky relu</td>
</tr>
<tr>
<td>fully connected</td>
<td>250/784</td>
<td>tanh</td>
</tr>
<tr>
<td>(D_p(x)), (D_n(x))</td>
<td>256/784</td>
<td>sigmoid</td>
</tr>
<tr>
<td>fully connected</td>
<td>256</td>
<td>leaky relu</td>
</tr>
<tr>
<td>fully connected</td>
<td>256</td>
<td>leaky relu</td>
</tr>
<tr>
<td>fully connected</td>
<td>1</td>
<td>sigmoid</td>
</tr>
<tr>
<td>leaky relu slope</td>
<td>0.2</td>
<td></td>
</tr>
<tr>
<td>mini-batch size for (X_p, X_u)</td>
<td>50, 100</td>
<td></td>
</tr>
<tr>
<td>learning rate</td>
<td>0.0003</td>
<td></td>
</tr>
<tr>
<td>optimizer</td>
<td>Adam(1, 0, 0.999)</td>
<td></td>
</tr>
<tr>
<td>weight, bias initialization</td>
<td>0, 0</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Specifications of network architecture and hyperparameters for USPS/MNIST dataset.
Table 2: The accuracy comparison on MNIST for \( N_i \in \{100, 50, 10, 5, 1\} \).

<table>
<thead>
<tr>
<th>( N_i )</th>
<th>( \text{Oracle PN} )</th>
<th>( \text{UPU} )</th>
<th>( \text{NNPU} )</th>
<th>( \text{GenPU} )</th>
<th>( \text{Oracle PN} )</th>
<th>( \text{UPU} )</th>
<th>( \text{NNPU} )</th>
<th>( \text{GenPU} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0.993</td>
<td>0.914</td>
<td>0.969</td>
<td>0.983</td>
<td>0.993</td>
<td>0.914</td>
<td>0.969</td>
<td>0.983</td>
</tr>
<tr>
<td>50</td>
<td>0.995</td>
<td>0.854</td>
<td>0.966</td>
<td>0.982</td>
<td>0.994</td>
<td>0.873</td>
<td>0.965</td>
<td>0.979</td>
</tr>
<tr>
<td>10</td>
<td>0.999</td>
<td>0.771</td>
<td>0.866</td>
<td>0.980</td>
<td>0.994</td>
<td>0.733</td>
<td>0.907</td>
<td>0.978</td>
</tr>
<tr>
<td>5</td>
<td>0.995</td>
<td>0.660</td>
<td>0.843</td>
<td>0.979</td>
<td>0.994</td>
<td>0.684</td>
<td>0.840</td>
<td>0.976</td>
</tr>
<tr>
<td>1</td>
<td>0.999</td>
<td>0.557</td>
<td>0.563</td>
<td>0.976</td>
<td>0.994</td>
<td>0.550</td>
<td>0.573</td>
<td>0.972</td>
</tr>
</tbody>
</table>

Table 3: The accuracy comparison on USPS for \( N_i \in \{50, 10, 5, 1\} \).

<table>
<thead>
<tr>
<th>( N_i )</th>
<th>( \text{UPU} )</th>
<th>( \text{GenPU} )</th>
<th>( \text{Oracle PN} )</th>
<th>( \text{NNPU} )</th>
<th>( \text{Oracle PN} )</th>
<th>( \text{GenPU} )</th>
<th>( \text{NNPU} )</th>
<th>( \text{UPU} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>0.890</td>
<td>0.965</td>
<td>0.965</td>
<td>0.965</td>
<td>0.960</td>
<td>0.965</td>
<td>0.945</td>
<td>0.945</td>
</tr>
<tr>
<td>10</td>
<td>0.735</td>
<td>0.890</td>
<td>0.955</td>
<td>0.955</td>
<td>0.725</td>
<td>0.920</td>
<td>0.935</td>
<td>0.935</td>
</tr>
<tr>
<td>5</td>
<td>0.620</td>
<td>0.630</td>
<td>0.930</td>
<td>0.930</td>
<td>0.630</td>
<td>0.965</td>
<td>0.925</td>
<td>0.925</td>
</tr>
<tr>
<td>1</td>
<td>0.540</td>
<td>0.610</td>
<td>0.940</td>
<td>0.940</td>
<td>0.555</td>
<td>0.635</td>
<td>0.920</td>
<td>0.920</td>
</tr>
</tbody>
</table>

Figure 4: Top: visualization of positive (left) and negative (right) digits generated using one positive ‘3’ label. Bottom: projected distributions of ‘3’ vs ‘5’, with ground truth (left) and generated (right).

Table 5: The accuracy comparison on USPS for \( N_i \in \{50, 10, 5, 1\} \).

Figure 5: Visualization of male (left) and female (right) faces generated by GenPU on CelebA 64 × 64 data.

To this end, the first 20,000 male and 20,000 female faces in CelebA are chosen as training set and the last 1,000 faces are used as test set. Then, 2,000 out of 20,000 male faces are randomly selected as positively labeled data. The architectures for generators and discriminators follow the design of the improved WGAN [Gulrajani et al., 2017]. Figure 5 illustrates the generated male and female faces, indicating GenPU is capable of producing visually appealing and diverse images that belong to the correct categories. A deep PN classifier is then trained on the synthetic images and achieves the accuracy of 87.9 which is better than 86.8 of NNPU and 62.5 of UPU.

5 Discussion

One key factor to the success of GenPU relies on the capability of underlying GAN in generating diverse samples with high quality standard. Only in this way, the ideal performance could be achieved by training a flexible classifier on those samples. However, it is widely known that the perfect training of the original GAN is quite challenging. GAN suffers from issues of mode collapse and mode oscillation, especially when high-dimensional data distribution has a large number of output modes. For this reason, the similar issue of the original GAN may happen to our GenPU when a lot of output submodes exist. Since it has empirically been shown that the original GAN equipped with JSD inclines to mimic the mode-seeking process towards convergence. Fortunately, our framework is very flexible in the sense that it can be established by switching to different underlying GAN variants with more effective distance metrics (i.e., integral probability metric (IPG) or JSD) or f-divergence. By doing so, the possible issue of the missing modes can be greatly reduced. Another possible solution is to extend single generator \( G_p \) (\( G_n \)) to multiple generators \( \{G_{p,i}\}_{i=1}^{J} \) (\( \{G_{n,i}\}_{i=1}^{J} \)) for the positive (negative) class, also by utilizing the parameter...
sharing scheme to leverage common information and reduce the computational load. Regarding other application domains, one direction of future work is to use GenPU for text classification. However, applying GAN to generating sequential semantically meaningful text is challenging, since GAN has difficulty in directly generating sequences of discrete tokens [Goodfellow et al., 2016]. More sophisticated underlying GANs need to be developed for this purpose.

Acknowledgments
This work was partially supported by JSPS KAKENHI (Grant No. 17K00326), NSFC China (Grant No. 61773129) and JST GREST (Grant No. JPMJCR1784).

References


