Generative Adversarial Positive-Unlabelled Learning

Ming Hou[1], Brahim Chaib-draa[2], Chao Li[1] and Qibin Zhao[1]
RIKEN AIP[1] Laval University[2]

2018.7.17@IJCAI

Outline

- Background
- Generative PU learning
- Experimental results

Problem Setting

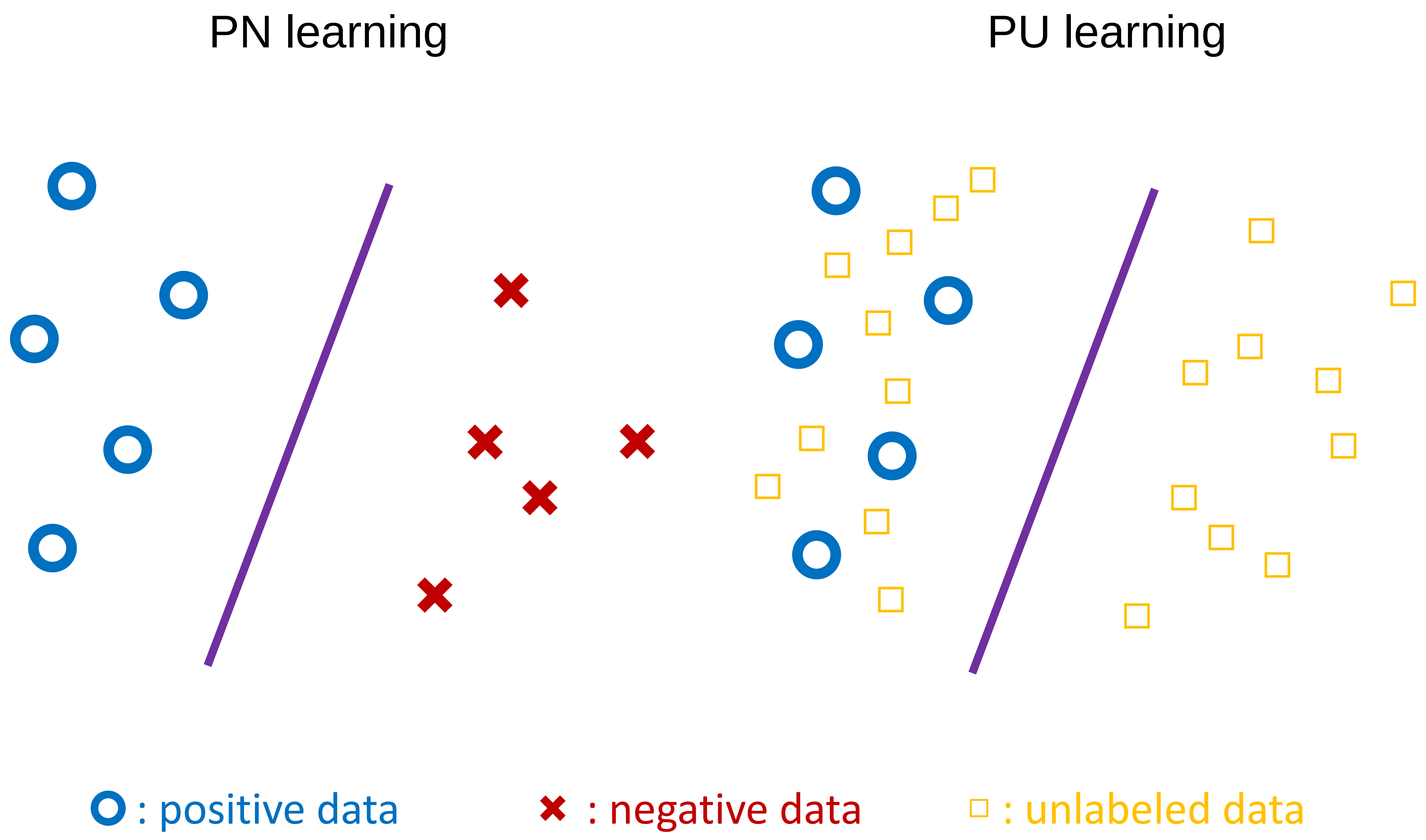


figure credit Gang Niu

Why PU learning?

- PN learning usually requires a large amount of training data
- PU learning is useful in context where
  - ✓ negative data are too expensive
  - ✓ negative data are too diverse
  - ✓ negative data are impure

- PU learning has been applied to applications
  - ✓ binary classification [Liu et al ICML02] [Li and Liu IJCAI03] [Elkan and Noto KDD08] [du Plessis NIPS14]
  - ✓ matrix completion [Hsieh et al ICML15]
  - ✓ sequential data [Li et al SDM09] [Nguyen et al IJCAI11]

Notations

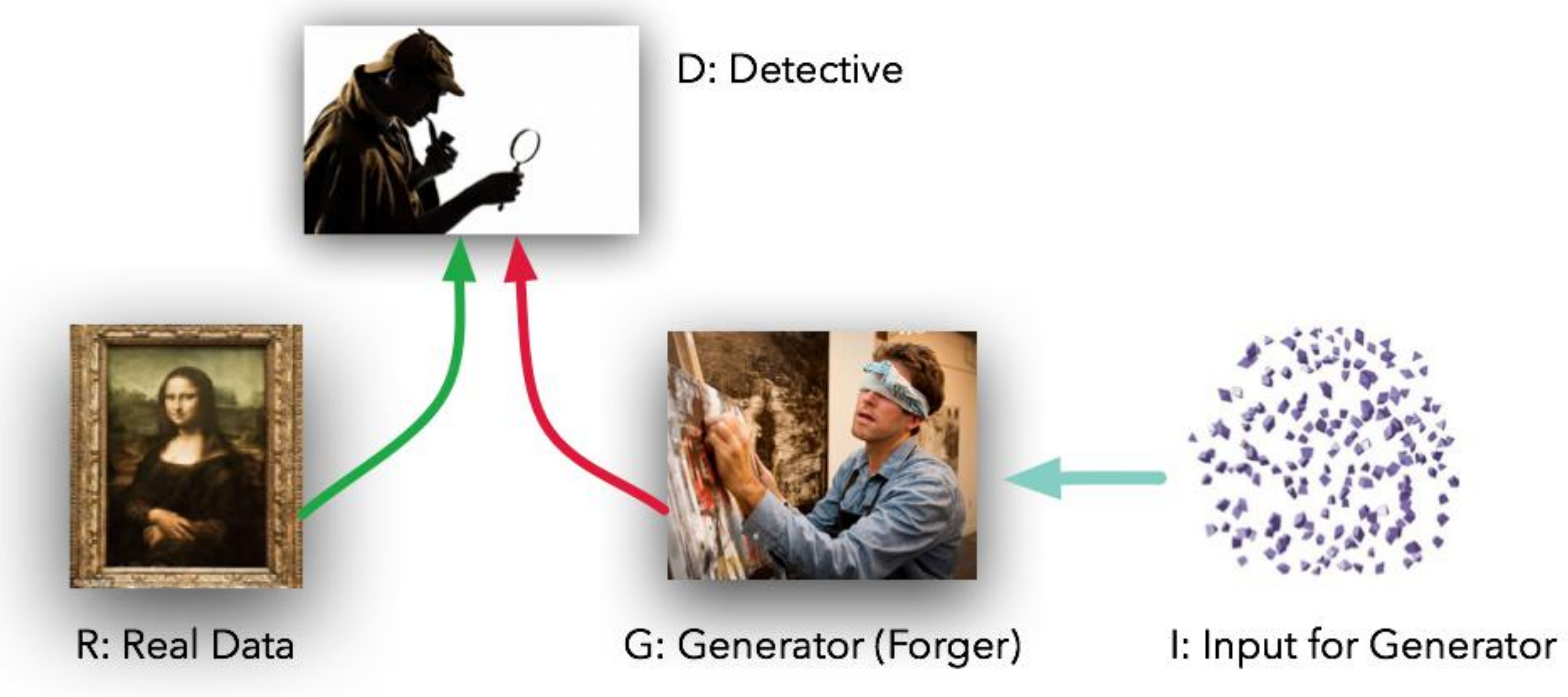
- Input & output random variables: $\mathbf{x} \in \mathbb{R}^d \quad y \in \{\pm 1\}$
- Underlying joint density: $p(\mathbf{x}, y)$
- P marginal & N marginal: $p_p(\mathbf{x}) = p(\mathbf{x}|y=1) \quad p_n(\mathbf{x}) = p(\mathbf{x}|y=-1)$
- U marginal: $p(\mathbf{x}) = \pi_p p(\mathbf{x}|y=1) + \pi_n p(\mathbf{x}|y=-1)$
- Class-prior probability: $\pi_p = p(y=1)$ assume to be known
- P data & N data: $\mathcal{X}_p = \{\mathbf{x}_p^i\}_{i=1}^{n_p} \sim p_p(\mathbf{x}) \quad \mathcal{X}_n = \{\mathbf{x}_n^i\}_{i=1}^{n_n} \sim p_n(\mathbf{x})$
- U data: $\mathcal{X}_u = \{\mathbf{x}_u^i\}_{i=1}^{n_u} \sim p(\mathbf{x})$

Recent Related Work

- Unbiased risk estimator for PU (UPU) [du Plessis et.al ICML15]
  - ✓ UPU is an unbiased & consistent estimator
  - ✓ UPU is nice for training linear-in-parameter
  - ✓ UPU seriously overfit to training deep neural networks

- Non-negative risk estimator for PU (NNPU) [Kiryo et.al NIPS17]
  - ✓ NNPU overcomes overfitting to some extent
  - ✓ NNPU is consistent but biased estimator
  - ✓ NNPU has a bias in $\mathcal{O}(exp(-\frac{1}{1/n_p+1/n_u}))$ performance might not be good for small P data

Generative Adversarial Nets (GAN)



- The minimax objective function of GAN [Goodfellow et.al NIPS14]

$$\min_G \max_D \mathcal{V}(G, D) = \min_G \max_D \; \mathbb{E}_{\mathbf{x}\sim p_x(\mathbf{x})} \log(D(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})} \log(1 - D(G(\mathbf{z})))$$

✓ binary classifier $D(\mathbf{x}) : \mathbf{x} \to [\pm 1]$

✓ transformation function $G(\mathbf{z}) : \mathbf{z} \to \mathbf{x}$

figure credit Dev Nag

Outline

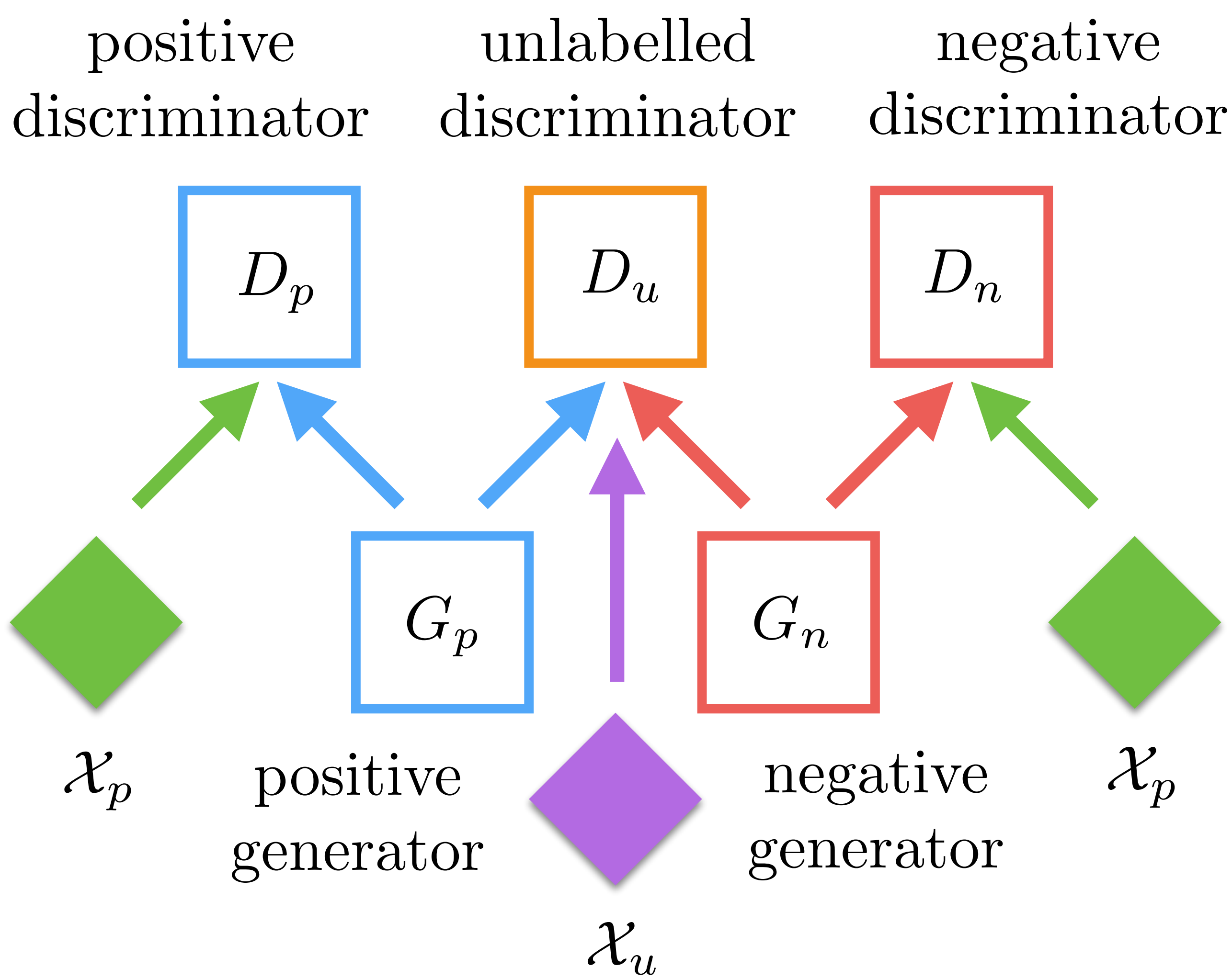
- Background
- Generative PU learning
- Experimental results

General Idea

- The goal of generative PU learning
  - ✓ to solve binary PU classification via generative model by leveraging GAN
  - ✓ to learn both positive and negative marginal distributions from P and U data

- The purposed solution
  1. couple multiple GANs to learn generator distributions using large U data and limited P data
  2. train deep PN classier on generated samples to find optimal decision boundary

General Framework



- In the viewpoint of G_p
  - ✓ guided by D_u to generate positive samples alike $\mathcal{X}_u$
  - ✓ guided by D_p to generate positive samples alike $\mathcal{X}_p$
- In the viewpoint of G_n
  - ✓ guided by D_u to generate negative samples alike $\mathcal{X}_u$
  - ✓ guided by D_n to generate negative samples **unlike** $\mathcal{X}_p$

Overall Objective Function

- The overall objective function can be decomposed, in the views of generators, as

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

- The first part corresponds to G_p can be split into two **standard** GAN components

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

✓ The value function of the first GAN_{G_p,D_p}

$$\mathcal{V}_{G_p,D_p}(G,D) = \mathbb{E}_{\mathbf{x}\sim p_p(\mathbf{x})} \log(D_p(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})} \log(1 - D_p(G_p(\mathbf{z})))$$

✓ The value function of the second GAN_{G_p,D_u}

$$\mathcal{V}_{G_p,D_u}(G,D) = \mathbb{E}_{\mathbf{x}\sim p_u(\mathbf{x})} \log(D_u(\mathbf{x})) + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})} \log(1 - D_u(G_p(\mathbf{z})))$$

Overall Objective Function Cont

- The overall objective function can be decomposed, in the views of generators, as

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

- The second part corresponds to G_n can be split into two GAN components

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

✓ The value function of the first GAN_{G_n, D_u}

$$\mathcal{V}_{G_n, D_u}(G, D) = \mathbb{E}_{\mathbf{x} \sim p_u(\mathbf{x})} \log(D_u(\mathbf{x})) + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} \log(1 - D_u(G_n(\mathbf{z})))$$

✓ The value function of the second GAN_{G_n, D_n}

$$\mathcal{V}_{G_n, D_n}(G, D) = \mathbb{E}_{\mathbf{x} \sim p_p(\mathbf{x})} \log(D_n(\mathbf{x})) + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} \log(1 - D_n(G_n(\mathbf{z})))$$

$$\boxed{\begin{aligned} D_n^\star &= \arg\max_{D_n} \mathbb{E}_{\mathbf{x} \sim p_p(\mathbf{x})} \log(D_n(\mathbf{x})) + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} \log(1 - D_n(G_n(\mathbf{z}))) \\ G_n^\star &= \arg\min_{G_n} -\mathcal{V}_{G_n, D_n^\star}(G, D_n^\star) \end{aligned}}$$

Theoretical Result

Theorem suppose the data distribution in the standard PU learning setting take the form of $p(\boldsymbol{x}) = \pi_p p_p(\boldsymbol{x}) + \pi_n p_n(\boldsymbol{x})$, where $p_p(\boldsymbol{x})$ and $p_n(\boldsymbol{x})$ are well-separated. Given the optimal discriminators, the minimax optimization problem with the overall objective function obtains its optimal solution if

$$p_{gp}(\boldsymbol{x}) = p_p(\boldsymbol{x})$$

$$p_{gn}(\boldsymbol{x}) = p_n(\boldsymbol{x})$$

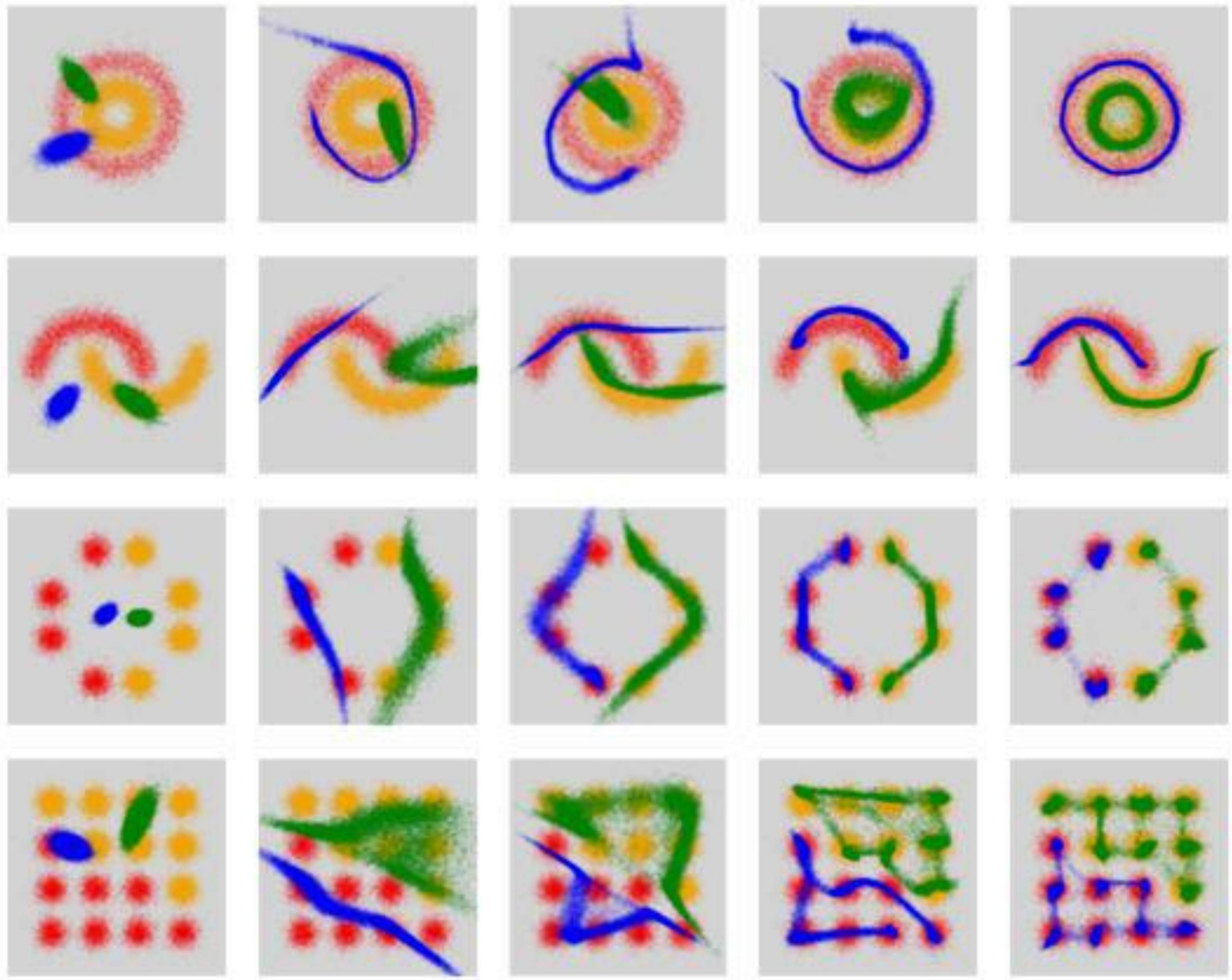
with the objective value of $-(\pi_p \lambda_p + \lambda_u) \log(4)$

Outline

- Background
- Generative PU learning
- Experimental results

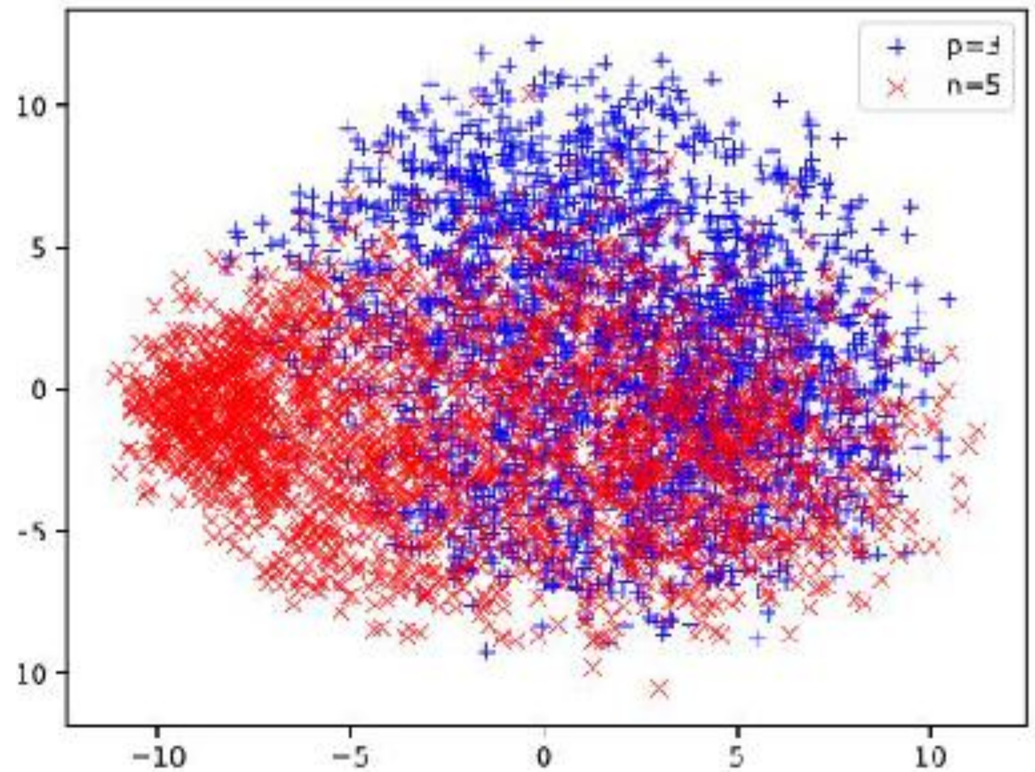
Simulation Result

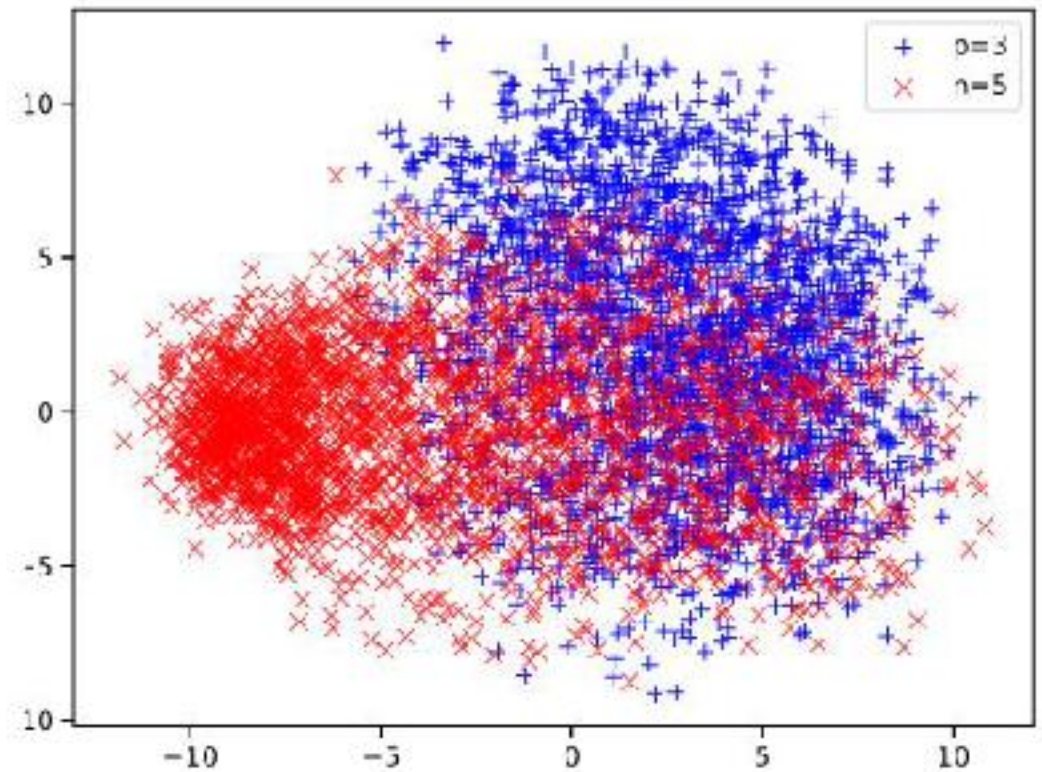
- Evolution of positive and negative samples produced by GenPU through time with 500 labeled P data and 9500 U data
- Adopt MLP as underlying GAN component



- The best accuracy with positively labeled data from 100 to 1

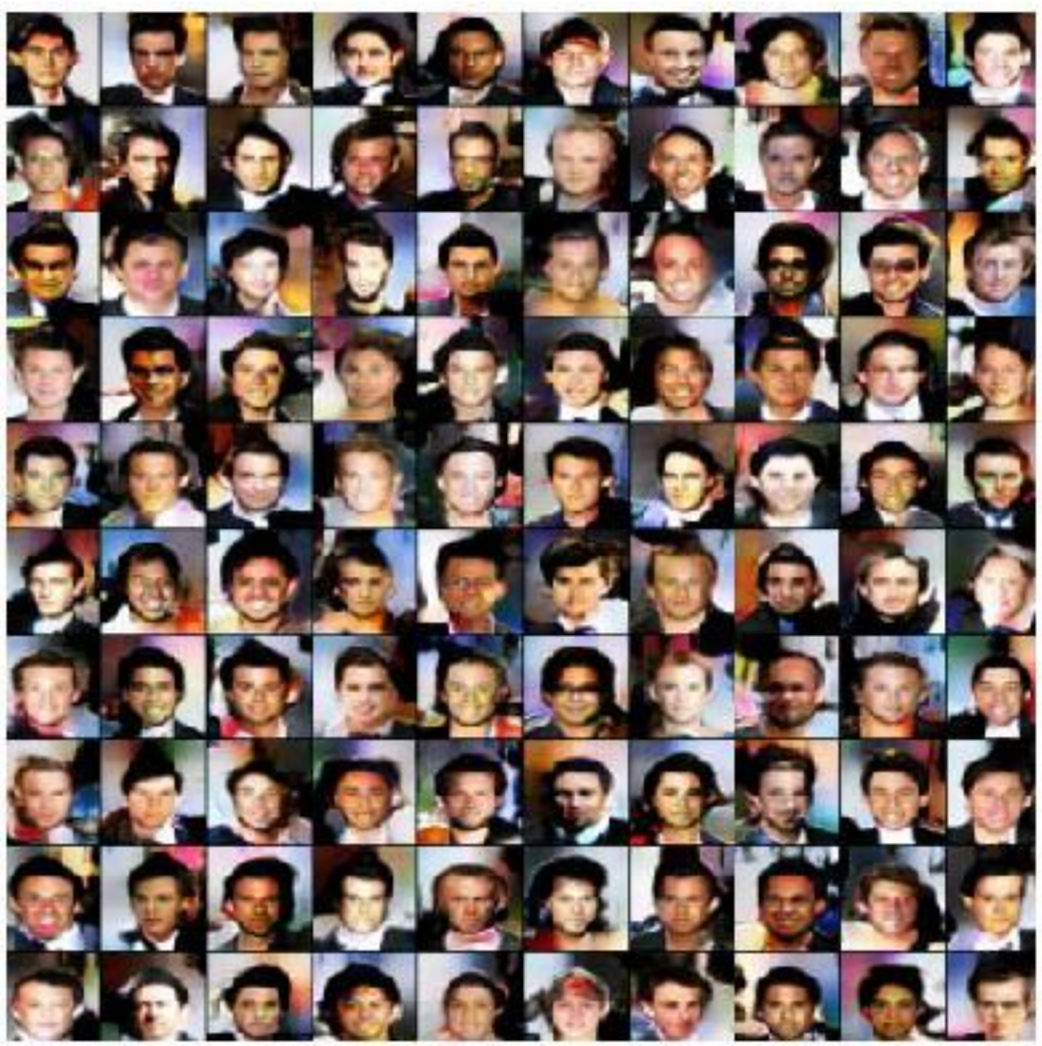
MNIST	‘3’ vs. ‘5’				‘8’ vs. ‘3’			
$N_l : N_u$	Oracle PN	UPU	NNPU	GenPU	Oracle PN	UPU	NNPU	GenPU
100 : 9900	.993	.914	.969	**.983**	.994	.932	.974	**.982**
50 : 9950	.993	.854	.966	**.982**	.994	.873	.965	**.979**
10 : 9990	.993	.711	.866	**.980**	.994	.733	.907	**.978**
5 : 9995	.993	.660	.843	**.979**	.994	.684	.840	**.976**
1 : 9999	.993	.557	.563	**.976**	.994	.550	.573	**.972**





CelebA Result

- Data dimensionality is 64 x 64 x 3 = 12288
- Experiment on 20000 male and 20000 female faces
- Randomly select 2000 male faces as labeled P data, leave rest 38000 as U data
- Adapt the improved WGAN [Gulrajani et al., 2017] as the underlying GANs
- Achieve better accuracy of 87.9 than 86.8 of NNPU and 62.5 of UPU



Conclusion

- Attacking PU task from generative model perspective using ensemble of GANs is novel and promising.
- Performance depends on the underlying GAN realization, and GenPU inherits the weakness of GAN, e.g., mode collapsing, mode oscillation.
- Applying GenPU to high-dimensional data is not easy, network architecture to be carefully designed.

Thank You!