Adversarial approach to structural estimation

Kaji, Manresa, Pouliot, 2023

Presented by Zixuan & Huixin

Date: Jan 24, 2025

Remark: GANs is a well-studied machine learning technique.

Data generating

Goodfellow et al. (2014) proposed an algorithm to generate data (x_g) that mimics the true data (x_r).
It consists of two *players*:

- **Generator** G that generates data $x_g \sim p_g$ given input $z \sim p_z$. It can be a function $G(z; \theta)$ parametrized by θ.
- **Discriminator** D that distinguishes between real data x_r and generated data x_g. It is a classifier $D(x; \lambda) \in [0, 1]$ parametrized by λ.

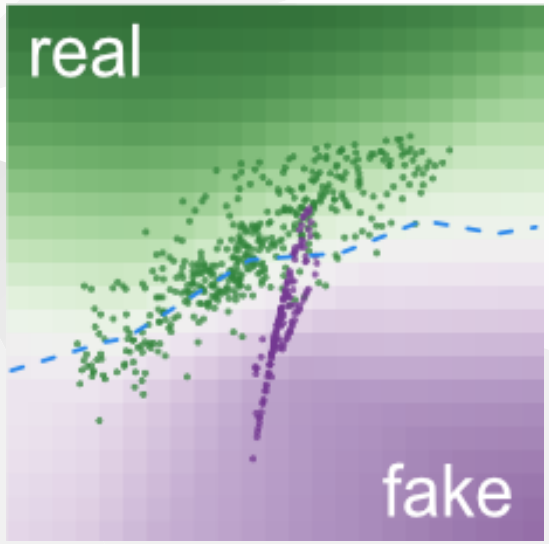
Two player game

The two *players* play against each other in the following sense.

$$\min_G \max_D L(D, G) = \mathbb{E}_{x_r \sim p_r(x)}[\log D(x_r)] + \mathbb{E}_{x_g \sim p_g(x)}[\log(1 - D(x_g))]$$

Empirically,

$$L(D, G) = \frac{1}{n} \sum_{i=1}^{n} \log D(x_i) + \frac{1}{m} \sum_{j=1}^{m} \log(1 - D(x_j))$$



Discriminator

Let's fix G first.

Given a λ, $D(x;\lambda)$ gives the probability that x is real.

Notice that

$\frac{1}{n}\sum_{i=1}^{n}\log D(x_i) + \frac{1}{m}\sum_{j=1}^{m}\log(1-D(x_j))$

is between

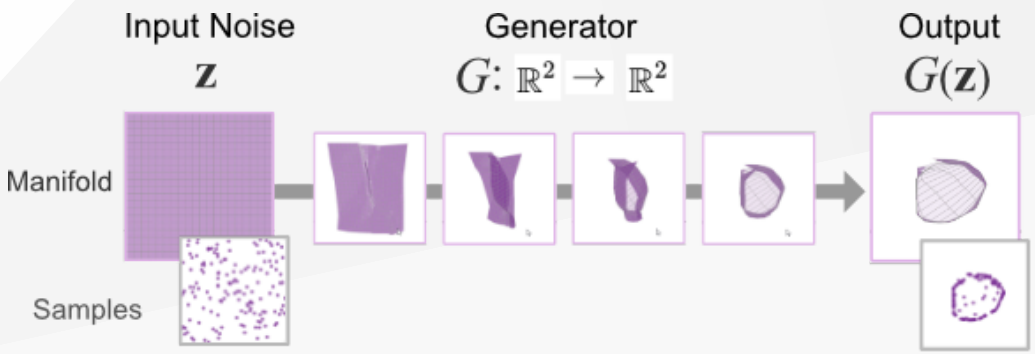
- $-\log 2$: when $D(x) = 0.5$ for all x.
- 0: when $D(x) = 1$ for real data and $D(x) = 0$ for generated data.
- Oracle loss: $D(x) = p_r(x)/(p_r(x)+p_g(x))$.

Generator

Now, fix D.
Given a parameter θ, $G(z;\theta)$ generates data that follows p_g.

Notice that finding the optimal G is equivalent to finding the optimal θ such that the generated data mimics the real data (p_g resembles p_r).

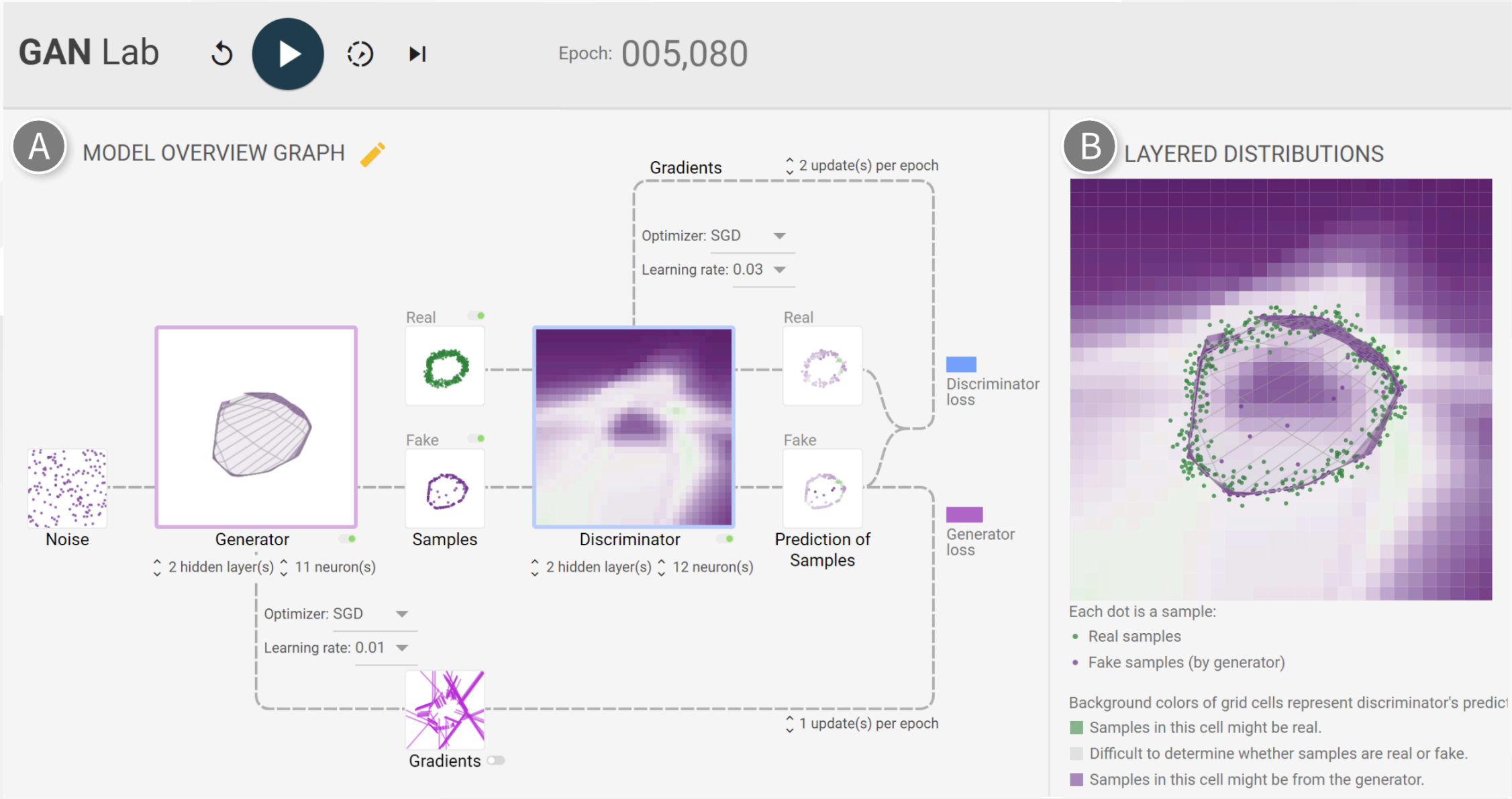
The generator is essentially estimating the true parameter θ_0 of the DGP.



Algorithm

- Initialize θ^0 and λ^0.
- While θ^k has not converged:
  i. Generate x_g from $G(z;\theta^k)$.
  ii. Train $D(x;\lambda)$ till convergence with $\{x_r\}$ and $\{x_g^k\}$. Update λ^{k+1}.
  iii. Compute the $L(D_{\lambda^{k+1}}, G_{\theta^k})$.
  iv. Minimize $L(D_{\lambda^{k+1}}, G_{\theta^k})$ with respect to θ. Update θ^{k+1}.

The two *players* play back and forth. When one improves, the other catches up.


GAN Lab
Epoch: 005,080
A
MODEL OVERVIEW GRAPH
Gradients
2 update(s) per epoch
Optimizer: SGD
Learning rate: 0.03
Real
Fake
Noise
Generator
2 hidden layer(s) 11 neuron(s)
Samples
Discriminator
2 hidden layer(s) 12 neuron(s)
Real
Fake
Prediction of Samples
Discriminator loss
Generator loss
Optimizer: SGD
Learning rate: 0.01
Gradients
1 update(s) per epoch
B
LAYERED DISTRIBUTIONS
Each dot is a sample:
Real samples
Fake samples (by generator)
Background colors of grid cells represent discriminator's predict
Samples in this cell might be real.
Difficult to determine whether samples are real or fake.
Samples in this cell might be from the generator.

Remarks

- **What is the main focus?**
  The **structural parameters** θ. The discriminator $D(x;\lambda)$ is like a propeller that pushes the generator $G(z;\theta)$ to find the optimal θ. Itself is not of interests. (The λ can be regarded as a nuisance parameter.)
- **What is the property of this generator estimator $\hat{\theta}$?**
  Consistency and efficiency both depend on the choice of D.
- **How to train D and G?**
  D is trained using Neural Networks, while G is not. While in the original GAN, both are trained with NNs.

Competing methods

> The adversarial estimator fills the gap between SMM and MLE.

MLE

$$\min_{\theta} L_{\theta} = -\frac{1}{2n} \sum_{i=1}^{n} \log p(x_i; \theta)$$

where $p(x; \theta)$ is the probability of observing the real data x.

AdE

$$\min_{\theta} M_{\theta}(D) = \frac{1}{n} \sum_{i=1}^{n} \log D(x_i) + \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G_{\theta}(Z_i)))$$

where $D(x; \lambda)$ is the probability assigned by the discriminator that x is real.

SMM

Given

$$\mathbb{E}_X(f(X|\theta)) = \theta$$

where $f(X|\theta)$ is some moment condition.
We replace this θ with a generator G such that

$$\mathbb{E}_Z(G(Z|\theta)) = \theta$$

Therefore, the theoretical moment condition we will be using is the following

$$\mathbb{E}_X(f(X|\theta) - \mathbb{E}_Z(G(Z|\theta))) = 0$$

SMM

We minimize

$$\min_{\theta} S_\theta = \left\{ \frac{1}{n} \sum_{i}^{n} \left\{ f(X_i) - \frac{1}{m} \sum_{j=1}^{m} G(Z_j|\theta) \right\} \right\}^\top \Omega \left\{ \frac{1}{n} \sum_{i}^{n} \left\{ f(X_i) - \frac{1}{m} \sum_{j=1}^{m} G(Z_j|\theta) \right\} \right\}$$

where real data has size n and generated data has size m.

Likelihood: $\mathbb{R}$

In MLE,

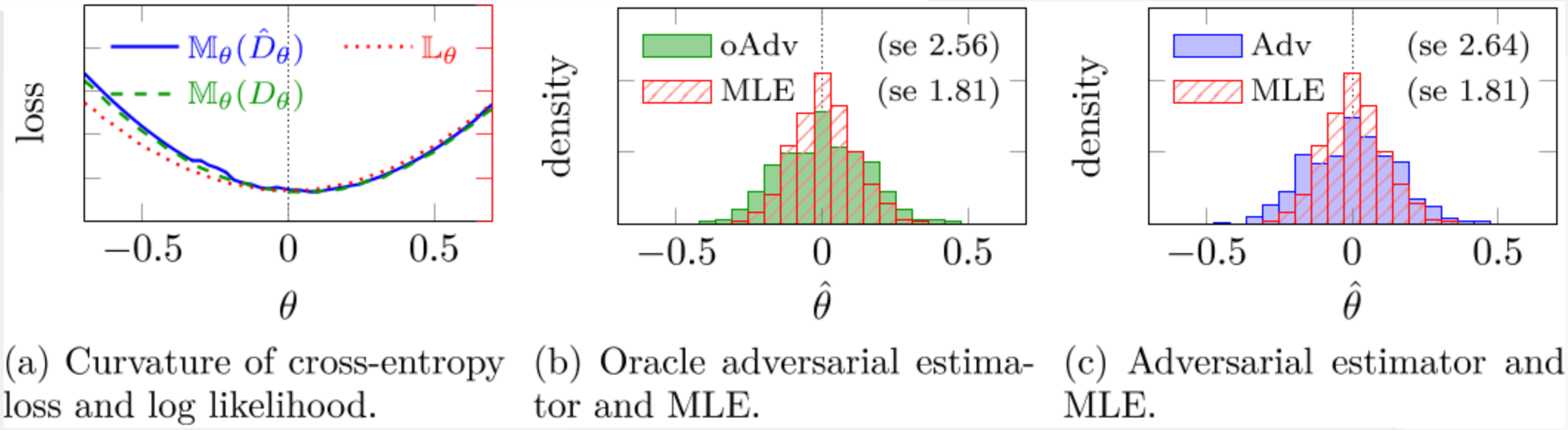
$$p(x) = \Lambda(x - \theta)(1 - \Lambda(x - \theta))$$

In AdE,

$$D(x; \lambda_0, \lambda_1)$$

We plot the L_θ **against** $M_\theta(D)$.

> An intuition behind efficiency is that the curvature of $M_\theta(\hat{D}_\theta)$ at θ_0 is proportional to the Fisher information. First, the curvature of L_θ is a quarter of the Fisher information, and so is the curvature of the oracle loss $M_\theta(D_\theta)$.



(a) Curvature of cross-entropy loss and log likelihood.

(b) Oracle adversarial estimator and MLE.

(c) Adversarial estimator and MLE.

Moments: $\mathbb{R}^d$

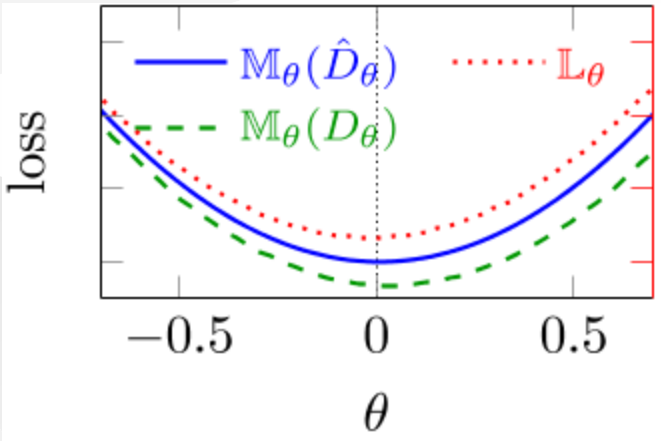
In SMM,

$$\mathbb{E}_X(f(X|\theta)) = \theta \quad f(X|\theta) \in \mathbb{R}^d$$

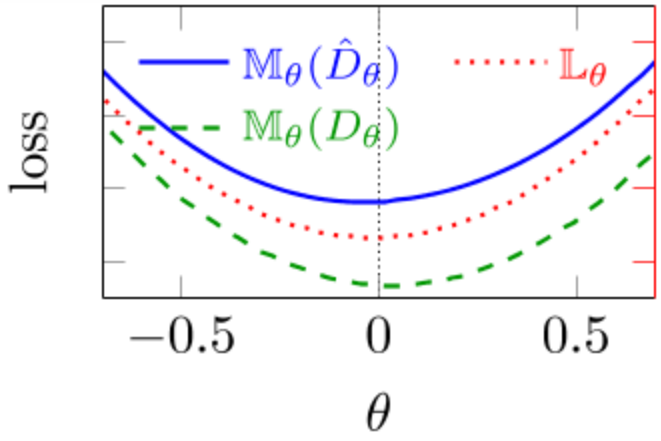
In AdE,

$$D(x, x^2, \ldots, x^d; \lambda_0, \ldots, \lambda_d)$$

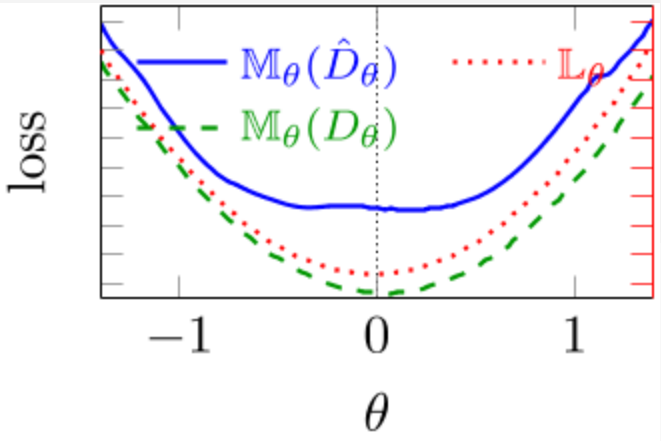
We plot L_θ **against** $M_\theta(D)$.



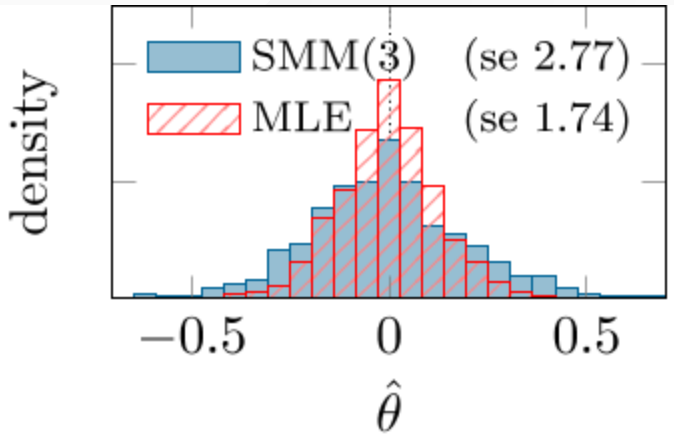
(a) Loss with 3 moments.



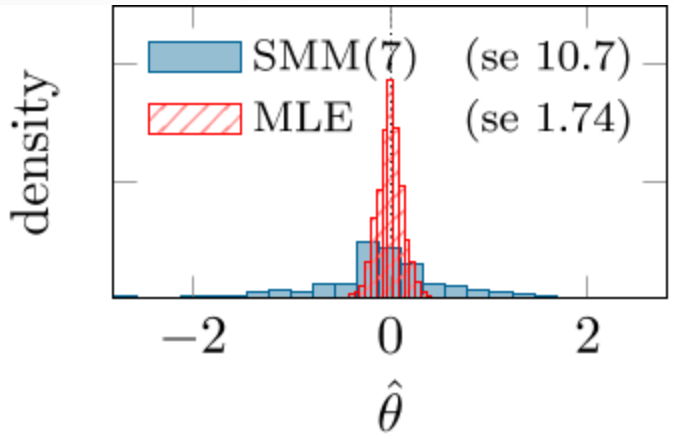
(b) Loss with 7 moments.



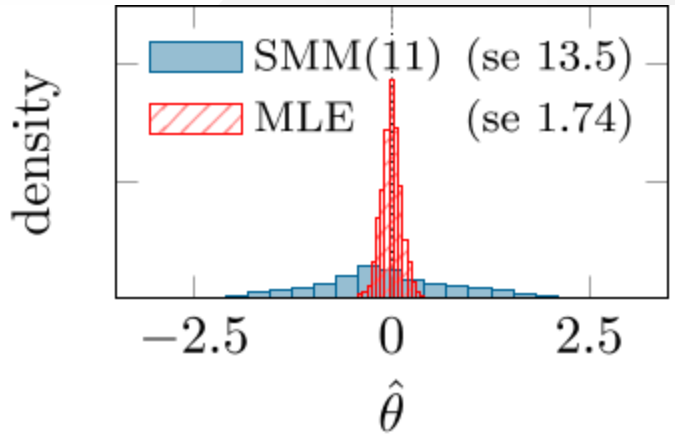
(c) Loss with 11 moments.



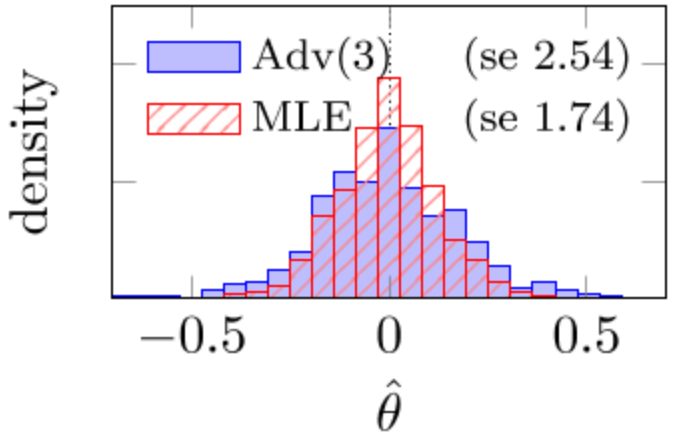
(a) SMM with 3 moments.



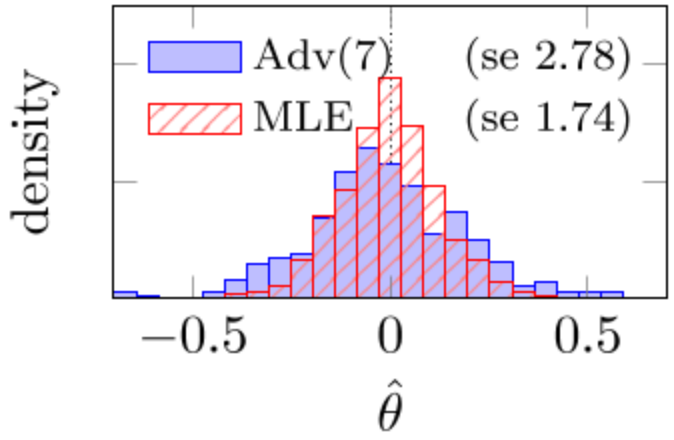
(b) SMM with 7 moments.



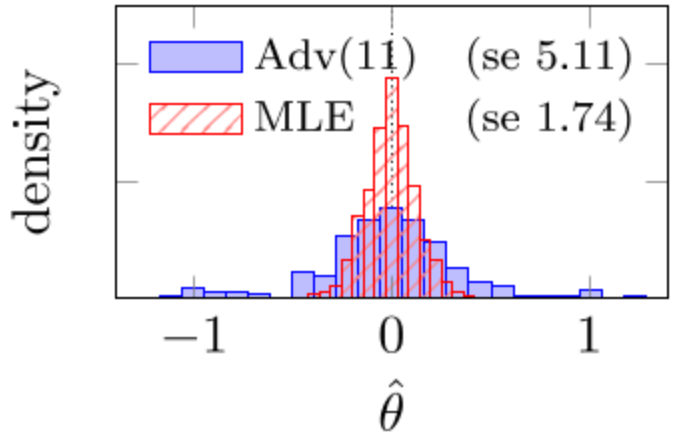
(c) SMM with 11 moments.



(d) Adversarial estimator with 3 moments.



(e) Adversarial estimator with 7 moments.



(f) Adversarial estimator with 11 moments.

Asymptotic properties

Consistency

The adversarial estimator is consistent if the estimated loss $\mathbb{M}_\theta(\hat{D}_\theta)$ converges uniformly to the oracle loss $\mathbb{M}_\theta(D_\theta)$ and $\hat{\theta}$ finds a global minimizer.

The next 3 statistical properties are derived under parametric specification of the generator $G(z;\theta)$.

Rate of convergence

$$h(\hat{\theta}, \theta_0) = O_P^*(n^{-1/2})$$

Asymptotic distribution

$$\sqrt{n}(\hat{\theta} - \theta_0) = 2\tilde{I}_{\theta_0}^{-1}\sqrt{n}\left[P_0(1 - D_{\theta_0})\dot{\ell}_{\theta_0} - P_0 D_{\theta_0}\dot{\ell}_{\theta_0} - \tilde{P}_0\tau_n\right] + o_P^*(1) \rightsquigarrow N(0, \tilde{I}_{\theta_0}^{-1} V \tilde{I}_{\theta_0}^{-1}),$$

where $V := \lim_{n\to\infty} 4P_0 D_{\theta_0}(1 - D_{\theta_0})\dot{\ell}_{\theta_0}\dot{\ell}_{\theta_0}^\top$.

Efficiency

If the model is correctly specified, the AdE is efficient.

$$\sqrt{n}(\hat{\theta} - \theta_0) = I_{\theta_0}^{-1}\sqrt{n}(P_0 - P_{\theta_0})\dot{\ell}_{\theta_0} + o_P^*(1) \rightsquigarrow N(0, I_{\theta_0}^{-1}).$$

GANs in Economics

In the paper by Athey et al. (2021),

> Using Wasserstein Generative Adversarial Networks for the design of Monte Carlo simulations

Why G is also trained using NNs?

This is because the objective is to **simulate data** that resembles the real data from a field experiment (LaLonde, 1986).

The focus is **not structural estimation** with interpretable parameters. But to **evaluate different estimators** of average treatment effect by **lots of replications.**

Simulation

For example, we want to evaluate three estimators,

$$\hat{\tau}^{\mathrm{cm}}, \hat{\tau}^{\mathrm{ht}}, \hat{\tau}^{\mathrm{dr}}$$

Given **one true data set** (field experiment), we can have

Experimental	estimate	s.e.
$\hat{\tau}^1$	1.79	0.63
$\hat{\tau}^2$	2.12	0.88
$\hat{\tau}^3$	1.79	0.57

Compare Estimators

They simulate 2000 replications (each with 1 million units) and compute the **RMSE, Bias, SDev, and Coverage** of the 95% confidence interval.

Method	RMSE	Bias	SDev	Coverage
$\hat{\tau}^1$	0.49	0.06	0.48	0.94
$\hat{\tau}^2$	0.58	0.00	0.58	0.96
$\hat{\tau}^3$	0.52	-0.06	0.51	0.88

Wasserstein Distance

The Wasserstein distance is a measure of the distance between two probability distributions over a metric space.

GANs in Finance

Paper by Xu et al.

> Using generative adversarial networks to synthesize artificial financial datasets

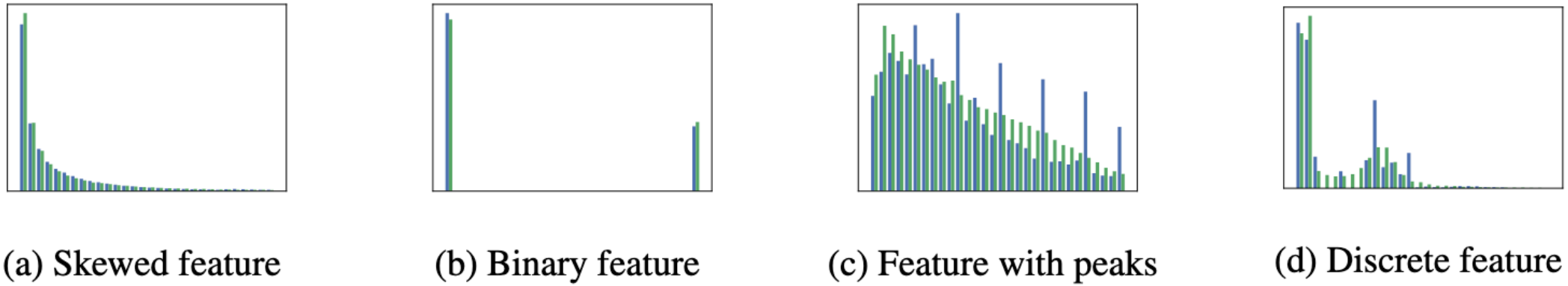
This paper evaluate the performance of GANs as a means to generate synthetic financial data.

To preserve the unique properties of financial data while maintaining customers' privacy, generating synthetic or artificial data can be a good way to address this problem.

The authors applied the following criteria to evaluate the generator:

1. Distributions of individual features in generated data match those in real data
2. Overall distributions of generated and real data match each other
3. Relationship between features and the target variable in real data is replicated in generated data

Looking at the histogram (Figure 4 in the paper), GANs were able to reproduce discrete distributions just as good as continues ones, and that for some continuous variables GANs tended to produce slightly smoothed versions of their distributions.



(a) Skewed feature (b) Binary feature (c) Feature with peaks (d) Discrete feature

Figure 4: Examples of feature histograms for real (blue) and generated (green) data. Experiment with Dataset C.

To test relationship between features and target variable in real and generated data, the authors compared two supervised ML models:

- one trained on real data,
- another one trained on data produced by GAN.

Dataset	Original data	Synthetic data
Dataset A	0.66	0.63
Dataset B	0.80	0.78
Dataset C	0.89	0.86

Table 2: The AUC scores of supervised model test for benchmark datasets.

Coding

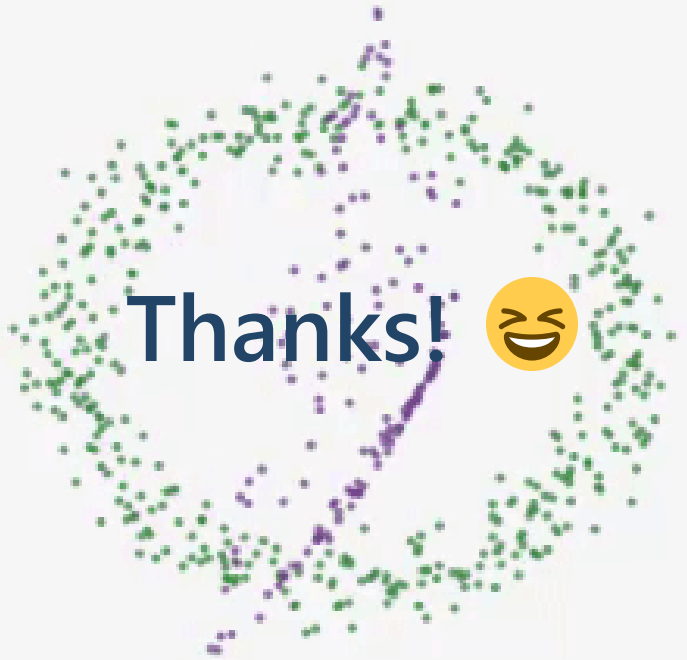
[Main Script Files – Execute each file to obtain results in the respective section]

- `main_logistic.m` : Main script file for Figures 1 and 2 in Section 3.1.1. Took 90h.
- `main_logistic2.m` : Script file for the large `m` exercise in Section 3.1.1. Took 0.5h.
- `main_logistic3.m` : Script file for the bootstrap exercise in Section 3.1.1. Took 3.5h.
- `main_misspec.m` : Main script file for Figure 3 in Section 3.1.2. Took 0.5h.
- `main_curse.m` : Main script file for Figures 4 and 5 in Section 3.1.3. Took 2h.
- `main_roy.m` : Main script file for Figures 6 and 7 in Section 3.2.1. Took 260h.
- `main_case.m` : Main script file for Figures 8 and 9 and Table 1 in Section 3.2.2. Took 300h.
- `main_case2.m` : Script file for the preparation for `'main_case3.m'`. Took 24h.
- `main_case3.m` : Script file for a faster alternative to `'main_case.m'`. Uses `'main_case2.mat'` generated by `'main_case2.m'`. Took 60h.

My goal is to replicate in `Python`.

- Well developed libraries: `PyTorch`.
- Parrallel.
- GPU.

LAYERED DISTRIBUTIONS



Each dot is a 2D data sample:

- Real samples
- Fake samples

Thanks! 😆

Appendix

- D^{θ} has good properties but is infeasible as we do not know $f_{\theta}(X)$.
- $\mathcal{D} = \mathcal{D}^{\mathcal{N}}$: multi-layer NN as a nonparametric (sieve) estimator of D^{θ}.
- **Result**: under *regularity conditions*, when using an appropriately chosen multi-layer NN as a discriminator, and as $N/H \to 0$:

$$\sqrt{N}(\hat{\theta}_G - \theta_G^0) \to N(0, I_{\theta_0}^{-1}),$$

where $I_{\theta_0}^{-1}$ is the information matrix.

- There are three main steps to obtain this result:
  i. Rate of convergence of the discriminator: $o_p(N^{-1/4})$,
  ii. Rate of convergence of the objective functions: $o_p(N^{-1/2})$,
  iii. Rate of convergence of $\hat{\theta}$: $O_p(N^{-1/2})$.