An Adversarial Approach to Structural Estimation

Tetsuya Kaji Elena Manresa Guillaume Pouliot

Presented by Zixuan, Huixin

1 Adversarial Framework

Kaji et al. (2023) propose a simulation-based method of estimation. From my understanding, this method is a specific case of the more general Generative Adversarial Networks (GANs) framework, with the restriction that the data-generating process is characterized by a structural model. Since there are existing simulation-based methods such as Simulated Method of Moments (SMM) and Simulation Maximum Likelihood (SML), the paper contributes to this strand of literature by both making comparison of performance and deriving formal statistical results. To provide a thorough understanding of Kaji et al. (2023), we will first delve into a detailed explanation of the ideas underlying GANs.

1.1 Mathematical Formulation

In this section, we lay out the basic mathematical framework of the adversarial approach. While adversarial min-max zero-sum games are not new, their implementation and adaptation in generative tasks—such as image generation—were first proposed by Goodfellow et al. (2014) and have been shown to perform exceptionally well under various settings. By iteratively training a discriminator that attempts to distinguish real data from fake data, and a generator that responds by producing increasingly realistic synthetic data, the algorithm can generate new objects that closely resemble existing ones. The mathematical formulation of this game is as follows:

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

Empirically, the loss function is:

$$L(D,G) = \frac{1}{n}\sum_{i=1}^{n} \log D(x_i) + \frac{1}{m}\sum_{j=1}^{m} \log(1 - D(x_j)). \tag{2}$$

Rather than directly explaining the objective function, I will disentangle the problem step by step. From the above expression, we can see that there are two components to optimize: the discriminator D and the generator G.

The generator works by taking input noise $z \sim p_z(z)$ and mapping it to the data space via $x_g = G(z)$. Since we know the input noise distribution $p_z(z)$ (typically normal or uniform) as well as the function $G(\cdot)$, we can compute the distribution of the generated data $p_g(x)$. At this point, we have two sets of data points: one synthetic, with a known distribution p_g, and one real, with an unknown distribution p_r.

Given sets of points $\{x_r\}_i$ and $\{x_g\}_i$, how can we train a classification model to distinguish between them? Specifically, for a given data point x, the classifier D outputs a value between 0 and 1. A *good* classifier should output values closer to 1 for real data and closer to 0 for synthetic data.

Recall that training a *good* classifier typically involves minimizing the cross-entropy loss, which is equivalent to maximizing the log-likelihood. The cross-entropy between two distributions p and q is defined as:

$$H(p,q) = -\mathbb{E}_p[\log q].$$

In the context of binary classification, for a given point x_i with label y_i, the cross-entropy loss is:

$$\begin{aligned} H(p,q) &= -\left(p(x \in \text{Real}) \log q(x \in \text{Real}) + p(x \in \text{Fake}) \log q(x \in \text{Fake})\right) \\ &= -\left(y_i \log D(x_i) + (1 - y_i) \log(1 - D(x_i))\right). \end{aligned}$$

Summing over all $n + m$ points, we obtain:

$$-\frac{1}{m+n}\sum_{i} \left(y_i \log D(x_i) + (1 - y_i) \log(1 - D(x_i))\right).$$

This is the cross-entropy loss of the classifier, which is equivalent to the log-likelihood function. Note that this is not identical to the adversarial loss $L(D,G)$ defined in Equation (2), which assigns different weights to real and synthetic points based on their counts m and n.

The reason for detailing this training process is that later we will compare the objective functions of maximum likelihood estimation (MLE), adversarial estimation (AdE), and sim-

ulated method of moments (SMM). Initially, I was confused about the difference between MLE and AdE because the inner maximization function of AdE closely resembles the log-likelihood objective of MLE. However, they serve entirely different purposes. While MLE directly maximizes the likelihood of observing the real data to estimate the true parameters, the inner likelihood maximization in AdE is an intermediate step that pushes the generator to recover the true parameters. Further details will be provided in Section 3.2.

With this in mind, let us analyze how the cross-entropy loss $L(G, D)$ changes as we vary D and G. First, fix $G \equiv p_g$ and examine how the choice of D affects $L(D, G)$. I summarize the key cases as follows:

- **Perfect** D: $D(x_i)$ assigns a value of 1 to all $x_i \in p_r$ and 0 to all $x_i \in p_g$. In this case, $D(x)$ is not a valid function because for two points $x_1 = x_2$ where $x_1 \in p_r$ and $x_2 \in p_g$, the discriminator predicts $D(x_1) = 1$ and $D(x_2) = 0$. However, this perfect classifier provides an upper bound for the loss, which is 0.

- **Ignorant** D: $D(x_i)$ assigns a value of $\frac{1}{2}$ to all x_i. In this case, the loss achieves its lower bound, which is $\log \frac{1}{2} + \log \frac{1}{2} = -2 \log 2$.

- **Oracle** D: $D(x_i) = \frac{p_r(x_i)}{p_r(x_i)+p_g(x_i)}$. This is the optimal discriminator, derived by taking the derivative of $L(D, G)$ with respect to D. Alternatively, it can be interpreted as the posterior probability of x_i being real, assuming equal priors of $\frac{1}{2}$. If the discriminator knows the distributions $p_r(x; \theta_0)$ and $p_g(x; \theta)$, this is the optimal discriminator.

- **Correctly Specified** D: $D(x_i; \lambda)$ is correctly specified if there exists a λ^* such that $D(x_i; \lambda^*) = \frac{p_r(x_i;\theta_0)}{p_r(x_i;\theta_0)+p_g(x_i;\theta)}$. In this case, the convergence of $D(x_i; \lambda)$ to the oracle $D(x_i; \lambda^*)$ is guaranteed.

- **Flexible** D: In practice, neither the oracle nor the correctly specified D is available. Instead, we often use a flexible parametric form for D, such as $D(x) = \Lambda(\lambda_0 + \lambda_1 x + \lambda_2 x^2 + \cdots)$. Alternatively, we can use a neural network classifier to approximate the oracle D. The convergence of D_{NN} to the oracle D is proved in .

Next, fix the discriminator D and examine how the choice of G affects $L(D, G)$. Note that the optimal discriminator is $D(x_i) = \frac{p_r(x_i)}{p_r(x_i)+p_g(x_i)}$. The optimal generator in this case is $G(z) \sim p_r(x)$, which renders the oracle D equivalent to the ignorant D[1]. If both the discriminator and the generator are at their optimal states, the loss function evaluates to $-2 \log 2$.

[1]Here, G is essentially saying to D: "I heard you've learned something and are no longer an ignorant young kid, but an oracle? Let me fool you again."

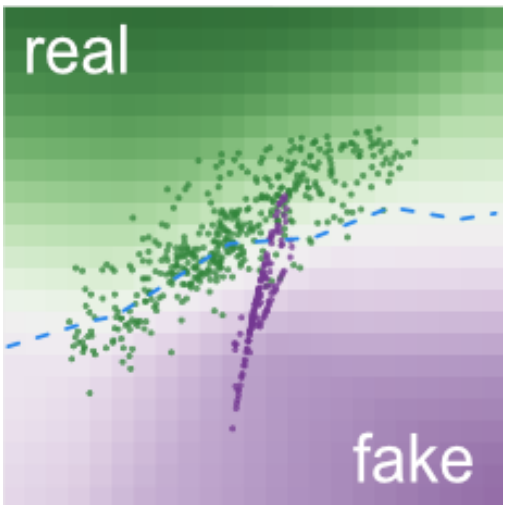


Figure 1: The real data points are in green, the fake data points are in purple. The grid color is greener when $D(x)$ is closer to 1, and more purple when closer to 0. Source: Kahng et al. (2018)

1.2 Algorithm

For ease of exposition, we first assume that $G(z;\theta)$ and $D(x;\lambda)$ are parametrized by θ and λ, respectively. We summarize it in an oversimplified way [2] as follows: One may resort to Figure 2 for a visual representation of the algorithm.

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

When training G, if the true distribution p_r is parametric (e.g., a logistic distribution parameterized by location and scale), the training of G reduces to estimating the true parameter θ_0 that characterizes p_r. If our structural model includes parameters θ with the true value θ_0 corresponding to the one that generates the observed data, the generator is trained as an estimator for θ_0. In the following section, we denote this adversarial generator estimator as AdE. One may notice that if the focus is on the generator, the discriminator can

[2] Refer to this paper Kaji et al. (2023) or the original one Goodfellow et al. (2014) for the detailed algorithm in pseudocode

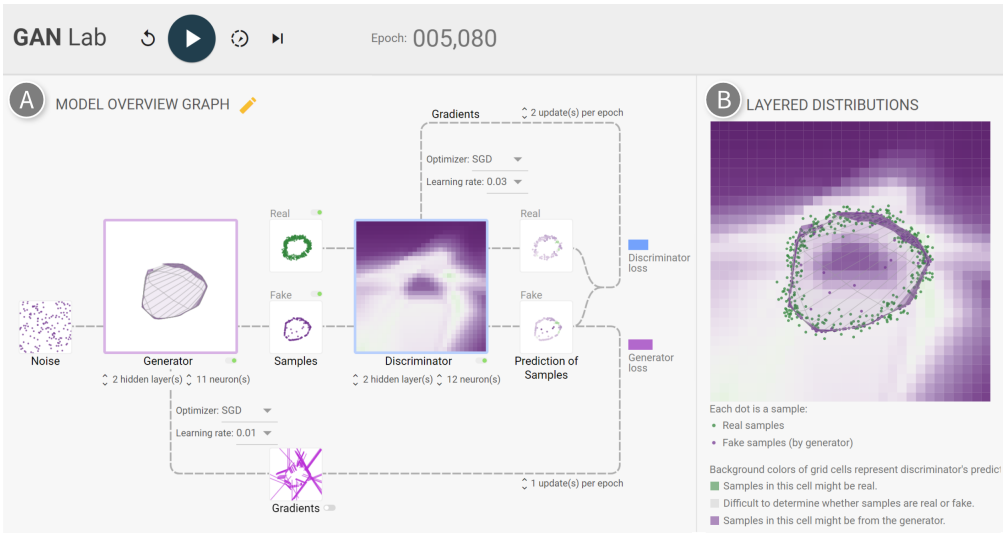


Figure 2: The upper-right loop represents the training of D in the step 2.b while the bottom loop represents the training of G in step 2. Source: Kahng et al. (2018)

be regarded as a nuisance intermediate step. However, despite being labeled as a nuisance, the discriminator plays a crucial role in determining the performance of AdE, including its convergence rate, asymptotic distribution, and efficiency. This will be further discussed in Section 3.

It is worth noting that in structural parameter estimation, G is typically parametric, whereas in the original GAN framework, G is often implemented using a neural network. This distinction arises from the different objectives of the tasks. If the goal is to generate data that mimics the true data, a neural network is a better choice than a restrictive parametric assumption. However, if the goal is to estimate the distribution of the true data (rather than generating new data), the parametric assumption provides interpretable parameters that are meaningful in an economic context.

The next section 2 will discuss two applications of GANs in economics and finance, both of which share the objective of generating new data.

2 Related Applications

2.1 GANs in Economics

The first paper to apply Wasserstein Generative Adversarial Networks (WGANs) in economics is Athey et al. (2024). Unlike traditional GANs, which aim to generate images that can fool human perception, the authors propose using GANs to generate synthetic samples

that resemble data collected from costly experiments. For example, consider a program that use conditional subsidy to encourage school enrollment. The experiment provides researchers with a single real sample of n individuals. Researchers can apply various estimators to this sample to estimate the average treatment effect. However, when **evaluating the performance of these estimators**, it is typically necessary to conduct Monte Carlo simulations using many synthetic samples. Often, the credibility of such simulations is questioned due to the specific design choices made in the simulation setup. For instance, an estimator may appear to perform better solely because of the particular simulation design. To address this issue, the authors propose using GANs to generate a large of number of synthetic samples based on which we can evaluate the estimators.

With only one real sample, we can obtain the **Estimate** and **s.e.** columns of Table 1. By simulating 2000 replications of the sample using GANs, it is possible to evaluate the **RMSE**, **Bias**, and **Coverage** of the estimators.

Method	**Estimate**	**s.e.**	**RMSE**	**Bias**	**Coverage**
$\hat{\tau}^1$	1.79	0.63	0.49	0.06	0.94
$\hat{\tau}^2$	2.12	0.88	0.58	0.00	0.96
$\hat{\tau}^3$	1.79	0.57	0.52	-0.06	0.88

Table 1: Comparison of estimators.

In addition to the difference in objectives—Athey et al. (2024) focus on generating data, while Kaji et al. (2023) focus on estimating parameters—the former uses Wasserstein Generative Adversarial Networks (WGANs), an improved version of GANs that employs the Wasserstein distance to measure the difference between the real distribution p_r and the generated distribution p_g.

From JS Divergence to Wasserstein Distance Recall the GANs objective function in Equation 1. If we substitute the oracle discriminator D_{oracle}, we obtain:

$$\begin{aligned} L(G, D_{\text{oracle}}) &= \int p_r(x) \log \frac{p_r(x)}{p_r(x) + p_g(x)} + p_g(x) \log \frac{p_g(x)}{p_r(x) + p_g(x)} \, dx \\ &= 2L_{JS}(p_r, p_g) - 2 \log 2, \end{aligned}$$

where $L_{JS}(p_r, p_g)$ is the Jensen-Shannon (JS) divergence between the real and generated distributions. This implies that if the discriminator approximates the oracle, the generator's loss function reduces to the JS divergence between p_r and p_g. However, the JS divergence has limitations, such as producing non-meaningful divergence values (e.g., ∞) when the supports

of p_r and p_g do not overlap. In contrast, the Wasserstein distance provides a meaningful measure of distance between distributions under all circumstances.

To illustrate this, we compare the loss functions for JS divergence and Wasserstein distance:

$$\min_G \max_D L(G, D) = \min_G (2L_{JS}(p_r, p_g) - 2\log 2), \tag{3}$$

$$\begin{aligned}\min_G \max_D L(G, D) &= \min_G W(p_r, p_g) \\ &= \min_G \max_{w \in W} \left(\mathbb{E}_{x_r \sim p_r}[f_w(x_r)] - \mathbb{E}_{x_g \sim p_g(x)}[f_w(x_g)]\right) \\ &= \min_G \max_{w \in W} \int (p_r(x) - p_g(x))\, f_w(x)\, dx,\end{aligned} \tag{4}$$

where $\{f_w\}_{w \in W}$ is a family of K-Lipschitz continuous functions. In this formulation, the discriminator's role is not to classify but to approximate a function f_w that captures the Wasserstein distance between p_r and p_g. This explanation closely follows Weng (2017), which provides a more detailed explanation of the Wasserstein GANs and its dual formulation.

The algorithm should follow the same structure as in Figure 2, but with modified loss function in Equation (4).

It would be interesting to see how the Wasserstein GANs perform in the context of structural estimation. This could be a potential extension of Kaji et al. (2023). However, it may be the case that it is not so much improvment to apply WGAN to economics context since we are not working with objects like images that often have non-overlapping supports.

2.2 GANs in Finance

A significant application of GANs in the finance sector is the generation of synthetic datasets. Synthetic data is particularly valuable as it can be tailored for specific purposes or scenarios where real data is either scarce or inaccessible, often due to privacy concerns or regulatory restrictions.

A prevalent issue in fraud detection using real datasets is class imbalance, where the dataset may represent a biased sample from reality, leading to suboptimal models. To address this, American Express Co. has explored the use of synthetic data to enhance their fraud detection models. Researchers at American Express published a study (Efimov et al., 2020) proposing a hybrid model combining Conditional GANs (CGANs) and Deep Regret Analytic GANs (DRAGANs) to generate synthetic datasets.

Efimov et al. (2020) assess the efficacy of GANs in generating synthetic data that repli-

cates the distribution of original data, thereby enabling the construction of machine learning models that perform comparably to those trained on real data.

The study introduces a novel GAN architecture, termed conditional DRAGAN (CDRAGAN), which integrates the strengths of CGANs and DRAGANs. Training standard GANs can be unstable due to issues such as gradient exploding or vanishing, which are exacerbated in applications involving structured data. DRAGANs, as introduced by Kodali et al. (2017), mitigate these issues by incorporating a regularization term in the discriminator's loss function to prevent sharp gradients and enhance convergence. CGANs, introduced by Mirza (2014), are designed to handle categorical features in training data by including a random noise vector z and a dummy features vector y derived from categorical features in the generator's input, while the generator's output consists solely of numerical features. The use of CDRAGAN is particularly motivated by the presence of categorical features in the dataset.

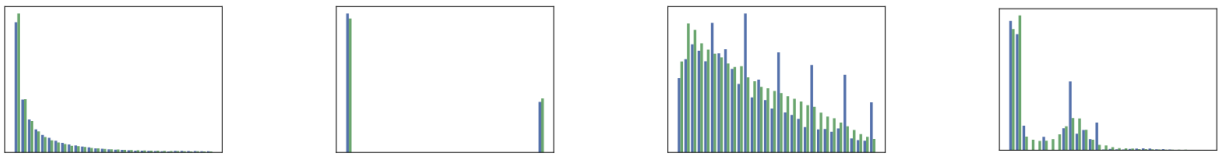
The authors applied the following criteria to evaluate the performance of the generated data:

1. The distributions of individual features in the generated data match those in the real data.

2. The overall distributions of generated and real data are similar.

3. The relationship between features and the target variable in the real data is replicated in the generated data.

Figure 3 presents histograms of real and generated data for three datasets. The GANs demonstrated the ability to reproduce discrete distributions as effectively as continuous ones, although for some continuous variables, the GANs tended to produce slightly smoothed versions of their distributions.



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

Figure 3: Examples of feature histograms for real (blue) and generated (green) data. Source: Efimov et al. (2020)

To evaluate the performance of synthetic data in training machine learning models, the authors compared the AUC scores of supervised models trained on real and generated data.

Specifically, they compared two supervised machine learning models: one trained on real data and another trained on data produced by GANs, using identical hyperparameters during the training phase. The Area Under the Curve (AUC) score was used to evaluate the performance of these models. The AUC scores for the three datasets are presented in Table 2. The models trained on synthetic data exhibited slightly lower performance on out-of-sample validation data, but the scores were very close. These results indicate that supervised models trained on generated data perform similarly to those trained on real data, demonstrating the potential of GANs to generate synthetic data for financial applications.

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. Source: Efimov et al. (2020)

3 Performance

3.1 Statistical Properties

The adversarial estimator proposed in this paper exhibits several desirable statistical properties, including consistency, a parametric rate of convergence, asymptotic normality, and efficiency under correct specification. These properties are established under a set of assumptions that ensure the estimator behaves well in both finite and large samples.

Consistency (Theorem 1) The adversarial estimator $\hat{\theta}$ converges to θ_0 if:

- The population loss $M_\theta(D_\theta)$ is uniquely minimized at θ_0,
- The classes $\{\log D_\theta\}$ and $\{\log(1 - D_\theta) \circ T_\theta\}$ satisfy uniform convergence,
- $\mathbb{M}_\theta(\hat{D}_\theta)$ uniformly converges to $M_\theta(D_\theta)$,
- $\hat{\theta}$ approximately minimizes $\mathbb{M}_\theta(\hat{D}_\theta)$.

Result: $\hat{\theta} \xrightarrow{p} \theta_0$.

Rate of Convergence (Theorem 2) Achieves $\sqrt{n}$-rate under:

- Parametric, smooth structural model (Assumption 1),
- $m/n \to \infty$ (Assumption 2),
- Orthogonality of discriminator estimation (Assumption 3),
- Quadratic curvature of $M_\theta(D_\theta)$ (Assumption 4).

Result: $h(\hat{\theta}, \theta_0) = O_p(n^{-1/2})$.

Asymptotic Distribution (Theorem 3) Under regularity conditions:

$$\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} N\left(0, \frac{1}{4}\bar{I}_{\theta_0}^{-1} V \bar{I}_{\theta_0}^{-1}\right),$$

where $V = \lim 4P_{\theta_0}[D_{\theta_0}(1 - D_{\theta_0})\dot{\ell}_{\theta_0}\dot{\ell}_{\theta_0}^\top]$.

Efficiency (Corollary 4) Under correct specification:

$$\sqrt{n}(\hat{\theta} - \theta_0) \xrightarrow{d} N(0, I_{\theta_0}^{-1}),$$

achieving the Cramér-Rao bound when $m \gg n$.

3.2 Comparison

The adversarial estimator fills the gap between SMM and MLE.

There are numerous existing methods for structural estimation. This section compares the newly proposed adversarial estimator (AdE) with maximum likelihood estimation (MLE), simulated method of moments (SMM), and simulated maximum likelihood (SML). The paper claims that AdE does not require a tractable likelihood or a correctly specified discriminator, nor does it suffer from the issues associated with an increasing number of moment conditions. To provide a comprehensive comparison, we include SML as a method for handling intractable likelihoods. Below, we outline the objective functions of each method.

MLE The maximum likelihood estimator minimizes the negative log-likelihood:

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

where $p(x;\theta)$ is the probability density of observing the real data x under the model parameterized by θ.

AdE The adversarial estimator minimizes the following objective:

$$\min_{\theta} M_\theta(D) = \frac{1}{n}\sum_{i=1}^{n} \log D(x_i) + \frac{1}{m}\sum_{j=1}^{m} \log\left(1 - D(G_\theta(z_j))\right),$$

where $D(x;\lambda)$ is the discriminator's probability that x is real, and $G_\theta(z)$ is the generator mapping noise z to synthetic data. Although the objective functions of MLE and AdE appear similar, $p(x;\theta)$ and $D(x;\lambda)$ represent fundamentally different quantities: the former is the likelihood of the data, while the latter is a discriminator estimating the probability that x belongs to the real sample.

SMM The simulated method of moments relies on moment conditions of the form $\mathbb{E}_X[f(X;\theta)] = \theta$, where $f(X;\theta)$ is a moment function. By replacing θ with a generator G, we impose the condition $\mathbb{E}_Z[G(Z;\theta)] = \theta$. The theoretical moment condition becomes:

$$\mathbb{E}_X\left[f(X;\theta) - \mathbb{E}_Z[G(Z;\theta)]\right] = 0.$$

The SMM objective function is:

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

where n and m are the sizes of the real and synthetic datasets, respectively. Consistency of SMM estimator requires $n \to \infty$ while holding m fixed. Unlike AdE, the generator G in SMM is not an estimator but a function used to construct moment conditions. Finding such a generator G that satisfies $\mathbb{E}_Z[G(Z;\theta)] = \theta$ can be challenging.

SML Simulated maximum likelihood (SML) addresses intractable likelihoods by approximating the likelihood function. Suppose the likelihood $p(x;\theta)$ is not analytically computable.

We introduce a function $G(x, z; \theta)$ such that $\mathbb{E}_z[G(x, z; \theta)] = p(x; \theta)$. Or we consider a more general form with both y_i and x_i. Since $p(y_i \mid x_i; \theta)$ is intractable, we simulate m draws z_{ir} for each (x_i, y_i) and approximate the likelihood as:

$$L_s(\theta) = \prod_{i=1}^{n} \left[\frac{1}{m} \sum_{r=1}^{m} G(y_i \mid x_i, z_{ir}; \theta) \right].$$

In log form, the SML objective becomes:

$$\ell_s(\theta) = \sum_{i=1}^{n} \log \left[\frac{1}{m} \sum_{r=1}^{m} G(y_i \mid x_i, z_{ir}; \theta) \right].$$

Under regularity conditions, the SML estimator $\hat{\theta}$ is consistent as $m \to \infty$ and $n \to \infty$.

3.3 Discussion

In the previous subsection, we outlined the objective functions for MLE, AdE, SMM, and SML. Focusing on the three simulation-based methods—AdE, SMM, and SML—we observe interesting differences in their requirements for sample size n and simulation size m:

- **SMM** requires $n \to \infty$ while holding m fixed to achieve a parametric rate of convergence.
- **SML** requires both $n \to \infty$ and $m \to \infty$ to achieve a parametric rate.
- **AdE** requires uniform convergence of $\mathbb{M}_\theta(\hat{D}_\theta)$ to $M_\theta(D_\theta)$ and, to achieve the parametric rate, $m/n \to \infty$.

One might question the necessity of AdE given the existence of SMM and SML. However, AdE offers a distinct advantage in scenarios where constructing a simulator function G is challenging. Specifically, SMM requires a function G such that $\mathbb{E}_Z[G(Z; \theta)] = \theta$, and SML requires G to satisfy $\mathbb{E}_z[G(x, z; \theta)] = p(x; \theta)$. In practice, finding such functions can be non-trivial or even infeasible. AdE circumvents this issue by leveraging a discriminator to guide the generator, eliminating the need for an explicit simulator function. This flexibility makes AdE particularly appealing for complex structural models.

That said, AdE is not without its challenges. Computational concerns arise due to the iterative nature of adversarial training, which involves alternating updates between the generator and discriminator. This process can be computationally intensive and sensitive to

hyperparameter choices, such as the learning rate and network architecture. Despite these challenges, AdE represents a powerful and flexible alternative to traditional simulation-based methods, especially in settings where constructing a simulator function is impractical.

References

Athey, S., Imbens, G. W., Metzger, J., and Munro, E. (2024). Using wasserstein generative adversarial networks for the design of monte carlo simulations. *Journal of Econometrics*, 240(2):105076.

Efimov, D., Xu, D., Kong, L., Nefedov, A., and Anandakrishnan, A. (2020). Using generative adversarial networks to synthesize artificial financial datasets.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Kahng, M., Thorat, N., Chau, D. H., Viégas, F. B., and Wattenberg, M. (2018). Gan lab: Understanding complex deep generative models using interactive visual experimentation. *IEEE transactions on visualization and computer graphics*, 25(1):310–320.

Kaji, T., Manresa, E., and Pouliot, G. (2023). An adversarial approach to structural estimation. *Econometrica*, 91(6):2041–2063.

Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). On convergence and stability of gans. *arXiv preprint arXiv:1705.07215*.

Mirza, M. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Weng, L. (2017). From gan to wgan. *lilianweng.github.io*.