NBER WORKING PAPER SERIES

USING WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS FOR THE DESIGN OF MONTE CARLO SIMULATIONS

Susan Athey
Guido W. Imbens
Jonas Metzger
Evan M. Munro

Working Paper 26566
http://www.nber.org/papers/w26566

NATIONAL BUREAU OF ECONOMIC RESEARCH
1050 Massachusetts Avenue
Cambridge, MA 02138
December 2019

Financial support from the Sloan Foundation and the Office of Naval Research under grant N00014-17-1-2131 is gratefully acknowledged, as well as gift funding through Schmidt Futures and the Golub Capital Social Impact Lab at Stanford. The views expressed herein are those of the authors and do not necessarily reflect the views of the National Bureau of Economic Research.

At least one co-author has disclosed a financial relationship of potential relevance for this research. Further information is available online at http://www.nber.org/papers/w26566.ack

Using Wasserstein Generative Adversarial Networks for the Design of Monte Carlo Simulations
Susan Athey, Guido W. Imbens, Jonas Metzger, and Evan M. Munro
NBER Working Paper No. 26566
December 2019
JEL No. C15

ABSTRACT

When researchers develop new econometric methods it is common practice to compare the performance of the new methods to those of existing methods in Monte Carlo studies. The credibility of such Monte Carlo studies is often limited because of the freedom the researcher has in choosing the design. In recent years a new class of generative models emerged in the machine learning literature, termed Generative Adversarial Networks (GANs) that can be used to systematically generate artificial data that closely mimics real economic datasets, while limiting the degrees of freedom for the researcher and optionally satisfying privacy guarantees with respect to their training data. In addition if an applied researcher is concerned with the performance of a particular statistical method on a specific data set (beyond its theoretical properties in large samples), she may wish to assess the performance, e.g., the coverage rate of confidence intervals or the bias of the estimator, using simulated data which resembles her setting. Tol illustrate these methods we apply Wasserstein GANs (WGANs) to compare a number of different estimators for average treatment effects under unconfoundedness in three distinct settings (corresponding to three real data sets) and present a methodology for assessing the robustness of the results. In this example, we find that (i) there is not one estimator that outperforms the others in all three settings, so researchers should tailor their analytic approach to a given setting, and (ii) systematic simulation studies can be helpful for selecting among competing methods in this situation.

Susan Athey
Graduate School of Business
Stanford University
655 Knight Way
Stanford, CA 94305
and NBER
athey@stanford.edu

Guido W. Imbens
Graduate School of Business
Stanford University
655 Knight Way
Stanford, CA 94305
and NBER
Imbens@stanford.edu

Jonas Metzger
Department of Economics
Stanford University
metzgerj@stanford.edu

Evan M. Munro
Graduate School of Business
Stanford University
655 Knight Way
Stanford, CA 94305
munro@stanford.edu

1 Introduction

There has been rapid progress in the development of predictive statistical methods in recent years, particularly in the field of machine learning (see for recent reviews in economics Mullainathan and Spiess [2017], Athey and Imbens [2019]). Some of this progress has been aided by the availability of a large number of benchmark real-world data sets: combined with the criterion of out-of-sample predictive accuracy these data sets define a shared objective for the scientific community, which new developments can be easily evaluated against. A main objective of many econometric methods however is the estimation of causal effects, as opposed to making predictions. Since the causal effect are unobserved in real-world data sets, such data sets cannot directly serve as benchmarks to evaluate the performance of causal inference methods.

Therefore, when researchers develop new econometric methods, it is common practice to compare the performance of the new methods to those of existing methods in Monte Carlo studies where researchers know the data generating distribution and thus the true causal effect that generated the data. In such Monte Carlo studies, artificial data are frequently generated using very simple distributions with a high degree of smoothness and limited association between variables. The researcher potentially has many degrees of freedom in the design of a Monte Carlo exercise which can affect the results. As a consequence, the performance of new methods in those settings may not be indicative of the performance in realistic settings where many variables have mixed discrete-continuous distributions, sometimes with long tails and outliers, and correlation patterns are complex. For a recent discussion of these issues, see Advani et al. [2019], Knaus et al. [2018].

In a similar but distinct setting, an applied researcher may have to decide on which particular statistical method to use on a specific data set. To make this decision, evidence on the properties of various estimators in a particular finite-sample setting can be useful, even when attractive theoretical properties in large sample are known to hold for some methods. In this situation, the researcher may wish to assess the performance, *e.g.,* the coverage rate of confidence intervals or the bias of an estimator, using simulated data. Again, this requires generating artificial data in a setting in which it is particularly important to structure the simulations to the actual data set where the researcher may wish to implement the method.

In this paper we discuss how Generative Adversarial Nets (GANs, Goodfellow et al. [2014]), and in particular GANs minimizing the Wasserstein distance (WGANs, Arjovsky

et al. [2017]) can be used to systematically generate data that closely mimic real data sets. Given an actual data set these methods allow researchers to systematically assess the performance of various estimators in settings that are substantially more realistic than those often used in Monte Carlo studies. Moreover, by tying the data generating process to real data sets they can at least partly pre-empt concerns that researchers chose particular simulation designs to favor their proposed methods. Additionally, the resulting data generating distributions can be shown to satisfy certain privacy guarantees with respect to the data they were trained on under some modifications, see Xie et al. [2018]. This would allow the scientific community to benefit from otherwise inaccessible confidential data sources.

After a basic review of WGANs, we apply them to a classic data set in the program evaluation literature, originally put together by LaLonde [1986]. We use the specific sample subsequently recovered by Dehejia and Wahba [1999] that is available online on Dehejia's website. We refer to this as the Lalonde-Dehejia-Wahba (LDW) data set. The LDW data set has some special features which make it a challenging setting for estimators of average treatment effects under unconfoundedness. It is thus an attractive starting point for comparing different estimators. First, we demonstrate how WGANs can generate artificial data in this setting. Second, we discuss how similar the generated data are to the actual sample. Third, we assess the properties of a set of estimators. Finally, we present approaches to evaluate various robustness properties of these results, such as robustness to sampling variation, size of the original data and WGAN hyperparameters.

We use three specific samples created from the LDW data to create a range of settings. First, in what we call the LDW-E (experimental sample), we use the data from the original experiment. Second, LDW-CPS (observational sample) contains the experimental treated units and data on individuals from the CPS comparison sample. Third, in the LDW-PSID (observational sample) we use the experimental treated units and data from the PSID comparison sample. In our analysis we compare the performance of thirteen estimators for the average effect for the treated proposed in the literature. As a baseline case we use the difference in means by treatment status. In addition we consider a number of more sophisticated estimators, all based on the assumption of unconfounded treatment assignment. Some of these use flexible estimators of the conditional outcome means, of the propensity score, or of both. These estimators are based on generalized linear models, random forests and neural nets, as well as balancing methods.

2 Wasserstein Generative Adversarial Networks

Suppose we have a sample of N observations on d_X-component vectors, $X_1, \ldots, X_N$, drawn randomly from a distribution with cumulative distribution function $F_X(\cdot)$, density $f_X(\cdot)$ and domain $\mathbb{X}$. We are interested in drawing new samples that are similar to samples drawn from this distribution.

2.1 Conventional Alternatives

A conventional approach in econometrics is to estimate the distribution $f_X(\cdot)$ using kernel density estimation (Silverman [2018], Härdle [1990]). Given a bandwidth h and a kernel $K(\cdot)$, the standard kernel density estimator is

$$\hat{f}_X(x) = \frac{1}{Nh^{d_X}} \sum_{i=1}^{N} K\left(\frac{X_i - x}{h}\right).$$

Conventional kernels include the Gaussian kernel and the Epanechnikov kernel. Bandwidth choices have been proposed to minimize squared-error loss. Standard kernel density estimators perform poorly in high-dimensional settings, and when the true data distribution has bounded support. In finite samples, kernel density estimation are susceptible to oversmoothing.

An alternative approach to generate new samples that are similar to an existing sample is the bootstrap (Efron [1982], Efron and Tibshirani [1994]), which estimates the cumulative distribution function as

$$\hat{F}_X(x) = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_{X_i \leq x}.$$

The major reason this does not work for our purposes is that it cannot generate observations that differ from those seen in the sample and that the sampled data contains an unrealistic amount of identical data points. Specifically in the particular problem we study this would lead to the difficulty that in the population the propensity score, the probability of receiving the treatment conditional on the covariates, would always be

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a recently developed alternative approach to generating data that are similar to a particular data set (Goodfellow et al. [2014], Arjovsky and Bottou [2017]). GANs can be thought of as implicitly estimating the distribution,

although they do not directly produce estimates of the density or distribution function at a particular point. Instead, they directly optimize the parameters of a model serving as data generating process called the *generator*, which is trained in a type of mini-max game against an adversarial model called the *discriminator*. We introduce both pieces individually before we bring them together. Liang [2018] derive theoretical properties of the implied distributions obtained from this class of algorithms. GANs have recently become popular in the machine learning literature, but have not received much attention yet in the econometrics literature. An exception is Kaji et al. [2019] who use WGANs for estimation of structural models.

2.2.1 Generator

Departing from more conventional approaches to specifying DGPs, the generator is defined as a non-stochastic mapping $g(\cdot;\theta_g) : \mathbb{Z} \to \mathbb{X}$, where θ_g denote the parameters of the generator and $\mathbb{Z}$ is some latent space, which often is, but need not be, the same dimension as $\mathbb{X}$ in general. For any distribution $p_Z(\cdot)$ over $\mathbb{Z}$, this mapping implicitly defines a distribution $p_{\theta_g}(\cdot)$ over $\mathbb{X}$ via

$$\tilde{X} = g(Z;\theta_g),\ Z \sim p_Z(\cdot). \implies \tilde{X} \sim p_{\theta_g}(\cdot).$$

$p_Z(\cdot)$ is chosen by the researcher to be simple to draw from (e.g. multivariate uniform or normal) and kept fixed throughout training. Optimization is performed over the parameters θ_g, by minimizing a notion of distance between the empirical distributions of the samples from p_{θ_g} and the original data. Before describing the optimization, we describe the concept of a generator using a concrete example, where both $\mathbb{Z}$ and $\mathbb{X}$ are scalars. Let $p_Z(\cdot)$ be the density of a standard normal distribution:

$$Z \sim \mathcal{N}_{d_g}(0,1).$$

Our generator could be a simple shift

$$g(z;\theta_g) = z + \theta_g,$$

where θ_g is one-dimensional as well. With this special choice for the generator, we can express the implied distribution of $\tilde{X}$ in closed form:

$$\tilde{X} = g(Z;\theta_g),\ Z \sim \mathcal{N}_{d_g}(0,1) \implies \tilde{X} \sim \mathcal{N}_{d_X}(\theta_g, 1).$$

In practice, the generator is usually parametrized by a neural network. In this case, we would not have access to a closed form expression of $p_{\theta_g}(\cdot)$, although often it is still straightforward

to draw samples from it. Next, we explain how the GAN methodology minimizes a well defined notion of distance between the distribution of our model $p_{\theta_g}(\cdot)$ and that of the data $p_X(\cdot)$ without requiring a closed-form expression for either of the two.

2.2.2 Discriminator

First, we examine notions of distances between distributions. For two distributions $\mathbb{P}$ and $\mathbb{P}'$ which are absolutely continuous with respect to the same measure μ, the Kullback-Leibler divergence is given by

$$KL(\mathbb{P}, \mathbb{P}') = \int \ln\left(\frac{\mathbb{P}(X)}{\mathbb{P}'(X)}\right) \mathbb{P}(X) d\mu(X).$$

This distance is used with $\mathbb{P}'$ equal to the empirical distribution in maximum likelihood estimation where it has attractive efficiency properties. However, it has been argued that a distance notion that is symmetric in $\mathbb{P}$ and $\mathbb{P}'$ would be preferable in the context of data generation (Huszár [2015]): while maximum likelihood incentivizes model distributions under which the data from the empirical distribution is likely, data generation might be more interested in the reverse: model distributions which produce data that is likely under the empirical distribution. Intuitively, this can be related to the perceived tendency of KL-minimizers to *oversmooth* relative to the true distribution. An objective that balances these aspects would be the Jensen-Shannon divergence

$$JS(\mathbb{P}, \mathbb{P}') = KL\left(\mathbb{P} \middle| \frac{\mathbb{P} + \mathbb{P}'}{2}\right) + KL\left(\mathbb{P}' \middle| \frac{\mathbb{P} + \mathbb{P}'}{2}\right).$$

Goodfellow et al. [2014] show that the JS divergence can be equivalently written as the solution to a particular optimization problem:

$$JS(\mathbb{P}, \mathbb{P}') = \ln 2 + \sup_{d:\mathbb{X}\to(0,1)} \left\{\frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}} \ln(d(x)) + \frac{1}{2}\mathbb{E}_{x\sim\mathbb{P}'} \ln(1 - d(x))\right\}.$$

The authors call the function $d(\cdot)$ the *discriminator*. It has a simple interpretation: imagine a data generating process that with equal probability samples x from either $\mathbb{P}$ or $\mathbb{P}'$. Then the above optimization problem corresponds to the maximum likelihood objective of a model which tries to classify which of the two distributions x was sampled from, where the optimal classifier is

$$d^*(x) = P(x \sim \mathbb{P}|x) = \frac{\mathbb{P}(x)}{\mathbb{P}(x) + \mathbb{P}'(x)}.$$

Let us examine how this would apply to the simple one-dimensional example from the previous subsection. Assume the original data was generated from a one-dimensional Gaussian with mean μ and unit variance, *i.e.*

$$X \sim \mathcal{N}(\mu, 1).$$

Then, we can plug in the Gaussian densities into the expression above to obtain the optimal discriminator d^*. For any given value of μ and θ_g, we can cancel out constant factors of the Gaussian densities to obtain, for $\sigma(x) = \frac{\exp(x)}{1+\exp(x)}$:

$$d^*(x) = \sigma(\mu^2 - \theta_g^2 + 2(\mu - \theta_g)x) = \sigma(\theta_{0d}^* + \theta_{1d}^* x),$$

for $\theta_{0d}^* = \mu^2 - \theta_g^2$ and $\theta_{1d}^* = 2(\mu - \theta_g)$. A key insight is that we do not require an analytical expression for either of the two densities to obtain the optimal discriminator. We can simply parametrize a model for the discriminator $d(x; \theta_d) = \sigma(\theta_{0d} + \theta_{1d} x)$ and optimize the likelihood of correctly classifying random samples from the two distributions. The maximum likelihood estimator for θ_d will converge to the optimal θ_d^*. This yields the JS divergence between the current generator and original data distribution and allows us to obtain gradients with which we can optimize the generator as described in the next section. Even if the true and generator distributions are very complex, we can *approximate* the optimal discriminator by maximizing the empirical analogue of the maximum likelihood objective with any sufficiently flexible function approximator $d(\cdot; \theta_d)$ taking values in $(0, 1)$. This yields the original GAN formulation.

2.3 Original GAN

Let $X_1, \ldots, X_{N_R}$ denote the original data as before and let $Z_1, \ldots, Z_{N_F}$ be a large number of samples from $p_Z(\cdot)$. Goodfellow et al. [2014] propose to jointly optimize for the discriminator and generator via the saddle-point objective

$$\min_{\theta_g} \max_{\theta_d} L(\theta_d, \theta_g),$$

where

$$L(\theta_d, \theta_g) = \frac{1}{N_R} \sum_{i=1}^{N_R} \ln d(X_i; \theta_d) + \frac{1}{N_F} \sum_{i=1}^{N_F} \ln \Big[1 - d\Big(g(Z_i; \theta_g); \theta_d\Big)\Big].$$

Both the generator and discriminator are parametric models, though typically very flexible ones, *e.g.,* neural networks. The joint optimization is carried out by switching back and

forth between updating θ_d and θ_g in the respective directions implied by the gradient of the objective $L(\theta_d, \theta_g)$. This procedure can be interpreted as a two player mini-max game with alternating better-response dynamics. Using the arguments from the previous subsection, the authors discuss assumptions under which this process converges to the saddle-point in which the discriminator yields the JS divergence, which the generator minimizes by mimicking the original data $p_X(\cdot)$. Further implementation details are described in Subsection 2.5. Let us examine how this would play out in our simplified one-dimensional example. Since both the original and discriminator distributions are Gaussian with constant variance, we are justified to restrict the search space for the discriminator to that of linear logistic regression functions as argued before. After aninitialization $(\theta_g^{(0)}, \theta_d^{(0)})$, we can optimize the saddle-point objective by iterating between the following two steps. Given values (θ_d^k, θ_g^k) after k steps of the algorithm we update the two parameters:

1. Update the discriminator parameter θ_d as

$$\theta_d^{k+1} = \arg\max_{\theta_d} L(\theta_d, \theta_g).$$

2. Update the generator parameter θ_g by taking a small step (small learning rate α) along the derivative:

$$\theta_g^{k+1} = \theta_g^k - \alpha \frac{\partial}{\partial \theta_g} L(\theta_d^{k+1}, \theta_g).$$

After optimizing the discriminator at each step, we get an estimate of the JS divergence and its gradient at the *current* value of θ_g. We thus need to re-optimize the discriminator after every gradient update of θ_g. Particularly when the discriminator is a neural network, a practical implementation would simply update θ_d for a few gradient steps only instead of solving its optimization until convergence. In our particular example, given a sufficiently large number of draws from $P_Z(\cdot)$, the process will converge to the JS minimizing value $\theta_g = \overline{X}$ implying a discriminator with $\theta_d = (\ln(N_R/(N_R + N_F)), 0)$ which cannot do better than guessing a constant probability of $N_R/(N_R+N_F)$ of the data being real. Note that in the saddlepoint optimization we may completely optimize the parameters of the discriminator, but we should only take small steps for the generator.

2.4 Wasserstein GANs

In practice, optimization of the original Goodfellow et al. [2014] GAN objective has proven to be rather difficult. Difficulties can arise when the discriminator becomes too good early

on, becoming "flat" around the samples from the generator and thus failing to provide useful gradient information to the generator. This is related to the fact that the JS divergence is infinite between two distributions with non-identical support. See Gulrajani et al. [2017], Arjovsky and Bottou [2017] for details. An attractive alternative to the Jensen-Shannon divergence is the Earth-Mover or Wasserstein distance (Arjovsky et al. [2017]):

$$W(\mathbb{P}, \mathbb{P}') = \inf_{\gamma \in \Pi(\mathbb{P}, \mathbb{P}')} \mathbb{E}_{(X,Y)\sim\gamma} \left[\|X - Y\| \right],$$

where $\Pi(\mathbb{P}, \mathbb{P}')$ is the set of joint distributions that have marginals equal to $\mathbb{P}$ and $\mathbb{P}'$. The term Earth-Mover distance comes from the interpretation that $W(\mathbb{P}, \mathbb{P}')$ is the amount of probability mass that needs to be transported to move from the distribution $\mathbb{P}$ to the distribution $\mathbb{P}'$. It is symmetric and remains well defined irrespective of the amount of overlap between the support of the distributions by introducing a natural notion of similarity between different supports. Arjovsky et al. [2017] exploit the fact that the Wasserstein distance, like the JS divergence, admits a dual representation

$$W(\mathbb{P}, \mathbb{P}') = \sup_{\|f\|_L \leq 1} \Big\{ \mathbb{E}_{X\sim\mathbb{P}} \left[f(X) \right] - \mathbb{E}_{X\sim\mathbb{P}'} \left[f(X) \right] \Big\},$$

where we take the supremum of the functions $f(\cdot)$ over all Lipschitz functions with Lipschitz constant equal to 1. The function $f(\cdot)$ is known as the *critic* and its optimized value implies an upper bound on how much any Lipschitz-continuous moment can differ between the two distributions. Suppose we parametrize the critic as $f(x; \theta_c)$. Ignoring the Lipschitz constraint, the empirical analogue of the optimization problem becomes

$$\min_{\theta_g} \max_{\theta_c} \left\{ \frac{1}{N_R} \sum_{i=1}^{N_R} f(X_i; \theta_c) - \frac{1}{N_F} \sum_{i=1}^{N_F} f(g(Z_i; \theta_g); \theta_c) \right\}. \tag{2.1}$$

Given the generator, we choose the parameters of the critic to maximize the difference between the average of $f(X_i; \theta_c)$ over the real data and the average over the generated data. We then choose the parameter of the generator θ_g, to minimize this maximum difference. For this objective to be well-behaved, it is important to restrict the search to parameters that ensure that the critic is Lipschitz with constant 1. The original WGAN formulation considered parameter clipping to ensure this constraint, which caused computational problems. Gulrajani et al. [2017] showed that these can be avoided by instead adding a penalty term to the objective function for the critic. This term directly penalizes the norm of the derivative of the critic $f(\cdot)$ with respect to its input along the lines connecting original and generated

data points, which ensures the critic sufficiently satisfies the constraint. Specifically, it has the form

$$\lambda \left\{ \frac{1}{m} \sum_{i=1}^{m} \left[\max \left(0, \left\| \nabla_{\hat{x}} f \left(\hat{X}_i; \theta_c \right) \right\|_2 - 1 \right) \right]^2 \right\},$$

where the $\hat{X}_i = \epsilon_i X_i + (1 - \epsilon_i)\tilde{X}_i$ are random convex combinations of the real and generated observations, with the ε_i re-drawn at each step. Note that here we use batches of the real and generated data of the same size m.

2.5 The Algorithm

Instead of using all the data in each step of the algorithm, we repeatedly use random batches of the real data with batch size m, denoted by $X_1, \ldots, X_m$, and each iteration generate the same number m of new fake observations from the input distribution, denoted by $Z_1, \ldots, Z_m$. The general algorithm is described in Algorithm 1. For the optimization we use a modification of the SGD (Stochastic Gradient Descent) algorithm (*e.g.,* Bottou [2010]), the Adam (Adaptive moment estimation, Kingma and Ba [2014]) algorithm. The Adam algorithm combines the estimate of the (stochastic) gradient with previous estimates of the gradient, and scales this using an estimate of the second moment of the unit-level gradients, the latter part being somewhat akin to the Berndt-Hall-Hall-Hausman algorithm proposed in Berndt et al. [1974]. Details on the specific version of the Adam algorithm we use, which adapts the learning rate (Baydin et al. [2017]) to reduce the time for hyperparameter search, are provided in the appendix. Our specific implementation uses dropout (Warde-Farley et al. [2013], Wager et al. [2013]) for regularization purposes. Without regularization, the generator may get close to the empirical distribution function.

2.6 Conditional WGANs

The algorithm discussed in Section 2.5 learns to generate draws from an unconditional distribution. In many cases we want to generate data from a conditional distribution. For example, for the causal settings that motivate this paper, we may wish to keep fixed the number of treated and control units. That would be simple to implement by just running two marginal WGANs. More importantly, we wish to generate potential treated and control outcomes given a common set of pre-treatment variables. For that reason it is important to generate data from a conditional distribution (Mirza and Osindero [2014], Odena et al. [2017], Liu et al. [2018], Kocaoglu et al. [2017]).

Algorithm 1 WGAN

▷ Tuning parameters:

m, batch size

$n_{critic} = 15$, number of critic iterations per iteration of the generator

$lr_0 = 0.0001$, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, parameters for Adam algorithm with hypergradient descent

$\lambda = 5$, penalty parameter for derivative of critic

▷ Starting Values:

$\theta_c = 0$ (critic), $\theta_g = 0$ (generator)

▷ Noise Distribution:

$p_Z(\cdot)$ is mean zero Gaussian with identity covariance matrix, dimension equal to that of x

while θ_g has not converged **do**

  ▷ Run n_{critic} training steps for the critic.

  for $t = 0, ..., n_{critic}$ **do**

    Sample $\{X_i\}_{i=1}^m \sim \mathcal{D}$ (a batch of size m from the real data, without replacement)

    Sample $\{Z_i\}_{i=1}^m \sim p_Z(z)$ noise.

    ▷ Generate m fake observations from the noise observations.

    $\tilde{X}_i \leftarrow g(Z_i; \theta_g)$ for $i = 1, \ldots, m$

    ▷ Compute penalty term $Q(\theta_c)$.

    Generate ϵ_i, $i = 1, \ldots, m$ from uniform distribution on $[0, 1]$

    Calculate $\hat{X}_i = \epsilon_i X_i + (1 - \epsilon_i)\tilde{X}_i$ convex combinations of real and fake observations

    $Q(\theta_c) \leftarrow \frac{1}{m} \sum_{i=1}^m \left[\max\left(0, \left\| \nabla_{\hat{x}} f\left(\hat{X}_i; \theta_c\right) \right\|_2 - 1 \right) \right]^2$

    ▷ Compute gradient with respect to the critic parameter θ_c.

    $\delta_{\theta_c} \leftarrow \nabla_{\theta_c} \left[\frac{1}{m} \sum_{i=1}^m f\left(X_i; \theta_c\right) - \frac{1}{m} \sum_{i=1}^m f\left(\tilde{X}_i; \theta_c\right) + \lambda Q(\theta_c) \right]$

    $\theta_c \leftarrow \text{Adam}(-\delta_{\theta_c}, \theta_c, \alpha, \beta_1, \beta_2)$ (update critic parameter using Adam algorithm)

    end for

  ▷ Run a single generator training step.

  Sample $\{Z_i\}_{i=1}^m \sim p_Z(z)$ noise.

  ▷ Compute gradients with respect to the generator parameters.

  $\delta_{\theta_g} \leftarrow \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m f\left(g(Z_i; \theta_g); \theta_c\right)$

  $\theta_g \leftarrow \text{Adam}(\delta_{\theta_g}, \theta_g, \alpha, \beta_1, \beta_2)$ (update generator parameter using Adam algorithm)

  end while

Suppose we have a sample of real data (X_i, V_i), $i = 1, \ldots, N_R$. We wish to train a generator to sample from the conditional distribution of $X_i|V_i$. The conditioning variables V_i are often referred to as *labels* in this literature. Let $g(x|v;\theta_g)$ be the generator, and let $f(x|v;\theta_c)$ be the critic. As before, both will be neural networks, although this is not essential. The specific algorithm is described in Algorithm 2. [1]

3 Simulating the Lalonde-Dehejia-Wahba Data

In this section we discuss using WGANs to simulate data for the Lalonde-Dehejia-Wahba (LDW) data.

3.1 Simulation Studies for Average Treatment Effects

In the setting of interest we have data on an outcome Y_i, a set of pretreatment variables X_i and a binary treatment W_i. We postulate that there exists for each unit in the population two potential outcomes $Y_i(0)$ and $Y_i(1)$, with the observed outcome equal to corresponding to the potential outcome for the treatment received, $Y_i = Y_i(W_i)$. We are interested in the average treatment effect for the treated,

$$\tau = \mathbb{E}[Y_i(1) - Y_i(0)|W_i = 1],$$

assuming unconfoundedness (Rosenbaum and Rubin [1983], Imbens and Rubin [2015]):

$$W_i \perp \Big(Y_i(0), Y_i(1)\Big) \Big| X_i,$$

and overlap

$$0 < \text{pr}(W_i = 1|X_i = x) < 1,$$

for all x in the support of the pre-treatment variables. Let $\mu(w, x) = \mathbb{E}[Y_i|W_i = w, X_i = x]$ (which by unconfoundedness is equal to $\mathbb{E}[Y_i(w)|X_i = x]$) be the conditional outcome mean, and let $e(x) = \text{pr}(W_i = 1|X_i = x)$ be the propensity score. There a large literature developing methods for estimating average and conditional average treatment effects in this literature (see Rubin [2006], Imbens [2004], Abadie and Cattaneo [2018] for general discussions).

[1] In a related paper, Kocaoglu et al. [2017] propose an architecture that preserves dependencies represented in a directed acyclic graph (DAG).

Algorithm 2 CWGAN

▷ Tuning parameters:

m, batch size

$n_{critic} = 15$, number of critic iterations per iteration of the generator

$lr_0 = 0.0001$, $\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, parameters for Adam algorithm with hypergradient descent

$\lambda = 5$, penalty parameter for derivative of critic

▷ Starting Values:

$\theta_c = 0$ (critic), $\theta_g = 0$ (generator)

▷ Noise Distribution:

$p_Z(z)$ is mean zero Gaussian with identity covariance matrix, dimension equal to that of x

while θ has not converged **do**

▷ Run n_{critic} training steps for the critic.

for $t = 0, ..., n_{critic}$ **do**

Sample $\{(X_i, V_i)\}_{i=1}^m \sim \mathcal{D}$ a batch from the real data and labels.

Sample $\{Z_i\}_{i=1}^m \sim p_Z(z)$ noise.

▷ Generate m fake observations $\tilde{X}_i$ corresponding to the m real labels V_i.

$\tilde{X}_i \leftarrow g(Z_i|V_i;\theta_g)$ for each i

▷ Compute penalty term $Q(\theta_c)$.

Generate ϵ_i, $i = 1, \ldots, m$ from uniform distribution on $[0, 1]$

Calculate $\hat{X}_i = \epsilon_i X_i + (1-\epsilon_i)\tilde{X}_i$ convex combinations of real and fake observations

$Q(\theta_c) \leftarrow \frac{1}{m}\sum_{i=1}^m \left[\max\left(0, \left\|\nabla_{\hat{x}} f\left(\hat{X}_i|V_i;\theta_c\right)\right\|_2 - 1\right)\right]^2$

▷ Compute gradient with respect to the critic parameter θ_c.

$\delta_{\theta_c} \leftarrow \nabla_{\theta_c}\left[\frac{1}{m}\sum_{i=1}^m f\left(X_i|V_i;\theta_c\right) - \frac{1}{m}\sum_{i=1}^m f\left(\tilde{X}_i|V_i;\theta_c\right) + \lambda Q(\theta_c)\right]$

$\theta_c \leftarrow \text{Adam}(-\delta_{\theta_c}, \theta_c, \alpha, \beta_1, \beta_2)$ (update critic parameter using Adam algorithm)

end for

▷ Run a single generator training step.

Sample $\{V_i\}_{i=1}^m \sim \mathcal{D}$ a batch of size m from the real labels.

Sample $\{Z_i\}_{i=1}^m \sim p_Z(z)$ noise.

▷ Compute gradients with respect to the generator parameters.

$\delta_{\theta_g} \leftarrow \nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^m f\left(g(Z_i|V_i;\theta_g)|V_i;\theta_c\right)$

$\theta_g \leftarrow \text{Adam}(\delta_{\theta_g}, \theta_g, \alpha, \beta_1, \beta_2)$ (update generator parameter)

Given a sample (X_i, W_i, Y_i) we cannot directly establish the true value of the average treatment effect, so standard Machine Learning comparisons of estimators based on cross-validation methods are not feasible. In this setting, researchers have often conducted simulation studies to assess the properties of proposed methods (Abadie and Imbens [2011], Belloni et al. [2014], Athey et al. [2018], Huber et al. [2013], Lechner and Wunsch [2013], Lechner and Strittmatter [2019], Knaus et al. [2018], Wendling et al. [2018]).

Most closely related in the spirit of creating simulation designs that are close to real data are Abadie and Imbens [2011], Schuler et al. [2017], Knaus et al. [2018]. Using the LDW sample Abadie and Imbens [2011] estimate a model for the conditional means and the propensity score allowing for linear terms and second order terms. To account for the mass points at zero, they model separately the probability of the outcome being equal to zero and outcome conditional on being positive. Schuler et al. [2017] also start with a real data set. They postulate a value for the conditional average treatment effect $\tau(x) = \mathbb{E}[Y_i(1) - Y_i(0)|X_i = x] = \mu(1, x) - \mu(0, x)$. They then use the empirical distribution of (W_i, X_i) as the true distribution. They estimate the conditional means $\mu(w, x)$ using flexible models, imposing the constraint implied by the choice of conditional average treatment effect $\tau(x)$. Given these estimates they estimate the residual distribution as the empirical distribution of $Y_i - \hat{\mu}(W_i, X_i)$. Then they impute outcomes for new samples using the estimated regression functions and random draws from the empirical residual distribution. Note that this procedure imposes homoskedasticity. Note also that the Schuler et al. [2017] choice for the joint distribution of (W_i, X_i) can create violations of the overlap requirement if the pre-treatment variables X_i are continuous. Because they specify the conditional average treatment effect that does not create problems for estimating the ground truth. Knaus et al. [2018] develop what they call empirical Monte Carlo methods where they use the empirical distribution of the covariates and the control outcome, combined with postulated individual level treatment effects and a flexibly estimated propensity score to generate artificial data.

3.2 The LDW Data

The data set we use in this paper was originally constructed by LaLonde [1986], and later recovered by Dehejia and Wahba [1999]. The version we use is available on Dehejia's website. This data set has been widely used in the program evaluation literature to compare different methods for estimating average treatment effects (e.g., Dehejia and Wahba [2002], Heckman

and Hotz [1989], Abadie and Imbens [2011]). We use three versions of the data. The first, which we refer to as the experimental sample, LDW-E, contains the observations from the actual experiment. This sample contains $N^{\rm exp} = 445$ observations, with $N_0^{\rm exp} = 260$ control observations and $N_1^{\rm exp} = N^{\rm exp} - N_0^{\rm exp} = 185$ treated observations. For each individual in this sample we observe a set of eight pre-treatment variables, denoted by X_i. These include two earnings measures, two indicators for ethnicity, marital status, and two education measures, and age. $\mathbf{X}_0^{\rm exp}$ denotes the $N_0^{\rm exp} \times 8$ matrix with each row corresponding to the pre-treatment variables for one of these units, and $\mathbf{X}_1^{\rm exp}$ denoting the $N_1^{\rm exp} \times 8$ for the treated units in this sample. Let $\mathbf{X}^{\rm exp} = \left(\mathbf{X}^{\rm exp_0 \top} \mathbf{X}^{\rm exp_1 \top}\right)^\top$ denote the $N^{\rm exp} \times 8$ matrix with all the covariates. Simiarly, let $\mathbf{Y}_0^{\rm exp}$ denote the $N_0^{\rm exp}$ vector of outcomes for the control units in this sample, and $\mathbf{Y}_1^{\rm exp}$ denote the $N_1^{\rm exp}$ vector of outcomes for the treated units, and let $\mathbf{W}_0^{\rm exp}$ denote the $N_0^{\rm exp}$ vector of treatment indicators for the control units in this sample (all zeros), and $\mathbf{W}_1^{\rm exp}$ denote the $N_1^{\rm exp}$ vector of outcomes for the treated units (all ones). The outcome is a measure of earnings in 1978.

The second sample is the CPS sample, LDW-CPS. It combines the treated observations from the experimental sample with $N_0^{\rm cps} = 15,992$ control observations drawn from the Current Population Survey, for a total of $N^{\rm cps} = N_1^{\rm exp} + N_0^{\rm cps} = 16,177$ observations. The third sample is the PSID sample, LDW-PSID. It combines the treated observations from the experimental sample with $N_0^{\rm psid} = 2,490$ control observations drawn from the Panel Survey of Income Dynamics, for a total of $N^{\rm psid} = N_1^{\rm exp} + N_0^{\rm psid} = 2,675$ observations. Table 1 presents summary statistics for the eight pretreatment variables in these samples, the treatment indicator and the outcome.

3.3 A Conditional WGAN for the LDW Data

Consider the experimental data set LDW-E. The goal is to create samples of $N^{\rm exp}$ observations, containing $N_0^{\rm exp} = 260$ control units and $N_1^{\rm exp} = 185$ treated units, where the samples are similar to the real sample. We proceed as follows. First, we run a conditional WGAN on the sample $\mathbf{X}^{\rm exp}$, conditional on $\mathbf{W}^{\rm exp}$. Let the parameters of the generator of the WGAN be $\theta_{g,X}^{\rm exp}$. During training of the models, each batch of training data contains treated and control units with equal probability, to avoid estimation issues when the fraction of treated units is close to zero (for example, it is equal to 0.011 in the CPS dataset).

In each case, for the generator we use a neural net with the following architecture. There

Table 1: Summary Statistics for Lalonde-Dehejia-Wahba Data

	Experimental trainees (185)		Experimental controls (260)		CPS controls (15,992)		PSID controls (2,490)	
	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.
black	0.84	(0.36)	0.83	(0.38)	0.07	(0.26)	0.25	(0.43)
hispanic	0.06	(0.24)	0.11	(0.31)	0.07	(0.26)	0.03	(0.18)
age	25.82	(7.16)	25.05	(7.06)	33.23	(11.05)	34.85	(10.44)
married	0.19	(0.39)	0.15	(0.36)	0.71	(0.45)	0.87	(0.34)
nodegree	0.71	(0.46)	0.83	(0.37)	0.30	(0.46)	0.31	(0.46)
education	10.35	(2.01)	10.09	(1.61)	12.03	(2.87)	12.12	(3.08)
earn '74	2.10	(4.89)	2.11	(5.69)	14.02	(9.57)	19.43	(13.41)
earn '75	1.53	(3.22)	1.27	(3.10)	13.65	(9.27)	19.06	(13.60)
treatment	1	-	0	-	0	-	0	-
earn '78	6.35	(7.87)	4.55	(5.48)	14.85	(9.65)	21.55	(15.56)

are three hidden layers in the neural net, with the number of inputs and outputs equal to $(d_X + M, 128)$, $(128, 128)$ and $(128, 128)$ respectively. Here d_X is the dimension of the vectors whose distribution we are modeling and M is the dimension of the conditioning variables. For generating the covariates conditional on the treatment, this is $d_X = 8$, and $M = 1$, and for generating the outcome variable conditional on the treatment this is $d_X = 1$, and $M = 8$. We use the rectified linear transformation, $a(z) = |a|\mathbf{1}_{z>0}$ in the hidden layers. For the final layer we have 128 inputs and d_X outputs. Here we use for binary variables a sigmoid transformation, for censored variables a rectified linear transformation, and for continuous variables the identify function. We use 10% dropout in the generator.

For the critic we use the same architecture with three layers, with the number of inputs and outputs equal to $(d_X + M, 128)$, $(128, 128)$ and $(128, 128)$ respectively. For the final layer we have 128 inputs, and 1 linear output.

We did not adapt the architectures to the individual settings, so these hyperparameters should not be thought of as optimal. In spite of this, they yield a well-performing WGAN. This is to emphasize that the exact architectural choices do not matter in settings like ours, so long as the overall size of the network is large enough to capture the complexity of the

data and the amount of regularization (*i.e.,* dropout probability) is high enough to avoid over-fitting.

Given the parameters for the generators, $\theta^{\rm exp}_{g,X|W}$, $\theta^{\rm exp}_{g,Y(W)|X,W}$, we first create a very large sample, with $N = 10^6$ units. We use this sample as our population for the simulations. To create the large sample, first we draw separately the covariates for the treated and control units using the generator with parameter $\theta^{\rm exp}_{g,X|W}$. In this step, we create the sample keeping the fraction of treated units equal to that in the sample. Next we draw independently $Y(0)$ and $Y(1)$ for each observation in this large sample, using the X and W as the conditioning variables, using the generators with parameters $\theta^{\rm exp}_{g,Y(W)|X,W}$. Unlike in any real dataset, we observe both $Y(0)$ and $Y(1)$ for each unit, simplifying the task of estimating the ground truth in the simulated data. We use this large sample to calculate the approximate true average effect for the treated as the average difference between the two potential outcomes for the treated units:

$$\tau = \frac{1}{N_1} \sum_{i:W_i=1} \Big(Y_i(1) - Y_i(0) \Big).$$

For this fixed population we report in Table 2 the means and standard deviations for the same ten variables as in Table 1. The means and standard deviations are fairly similar. However, the fact that the first two moments of the generated data closely match those of the actual data is only limited comfort. There are simple ways in which to generate data for which the first two moments of each of the variables match exactly those of the actual data, such as the standard bootstrap or a multivariate normal distribution. However, our generator allows us to generate new samples that contain observations not seen in the actual data, and with no duplicate observations. For some of the estimators, notably the nearest neighbor matching estimators, this can make a substantial difference.

In Figures 1-5 we present some graphical evidence on the comparison of the actual data and the generate data for the CPS control sample. In general the generated data and the actual data are quite similar. This is true for the first two moments, the marginal distributions, the correlations, as well as the conditional distributions. In particular it is impressive to see in Figure 5 that the conditional distribution of earnings for two groups for the actual data (those with 1974 earnings positive or zero), which have substantially different shapes, is in both cases still well matched by the artificial data.

Given the fact that fit models to the conditional expectation of the outcome variable earnings in 1978 conditional on the treatment and the other variables, it is of interest to

Table 2: Summary Statistics for WGAN-Generated Data Based on LDW Data

	Experimental trainees		Experimental controls		CPS controls		PSID controls	
	mean	s.d.	mean	s.d.	mean	s.d.	mean	s.d.
black	0.86	(0.35)	0.80	(0.40)	0.10	(0.30)	0.27	(0.44)
hispanic	0.05	(0.23)	0.10	(0.30)	0.05	(0.23)	0.03	(0.18)
age	25.16	(6.15)	25.00	(7.27)	33.14	(10.96)	34.42	(10.67)
married	0.16	(0.37)	0.10	(0.31)	0.71	(0.46)	0.87	(0.34)
nodegree	0.72	(0.45)	0.84	(0.37)	0.31	(0.46)	0.31	(0.46)
education	10.43	(1.74)	10.1	(1.63)	11.79	(2.81)	11.99	(3.01)
earn '74	2.21	(5.29)	1.69	(5.08)	13.46	(9.60)	20.48	(12.51)
earn '75	1.66	(2.90)	1.08	(2.66)	12.84	(9.15)	19.67	(12.69)
earn '78	6.66	(6.69)	4.19	(4.91)	14.13	(9.55)	21.06	(13.29)

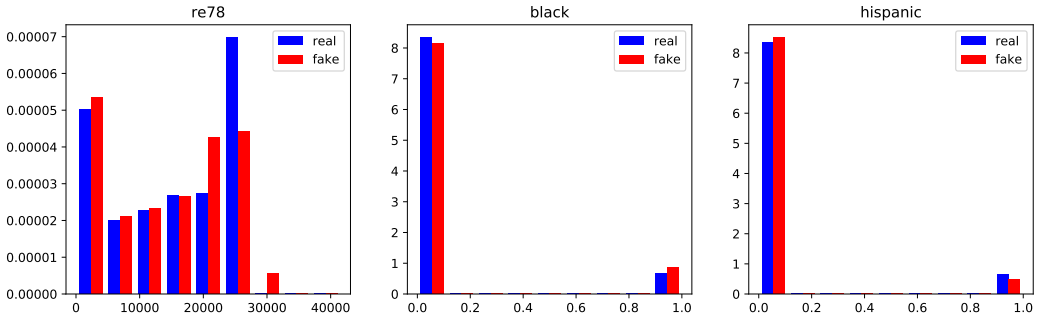

Figure 1: Histograms for CPS Data, Earnings 1978, Black, Hispanic

assess how well this conditional expectation is approximated by a linear function. To do so we fit a linear model, a random forest and a neural net to this regression, and compare the goodness of fit out of sample. We do so both with the actual data and the generated data.

We find that for the experimental and CPS samples the model fit is similar between the real and generated data, for all three models. That is not true for the PSID sample. There the fit for the real data is substantially worse than for the generated data. This suggests the WGAN does not match the real data as well in this case, and the rankings based on the WGAN should not be taken as seriously for the PSID data.

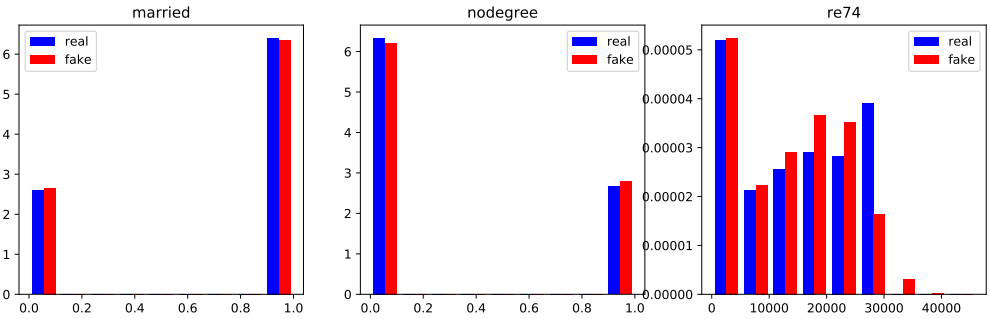

Figure 2: Histograms for CPS Data, Married, No Degree, Earnings 1974

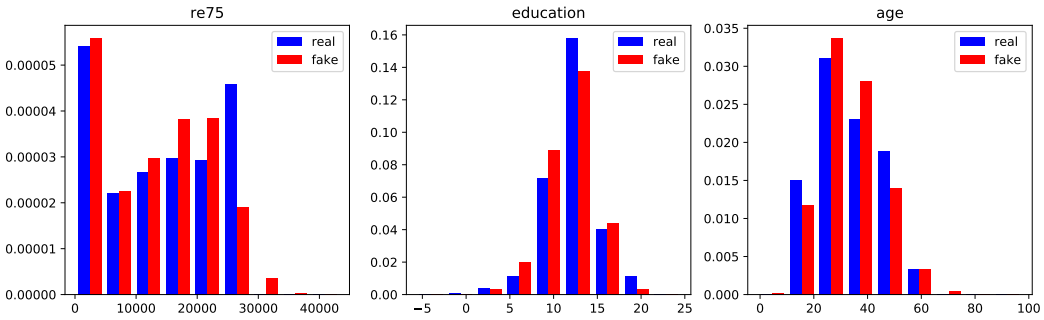

Figure 3: Histograms for CPS Data, Earnings 1975, Education, Age

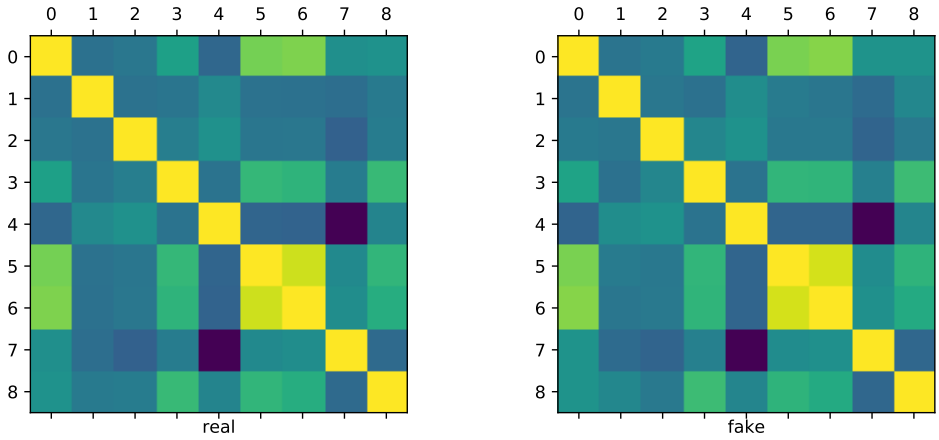

Figure 4: Correlations for CPS Data

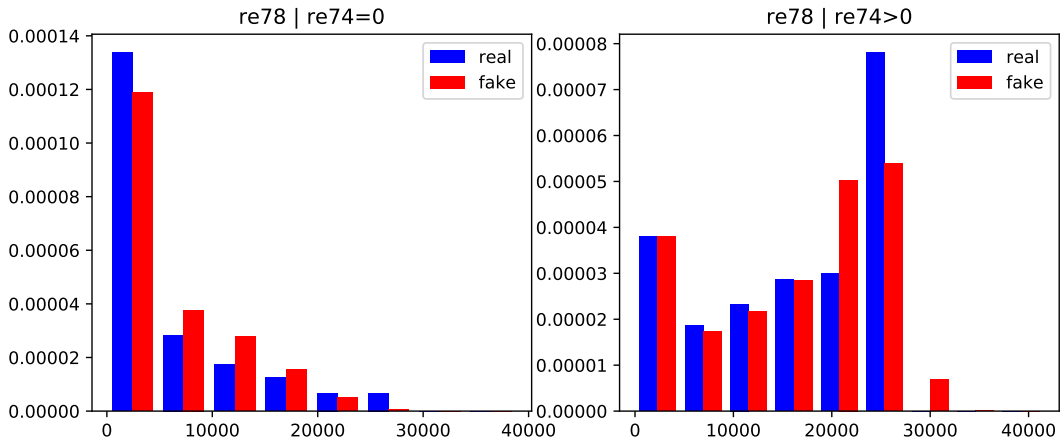

Figure 5: Conditional Histograms for CPS Data, Earnings 1978 Given Earnings 1974 Positive or Zero

Next we repeatedly construct samples of size N^{exp}, with N_0^{exp} control units and N_1^{exp} treated units. From our large population of size $N = 10^6$, we randomly draw N_0^{exp} observations with $W_i = 0$, and record their X_i, their $W_i = 0$, and their outcome $Y_i = Y_i(0)$. Next we draw randomly N_1^{exp} observations with $W_i = 1$ and record their X_i, their $W_i = 1$, and their outcome $Y_i = Y_i(1)$. This gives us our sample with N^{exp} units, with N_0^{exp} control units and N_1^{exp} treated units. If we want another sample we use the next N_0^{exp} control units and the next N_1^{exp} treated observations.

4 Comparing Estimators for Average Treatment Effects

In this section we implement the WGANs to generate data sets to compare different estimators for the average treatment effects. We do this in three settings, first with the experimental LDW-E data, second with the LDW-CPS comparison group, and third with the LDW-PSID comparison group.

4.1 Estimators

We compare thirteen estimators for the average effect for the treated. Nine of them fit into a set where we compare three methods for estimating the two nuisance functions, the propensity score $e(x)$ and the conditional outcome mean $\mu(0, x)$ (linear and logit models,

Table 3: Out-of-Sample Goodness of Fit (R^2) on Real and Generated Data

		Experimental Controls	CPS Controls	PSID Controls
Real Data	Linear Model	-0.17	0.47	0.57
Real Data	Random Forest	-0.14	0.48	0.58
Real Data	Neural Net	-0.26	0.48	0.56
Generated Data	Linear Model	-0.03	0.49	0.75
Generated Data	Random Forest	0.00	0.51	0.75
Generated Data	Neural Net	-0.15	0.51	0.74

random forests, and neural nets), with three ways of combining these estimates of the nuisance functions (difference in estimated conditional means, propensity score weighting, and double robust methods), and four are stand-alone estimators. All estimators that involve estimating the propensity score use trimming on the estimated propensity score, dropping observations with an estimated propensity score larger than 0.95. See Crump et al. [2009] for discussions of the importance of trimming in general.

For estimating the two nuisance functions $e(x)$ and $\mu(0,x)$ we consider three methods:

1. A logit model for the propensity score and a linear model for the conditional outcome mean, given the set of eight pre-treatment variables. Denote the estimator for the conditional mean by $\hat{\mu}^{\text{lm}}(0,x)$, and the estimator for the propensity score by $\hat{e}^{\text{lm}}(x)$.

2. Random Forests for the propensity score and the conditional outcome mean. Denote the estimator for the conditional mean by $\hat{\mu}^{\text{rf}}(0,x)$, and the estimator for the propensity score by $\hat{e}^{\text{rf}}(x)$.

3. Neural Nets for the propensity score and the conditional outcome mean.Denote the estimator for the conditional mean by $\hat{\mu}^{\text{nn}}(0,x)$, and the estimator for the propensity score by $\hat{e}^{\text{nn}}(x)$. See Farrell et al. [2018] for theoretical analysis of neural nets.

We also consider three methods for incorporating the estimated nuisance functions into an estimator for the ATE:

1. Use the estimated conditional outcome mean by averaging the difference between the realized outcome and the esitmated control outcome, averaging this over the treated observations:

$$\hat{\tau}^{\rm cm} = \frac{1}{N_1} \sum_{i:W_i=1} \Big(Y_i - \hat{\mu}(0, X_i) \Big).$$

2. Use the estimated propensity score to weight the control observations:

$$\hat{\tau}^{\rm ht} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - (1-W_i) Y_i \frac{\hat{e}(X_i)}{1-\hat{e}(X_i)} \Big/ \frac{1}{N_1} \sum_{j=1}^{N} (1-W_j) \frac{\hat{e}(X_j)}{1-\hat{e}(X_j)} \right).$$

3. Use both the estimated conditional mean and the estimated propensity score in a double robust approach:

$$\hat{\tau}^{\rm dr} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - \hat{\mu}(0, X_i) - (1-W_i)(Y_i - \hat{\mu}(0, X_i) \frac{\hat{e}(X_i)}{1-\hat{e}(X_i)} \right).$$

   Note that for the neural net and the random forest implementation, we use sample splitting as in Chernozhukov et al. [2018].

4.2 Estimates for LDW Data

First we use all the estimators on the actual LDW data. The results are reported in Table 4.

4.3 Simulation Results for the Experimental Control Sample

First we report results for the comparison of all the estimators for the experimental sample in Table 5. We draw 10,000 samples from the population distribution and calculate the estimated treatment effect for each sample and each of the thirteen estimators. The population value of the treatment effect is the average treatment effect for the generated population of one million individuals. We report the average bias of each estimator across the 10,000 samples, the standard deviation for each estimator across the 10,000 samples, the root-mean-squared error (RMSE) and the coverage rates over the 10,000 replications.

For the experimental sample, the RMSEs for the different estimators are fairly similar, ranging from 0.064 (for the residual balancing estimator) to 0.77 for the bias corrected matching estimator. Since there is balance between the treatment and control group due to random assignment of the treatment, it is not surprising that all methods perform fairly

Table 4: Estimates Based on LDW Data

	Experimental		CPS		PSID	
	estimate	s.e.	estimate	s.e.	estimate	s.e.
Difference in Means	1.79	(0.67)	-8.50	(0.58)	-15.20	(0.66)
Bias Corrected Matching	1.90	-	2.35	-	1.47	-
Outcome Models						
Linear	1.00	(0.57)	0.69	(0.60)	0.79	(0.60)
Random Forest	1.73	(0.58)	0.92	(0.6)	0.06	(0.63)
Neural Nets	2.07	(0.59)	1.43	(0.59)	2.12	(0.59)
Propensity Score Weighting						
Linear	1.81	(0.83)	1.18	(0.77)	1.26	(1.13)
Random Forest	1.78	(0.94)	0.65	(0.77)	-0.46	(1.00)
Neural Nets	1.92	(0.87)	1.26	(0.93)	0.10	(1.28)
Double Robust Methods						
Linear	1.80	(0.67)	1.27	(0.65)	1.50	(0.97)
Random Forest	1.84	(0.8)	1.46	(0.63)	1.34	(0.85)
Neural Nets	2.15	(0.74)	1.52	(0.75)	1.14	(1.08)
Causal Forest	1.71	(0.68)	1.62	(0.66)	1.25	(0.82)
Residual Balancing	1.80	(0.68)	1.02	(0.62)	1.19	(0.79)

well. The biases are low relative to the standard deviations, and as a result the coverage rates for the nominal 95% confidence intervals are accurate.

Table 5: Estimates Based on LDW Experimental Data (10,000 Replications)

	Experimental				
Method	RMSE	bias	s.d.	coverage	power
Difference in Means	0.65	0.27	0.58	0.91	1
Bias Corrected Matching	0.72	-0.06	0.72	0.71	1
Outcome Models					
Linear	0.64	0.13	0.62	0.87	1
Random Forest	0.61	0.07	0.60	0.90	1
Neural Nets	0.97	0.33	0.91	0.74	1
Propensity Score Models					
Linear	0.66	0.16	0.64	0.98	1
Random Forest	0.63	0.12	0.62	0.99	1
Neural Networks	0.68	0.26	0.62	0.97	1
Double Robust Methods					
Linear	0.65	0.16	0.63	0.94	1
Random Forest	0.65	0.07	0.65	0.95	1
Neural Nets	0.72	0.21	0.69	0.94	1
Causal Forest	0.59	0.03	0.59	0.95	1
Residual Balancing	0.63	0.15	0.62	0.95	1

4.4 Simulation Results for the CPS Control Sample

Next, we report results for the comparison of the twelve estimators for the CPS comparison sample in Table 6. As expected, given the substantial differences in characteristics between the treatment group and the control group, in this exercise we see substantial differences in the performances of the different estimators. The methods relying on outcome modeling only, or propensity score modelling only, do poorly other than those that rely on neural nets for estimation of nuisance parameters. The flexible double robust methods do well here. The biases for some of the estimators are substantial, contributing to their confidence intervals having poor coverage rates.

Table 6: Estimates Based on LDW-CPS Data (10,000 Replications)

	CPS				
Method	RMSE	bias	s.d.	coverage	power
Difference in Means	9.93	-9.92	0.41	0.00	0.00
Bias Corrected Matching	0.78	-0.49	0.61	0.11	0.94
Outcome Models					
Linear	0.76	-0.62	0.45	0.68	0.97
Random Forest	0.78	-0.64	0.45	0.62	0.96
Neural Nets	0.50	-0.01	0.50	0.89	1.00
Propensity Score Weighting					
Linear	0.51	-0.25	0.45	0.98	1.00
Random Forest	0.84	-0.71	0.45	0.84	0.95
Neural Nets	0.49	-0.02	0.49	0.99	1.00
Double Robust Methods					
Linear	0.49	-0.19	0.45	0.94	1.00
Random Forest	0.48	-0.12	0.47	0.93	1.00
Neural Nets	0.48	0.00	0.48	0.96	1.00
Causal Forest	0.47	-0.09	0.46	0.93	1.00
Residual Balancing	0.65	-0.48	0.44	0.80	0.99

4.5 Simulation Results for the PSID Control Sample

Third, we report results for the comparison of the twelve estimators for the psid comparison sample in Table 7. Here the linear methods do suprisingly well, outperforming all the methods using random forests and neural nets with the exception of the Causal Forest. The double robust methods do reasonably well overall. Note that the linear methods do particularly well in terms of bias. Recall, however, that the WGAN generated data were not as similar to the real data in this case as for the CPS and experimental sample, as evidenced by the goodness of fit measures in Table 3. These results may therefore not be quite as reliable as in the other cases.

Table 7: Estimates Based on LDW-PSID Data (10,000 Replications)

	PSID				
Method	RMSE	bias	s.d.	coverage	power
Baselines					
Difference in Means	16.96	-16.95	0.54	0.00	0.00
Bias Corrected Matching	1.12	0.40	1.05	0.23	0.97
Outcome Models					
Linear	0.89	-0.59	0.67	0.75	0.96
Random Forest	1.82	-1.68	0.70	0.18	0.55
Neural Nets	2.48	0.12	2.47	0.55	0.92
Propensity Score Weighting					
Linear	1.07	-0.16	1.06	0.96	0.93
Random Forest	2.09	-1.91	0.86	0.46	0.43
Neural Nets	1.63	-1.15	1.15	0.82	0.71
Double Robust Methods					
Linear	1.14	0.46	1.05	0.91	0.98
Random Forest	0.98	-0.37	0.91	0.88	0.93
Neural Nets	1.11	0.20	1.09	0.90	0.95
Causal Forest	0.97	-0.53	0.82	0.82	0.93
Residual Balancing	0.71	0.17	0.68	0.93	1.00

5 Robustness of the Simulations

The algorithm developed in this paper leads, for a given data set, to a RMSE for each estimator, and, based on that, a unique ranking of a set of estimators. However, it does not come with a measure of robustness of that ranking or the RMSE it is based on. In principle, the estimated RMSEs and the implied ranking of the estimators could be quite different if we change the set up. In particular we may be concerned with the robustness of the bias component of the RMSE. In this section we discuss a number approaches to assessing how robust the rankings are.

5.1 Robustness to Sample

We apply the WGANs to $M = 10$ samples drawn without replacement from the original sample. Each sample is 80% of the size of the original sample. We use these subsamples to train a WGAN and for each WGAN, draw 10,000 samples from the population distribution

and calculate RMSE, bias, standard deviation, coverage, and power. The main question of interest is by how much the results vary across the different subsamples of the data. The table gives the average of each metric of interest, calculated across the 10 different synthetic populations trained from the 10 different subsamples of original data. The averages are close to the point estimates of the metrics from the full sample. We also show the standard deviation; for most metrics and estimators, the metric estimates vary little for synthetic populations trained on different 80% subsamples of the dataset.

Table 8: Robustness of Ranking for LDW-CPS, Average and Standard Deviations of Metrics over $M = 10$ Samples Drawn from Original Sample

Method	RMSE	bias	s.d.	coverage	power
DIFF	9.64 (0.77)	-9.63 (0.77)	0.39 (0.03)	0.00 (0.00)	0.00 (0.00)
BCM	0.79 (0.29)	0.05 (0.60)	0.61 (0.03)	0.12 (0.05)	0.64 (0.22)
Outcome Models					
L	0.69 (0.30)	-0.1 (0.63)	0.44 (0.04)	0.72 (0.27)	0.57 (0.30)
RF	0.61 (0.12)	-0.41 (0.16)	0.44 (0.04)	0.78 (0.09)	0.36 (0.26)
NN	0.50 (0.04)	0.02 (0.09)	0.5 (0.04)	0.87 (0.01)	0.65 (0.23)
Propensity Score Weighting					
L	0.51 (0.07)	-0.12 (0.22)	0.45 (0.04)	0.99 (0.01)	0.57 (0.28)
RF	0.68 (0.14)	-0.51 (0.17)	0.44 (0.04)	0.95 (0.05)	0.30 (0.24)
NN	0.49 (0.04)	-0.03 (0.05)	0.49 (0.04)	0.99 (0.00)	0.61 (0.27)
Double Robust Methods					
L	0.50 (0.05)	-0.07 (0.22)	0.45 (0.04)	0.94 (0.02)	0.60 (0.27)
RF	0.47 (0.04)	-0.02 (0.09)	0.46 (0.04)	0.93 (0.01)	0.63 (0.25)
NN	0.49 (0.05)	0.01 (0.05)	0.49 (0.05)	0.95 (0.00)	0.63 (0.26)
CF	0.48 (0.04)	-0.07 (0.09)	0.46 (0.04)	0.93 (0.02)	0.60 (0.28)
RB	0.61 (0.18)	-0.33 (0.31)	0.45 (0.04)	0.80 (0.16)	0.41 (0.27)

5.2 Robustness to Model Architecture

We also investigate the robustness to the architecture of the critic and generator, within a similar complexity class of neural networks. Recall that the architecture of the generator and critic both have three hidden layers, with dimensions $(d_X + M, 128)$, $(128, 128)$ and $(128, 128)$. The first alternative architecture (Alt1) considered has a generator hidden layer with dimensions $[64, 128, 256]$ and a critic hidden layer with dimensions $[256, 128, 64]$. The second alternative architecture (Alt2) considered has a generator hidden layer with dimen-

sions $[128, 256, 64]$ and a critic hidden layer with dimensions $[64, 256, 128]$. We do not find that our results are overly sensitive to a certain WGAN architecture. We find that the RMSE, bias, and standard deviation estimates or each treatment effect estimator are similar for the main and two alternative specifications.

Table 9: Robustness to Model Architecture for LDW-CPS

	RMSE			bias			s.d.		
Method	Main	Alt1	Alt2	Main	Alt1	Alt2	Main	Alt1	Alt2
DIFF	9.93	10.34	9.78	-9.92	-10.33	-9.77	0.41	0.45	0.43
BCM	0.78	0.87	0.65	-0.49	-0.56	0.07	0.61	0.67	0.64
Outcome Models									
L	0.76	1.08	0.75	-0.62	-0.95	-0.55	0.45	0.51	0.51
RF	0.78	0.77	0.72	-0.64	-0.59	-0.54	0.45	0.49	0.48
NN	0.50	0.55	0.55	-0.01	0.01	-0.02	0.50	0.55	0.55
Propensity Score Weighting									
L	0.51	0.57	0.68	-0.25	-0.27	-0.46	0.45	0.51	0.51
RF	0.84	0.88	0.84	-0.71	-0.72	-0.68	0.45	0.50	0.49
NN	0.49	0.54	0.53	-0.02	-0.01	-0.03	0.49	0.54	0.53
Double Robust Methods									
L	0.49	0.55	0.66	-0.19	-0.22	-0.42	0.45	0.51	0.51
RF	0.48	0.52	0.53	-0.12	-0.04	-0.10	0.47	0.52	0.53
NN	0.48	0.54	0.53	0.00	0.02	0.03	0.48	0.54	0.53
CF	0.47	0.53	0.52	-0.09	-0.10	-0.06	0.46	0.52	0.51
RB	0.65	0.61	0.71	-0.48	-0.35	-0.50	0.44	0.50	0.51

5.3 Robustness to Size of Training Data

Next we change the size of the training sample to some fraction of the original sample. This is likely to make the generator more smooth because it has fewer data to be trained on. We still generate samples from the generator that are the same size as the original sample. The results are in Table 10.

Table 10: RMSE for Estimators on LDW-CPS for Different Training Data Sizes

	RMSE								
Fraction Used → Method	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
DIFF	7.38	10.07	9.82	10.14	10.18	8.31	10.81	8.73	9.09
BCM	0.54	0.67	0.78	0.57	1.30	1.79	0.80	0.66	0.63
Outcome Models									
L	1.37	1.27	0.56	0.44	0.92	1.44	0.50	0.48	0.92
RF	0.40	0.47	0.61	0.47	0.45	0.52	0.72	0.45	0.74
NN	0.37	0.47	0.55	0.46	0.48	0.54	0.53	0.52	0.51
Propensity Score Weighting									
L	0.37	0.45	0.50	0.43	0.43	0.68	0.54	0.47	0.69
RF	0.35	0.48	0.61	0.52	0.45	0.74	0.79	0.53	0.91
NN	0.38	0.45	0.53	0.44	0.47	0.54	0.55	0.49	0.52
Double Robust Methods									
L	0.37	0.45	0.50	0.43	0.43	0.67	0.53	0.47	0.65
RF	0.37	0.45	0.51	0.43	0.47	0.52	0.51	0.48	0.49
NN	0.36	0.45	0.52	0.43	0.46	0.53	0.53	0.48	0.50
CF	0.36	0.45	0.52	0.43	0.47	0.51	0.51	0.47	0.51
RB	0.45	0.52	0.55	0.42	0.45	0.52	0.62	0.47	0.65

6 Conclusion

In this paper we show how WGANs can be used to tie Monte Carlo studies to real data. This has the benefit of ensuring that simulation studies are grounded in realistic settings, removing the suspicion that they were chosen to support the preferred methods. In this way the simulations studies will be more credible to the readers. We illustrate these methods comparing different estimators for average treatment effects using the Lalonde-Dehejia-Wahba data. There are a number of findings. First, in the three different settings, the experimental data, the CPS control group and the PSID control group, different estimators emerge at the top. Within a particular sample the results appear to be relatively robust to changes in the analysis (e.g., changing the sample size, or doing the two-stage WGAN robustness analysis). Second, the preference in the theoretical literature for double robust estimators is broadly mirrored in our results. Although the flexible double robust estimators (using

random forests or neural nets) do not always outperform the other estimators, the loss in terms of root-mean-squared-error is always modest, where other estimators often perform particularly poorly in some settings. If one were to look for a single estimator in all settings, our recommendation would thereefore be the double robust estimator using random forests or neural nets. However, one may do better in a specific setting by using the WGANs to assess the relative performance of a wider range of estimators.

Appendix: The Estimators

1. Difference in Means (DIFF)

$$\tau^{\text{dm}} = \frac{1}{N_1} \sum_{i:W_i=1} Y_i - \frac{1}{N_0} \sum_{i:W_i=0} Y_i.$$

2. The Bias-Adjusted Matching estimator (BCM)

   (a) Match all treated units with replacement to control units using diagonal version of Mahalanobis matching.

   (b) Regress difference between treated and control outcome for matched pairs on difference in covariates. See Abadie and Imbens [2011].

3. Conditional Outcome Model, Linear Model (LIN)

$$\hat{\tau}^{\text{cm,lm}} = \frac{1}{N_1} \sum_{i:W_i=1} \Big(Y_i - \hat{\mu}^{\text{lm}}(0, X_i) \Big).$$

4. Conditional Outcome Model, Random Forest (RF)

$$\hat{\tau}^{\text{cm,rf}} = \frac{1}{N_1} \sum_{i:W_i=1} \Big(Y_i - \hat{\mu}^{\text{rf}}(0, X_i) \Big).$$

5. Conditional Outcome Model, Neural Nets (NN)

$$\hat{\tau}^{\text{cm,nn}} = \frac{1}{N_1} \sum_{i:W_i=1} \Big(Y_i - \hat{\mu}^{\text{nn}}(0, X_i) \Big).$$

6. The Horowitz-Thompson Estimator, Logit Model (LIN)

$$\hat{\tau}^{\text{ht,lm}} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - (1-W_i) Y_i \frac{\hat{e}^{\text{lm}}(X_i)}{1-\hat{e}^{\text{lm}}(X_i)} \Big/ \frac{1}{N_1} \sum_{j=1}^{N} (1-W_j) \frac{\hat{e}^{\text{lm}}(X_j)}{1-\hat{e}^{\text{lm}}(X_j)} \right).$$

   See Hirano et al. [2003].

7. The Horowitz-Thompson Estimator, Random Forest (RF)

$$\hat{\tau}^{\text{ht,rf}} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - (1-W_i) Y_i \frac{\hat{e}^{\text{rf}}(X_i)}{1-\hat{e}^{\text{rf}}(X_i)} \Big/ \frac{1}{N_1} \sum_{j=1}^{N} (1-W_j) \frac{\hat{e}^{\text{rf}}(X_j)}{1-\hat{e}^{\text{rf}}(X_j)} \right).$$

8. The Horowitz-Thompson Estimator, Neural Net (NN)

$$\hat{\tau}^{\text{ht,nn}} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - (1-W_i) Y_i \frac{\hat{e}^{\text{nn}}(X_i)}{1-\hat{e}^{\text{nn}}(X_i)} \Big/ \frac{1}{N_1} \sum_{j=1}^{N} (1-W_j) \frac{\hat{e}^{\text{nn}}(X_j)}{1-\hat{e}^{\text{nn}}(X_j)} \right).$$

9. The Double Robust Estimator, Linear and Logit Model (LIN)

$$\hat{\tau}^{\text{dr,lm}} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - \hat{\mu}^{\text{lm}}(0, X_i) - (1 - W_i)(Y_i - \hat{\mu}^{\text{lm}}(0, X_i) \frac{\hat{e}^{\text{lm}}(X_i)}{1 - \hat{e}^{\text{lm}}(X_i)} \right).$$

10. The Double Robust Estimator, Random Forest (RF)

$$\hat{\tau}^{\text{dr,rf}} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - \hat{\mu}^{\text{rf}}(0, X_i) - (1 - W_i)(Y_i - \hat{\mu}^{\text{rf}}(0, X_i) \frac{\hat{e}^{\text{rf}}(X_i)}{1 - \hat{e}^{\text{rf}}(X_i)} \right).$$

11. The Double Robust Estimator, Neural Nets (NN)

$$\hat{\tau}^{\text{dr,nn}} = \frac{1}{N_1} \sum_{i:W_i=1} \left(Y_i - \hat{\mu}^{\text{nn}}(0, X_i) - (1 - W_i)(Y_i - \hat{\mu}^{\text{nn}}(0, X_i) \frac{\hat{e}^{\text{nn}}(X_i)}{1 - \hat{e}^{\text{nn}}(X_i)} \right).$$

12. Residual Balancing Estimator (RB)

    (a) Estimate conditional outcome mean for controls by elastic net.

    (b) Construct weights that balance control covariates to average covariate values for treated.

    (c) Combine to estimate average outcome for treated units under control treatment. See Athey et al. [2018].

13. Causal Forest Estimator (CF) See Athey et al. [2019].

Appendix: The Adam Algorithm

Algorithm 3 Adam

▷ Tuning parameters:
    $m =$, batch size
    α, step size
    β_1,
    β_2,
    $\epsilon = 10^{-8}$,
▷ Starting Values:
    $\theta = 0$, $m_0 = 0$, $v_0 = 0$, $t = 0$
while θ has not converged **do**
    $\triangleright$ $t \leftarrow t + 1$
    Sample $\{Z_i\}_{i=1}^m$.
    ▷ Compute gradient
    $\delta_\theta \leftarrow \frac{1}{m} \sum_{i=1}^m \nabla_\theta f\left(Z_i; \theta_t\right)$
    $\gamma_\theta \leftarrow \frac{1}{m} \sum_{i=1}^m \left(\nabla_\theta f\left(Z_i; \theta_t\right)\right)^2$
    $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)\delta_\theta$
    $\hat{m}_t = m_t / (1 - \beta_1^t)$
    $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)\gamma_\theta$
    $\hat{v}_t = v_t / (1 - \beta_2^t)$ ▷ Update θ
    $\theta_t \leftarrow \theta_{t-1} - \alpha \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (update generator parameter)
    end while

References

Alberto Abadie and Matias D Cattaneo. Econometric methods for program evaluation. Annual Review of Economics, 10:465–503, 2018.

Alberto Abadie and Guido W Imbens. Bias-corrected matching estimators for average treatment effects. Journal of Business & Economic Statistics, 29(1):1–11, 2011.

Arun Advani, Toru Kitagawa, and Tymon Słoczyński. Mostly harmless simulations? using monte carlo studies for estimator selection. Journal of Applied Econometrics, 2019.

Martin Arjovsky and Léon Bottou. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862, 2017.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. arXiv preprint arXiv:1701.07875, 2017.

Susan Athey and Guido W Imbens. Machine learning methods that economists should know about. Annual Review of Economics, 11, 2019.

Susan Athey, Guido W Imbens, and Stefan Wager. Approximate residual balancing: debiased inference of average treatment effects in high dimensions. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 80(4):597–623, 2018.

Susan Athey, Julie Tibshirani, Stefan Wager, et al. Generalized random forests. The Annals of Statistics, 47(2):1148–1178, 2019.

Atilim Gunes Baydin, Robert Cornish, David Martinez Rubio, Mark Schmidt, and Frank Wood. Online learning rate adaptation with hypergradient descent. arXiv preprint arXiv:1703.04782, 2017.

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. Inference on treatment effects after selection among high-dimensional controls. The Review of Economic Studies, 81(2):608–650, 2014.

Ernst R Berndt, Bronwyn H Hall, Robert E Hall, and Jerry A Hausman. Estimation and inference in nonlinear structural models. In Annals of Economic and Social Measurement, Volume 3, number 4, pages 653–665. NBER, 1974.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010, pages 177–186. Springer, 2010.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. The Econometrics Journal, 21(1):C1–C68, 2018.

Richard K Crump, V Joseph Hotz, Guido W Imbens, and Oscar A Mitnik. Dealing with limited overlap in estimation of average treatment effects. Biometrika, pages 187–199, 2009.

Rajeev H Dehejia and Sadek Wahba. Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs. Journal of the American statistical Association, 94(448): 1053–1062, 1999.

Rajeev H Dehejia and Sadek Wahba. Propensity score-matching methods for nonexperimental causal studies. Review of Economics and statistics, 84(1):151–161, 2002.

Bradley Efron. The jackknife, the bootstrap and other resampling plans. SIAM, 1982.

Bradley Efron and Robert J Tibshirani. An Introduction to the Bootstrap, volume 57. Chapman & Hall/CRC, 1994.

Max H Farrell, Tengyuan Liang, and Sanjog Misra. Deep neural networks for estimation and inference: Application to causal effects and other semiparametric estimands. arXiv preprint arXiv:1809.09953, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In Advances in neural information processing systems, pages 5767–5777, 2017.

Wolfgang Härdle. Applied nonparametric regression. Number 19. Cambridge university press, 1990.

James J Heckman and V Joseph Hotz. Choosing among alternative nonexperimental methods for estimating the impact of social programs: The case of manpower training. Journal of the American statistical Association, 84(408):862–874, 1989.

Keisuke Hirano, Guido W Imbens, and Geert Ridder. Efficient estimation of average treatment effects using the estimated propensity score. Econometrica, 71(4):1161–1189, 2003.

Martin Huber, Michael Lechner, and Conny Wunsch. The performance of estimators based on the propensity score. Journal of Econometrics, 175(1):1–21, 2013.

Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? arXiv preprint arXiv:1511.05101, 2015.

Guido Imbens. Nonparametric estimation of average treatment effects under exogeneity: A review. Review of Economics and Statistics, pages 1–29, 2004.

Guido W Imbens and Donald B Rubin. Causal Inference in Statistics, Social, and Biomedical Sciences. Cambridge University Press, 2015.

T Kaji, Elena Manresa, and Guillaume Poulio. Artificial intelligence for structural estimation. Technical report, New York University, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

Michael Knaus, Michael Lechner, and Anthony Strittmatter. Machine learning estimation of heterogeneous causal effects: Empirical monte carlo evidence. 2018.

Murat Kocaoglu, Christopher Snyder, Alexandros G Dimakis, and Sriram Vishwanath. Causalgan: Learning causal implicit generative models with adversarial training. arXiv preprint arXiv:1709.02023, 2017.

Robert J LaLonde. Evaluating the econometric evaluations of training programs with experimental data. The American economic review, pages 604–620, 1986.

Michael Lechner and Anthony Strittmatter. Practical procedures to deal with common support problems in matching estimation. Econometric Reviews, 38(2):193–207, 2019.

Michael Lechner and Conny Wunsch. Sensitivity of matching-based program evaluations to the availability of control variables. Labour Economics, 21:111–121, 2013.

Tengyuan Liang. On how well generative adversarial networks learn densities: Nonparametric and parametric results. arXiv preprint arXiv:1811.03179, 2018.

Yifan Liu, Zengchang Qin, Tao Wan, and Zhenbo Luo. Auto-painter: Cartoon image generation from sketch by using conditional wasserstein generative adversarial networks. Neurocomputing, 311:78–87, 2018.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.

Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. Journal of Economic Perspectives, 31(2):87–106, 2017.

Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier gans. In Proceedings of the 34th International Conference on Machine Learning-Volume 70, pages 2642–2651. JMLR. org, 2017.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. Biometrika, 70(1):41–55, 1983.

Donald B Rubin. Matched sampling for causal effects. Cambridge University Press, 2006.

Alejandro Schuler, Ken Jung, Robert Tibshirani, Trevor Hastie, and Nigam Shah. Synth-validation: Selecting the best causal inference method for a given dataset. arXiv preprint arXiv:1711.00083, 2017.

Bernard W Silverman. Density estimation for statistics and data analysis. Routledge, 2018.

Stefan Wager, Sida Wang, and Percy S Liang. Dropout training as adaptive regularization. In Advances in neural information processing systems, pages 351–359, 2013.

David Warde-Farley, Ian J Goodfellow, Aaron Courville, and Yoshua Bengio. An empirical analysis of dropout in piecewise linear networks. arXiv preprint arXiv:1312.6197, 2013.

T Wendling, K Jung, A Callahan, A Schuler, NH Shah, and B Gallego. Comparing methods for estimation of heterogeneous treatment effects using observational data from health care databases. Statistics in medicine, 37(23):3309–3324, 2018.

Liyang Xie, Kaixiang Lin, Shu Wang, Fei Wang, and Jiayu Zhou. Differentially private generative adversarial network, 2018.