Using Wasserstein Generative Adversarial Networks for the Design of Monte Carlo Simulations*

Susan Athey† Guido W. Imbens‡ Jonas Metzger§ Evan Munro¶

September 2019

Abstract

Researchers often use artificial data to assess the performance of new econometric methods. In many cases, the data generating processes used in these Monte Carlo studies do not resemble real data sets, and instead reflect arbitrary decisions made by the researchers. Often the data generating processes are derived from simple functional forms and impose regularity in a variety of ways. As a result potential users of the methods are rarely persuaded by Monte Carlo studies that new methods will perform equally well in practice. Here, we propose designing Monte Carlo studies in a different way, in particular using Wasserstein Generative Adversarial Networks (WGANs) as a method for systematically generating artificial data that mimics closely a given real data set. Our approach limits the degrees of freedom for the researcher, while still allowing the researcher to know the "ground truth" in order to compare the performance of estimators. We apply the methods to compare a number of different estimators for average treatment effects under unconfoundedness; we consider three distinct settings (corresponding to three real data sets that serve as the benchmark for the WGANs). In this example, we find that (*i*) there is not one estimator that outperforms the others in all three settings, so that it will be helpful to tailor the analytic approach to the specific setting, and (*ii*) systematic simulation studies can be helpful for selecting among competing methods in a given setting.

JEL Classification: C14, C21, C52

Keywords: Generative Adverserial Networks, Machine Learning, Neural Nets, Average Treatment Effects, Potential Outcomes

*We are grateful for comments by participants in the conference in honor of Whitney Newey in April 2019, and especially for discussions with Whitney. We also want to acknowledge exceptional research assistance by Cole Kissane. Code for the calculations in this paper is available at `https://github.com/evanmunro/design-of-simulations`.

†Graduate School of Business, Stanford University, and NBER. Electronic correspondence: athey@stanford.edu.

‡Graduate School of Business, and Department of Economics, Stanford University, and NBER. Electronic correspondence: imbens@stanford.edu.

§Department of Economics, Stanford University. Electronic correspondence: metzgerj@stanford.edu.

¶Graduate School of Business, Stanford University. Electronic correspondence: munro@stanford.edu,

1 Introduction

When researchers in econometrics develop new econometric methods it is common practice to assess the performance of the new methods in Monte Carlo studies. In such Monte Carlo studies, artificial data are often generated using very simple distributions with a high degree of smoothness and limited dependence between different variables. The performance of the 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]. In this paper we discuss how Generative Adversial 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 actual data sets. Given an actual data set these methods allow researchers to systematically assess the performance of new methods in a realistic setting. Moreoever, it would pre-empt concerns that researchers chose particular simulation designs to favor their proposed methods.

An important feature of many econometric methods is that they focus on causal effects, rather than predictions. This means that in general we need to have more than simply a single real data set to evaluate new methods: we need methods that allow us to generate large amounts of data beyond any given sample to establish the ground truth. This is where the WGANs can be very helpful.

We apply these ideas 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 that make it a challenging setting for estimators for average treatment effects under unconfoundedness. That makes it 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 evaluate the robustness of these results to sampling variation. Our approach to robustness relies on starting with a very large dataset, and repeating our entire analysis treating different samples from the large dataset as the "real" dataset. The very large dataset is generated using the WGAN that was based on our original real dataset.

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 twelve 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, or neural nets.

2 Wasserstein Generative Adversarial Networks

Suppose we have a sample of N observations on K-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

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

Conventional kernels include the Gaussian kernel and the Epanechnikov kernel. Bandwidth choices have been proposed to minimize squared-error loss. These methods tend to perform poorly especially in high-dimensional settings. The estimated densities tend to oversmooth and look substantially different from the actual ones.

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.

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a recently developed an alternative approach to generating data that are similar to a particular sample (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. GANs have two components. The first is a *generator* that is a flexible parametric model for the distribution of data. The second is a *discriminator* that classifies observations as coming from the training sample rather than from the generator. If the generator performs well, the discriminator cannot distinguish between the training data and the artificial data. The training procedure for the generative model then tries to maximize the probability of the discriminator making a mistake. This can be viewed as a two-player minimax game. In typical applications, and in our application, both the generative model and the discriminator are multi-layer neural nets, but in principle they could be simpler models. GANs have recently become popular in the machine learning literature, but have not received much attention in the econometrics literature. An exception is Kaji et al. [2019] who use WGANs for estimation of structural models.

The starting point is a random noise distribution, $p_Z(\cdot)$, with domain $\mathbb{Z}$, selected by the researcher. This can be a multivariate uniform or Gaussian distribution. Second, we have the generator, a mapping from $\mathbb{Z}$ to the data space $\mathbb{X}$ with parameter θ_g. Call this mapping $G(z;\theta_g)$. This may be a multi-layer neural net but could be a much simpler model. For a given parameter θ_g the generator, in combination with the distribution of the inputs $p_Z(\cdot)$ implies a distribution for $\tilde{X}$. We denote this distribution by $p_{\theta_g}(\cdot)$. Next, we have a second model, the discriminator $D(x;\theta_d)$ with parameter θ_d. The output of the generator is the probability that x came from the real data rather than from the generator. Again, typically the discriminator is a multi-layer neural net, but it could be a very simple model. We train the discriminator, using samples from the training data and samples drawn from $p_{\theta_g}(\cdot)$, to

maximize the probability of assigning the correct label to the generated and the real data. At the same time we train the generator to minimize the the logarithm of the probability of assigning the generated data to the correct label, that is we minimize the logarithm of $1 - D(\tilde{X}, \theta_d)$ for observations from the generated data.

2.3 A Simple Example

To fix ideas, let us consider a GAN in a very simple example where both models, the generator and the discriminator, are very parsimonious parametric models rather than multi-layer neural nets. We have a random sample $X_1, \ldots, X_{N_R}$, the "real" data, with N_R observations. Suppose the distribution of the scalar input is a Normal distribution with mean zero and unit variance,

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

Suppose also that the generator is an additive shift:

$$G(z; \theta_g) = z + \theta_g.$$

This implies a model for the generated data that is $\mathcal{N}(\theta_g, 1)$:

$$p_{\theta_g}(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}(x - \theta_g)^2\right).$$

This is of course a very simple model, and given this model we can estimate θ_g on the real data just as the sample average:

$$\hat{\theta}_g = \frac{1}{N_R} \sum_{i=1}^{N_R} X_i.$$

Note that if we generated our artificial observations from a Normal distribution $\mathcal{N}(\hat{\theta}_g, 1)$, the artificial data would look very similar to the real data, and as a result any discriminator would find it impossible to systematically distinguish between the real data and the artificial data.

Let us see how the GAN analyses this problem. For the discriminator we use a logit model with two parameters, $\theta_d = (\theta_{d0}, \theta_{d1})$, with outcome $Y = R$ (a real observation) or $Y = F$ (a fake observation):

$$D(x; \theta_{d0}, \theta_{d1}) = \text{pr}(Y = F | X = x) = \frac{\exp(\theta_{d0} + \theta_{d1} x)}{1 + \exp(\theta_{d0} + \theta_{d1} x)}.$$

For the real data we create an outcome $Y_i = R$. We also generate a sample of N_F observations from the input distribution, $Z_{N_R+1}, \ldots, Z_{N_R+N_F}$, and create an outcome for

these fake observations, $Y_i = F$. Given the parameter for the generator, θ_g, the discriminator chooses θ_d to maximize the standard logit log likelihood function:

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

where

$$L(\theta_d, \theta_g) = \sum_{i:Y_i=R} \ln D(X_i; \theta_d) + \sum_{i:Y_i=F} \ln \left[1 - D\Big(G(Z_i; \theta_g^k); \theta_d\Big)\right].$$

We then choose the parameter θ_g for the generator to minimize the concentrated log likelihood function for the discriminant:

$$\min_{\theta_g} \max_{\theta_d} \left\{ \sum_{i:Y_i=R} \ln D(X_i; \theta_d) + \sum_{i:Y_i=F} \ln \left[1 - D\Big(G(Z_i; \theta_g^k); \theta_d\Big)\right] \right\}.$$

Let us implement this 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

$$\theta_d^{k+1} = \arg\max_{\theta_d} \left\{ \sum_{i:Y_i=R} \ln D(X_i; \theta_d) + \sum_{i:Y_i=F} \ln \Big[1 - D\Big(G(Z_i; \theta_g^k); \theta_d\Big)\Big] \right\}.$$

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

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

To see what this algorithm leads to, consider when the generator is successful in the sense that the discriminator is unable to tell apart the real and fake data. This happens when the solution for the parameter for the discriminant is $(\theta_{d0}, 0)$. Then, inspecting the optimization problem for θ_d, it must be the case that θ_{d0} is equal to $\theta_{d0} = \ln(N_R/(N_R + N_F))$, and thus $\theta_d = (\ln(N_R/(N_R + N_F), 0)$. For this to be the solution to the optimization problem for θ_d, it must be the case that

$$\hat{\theta}_g = \frac{1}{N_R} \sum_{i:Y_i=R} X_i - \frac{1}{N_F} \sum_{i:Y_i=F} Z_i.$$

If the number of observations from the generated model (N_F) is large, then $\sum_{i:Y_i=F} Z_i/N_f \approx 0$, and this will converge to $\hat{\theta}_g = \sum_{i=1}^N X_i/N_R$.

Note that there are some subtleties in this algorithm. An important one is that at each iteration we can solve for the optimal value of θ_d, given the current value for θ_g. However, in

the second step of each iteration we cannot optimize for θ_g given θ_d. In this simple example it is easy to see that as a function of θ_g the log likelihood function is monotone, and so θ_g would diverge. We avoid that by taking a small step in the direction of the derivative.

2.4 Earth Moving Distance and Wasserstein Generative Adversial Networks

The simple GAN as decribed in the previous section does not work well in more complex settings. An attractive alternative is based on a modification of the distance measure. First, for two distributions $\mathbb{P}$ and $\mathbb{P}'$, let the Kullback-Leibler divergence be

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

where both $\mathbb{P}$ and $\mathbb{P}'$ are absolutely continuous with respect to the same measure. Also define the Jensen-Shannon divergence as

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

Goodfellow et al. [2014] and Arjovsky et al. [2017]) show that the GAN as described in Section 2.2 chooses in large samples the generator $p_\theta(\cdot)$ to minimize the Jensen-Shannon divergence between the generator and the distribution $p_X(\cdot)$ that generates the true data. This Jensen-Shannon divergence metric turns out to have some disadvantages, especially when there is a difference in the support of the distributions. This can lead to substantial computational problems where the discriminator is too good early on so that it classifies the real and fake observations well, and as a result the derivatives for the generator vanish. 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 well defined irrespective of the support of the distributions.

Arjovsky et al. [2017] show that an alternative representation of the Wasserstein distance is

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

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*.

Suppose we parametrize the critic as $f(x; \theta_c)$. Ignoring the Lipschitz constraint the optimization problem would be

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

Given the generator, we choose the parameters of the critic, θ_c, 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. Again we need to be careful in taking small steps for the generator, whereas we can take many steps for the critic. Ideally we would restrict the search to parameters that ensure that the critic is Lipschitz with constant 1. That is difficult computationally. In the literature two approaches have been followed (Gulrajani et al. [2017]). The first solution is to restrict the parameters to lie in some compact set, the Cartesian product of $[-\varepsilon, \varepsilon]$. That implies that the function $f(\cdot)$ is $K-$Lipschitz for some K, so that we mininimize K times the Wasserstein distance, for some unknown K. This approach, known as *clipping*, still turns out to have computational problems. A second suggestion is to add a penalty term to the objective function for the critic. The penalty term depends on the average the square of the positive part of the difference between the norm of the derivative of $f(\cdot)$ with respect to the input and one. Specificially, 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 a convex combination of the real and generated observations. 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 2. 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 Adam algorithm are provided in the appendix. Our specific algorithm uses dropout (Warde-Farley et al. [2013]) for regularization purposes as well as a modification of the Adam algorithm that adapts the learning rate (Baydin et al. [2017]), thus reducing the time for hyperparameter search.

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]).

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.

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

Algorithm 1 WGAN

▷ Tuning parameters:
m, batch size
$n_{critic} = 8$, number of critic iterations per iteration of the generator
$lr = 0.0001$,$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$, parameters 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_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

Algorithm 2 CWGAN

▷ Tuning parameters:

m, batch size

$n_{critic} = 8$, 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 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 $Q(\theta_c)$.

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

▷ 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)

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 Imbens [2004], Abadie and Cattaneo [2018] for surveys).

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]). 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.

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^{exp} observations, with $N_0^{\text{exp}} = 260$ control observations and $N_1^{\text{exp}} = N^{\text{exp}} - N_0^{\text{exp}} = 185$ 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. $\mathbf{X}_0^{\text{exp}}$ denotes the $N_0^{\text{exp}} \times 8$ matrix with each row corresponding to the pre-treatment variables for one of these units, and $\mathbf{X}_1^{\text{exp}}$ denoting the $N_1^{\text{exp}} \times 8$ for the treated units in this sample. Let $\mathbf{X}^{\text{exp}} = \left(\mathbf{X}^{\text{exp}_0\top}\mathbf{X}^{\text{exp}_1\top}\right)^\top$ denote the $N^{\text{exp}} \times 8$ matrix with all the covariates. Simiarly, let $\mathbf{Y}_0^{\text{exp}}$ denote the N_0^{exp} vector of outcomes for the control units in this sample, and $\mathbf{Y}_1^{\text{exp}}$ denote the N_1^{exp} vector of outcomes for the treated units, and let $\mathbf{W}_0^{\text{exp}}$ denote the N_0^{exp} vector of treatment indicators for the control units in this sample (all zeros), and $\mathbf{W}_1^{\text{exp}}$ denote the N_1^{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 $N_0^{\text{cps}} = 15,992$ control observations drawn from the Current Population Survey, for a total of $N^{\text{cps}} = N_1^{\text{exp}} + N_0^{\text{cps}} = 16,177$ observations. The third sample is the PSID sample, LDW-PSID. It combines the treated observations from the experimental sample with $N_0^{\text{psid}} = 2,490$ control observations drawn from the Panel Survey of Income Dynamics, for a total of $N^{\text{psid}} = N_1^{\text{exp}} + N_0^{\text{psid}} = 2,675$ observations.

Table 1 presents summary statistics for the eight pretreatment variables in these samples, the treatment indicator and the outcome.

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)

3.3 A Conditional WGAN for the LDW Data

Consider the experimental data set LDW-E. The goal is to create samples of N^{exp} observations, containing $N_0^{\text{exp}} = 260$ control units and $N_1^{\text{exp}} = 185$ treated units, where the samples are similar to the real sample. We proceed as follows. First, we run a conditional WGAN on the sample $\mathbf{X}^{\text{exp}}$, conditional on $\mathbf{W}^{\text{exp}}$. Let the parameters of the generator of the WGAN be $\theta_{g,X}^{\text{exp}}$. 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 are three hidden layers in the neural net, with the number of inputs and outputs equal to $(K + M, 128)$, $(128, 128)$ and $(128, 128)$ respectively. Here K is the dimension of the vectors

whose distribution we are modeling and M is the dimension of the conditioning variables. For the unconditional WGAN this is $K = 8$, and $M = 1$, and for the conditional WGANs this is $K = 1$, 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 K outputs. Here we use for binary variables a sigmoid transformation, for censored variables a rectified linear transformation, and for continuous variables the identify function.

For the critic we use the same architecture with three layers, with the number of inputs and outputs equal to $(K + M, 128)$, $(128, 128)$ 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, $\theta^{\exp}_{g,X|W}$, $\theta^{\exp}_{g,Y(W)|X,W}$, we first create a very large sample, with $N = 10^7$ 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 $\theta^{\exp}_{g,X|W}$. 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 $\theta^{\exp}_{g,Y(W)|X,W}$. 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.

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

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

	experimental				cps				psid			
	treated		control		treated		control		treated		control	
	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd
black	0.85	(0.35)	0.83	(0.38)	0.83	(0.37)	0.09	(0.28)	0.84	(0.36)	0.29	(0.45)
hispanic	0.07	(0.25)	0.10	(0.30)	0.05	(0.22)	0.08	(0.28)	0.07	(0.16)	0.03	(0.16)
age	26.4	(7.0)	26.0	(7.0)	25.6	(7.0)	33.2	(11.3)	26.9	(7.3)	36.5	(11.2)
married	0.19	(0.39)	0.18	(0.38)	0.19	(0.39)	0.72	(0.45)	0.19	(0.39)	0.85	(0.35)
nodegree	0.72	(0.45)	0.83	(0.37)	0.71	(0.45)	0.29	(0.45)	0.72	(0.45)	0.33	(0.47)
education	9.9	(2.0)	9.8	(1.6)	10.3	(2.0)	11.9	(2.9)	9.8	(2.0)	11.4	(3.0)
earn '74	2.13	(4.89)	2.121	(5.35)	2.02	(4.15)	14.61	(9.95)	1.82	(4.16)	18.02	(12.40)
earn '75	1.50	(2.93)	1.42	(3.39)	1.43	(2.86)	14.32	(9.60)	0.93	(2.99)	16.72	(11.92)
earn '78	6.21	(6.35)	4.40	(4.70)	6.00	(5.34)	15.49	(9.46)	5.13	(5.44)	19.28	(13.20)

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 has substantially different shapes, is still well matched by the artificial data. 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. 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.

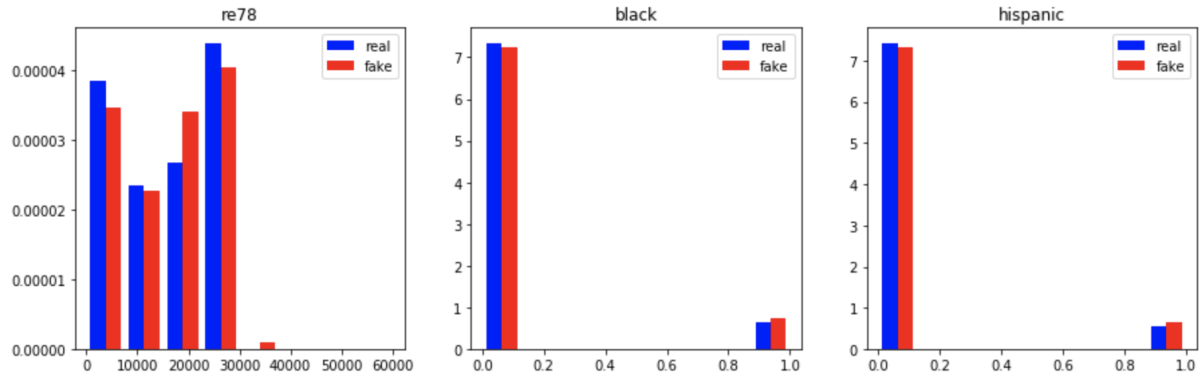

Figure 1: Histograms for CPS Data, Earnings 1978, Black, Hispanic

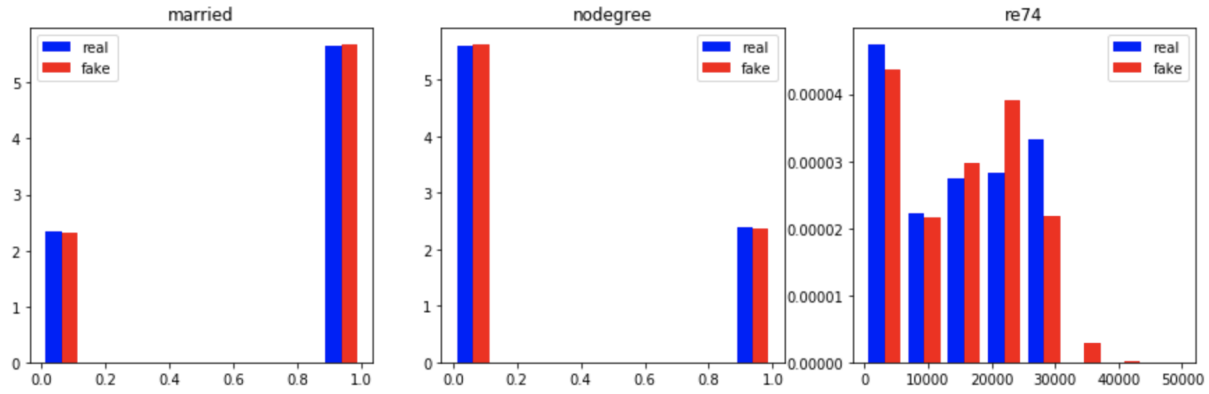

Figure 2: Histograms for CPS Data, Married, No Degree, Earnings 1974

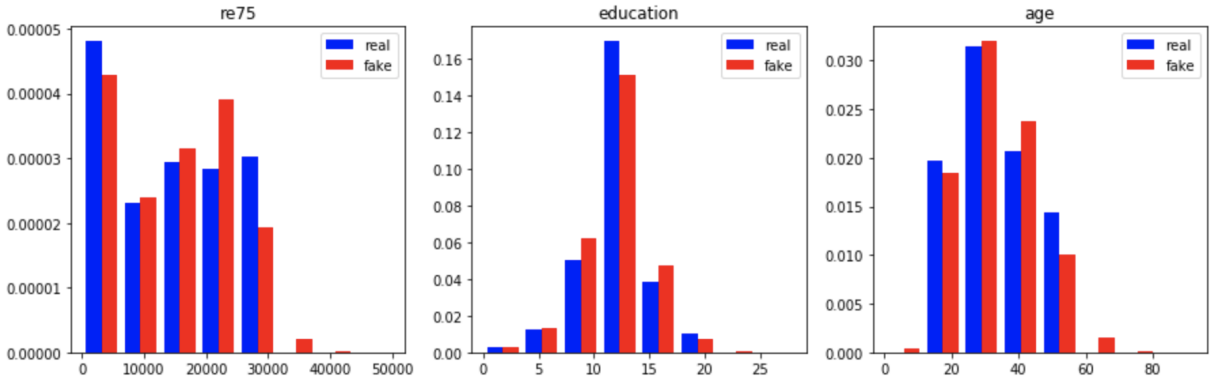

Figure 3: Histograms for CPS Data, Earnings 1975, Education, Age

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 $N = 10^7$, 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.

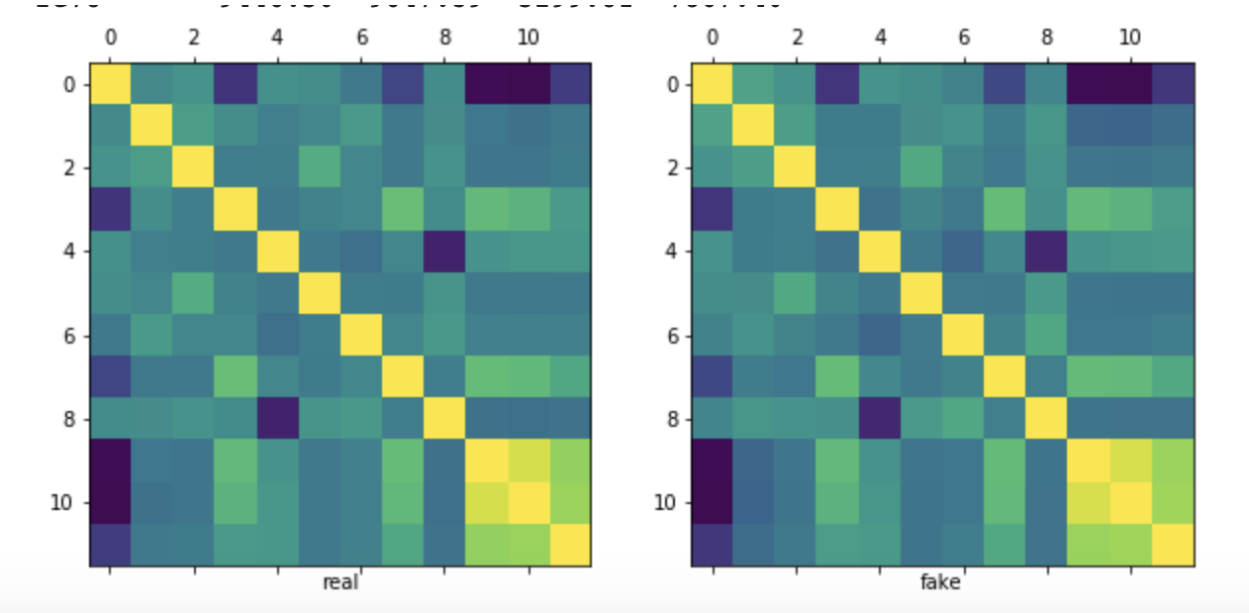

Figure 4: Correlations for CPS Data

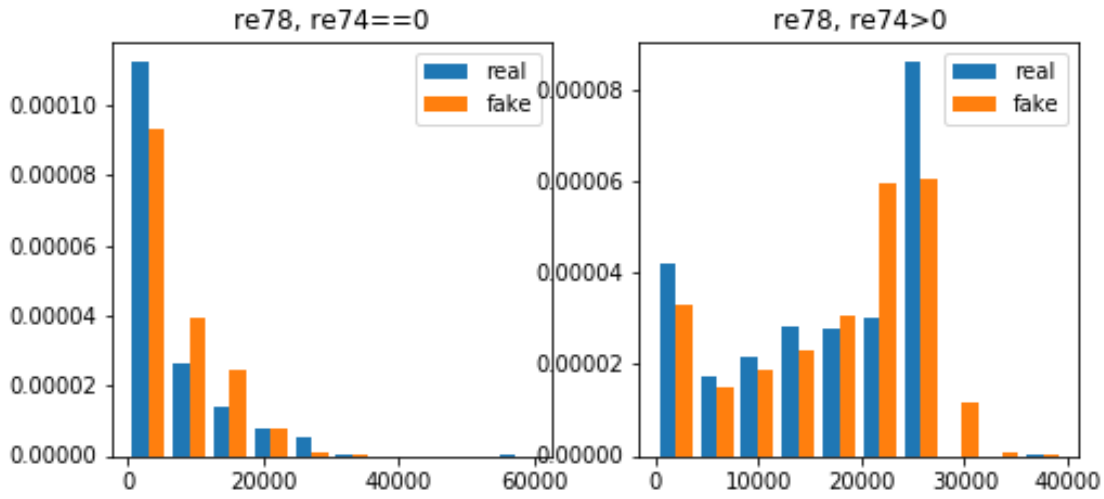

Figure 5: Conditional Histograms for CPS Data, Earnings 1978 Given Earnings 1974 Positive or Zero

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 eleven 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, 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 two 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. In future versions we intend to include additional estimators including the Residual Balancing and dragonnet estimators (Athey et al. [2018], Shi et al. [2019]).

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 $\hat{\mu}^{\rm lm}(0,x)$, and the estimator for the propensity score by $\hat{e}^{\rm lm}(x)$.

2. Random Forests for the propensity score and the conditional outcome mean. Denote the estimator for the conditional mean by $\hat{\mu}^{\rm rf}(0,x)$, and the estimator for the propensity score by $\hat{e}^{\rm rf}(x)$. See Athey and Wager [forthcoming] for an analysis using generalized random forests Athey et al. [2019] in doubly robust estimation.

3. Neural Nets for the propensity score and the conditional outcome mean.Denote the estimator for the conditional mean by $\hat{\mu}^{\rm nn}(0,x)$, and the estimator for the propensity score by $\hat{e}^{\rm nn}(x)$. SeeFarrell 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 we do not use the sample splitting here.

4.2 Estimates for LDW Data

First we use all the estimators on the actual LDW data. The results are reported in Table 3.

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 4. We draw 10,000 samples from the population distribution. We report the bias of the estimators, and standard deviation, the root-mean-squared-error, the coverage rates over the 10,000 replications, and the rmse ranking over the estimators. The rmse's 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 well. The ibiases are low relative to the standard deviations, and as a result the coverage rates for the nominal 95% confidence intervals are accurate.

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 5. As expected, given the substantial differences in characteristics between the treatment group and the control group, in this exercise we see substantial differences

Table 3: Estimates Based on LDW Data

	Experimental		CPS		PSID	
	est	s.e.	est	s.e.	est	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)

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.

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 6. Here the linear methods do suprisingly well, outperforming all the methods using random forests and neural nets. The double robust methods do reasonably

Table 4: Estimates Based on LDW Experimental Data (10,000 Replications)

Estimator	rmse	rmse rank	bias	sdev	coverage
Difference in Means	0.62	9	-0.29	0.55	0.90
Bias Corrected Matching	0.64	10	-0.08	0.64	
Outcome Models					
Linear	0.56	2	-0.06	0.56	0.90
Random Forest	0.58	4	-0.15	0.56	0.89
Neural Nets	0.65	11	-0.17	0.63	0.85
Propensity Score Weighting					
Linear	0.56	3	-0.04	0.56	0.99
Random Forest	0.60	7	-0.17	0.58	0.99
Neural Nets	0.59	5	-0.11	0.58	0.99
Double Robust Methods					
Linear	0.56	1	-0.04	0.56	0.95
Random Forest	0.60	8	-0.08	0.60	0.95
Neural Nets	0.59	6	-0.09	0.59	0.95

well overall. Note that the linear methods do particularly well in terms of bias.

4.6 How Credible are the Rankings

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. The question arises becaues in principle the ranking of the estimators could be quite different if we change the environment slightly. In particular we may be concerned with the robustness of the bias component of the rmse. In this section we discuss two approaches to assessing how robust the rankings are.

In the first approach we change the number of observations we draw from the population distribution generated by the WGAN. We focus here on the PSID estimates. Given

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

Estimator	rmse	rank	bias	sdev	coverage
Difference in Means	10.50	11	-10.49	0.40	0.00
Bias Corrected Matching	0.71	7	-0.37	0.61	0.00
Outcome Models					
Linear	0.77	8	-0.62	0.45	0.67
Random Forest	0.80	9	-0.67	0.44	0.62
Neural Nets	0.51	3	-0.10	0.50	0.89
Propensity Score Weighting					
Linear	0.66	6	-0.47	0.46	0.95
Random Forest	0.89	10	-0.77	0.45	0.86
Neural Nets	0.52	4	0.09	0.51	0.98
Double Robust Methods					
Linear	0.64	5	-0.45	0.45	0.84
Random Forest	0.50	1	-0.13	0.48	0.92
Neural Nets	0.50	2	0.01	0.50	0.96

the estimated WGAN, we originally drew many samples with $N_1 = 185$ treated units and $N_0 = 2,490$ control units. Now we change this by drawing repeatedly samples with $N_1 = 945$ treated units and $N_0 = 12,450$ control units, five times as many from each group as previously. In the second part of Table 6 we present the same statistics we reported previously for these samples. We find that for estimators that are increasingly flexible as the sample size increases, those involving random forests or neural nets, and the matching estimator, the bias goes down, sometimes substantially. The standard deviation goes down for all estimators, roughly to the extent expected. As a result the rankings do not change much. It remains the case that the linear model and the logit propensity score weighting do well, but the double robust methods based on random forests and neural nets remain competitive. It would be interesting to see if there are other robustness analyses that would change the

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

	185 treated, 2,490 Controls					945 Treated, 12,450 Controls				
Estimator	rmse	rank	bias	sdev	cov	rmse	rank	bias	sdev	cov
DIM	15.18	12	-15.17	0.48	0.00	15.17	11	-15.17	0.21	0.00
BCM	0.88	8	0.42	0.77	0.00	0.51	7	0.38	0.35	0.00
Outcome Models										
L	0.57	1	0.09	0.56	0.88	0.27	1	0.09	0.25	0.86
RF	0.97	10	-0.79	0.57	0.52	0.63	10	-0.57	0.27	0.22
NN	1.20	11	0.85	0.85	0.48	0.62	9	0.50	0.36	0.36
Propensity Score Weighting										
L	0.67	2	-0.01	0.67	0.98	0.29	2	-0.02	0.29	0.99
RF	0.91	9	-0.65	0.64	0.94	0.53	8	-0.44	0.29	0.81
NN	0.83	7	-0.40	0.73	0.96	0.31	3	0.06	0.30	0.98
Double Robust Methods										
L	0.72	4	0.27	0.67	0.93	0.38	6	0.23	0.30	0.87
RF	0.70	3	0.11	0.69	0.91	0.33	4	0.07	0.32	0.89
NN	0.76	6	0.35	0.67	0.90	0.34	5	0.17	0.30	0.91

relative performance of the linear regression estimator here.

In the second approach we do the following. Given our trained WGAN, we take $M = 10$ random samples from its distribution. Now we treat in turn each of these M artificial samples as the original sample. We then go through the same procedure as before, estimating a WGAN on the m-th artificial sample, and comparing the twelve estimators on the distribution implied by the WGAN estimated on the m-th sample. We then compare the rmse from the procedure applied to the m-th artificial sample to the rmse based on the WGAN trained on the actual sample. In Table 7 we report the results for the LDW-CPS. We report the average, over the $M = 10$ replications/samples the rmse, bias and standard deviation, and we report the standard deviation over these M replications. The methods that did well on the original sample typically do well in these M replications. For example the double

robust random forest had an rmse of 0.50, whereas the linear double robust method has on the original sample an rmse of 0.64. Over the M artificial samples the rmse for the double robust RF was 0.45, with a standard deviation of 0.05. For the double robust L method the average rmse is 0.66, with a standard deviation of 0.05. It typically ranks below the double robust RF method.

For this sample the tentative conclusion is that the flexible double robust methods (using neural nets or random forests do well), consistent with the theory. The neural net version of propensity score weighting also does well, but since this is less consistent with theory, we would not recommend this.

Both these exercises suggest that the WGAN-based simulations do a good job in recovering the bias, and that ranking estimators based on such WGANs may be an effective way of choosing between competing methods.

5 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, and not chosen to support the preferred methods. In this way the simulations studies will be more informative to the readers. We illustrate these methods comparing different estimators for average treatment effects using the Lalonde-Dehejia-Wahba. 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.

Table 7: Robustness of Ranking for LDW-CPS, Average and Standard Deviations of Metrics over $M = 10$ Samples

	Original Estimator			Ave Simulations			Standard Dev Sims		
	RMSE	Bias	sdev	RMSE	Bias	sdev	RMSE	Bias	sdev
DIM	10.50	-10.49	0.40	10.46	-10.45	0.38	0.52	0.52	0.03
BCM	0.71	-0.37	0.61	0.78	-0.11	0.59	0.15	0.55	0.03
Outcome Models									
L	0.77	-0.62	0.45	0.89	-0.66	0.42	0.40	0.61	0.02
RF	0.80	-0.67	0.44	0.66	-0.50	0.40	0.19	0.24	0.03
NN	0.51	-0.10	0.50	0.50	-0.10	0.46	0.05	0.17	0.03
Propensity Score Weighting									
L	0.66	-0.47	0.46	0.66	-0.44	0.43	0.24	0.35	0.04
RF	0.89	-0.77	0.45	0.74	-0.60	0.40	0.23	0.27	0.03
NN	0.52	0.09	0.51	0.46	0.06	0.45	0.05	0.06	0.05
Double Robust Methods									
L	0.64	-0.45	0.45	0.66	-0.44	0.43	0.23	0.34	0.04
RF	0.50	-0.13	0.48	0.45	-0.09	0.43	0.05	0.13	0.04
NN	0.50	0.01	0.50	0.45	-0.02	0.44	0.05	0.05	0.05

Appendix: The Estimators

1. Difference in Means

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

2. The Bias-Adjusted Matching estimator

   Bias Corrected Matching: 1. match all treated units with replacement to control units using diagonal version of Mahalanobis matching. 2. 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

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

4. Conditional Outcome Model, Random Forest

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

5. Conditional Outcome Model, Neural Nets

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

6. The Horowitz-Thompson Estimator, Logit Model

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

See Hirano et al. [2003].

7. The Horowitz-Thompson Estimator, Random Forest

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

8. The Horowitz-Thompson Estimator, Neural Net

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

9. The Double Robust Estimator, Linear and Logit Model

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

10. The Double Robust Estimator, Random Forest

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

11. The Double Robust Estimator, Neural Nets

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

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 Stefan Wager. Estimating treatment effects with causal forests: An application. Observational Studies, forthcoming.

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.

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. 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.

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.

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.

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.

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.

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.

Claudia Shi, David M Blei, and Victor Veitch. Adapting neural networks for the estimation of treatment effects. arXiv preprint arXiv:1906.02120, 2019.

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

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.