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

Susan Athey[1] Guido W.Imbens[1] J.Metzger[1] Evan Munro[1]

[1]Department of Econometrics, Stanford University

Bocconi, PhD Statistics and Computer Science, 10 June 2024

Motivation I

The objective of many econometric studies is assessing the performance of new causal estimators in comparison to existing ones.

This is often done through Monte Carlo simulations with researcher-chosen distributions that allow access to the true causal effects underlying the data.

Limitations:

1. Chosen distributions (high smoothness, limited dependence between variables) may be unrepresentative of real-world settings.
2. Researchers may tailor simulation designs to favor their proposed methods.

Motivation II

An applied researcher, faced with choosing a method of causal inference for his data, might be interested in comparing estimators in a specific finite-sample setting, even when attractive large-sample properties hold for some methods. Rankings of estimators proposed in MC and EMC studies have been shown to be inconsistent in this setting. (Advani et al.[1])

The authors[3] show how Wasserstein GANs can be used to:

1. Systematically generate samples closely mimicking existing data
2. Tie MC simulations to real data generating processes
3. Assess the performance of causal inference estimators in settings that are more realistic than those commonly used in MC studies

Table of Contents

Setting

Assume to have a sample of N observations on d_X-component vectors, $X_1, ..., X_N$, drawn randomly from a distribution with cumulative $P(\cdot)$, density $p_X(\cdot)$ and domain $\mathbb{X}$.

Such distribution is unknown, and we want to obtain samples that are as similar as possible to those drawn from it. Common solutions to this problem involve the parametric or non-parametric estimation of $p_X(\cdot)$ through a distribution $\hat{p}_X(\cdot)$.

Conventional Approaches to Non-parametric Estimation

- Bootstrapping Techniques: Unsuitable for comparing ATT estimators based on the propensity score.

$$\hat{P}(x) = \frac{1}{N} \sum_{i=1}^{N} 1_{X_i \leq x}$$

- Kernel Density Estimation (bandwidth h, kernel $K(\cdot)$): Performs poorly in high-dimensional settings and when involved distributions have bounded support.

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

Parametric Approaches

We can define a parametric family of densities $(P_\theta)_{\theta \in R^d}$, and estimate the parameters that maximize the likelihood of the data

$$\max_{\theta \in R_D} \frac{1}{N} \sum_{i=1}^{N} \log P_\theta(x^{(i)})$$

Asymptotically, this is equivalent to minimizing the KL divergence between the true data distribution and the estimated one.

- Choice of the parametric family is tipically subject to researcher's discretion
- A noise component must be introduced in the estimated model for the problem to be well-defined, this may degrade the quality of generated samples.

Generative Models

Rather than estimating the density $\hat{p}_X$, we can define a latent random variable Z, and pass it through a parametric function (neural nets)

$g(z, \gamma) : Z \to X$, such that $\tilde{X} = g(Z, \gamma)$ and
$Z \sim p_Z(\cdot) \implies \tilde{X} \sim p_\gamma(\cdot)$.

We can sample directly from p_γ and fine tune its parameters to make it as similar as possible to p_X.

1. We can represent distributions confined to lower dimensional manifolds.
2. The ability to easily generate samples is often more useful than estimating the numerical value of the density at a specific point.

Table of Contents

Generative Adversarial Networks (GoodFellow et al.[6])

GANs jointly optimize the discriminator and generator networks through the saddle point objective:

$$\min_{\gamma} \max_{\phi} L(\gamma, \phi)$$

The objective function is minimized with respect to the generator's parameters, and maximized with respect to the discriminator's. Since the discriminator solves a binary classification problem, it can be written (from the cross-entropy loss), as:

$$L(\gamma, \phi) = \frac{1}{N_R} \sum_{i=1}^{N_R} \ln d(X_i, \phi) + \frac{1}{N_F} \sum_{i=1}^{N_F} \ln \left[1 - d\left(g(Z_i, \gamma), \phi\right)\right]$$

Where $p_Z(\cdot)$ is the researcher-chosen latent distribution, which is mapped by the generator through $p_\gamma(\cdot)$.

Limitations of GANs

Assuming an infinite number of samples, a fixed generator, and substituting for the optimal discriminator $d^*(x) = \frac{p_X(x)}{p_X(x)+p_\gamma(x)}$ we can rewrite the error function C as a function of the generator network p_γ:

$$C(p_\gamma) = -\ln(4) + KL\left(p_X \,\middle\|\, \frac{p_X+p_\gamma}{2}\right) + KL\left(p_\gamma \,\middle\|\, \frac{p_X+p_\gamma}{2}\right)$$

In points in the data space where p_X and p_γ do not overlap the JS is saturated, since its KL components are infinite.

Although GANs do not directly minimize such divergences, in practice this means a discriminator can quickly learn to classify data not in the support of p_X as fake. This leads to vanishing gradients for ϕ, small updates of γ and slow training.

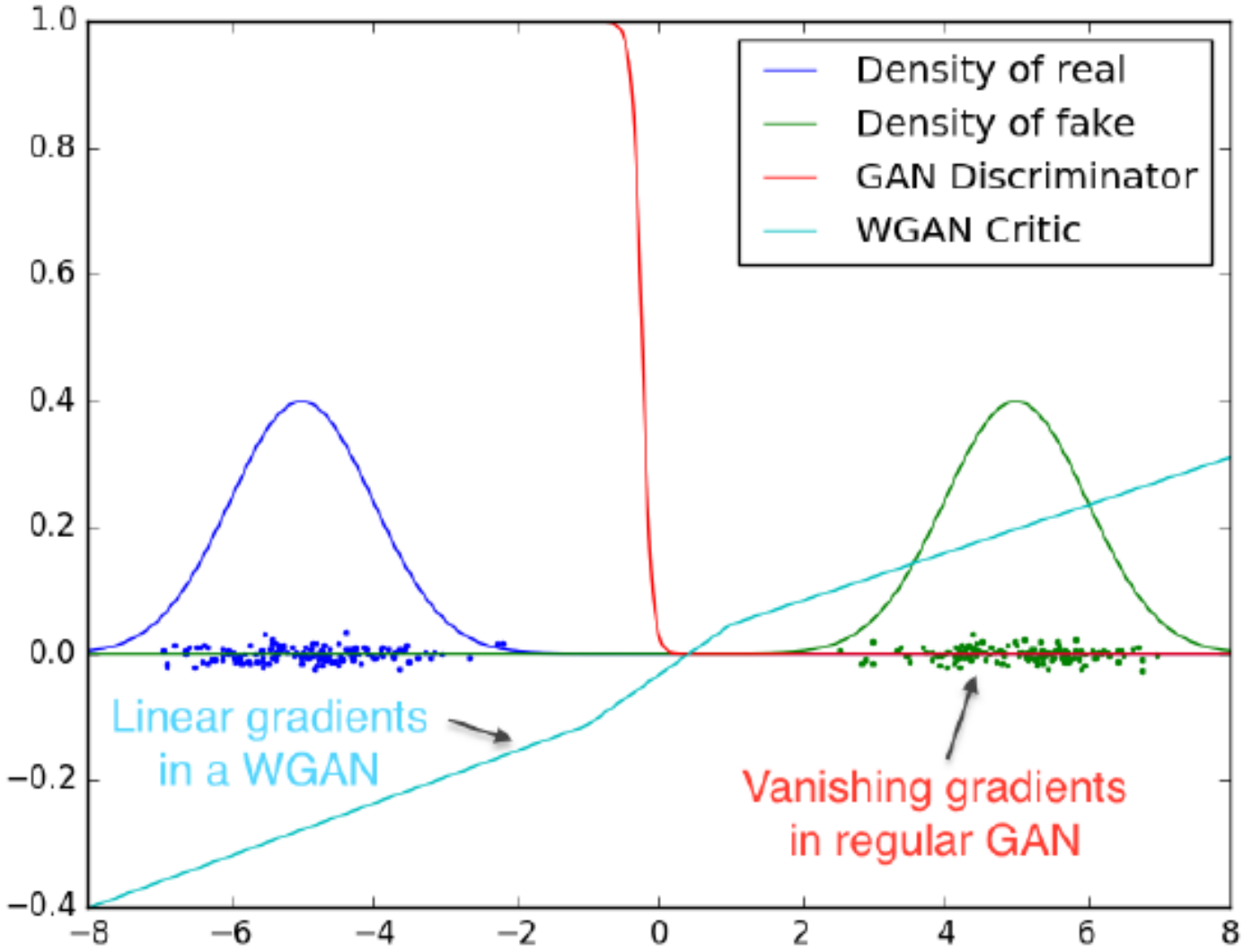
Wasserstein GAN(Arjovsky et al.[2])

Wasserstein GAN modifies the loss to reflect how 'horizontally' distant p_X and p_γ are in the data space. The proposed alternative to the JS Divergence is the Earth-Mover(Wasserstein) distance, which, exploiting the Kantorovich-Rubinstein duality, can be written as:

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

By parametrizing the critic as $f(x, \phi)$, assuming a Lipschitz continuity constraint with constant 1 is imposed on the critic, we obtain the W-GAN optimization problem:

$$\min_{\gamma} \max_{\phi} \left\{ \frac{1}{N_R} \sum_{i=1}^{N_R} f(X_i, \phi) - \frac{1}{N_F} \sum_{i=1}^{N_F} f(g(Z_i, \gamma), \phi) \right\}$$


Density of real
Density of fake
GAN Discriminator
WGAN Critic
Linear gradients
in a WGAN
Vanishing gradients
in regular GAN
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−8
−6
−4
−2
0
2
4
6
8

WGAN-GP(Gulrajani et al.[7])

In practice, Arjovsky et al. ensure the Lipschitz constraint by limiting the gradient $\nabla_x f(X, \phi)$ through weight clipping, biasing the critic towards much simpler functions.

Gulrajani et al. address this by introducing a penalty term on the norm of the gradient of the critic with respect to its input along lines connecting original and fake data points, leading to Gradient Penalty Wasserstein GAN. The following penalization term is added to the loss during training:

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

Where $\hat{X}_i = e_i X_i + (1 - e_i)\tilde{X}_i$ are random convex combinations of real and generated observations for the considered data batch, and e_i are re-sampled at each update of ϕ

Training Differences with GAN

- In W-GAN, the output of the critic is a scalar. It faces a regression problem.
- The generator improves faster with an optimal critic, more than 1 gradient update of ϕ is often required for each gradient update of γ.
- RMSProp for SGD is suggested by Arjovsky et al. Gulrajani et al. show ADAM leads to faster convergence in W-GAN-GP.
- Slower training due to penalized gradients, but more consistent results.
- The loss function has a meaningful interpretation, as it is an approximation of the EM distance. (Can't use it to compare models based on different critics, as the hyperparameter K will change).

Table of Contents

The Lalonde-Dehejia-Wahba-Dataset

	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)	0.3	(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	2.1	(4.89)	2.11	(5.69)	14.02	(9.57)	19.43	(13.41)
earn '75	1.53	(3.22)	1.27	(3.1)	13.65	(9.27)	19.06	(13.6)
earn '78	6.35	(7.87)	4.55	(5.48)	14.85	(9.65)	21.55	(15.56)

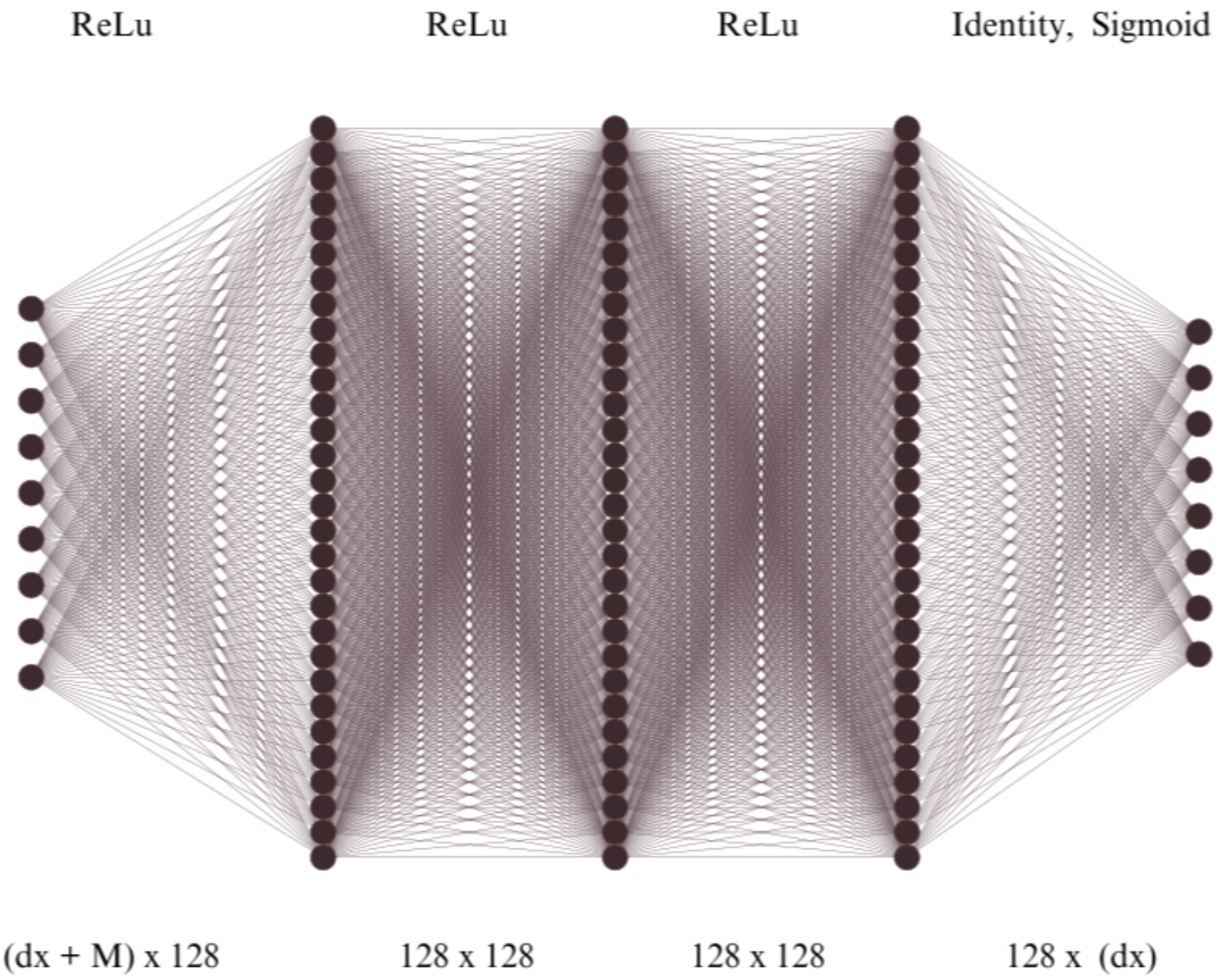
Sampling from Conditional Distributions, C-WGAN

In the causal setting that motivates the paper, we are interested in sampling from the following conditional distributions:

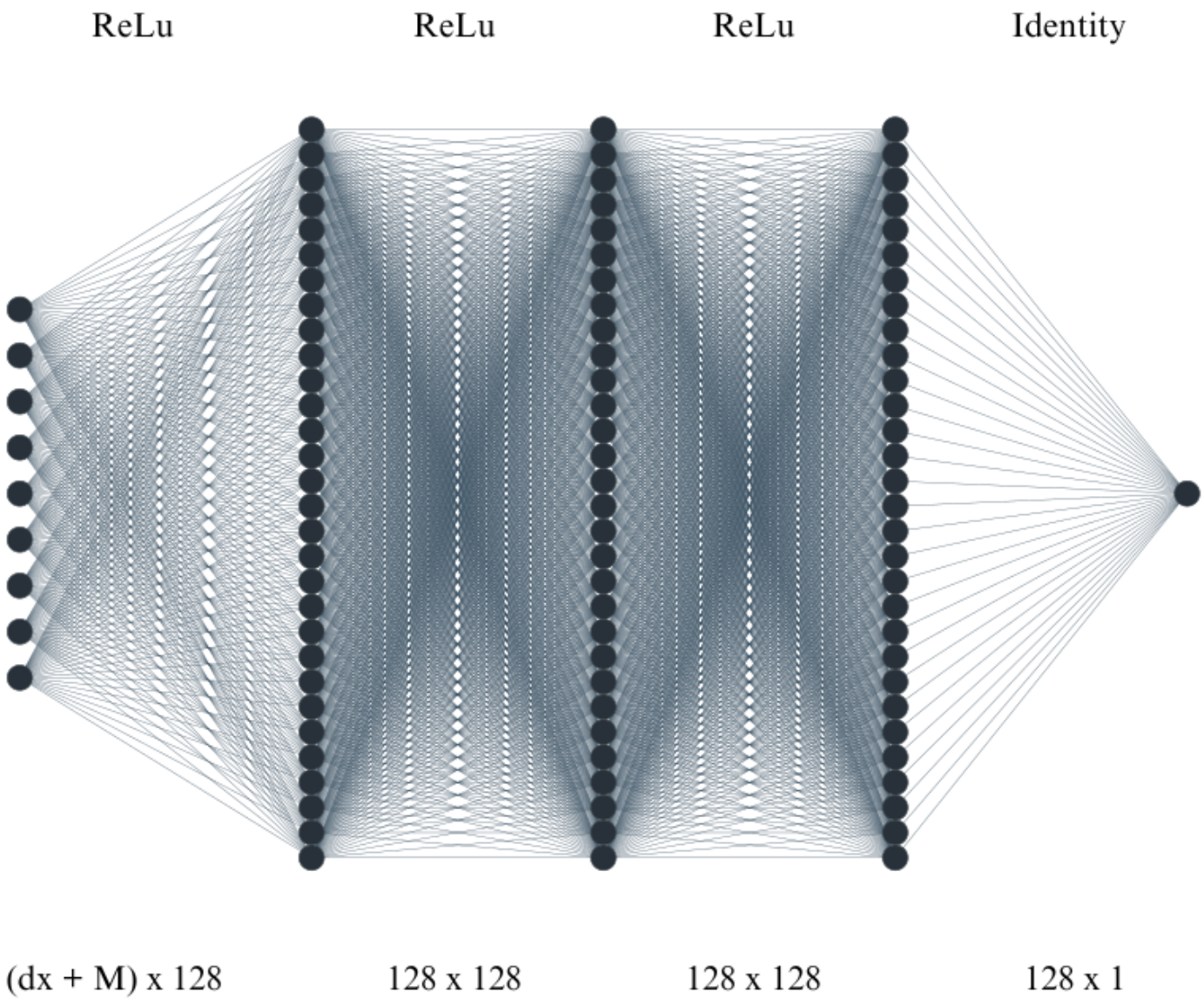
- $P(X_i|T_i)$ the density of covariates X_i conditional on treatment T_i.
- $P(Y_i|X_i, T_i)$ the density of the outcome variable conditional on the covariates and treatment.

This can be achieved under minimal modifications of the W-GAN setup. A separate W-GAN is used for each of the two distributions.

C-WGAN Architecture: Generators $X_i|T_i$ $Y_i|X_i, T_i$



C-WGAN Architecture: Critics $X_i|T_i$ $Y_i|X_i, T_i$



Algorithmic choices

- Dropout for the Generator: avoid that the generator gets too close to the empirical distribution. This compromises slightly the interpretation of the loss function.
- ADAM is used for optimization. (Gulrajani et al.)
- Tests with shallower neural nets led to too high sensitivity of hyperparameters choices.

The Algorithm I

1: **Tuning parameters:**
2: m, batch size
3: $n_{critic} = 15$
4: $lr_0 = 0.0001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ (Adam)
5: $\lambda = 5$, critic's penalty

6: **Starting Values:**
7: $\phi = 0$ (critic), $\gamma = 0$ (generator)

8: **Latent Distribution:**
9: $p_Z(z)$ distributed as $N(0, I)$

The Algorithm II

while (ϕ, γ) have not converged **do**
  for $t = 0$ to n_{critic} **do**
    Sample $\{(X_i, V_i)\}_{i=1}^{m} \sim \mathcal{D}$ a batch from the real data and labels.
    Sample $\{Z_i\}_{i=1}^{m} \sim p_Z(z)$
    $\tilde{X}_i \leftarrow g(Z_i|V_i, \gamma)$ for each of the m observations

    Compute penalty term $Q(\phi)$.
    Generate ϵ_i, $i = 1, \ldots, m$ from $U(0, 1)$
    Calculate $\hat{X}_i = \epsilon_i X_i + (1 - \epsilon_i)\tilde{X}_i$
    $Q(\phi) \leftarrow \frac{1}{m} \sum_{i=1}^{m} \left[\left(\left\| \nabla_{\hat{x}} f(\hat{X}_i|V_i, \phi) \right\|_2 - 1 \right)^2 \right]$

    Compute gradient with respect to ϕ and update
    $\delta_\phi \leftarrow \nabla_\phi \left[\frac{1}{m} \sum_{i=1}^{m} f(X_i|V_i, \phi) - \frac{1}{m} \sum_{i=1}^{m} f(\tilde{X}_i|V_i, \phi) + \lambda Q(\phi) \right]$
    $\phi \leftarrow \text{Adam}(-\delta_\phi, \phi, lr_0, \beta_1, \beta_2, \epsilon)$
  end for

The Algorithm III

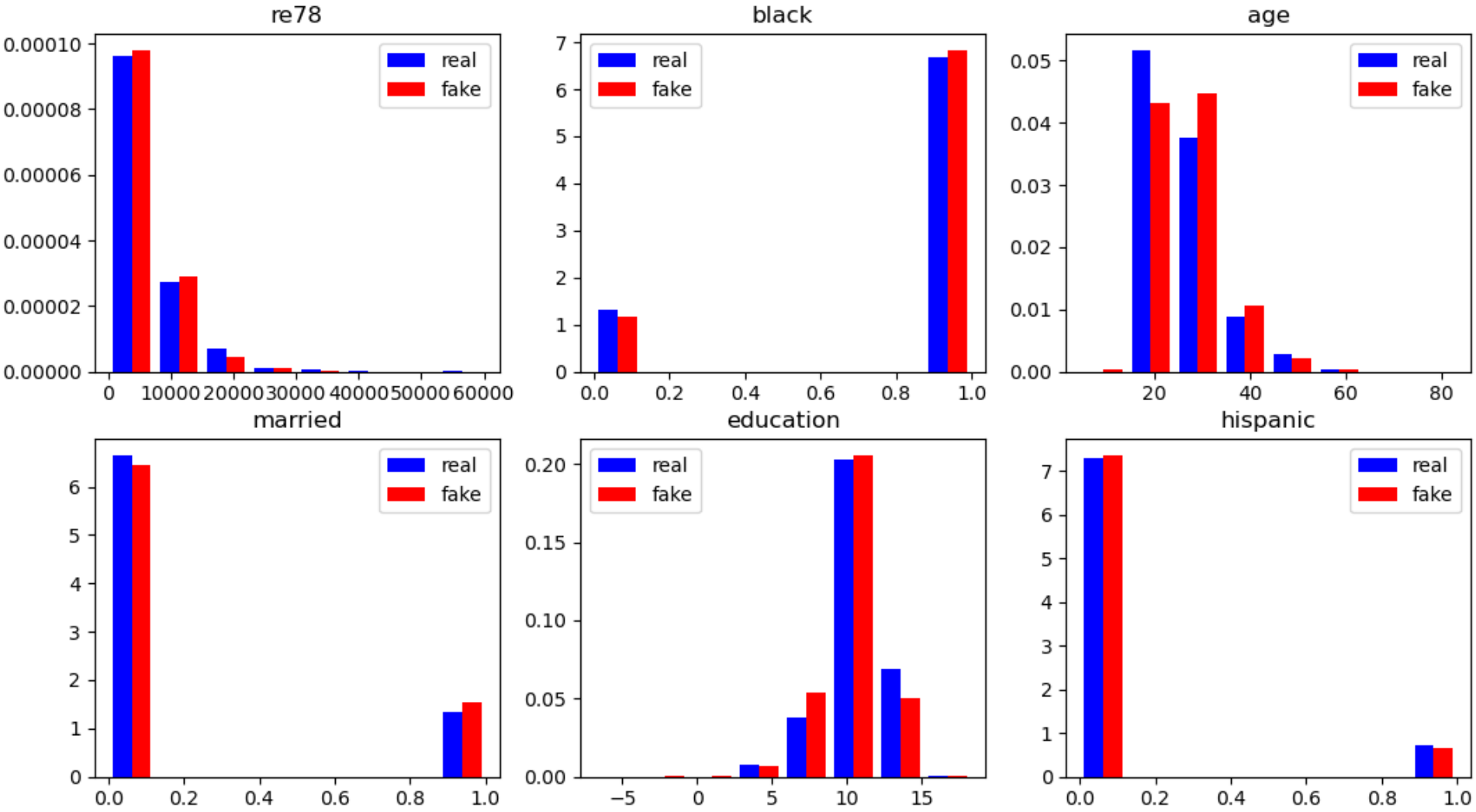
Run a single generator training step.
Sample $\{V_i\}_{i=1}^m \sim \mathcal{D}$ a batch of size m from the real labels.
Sample $\{Z_i\}_{i=1}^m \sim p_Z(z)$

Compute gradients with respect to γ and update
$\delta_\gamma \leftarrow \nabla_\gamma \left[\frac{1}{m}\sum_{i=1}^m f(g(Z_i|V_i,\gamma)|V_i,\phi)\right]$
$\gamma \leftarrow \text{Adam}(\delta_\gamma, \gamma, lr_0, \beta_1, \beta_2, \epsilon)$
end while

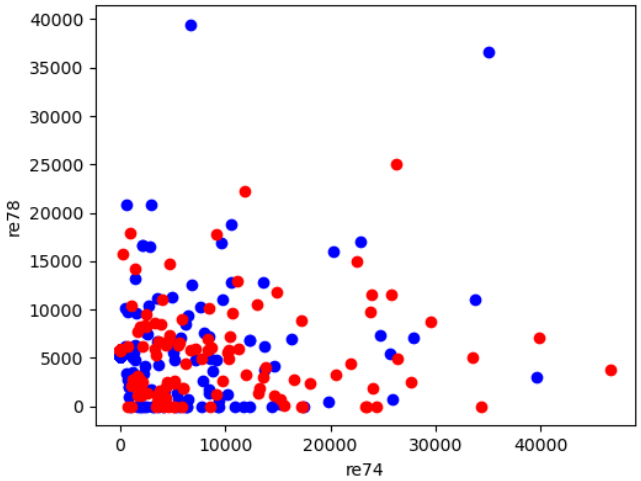
Comparison of Moments (LDW-E)

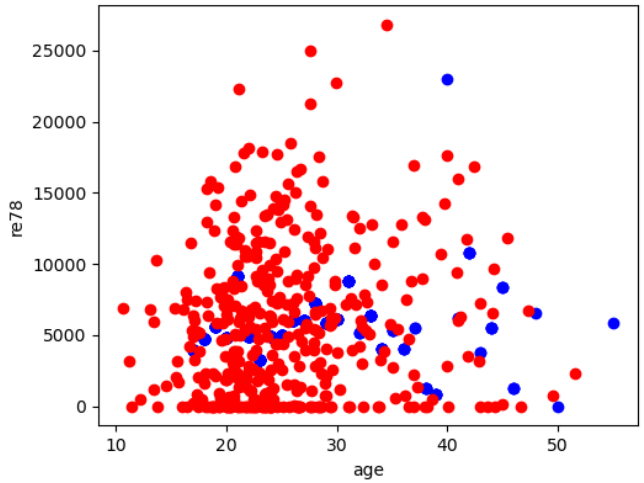
Comparison of Means (LDW-E)				
t	0		1	
Source	Fake	Real	Fake	Real
age	24.18	25.05	24.89	25.82
education	10.03	10.09	10.40	10.35
black	0.82	0.83	0.87	0.84
hispanic	0.10	0.11	0.05	0.06
married	0.15	0.15	0.19	0.19
nodegree	0.85	0.83	0.68	0.71
re74	1238.91	2107.03	1606.40	2095.57
re75	660.05	1266.91	953.69	1532.06
re78	3828.40	4554.80	3627.72	6349.14
Comparison of Standard Deviations (LDW-E)				
age	6.27	7.06	6.16	7.16
education	1.50	1.61	1.86	2.01
black	0.38	0.38	0.34	0.36
hispanic	0.29	0.31	0.23	0.24
married	0.36	0.36	0.39	0.39
nodegree	0.36	0.37	0.47	0.46
re74	3804.63	5687.91	4091.16	4886.62
re75	2031.49	3102.98	2366.83	3219.25
re78	4425.63	5483.84	6475.03	7867.40

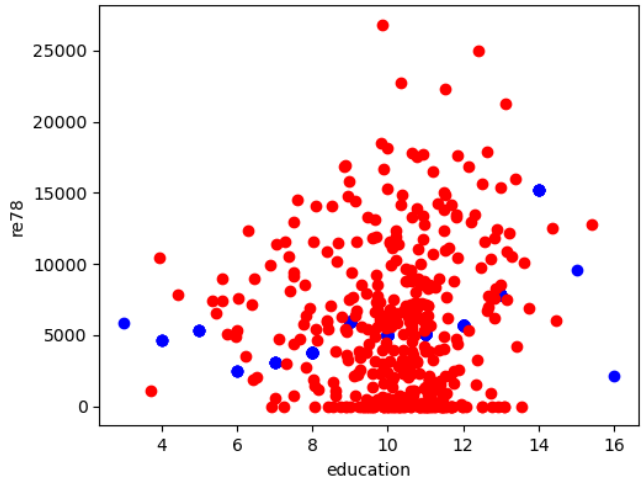
Marginal Distributions (LDW-E)

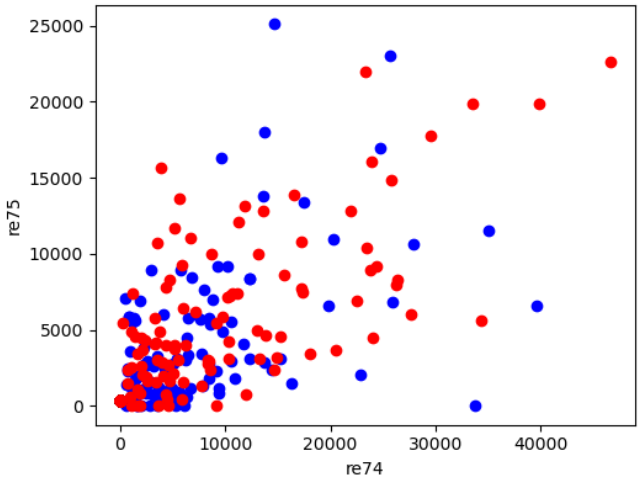


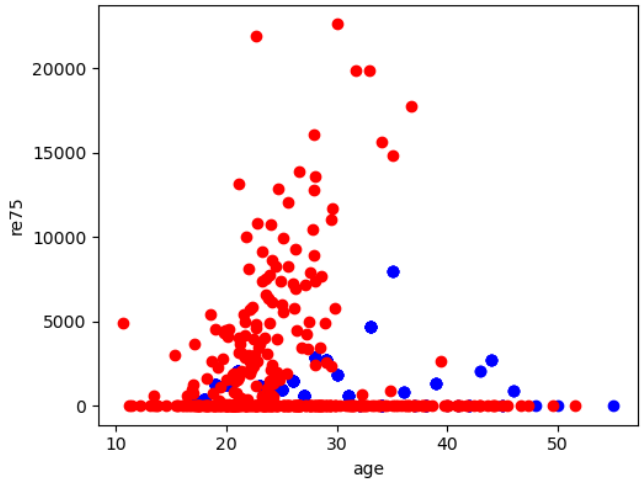
Correlations (LDW-E)

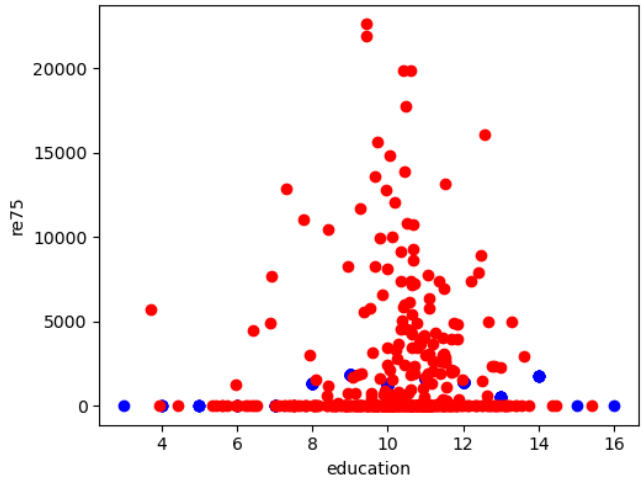




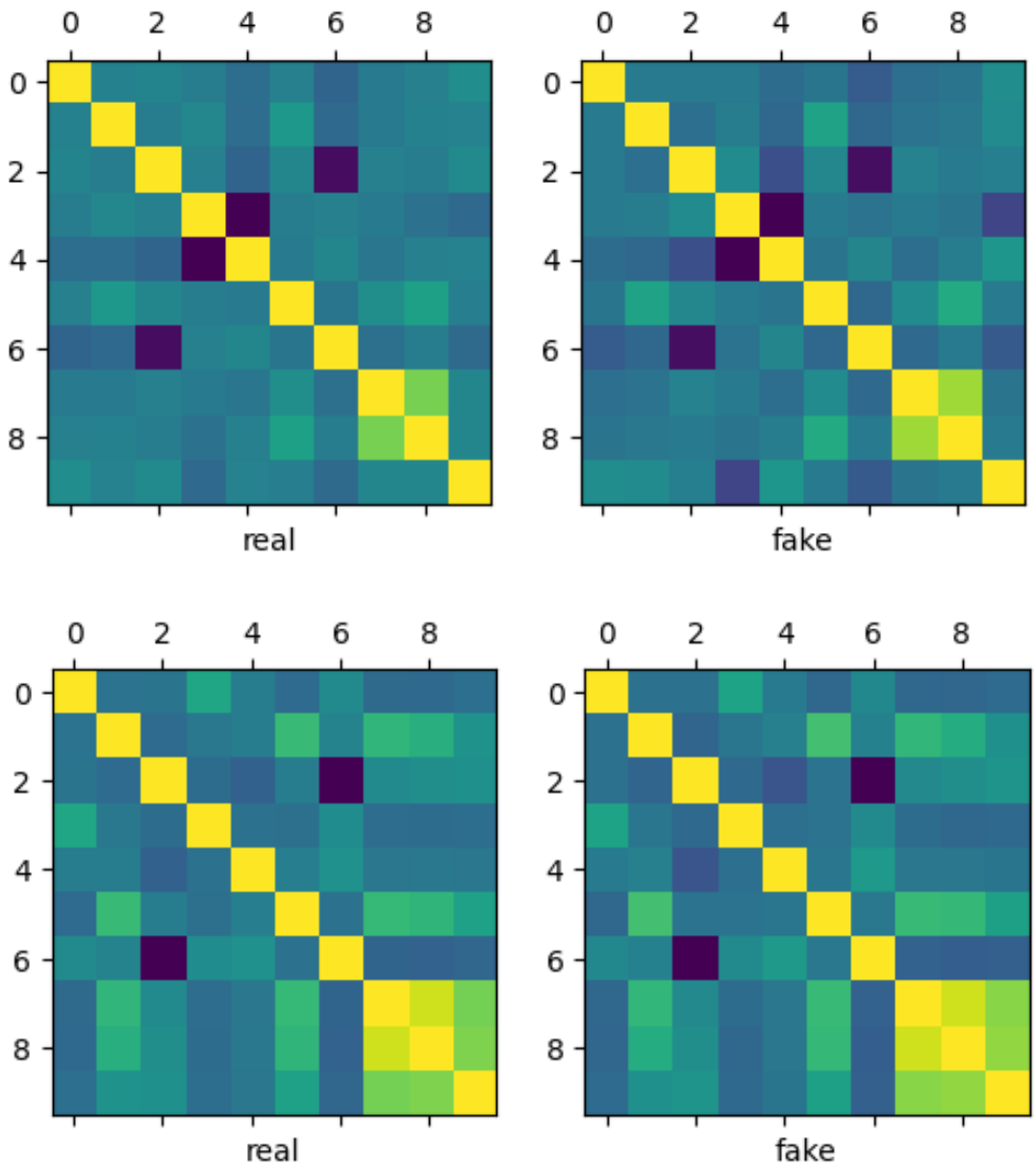




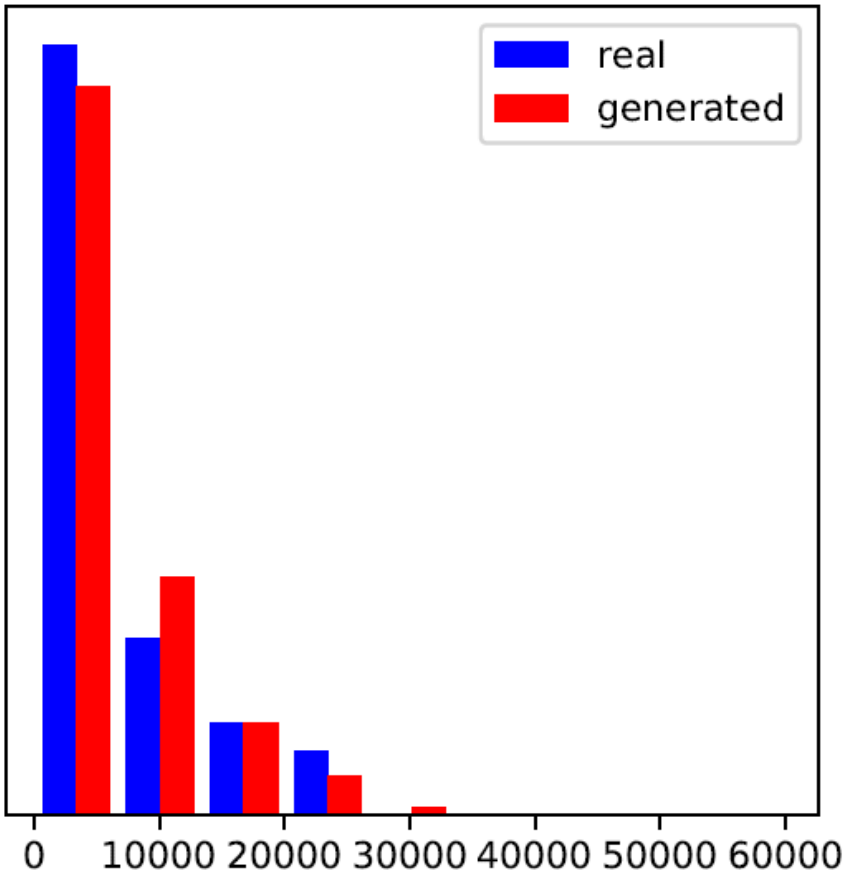




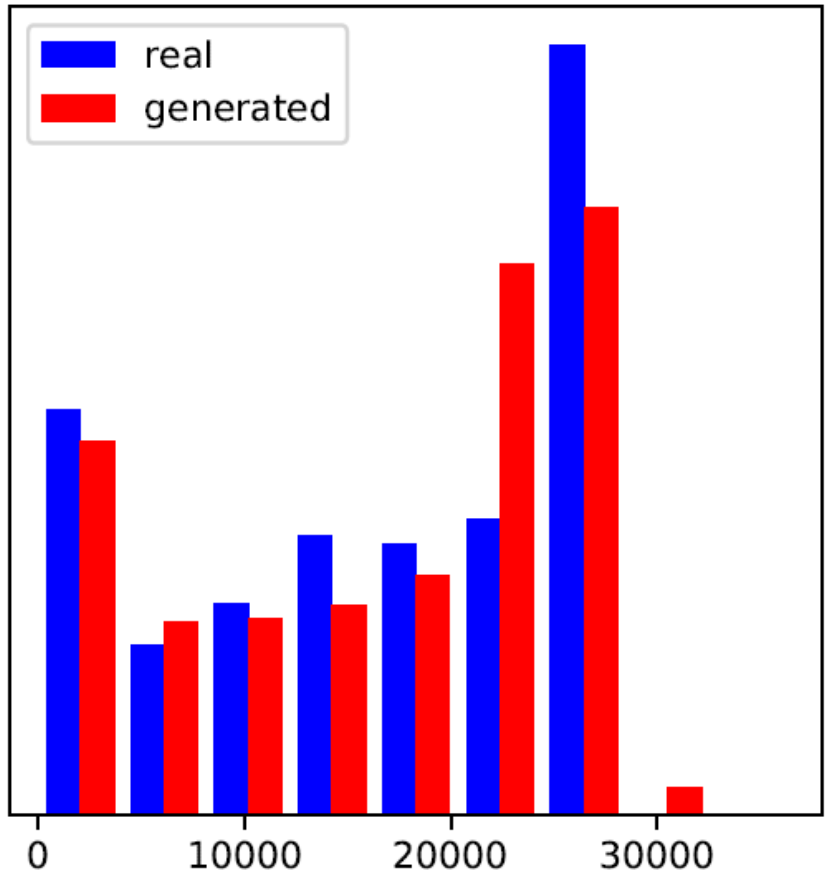
Variance - Covariance (LDW-E / LDW-CPS)



Conditional Distributions (LDW-CPS)



Earnings 1978 | Earnings 1974=0



Earnings 1978 | Earnings 1974>0

WD and out-of-sample R^2

- Wasserstein Distance between real and generated data is calculated using linear programming (Cuturi, 2013[5]), comparing the generated model and a model fitted with a multivariate normal.

Dataset	GAN	Multivariate Normal	Ratio
Experimental controls	1606	4972	0.32
CPS controls	1629	5005	0.33
PSID controls	3152	5586	0.56

- Out of sample fit of three models fitting a linear function for the conditional expectation of the outcome variable is very similar between real and generated data.

Table of Contents

Standard setting for Simulation studies for ATT

We are interested in comparing estimators for ATT:

$$\tau = \mathbb{E}[Y_i(1) - Y_i(0) \mid T_i = 1]$$

assuming unconfoundedness and overlap:

$$T_i \perp (Y_i(1), Y_i(0)) \mid X_i$$

$$0 < \Pr(T_i = 1 \mid X_i = x) < 1 \quad \text{, for each } x$$

Baseline Estimation of ATT

- Having trained a generator for $P(Y_i|X_i, T_i)$, we can generate both potential outcomes for each observation. This allows for a baseline estimation of ATT in the generated samples as:

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

.

- First, a large (10^6) sample of covariates conditional on treatment is generated: $\tilde{X}_i = g(Z_i|T_i, \gamma)$, keeping the fraction of treated units equal to that in the true sample.
- Then $Y(0), Y(1)$ are drawn independently for each observation, conditioning on the generated covariates and exchanging the treatment status for units in the treatment group:

$$\tilde{Y}_i(0) = g(Z_i|\tilde{X}_i, T_i = 1; \gamma)$$
$$\tilde{Y}_i(1) = g(Z_i|\tilde{X}_i, T_i = 1; \gamma)$$

Estimators

The propensity score $e(x) = Pr(T_i = 1 \mid X_i = x)$ and the conditional outcome mean $\mu(t, x) = \mathbb{E}[Y_i \mid T_i = t, X_i = x]$ are first estimated with three methods; Linear Models, Random Forests and Neural Networks.

Then, they are combined in the computation of ATT with three proposed estimators:

$$\hat{\tau}^{cm} = \frac{1}{N_1} \sum_{i:T_i=1} \left(Y_i - \hat{\mu}(0, X_i)\right)$$

$$\hat{\tau}^{ht} = \sum_i \left(\frac{T_i}{N_1} Y_i - \frac{(1-T_i) Y_i \hat{e}(X_i)}{1-\hat{e}(X_i)} \Big/ \sum_{j=1}^{N} \frac{(1-T_j)\hat{e}(X_j)}{1-\hat{e}(X_j)} \right)$$

$$\hat{\tau}^{dr} = \sum_i \left(\frac{T_i}{N_1} \left(Y_i - \hat{\mu}(0; X_i)\right) - \frac{(1-T_i)(Y_i - \hat{\mu}(0, X_i))\hat{e}(X_i)}{1-\hat{e}(X_i)} \Big/ \sum_{j=1}^{N} \frac{(1-T_j)\hat{e}(X_j)}{1-\hat{e}(X_j)} \right)$$

Results LDW-E (2000 samples)

method	rmse	bias	sdev	coverage
Baselines				
DIFF	0.49	0.06	0.48	0.94
BCM	0.58	0.00	0.58	0.96
Outcome Models				
L	0.52	-0.06	0.51	0.88
RF	0.51	-0.07	0.50	0.88
NN	1.32	0.04	1.32	0.75
Propensity Score Models				
L	0.52	-0.08	0.52	0.99
RF	0.52	-0.06	0.51	0.99
NN	0.52	0.01	0.52	0.99
Doubly Robust Methods				
L	0.51	-0.08	0.51	0.95
RF	0.52	-0.04	0.52	0.95
NN	0.79	-0.05	0.79	0.95
CF	0.50	-0.09	0.49	0.94
RB	0.52	-0.09	0.51	0.95

- Similar RMSEs. No particular ranking emerges.
- All methods perform fairly well, as the number of treated and control units in the sample is balanced.

Results LDW-CPS (2000 samples)

method	rmse	bias	sdev	coverage
Baselines				
DIFF	11.12	-11.11	0.45	0.00
BCM	0.73	0.07	0.73	0.96
Outcome Models				
L	2.14	-2.08	0.51	0.02
RF	1.00	-0.87	0.51	0.54
NN	0.63	0.14	0.61	0.88
Propensity Score Models				
L	0.51	0.00	0.51	0.98
RF	1.00	-0.87	0.50	0.73
NN	0.65	0.23	0.61	0.94
Doubly Robust Methods				
L	0.53	0.03	0.53	0.96
RF	0.54	-0.05	0.54	0.93
NN	0.62	0.20	0.58	0.94
CF	0.55	0.11	0.53	0.91
RB	0.57	-0.22	0.52	0.89

- Double-robust estimators perform better
- Bias quite large for Conditional Outcome Models, leading to low coverage rates

Robustness Checks

The ranking of estimators implied by the paper might not be robust to alternative specifications.

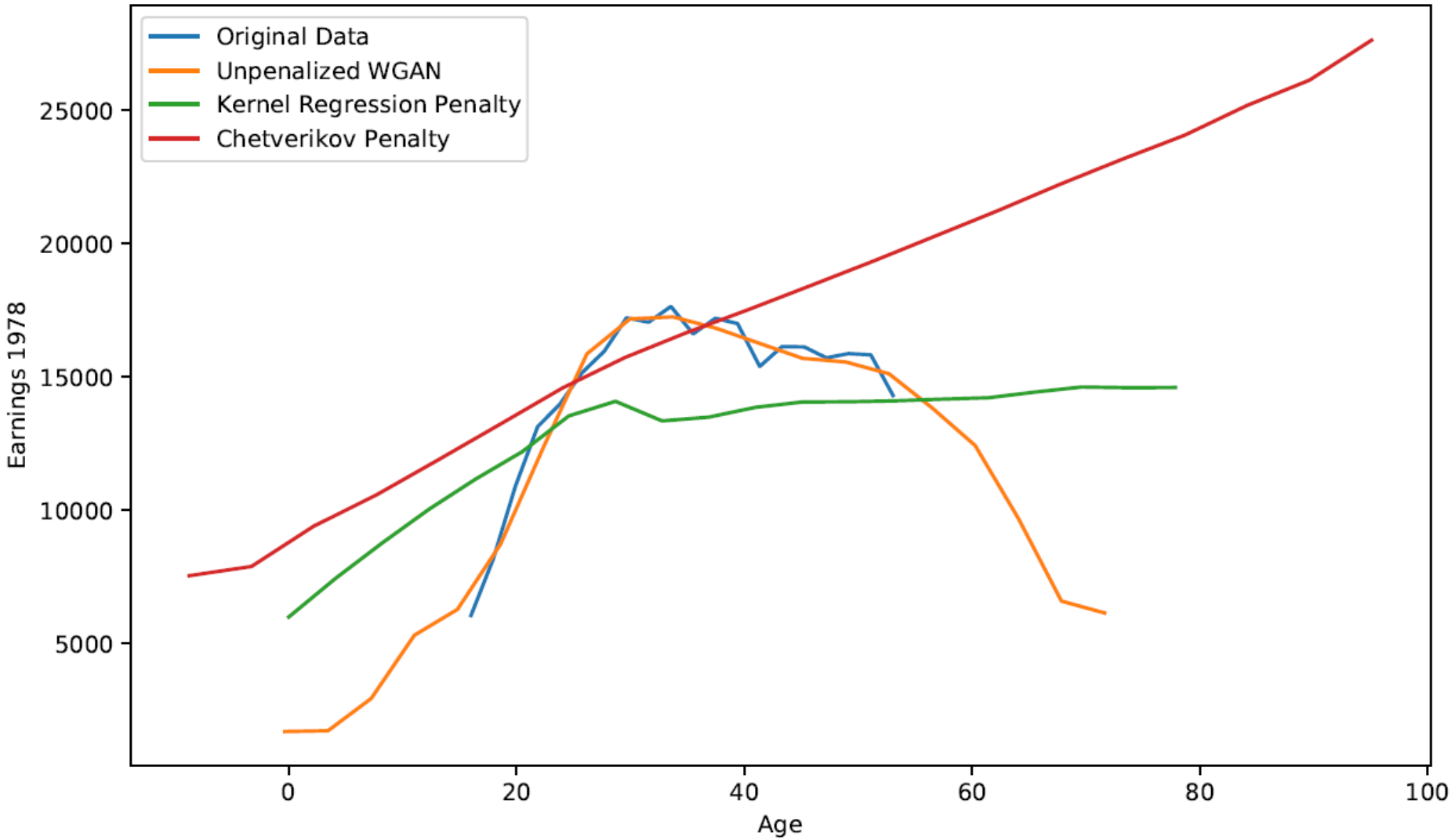
1. Robustness to sub-samples of the data: 10 samples of 80% size of original are used; the ranking is consistent for each of them.
2. Robustness to Architecture: The authors attempt using a different structure for the neural nets, shifting the number of hidden units in each layer from 80 to 256.
3. Robustness to Size of training data. No significant differences even up to 0.2 of original training sample.

Table of Contents

Imposing Restrictions

- We may wish to generate data from a restricted set of distributions. For instance, $\mu(x) = E[Y_i|X_i = x]$ may be monotonically increasing in x. (Demand Functions).
- This can be implemented by penalizing the objective function in the direction which increases the value of a test statistic $T(g(Z_1, \gamma), \ldots, g(Z_M, \gamma))$, with null hypothesis that the regression function between X and Y is non-decreasing. (add $\lambda T(\gamma)$ to the objective).
- The test statistic the authors chose was proposed by Chetverikov (2018)[4]. They focus on the case where Y_i is earnings in 1978 and X_i is age. The regression function is far from monotone in the true sample: imposing monotonicity changes the joint distribution of the two variables. The relationship is described through Kernel Regression.

Penalized Kernel Regression, LDW-CPS



Penalized Kernel Regression, LDW-E

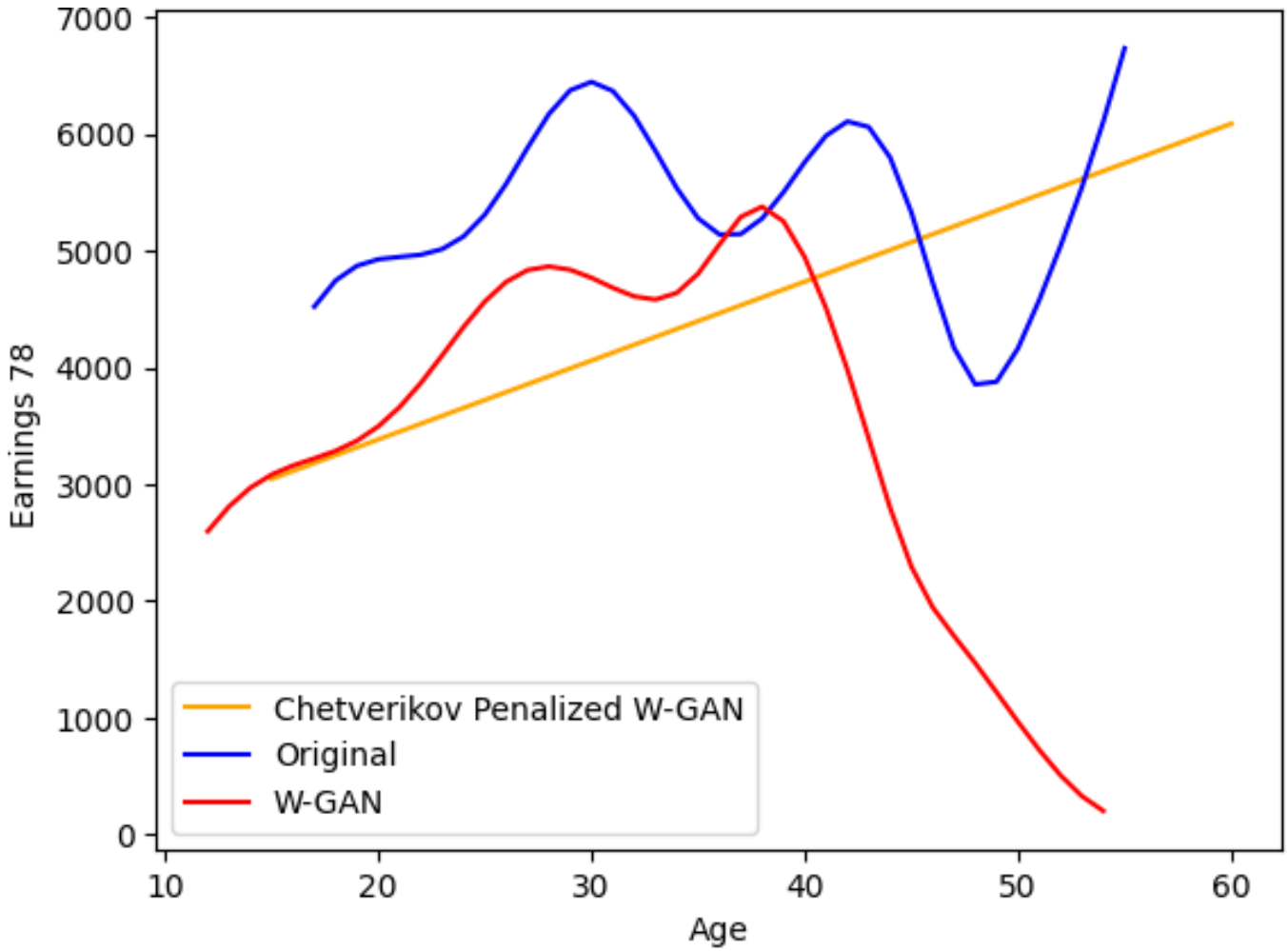
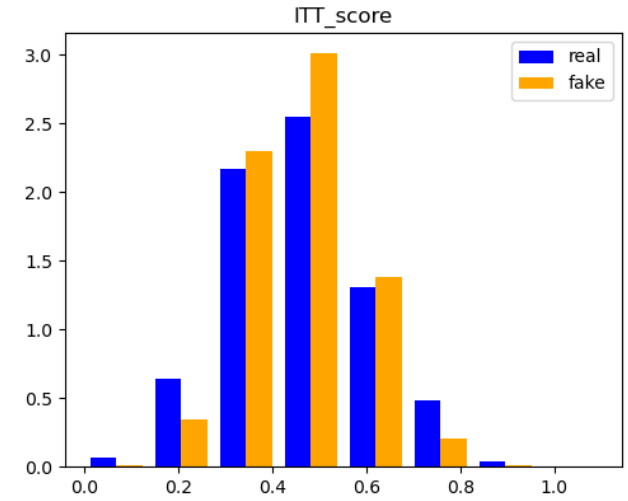


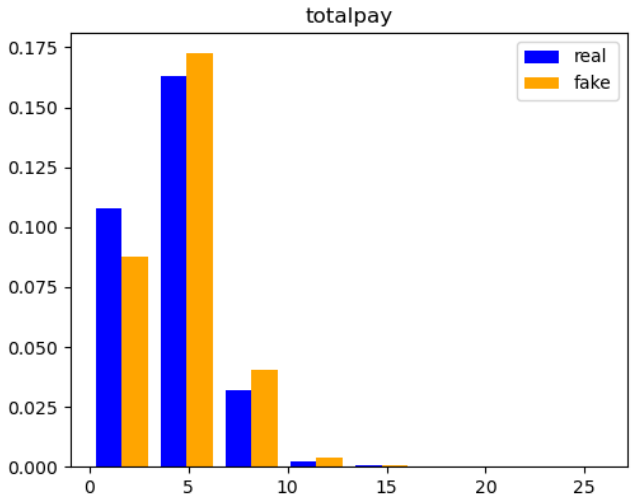
Table of Contents

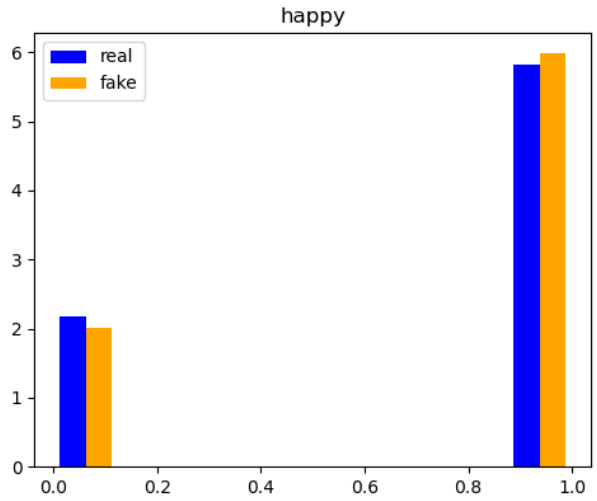
RCT(De Ree et al.[8]) - Cross Section

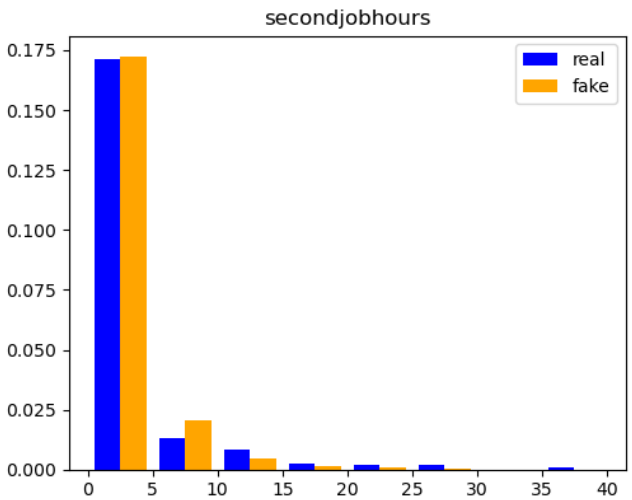
Comparison of Means (D-RCT)				
t	0		1	
Source	Fake	Real	Fake	Real
certified	0.53	0.53	0.72	0.72
secondjob	0.26	0.26	0.21	0.22
secondjobhours	1.52	2.23	1.10	1.52
basepay	2.37	2.34	2.41	2.35
additionalpay	0.85	0.82	0.95	0.90
totalpay	4.18	4.03	4.86	4.68
problems	0.38	0.43	0.27	0.33
happy	0.71	0.68	0.82	0.81
absent	0.15	0.13	0.09	0.12
assets	0.41	0.33	0.38	0.34
ITTscore	0.46	0.46	0.46	0.46
Comparison of Standard Deviations (D-RCT)				
certified	0.50	0.50	0.45	0.45
secondjob	0.44	0.44	0.41	0.42
secondjobhours	3.58	5.56	3.09	4.22
basepay	0.98	0.93	0.85	0.85
additionalpay	0.90	0.83	1.07	0.94
totalpay	2.25	2.14	2.30	2.23
problems	0.49	0.50	0.45	0.47
happy	0.46	0.47	0.38	0.39
absent	0.36	0.34	0.29	0.32
assets	0.26	0.24	0.25	0.25
ITTscore	0.12	0.15	0.11	0.15

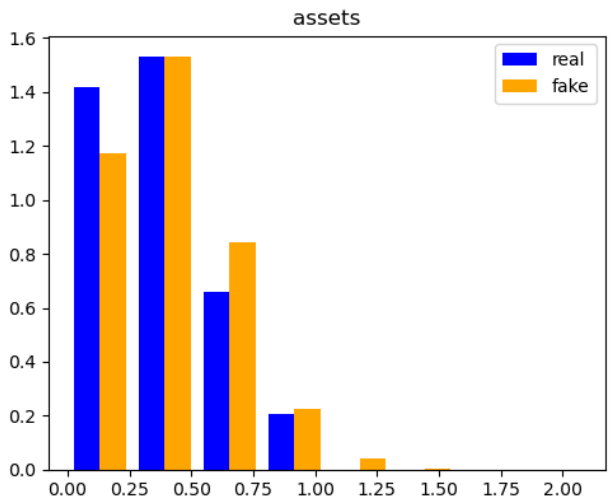
Marginals

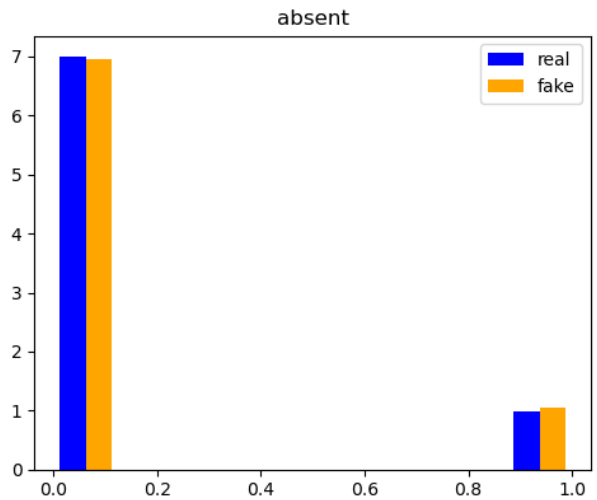

ITT_score
real
fake
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0.0
0.2
0.4
0.6
0.8
1.0


totalpay
real
fake
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0
5
10
15
20
25

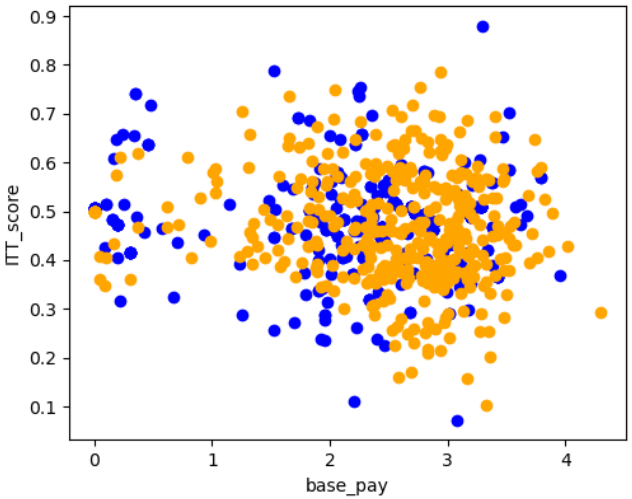

happy
real
fake
0
1
2
3
4
5
6
0.0
0.2
0.4
0.6
0.8
1.0

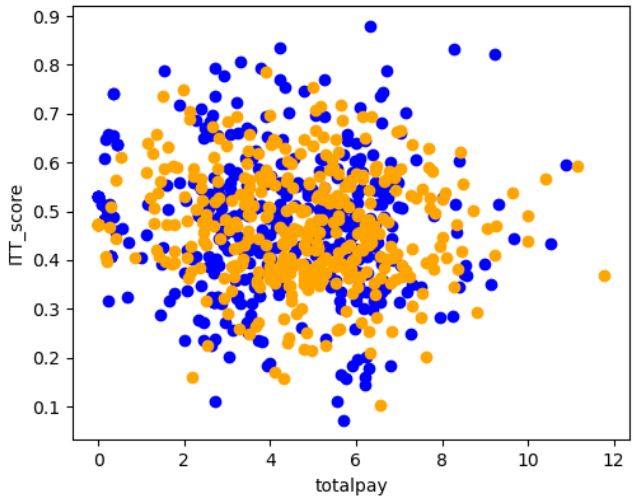

secondjobhours
real
fake
0.000
0.025
0.050
0.075
0.100
0.125
0.150
0.175
0
5
10
15
20
25
30
35
40

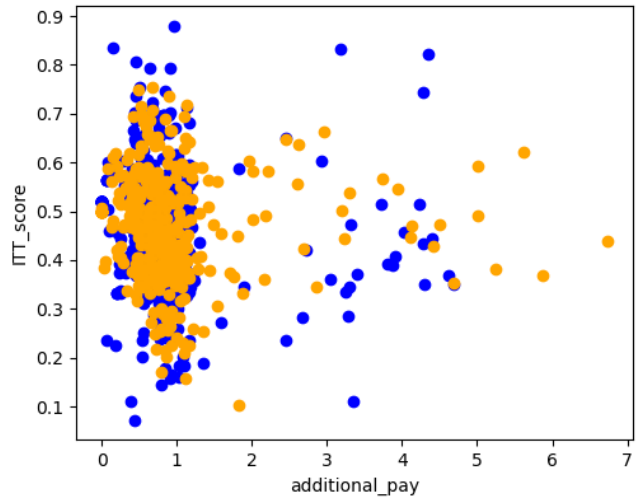

assets
real
fake
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

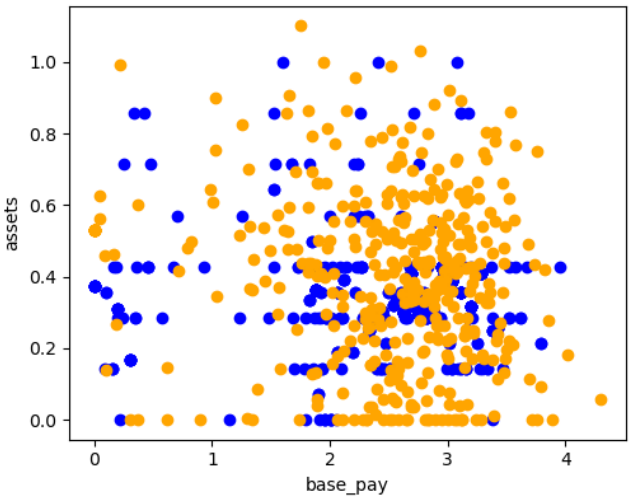

absent
real
fake
0
1
2
3
4
5
6
7
0.0
0.2
0.4
0.6
0.8
1.0

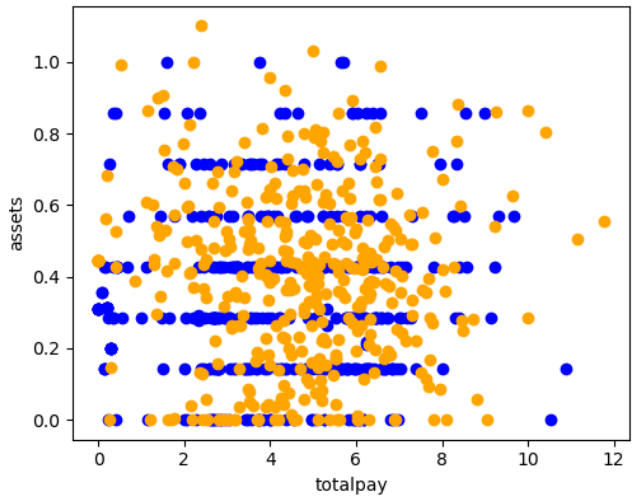
Correlations

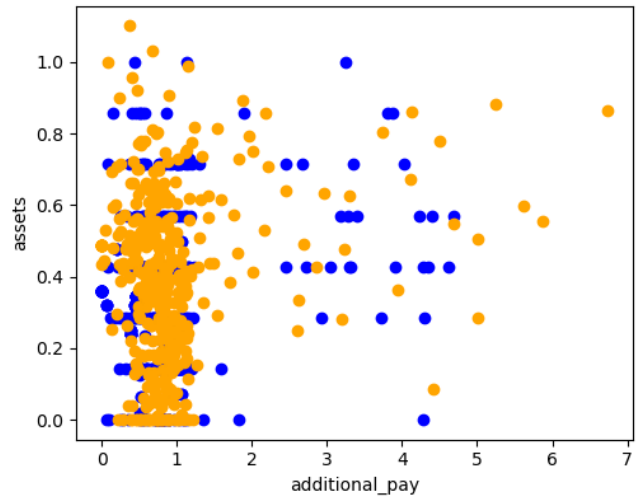




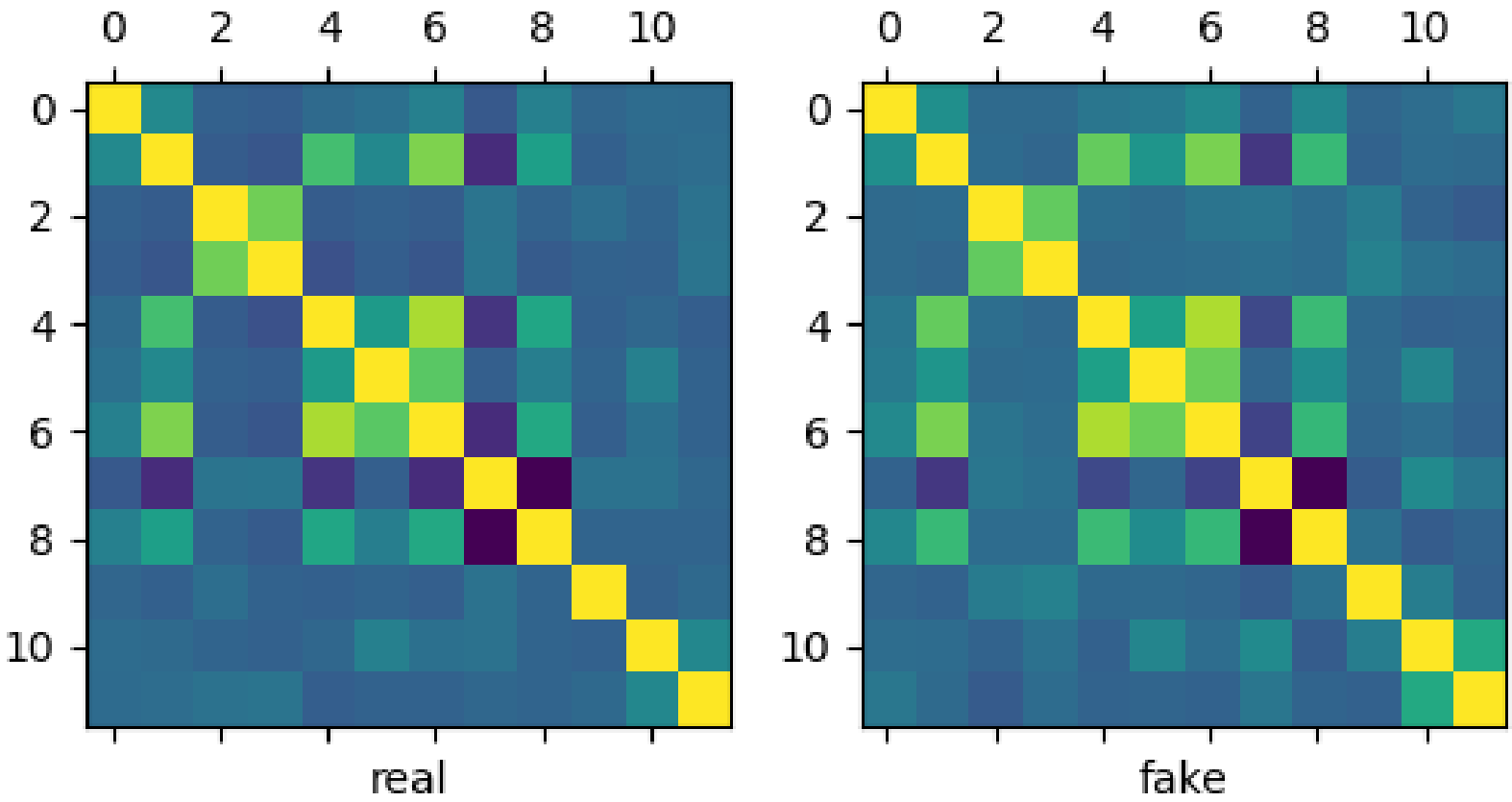








Variance - Covariance Matrix



Bibliography I

[1] Arun Advani, Toru Kitagawa, and Tymon Słoczyński. "Mostly harmless simulations? Using Monte Carlo studies for estimator selection". In: *Journal of Applied Econometrics* 34.6 (Sept. 2019), pp. 893–910.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. *Wasserstein GAN*. 2017. arXiv: 1701.07875 [stat.ML].

[3] Susan Athey et al. *Using Wasserstein Generative Adversarial Networks for the Design of Monte Carlo Simulations*. 2020. arXiv: 1909.02210 [econ.EM].

[4] Denis Chetverikov. "Testing Regression Monotonicty in Econometric Models". In: *Econometric Theory* 35.4 (Sept. 2018), pp. 729–776. ISSN: 1469-4360.

Bibliography II

[5] Marco Cuturi. *Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances*. 2013. arXiv: 1306.0895 [stat.ML].

[6] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[7] Ishaan Gulrajani et al. *Improved Training of Wasserstein GANs*. 2017. arXiv: 1704.00028 [cs.LG].

[8] Joppe de Ree et al. "Double for Nothing? Experimental Evidence on an Unconditional Teacher Salary Increase in Indonesia*". In: *The Quarterly Journal of Economics* 133.2 (Nov. 2017), pp. 993–1039. ISSN: 0033-5533. DOI: 10.1093/qje/qjx040.