# Estimation with GANs

Master's Thesis

Presented to the
Department of Economics at the
Rheinische Friedrich-Wilhelms-Universität Bonn

In Partial Fulfillment of the Requirements for the Degree of
Master of Science (M.Sc.)

Supervisor: Prof. Dr. Joachim Freyberger

Submitted in September 2024 by

Marvin Benedikt Riemer

Matriculation Number: 2799234

# Contents

# List of Figures

# List of Tables

# 1 Introduction

Welcome to my thesis! It is based on the paper Kaji, Manresa, and Pouliot (2023).

# 2 Method

Consider the problem of estimating the parameters of a structural economic model. For $k \in \{1, \ldots, K\}$, let

$$Y_k = f_\Theta(X_k, Z_k;), \tag{1}$$

where $Y_k$ is a vector of outcome variables influenced by a vector of noise variables $Z_k$ via a set of (possibly endogeneous) variables $X_k$. The strength and functional form of the relationships between the variables is defined by a function $f$ and its parameters $\Theta$.

**Algorithm 1** Adversarial estimation

```
Initialize
for number of iterations do
    Update the discriminator
    Update the generator
end for
```

The GAN was first proposed by I. J. Goodfellow et al. (2014) (later published as I. Goodfellow et al. (2020)).

# 3 Simulation

Now I undertake a simulation study in Python, utilizing the library `pytorch` (Ansel et al. (2024)).

First, I reproduce parts of the author's simulation in the scientific Python stack, more precisely, using the packages numpy, scipy, and scikit-learn (Harris et al. (2020), Virtanen et al. (2020), and Pedregosa et al. (2011), respectively). My code is available

The replication package can be downloaded from the journal website. It contains the author's simulation code, written in Matlab. As the authors state in the readme file, the simulations for the Roy model are contained in the files `main_roy.m` (Figures 6, 7) and `main_case.m` (Figures 8, 9, and Table I). They draw on functions in other files to simulate data and calculate losses.

Both main files share a general structure: After setting parameters of the simulation itself (e.g. sample sizes, number of simulation runs) and the Roy model, the values of loss functions

are calcuted along a grid and then rendered to created Figures 6 and 8. Thereafter, real and fake observations are generated and the estimation is performed on them

The authors' code for the neural network discriminator is in `NND.m`. It uses Matlab's `patternnet` and `train`. The scientific Python stack comes with limited support for neural networks, but I can sufficiently approximate the author's discriminator using `sklearn.neural_network.MLPClassifier`.

Following the authors, I create a net with 1 hidden layer containing 10 nodes, followed by the tanh activation function. Inspecting sklearn's source code reveals that a logistic output activation function is automatically set. Because the conjugate-gradient descent algorithm is not available to train `MLPClassifier`, I use the Adam algorithm (Diederik (2014)). It is popular for training neural networks and achieves comparable results in my case.

`MLPClassifier`'s default convergence criteria cause my code to raise warnings about non-convergence of the discriminator nets. This is not completely mitigated even by setting `max_iter` (the maximum number of iterations of the optimizer) to 2000 (10 times the default value), at the cost of a longer runtime. Nevertheless, the networks converge well enough under the default settings. Leaving `max_iter` at 200, but increasing `tol`, the tolerance of the convergence criterium, five- or tenfold mitigates the warnings but results in flatter and less smooth loss functions.

The authors also set the normalization and regularization parameters of `patternnet`. Since these are handled differently in `MLPClassifier`, I do not translate this adaption.

My simulations show that these modifications do not significantly alter the shape of the loss curves.

For the outer optimization loop that trains the generator, the authors use the custom `fminsearchcon` function (D'Errico (2024)). This is a wrapper that adds support for bounds and nonlinear constraints to Matlab's built-in `fminsearch`, which employs the Nelder-Mead simplex algorithm (Lagarias et al. (1998)) to minimize a function without computing gradients. I employ `scipy.optimize.minimize`, which natively supports the Nelder-Mead algorithm with bounds and nonlinear constraints. I set an option to perform a version of the Nelder-Mead algorithm that's adapted to higher-dimensional problems, which shows improved convergence in my simulation.

I employ the `mp` module from Python's standard library to parallelize simulation runs on an HPC cluster (cf. A).

# 4 Conclusion

This section concludes.

# Appendix A Acknowledgement of system use

The author gratefully acknowledges the granted access to the Marvin cluster hosted by the University of Bonn.

# References


**Ansel, Jason, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, et al.** 2024. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation." In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24).* ACM. https://doi.org/10.1145/3620665.3640366. [1]

**D'Errico, John.** 2024. *fminsearchbnd, fminsearchcon.* https://www.mathworks.com/matlabcentral/fileexchange/8277-fminsearchbnd-fminsearchcon. MATLAB Central File Exchange. Accessed September 12, 2024. [2]

**Diederik, P Kingma.** 2014. "Adam: A method for stochastic optimization." *(No Title).* [2]

**Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.** 2020. "Generative adversarial networks." *Communications of the ACM* 63 (11): 139–44. [1]

**Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.** 2014. *Generative Adversarial Networks.* eprint: arXiv:1406.2661. [1]

**Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al.** 2020. "Array programming with NumPy." *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2. [1]

**Kaji, Tetsuya, Elena Manresa, and Guillaume Pouliot.** 2023. "An adversarial approach to structural estimation." *Econometrica* 91 (6): 2041–63. [1]

**Lagarias, Jeffrey C, James A Reeds, Margaret H Wright, and Paul E Wright.** 1998. "Convergence properties of the Nelder–Mead simplex method in low dimensions." *SIAM Journal on optimization* 9 (1): 112–47. [2]

**Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al.** 2011. "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30. [1]

**Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al.** 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17: 261–72. https://doi.org/10.1038/s41592-019-0686-2. [1]

# Selbstständigkeitserklärung

Ich versichere hiermit, dass ich die vorstehende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass die vorgelegte Arbeit noch an keiner anderen Hochschule zur Prüfung vorgelegt wurde und dass sie weder ganz noch in Teilen bereits veröffentlicht wurde. Wörtliche Zitate und Stellen, die anderen Werken dem Sinn nach entnommen sind, habe ich in jedem einzelnen Fall kenntlich gemacht.

12. September 2024

Marvin Benedikt Riemer