Journal of Machine Learning Research 23 (2022) 1-6 Submitted 7/21; Revised 12/21; Published 1/22

DoubleML – An Object-Oriented Implementation of Double Machine Learning in Python

Philipp Bach[†] PHILIPP.BACH@UNI-HAMBURG.DE
Victor Chernozhukov[‡] VCHERN@MIT.EDU
Malte S. Kurz[†*] MALTE.SIMON.KURZ@UNI-HAMBURG.DE
Martin Spindler[†] MARTIN.SPINDLER@UNI-HAMBURG.DE

[†]*Faculty of Business Administration, University of Hamburg, Moorweidenstraße 18, 20148 Hamburg, Germany*
[‡]*Department of Economics and Center for Statistics and Data Science, Massachussets Institute of Technology, 50 Memorial Drive, Cambridge, MA 02142, USA*

Editor: Joaquin Vanschoren

Abstract

`DoubleML` is an open-source Python library implementing the double machine learning framework of Chernozhukov et al. (2018) for a variety of causal models. It contains functionalities for valid statistical inference on causal parameters when the estimation of nuisance parameters is based on machine learning methods. The object-oriented implementation of `DoubleML` provides a high flexibility in terms of model specifications and makes it easily extendable. The package is distributed under the MIT license and relies on core libraries from the scientific Python ecosystem: `scikit-learn`, `numpy`, `pandas`, `scipy`, `statsmodels` and `joblib`. Source code, documentation and an extensive user guide can be found at `https://github.com/DoubleML/doubleml-for-py` and `https://docs.doubleml.org`.

Keywords: machine learning, causal inference, causal machine learning, Python

1. Introduction

The double/debiased machine learning (DML) framework (Chernozhukov et al., 2018) provides methods for valid statistical inference in structural equation models while exploiting the excellent prediction quality of machine learning (ML) methods for potentially high-dimensional and non-linear nuisance functions. The Python package `DoubleML` implements DML for partially linear and interactive regression models and is primarily based on the machine learning package `scikit-learn` (Pedregosa et al., 2011). A key element of the theoretical DML framework are so-called Neyman orthogonal score functions which also form the central part of the object-oriented implementation of `DoubleML`. The object-oriented structure also makes the package highly flexible with regards to model specifications and facilitates easy extensions.

The following sections describe the key ingredients of DML, give an introduction to the architecture of `DoubleML`, list quality standards under which the project is developed, compare the package to related software and provide some outlook to future extensions.

∗. Corresponding author.

2. Key Ingredients of Double Machine Learning

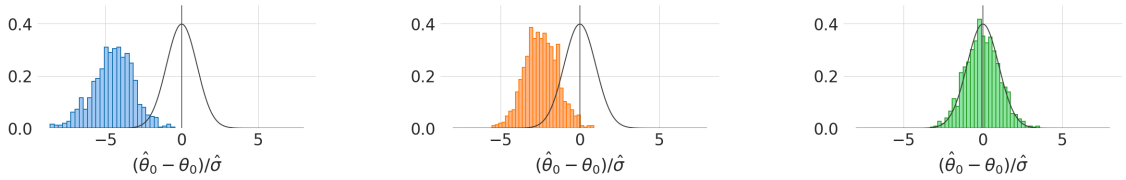
Exemplarily we consider the partially linear regression (PLR) model

$$Y = D\theta_0 + g_0(X) + \zeta, \quad \mathbb{E}(\zeta|D, X) = 0, \tag{1}$$
$$D = m_0(X) + V, \quad \mathbb{E}(V|X) = 0, \tag{2}$$

where a researcher is interested in estimating the causal effect of the treatment variable, D, on the outcome, Y, provided a high-dimensional vector of confounders, X. The DML framework has three key ingredients.

The first key ingredient is the Neyman orthogonality condition. To estimate the causal parameter θ_0, we solve the empirical analog of the moment condition $\mathbb{E}(\psi(W; \theta_0, \eta_0)) = 0$, where $W = (Y, D, X)$, $\psi(\cdot)$ is called the score function and $\eta_0 = (g_0, m_0)$ are the nuisance functions. The score function satisfies the Neyman orthogonality condition if $\partial_\eta \mathbb{E}(\psi(W; \theta_0, \eta))\big|_{\eta=\eta_0} = 0$, where ∂_η denotes the pathwise Gateaux derivative operator. Using ML estimators to approximate the nuisance functions η_0 introduces a regularization bias. Neyman orthogonality implies that this bias has no first-order effect on the estimation of the causal parameter θ_0.

To illustrate this effect, we simulate data from a PLR model. A naive ML approach consists of estimating g_0 with ML methods, for example using random forests, and then plugging-in predictions $\hat{g}_0$ to eventually obtain a naive estimate of θ_0 from an OLS regression of Equation (1). The arising bias is substantial as illustrated in Figure 1a. As an alternative, we can partial out the effect of X on Y and X on D by estimating $\hat{g}_0$ and $\hat{m}_0$ with ML methods. θ_0 can then be estimated from an OLS regression of $Y - \hat{g}_0(X)$ on $D - \hat{m}_0(X)$. This approach implements a Neyman orthogonal score function that identifies θ_0. As shown in Figure 1c, the corresponding estimator is robust to the regularization bias.



(a) Non-Orthogonal Score (b) No Sample Splitting (c) Double Machine Learning

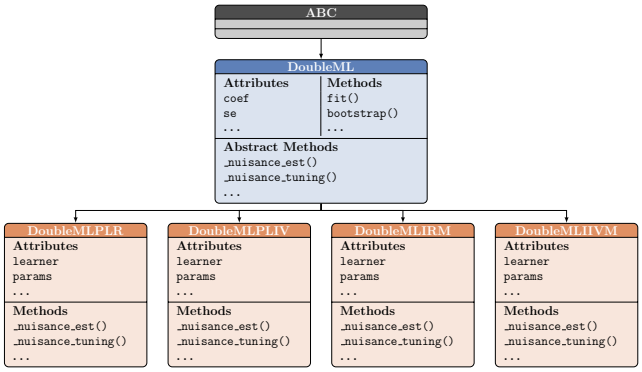
Figure 1: The Effect of the Key Ingredients of Double Machine Learning.

The second key ingredient for DML are high-quality ML methods for estimating the nuisance functions η_0. It is possible to use various ML methods, including lasso, random forests, regression trees, boosting, neural nets or ensemble methods to estimate the nuisance functions. Chernozhukov et al. (2018, Section 3) state formal conditions for the estimation quality and provide a discussion how the selection of appropriate ML methods depends on the structural assumptions for η_0, like for example sparsity.

The third key ingredient for DML is sample splitting. An overfitting bias arises if the nuisance functions are estimated on the same sample as the parameter θ_0 (see Figure 1b vs. 1c). This bias can be overcome by sample splitting, i.e., the nuisance functions η_0 are estimated on one half of the data and the score function is solved for the target parameter

θ_0 on the other half of the data. Sample splitting in K folds is applicable and the usage of repeated cross-fitting is recommended to obtain more efficient estimates.

3. Software Architecture and API




```
import numpy as np
from doubleml import DoubleMLIRM
from doubleml import DoubleMLData
from doubleml.datasets import make_irm_data
from sklearn.ensemble import \
        RandomForestRegressor, \
        RandomForestClassifier
np.random.seed(3141)
df = make_irm_data(return_type='DataFrame',
        n_obs=1000, theta=0.5)
# DoubleMLData from a pandas.DataFrame
dml_data = DoubleMLData(df, y_col='y', d_cols='d')
dml_model = DoubleMLIRM(dml_data,
        RandomForestRegressor(max_depth=5),
        RandomForestClassifier(max_depth=5),
        score='ATE')
dml_model.fit().summary
     coef  std err        t  P>|t|  2.5 %  97.5 %
d 0.5161   0.0750 6.8805 0.0000 0.3691  0.6631
```

Figure 2: Class Structure and API Demonstration of the `DoubleML` Package.

Figure 2 provides a summary of the object-oriented structure and a code snippet demonstrating the API of the the `DoubleML` package. The central class of the package is the abstract base class `DoubleML` which implements all key functionalities: The estimation of the causal parameters, standard errors, t-statistics and confidence intervals, but also methods for valid simultaneous inference through p-value adjustments and the estimation of joint confidence regions via a multiplier bootstrap approach. The different model classes for partially linear regression, partially linear instrumental variable regression, interactive regression and interactive instrumental variable regression are inherited from `DoubleML`. Model classes only have to add the model-specific parts which are the estimation of the nuisance models with ML methods, functionalities for the tuning of these ML methods and methods to compute the model-specific Neyman orthogonal score functions. A key property that all mentioned model classes share is the linearity of the score functions $\psi(\cdot)$ in the parameter θ_0 which is central for the object-oriented and flexible implementation of `DoubleML`.

`DoubleML` gives the user a high flexibility with regard to the specification of DML models including: The choice of ML methods for approximating the nuisance functions, different resampling schemes, selecting among the DML algorithms `dml1` and `dml2` (see Chernozhukov et al., 2018, Section 3) and choosing from different Neyman orthogonal score functions. The object-oriented structure further makes it easy to extend the `DoubleML` package in different directions. For example new model classes with appropriate Neyman orthogonal score function could be inherited from `DoubleML`, e.g., to add an implementation of DML difference-in-differences models (Chang, 2020). The package features `callables` as score functions which makes it easy to extend existing model classes, for example the second-order orthogonal score for the PLR model by Mackey et al. (2018) could be added in this way. Additionally, the resampling schemes are customizable in a flexible way, which for example made it possible to implement the multiway-cluster robust DML model (Chiang et al., 2021) with the `DoubleML` package.

4. Development

Releases of the `DoubleML` package are available via PyPI and conda-forge. The source code of the package and the user guide is hosted on GitHub at `https://github.com/DoubleML/doubleml-for-py` and `https://github.com/DoubleML/doubleml-docs`. Collaborative work on the project is possible via the usual GitHub workflows in issues and pull requests for discussions, feature requests or bug reports. Continuous integration with GitHub Actions ensures backward compatibility and facilitates easy integration of new code. In the project we follow `PEP8` style standards and the package is equipped with an extensive unit test suite which leads to a 99% test coverage. The code quality is rated `A` by LGTM and codacy. Documentation consists of installation instructions, a getting started manual, an extensive user guide and a detailed API reference. It is generated with `sphinx` and hosted via GitHub Pages at `https://docs.doubleml.org`. The package is distributed under the MIT license and relies on core libraries from the scientific Python ecosystem: `scikit-learn` (Pedregosa et al., 2011), `numpy` (Harris et al., 2020), `pandas` (McKinney, 2010), `scipy` (Virtanen et al., 2020), `statsmodels` (Seabold and Perktold, 2010) and `joblib`.

5. Comparison to Related Software

The Python package `DoubleML` was developed together with an R twin `DoubleML` (Bach et al., 2021). The packages have a similar software architecture and API. While the core functionalities are almost the same, a main difference is that the Python package is build on `scikit-learn` (Pedregosa et al., 2011) and the R package on `mlr3` (Lang et al., 2019).

Other Python packages for causal machine learning include `EconML` (Battocchi et al., 2021) and `CausalML` (Chen et al., 2020). These packages have a focus on estimation of effect heterogeneity. `EconML` offers a collection of different estimation methods for conditional average treatment effects, among others based on the double machine learning approach.

The object-oriented architecture of `DoubleML` is centered around orthogonal moment conditions for valid statistical inference. The implementation closely follows the moment condition based theoretical framework in Chernozhukov et al. (2018) for the estimation of causal and structural parameters like for example the average treatment effect, the average treatment effect of the treated or the local average treatment effect. A focus of `DoubleML` is on the extensibility to new model classes which automatically inherit advanced methodology for statistical inference, like for example clustered standard errors, methods for simultaneous inference or the computationally efficient multiplier bootstrap.

6. Future Work

`DoubleML` is under active development and we here want to share some ideas for future extensions. The `DoubleML` packages builds on `scikit-learn`, but we intend to design a flexible API that makes it possible to use learners from other packages like `Keras` or `XGBoost` as well. Furthermore, there are many recently developed extensions to the DML framework (Chang, 2020; Semenova et al., 2017; Mackey et al., 2018; Kallus and Uehara, 2020; Narita et al., 2021) that we would like to integrate into the `DoubleML` package. Another interesting path for future research are cloud computing technologies, for example Kurz (2021) uses serverless computing for estimating DML models with the `DoubleML` package.

Acknowledgments

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project Number 431701914.

References

P. Bach, V. Chernozhukov, M. S. Kurz, and M. Spindler. DoubleML – An object-oriented implementation of double machine learning in R, 2021. arXiv:2103.09603 [stat.ML].

K. Battocchi, E. Dillon, M. Hei, G. Lewis, P. Oka, M. Oprescu, and V. Syrgkanis. EconML: A Python package for ML-based heterogeneous treatment effects estimation, 2021. URL `https://github.com/microsoft/EconML`. Version 0.11.1.

N.-C. Chang. Double/debiased machine learning for difference-in-differences models. *The Econometrics Journal*, 23(2):177–191, 2020. doi: 10.1093/ectj/utaa001.

H. Chen, T. Harinen, J.-Y. Lee, M. Yung, and Z. Zhao. CausalML: Python package for causal machine learning, 2020. arXiv:2002.11631 [cs.CY].

V. Chernozhukov, D. Chetverikov, M. Demirer, E. Duflo, C. Hansen, W. Newey, and J. Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018. doi: 10.1111/ectj.12097.

H. D. Chiang, K. Kato, Y. Ma, and Y. Sasaki. Multiway cluster robust double/debiased machine learning. *Journal of Business & Economic Statistics*, 2021. doi: 10.1080/07350015.2021.1895815.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. Fernández del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, 2020. doi: 10.1038/s41586-020-2649-2.

N. Kallus and M. Uehara. Double reinforcement learning for efficient off-policy evaluation in markov decision processes. *Journal of Machine Learning Research*, 21(167):1–63, 2020.

M. S. Kurz. Distributed double machine learning with a serverless architecture. In *Companion of the ACM/SPEC International Conference on Performance Engineering*, pages 27–33, 2021. doi: 10.1145/3447545.3451181.

M. Lang, M. Binder, J. Richter, P. Schratz, F. Pfisterer, S. Coors, Q. Au, G. Casalicchio, L. Kotthoff, and B. Bischl. mlr3: A modern object-oriented machine learning framework in R. *Journal of Open Source Software*, 2019. doi: 10.21105/joss.01903.

L. Mackey, V. Syrgkanis, and I. Zadik. Orthogonal machine learning: Power and limitations. In *Proceedings of the 35th International Conference on Machine Learning*, pages 3375–3383, 2018.

W. McKinney. Data structures for statistical computing in Python. In *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. doi: 10.25080/Majora-92bf1922-00a.

Y. Narita, S. Yasui, and K. Yata. Debiased off-policy evaluation for recommendation systems. In *Proceedings of the 15th ACM Conference on Recommender Systems*, pages 372–379, 2021. doi: 10.1145/3460231.3474231.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference*, pages 92–96, 2010. doi: 10.25080/Majora-92bf1922-011.

V. Semenova, M. Goldman, V. Chernozhukov, and M. Taddy. Estimation and inference about heterogeneous treatment effects in high-dimensional dynamic panels, 2017. arXiv:1712.09988 [stat.ML].

P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C J Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.