EconML: A Machine Learning Library for Estimating Heterogeneous Treatment Effects

Miruna Oprescu
Microsoft Research
mopresCu@microsoft.com

Vasilis Syrgkanis
Microsoft Research
vasy@microsoft.com

Keith Battocchi
Microsoft Research
kebatt@microsoft.com

Maggie Hei
Microsoft Research
maggie.hei@microsoft.com

Greg Lewis
Microsoft Research
glewis@microsoft.com

Abstract

We introduce EconML, a Python library comprised of state-of-the-art techniques for the estimation of heterogeneous treatment effects from observational data via machine learning. We highlight the features of EconML, present a common API to automate complex causal inference problems, and showcase the usage of EconML to real heterogeneous treatment effect estimation problems.

1 Introduction

One of the biggest promises of machine learning is the automation of decision making in a multitude of application domains. A core problem that arises in most data-driven personalized decision scenarios is the estimation of heterogeneous treatment effects: what is the effect of an intervention on an outcome of interest as a function of a set of observable characteristics of the treated sample? For instance, this problem arises in personalized pricing, where the goal is to estimate the effect of a price discount on the demand as a function of characteristics of the consumer. Similarly, it arises in medical trials where the goal is to estimate the effect of a drug treatment on the clinical response of a patient as a function of patient characteristics. In many such settings we have an abundance of observational data, where the treatment was chosen via some unknown policy and the ability to run control A/B tests is limited.

The EconML package implements recent techniques in the literature at the intersection of econometrics and machine learning that tackle the problem of heterogeneous treatment effect estimation via machine learning-based approaches. These novel methods offer large flexibility in modeling the effect heterogeneity (via techniques such as random forests, boosting, lasso and neural nets), while at the same time leverage techniques from causal inference and econometrics to preserve the causal interpretation of the learned model and many times also offer statistical validity via the construction of valid confidence intervals.

EconML implements techniques from recent academic works from leading groups in the field. Examples include Double Machine Learning (see e.g. [2], [4], [8], [10], [3], [5]), Causal Forests (see e.g. [13], [1], [11]), Deep Instrumental Variables (see e.g. [6]), Non-parametric Instrumental Variables ([9], [12]), and meta-learners (see e.g. [7]). The library brings together all these diverse techniques under a common Python API.

2 Problem Statement

We begin by formulating the abstract problem that is addressed by the library. Subsequently, we will also provide a formulation in the structural equations notation for readers more familiar with that

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

notation.

The methods developed in our library tackle the following general problem: let Y(t) denote the random variable that corresponds to the value of the outcome of interest if we were to treat a sample with treatment $\mathbf{t} \in T$. Given two vectors of treatments $\mathbf{t}_0, \mathbf{t}_1 \in T$, a vector of covariates $\mathbf{x}$ and a random vector of potential outcomes $Y(\mathbf{t})$, we want to estimate the quantity:

$$\tau(\mathbf{t}_0, \mathbf{t}_1, \mathbf{x}) = \mathbb{E}[Y(\mathbf{t}_1) - Y(\mathbf{t}_0) | X = \mathbf{x}] \tag{1}$$

We will refer to the latter quantity as the heterogeneous treatment effect of going from treatment $\mathbf{t}_0$ to treatment $\mathbf{t}_1$ conditional on observables $\mathbf{x}$. If treatments are continuous, then one might also be interested in a local effect around a treatment point. The latter translates to estimating a local gradient around a treatment vector conditional on observables:

$$\partial\tau(\mathbf{t}, \mathbf{x}) = \mathbb{E}[\nabla_{\mathbf{t}} Y(\mathbf{t}) | X = \mathbf{x}] \tag{2}$$

We will refer to the latter as the heterogeneous marginal effect. Finally, we might not only be interested in the effect but also in the actual counterfactual prediction, i.e. estimating the quatity:

$$\mu(\mathbf{t}, \mathbf{x}) = \mathbb{E}[Y(\mathbf{t}) | X = \mathbf{x}] \tag{3}$$

We assume we have data that are generated from some collection policy. In particular, we assume that we have data of the form: $\{Y_i(T_i), T_i, X_i, W_i, Z_i\}$, where $Y_i(T_i)$ is the observed outcome for the chosen treatment, T_i is the treatment, X_i are the covariates used for heterogeneity, W_i are other observable covariates that we believe are affecting the potential outcome $Y_i(T_i)$ and potentially also the treatment T_i; and Z_i are variables that affect the treatment T_i but do not directly affect the potential outcome. We will refer to variables W_i as controls and variables Z_i as instruments. The variables X_i can also be thought of as control variables, but they are special in the sense that they are a subset of the controls with respect to which we want to measure treatment effect heterogeneity. We will refer to them as features.

Structural Equations

We can equivalently describe the data and the quantities of interest via the means of structural equations. In particular, suppose that we observe i.i.d. samples $\{Y_i, T_i, X_i, W_i, Z_i\}$ from some joint distribution and we assume the following structural equation model of the world:

$$Y = g(T, X, W, \epsilon) \tag{4}$$
$$T = f(X, W, Z, \eta) \tag{5}$$

where ϵ and η are noise random variables that are independent of X, Z, T, W but could be potentially correlated with each other. The target quantity that we want to estimate can then be expressed as:

$$\tau(\mathbf{t}_0, \mathbf{t}_1, \mathbf{x}) = \mathbb{E}[g(\mathbf{t}_1, X, W, \epsilon) - g(\mathbf{t_0}, X, W, \epsilon) | X = \mathbf{x}] \tag{6}$$
$$\partial\tau(\mathbf{t}, \mathbf{x}) = \mathbb{E}[\nabla_{\mathbf{t}} g(\mathbf{t}, X, W, \epsilon) | X = \mathbf{x}] \tag{7}$$

where in these expectations, the random variables W, ϵ are taken from the same distribution as the one that generated the data. In other words, there is a one-to-one correspondence between the potential outcomes formulation and the structural equations formulation in that the random variable $Y(\mathbf{t})$ is equal to the random variable $g(t, X, W, \epsilon)$, where X, W, ϵ is drawn from the distribution that generated each sample in the data set.

3 Unified API

The base class of all the methods in our API has the following signature:

```
class BaseCateEstimator

    def fit(self, Y, T, X=None, W=None, Z=None, inference=None):
        ''' Estimates the counterfactual model from data, i.e. estimates
        functions $\tau(\cdot,\cdot,\cdot)$, $\partial\tau(\cdot,\cdot)$ and $\mu(\cdot,\cdot)$

        Parameters:
        Y: $(n \times d_y)$ matrix of outcomes for each sample
        T: $(n \times d_t)$ matrix of treatments for each sample
        X: optional $(n \times d_x)$ matrix of features for each sample
        W: optional $(n \times d_w)$ matrix of controls for each sample
        Z: optional $(n \times d_z)$ matrix of instruments for each sample
        inference: optional string, 'Inference' instance, or None
            Method for performing inference.  All estimators support
            'bootstrap' (or an instance of 'BootstrapInference'), some
            support other methods as well.
        '''

    def effect(self, X=None, *, T0, T1):
        ''' Calculates the heterogeneous treatment effect $\tau(\cdot,\cdot,\cdot)$ between two
        treatment points conditional on a vector of features on a set
        of $m$ test samples $\{T0_i, T1_i, X_i\}$

        Parameters:
        T0: $(m \times d_t)$ matrix of base treatments for each sample
        T1: $(m \times d_t)$ matrix of target treatments for each sample
        X: optional $(m \times d_x)$ matrix of features for each sample

        Returns:
        tau: $(m \times d_y)$ matrix of heterogeneous treatment effects on each
            outcome for each sample
        '''

    def marginal_effect(self, T, X=None):
        ''' Calculates the heterogeneous marginal effect $\partial\tau(\cdot,\cdot)$ around a base
        treatment point conditional on a vector of features on a set of $m$
        test samples $\{T_i, X_i\}$

        Parameters:
        T: $(m \times d_t)$ matrix of base treatments for each sample
        X: optional $(m \times d_x)$ matrix of features for each sample

        Returns:
        grad_tau: $(m \times d_y \times d_t)$ matrix of heterogeneous marginal effects on
            each outcome for each sample
        '''

    def effect_interval(self, X=None, *, T0=0, T1=1, alpha=0.1):
        ''' Confidence intervals for the quantities $\tau(\cdot,\cdot,\cdot)$ produced by the
        model. Available only when inference is not None, when calling the
        fit method.

        Parameters:
        X: optional $(m \times d_x)$ matrix of features for each sample
        T0: optional $(m \times d_t)$ matrix of base treatments for each sample
        T1: optional $(m \times d_t)$ matrix of target treatments for each sample
        alpha: optional float in [0, 1] of the (1-alpha) level of confidence

        Returns:
        lower, upper : tuple of the lower and the upper bounds of the
            confidence interval for each quantity.
        '''
```

Listing 1: Base CATE Estimator Class

Through this unified API, the EconML library can be extended with arbitrary heterogeneous treatment effect estimation methods.

4 Usage Examples: Orange Juice Elasticity

We applied two of the techniques implemented in EconML, namely the Double Machine Learning technique ([2]) and the Orthogonal Random Forest ([11]), to estimate the effect of orange juice price on demand.

To this end, we use Dominick's dataset, a popular historical dataset of store-level orange juice prices and sales provided by University of Chicago Booth School of Business. The dataset is comprised of a large number of features W, but economics researchers might only be interested in learning the elasticity of demand as a function of a few variables x such as income or education. Thus, the methods in [2] and [11] are ideal candidates for this exercise.

We take the features of heterogeneity x to be the average customer income and the controls W to be all other features, including orange juice brand information and customer demographics such as the age, education level, etc. Our results (along with code snippets), depicted in Figs. 1, 2, and 3, unveil the natural phenomenon that lower income consumers are more price-sensitive.

```
from econml.dml import LinearDMLCateEstimator
est = LinearDMLCateEstimator(
            model_y=RandomForestRegressor(),
            model_t=RandomForestRegressor()
            )
est.fit(Y, T, X, W)
te_pred = est.effect(X_test)
```

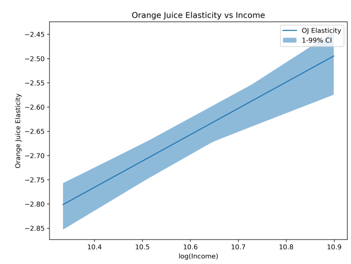

Figure 1: Double Machine Learning (DML) application with linear treatment effect assumption. Left: code snippet from EconML. Right: DML estimates for the effect of orange juice price on demand by income level. The shaded region depicts the 1%-99% confidence interval obtained via bootstrap.

```
from econml.dml import LinearDMLCateEstimator
est = LinearDMLCateEstimator(
            model_y=RandomForestRegressor(),
            model_t=RandomForestRegressor(),
            featurizer=PolynomialFeatures(degree=4)
            )
est.fit(Y, T, X, W)
te_pred = est.effect(X_test)
```

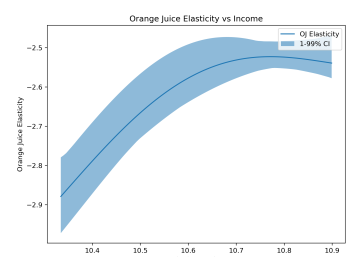

Figure 2: DML application with polynomial treatment effect assumption. Left: code snippet from EconML. Right: DML estimates for the effect of orange juice price on demand by income level. The shaded region depicts the 1%-99% confidence interval obtained via bootstrap.

5 Conclusion

The EconML library is a versatile tool for estimating heterogeneous treatment effects from observational data. With a common API for the different estimation methods, state-of-the art techniques can be continually added to the framework.

We highlight the following features of EconML:

- Built-in inference methods (confidence intervals)
- Built-in cross-validation
- Interpretability tools

```python
from econml.ortho_forest import ContinuousTreatmentOrthoForest
est = ContinuousTreatmentOrthoForest(
        n_trees=n_trees, min_leaf_size=min_leaf_size,
        max_depth=max_depth, subsample_ratio=subsample_ratio,
        bootstrap=bootstrap,
        model_T=Lasso(alpha=0.1),
        model_Y=Lasso(alpha=0.1)
        )
est.fit(Y, T, X, W)
te_pred = est.effect(X_test)
```

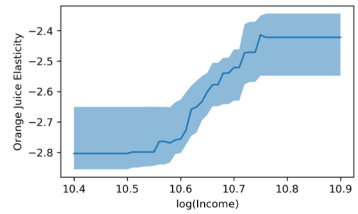

Figure 3: Orthogonal Random Forest (ORF) application with non-parametric treatment effect. Left: code snippet from EconML. Right: ORF estimates for the effect of orange juice price on demand by income level. The shaded region depicts the 1%-99% confidence interval obtained via bootstrap.

- Built on standard Python packages for machine learning and data analysis
- Flexible and reusable for various heterogeneous treatment effect applications
- Open source: Available on GitHub at github.com/microsoft/EconML

We hope that this tool will continue to grow and bring value to causal inference researchers and data scientists alike.

References

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

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

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

[4] Victor Chernozhukov, Matt Goldman, Vira Semenova, and Matt Taddy. Orthogonal machine learning for demand estimation: High dimensional causal inference in dynamic panels. *arXiv preprint arXiv:1712.09988*, 2017.

[5] Dylan J Foster and Vasilis Syrgkanis. Orthogonal statistical learning. *arXiv preprint arXiv:1901.09036*, 2019.

[6] Jason Hartford, Greg Lewis, Kevin Leyton-Brown, and Matt Taddy. Deep iv: A flexible approach for counterfactual prediction. In *International Conference on Machine Learning*, pages 1414–1423, 2017.

[7] Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Meta-learners for estimating heterogeneous treatment effects using machine learning. *arXiv preprint arXiv:1706.03461*, 2017.

[8] Lester Mackey, Vasilis Syrgkanis, and Ilias Zadik. Orthogonal machine learning: Power and limitations. *arXiv preprint arXiv:1711.00342*, 2017.

[9] Whitney K. Newey and James L. Powell. Instrumental variable estimation of nonparametric models. *Econometrica*, 71(5):1565–1578, 2003.

[10] Xinkun Nie and Stefan Wager. Quasi-oracle estimation of heterogeneous treatment effects. *arXiv preprint arXiv:1712.04912*, 2017.

[11] Miruna Oprescu, Vasilis Syrgkanis, and Zhiwei Steven Wu. Orthogonal random forest for heterogeneous treatment effect estimation. *arXiv preprint arXiv:1806.03467*, 2018.

[12] Vasilis Syrgkanis, Victor Lei, Miruna Oprescu, Maggie Hei, Keith Battocchi, and Greg Lewis. Machine learning estimation of heterogeneous treatment effects with instruments. *arXiv preprint arXiv:1905.10176*, 2019.

[13] Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.