Supporting information Appendix for Pichler & Hartig – Can machine learning be used for causal inference?

Table of contents

**Summary:** This document provides supporting information on Pichler & Hartig – Can machine learning be used for causal inference.

## 1 Boosting and regression trees

### 1.1 Unbiasedness

Random forest (RF) and boosted regression trees (BRT) showed bias in both scenarios, with and without collinearity, raising the question of whether the bias is caused by the boosting/bagging or the regression trees themselves. For RF, we know that the observed spillover effect is caused by the random subsampling (mtry parameter) in the algorithm, which explains the bias.

For BRT, however, it is unclear what is causing the bias (boosting or regression trees) because each member in the ensemble is always presented with all features (at least with the default hyperparameters, the BRT implementation in xgboost has options to use bootstrap samples for each tree and also subsamples of columns in each tree (or node), see Chen and Guestrin (2016)).

To understand how boosting and regression trees affect effect estimates, we simulated three different scenarios (Fig. S1, first column) without collinearity (Fig. S1a) and with collinearity (Fig. S1a, b) (we sampled 2000 observations from each data generating model (Fig. S1, first column) and estimated effects using MCE (100 repititions)).

|  |
| --- |
| **Figure S** 1: Bias on effect estimates for different ML algorithms (LM = liner regression model (OLS), RT LC = regression tree with low complexity (depth), RT HC = regression tree with high complexity, Linear Booster, Tree Booster LC = tree booster with low complexity, Tree Booster HC = tree boster with high complexity) in three different simulated causal scenarios (a, b, and c). Sample sizes are so large that stochastic effects can be excluded (2000 observations). Effects of the ML models were inferred using marginal conditional effects. Row a) shows results for simulations with uncorrelated features with the true effect sizes . Row b) shows the results for simulations with X1 and X2 being strongly correlated (Pearson correlation factor = 0.9) but only X1 has an effect on y (mediator) and row c) shows the results for X1 and X2 being strongly correlated (Pearson correlation factor = 0.9) with X1 and X2 having effects on Y (confounder scenario). |

We found that the regression tree (RT) is unable to estimate unbiased effects (Fig. S1), regardless of the presence or absence of collinearity or the complexity of the RT (depth of the regression trees). Without collinearity, effects in regression trees were biased toward zero, less so with higher complexity (Fig. S1). With collinearity, there was a small spillover effect for the RT with high complexity (Fig. S1b) to the collinear zero effect (X2), similar to an l2 regularization. When the collinear feature (X2) had an effect (Fig. S1c), we found a stronger absolute bias for the smaller of the two collinear effects (X2), confirming our expectation that RTs show a greedy effect. This greedy behavior was particularly strong for the low complexity RT (Fig. S1c).

To answer the question of how boosting affects the greediness and spillover effects of RT, we first investigated the behavior of a linear booster because of the well-known behavior of OLS under collinearity. And indeed, we found that the linear booster was unbiased in all three scenarios (compare LM and linear booster in Fig. S1), showing that boosting itself can produce unbiased effects.

Now, comparing the vanilla BRTs with low and high complexity (depth of individual trees) with the linear booster and the RTs, we found similar biases as for the RTs, in terms of spillover with a collinear zero effect and the greediness effect in the presence of a weaker collinear effect (Fig. S1).

### 1.2 Understanding boosting

Intuitive boosting shouldn’t work because it’s basically a regression of residuals. That is, and in the case of collinearity, the stronger of two collinear features in the first model would absorb the effect of the weaker second feature that, for example, causes the omitted variable bias (the effect of the missing confounder is absorbed by the collinear effect).

|  |
| --- |
| **Figure S** 2: Changes of effects within boosting. (A) shows the total effect of ensemble (linear booster) until the n-th ensemble member. (B) shows the effects of the n-th ensemble member. X1 and X2 were correlated (Pearson correlationf factor = 0.9). |

Looking at the evolution of the total effect within a linear booster model (Fig. S2a), we found indeed that the first members of the ensemble absorb the effect of the collinear effect (effect of X1 is absorbed by X1, Fig. S2a), but as members are added to the ensemble, the collinear effect (X2) slowly recovers the effect of the stronger collinear effect until both are at their correct effect estimate (Fig. S2a). This retrieval works by reversing the sign of each member’s effect, so that X1, which initially has an effect of 1.5 (because it absorbed the effect of X2), has small negative effects in subsequent trees, while X2, which is initially estimated at 0, has small positive effects (Fig. S2b).

## 2 Extending MCE to two-way interactions

MCE can be extended to $n$-dimensions to detect way feature interactions. Here, we extended MCEs to two dimensions to detect two-way feature interactions by asking what the change is of when features and change together:

We can approximate with the finite difference method:

and are set to and . All features are centered and standardized.

## 3 Proof of concept - Additional results

### 3.1 Addtional scenarios

|  |
| --- |
| **Figure S** 3: Bias on effect estimates for different ML algorithms in two different simulated causal simulations (a and b). Sample sizes are so large that stochastic effects can be excluded. Effects of the ML models were inferred using average conditional effects. Row a) shows results for simulations with with features (Pearson correlation factor = 0.5) with effect sizes (X1: 1.0, X2: -0.5, X3: 1.0). Row b) shows the results for simulations with X1 and X2 being strongly correlated (Pearson correlation factor = 0.99) but only X1 has an effect on y. |

### 3.2 Additional models

[,1] [,2]  
[1,] 1.000000e+00 0.7071068  
[2,] 0.000000e+00 0.7071068  
[3,] -7.071068e-01 0.0000000  
[4,] -1.836970e-16 1.7071068

|  |
| --- |
| **Figure S** 4: Bias on effect estimates for different ML algorithms in two different simulated causal simulations (a and b). Sample sizes are so large that stochastic effects can be excluded. Effects of the ML models were inferred using average conditional effects. Row a) shows results for simulations with with features (Pearson correlation factor = 0.5) with effect sizes (X1: 1.0, X2: -0.5, X3: 1.0). Row b) shows the results for simulations with X1 and X2 being strongly correlated (Pearson correlation factor = 0.99) but only X1 has an effect on y. |

## 4 Hyperparameter tuning

We performed a hyperparameter search to check if and how hyperparameters influence differently or equally effect estimates and the prediction error, so does a model tune after the prediction error has biased effects? For that, we created simulation scenarios with 50, 100, 600, and 2000 observations and 100 features with effects () , and to were equally spaced between 0.0 to 1.0 so that and .

Features were sampled from a multivariate normal distribution and all features were randomly correlated (Variance-covariance matrix was sampled from a LKJ-distribution with .

1,000 combinations of hyper-parameters were randomly drawn (Table S1). For each draw of hyperparameters, the data simulation and model fitting was repeated 20 times. Effect sizes of X1 and X2 were recorded (for each hyperparameter combination and for each reptition). Moreover, bias, variance, and mean square error (MSE) were recorded for the predictions on a holdout of the same size as the training data.

**Table S** 1: Overview over hyper-parameters for Neural Network, Boosted Regression Tree, and Random Forest

| Algorithm | Hyper-parameter | Range |
| --- | --- | --- |
| Neural Network | activation function | [relu, leaky\_relu, tanh, selu, elu, celu, gelu] |
|  | depth | [1, 8] |
|  | width | [2, 50] |
|  | batch size (sgd) | [1, 100] in percent |
|  | lambda | [2.65e-05, 0.16] |
|  | alpha | [0, 1.0] |
| Boosted Regression Tree | eta | [0.01, 0.4] |
|  | max depth | [2, 25] |
|  | subsample | [0.5, 1] |
|  | max tree | [30, 125] |
|  | lambda | [1, 20] |
| Random Forest | mtry | [0, 1] in percent |
|  | min node size | [2, 70] |
|  | max depth | [2, 50] |
|  | regularization factor | [0, 1] |
| Elastic net | alpha | [0, 1.0] |
|  | lambda | [0, 1.0] |

### 4.1 Results hyperparameter tuning

|  |
| --- |
| **Figure S** 5: Results of hyperparameter tuning for Neural Networks (NN), Boosted Regression Trees (BRT), Random Forests (RF), and Elastic Net (EN) for 50 observations with 100 features. The influence of the hyperparameters on the effect X1 (bias, variance, and MSE), the true simulated effect X1 = 1.0, and the predictions of the model (bias, variance, and MSE) were estimated by a multivariate generalized additive model (GAM). Categorical hyperparameters (activation function in NN) were estimated as fixed effects. The responses (bias, variance, MSE) were centered so that the categorical hyperparameters correspond to the intercepts. The variable importance of the hyperparameters was estimated by a random forest with the MSE of the effect X1 (first plot) or the prediction (second plot) as the response. Orange dots correspond to the best predicted set of hyperparameters (based on a random forest), in the first plot for the minimum MSE of the effect for X1 and in the second plot for the minimum MSE of the predictions. |

|  |
| --- |
| **Figure S** 6: Results of hyperparameter tuning for Neural Networks (NN), Boosted Regression Trees (BRT), Random Forests (RF), and Elastic Net (EN) for 600 observations with 100 features. The influence of the hyperparameters on the effect X1 (bias, variance, and MSE), the true simulated effect X1 = 1.0, and the predictions of the model (bias, variance, and MSE) were estimated by a multivariate generalized additive model (GAM). Categorical hyperparameters (activation function in NN) were estimated as fixed effects. The responses (bias, variance, MSE) were centered so that the categorical hyperparameters correspond to the intercepts. The variable importance of the hyperparameters was estimated by a random forest with the MSE of the effect X1 (first plot) or the prediction (second plot) as the response. Orange dots correspond to the best predicted set of hyperparameters (based on a random forest), in the first plot for the minimum MSE of the effect for X1 and in the second plot for the minimum MSE of the predictions. |

### 4.2 Optimal hyperparameters

The hyperparameters were chosen based on the lowest MSE for the predictive performance of the models (Table S2) and the lowest MSE for the effect () on X1 (Table S3). The selection of the best hyperparameters was done by first fitting a random forest (default parameters) with the MSE as response and the hyperparameters as features, and then using the set of hyperparameters that predicted the lowest MSE.

**Table S** 2: Best predicted set of hyperparameterfor ML algorithms (tuned after MSE of predictions)

| Algorithm | Hyperparameter | n = 50 | n = 100 | n = 600 |
| --- | --- | --- | --- | --- |
| NN | activations | celu | selu | selu |
|  | sgd | 0.944 | 0.348 | 0.098 |
|  | depth | 1 | 1 | 1 |
|  | width | 24 | 20 | 35 |
|  | alpha | 0.939 | 0.821 | 0.693 |
|  | lambda | 0.003 | 0.02 | 0.019 |
| BRT | eta | 0.072 | 0.126 | 0.245 |
|  | max\_depth | 2 | 2 | 2 |
|  | subsample | 0.666 | 0.511 | 0.77 |
|  | lambda | 9.073 | 8.888 | 8.21 |
|  | max\_tree | 117 | 109 | 110 |
| RF | mtry | 0.129 | 0.466 | 0.792 |
|  | min.node.size | 12 | 2 | 3 |
|  | max.depth | 21 | 19 | 47 |
|  | regularization.factor | 0.914 | 0.874 | 0.736 |
| EN | alpha | 0.007 | 0.008 | 0.025 |
|  | lambda | 0.286 | 0.028 | 0.006 |

**Table S** 3: Best predicted set of hyperparameterfor ML algorithms (tuned after MSE of effect X1)

| Algorithm | Hyperparameter | n = 50 | n = 100 | n = 600 |
| --- | --- | --- | --- | --- |
| NN | activations | selu | selu | selu |
|  | sgd | 0.391 | 0.395 | 0.112 |
|  | depth | 3 | 3 | 2 |
|  | width | 18 | 40 | 19 |
|  | alpha | 0.135 | 0.613 | 0.332 |
|  | lambda | 0.009 | 0.011 | 0.002 |
| BRT | eta | 0.252 | 0.327 | 0.393 |
|  | max\_depth | 11 | 17 | 3 |
|  | subsample | 0.514 | 0.584 | 0.523 |
|  | lambda | 9.051 | 7.779 | 9.053 |
|  | max\_tree | 71 | 102 | 124 |
| RF | mtry | 0.137 | 0.926 | 0.462 |
|  | min.node.size | 2 | 4 | 9 |
|  | max.depth | 31 | 29 | 29 |
|  | regularization.factor | 0.683 | 0.894 | 0.587 |
| EN | alpha | 0.011 | 0 | 0.011 |
|  | lambda | 0.016 | 0.018 | 0.009 |

## 5 Additional results for data-poor scenarios

### 5.1 Prediction error of scenarios

|  |
| --- |
| **Figure S** 7: Prediction error (mean square error, MSE) of data poor simulations with optimal hyperparameters either tuned after the best MSE of the effect size (red) or the best MSE of the prediction error (blue). |

## 6 Data-poor scenarios without collinearity

### 6.1 Bias and variance of effects

|  |
| --- |
| **Figure S** 8: Bias and variance of estimated effects in data-poor situations. N = 50, 100, and 600 observations of 100 weakly correlated features were simulated. The effects of X1 and X2 were 1.0 and 0.0. The other 98 effects were equally spaced between 0 and 1. Models were fitted to the simulated data (1000 repititions) with the optimal hyperparameters (except for LM, which doesn’t have hyperparameters). Hyperparameters were chosen based on the minimum MSE of an effect (green) or the prediction error (red). Bias and variance were calculated for X1 and X2. Effects were approximated using ACE. |

### 6.2 Predictions error of scenarios (without collinearity)

### 6.3 Prediction error of scenarios

|  |
| --- |
| **Figure S** 9: Prediction error (mean square error, MSE) of data poor simulations with optimal hyperparameters either tuned after the best MSE of the effect size (red) or the best MSE of the prediction error (blue). |

## 7 Detecting interactions

## 8 Weighted MCE

If the instances of a feature x\_j are not uniformly distributed, we propose to calculate a weighted with the being, for example, the inverse probabilities of an estimated density function over the feature space of .

To demonstrate the idea of weighted MCE, we simulated a scenario with one feature where the for values of the feature and for the other feature values (Fig. S4). The feature was sampled from a log-Normal distribution. We fitted a linear regression model and a NN on the data and compared the effect estimated by the LM, the unweighted MCE, and the weighted MCE.

The LM estimated an effect of 1.48, the unweighted MCE was 1.95, and the weighted MCE was 1.48 (Fig. S16).

|  |
| --- |
| **Figure S** 10: Simulation example with non-uniform sampled feature X1 (log normal distributed). The red line is the effect estimated by a LM OLS. The blue line is the effect reported by an unweighted MCE from a NN. The green line is the effect reported by a weighted MCE from a NN. |

## 9 Case study - RMSE

**Table** **:** In-sample R2 of BRT, RF, NN, and LM in predicting the risk of Lung Cancer

| **name** | **BRT** | **RF** | **NN** | **LM** |
| --- | --- | --- | --- | --- |
| Causal ML | 0.58 | 0.54 | 0.63 | 0.64 |
| Conventional ML 1 | 0.84 | 0.84 | 0.87 | 0.87 |

**Table** **:** Out-of-sample R2 of BRT, RF, NN, and LM in predicting the risk of Lung Cancer with intervention on Lung Volume

| **name** | **BRT** | **RF** | **NN** | **LM** |
| --- | --- | --- | --- | --- |
| Causal ML | 0.58 | 0.54 | 0.64 | 0.64 |
| Conventional ML 1 | 0.45 | 0.48 | 0.41 | 0.41 |

**Table** **:** Out-of-sample R2 of BRT, RF, NN, and LM in predicting the risk of Lung Cancer with changed correlation structure because of the unobservable confounder Stress

| **name** | **BRT** | **RF** | **NN** | **LM** |
| --- | --- | --- | --- | --- |
| Conventional ML 1 | 0.60 | 0.55 | 0.67 | 0.68 |
| Causal ML | 0.85 | 0.73 | 0.94 | 0.97 |
| Conventional ML 2 | 0.89 | 0.75 | 0.97 | 0.99 |

## 10 Learning in neural networks

Chen, Tianqi, and Carlos Guestrin. 2016. “Xgboost: A Scalable Tree Boosting System.” In *Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining*, 785–94.