Modified Causal Forest: Optimal Policy Estimation

Section 1: General information

Welcome to the mcf estimation and optimal policy package.
This report provides you with a summary of specifications and results. More detailed information can be found in the respective output files. Figures and data (in csv-format, partly to recreate the figures on your own) are provided in the output path as well.

Output information for OPTIMAL POLICY ANALYSIS
Path for all outputs:
Q:\SEW\Projekte\MLechner\Projekte und Angebote\Unicef\Kasachstan\Workshops\Astana\Wednesday_examples070/example/outputOPTPT
Detailed text output:
Q:\SEW\Projekte\MLechner\Projekte und Angebote\Unicef\Kasachstan\Workshops\Astana\Wednesday_examples070/example/outputOPTPT/OptPolicy.0.7.0.txt
Summary text output:
Q:\SEW\Projekte\MLechner\Projekte und Angebote\Unicef\Kasachstan\Workshops\Astana\Wednesday_examples070/example/outputOPTPT/OptPolicy.0.7.0_Summary.txt

BACKGROUND

The optimal policy module offers three (basic) algorithms that can be used to exploit fine grained knowledge about effect heterogeneity to obtain decision rules. The current version is implemented for discrete treatments only.
There is also an option for different fairness adjustments.

The BEST_POLICY_SCORE algorithm is based on assigning the treatment that has the highest impact at the unit (e.g., individual) level. If the treatment heterogeneity is known (not estimated), this will lead to the best possible result. This algorithm is computationally not burdensome. However, it will not be easy to understand how the implied rules depends on the features of the unit. Its statistical properties are also not clear (for estimated treatment heterogeneity) and there is a certain danger of overfitting, which could lead to an unsatisfactory out-of-training-sample performance.

The BPS_CLASSIFIER classifier algorithm runs a classifier for each of the allocations obtained by the BEST_POLICY_SCORE algorithm. One advantage of this approach compared to the BEST_POLICY_SCORE algorithm is that prediction of the allocation of (new) observations is fast because it does not require to recompute the policy score (as it is the case with the BEST_POLICY_SCORE algorithm). The specific classifier is selected among four different classifiers from scikit-learn, namely a simple neural network, two classification random forests with minimum leaf size of 2 and 5, and ADDABoost. The selection is a made according to the out-of-sample performance of the Accuracy Score of scikit-learn.

The POLICY TREE algorithm builds optimal shallow decision trees. While these trees are unlikely to lead to gloablly optimal allocations, and are computationally much more expensive, they have the advantage that the decision rule is much easier to understand and that some statistical properties are known, at least for certain versions of such decision trees (e.g., Zhou, Athey, Wager, 2023). The basic algorithmic implementation follows the recursive algorithm suggested by Zhou, Athey, Wager

(2023) with three (more substantial) deviations (=extensions).
Extension 1: Since using One Hot Encoding for categorical variables may lead to rather extreme leaves for such variables with many different values when building (shallow) trees (splitting one value against the rest), a more sophisticated procedure is used that allows to have several values of the categorical variables on both sides of the split.
Extension 2: Constraints are allowed for. They are handled in a sequential manner: First, an approximate treatment-specific cost vector is obtained and used to adjust the policy score accordingly. Second, trees that violate the constraints are removed (to some extent, optional).
Extensions 3: There are a several options implemented to reduce the computational burden, which are discussed below in the section showing the implementation of the policy score.

References
-Zhou, Z., S. Athey, S. Wager (2023): Offline Multi-Action Policy Learning: Generalization and Optimization, Operations Research, INFORMS, 71(1), 148-183.

Modified Causal Forest: Optimal Policy Estimation

Section 2: Optimal Policy

METHOD
The assignment rule is based on allocating units using a shallow decision tree of depth 4 (based on 2 optimal trees, depth of 1st tree: 2, depth of 2nd tree: 2).

VARIABLES provided
Policy scores: y_pot0, y_pot1, y_pot2
IATEs relative to first treatment state: iate1vs0, iate2vs0
Treatment dependent variables for descriptive analysis: zero, ite1vs0, ite2vs0, x_cont0, iate1vs0, iate2vs0
Treatment: treat
Identifier: id
Oderered features of units: x_cont0, x_cont1, x_cont2, x_ord0, x_ord1, x_ord2
Categorical / unorderered features of units: x_unord0, x_unord1, x_unord2
Features used for variable importance statistics without transformations: x_cont0, x_cont1, x_cont2
Features that are transformed to indicator/dummy variables for variable importance computations (only): x_unord0

COSTS
No user provided costs of specific treatments.

RESTRICTIONS of treatment shares
The following restrictions on the treatment shares are specified 100%, 100%, 30.0%.

FAIRNESS
No fairness adjustments performed.

Section 2.1: Optimal Policy: Training

COMPUTATION
6 logical cores are used for processing.
Continuous variables are internally split for best use of cpu ressources.

DATA PREPARATION
Variables without variation are removed.
Variables that are perfectly correlated with other variables are removed.
Dummy variables with less than 10 observations in the smaller group are removed.
Rows with any missing values for variables needed for training are removed.

RESTRICTIONS on treatment shares
Restrictions are taken into account by modifying the policy scores with artificial costs. These artificial costs are computed such that a Black-Box allocation will respect the constraints automatically.
If the allocated treatment shares are not close enough to the desired shares, then these artificial costs can be adjusted by specifying/changing the cost multiplier (keyword "costs_of_treat_mult").

Modified Causal Forest: Optimal Policy Estimation

COMPUTATIONAL EFFICIENCY
Optimal policy trees are computationally very demanding. Therefore, several approximation parameters are used.
Instead of evaluating all values of continuous variables and combinations of values of categorical variables when splitting, only 100 values are considered. These values are equally spaced for continuous variables and random combinations for categorical variables. This number is used for EVERY splitting decision, i.e. the approximation improves the smaller the data in the leaf becomes. Increasing this value can significantly improve the computational performance at the price of a certain approximation loss.
The depth of the tree is also a key parameter. Usually, it is very hard to estimate trees beyond the depth of 4 (16 leaves) with reasonably sized training data. There are two options to improve the computational performance. The first one is to reduce the depth (leading to loss of efficiency but a gain in interpretability). The second option is to split the tree building into several steps.
In this application, this two-step tree buildung option is implemented in the following way: After buildung the first tree of depth 2, in each leaf of this tree, a second optimal tree of depth 2 is built. Subsequently, these trees are combined to form the final tree of depth 4. For given final tree depth, the more similar the depths of the two trees are, the faster the algorithm. However, the final tree will of course be subject to an additional approximation error.
Another parameter crucial for performance is the minimum leaf size. Too small leaves may be undesirable for practical purposes (and they increase computation times). The minimum leaf size in this application is set to 4.
In addition, the user may reduce the size of the training data to increase speed, but this will increase sampling noise.

CATEGORICAL VARIABLES
There are two different approximation methods for larger categorical variables. Since we build optimal trees, for categorical variables we need to check all possible combinations of the different values that lead to binary splits. This number could indeed be huge. Therefore, we compare only 200 different combinations. The available methods differ on how these methods are implemented. In this application, at each possible split, we sort the values of the categorical variables according to the values of the policy scores as one would do for a standard random forest. If this set is still too large, a random sample of the entailed combinations is drawn.

STRUCTURE OF FINAL TREE (using data from training fair)

Leaf information for estimated policy tree

Depth of 1st tree: 2, depth of 2nd tree: 2, total depth: 4
- -
Leaf 00: x_cont0 <= -0.842 x_cont1 <=   1.068 x_cont2 <= -0.681 x_cont2 <= -0.778
Alloc Treatment: 0   Obs: 56
- -
Leaf 01: x_cont0 <= -0.842 x_cont1 <=   1.068 x_cont2 <= -0.681 x_cont2 >   -0.778
Alloc Treatment: 2   Obs: 6
- -
Leaf 10: x_cont0 <= -0.842 x_cont1 <=   1.068 x_cont2 >   -0.681 x_cont1 <=   0.204

Modified Causal Forest: Optimal Policy Estimation

Alloc Treatment: 0 Obs: 102
- -
Leaf 11: x_cont0 <= -0.842 x_cont1 <= 1.068 x_cont2 > -0.681 x_cont1 > 0.204
Alloc Treatment: 1 Obs: 38
- -
Leaf 20: x_cont0 <= -0.842 x_cont1 > 1.068 x_ord0 <= 0.500 x_cont2 <= 0.243
Alloc Treatment: 2 Obs: 21
- -
Leaf 21: x_cont0 <= -0.842 x_cont1 > 1.068 x_ord0 <= 0.500 x_cont2 > 0.243
Alloc Treatment: 1 Obs: 11
- -
Leaf 30: x_cont0 <= -0.842 x_cont1 > 1.068 x_ord0 > 0.500 x_unord1 In: 0 7 9
Alloc Treatment: 0 Obs: 6
- -
Leaf 31: x_cont0 <= -0.842 x_cont1 > 1.068 x_ord0 > 0.500 x_unord1 Not in: 0 7 9
Alloc Treatment: 1 Obs: 15
- -
Leaf 40: x_cont0 > -0.842 x_unord0 In: 0 7 8 x_cont2 <= 0.965 x_cont1 <= 1.291
Alloc Treatment: 2 Obs: 103
- -
Leaf 41: x_cont0 > -0.842 x_unord0 In: 0 7 8 x_cont2 <= 0.965 x_cont1 > 1.291
Alloc Treatment: 1 Obs: 13
- -
Leaf 50: x_cont0 > -0.842 x_unord0 In: 0 7 8 x_cont2 > 0.965 x_unord2 In: 1 5 9
Alloc Treatment: 2 Obs: 10
- -
Leaf 51: x_cont0 > -0.842 x_unord0 In: 0 7 8 x_cont2 > 0.965 x_unord2 Not in: 1 5 9
Alloc Treatment: 1 Obs: 23
- -
Leaf 60: x_cont0 > -0.842 x_unord0 Not in: 0 7 8 x_unord2 In: 0 2 4 5 7 9 x_cont1 <= -1.650
Alloc Treatment: 0 Obs: 6
- -
Leaf 61: x_cont0 > -0.842 x_unord0 Not in: 0 7 8 x_unord2 In: 0 2 4 5 7 9 x_cont1 > -1.650
Alloc Treatment: 1 Obs: 346
- -
Leaf 70: x_cont0 > -0.842 x_unord0 Not in: 0 7 8 x_unord2 Not in: 0 2 4 5 7 9 x_cont2 <= 0.302
Alloc Treatment: 1 Obs: 154
- -
Leaf 71: x_cont0 > -0.842 x_unord0 Not in: 0 7 8 x_unord2 Not in: 0 2 4 5 7 9 x_cont2 > 0.302
Alloc Treatment: 2 Obs: 90
- -
NOTE: Splitpoints displayed for ordered variables are midpoints between observable values (e.g., 0.5 for a variable with values of 0 and 1).

Modified Causal Forest: Optimal Policy Estimation

Section 2.2: Optimal Policy: Evaluation of Allocation(s)

Main evaluation results.

Note: The output files contain relevant additional information, like a descriptive analysis of the treatment groups and variable importance statistics.

Evaluation of treatment allocation

Allocation	*Value function*	*Share of 0 in %*	*Share of 1 in %*	*Share of 2 in %*
All Policy Tree	1.1797	17.0	60.0	23.0
All observed	0.5182	33.3	33.4	33.3
All random	0.6532	30.2	34.0	35.8
Switchers Policy Tree	1.2189	17.98	57.46	24.56
Switchers random	0.68	30.38	33.38	36.24

Note: Allocation analysed is the SAME as the one obtained from the training data.

Evaluation of treatment allocation

Allocation	*Value function*	*Share of 0 in %*	*Share of 1 in %*	*Share of 2 in %*
All Policy Tree	0.9848	19.5	58.6	21.9
All observed	0.4973	33.3	33.4	33.3
All random	0.6461	30.2	34.0	35.8
Switchers Policy Tree	1.0206	22.71	56.49	20.8
Switchers random	0.7073	29.33	35.05	35.62

Note: Allocation analysed is DIFFERENT from the one obtained from the training data.