Package ‘policytree’

January 16, 2020

Title Policy Learning via Doubly Robust Empirical Welfare Maximization over Trees
Version 0.9.0
Depends R (>= 3.5.0)
License GPL-3
Encoding UTF-8
LazyData true
Suggests testthat (>= 2.1.0), DiagrammeR
RoxygenNote 7.0.1
LinkingTo Rcpp, BH
Imports Rcpp, grf (>= 1.0.0)
URL https://github.com/grf-labs/policytree
NeedsCompilation yes
Author Zhengyuan Zhou [aut],
        Susan Athey [aut],
        Stefan Wager [aut],
        Ayush Kanodia [aut],
        Erik Sverdrup [cre]
Maintainer Erik Sverdrup <erikcs@stanford.edu>
Repository CRAN
Date/Publication 2020-01-16 11:40:02 UTC

R topics documented:

double_robust_scores.causal_forest ........................................ 2
gen_data_epl ........................................................................ 3
gen_data_mapl ...................................................................... 4
multi_causal_forest .............................................................. 4
double_robust_scores.causal_forest

Matrix $\Gamma$ of scores for each treatment $a$

Description

Computes a matrix of double robust scores $\Gamma_{ia} = \mu_a(x) + \frac{1}{\sigma_a(x)} (Y_i - \mu_a(x)) 1(A_i = a)$

Usage

## S3 method for class 'causal_forest'
double_robust_scores(object, ...)

## S3 method for class 'instrumental_forest'
double_robust_scores(object, compliance.score = NULL, ...)

## S3 method for class 'multi_causal_forest'
double_robust_scores(object, ...)

double_robust_scores(object, ...)

Arguments

- object: An appropriate causal forest type object
- ...: Additional arguments
- compliance.score: An estimate of the causal effect of Z on W. i.e., $\Delta(X) = E(W \mid X, Z = 1) - E(W \mid X, Z = 0)$, for each sample $i = 1, ..., n$. If NULL (default) then this is estimated with a causal forest.

Details

This is the matrix used for CAIPWL (Cross-fitted Augmented Inverse Propensity Weighted Learning)

Value

A matrix of scores for each treatment
Methods (by class)

- **causal_forest**: Scores \((\Gamma_0, \Gamma_1)\)
- **instrumental_forest**: Scores \((-\Gamma, \Gamma)\)
- **multi_causal_forest**: Matrix \(\Gamma\) of scores for each treatment \(a\)

Note

For instrumental_forest this method returns \((-\Gamma_i, \Gamma_i)\) where \(\Gamma_i\) is the double robust estimator of the treatment effect as in eqn. (52) in Athey and Wager (2017).

References


Examples

```r
n <- 500
p <- 10
d <- 3
X <- matrix(runif(n * p), n, p)
Y <- runif(n)
W <- sample(1:d, n, replace = TRUE)
forests <- multi_causal_forest(X = X, Y = Y, W = W)
double_robust_scores(forests)
```

---

**gen_data_epl**

Example data generating process from Efficient Policy Learning

**Description**

The DGP from section 5.2 in Athey and Wager (2017)

**Usage**

```r
gen_data_epl(n, type = c("continuous", "jump"))
```

**Arguments**

- `n` Number of observations
- `type` tau is "continuous" (default - equation 54) or exhibits "jumps" (equation 55)

**Value**

A list
References


---

`gen_data_mapl`  
*Example data generating process from Offline Multi-Action Policy Learning: Generalization and Optimization*

---

Description

The DGP from section 6.4.1 in Zhou, Athey, and Wager (2018): There are \( d = 3 \) actions \((a_0, a_1, a_2)\) which depend on 3 regions the covariates \( X \) reside in. Observed outcomes: \( Y \sim N(\mu_{a_i}(X), 4) \)

Usage

`gen_data_mapl(n, p = 10, sigma2 = 4)`

Arguments

- **n**: Number of observations \( X \).
- **p**: Number of features (minimum 7). Default is 10.
- **sigma2**: Noise variance. Default is 4.

Value

A list with realized action \( a_i \), region \( r_i \), conditional mean \( \mu \), outcome \( Y \) and covariates \( X \)

References


---

`multi_causal_forest`  
*One vs. all causal forest for multiple treatment effect estimation*

---

Description

For \( k \) treatments this "naive" multivariate-grf proceeds by fitting \( k \) separate causal forests where in forest \( i \) the treatment assignment vector is one-hot encoded for treatment \( i \). The steps are:

1. Estimate propensities for each action \( 1..k \): \( e_k \). This is done with \( k \) separate regression forests with propensities normalized to sum to 1 at the final step.
2. Estimate the expected response \( m(x) = E(Y \mid X_i) \) marginalizing over treatment. This is done with one regression forest.
3. Estimate each \( \tau_i \) with a causal forest
multi_causal_forest

Usage

multi_causal_forest(
  X,
  Y,
  W,
  Y.hat = NULL,
  W.hat = NULL,
  num.trees = 2000,
  num.trees.orthog = max(50, num.trees/4),
  sample.weights = NULL,
  clusters = NULL,
  equalize.cluster.weights = FALSE,
  sample.fraction = 0.5,
  mtry = min(ceiling(sqrt(ncol(X)) + 20), ncol(X)),
  min.node.size = 5,
  honesty = TRUE,
  honesty.fraction = 0.5,
  honesty.prune.leaves = TRUE,
  alpha = 0.05,
  imbalance.penalty = 0,
  stabilize.splits = TRUE,
  ci.group.size = 2,
  tune.parameters = "none",
  tune.num.trees = 200,
  tune.num.reps = 50,
  tune.num.draws = 1000,
  compute.oob.predictions = TRUE,
  orthog.boosting = FALSE,
  num.threads = NULL,
  seed = runif(1, 0, .Machine$integer.max)
)

Arguments

X         The covariates used in the causal regression.
Y         The outcome (must be a numeric vector with no NAs).
W         The treatment assignment (must be a categorical vector with no NAs).
           If Y.hat = NULL, these are estimated using a separate regression forest. See
           section 6.1.1 of the GRF paper for further discussion of this quantity. Default is
           NULL.
W.hat     Matrix with estimates of the treatment propensities $E[W_k | X_i]$. If W.hat =
           NULL, these are estimated using a k separate regression forests. Default is
           NULL.
num.trees Number of trees grown in the forest. Note: Getting accurate confidence intervals
           generally requires more trees than getting accurate predictions. Default is 2000.
num.trees.orthog
Number of trees used in Y.hat and W.hat estimation (optional). Default is max(50, num.trees / 4).

sample.weights
(Experimental) Weights given to each sample in estimation. If NULL, each observation receives the same weight. Note: To avoid introducing confounding, weights should be independent of the potential outcomes given X. Default is NULL.

clusters
Vector of integers or factors specifying which cluster each observation corresponds to. Default is NULL (ignored).

equalize.cluster.weights
If FALSE, each unit is given the same weight (so that bigger clusters get more weight). If TRUE, each cluster is given equal weight in the forest. In this case, during training, each tree uses the same number of observations from each drawn cluster: If the smallest cluster has K units, then when we sample a cluster during training, we only give a random K elements of the cluster to the tree-growing procedure. When estimating average treatment effects, each observation is given weight 1/cluster size, so that the total weight of each cluster is the same. Note that, if this argument is FALSE, sample weights may also be directly adjusted via the sample.weights argument. If this argument is TRUE, sample.weights must be set to NULL. Default is FALSE.

sample.fraction
Fraction of the data used to build each tree. Note: If honesty = TRUE, these subsamples will further be cut by a factor of honesty.fraction. Default is 0.5.

mtry
Number of variables tried for each split. Default is $\sqrt{p} + 20$ where p is the number of variables.

min.node.size
A target for the minimum number of observations in each tree leaf. Note that nodes with size smaller than min.node.size can occur, as in the original randomForest package. Default is 5.

honesty
Whether to use honest splitting (i.e., sub-sample splitting). Default is TRUE. For a detailed description of honesty, honesty.fraction, honesty.prune.leaves, and recommendations for parameter tuning, see the grf algorithm reference.

honesty.fraction
The fraction of data that will be used for determining splits if honesty = TRUE. Corresponds to set $J_1$ in the notation of the paper. Default is 0.5 (i.e. half of the data is used for determining splits).

honesty.prune.leaves
If true, prunes the estimation sample tree such that no leaves are empty. If false, keep the same tree as determined in the splits sample (if an empty leave is encountered, that tree is skipped and does not contribute to the estimate). Setting this to false may improve performance on small/marginally powered data, but requires more trees (note: tuning does not adjust the number of trees). Only applies if honesty is enabled. Default is TRUE.

alpha
A tuning parameter that controls the maximum imbalance of a split. Default is 0.05.

imbalance.penalty
A tuning parameter that controls how harshly imbalanced splits are penalized. Default is 0.
**multi_causal_forest**

Specifies the following parameters:

- **stabilize.splits**: Whether or not the treatment should be taken into account when determining the imbalance of a split. Default is TRUE.
- **ci.group.size**: The forest will grow ci.group.size trees on each subsample. In order to provide confidence intervals, ci.group.size must be at least 2. Default is 2.
- **tune.parameters**: A vector of parameter names to tune. If "all": all tunable parameters are tuned by cross-validation. The following parameters are tunable: ("sample.fraction", "mtry", "min.node.size", "honesty.fraction", "honesty.prune.leaves", "alpha", "imbalance.penalty"). If honesty is false these parameters are not tuned. Default is "none" (no parameters are tuned).
- **tune.num.trees**: The number of trees in each 'mini forest' used to fit the tuning model. Default is 200.
- **tune.num.reps**: The number of forests used to fit the tuning model. Default is 50.
- **tune.num.draws**: The number of random parameter values considered when using the model to select the optimal parameters. Default is 1000.
- **compute.oob.predictions**: Whether OOB predictions on training set should be precomputed. Default is TRUE.
- **orthog.boosting** (experimental) If TRUE, then when Y.hat = NULL or W.hat is NULL, the missing quantities are estimated using boosted regression forests. The number of boosting steps is selected automatically. Default is FALSE.
- **num.threads**: Number of threads used in training. By default, the number of threads is set to the maximum hardware concurrency.
- **seed**: The seed of the C++ random number generator.

**Value**

A trained multi causal forest object (collection of causal forests). If tune.parameters is enabled, then tuning information will be included through the tuning.output attribute of each forest.

**Examples**

```r
n <- 500
p <- 10
d <- 3
X <- matrix(runif(n * p), n, p)
Y <- runif(n)
W <- sample(1:d, n, replace = TRUE)
mcf <- multi_causal_forest(X = X, Y = Y, W = W)
mcf

# Treatments may be labeled arbitrarily
W <- sample(c("TreatmentA", "TreatmentB", "TreatmentC"), n, replace = TRUE)
mcf.named <- multi_causal_forest(X = X, Y = Y, W = W)
mcf.named
```
plot.policy_tree  

Plot a policy_tree tree object.

Description
Plot a policy_tree tree object.

Usage
```r
## S3 method for class 'policy_tree'
plot(x, ...)
```

Arguments
- `x`: The tree to plot
- `...`: Additional arguments (currently ignored).

policy_tree  

Fit a policy with exact tree search

Description
Finds the optimal (maximizing the sum of rewards) depth L tree by exhaustive search. If the optimal action is the same in both the left and right leaf of a node, the node is pruned.

Usage
```r
policy_tree(X, Gamma, depth = 2)
```

Arguments
- `X`: The covariates used. Dimension \( N \times p \) where \( p \) is the number of features.
- `Gamma`: The rewards for each action. Dimension \( N \times d \) where \( d \) is the number of actions.
- `depth`: The depth of the fitted tree. Default is 2.

Value
A policy_tree object.

References
Examples

```r
n <- 50
p <- 10
d <- 3
features <- matrix(rnorm(n * p), n, p)
rewards <- matrix(rnorm(n * d), n, d)
tree <- policy_tree(features, rewards, depth = 2)
tree
```

```r
predict.multi_causal_forest
Predict with multi_causal_forest
```

Description

Computes estimates of $\tau_a(x)$

Usage

```r
## S3 method for class 'multi_causal_forest'
predict(object, newdata = NULL, ...)
```

Arguments

- `object`: The trained forest.
- `newdata`: Points at which predictions should be made. If NULL, makes out-of-bag predictions on the training set instead (i.e., provides predictions at Xi using only trees that did not use the i-th training example). Note that this matrix should have the number of columns as the training matrix, and that the columns must appear in the same order.
- `...`: Additional arguments passed to grf::predict.causal_forest.

Value

List containing matrix of predictions and other estimates (debiased error, etc.) for each treatment.

Examples

```r
n <- 250
p <- 10
d <- 3
X <- matrix(runif(n * p), n, p)
Y <- runif(n)
W <- sample(1:d, n, replace = TRUE)
mcf <- multi_causal_forest(X = X, Y = Y, W = W)
```
predict.policy_tree

Description

Predict values based on fitted policy_tree object.

Usage

## S3 method for class 'policy_tree'
predict(object, newdata, ...)

Arguments

- **object**: policy_tree object
- **newdata**: A data frame with features
- **...**: Additional arguments (currently ignored).

Value

A vector of predictions. Each element is an integer from 1 to d where d is the number of columns in the reward matrix.

Examples

```
n <- 50
p <- 10
d <- 3
features <- matrix(rnorm(n * p), n, p)
rewards <- matrix(rnorm(n * d), n, d)
tree <- policy_tree(features, rewards, depth = 2)
print(tree)
predict(tree, features)
```
Print a multi_causal_forest object.

Usage

```r
defprint.multi_causal_forest(x, ...)```

Arguments

- `x` The object to print.
- `...` Additional arguments (currently ignored).

Print a policy_tree object.

Usage

```r
defprint.policy_tree(x, ...)```

Arguments

- `x` The tree to print.
- `...` Additional arguments (currently ignored).
Index

double_robust_scores
  (double_robust_scores.causal_forest), 2

double_robust_scores.causal_forest, 2

gen_data_epl, 3

gen_data_mapl, 4

multi_causal_forest, 4

plot.policy_tree, 8
policy_tree, 8
predict.multi_causal_forest, 9
predict.policy_tree, 10
print.multi_causal_forest, 11
print.policy_tree, 11