Package 'fastpolicytree'

June 24, 2025

Type Package

Title Constructs Policy Trees from Covariate and Reward Data

Version 1.0

Description Constructs optimal policy trees which provide a rule-based treatment prescription policy. Input is covariate and reward data, where, typically, the rewards will be doubly robust reward estimates.
This package aims to construct optimal policy trees more quickly than the existing 'policytree' package and
is intended to be used alongside that package.
For more details see Cussens, Hatamyar, Shah and Kreif (2025) <doi:10.48550/arXiv.2506.15435>.

URL https://github.com/jcussens/tailoring

Suggests policytree

Imports Rcpp (>= 1.0.7)

LinkingTo Rcpp

RoxygenNote 7.3.2

Encoding UTF-8

License GPL (>= 3)

NeedsCompilation yes

Author James Cussens [aut, cre] (ORCID:
<https://orcid.org/0000-0002-1363-2336>),
Julia Hatamyar [ctb],
Vishalie Shah [ctb],
University of Bristol [cph],
MRC [fnd]

Maintainer James Cussens <james.cussens@bristol.ac.uk>

Repository CRAN

Date/Publication 2025-06-24 08:50:06 UTC

Contents

`fastpolicytree` *Construct an optimal policy tree from covariate and reward data*

Description

This function accepts almost the same input and generates the same type of output as the policy_tree function in the policytree package. The only difference is that this function has no 'split.step' argument (since it is effectively hard-coded to the value 1).

Usage

```
fastpolicytree(
  X,
  Gamma,
  depth = 3,
  min.node.size = 1,
  strategy.datatype = 2,
  strategy.find.reward.ub = FALSE,
  strategy.find.dummy.split.reward = FALSE,
  strategy.use.last.rewards = TRUE,
  strategy.use.cutoffs = FALSE,
  strategy.use.cache = TRUE,
  strategy.exploitbinaryvars = TRUE
)
```

Arguments

`X`	The covariates used. Dimension $N * p$ where p is the number of features.
`Gamma`	The rewards for each action. Dimension $N * d$ where d is the number of actions.
`depth`	The depth of the fitted tree. Default is 3.
`min.node.size`	An integer indicating the smallest terminal node size permitted. Default is 1.
`strategy.datatype`	If set to 0 policytree style sorted sets are used to represent datasets during solving. If set to 1 then unsorted sets are used which are sorted 'on demand'. If set to to 2 then the choice of representation is decided automatically. Default is 2 (choice is automatically made).
`strategy.find.reward.ub`	If TRUE upper bounds on rewards are computed. Default is FALSE

`strategy.find.dummy.split.reward`
: If TRUE then the reward for a dummy split (where the left split has no datapoints) is computed. Default is FALSE.

`strategy.use.last.rewards`
: If TRUE an upper bound on the reward for a split is computed from the reward for the most recent split value for the current covariate. Default is TRUE

`strategy.use.cutoffs`
: If TRUE then tree finding is aborted if it can be deduced that the reward for the tree is beaten by some existing tree. Default is FALSE

`strategy.use.cache`
: If TRUE a cache of optimal trees for (sub-)datasets is used. Default is TRUE

`strategy.exploitbinaryvars`
: If TRUE then covariates with only 2 values are treated specially. Default is TRUE

Value

A policy_tree object.

Examples

```
X <-  data.frame(
 X1=c(-0.32, 0.16, 0.34, 1.24, 0.22, 0.45, 1.48, 0.65,-0.93,-1.11),
 X2=c(-0.58, 0.90,-0.22, 1.54,-0.57,-1.08,-1.42,-1.98,-0.02, 0.05),
 X3=c(0.70,-1.49, 0.36,-0.05,-0.14, 1.57,-0.18,-1.98,-1.77,-1.25),
 X4=c(0.21, 0.34, 0.60,-0.05,-0.66,-0.69, 0.52, 0.31,-0.03, 1.09),
 X5=c(0.16, 0.96,-1.07,-0.97, 2.02,-0.43,-0.79,-2.08, 1.21, 0.39))
Gamma  <- data.frame(
 control=c(0.8502363,-1.4950411,1.9608062,0.7487925,2.9718517,
  0.8952429,-0.2563680,5.9945581,-1.8485703,-1.2840477),
 treat=c(-2.91607259,-2.25464535, 0.28214637,-0.17284650,-0.09480810,
  1.48786125,2.08600119,-2.05283394,0.72903608,-0.04416392))
tree3  <- fastpolicytree(X,Gamma)
tree3
tree2  <- fastpolicytree(X,Gamma,depth=2)
tree2
# to get a human-readable display of the trees use the
# policytree package...
#library(policytree)
#print(tree3)
#print(tree2)
```

`githash` *Returns git hash for compiled C code*

Description

Returns git hash for compiled C code

Usage

```
githash()
```

Value

A string which is the relevant githash

Index