AHB-vignette

Overview

Introduction

AHB (Adaptive Hyper-box) Matching is an interpretable matching method for causal inference. It matches units with others in unit-specific, hyper-box-shaped regions of the covariate space. These regions are large enough that many matches are created for each unit and small enough that the treatment effect is roughly constant throughout.For more details, see the below section *Description of the Algorithm* or the original AHB matching paper, linked here.

Making Matches

We can start by loading AHB…

```
library(AHB)
```

… and generating some toy data using the included `gen_data` function.

```
set.seed(45)
n <- 50
p <- 5
data <- gen_data(n, p) # Data we would like to match
holdout <- gen_data(n, p) # Data we will train on, to generate outcome model Y
```

Note that all our covariates can either be continuous, categorical or mixed (continous and categorical). In addition to the covariates to match on, `data` and `holdout` contains an outcome and a treated column:

```
names(data)
#> [1] "X1"      "X2"      "X3"      "X4"      "X5"      "outcome" "treated"
```

The outcome `Y` must be numeric, either binary or continuous. AHB matching focuses on binary treatments and the treatment column must either be logical or binary numeric.

From here, we can run AHB with its default parameters. This will match units on the covariates – here, `X1`, `X2`, `X3`, `X4`, `X5` – and output information about the matches that were made.

```
AHB_fast_out <- AHB_fast_match(data = data, holdout = holdout)
```

```
#> Running AHB_fast_matching
#> Matching unit 1 of 23
Matching unit 2 of 23
Matching unit 3 of 23
Matching unit 4 of 23
Matching unit 5 of 23
Matching unit 6 of 23
Matching unit 7 of 23
Matching unit 8 of 23
Matching unit 9 of 23
Matching unit 10 of 23
Matching unit 11 of 23
Matching unit 12 of 23
Matching unit 13 of 23
Matching unit 14 of 23
Matching unit 15 of 23
Matching unit 16 of 23
Matching unit 17 of 23
Matching unit 18 of 23
Matching unit 19 of 23
Matching unit 20 of 23
Matching unit 21 of 23
Matching unit 22 of 23
Matching unit 23 of 23
#> Time to match 23 units: 3.89 secs
AHB_MIP_out <- AHB_MIP_match(data = data, holdout = holdout)
#> Running AHB_MIP_matching
#> Matching unit 1 of 23
Matching unit 2 of 23
Matching unit 3 of 23
Matching unit 4 of 23
Matching unit 5 of 23
Matching unit 6 of 23
Matching unit 7 of 23
Matching unit 8 of 23
Matching unit 9 of 23
Matching unit 10 of 23
Matching unit 11 of 23
Matching unit 12 of 23
Matching unit 13 of 23
Matching unit 14 of 23
Matching unit 15 of 23
Matching unit 16 of 23
Matching unit 17 of 23
Matching unit 18 of 23
Matching unit 19 of 23
Matching unit 20 of 23
```

```
Matching unit 21 of 23
Matching unit 22 of 23
Matching unit 23 of 23
Time to match 23 units: 1.36 secs
```

By default, either `AHB_fast_match` or `AHB_MIP_match` returns a list with 5 entries respectively:

```
names(AHB_fast_out)
#> [1] "data"     "units_id" "CATE"     "bins"     "MGs"      "verbose"
```

Take return of `AHB_fast_match` as an example.

The first, `AHB_fast_out$data` is a dataframe that was matched. If holdout is not a numeric value, the `AHB_fast_out$data` is the same as the data input into `AHB_fast_match()`. If holdout is a numeric scalar between 0 and 1, `AHB_fast_out$data` is the remaining proportion of data that were matched.

```
all(AHB_fast_out$data == data)
#> [1] TRUE
```

The second, `AHB_fast_out$units_id` is a integer vector that contains the unit id of treated units in dataset matched. And we aan get matched group for each treated unit below in `AHB_fast_out$MGs`

```
AHB_fast_out$units_id
#>  [1]  6  7  8 12 13 14 15 16 22 23 25 26 28 31 32 33 34 37 41 46 48 49 50
```

The third, `AHB_fast_out$CATE` is a numeric vector with the conditional average treatment effect estimates for each matched group in `AHB_fast_out$MGs`.

```
AHB_fast_out$CATE
#>  [1]  0.16914765  4.25611355  0.12645334  5.21728087 -1.35194378 -1.21309734
#>  [7]  0.70083876 -1.67701821  5.21728087  5.95781011 -1.05922729  0.75490075
#> [13] -1.95251236  0.06112651  5.21728087 -0.74869560  0.70176757  6.41431021
#> [19] -0.44165784 -1.76698547 -0.75484083 -0.10907419 -0.99145127
```

The fourth `AHB_fast_out$bins` is an array of two lists where the first list contains lower bounds and the other one contains upper bounds for each test treated unit. Each row of each lists is a vector containing the lower or upper bounds of the hpyer-box for each treated unit in `AHB_fast_out$units_id`. `AHB_fast_out$bins[,1]` contains the lower bounds of hpyer-boxes while `AHB_fast_out$bins[,2]` contains the upper bounds of the upper bounds of hpyer-boxes.

For instance, if we want to know the hpyer-box of a paticular unit in `AHB_fast_out$units_id`, we just need to find the corresponding row of `AHB_fast_out$bins[,1]` to get the lower bounds and the corresponding row of `AHB_fast_out$bins[,2]` to get upper bounds.

```
print("Lower bounds for the first hyper_box: ")
```

```
#> [1] "Lower bounds for the first hyper_box: "
AHB_fast_out$bins[1,,1]
#> [1] -2.022415 -1.418114 -3.277960  2.102896 -3.164994
print("Upper bounds for the first hyper_box: ")
#> [1] "Upper bounds for the first hyper_box: "
AHB_fast_out$bins[1,,2]
#> [1] -1.311098  4.574344  4.856455  2.666419  4.211783
```

The fifth is `AHB_fast_out$MGs` A list of all the matched groups formed by `AHB_fast_match`. For each test treated unit in `AHB_fast_out$units_id`, each row contains all unit_id of the other units that fall into its box, including itself. For instance, the frist row of `AHB_fast_out$MGs` represents the first hpyer-box, which means that the frist row of `AHB_fast_out$MGs` has all unit_ids that fall into the first hyper-box.

```
head(AHB_fast_out$MGs)
#> [[1]]
#> [1] 1 6
#>
#> [[2]]
#> [1] 22  7 32 43
#>
#> [[3]]
#> [1]  8 27
#>
#> [[4]]
#> [1] 12 22  7 29 32 43
#>
#> [[5]]
#> [1] 48 36 13
#>
#> [[6]]
#> [1] 14 28 10
```

Analyzing Matches

`ATE(AHB_fast_out)` and `ATT(AHB_fast_out)` take in the output of a call to `AHB_fast_match` and return the estimated average treatment effect and the estimated average treatment effect on the treated, respectively.

```
ATE(AHB_out = AHB_fast_out)
#> [1] 2.26585
ATT(AHB_out = AHB_fast_out)
#> [1] 2.384093
```

Description of Arguments

Below are brief descriptions of the main arguments that may be passed to `AHB_fast_match` and `AHB_MIP_match`. For their complete descriptions, and those of all acceptable arguments, please refer to the documentation.

Data Arguments

These are arguments that govern the format in which data is passed to `AHB_fast_match` or `AHB_MIP_match`.

- `data`: Either a data frame or a path to a .csv file to be read. If holdout is not a numeric value, this is the data to be matched. If holdout is a numeric scalar between 0 and 1, that proportion of data will be made into a holdout set and only the remaining proportion of data will be matched. Covariates can either be categorical, continuous or mixed (categorical and continuous). Outcome must be either binary or continuous (both numeric) with a outcome name . Treatment must be described by a logical or binary column with treated name.
- `holdout`: A data frame, a path to a .csv file to be read or a numeric scalar between 0 and 1. If a numeric scalar, that proportion of data will be made into a holdout set and only the remaining proportion of data will be matched. Otherwise, a dataframe or a path to a .csv file. In this case, restrictions on column types are the same as for `data`. Must have same column names and order as `data`.
- `treated_column_name`: A character with the name of the column to be used as treatment in `data`. Defaults to 'treated'.
- `outcome_column_name`: A character with the name of the column to be used as outcome in `data`. Defaults to 'outcome'.

Algorithmic Arguments

These are arguments that deal with features of the underlying AHB matching algorithm.

Arguments for AHB_MIP_match

- `black_box`: Denotes the method to be used to generate outcome model Y. If "BART" and cv = "F", uses `dbarts::bart` with keeptrees = TRUE, keepevery = 10, verbose = FALSE, k = 2 and ntree =200 and then the default predict method to estimate the outcome. If "BART" and cv = "T", k and ntree will be best values from cross validation. Defaults to 'BART'. There will be multiple choices about black_box in the future.
- `cv` A logical scalar. If "T", do cross-validation on the train set to generate outcome model Y . Defaults to "T".
- `gamma0`: A numeric value, one of hyperparameters in global MIP that controls the weight placed on the outcome function portion of the loss. Defaults to 3.
- `gamma1`: A numeric value, one of hyperparameters in global MIP that controls the weight placed on

the outcome function portion of the loss. Defaults to 3.
- `beta`: A numeric value, one of hyperparameters in global MIP that controls the weight placed on the number of units in the box. Defaults to 2.
- `m`: A numeric value, the at least number of control units that the box contains when estimating causal effects for a single treatment unit. Defaults to 1.
- `M`: A numeric value, a large positive constant that controls the weight placed on decision variable wij, which is an indicator for whether a unit is in the box.Defaults to 1e5.
- `n_prune`: a numeric value, the number of candidate units selected to run on mip for constructing the box. Dataset mentioned below is refered to the dataset for matching. If you match a small dataset with the number of units smaller than 400, it will run MIP on all dataset for each treated unit. If you match larger dataset and your memory of your computer cannot support such much computation or the speed is extremely slow, plase adjust n_prune below 400 or even smaller. The smaller number of candidate units selected to run the mip on for constructing the box, the faster this program runs.Defualt n_prune = 0.1* nrow(dataset).
- `MIP_solver` Denotes the method to be used to solve MIP problem. Optional, if “Rcplex”, use Rcplex as solver, which is faster than Rglpk but not free. if “Rglpk”, use Rglpk as solver, which is free and easy to be installed.

Arguments for AHB_fast_match

- `black_box`: Denotes the method to be used to generate outcome model Y. If “BART” and cv = F, uses with keeptrees = TRUE, keepevery = 10, verbose = FALSE, k = 2 and ntree =200 and then the default predict method to estimate the outcome. If “BART” and cv = T, k and ntree will be best values from cross validation. Defaults to ‘BART’. There will be multiple choices about black_box in the future.
- `cv` A logical scalar. If “T”, do cross-validation on the train set to generate outcome model Y . Defaults to “T”.
- `C`: a parameter in `AHB_fast_match`, a positive scalar. Determines the stopping condition for Fast AHB. When the variance in a newly expanded region exceeds C times the variance in the previous expansion region, the algorithm stops. Thus, higher C encourages coarser bins while lower C encourages finer ones. The user should analyze the data with multiple values of C to see how robust results are to its choice.
- `n_prune`: A numeric value, the number of candidate units selected to construct the box.If computation speed is very slow, please adjust it into smaller values. When n_prune > 300, `AHB_fast_match` usually runs faster than `AHB_MIP_match`. Defualt n_prune = 0.1* nrow(dataset).

Description of the Algorithm

AHB matching is a matching algorithm for observational data that matches units with others in unit-specific, hyper-box-shaped regions of the covariate space. These regions are large enough that many matches are created for each unit and small enough that the treatment effect is roughly constant throughout. The regions are found as either the solution to a mixed integer program `AHB_MIP_match` , or using a (fast) approximation algorithm `AHB_fast_match`. The result is an interpretable and tailored estimate of a causal effect for each unit.

The approach learns an optimal adaptive coarsening of the covariate space from a model trained on a

separate training dataset, leading to accurate estimates of the treatment effect and interpretable matches. The matched group for a unit consists of all units within a learned unit-specific high dimensional hyper-box. These hyper-boxes are constructed so that they (1) contain enough units for reliable treatment effect estimates, and also so that (2) units within each box have similar pre-treatment outcomes, which lowers the bias of the estimated treatment effect. This is achieved by learning hyper-boxes such that the variability in treatment effect estimates is reduced. This also allows us to avoid black-box summaries (propensity or prognostic scores) and ad-hoc pre-specified metrics given by the users. Our estimates are interpretable. First, they are case-based: each individual's estimate can be explained in terms of the other units they are matched with. Second, the choice of cases is itself interpretable: if two units are matched together, it is because they fall in the same easily-described hyper-box.

For more details, see the AHB matching paper