# Dissertation_Code

David McCoy

2023-06-22

## Example

First load the package and other packages needed

```
library(CVtreeMLE)
library(sl3)
library(pre)
library(partykit)
library(kableExtra)
library(ggplot2)

set.seed(429153)
```

## Simulate Data

```
sim_data <- simulate_mixture_cube(
  n_obs = 800,
  splits = c(0.99, 2.0, 2.5),
  mins = c(0, 0, 0),
  maxs = c(3, 4, 5),
  subspace_assoc_strength_betas = c(
    0, 0, 0, 0,
    0, 0, 6, 0
  )
)
```

Let’s take a look at the data:

```
head(sim_data)
```

```
##           age        bmi        sex        M1         M2        M3          y
## 1  0.01651435 -0.4227082 -1.0221195 1.7594922 0.03442708 2.7936966 -0.9910446
## 2  0.19072911  0.4842019  0.9771362 0.1961772 2.34932053 1.3962661  1.1728384
## 3 -0.18790449  0.4828171 -1.0221195 0.4488381 0.04331044 2.6834768 -1.2116326
## 4 -0.19596384 -1.1133632 -1.0221195 0.1387679 2.78777587 0.6990761 -1.2167154
## 5  0.26243848  0.6081797  0.9771362 1.6475103 1.33051234 1.6460804  1.2452379
## 6 -1.32782405 -1.0698419 -1.0221195 1.4097762 0.00406810 2.8495084 -2.3520191
```

Using the `simulate_mixture_cube` we generate 800 observations that are exposed to three variables with min values being 0 for all and max values being 3,4, and 5. In each variable we define split points 0.99, 2.0, and 2.5. Given the eight regions in the cube, the `subspace_assoc_strength_betas` parameter is where we put the outcome in a specific region.

The indices correspond to an area in the cube:

1. All mixtures lower than specified thresholds
2. M1 is higher but M2 and M3 are lower
3. M2 is higher but M1 and M3 are lower
4. M1 and M2 are higher and M3 is lower
5. M3 is higher and M1 and M2 are lower
6. M1 and M3 are higher and M2 is lower
7. M2 and M3 are higher and M1 is lower
8. All mixtures are higher than thresholds

So here - we put 6 at index 7 which means the outcome is 6 when M2 and M3 are higher and M1 is lower than their respective split points. The outcome is 0 in all other regions.

## Run `CVtreeMLE`

We will now pass the simulated data and variable names for each node in O = W,A,Y to the `CVtreeMLE` function.

```r
ptm <- proc.time()

sim_results <- CVtreeMLE(
  data = sim_data,
  w = c("age", "sex", "bmi"),
  a = c(paste("M", seq(3), sep = "")),
  y = "y",
  n_folds = 5,
  parallel_cv = TRUE,
  seed = 2333,
  parallel_type = "multi_session",
  family = "continuous",
  num_cores = 6
)

proc.time() - ptm
```

```
##    user  system elapsed 
##  95.819  24.804 447.939
```

Note that above, there are default estimators for all parameters if they are not passed to the function. Here we just use the out of the box estimators that are defined in `utils_create_sls.R`. These estimators are chosen to be both non-parametric but also not too computationally demanding. Examples of estimators used by default are random forest, xgboost, elastic net, and glms. Users can also pass in their own custom stacks of learners. We also see here that, using 6 cores with these learners on our simulated data with 800 observations and 6 variables, our run time is 6 minutes. This can be greatly improved by increasing the num_cores parameter.

## Results

We can look at the pooled TMLE results for this model. Let's see if `CVtreeMLE` identified the current rule in all our folds:

```
mixture_results <- sim_results$`Pooled TMLE Mixture Results`
mixture_results %>%
  dplyr::filter(Proportion_Folds == 1.0)
```

```
##   Mixture ATE Standard Error Lower CI Upper CI P-value P-value Adj     Vars
## 1       3.259          0.158    2.949    3.568       0           0    M1-M2
## 2       5.935          0.037    5.862    6.007       0           0 M1-M2-M3
##    RMSE
## 1 2.128
## 2 1.069
##                                                                          Union_Rule
## 1                            M1 >= 0.002 & M1 <= 0.966 & M2 >= 1.336 & M2 <= 3.968
## 2 M1 >= 0.002 & M1 <= 0.989 & M2 >= 1.966 & M2 <= 3.968 & M3 >= 2.436 & M3 <= 4.99
##   Proportion_Folds
## 1                1
## 2                1
```

Above, the estimated mixture ATE for this rule is 5.94 (5.84 - 6.03), which covers our true mixture ATE used to generate the data which was 6. The estimated mixture ATE is interpreted as: the average counterfactual mean outcome if all individuals were exposed to the rule shown in `Union Rule` compared to if all individuals were unexposed is 5.94. That is, those individuals who are exposed to this rule have an outcome that is 5.94 higher compared to those that are not exposed to this rule. The standard error, confidence intervals and p-values are derived from the influence curve of this estimator.

We can also look at the v-fold specific results. This gives the analyst the ability to investigate how stable the estimates and rules are. These results are the same as standard sample splitting techniques and therefore have proper variance estimates and p-values. Below we show the v-fold specific interactions found with fold specific estimates of our ATE target parameter and variance estimates from the fold specific IC.

```
mixture_v_results <- sim_results$`V-Specific Mix Results`
mixture_v_results$`M1-M2-M3`
```

```
##     ate    se lower_ci upper_ci p_val p_val_adj  rmse
## 1 5.893 0.066   5.7630   6.0230     0         0 1.184
## 2 5.946 0.043   5.8610   6.0300     0         0 0.997
## 3 5.946 0.109   5.7320   6.1600     0         0 1.178
## 4 5.940 0.114   5.7160   6.1630     0         0 1.300
## 5 5.948 0.071   5.8090   6.0880     0         0 1.113
## 6 5.935 0.190   5.5627   6.3077     0         0 1.089
##                                                                            mix_rule
## 1                                             M3 > 2.468 & M2 > 1.975 & M1 <= 0.986
## 2                                             M3 > 2.481 & M1 <= 0.995 & M2 > 1.975
## 3                                             M2 > 2.006 & M3 > 2.408 & M1 <= 0.985
## 4                                             M3 > 2.481 & M2 > 1.966 & M1 <= 0.986
## 5                                             M3 > 2.481 & M2 > 1.975 & M1 <= 0.989
## 6 M1 >= 0.002 & M1 <= 0.989 & M2 >= 1.966 & M2 <= 3.968 & M3 >= 2.436 & M3 <= 4.99
##     fold variables
## 1      1  M1-M2-M3
## 2      2  M1-M2-M3
## 3      3  M1-M2-M3
## 4      4  M1-M2-M3
## 5      5  M1-M2-M3
## 6 Pooled  M1-M2-M3
```

In v-fold specific results we also give a pooled estimate. This is different than the pooled TMLE estimate. Here we simply take the weighted average of the fold specific ATEs and the harmonic mean of the variances. This is similar to meta-analysis approaches.

We can plot our v-fold mixture results findings using the `plot_mixture_results` function. This will return a list of plots with names corresponding to the interactions found.

This plot shows the ATE specific for each fold and for the weighted-mean results over the fold with corresponding pooled variance. The rule is the union rule which includes all observations that were indicated by the fold specific rules.

`CVtreeMLE` also data-adaptively identifies thresholds in the marginal space. This feature is described in the vignette. In the marginal setting, partitions are found for each mixture variable individually and the ATE is in reference to the baseline (lowest leaf) value.

Additional details for this and other features are given in the vignette.