Implementing Double Robust Estimators: Some Practical Considerations

Ashley I Naimi

Oct 2022

Contents

1 Fitting Large DR Estimators: Some Practical Considerations

In this session, we are going to finally implement both `tmle3` and `AIPW` with some strategies employed in practice. These strategies are meant to deal with challenges that often arise when using machine learning methods for estimating causal effects. For example:

1) While it is recommended to use a large super learner library with many algorithms, and use a wide array of tuning parameter values for each algorithm, running the code to estimate treatment effects with TMLE and AIPW often requires (sometimes considerable) debugging. To avoid extensive delays while trying to debug code, it would be useful to employ a strategy that enables us to run simple versions of the algorithms and add complexity in increments.

2) After running the TMLE or AIPW algorithms with the richly specified (and slow!) super learner algorithms, we often need to revisit the model to evaluate properties of the fit. However, failing to document or report something that is needed later requires re-fitting the (slow!) algorithm.

3) It is often important to visualize and evaluate important properties of the models, including propensity score distributions, outcome predictions, super learner coefficient and risks, and other things. Yet these elements can be buried deep within the objects created by `TMLE` or `AIPW`.

In the code that follows, I try to demonstrate some practical strategies you can take to simplify dealing with these issues.

2 Using sl3 with tmle3

We start with TMLE:

```
library(tmle3)
library(sl3)
library(tidyverse)
library(here)
```

```
scale_ <- function(x) {
    (x - mean(x, na.rm = TRUE))/sd(x, na.rm = TRUE)
}
nhefs <- read_csv(here("data", "nhefs.csv")) %>%
    mutate(wt_delta = as.numeric(wt82_71 >
        median(wt82_71)), age = scale_(age),
        sbp = scale_(sbp), dbp = scale_(dbp),
        price71 = scale_(price71), tax71 = scale_(tax71)) %>%
    select(-wt82_71)

head(nhefs)
```

```
## # A tibble: 6 x 11
##    seqn  qsmk   sex     age income    sbp     dbp price71  tax71  race wt_delta
##   <dbl> <dbl> <dbl>   <dbl>  <dbl>  <dbl>   <dbl>   <dbl>  <dbl> <dbl>    <dbl>
## 1   233     0     0 -0.112      19  2.50   1.74     0.197  0.204     1        0
## 2   235     0     0 -0.614      18 -0.284  0.224    0.922  1.44      0        1
## 3   244     0     1  1.06       15 -0.712 -0.251   -2.53  -2.38      1        1
## 4   245     0     0  2.06       15  1.06   0.0342  -2.81  -2.51      1        1
## 5   252     0     0 -0.279      18 -0.551 -0.0609   0.922  1.44      0        1
## 6   257     0     1 -0.0286     11  0.680  0.509    0.314  0.450     1        1
```

```
# CREATE SUPERLEARNER LIBRARY choose
# base learners

sl3_list_learners("binomial")
```

```
##  [1] "Lrnr_bartMachine"                "Lrnr_bayesglm"
##  [3] "Lrnr_bound"                      "Lrnr_caret"
##  [5] "Lrnr_cv_selector"                "Lrnr_dbarts"
##  [7] "Lrnr_earth"                      "Lrnr_ga"
##  [9] "Lrnr_gam"                        "Lrnr_gbm"
## [11] "Lrnr_glm"                        "Lrnr_glm_fast"
## [13] "Lrnr_glmnet"                     "Lrnr_grf"
## [15] "Lrnr_gru_keras"                  "Lrnr_h2o_glm"
## [17] "Lrnr_h2o_grid"                    "Lrnr_hal9001"
## [19] "Lrnr_lightgbm"                    "Lrnr_lstm_keras"
## [21] "Lrnr_mean"                        "Lrnr_nnet"
## [23] "Lrnr_nnls"                        "Lrnr_optim"
## [25] "Lrnr_pkg_SuperLearner"            "Lrnr_pkg_SuperLearner_method"
## [27] "Lrnr_pkg_SuperLearner_screener" "Lrnr_polspline"
## [29] "Lrnr_randomForest"                "Lrnr_ranger"
## [31] "Lrnr_rpart"                       "Lrnr_screener_correlation"
## [33] "Lrnr_solnp"                       "Lrnr_stratified"
## [35] "Lrnr_svm"                         "Lrnr_xgboost"
```

```
lrnr_mean <- make_learner(Lrnr_mean)
lrnr_glm <- make_learner(Lrnr_glm)

# ranger learner
grid_params <- list(num.trees = c(250, 500,
    1000, 2000), mtry = c(2, 4, 6), min.node.size = c(50,
    100))
grid <- expand.grid(grid_params, KEEP.OUT.ATTRS = FALSE)
lrnr_ranger <- vector("list", length = nrow(grid))
for (i in 1:nrow(grid)) {
    lrnr_ranger[[i]] <- make_learner(Lrnr_ranger,
        num.trees = grid[i, ]$num.trees,
        mtry = grid[i, ]$mtry, min.node.size = grid[i,
            ]$min.node.size)
}
########################################################################
lrnr_ranger <- make_learner(Lrnr_ranger)  ###########  FLAG! ###########
########################################################################

# glmnet learner
grid_params <- seq(0, 1, by = 0.25)
lrnr_glmnet <- vector("list", length = length(grid_params))
for (i in 1:length(grid_params)) {
    lrnr_glmnet[[i]] <- make_learner(Lrnr_glmnet,
        alpha = grid_params[i])
}
########################################################################
lrnr_glmnet <- make_learner(Lrnr_glmnet)  ###########  FLAG! ###########
########################################################################

# xgboost learner
grid_params <- list(max_depth = c(2, 4, 6,
    8), eta = c(0.01, 0.1, 0.2), nrounds = c(50,
    100, 500))
grid <- expand.grid(grid_params, KEEP.OUT.ATTRS = FALSE)
lrnr_xgboost <- vector("list", length = nrow(grid))
for (i in 1:nrow(grid)) {
    lrnr_xgboost[[i]] <- make_learner(Lrnr_xgboost,
        max_depth = grid[i, ]$max_depth,
        eta = grid[i, ]$eta)
}
########################################################################
lrnr_xgboost <- make_learner(Lrnr_xgboost)  ##########  FLAG! ##########
########################################################################

# earth learner
grid_params <- c(2, 3, 4, 5, 6)
lrnr_earth <- vector("list", length = length(grid_params))
for (i in 1:length(grid_params)) {
    lrnr_earth[[i]] <- make_learner(Lrnr_earth,
        degree = grid_params[i])
}
########################################################################
lrnr_earth <- make_learner(Lrnr_earth)  ############  FLAG! ############
########################################################################

sl_ <- make_learner(Stack, unlist(list(lrnr_mean,
    lrnr_glm, lrnr_ranger, lrnr_glmnet, lrnr_xgboost,
    lrnr_earth), recursive = TRUE))

# DEFINE SL_Y AND SL_A We only need
# one, because they're the same

Q_learner <- Lrnr_sl$new(learners = sl_,
    metalearner = Lrnr_nnls$new(convex = T))
g_learner <- Lrnr_sl$new(learners = sl_,
    metalearner = Lrnr_nnls$new(convex = T))
learner_list <- list(Y = Q_learner, A = g_learner)

########################################################################

# PREPARE THE THINGS WE WANT TO FEED IN
# TO TMLE3
ate_spec <- tmle_ATE(treatment_level = 1,
    control_level = 0)

nodes_ <- list(W = c("age", "sbp", "dbp",
    "price71", "tax71", "sex", "income",
    "race"), A = "qsmk", Y = "wt_delta")

# RUN TMLE3
set.seed(123)
tmle_fit_ <- tmle3(ate_spec, nhefs, nodes_,
    learner_list)

print(tmle_fit_)
```

```
## A tmle3_Fit that took 1 step(s)
##    type                param  init_est  tmle_est         se      lower
## 1:  ATE ATE[Y_{A=1}-Y_{A=0}] 0.1290174 0.1440591 0.03031777 0.08463742
##        upper psi_transformed lower_transformed upper_transformed
## 1: 0.2034809       0.1440591        0.08463742         0.2034809
```

```r
saveRDS(tmle_fit_, here("misc", "tmle_fit-preliminary.rds"))

tmle_task <- ate_spec$make_tmle_task(nhefs,
    nodes_)

initial_likelihood <- ate_spec$make_initial_likelihood(tmle_task,
    learner_list)

## save propensity score for diagnosis
propensity_score <- initial_likelihood$get_likelihoods(tmle_task)$A
propensity_score <- propensity_score * nhefs$qsmk +
    (1 - propensity_score) * (1 - nhefs$qsmk)

# min and max
print(min(propensity_score))
```

```
## [1] 0.09530209
```

```r
print(max(propensity_score))
```

```
## [1] 0.5600149
```

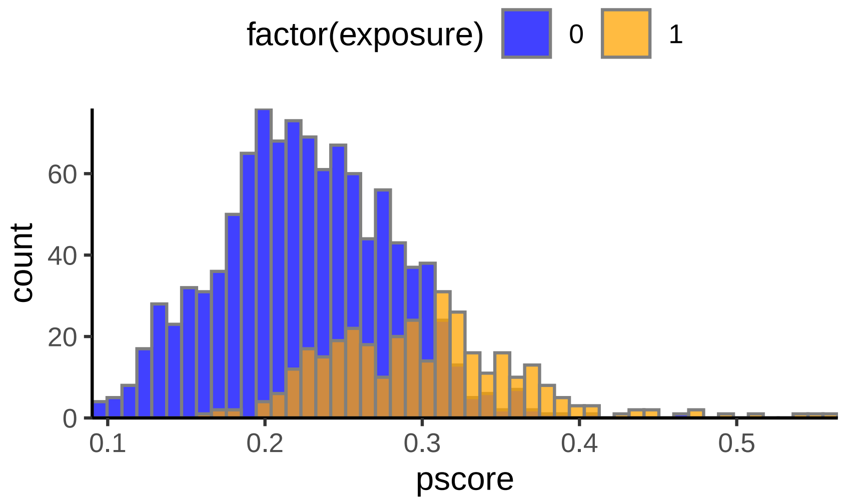
```r
plap_ <- tibble(exposure = nhefs$qsmk, pscore = propensity_score)
plap_ <- plap_ %>%
    mutate(sw = exposure * (mean(exposure)/propensity_score) +
        (1 - exposure) * ((1 - mean(exposure))/(1 -
            propensity_score)))

# distribution of PS and stabilized
# weights
summary(plap_$sw)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4355  0.8998  0.9597  0.9487  1.0203  1.5539
```

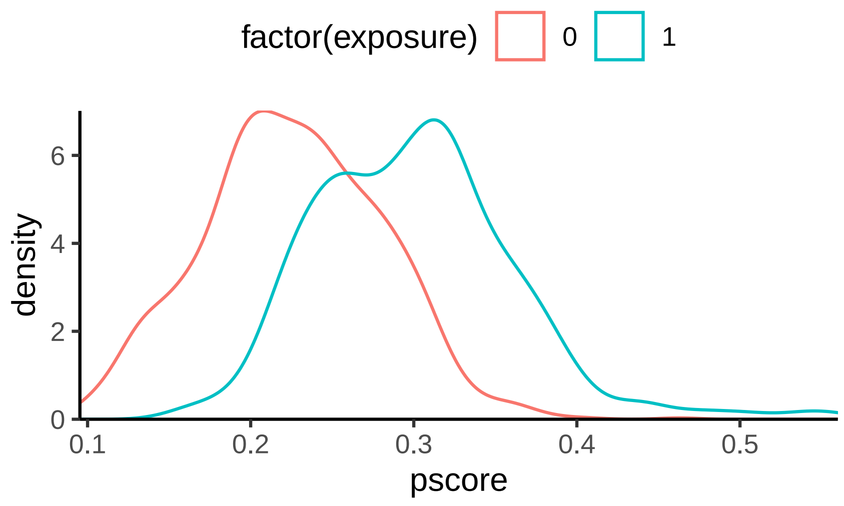
```
summary(propensity_score)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0953  0.1995  0.2408  0.2443  0.2863  0.5600
```

```
# ps overlap plot
ggplot(plap_) + geom_histogram(aes(pscore,
    fill = factor(exposure)), colour = "grey50",
    alpha = 0.75, bins = 50, position = "identity") +
    scale_fill_manual(values = c("blue",
        "orange")) + scale_x_continuous(expand = c(0,
    0)) + scale_y_continuous(expand = c(0,
    0))
```



```
ggsave(here("figures", "ps_overlap_hist-2022_06_02.pdf"),
    width = 15, height = 15, units = "cm")

ggplot(plap_) + geom_density(aes(pscore,
    color = factor(exposure))) + scale_fill_manual(values = c("blue",
    "orange")) + scale_x_continuous(expand = c(0,
    0)) + scale_y_continuous(expand = c(0,
    0))
```



```
ggsave(here("figures", "ps_overlap_dens-2022_06_02.pdf"),
    width = 15, height = 15, units = "cm")

# save outcome predictions for
# diagnosis
outcome_preds <- initial_likelihood$get_likelihoods(tmle_task)$Y

# super learner coefficients for PS
# model
g_fit <- tmle_fit_$likelihood$factor_list[["A"]]$learner
g_fit$fit_object$full_fit$learner_fits$Lrnr_nnls_TRUE
```

```
## [1] "Lrnr_nnls_TRUE"
##                                          lrnrs   weights
## 1:                                   Lrnr_mean 0.1913551
## 2:                               Lrnr_glm_TRUE 0.5800473
## 3:                Lrnr_ranger_500_TRUE_none_1 0.1307639
## 4: Lrnr_glmnet_NULL_deviance_10_1_100_TRUE_FALSE 0.0000000
## 5:                           Lrnr_xgboost_20_1 0.0000000
## 6:             Lrnr_earth_2_3_backward_0_1_0_0 0.0978336
```

```
# super learner coefficients for
# outcome model
```

```
Q_fit <- tmle_fit_$likelihood$factor_list[["Y"]]$learner
Q_fit$fit_object$full_fit$learner_fits$Lrnr_nnls_TRUE
```

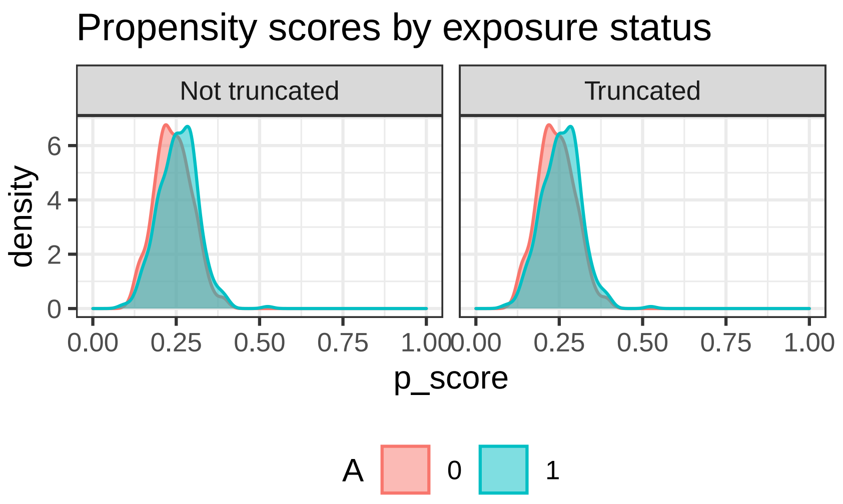
```
## [1] "Lrnr_nnls_TRUE"
##                                            lrnrs    weights
## 1:                                     Lrnr_mean 0.07948012
## 2:                                 Lrnr_glm_TRUE 0.50585733
## 3:                    Lrnr_ranger_500_TRUE_none_1 0.00000000
## 4: Lrnr_glmnet_NULL_deviance_10_1_100_TRUE_FALSE 0.00000000
## 5:                             Lrnr_xgboost_20_1 0.02393456
## 6:                Lrnr_earth_2_3_backward_0_1_0_0 0.39072798
```

3 Using sl3 with AIPW

The procedure for using AIPW is simpler, but there's some important context. If you were able to install the development version of AIPW (from GitHub), you can use `sl3` to estimate the propensity score and outcome model. This is the procedure that I often use, which makes it simple in that I can use the super learner object I create for TMLE:

```
# detach('package:AIPW', unload=TRUE)
remotes::install_github("yqzhong7/AIPW")
library(AIPW)

stacklearner <- Stack$new(lrnr_mean, lrnr_glm,
    lrnr_ranger, lrnr_glmnet, lrnr_xgboost,
    lrnr_earth)

metalearner <- Lrnr_nnls$new(convex = T)

sl.lib <- Lrnr_sl$new(learners = stacklearner,
    metalearner = metalearner)

outcome <- nhefs$wt_delta
exposure <- nhefs$qsmk
```

```
covariates <- nhefs[, c("age", "sbp", "dbp",
    "price71", "tax71", "sex", "income",
    "race")]

set.seed(123)
AIPW_SL <- AIPW$new(Y = outcome, A = exposure,
    W = covariates, Q.SL.library = sl.lib,
    g.SL.library = sl.lib, k_split = 10,
    verbose = FALSE)$fit()$summary(g.bound = 0.025)$plot.p_score()
```

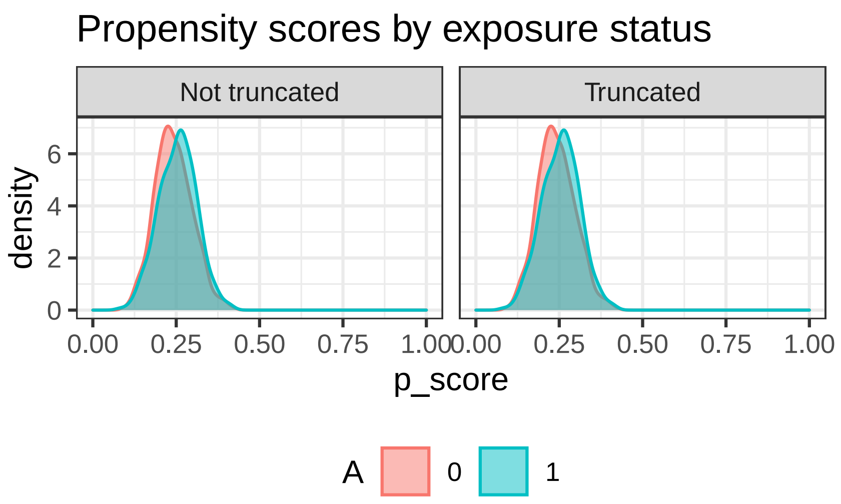


However, if unable to install the development version of AIPW, you will have to use the older version of the Super Learner:

```
library(SuperLearner)

listWrappers()
```

```
##  [1] "SL.bartMachine"      "SL.bayesglm"         "SL.biglasso"
##  [4] "SL.caret"            "SL.caret.rpart"      "SL.cforest"
##  [7] "SL.earth"            "SL.extraTrees"       "SL.gam"
## [10] "SL.gbm"              "SL.glm"              "SL.glm.interaction"
## [13] "SL.glmnet"           "SL.ipredbagg"        "SL.kernelKnn"
## [16] "SL.knn"              "SL.ksvm"             "SL.lda"
## [19] "SL.leekasso"         "SL.lm"               "SL.loess"
## [22] "SL.logreg"           "SL.mean"             "SL.nnet"
## [25] "SL.nnls"             "SL.polymars"         "SL.qda"
## [28] "SL.randomForest"     "SL.ranger"           "SL.ridge"
## [31] "SL.rpart"            "SL.rpartPrune"       "SL.speedglm"
## [34] "SL.speedlm"          "SL.step"             "SL.step.forward"
## [37] "SL.step.interaction" "SL.stepAIC"          "SL.svm"
## [40] "SL.template"         "SL.xgboost"
## [1] "All"
## [1] "screen.corP"           "screen.corRank"        "screen.glmnet"
## [4] "screen.randomForest"   "screen.SIS"            "screen.template"
## [7] "screen.ttest"          "write.screen.template"
```

```
lrnr_mean <- "SL.mean"
lrnr_glm <- "SL.glm"

# ranger learner
lrnr_ranger = create.Learner("SL.ranger",
    tune = list(num.trees = c(250, 500, 1000,
        2000), mtry = c(2, 4, 6), min.node.size = c(50,
        100)))
########################################################################
lrnr_ranger <- NULL
lrnr_ranger$names <- "SL.ranger"  ###########  FLAG! ###########
########################################################################

# glmnet learner
grid_params <- seq(0, 1, by = 0.25)
lrnr_glmnet = create.Learner("SL.glmnet",
    tune = list(alpha = grid_params))
########################################################################
lrnr_glmnet <- NULL
lrnr_glmnet$names <- "SL.glmnet"  ###########  FLAG! ###########
########################################################################

# xgboost learner
lrnr_xgboost = create.Learner("SL.xgboost",
    tune = list(max_depth = c(2, 4, 6, 8),
        eta = c(0.01, 0.1, 0.2), nrounds = c(50,
            100, 500)))
########################################################################
lrnr_xgboost <- NULL
lrnr_xgboost$names <- "SL.xgboost"  ###########  FLAG! ###########
########################################################################

# earth learner
grid_params <- c(2, 3, 4, 5, 6)
lrnr_earth = create.Learner("SL.earth", tune = list(degree = grid_params))
########################################################################
lrnr_earth <- NULL
lrnr_earth$names <- "SL.earth"  ###########  FLAG! ###########
########################################################################

sl.lib <- c(lrnr_mean, lrnr_glm, lrnr_ranger$names,
    lrnr_glmnet$names, lrnr_xgboost$names,
    lrnr_earth$names)

outcome <- nhefs$wt_delta
exposure <- nhefs$qsmk
covariates <- nhefs[, c("age", "sbp", "dbp",
    "price71", "tax71", "sex", "income",
    "race")]

set.seed(123)
AIPW_SL <- AIPW$new(Y = outcome, A = exposure,
    W = covariates, Q.SL.library = sl.lib,
    g.SL.library = sl.lib, k_split = 10,
    save.sl.fit = T, verbose = FALSE)$fit()$summary(g.bound = 0.025)$plot.p_score()
```



```
# stratified_fit()$

# AIPW_SL$libs$Q.fit AIPW_SL$libs$g.fit

print(AIPW_SL$result, digits = 2)
```

```
##                   Estimate    SE 95% LCL 95% UCL    N
## Risk of exposure      0.61 0.027   0.559    0.66  340
## Risk of control       0.47 0.015   0.437    0.50 1054
## Risk Difference       0.14 0.031   0.085    0.21 1394
## Risk Ratio            1.31 0.054   1.180    1.46 1394
## Odds Ratio            1.80 0.126   1.405    2.31 1394
```