Evaluating the Benefits of Sample Splitting for Double Machine Learning

Michael Sarkis, Jack McCarthy

4/24/2022

Double Machine Learning

In this section we will replicate the methods put forth in the paper by Chernozhukov et al., dubbed "CCD-DHNR", which establishes an unbiased Bayesian machine learning framework for treatment effect estimation. This work deals with treatment effect estimation for complex data which has a high number of confounding parameters relative to a low dimensional treatment effect.

Simulated Dataset

First we will generate a toy dataset on which we may test each method. We will generate data according to the following partial-linear model, which relates the treatment linearly to the response and the covariates non-linearly to both the response and the treatment:

$$\begin{aligned} Y_i &= D_i\theta + g_0(X_i) + \epsilon_i, \quad && \epsilon_i \sim N(0,1), \\ D_i &= m_0(X_i) + \tau_i, && \tau_i \sim N(0,1), \end{aligned}$$

where Y is the outcome, D is the treatment, and X is the vector of covariates. $g_0(X)$ and $m_0(X)$ relate the covariates to the value of the response and the treatment respectively. We define these "nuisance" functions to be the following non-linear form:

$$\begin{aligned} g_0(x) &= \sin(x_1) + \cos(x_2) + \sigma(x_3), \\ m_0(x) &= \sin(x_4) + \cos(x_5) + \sigma(x_6), \end{aligned}$$

where $\sigma(x)$ is the sigmoid function:

$$\sigma(x) = \frac{1}{1+e^{-x}}.$$

We also define $X_i \sim N(0, \Sigma)$ where $\Sigma_{jk} = 0.5^{|j-k|}$.

Assuming a continuous response and treatment, N total observations, a p-dimensional covariate vector, and a true value of $\theta = 0.5$, we may generate a toy dataset as follows:

```r
# data dimensions
N = 250
p = 100
theta = 0.5
```

```
# covariance matrix
i_mat <- matrix(rep(1:p, p), p)
j_mat <- matrix(rep(1:p, each=p), p)
Sigma <- 0.5^abs(i_mat - j_mat)

generate_data <- function(N, p, theta, S) {
  # generate covariates
  X <- mvrnorm(n=N, mu=rep(0, p), Sigma=S)

  # generate treatment
  D <- m_0(X) + rnorm(N)

  # generate response
  Y <- D*theta + g_0(X) + rnorm(N)

  data <- as.data.frame(cbind(Y, D, X))
}
```

where we will use the **generate_data** function to obtain distributions on the predicted value of θ through multiple fits to many random datasets.

Naive ML

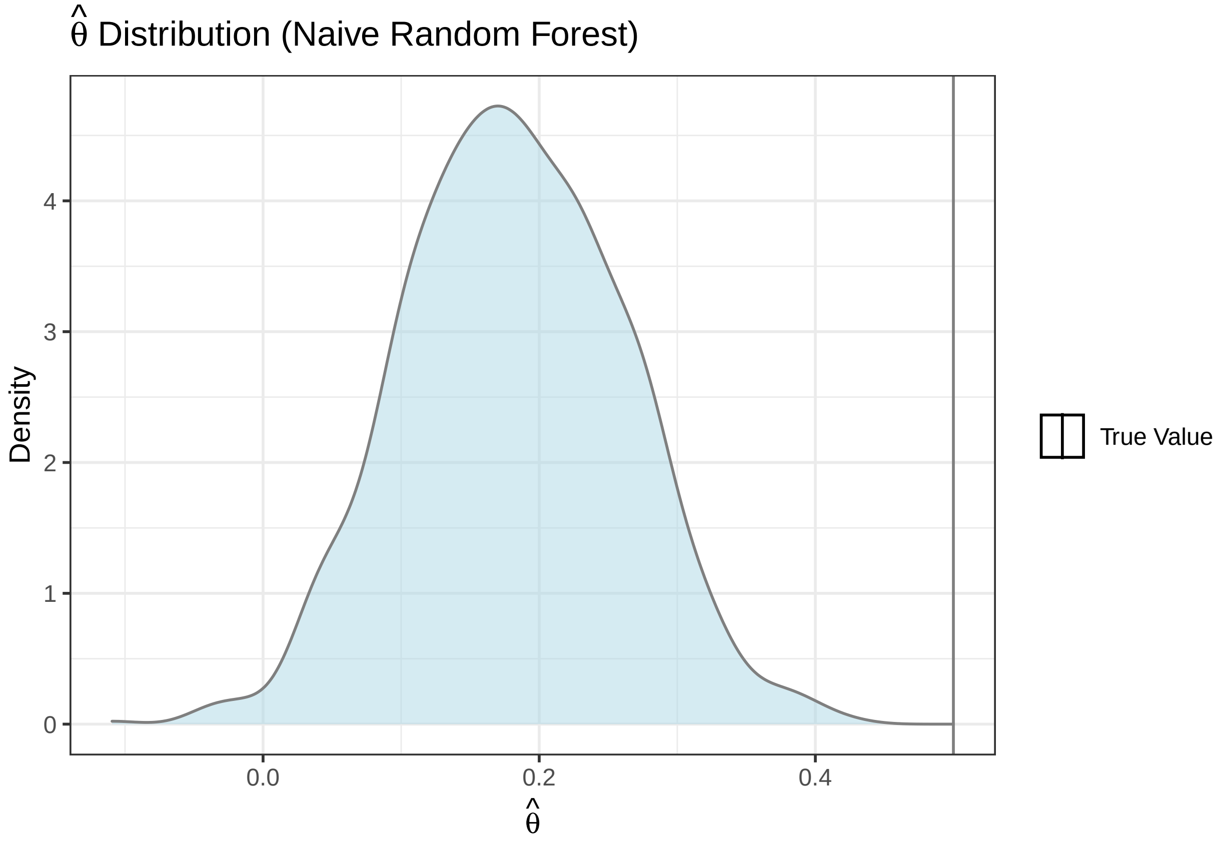
A naive approach to estimating θ would be to estimate $D\theta + g_0(X)$ using some machine learning method. In line with the demonstration in Chernozhukov et al., we will split the samples into two index sets of equal size, S_1 (auxiliary) and S_2 (primary). We will then use the auxiliary set generate the estimate $D\hat{\theta} + \hat{g}_0(X)$, and use the primary set to estimate θ as

$$\hat{\theta} = \left(\frac{1}{n} \sum_{i \in S_2} D_i^2 \right)^{-1} \frac{1}{n} \sum_{i \in S_2} D_i (Y_i - \hat{g}_0(X_i)).$$

We do so for our simulated dataset below using random forest regressors.

```
n_sim <- 1000
theta.est <- numeric(n_sim)

for (i in 1:n_sim) {
  # obtain new data
  data <- generate_data(N, p, theta, Sigma)

  # sample splitting
  idx <- sample.int(N, N/2, replace=F)
  s1 <- data[idx,]
  s2 <- data[-idx,]

  # fit random forest
  rf <- randomForest(Y ~ ., ntree=100, data=s1)

  # obtain theta estimate
  g.hat <- predict(rf, s2)
  theta.est[i] <- with(s2, mean(D * (Y - g.hat)) / mean(D^2))
}

save(theta.est, file="theta_naive.RData")
```

Having obtained a vector of 1000 estimates of $\hat{\theta}$, we may plot the distribution of said estimates to assess whether or not this approach is biased.



As we can see, the estimates of θ are quite distant from the true value of 0.5, indicating this naive ML method for estimating treatment effects is biased. Double machine learning was introduced to address this issue.

Double ML

Instead of estimating $D\theta + g_0(X)$, we may instead first estimate $V = D - m_0(X)$ by regressing D on X and obtaining the residuals. This will give the estimate $\hat{V} = D - \hat{m}_0(X)$, which we use in conjunction with $\hat{g}_0(X)$ to obtain the following estimate of θ:

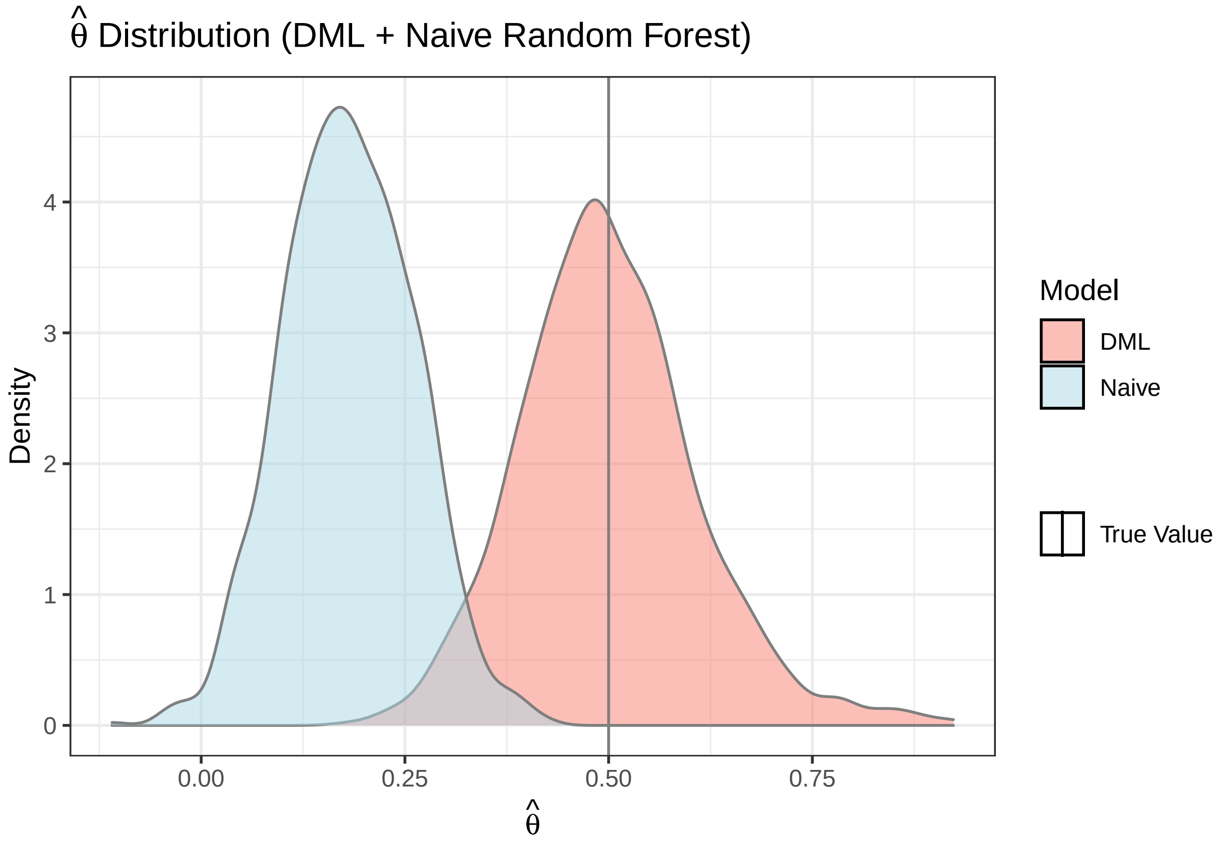
$$\hat{\theta} = \left(\frac{1}{n} \sum_{i \in S_2} \hat{V}_i D_i \right)^{-1} \frac{1}{n} \sum_{i \in S_2} \hat{V}_i (Y_i - \hat{g}_0(X_i)).$$

This method is implemented in R below, again with random forest regressors.

```
theta.dml <- numeric(n_sim)
```

```r
for (i in 1:n_sim) {
  # obtain new data
  data <- generate_data(N, p, theta, Sigma)

  # sample splitting
  idx <- sample.int(N, N/2, replace=F)
  s1 <- data[idx,]
  s2 <- data[-idx,]

  # fit treatment
  rf.D <- randomForest(D ~ . - Y, ntree=100, data=s1)

  # fit response
  rf.Y <- randomForest(Y ~ . - D, ntree=100, data=s1)

  # obtain theta estimate
  V.hat <- s2$D - predict(rf.D, s2)
  g.hat <- predict(rf.Y, s2)
  theta.dml[i] <- with(s2, mean(V.hat * (Y - g.hat)) / mean(V.hat * D))
}
```

Having accumulated 1000 samples of $\hat{\theta}$, we may again investigate their distribution in comparison to the previous naive method.



This time, the distribution appears centered directly around the true value of $\theta = 0.5$, which indicates that DML was able to avoid bias in treatment effect estimation.

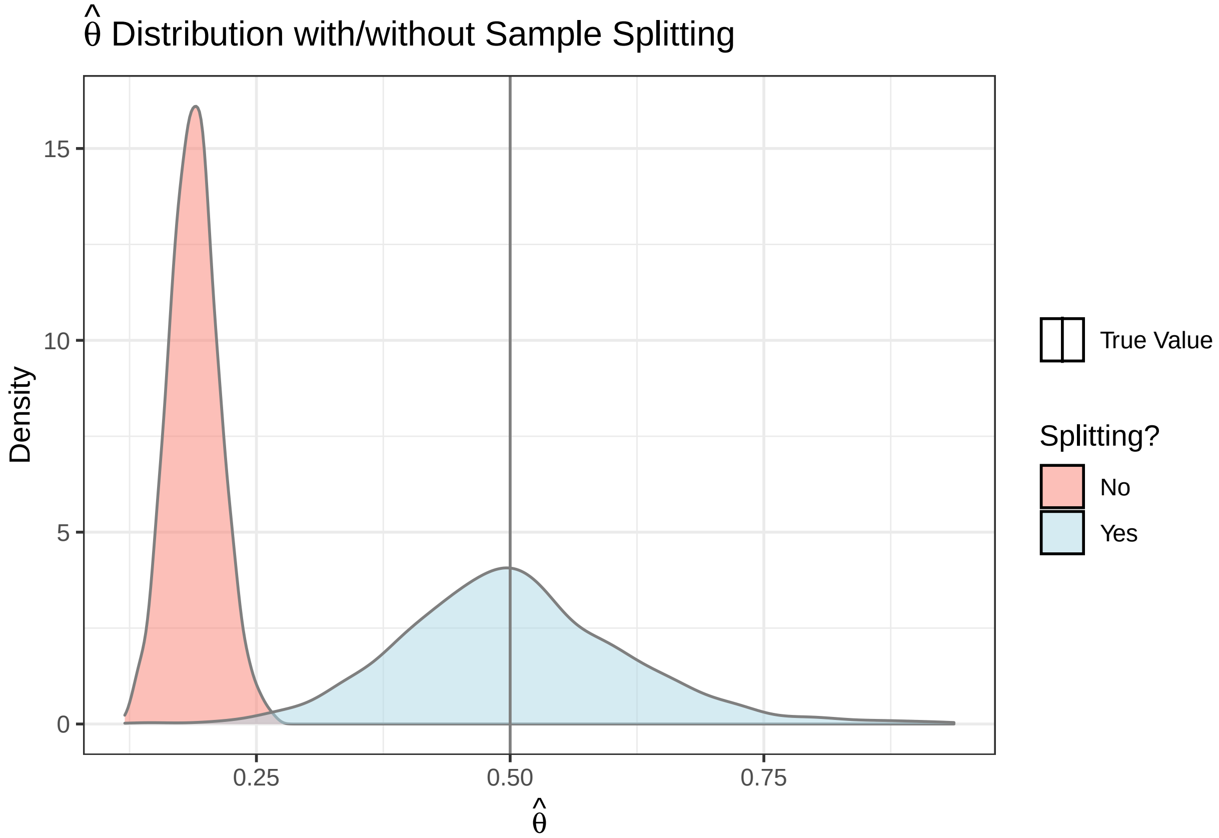
We have therefore been able to replicate the results of Chernozhukov et al., demonstrating the utility of double machine learning in producing unbiased treatment effect estimates. Notably, the examples above both made use of sample splitting in order to produce these estimates. The actual benefit of this intermediate step will be the subject of investigation in the following section.

Sample Splitting

Sample splitting entails dividing the data into two groups, say S_1 and S_2 for function approximation and effect estimation respectively. This was done in both examples above in line with the methods introduced in Chernozhukov et al., but now we will investigate how important this step is to obtaining unbiased treatment effect estimates.

Value of Sample Splitting

To start, we will repeat the above DML simulation with and without sample splitting. When not using sample splitting, both the function approximation and effect estimation will be done using the entire dataset. The following plot shows the distributions of $\hat{\theta}$ for each method:



It is clear from these distributions that bias is not avoided by double machine learning when we neglect sample splitting. We therefore require that sample splitting is used in the DML process.

Cross-Fitting

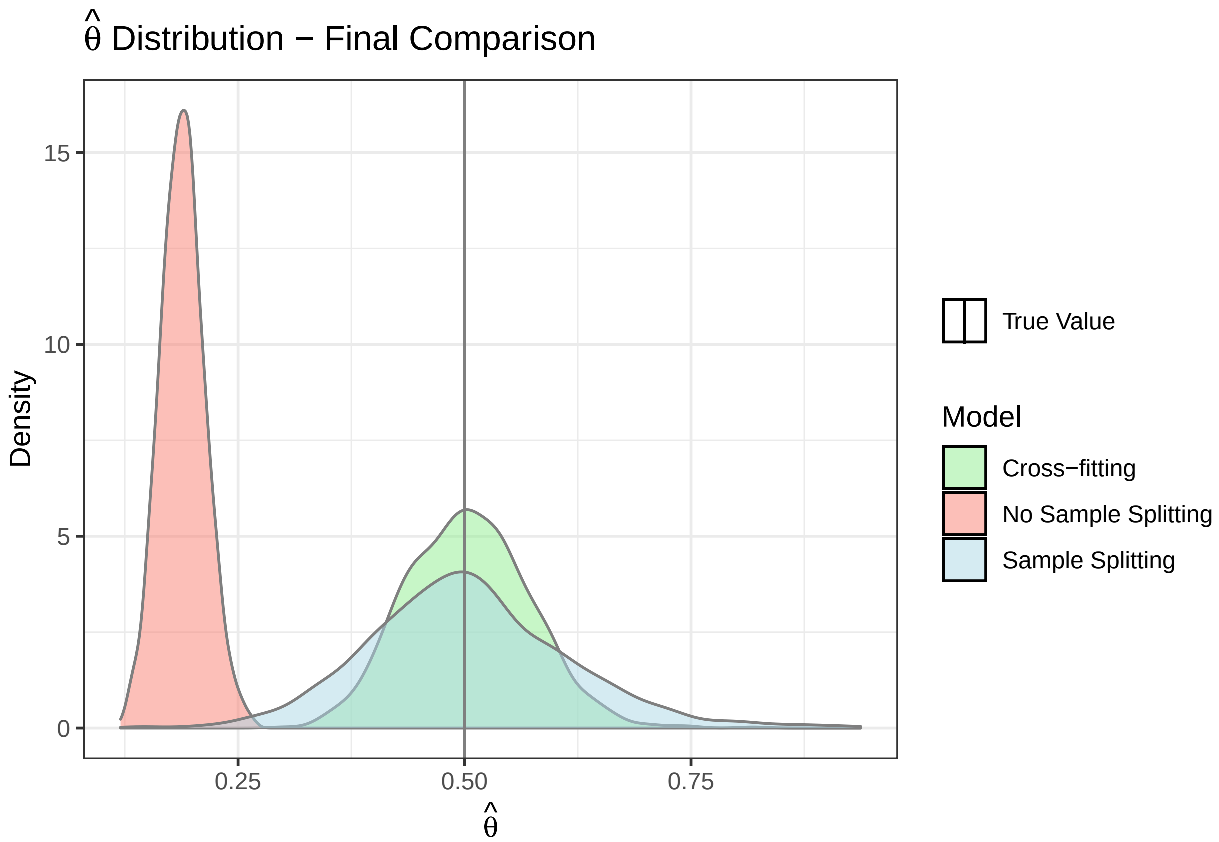
Cross-fitting is one more technique that we may apply in double ML to achieve unbiased treatment effect estimates. Cross-fitting entails taking the weighted mean of the estimates generated via sample splitting, with the first estimate training on the auxiliary set and estimating on the primary set, and the second estimate training on the primary set and estimating on the auxiliary set. Put more rigorously, the full process with cross-fitting is the following (where superscripts denote the set used to obtain estimates):

1. Split data into sets S_1 and S_2
2. Obtain $\hat{g_0}^{(1)}(X)$ and $\hat{g_0}^{(2)}(X)$ via S_1 and S_2 respectively
3. Obtain residuals $\hat{V}^{(1)} = D^{(1)} - \hat{m_0}^{(1)}(X^{(1)})$ and $\hat{V}^{(2)} = D^{(2)} - \hat{m_0}^{(2)}(X^{(2)})$
4. Estimate $\hat{\theta}^{(1)}$ and $\hat{\theta}^{(2)}$ as shown in the previous section
5. Generate final estimate $\hat{\theta} = \frac{|S_1|\hat{\theta}^{(1)}+|S_2|\hat{\theta}^{(2)}}{|S_1|+|S_2|}$

We implement this process in R below.

```r
theta.cross <- numeric(n_sim)

for (i in 1:n_sim) {
  # obtain new data
  data <- generate_data(N, p, theta, Sigma)

  # sample splitting
  idx <- sample.int(N, N/2, replace=F)
  s1 <- data[idx,]
  s2 <- data[-idx,]

  # fit treatment
  rf.D.ss.1 <- randomForest(D ~ . - Y, ntree=100, data=s1)
  rf.D.ss.2 <- randomForest(D ~ . - Y, ntree=100, data=s2)

  # fit response
  rf.Y.ss.1 <- randomForest(Y ~ . - D, ntree=100, data=s1)
  rf.Y.ss.2 <- randomForest(Y ~ . - D, ntree=100, data=s2)

  # obtain theta estimate 1
  V.hat.ss <- s2$D - predict(rf.D.ss.1, s2)
  g.hat.ss <- predict(rf.Y.ss.1, s2)
  theta.1 <- with(s2, mean(V.hat.ss * (Y - g.hat.ss)) / mean(V.hat.ss * D))

  # obtain theta estimate 2
  V.hat.ss <- s1$D - predict(rf.D.ss.2, s1)
  g.hat.ss <- predict(rf.Y.ss.2, s1)
  theta.2 <- with(s1, mean(V.hat.ss * (Y - g.hat.ss)) / mean(V.hat.ss * D))

  # obtain final estimate
  theta.cross[i] <- mean(c(theta.1, theta.2))
}
```

Finally, we may compare the methods of cross-fitting, sample splitting, and no sample splitting by evaluating the distributions of the estimates they generate.



It is clear that we see marked improvements with the addition of each technique. Sample splitting eliminates bias in the treatment effect estimates and cross-fitting improves the precision around the true value (as seen in the lower dispersion in the distribution of $\hat{\theta}$). As such, we have not only demonstrated the necessary value of sample splitting, but the additional benefit of cross-fitting in obtaining treatment effect estimates.

Conclusion

Double machine learning is a powerful technique for treatment effect estimation with high-dimensional confounders, such as the case addressed above. While a more naive approach would produce biased estimates, DML is able to side-step this issue by utilizing a two-pronged machine learning process, in which one model is built to predict the response and one is built to predict the treatment (both based on the covariates). However, as demonstrated above, the benefits of DML will not be realized unless used in conjunction with sample splitting. Finally, we showed that estimates with lower uncertainty could be obtained with the implementation of cross-fitting.