Estimating Heterogeneous Exposure Effects using CL-BART

Jacob Englert

18 April 2024

Setup

First, ensure you have loaded the `clbart` package into your R session. We will also make use of the `tidyverse` package for some data manipulation and plotting.

```
# devtools::install_github('jacobenglert/clbart')
library(clbart)
library(tidyverse)
```

Simulate Data

To mimic the case-crossover design, we will follow 10,000 individuals for 1 year. We will consider 10 time-invariant covariates `w`, and 5 time-varying covariates `x`.

```
# Setup
set.seed(1)
n       <- 10000
start   <- as.Date("2023-01-01")
end     <- as.Date("2023-12-31")
dates   <- seq.Date(start, end, by = 'day')
t       <- length(dates)
p_w     <- 10
p_x     <- 5

# Simulate shared exposure time-series
z <- rnorm(t, sin(seq(0, 2 * pi, length.out = t)))

# Simulate independent time-varying confounders
x <- replicate(p_x, runif(t))
colnames(x) <- paste0('X', 1:p_x)

# Simulate independent time-invariant moderators
w <- replicate(p_w, runif(n))
colnames(w) <- paste0('W', 1:p_w)

# Construct data frame of time-varying variables
xz <- data.frame(x, Z = z, date = dates) |>
  mutate(year = year(date), month = month(date), dow = wday(date))

# Data frame of all individuals at all times
data <- cross_join(data.frame(strata = 1:n, w), xz)
```

We then simulate the outcome based on coefficients. For the exposure effect, we use a modification of the classic Friedman function [Friedman, 2001].

```
# Heterogeneous exposure log odds-ratios
f0 <- function(x) 10 * sin(pi*x[,1]*x[,2]) + 20*(x[,3]-.5)^2 + 10*x[,4] + 5*x[,5]
f <- function(x) (f0(x) - 14) / 15
tau <- f(data[,colnames(w)])

# Fixed effects
alpha <- -8 # intercept
beta <- log(c(0.5, 0.8, 1.0, 1.2, 2.0)) # confounder log odds-ratios

# Calculate probabilities
expit <- function(x) exp(x) / (1 + exp(x))
p <- expit(alpha + as.matrix(data[,colnames(x)]) %*% beta + tau * data$Z)[,1]

# Store probabilities and log-odds in dataset
data$tau <- tau
data$p <- p

# Generate outcome
y <- rbinom(n * t, 1, data$p)
```

Implement the Case-Crossover Design

We then filter down to the cases only (as this is what is typically observed), and implement the time-stratified case-crossover design. This involves matching each observed case to the other days in the calendar month which share the same day of the week.

```
# Implement time-stratified case-crossover design
cco <- data[which(y == 1),] |>  # subset cases
  mutate(strata = row_number()) |>  # assign unique ID (strata) to each case
  select(-all_of(c(colnames(x), 'Z'))) |>
  left_join(xz, by = c('year', 'month', 'dow'), relationship = 'many-to-many') |>
  mutate(Y = ifelse(date.x == date.y, 1, 0)) |> # create case variable
  select(-date.x) |>
  rename(date = date.y)
```

Fitting the Model

CL-BART seeks to estimate parameters from the following likelihood:

$$\pi(\mathbf{y} \mid \tau(\cdot), \beta) = \prod_{i=1}^{n} \pi(\mathbf{y}_i \mid \tau(\cdot), \beta) = \prod_{i=1}^{n} \frac{\exp\left\{\mathbf{x}_{it_i}^T \beta + \tau(\mathbf{w}_i) z_{it_i}\right\}}{\sum_{t \in \mathcal{W}_i} \exp\left\{\mathbf{x}_{it}^T \beta + \tau(\mathbf{w}_i) z_{it}\right\}}$$

where $\mathbf{y}$, $\mathbf{x}$, $\mathbf{w}$, and z are the matrices/vectors we have generated thus far. The referent window $\mathcal{W}_i$ represents all observed times for individual i. We incorporate this using the `strata` vector, where `strata` represents the groupings of observations by individual.

```
# Prepare CCO data for model
w <- cco[colnames(w)]
x <- cco[colnames(x)]
```

```r
z <- cco$Z
y <- cco$Y
strata <- cco$strata
```

Cl-BART uses Bayesian additive regression trees [Chipman et al., 2010] to estimate the exposure moderating function $\tau(\cdot)$, which has several options for hyperparameter settings. Fitting a `clbart` model requires specifying the hyperparameters via `Hypers()` and the options via `Opts()`. For this example we run a 20 tree model for 2500 MCMC iterations, saving the final 500 iterations. We also set the `update_s` and `update_alpha` options to `TRUE`, which improves variable selection via the Dirichlet prior modification introduced in Linero [2018].

```r
# Specify hyperparameters for fitting clbart model
hypers <- Hypers(num_tree = 20, k = 1)
opts <- Opts(update_s = TRUE, update_alpha = TRUE,
             num_burn = 2000, num_thin = 1, num_save = 500)

# Fit clbart model
fit <- clbart(w, x, y, z, strata, hypers, opts)
```

Summarizing Model Results

Once the model is fit (make take a while), we can examine it.

```r
# Obtain a brief summary of the model fit
fit_sum <- summary(fit)

# Check beta (confounder) estimates
fit_sum$beta_stats
#>    post.mean post.2.5 post.97.5
#> X1     -0.87    -1.04     -0.66
#> X2     -0.27    -0.47     -0.04
#> X3     -0.17    -0.39      0.03
#> X4      0.31     0.09      0.50
#> X5      0.82     0.63      1.01
beta
#> [1] -0.6931472 -0.2231436  0.0000000  0.1823216  0.6931472
```

The estimated β coefficients are similar to what we specified above.

More interestingly, we can plot the predictions for $\tau(\mathbf{w})_i$, stored in `fit$lambda_est`. These are the posterior mean BART predictions. By default, only the posterior means are returned. However, setting `store_lambda = TRUE` inside of `Opts()` will keep the entire posterior distribution of all predictions.

```r
# Plot individual BART predictions versus the true association
plot(fit$lambda_est ~ cco$tau,
     xlab = 'Truth', ylab = 'Prediction')
abline(a = 0, b = 1, col = 'red', lty = 2)
```

We can also view the proportion of splits based on each moderating covariate, the probability of splitting based on each covariate, and the posterior inclusion probability for each covariate.

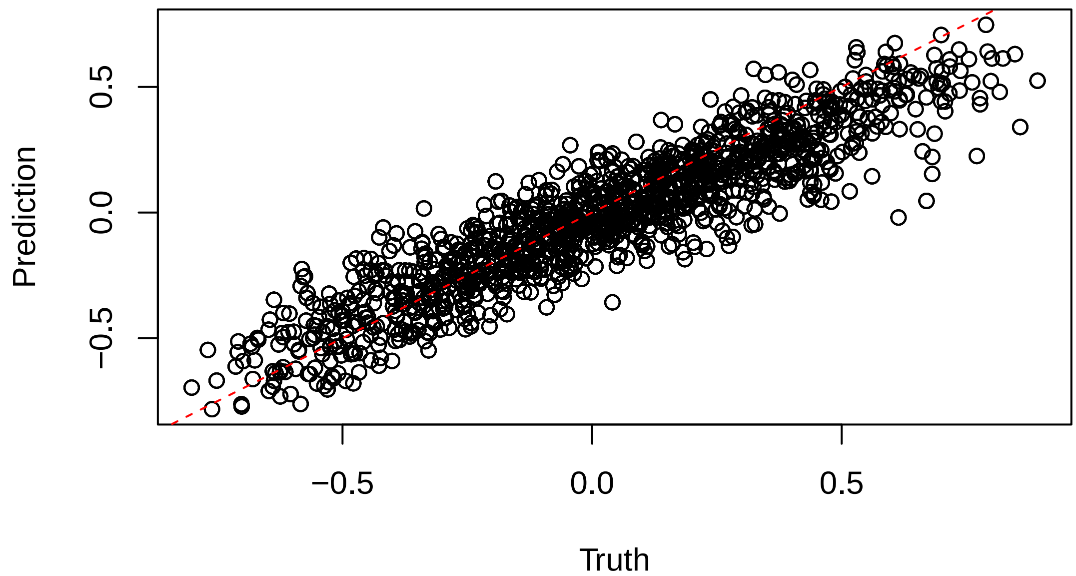

Figure 1: CL-BART Predictions vs. Truth

```
# Check variable importance measures
fit_sum$var_imp
#>     split.props split.probs  PiP
#> W1         0.15        0.13 1.00
#> W2         0.18        0.15 1.00
#> W3         0.16        0.14 0.99
#> W4         0.14        0.12 1.00
#> W5         0.10        0.10 0.92
#> W6         0.06        0.07 0.71
#> W7         0.05        0.07 0.69
#> W8         0.05        0.08 0.65
#> W9         0.07        0.08 0.81
#> W10        0.04        0.07 0.63
```

Here we see that the first 5 covariates are split upon more often and included more often.

We can visualize the marginal effect of each moderator using accumulated local effects [Apley and Zhu, 2020]. For Bayesian models, we can compute the ALE using the `bayes_ale()` function from the `pdpd` R package available here.

```
library(pdpd)
f_hat <- function(w) predict(fit, list(w = w), type = "bart", posterior = TRUE)
firsts <- match(unique(strata), strata)

# Compute ALE
marg_ale <- lapply(colnames(w),
```

```
                \(v) bayes_ale(w[firsts,], f_hat, vars = v, k = 40, f = f))
```

Now we plot the marginal ALE functions.

```
marg_ale |>
  bind_rows() |>
  ggplot(aes(x = x, y = est, ymin = lcl, ymax = ucl)) +
  geom_line() +
  geom_line(aes(y = truth), col = 'red', lty = 2) +
  geom_ribbon(alpha = 0.2) +
  facet_wrap(~factor(var, colnames(w)), nrow = 2) +
  theme_bw() +
  labs(x = 'Covariate Value',
       y = 'Posterior Mean ALE w/ 95% Credible Interval')
```

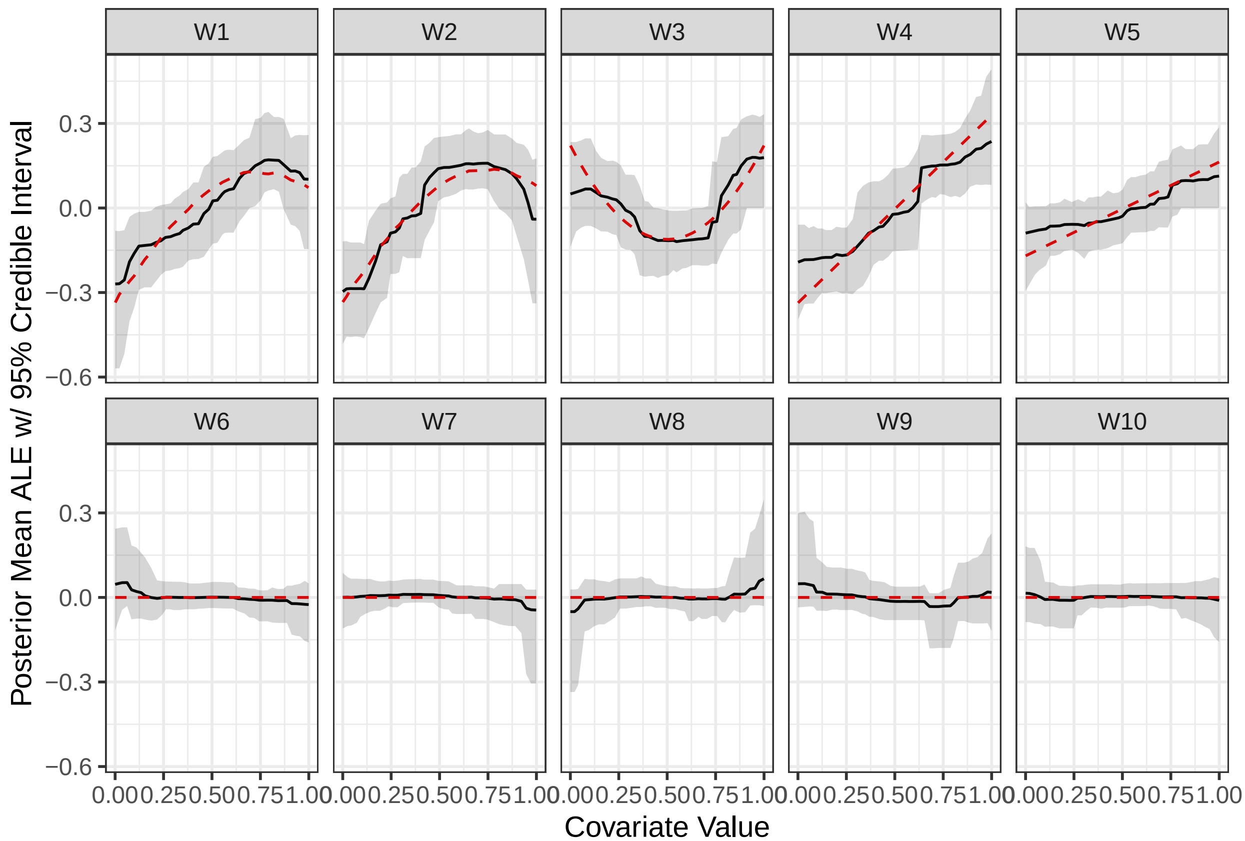

Figure 2: Accumulated Local Effects Plots for Effect Moderators

Even with only 20 trees, the model does a fairly good job of approximates the true functional forms of the important effect moderators, and accurately estimates null effects for the unimportant covariates.

Diagnostics

Posterior samples for most quantities are available in the model. This makes it easy to assess trace plots, such as for the average BART prediction.

```r
plot(fit$lambda_mean_overall, type = 'l',
     xlab = 'Posterior Sample Index', ylab = 'Average Prediction')
```

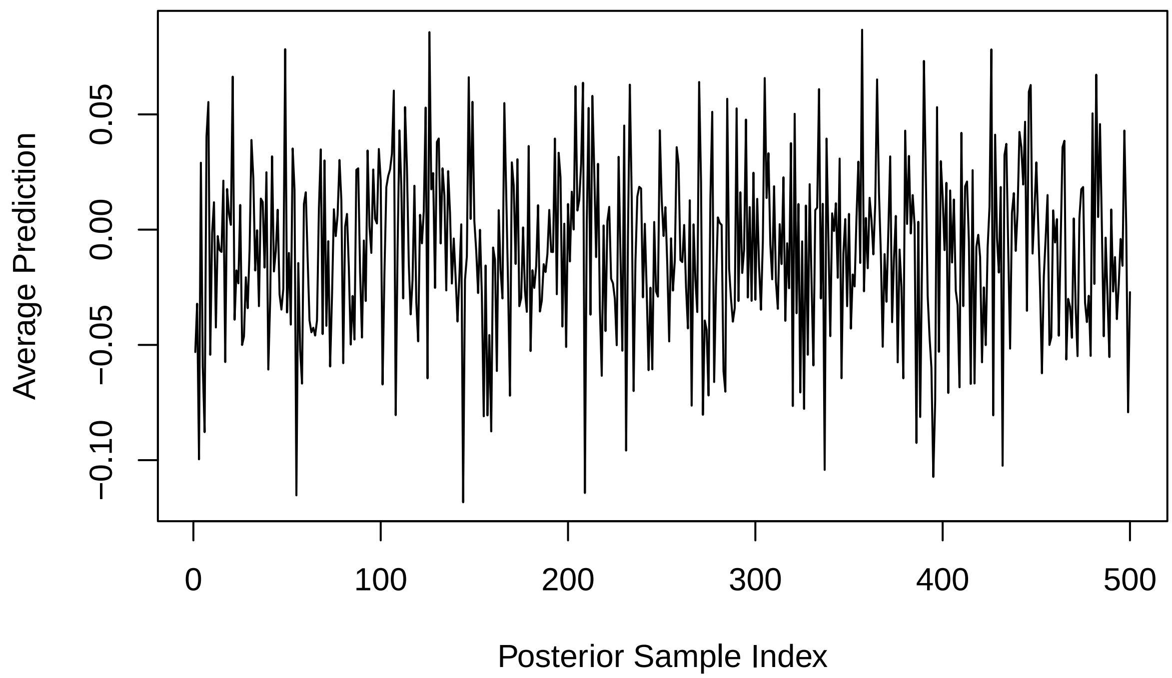

Figure 3: Trace Plot of Average BART Prediction

We can also extract the model WAIC. Which may be useful to compare multiple CL-BART models.

```r
fit$WAIC
#> [1] 3817.866
```

References

Daniel W. Apley and Jingyu Zhu. Visualizing the Effects of Predictor Variables in Black Box Supervised Learning Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 82(4):1059–1086, September 2020. ISSN 1369-7412, 1467-9868. doi: 10.1111/rssb.12377. URL https://academic.oup.com/jrsssb/article/82/4/1059/7056085.

Hugh A. Chipman, Edward I. George, and Robert E. McCulloch. BART: Bayesian additive regression trees. *Ann. Appl. Stat.*, 4(1), March 2010. ISSN 1932-6157. doi: 10.1214/09-AOAS285. URL https://projecteuclid.org/journals/annals-of-applied-statistics/volume-4/issue-1/BART-Bayesian-additive-regression-trees/10.1214/09-AOAS285.full.

Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 29(5), October 2001. ISSN 0090-5364. doi: 10.1214/aos/1013203451. URL https:

//projecteuclid.org/journals/annals-of-statistics/volume-29/issue-5/Greedy-function-approximation-A-gradient-boosting-machine/10.1214/aos/1013203451.full.

Antonio R. Linero. Bayesian Regression Trees for High-Dimensional Prediction and Variable Selection. *Journal of the American Statistical Association*, 113(522):626–636, April 2018. ISSN 0162-1459, 1537-274X. doi: 10.1080/01621459.2016.1264957. URL https://www.tandfonline.com/doi/full/10.1080/01621459.2016.1264957.