Neyman Orthogonality for the Partially Linear Regression Model

PLSC 40601

2024-04-25

Example

$$Y = D\theta_0 + g_0(X) + U, \qquad \mathrm{E}[U|X, D] = 0,$$

$$D = m_0(X) + V, \qquad \mathrm{E}[V|X] = 0$$

- Y is the dependent variable.
- θ_0 is the causal parameter of interest.
- D is the binary treatment assignment.
- X is the covariate matrix.
- $g(X)$ is a nuisance function describing how X affects Y.
- $m(X)$ is a nuisance function describing how X affects D.

The associated score function for (1.5) is given by:

$$\psi(W; \theta, g) = (Y - g(X) - \theta D) \times (D - m(X))$$

The score function describes how the likelihood of observing the given data changes as each parameter is varied slightly. Section 2 gives some different ways we can construct score functions. At the true value of the parameter, the expectation of score function is zero. So we want to find the value of θ that minimizes the score function. We do this by choosing a $\check{\theta}$ that minimizes the empirical expectation of the score function, i.e., the average score across all observations.

Here, we estimate the nuisance functions $\hat{g}(X)$ and $\hat{m}(X)$ and then plug them into the estimating equation for θ:

$$\check{\theta} = \frac{\sum_{i=1}^{n}(Y_i - \hat{g}(X_i))(D_i - \hat{m}(X_i))}{\sum_{i=1}^{n} D_i(D_i - \hat{m}(X_i))}$$

We can check that the empirical expectation of the score function is zero (or approximately zero) by calculating the average score across all observations at the selected value of $\check{\theta}$.

$$\mathrm{E}_n[\psi(W; \theta, g)] = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{g}(X_i) - \check{\theta}D_i)(D_i - \hat{m}(X_i)) \approx 0$$

The Neyman orthogonality condition is given by:

$$\partial_\eta \mathrm{E}[\psi(W; \theta_0, g)]|_{\eta=\eta_0} = 0$$

Neyman orthogonality implies that the score function used to estimate a parameter of interest is insensitive to small perturbations in the nuisance parameter estimates. The "orthogonality" in the term comes from the idea that the partial derivative (or Gateaux derivative in functional spaces) of the expected value of the score function with respect to the nuisance parameter(s) equals zero at the true value(s) of the parameter(s).

This implies that the score function is orthogonal (in the sense of having zero covariance) to perturbations in nuisance parameters.

The partial derivatives of the score function with respect to the nuisance parameters are:

$$\frac{\partial \psi}{\partial g} = -(D - m(X))$$

$$\frac{\partial \psi}{\partial m} = -(Y - g(X) - \theta D)$$

The expectation conditions for orthogonality are:

$$\mathrm{E}\left[\frac{\partial \psi}{\partial g}\right] = 0 \quad \text{and} \quad \mathrm{E}\left[\frac{\partial \psi}{\partial m}\right] = 0$$

This implies that $m(X)$ should be chosen such that it is the expected value of D given X (i.e., $m(X) = \mathrm{E}[D|X]$), and $g(X)$ should correctly specify $\mathrm{E}[Y - \theta D|X]$. When evaluated at the true values of the nuisance parameters, this is given from the problem definition.

Code

```
# Load necessary libraries
library(glmnet) # for lasso
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-7
```

```
# Set seed for reproducibility
# set.seed(60637)

# Functions
fit_g <- function(X, resid) cv.glmnet(X, resid, alpha = 1)
fit_m <- function(X, D) glm(paste0('Y ~ ', paste0('X', 1:p, collapse = ' + ')),
                            family = "binomial", data.frame(Y = D, X))

## Simulate data ----
n <- 1e5 # Number of observations
p <- 10 # Number of covariates
nfolds <- 5 # Folds for cross fitting
folds <- sample(rep(1:nfolds, each = n / nfolds))

X <- matrix(rnorm(n * p), ncol = p)
beta_X <- runif(p, -1, 1)
gamma_X <- runif(p, -1, 1)
U <- rnorm(n, sd = 1)

g0 <- X %*% beta_X  # True g(X)
m0 <- 1/(1 + exp(-X %*% gamma_X))  # True m(X)
theta0 <- 0.5 # True theta
D <- rbinom(n, 1, m0)
Y <- D * theta0 + g0 + U
```

```r
# Estimation ----
g_hat_model <- m_hat_model <- vector("list", nfolds)
theta_naive_est <- theta_neyman_est <- g_hat_est <- m_hat_est <- V_hat_est <- numeric(n)

#* Naive estimation ----
# Fit g_hat using iterative methods where we alternate between estimating g_hat and theta
# (we won't use this theta for estimation)
for (k in 1:nfolds) {
  train_indices <- (folds != k)
  test_indices <- (folds == k)

  # Alternating minimization for g_hat
  convergence_threshold <- 1e-5
  max_iter <- 100
  iter <- 1
  changes <- Inf

  # Estimate theta given a starting g_hat
  g_hatX <- mean(Y[train_indices])
  theta_hat <- sum((Y[train_indices] - g_hatX) * D[train_indices]) / sum(D[train_indices]^2)

  while (iter <= max_iter && changes > convergence_threshold) {

    # Calculate residuals
    resid <- Y[train_indices] - D[train_indices] * theta_hat

    # Refit g_hat
    g_hat_model[[k]] <- fit_g(X[train_indices, ], resid)

    # Check convergence
    g_hatX_new <- predict(g_hat_model[[k]], s = "lambda.min",
                          newx = X[train_indices, ])
    changes <- sum((g_hatX - g_hatX_new)^2)
    iter <- iter + 1

    # Estimate theta given g_hat for the next iteration
    g_hatX <- g_hatX_new
    theta_hat <- sum((Y[train_indices] - g_hatX) * D[train_indices]) / sum(D[train_indices]^2)

  }

  # Save g_hat estimates conditional on X using cross-fit models
  g_hat_est[test_indices] <- predict(g_hat_model[[k]],
                                     s = "lambda.min",
                                     newx = X[test_indices, ])

  # Calculate theta estimate for each fold from (1.3)
  theta_naive_est[test_indices] <- {sum((Y[test_indices] - g_hat_est[test_indices]) *
                                          D[test_indices]) / sum(D[test_indices]^2)}
}


# Aside: Check that (1.3) gives the same result as the linear estimate
```

```r
sum((Y - g_hat_est) * D) / sum(D^2)
```

```
## [1] 0.4794493
```

```r
coef(lm(Y - g_hat_est ~D-1))
```

```
##         D
## 0.4794493
```

```r
# Score
score_naive <- (Y - g_hat_est - theta_naive_est * D) * D
mean(score_naive) # should be zero by design
```

```
## [1] 1.078183e-15
```

```r
# Estimate + standard error
mean(theta_naive_est)
```

```
## [1] 0.4794633
```

```r
sd(score_naive) / sqrt(n)
```

```
## [1] 0.002239904
```

```r
# Alternate variance estimation
sqrt(mean((Y - g_hat_est - theta_neyman_est*D)^2*D^2)/(mean(D^2)*mean(D^2)))/sqrt(n)
```

```
## [1] 0.004953692
```

```r
#* Neyman orthogonal estimation ----

# Fit m_hat in each of the folds
for(k in 1:nfolds){
  train_indices <- (folds != k)
  test_indices <- (folds == k)

  m_hat_model[[k]] <- fit_m(X[train_indices,],
                            D[train_indices])

  # Save m_hat estimates conditional on X using cross-fit models
  m_hat_est[test_indices] <- predict(m_hat_model[[k]],
                                     newdata =
                                       data.frame(X[test_indices,]),
                                     type = "response")

  # Save V_hat estimates
  V_hat_est[test_indices] <- D[test_indices] - m_hat_est[test_indices]

  # Get theta estimate for each fold from (1.5)
  theta_neyman_est[test_indices] <- {
    sum((Y[test_indices] - g_hat_est[test_indices]) *
          V_hat_est[test_indices]) / sum(D[test_indices]*V_hat_est[test_indices])}
}

# Score
score_neyman <- (Y - g_hat_est - theta_neyman_est * D)*(D - m_hat_est)
mean(score_neyman) # should be zero by design
```

```
## [1] 2.782519e-16
```

```
# Score evaluated at truth
# (Y - g0 - theta0 * D)*(D - m0)
sum((Y - g0) * (D-m0)) / sum(D*(D-m0))
```

```
## [1] 0.484031
```

```
# Estimate + standard error
mean(theta_neyman_est)
```

```
## [1] 0.4841574
```

```
sd(score_neyman)/sqrt(n)
```

```
## [1] 0.001311774
```

```
# Linear estimate
summary(lm(Y - g_hat_est ~V_hat_est-1))
```

```
##
## Call:
## lm(formula = Y - g_hat_est ~ V_hat_est - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.9885 -0.4397  0.2384  0.9237  4.7527
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## V_hat_est 0.484090   0.007933   61.03   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.039 on 99999 degrees of freedom
## Multiple R-squared:  0.03591,    Adjusted R-squared:  0.0359
## F-statistic:  3724 on 1 and 99999 DF,  p-value: < 2.2e-16
```

```
# Alternate variance estimation
sqrt(mean((Y - g_hat_est - theta_neyman_est*D)^2*V_hat_est^2)/(mean(V_hat_est^2)*mean(V_hat_est^2)))/sq
```

```
## [1] 0.0076477
```