Homework 5 Solutions

Additional Question 1

In this question, we will estimate the ATT of a job training program using a number of different techniques that we discussed in class.

One thing to note: for regression, regression adjustment, propensity score re-weighting, and doubly robust, we should get exactly the same answers, but we may be get different estimates when we use machine learning, due to sample splitting, choosing tuning parameters, etc. The same is true for the bootstrap standard errors—they should be broadly similar, but we would not expect that mine and yours will be literally the same number.

Part (a)

To estimate the ATT using regression adjustment, we first need to calculate $\hat{\beta}$ from the regression of Y on X using untreated observations only. Once we have this estimate, we can compute

$$\widehat{ATT} = \frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i'\hat{\beta}$$

```
# load packages used later
library(pbapply)
library(randomForest)

# load data
data <- as.data.frame(haven::read_dta("jtrain_observational.dta"))
Y <- data$re78
D <- data$train
p <- mean(D)
n <- nrow(data)
X <- model.matrix(~age + educ + black + hisp + married + re75 + unem75, data=data)
# run regression using untreated observations only
bet <- solve(t(X)%*%(X*as.numeric((1-D)/p)))%*%t(X)%*%as.numeric(Y*(1-D)/p)

att1 <- mean(D*Y/p)
att2 <- sum(apply(D*X/p,2,mean)*as.numeric(bet))
att <- att1 - att2

# report estimate of att
round(att,3)
```

```
[1] 0.859
```

The outcome is in 1000's of dollars, so this indicates that we are estimating that job training increased yearly earnings by $859.

As a side-comment, it's not immediately clear if this should be interpreted as a large effect or not. One way to think about this is to compute: $ATT/\mathbb{E}[Y(0)|D=1]$ (i.e., the relative size of the ATT compared to what the average outcome would have been absent the treatment). Further,

notice that this is equal to $ATT/(\mathbb{E}[Y|D=1]-ATT)$ (which holds by adding and subtracting $\mathbb{E}[Y(1)|D=1]$ in the denominator). If we compute this, we get that we have estimated that yearly earnings as about 16% higher from job training relative to what they would have been in the absence of job training.

Part (b)

Notice that

$$\begin{aligned}
\sqrt{n}(\widehat{ATT}-ATT) &= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i'\hat{\beta}\right) - \sqrt{n}\left(\mathbb{E}\left[\frac{D}{p}Y\right] - \mathbb{E}\left[\frac{D}{p}X'\right]\beta\right) \\
&= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \mathbb{E}\left[\frac{D}{p}Y\right]\right) - \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i' - \mathbb{E}\left[\frac{D}{p}X'\right]\right)\hat{\beta} \\
&\qquad - \mathbb{E}\left[\frac{D}{p}X'\right]\sqrt{n}(\hat{\beta}-\beta) \\
&= \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}Y_i - \mathbb{E}\left[\frac{D}{p}Y\right]\right) - \sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n}\frac{D_i}{p}X_i' - \mathbb{E}\left[\frac{D}{p}X'\right]\right)\beta \\
&\qquad - \mathbb{E}\left[\frac{D}{p}X'\right]\sqrt{n}(\hat{\beta}-\beta) + o_p(1)
\end{aligned}$$

where the first line holds by definition, the second line adds and subtracts $\mathbb{E}[(D/p)X']\hat{\beta}$, and the third equality holds because $\hat{\beta} \xrightarrow{p} \beta$ (and by the CMT). Recalling that,

$$\sqrt{n}(\hat{\beta}-\beta) = \mathbb{E}\left[\frac{(1-D)}{(1-p)}XX'\right]^{-1}\frac{1}{\sqrt{n}}\sum_{i=1}^{n}\frac{(1-D_i)}{(1-p)}X_ie_i + o_p(1)$$

we have that

$$\sqrt{n}(\widehat{ATT}-ATT) = \frac{1}{\sqrt{n}}\sum_{i=1}^{n}(A_i - B_i - C_i) + o_p(1)$$

where

$$\begin{aligned}
A_i &= \frac{D_i}{p}Y_i - \mathbb{E}\left[\frac{D}{p}Y\right] \\
B_i &= \left(\frac{D_i}{p}X_i' - \mathbb{E}\left[\frac{D}{p}X'\right]\right)\beta \\
C_i &= \mathbb{E}\left[\frac{D}{p}X'\right]\mathbb{E}\left[\frac{(1-D)}{(1-p)}XX'\right]^{-1}\frac{(1-D_i)}{(1-p)}X_ie_i
\end{aligned}$$

Thus, $\sqrt{n}(\widehat{ATT}-ATT) \xrightarrow{d} N(0,V)$ where $V = \mathbb{E}[(A-B-C)^2]$ (we can square here since ATT is a scalar). We can consistently estimate V by replacing all of the population averages by sample averages and replacing β with its consistent estimate $\hat{\beta}$.

```
A2 <- mean(D/p*Y)
Ai <- D/p*Y - A2
```

```
XDp <- X*as.numeric(D/p)
B2 <- colMeans(XDp)
# sweep subtracts a vector from each row of a matrix
Bi <- sweep(XDp, 2, B2) %*% bet

C2 <- t(B2)
XUp <- X*as.numeric((1-D)/(1-p))
C3 <- t(X)%*%XUp/n
ehat <- Y - X%*%bet
XUpe <- XUp*as.numeric(ehat)
Ci <- as.numeric(C2%*%solve(C3)%*%t(XUpe))

V <- mean( (Ai-Bi-Ci)^2 )
se <- sqrt(V)/sqrt(n)
round(se,3)
```

```
[1] 0.903
```

This indicates that we cannot reject that job training had no effect earnings at conventional significance levels.

Part (c)

Let's move to computing standard errors using the bootstrap. Towards, this end let's write a function that takes in some data and computes an estimate of ATT (this is essentially just the same code that we used before).

```
compute.att <- function(data) {
  Y <- data$re78
  D <- data$train
  p <- mean(D)
  X <- model.matrix(~age + educ + black + hisp + married + re75 + unem75, data=data)
  # run regression using untreated observations only
  bet <- solve(t(X)%*%(X*as.numeric((1-D)/p)))%*%t(X)%*%as.numeric(Y*(1-D)/p)

  att1 <- mean(D*Y/p)
  att2 <- sum(apply(D*X/p,2,mean)*as.numeric(bet))
  att <- att1 - att2
  att
}
```

There is a subtle issue about whether we should treat p as being known or estimated. Above I treated it like it was known. And, for this reason, I am going to draw bootstrap samples from the treated group and untreated group separately (it is not a big deal if you didn't do this though, just noting it so you can understand the code).

```r
# now bootstrap
biters <- 1000
treated_data <- subset(data, train==1)
untreated_data <- subset(data, train==0)
n1 <- nrow(treated_data)
n0 <- nrow(untreated_data)
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att(boot_data)
})

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

round(boot_se, 3)
```

```
[1] 0.861
```

These standard errors are similar to the ones we computed before.

Part (d)

For this part, we are just going to run a regression of Y on D and X.

```r
X2 <- cbind(X,D)
bet2 <- solve(t(X2)%*%X2)%*%t(X2)%*%Y
round(bet2,3)
```

```
              [,1]
(Intercept) -0.061
age         -0.057
educ         0.604
black       -0.597
hisp         2.547
```

```
married      1.530
re75         0.788
unem75      -0.079
D            0.525
```

```
ehat <- Y - X2%*%bet2
X2e <- X2*as.numeric(ehat)
Omeg2 <- t(X2e)%*%X2e/n
Q2 <- t(X2)%*%X2/n
V2 <- solve(Q2)%*%Omeg2%*%solve(Q2)
se2 <- sqrt(diag(V2))/sqrt(n)
round(se2,3)
```

```
(Intercept)         age        educ       black        hisp     married
      1.588       0.025       0.096       0.462       1.271       0.517
       re75      unem75           D
      0.036       0.967       0.884
```

The estimated coefficient on D is somewhat closer to 0 than we computed in the first part while the standard errors are about the same. In some sense, this doesn't appear to matter much, because in both cases we are estimating a small positive (and not statistically significant effect) of job training. However, this is mostly a result of us not being able to precisely estimate effects of job training. That said, our earlier point estimate is about 64% larger than the one from the regression which is arguably meaningfully different even though the identifying assumptions are the same.

Part (e)

Next, we will estimate the ATT using propensity score re-weighting. Since we will use the bootstrap to compute standard errors, let's write a function that computes an estimate of the ATT given some data—we'll use this function to estimate the ATT itself and inside our bootstrap iterations.

```
compute.att_ipw <- function(data) {
  Y <- data$re78
  D <- data$train
  p <- mean(D)

  logit_est <- glm(train ~ age + educ + black + hisp + married + re75 + unem75,
                   data=data, family=binomial(link="logit"))
  pscore <- predict(logit_est, type="response")
  att <- mean( ( (D/p) - (1-D)/p*pscore/(1-pscore) ) * Y )
  att
}
att_ipw <- compute.att_ipw(data)
```

and now code to compute bootstrapped standard errors

```
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att_ipw(boot_data)
})

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

# report results
data.frame(att_ipw=round(att_ipw,3), se=round(boot_se, 3))
```

```
  att_ipw    se
1   0.458 1.022
```

This estimate is somewhat lower than regression adjustment and the bootstrap standard errors have a similar magnitude.

Part (f)

Next, we will estimate the *ATT* using the doubly robust approach discussed in class.

```
compute.att_dr <- function(data) {
  Y <- data$re78
  D <- data$train
  p <- mean(D)
  X <- model.matrix(~age + educ + black + hisp + married + re75 + unem75, data=data)
  bet <- solve(t(X)%*%(X*as.numeric((1-D)/p)))%*%t(X)%*%as.numeric(Y*(1-D)/p)
  m <- X%*%bet

  logit_est <- glm(train ~ age + educ + black + hisp + married + re75 + unem75,
                   data=data, family=binomial(link="logit"))
  pscore <- predict(logit_est, type="response")
  att <- mean( ( (D/p) - (1-D)/p*pscore/(1-pscore) ) * ( Y - m ) )
  att
```

```
}
att_dr <- compute.att_dr(data)
```

and now code to compute bootstrapped standard errors

```
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att_dr(boot_data)
})

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

# report results
data.frame(att_dr=round(att_dr,3), se=round(boot_se, 3))
```

```
  att_dr se
1   0.62  1
```

Here the estimate is in between regression adjustment and propensity score re-weighting. The bootstrap standard errors are similar to the ones we computed before.

Part (g)

Next, we will estimate the ATT using machine learning.

```
set.seed(1234)

compute.att_ml <- function(data) {

  # create two folds
  data$fold <- sample(1:2, n, replace=TRUE)
  fold1 <- subset(data, fold==1)
  fold2 <- subset(data, fold==2)
```

```
  # inner function to compute an att for f2 using f1 to estimate the
  # preliminary models
  ml_att <- function(f1, f2) {
    # use f1 to estimate the first step models
    Dmod <- randomForest(as.factor(train) ~ age + educ + black + hisp +
                           married + re75 + unem75, data=f1)
    Ymod <- randomForest(re78 ~ age + educ + black + hisp +
                           married + re75 + unem75, data=f1)

    # get predictions with f2
    pscore <- predict(Dmod, newdata=f2, type="prob")[,2] # this gets p(d=1|x)
    out_reg <- predict(Ymod, newdata=f2)

    # compute att(k) with f2
    D <- f2$train
    p <- mean(D)
    Y <- f2$re78
    att1 <- mean(D/p*(Y-out_reg))
    att2 <- mean((1-D)/p * pscore/(1-pscore) * (Y-out_reg) )
    att <- att1-att2
    att
  }

  # cross splitting
  ml1 <- ml_att(fold1,fold2)
  # reverse roles
  ml2 <- ml_att(fold2,fold1)
  # average
  mean(c(ml1,ml2))
}

att_ml <- compute.att_ml(data)
```

and now code to compute bootstrapped standard errors

```
# note: the bootstrap takes longer here compared to other approaches
# below I used parallel processing to speed things up
# but it is also ok to decrease the number of bootstrap iterations (or
# to just wait a while longer...)
biters <- 1000
boot_res <- pblapply(1:biters, function(b) {
  # draw new data with replacement

  boot_treated_rows <- sample(1:n1, size=n1, replace=TRUE)
  boot_treated <- treated_data[boot_treated_rows,]
  boot_untreated_rows <- sample(1:n0, size=n0, replace=TRUE)
  boot_untreated <- untreated_data[boot_untreated_rows,]
```

```
  boot_data <- rbind.data.frame(boot_treated, boot_untreated)

  # alternative code that doesn't treat p as fixed
  #boot_rows <- sample(1:n, size=n, replace=TRUE)
  #boot_data <- data[boot_rows,]

  compute.att_ml(boot_data)
}, cl=10)

# run bootstrap
boot_res <- do.call("rbind", boot_res)

# compute bootstrap standard errors
boot_se <- apply(boot_res, 2, sd)

# report results
data.frame(att_ml=round(att_ml,3), se=round(boot_se, 3))
```

```
  att_ml    se
1  1.092 1.232
```

This is somewhat larger than our earlier estimates though the standard errors are also somewhat larger.