Assignment 2: ML

Causal Inference and Machine Learning

10/2/2020

In this assignment we'll start with data from an experiment and then move on to the harder case of observational data

Part 1: A field experiment

In this exercise we're going to look at the effect of a educational television program The Electric Company on children's reading ability. We'll simplify the design a bit for our analysis.

It was a fairly expensive show (but it had Morgan Freeman, so it was probably worth the money). In the exercise we'll look at what reading gains it may have made in the reading scores of its target audience: 1 through 4th graders.

Here we're looking at a two location trial that randomized at the level of school classes. (Actually it paired classes, but we'll ignore that for now). Each class was either treated (watch the program) or control (did not watch the program). The dependent variable is the score on a reading test at the end of the year: `post.score`. Our variables are therefore

- `post.score`: reading score at the end of the year
- `pre.score`: reading score before treatment assignment
- `city`: `F` (Fresno) or `Y` (Youngstown)
- `grade`: grade the class: 1-4
- `treatment`: `T` (treated group) or `C` (control group)
- `pair`: index of the treated and control pair (we ignore this here)
- `supp`: whether the program `R` replaced or supplemented `S` a reading activity

You can find the data in a file called "electric-company-long.csv" in the `data` folder. I'm going to assume you call it `electric` when you read it in.

```r
library(tidyverse)
set.seed(75)#for consistent results

electric <- read_csv("data/electric-company-long.csv") %>%
  mutate(grade = factor(grade),                       # make grade a factor
         treatment = ifelse(treatment == "T", 1, 0)) # make treatment an indicator
```

A first analysis

To start us off, compute and report the simple difference in mean `post.score` by treatment status, noting both the point estimate and its precision.

```r
#performing a regression
summary(mean_diff <- lm(post.score~treatment, electric))
```

```
Call:
```

```
lm(formula = post.score ~ treatment, data = electric)

Residuals:
    Min      1Q  Median      3Q     Max 
-55.778  -9.935   4.872  13.397  23.679 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   94.321      1.794   52.58   <2e-16 ***
treatment      5.657      2.537    2.23   0.0269 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.58 on 190 degrees of freedom
Multiple R-squared:  0.0255,    Adjusted R-squared:  0.02037 
F-statistic: 4.973 on 1 and 190 DF,  p-value: 0.02692
```

From the regression results, the point estimate is 5.657 and the precision/standard error is 17.58

A second analysis

Now let's try a totally different model at it. Load up the `randomForest` package (or `ranger` if you prefer), installing as necessary.

```r
library(randomForest)
```

Let's take look at the `randomForest` function. Looks like it takes a formula, which is convenient, so run that model with all the variables we have. We'll still be interested in the ATE.

Let's first try something unwise - mostly just for practice: use only the treatment variable to predict `post.score`, as we did above. randomForest has a formula interface like lm (see its help page), but as we'll see, it's not great. Nevertheless that's the easiest one to use here

```r
#trial with one variable - treatment
randomForest(post.score~treatment, data = electric)
```

```

Call:
 randomForest(formula = post.score ~ treatment, data = electric) 
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 1

          Mean of squared residuals: 312.4644
                    % Var explained: 0.4
```

Now, how to get a treatment effect out of this thing when there's no coefficient to look at?

By definition the ATE is the average Y(1) - Y(0). And we have model that, if we trust it, can give us both values. We can just make a data set which is like the regular one but where every class is treated, another one where every class is untreated, get predictions for each data set, subtract them, and average them.

This will, if course, work for your `lm` models, but we don't in those cases need to go through this process because there's happens to be a parameter summary in this case. Here's the process for the first model we fit. (Assuming that we called the first difference in means model `mod0`)

```r
mod0 <- lm(post.score ~ treatment, data = electric)
Y1 <- predict(mod0, newdata = mutate(electric, treatment = 1))
Y0 <- predict(mod0, newdata = mutate(electric, treatment = 0))
ATE <- mean(Y1 - Y0)
ATE
```

```
[1] 5.657292
```

```r
coef(mod0)[["treatment"]]
```

```
[1] 5.657292
```

Now do the same for a `randomForest` model. Through some quirk of bad R programming from our ML friends, the `predict` function for this model class only works properly if we *don't* use the formula interface. This is annoying, but easily gotten around: we'll make our own variables. Wait, no, of course we won't. If we wanted to make dummy variables and interactions by hand we'd be using Stata already.

We'll get R to do that part too, using `model.matrix` (which is what `lm` uses in the background anyway). So let's get a matrix of predictors, minus the intercept, which we won't need for `randomForests`

```r
# use the formula interface, like in lm
X <- model.matrix(post.score ~ treatment, data = electric)
head(X, 5)
```

```
  (Intercept) treatment
1           1         1
2           1         1
3           1         1
4           1         1
5           1         1
```

```r
# It's more convenient if we rewrap this thing back as a data.frame
X <- as.data.frame(X)
```

Right, now that's `X` we want. All we need next is a `y`. `electric$post.score` will work nicely as that.

OK now fit a random forest, ignoring all the fancy options for now.

```r
modrf <- randomForest(x = X, y = electric$post.score, data = electric)
```

Now you've got a model, construct the potential outcomes (the Y1 and Y0 vectors) and then compute the ATE

```r
Y1 <- predict(modrf, newdata = mutate(X, treatment = 1))
Y0 <- predict(modrf, newdata = mutate(X, treatment = 0))
ATE_rf <- mean(Y1 - Y0)
ATE_rf
```

```
[1] 2.879027
```

Do the two ATEs agree? No the two ATEs do not agree, ATE_rf is 2.879027 while ATE from a linearly fitted model is 5.657292.

Third Analysis

OK, now we've got the basic process down, let's give the random forest a bit more to work with. We'd *hope* that it could pick out some interactions and conditional treatment effects (they're definitely in there).

Create a new `X` that includes `pre.score`, `supp`, `grade`, and `city`, using the code above as a template.

```
X <- as.data.frame(
  model.matrix(post.score ~ treatment + city + grade + pre.score + supp,
               data = electric))
```

and fit a new random forest. What is the `ATE` in *this* model?

```
modrf2 <- randomForest(x = X, y = electric$post.score,
                       data = electric)

Y1 <- predict(modrf2, newdata = mutate(X, treatment = 1))
Y0 <- predict(modrf2, newdata = mutate(X, treatment = 0))
ATE_rf2 <- mean(Y1 - Y0)
ATE_rf2
```

```
[1] 2.504612
```

Now let's take a look at the conditional effects (remember our lm model has none of these, but the tree is inherently conditional).

We'll use the fact that we've got all the potential outcomes in `Y1` and `Y0`, and the definition of conditional ATE to get grade-specific ATEs. Here's one way to estimate a grade 1 treatment effect.

```
mean(Y1[electric$grade == 1] - Y0[electric$grade == 1])
```

```
[1] 4.306128
```

What do the others look like?

```
#estimating grade 2 treatment effects
mean(Y1[electric$grade == 2] - Y0[electric$grade ==2])
```

```
[1] 3.123949
```

```
#estimating grade 3 treatment effects
mean(Y1[electric$grade == 3] - Y0[electric$grade == 3])
```

```
[1] 0.5470857
```

```
#estimating grade 4 treatment effects
mean(Y1[electric$grade == 4] - Y0[electric$grade == 4])
```

```
[1] 1.564669
```

Fourth Analysis

But we know from the theory noted in the Double ML, that we'll do better causal inference if we treat that treatment variable like the treatment variable that it is (phew). For this we need some cross fitting. We could do this by hand - fit the m0 model, then the g0 model, then regress the residuals - but it's easier to get R to do all this.

We'll use `grf` package, specifically the `causal_forest` function to combine random forests with the Double ML fitting procedure.

If that package is not installed, install it (just once, and not in the Rmd please). Now lets some new treatment effects. We'll need *three* ingredients for this function:

- `X` like before but *without* the treatment variable
- `Y` just like before
- `W` the treatment variable

Note that everything here has to be numeric, so we really couldn't skip the `model.matrix` part.

```r
library(grf)

#defining W
W <- electric$treatment
#definig X
X <- as.data.frame(
  model.matrix(post.score ~ W + city + grade + pre.score + supp,
               data = electric))
#definig Y
Y <- electric$post.score
```

If you use the old `X` you made above, don't forget to remove the treatment variable, and add it to the function as `W`, otherwise the propensity score model will be quite implausibly good... Also, for good measure you should tune all the tuneable parameters. It won't take long (the help page will show you how)

```r
#attaching W to X and predicting W
forest_W <- regression_forest(X,W, tune.parameters = "all")
w_hat <- predict(forest_W)$predictions

#attaching Y to X and predicting Y
forest_y <- regression_forest(X,Y, tune.parameters = "all")
y_hat <- predict(forest_y)$predictions

#calculating a weighted sum of variable importance
forest_y_varimpo <- variable_importance(forest_y)

#reducing the agression to avoid training the model on very few variables
selected_vars <- which(forest_y_varimpo/mean(forest_y_varimpo) > 0.2)

tau_forest <- causal_forest(X[, selected_vars], Y, W,
                            W.hat = w_hat, Y.hat = y_hat,
                            tune.parameters = "all")
#checking whether predictions are well calibrated
test_calibration(tau_forest)
```

```
Best linear fit using forest predictions (on held-out data)
as well as the mean forest prediction as regressors, along
with one-sided heteroskedasticity-robust (HC3) SEs:

                                Estimate Std. Error  t value    Pr(>t)
mean.forest.prediction           2.25682    0.43912   5.1395 3.405e-07 ***
differential.forest.prediction -36.76050    2.88178 -12.7562         1
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now let's get some ATEs out. Happily, the authors know we want this kind of thing so there are a couple of convenient ATE computation functions. Use `average_treatment_effect`

```
#average treatment effect on the full sample
ATE_tf <- average_treatment_effect(tau_forest, target.sample = "all")
ATE_tf
```

```
  estimate    std.err
158.745492   0.907677
```

```
#estimating ATE on the treated
ATE_tf_t <- average_treatment_effect(tau_forest, target.sample = "treated")
ATE_tf_t
```

```
  estimate    std.err
160.531505   1.170054
```

```
#estimating ATEs on the control
ATE_tf_ctrl <- average_treatment_effect(tau_forest, target.sample = "control")
ATE_tf_ctrl
```

```
  estimate    std.err
154.717442   1.604823
```

```
#estimating the weighted ATE
ATE_tf_w <- average_treatment_effect(tau_forest, target.sample = "overlap")
ATE_tf_w
```

```
estimate  std.err
160.3691 159.8658
```

(This is doing our Y1 and Y0 creation, comparison, and averaging in the background and in a mildly more sophisticated fashion).

Better yet, we can ask for conditional effects too by specifying a subset of the cases to consider, e.g. `electric$grade == 4`. What does this debiased model think the grade-specific treatment effects are?

```
#estimating CATE when grade==1
CATE_tf_gr_1 <- average_treatment_effect(tau_forest, subset=electric$grade==1, method="AIPW")
CATE_tf_gr_1
```

```
  estimate    std.err
161.032135   3.174885
```

```
#estimating CATE when grade==2
CATE_tf_gr_2 <- average_treatment_effect(tau_forest, subset = electric$grade==2, method = "AIPW")
CATE_tf_gr_2
```

```
  estimate    std.err
159.722253   1.424182
```

```
#estimating CATE when grade==3
CATE_tf_gr_3 <- average_treatment_effect(tau_forest, subset = electric$grade==3, method = "AIPW")
CATE_tf_gr_3
```

```
  estimate    std.err
156.572311   1.080234
```

```
#estimating CATE when grade==4
CATE_tf_gr_4 <- average_treatment_effect(tau_forest, subset = electric$grade==4, method = "AIPW")
CATE_tf_gr_4
```

```
  estimate    std.err
156.950023   1.016768
```

Compare and contrast

Are these ATEs right? Well, naturally that's hard to answer, but if we take the time to specify a really careful linear model, e.g. this one with a bucket of interactions

```
mod_flm <- lm(post.score ~ treatment + grade + treatment:grade +
              pre.score + pre.score:grade +
              supp + supp:grade + supp:grade + city,
                  data = electric)
```

then we can compare them to the causal forest. How about you do that? Borrow the code you used above.

```
Y1 <- predict(mod_flm, newdata = mutate(electric, treatment = 1))
Y0 <- predict(mod_flm, newdata = mutate(electric, treatment = 0))
ATE_mod_flm <- mean(Y1 - Y0)

ATE_grade_1 <- mean(Y1[electric$grade == 1] - Y0[electric$grade == 1])
ATE_grade_1
```

```
[1] 8.756314
```

```
ATE_grade_2 <- mean((Y1[electric$grade == 2] - Y0[electric$grade == 2]))
ATE_grade_2
```

```
[1] 4.306566
```

```
ATE_grade_3 <- mean((Y1[electric$grade == 3] - Y0[electric$grade == 3]))
ATE_grade_3
```

```
[1] 1.916245
```

```
ATE_grade_4 <- mean((Y1[electric$grade == 4] - Y0[electric$grade == 4]))
ATE_grade_4
```

```
[1] 1.640197
```

To the extent these are similar, then our causal forest has discovered all those interactions, and perhaps more.

Foreign direct investment

Time to try this out in a observational case. Let's revisit the Jensen data…

```
fdi <- foreign::read.dta("data/jensen-rep.dta")
```

Fit a causal forest to this data, using `regime` as the treatment variable. Following the lecture slides, you can use `Fvar5` as the dependent variable and everything else as `X`.

What do you find?

I find that in these highly weighted countries (Fvar5), their slightly positive average partial effects communicates that their non-democratic regime attracts less FDI. Unless they undergo aggressive regime changes to democratic, their FDI attraction will yield even higher positive results.

Hint: you'll want to use `average_partial_effect` to get the equivalent of a treatment slope parameter. Also, don't forget to tune the parameters.

```
#defining X
X <- as.data.frame(
  model.matrix(Fvar5 ~., data = fdi))
#var5 + market + lgdppc + gdpgrowt + tradeofg + overallb + year + country + generalg + d2 +d3
#defining W
W <- fdi$regime

#defining Y
Y <- fdi$Fvar5

#fitting a causal forest model
mod_fdi <- causal_forest(X, Y, W, tune.parameters="all")

#calculating the avearge partial effect
APE_fdi <- average_partial_effect(mod_fdi)
APE_fdi
```

```
  estimate    std.err
0.06868520 0.06706185
```