Quantitative Research Methods IV - 17.806

Recitation, Week 6.
Topic: Causal Machine Learning I.

Benjamín Muñoz

March 17, 2023

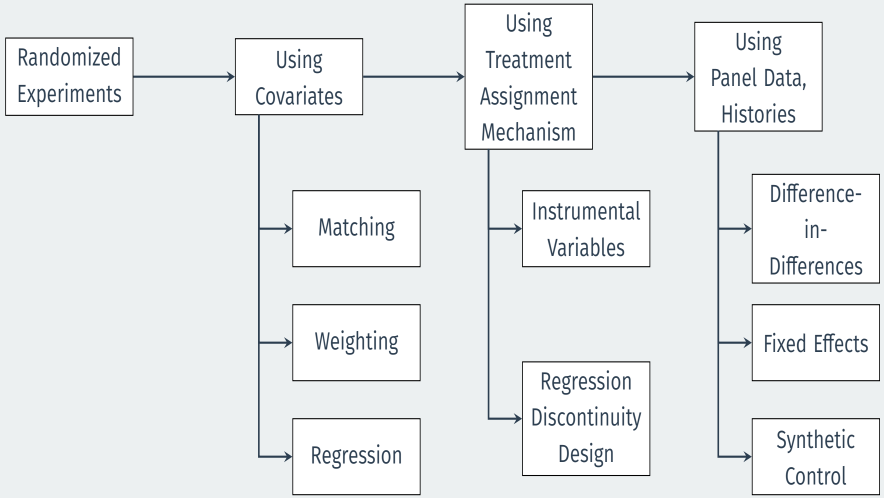
MIT

Massachusetts Institute of Technology

Table of contents

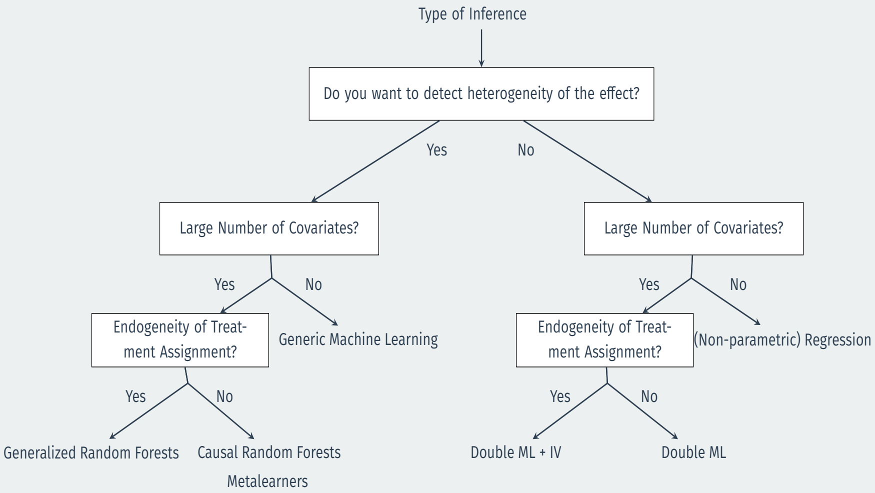
1/ Theoretical Roadmap

Identification Strategies Quant II



Randomized Experiments
Using Covariates
Using Treatment Assignment Mechanism
Using Panel Data, Histories
Matching
Weighting
Regression
Instrumental Variables
Regression Discontinuity Design
Difference-in-Differences
Fixed Effects
Synthetic Control

Causal Machine Learning


Type of Inference
Do you want to detect heterogeneity of the effect?
Yes
No
Large Number of Covariates?
Large Number of Covariates?
Yes
No
Yes
No
Endogeneity of Treat-ment Assignment?
Generic Machine Learning
Endogeneity of Treat-ment Assignment?
(Non-parametric) Regression
Yes
No
Yes
No
Generalized Random Forests
Causal Random Forests
Metalearners
Double ML + IV
Double ML

What Type of Problems Can Machine Learning Solve?

- Machine Learning techniques cannot solve causal identification problems.
    - How much can we learn about parameters from infinite amount of data? (Manski, 2007).
- The predictive power of ML tools can indeed solve some inferential issues.
    - : How much can we learn about parameters from infinite amount of data? (Manski, 2007).
    - However, these techniques create potential new problems, which must be taken into consideration (efficiency, bias, etc.).

What kind of problems will we try to solve?

1. High Dimensionality of **X**:
    - I have a large number of covariates ($p >> n$). How can I use them to justify my identification strategy (Selection on Observables)?
2. Heterogeneous Treatment Effects:
    - The typical Quantity of Interest is the Conditional Average Treatment Effect. But what if I want a different QoI? (Fundamental Problem of Causal Inference).
    - A typical strategy is to use a linear interactive model. But these models impose strong assumptions (see Hainmueller, Mummolo, and Xu, 2019).
    - The best practice is to pre-register my studio. But usually, we don't. So how can we protect against the problem of multiple tests?

2/ Causal Inference Review

Design Trumps Analysis

- Ignorability $\rightsquigarrow$ Ideal setup (e.g, experiment)

$$D_i \perp \{Y_i^0, Y_i^1\}$$

- At least we want conditional ignorability

$$D_i \perp \{Y_i^0, Y_i^1\} \mid X$$

- Additional Assumption: Common Support

$$0 < P(D_i = 1|X_i = x) < 1 \forall x \in \mathcal{X}$$

Design Trumps Analysis

- **Regression:** Two extra assumptions:
    - Constant Treatment Effect ($\tau = Y_i^1 - Y_i^0$ for all i).
    - Linearity ($Y_{di} = \beta_0 + d\beta_1 + \gamma^T \boldsymbol{X} + \epsilon_i$).
- OLS approach: modeling $E(Y|X)$ correctly.
- **Matching:** nonparametric method of adjustment for treatment assignment patterns.
- Propensity score approach: modeling $E(T|X)$ correctly.
- We'll need to specify at least one of two models correctly:

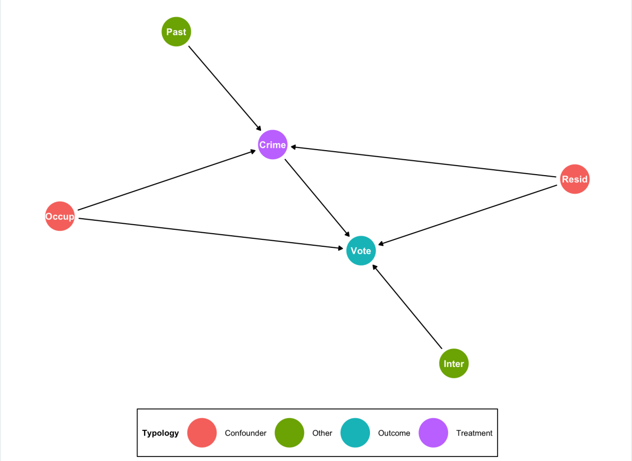
Outcome model: $E(Y|X) = f(X)$

Treatment assignment mechanism: $E(T|X) = g(X)$

Design Trumps Analysis

- DAG Perspective – Can Block Either Backdoor path.

Figure 1: Confounding Paths in DAG



Regression Estimation

- Bias and Variance Tradeoff.

- The analyst has to correctly specify the model fairly precisely.
- Including irrelevant variables is ok, but only if there aren't too many.

  $\rightsquigarrow$ too many variables lead to less precise estimate (high variance).

3/ High Dimensionality of X

High Dimensionality of X

- Definition: $p >> n$.
- Parametric Asymptotics: p is fixed as $n \longrightarrow \infty$. Nonparametric regression $p \longrightarrow \infty$ but at a much slower rate than n.
- Typical Regression Setting: if $p > n$, then $\widehat{\beta_{OLS}}$ is not uniquely defined. For $p < n$ but *large*, $\widehat{\beta_{OLS}}$ can be unstable and have high variance.
- ML based approach can:
  - deal with arbitrary interactions or flexible specification.
  - regularize irrelevant terms.
- **Sparsity:** only a subset of the elements of β are nonzero.
- **Approximate Sparsity:** generalization of Sparsity.
  - Recasts the High-Dimensional problem in a variable selection framework.

High Dimensionality of X

- Algorithms
    1. **Post-LASSO:** Choose correct variables for outcome model (naive-Lasso).
    2. **Double Selection:**Choose correct variables for treatment and outcome model.
    3. **Double/Debiased ML:** Choose correct variables for treatment and outcome model and allow for flexible funcional form.

- Model Selection and Estimation cannot be achieved optimally at the same time.
    - Chances of mistakes in selection step = Contamination of estimation+inference step. Regularization Bias

Post Lasso (Naïve LASSO)

- One (naive) approach is to focus on just the outcome model.
- Instead of specifying a small set of covariates that almost all matter, analyst specifies a large number of covariates that mostly do not.
- Use a Lasso model to choose the ones that are significant (non-zero covariates).
- Coverage probability of standard confidence intervals can be far from the nominal level (downward bias). Affected by (and increasing in) the correlation between D and X Omitted-Variable Bias (Zero if $\beta = 0$ or $\rho = 0$).

Post Lasso (Naïve LASSO)

R Code

```
#Load package
library(glmnet)

# CV is not how chernozhukov et al choose

# Run the Outcome model
lasso <- cv.glmnet(x = X, y = y)

# Choose which variables to keep
keep <- as.matrix(coef(lasso, s = "lambda.min")[-1]) !=0

# Run OLS
lm(y ~ T + X[, keep])
```

Double Selection

- Concern with post Lasso: it might miss weak predictors of the outcome that are strong predictors of the treatment.
- This is particularly a concern if the outcome model is non-sparse or if covariates are highly correlated.
- Want to have the covariates and functional form suitable for outcome and treatment models.
- Solution is to select covariates that predict both the treatment and the outcome $\rightsquigarrow$ Double Selection.
- You can put many covariates as you want as well as their interactions.
- Under approximate sparsity, asymptotic normality
- **Post-Regularization LASSO** (Partialling Out LASSO): Gains in efficiency. Uses only the relevant components of X to separately demean Y and D, leading to a greater parsimony,

Double Selection

R Code

```r
# CV is not how chernozhukov et al choose
# the penalty term. decide outcome model
outcome_model <- cv.glmnet(x = X, y = y)

# Choose covariates whose coefficients are not zero
keep_outcome_model <- as.matrix(coef(outcome_model, s = ”lambda.min”)[-1]) != 0

# Decide treatment model
treat_model <- cv.glmnet(x = X, y = T, family = ”binomial”)

# Choose covariates whose coefficients are not zero
keep_treat_model <- as.matrix(coef(treat_model, s = ”lambda.min”)[-1]) != 0

# Run regression on chosen variables
lm(y ˜ T + X[, keep_outcome_model|keep_treat_model])

### Alternative Code
library(hdm)

#### Implement Double Selection with LASSO
dsl <- rlassoEffect(x = X, d = T, y = Y, method = ”double selection”, post = TRUE)
summary(dsl)
```

Double ML

- In principle, double selection can be used to protect us from many forms of model misspecification.
- For example, a high order polynomial can fit most response functions very flexibly.
- However, no guarantees that this function is sparse or efficiently estimated.
- Double ML – tries to overcome this by using flexible ML methods to model the covariates.
- Suppose true model is:

$$Y = \tau T + g(X) + \epsilon$$
$$T = m(X) + \eta$$

Double ML

- If we knew $m(\cdot)$ and $g(\cdot)$, recovering τ would be trivial.

  $\rightsquigarrow$ [FWL Theorem] Just regress $u = Y - g(X)$ on $e = T - m(X)$.
- Immunization/Orthogonalization Procedure.
- **Intuition:** remove a part of T and Y that can be explained by X (i.e., partialing out, obtain residual), and then run a regression of them.
- Instead, we will use machine learning to flexibly model $\hat{g}$ and $\hat{f}$.
- The problem with this is overfitting – that we'll capture noise or the effect of the treatment in our estimates.
- The solution is **cross fitting** – fit $m(X)$ and $g(X)$ within one part of the sample and estimate the residuals on the other.
- Sample Splitting reduces the dependency between the estimation stages and can improve performance.

Double ML Implementation

R Code

```
# also check slide 11
# for each fold k,
#get_resids <- function(X,y,treat, fold, folds) {
                d <- data.frame(y = y[fold != folds], X = X[fold != folds,])
                outcome_model <- ranger(y ~ ., data = d)
                d <- data.frame(treat = treat[fold != folds], X = X[fold != folds,])
                treat_model <- ranger(treat ~ ., data = d)
                V_hat <- treat[fold == folds] - predict(treat_model, newx = X[fold == folds,])
                W_hat <- y[fold == folds] - predict(outcome_model, newx=X[fold == folds,])
                mod <- lm(W_hat~V_hat)
                tau <- coef(mod)[2]
                epsilon <- resid(mod)
                return(list(tau = tau, epsilon = epsilon, v_hat = V_hat))
        }
# naive approach to conduct k-fold cross validation
        folds <- sample(1:k, nrow(X), replace = TRUE)
        Vsqrd <- 0 VTimesEpsilon <- 0 tau <- rep(NA, k) for(i in 1:k) {
                results <- get_resids(X, y, treat = treat, fold = i, folds = folds)
                tau[i] <- results$tau
                Vsqrd <-  Vsqrd + sum(results$v_hat^2)
                VTimesEpsilon <- VTimesEpsilon + t(results$v_hat^2) %*% results$epsilon^2
        }
        tau <- mean(tau)
        sesqrd <- (Vsqrd/nrow(X))^(-2)*(VTimesEpsilon/nrow(X))
        se <- sqrt(sesqrd/nrow(X))
```

Double ML Implementation

R Code

```
### Alternative Code
library(DoubleML)
library(mlr3)
library(mlr3learners)

#### Define data for DDML
dml_data_sim <- double_ml_data_from_matrix(X = X, y= Y, d = T)

#### Select LASSO learner
learner  <- lrn("regr.cv_glmnet", s="lambda.min")
ml_l_sim <- learner$clone()
ml_m_sim <- learner$clone()

#### Execute DDML
obj_dml_plr_sim <- DoubleMLPLR$new(dml_data_sim, ml_l=ml_l_sim, ml_m=ml_m_sim)
obj_dml_plr_sim$fit()
print(obj_dml_plr_sim)
```