Quantitative Research Methods IV - 17.806

Recitation, Week 9.

Topic: Causal Inference with Machine Learning

Jingtian Chen

April 4, 2025

MIT

Massachusetts Institute of Technology

Table of contents

1. Theoretical Roadmap

2. High-dimensional Causal Inference

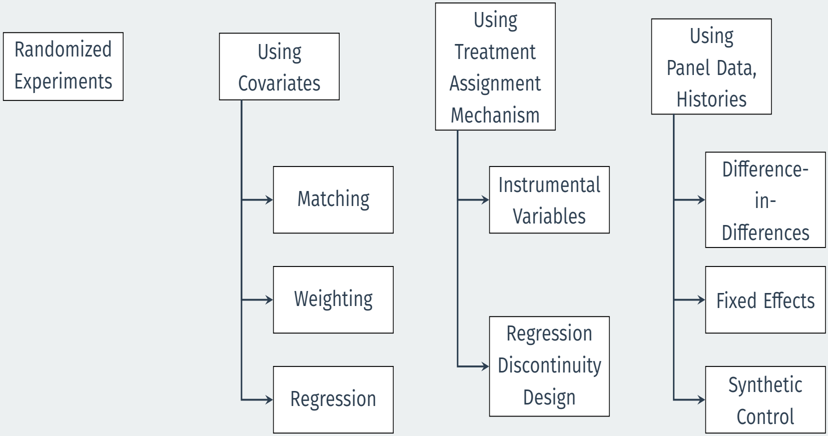
3. Heterogeneous Treatment Effects

Introduction

- Problem Set 4: due April 8 by 3 PM
- Review of causal inference with ML
- Problem set hints

1/ Theoretical Roadmap

Identification Strategies Quant II



What Type of Problems Can Machine Learning Solve?

- Machine Learning cannot solve causal identification problems.
- The predictive power of ML can solve some inferential issues.

1. High Dimensionality of $\mathbf{X}$:
   - 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:
   - A typical strategy is to use a linear interactive model. But these models impose strong assumptions (see Hainmueller, Mummolo, and Xu, 2019).

Causal Inference Review

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

$$T \perp \{Y(0), Y(1)\}$$

- At least we want conditional ignorability

$$T \perp \{Y(0), Y(1)\} \mid X$$

- Additional Assumption: Common Support

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

Causal Inference Review

- Conditional ignorability:

$$T \perp \{Y(0), Y(1)\} \mid X$$

- We'll need to specify at least one of two models correctly:

$$\text{Outcome model (e.g., OLS)} : E(Y|X) = f(X, T)$$
$$\text{Treatment assignment mechanism (e.g., PS):} \; E(T|X) = g(X)$$

- DAG Perspective – Can Block Either Backdoor path

$$\begin{tikzcd} X \arrow[r] \arrow[rr, bend left] & T \arrow[r] & Y \end{tikzcd}$$

Regression Estimation

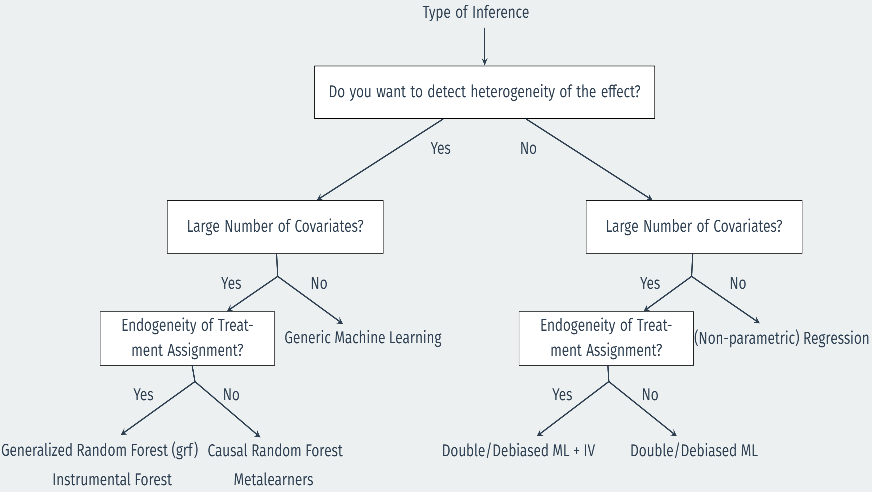
- Bias and Variance Tradeoff.

- 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).

2/ High-dimensional Causal Inference

Causal Machine Learning



High Dimensionality of X

- Definition: $p >> n$.
- 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.
- 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.

Post Lasso (Naïve LASSO)

- One (naive) approach is to focus on just the outcome model
- Include a large number of covariates (and their interactions)
- Use a Lasso model to choose non-zero covariates

R Code

```
#Load package
library(glmnet)

# 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.

Double Selection

R Code

```
# 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/Debiased 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.

Double/Debiased ML

- Suppose true model is:

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

- If we knew $m(\cdot)$ and $g(\cdot)$, recovering τ would be trivial.
  $\rightsquigarrow$ [Frisch-Waugh-Lovell 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.

Double/Debiased ML Implementation

R Code

```
# 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/Debiased 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)
```

3/ Heterogeneous Treatment Effects

Heterogeneous Treatment Effects

- Degree to which different treatments have differential causal effects on each unit.
- Multiple justifications:
    1. Alternative Quantities of Interest.
    2. Selecting the most effective treatment (optimal allocation).
    3. Subgroup Analysis: identifying subgroups of observations for which the treatment in particularly efficacious or deleterious.
    4. Avoid strong functional form assumptions.

Conditional Average Treatment Effects

- CATE: $\tau(D; X) = \mathbb{E}(Y_i^1 - Y_i^0 | X_i = x)$

1. Linear Interactive Model:

$$Y = \beta_0 + \beta_1 D + \beta_2 M + \beta_3 D \times M + \gamma X + \epsilon$$

   - **Extra Assumptions:** the treatment effect varies linearly with M, and for each value of M, the treatment and control groups should have a sufficient number of overlapping cases (see **Hainmueller, Mummolo, and Xu, 2019**).

2. Machine learning tools to estimate the heterogeneous treatment effects:
   - Loss function approach: Squared Loss SVM with separate LASSO constraints (**Imai and Ratkovic, 2013**), R-learner (**Nie and Wager, 2021**).
   - Construct potential outcomes: X-learner (**Künzel et al., 2019**)

Causal Forests

- **Tree:** method that recursively partitions the high-dimensional covariate space into smaller units.
- Prediction $\neq$ Detection of Heterogeneity.
  - Minimization of within-leaf variance of Y (all the cases belonging to the same leaf should be homogeneous in terms of the outcome) $\rightsquigarrow$ Minimization of the within-leaf variance of the estimated treatment effects (inter-leaf variation should be large).
  - New Splitting Rule:

$$-\widehat{\mathrm{EMSE}}_{\tau}(\mathrm{S}^{\mathrm{Tr}}, \mathrm{N}^{\mathrm{est}}, \sqcap) = \underbrace{\frac{1}{N^{Tr}} \sum_{i \in S^{Tr}} \hat{\tau}^2(X_i | S^{Tr}, \sqcap)}_{\substack{\text{Variance of Treatment Effects} \\ \text{across Leaves} \\ \text{Prefer Leaves with} \\ \text{Heterogeneous Effects Across} \\ \text{Leaves}}} - \underbrace{\left(\frac{1}{N^{Tr}} + \frac{1}{N^{Est}}\right) \sum_{l \in \sqcap} \left(\frac{S^2_{S^{Tr}_{Treat}}(l)}{p} + \frac{S^2_{S^{Tr}_{Control}}(l)}{1-p}\right)}_{\substack{\text{Uncertainty about Leaf} \\ \text{Treatment Effects} \\ \text{Prefer Leaves with Good Fit} \\ \text{(Leaf-Specific Effects estimated} \\ \text{Precisely)}}}$$

Causal Forests

- **Honest Procedure:** not use the same information for selecting the model structure as for estimation given a model structure.
  - One (independent) split of the data is used to learn the tree structure/-partition, and the second split of the data is used to conduct inference (estimation of treatment effects).

R Code

```r
# Run Causal Forests (Basic Algorithm)
library(grf)

# Estimate Causal Forest
cf <- causal_forest(X = X, Y = Y, W = W, num.trees = 10000,
                    honesty = TRUE, honesty.fraction = 0.5,
                    tune.parameters = "all", seed = 17806)

# Estimate Predicted Values (CATEs)
pred <- predict(object = cf, newdata = newX,
                estimate.variance = TRUE)$predictions

# Use expanded algorithm for Problem 3.
```

Metalearners

- Meta-algorithms decompose estimating the CATE into several subregression problems that can be solved with any supervised ML method.
- Combination of **base learners** in a specific manner while allowing the base learners to take any form.

1. **S-Learner** (Single) $\rightsquigarrow$ using all of the features and the treatment indicator (without giving to D a special role).

$$\mu(x) = \mathbb{E}[Y|X = x, D = d]$$
$$\hat{\tau}(x) = \hat{\mu}(x, D = 1) - \hat{\mu}(x, D = 0)$$

   - Risk of dropping the treatment.
   - can be biased toward 0

Metalearners

2. **T-Learner** (Two) $\rightsquigarrow$ use base learners to estimate the conditional expectations of the outcomes separately for control and treatment groups.

$$\mu_0(x) = \mathbb{E}[Y|D = 0, X = x]$$
$$\mu_1(x) = \mathbb{E}[Y|D = 1, X = x]$$
$$\hat{\tau}(x) = \hat{\mu}_1(x) - \hat{\mu}_0(x)$$

- Ignore group size.
- Works well when there are no common trends in μ_0 and μ_1

Metalearners

3. **X-Learner** $\rightsquigarrow$ uses each observation in the training set in an X-like shape (Sample Splitting for Fundamental Problem of Causal Inference).

   3.1 Estimate the response functions:

$$\mu_0(x) = \mathbb{E}[Y|D = 0, X = x]$$
$$\mu_1(x) = \mathbb{E}[Y|D = 1, X = x]$$

   3.2 Impute the individual treatment effects (for the Treated group with the control-outcome estimator and for the Control group with the treatment-outcome estimator):

$$\tilde{D}_i^1 = Y_i^1 - \hat{\mu}_0(X_i^1)$$
$$\tilde{D}_i^0 = \hat{\mu}_1(X_i^0) - Y_i^0$$

Metalearners

3.3 Use any learner to estimate/predict the imputed treatment effects for each group:

$$\hat{\tau}_1(x) = \mathbb{E}[\tilde{D_i^1}|D = 1, X = x]$$
$$\hat{\tau}_0(x) = \mathbb{E}[\tilde{D_i^0}|D = 0, X = x]$$

3.4 Define the CATE estimate by the weighted average of the two estimates in 3.3:

$$\hat{\tau}(x) = g(x)\hat{\tau}_0(x) + (1 - g(x))\hat{\tau}_1(x)$$

- $g(x) \in [0, 1]$. Good option =Propensity Score.
- Efficient use of unbalanced design.