Lec12: Causal Machine Learning I

Isidoro Garcia Urquieta

2021

Agenda

- Omitted Variable Bias
- Balancing Scores: `e(x)`, IPW, AIPW
- Double Selections LASSO
- Double Debiased ML
- Residual Balancing
- Causal Trees
- Double Debiased ML for HTE
- Causal forests
- Generalized Random Forests

Set up

Para poder hablar de Causal Machine Learning tenemos que volver brevemente a los basics de inferencia causal

Queremos estimar τ de manera insesgada:

$$Y_i = X\beta + \tau T_i + \epsilon_i$$

Donde: - Y_i es la variable sobre la que queremos evaluar el impacto

- X es una matriz (de alta dimensionalidad) de controles
- T_i es el tratamiento sobre el que nos gustaría evaluar el impacto sobre Y_i ceteris paribus
  - Noten como T_i puede ser dicotómico, categórico o continuo.
- ϵ_i es ruido blanco

Como combinar Inferencia Causal con Machine Learning?

De manera general, la inferencia causal trata de **inferir** sobre **derivadas** o cambios en la métrica Y_i en lugar de la Y_i misma (el nivel)

$$\tau_i = \frac{\partial Y_i}{\partial T_i}$$

Por otro lado, los modelos de Machine Learning que vimos buscan **predecir** Y_i fuera de la muestra:

$$\hat{Y}_i = f(\hat{X}_i) + \epsilon_i$$

Son problemas fundamentalmente distintos!

Un grupo de autores: (Christian Hansen, Susan Athey, Guido Imbens, Stephan Wager, Belloni, Victor Chernozshukov, Matt Taddy, Esther Duflo, Robert Tibshirani) se pusieron a pensar cómo aprovechar ML y aplicarlo a **predecir e inferir impactos causales (derivadas)**

Omitted Variable Bias

Recuerden lo que origina el sesgo en la estimación de τ. Imaginen que tenemos lo siguiente:

- Modelo real: $Y_i = \beta_0 + \beta_1 X_{1i} + \tau T_i + \epsilon_i$
- Modelo estimado: $Y_i = \beta_0 + \tau T_i + \psi_i$
- $\psi_i = \epsilon_i + \beta_1 X_{1i}$

Veamos la $\hat{\tau}$ del modelo estimado:

$$\hat{\tau} = \frac{\Sigma_i^N (T_i - \bar{T})(Y_i)}{\Sigma_i^N (T_i - \bar{T})^2}$$

Sustituyo Y_i

$$\hat{\tau} = \frac{\Sigma_i^N (T_i - \bar{T})(\beta_0 + \tau T_i + \psi_i)}{\Sigma_i^N (T_i - \bar{T})^2}$$

Omitted Variable Bias II

Distribuyo los términos:

$$\hat{\tau} = \frac{\Sigma_i^N (T_i - \bar{T})\beta_0}{\Sigma_i^N (T_i - \bar{T})^2} + \frac{\Sigma_i^N (T_i - \bar{T})\tau T_i}{\Sigma_i^N (T_i - \bar{T})^2} + \frac{\Sigma_i^N (T_i - \bar{T})\psi_i}{\Sigma_i^N (T_i - \bar{T})^2}$$

$$\hat{\tau} = \beta_0 \frac{\Sigma_i^N (T_i - \bar{T})}{\Sigma_i^N (T_i - \bar{T})^2}{}^{0} + \tau \frac{\Sigma_i^N (T_i - \bar{T}) T_i}{\Sigma_i^N (T_i - \bar{T})^2}{}^{1} + \frac{\Sigma_i^N (T_i - \bar{T})\psi_i}{\Sigma_i^N (T_i - \bar{T})^2}$$

$$\hat{\tau} = \tau + \frac{\Sigma_i^N (T_i - \bar{T})\psi_i}{\Sigma_i^N (T_i - \bar{T})^2}$$

Sustiyo $\psi_i = \beta_1 X_{1i} + \epsilon_i$

$$\hat{\tau} = \tau + \frac{\Sigma_i^N (T_i - \bar{T})(\beta_1 X_{1i})}{\Sigma_i^N (T_i - \bar{T})^2} = \tau + \beta_1 \frac{\Sigma_i^N (T_i - \bar{T})(X_{1i})}{\Sigma_i^N (T_i - \bar{T})^2}$$

Noten como $\frac{\Sigma_i^N (T_i - \bar{T})(X_{1i})}{\Sigma_i^N (T_i - \bar{T})^2}$ es el coeficiente de la regresión $X_{1i} = \delta_o + \delta_1 T_i$.

Omitted Variable Bias III

Por lo tanto:

$$\hat{\tau} = \tau + \beta_1 \delta_1$$

El omitted variable bias es el producto de:

- β_1: La relevancia de la variable omitida X_{1i} sobre Y_i
- δ_1: La relevancia de la variable omitida X_{1i} sobre T_i

Si la variable omitida X_{1i} no tiene relación con el tratamiento T_i (δ_1=0), el sesgo es cero y el estimador es causal.

De igual manera, si la variable omitida X_{1i} no tiene relación con Y_i, el sesgo es cero y el estimador es causal

Inferencia Causal con Big Data

En el mundo de Big Data es mucho más probable que encontremos muchos desbalances entre los grupos de tratamiento.

Aún en la presencia de un experimento, es muy probable que haya varias variables desbalanceadas.

Que hacemos

- Controlamos por todas las variables para rebalancearlas?

Inferencia Causal con Big Data

En el mundo de Big Data es mucho más probable que encontremos muchos desbalances entre los grupos de tratamiento.

Aún en la presencia de un experimento, es muy probable que haya varias variables desbalanceadas.

Que hacemos

- Controlamos por todas las variables para rebalancearlas?
- Puede ser, pero podemos perder muchos grados de libertad

Inferencia Causal con Big Data

En el mundo de Big Data es mucho más probable que encontremos muchos desbalances entre los grupos de tratamiento.

Aún en la presencia de un experimento, es muy probable que haya varias variables desbalanceadas.

Que hacemos

- Controlamos por todas las variables para rebalancearlas?
- Puede ser, pero podemos perder muchos grados de libertad
- Mejor usemos el propensity score $P(treat)$!

Propiedades del Propensity Score/Mecanismo asignación

Recordemos las 3 propiedas que deben cumplirse para hacer inferencia causal:

1. **Asignación individual (SUTVA):** Stable Unit Treatment Value Assumptions (SUTVA). Esto significa no contaminación entre grupos (non-interference)
2. **Asignación Probabilística (Overlap):** Cada individuo tiene una probabilidad de pertenecer a los grupos de tratamiento conocida y $\in (0, 1)$
3. **Unconfoundness (Excludability):** La asignación es independiente a las características de los usuarios.

Balancing Score

Propensity score:

$$e(X_i) = E[T_i|X_i] \iff {}_{T_i \in \{0,1\}} P(T_i|X_i)$$

El propensity score es muy importante para generar balance. Esto es especialmente cierto en la presencia de una X_i de alta dimensionalidad.

Rosembaum (1983) probó que si estimamos $e(X_i)$ y controlamos por este, **podemos alcanzar balance en todas las** X_i. Esto es genial! Podemos control por una sola variable y alcanzar balances de alta dimensionalidad.

- Esto convierte al propensity score en un **Balancing Score**

Todavía tenemos que ver cómo:

1. Cómo estimar $e(X_i)$ y
2. Cómo usarlo para alcanzar unconfoundness

Balancing Score II

Algunos de los métodos más comunes que utilizan el Propensity Score son:

- **Inverse-Weightening Estimators**: Usas $e(X_i)$ como peso de la regresión para alcanzar balance.
- **Propensity Score Matching**: Usas $e(X_i)$ para contruir pares de observaciones de tratamiento y control muy similares mediante funciones de distancia (i.e. Malahanobis). El gran problema (como todo estimador no paramétrico) es que son computacionalmente pesadas y las funciones de distancia pueden ser ad-hoc.
- **Blocking**: Usas $e(X_i)$ para construir bloques donde el score sea muy similar tal que alcances unconfoundness. Una forma común es usar Dalenius Hodge.

Inverse Probability Weightening Estimators (IPW)

Veamos la lógica matemática de IPW:

$$T_i \in 0, 1$$

$$Y_i^{obs} = Y_i(1)\,T_i + Y_i(0)(1 - T_i)$$

$$E\Big[\frac{Y_i^{obs}\,T_i}{e(X_i)}\Big] = E[Y(1)]$$

$$E\Big[\frac{Y_i^{obs}(1 - T_i)}{1 - e(X_i)}\Big] = E[Y(0)]$$

Con estas podemos estimar $\hat{\tau}$ como:

$$\hat{\tau} = \frac{1}{N}\Sigma_{i=1}^{N}\Big(\frac{T_i * Y_i}{e(X_i)} - \frac{(1 - T_i) * Y_i}{1 - e(X_i)}\Big)$$

IPW Prueba

Noten primero como:

$$Y_i^{obs} * T_i = Y_i(1)\,T_i^2 + Y_i(0)(1 - T_i) * T_i = Y_i(1) * T_i$$

Esto se cumple dado que 1) $T_i^2 = T_i$ para $T_i \in 0, 1$ y 2) $T_i(1 - T_i)$ es siempre cero

$$Y_i^{obs} * (1 - T_i) = Y_i(1)\,T_i(1 - T_i) + Y_i(0)(1 - T_i)^2 = Y_i(0)$$

A partir de ahí el resto de la prueba es Ley de Esperanzas Iteradas:

$$E\Big[\frac{Y_i^{obs}\,T_i}{e(X_i)}\Big] = E_x\Big[E\Big[\frac{Y_i(1) * T_i}{e(X_i)}\Big|X_i\Big]\Big] =$$

$$E_x\Big[E\Big[\frac{E[Y_i(1)|X_i] * E[T_i|X_i]}{e(X_i)}\Big|X_i\Big]\Big] = E[Y(1)]$$

Inverse Probability Weightening Estimators (IPW) II

$$\hat{\tau} = \frac{1}{N}\Sigma_{i=1}^{N}\left(\frac{T_i * Y_i}{e(X_i)} - \frac{(1-T_i) * Y_i}{1-e(X_i)}\right)$$

Noten igual como, para cada i, tenemos (si sumamos la fracción):

$$Y_i^* = Y_i\left(\frac{T_i - e(X_i)}{e(X_i) * (1-e(X_i))}\right) = \tau_i$$

Wow! Ya tenemos un estimador observable del impacto a nivel individuo!

Significa que con esto ya podemos hacer Causal ML? No todavía :(

Algunos problemas con estos estimadores:

- Son muy inestables cuando $e(X_i)$ está cerca de 0 o 1 (overlap!!!)
- Requieren una estimación casi perfecta de $e(X_i)$

Augmented Inverse Probability Weightening (AIPW)

Los estimadores doble robustos son muy parecidos a los estimadores IPW, pero ajustan por $\bar{Y}, e(\bar{X}_i)$.

Esto genera que:

1. Son menos inestables que IPW
2. No requieren una estimación

Un ejemplo importante es el AIPW:

$$\tau_{AIPW} = \frac{1}{N} = \Sigma_{i=1}^{N} \Big(I\hat{PW} - I\bar{PW} * [e(\hat{X}_i) * \hat{Y}_{|T_i=0} - (1 - e(\hat{X}_i)) * \hat{Y}_{|T_i=1}] \Big)$$

Resumen de $e(X_i)$

- El propensity score es la mejor manera de balancear muchas X_s
- La mejor estimación de $e(X_i)$ es la que logra balances en todas las X_s
- Se requiere overlap $0 < e(X_i) < 1$
- Con el IPW, podemos dividir Y entre la probabilidad de estar tratado. Con esto podemos encontrar la variable latente Y_i^* que nos permite estimar $\hat{\tau}$ directamente.
- Estos estimadores son inestables a propensity scores cercanos a 0 o 1.
- Por ende, existen los estimadores doble robustos. Estos estiman $\hat{Y}, \hat{\tau}$. Su nombre viene de que no se necesita que las estimaciones sean perfectas.

Double LASSO (Belloni, Chernoszhukov y Hansen)

Los primeros intentos de usar Machine Learning en inferencia causal involucraron LASSO. Veamos cómo:

$$\hat{\tau} = \tau + \beta_1 \delta_1$$

- Estimas LASSO sobre $T_i = X\beta + \psi_i$. En principio, vas a dejar sólo las variables que tengan $\delta \neq 0$. (First Stage)

Double LASSO (Belloni, Chernoszhukov y Hansen)

Los primeros intentos de usar Machine Learning en inferencia causal involucraron LASSO. Veamos cómo:

$$\hat{\tau} = \tau + \beta_1 \delta_1$$

- Estimas LASSO sobre $T_i = X\beta + \psi_i$. En principio, vas a dejar sólo las variables que tengan $\delta \neq 0$. (First Stage)
- Estimas LASSO sobre $Y_i = X\beta + \epsilon_i$ (SIN T_i). En principio, vas a dejar sólo variables que tengan $\beta \neq 0$. (Second Stage)

Double LASSO (Belloni, Chernoszhukov y Hansen)

Los primeros intentos de usar Machine Learning en inferencia causal involucraron LASSO. Veamos cómo:

$$\hat{\tau} = \tau + \beta_1 \delta_1$$

- Estimas LASSO sobre $T_i = X\beta + \psi_i$. En principio, vas a dejar sólo las variables que tengan $\delta \neq 0$. (First Stage)
- Estimas LASSO sobre $Y_i = X\beta + \epsilon_i$ (SIN T_i). En principio, vas a dejar sólo variables que tengan $\beta \neq 0$. (Second Stage)
- Estimas OLS con la **UNIÓN** de las X_s seleccionadas en ambas etapas.

Double LASSO (Belloni, Chernoszhukov y Hansen)

Los primeros intentos de usar Machine Learning en inferencia causal involucraron LASSO. Veamos cómo:

$$\hat{\tau} = \tau + \beta_1 \delta_1$$

- Estimas LASSO sobre $T_i = X\beta + \psi_i$. En principio, vas a dejar sólo las variables que tengan $\delta \neq 0$. (First Stage)
- Estimas LASSO sobre $Y_i = X\beta + \epsilon_i$ (SIN T_i). En principio, vas a dejar sólo variables que tengan $\beta \neq 0$. (Second Stage)
- Estimas OLS con la **UNIÓN** de las X_s seleccionadas en ambas etapas.
- A esto se le llama **Two Stage Post LASSO** for Causal Inference.

Double LASSO (Belloni, Chernoszhukov y Hansen)

Los primeros intentos de usar Machine Learning en inferencia causal involucraron LASSO. Veamos cómo:

$$\hat{\tau} = \tau + \beta_1 \delta_1$$

- Estimas LASSO sobre $T_i = X\beta + \psi_i$. En principio, vas a dejar sólo las variables que tengan $\delta \neq 0$. (First Stage)
- Estimas LASSO sobre $Y_i = X\beta + \epsilon_i$ (SIN T_i). En principio, vas a dejar sólo variables que tengan $\beta \neq 0$. (Second Stage)
- Estimas OLS con la **UNIÓN** de las X_s seleccionadas en ambas etapas.
- A esto se le llama **Two Stage Post LASSO** for Causal Inference.
- Este estimador **acota** el sesgo por variables omitidas a valores muy pequeños. Esto es pues cualquier variable no incluida por construcción tiene $\beta \simeq 0$ y $\delta \simeq 0$.

Double LASSO (Belloni, Chernoszhukov y Hansen) II

Algunas notas sobre este método:

- El nombre Two Stage Post LASSO viene de las dos etapas. La parte POST viene de que en la tercera etapa se usa OLS en lugar de LASSO. Esto tiene los beneficios de obtener intervalos de confianza y p-values.
- Es importantísimo incluir la unión de la variables seleccionadas en cada etapa. Esto hace que el modelo sobre los impactos de T sobre Y sea **estructural**. Es decir, se incluyen **todo** lo que genera sesgo por variables omitidas
- El LASSO te ayuda a estimar no sólo las variables relevantes sino la forma funcional de la primera etapa ($T_i = g(X_i) + \psi_i$) y de la segunda etapa ($Y_i = m(X_i) + \epsilon_i$)

Double LASSO (Belloni, Chernoszhukov y Hansen) III

- El Two-Stage Post LASSO es un **estimador doble robusto**. Estimas $m(\hat{X}_i)$ y $g(\hat{X}_i)$, pero no necesitas una estimación perfecta de ninguna para que el Omitted Variable Bias ($\beta * \delta$) si hacemos un trabajo decente en ambas
- X puede ser una matriz de alta dimensionalidad por dos razones: 1) Ahora tenemos mucha información (Big data!) y 2) La estimación de las funciones. Esto hace que estas matrices sean `sparse`

Double Debiased Machine Learning

Chernoszhukov, Chetverikov, Duflo y Hansen (2018) propusieron un método parecido al de Belloni (2014), pero con dos mejoras sustanciales:

1. Se puede usar cualquier método de Machine Learning en las dos etapas
2. Sacan un método de estimación **Cross-Fitting** que permite insesgar los estimadores.

El problema se motiva así:

$$y_i = \tau T_i + m(X_i) + \epsilon_i$$

$$T_i = g(X_i) + u_i$$

Donde $m(X_i)$ y $g(X_i)$ son **nuisance functions** desconocidas de todas las variables. Estas pueden tener cualquier forma (lineal y no lineal) y ser estimadas con cualquier método de ML.

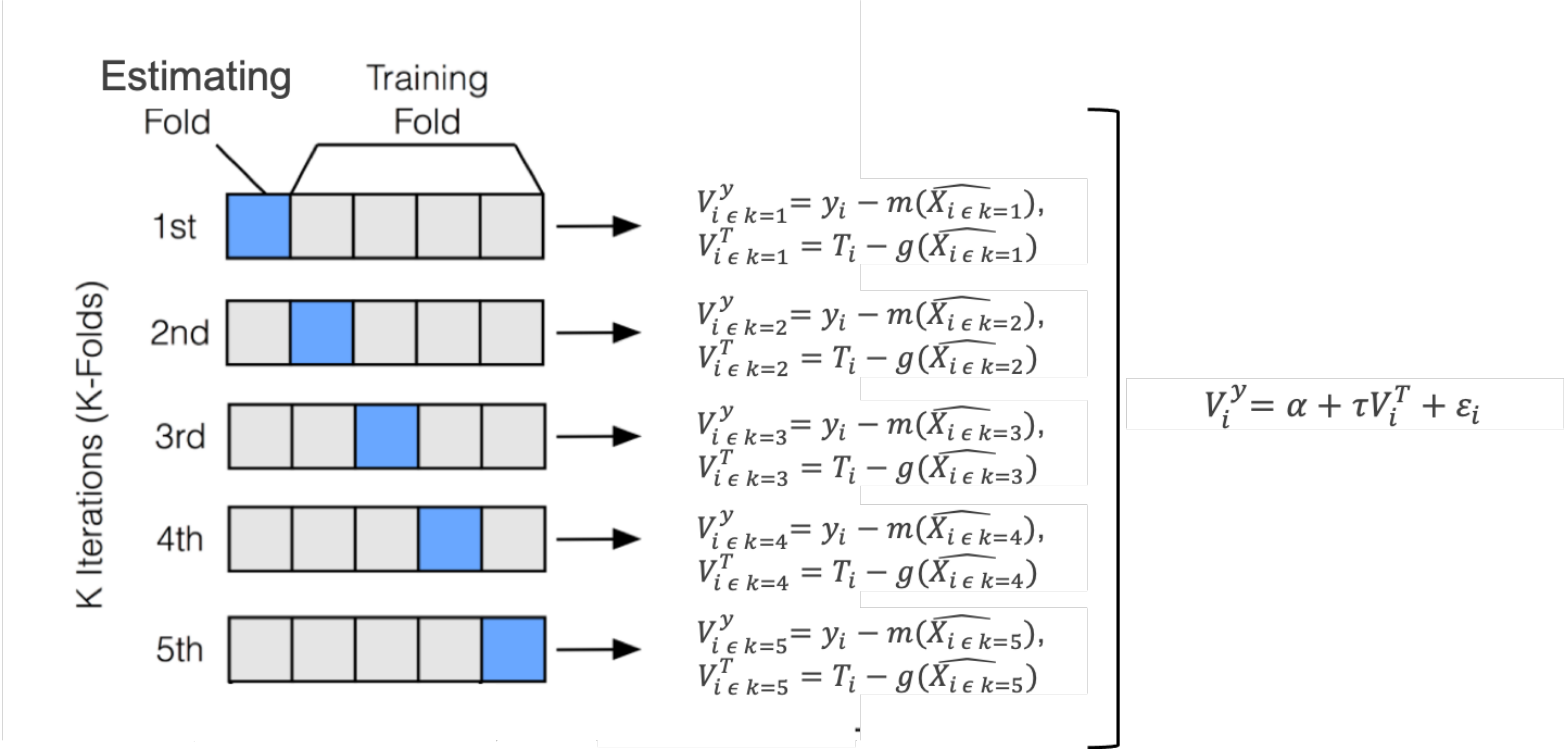
Double Debiased Machine Learning II

Pasos:

1. Divide tu muestra en K partes de manera aleatoria
2. Estima $g(\hat{X_{i \in K^c}})$ en K-1 partes usando LASSO, Random Forest, XGB u otro método.
3. Predice $g(\hat{X_{i \in K}})$ y $V^y_{i \in K} = y_{i \in K} - g(\hat{X_{i \in K}})$
4. Estima $m(\hat{X_{i \in K^c}})$ en K-1 partes usando LASSO, Random Forest, XGB u otro método.
5. Predice $m(\hat{X_{i \in K}})$ y $V^T_{i \in K} = T_{i \in K} - m(\hat{X_{i \in K}})$
6. Obtén τ al correr la regresión de $V^y_i = \alpha + V^T_i + \psi_i$

Double Debiased Machine Learning II

Noten la parte del **cross-fitting**. Entrenas en todas vs una parte las nuisance functions y estimas (creas residuales) en la otra.



Cross-fitting

Los autores muestran que el cross-fitting es **crucial** para obtener estimadores insesgados/causales. Esto también se puede denominar **honesty**.

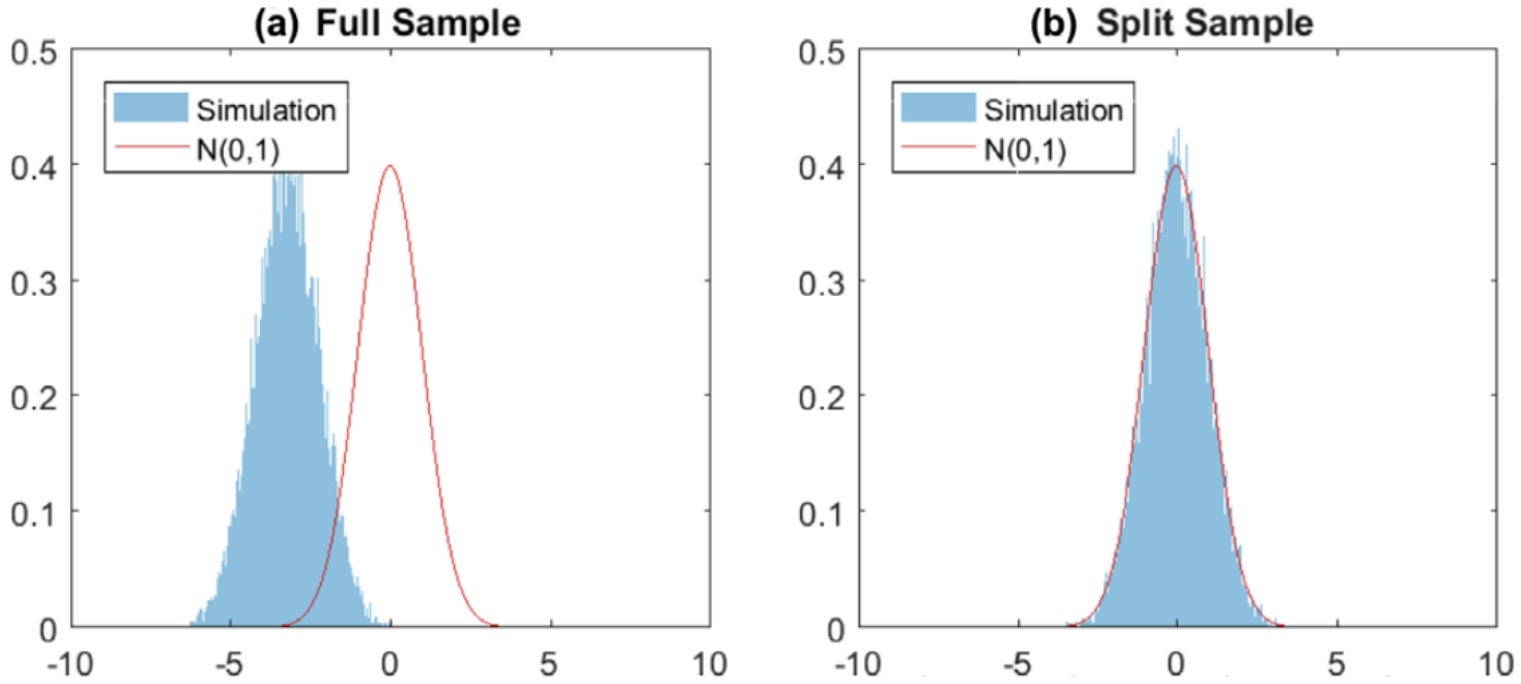


Figure 2. Comparison of full-sample and cross-fitting procedures. [Colour figure can be viewed at wileyonlinelibrary.com]

Double Debiased Machine Learning II

Finalmente, noten como el algoritmo:

- Aprovecha ML para crear funciones que detectan todas las señales. Esto reduce los errores a un mínimo, pues nos da podemos estimar funciones muy complejas
- Usa el concepto de **partialling out**. $m(X_i)$ y $g(X_i)$ contienen la varianza de y_i y T_i explicada por X.
- Al crear los residuales V_i^y y V_i^T, estamos estimando la relacion entre toda la varianza no explicada por X_i de y_i vs T_i. Ergo, causal!
- Finalmente, la estimación es doble robusta porque, al estimar ambas nuisance functions, no se necesita perfección en ninguna.

DDML en R

```
##############################
# Creando la k aleatoriamente
##############################
base_training <-
  base_training %>%
  arrange(NUM_CLIE)

k<-treatment_assign(data = base_training, share_control = 0.2,
                    n_t = 4,  strata_varlist = "NUM_CLIE",
                    missfits = "global",
                    seed = 1900, key = "NUM_CLIE")$data
k<-
  k %>%
  mutate(k = treat + 1) %>%
   ungroup()
base_training<-bind_cols(base_training, k %>% select(k))
k<-k$k
```

DDML en R II

```
# Cross-fitting
modelo<-map_dfr(1:5,
   function(a) {
     treat_fit <-gamlr(x = X[k!=a, , drop= F],
                       y = trat[k !=a],family="binomial")
     treat_hat<-as.numeric(predict(treat_fit,
                                   newdata = X[k==a, , drop= F],
                                   type = "response"))

     monto_fit <-gamlr(x = X[k!=a, , drop= F],
                       y = monto_ret[k !=a])
     monto_hat<-as.numeric(predict(monto_fit,
                                   newdata = X[k==a, , drop= F],
                                   type = "response"))
```

DDML en R III

```
      treat_resid <- trat[k==a] - treat_hat
      resid_monto <- monto_ret[k==a]- monto_hat

      fits<-bind_cols("treat_hat" = treat_hat,
                      "monto_hat"= monto_hat,
                      "resid_treat" = treat_resid,
                      "resid_monto" = resid_monto)
   })

# monto_ret
monto_ret_ate<-lm(resid_monto~ resid_treat,
                  data = modelo) %>%
   tidy()
```

Approximate Residual Balancing

Athey y Wager (2019) argumentaron que el Two Stage Post LASSO puede ser mejorado de otra manera.

En lugar de estimar el propensity score (con las dificultades de ello) para rebalancear a la Rosenbaum, **usan pesos** para balancear los grupos.

Este proceso usa los pesos para construir grupos tan balanceados como en un experimento. Los autores argumentan que esto es mejor ya que balancea por todas las variables en X_i y no sólo las importantes.

Pasos:

1. Estimas $y_i = \tau T_i + m(X_i) + \epsilon_i$ para $i \in T_i = 0$. Este se encarga de las señales fuertes en el control
2. Estimas $y_i = \tau T_i + m(X_i) + \epsilon_i$ para $i \in T_i = 1$. Este se encarga de las señales fuertes en el tratamiento
3. Calculas los pesos que balancean los grupos lo más que se pueda.
4. Usas los pesos de (3) para estimar τ

balanceHD en R

Los autores crearon una librería para usar su algoritmo en `devtools::install_github("swager/balanceHD")`.

Estimate ATE via approximate residual balancing

Description

Estimate ATE via approximate residual balancing

Usage

```
residualBalance.ate(X, Y, W, target.pop = c(0, 1),
  allow.negative.weights = FALSE, zeta = 0.5, fit.method = c("elnet",
  "none"), alpha = 0.9, scale.X = TRUE, estimate.se = FALSE,
  optimizer = c("mosek", "pogs", "pogs.dual", "quadprog"),
  bound.gamma = FALSE, verbose = FALSE)
```

Arguments

`X`	the input features
`Y`	the observed responses
`W`	treatment/control assignment, coded as 0/1
`target.pop`	which population should the treatment effect be estimated for? (0, 1): average treatment effect for everyone 0: average treatment effect for controls 1: average treatment effect for treated
`allow.negative.weights`	whether negative gammas are allowed for balancing
`zeta`	tuning parameter for selecting approximately balancing weights
`fit.method`	the method used to fit mu(x, w) = E[Y | X = x, W = w]
`alpha`	tuning paramter for glmnet
`scale.X`	whether non-binary features should be noramlized
`estimate.se`	whether to return estimate of standard error
`optimizer`	which optimizer to use for approximate balancing
`bound.gamma`	Whether upper bound on gamma should be imposed. This is required to guarantee asymptotic normality, but increases computational cost.
`verbose`	whether the optimizer should print progress information

Value

ATE estimate, along with (optional) standard error estimate

Conclusiones

En esta clase vimos los primeros algoritmos para hacer inferencia causal en Big Data apalancándonos de algoritmos de ML:

- El sesgo por variables omitidas es $\beta * \delta$ en $\hat{\tau} = \tau + \beta_1 \delta_1$
- Los propensity scores son **cruciales** para alcanzar balances en altas dimensiones.
- Estos se pueden 1) Estimar directamente via ML o 2) Resolver por los pesos que generan mayor balance
- El partialling out (conditional unconfoundness) está detrás de todos estos algoritmos
- Finalmente, el cross-fitting ó honesty (depende del autor) es un ejercicio importantísimo para asegurar estimadores insesgados.

La siguiente clase veremos como estos algoritmos (DDML) y otros nuevos (Causal Forests) extienden esta literatura para encontrar no sólo τ sino τ_i