

ML for SS: Causal Machine Learning

Dr. Richard M. Crowley

rcrowley@smu.edu.sg

<https://rmc.link/>



Overview

Papers

Chernozhukov et al. 2017 AAER

- Introduces a ML-based method for causal identification useful in standard DID and IV approaches
 - Focused on calculating ATE and ATTE

Deryugina et al. 2019 AAER

- Uses XGBoost + DoubleML for an interesting application: deaths caused by pollution
- The study is well designed, showing results using both traditional econometrics and ML-based econometrics

Crowley et al. 2023a Working

- Examines the impact of misinformation laws in China
- Implements DoubleML-DiD to measure the effect

Technical Discussion: DoubleML

Python

- Use the [DoubleML](#) library
 - For a basic DoubleML model
 - For a DiD model from Crowley et al. (2023a)

R

- The [doubleML](#) library is available in R as well
 - We'll try this for a model with clustered standard errors
- [{dm1mt}](#) for multilevel treatments using DoubleML
 - We'll try this for a model with clustered standard errors
- Both the above are implemented in Crowley et al. (2023b)
- The AER paper's source code is all in R!



Double ML: Theory

Background

- There are a number of relevant papers published in economics in recent years developing and using Double ML
- The method is developed largely from:
 - Chernozhukov et al. (2017 AER), “Double/debiased/Neyman machine learning of treatment effects”
 - Chernozhukov et al. (2018 Econometrics J), “Double/debiased machine learning for treatment and structural parameters.”

Impact or overlap with methodological work by Susan Athey, Matthew Gentzkow, Trevor Hastie, Guido Imbens, Matt Taddy, and Stefan Wager

What is Double ML?

1. Split your sample as you would for K -fold cross validation, into sets $\{I_k\}_{k \in \{1, \dots, K\}}$
 - K samples of N/K observations each
 - Let $I_k^c = \cup\{I_j\}_{j \neq k}$
2. Construct K estimators using a machine learning estimator over nuisance parameters (e.g., controls) applied to the data I_k^c
3. Average the K estimators to obtain a final estimator
 - This average estimator is approximately unbiased and normally distributed
 - The estimator is also asymptotically efficient

And repeat. Bootstrap this out and take the mean or median of the estimators

Where Double ML excels: Endogenous treatment

- Suppose a policy affects a subset of individuals (people, corporations, etc.)
- Suppose individuals have the ability to alter their treatment status
 - E.g., state laws (move), labor laws, etc.
- Linear controls may be insufficient to claim causality of the treatment on anything

There are a lot of older methods that try to address this, though incompletely

1. Linear controls
2. Propensity score adjustments (e.g., weighting)
3. Matching methods
4. “doubly-robust” estimators

Why is machine learning needed?

- Suppose a true form of a specification is as follows
 - T is a treatment indicator, C is a vector of controls

$$Y = g_0(T, C) + \varepsilon_1$$

$$T = m_0(C) + \varepsilon_2$$

- We often assume g_0 to be something like $\alpha + \theta_0 T + \gamma \cdot C$
- We often assume m_0 to be a constant (i.e., assume that T is exogenous)

We know these assumptions aren't true! (in many cases)

Why is machine learning needed?

How can we estimate a more general form for g_0 and m_0 ?

- We could use a more flexible econometric approach, such as including interactions between T and C
 - This is still very restrictive: purely linear
- We could include transformations of C and its interactions
 - This is still restrictive: T is additively separable
- We could use a nonparametric estimator!
 - This is where machine learning is very useful: efficient and reasonably accurate nonparametric estimation
 - LASSO, random forest, XGBoost, etc.

Model variants

- Interactive regression model (IRM)
 - The model described in the previous slides
- Partially linear regression model (PLR)
 - Use if you can separate your treatment effect from the controls but suspect nonlinear effects of controls
 - Solves $Y = \theta_0 T + g_0(C) + \varepsilon_0$ and $T = m_0(C) + \varepsilon_2$
- There are also instrumental variable variants of both IRM and PLR

What does this give us?

- Average treatment effect (ATE)
 - How does the treatment effect the outcome across groups?
 - $\mathbb{E} [g_0(1, C) - g_0(0, C)]$
- Average treatment effect of the treated (ATTE)
 - How does the treatment effect only those under the treatment?
 - $\mathbb{E} [g_0(1, C) - g_0(0, C) | T = 1]$

Reconciling these slides notation with the paper

- These slides use a somewhat simpler oriented notation.
- Reconciliation from slides to papers:
 - T is D
 - C is X
 - ε_0 is U or ζ depending on the paper
 - ε_1 is V



Implementing DoubleML

Walking through an implementation of DoubleML

Problem: How does 401k participation impact wealth?

- This problem is walked through in Chernozhukov et al. (2017 AER, Web Appendix)
 - The R code for the AER paper is available from AER as well
 - Quite clean code at that!
- We will implement this in python using the [DoubleML](#) library
 - Which Chernozhukov was involved in the development of

Importing the data

- Conveniently, the data is available from the DoubleML package

```
# Grab the dataset
import doubleml.datasets
df = dml.datasets.fetch_401K('DataFrame')
df
```

	nifa	net_tfa	tw	age	inc	...	twoearn	e401	p401	pira	hown
0	0.0	0.0	4500.0	47	6765.0	...	0	0	0	0	1
1	6215.0	1015.0	22390.0	36	28452.0	...	0	0	0	0	1
2	0.0	-2000.0	-2000.0	37	3300.0	...	0	0	0	0	0
3	15000.0	15000.0	155000.0	58	52590.0	...	1	0	0	0	1
4	0.0	0.0	58000.0	32	21804.0	...	0	0	0	0	1
...
9910	98498.0	98858.0	157858.0	52	73920.0	...	0	1	1	0	1
9911	287.0	6230.0	15730.0	41	42927.0	...	1	1	1	1	1
9912	99.0	6099.0	7406.0	40	23619.0	...	0	1	0	1	0
9913	0.0	-32.0	2468.0	47	14280.0	...	0	1	1	0	0
9914	4000.0	5000.0	8857.0	33	11112.0	...	0	1	1	0	0

[9915 rows x 14 columns]

Using your own data

- We can also do this manually, by importing the Stata file from AER
- We then need to prep the data into the format `DoubleML` expects
 - This is fairly straightforward, just defining our Y, treatment, and control variables



```
df = pd.read_stata('../Data/S8_sipp1991.dta')

y = 'net_tfa'
treat = 'e401'
controls = [x for x in df.columns.tolist() if x not in [y, treat]]

df_dml = dml.DoubleMLData(df, y_col=y, d_cols=treat, x_cols=controls)
```

What is the data format used by DoubleML?

```
Python | print(df_dml)

===== DoubleMLData Object =====

----- Data summary -----
Outcome variable: net_tfa
Treatment variable(s): ['e401']
Covariates: ['nifa', 'tw', 'age', 'inc', 'fsize', 'educ', 'db', 'marr', 'twoearn', 'p401', 'pira', 'hown']
Instrument variable(s): None
No. Observations: 9915


----- DataFrame info -----
<class 'pandas.core.frame.DataFrame'>
Int64Index: 9915 entries, 0 to 9914
Columns: 14 entries, nifa to hown
dtypes: float32(4), int8(10)
memory usage: 329.2 KB
```

- Pandas dataframe
- A pre-specified outcome variable
- One or more treatment indicators
- One or more controls
- Optional instruments


Set up the Nuisance functions

- Recall that there are two functions, m_0 and g_0 that need to be solved for this method
 - m_0 must match the treatment's form, g_0 the DV's form
- We can specify any form for these that we want, if they follow Scikit-learn's API


g_0 : *Continuous GBM*

```
 g_0 = GradientBoostingRegressor(  
    loss='ls',  
    learning_rate=0.01,  
    n_estimators=1000,  
    subsample=0.5,  
    max_depth=2  
)
```

m_0 : *Binary GBM*

```
 m_0 = GradientBoostingClassifier(  
    loss='exponential',  
    learning_rate=0.01,  
    n_estimators=1000,  
    subsample=0.5,  
    max_depth=2  
)
```

Run the DML model: Average Treatment Effects

```
 # Fix the random number generator for replicability  
np.random.seed(1234)  
# Run the model  
dml_model_irm = dml.DoubleMLIRM(df_dml, g_0, m_0)  
# Output the model's findings  
print(dml_model_irm.fit())
```

=====
DoubleMLIRM Object
=====

----- Data summary -----

Outcome variable: net_tfa

Treatment variable(s): ['e401']

Covariates: ['nifa', 'tw', 'age', 'inc', 'fsize', 'educ', 'db', 'marr', 'twoearn', 'p401', 'pira', 'hown']

Instrument variable(s): None

No. Observations: 9915

----- Score & algorithm -----

Score function: ATE

DML algorithm: dml2

----- Resampling -----

No. folds: 5

No. repeated samples: 1

Run the DML model: ATTE

- ATTE: Average Treatment Effects of the Treated

```
# Run the model
dml_model_irm_ATTE = dml.DoubleMLIRM(df_dml, g_0, m_0, score='ATTE')
# Output the model's findings
print(dml_model_irm_ATTE.fit())
```

==== DoubleMLIRM Object =====

----- Data summary -----

Outcome variable: net_tfa

Treatment variable(s): ['e401']

Covariates: ['nifa', 'tw', 'age', 'inc', 'fsize', 'educ', 'db', 'marr', 'twoearn', 'p401', 'pira', 'hown']

Instrument variable(s): None

No. Observations: 9915

----- Score & algorithm -----

Score function: ATTE

DML algorithm: dml2

----- Resampling -----

No. folds: 5

No. repeated samples splits: 1

Other twists on the model

1. Change the machine learning backend

- Our models used `dm12`
- You can switch to `dm11` using `dm1_procedure='dm11'`
- `dm11` follows the math in these slides
 - Solve for a condition equal to zero for each model, and then average the estimators
 - `dm12` solves for the average of the condition being equal to zero overall

2. Run multiple iterations of the model

- The paper uses 100 iterations, emulate this by adding `n_rep=100`

3. Change the machine learning models fed to the DoubleML model

- An example of using “Histogram-based Gradient Boosting” is in the Jupyter notebook
 - This is a much faster GBM-like model



DoubleML DiD

An example of DML DiD

- On the next slides, I present the code for the DML DiD test from Crowley et al. 2023a
- The code uses an estimator that is doubly robust following Sant'Anna and Zhao (2020)
- The code uses the DiD approach from Chang (2020)

Pros

- DiD with DoubleML!

Cons

- Hard to use many Fixed Effects
- No model for clustered standard errors (yet)
- Only works with basic DiD estimation

Initializing the model in Python


Here we define m_0 in `ml_m` and g_0 in `ml_g`. This paper uses XGBoost as the underlying algorithm, so the model is fully nonparametric.



```
# Set up the ML functions to use within the DoubleML routine
# Initial
ml_g = ml_l_xgb = XGBRegressor(objective = "reg:squarederror", eta = 0.1,
                              n_estimators = 100)
ml_m = XGBClassifier(use_label_encoder = False,
                    objective = "binary:logistic",
                    eval_metric = "logloss",
                    eta = 0.1, n_estimators = 100)

# Set a random seed for replicability
np.random.seed(65474)
```

Setting up the data and estimator

```
 # Build the data object
data = df[['Avg_Fake', 'treat', 'post_reg', 'Size_w', 'ROA_w', 'Market_to_Book_w',
          'Leverage_w', 'SOE_indicator', 'dailyret_w', 'retvol_qtr_w',
          'disclosure_rating', 'avg_daily_overall_tone_w', 'log_avg_total_words',
          'log_daily_post_count']]
obj_dml_data = dml.DoubleMLData(data, 'Avg_Fake', 'treat', t_col='post_reg')

# Compile the DMLDiD estimator
dml_did_obj = dml.DoubleMLDIDCS(obj_dml_data, ml_g, ml_m)

# Run and output the results
output = dml_did_obj.fit() # Takes 7 minutes using 16 cores

print(output) # shown on the next slide
```

- The above code is very similar to our base DoubleML model
 - Just note the use of `dml.DoubleMLDIDCS()` instead of `dml.DoubleMLIRM()`

Model output

```
===== DoubleMLDIDCS Object =====  
  
----- Data summary -----  
Outcome variable: Avg_Fake  
Treatment variable(s): ['treat']  
Covariates: ['Size_w', 'ROA_w', 'Market_to_Book_w', 'Leverage_w', 'SOE_indicator', 'dailyret_w', 'retvol_qtr_w',  
'disclosure_rating', 'avg_daily_overall_tone_w', 'log_avg_total_words', 'log_daily_post_count']  
Instrument variable(s): None  
Time variable: post_reg  
No. Observations: 2633704  
  
----- Score & algorithm -----  
Score function: observational  
DML algorithm: dml2
```

Machine Learning



DoubleML with clustered standard errors

An example of DML with clustered SEs

- On the next slides, I present the code for the clustered standard error DoubleML model from Crowley, Lou, Tan, and Zhang (2023b)
- The model is from Chiang et al. (2022)
 - Bootstrapping and sampling over the clustering variables

Pros

- Clustering with DoubleML!
 - One-way
 - Two-way

Cons

- Hard to use many Fixed Effects
- Only works with basic treatment effect estimation
- High computational cost and data requirements

Initializing the model in R

```
R # Required imports
library('hdm')
library('DoubleML')
library('mlr3')
library('mlr3learners')

# Note that all the data is in `df`

# Data setup (building a data.frame for it)
formula = formula(~ -1 + mcap + roa + dt_at + log_followers + log_friends +
                  log_total_tweets + as.factor(year) + event + as.factor(sic1))
data_transf = data.frame(model.matrix(formula, df))

# Set up the learners using glmnet under the hood
lasso_l = lrn("regr.cv_glmnet", nfolds = 10, s = "lambda.min")
lasso_m = lrn("classif.cv_glmnet", nfolds = 10, s = "lambda.min")
```

- Here we use a LASSO model, so this is only semi-parametric
 - `lasso_l` is g_0 , `lasso_m` is m_0

Setting up the data and estimator

```
R # Constructing the DML data object
y_col = 'log_tweets_num'
d_col = 'misinformation'
cluster_cols = c('region', 'year')
dml_df = cbind(df[c(y_col, d_col, cluster_cols)],
              data_transf)

dml_data = DoubleMLClusterData$new(dml_df,
                                   y_col=y_col,
                                   d_cols=d_col,
                                   cluster_cols=cluster_cols,
                                   x_cols=names(data_transf))

set.seed(1111)
#dml_data$z_cols = z_col
#dml_data$cluster_cols = c('model.id', 'cdid')
n_rep = 10
dml_pl = DoubleMLPLR$new(dml_data,
                        lasso_l, lasso_m,
                        n_folds=2, n_rep=n_rep)
```

- Note that the R `doubleml` package allows for specifying clustering via the `cluster_cols` argument in the data construction function

Model output

```
R |dml_pl$summary()
```

Estimates and significance testing of the effect of target variables

	Estimate.	Std. Error	t value	Pr(> t)
misinformation	-0.6509	0.2956	-2.202	0.0277 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

- Thi model does not give coefficient estimates



DoubleML with multilevel treatments

An example of multilevel treatment DML

- On the next slides, I present the code for the multilevel treatment DoubleML model from Crowley, Lou, Tan, and Zhang (2023b)
- The model is from Knaus (2020)

Pros

- Supports any level of treatment, 2 levels or more
- Useful with stacked regressions
- Can also compare the ATEs across levels

Cons

- Hard to use many Fixed Effects
- No model for clustered standard errors (yet)

Initializing the model

```
R | # imports  
  | library(dmlmt)
```

- No need to set up the ML functions with `{dmlmt}`
 - It uses cross-validated post-LASSO by default

Setting up the data and estimator

```
R # Build the data
Y = df$log_tweets_num
tier1 <- c('cn2', 'sg', 'ru')
tier2 <- c('de', 'fr')
tier3 <- c('be', 'it', 'nl')
D_mult = ifelse(df$event %in% tier1, 1,
               ifelse(df$event %in% tier2, 2,
                     ifelse(df$event %in% tier3, 3,
                           0))) * df$post * df$treat
X4 = model.matrix(~ -1 + mcap + roa + dt_at + log_followers + log_friends +
                 log_total_tweets + as.factor(year) + as.factor(sic1),
                 data = df)

# Compile and run DMLDiD estimator
stand_l_bin4 <- dmlmt(X4, D_mult, Y, pl=TRUE)
```

- No need to build a DoubleML data object
 - **pl=TRUE** specifies to use post-LASSO
 - Post-LASSO is slower but more rigorous

Model output

```
Multiple treatment
```

```
Potential outcomes:
```

```
# Treatment 0 4.7123 0.0200
# Treatment 1 4.0268 0.2991
# Treatment 2 4.8360 0.0595
# Treatment 3 4.4298 0.0928
#
# Average effects
# TE      SE      t      p
# T1 - T0 -0.685500 0.299019 -2.2925 0.0218828 *
# T2 - T0  0.123747 0.059220  2.0896 0.0366596 *
# T3 - T0 -0.282490 0.092132 -3.0661 0.0021701 **
# T2 - T1  0.808240 0.204210  3.9581 0.0000142 **
```

- Level 0 is the control level; levels 1-3 are treatments
- Note that the output shows us the ATE of the treatments and the difference across treatments



Conclusion

Wrap-up

Double Machine Learning can help in cleanly identifying treatment effects

- Easy to implement as well!
- Many models now available, making it more general-purpose

ML and Econometrics are not at odds with one another

- You can use ML to strengthen an econometric framework

ML is essentially just another tool in the econometrics toolbox!

Packages used for these slides

Python

- doubleML
- numpy
- pandas

R

- {doubleML}
- {dm1mt}
- glmnet
- {hdm}
- kableExtra
- knitr
- mlr3
- mlr3learners
- quarto
- reticulate
- revealjs

References

- Chang, Neng-Chieh. “Double/debiased machine learning for difference-in-differences models.” *The Econometrics Journal* 23, no. 2 (2020): 177-191.
- Chernozhukov, Victor, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, and Whitney Newey. “Double/debiased/neyman machine learning of treatment effects.” *American Economic Review* 107, no. 5 (2017): 261-65.
- Chiang, Harold D., Kengo Kato, Yukun Ma, and Yuya Sasaki. “Multiway cluster robust double/debiased machine learning.” *Journal of Business & Economic Statistics* 40, no. 3 (2022): 1046-1056.
- Crowley, Richard M., Yun Lou, Samuel T. Tan, and Liandong Zhang. “Does Misinformation Regulation Reduce Fake News in Financial Markets? Evidence from East Guba.” Working paper, Singapore Management University (2023a).
- Crowley, Richard M., Yun Lou, Samuel T. Tan, and Liandong Zhang. “Misinformation Regulations: Early Evidence on Corporate Social Media Strategy.” Working paper, Singapore Management University (2023b).
- Deryugina, Tatyana, Garth Heutel, Nolan H. Miller, David Molitor, and Julian Reif. “The mortality and medical costs of air pollution: Evidence from changes in wind direction.” *American Economic Review* 109, no. 12 (2019): 4178-4219.
- Knaus, Michael C. “A double machine learning approach to estimate the effects of musical practice on student’s skills.” *Journal of the Royal Statistical Society Series A: Statistics in Society* 184, no. 1 (2021): 282-300.
- Sant’Anna, Pedro HC, and Jun Zhao. “Doubly robust difference-in-differences estimators.” *Journal of Econometrics* 219, no. 1 (2020): 101-122.

