Double Machine Learning (DML)

Professor Song Yao
Olin Business School

Customer Analytics

Why do we need yet another causal inference tool?

Challenges in the New Era

- High dimension data
  - Linear/Logit break down—esp. difficult with complex interactions
  - Matching (KNN/PSM/IPW) suffers poor overlap
- How about use ML when calibrating propensity scores?
  - Overfitting bias
  - Sensitivity to misspecification
  - Standard error not properly accounted for
- What we need:
  - Handle high-dimentional data
  - ML's flexibility
  - Valid causal effect inference

Double Machine Learning (DML)

Chernozhukov et al. (2018)*

- Combine ML with *orthogonalization*
  - Handle high-dimensional data
  - ML
  - Maintain valid causal estimates and inference
- Widely used in the industry

* In the original paper, the model is called "Double/Debiased Machine Learning"

What is Orthogonalization?

Frisch-Waugh-Lovell (FWL) Theorem $Y = \beta_1 D + \beta_2 X + \epsilon$

- We are interested in β_1—X is called "nuisance" as we don't care about it
- We normally regress Y on D and X
  - $\hat{Y} = \hat{\beta}_1 D + \hat{\beta}_2 X$
- Alternatively, an overkill and redundant approach with THREE regressions
  - Regress Y on X and get the residual $\tilde{Y}$,
    $\hat{Y}* = \hat{\gamma}_2 X;\ \tilde{Y} = Y - \hat{Y}*$
  - Regress D on X and get the residual $\tilde{D}$,
    $\hat{D}* = \hat{\theta}_2 X;\ \tilde{D} = D - \hat{D}*$
  - When regressing $\tilde{Y}$ on $\tilde{D}$, we gets exactly the same estimate $\hat{\beta}_1$

An Numerical Example to Demo FWL

Ice Cream Sales Data

```python
# load data
icecream = pd.read_csv('https://songyao21.github.io/course_data/ice_cream_sales.csv')
display(icecream)
```

	temp	weekday	cost	price	sales
0	17.3	6	1.5	5.6	173
1	25.4	3	0.3	4.9	196
2	23.3	5	1.5	7.6	207
3	26.9	1	0.3	5.3	241
4	20.2	1	1.0	7.2	227
...	...	...	...	...	...
9995	24.1	5	0.3	5.7	184
9996	26.1	2	0.3	5.3	191
9997	22.0	6	0.3	5.2	171
9998	21.9	1	1.0	7.5	214
9999	20.0	7	0.5	5.9	237

- sales: outcome
- price: treatment
- temp, weekday, cost:
  - control covariates (nuisance)

An Numerical Example to Demo FWL

```python
### OLS
OLS = smf.ols("sales ~ price + temp + C(weekday) + cost",
              data=icecream).fit()
print(OLS.summary().tables[1])
```

```
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept        201.0207      1.005    200.072      0.000     199.051     202.990
C(weekday)[T.2]  -33.3613      0.433    -76.978      0.000     -34.211     -32.512
C(weekday)[T.3]  -32.9829      0.441    -74.780      0.000     -33.847     -32.118
C(weekday)[T.4]  -32.9927      0.440    -75.008      0.000     -33.855     -32.130
C(weekday)[T.5]  -32.9545      0.437    -75.456      0.000     -33.811     -32.098
C(weekday)[T.6]  -32.9413      0.443    -74.340      0.000     -33.810     -32.073
C(weekday)[T.7]    0.3027      0.420      0.721      0.471      -0.521       1.126
price             -3.9746      0.113    -35.316      0.000      -4.195      -3.754
temp               1.8523      0.029     64.416      0.000       1.796       1.909
cost               3.2713      0.272     12.033      0.000       2.738       3.804
==================================================================================
```

```python
### Frisch-Waugh-Lovell
resid_y_ols = smf.ols("sales ~ temp + C(weekday) + cost", data=icecream).fit()
resid_p_ols = smf.ols("price ~ temp + C(weekday) + cost", data=icecream).fit()
smf.ols("sales_res ~ price_res",
        data=icecream.assign(sales_res=resid_y_ols.resid, # sales residuals
                             price_res=resid_p_ols.resid) # price residuals
       ).fit().summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-2.24e-13	0.114	-1.97e-12	1.000	-0.223	0.223
price_res	-3.9746	0.112	-35.330	0.000	-4.195	-3.754

Frisch-Waugh-Lovell Helps with Orthogonalization

- Regress Y on X and get the residual
  - If correctly specified, the residual $\tilde{Y}$ is uncorrelated to X (orthogonal)
- Regress D on X and get the residual,
  - If correctly specified, the residual $\tilde{D}$ is uncorrelated to X (orthogonal)
- The above two regressions "remove all the influences" of X
- Regressing $\tilde{Y}$ on $\tilde{D}$ is free of the influence of X

How Does ML Help with Orthogonalization?

Need Correct Specifications in the First Two Regressions

- However, we need the first two regressions are correctly specified (e.g., properly account for nonlinearity and interactions, etc.)

- $Y = \gamma_2 X$ may be incorrectly specified
  - The residual has some remnants of X
- $D = \theta_2 X$ may be incorrectly specified
  - The residual has some remnants of X
- So when regress the residual on residual, we may still have the effect of X

Frisch-Waugh-Lovell on Steroids

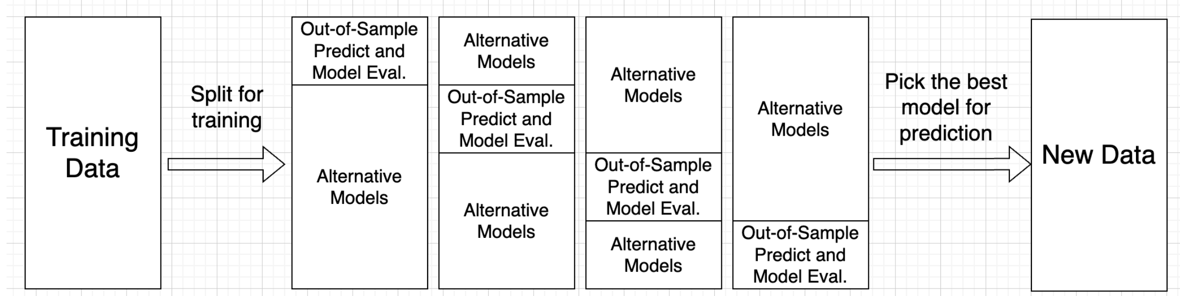
- ML comes to the rescue
  - $Y = g(X)$, ML-based estimation
  - $D = f(X)$, ML-based estimation (this is reason behind the "Double")
  - Get the residuals from the first and the second regressions
  - Regress residual on residual to get the effect of D on Y

Frisch-Waugh-Lovell on Steroids

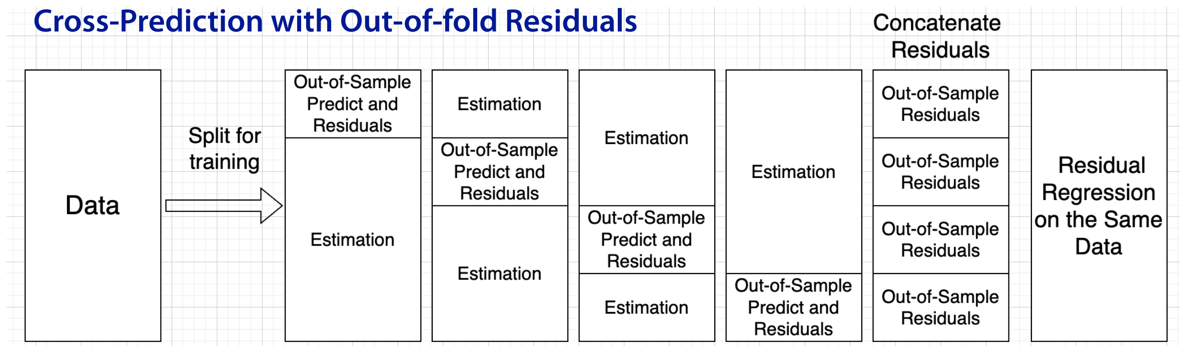
- To mitigate overfitting (and data leakage), we will also use *K-fold cross prediction to obtain out-of-fold residuals*
- Overfit $g(X)$:
  - Reduce bias so the residual $\tilde{Y}$ becomes too small—part of D's effect on Y is captured by $g(X)$
  - In the residual regression, D's estimate is biased towards 0
- Overfit $f(X)$:
  - Reduce bias so the residual $\tilde{D}$ becomes too small—less variation in $\tilde{D}$
  - In the residual regression, D's estimate may have a large standard error and become insignificant (*lack of statistical power*)

K-fold Cross Prediction and Out-of-fold Residuals

Cross-Validation



Cross-Prediction with Out-of-fold Residuals



ML Algorithms: Which One to Use?

LightGBM and RF are the most common choices

Use Case	**Recommended ML Model**	**Why**
Tabular data (e.g., all data we consider in this course)	**Random Forest, LightGBM**	Handles nonlinearity, interactions, minimal tuning
High-dimensional sparse data	**Lasso / ElasticNet**	Strong regularization, interpretable
Large-scale tabular + GPU	**XGBoost, LightGBM**	Fast, scalable
Deep interactions / complex data	**Neural nets**	Powerful but harder to tune, risk of overfitting
Mixed continuous/categorical	**CatBoost**	Handles both well

DML Implementation—Setup

Using LGBM

```python
## DML with LGBMRegressor
from lightgbm import LGBMRegressor
from sklearn.model_selection import cross_val_predict

# specify the variables
Y = icecream["sales"]
D = icecream["price"]
X = icecream[["temp", "weekday", "cost"]]

# Create the base model
lgbm = LGBMRegressor(
    objective='regression',
    max_depth=3,
    random_state=42,
    n_jobs=-1,  # Use all available cores
    verbose=-1  # Not show progress, replace with 1 to show progress
)
```


DML Implementation—Three Regressions

```python
## Debiasing Y, getting residuals of Y~X
icecream['sales_resid'] = Y - cross_val_predict(lgbm, X, Y, cv=5)
## Debiasing D, getting residuals of D~X
icecream['price_resid'] = D - cross_val_predict(lgbm, X, D, cv=5)
## Estimate the DML model with the residuals regression
DML_model = smf.ols(formula='sales_resid ~ price_resid', data=icecream).fit()
DML_model.summary().tables[1]
```

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	-0.0080	0.072	-0.110	0.912	-0.150	0.134
price_resid	$\boxed{-3.8890}$	0.071	-54.498	0.000	-4.029	-3.749

Was -3.97 with OLS

DML Implementation—Using a Library (DoubleML, EconML, and many others)

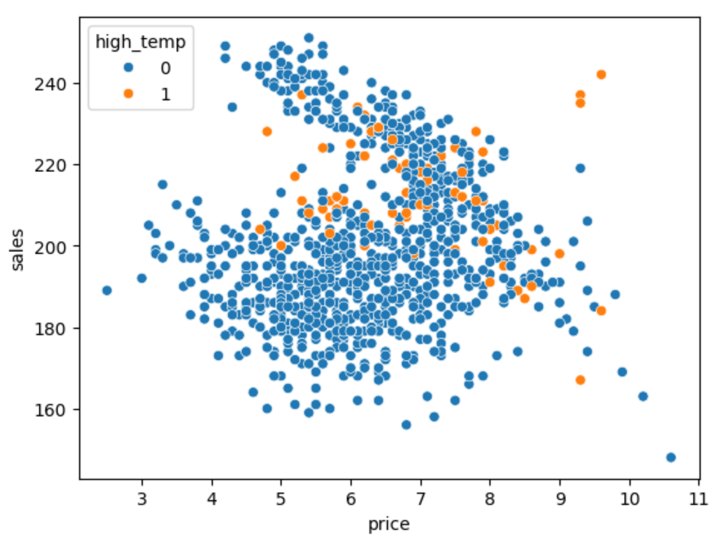
```python
from doubleml import DoubleMLData, DoubleMLPLR
from lightgbm import LGBMRegressor
from sklearn.ensemble import RandomForestRegressor

# Prepare the data for DoubleML
data = DoubleMLData(icecream, y_col='sales', d_cols='price',
                    x_cols=['temp', 'weekday', 'cost'])

# Specify the machine learning methods for nuisance parameters
learner = LGBMRegressor(objective='regression', max_depth=5, n_estimators=100,
                        n_jobs=-1, random_state=42, verbose=-1)
# Alternatively, you can use RandomForestRegressor
# learner = RandomForestRegressor(max_depth=5, n_estimators=100,
#                                 random_state=42, n_jobs=-1, verbose=0)

# Initialize the DoubleMLPLR model
## Set the random state globally to assure reproductibility,
## because DoubleMLPLR cannot set it within the setup or fit step.
np.random.seed(42)
random.seed(42)
dml_plr = DoubleMLPLR(data, ml_m=learner, ml_l=learner, n_folds=5)

# Fit the model
dml_plr.fit()

# Print the results
print(dml_plr.summary)
```

```
           coef   std err          t           P>|t|     2.5 %    97.5 %
price -3.898185  0.103646 -37.610414  1.452650e-309 -4.101328 -3.695042
```


Why the difference in the estimates?

Price is Higher in Summer—So is Demand

- Price sensitivity (elasticity) is lower as measured in abs value.



Connection to Other Methods

Close Ties to Other Models

- Should consider use several as robustness checks, when possible
- Similar to IV—Orthogonalization
  - IV—Isolate the effect uncorrelated with confounders
  - DML—Isolate the effect uncorrelated with observables
    - DML still need to be carefully about unobserved confounders
    - Need large covariates/features space
- Complementary to matching—mitigate poor matching

Limitations of DML

Not a magic wand

- Relies on observables to control confounders
  - Assumes no unobserved confounding
  - Cannot address unobserved confounders
- More complex and computational intensive
  - Requires multiple ML models and cross-fitting
  - Harder to interpret and debug