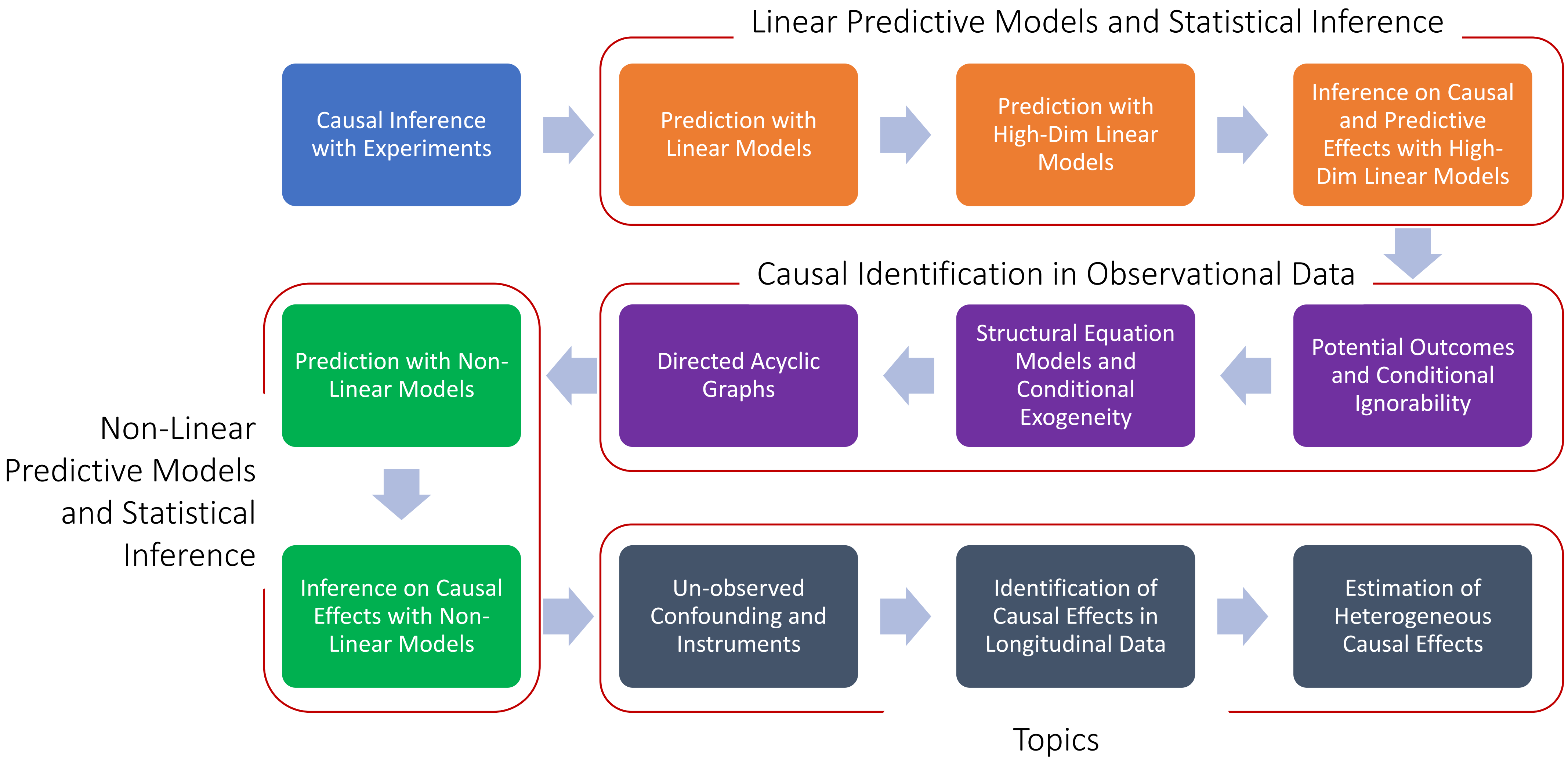
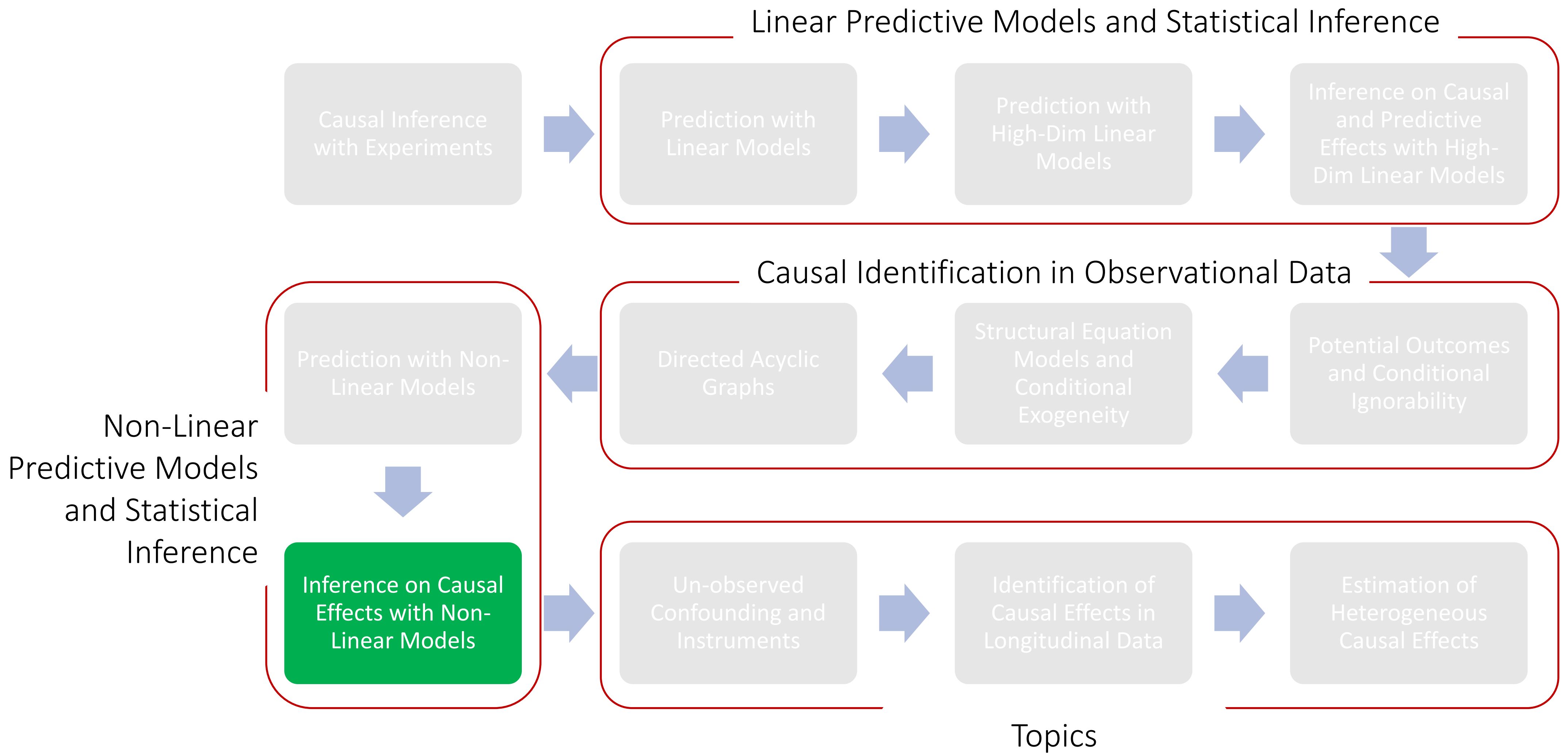
MS&E 228: Inference with Modern Non-Linear Prediction

Vasilis Syrgkanis

MS&E, Stanford


Linear Predictive Models and Statistical Inference
Causal Inference with Experiments
Prediction with Linear Models
Prediction with High-Dim Linear Models
Inference on Causal and Predictive Effects with High-Dim Linear Models
Causal Identification in Observational Data
Potential Outcomes and Conditional Ignorability
Structural Equation Models and Conditional Exogeneity
Directed Acyclic Graphs
Non-Linear Predictive Models and Statistical Inference
Prediction with Non-Linear Models
Inference on Causal Effects with Non-Linear Models
Un-observed Confounding and Instruments
Identification of Causal Effects in Longitudinal Data
Estimation of Heterogeneous Causal Effects
Topics

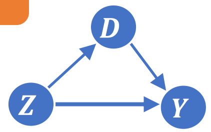

Linear Predictive Models and Statistical Inference
Causal Inference with Experiments
Prediction with Linear Models
Prediction with High-Dim Linear Models
Inference on Causal and Predictive Effects with High-Dim Linear Models
Causal Identification in Observational Data
Potential Outcomes and Conditional Ignorability
Structural Equation Models and Conditional Exogeneity
Directed Acyclic Graphs
Prediction with Non-Linear Models
Non-Linear Predictive Models and Statistical Inference
Inference on Causal Effects with Non-Linear Models
Un-observed Confounding and Instruments
Identification of Causal Effects in Longitudinal Data
Estimation of Heterogeneous Causal Effects
Topics

Recap of Last Lecture

Causal Inference Pipeline

Theory

DAGs/ASEMs



a_d
$\parallel$
$E\big[E[Y|D = d, Z]\big]$

Construct estimate $\hat{g}(d, z)$ of predictive model
$E[\,Y \mid D = d, Z = z\,]$
Estimate mean counterfactual
$\hat{a}_d = E_n[\hat{g}(d, Z)]$

$a_d \in \hat{a}_d \pm 2\hat{\sigma}$

Practice

Data Collection → Domain Assumption Elicitation → Identification → Estimation (training) → Validation (testing) → Inference (Confidence Intervals) → Sensitivity Analysis → Decision Policy → Adaptive Experiments → Data Collection

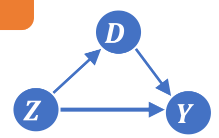
So I need to estimate this **bigger predictive model, predicting the outcome from the treatment $\boldsymbol{D}$ and the controls $\boldsymbol{S}$,** in some manner. Then average the predictive values over S fixing $d = 1$.

$a_d \in \hat{a}_d \pm (2\hat{\sigma} + \epsilon)$

Causal Inference Pipeline

Theory

DAGs/ASEMs



$$a_d = E\big[E[Y|D=d,Z]\big]$$

Construct estimate $\hat{g}(d,z)$ of predictive model

$$E[\,Y \mid D = d, Z = z\,]$$

Estimate mean counterfactual

$$\hat{a}_d = E_n[\hat{g}(d,Z)]$$

$$a_d \in \hat{a}_d \pm 2\hat{\sigma}$$

Practice

Data Collection → Domain Assumption Elicitation → Identification → Estimation (training) → Validation (testing) → Inference (Confidence Intervals) → Sensitivity Analysis → Decision Policy → Adaptive Experiments → Data Collection

There is no way to validate my causal estimate $\hat{\alpha}_d$, given the data that I have! I cannot observe counterfactuals. The best alternative is to **quantify uncertainty of my estimate** to see how confident I should be in the result.

$$a_d \in \hat{a}_d \pm (2\hat{\sigma} + \epsilon)$$

Goals for Today

- Methods for Confidence Intervals for ATE with non-linear models
- General Neyman Orthogonality Framework (Double/Debiased ML)
- Methods for Confidence Intervals for ATE in a partially-linear model
- Sample-splitting and cross-fitting

- Proof sketch of main theorem*

The Example Problem

Identification under Conditional Ignorability

- Once we condition on enough variables X that affect treatment assignment, remnant variation in D is exogenous (as-if trial)

$$Y^{(d)} \perp\!\!\!\perp D \mid X \quad \text{(conditional ignorability)}$$

- Why useful:

$$\begin{aligned} E[\,Y \mid D = d, X\,] &= E\left[\,Y^{(D)} \mid D = d, X\,\right] \\ &= E\left[\,Y^{(d)} \mid D = d, X\,\right] = E\left[\,Y^{(d)} \mid X\,\right] \end{aligned}$$

- Average treatment effect is "identified" as (g-formula):

$$\begin{aligned} \theta_0 = E\left[Y^{(1)} - Y^{(0)}\right] &= E\left[E\left[\,Y^{(1)} - Y^{(0)} \mid X\,\right]\right] \\ &= E\left[E[Y|D = 1, X] - E[\,Y \mid D = 0, X\,]\right] \end{aligned}$$

Let’s take it to data

- We observe n samples $Z_1, \ldots, Z_n$ where $Z_i = (X_i, D_i, Y_i)$

- Want to estimate average effect θ_0, which satisfies:

$$\theta_0 = E[g_0(1, X) - g_0(0, X)]$$

- Where:

$$g_0(D, X) \coloneqq E[\,Y \mid D, X\,]$$

- We want to be able to use ML to learn regression function g_0!

What do we want from $\hat{\theta}$?

- Ideally parametric rates for θ_0 even when we have slower rates for g_0
- Ideally construction of confidence intervals for θ_0

- One approach. Asymptotic normality

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \to_d N\left(0, \sigma^2\right)$$

- Implies construction of approximately correct confidence intervals

with prob. ≈ 95%: $\theta_0 \in \left[\hat{\theta} \pm 1.96\hat{\sigma}/\sqrt{n}\right]$

Natural Estimation Algorithm

- Estimate $\hat{g}$ of g_0 from data
- Calculate empirical plug-in average:

$$\hat{\theta} \coloneqq \frac{1}{n}\sum_{i=1}^{n} \hat{g}(1, X) - \hat{g}(0, X)$$

Natural Algorithm Gone Wrong

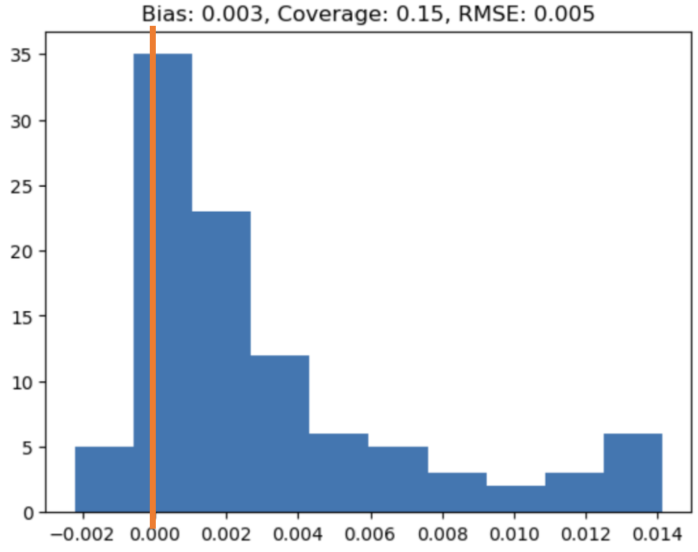
```
def est(X, D, y): # direct non-orthogonal estimator of average effect
    est = RandomForestRegressor(min_samples_leaf=20)
    est.fit(np.hstack([D.reshape(-1, 1), X]), y)
    ones = np.hstack([np.ones((X.shape[0], 1)), X])
    zeros = np.hstack([np.zeros((X.shape[0], 1)), X])
    preds = est.predict(ones) - est.predict(zeros)
    return np.mean(preds), np.std(preds)/np.sqrt(X.shape[0])
```

Simple Example

$X \sim N(0, I_{20})$

$D \sim \text{Binomial}\big(0.5 + \text{clip}(X_0, -0.4, 0.4)\big)$

$y \sim \theta_0 D + X_0 + X_1 + N(0,1)$



Natural Estimation Algorithm (Draft 2)

- Split the data in half S_1, S_2
- On first half S_1, estimate $\hat{g}$ of g_0
- Calculate empirical plug-in average on second half S_2:

$$\hat{\theta} \coloneqq \frac{1}{|S_2|} \sum_{i \in S_2} \hat{g}(1, X) - \hat{g}(0, X)$$

Natural Estimation Algorithm (Draft 3)

- Split data in K parts, $S_1, \dots, S_K$
- For each part k, estimate $\hat{g}_k$ using data from all parts except S_k
- Calculate average over all data:

$$\hat{\theta} = \frac{1}{n} \sum_{k=1}^{K} \sum_{i \in S_k} \hat{g}_k(1, X) - \hat{g}_k(0, X)$$

Natural Algorithm (Draft 3) Gone Wrong

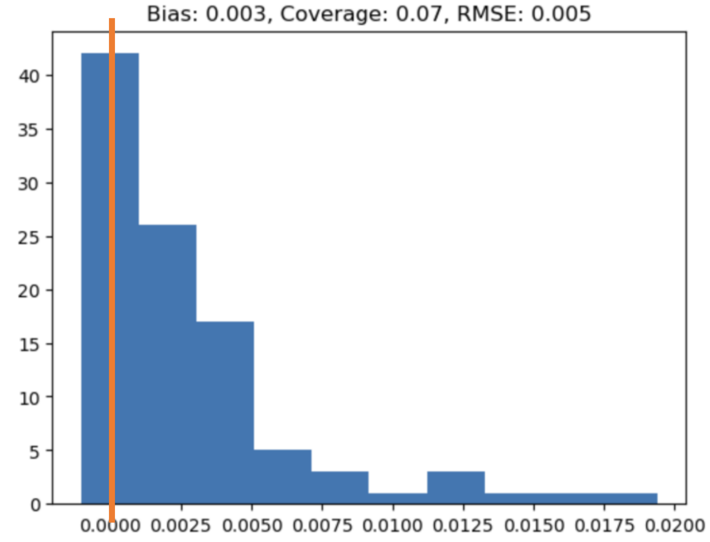
```
def est2(X, D, y): # direct non-orthogonal estimator with sample splitting
    effects = np.zeros(X.shape[0])
    for train, test in KFold(n_splits=3).split(X):
        est = RandomForestRegressor(min_samples_leaf=20)
        est.fit(np.hstack([D[train].reshape(-1, 1), X[train]]), y[train])
        ones = np.hstack([np.ones((X[test].shape[0], 1)), X[test]])
        zeros = np.hstack([np.zeros((X[test].shape[0], 1)), X[test]])
        effects[test] = est.predict(ones) - est.predict(zeros)
    return np.mean(effects), np.std(effects)/np.sqrt(X.shape[0])
```

Simple Example

$X \sim N(0, I_{20})$

$D \sim \text{Binomial}\big(0.5 + \text{clip}(X_0, -0.4, 0.4)\big)$

$y \sim \theta_0 D + X_0 + X_1 + N(0,1)$



When is estimate $\hat{\theta}$ $\sqrt{n}$-asymptotically normal?

When is estimate $\hat{\theta}$ $\sqrt{n}$-asymptotically normal?
We need to change the moment we use

Debiased Machine Learning

Average Causal Effect Example

- We observe n samples $Z_1, \ldots, Z_n$ where $Z_i = (X_i, D_i, Y_i)$
- Want to estimate average effect θ_0, which satisfies:

$$\theta_0 \coloneqq E[g_0(1, X) - g_0(0, X)]$$

- Where:

$$g_0(D, X) \coloneqq E[\, Y \mid D, X \,]$$

- The identification formula for θ_0 is sensitive to variations in g
- Any bias or error in g propagates to bias or error in moment and $\hat{\theta}$
- Can we add a correction that corrects the biases of $\hat{g}$

Better Formula for ATE

Regression residual is a proxy that g is biased

- Key Idea. Add a debiasing correction

$$M(g,a) = E[g(1,X) - g(0,X)] + E\left[a(D,X)\,\boxed{\left(Y\ - g(D,X)\right)}\right]$$

- What is a_0?
- Insensitivity: Take derivative with respect to g at θ_0, g_0, a_0 in any direction $\nu \in G$

$$\frac{\partial}{\partial t} M(g_0 + t\,\nu, a_0)\Big|_{t=0} = E[\nu(1,X) -\ \nu(0,X)] - E[a(D,X)\,\nu(D,X)] = 0$$

- If this holds then if g is very wrong but a is correct:

$$\theta = E[a_0(D,X)Y] = E\left[a_0(D,X)E[\,Y \mid D,X\,]\right]$$
$$= E[a_0(D,X)g(D,X)] = E[g(1,X) - g(0,X)]$$

Inverse Propensity Weighting (IPW)

- The following works: inverse propensity scoring

$$a_0(D, X) = \frac{D}{\Pr[D = 1|X]} - \frac{1 - D}{\Pr[D = 0|X]}$$

- Sketch:

$$\begin{aligned} E\left[\frac{D}{\Pr[D = 1|X]} g(D, X)\right] &= E\left[\frac{D}{\Pr[D = 1|X]} g(1, X)\right] \\ &= E\left[\frac{E[D|X]}{\Pr[D = 1|X]} g(1, X)\right] \\ &= E[g(1, X)] \end{aligned}$$

New Formula is Insensitive

$$M(g,a) = E[g(1,X) - g(0,X)] + E\left[a(D,X)\left(Y - g(D,X)\right)\right]$$

- Take derivative with respect to g at g_0, a_0 in any direction $\nu \in G$

$$\frac{\partial}{\partial t} M(g_0 + t\,\nu, a_0)\Big|_{t=0} = E[\nu(1,X) - \nu(0,X)] - E[a(D,X)\,\nu(D,X)] = 0$$

Take derivative with respect to a at g_0, a_0 in any direction $\nu \in A$

$$\frac{\partial}{\partial t} M(g_0, a_0 + t\nu)\Big|_{t=0} = E\left[\nu(D,X)\left(Y - g_0(D,X)\right)\right] = 0$$

Asymptotic Normality of De-biased Estimate

$$\hat{\theta} \coloneqq E_n\left[\hat{g}(1,X) - \hat{g}(0,X) + \hat{a}(D,X) \cdot \left(Y \ - \ \hat{g}(D,X)\right)\right]$$

- *Assume* that propensities are bounded away from 0 and 1 (strict overlap)
- *Assume* $\hat{g}, \hat{a}$ estimated on separate sample (or cross-fitting), are consistent and:

$$\sqrt{n}\, E\left[\left(a_0(D,X) - \hat{a}(D,X)\right)\left(\hat{g}(D,X) - g_0(D,X)\right)\right] \to_p 0$$

- *Assume* random variables $Y, a(D,X), g(D,X)$ have bounded fourth moments
- Then:

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \to_d N\left(0, \sigma^2\right), \qquad \hat{\sigma}^2 = Var_n\left(\hat{g}(1,X) - \hat{g}(0,X) + \hat{a}(X) \cdot \left(Y - \hat{g}(X)\right)\right)$$

Python Pseudocode

```
cv = KFold(n_splits=nfolds, shuffle=True, random_state=123)
yhat0, yhat1 = np.zeros(y.shape), np.zeros(y.shape)
# we will fit a model E[Y| D, X] by fitting a separate model for D==0
# and a separate model for D==1.
for train, test in cv.split(X, y):
    # train a model on training data that received zero and predict on all test data
    yhat0[test] = modely.fit(X[train][D[train]==0], y[train][D[train]==0]).predict(X[test])
    # train a model on training data that received one and predict on all test data
    yhat1[test] = modely.fit(X[train][D[train]==1], y[train][D[train]==1]).predict(X[test])
# prediction for observed treatment
yhat = yhat0 * (1 - D) + yhat1 * D
# propensity scores
Dhat = cross_val_predict(modeld, X, D, cv=cv, method='predict_proba', n_jobs=-1)[:, 1]
Dhat = np.clip(Dhat, trimming, 1 - trimming)
# doubly robust quantity for every sample
drhat = yhat1 - yhat0 + (y - yhat) * (D/Dhat - (1 - D)/(1 - Dhat))
point = np.mean(drhat)
var = np.var(drhat)
stderr = np.sqrt(var / X.shape[0])
return point, stderr, yhat, Dhat, y - yhat, D - Dhat, drhat
```

Continuous Treatments under Partial Linearity

Partially Linear Model

- Relevant in many applications: dose-response curve in healthcare, effect of price on demand, return-on-investment
- Assume conditional exogeneity
$$Y^{(d)} \perp\!\!\!\perp D \mid X$$
- Assume partially linear response
$$Y = \theta_0 D + f_0(X) + \epsilon, \qquad E[\,\epsilon \mid D, X\,] = 0$$
- Equivalently, a partial linearity condition on the conditional expectation function
$$g_0(D, X) = E[\,Y \mid D, X\,] = \theta_0 D + f_0(X)$$
- Parameter of interest θ_0 is constant marginal effect of treatment

Generalization of FWL Theorem

- Let's define a slight variant of residualization

$$\tilde{V} = V - E[V|X]$$

- Generalization of FWL theorem to partially linear models

$$\tilde{Y} = \theta_0 \tilde{D} + \epsilon, \qquad E\left[\epsilon \middle| \tilde{D}\right] = 0$$

- Let's consider the residual outcome

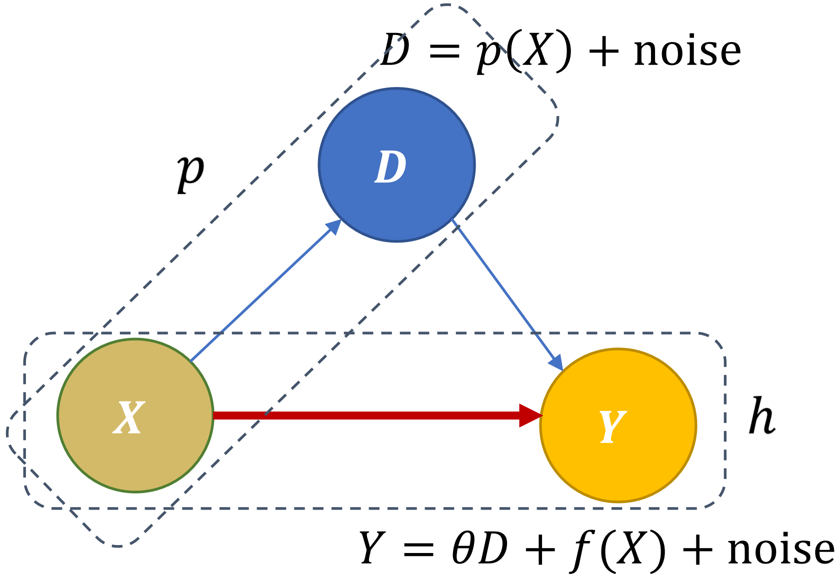
Regression model $h_0(X)$ predicting the outcome from the controls

$$\begin{aligned}\tilde{Y} &= Y - \boxed{E[Y|X]} \\ &= \theta_0 D + f_0(X) + \epsilon - E[\theta_0 D + f_0(X) + \epsilon|X] \\ &= \theta_0 D + f_0(X) + \epsilon - \theta_0 E[D|X] - f_0(X) \\ &= \theta_0(D - \boxed{E[D|X]}) + \epsilon\end{aligned}$$

Regression model $p_0(X)$ predicting the treatment from the controls

Orthogonal Method: Double ML

- **Double ML.** Split samples in half
  - Regress $Y \sim X$ with ML on first half, to get estimate $\hat{h}(S)$ of $E[Y|X]$
  - Regress $D \sim X$ with ML on first half, to get estimate $\hat{p}(S)$ of $E[D|X]$



Orthogonal Method: Double ML

- **Double ML.** Split samples in half
  - Regress $Y \sim X$ with ML on first half, to get estimate $\hat{h}(S)$ of $E[Y|X]$
  - Regress $D \sim X$ with ML on first half, to get estimate $\hat{p}(S)$ of $E[D|X]$
  - Construct residuals on other half, $\hat{D} := D - \hat{p}(X)$ and $\hat{Y} := Y - \hat{h}(X)$
  - Run OLS on residuals: $\hat{Y} \sim \hat{D}$ to get $\hat{\theta}$

- Final OLS, *in population limit*, equivalent to solving normal equation:

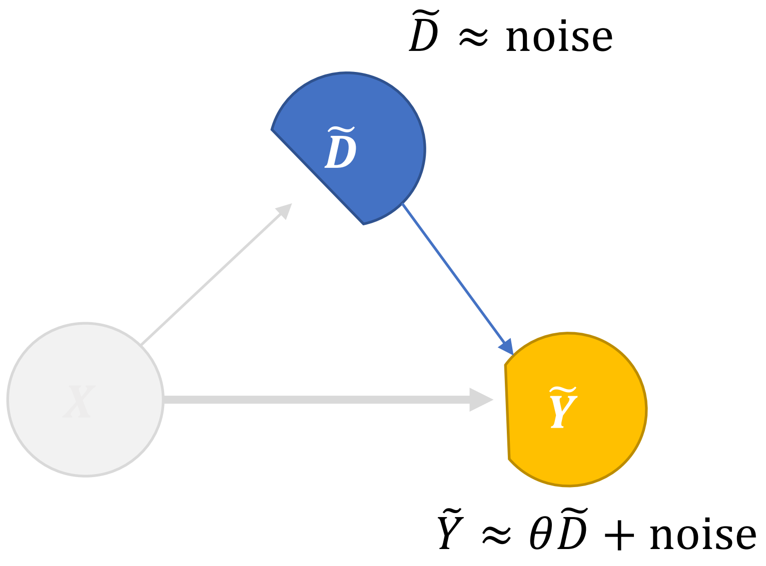
$$E\left[\left(\tilde{Y} - \theta\tilde{D}\right)\tilde{D}\right] = 0$$

- Define the *formula*:

$$M(\theta;\ h, p) = E\left[\left(Y - h(X) - \theta\left(D - p(X)\right)\right)\ \left(D - p(X)\right)\right]$$

- Final OLS, *in population limit*, equivalent to solving for θ:

$$M(\theta; h_0, p_0) = 0$$



Insensitivity of Double ML Method

- The *formula* M is insensitive to the nuisance functions h, p

$$M(\theta, h, p) = E\left[\Big(Y - h(X) - \theta\,\big(D - p(X)\big)\Big)\ \big(D - p(X)\big)\right]$$

- Directional derivative with respect to h

$$\frac{\partial}{\partial t} M(\theta_0; h_0, p_0 + t\,\nu)\bigg|_{t=0} = -E\big[\nu(X)\big(D - p_0(X)\big)\big] = E\big[\nu(X)E[\,D - p_0(X) \mid X\,]\big] = 0$$

- Directional derivative with respect to p

$$\frac{\partial}{\partial t} M(\theta_0;\ h_0, p_0 + t\,\nu)\bigg|_{t=0} = E\big[\nu(X)\big(D - p_0(X)\big)\big] - E\left[\Big(Y - h_0(X) - \theta_0\big(D - p_0(X)\big)\Big)\,\nu(X)\right] = 0$$

Asymptotic Normality of DoubleML Estimate

$$E_n\left[\left(\hat{Y} - \theta \widehat{D}\right)\widehat{D}\right] = 0 \Leftrightarrow \hat{\theta} = \frac{E_n\left[\hat{Y}\widehat{D}\right]}{E_n\left[\widehat{D}^2\right]}$$

- *Assume* that $E\left[\widetilde{D}^2\right] = E[Var(D \mid X)] > 0$ (average overlap)
- *Assume* $\hat{h}, \hat{p}$ estimated on separate sample (or cross-fitting), are consistent and:

$$\sqrt{n}\left(\mathrm{RMSE}\left(\hat{h}\right) \cdot \mathrm{RMSE}(\hat{p}) + \mathrm{RMSE}(\hat{p})^2\right) \to_p 0$$

- *Assume* random variables Y, D, X have bounded fourth moments

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \to_d N\left(0, \sigma^2\right),$$

Estimate asymptotically normal

$$\sigma^2 := \frac{E\left[\left(\tilde{Y} - \theta_0 \widetilde{D}\right)^2 \widetilde{D}^2\right]}{E\left[\widetilde{D}^2\right]^2},$$

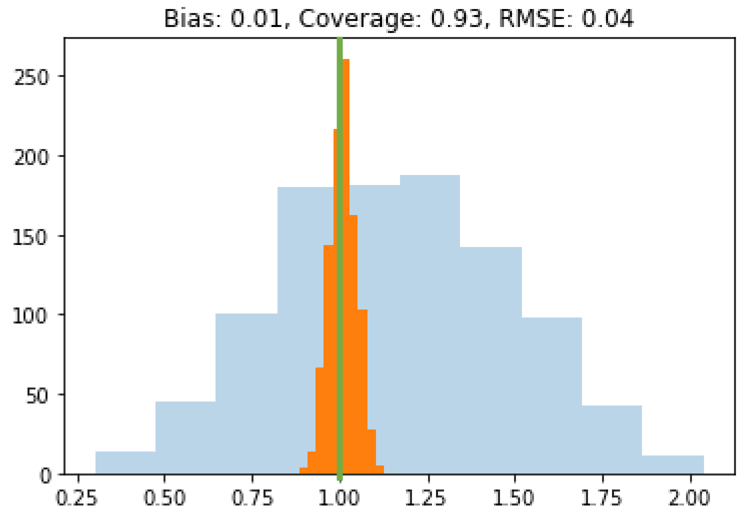
Asymptotic variance

$$\hat{\sigma}^2 = \frac{E_n\left[\left(\hat{Y} - \hat{\theta}\,\widehat{D}\right)^2 \widehat{D}^2\right]}{E_n\left[\widehat{D}^2\right]^2}$$

Estimate of variance $\Rightarrow$ 95% CI $\left[\theta \pm 1.96 \frac{\hat{\sigma}}{\sqrt{n}}\right]$

Natural Algorithm (Draft 3) Gone Right

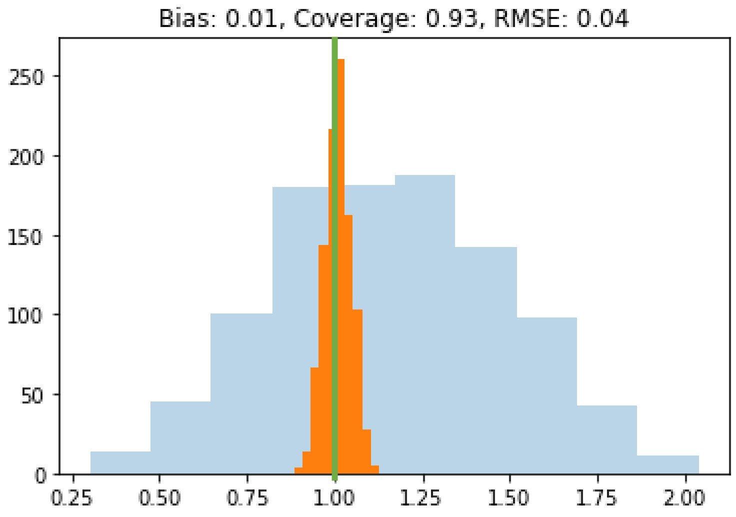
```
def dml2(X, D, y): # orthogonal dml with sample-splitting
    est_y = RandomForestRegressor(min_samples_leaf=20)
    yres = y - cross_val_predict(est_y, X, y, cv=3)
    est_t = RandomForestRegressor(min_samples_leaf=20)
    Dres = D - cross_val_predict(est_t, X, D, cv=3)
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```



Natural Algorithm (Draft 3) Gone Right

```
from econml.dml import LinearDML

dml = LinearDML(model_y=RandomForestRegressor(min_samples_leaf=20),
                model_t=RandomForestRegressor(min_samples_leaf=20))
est.fit(y, D, W=X).effect_inference()
```



Proof sketch

$$\sqrt{n}(\hat{\theta} - \theta_0) = \sqrt{n}\left(\frac{E_n[\hat{Y}\widehat{D}]}{E_n[\widehat{D}^2]} - \theta_0\right)$$

$$\approx \sqrt{n}\left(\frac{E_n[\tilde{Y}\widetilde{D}]}{E_n[\widetilde{D}^2]} - \theta_0\right)$$

Due to insensitivity of the formula and fast enough rates of the nuisance models, we can ignore the error in the residuals

$$= \sqrt{n}\left(\frac{E_n[\tilde{Y}\widetilde{D}]}{E_n[\widetilde{D}^2]} - \frac{E_n[\widetilde{D}^2]}{E_n[\widetilde{D}^2]}\theta_0\right)$$

$$= \sqrt{n}\left(\frac{E_n[(\tilde{Y} - \theta_0\widetilde{D})\widetilde{D}]}{E_n[\widetilde{D}^2]}\right)$$

$$\approx \sqrt{n}\left(\frac{E_n[(\tilde{Y} - \theta_0\widetilde{D})\widetilde{D}]}{E[\widetilde{D}^2]}\right) = \sqrt{n}E_n\left[\frac{(\tilde{Y} - \theta_0\widetilde{D})\widetilde{D}}{E[\widetilde{D}^2]}\right] \to_d N\left(0, Var\left(\frac{(\tilde{Y} - \theta_0\widetilde{D})\widetilde{D}}{E[\widetilde{D}^2]}\right)\right)$$

By law of large numbers, the denominator can be replaced by the expectation, $E_n[\widetilde{D}^2] \approx E[\widetilde{D}^2]$

Central Limit Theorem

(Extended) Partially Linear Model

- Relevant in many applications: dose-response curve in healthcare, effect of price on demand, return-on-investment
- Assume conditional exogeneity

$$Y^{(d)} \perp\!\!\!\perp D \mid X$$

- Assume partially linear response, for a known feature map ϕ

$$Y = \theta_0{}'\phi(D, X) + f_0(X) + \epsilon, \qquad E[\,\epsilon \mid D, X\,] = 0$$

- Parameter of interest θ_0
- **Example:** in pricing applications $\phi(D, X) = \left(D, D^2, X \cdot D, X \cdot D^2\right)$

Orthogonal Method: (Extended) Double ML

- **Double ML.** Split samples in half
  - Regress $Y \sim X$ with ML on first half, to get estimate $\hat{h}(S)$ of $E[Y|X]$
  - Regress $\phi(D, X) \sim X$ with ML on first half, to get estimate $\hat{p}(S)$ of $E[\phi(D, X)|X]$
  - Construct residuals on other half, $\widetilde{D} \coloneqq \phi(D, X) - \hat{p}(X)$ and $\tilde{Y} \coloneqq Y - \hat{h}(X)$
  - Run OLS on residuals: $\tilde{Y} \sim \widetilde{D}$ to get $\hat{\theta}$

- Final OLS, in population limit, equivalent to solving normal equation:

$$E\left[\left(\tilde{Y} - \theta' \widetilde{D}\right)\widetilde{D}\right] = 0$$

- Define the *formula*:

$$M(\theta, h, p) = E\left[\left(Y - h(X) - \theta'\left(\phi(D, X) - p(X)\right)\right)\ \left(\phi(D, X) - p(X)\right)\right]$$

- OLS is equivalent to solving with respect to θ:

$$M(\theta, h_0, p_0) = 0$$

Asymptotic Normality of DoubleML Estimate

$$E_n\left[\left(\hat{Y} - \theta\widehat{D}\right)\widehat{D}\right] = 0 \Leftrightarrow \hat{\theta} = E_n\left[\widehat{D}\widehat{D}'\right]^{-1} E_n\left[\widehat{D}\hat{Y}\right]$$

- *Assume* that $E\left[\widetilde{D}\widetilde{D}'\right] \succcurlyeq 0$ (minimum eigenvalue bounded away)
- *Assume* $\hat{h}, \hat{p}$ estimated on separate sample (or cross-fitting), are consistent and:

$$\sqrt{n}\left(\mathrm{RMSE}\left(\hat{h}\right) \cdot \mathrm{RMSE}(\hat{p}) + \mathrm{RMSE}(\hat{p})^2\right) \rightarrow_p 0$$

- *Assume* random variables Y, D, X have bounded fourth moments

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \rightarrow_d N(0, V), \qquad \hat{V} \coloneqq E_n\left[\widehat{D}\widehat{D}'\right]^{-1} E_n\left[\left(\hat{Y} - \hat{\theta}\widehat{D}\right)^2 \widehat{D}\widehat{D}'\right] E_n\left[\widehat{D}\widehat{D}'\right]^{-1}$$

Sandwich formula for heteroskedasticity-robust standard errors in OLS packages (HC0)

Practical Variants of Cross-Fitting

Stacking and Model Selection

- If we want to choose among many models or perform stacking, we can just use a stacked or automl model in place of each ML model

Stacking ML Models

```
def dml2(X, D, y): # orthogonal dml with sample-splitting
    est_y = StackingRegressor([rf, nnet, gbf, lasso])
    yres = y - cross_val_predict(est_y, X, y, cv=3)
    est_t = StackingRegressor([rf, nnet, gbf, lasso])
    Dres = D - cross_val_predict(est_t, X, D, cv=3)
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

AutoML Models

```
from flaml import AutoML

def dml2(X, D, y): # orthogonal dml with sample-splitting
    est_y = AutoML()
    yres = y - cross_val_predict(est_y, X, y, cv=3)
    est_t = AutoML()
    Dres = D - cross_val_predict(est_t, X, D, cv=3)
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

Stacking and Model Selection

- If we want to choose among many models or perform stacking, we can just use a stacked or automl model in place of each ML model
- Model selection or stacking done many times within each training fold
- Computationally expensive and statistically lossy

- Can we use all the data to at least select among models?

Semi-Crossfitting Estimation Algorithm

Split the data in half *(in practice K folds)*

- On first half, estimate $\hat{g}_1^{(1)}, \dots, \hat{g}_1^{(\mathrm{L})}$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2^{(1)}, \dots, \hat{g}_2^{(\mathrm{L})}$ of g_0 and predict on first half

- Choose the model $\ell \in \{1, \dots, L\}$ that optimizes out-of-sample RMSE

- On all data, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) \coloneqq \frac{1}{n} \sum_{i \in S_1} m\left(Z_i; \hat{\theta}, \hat{g}_2^{(\ell)}\right) + \frac{1}{n} \sum_{i \in S_2} m\left(Z_i; \hat{\theta}, \hat{g}_1^{(\ell)}\right) = 0$$

Semi-Crossfitting

```
def dml2(X, D, y): # orthogonal dml with semi-crossfitting
    # cross val predict with many models
    est_y = [rf, gbf, lasso]
    yres = np.array([y - cross_val_predict(est, X, y, cv=3) for est in est_y])
    est_d = [rf, gbf, lasso]
    Dres = np.array([D - cross_val_predict(est, X, D, cv=3) for est in est_d])
    # select models with best out of fold performance
    best_y = np.argmin(np.mean(yres**2, axis=1))
    best_d = np.argmin(np.mean(Dres**2, axis=1))
    yres = yres[best_y]
    Dres = Dres[best_d]
    # go with their corresponding residuals
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

Semi-Crossfitting

- If the number of models L is small, then "spillover" is ok and approach still works. For practical purposes L should be thought as constant.
- Under further regularity, provably asymptotic normality holds if

$$\sqrt{\log(L)} = \mathrm{o}\left(\mathrm{n}^{1/4}\right)$$

Semi-Crossfitting with Stacking

Split the data in half *(in practice K folds)*

- On first half, estimate $\hat{g}_1^{(1)}, \dots, \hat{g}_1^{(\mathrm{L})}$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2^{(1)}, \dots, \hat{g}_2^{(\mathrm{L})}$ of g_0 and predict on first half

- Construct weights $\alpha = (\alpha_1, \dots, \alpha_L)$ on the models using all the data (stacking)
- Define stacked models $\hat{g}_k^{(\alpha)} = \sum_{j=1}^{L} a_j \cdot \hat{g}_k^{(j)}$

- On all data, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) \coloneqq \frac{1}{n} \sum_{i \in S_1} m\left(Z_i; \hat{\theta}, \hat{g}_2^{(\alpha)}\right) + \frac{1}{n} \sum_{i \in S_2} m\left(Z_i; \hat{\theta}, \hat{g}_1^{(\alpha)}\right) = 0$$

Semi-Crossfitting with Stacking

```
def dml2(X, D, y): # orthogonal dml with semi-crossfitting and stacking
    # cross val predict with many models
    est_y = [rf, gbf, lasso]
    ypreds = np.array([cross_val_predict(est, X, y, cv=3) for est in est_y]).T
    est_d = [rf, gbf, lasso]
    Dpreds = np.array([cross_val_predict(est, X, D, cv=3) for est in est_d]).T
    # calculate stacked residuals by finding optimal coefficients
    # and weigthing out-of-sample predictions by these coefficients
    yres = y - LinearRegression().fit(ypreds, y).predict(ypreds)
    Dres = D - LinearRegression().fit(Dpreds, D).predict(Dpreds)
    # go with the stacked residuals
    theta = np.mean(yres * Dres) / np.mean(Dres**2)
    var = np.mean((Dres**2) * (yres - theta*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return theta, stderr
```

Semi-Crossfitting

- If the number of models L is small, then “spillover” is ok and approach still works. For practical purposes L should be thought as constant.
- Under further regularity, provably asymptotic normality holds if

$$\sqrt{L} = \mathrm{o}\left(\mathrm{n}^{1/4}\right)$$

Equivalent view of cross-fitting with stacking (lens of FWL theorem)

- Construct out of fold predictions based on many ML models
- Use these predictions as engineered features X in a simple OLS regression on D, X
- Use the coefficient and standard error of D from this final OLS

General Theory

Estimation from Moment Restrictions

- Observe samples $Z_1, \dots, Z_n$ i.i.d. from data distribution D
- Parameter of interest $\theta_0 \in \mathbb{R}^d$ is identified as the solution to a set of *population formulas*

$$M(\theta_0; g_0) \coloneqq E_{Z\sim D}[m(Z; \theta_0, g_0)] = 0$$

Vector of moment restrictions that distribution D needs to satisfy and which have a unique solution with respect to θ

- $g_0 \in G$ a function we don't care (*nuisance function*) but need to estimate from data

Examples.

$$m(Z; \theta, g, a) = g(1, X) - g(0, X) - a(D, X)\big(Y - g(D, X)\big) - \theta$$ *(ATE for binary treatment)*

$$m(Z; \theta, h, p) = \Big(Y - h(X) - \theta\big(D - p(X)\big)\Big) \cdot \big(D - p(X)\big)$$ *(ATE under PLR model)*

Cross-fitting Estimation Algorithm

Split the data in half

- On first half, estimate $\hat{g}_1$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2$ of g_0 and predict on first half

On all data, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) := \frac{1}{n} \sum_{i \in S_1} m(Z_i; \hat{\theta}, \hat{g}_2) + \frac{1}{n} \sum_{i \in S_2} m(Z_i; \hat{\theta}, \hat{g}_1) = 0$$

In practice use $K \approx 3 \text{ to } 5$ folds: for each fold, train on all other folds and predict on that fold

Neyman Orthogonality (Insensitivity)

Moment $M(\theta, g)$ is Neyman orthogonal if for any $\nu \in G - g_0$:

$$\frac{\partial}{\partial t} M(\theta_0, g_0 + t\,\nu)\Big|_{t=0} = 0$$

Main Theorem

If moment is Neyman orthogonal and RMSE of $\hat{g}$ is $o_p\left(n^{-1/4}\right)$, plus regularity conditions

$$\sqrt{n}\,\left(\hat{\theta} - \theta_0\right) \to N\left(0, J_0^{-1}\Sigma\left(\mathrm{J}_0^{-1}\right)^{\top}\right)$$

where $J_0 \coloneqq \nabla_\theta M(\theta_0, g_0)$ and $\Sigma \coloneqq E\left[m(Z;\theta_0, g_0)\, m(Z;\theta_0, g_0)^{\top}\right]$

Automatic Debiasing

- Quite generally, start with any formula that identifies your parameter
- You can turn it into an "insensitive formula" to nuisance functions
- Typically add a debiasing term multiplied by an appropriate α function

See e.g.

[2110.03031] RieszNet and ForestRiesz: Automatic Debiased Machine Learning with Neural Nets and Random Forests

[2307.04527] Automatic Debiased Machine Learning for Covariate Shifts

[2203.13887] Automatic Debiased Machine Learning for Dynamic Treatment Effects (arxiv.org)

Example:
Parameters Defined via Linear Functionals

- Suppose parameter of interest defined as:

$$\theta_0 = E[m(Z; g_0)], \qquad g_0(X) \coloneqq E[Y|X]$$

for some known moment m that is linear in g

- Examples
  - Average Treatment Effect: $g_0(D, X) = E[Y|D, X]$, $\theta_0 \coloneqq E[g_0(1, X) - g_0(0, X)]$
  - Average Policy Effect: $g_0(D, X) \coloneqq E[Y|D, X]$, $\theta_0 \coloneqq E\left[\pi(X) \cdot \left(g_0(1, X) - g_0(0, X)\right)\right]$
  - Average Derivative: $g_0(D, X) \coloneqq E[Y|D, X]$, $\theta_0 \coloneqq E[\partial_D g_0(D, X)]$

De-biased Moment

- Suppose parameter of interest defined as of the form:

$$\theta_0 = E[m(Z; g_0)], \qquad g_0(X) \coloneqq E[Y|X]$$

for some known moment m that is linear in g

- Then debiased version of the moment is of the form:

$$\theta_0 = E\left[m(Z; g_0) + a_0(X) \cdot \left(Y - g_0(X)\right)\right]$$

- $a_0(X)$: Riesz Representer (RR) of linear functional

$$\forall g: E[m(Z; g)] = E[a_0(X) \cdot g(X)]$$

Automatic Debiasing

Traditionally: characterize how RR looks like and perform plug-in estimation

Average Treatment Effect:

$$\theta := E[g(1,X) - g(0,X)], \qquad a(D,X) := \frac{D}{p(X)} - \frac{(1-D)}{1-p(X)}$$

Average Policy Effect:

$$\theta := E\big[\pi(X) \cdot \big(g(1,X) - g(0,X)\big)\big], \qquad a(D,X) := \pi(X)\left(\frac{D}{p(X)} - \frac{(1-D)}{1-p(X)}\right)$$

Average Derivative:

$$\theta := E[\partial_D g(D,X)], \qquad a(D,X) := -\partial_D \log\big(f(D|X)\big)$$

Can we circumvent the characterization step and estimate $\boldsymbol{a}$ directly? [Newey'94, Chen-Liao'14, Chernozhukov, Newey, Singh'18, Smucler, Rotnitzky, Robins, 19, Chernozhukov, Newey, S., Singh'19-21, Chernozhukov, Newey, Quintas-Martinez, S.'21]

Automatic Debiasing

[2104.14737] Automatic Debiased Machine Learning via Neural Nets for Generalized Linear Regression

- The RR is the minimizer of the loss

$$L(a) \coloneqq E\left[a(X)^2 - 2\, m(Z; a)\right]$$

- By RR property of a_0:

$$E[m(Z; a)] = E[a_0(X) \cdot a(X)]$$

- Loss is equivalent to an incomplete square loss

$$L(a) \coloneqq E\left[a(X)^2 - 2\, a_0(X) \cdot a(X)\right]$$

- Therefore,

$$L(a) - L(a_0) = E\left[\left(a(X) - a_0(X)\right)^2\right]$$

- Fast statistical learning rates based on modern statistical learning theory techniques can be derived based on this interpretation + practical ML algorithms (RieszNet, ForestRiesz)

Appendix:
General Theory (Expanded)

Main Theorem (expanded)

Define RMSE: $\|h\|_{L^2} = \sqrt{E[h(X)^2]}$

- If moment is Neyman orthogonal and RMSE of $\hat{g}$ goes down at rate $n^{1/4}$, plus regularity conditions

$$n^{1/4}\|\hat{g} - g_0\|_{L^2} \approx 0$$

- Then the estimate $\hat{\theta}$ is *asymptotically linear*

$$\sqrt{n}(\hat{\theta} - \theta_0) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -J_0^{-1}\, m(Z; \theta_0, g_0), \qquad J_0 := \partial_\theta E[m(Z; \theta_0, g_0)]$$

Jacobian of moments with respect to parameter; relates to identification strength

influence function

influence is a linear transformation of the moment; transforming from "moment space" to "parameter space"

- Consequently, it is *asymptotically normal*

$$\sqrt{n}\,(\hat{\theta} - \theta_0) \sim_a N(0, V), \qquad V := E[\phi_0(Z)\phi_0(Z)']$$

Covariance of the influence function

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{V}\ell}{n}}\right], \qquad \hat{V} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) := -\hat{J}^{-1} m(Z; \hat{\theta}, \hat{g}), \qquad \hat{J} = \partial_\theta E_n\left[m(Z; \hat{\theta}, \hat{g})\right]$$

Empirical variance of approximate influence

Approximate influence function

Empirical average of jacobian

Python Pseudocode

```
# General DML pseudocode
def general_dml(Z, ell, nfolds, moment, jacobian, nuisance
    # construct out-of-fold predictions from the nuisan
    ghat = cross_val_predict(nuisance_estimator, Z, cv=nfol
      use these predictions to define the empirical moment
      with respect to theta
      moment = lambda theta: np.mean(moment(Z, theta, ghat), axis=0)
    # solve for the empirical moment equation equals zero
    thetahat = fsolve(avg_moment)
    # calculate empirical jacobian with respect to theta, evaluated
    # at the estimates thetahat and ghat
    Jhat = np.mean(jacobian(Z, thetahat, ghat), axis=0)
    # construct approximate influence function for each sample
    phihat = - moment(Z, thetahat, ghat) @ np.linalg.pinv(Jhat).T
    # variance estimate
    var = phihat.T @ phihat / phihat.shape[0]
    # estimate and standard error for any projection ell of the pa
    point = ell @ thetahat
    stderr_ell = np.sqrt(ell.T @ var @ ell / phihat.shape[0])
    return point, stderr_ell
```

Estimate $\hat{g}$ out of fold and predict

Construct cross-fitted moment function
$\theta \to E_n[m(Z;\theta,\hat{g})]$

Solve that moment = 0 wrt θ

Calculate
$\hat{J} := \partial_\theta E_n\left[m(Z;\hat{\theta},\hat{g})\right]$

Approximate influence function
$Z \to \hat{\phi}(Z) := \hat{J}^{-1} m(Z;\hat{\theta},\hat{g})$

Empirical covariance of estimate
$\hat{V} = E_n\left[\hat{\phi}(Z)\hat{\phi}(Z)'\right]$

Main Theorem (linear moments)

- If moments are linear

$$m(Z;\theta,g) = \nu(Z;g) - a(Z;g)\theta$$

- Estimate is closed form:

$$\hat{\theta} = \hat{J}^{-1}E_n[\nu(Z;g)], \qquad \hat{J} = E_n[a(Z;g)]$$

- Then the estimate $\hat{\theta}$ is *asymptotically linear*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \approx \sqrt{n}\, E_n[\phi_0(Z)], \qquad \phi_0(Z) = -J_0^{-1}\, m(Z;\theta_0, g_0), \qquad J_0 \coloneqq E[a(Z;\ g_0)]$$

- Consequently, it is *asymptotically normal*

$$\sqrt{n}\left(\hat{\theta} - \theta_0\right) \sim_a N(0,V), \qquad V \coloneqq E[\phi_0(Z)\phi_0(Z)']$$

- *Confidence intervals* for any projection based on estimate of variance are asymptotically valid

$$\ell'\theta \in \left[\ell'\hat{\theta} \pm c\sqrt{\frac{\ell'\hat{V}\ell}{n}}\right], \qquad \hat{\mathrm{V}} = \mathrm{Var_n}\left(\hat{\phi}(Z)\right), \qquad \hat{\phi}(Z) \coloneqq -\hat{J}^{-1}m\left(Z;\hat{\theta},\hat{g}\right), \qquad \hat{J} = E_n[a(Z;\hat{g})]$$

Python Pseudocode

```
# General DML pseudocode for linear moments: m(Z; theta, g)
def general_dml_linear_moment(Z, ell, nfolds, alpha, nu, ja
    # construct out-of-fold predictions from the nuisance
    ghat = cross_val_predict(nuisance_estimator, Z, cv=nfolds)
    # use these predictions to define the empirical moment equation
    # and solve explicitly with respect to theta
    Jhat = np.mean(alpha(Z, ghat), axis=0)
    avg_nu = np.mean(nu(Z, ghat), axis=0)
    # solve for the empirical moment equation equals zero
    invJhat = np.linalg.pinv(Jhat)
    thetahat = invJhat @ avg_nu
    # construct approximate influence function for each sample
    phihat = - (nu(Z, ghat) - alpha(Z, ghat) @ thetahat) @ invJhat.T
    # variance estimate
    var = phihat.T @ phihat / phihat.shape[0]
    # estimate and standard error for any projection ell of the param
    point = ell @ thetahat
    stderr_ell = np.sqrt(ell.T @ var @ ell / phihat.shape[0])
    return point, stderr_ell
```

Estimate $\hat{g}$ out of fold and predict

cross-fitted jacobian $\hat{J} := E_n[a(Z;\hat{g})]$

cross-fitted offset $E_n[\nu(Z;\hat{g})]$

Closed form solution: $\theta = \hat{J}^{-1} E_n[\nu(Z;\hat{g})]$

Approximate influence function

$Z \to \hat{\phi}(Z) := \hat{J}^{-1} m(Z;\hat{\theta},\hat{g})$

Empirical covariance of estimate

$\hat{V} = E_n[\hat{\phi}(Z)\hat{\phi}(Z)']$

Proving the Main Theorem

Linear in θ Moments

- We will restrict attention to a broad class that simplifies proof

- Moment is linear in target parameter

$$m(Z;\theta,g) = a(Z;g)'\theta + \nu(Z;g)$$

- Expected moment also linear in θ

$$M(\theta,g) = A(g)'\theta + V(g)$$

Proof Ingredients: Linear in θ Moments

- Since $M_n(\hat{\theta}, \hat{g}) = 0$ we expect by concentration and sample splitting $M(\hat{\theta}, \hat{g}) \approx n^{-1/2}$
- Since $M(\theta_0, g_0) = 0$ we expect by Neyman orthogonality $M(\theta_0, \hat{g}) \approx RMSE(\hat{g})^2 = o(n^{-1/2})$
- Since moment is linear in θ: $A(\hat{g})\,(\hat{\theta} - \theta_0) = M(\hat{\theta}, \hat{g}) - M(\theta_0, \hat{g})$
- Since A is Lipschitz and $\hat{g} \to g_0$: $A(g_0)\,(\hat{\theta} - \theta_0) = M(\hat{\theta}, \hat{g}) - M(\theta_0, \hat{g}) + \mathrm{o_p}(\|\hat{\theta} - \theta_0\|)$
- Since $A(g_0)$ is invertible: $\|\hat{\theta} - \theta_0\| = \mathrm{O}(\|M(\hat{\theta}, \hat{g})\| + \|M(\theta_0, \hat{g})\|) = O_p(n^{-1/2})$

- More fine-grained analysis of $M(\hat{\theta}, \hat{g})$ term, shows: $\sqrt{n}M(\hat{\theta}, \hat{g}) \to \mathrm{N}(0, \mathrm{V})$

Proof of Main Theorem (visually)

CLT
+
sample-splitting
+
concentration
+
$\hat{g} \to g_0$

$M_n(\hat{\theta}, \hat{g}) = 0$

$N(0, V) \leftarrow$

$A(\hat{g})\,(\hat{\theta} - \theta_0) = M(\hat{\theta}, \hat{g}) - M(\theta_0, \hat{g})$

2[nd] order Taylor
+
orthogonality
+
$\|\hat{g} - g_0\| = o_p(n^{-1/4})$

$= o_p(n^{-1/2})$

$o_p(\|\hat{\theta} - \theta_0\|) =$

$M(\theta_0, g_0) = 0$

$A(g_0)\,(\hat{\theta} - \theta_0)$

Lipschitz $A(g)$
+
$\hat{g} \to g_0$

Proof of Main Theorem (algebraically)

- Since moment $M(\theta, g)$ is linear in θ and $M_n(\hat{\theta}, \hat{g}) = 0$ and $M(\theta_0, g_0) = 0$

$$\begin{aligned} A(\hat{g})\left(\hat{\theta} - \theta_0\right) &= M\left(\hat{\theta}, \hat{g}\right) - M(\theta_0, \hat{g}) \\ &= M\left(\hat{\theta}, \hat{g}\right) - M_n\left(\hat{\theta}, \hat{g}\right) + M(\theta_0, g_0) - M(\theta_0, \hat{g}) \end{aligned}$$

- Since $\mathrm{RMSE}(\hat{g}) = \|\hat{g} - g_0\| = o_p(1)$

$$A(g_0)\left(\hat{\theta} - \theta_0\right) = A(\hat{g})\left(\hat{\theta} - \theta_0\right) + \mathrm{o_p}\left(\left\|\hat{\theta} - \theta_0\right\|\right)$$

- Thus

$$A(g_0)\left(\hat{\theta} - \theta_0\right) = \underbrace{M\left(\hat{\theta}, \hat{g}\right) - M_n\left(\hat{\theta}, \hat{g}\right)}_{\substack{\to N(0, V) \\ \text{via CLT} \\ +\ \text{sample-splitting} \\ +\ \text{concentration} \\ +\ \hat{g} \to g_0}} + \underbrace{M(\theta_0, g_0) - M(\theta_0, \hat{g})}_{\substack{= o_p\left(n^{-1/2}\right) \\ \text{via orthogonality} \\ +\ \|\hat{g} - g_0\| = o_p\left(n^{-1/4}\right)}} + \mathrm{o_p}\left(\left\|\hat{\theta} - \theta_0\right\|\right)$$

Proof of Main Theorem: Orthogonality

- By Neyman orthogonality and bounded second derivative of $M(\theta_0, g)$ w.r.t. g
  $M(\theta_0, g_0) - M(\theta_0, \hat{g}) = D_g M(\theta_0, g_0)[g_0 - g] + O\left(\|\hat{g} - g_0\|^2\right) = o_p\left(n^{-1/2}\right)$

- Thus

$$A(g_0)\left(\hat{\theta} - \theta_0\right) = \underbrace{M\left(\hat{\theta}, \hat{g}\right) - M_n\left(\hat{\theta}, \hat{g}\right)}_{G_n\left(\hat{\theta}, \hat{g}\right)} + \mathrm{o_p}\left(n^{-1/2} + \left\|\hat{\theta} - \theta_0\right\|\right)$$

Proof of Main Theorem: Sample-Splitting (1)

- Let $G_n(\theta, g) = M(\theta, g) - M_n(\theta, g)$

$$G_n(\hat{\theta}, \hat{g}) = G_n(\theta_0, g_0) + G_n(\hat{\theta}, \hat{g}) - G_n(\theta_0, \hat{g}) + G_n(\theta_0, \hat{g}) - G_n(\theta_0, g_0)$$

- Linearity of moment + (sample-splitting and concentration $\Rightarrow$ $\|A(\hat{g}) - A_n(\hat{g})\| = o_p(1)$):

$$G_n(\hat{\theta}, \hat{g}) - G_n(\theta_0, \hat{g}) = \left(A(\hat{g}) - A_n(\hat{g})\right)\left(\hat{\theta} - \theta_0\right) = o_p\left(\|\hat{\theta} - \theta_0\|\right)$$

- Thus

$$A(g_0)\left(\hat{\theta} - \theta_0\right) = G_n(\theta_0, g_0) + G_n(\theta_0, \hat{g}) - G_n(\theta_0, g_0) + \mathrm{o_p}\left(n^{-1/2} + \|\hat{\theta} - \theta_0\|\right)$$

Proof of Main Theorem: Sample-Splitting (2)

- Note for $X_i = m(Z_i; \theta_0, \hat{g}) - m(Z_i; \theta_0, \hat{g}) - E[m(Z_i; \theta_0, \hat{g}) - m(Z_i; \theta_0, \hat{g})]$

$$G_n(\theta_0, \hat{g}) - G_n(\theta_0, g_0) = \frac{1}{n_2} \sum_{i \in S_2} X_i$$

- By sample splitting, X_i are i.i.d. with $E[X_i] = 0$. By variance decomposition (concentration)

$$\left\| \frac{1}{n_2} \sum_{i \in S_2} X_i \right\|_{L_2} \leq \sqrt{\frac{E\left[X_i^2\right]}{n}}$$

- Thus

$$\|G_n(\theta_0, \hat{g}) - G_n(\theta_0, g_0)\|_{L_2} \leq \frac{1}{\sqrt{n}} \sqrt{E\left[\left(m(Z_i; \theta_0, \hat{g}) - m(Z_i; \theta_0, \hat{g})\right)^2\right]} = O\left(\frac{\|\hat{g} - g_0\|}{\sqrt{n}}\right) = o_p\left(n^{-1/2}\right)$$

Concluding

- So far

$$A(g_0)\left(\hat{\theta}-\theta_0\right) = G_n(\theta_0, g_0) + \mathrm{o_p}\left(n^{-1/2} + \left\|\hat{\theta}-\theta_0\right\|\right)$$

- Since $A(g_0)$ is invertible and $G_n(\theta_0, g_0) = O_p\left(n^{-1/2}\right)$ by concentration

$$\left\|\hat{\theta}-\theta_0\right\| = O_p\left(n^{-1/2}\right)$$

- Thus, we have asymptotic linearity

$$\sqrt{n}\left(\hat{\theta}-\theta_0\right) = \sqrt{n}A(g_0)^{-1}G_n(\theta_0, g_0) + \mathrm{o_p}(1) = \frac{1}{\sqrt{n_2}}\sum_{i\in S_2} A(g_0)^{-1}m(Z_i;\theta_0, g_0) + o_p(1)$$

- By CLT we get the theorem

Main Theorem

- If moment is Neyman orthogonal and RMSE of $\hat{g}$ is $o_p\left(n^{-1/4}\right)*$

$$\sqrt{n}\left(\hat{\theta}-\theta_0\right) \to N\left(0, A^{-1}\Sigma\left(\mathrm{A}^{-1}\right)^{\top}\right)$$

- $A = \nabla_\theta M(\theta_0, g_0)$ and $\Sigma = E\left[m(Z;\theta_0, g_0)\, m(Z;\theta_0, g_0)^{\top}\right]$

*plus regularity conditions