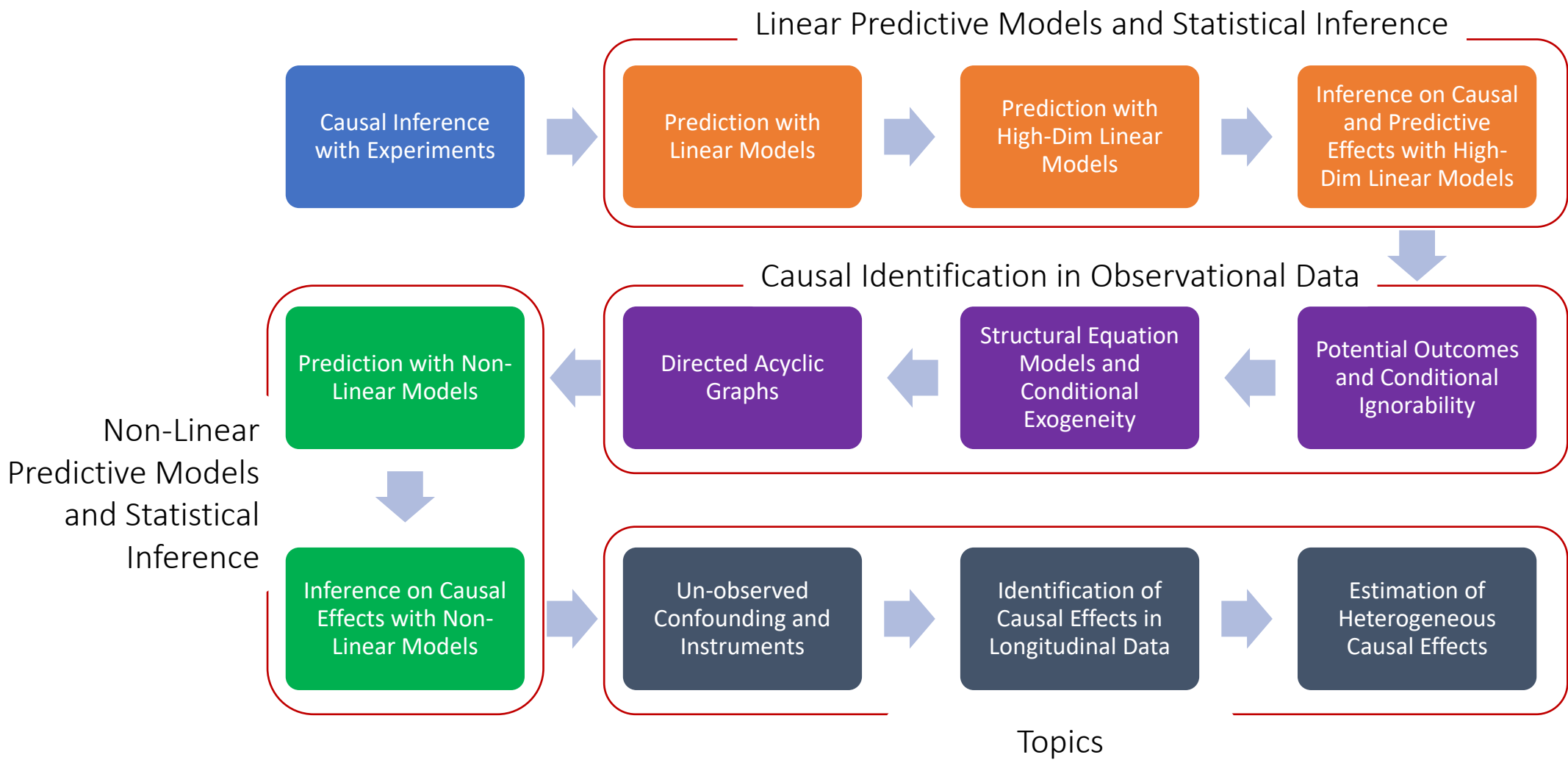
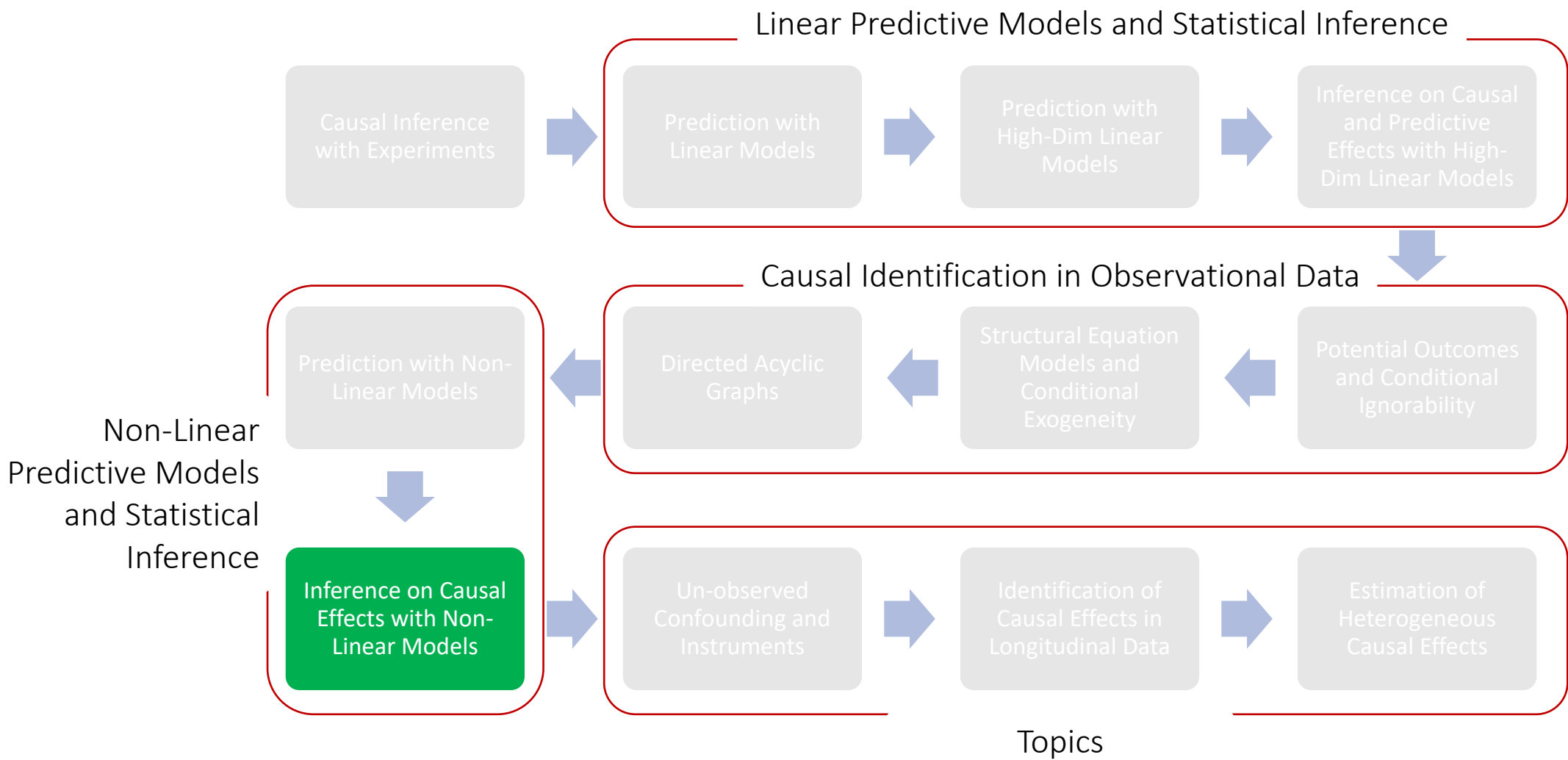


MS&E 228: Inference with Modern Non-Linear Prediction

Vasilis Syrgkanis

MS&E, Stanford

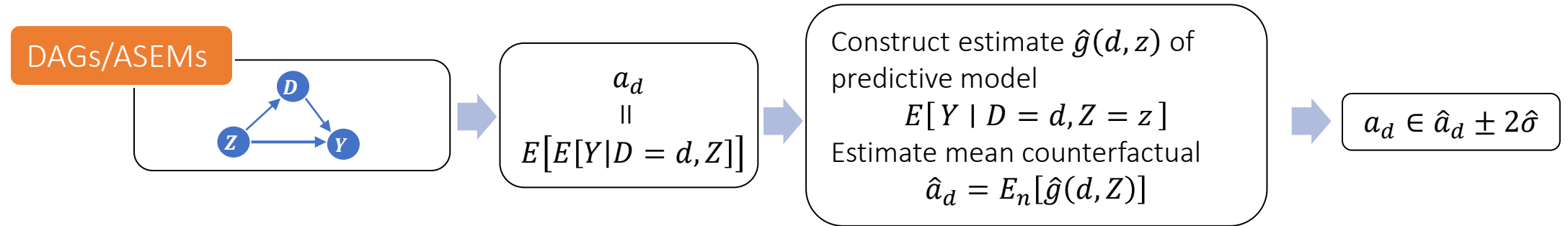




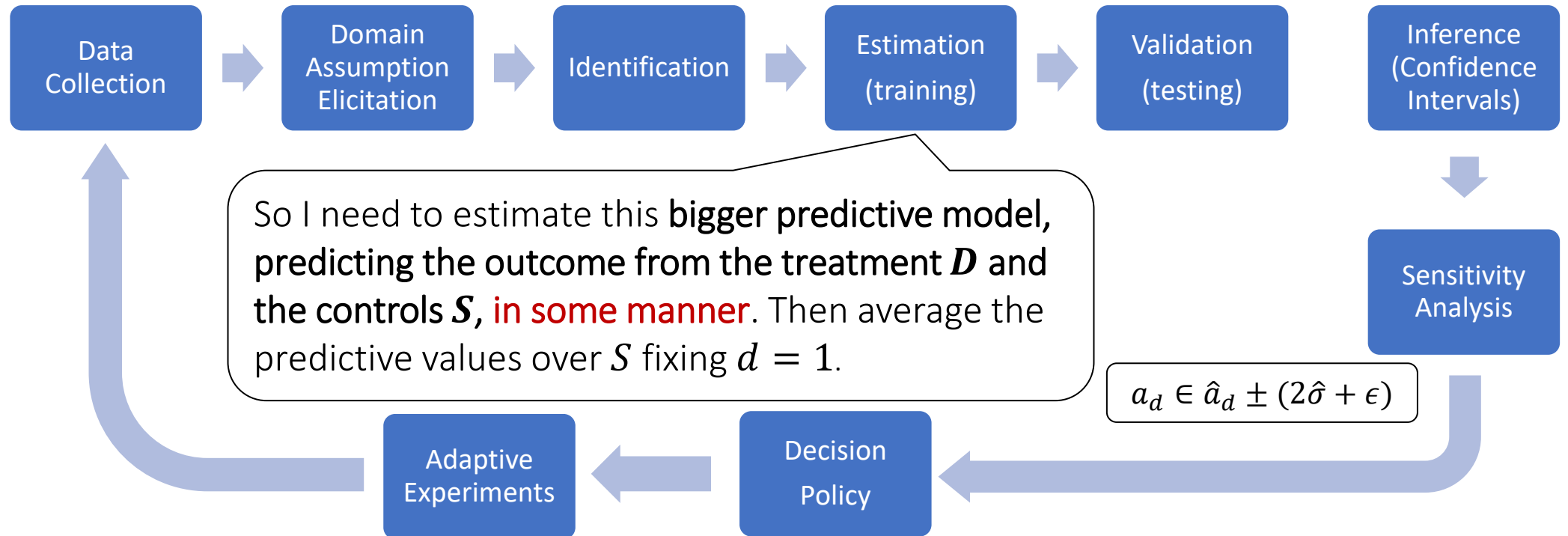
Recap of Last Lecture

Causal Inference Pipeline

Theory

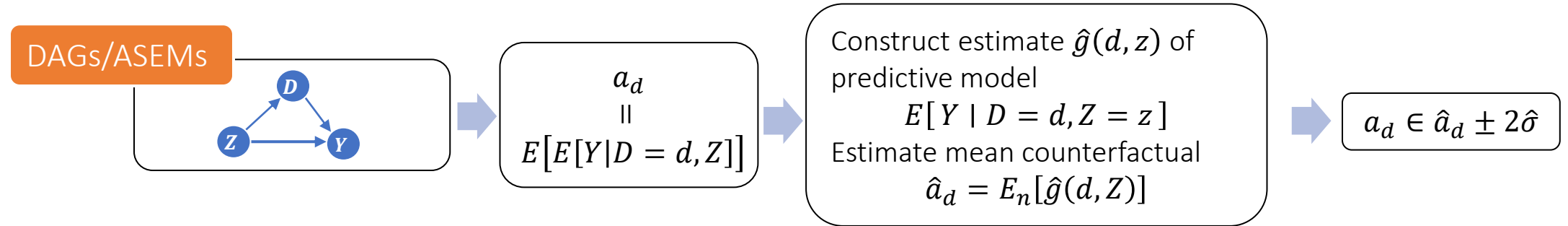


Practice

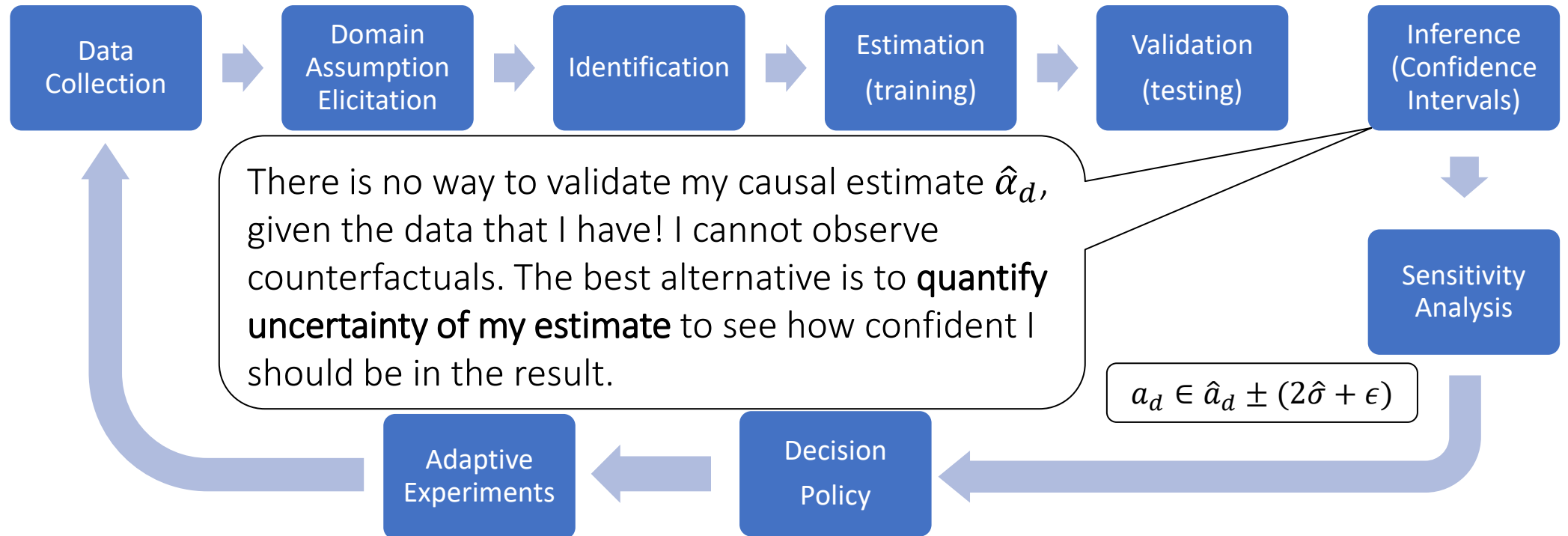


Causal Inference Pipeline

Theory



Practice



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)

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

- Why useful:

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

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

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

Let's take it to data

- We observe n samples Z_1, \dots, 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:

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

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

A General Estimation Framework

Semi-Parametric Moment Restrictions

- Observe samples Z_1, \dots, Z_n i.i.d. from data distribution D
- Distribution D satisfies vector of restrictions (aka moment conditions)
$$M(\theta_0, g_0) := E_{Z \sim D}[m(Z; \theta_0, g_0)] = 0$$
- $\theta_0 \in R^d$ finite dimensional target parameter of interest
- $g_0 \in G$ potentially infinite dimensional (an un-known function) we don't care (nuisance)
- g_0 is un-known and needs to be estimated from data

ATE under Conditional Exogeneity

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

- Want to estimate average effect θ_0 , which satisfies:

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

- Where:

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

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

Given n samples we want to produce estimate $\hat{\theta}$

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

- Consistency

$$\hat{\theta} \rightarrow_p \theta_0$$

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

- Finite sample parametric rate:

$$\|\hat{\theta} - \theta_0\| = o_p\left(\sqrt{d/n}\right)$$

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

- Asymptotic normality

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

- Construction of confidence intervals:

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

- Calculation of p -value for zero effect

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

- Asymptotic linearity

$$\sqrt{n}(\hat{\theta} - \theta_0) = \frac{1}{\sqrt{n}} \sum_{i=1}^n \phi_0(Z_i) + o_p(1)$$

- Consistency of bootstrap confidence intervals

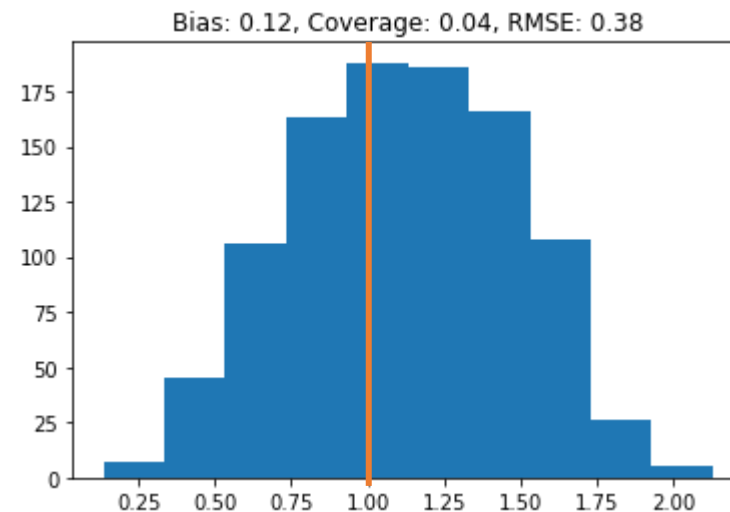
Natural Estimation Algorithm

- Estimate \hat{g} of g_0 from data
- Return solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) := \frac{1}{n} \sum_{i=1}^n m(Z_i; \hat{\theta}, \hat{g}) = 0$$

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])
```



Natural Estimation Algorithm (Draft 2)

- Split the data in half
- On first half, estimate \hat{g} of g_0
- On second half, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) := \frac{1}{n_2} \sum_{i \in S_2} m(Z_i; \hat{\theta}, \hat{g}) = 0$$

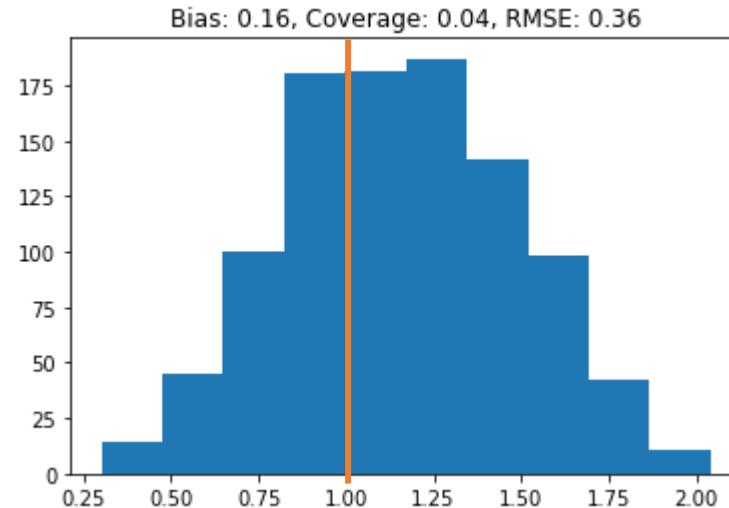
Natural Estimation Algorithm (Draft 3)

- 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
- Return solution $\hat{\theta}$ to cross-fitted empirical moment equation:

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

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])
```

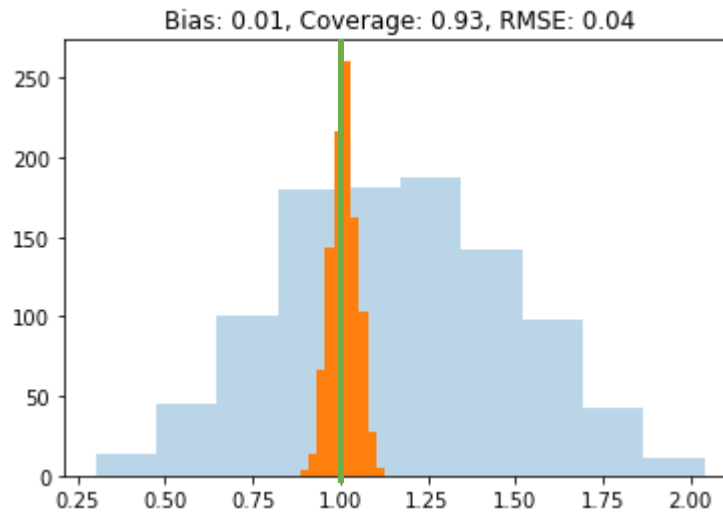


When is estimate $\hat{\theta}$ \sqrt{n} -asymptotically normal?

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)
    final = LinearRegression(fit_intercept=False).fit(Dres.reshape(-1, 1), yres)
    point = final.coef_[0]
    var = np.mean((Dres**2) * (yres - point*Dres)**2) / np.mean(Dres**2)
    stderr = np.sqrt(var / X.shape[0])
    return point, stderr
```

TO BE UNCOVERED!



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

Debiasing Intuition

ATE under Conditional Exogeneity

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

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

- Where:

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

- The moment $M(\theta, g)$ 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 Moment for ATE

- Add a “debiasing” correction:

$$\tilde{M}(\theta, g, a) = M(\theta, g) + E[a(D, X) (Y - g(D, X))]$$

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

$$\left. \frac{\partial}{\partial t} \tilde{M}(\theta_0, g_0 + t v, a_0) \right|_{t=0} = E[v(1, X) - v(0, X)] - E[a(D, X) v(D, X)] = 0$$

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

$$\begin{aligned} \theta &= E[a(D, X)Y] = E[a(D, X)E[Y | D, X]] \\ &= E[a(D, X)g(D, X)] = E[g(1, X) - g(0, X)] \end{aligned}$$

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 Moment is Insensitive

$$\tilde{M}(\theta, g, a) = M(\theta, g) + E[a(D, X) (Y - g(D, X))]$$

- Take derivative with respect to g at θ_0, g_0, a_0 in any direction $v \in G$

$$\left. \frac{\partial}{\partial t} \tilde{M}(\theta_0, g_0 + t v, a_0) \right|_{t=0} = E[v(1, X) - v(0, X)] - E[a(D, X) v(D, X)]$$

$$= 0$$

- Take derivative with respect to a at θ_0, g_0, a_0 in any direction $v \in A$

$$\left. \frac{\partial}{\partial t} \tilde{M}(\theta_0, g_0, a_0 + t v) \right|_{t=0} = E[v(D, X) (Y - g_0(D, X))]$$

$$= 0$$

Asymptotic Normality of De-biased Estimate

$$\hat{\theta} := E_n[m(Z; \hat{g}) + \hat{a}(X) \cdot (Y - \hat{g}(X))]$$

- Assume moment satisfies mean-squared-continuity (MSC)

$$\sqrt{E[m(Z; g)^2]} \leq M\sqrt{E[g(X)^2]}$$

- Assume \hat{g}, \hat{a} estimated on separate sample (or cross-fitting), are consistent and:

$$\sqrt{n} E[(a_0(X) - \hat{a}(X))(\hat{g}(X) - g_0(X))] \rightarrow_p 0$$

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

$$\sqrt{\frac{n}{\text{Var}_n(m(Z; \hat{g}) + \hat{a}(X) \cdot (Y - \hat{g}(X)))}} (\hat{\theta} - \theta_0) \rightarrow_d N(0,1)$$

General Theory: Neyman Orthogonality

Formal Definition

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

$$D_g M(\theta_0, g_0)[v] := \left. \frac{\partial}{\partial t} M(\theta_0, g_0 + t v) \right|_{t=0} = 0$$

Sample-Splitting Estimation Algorithm

- Split the data in half
- On first half, estimate \hat{g} of g_0
- On second half, solution $\hat{\theta}$ to empirical plug-in moment equation:

$$M_n(\hat{\theta}, \hat{g}) := \frac{1}{n_2} \sum_{i \in S_2} m(Z_i; \hat{\theta}, \hat{g}) = 0$$

Main Theorem

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

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

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

*plug regularity conditions

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

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

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

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

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

influence function

influence is a linear transformation of the moment; transforming from

- Consequently, it is **asymptotically normal**

$$\sqrt{n}(\hat{\theta} - \theta_0) \underset{d}{\sim} N(0, V), \quad V = E[\phi_0(Z)\phi_0(Z)']$$

Covariance of the influence function

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

$$\ell' \theta \in \left[\ell' \hat{\theta} \pm c \sqrt{\frac{\ell' \hat{V} \ell}{n}} \right],$$

$$\hat{V} = \text{Var}_n(\hat{\phi}(Z)),$$

Empirical variance of approximate influence

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

Approximate influence function

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

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 nuisance
```

```
    ghat = cross_val_predict(nuisance_estimator, Z, cv=nfold
```

```
    # use these predictions to define the empirical moment
```

```
    with respect to theta
```

```
    avg_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 parameters
```

```
    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
 $\theta \rightarrow E_n[m(Z; \theta, \hat{g})]$

Solve that
moment = 0
wrt θ

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

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

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

Main Theorem (linear moments)

- If moments are linear

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

- Estimate is closed form:

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

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

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

- Consequently, it is *asymptotically normal*

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

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

$$\ell' \theta \in \left[\ell' \hat{\theta} \pm c \sqrt{\frac{\ell' \hat{V} \ell}{n}} \right], \quad \hat{V} = \text{Var}_n(\hat{\phi}(Z)), \quad \hat{\phi}(Z) := -\hat{J}^{-1} m(Z; \hat{\theta}, \hat{g}), \quad \hat{J} = E_n[a(Z; \hat{g})]$$

Python Pseudocode

```
# General DML pseudocode for linear moments: m(Z; theta, g)
def general_dml_linear_moment(Z, ell, nfolds, alpha, nu, jacob):
    # construct out-of-fold predictions from the nuisance estimator
    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 parameter
    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 $E_n[v(Z; \hat{g})]$

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

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

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

ATE under Conditional Exogeneity

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

- Want to estimate average effect θ_0 , which satisfies:

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

- Where:

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

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

Better Moment for ATE

- Add a “debiasing” correction:

$$\tilde{M}(\theta, g, a) = M(\theta, g) + E[a(D, X) (Y - g(D, X))]$$

- What is a_0 ? Should be such that

$$E[a_0(D, X) g(D, X)] = E[g(1, X) - g(0, X)]$$

- The following works: inverse propensity scoring

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

- Doubly robust estimation algorithm

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*

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

- Consequently, it is *asymptotically normal*

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

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

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

Inference with Doubly Robust Algorithm

- Moment is Neyman orthogonal

$$m(Z; \theta, g) := g(1, X) - g(0, X) - a(D, X) (Y - g(D, X)) - \theta$$

- If RMSE of propensity and regression model goes down at rate $n^{1/4}$, plus regularity conditions
- Then the estimate $\hat{\theta}$ is *asymptotically linear*

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

- Consequently, it is *asymptotically normal*

$$\sqrt{n}(\hat{\theta} - \theta_0) \sim_a N(0, V), \quad V := E[m(Z; \theta_0, g_0)^2]$$

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

$$\ell' \theta \in \left[\ell' \hat{\theta} \pm c \sqrt{\frac{\ell' \hat{V} \ell}{n}} \right], \quad \hat{V} = \text{Var}_n(\hat{\phi}(Z)), \quad \hat{\phi}(Z) := -m(Z; \hat{\theta}, \hat{g}), \quad \hat{J} = \mathbf{1}$$

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

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

- Assume partially linear response

$$g_0(D, X) = E[Y \mid D, X] = \theta_0 D + f_0(X)$$

- Parameter of interest θ_0 is constant marginal effect of treatment

Partially Linear Model

- By definition of CEF we have the decomposition

$$Y = g_0(D, X) + \epsilon = \theta_0 D + f_0(X) + \epsilon, \quad E[\epsilon|D, X] = 0$$

- By definition of g_0 , for any function $q(D, X)$

$$E[(Y - \theta_0 D - f_0(X))q(D, X)] = E[E[Y - g_0(D, X) | D, X] q(D, X)] = 0$$

- Direct non-orthogonal method, estimate \hat{f} and solve:

$$E[(Y - \theta D - \hat{f}(X))D] = 0$$

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

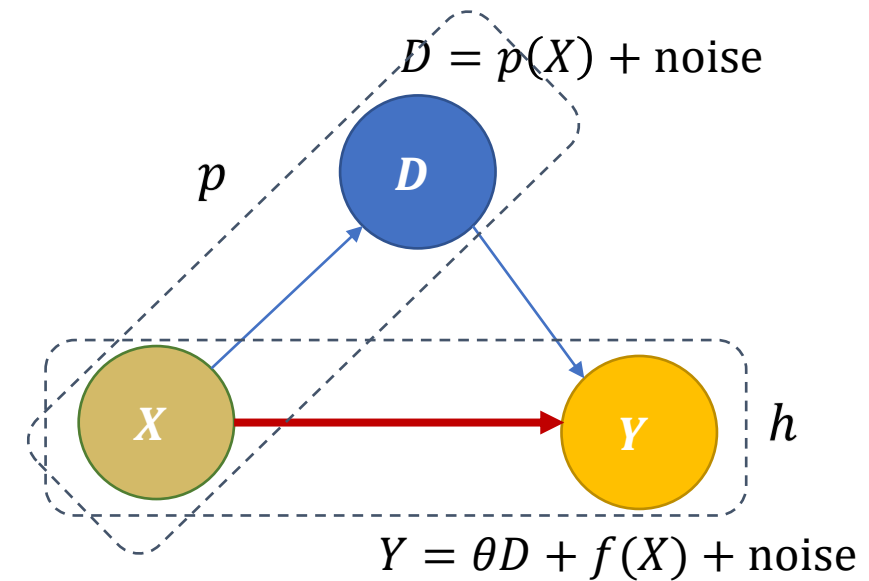
$$\tilde{Y} = \theta_0 \tilde{D} + \epsilon, \quad E[\epsilon|\tilde{D}] = 0$$

- Let's consider the residual outcome

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

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

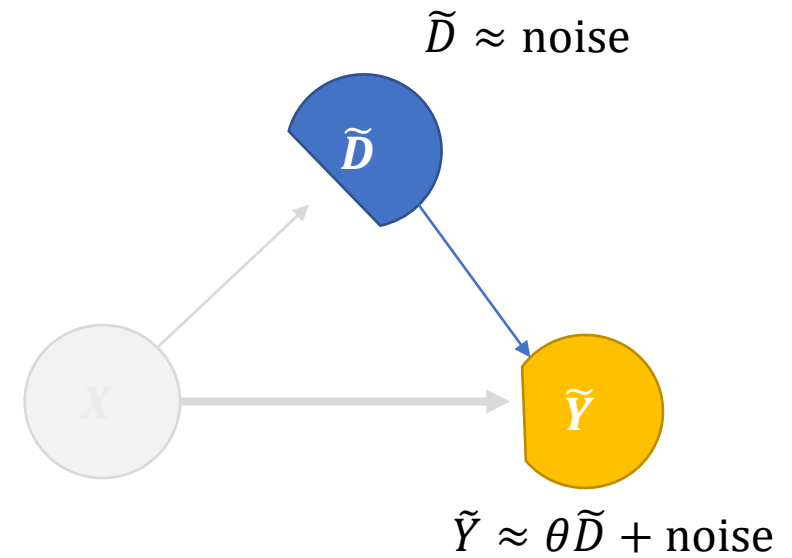
- Regress $Y \sim X$ with ML on first half, to get estimate $\hat{h}(S)$ of $E[Y|X]$
- Regress $D \sim X$ with ML on first half, to get estimate $\hat{p}(S)$ of $E[D|X]$
- Construct residuals on other half, $\tilde{D} := D - \hat{p}(X)$ and $\tilde{Y} := Y - \hat{h}(X)$
- Run OLS on residuals: $\tilde{Y}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

- OLS equivalent to solving moment condition:

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

- Orthogonal Moment condition:

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



Practical Variants of Sample-Splitting

Cross-fitting and semi-cross-fitting

Cross-fitting

- Sample splitting is statistically lossy
- Only half of the data are used for the final parameter estimation
- Can we utilize all the data?
- *Cross-fitting*: analogous to cross-validation
- Use the second half to train g and predict on first half
- Then calculate parameter using all the data

Cross-fitting Estimation Algorithm

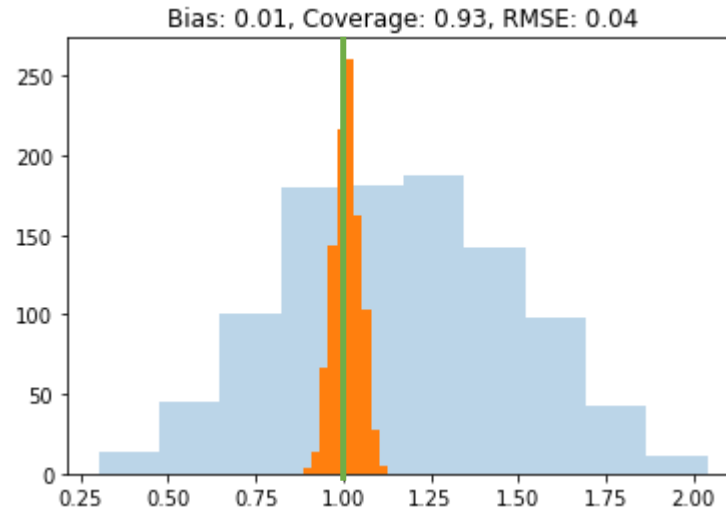
- Split the data in half
- On first half, estimate \hat{g}_1 of g_0 and predict on second half
- On second half, estimate \hat{g}_2 of g_0 and predict on first half
- On all data, solution $\hat{\theta}$ to empirical plug-in moment equation:

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

- In practice do this with $K \approx 3$ to 5 folds: for each fold k train on all other folds and predict on fold k

Natural Algorithm (Draft 3) Gone Right

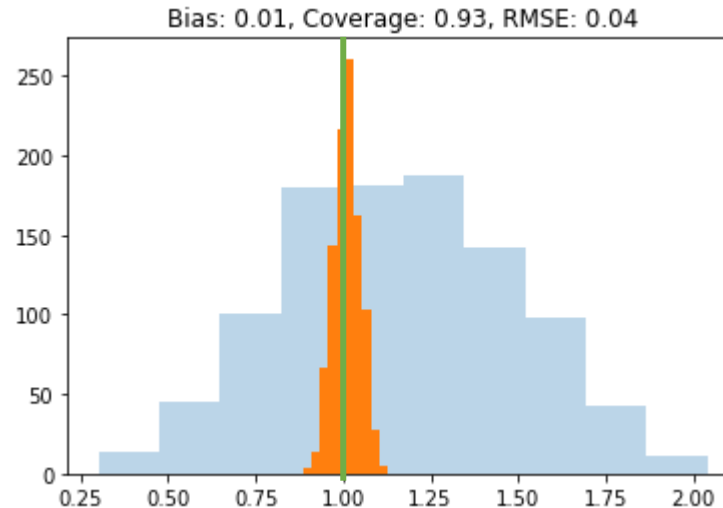
```
def dml2(X, D, y): # orthogonal dml with sample-splitting
    est_y = RandomForestRegressor(min_samples_leaf=20)
    yres = y - cross_val_predict(est_y, X, y, cv=3)
    est_t = RandomForestRegressor(min_samples_leaf=20)
    Dres = D - cross_val_predict(est_t, X, D, cv=3)
    theta = np.mean(yres * 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()
```



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-Cross-fitting Estimation Algorithm

- Split the data in half (*in practice K folds*)
- On first half, estimate $\hat{g}_1^{(1)}, \dots, \hat{g}_1^{(L)}$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2^{(1)}, \dots, \hat{g}_2^{(L)}$ 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:

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

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

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

Semi-Cross-fitting with Stacking

- Split the data in half (*in practice K folds*)
- On first half, estimate $\hat{g}_1^{(1)}, \dots, \hat{g}_1^{(L)}$ of g_0 and predict on second half
- On second half, estimate $\hat{g}_2^{(1)}, \dots, \hat{g}_2^{(L)}$ of g_0 and predict on first half
- Construct weights $\alpha_1, \dots, \alpha_\ell$ on the models using all the data (stacking)

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

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

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 weighing 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
$$\sqrt{L} = o(n^{1/4})$$

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

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

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

- 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 $\hat{g} \rightarrow g_0$: $A(g_0)(\hat{\theta} - \theta_0) = M(\hat{\theta}, \hat{g}) - M(\theta_0, \hat{g}) + o_p(\|\hat{\theta} - \theta_0\|)$
- Since $A(g_0)$ is invertible: $\|\hat{\theta} - \theta_0\| = O(\|M(\hat{\theta}, \hat{g})\| + \|M(\theta_0, \hat{g})\|) = O_p(n^{-1/2})$
- More fine-grained analysis of $M(\hat{\theta}, \hat{g})$ term, shows: $\sqrt{n}M(\hat{\theta}, \hat{g}) \rightarrow N(0, V)$

Proof of Main Theorem (visually)

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

$N(0, V) \leftarrow$

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

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

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

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

$$M(\theta_0, g_0) = 0$$

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

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

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

Proof of Main Theorem (algebraically)

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

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

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

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

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

- Thus

$$A(g_0) (\hat{\theta} - \theta_0) = \underbrace{M(\hat{\theta}, \hat{g}) - M_n(\hat{\theta}, \hat{g})}_{\rightarrow N(0, V)} + \underbrace{M(\theta_0, g_0) - M(\theta_0, \hat{g})}_{= o_p(n^{-1/2})} + o_p(\|\hat{\theta} - \theta_0\|)$$

$\rightarrow N(0, V)$
 via CLT
 + sample-splitting
 + concentration
 + $\hat{g} \rightarrow g_0$

$= o_p(n^{-1/2})$
 via orthogonality
 + $\|\hat{g} - g_0\| = o_p(n^{-1/4})$

Proof of Main Theorem: Orthogonality

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

- Thus

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

$G_n(\hat{\theta}, \hat{g})$

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 $\Rightarrow \|A(\hat{g}) - A_n(\hat{g})\| = o_p(1)$):
$$G_n(\hat{\theta}, \hat{g}) - G_n(\theta_0, \hat{g}) = (A(\hat{g}) - A_n(\hat{g})) (\hat{\theta} - \theta_0) = o_p(\|\hat{\theta} - \theta_0\|)$$
- Thus
$$A(g_0) (\hat{\theta} - \theta_0) = G_n(\theta_0, g_0) + G_n(\theta_0, \hat{g}) - G_n(\theta_0, g_0) + o_p(n^{-1/2} + \|\hat{\theta} - \theta_0\|)$$

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)

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

- Thus

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

Concluding

- So far

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

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

- Thus, we have asymptotic linearity

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

- By CLT we get the theorem

Main Theorem

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

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

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

*plus regularity conditions