High Dimensionality and Double Machine Learning

Causal Inference using Machine Learning
Master in Economics, UNT

Andres Mena

Spring 2024

Content

High-Dimensional Linear Regression and LASSO

Setting: We consider a high-dimensional linear model

$$Y = \beta' X + \varepsilon, \quad \varepsilon \perp X,$$

where the dimension of X (denoted p) can be very large, possibly much larger than n, the sample size.

High-Dimensional Linear Regression and LASSO

Setting: We consider a high-dimensional linear model

$$Y = \beta' X + \varepsilon, \quad \varepsilon \perp X,$$

where the dimension of X (denoted p) can be very large, possibly much larger than n, the sample size.

Challenge: Ordinary Least Squares (OLS) is prone to overfitting when p is large. High complexity leads to large variance of estimates and poor out-of-sample performance.

High-Dimensional Linear Regression and LASSO

Setting: We consider a high-dimensional linear model

$$Y = \beta' X + \varepsilon, \quad \varepsilon \perp X,$$

where the dimension of X (denoted p) can be very large, possibly much larger than n, the sample size.

Challenge: Ordinary Least Squares (OLS) is prone to overfitting when p is large. High complexity leads to large variance of estimates and poor out-of-sample performance.

Idea of LASSO: The LASSO estimator imposes a penalty on the size of the coefficients to control complexity:

$$\widehat{\beta}^{\text{LASSO}} = \arg\min_{b \in \mathbb{R}^p} \sum_{i=1}^{n} (Y_i - b' X_i)^2 + \lambda \sum_{j=1}^{p} |b_j|.$$

High-Dimensional Linear Regression and LASSO

Setting: We consider a high-dimensional linear model

$$Y = \beta' X + \varepsilon, \quad \varepsilon \perp X,$$

where the dimension of X (denoted p) can be very large, possibly much larger than n, the sample size.

Challenge: Ordinary Least Squares (OLS) is prone to overfitting when p is large. High complexity leads to large variance of estimates and poor out-of-sample performance.

Idea of LASSO: The LASSO estimator imposes a penalty on the size of the coefficients to control complexity:

$$\widehat{\beta}^{\text{LASSO}} = \arg\min_{b \in \mathbb{R}^p} \sum_{i=1}^{n} (Y_i - b' X_i)^2 + \lambda \sum_{j=1}^{p} |b_j|.$$

This penalty shrinks some coefficients towards zero, performing both regularization and variable selection.

Bias-Variance Trade-off in LASSO (1)

Bias-Variance Trade-off:

- **Variance Reduction:** By penalizing large coefficients, LASSO reduces variance in estimates, making predictions more stable.

Bias-Variance Trade-off in LASSO (1)

Bias-Variance Trade-off:

- **Variance Reduction:** By penalizing large coefficients, LASSO reduces variance in estimates, making predictions more stable.
- **Introduction of Bias:** The penalty $\lambda \sum |b_j|$ shrinks coefficients towards zero. This shrinkage induces bias in the estimates $\widehat{\beta}^{\text{LASSO}}$ compared to OLS.

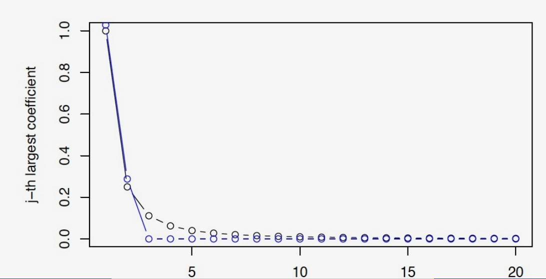
$$|\widehat{\beta}^{\text{LASSO}}| \;\leq\; |\widehat{\beta}^{\text{OLS}}|$$

Bias-Variance Trade-off in LASSO (1)

Bias-Variance Trade-off:

- **Variance Reduction:** By penalizing large coefficients, LASSO reduces variance in estimates, making predictions more stable.
- **Introduction of Bias:** The penalty $\lambda \sum |b_j|$ shrinks coefficients towards zero. This shrinkage induces bias in the estimates $\widehat{\beta}^{\text{LASSO}}$ compared to OLS.

$$|\widehat{\beta}^{\text{LASSO}}| \leq |\widehat{\beta}^{\text{OLS}}|$$



Optimal Penalties: Balancing Bias-Variance and Sparsity

Optimal Trade-off in LASSO:

- Carefully choosing λ balances bias and variance.

Optimal Penalties: Balancing Bias-Variance and Sparsity

Optimal Trade-off in LASSO:

- Carefully choosing λ balances bias and variance.
- A larger λ increases bias (more shrinkage) but lowers variance.

Optimal Penalties: Balancing Bias-Variance and Sparsity

Optimal Trade-off in LASSO:

- Carefully choosing λ balances bias and variance.
- A larger λ increases bias (more shrinkage) but lowers variance.
- A smaller λ reduces bias but may increase variance.

Optimal Penalties: Balancing Bias-Variance and Sparsity

Optimal Trade-off in LASSO:

- Carefully choosing λ balances bias and variance.
- A larger λ increases bias (more shrinkage) but lowers variance.
- A smaller λ reduces bias but may increase variance.

Goal: Find the sweet spot where predictive performance is optimized.

Optimal Penalties: Balancing Bias-Variance and Sparsity

Optimal Trade-off in LASSO:

- Carefully choosing λ balances bias and variance.
- A larger λ increases bias (more shrinkage) but lowers variance.
- A smaller λ reduces bias but may increase variance.

Goal: Find the sweet spot where predictive performance is optimized.

Penalty and Induced Sparsity:

- LASSO sets coefficients to zero if their marginal predictive benefit does not outweigh the penalty cost.

Optimal Penalties: Balancing Bias-Variance and Sparsity

Optimal Trade-off in LASSO:

- Carefully choosing λ balances bias and variance.
- A larger λ increases bias (more shrinkage) but lowers variance.
- A smaller λ reduces bias but may increase variance.

Goal: Find the sweet spot where predictive performance is optimized.

Penalty and Induced Sparsity:

- LASSO sets coefficients to zero if their marginal predictive benefit does not outweigh the penalty cost.
- $$\widehat{\beta}_j = 0 \text{ if } \left| \frac{\partial}{\partial \widehat{\beta}_j} \sum_i (Y_i - \widehat{\beta}' X_i)^2 \right| < \lambda.$$

Optimal Penalties: Balancing Bias-Variance and Sparsity

Optimal Trade-off in LASSO:

- Carefully choosing λ balances bias and variance.
- A larger λ increases bias (more shrinkage) but lowers variance.
- A smaller λ reduces bias but may increase variance.

Goal: Find the sweet spot where predictive performance is optimized.

Penalty and Induced Sparsity:

- LASSO sets coefficients to zero if their marginal predictive benefit does not outweigh the penalty cost.
- $$\widehat{\beta}_j = 0 \text{ if } \left| \frac{\partial}{\partial \widehat{\beta}_j} \sum_i (Y_i - \widehat{\beta}' X_i)^2 \right| < \lambda.$$
- This condition ensures that only variables with sufficiently high marginal predictive contribution survive the penalty, inducing sparsity in the model.

Predictive Performance of LASSO: Theoretical Guarantees

Theorem (Predictive Performance of Lasso)

Under approximate sparsity and suitable regularity conditions, if we choose λ as recommended (e.g., $\lambda \propto \sigma\sqrt{n\log(\max\{p,n\})}$) and let s be the effective dimension (number of important parameters):

$$\sqrt{E_X[(\beta'X - \widehat{\beta}'X)^2]} \leq \text{const} \cdot \sqrt{E[\varepsilon^2]}\sqrt{\frac{s\log(\max\{p,n\})}{n}}.$$

Moreover, with high probability:

- The number of regressors selected by LASSO is of order s.
- If $s\log(\max\{p,n\})/n$ is small, LASSO is close to the best linear predictor.

Predictive Performance of LASSO: Theoretical Guarantees

Theorem (Predictive Performance of Lasso)

Under approximate sparsity and suitable regularity conditions, if we choose λ as recommended (e.g., $\lambda \propto \sigma\sqrt{n \log(\max\{p, n\})}$) and let s be the effective dimension (number of important parameters):

$$\sqrt{E_X[(\beta' X - \widehat{\beta}' X)^2]} \leq \text{const} \cdot \sqrt{E[\varepsilon^2]}\sqrt{\frac{s \log(\max\{p, n\})}{n}}.$$

Moreover, with high probability:

- The number of regressors selected by LASSO is of order s.
- If $s \log(\max\{p, n\})/n$ is small, LASSO is close to the best linear predictor.

Implication: LASSO adapts to unknown sparsity, suffering a modest $\sqrt{\log(\max\{p, n\})}$ factor over the ideal $\sqrt{s/n}$ rate.

Conclusions on LASSO's Properties

- **High Bias in $\widehat{\beta}$:** LASSO deliberately shrinks coefficients, introducing bias relative to OLS.

Conclusions on LASSO's Properties

- **High Bias in $\widehat{\beta}$:** LASSO deliberately shrinks coefficients, introducing bias relative to OLS.
- **Low Variance:** The shrinkage reduces variance, leading to more stable predictions.

Conclusions on LASSO's Properties

- **High Bias in $\widehat{\beta}$:** LASSO deliberately shrinks coefficients, introducing bias relative to OLS.
- **Low Variance:** The shrinkage reduces variance, leading to more stable predictions.
- **Sparsity:** By setting some coefficients to zero, LASSO performs variable selection and yields parsimonious models.

Conclusions on LASSO's Properties

- **High Bias in $\widehat{\beta}$:** LASSO deliberately shrinks coefficients, introducing bias relative to OLS.
- **Low Variance:** The shrinkage reduces variance, leading to more stable predictions.
- **Sparsity:** By setting some coefficients to zero, LASSO performs variable selection and yields parsimonious models.
- **Improved Predictive Performance for Y:** Despite the bias in $\widehat{\beta}$, the overall prediction $\widehat{\beta}'X$ often outperforms OLS predictions out-of-sample, especially when p is large and s is relatively small.

Bottom Line: LASSO trades off some bias in parameter estimates for substantial gains in predictive stability and interpretability. This is useful for high-dimensional econometric applications.

Content

Naive Estimator (1)

Naive Estimator Setup:

- Consider two separate LASSO regressions:

$$\widehat{\beta}_1 = \arg\min_{\beta_1} \sum_{i:D_i=1} (Y_i - W_i'\beta_1)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_{1j}|,$$

and

$$\widehat{\beta}_0 = \arg\min_{\beta_0} \sum_{i:D_i=0} (Y_i - W_i'\beta_0)^2 + \lambda_0 \sum_{j=1}^{p} |\beta_{0j}|.$$

Naive Estimator (1)

Naive Estimator Setup:

- Consider two separate LASSO regressions:

$$\widehat{\beta}_1 = \arg\min_{\beta_1} \sum_{i:D_i=1} (Y_i - W_i'\beta_1)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_{1j}|,$$

and

$$\widehat{\beta}_0 = \arg\min_{\beta_0} \sum_{i:D_i=0} (Y_i - W_i'\beta_0)^2 + \lambda_0 \sum_{j=1}^{p} |\beta_{0j}|.$$

- Then form the naive ATE estimator:

$$\widehat{\alpha} = \frac{1}{N} \sum_{i=1}^{N} W_i'(\widehat{\beta}_1 - \widehat{\beta}_0).$$

Naive Estimator (2)

Neyman Orthogonality and Bias:

- The moment condition for the naive estimator is:

$$E[W'(\beta_1 - \beta_0) - \alpha] = 0.$$

Naive Estimator (2)

Neyman Orthogonality and Bias:

- The moment condition for the naive estimator is:

$$E[W'(\beta_1 - \beta_0) - \alpha] = 0.$$

- Derivatives w.r.t. β_1 and β_0:

$$\frac{\partial}{\partial \beta_1} E[W'(\beta_1 - \beta_0) - \alpha] = E[W] \neq 0, \quad \frac{\partial}{\partial \beta_0} E[W'(\beta_1 - \beta_0) - \alpha] = -E[W]$$

Naive Estimator (2)

Neyman Orthogonality and Bias:

- The moment condition for the naive estimator is:

$$E[W'(\beta_1 - \beta_0) - \alpha] = 0.$$

- Derivatives w.r.t. β_1 and β_0:

$$\frac{\partial}{\partial \beta_1} E[W'(\beta_1 - \beta_0) - \alpha] = E[W] \neq 0, \quad \frac{\partial}{\partial \beta_0} E[W'(\beta_1 - \beta_0) - \alpha] = -E[W$$

- **Not Neyman Orthogonal:** Since the derivatives are not zero, the moment condition is not orthogonal. Thus, the estimation error in $(\widehat{\beta}_1 - \widehat{\beta}_0)$ induces slow bias convergence for $\widehat{\alpha}$.

Naive Estimator (2)

Neyman Orthogonality and Bias:

- The moment condition for the naive estimator is:

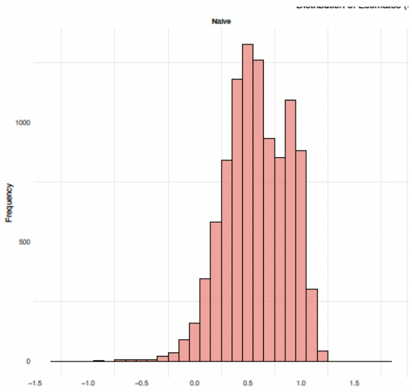
$$E[W'(\beta_1 - \beta_0) - \alpha] = 0.$$

- Derivatives w.r.t. β_1 and β_0:

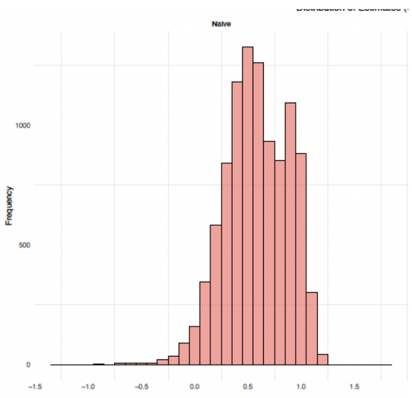
$$\frac{\partial}{\partial \beta_1} E[W'(\beta_1 - \beta_0) - \alpha] = E[W] \neq 0, \quad \frac{\partial}{\partial \beta_0} E[W'(\beta_1 - \beta_0) - \alpha] = -E[W]$$

- **Not Neyman Orthogonal:** Since the derivatives are not zero, the moment condition is not orthogonal. Thus, the estimation error in $(\widehat{\beta}_1 - \widehat{\beta}_0)$ induces slow bias convergence for $\widehat{\alpha}$.
- **Consequence:** The bias converges at a slower than $\sqrt{n}$-rate, rendering standard inference methods unreliable for moderate sample sizes.

Why the Naive Estimator Performs Poorly



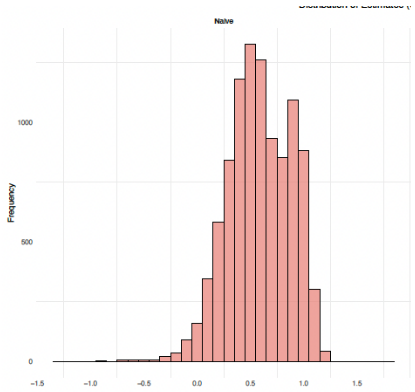
Why the Naive Estimator Performs Poorly



Key Insight:

- The naive method selects only strong predictors of Y.

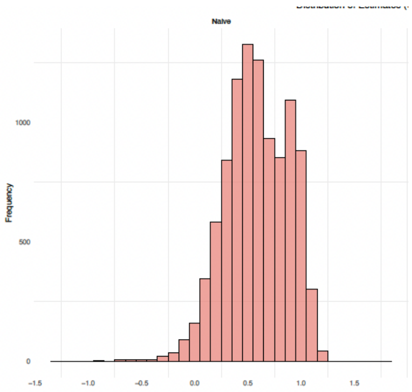
Why the Naive Estimator Performs Poorly



Key Insight:

- The naive method selects only strong predictors of Y.
- Weak Y-predictors that strongly predict D are dropped, causing severe omitted variable bias.

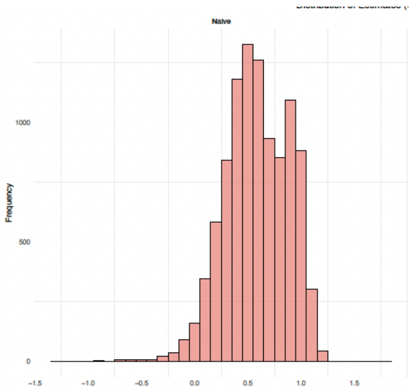
Why the Naive Estimator Performs Poorly



Key Insight:

- The naive method selects only strong predictors of Y.
- Weak Y-predictors that strongly predict D are dropped, causing severe omitted variable bias.
- The orthogonal approach (Double Lasso) solves two prediction problems (for Y and D) and includes controls that matter for either, thus "de-confounding" the residuals.

Why the Naive Estimator Performs Poorly



Key Insight:

- The naive method selects only strong predictors of Y.
- Weak Y-predictors that strongly predict D are dropped, causing severe omitted variable bias.
- The orthogonal approach (Double Lasso) solves two prediction problems (for Y and D) and includes controls that matter for either, thus "de-confounding" the residuals.

Homework: Replicate CIML Example 4.3.1.

Double Lasso Procedure

Partialling-Out via Frisch-Waugh-Lovell:

$$Y = \alpha D + \beta' W + \epsilon.$$

Double Lasso Procedure

Partialling-Out via Frisch-Waugh-Lovell:

$$Y = \alpha D + \beta' W + \epsilon.$$

After partialling out W:

$$\tilde{Y} = \alpha \tilde{D} + \tilde{\epsilon}, \quad \text{with } E[\tilde{\epsilon}\tilde{D}] = 0.$$

Double Lasso Procedure

Partialling-Out via Frisch-Waugh-Lovell:

$$Y = \alpha D + \beta' W + \epsilon.$$

After partialling out W:

$$\tilde{Y} = \alpha \tilde{D} + \tilde{\epsilon}, \quad \text{with } E[\tilde{\epsilon}\tilde{D}] = 0.$$

Double Lasso Steps:

1. Run LASSO: Y on W and D on W to get residuals:

$$\tilde{Y} = Y - \widehat{\gamma}'_{YW} W, \quad \tilde{D} = D - \widehat{\gamma}'_{DW} W.$$

Double Lasso Procedure

Partialling-Out via Frisch-Waugh-Lovell:

$$Y = \alpha D + \beta' W + \epsilon.$$

After partialling out W:

$$\tilde{Y} = \alpha \tilde{D} + \tilde{\epsilon}, \quad \text{with } E[\tilde{\epsilon}\tilde{D}] = 0.$$

Double Lasso Steps:

1. Run LASSO: Y on W and D on W to get residuals:

$$\tilde{Y} = Y - \widehat{\gamma}'_{YW} W, \quad \tilde{D} = D - \widehat{\gamma}'_{DW} W.$$

2. Run OLS of $\tilde{Y}$ on $\tilde{D}$:

$$\widehat{\alpha} = \frac{\sum \tilde{D}_i \tilde{Y}_i}{\sum \tilde{D}_i^2}.$$

Double Lasso Conditions

Approximate Sparsity: For good performance, we need the coefficients for Y-on-W and D-on-W to be approximately sparse:

$$|\gamma_{YW}|_{(j)} \leq Aj^{-a}, \quad |\gamma_{DW}|_{(j)} \leq Aj^{-a}, \quad a > 1.$$

Double Lasso Conditions

Approximate Sparsity: For good performance, we need the coefficients for Y-on-W and D-on-W to be approximately sparse:

$$|\gamma_{YW}|_{(j)} \leq Aj^{-a}, \quad |\gamma_{DW}|_{(j)} \leq Aj^{-a}, \quad a > 1.$$

Tuning:

- Choose λ_1, λ_2 (penalties) using theory-driven plug-in rules or related methods.

Double Lasso Conditions

Approximate Sparsity: For good performance, we need the coefficients for Y-on-W and D-on-W to be approximately sparse:

$$|\gamma_{YW}|_{(j)} \leq Aj^{-a}, \quad |\gamma_{DW}|_{(j)} \leq Aj^{-a}, \quad a > 1.$$

Tuning:

- Choose λ_1, λ_2 (penalties) using theory-driven plug-in rules or related methods.
- Ensures that the first-step estimation does not have a first-order effect on $\widehat{\alpha}$.

Double Lasso Conditions

Approximate Sparsity: For good performance, we need the coefficients for Y-on-W and D-on-W to be approximately sparse:

$$|\gamma_{YW}|_{(j)} \leq Aj^{-a}, \quad |\gamma_{DW}|_{(j)} \leq Aj^{-a}, \quad a > 1.$$

Tuning:

- Choose λ_1, λ_2 (penalties) using theory-driven plug-in rules or related methods.
- Ensures that the first-step estimation does not have a first-order effect on $\widehat{\alpha}$.

Without these conditions, and relying solely on cross-validation, performance may suffer in moderate samples.

Inference Under Double Lasso

Theorem 4.2.1 (Adaptive Inference with Double Lasso in High-Dimensional Regression)

Under the stated approximate sparsity, the conditions required for Theorem 3.2.1 (e.g. restricted isometry), and additional regularity conditions, the estimation error in D_i and Y_i has no first-order effect on $\widehat{\alpha}$, and

$$\sqrt{n}(\widehat{\alpha} - \alpha) \xrightarrow{d} N(0, V),$$

where

$$V = \left(E[D^2]\right)^{-1} E[D^2\epsilon^2] \left(E[D^2]\right)^{-1}.$$

Inference Under Double Lasso

Theorem 4.2.1 (Adaptive Inference with Double Lasso in High-Dimensional Regression)

Under the stated approximate sparsity, the conditions required for Theorem 3.2.1 (e.g. restricted isometry), and additional regularity conditions, the estimation error in D_i and Y_i has no first-order effect on $\widehat{\alpha}$, and

$$\sqrt{n}(\widehat{\alpha} - \alpha) \xrightarrow{d} N(0, V),$$

where

$$V = \left(E[D^2]\right)^{-1} E[D^2\epsilon^2] \left(E[D^2]\right)^{-1}.$$

he standard error of $\widehat{\alpha}$ is given by:

$$\mathrm{SE}(\widehat{\alpha}) = \sqrt{\frac{\widehat{V}}{n}},$$

where $\widehat{V}$ is a consistent estimator of V.

Inference Under Double Lasso

Theorem 4.2.1 (Adaptive Inference with Double Lasso in High-Dimensional Regression)

Under the stated approximate sparsity, the conditions required for Theorem 3.2.1 (e.g. restricted isometry), and additional regularity conditions, the estimation error in D_i and Y_i has no first-order effect on $\widehat{\alpha}$, and

$$\sqrt{n}(\widehat{\alpha} - \alpha) \xrightarrow{d} N(0, V),$$

where

$$V = \left(E[D^2]\right)^{-1} E[D^2 \epsilon^2] \left(E[D^2]\right)^{-1}.$$

he standard error of $\widehat{\alpha}$ is given by:

$$\text{SE}(\widehat{\alpha}) = \sqrt{\frac{\widehat{V}}{n}},$$

where $\widehat{V}$ is a consistent estimator of V.

A 95% confidence interval for α is:

$$\left[\widehat{\alpha} \pm 1.96 \cdot \sqrt{\frac{\widehat{V}}{n}}\right].$$

Neyman Orthogonality

Concept: Neyman orthogonality ensures local insensitivity of the target parameter to perturbations in nuisance parameters.

Neyman Orthogonality

Concept: Neyman orthogonality ensures local insensitivity of the target parameter to perturbations in nuisance parameters.

Key Idea:

- The target parameter α depends on nuisance parameters $\eta = (\gamma_D, \gamma_Y)$, representing projection coefficients.

Neyman Orthogonality

Concept: Neyman orthogonality ensures local insensitivity of the target parameter to perturbations in nuisance parameters.

Key Idea:

- The target parameter α depends on nuisance parameters $\eta = (\gamma_D, \gamma_Y)$, representing projection coefficients.
- The Double Lasso estimator $\widehat{\alpha}(\eta)$ satisfies:

$$\partial_\eta \alpha(\eta_0) = 0,$$

  where $\eta_0 = (\gamma_D^0, \gamma_Y^0)$ are the true nuisance parameters.

Neyman Orthogonality

Concept: Neyman orthogonality ensures local insensitivity of the target parameter to perturbations in nuisance parameters.

Key Idea:

- The target parameter α depends on nuisance parameters $\eta = (\gamma_D, \gamma_Y)$, representing projection coefficients.
- The Double Lasso estimator $\widehat{\alpha}(\eta)$ satisfies:

$$\partial_\eta \alpha(\eta_0) = 0,$$

  where $\eta_0 = (\gamma_D^0, \gamma_Y^0)$ are the true nuisance parameters.

Implication:

- Estimation errors in first-step Lasso regressions for nuisance components do not propagate into first-order bias for $\widehat{\alpha}$.
- Enables $\sqrt{n}$-consistent inference on α, even in high-dimensional settings.

Neyman Orthogonality

Derivation:

- The Double Lasso residuals are defined as:

$$\tilde{Y}(\eta) = Y - \eta_Y^\top W, \quad \tilde{D}(\eta) = D - \eta_D^\top W.$$

Neyman Orthogonality

Derivation:

- The Double Lasso residuals are defined as:

$$\tilde{Y}(\eta) = Y - \eta_Y^\top W, \quad \tilde{D}(\eta) = D - \eta_D^\top W.$$

- $\alpha(\eta)$ is implicitly defined via the moment condition:

$$M(a, \eta) = E\left[(\tilde{Y}(\eta) - a\tilde{D}(\eta))\tilde{D}(\eta)\right] = 0.$$

Neyman Orthogonality

Derivation:

- The Double Lasso residuals are defined as:

$$\tilde{Y}(\eta) = Y - \eta_Y^\top W, \quad \tilde{D}(\eta) = D - \eta_D^\top W.$$

- $\alpha(\eta)$ is implicitly defined via the moment condition:

$$M(a, \eta) = E\left[(\tilde{Y}(\eta) - a\tilde{D}(\eta))\tilde{D}(\eta)\right] = 0.$$

- Using the Implicit Function Theorem:

$$\partial_\eta \alpha(\eta_0) = -\left(\partial_a M(\alpha, \eta_0)\right)^{-1} \partial_\eta M(\alpha, \eta_0).$$

Neyman Orthogonality

- The derivative of $M(a, \eta)$ with respect to η is:

$$\partial_\eta M(\alpha, \eta_0) = (\partial_{\eta_Y} M(\alpha, \eta_0), \partial_{\eta_D} M(\alpha, \eta_0))$$

where each term is computed as follows:

Neyman Orthogonality

- The derivative of $M(a, \eta)$ with respect to η is:

$$\partial_\eta M(\alpha, \eta_0) = (\partial_{\eta_Y} M(\alpha, \eta_0), \partial_{\eta_D} M(\alpha, \eta_0))$$

where each term is computed as follows:

- For $\partial_{\eta_Y} M(\alpha, \eta_0)$:

$$\partial_{\eta_Y} M(\alpha, \eta_0) = -E[W \tilde{Y}(\eta_0)] + 2\alpha E[W \tilde{D}(\eta_0)].$$

Substituting:

$$\tilde{Y}(\eta_0) = Y - \eta_Y^\top W \quad \text{and} \quad \tilde{D}(\eta_0) = D - \eta_D^\top W,$$

gives:

$$\partial_{\eta_Y} M(\alpha, \eta_0) = -E[W(Y - \eta_Y^\top W)] + 2\alpha E[W(D - \eta_D^\top W)] = 0.$$

Neyman Orthogonality

- The derivative of $M(a, \eta)$ with respect to η is:

$$\partial_\eta M(\alpha, \eta_0) = (\partial_{\eta_Y} M(\alpha, \eta_0), \partial_{\eta_D} M(\alpha, \eta_0))$$

  where each term is computed as follows:

- For $\partial_{\eta_Y} M(\alpha, \eta_0)$:

$$\partial_{\eta_Y} M(\alpha, \eta_0) = -E[W\tilde{Y}(\eta_0)] + 2\alpha E[W\tilde{D}(\eta_0)].$$

  Substituting:

$$\tilde{Y}(\eta_0) = Y - \eta_Y^\top W \quad \text{and} \quad \tilde{D}(\eta_0) = D - \eta_D^\top W,$$

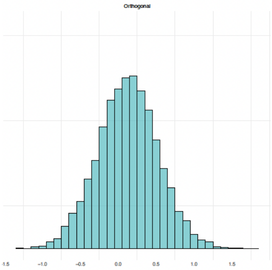
  gives:

$$\partial_{\eta_Y} M(\alpha, \eta_0) = -E[W(Y - \eta_Y^\top W)] + 2\alpha E[W(D - \eta_D^\top W)] = 0.$$

- For $\partial_{\eta_D} M(\alpha, \eta_0)$:

$$\partial_{\eta_D} M(\alpha, \eta_0) = E[W\tilde{D}(\eta_0)] = E[W(D - \eta_D^\top W)] = 0.$$

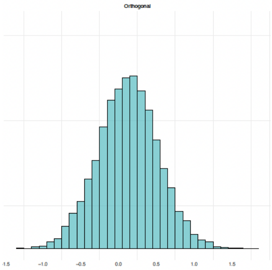
Summary and Conclusions



Key Takeaways:

- The naive LASSO-based estimator fails because it is not Neyman orthogonal, leading to slow bias convergence.

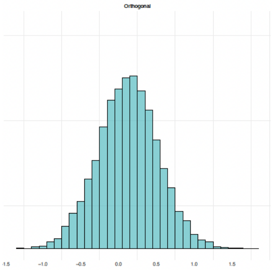
Summary and Conclusions



Key Takeaways:

- The naive LASSO-based estimator fails because it is not Neyman orthogonal, leading to slow bias convergence.
- Double Lasso solves two separate prediction problems (Y on W and D on W) and uses residuals to form an orthogonal moment condition.

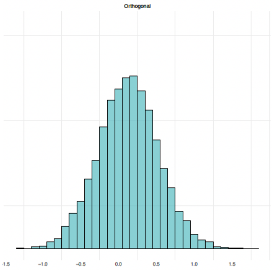
Summary and Conclusions



Key Takeaways:

- The naive LASSO-based estimator fails because it is not Neyman orthogonal, leading to slow bias convergence.
- Double Lasso solves two separate prediction problems (Y on W and D on W) and uses residuals to form an orthogonal moment condition.
- Under approximate sparsity and proper tuning, Double Lasso achieves $\sqrt{n}$-rate inference and classical-style confidence intervals for the target parameter.

Summary and Conclusions



Key Takeaways:

- The naive LASSO-based estimator fails because it is not Neyman orthogonal, leading to slow bias convergence.
- Double Lasso solves two separate prediction problems (Y on W and D on W) and uses residuals to form an orthogonal moment condition.
- Under approximate sparsity and proper tuning, Double Lasso achieves $\sqrt{n}$-rate inference and classical-style confidence intervals for the target parameter.
- Neyman orthogonality is the key principle ensuring that errors in nuisance estimation do not propagate to first-order bias in the final estimate.

Content

Partially Linear Model (PLM)

Model Setup:

- The PLM captures both linear and nonlinear relationships:

$$Y = \beta D + g(X) + \epsilon, \quad \text{where } E[\epsilon|D, X] = 0.$$

Partially Linear Model (PLM)

Model Setup:

- The PLM captures both linear and nonlinear relationships:

$$Y = \beta D + g(X) + \epsilon, \quad \text{where } E[\epsilon|D, X] = 0.$$

- Components:
  - Y: Outcome variable.
  - D: Regressor of interest (e.g., treatment or exposure).
  - X: High-dimensional control variables or features.
  - $g(X)$: Nonlinear component capturing the effect of X.

Partially Linear Model (PLM)

Model Setup:

- The PLM captures both linear and nonlinear relationships:

$$Y = \beta D + g(X) + \epsilon, \quad \text{where } E[\epsilon|D, X] = 0.$$

- Components:
  - Y: Outcome variable.
  - D: Regressor of interest (e.g., treatment or exposure).
  - X: High-dimensional control variables or features.
  - $g(X)$: Nonlinear component capturing the effect of X.
- β: Measures the predictive or causal effect of D on Y, controlling for X.

Residualized Vectors

Partialling-out X:

- Define residualized Y and D by removing the influence of X:

$$\tilde{Y} := Y - \ell(X), \quad \tilde{D} := D - m(X),$$

where:

$$\ell(X) := E[Y|X], \quad m(X) := E[D|X].$$

Residualized Vectors

Partialling-out X:

- Define residualized Y and D by removing the influence of X:

$$\tilde{Y} := Y - \ell(X), \quad \tilde{D} := D - m(X),$$

where:

$$\ell(X) := E[Y|X], \quad m(X) := E[D|X].$$

- Substituting into the PLM:

$$\tilde{Y} = \beta\tilde{D} + \epsilon, \quad \text{with } E[\epsilon\tilde{D}] = 0.$$

Residualized Vectors

Partialling-out X:

- Define residualized Y and D by removing the influence of X:

$$\tilde{Y} := Y - \ell(X), \quad \tilde{D} := D - m(X),$$

  where:

$$\ell(X) := E[Y|X], \quad m(X) := E[D|X].$$

- Substituting into the PLM:

$$\tilde{Y} = \beta\tilde{D} + \epsilon, \quad \text{with } E[\epsilon\tilde{D}] = 0.$$

Key Insight:

- Residualization isolates the effect of D on Y by removing confounding through X.
- The moment condition $E[\tilde{Y} - \beta\tilde{D}]\tilde{D} = 0$ identifies β.

FWL Partialling-Out for PLM

Theorem

- Suppose Y, D, and X have bounded second moments.

FWL Partialling-Out for PLM

Theorem

- Suppose Y, D, and X have bounded second moments.
- Then β is identified by the population regression of $\tilde{Y}$ on $\tilde{D}$:

$$\beta := \arg\min_b E[(\tilde{Y} - b\tilde{D})^2].$$

FWL Partialling-Out for PLM

Theorem

- Suppose Y, D, and X have bounded second moments.
- Then β is identified by the population regression of $\tilde{Y}$ on $\tilde{D}$:

$$\beta := \arg\min_b E[(\tilde{Y} - b\tilde{D})^2].$$

- Explicit solution:

$$\beta = \frac{E[\tilde{Y}\tilde{D}]}{E[\tilde{D}^2]}.$$

FWL Partialling-Out for PLM

Theorem

- Suppose Y, D, and X have bounded second moments.
- Then β is identified by the population regression of $\tilde{Y}$ on $\tilde{D}$:

$$\beta := \arg\min_b E[(\tilde{Y} - b\tilde{D})^2].$$

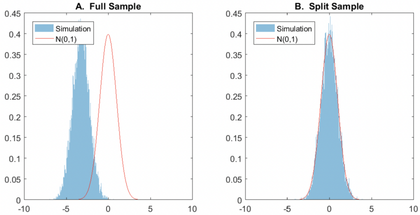
- Explicit solution:

$$\beta = \frac{E[\tilde{Y}\tilde{D}]}{E[\tilde{D}^2]}.$$

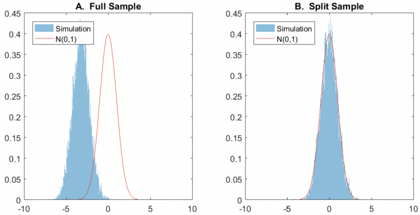
Interpretation:

- β is the regression coefficient of residualized Y on residualized D, generalizing the Frisch-Waugh-Lovell theorem to PLMs.
- Residuals ensure confounding from X is removed.

Cross-Fitting and Overfitting



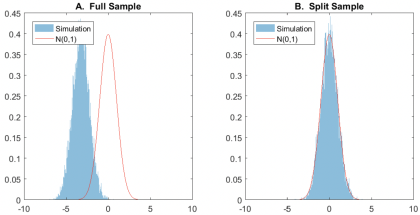
Cross-Fitting and Overfitting



Why Cross-Fitting?

- Overfitting during the estimation of residualized quantities (e.g., $\tilde{Y}, \tilde{D}$) can introduce bias into the estimation of β.

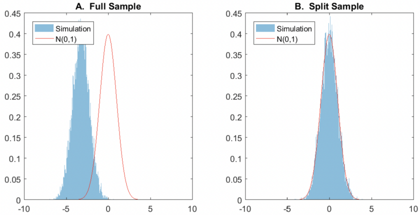
Cross-Fitting and Overfitting



Why Cross-Fitting?

- Overfitting during the estimation of residualized quantities (e.g., $\tilde{Y}, \tilde{D}$) can introduce bias into the estimation of β.
- Data-driven tuning methods like cross-validation often lead to mild overfitting, as the same data is used repeatedly.

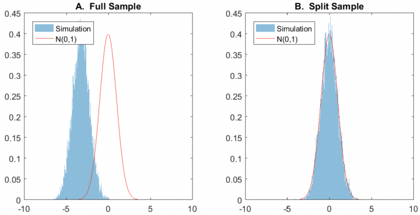
Cross-Fitting and Overfitting



Why Cross-Fitting?

- Overfitting during the estimation of residualized quantities (e.g., $\tilde{Y}, \tilde{D}$) can introduce bias into the estimation of β.
- Data-driven tuning methods like cross-validation often lead to mild overfitting, as the same data is used repeatedly.
- Cross-fitting mitigates this issue by using sample-splitting:
  - Nuisance parameters ($\ell(X), m(X)$) are estimated on one part of the data.
  - Residuals ($\tilde{Y}, \tilde{D}$) are computed on another.

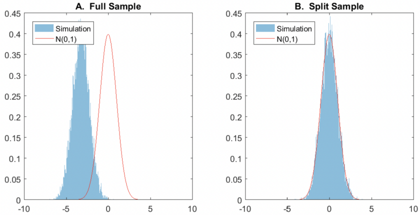
Cross-Fitting and Overfitting



Why Cross-Fitting?

- Overfitting during the estimation of residualized quantities (e.g., $\tilde{Y}, \tilde{D}$) can introduce bias into the estimation of β.
- Data-driven tuning methods like cross-validation often lead to mild overfitting, as the same data is used repeatedly.
- Cross-fitting mitigates this issue by using sample-splitting:
  - Nuisance parameters ($\ell(X), m(X)$) are estimated on one part of the data.
  - Residuals ($\tilde{Y}, \tilde{D}$) are computed on another.
- Even when using simple learners like LASSO, cross-fitting is recommended unless theoretical penalty levels (λ) are used.

Cross-Fitting and Overfitting



Why Cross-Fitting?

- Overfitting during the estimation of residualized quantities (e.g., $\tilde{Y}, \tilde{D}$) can introduce bias into the estimation of β.
- Data-driven tuning methods like cross-validation often lead to mild overfitting, as the same data is used repeatedly.
- Cross-fitting mitigates this issue by using sample-splitting:
  - Nuisance parameters $(\ell(X), m(X))$ are estimated on one part of the data.
  - Residuals $(\tilde{Y}, \tilde{D})$ are computed on another.
- Even when using simple learners like LASSO, cross-fitting is recommended unless theoretical penalty levels (λ) are used.

Practical Implication:

- Cross-fitting ensures robust inference by preventing overfitting from contaminating estimates of the parameter of interest.
- Especially crucial when using complex machine learning methods.

Double Machine Learning (DML) Procedure

Steps:

1. **Cross-Fitting:**
   - Partition data into K folds: $\{1, ..., n\} = \cup_{k=1}^{K} I_k$.
   - For each fold k, estimate $\ell(X)$ and $m(X)$ using data outside I_k.
   - Compute cross-fitted residuals for $i \in I_k$:

$$\tilde{Y}_i = Y_i - \ell^{[-k]}(X_i), \quad \tilde{D}_i = D_i - m^{[-k]}(X_i).$$

Double Machine Learning (DML) Procedure

Steps:

1. **Cross-Fitting:**
   - Partition data into K folds: $\{1, ..., n\} = \cup_{k=1}^{K} I_k$.
   - For each fold k, estimate $\ell(X)$ and $m(X)$ using data outside I_k.
   - Compute cross-fitted residuals for $i \in I_k$:

$$\tilde{Y}_i = Y_i - \ell^{[-k]}(X_i), \quad \tilde{D}_i = D_i - m^{[-k]}(X_i).$$

2. **OLS Regression:**
   - Regress $\tilde{Y}_i$ on $\tilde{D}_i$ to estimate β:

$$\hat{\beta} = \frac{\frac{1}{n}\sum \tilde{Y}_i \tilde{D}_i}{\frac{1}{n}\sum \tilde{D}_i^2}.$$

Double Machine Learning (DML) Procedure

Steps:

1. **Cross-Fitting:**
   - Partition data into K folds: $\{1, ..., n\} = \cup_{k=1}^{K} I_k$.
   - For each fold k, estimate $\ell(X)$ and $m(X)$ using data outside I_k.
   - Compute cross-fitted residuals for $i \in I_k$:

$$\tilde{Y}_i = Y_i - \ell^{[-k]}(X_i), \quad \tilde{D}_i = D_i - m^{[-k]}(X_i).$$

2. **OLS Regression:**
   - Regress $\tilde{Y}_i$ on $\tilde{D}_i$ to estimate β:

$$\hat{\beta} = \frac{\frac{1}{n}\sum \tilde{Y}_i \tilde{D}_i}{\frac{1}{n}\sum \tilde{D}_i^2}.$$

3. **Inference:**
   - Construct standard errors using:

$$\hat{V} = \left(\frac{1}{n}\sum \tilde{D}_i^2\right)^{-1} \left(\frac{1}{n}\sum \tilde{D}_i^2 \epsilon_i^2\right) \left(\frac{1}{n}\sum \tilde{D}_i^2\right)^{-1}.$$

   - Confidence Interval:

$$\left[\hat{\beta} \pm 1.96 \cdot \sqrt{\frac{\hat{V}}{n}}\right].$$

Summary and Extensions

Key Takeaways:

- The PLM framework combines flexibility (nonlinearity via $g(X)$) with interpretability (linear effect of D).
- DML ensures valid inference on β despite high-dimensional X.
- Cross-fitting mitigates overfitting from machine learning-based nuisance parameter estimation.

Summary and Extensions

Key Takeaways:

- The PLM framework combines flexibility (nonlinearity via $g(X)$) with interpretability (linear effect of D).
- DML ensures valid inference on β despite high-dimensional X.
- Cross-fitting mitigates overfitting from machine learning-based nuisance parameter estimation.

When PLM Fails:

- DML estimates the best linear predictor (BLP) of $\tilde{Y}$ in terms of $\tilde{D}$, even if PLM does not hold.
- In this case, β captures an approximation of the effect of D on Y.

Content

Defining the Moment Condition

Key Ingredients

- DML is based on the **method-of-moments** framework, targeting a low-dimensional parameter of interest, θ_0, defined via the moment condition:

$$E[\psi(W; \theta_0, \eta_0)] = 0,$$

  where:
  - ψ: Score function.
  - W: Data vector.
  - θ_0: Parameter of interest.
  - η_0: Nuisance parameters (unknown high-dimensional functions).

Defining the Moment Condition

Key Ingredients

- DML is based on the **method-of-moments** framework, targeting a low-dimensional parameter of interest, θ_0, defined via the moment condition:

$$E[\psi(W; \theta_0, \eta_0)] = 0,$$

  where:
  - ψ: Score function.
  - W: Data vector.
  - θ_0: Parameter of interest.
  - η_0: Nuisance parameters (unknown high-dimensional functions).
- **Interpretation:** θ_0 is identified when the above equation holds.

Neyman Orthogonality

Key Concept

- A score function $\psi(W; \theta, \eta)$ satisfies **Neyman orthogonality** if:

$$\frac{\partial}{\partial \eta} E[\psi(W; \theta_0, \eta)] \Big|_{\eta = \eta_0} = 0.$$

Neyman Orthogonality

Key Concept

- A score function $\psi(W;\theta,\eta)$ satisfies **Neyman orthogonality** if:

$$\frac{\partial}{\partial\eta} E[\psi(W;\theta_0,\eta)]\bigg|_{\eta=\eta_0} = 0.$$

- **Importance:** Eliminates first-order bias from errors in the nuisance parameter estimates, $\hat{\eta}$.

Gateaux Derivative

Definition

- The **Gateaux derivative** formalizes sensitivity to small perturbations:

$$\frac{\partial}{\partial \eta} E[\psi(W;\theta,\eta)][\Delta] := \frac{\partial}{\partial t} E[\psi(W;\theta,\eta + t\Delta)]\bigg|_{t=0}.$$

Gateaux Derivative

Definition

- The **Gateaux derivative** formalizes sensitivity to small perturbations:

$$\frac{\partial}{\partial \eta} E[\psi(W;\theta,\eta)][\Delta] := \frac{\partial}{\partial t} E[\psi(W;\theta,\eta + t\Delta)]\Big|_{t=0}.$$

- **Implication:** Neyman orthogonality implies:

$$\frac{\partial}{\partial \eta} E[\psi(W;\theta_0,\eta_0)][\Delta] = 0, \quad \forall \Delta.$$

Gateaux Derivative

Definition

- The **Gateaux derivative** formalizes sensitivity to small perturbations:

$$\frac{\partial}{\partial \eta} E[\psi(W;\theta,\eta)][\Delta] := \frac{\partial}{\partial t} E[\psi(W;\theta,\eta + t\Delta)]\bigg|_{t=0}.$$

- **Implication:** Neyman orthogonality implies:

$$\frac{\partial}{\partial \eta} E[\psi(W;\theta_0,\eta_0)][\Delta] = 0, \quad \forall \Delta.$$

- **Admissible Directions:** Δ is admissible if $\eta_0 + t\Delta$ stays in the parameter space for small t.

Good Learners for Nuisance Functions

Requirements for High-Quality Learners

- Learners must approximate the true nuisance parameters η_0 well:

$$n^{1/4}\|\hat{\eta} - \eta_0\|_{L^2} \approx 0.$$

Good Learners for Nuisance Functions

Requirements for High-Quality Learners

- Learners must approximate the true nuisance parameters η_0 well:

$$n^{1/4}\|\hat{\eta} - \eta_0\|_{L^2} \approx 0.$$

- Examples of Machine Learning Methods:
  1. **LASSO:** For sparsely parameterized η_0.
  2. **Random Forests:** For tree-like structures in η_0.
  3. **Deep Neural Networks:** For η_0 approximable by sparse deep nets.
  4. **Ensemble Models:** Combining methods to leverage strengths of each.

Good Learners for Nuisance Functions

Requirements for High-Quality Learners

- Learners must approximate the true nuisance parameters η_0 well:

$$n^{1/4}\|\hat{\eta} - \eta_0\|_{L^2} \approx 0.$$

- Examples of Machine Learning Methods:
  1. **LASSO:** For sparsely parameterized η_0.
  2. **Random Forests:** For tree-like structures in η_0.
  3. **Deep Neural Networks:** For η_0 approximable by sparse deep nets.
  4. **Ensemble Models:** Combining methods to leverage strengths of each.
- Cross-validation and careful tuning are critical for robust performance.

Cross-Fitting

Why Cross-Fitting?

- Prevents **overfitting**, which occurs when nuisance parameter estimates are correlated with the same data used for inference.

Cross-Fitting

Why Cross-Fitting?

- Prevents **overfitting**, which occurs when nuisance parameter estimates are correlated with the same data used for inference.
- Mechanism:
  1. Split data into K folds.
  2. Use $K - 1$ folds to estimate nuisance parameters ($\hat{\eta}$).
  3. Use the left-out fold to compute residuals and estimate the target parameter.

Cross-Fitting

Why Cross-Fitting?

- Prevents **overfitting**, which occurs when nuisance parameter estimates are correlated with the same data used for inference.
- Mechanism:
  1. Split data into K folds.
  2. Use $K-1$ folds to estimate nuisance parameters ($\hat{\eta}$).
  3. Use the left-out fold to compute residuals and estimate the target parameter.
- **Outcome:** Avoids biases arising from overfitting complex machine learning methods.

Example 1: Partially Linear Model (PLM)

Moment Condition for PLM

$$\psi(W;\theta,\eta) = (Y - \ell(X) - \theta(D - m(X)))\,(D - m(X)).$$

- $W = (Y, D, X)$: Observable variables.
- $\eta = (\ell, m)$: Nuisance parameters.
    - $\ell(X) = E[Y|X]$, $m(X) = E[D|X]$.

Example 1: Partially Linear Model (PLM)

Moment Condition for PLM

$$\psi(W;\theta,\eta) = (Y - \ell(X) - \theta(D - m(X)))\,(D - m(X)).$$

- $W = (Y, D, X)$: Observable variables.
- $\eta = (\ell, m)$: Nuisance parameters.
  - $\ell(X) = E[Y|X]$, $m(X) = E[D|X]$.

Neyman Orthogonality

- Using elementary calculations:

$$\frac{\partial}{\partial\eta} E[\psi(W;\theta,\eta)]\Big|_{\eta=\eta_0} = 0.$$

Example 1: Partially Linear Model (PLM)

Moment Condition for PLM

$$\psi(W;\theta,\eta) = (Y - \ell(X) - \theta(D - m(X)))\,(D - m(X)).$$

- $W = (Y, D, X)$: Observable variables.
- $\eta = (\ell, m)$: Nuisance parameters.
  - $\ell(X) = E[Y|X]$, $m(X) = E[D|X]$.

Neyman Orthogonality

- Using elementary calculations:

$$\frac{\partial}{\partial\eta} E[\psi(W;\theta,\eta)]\Big|_{\eta=\eta_0} = 0.$$

Interpretation: $\psi(W;\theta,\eta)$ generalizes residualization in linear models, enabling robust inference.

Example 2: Doubly Robust IPW

Doubly Robust IPW Estimator

$$\psi(W;\theta,\eta) = (g(1,X) - g(0,X)) + H(D,X)(Y - g(D,X)) - \theta,$$

where:

$$H(D,X) = \frac{D}{m(X)} - \frac{(1-D)}{1-m(X)}.$$

Example 2: Doubly Robust IPW

Doubly Robust IPW Estimator

$$\psi(W;\theta,\eta) = (g(1,X) - g(0,X)) + H(D,X)(Y - g(D,X)) - \theta,$$

where:

$$H(D,X) = \frac{D}{m(X)} - \frac{(1-D)}{1-m(X)}.$$

- $g(D,X) = E[Y|D,X]$, $m(X) = P[D=1|X]$.
- nuisance parameters: $\eta = (g,m)$.
- Neyman Orthogonality:

$$\frac{\partial}{\partial \eta} E[\psi(W;\theta,\eta)] = 0.$$

Generic DML Algorithm

1. **Input:** Data $\{W_i\}_{i=1}^n$, Neyman orthogonal score $\psi(W;\theta,\eta)$, and machine learning methods for η.

Generic DML Algorithm

1. **Input:** Data $\{W_i\}_{i=1}^n$, Neyman orthogonal score $\psi(W;\theta,\eta)$, and machine learning methods for η.
2. **Cross-Fitting:**
   - Split data into K folds.
   - Train $\hat{\eta}[k]$ on $K-1$ folds and compute expectations on the left-out fold.

Generic DML Algorithm

1. **Input:** Data $\{W_i\}_{i=1}^n$, Neyman orthogonal score $\psi(W;\theta,\eta)$, and machine learning methods for η.
2. **Cross-Fitting:**
   - Split data into K folds.
   - Train $\hat{\eta}[k]$ on $K-1$ folds and compute expectations on the left-out fold.
3. **Moment Estimation:**

$$\hat{M}(\theta,\hat{\eta}) = \frac{1}{n}\sum_{i=1}^{n}\psi(W_i;\theta,\hat{\eta}[k(i)]).$$

4. **Variance and Confidence Intervals:**
   - Estimate the asymptotic variance of $\hat{\theta}$ as:

$$\hat{V} = \frac{1}{n}\sum_{i=1}^{n}[\hat{\phi}(W_i)\hat{\phi}(W_i)'] - \frac{1}{n}\sum_{i=1}^{n}[\hat{\phi}(W_i)]\frac{1}{n}\sum_{i=1}^{n}[\hat{\phi}(W_i)]', \quad \hat{\phi}(W_i) = -\hat{J}_0^{-1}\psi(W_i;\hat{\theta},\hat{\eta}[k($$

Generic DML Algorithm

1. **Input:** Data $\{W_i\}_{i=1}^n$, Neyman orthogonal score $\psi(W;\theta,\eta)$, and machine learning methods for η.
2. **Cross-Fitting:**
   - Split data into K folds.
   - Train $\hat{\eta}[k]$ on $K-1$ folds and compute expectations on the left-out fold.
3. **Moment Estimation:**

$$\hat{M}(\theta,\hat{\eta}) = \frac{1}{n}\sum_{i=1}^{n}\psi(W_i;\theta,\hat{\eta}[k(i)]).$$

4. **Variance and Confidence Intervals:**
   - Estimate the asymptotic variance of $\hat{\theta}$ as:

$$\hat{V} = \frac{1}{n}\sum_{i=1}^{n}[\hat{\phi}(W_i)\hat{\phi}(W_i)'] - \frac{1}{n}\sum_{i=1}^{n}[\hat{\phi}(W_i)]\frac{1}{n}\sum_{i=1}^{n}[\hat{\phi}(W_i)]', \quad \hat{\phi}(W_i) = -\hat{J}_0^{-1}\psi(W_i;\hat{\theta},\hat{\eta}[k($$

   - Confidence interval:

$$[\hat{\theta} \pm z_{1-\alpha/2}\sqrt{\hat{V}/n}].$$