# Adversarial Monte Carlo Meta-Learning of Optimal Prediction Procedures

**Alex Luedtke (♙)**

This presentation builds on collaborations with:

**Marco Carone (♗)**
**Noah Simon (♗)**
**Oleg Sofrygin (♘)**
**Hongxiang Qiu (♖)**
**Incheoul Chung (♙)**

♙ : Department of Statistics, University of Washington
♗ : Department of Biostatistics, University of Washington
♘ : Division of Research, Kaiser Permanente Northern California
♖ : Department of Statistics and Data Science, University of Pennsylvania

# Setting

Let $(X, A, Y) \sim P$, where

- $X$ is a feature vector with support in $\mathbb{R}^q$
- $A$ is a subsequent binary treatment
- $Y$ is a continuous outcome

**Setting 1:** Observe an iid dataset $\boldsymbol{D} := (X_i, Y_i)_{i=1}^n$ and want to develop an estimator of a **regression function**:

$$x \mapsto E_P[Y|X = x]\,.$$

**Setting 2:** Observe an iid dataset $\boldsymbol{D} := (X_i, A_i, Y_i)_{i=1}^n$ and want to develop an estimator of a **conditional average treatment effect (CATE) function**:

$$x \mapsto E_P[Y|A = 1, X = x] - E_P[Y|A = 0, X = x]\,.$$

# Setting

Let $(X, A, Y) \sim P$, where

- $X$ is a feature vector with support in $\mathbb{R}^q$
- $A$ is a subsequent binary treatment
- $Y$ is a continuous outcome

**Setting 1:** Observe an iid dataset $\boldsymbol{D} := (X_i, Y_i)_{i=1}^n$ and want to develop an estimator of a **regression function**:

$$x \mapsto \underbrace{E_P[Y|X = x]}_{\theta_P(x)}.$$

**Setting 2:** Observe an iid dataset $\boldsymbol{D} := (X_i, A_i, Y_i)_{i=1}^n$ and want to develop an estimator of a **conditional average treatment effect (CATE) function**:

$$x \mapsto \underbrace{E_P[Y|A = 1, X = x] - E_P[Y|A = 0, X = x]}_{\Delta_P(x)}.$$

# Estimators

Throughout, I'll use "estimator" to refer to any map $T$ from a dataset $\boldsymbol{D}$ to a function mapping from features in $\mathbb{R}^q$ to $\mathbb{R}$.

The set of allowable estimators will be denoted by $\mathcal{T}$.

# Mean squared error risk for estimating the regression function

For a given $P$, performance of an estimator $T$ for estimating the regression function will be adjudicated via its standardized mean-squared error:

$$R(T, P) = E_P \left[ \int \frac{[T(\boldsymbol{D})(x) - \theta_P(x)]^2}{\sigma_P^2} dP(x) \right],$$

where $\sigma_P^2 := \mathrm{Var}_P(Y - E_P[Y|X])$.

- The standardization factor $\sigma_P^2$ is the **semiparametric efficiency bound for estimating** $E_P[\theta_P(X)]$ in a model where the marginal distribution of $X$ is known.

# Mean squared error risk for estimating the regression function

For estimating the CATE, we use the following risk function:

$$R(T, P) = E_P \left[ \int \frac{[T(\boldsymbol{D})(x) - \Delta_P(x)]^2}{\sigma_P^2} dP(x) \right],$$

where $\sigma_P^2 := \mathrm{Var}_P \left( \frac{2A-1}{P(A|X)} \{ Y - E_P[Y|A, X] \} \right)$.

- The standardization factor $\nu_P^2$ is the **semiparametric efficiency bound for estimating** $E_P[\Delta_P(X)]$ in a model where the marginal distribution of $X$ is known.

# Bayes and maximal risks

Because the true underlying distribution $P$ is not known, the risk $R(T, P)$ is not known either.

Instead, the performance of an estimator can be quantified via its Bayes risk relative to the prior $\Pi$:

$$r(T, \Pi) := \int R(T, P)\Pi(dP)$$

or its maximal risk over the statistical model $\mathcal{P}$:

$$\sup_{P \in \mathcal{P}} R(T, P).$$

# Γ-maximal risk

I'll consider a compromise between the Bayes and maximal risks.

For a collection of priors Γ, I'll use the **Γ-maximal risk** (Berger, 1985) :

$$\sup_{\Pi \in \Gamma} r(T, \Pi).$$

- When Γ is a singleton, the Γ-maximal risk is a Bayes risk.
- When Γ is unrestricted, the Γ-maximal risk is the maximal risk.

# Γ-minimax estimators

An estimator $T^\star$ is called **Γ-minimax** if

$$\sup_{\Pi \in \Gamma} r(T^\star, \Pi) = \inf_{T \in \mathcal{T}} \sup_{\Pi \in \Gamma} r(T, \Pi).$$

**Problem:** A closed-form expression for a Γ-minimax estimator is rarely known.

**Solution:** Use **numerical methods** to construct a Γ-minimax estimator.

# Existing works

Here are some key existing works for numerically constructing $\Gamma$-minimax estimators:

- **When $\Gamma$ is unrestricted:** Nelson (1966) and Kempthorne (1987)
- **When $\Gamma$ is to a finite mixture of $k$ fixed priors:** Chamberlain (2000)

All above listed approaches assume that it is easy to derive the Bayes estimator for a given prior, which may not be true in practice.

- **When $\Gamma$ is a singleton:** Hochreiter et al. (2001), Finn et al. (2017), and Garnelo et al. (2018)

# An iterative strategy for constructing a Γ-minimax estimator

Under conditions, Γ-minimax decision problems can be reformulated as Bayesian decision problems under a **least favorable prior** (LFP):

$$\underbrace{\max_{\Pi}\min_{T} r(T,\Pi)}_{\text{Bayes risk under LFP}} \quad = \quad \underbrace{\min_{T}\max_{\Pi} r(T,\Pi)}_{\Gamma\text{-minimax risk}}.$$

This suggests the following iterative learning scheme:

**Two agents compete against each other as follows:**

Game #1

The Statistician selects an estimator $T$.

# An iterative strategy for constructing a Γ-minimax estimator

Under conditions, Γ-minimax decision problems can be reformulated as Bayesian decision problems under a **least favorable prior** (LFP):

$$\underbrace{\max_{\Pi}\min_{T} r(T,\Pi)}_{\text{Bayes risk under LFP}} \quad = \quad \underbrace{\min_{T}\max_{\Pi} r(T,\Pi)}_{\Gamma\text{-minimax risk}}.$$

This suggests the following iterative learning scheme:

**Two agents compete against each other as follows:**

Game #1

Nature selects a prior Π.
A distribution $P$ is drawn from Π, a data set is drawn from $P$, and the performance of $T$ is evaluated.

# An iterative strategy for constructing a Γ-minimax estimator

Under conditions, Γ-minimax decision problems can be reformulated as Bayesian decision problems under a **least favorable prior** (LFP):

$$\underbrace{\max_{\Pi}\min_{T} r(T,\Pi)}_{\text{Bayes risk under LFP}} \quad = \quad \underbrace{\min_{T}\max_{\Pi} r(T,\Pi)}_{\Gamma\text{-minimax risk}}.$$

This suggests the following iterative learning scheme:

**Two agents compete against each other as follows:**

Game #2

Having learned from previous game, the Statistician selects an estimator $T$ whose Bayes risk is lower for Nature's earlier prior.

# An iterative strategy for constructing a Γ-minimax estimator

Under conditions, Γ-minimax decision problems can be reformulated as Bayesian decision problems under a **least favorable prior** (LFP):

$$\underbrace{\max_{\Pi}\min_{T} r(T,\Pi)}_{\text{Bayes risk under LFP}} \quad = \quad \underbrace{\min_{T}\max_{\Pi} r(T,\Pi)}_{\Gamma\text{-minimax risk}}.$$

This suggests the following iterative learning scheme:

**Two agents compete against each other as follows:**

Game #2

Having learned from previous game, Nature seeks to select a less favorable prior $\Pi$ for the Statistician's earlier estimator.

# An iterative strategy for constructing a Γ-minimax estimator

Under conditions, Γ-minimax decision problems can be reformulated as Bayesian decision problems under a **least favorable prior** (LFP):

$$\underbrace{\max_{\Pi}\min_{T} r(T,\Pi)}_{\text{Bayes risk under LFP}} \quad = \quad \underbrace{\min_{T}\max_{\Pi} r(T,\Pi)}_{\Gamma\text{-minimax risk}}.$$

This suggests the following iterative learning scheme:

**Two agents compete against each other as follows:**

Game #2

A distribution $P$ is drawn from $\Pi$, a data set is drawn from $P$, and the performance of $T$ is evaluated.

# An iterative strategy for constructing a Γ-minimax estimator

Under conditions, Γ-minimax decision problems can be reformulated as Bayesian decision problems under a **least favorable prior** (LFP):

$$\underbrace{\max_{\Pi} \min_{T} r(T, \Pi)}_{\text{Bayes risk under LFP}} \quad = \quad \underbrace{\min_{T} \max_{\Pi} r(T, \Pi)}_{\Gamma\text{-minimax risk}} .$$

This suggests the following iterative learning scheme:

**Two agents compete against each other as follows:**

...

# An iterative strategy for constructing a Γ-minimax estimator

Under conditions, Γ-minimax decision problems can be reformulated as Bayesian decision problems under a **least favorable prior** (LFP):

$$\underbrace{\max_{\Pi}\min_{T} r(T,\Pi)}_{\text{Bayes risk under LFP}} \quad = \quad \underbrace{\min_{T}\max_{\Pi} r(T,\Pi)}_{\Gamma\text{-minimax risk}}.$$

This suggests the following iterative learning scheme:

**Two agents compete against each other as follows:**

Many games later...

**Desired result:** Both play optimally.
In particular, the Statistician selects a Γ-minimax estimator.

# Parameterizing the Statistician's strategy

In our experiments, we have found it effective to **let the class $\mathcal{T}$ of estimators be a neural network class**.

- I'll describe a particularly interesting neural network class later in this talk.

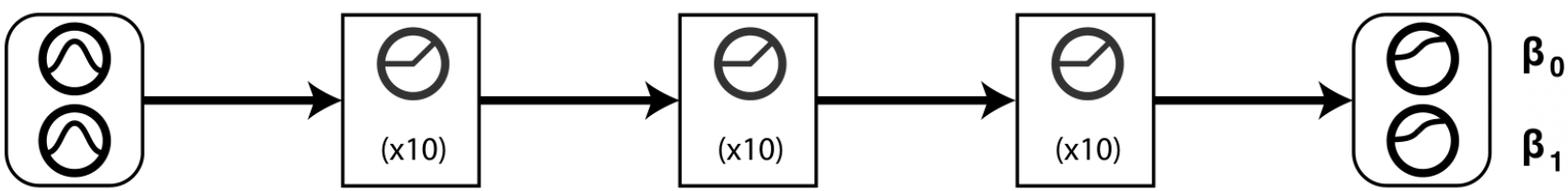
# Parameterizing Nature's strategy

When implementing our iterative scheme, we must specify a form for the prior distribution.

At each iteration, we simulate a batch of distributions from the current prior.

Consequently, our procedure is easiest to implement when the prior is easy to sample from.

**One way of achieving this:** At each step $k$, parameterize the prior as a **generator network** $G_{g(k)}$, indexed by $g(k) \in \mathbb{R}^w$, that takes as input a source of randomness $Z \sim P_Z$ and outputs a distribution $G_{g(k)}(Z)$.



# Pseudocode for iterative scheme

For each $P$, we require access to a generator $H_P$ that takes as input a source of randomness $U \sim P_U$ and outputs data $(X, Y)$ that has distribution $P$.

**initialize** parameters $t(1)$ and $g(1)$ for the Statistician's network and Nature's network.

**for** $k = 1$ to $K - 1$ **do**

  Sample $Z \sim P_Z$ and let $P_{g(k)} = G_{g(k)}(Z)$. ▷ Draw $P_{g(k)}$ from current prior.

  **for** $j = 0$ to $n$ **do** ▷ Draw data from $P_{g(k)}$.

    Sample $U_j \sim P_U$ and let $(X_{g(k),j}, Y_{g(k),j}) = H_{P_{g(k)}}(U_j)$.

  **end for**

  Let $\boldsymbol{D}_{g(k)} = (X_{g(k),j}, Y_{g(k),j})_{j=1}^{n}$. ▷ Define observed dataset.

  Update the Statistician's strategy:

$$t(k+1) = t(k) - \eta_k \nabla_{t(k)} L(T_{t(k)}(\boldsymbol{D}_{g(k)})(X_{g(k),0}), P_{g(k)}).$$

  Update Nature's strategy:

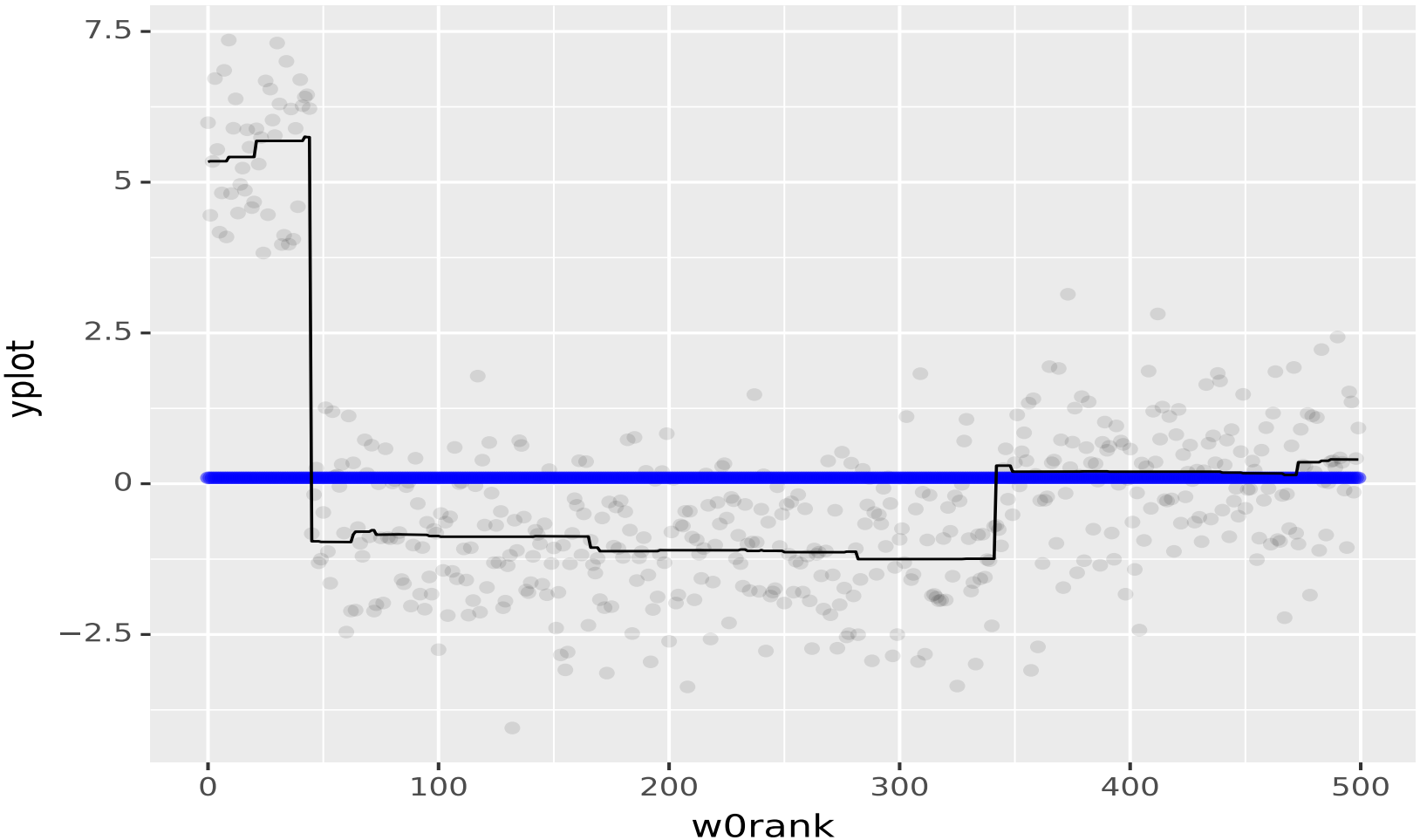
$$g(k+1) = g(k) + \delta_k \nabla_{g(k)} L\Big(T_{t(k)}\left(\boldsymbol{D}_{g(k)}\right)\left(X_{g(k),0}\right), P_{g(k)}\Big).$$

**end for**

**return** the estimator $T_{t(K)}$.

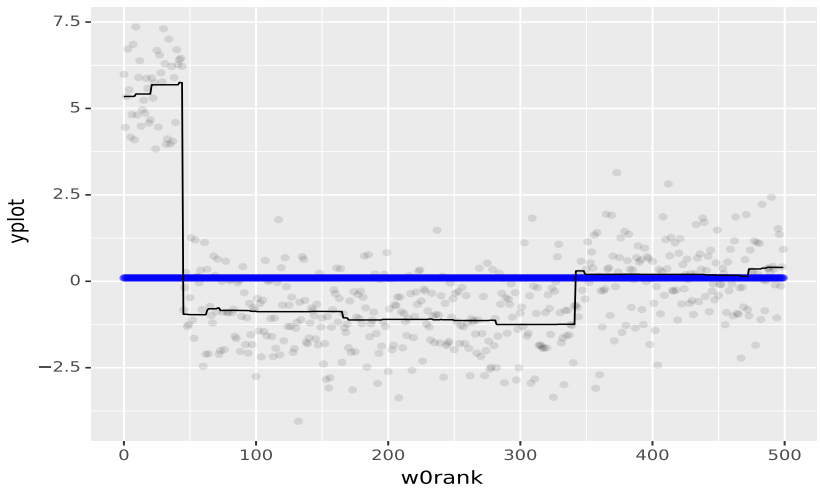
# Improvements in performance of estimator constructed with AMC over time when trained under fused lasso additive regression model ($n = 500$)

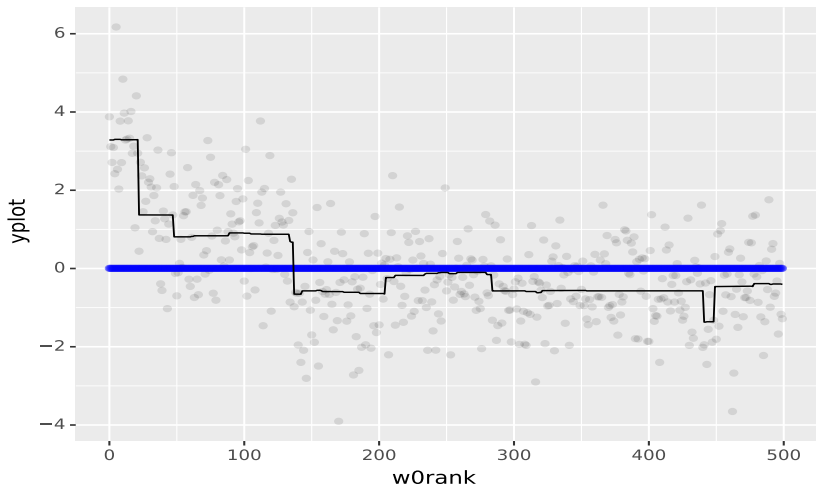
Our estimator essentially returned a constant upon initialization:

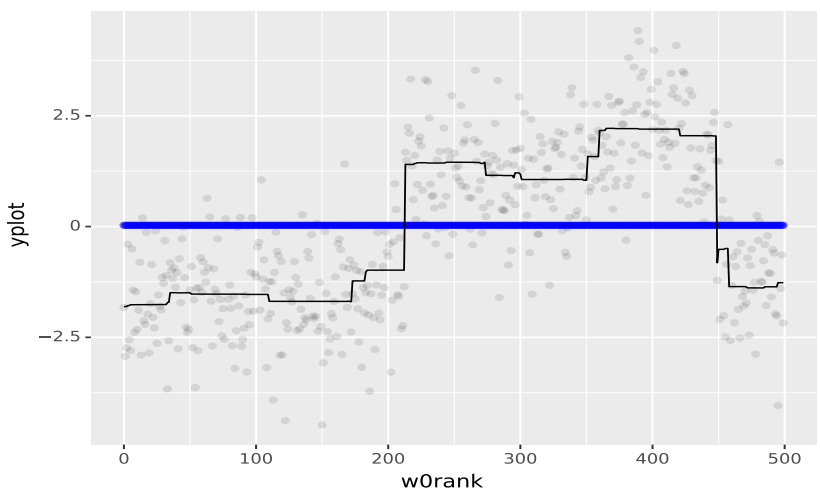


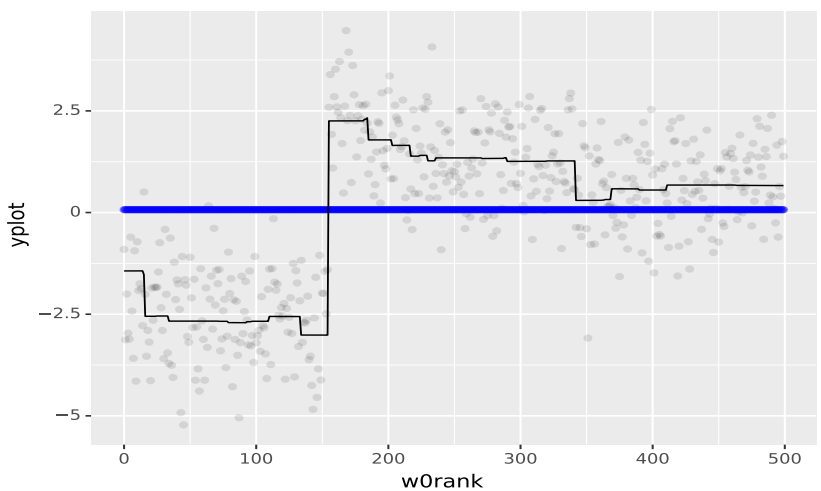
# Improvements in performance of estimator constructed with AMC over time when trained under fused lasso additive regression model ($n = 500$)

Our estimator essentially returned a constant upon initialization:



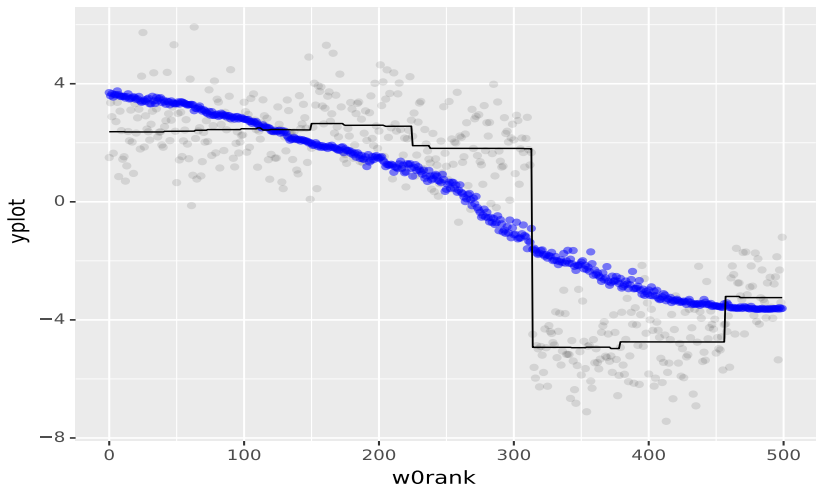



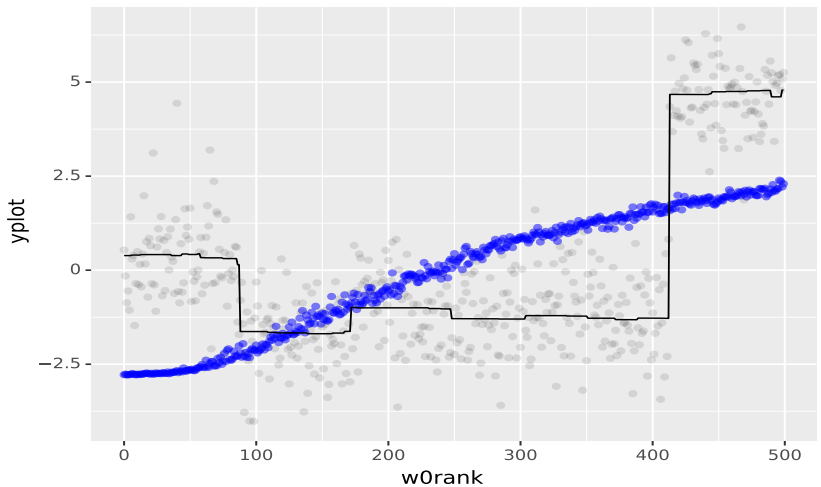



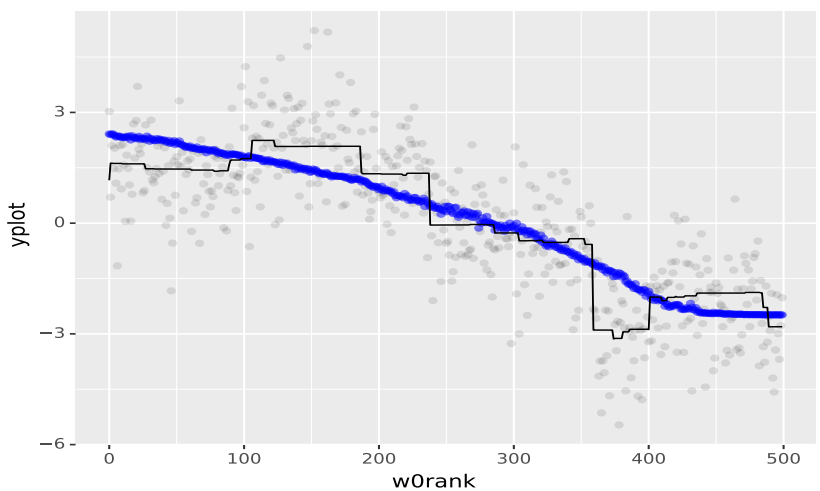



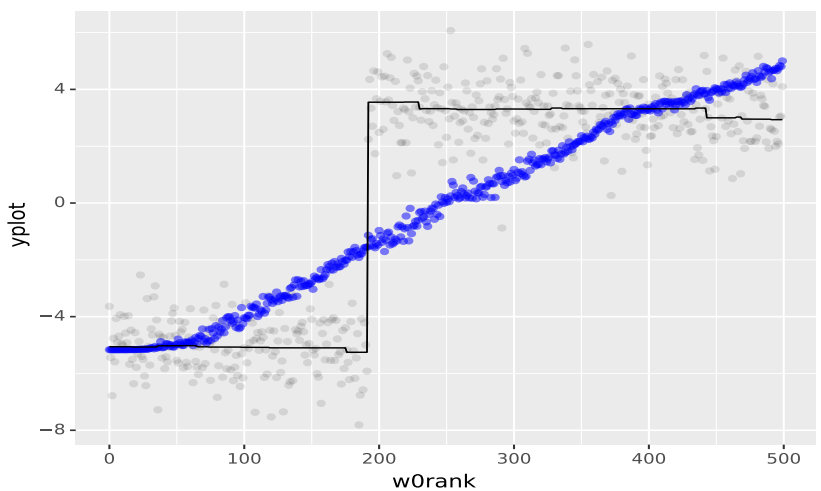
# Improvements in performance of estimator constructed with AMC over time when trained under fused lasso additive regression model ($n = 500$)

After 2 hours, it learned to capture the linear trend:



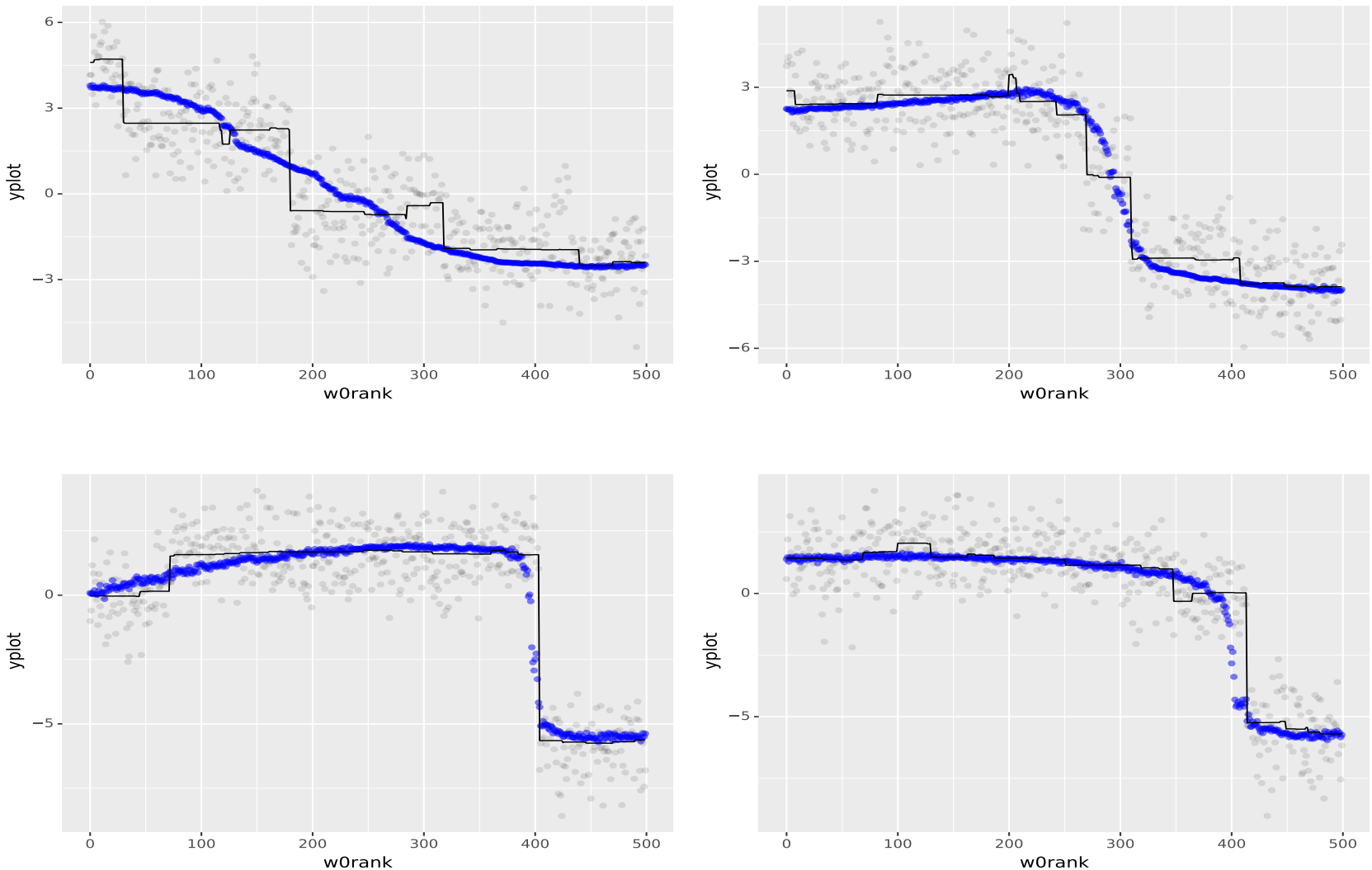



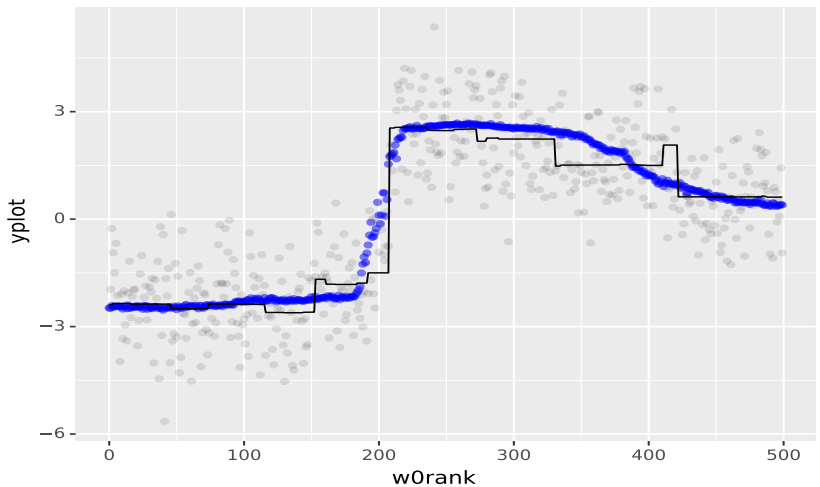
# Improvements in performance of estimator constructed with AMC over time when trained under fused lasso additive regression model ($n = 500$)

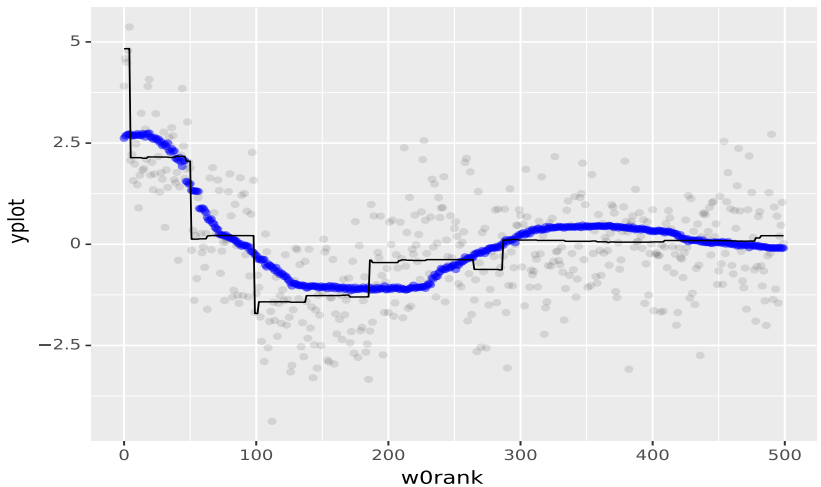
After 6 hours, it began to learn nonlinear trends:

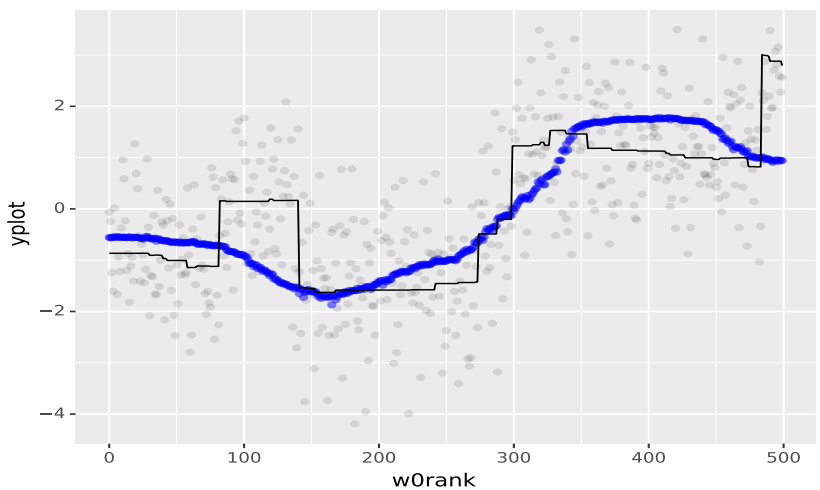


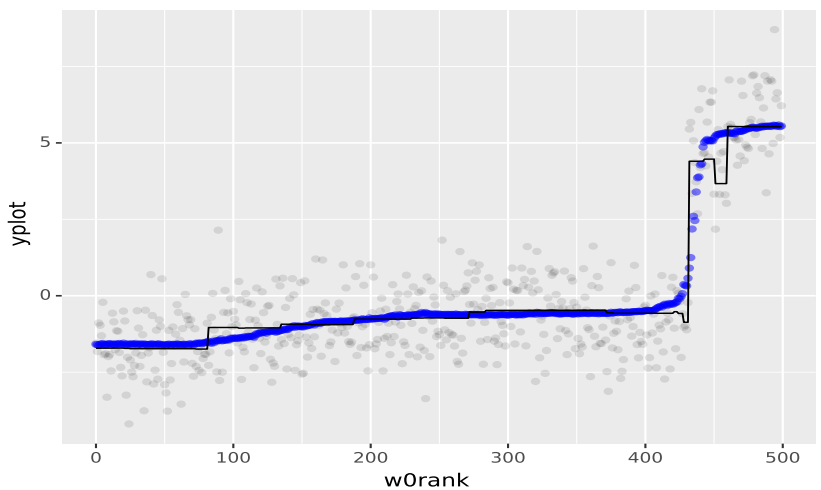
# Improvements in performance of estimator constructed with AMC over time when trained under fused lasso additive regression model ($n = 500$)

It continued improving over the next few hours. At 15 hours:



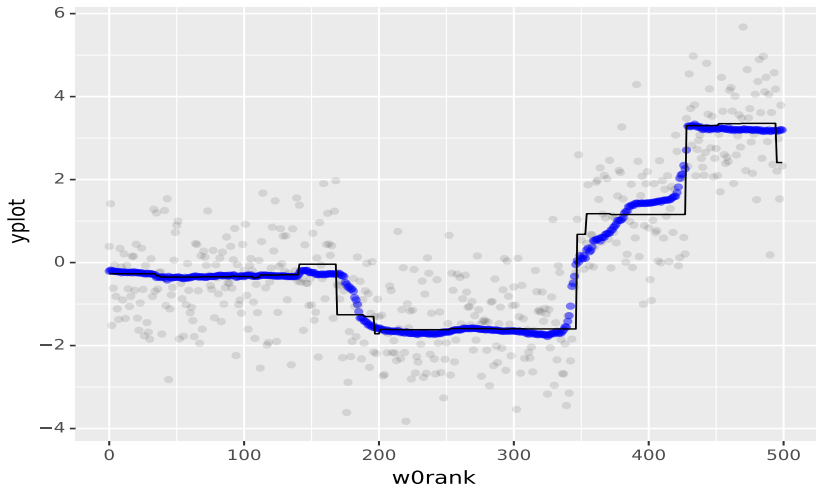



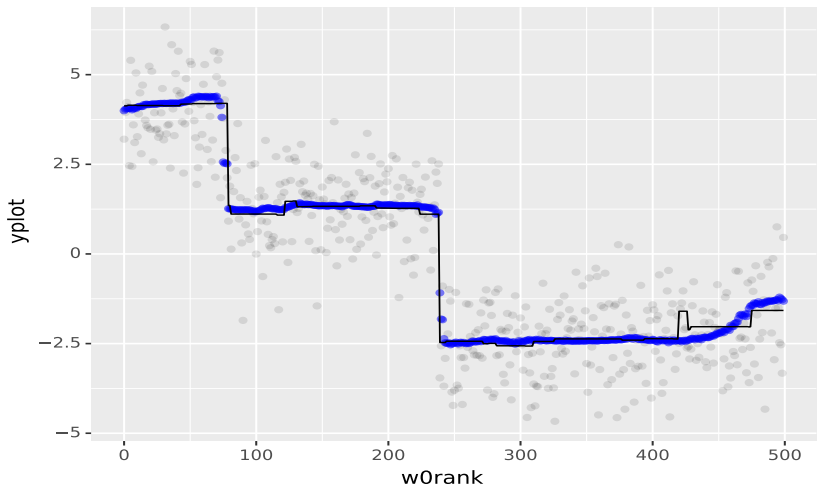



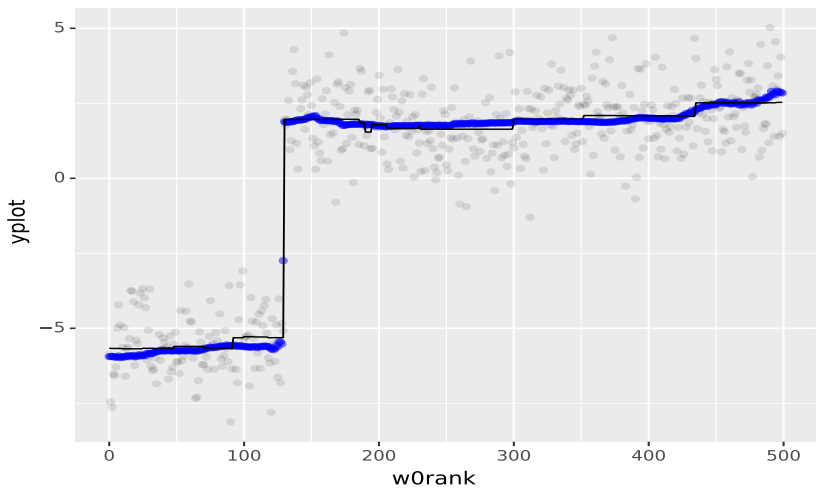



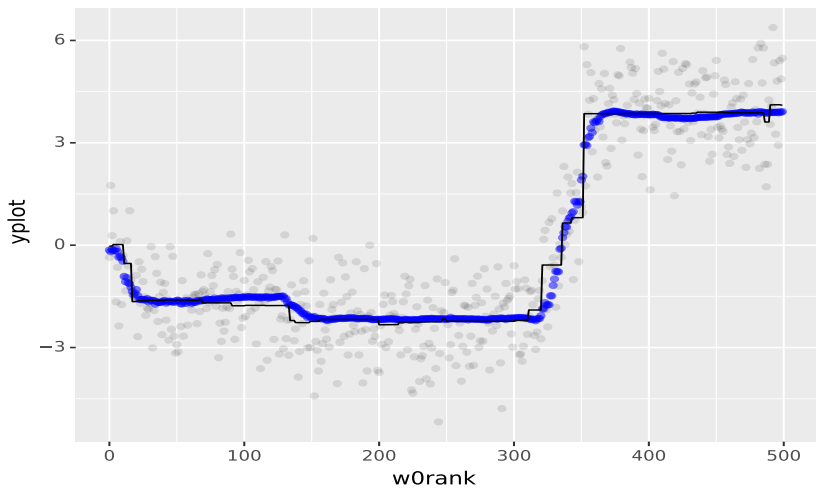
# Improvements in performance of estimator constructed with AMC over time when trained under fused lasso additive regression model ($n = 500$)

After several more days, our predictions look as follows:



# Questions?

# Overview of this section

Optimizing over a rich class of estimators $\mathcal{T}$ numerically is challenging.

If $\mathcal{T}_1 \subseteq \mathcal{T}$ contains a $\Gamma$-minimax estimator $T^\star$, that is, an estimator for which

$$\sup_{\Pi \in \Gamma} r(T^\star, \Pi) = \inf_{T \in \mathcal{T}} \sup_{\Pi \in \Gamma} r(T, \Pi),$$

then nothing should be lost by restricting the optimization to $\mathcal{T}_1$.

We've derived a Hunt-Stein-type theorem that provides the form of such a class $\mathcal{T}_1$ in many problems.

# Notation

On the upcoming slides, I'll write an observed dataset in a regression problem as $\boldsymbol{d} = (\boldsymbol{x}, \boldsymbol{y})$.

- $\boldsymbol{x}$ is an $n \times q$ design matrix.
- $\boldsymbol{y}$ is an $n \times 1$ vector of outcomes.

# Suffices to focus on equivariant estimators in many regression problems

**Theorem:** Under conditions, there is a $\Gamma$-minimax estimator $T^\star$ that satisfies the following for all datasets $(\boldsymbol{x}, \boldsymbol{y})$ and evaluation points $x_0$:

1. **Invariance to permutations of observations:** For all $B \in \mathcal{B}_n$,

$$T^\star(B\boldsymbol{x}, B\boldsymbol{y})(x_0) = T^\star(\boldsymbol{x}, \boldsymbol{y})(x_0),$$

   where $\mathcal{B}_s$ denotes the collection of all $s \times s$ permutation matrices.

2. **Invariance to permutations of features:** For all $B \in \mathcal{B}_q$,

$$T^\star(\boldsymbol{x}B, \boldsymbol{y})(x_0 B) = T^\star(\boldsymbol{x}, \boldsymbol{y})(x_0).$$

3. **Invariance to increasing affine transformations of features:** The output of $T^\star$ remains unchanged if, for each feature $j \in \{1, 2, \ldots, q\}$, an affine transformation is applied to that feature.

4. **Equivariance to increasing affine transformations of outcomes:** For all $b > 0$ and $c \in \mathbb{R}$,

$$T^\star(\boldsymbol{x}, b\boldsymbol{y} + c)(x_0) = bT^\star(\boldsymbol{x}, \boldsymbol{y})(x_0) + c.$$

# Example of invariance to permutations of observations

The estimator $T^\star$ outputs the **same prediction** on

| $x$ | | | $y$ |
|---|---|---|---|
| 2.5 | 3.1 | 4.8 | 7.8 |
| 5.8 | 1.2 | 0.9 | 3.6 |
| 4.9 | 6.6 | 2.8 | 4.6 |
| 1.9 | 0.7 | 1.9 | 9.0 |

| $x_0$ | | |
|---|---|---|
| 4.3 | 7.5 | 1.7 |

and

| $x$ | | | $y$ |
|---|---|---|---|
| 5.8 | 1.2 | 0.9 | 3.6 |
| 2.5 | 3.1 | 4.8 | 7.8 |
| 1.9 | 0.7 | 1.9 | 9.0 |
| 4.9 | 6.6 | 2.8 | 4.6 |

| $x_0$ | | |
|---|---|---|
| 4.3 | 7.5 | 1.7 |

# Example of invariance to permutations of features

The estimator $T^\star$ outputs the **same prediction** on

| $\boldsymbol{x}$ | | | $\boldsymbol{y}$ |
|---|---|---|---|
| 2.5 | 3.1 | 4.8 | 7.8 |
| 5.8 | 1.2 | 1.1 | 3.6 |
| 4.9 | 6.6 | 2.8 | 4.6 |
| 1.9 | 0.7 | 1.9 | 9.0 |

$x_0$

4.3 7.5 1.7

and

| $\boldsymbol{x}$ | | | $\boldsymbol{y}$ |
|---|---|---|---|
| 3.1 | 4.8 | 2.5 | 7.8 |
| 1.2 | 1.1 | 5.8 | 3.6 |
| 6.6 | 2.8 | 4.9 | 4.6 |
| 0.7 | 1.9 | 1.9 | 9.0 |

$x_0$

7.5 1.7 4.3

# Similar results hold for CATE estimation

A similar theorem can be derived for CATE estimation.

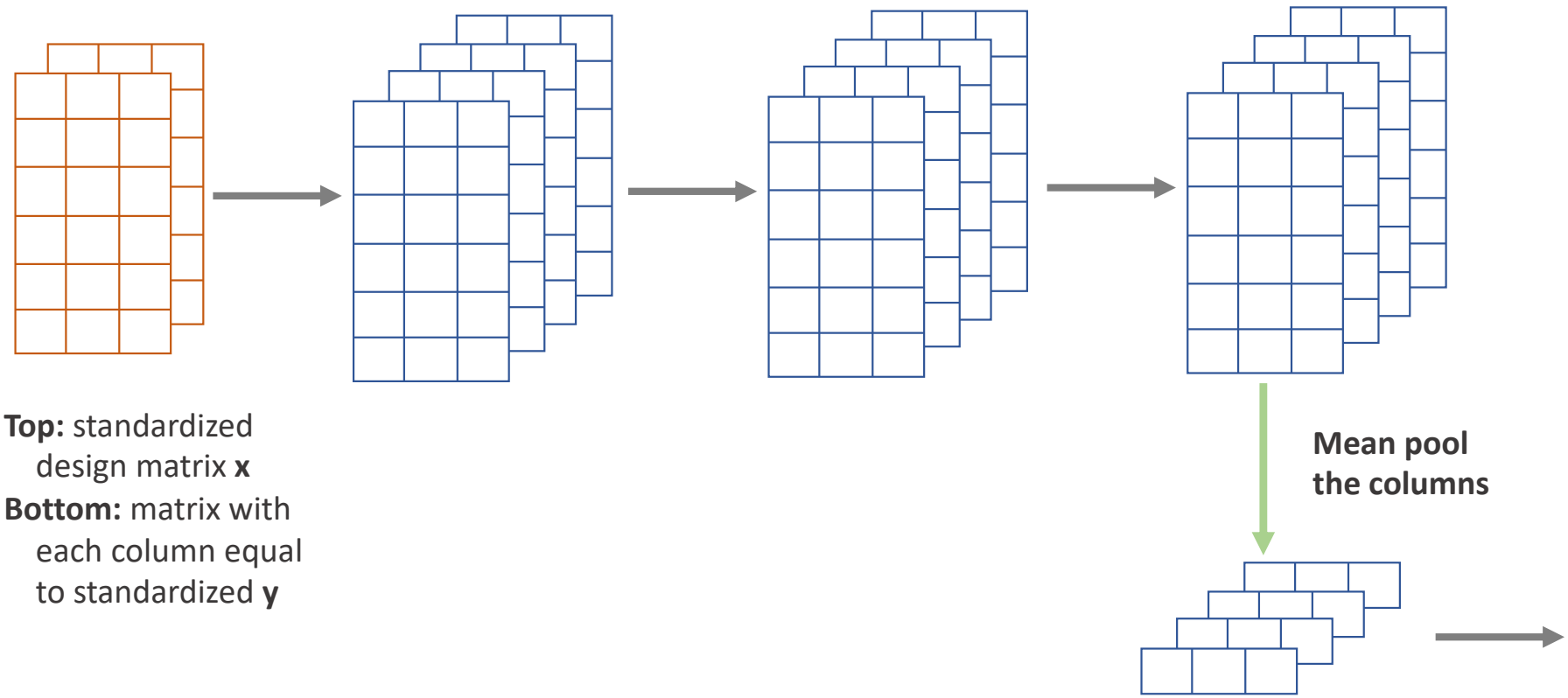
# Summary of this section

In Luedtke et al. (2021), we introduced a neural network architecture to parameterize the estimator $T$.

This estimator class satisfies all of the equivariance properties suggested by our Hunt-Stein-type theorem.

Each estimator in this class sequentially transforms the data via the composition of 4 functions, which we refer to as modules.

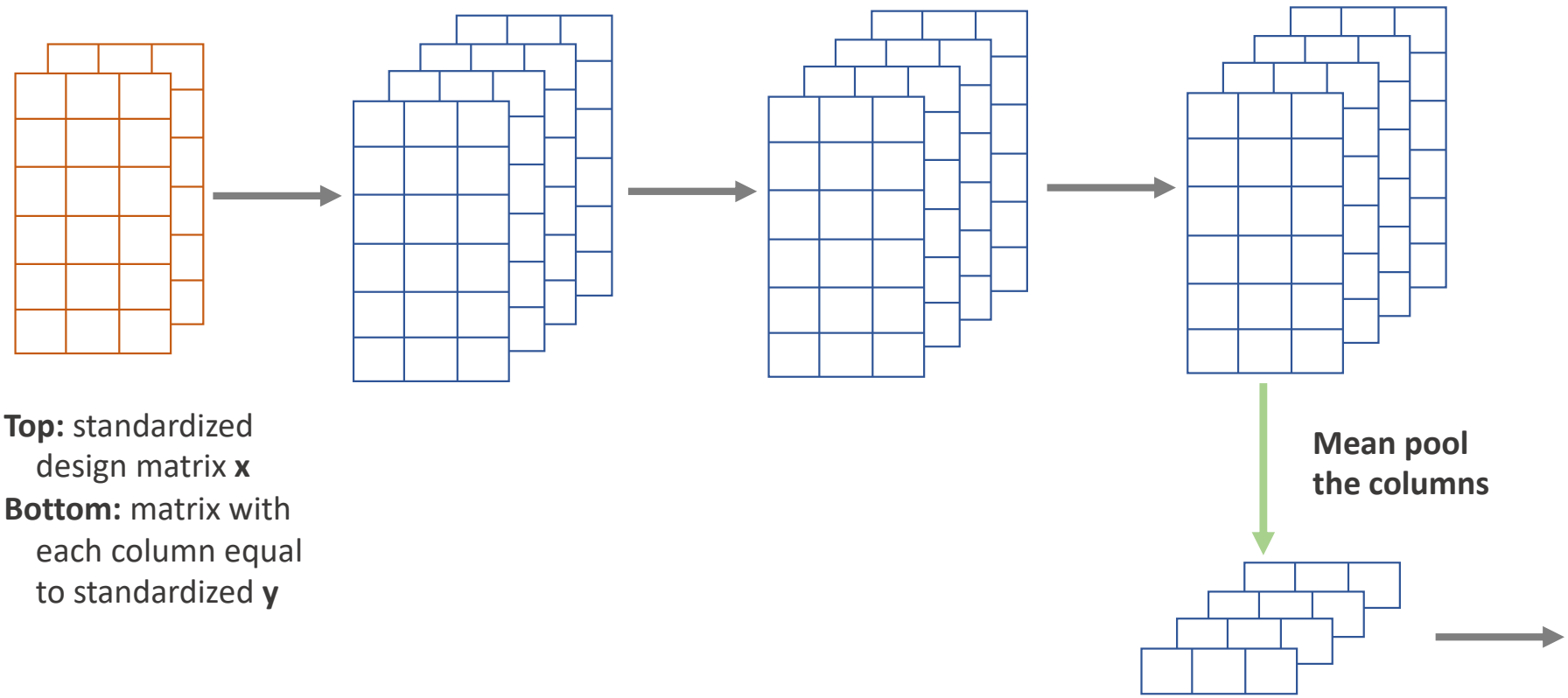
# Summary of architecture: module 1 (Hartford et al., 2018)




**Top:** standardized design matrix **x**

**Bottom:** matrix with each column equal to standardized **y**

# Summary of architecture: module 1 (Hartford et al., 2018)

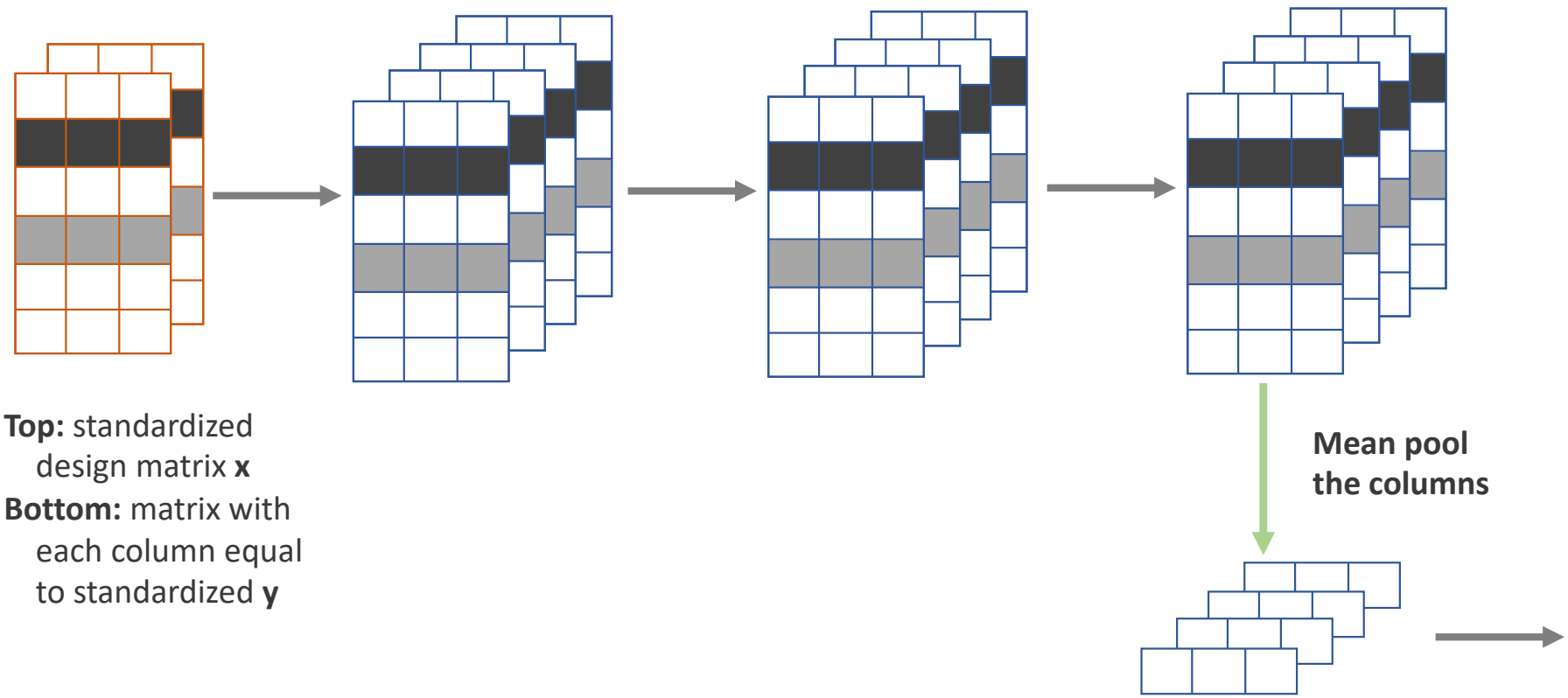



**Top:** standardized design matrix **x**

**Bottom:** matrix with each column equal to standardized **y**

**This module is invariant to permutations of the rows of its input.**

# Summary of architecture: module 1 (Hartford et al., 2018)

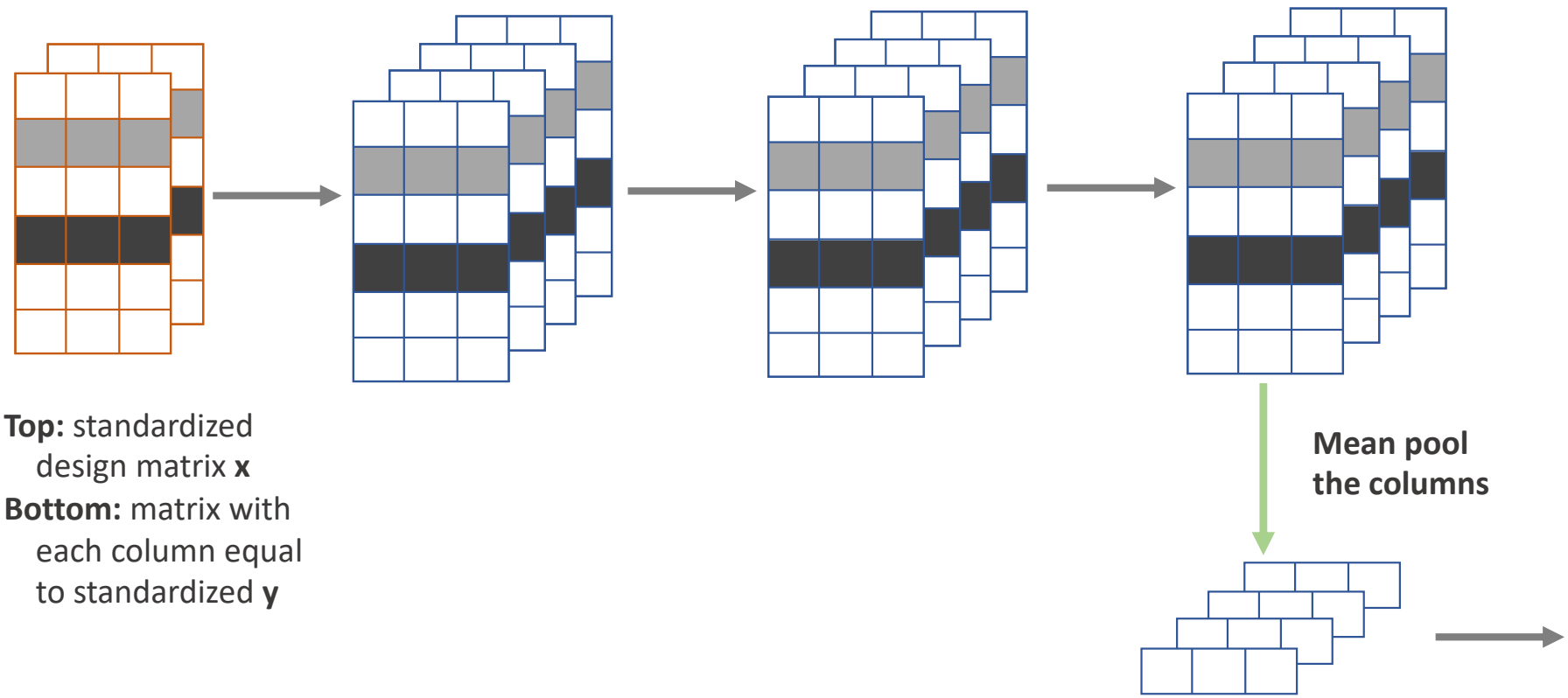



**Top:** standardized design matrix **x**

**Bottom:** matrix with each column equal to standardized **y**

**This module is invariant to permutations of the rows of its input.**

# Summary of architecture: module 1 (Hartford et al., 2018)

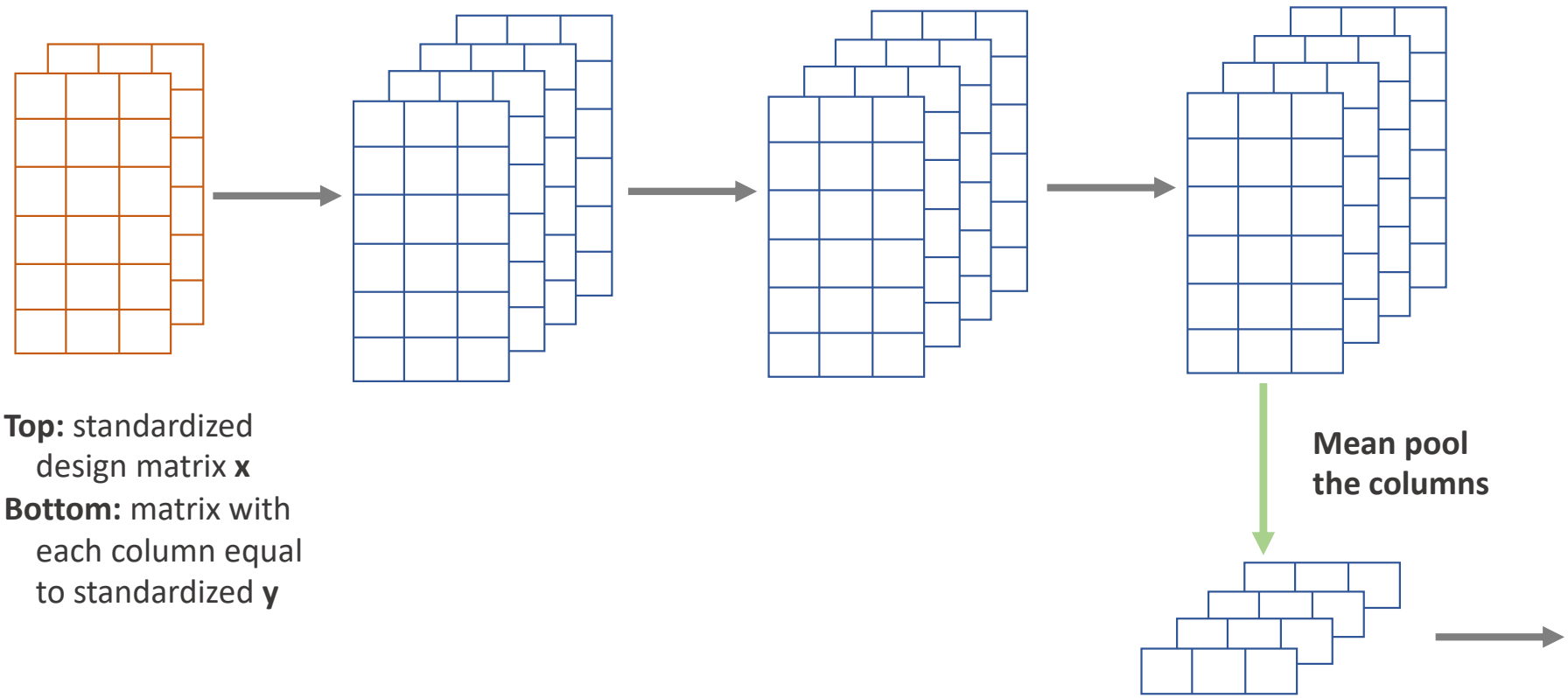



**Top:** standardized design matrix **x**

**Bottom:** matrix with each column equal to standardized **y**

**The module is invariant to permutations of the rows of its input.**

# Summary of architecture: module 1 (Hartford et al., 2018)

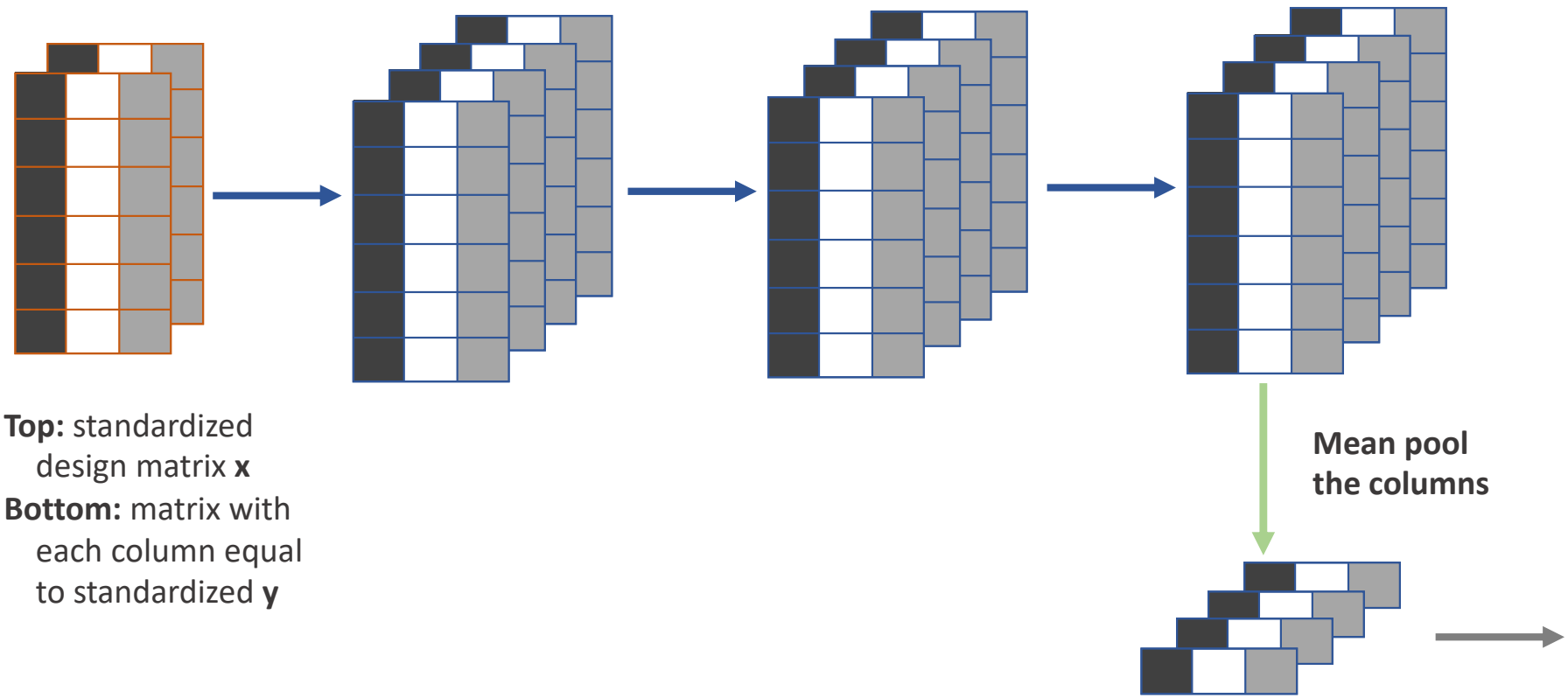



**Top:** standardized design matrix **x**

**Bottom:** matrix with each column equal to standardized **y**

**And equivariant to permutations of the columns.**

# Summary of architecture: module 1 (Hartford et al., 2018)

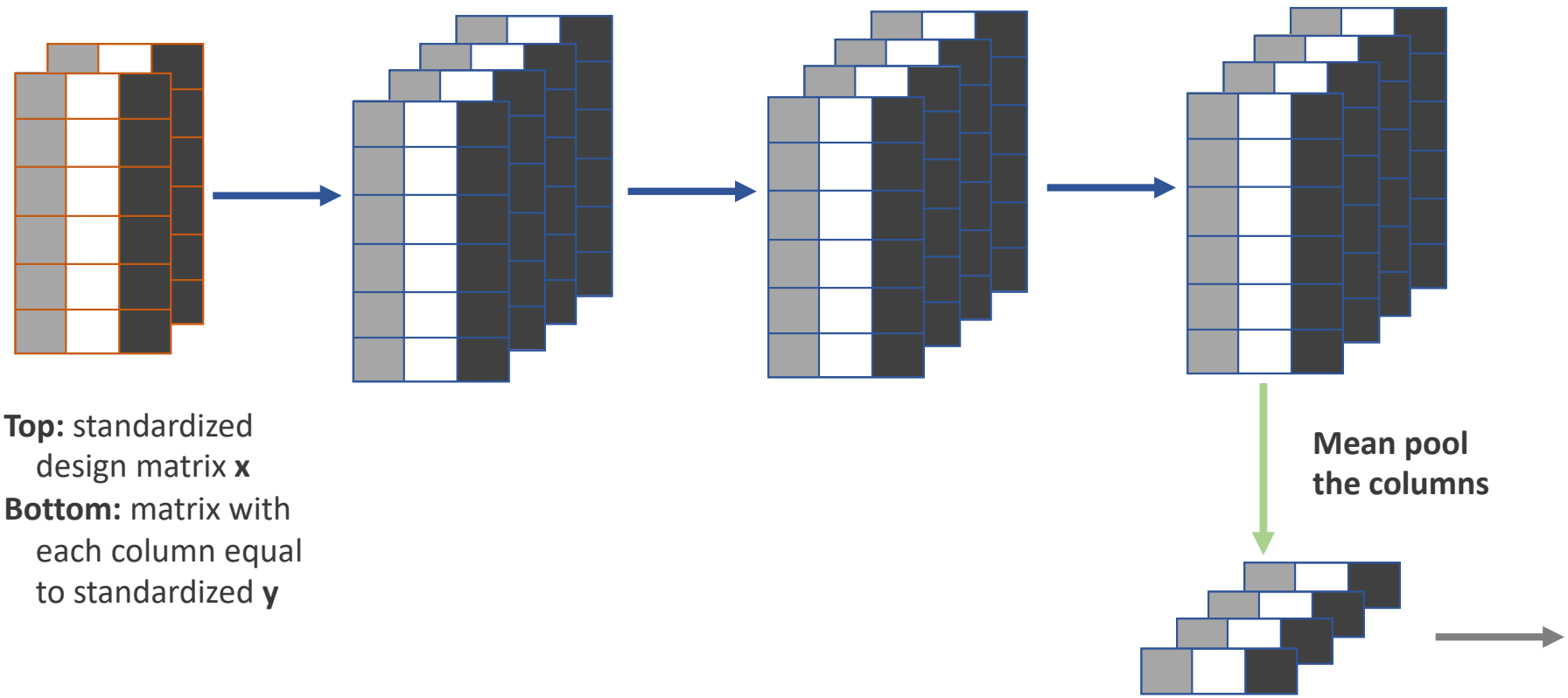



**Top:** standardized design matrix **x**

**Bottom:** matrix with each column equal to standardized **y**

**And equivariant to permutations of the columns.**

# Summary of architecture: module 1 (Hartford et al., 2018)

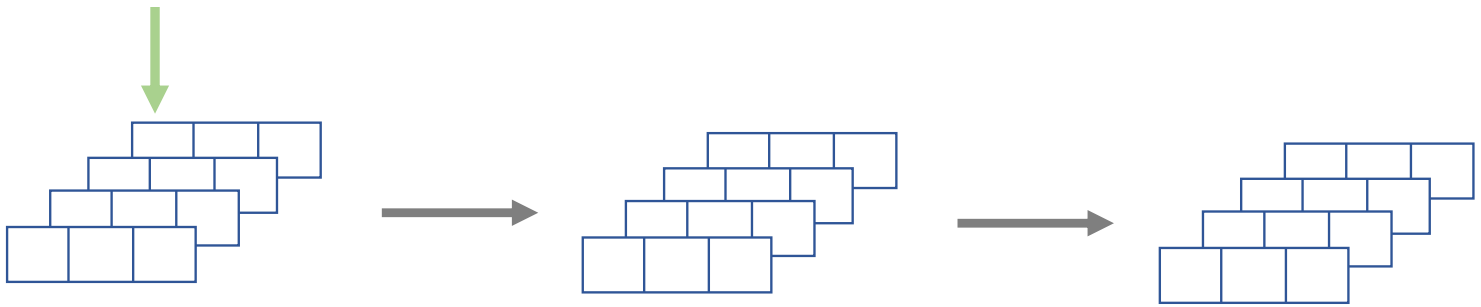



**Top:** standardized design matrix **x**

**Bottom:** matrix with each column equal to standardized **y**

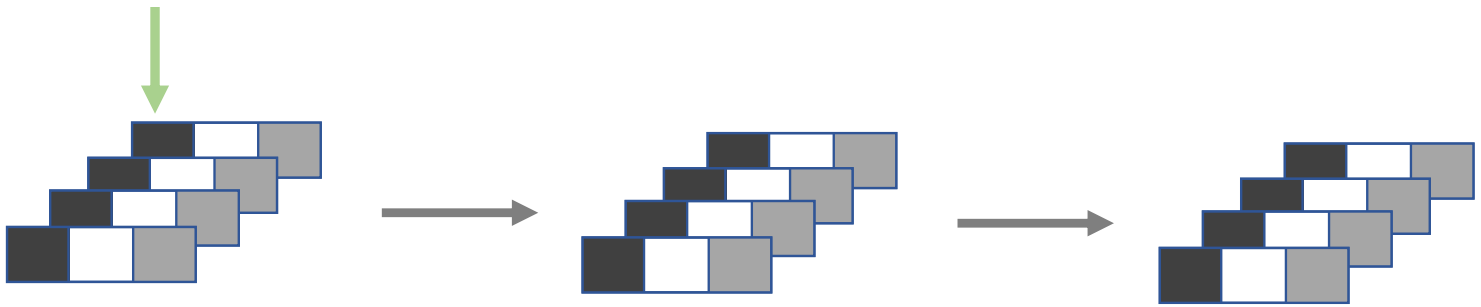
**And equivariant to permutations of the columns.**

# Summary of architecture: module 2 (Zaheer et al., 2017)



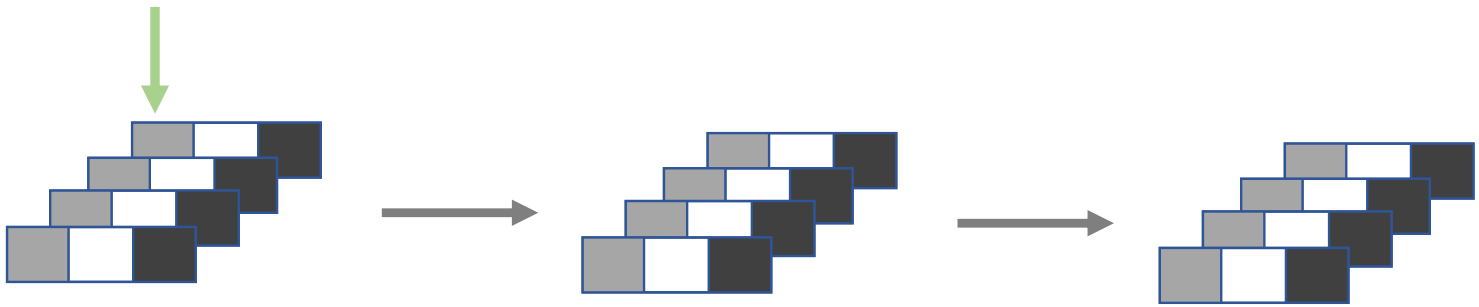
**This module is equivariant to permutations of the columns of its input.**

# Summary of architecture: module 2 (Zaheer et al., 2017)



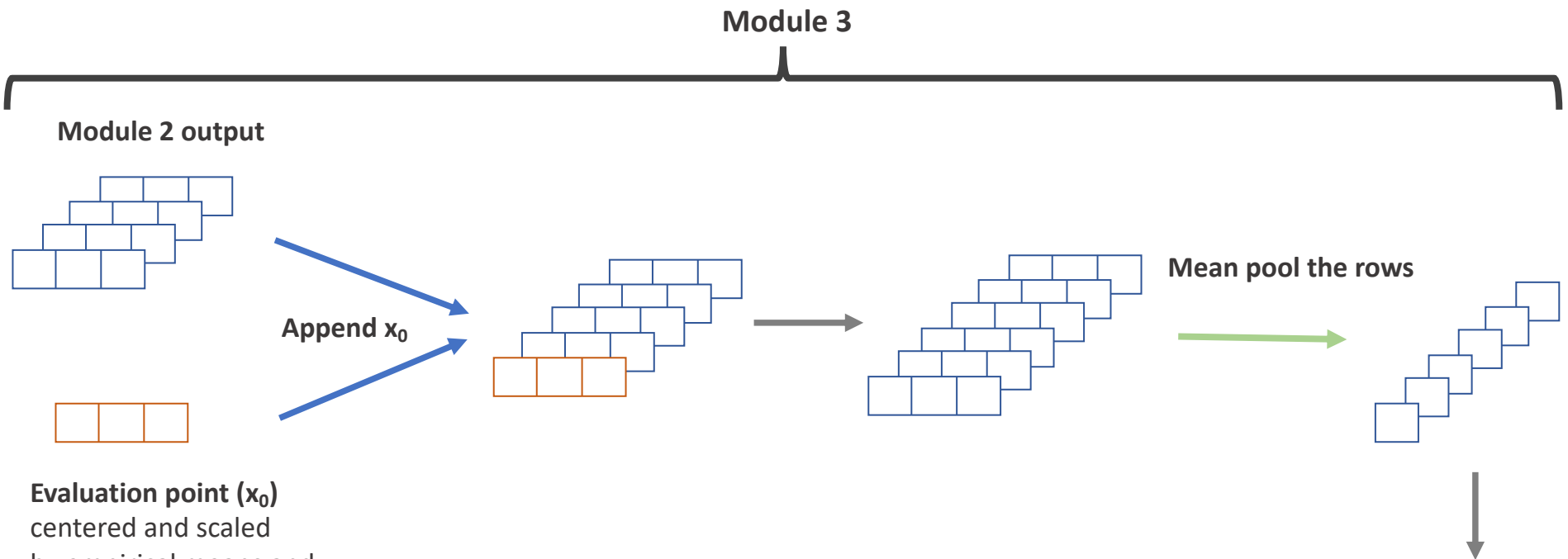
**This module is equivariant to permutations of the columns of its input.**

# Summary of architecture: module 2 (Zaheer et al., 2017)



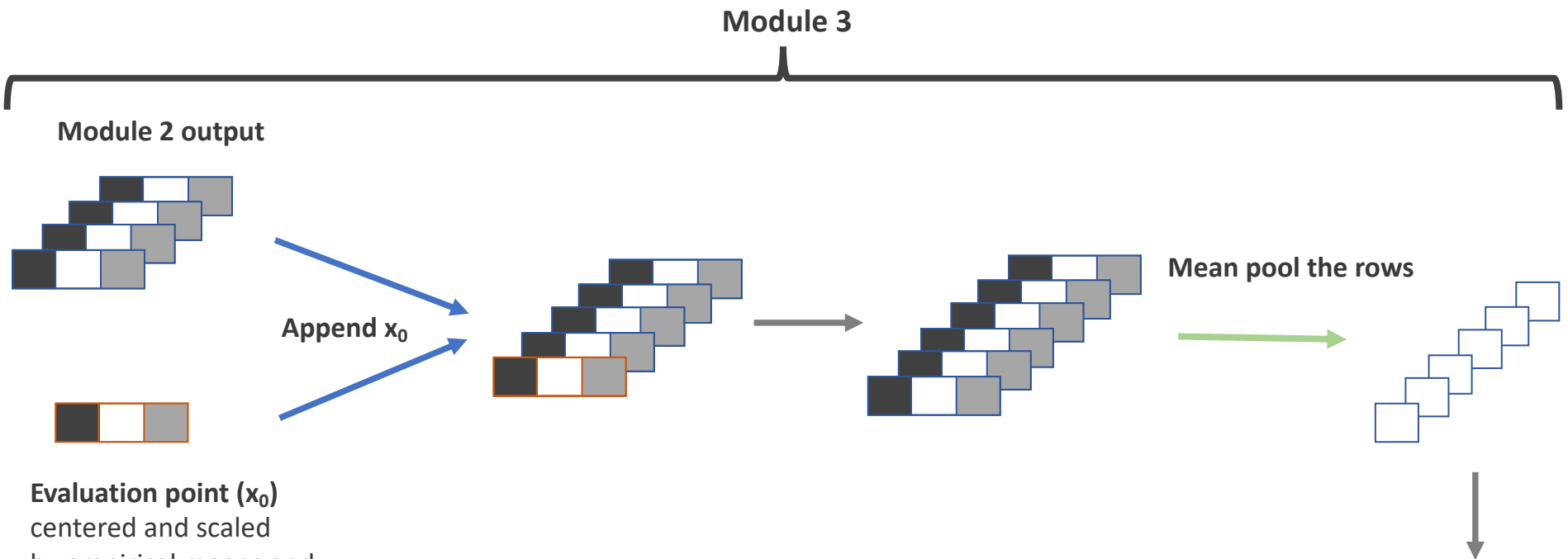
**This module is equivariant to permutations of the columns of its input.**

# Summary of architecture: returning a prediction at an evaluation point



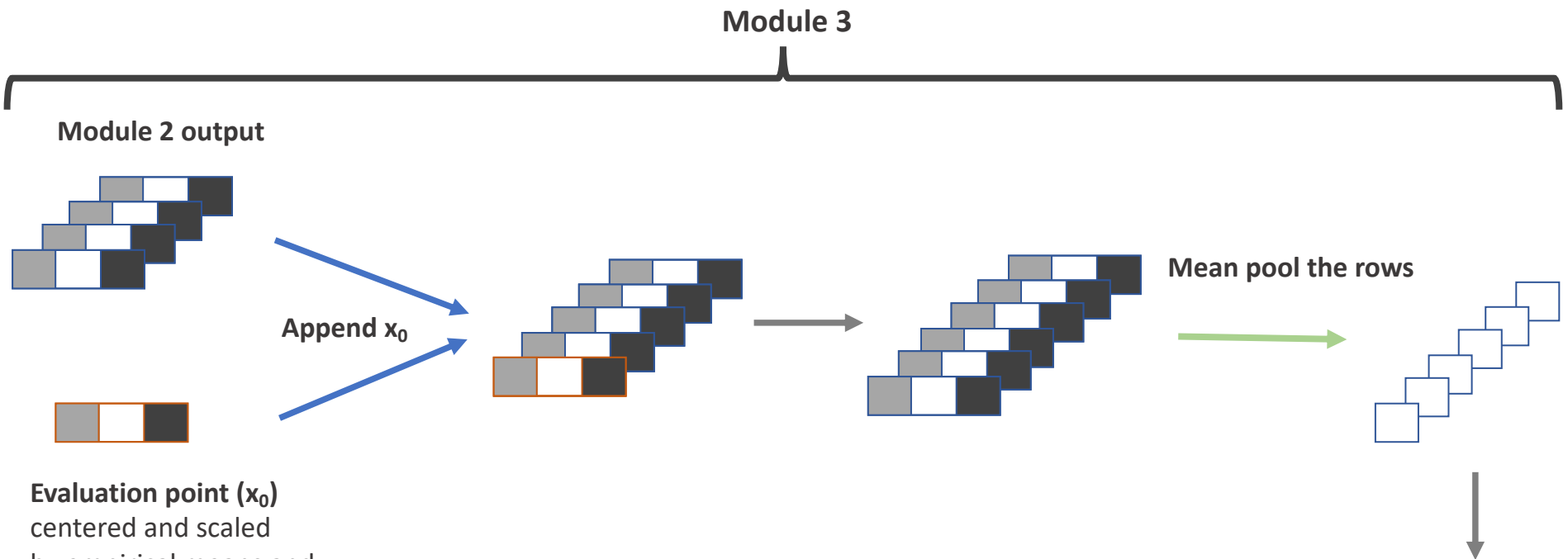

**Module 3 is invariant to permutations of the columns of its input.**

# Summary of architecture: returning a prediction at an evaluation point




**Module 3 is invariant to permutations of the columns of its input.**

# Summary of architecture: returning a prediction at an evaluation point




**Module 3 is invariant to permutations of the columns of its input.**

# Summary of architecture: returning a prediction at an evaluation point

A nearly identical architecture works for CATE estimation.

# Model for our experiments

Each distribution $P \in \mathcal{P}$ is indexed by a positive definite covariate matrix $\Sigma$ and a regression function $\theta_P$ in a set $\Theta$, and

$$X \sim MVN(\mathbf{0}_{10}, \Sigma),$$
$$Y|X \sim N\Big(\theta_P(X), 1\Big).$$

We consider several values of $\Theta$, each indexed by a sparsity level $s \in \{1, 2, \ldots, 10\}$:

1. **Linear regression:**

$$\Theta_s^{\text{lin}} = \Big\{ x \mapsto \beta^\top x \,:\, \|\beta\|_0 \leq s,\, \|\beta\|_1 \leq 5 \Big\}.$$

2. **Fused lasso additive model (FLAM) (Petersen et al., 2014):**

$$\Theta_s^{\text{flam}} = \left\{ x \mapsto \sum_{j=1}^{10} \mu_j(x_j) \,:\, \sum_{j=1}^{10} \|\mu_j\|_{\text{tv}} \leq 10,\ \ \mu_j \neq 0 \text{ for at most } s \text{ values of } j \right\}.$$

# AMC implementation

Used the described neural network with a total of 26 hidden layers.

- Total of $\approx$ 650k parameters.

Used AMC to construct each estimator over 1 million iterations.

- 1 Tesla V100 GPU per estimator.
- Constructing each estimator took **3-5 days**.

**Though constructing an estimator is computationally expensive, evaluating one is not.**

- When $n = 500$ and $q = 10$, each estimator can be evaluated and predictions can be made in $<$**0.05 seconds** on a standard CPU.

# Results for regression function estimation on benchmark datasets from the UCI Machine Learning Repository

| | OLS | Lasso | AMC Linear (ours) | FLAM | AMC FLAM (ours) | Stacked Existing | Stacked AMC (ours) | Stacked Both (ours) |
|---|---|---|---|---|---|---|---|---|
| college | 0.414 | 0.397 | **0.377** | **0.392** | 0.395 | 0.358 | 0.354 | **0.348** |
| happiness | **0.270** | 0.277 | 0.275 | 0.315 | **0.311** | 0.280 | 0.261 | **0.256** |
| hitters | 0.667 | **0.660** | 0.662 | 0.626 | **0.619** | 0.602 | 0.615 | **0.585** |
| wine-red | 0.768 | **0.737** | 0.746 | 0.826 | **0.776** | 0.737 | 0.737 | **0.731** |
| wine-white | 0.833 | **0.814** | 0.824 | 0.899 | **0.860** | 0.809 | 0.815 | **0.802** |

# Results for regression function estimation on benchmark datasets from the UCI Machine Learning Repository

| | OLS | Lasso | AMC Linear (ours) | FLAM | AMC FLAM (ours) | Stacked Existing | Stacked AMC (ours) | Stacked Both (ours) |
|---|---|---|---|---|---|---|---|---|
| college | 0.414 | 0.397 | **0.377** | **0.392** | 0.395 | 0.358 | 0.354 | **0.348** |
| happiness | **0.270** | 0.277 | 0.275 | 0.315 | **0.311** | 0.280 | 0.261 | **0.256** |
| hitters | 0.667 | **0.660** | 0.662 | 0.626 | **0.619** | 0.602 | 0.615 | **0.585** |
| wine-red | 0.768 | **0.737** | 0.746 | 0.826 | **0.776** | 0.737 | 0.737 | **0.731** |
| wine-white | 0.833 | **0.814** | 0.824 | 0.899 | **0.860** | 0.809 | 0.815 | **0.802** |

# Results for regression function estimation on benchmark datasets from the UCI Machine Learning Repository

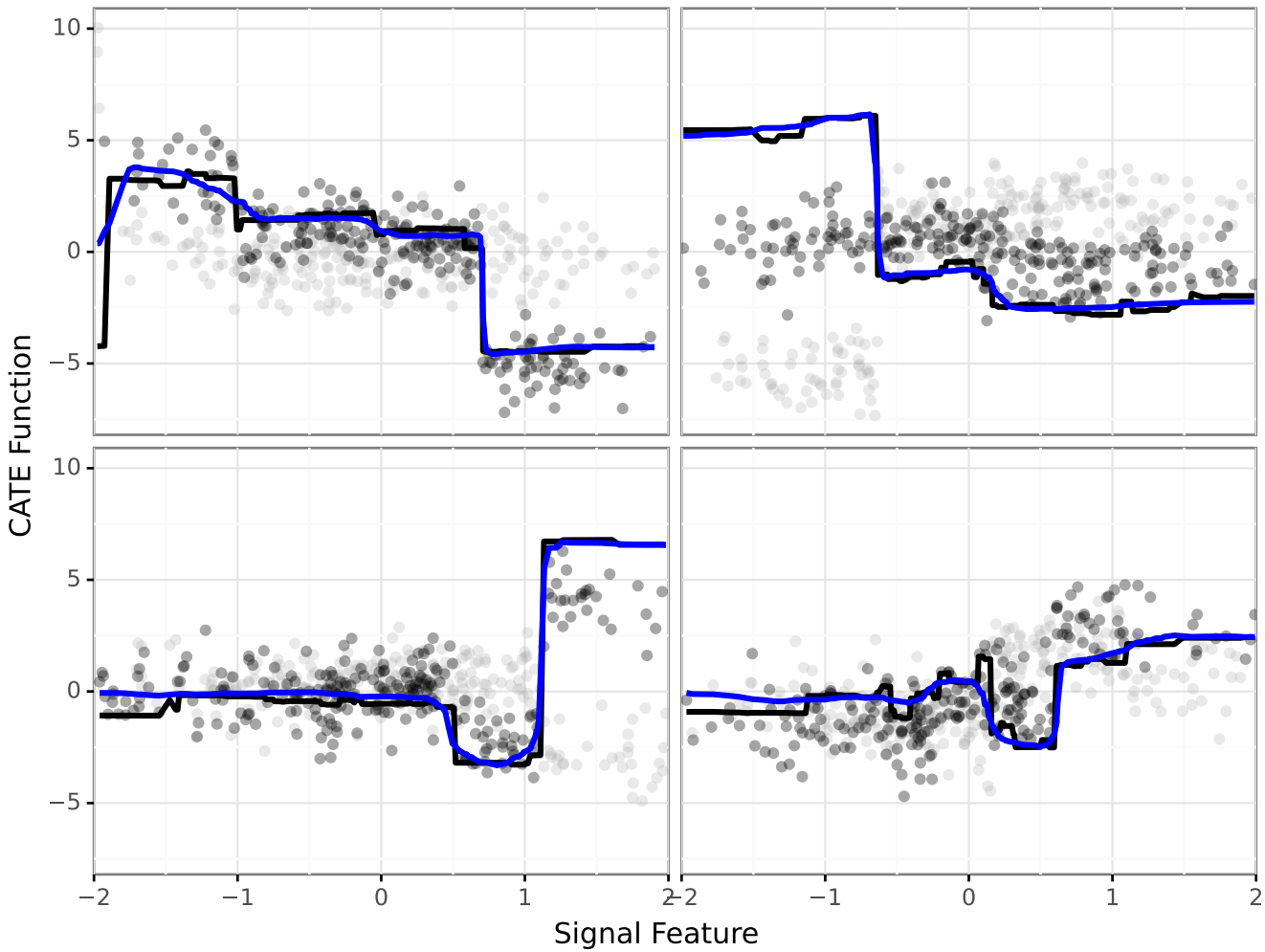
| | OLS | Lasso | AMC Linear (ours) | FLAM | AMC FLAM (ours) | Stacked Existing | Stacked AMC (ours) | Stacked Both (ours) |
|---|---|---|---|---|---|---|---|---|
| college | 0.414 | 0.397 | **0.377** | **0.392** | 0.395 | 0.358 | 0.354 | **0.348** |
| happiness | **0.270** | 0.277 | 0.275 | 0.315 | **0.311** | 0.280 | 0.261 | **0.256** |
| hitters | 0.667 | **0.660** | 0.662 | 0.626 | **0.619** | 0.602 | 0.615 | **0.585** |
| wine-red | 0.768 | **0.737** | 0.746 | 0.826 | **0.776** | 0.737 | 0.737 | **0.731** |
| wine-white | 0.833 | **0.814** | 0.824 | 0.899 | **0.860** | 0.809 | 0.815 | **0.802** |

# Results for regression function estimation on benchmark datasets from the UCI Machine Learning Repository

| | OLS | Lasso | AMC Linear (ours) | FLAM | AMC FLAM (ours) | Stacked Existing | Stacked AMC (ours) | Stacked Both (ours) |
|---|---|---|---|---|---|---|---|---|
| college | 0.414 | 0.397 | **0.377** | **0.392** | 0.395 | 0.358 | 0.354 | **0.348** |
| happiness | **0.270** | 0.277 | 0.275 | 0.315 | **0.311** | 0.280 | 0.261 | **0.256** |
| hitters | 0.667 | **0.660** | 0.662 | 0.626 | **0.619** | 0.602 | 0.615 | **0.585** |
| wine-red | 0.768 | **0.737** | 0.746 | 0.826 | **0.776** | 0.737 | 0.737 | **0.731** |
| wine-white | 0.833 | **0.814** | 0.824 | 0.899 | **0.860** | 0.809 | 0.815 | **0.802** |

# Examples of AMC fits when trained to estimate the CATE

In ongoing experiments, we are showing that AMC can also be used to develop performant estimators of the **CATE** function:




*Black points are treated test points ($A = 1$) and grey points are untreated ($A = 0$).*

# Concluding remarks

Have presented a new approach to adversarially construct regression and CATE function estimators.

- AMC can also be used in **more general statistical decision problems** – see Luedtke et al. (2020) and Qiu and Luedtke (2020).

Currently exploring using AMC to construct **adaptive estimators** in regression/CATE estimation settings.

# Works on AMC

A. Luedtke, M. Carone, N. Simon, and O. Sofrygin. "Learning to learn from data: Using deep adversarial learning to construct optimal statistical procedures". In: *Science Advances* 6.9 (2020), eaaw2140

A. Luedtke, I. Chung, and O. Sofrygin. "Adversarial Monte Carlo Meta-Learning of Optimal Prediction Procedures". In: *Journal of Machine Learning Research* 22.255 (2021), pp. 1–67

H. Qiu and A. Luedtke. "Adversarial meta-learning of Gamma-minimax estimators that leverage prior knowledge". In: *Electronic journal of statistics* 17.2 (2023), p. 1996

A. Luedtke and I. Chung. "Adversarial Monte Carlo Meta-Learning of Conditional Average Treatment Effects". In: *Handbook of Statistical Methods for Precision Medicine*. Chapman and Hall/CRC, 2024, pp. 237–248

Thank you!

# Acknowledgements

I'm grateful to the **NIH** for supporting this research through a **New Innovator Award** (1 DP2 LM013340).

I'm also grateful to **Amazon** for supporting this research through an **AWS Machine Learning Research Award**.

# Complete Bibliography I


[1] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Science & Business Media, 1985.

[2] W. Nelson. "Minimax solution of statistical decision problems by iteration". In: *The Annals of Mathematical Statistics* (1966), pp. 1643–1657.

[3] P. J. Kempthorne. "Numerical specification of discrete least favorable prior distributions". In: *SIAM Journal on Scientific and Statistical Computing* 8.2 (1987), pp. 171–184.

[4] G. Chamberlain. "Econometric applications of maxmin expected utility". In: *Journal of Applied Econometrics* 15.6 (2000), pp. 625–644.

[5] S. Hochreiter, A. S. Younger, and P. R. Conwell. "Learning to learn using gradient descent". In: *International Conference on Artificial Neural Networks*. Springer. 2001, pp. 87–94.

[6] C. Finn, P. Abbeel, and S. Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org. 2017, pp. 1126–1135.

[7] M. Garnelo et al. "Conditional neural processes". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1704–1713.

[8] J. Hartford et al. "Deep models of interactions across sets". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 1909–1918.

[9] M. Zaheer et al. "Deep sets". In: *Advances in neural information processing systems*. 2017, pp. 3391–3401.

[10] A. Petersen, D. Witten, and N. Simon. "Fused lasso additive model". In: *Journal of Computational and Graphical Statistics* 25.4 (2016), pp. 1005–1025.

[11] A. Luedtke et al. "Learning to learn from data: Using deep adversarial learning to construct optimal statistical procedures". In: *Science Advances* 6.9 (2020), eaaw2140.

[12] H. Qiu and A. Luedtke. "Adversarial meta-learning of Gamma-minimax estimators that leverage prior knowledge". In: *Electronic journal of statistics* 17.2 (2023), p. 1996.

[13] A. Luedtke, I. Chung, and O. Sofrygin. "Adversarial Monte Carlo Meta-Learning of Optimal Prediction Procedures". In: *Journal of Machine Learning Research* 22.255 (2021), pp. 1–67.

[14] A. Luedtke and I. Chung. "Adversarial Monte Carlo Meta-Learning of Conditional Average Treatment Effects". In: *Handbook of Statistical Methods for Precision Medicine*. Chapman and Hall/CRC, 2024, pp. 237–248.

[15] T. Lin, C. Jin, and M. Jordan. "On gradient descent ascent for nonconvex-concave minimax problems". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6083–6093.

# Complete Bibliography II


[16] G. W. Brown. "Iterative solution of games by fictitious play". In: *Activity analysis of production and allocation* 13.1 (1951), pp. 374–376.

[17] I. M. Johnstone and K. B. MacGibbon. "Minimax estimation of a constrained Poisson vector". In: *The Annals of Statistics* 20.2 (1992), pp. 807–831.

[18] E. Gourdin, B. Jaumard, and B. MacGibbon. "Global optimization decomposition methods for bounded parameter minimax risk evaluation". In: *SIAM Journal on Scientific Computing* 15.1 (1994), pp. 16–35.

[19] C. M. Schafer and P. B. Stark. "Constructing confidence regions of optimal expected size". In: *Journal of the American Statistical Association* 104.487 (2009), pp. 1080–1089.

[20] B. Bryan et al. "Efficiently computing minimax expected-size confidence regions". In: *Proceedings of the 24th international conference on Machine learning*. ACM. 2007, pp. 97–104.

# Extra Slides

# Implementing adversarial Monte Carlo meta-learning with other optimization schemes

The algorithm on the preceding slide is a form of stochastic gradient descent ascent (SGDA) (e.g., Lin et al., 2020).

Many variants are possible. Here are three examples:

- **Stochastic gradient descent with max oracle** (Luedtke et al., 2020; Lin et al., 2020): converges to a near-equilibrium point under conditions, but computationally costly.
- **Stochastic extragradient methods** (Mischenko, 2020): sometimes have better convergence properties than does SGDA.
- **Fictitious play** (Brown, 1951; Qiu and Luedtke, 2020): will converge when $\Gamma$ consists of mixtures of a fixed set of $k$ priors.

# Summary of key conditions for theorem

$\Gamma$ is preserved under the following transformations:

1. **Permutations of features:** $\Pi \in \Gamma$ and $B \in \mathcal{B}_q$ implies that $\Pi \circ f_1^{-1} \in \Gamma$, where $f_1(P)$ is the distribution of $(BX, Y)$ when $(X, Y) \sim P$.

2. **Increasing affine transformations of features:** $\Pi \in \Gamma$, $b \in \mathbb{R}^q$, and $c \in (0, \infty)^q$ implies that $\Pi \circ f_2^{-1} \in \Gamma$, where $f(P)$ is the distribution of $(b + c \odot X, Y)$ when $(X, Y) \sim P$.

3. **Increasing affine transformations of outcome:** $\Pi \in \Gamma$, $b \in \mathbb{R}$, and $c > 0$ implies that $\Pi \circ f_3^{-1} \in \Gamma$, where $f(P)$ is the distribution of $(X, b + cY)$ when $(X, Y) \sim P$.

*Additional technical regularity conditions can be found in Luedtke et al. (2021).*

# Ablation study: evaluating importance of permutation invariance

Fold-change in MSEs for modifications of AMC in the FLAM regression settings, as compared to the performances of AMC FLAM under our proposed architecture.

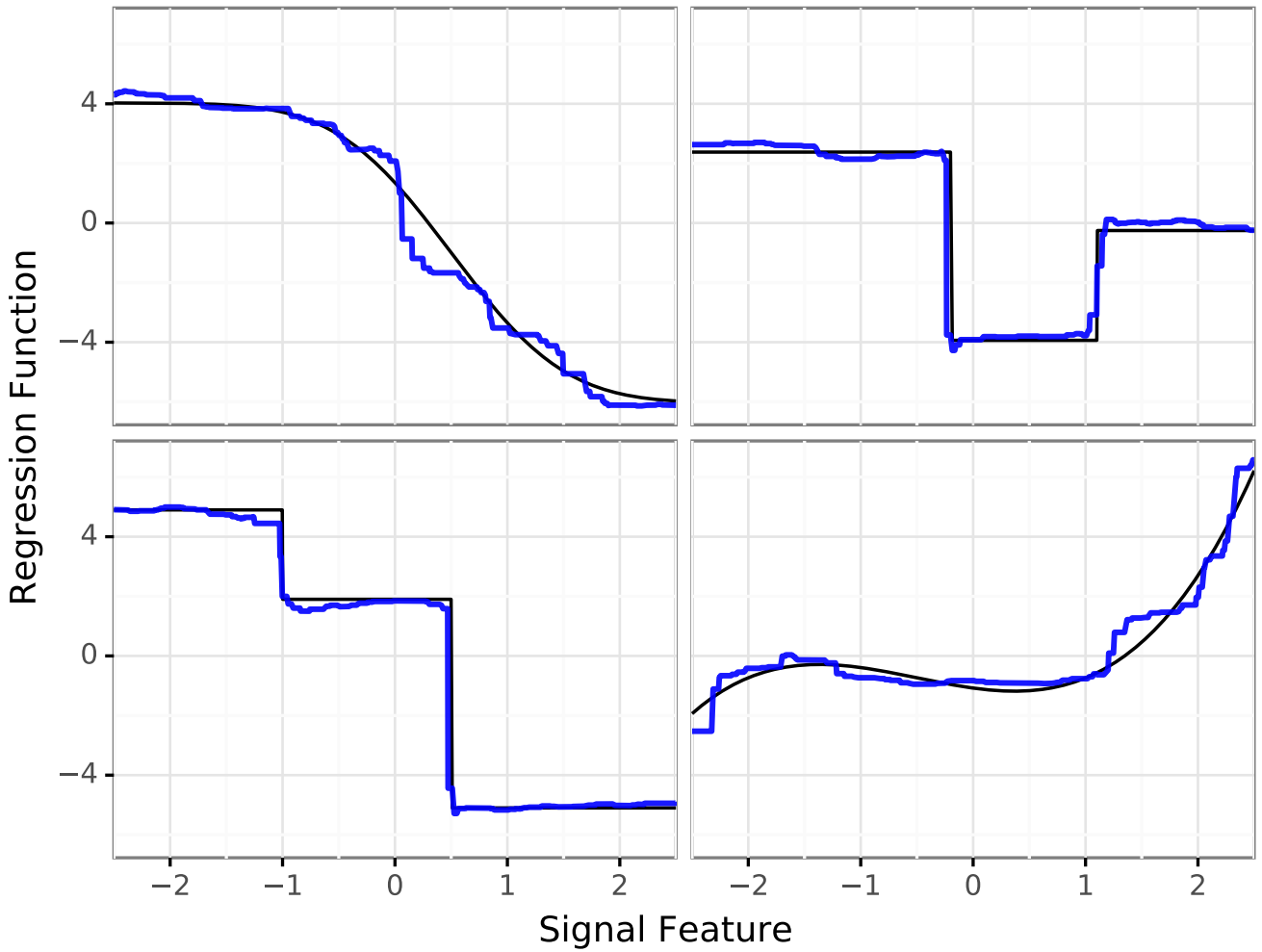
**(a) Sparse signal**

| Not invariant to | Scenario 1 | | Scenario 2 | | Scenario 3 | | Scenario 4 | |
|---|---|---|---|---|---|---|---|---|
| permutations of: | $n$=100 | 500 | 100 | 500 | 100 | 500 | 100 | 500 |
| observations | 6.98 | 38.29 | 5.82 | 29.93 | 5.03 | 27.58 | 4.29 | 13.08 |
| features | 1.01 | 0.95 | 1.16 | 1.09 | 1.02 | 0.98 | 1.01 | 0.99 |

**(b) Dense signal**

| Not invariant to | Scenario 1 | | Scenario 2 | | Scenario 3 | | Scenario 4 | |
|---|---|---|---|---|---|---|---|---|
| permutations of: | $n$=100 | 500 | 100 | 500 | 100 | 500 | 100 | 500 |
| observations | 1.86 | 14.68 | 1.69 | 8.60 | 1.97 | 14.20 | 1.51 | 4.70 |
| features | 1.05 | 2.55 | 0.99 | 1.98 | 1.09 | 3.02 | 1.04 | 1.67 |

# Examples of AMC fits when trained under FLAM regression model ($n = 500$) at sparsity level $s = 1$




*The four regression functions displayed above are derived from the four scenarios considered in the simulation study from Petersen et al. (2014).*

# Performance in FLAM regression simulations

Mean squared errors (MSEs) based on datasets of size $n$ in FLAM regression settings.

**(a) Sparse signal ($s = 1$)**

| | Scenario 1 | | Scenario 2 | | Scenario 3 | | Scenario 4 | |
|---|---|---|---|---|---|---|---|---|
| | $n$=100 | 500 | 100 | 500 | 100 | 500 | 100 | 500 |
| FLAM | 0.44 | **0.12** | 0.47 | 0.17 | 0.38 | 0.11 | 0.51 | 0.19 |
| AMC (ours) | **0.34** | **0.12** | **0.18** | **0.06** | **0.27** | **0.10** | **0.17** | **0.08** |

**(b) Denser signal ($s = 4$)**

| | Scenario 1 | | Scenario 2 | | Scenario 3 | | Scenario 4 | |
|---|---|---|---|---|---|---|---|---|
| | $n$=100 | 500 | 100 | 500 | 100 | 500 | 100 | 500 |
| FLAM | **0.59** | 0.17 | 0.65 | 0.24 | **0.53** | 0.16 | 0.76 | 0.36 |
| AMC (ours) | 1.20 | **0.15** | **0.47** | **0.08** | 0.87 | **0.12** | **0.30** | **0.09** |

# Performance in sparse linear regression simulations

MSEs based on datasets of size $n$ in linear regression settings.

**(a) Sparse signal ($s = 1$)**

| | Boundary | | Interior | |
|---|---|---|---|---|
| | $n$=100 | 500 | 100 | 500 |
| OLS | 0.12 | 0.02 | 0.12 | 0.02 |
| Lasso | 0.06 | 0.01 | **0.06** | **0.01** |
| AMC (ours) | **0.02** | <**0.01** | 0.11 | 0.04 |

**(b) Denser signal ($s = 5$)**

| | Boundary | | Interior | |
|---|---|---|---|---|
| | $n$=100 | 500 | 100 | 500 |
| OLS | 0.13 | **0.02** | 0.13 | **0.02** |
| Lasso | 0.11 | **0.02** | 0.09 | **0.02** |
| AMC (ours) | **0.10** | **0.02** | **0.08** | **0.02** |

# Experiments evaluating AMC for estimation of the CATE

In preparation for a manuscript submission, I'm also in the process of running numerical experiments evaluating AMC for CATE estimation.

These simulations are nearly identical to those already displayed, except:

1. A randomized treatment indicator $A \sim \mathrm{Bern}(1/2)$ is observed.
2. Rather than assuming that the regression function $x \mapsto E[Y|X = x]$ belongs to a linear regression or fused lasso additive model class, instead assume that the outcome regressions $x \mapsto E[Y|A = 0, X = x]$ and $x \mapsto E[Y|A = 1, X = x]$ both belong to such a class.

# Performance for CATE estimation when outcome regression is linear

We evaluate performance when $n = 500$ and $X \sim MVN(\mathbf{0}_{10}, \mathrm{Id})$ and $E_P[Y|A = a, X = x] = 0.5\beta(2a - 1)x_1$, so that

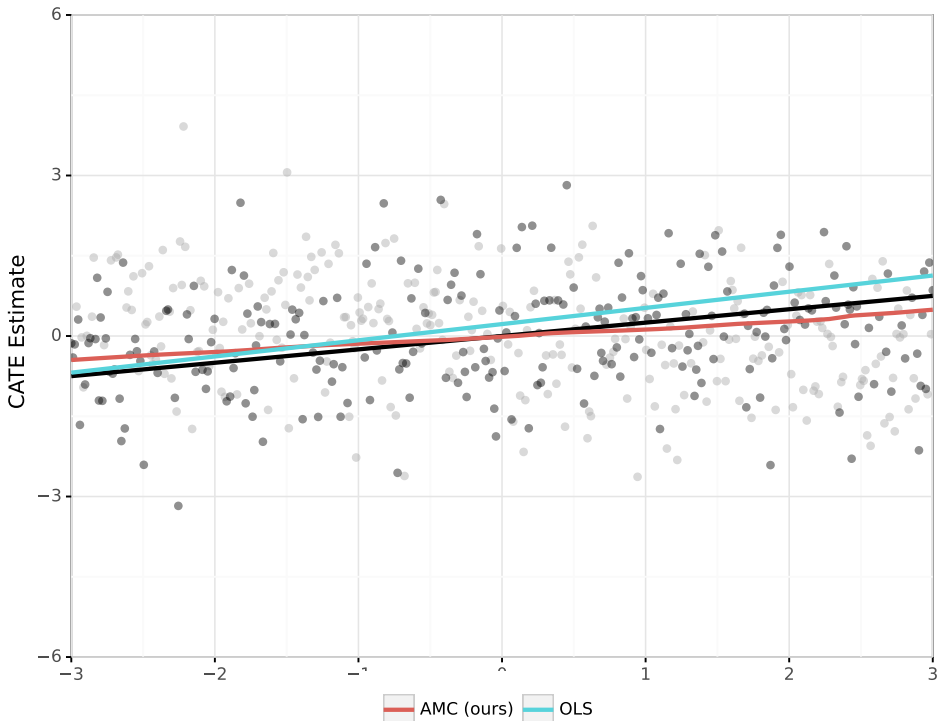
$$\Delta_P(x) = \beta x_1.$$

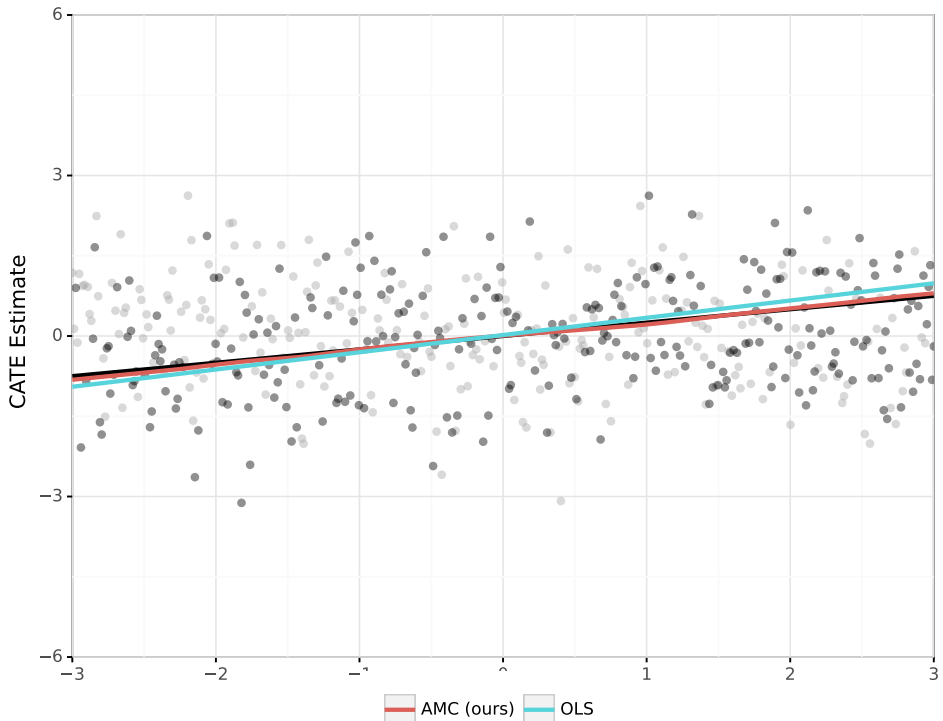
For different choices of $\beta$, we'll display the MSE along with two example fits on a dataset of size $n = 500$.

- Performance compared to that of a plug-in estimator that estimates outcome regressions with ordinary-least squares (OLS) regression.
- Given that $n \gg q$ and the truth is linear in this setting, OLS regression is expected to perform very well in this setting.

# Performance for CATE estimation when outcome regression is linear ($\beta = 0.5$)

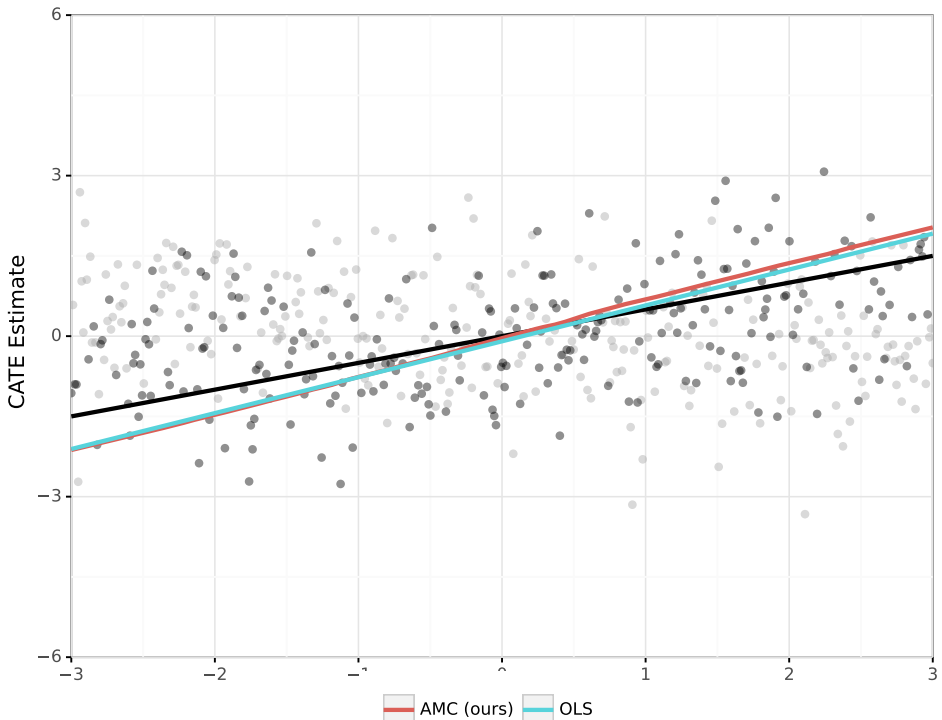
| | MSE |
|---|---|
| OLS | 0.017 |
| AMC (ours) | **0.015** |

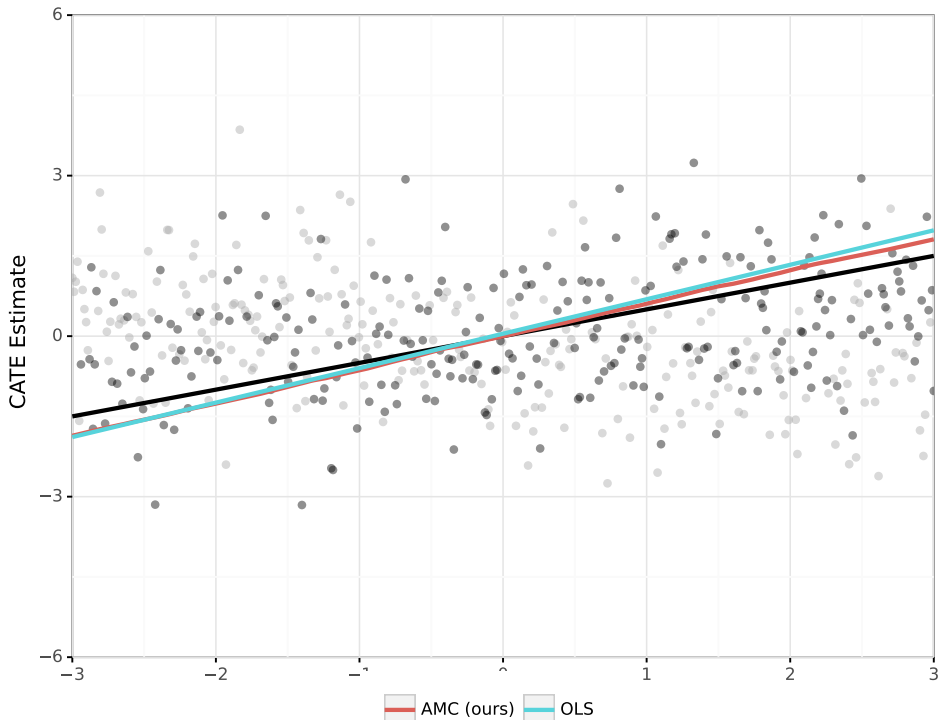


# Performance for CATE estimation when outcome regression is linear ($\beta = 1$)

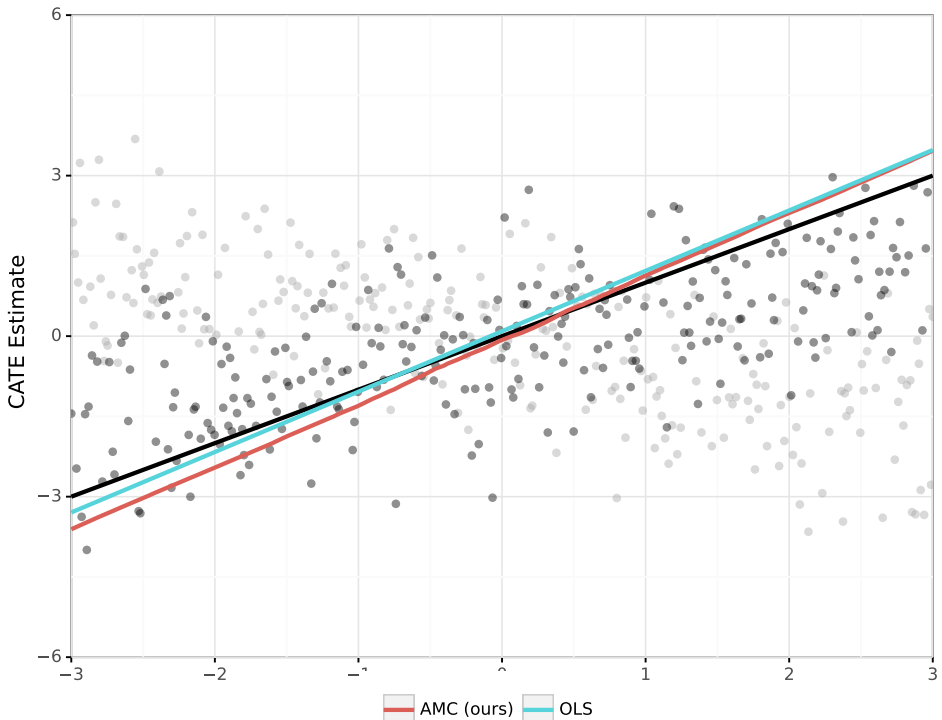
| | MSE |
|---|---|
| OLS | 0.017 |
| AMC (ours) | **0.012** |

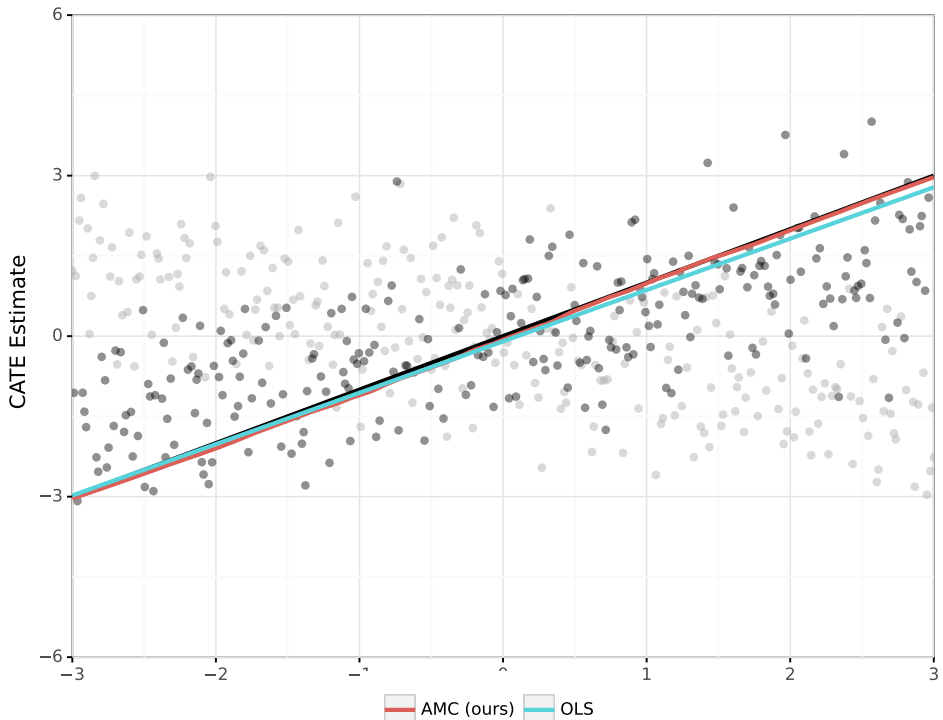


# Performance for CATE estimation when outcome regression is linear ($\beta = 2$)

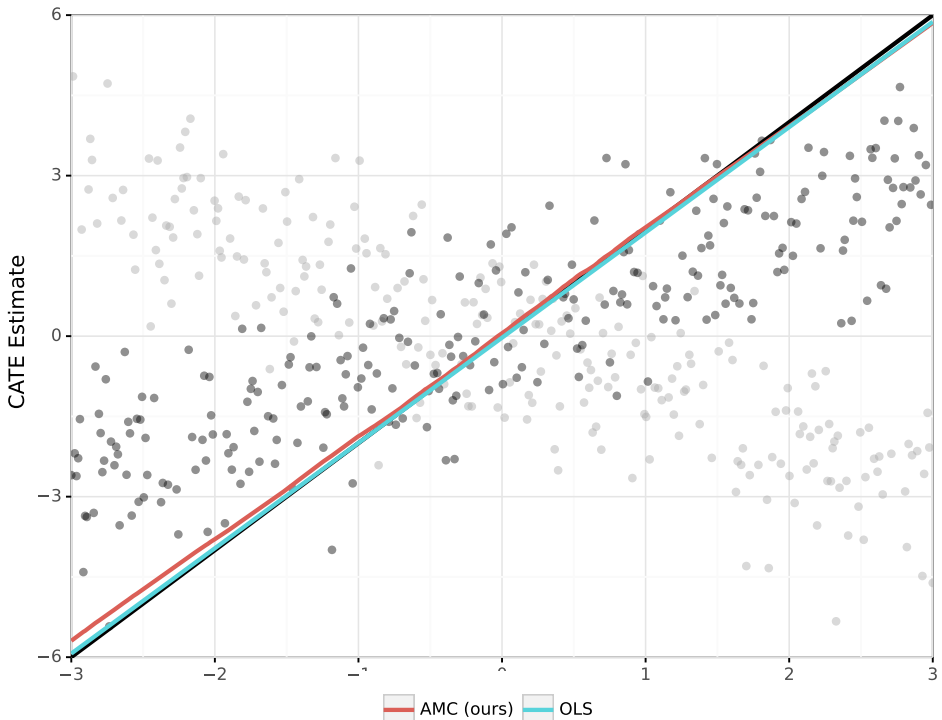
| | MSE |
|---|---|
| OLS | 0.017 |
| AMC (ours) | **0.015** |

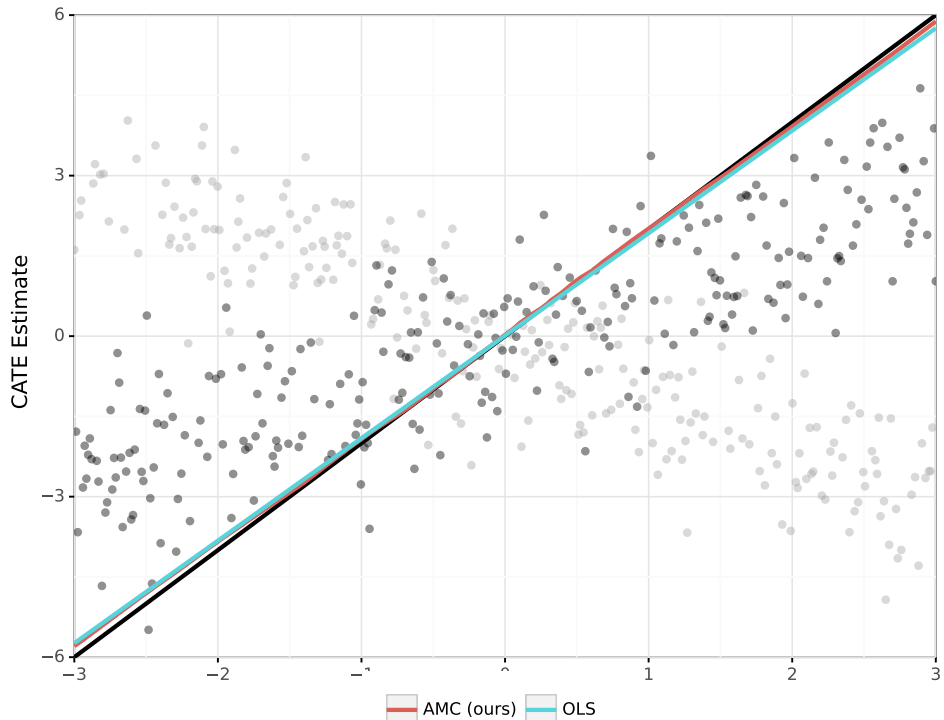


# Performance for CATE estimation when outcome regression is linear ($\beta = 4$)

| | MSE |
|---|---|
| OLS | **0.017** |
| AMC (ours) | 0.024 |



# Example of invariance to increasing affine transformations of features

The estimator $T^\star$ outputs the **same prediction** on

| $\boldsymbol{x}$ | | | $\boldsymbol{y}$ |
|---|---|---|---|
| 2.5 | 3.1 | 4.8 | 7.8 |
| 5.8 | 1.2 | 1.1 | 3.6 |
| 4.9 | 6.6 | 2.8 | 4.6 |
| 1.9 | 0.7 | 1.9 | 9.0 |
| $-1$ | $+2$ | $\times 2 - 1$ | |

| $x_0$ | | |
|---|---|---|
| 4.3 | 7.5 | 1.7 |
| $-1$ | $+2$ | $\times 2 - 1$ |

and

| $\boldsymbol{x}$ | | | $\boldsymbol{y}$ |
|---|---|---|---|
| 1.5 | 5.1 | 8.6 | 7.8 |
| 4.8 | 3.2 | 1.2 | 3.6 |
| 3.9 | 8.6 | 4.6 | 4.6 |
| 0.9 | 2.7 | 2.8 | 9.0 |

| $x_0$ | | |
|---|---|---|
| 3.3 | 9.5 | 2.4 |

# Example of equivariance to increasing affine transformations of outcome

If the estimator $T^\star$ outputs $\mathbf{2}$ on

| $\boldsymbol{x}$ | | | $\boldsymbol{y}$ |
|---|---|---|---|
| 2.5 | 3.1 | 4.8 | 7.8 |
| 5.8 | 1.2 | 0.9 | 3.6 |
| 4.9 | 6.6 | 2.8 | 4.6 |
| 1.9 | 0.7 | 1.9 | 9.0 |

$\times 0.5 + 3$

| $x_0$ | | |
|---|---|---|
| 4.3 | 7.5 | 1.7 |

then it outputs $\mathbf{2} \times 0.5 + 3 = 4$ on

| $\boldsymbol{x}$ | | | $\boldsymbol{y}$ |
|---|---|---|---|
| 2.5 | 3.1 | 4.8 | 6.9 |
| 5.8 | 1.2 | 0.9 | 4.8 |
| 4.9 | 6.6 | 2.8 | 5.3 |
| 1.9 | 0.7 | 1.9 | 7.5 |

| $x_0$ | | |
|---|---|---|
| 4.3 | 7.5 | 1.7 |

# Similar Hunt-Stein-type theorem holds for CATE estimation

**A similar Hunt-Stein-type theorem can be derived for CATE estimation.**

The only condition that changes is that, rather than having that

$$T^\star(\boldsymbol{x}, \boldsymbol{a}, b\boldsymbol{y} + c)(x_0) = bT^\star(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})(x_0) + c$$

for all $b \in \mathbb{R}$ and $c > 0$, we have that

$$T^\star(\boldsymbol{x}, \boldsymbol{a}, b\boldsymbol{y} + c)(x_0) = bT^\star(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})(x_0).$$

This makes sense in light of the fact that

$$E[bY + c|A = 1, X] - E[bY + c|A = 0, X] = b\left(E[Y|A = 1, X] - E[Y|A = 0, X]\right).$$

# Desirable features of architecture

Can be evaluated in settings where there are different numbers of observations $n$ and features $q$ than were used during training.

Computational and space complexity of evaluating estimator are both $O(nq)$.

# Existing work when $\Gamma$ is unrestricted (1/2)

**1966:** Nelson described an algorithm to iteratively construct unfavorable priors as a mixture between the current prior and a numerically derived less favorable prior.

**1987:** Kempthorne proposed a similar algorithm for the special case that the statistical model is one-dimensional. This algorithm iteratively updates discrete priors with finite support and reduces computational burden by allowing the support to shrink at each iteration.

Both of the above works provided proofs that the proposed estimators converge to the minimax optimal estimator as the number of iterations grows.

**1992:** Johnstone and MacGibbon present a related method for deriving the least favorable prior when estimating the mean of a Poisson vector.

**1994:** Gourdin, Jaumard, and MacGibbon present a global optimization procedure for identifying this discretely-supported least favorable prior.

**All above listed approaches assume that it is easy to derive the Bayes estimator for a given discrete prior.**

# Existing work when $\Gamma$ is a finite mixture class (2/2)

**2000:** Chamberlain formulates an algorithm for general decision problems that involves solving a convex optimization problem.

**2006:** Schafer and Stark provide a method for constructing confidence regions of minimal size (published in *JASA* in 2009). An optimal strategy is found via fictitious play.

**2007:** Bryan, McMahan, Schafer, and Schneider show that leveraging the near-sparsity of the problem can more efficiently lead to a solution.

**If the number of support points for the prior distribution is large, then it will be difficult to derive the least favorable prior distribution in $\Gamma$ and to compute the corresponding Bayes estimator.**