This is an Accepted Manuscript of a book chapter published by CRC Press in *Handbook of Statistical Methods for Precision Medicine* on 23 October 2024, available online: https://doi.org/10.1201/9781003216223-11

Adversarial Monte Carlo Meta-Learning of Conditional Average Treatment Effects

Alex Luedtke and Incheoul Chung

Department of Statistics, University of Washington, USA

May 16, 2022

Abstract

Traditionally, statistical procedures have been derived via analytic calculations whose validity often relies on sample size growing to infinity. In this chapter, we describe how to use deep adversarial learning to numerically construct a statistical procedure that performs well even in small samples. More concretely, we frame the meta-learning of conditional average treatment effect estimators as a search for an optimal strategy in a two-player game. In this game, Nature selects a prior over distributions that generate labeled data consisting of covariates, treatment, and an associated outcome, and the Estimator observes data sampled from a distribution drawn from this prior. The Estimator's objective is to learn a function that maps from a new feature to an estimate of the conditional average treatment effect. We argue that, under reasonable conditions, the Estimator's has an optimal strategy that is equivariant to shifts and rescalings of the outcome and is invariant to permutations of the observations and to shifts, rescalings, and permutations of the features. We introduce a neural network architecture that satisfies these properties.

1 Problem Formulation and Objective

Suppose we observe a dataset consisting of observations (X_i, A_i, Y_i), $i = 1, 2, \ldots, n$, drawn independently from a distribution P belonging to some known model $\mathcal{P}$, where X_i is a continuously distributed covariate with support contained in $\mathcal{X} := \mathbb{R}^p$, A_i is a treatment with support $\mathcal{A} := \{0, 1\}$, and Y_i is an outcome with support contained in $\mathcal{Y} := \mathbb{R}$. This dataset can be written as $\boldsymbol{D} := (\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y})$, where $\boldsymbol{X}$ is the $n \times p$ matrix for which row i contains X_i and $\boldsymbol{A}$ and $\boldsymbol{Y}$ are the n-dimensional vectors for which entry i contains A_i and Y_i, respectively. The objective is to develop an estimator of the conditional average treatment effect function (CATE) θ_P that maps from an evaluation point x_0 to $\mathrm{E}_P[Y|A = 1, X = x_0] - \mathrm{E}_P[Y|A = 0, X = x_0]$. An

Generous support was provided by the National Institutes of Health (NIH) under award number DP2-LM013340. The content is solely the responsibility of the authors and does not necessarily represent the official views of the NIH.

estimator T belongs to the collection $\mathcal{T}$ of operators that take as input a dataset $\boldsymbol{d} := (\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})$ and output a prediction function $T(\boldsymbol{d}) : \mathcal{X} \to \mathbb{R}$, where here and throughout we use $\boldsymbol{d} = (\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})$ to denote a possible realization of the random variable $\boldsymbol{D} = (\boldsymbol{X}, \boldsymbol{A}, \boldsymbol{Y})$. Examples of estimators include Bayesian additive regression trees (Hill, 2011), causal forests (Athey et al., 2019), and stacked ensemble learners (Luedtke and van der Laan, 2016). We quantify the performance of an estimator T at a distribution P via its standardized mean-squared error (MSE), namely

$$R(T, P) := \mathrm{E}_P \left[\int \frac{[T(\boldsymbol{D})(x_0) - \theta_P(x_0)]^2}{\sigma_P^2} dP_X(x_0) \right], \tag{1}$$

where the expectation above is over the draw of $\boldsymbol{D}$ under sampling from P, P_X denotes the marginal distribution of X implied by P, and $\sigma_P^2 := \mathrm{E}_P[\mathrm{Var}_P(Y \mid A, X)/\pi_P(A \mid X)^2]$ with $\pi_P(a \mid x) := P(A = a \mid X = x)$ denoting the propensity to receive treatment a given covariate value x. The standardization factor is chosen to scale with the semiparametric efficiency bound (Bickel et al., 1993) for estimating a smooth functional of θ_P, namely the marginal average treatment effect $q(\theta_P) := \int \theta_P(x) dP_X(x)$ in a model that may differ from $\mathcal{P}$, namely the model that is nonparametric up to the fact that P_X is known. The rationale for including this standardization factor is that, when estimating a smooth functional of θ_P is difficult (σ_P^2 is large), estimating θ_P itself must also be difficult. Consequently, there is little reason to disfavor an estimator for its performance at P if the estimation of $q(\theta_P)$ is inherently challenging.

Since P is not known in practice, $R(T, P)$ cannot be used to quantify the performance of an estimator T. Hence, another performance criterion must be sought. In this chapter, we focus on a general criterion known as the Γ-maximal risk (Berger, 1985). To define this notion, we let Γ denote a collection of priors Π with support on the statistical model $\mathcal{P}$. For a given prior Π, the Bayes risk of T is defined as $r(T, \Pi) := \int R(T, P) \Pi(dP)$, and the Γ-maximal risk is defined as $\sup_{\Pi \in \Gamma} r(T, \Pi)$. For a given prior Π, the Bayes risk expresses the expected performance of the estimator T under the prior assumptions encoded in Π. The Γ-maximal risk provides a means to avoid selecting a single prior, and instead considers the worst-case performance, in terms of Bayes risk, over a collection of priors. At one extreme, when Γ is the singleton collection $\{\Pi\}$, the Γ-maximal risk and the Bayes risk coincide. At the other extreme, when Γ contains all priors with support on $\mathcal{P}$, the Γ-maximal risk instead coincides with the maximal risk $\sup_{P \in \mathcal{P}} R(T, P)$. Thus, the Γ-maximal risk provides a natural means to interpolate between adjudicating performance via a single set of prior beliefs, as is done by the Bayes risk, and via a worst-case analysis, as is done by the maximal risk.

An estimator $T^\star$ is called Γ-minimax if it achieves the minimal possible Γ-maximal risk, that is, if

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

Unfortunately, Γ-minimax estimators are rarely available in closed form, and so they have seen limited use in practice. In this chapter, we will describe how recently developed meta-learning techniques can be used to iteratively construct Γ-minimax estimators. Many of our arguments will allow for a general choice of Γ, and so, as special cases, the framework we describe can be used to iteratively approximate a Bayes estimator (when Γ is a singleton) or a minimax estimator (when Γ is large or unrestricted).

Regularity Conditions and Notation

Throughout we assume that, for all $P \in \mathcal{P}$, $\mathrm{E}_P[Y^2] < \infty$ and that $Y - \mathrm{E}_P[Y \mid A, X]$ is a continuous random variable. We also suppose that the strong positivity assumption holds, namely, that there exists a $\delta > 0$ such that, for all $P \in \mathcal{P}$, the following holds P-almost surely: $\min_{a \in \{0,1\}} P(A = a \mid X) \geq \delta$. Notably, these conditions imply that $\sigma^2_P \in (0, \infty)$.

We now introduce the notation and conventions that we use. For any dataset $\boldsymbol{d} = (\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})$ and mapping f with domain $\mathcal{D} := \mathcal{X}^n \times \mathcal{A}^n \times \mathcal{Y}^n$, we let $f(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y}) := f(\boldsymbol{d})$. We take all vectors to be column vectors when they are involved in matrix operations. We write $\odot$ to mean the entrywise product and $b^{\odot 2}$ to mean $b \odot b$. For an $m_1 \times m_2$ matrix b, we let b_{i*} denote the i^{th} row, b_{*j} denote the j^{th} column, $\bar{b} := \frac{1}{m_1} \sum_{i=1}^{m_1} b_{i*}$ denote the column means, and $s(b)^{\odot 2} := \frac{1}{m_1} \sum_{i=1}^{m_1} (b_{i*} - \bar{b})^{\odot 2}$ denote the column standard variances. When we standardize a vector b as $[b - \bar{b}]/s(b)$, we always use the convention that $0/0 = 0$. We apply the same division-by-zero convention when standardizing the columns of an $m_1 \times m_2$ matrix b, where we write this standardization as $\frac{b - \bar{b}}{s(b)}$ to denote the $m_1 \times m_2$ matrix for which row i is equal to $[b_i - \bar{b}]/s(b)$. We write $[b \,|\, c]$ to denote the column concatenation of two matrices. For an $m_1 \times m_2 \times m_3$ array b, we let b_{i**} denote the $m_2 \times m_3$ matrix with entry (j, k) equal to b_{ijk}, b_{i*k} denote the m_2-dimensional vector with entry j equal to b_{ijk}, etc. For $b \in \mathbb{R}$ and $c \in \mathbb{R}^k$, we write $b + c$ to mean $b\mathbf{1}_k + c$.

2 Algorithm for Iteratively Constructing a CATE Estimator

We now present two algorithms for constructing a CATE estimator that performs well in terms of the Γ-maximal risk. These algorithms represent minimax optimization schemes that iteratively update the strategies of two players in a zero-sum game. One of these strategies corresponds to an estimator belonging to $\mathcal{T}$, and the other to a prior belonging to Γ. The estimator is repeatedly evaluated on simulated datasets and updated to perform as well as possible against the current prior, while the prior is also updated to be as unfavorable as possible for the current estimator. The proposed algorithms represent special cases of adversarial Monte Carlo meta-learning (AMC), which is a general approach for iteratively constructing (Γ-)minimax estimators introduced in Luedtke et al. (2020). This approach builds on earlier schemes developed by the statistics, econometrics, and machine learning communities — see the end of this chapter for a review.

The iterative schemes discussed in what follows will be easiest to implement when priors in Γ are easy to sample from. In this section, we focus on two types of collections Γ for which this is the case. In the first, the priors in Γ correspond to finite mixtures of m fixed priors $\Pi^{(1)}, \ldots, \Pi^{(m)}$, that is, is equal to $\Gamma_{\text{mix}} = \{\sum_{j=1}^m \alpha_j \Pi^{(j)} : \alpha \in \Delta_{m-1}\}$, where Δ_{m-1} denotes the $(m-1)$-simplex (Chamberlain, 2000). In the second, the priors in Γ are parameterized by a generator function G_g (Goodfellow et al., 2014), where g is a finite-dimensional index parameter that belongs to $\mathbb{R}^\ell$, $\ell < \infty$. Each G_g takes as input a source of noise U drawn from a user-specified distribution ν_u – such as a standard multivariate normal distribution – and outputs the parameters indexing a distribution in $\mathcal{P}$ (Luedtke et al., 2020). Though this form of sampling limits to parametric families $\mathcal{P}$, the number of parameters indexing this family may be much larger than the sample size n, which can, for all practical purposes, lead to a nonparametric estimation problem. For each g, we let Π_g denote the distribution of $G_g(U)$ when $U \sim \nu_u$. The collection Γ then takes

Algorithm 1 Adversarially learn a CATE estimator when Γ consists of mixtures of m priors.

Requires step sizes η_1, η_2 and exploration probability $\epsilon \in (0, 1)$
Initialize estimator T_t and mixture weights $\alpha = (\alpha_j)_{j=1}^m \in \Delta_{m-1}$
for K iterations **do**
  Explore with probability ϵ.
  If exploring **then** draw $M \sim \text{Uniform}\{1, 2, \ldots, m\}$.
  else draw $M \sim \text{Categorical}(\alpha)$. ▷ returns j with probability α_j.
  Let $P \sim \Pi^{(M)}$, $(X_i, A_i, Y_i)_{i=0}^n \overset{\text{iid}}{\sim} P$, and $\boldsymbol{D}$ be the dataset containing $(X_i, A_i, Y_i)_{i=1}^n$.
  Let $\text{Loss} = [T_t(\boldsymbol{D})(X_0) - \theta_P(X_0)]^2/\sigma_P^2$.
  Update estimator: $t = t - \eta_1 \nabla_t \text{Loss}$. ▷ Loss depends on t through T_t.
  Update M-th entry of prior mixture weight: $\alpha_M = \alpha_M + \eta_2 \frac{1-\epsilon}{(1-\epsilon)\alpha_M + \epsilon/m} \text{Loss}$.
  Project back to simplex: Replace α by its Euclidean projection onto Δ_{m-1}.
end for

the form $\Gamma_{\text{gen}} := \{\Pi_g : g \in \gamma\}$, where γ is a finite-dimensional index set.

The presented algorithms require that the class of estimators $\mathcal{T}$ is finite-dimensional, that is, that $\mathcal{T} = \{T_t : t \in \mathbb{R}^d\}$, where $d < \infty$. In our experiments, we focus on neural network classes for $\mathcal{T}$. More generally, we require that $t \mapsto T_t(\boldsymbol{d})(x_0)$ is differentiable for all datasets $\boldsymbol{d}$ and test points x_0. In the next section, we will describe a neural network architecture that satisfies certain group invariance properties that can be useful to impose on a class of CATE estimators.

Algorithm 1 provides an AMC scheme to adversarially construct a CATE estimator in cases where Γ consists of a finite mixture of m fixed priors ($\Gamma = \Gamma_{\text{mix}}$). This algorithm represents a variant of stochastic gradient descent ascent (Lin et al., 2019) that iteratively updates the parameters t and α indexing the estimator and prior, respectively, via unbiased estimates of the gradients of the Bayes risk function $(t, \alpha) \mapsto r(T_t, \Pi_\alpha)$ with respect to t and α, where T_t and Π_α denote the current estimator and prior, respectively. To update the estimator, this unbiased gradient estimate is obtained by differentiating the standardized squared-error loss $[T_t(\boldsymbol{D})(X_0) - \theta_P(X_0)]^2/\sigma_P^2$ in the parameter t for a randomly drawn distribution P, dataset $\boldsymbol{D}$, and test point X_0. To update the prior, this unbiased gradient estimate is instead obtained via a variant of the likelihood ratio method (Glynn, 1987). To account for the fact that the mixture weights α must belong to the simplex, the gradient updates to α are projected to the simplex Δ_{m-1}, so that the update to α corresponds to a projected stochastic gradient ascent step (Nemirovski et al., 2009). All gradients used in this and the upcoming algorithm can be computed via backpropagation as implemented in standard software packages such as TensorFlow or Pytorch (Abadi et al., 2015; Paszke et al., 2019).

We now provide an algorithm for cases where the priors in Γ are parameterized via a generator function ($\Gamma = \Gamma_{\text{gen}}$). To simplify the presentation, we will suppose that the propensity function π_P is known when considering these cases, which makes it so that $\pi_P = \pi_{P'}$ for all possible distributions $P, P' \in \mathcal{P}$; such a condition is plausible, for example, in a randomized experiment. Algorithm 2 provides an implementation of AMC that iteratively updates the parameter g indexing the generator function G_g and the parameter t indexing the estimator T_t via stochastic gradient descent ascent (Lin et al., 2019). Implementing this algorithm requires the ability to differentiate realized datasets through the parameters indexing the prior. To ensure

Algorithm 2 Adversarially learn a CATE estimator when prior is parameterized as a generator.

Requires step sizes η_1, η_2
Initialize estimator T_t and generator G_g
for K iterations **do**
  Independently draw $U \sim \nu_u$ and $V_0, V_1, \ldots, V_p \overset{\text{iid}}{\sim} \nu_v$.
  Let $P = G_g(U)$, $(X_i, A_i, Y_i) = H_P(V_i)$, $i = 0, 1, \ldots, n$, and $\boldsymbol{D} = (X_i, A_i, Y_i)_{i=1}^n$.
  Let Loss $= [T_t(\boldsymbol{D})(X_0) - \theta_P(X_0)]^2/\sigma_P^2$.
  Update estimator: $t = t - \eta_1 \nabla_t \text{Loss}$. ▷ Loss depends on t through T_t.
  Update prior: $g = g + \eta_2 \nabla_g \text{Loss}$. ▷ Loss depends on g through P, X_0, and $\boldsymbol{D}$.
end for

that this is possible, for each $P \in \mathcal{P}$, the algorithm requires access to a generator function $H_P : \mathcal{V} \to \mathcal{X} \times \mathcal{A} \times \mathcal{Y}$ such that $H_P(V)$ has the same distribution as $(X, A, Y) \sim P$ when noise V is drawn from a user-specified distribution ν_v with support on $\mathcal{V}$. The algorithm further requires that $g \mapsto H_{G_g(u)}(v)$ is differentiable at g_0 for all realizations of the noise u in the support of ν_u and v in the support of ν_v. An example of a setting where such a function H_P can be defined is provided at the end of this section. Note that there is no reason to expect that $g \mapsto H_{G_g(u)}(v)$ will be everywhere differentiable if the support of X or Y is discrete — consequently, Algorithm 2 will not apply in those settings. This problem can be avoided by modifying the algorithm to instead use the likelihood ratio method to obtain an unbiased gradient estimate (Glynn, 1987). The likelihood ratio method could similarly be used to obtain an unbiased gradient estimate in cases where the propensity function π_P is not known.

Many variants of Algorithms 1 and 2 are possible. For example, rather than computing the loss on a single dataset, a batch of multiple datasets can be sampled, and the loss can correspond to the average loss across all datasets in this batch. As another example, two-timescale learning rates could be used so that η_1 and η_2 both converge to zero as the number of iterations increases, but η_2 converges to zero at a faster rate than does η_1, that is, $\eta_2/\eta_1 \to 0$ as the number of iterations increases. Such two-timescale learning rate strategies have proven to be effective in stabilizing the optimization problem pursued by generative adversarial networks (Heusel et al., 2017). As a final example, an alternative first-order minimax optimization scheme could be employed, such as a stochastic extragradient method (Mishchenko et al., 2020).

We now provide an example of a setting where the generator functions G_g and H_P needed to implement Algorithm 2 can be defined so that $g \mapsto H_{G_g(u)}(v)$ is differentiable. Suppose that the model $\mathcal{P}$ consists of all distributions P for which there exist $(\lambda_P, \beta_P) \in (0, \infty) \times \mathbb{R}^4$ such that $X \sim \text{Exponential}(\lambda_P)$, $A|X \sim \text{Bernoulli}(1/2)$, and $Y|A, X \sim N(\beta_P^\top(1, A, X, AX), 1)$. Further suppose that the generator G_g is a multilayer perceptron with weights g mapping from $\mathbb{R}^k$ to $(0, \infty) \times \mathbb{R}^4$ and that the activation functions used to define this neural network are differentiable. Let ν_u be a $N(\mathbf{0}_k, \text{Id}_k)$ distribution. A draw from a prior Π_g can be obtained by sampling $U \sim \nu_u$ and letting $(\lambda_P, \beta_P) = G_g(U)$; the value of the parameters (λ_P, β_P) then fully determines the value of $P \in \mathcal{P}$. We now provide the form of a generator H_P and a noise distribution ν_v such that $H_P(V)$ has distribution P when $V \sim \nu_v$. In particular, let $V = (V_1, V_2, V_3)$ be a vectored-valued random variable whose three entries consist of mutually independent draws from a standard uniform distribution (V_1), Bernoulli distribution with

success probability 1/2 (V_2), and standard normal distribution (V_3). Let H_P be the function mapping from (v_1, v_2, v_3) to $(f_{P,1}(v_1), v_2, f_{P,3}(v_1, v_2, v_3))$, where $f_{P,1}(v_1) = -\log(1 - v_1)/\lambda_P$ is the inverse cumulative distribution function of an Exponential(λ) distribution evaluated at v_1 and $f_{P,3}(v_1, v_2, v_3) = \beta_P^\top(1, v_2, f_1(v_1), v_2 f_1(v_1)) + v_3$. By the chain rule, it can be verified that, for any $u \in \mathbb{R}^k$ and $v \in (0,1) \times \{0,1\} \times \mathbb{R}$, $g \mapsto H_{G_g(u)}(v)$ is a differentiable function.

3 Class of Estimators

3.1 Desired Invariance Properties

Before introducing the class of estimators that we propose using when implementing Algorithms 1 and 2, we describe several invariance properties that we will ensure that the estimators it contains satisfy. To do this, we will need to introduce some notation. For $k \in \mathbb{N}$, we let $\mathcal{B}_k$ denote the collection of all $k \times k$ permutation matrices and $\mathcal{D}_0 := \{(\boldsymbol{d}, x_0) \in \mathcal{D} \times \mathcal{X} : s(\boldsymbol{y}) \neq 0, s(\boldsymbol{x})_j \neq 0 \, \forall \, j = 1, 2, \ldots, p\}$. When X and Y are continuous random variables under sampling from P, a dataset-test-point pair $(\boldsymbol{D}, X_0)$ that is sampled according to P belongs to $\mathcal{D}_0$ with probability one. With this notation in hand, we are now in a position to state the invariance properties that we would like all estimators in $\mathcal{T}$ to satisfy. In particular, we want that, for all $T \in \mathcal{T}$, the following properties are satisfied for all $(\boldsymbol{d}, x_0) \in \mathcal{D}_0$:

T1) *Invariant to permutations of the observations:* For all $B \in \mathcal{B}_n$, $T(B\boldsymbol{x}, B\boldsymbol{a}, B\boldsymbol{y})(x_0) = T(\boldsymbol{d})(x_0)$.

T2) *Invariant to permutations of the covariates:* For all $B \in \mathcal{B}_p$, $T(\boldsymbol{x}B, \boldsymbol{a}, \boldsymbol{y})(B^\top x_0) = T(\boldsymbol{d})(x_0)$.

T3) *Invariant to increasing affine transformations of the covariates:* For all $b \in \mathbb{R}^p$ and $c \in (0, \infty)^p$, $T(\boldsymbol{x}^{b,c}, \boldsymbol{a}, \boldsymbol{y})(b + c \odot x_0) = T(\boldsymbol{d})(x_0)$, where $\boldsymbol{x}^{a,b}$ is the $n \times p$ matrix with row i equal to $a + b \odot \boldsymbol{x}_{i*}$.

T4) *Invariant to shifts of the outcomes:* For all $b \in \mathbb{R}$, $T(\boldsymbol{x}, \boldsymbol{a}, b + \boldsymbol{y})(x_0) = T(\boldsymbol{d})(x_0)$.

T5) *Equivariant to increasing linear transformations of the outcomes:* For all $c > 0$, $T(\boldsymbol{x}, \boldsymbol{a}, c\boldsymbol{y})(x_0) = cT(\boldsymbol{d})(x_0)$.

In the context of regression problems, Luedtke et al. (2021) motivated similar conditions (Eqs. 4 and 5 from that work) by showing that the corresponding decision problem is invariant under the action of an appropriately defined group (Lemmas 10 and 11 from that work). They then derived a variant of the Hunt-Stein theorem (Hunt and Stein, 1946) to establish that, for many natural collections of priors Γ, there exists an invariant Γ-minimax regression estimator. Though these arguments could be directly modified to establish similar results in CATE estimation settings, doing so is beyond the scope of this work. Instead, a hybrid approach is pursued in what follows: the rationale for imposing T1 and T2 is formally justified via Jensen's inequality, while an informal, but intuitive, rationale is provided for imposing T3-T5.

Our rationale for imposing T3-T5 is based on several properties of conditional expectations. We begin by motivating T4 and T5. Fix $(b, c) \in \mathbb{R} \times (0, \infty)$. By the linearity of expectation, the CATE $\theta_P(x_0) := E_P[Y|A = 1, X = x_0] - E_P[Y|A = 0, X = x_0]$ of A on an outcome Y can be

related to the CATE $\tilde{\theta}_P(x_0) := E_P[\tilde{Y}|A=1, X=x_0] - E_P[\tilde{Y}|A=0, X=x_0]$ of A on the linear transformation $\tilde{Y} := b + cY$ of this outcome as follows:

$$\begin{aligned}\tilde{\theta}_P(x_0) &= E_P[b+cY|A=1, X=x_0] - E_P[b+cY|A=0, X=x_0] \\ &= c(E_P[Y|A=1, X=x_0] - E_P[Y|A=0, X=x_0]) = c\,\theta_P(x_0).\end{aligned}$$

This suggests that an estimate $T(\boldsymbol{x}, \boldsymbol{a}, b + c\boldsymbol{y})$ of $\tilde{\theta}_P$ should be equal to c times the estimate $T(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})$ of θ_P, which is precisely what T4 and T5 require when taken in combination. In principle we could ask for even stronger version of T5, namely $T(\boldsymbol{x}, \boldsymbol{a}, c\boldsymbol{y}) = cT(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{y})$ for all $c \in \mathbb{R}$, rather than just for $c > 0$. Because the estimators that we present later in this work do not satisfy this stronger invariance property, we do not comment on it further here. We now motivate T3. Fix $(b, c) \in \mathbb{R}^p \times (0, \infty)^p$. By the injectivity of the map $x \mapsto b + c \odot x$, the CATE $\theta_P(x_0) := E_P[Y|A=1, X=x_0] - E_P[Y|A=0, X=x_0]$ of A on an outcome Y can be related to the CATE $\bar{\theta}_P(\bar{x}_0) := E_P[Y|A=1, \bar{X}=\bar{x}_0] - E_P[Y|A=0, \bar{X}=\bar{x}_0]$ of A based on the transformed covariates $\bar{X} := b + c \odot X$ being equal to $\bar{x}_0 := b + c \odot x_0$ as follows:

$$\begin{aligned}\bar{\theta}_P(\bar{x}_0) &= E_P[Y|A=1, \bar{X}=\bar{x}_0] - E_P[Y|A=0, \bar{X}=\bar{x}_0] \\ &= E_P[Y|A=1, X=x_0] - E_P[Y|A=0, X=x_0] = \theta_P(x_0).\end{aligned}$$

This suggests that estimates of $\bar{\theta}_P(\bar{x}_0)$ based on a given dataset should coincide with those of $\theta_P(x_0)$ based on that same dataset, which is exactly what T3 requires.

We now provide justification for imposing T1. This condition will be seen to be reasonable in light of the fact that the observations are independent and identically distributed, and therefore exchangeable. This exchangeability can be used to show that, for any estimator T, there exists an invariant estimator T_1 that uniformly performs at least as well as T in the sense that $R(T_1, P) \leq R(T, P)$ for all $P \in \mathcal{P}$. As a consequence of this, T_1 achieves a Γ-minimax risk that is at least as small as that of T. By Jensen's inequality and the convexity of $R(\cdot, P)$ for each P, T_1 can be chosen to be equal to $T_1(\boldsymbol{d})(x_0) = (n!)^{-1} \sum_{B \in \mathcal{B}_n} T(B\boldsymbol{x}, B\boldsymbol{a}, B\boldsymbol{y})(x_0)$, which corresponds to the average of the evaluations of T across permutations of the n observations in $\boldsymbol{d}$. In fact, under slightly stronger conditions that make it so that Jensen's inequality is strict (details omitted), T_1 can be shown to dominate T in the sense that $R(T_1, P)$ is strictly smaller than $R(T, P)$ for one or more $P \in \mathcal{P}$. Imposing that all estimators in $\mathcal{T}$ satisfy T1 makes it so that $T_1 = T$ for all $T \in \mathcal{T}$, so that there is no estimator in $\mathcal{T}$ that can be dominated by simply averaging across its evaluations on permutations of the observations.

In cases where nothing is known *a priori* about the ordering of the different entries of the covariate X, Jensen's inequality can similarly be used to justify T2. In particular, in what follows we consider cases where Γ is agnostic to the ordering of the covariates in the sense that $\Pi \in \Gamma$ implies that $\Pi \circ f_B^{-1} \in \Gamma$ for all $B \in \mathcal{B}_p$, where $f_B(P)$ is the pushforward measure $P \circ g_B^{-1}$ with $g_B(x, a, y) = (B^\top x, a, y)$. In such cases, for any $T \in \mathcal{T}$, it will be shown that there exists an estimator T_2 that performs at least as well as T in terms of Γ-maximal risk, in the sense that

$$\sup_{\Pi \in \Gamma} r(T_2, \Pi) \leq \sup_{\Pi \in \Gamma} r(T, \Pi). \tag{2}$$

The estimator T_2 can be evaluated by averaging the estimates obtained from T across permutations of the covariates. In particular, T_2 takes the form $T_2(\boldsymbol{d})(x_0) = (p!)^{-1} \sum_{B \in \mathcal{B}_p} T_B(\boldsymbol{d})(x_0)$, where

$T_B(\boldsymbol{d})(x_0) = T(\boldsymbol{x}B, \boldsymbol{a}, \boldsymbol{y})(B^\top x_0)$. To see this, let $h_B(x) := B^\top x$ for $B \in \mathcal{B}_p$ and note that, for any $P \in \mathcal{P}$, Jensen's inequality and the change of variables formula for pushforward measures show that

$$R(T_2, P) \leq \frac{1}{p!} \sum_{B \in \mathcal{B}_p} R(T_B, P) = \frac{1}{p!} \sum_{B \in \mathcal{B}_p} \mathrm{E}_{P \circ g_B^{-1}} \left[\int \frac{[T(\boldsymbol{D})(x_0) - \theta_P(Bx_0)]^2}{\sigma_P^2} dP_X \circ h_B^{-1}(x_0) \right].$$

Combining the above with the fact that $\theta_P(Bx_0) = \theta_{P \circ g_B^{-1}}(x_0)$, which holds since $E_P[Y|A = a, X = Bx_0] = E_P[Y|A = a, B^\top X = x_0]$ for $a \in \{0, 1\}$, we see that $R(T_2, P) \leq (p!)^{-1} \sum_{B \in \mathcal{B}_p} R(T, P \circ g_B^{-1})$. Integrating P against Π on both sides and noting that $\int R(T, P \circ g_B^{-1}) \Pi(dP) = r(T, \Pi \circ f_B^{-1})$ then shows that

$$r(T_2, \Pi) \leq \frac{1}{p!} \sum_{B \in \mathcal{B}_p} r(T, \Pi \circ f_B^{-1}) \leq \max_{B \in \mathcal{B}_p} r(T, \Pi \circ f_B^{-1}) \leq \sup_{\Pi' \in \Gamma} r(T, \Pi'),$$

where the final inequality used that $\Pi \circ f_B^{-1}$ is contained in Γ. As Π was an arbitrary element of Γ, taking a supremum in Π on the left-hand side shows that (2) holds. Imposing that all estimators in $\mathcal{T}$ satisfy T2 makes it so that $T_2 = T$ for all $T \in \mathcal{T}$, so that there is no estimator in $\mathcal{T}$ that can be dominated by simply averaging across its evaluations on permutations of the covariates.

3.2 Proposed Class of Estimators

Algorithm 3 presents the class of CATE estimators $\mathcal{T}$ that we propose to use when employing Algorithm 1 or 2. This class is the same as the one introduced in Luedtke et al. (2021) in the context of regression problems — namely, in the estimation of $x \mapsto E[Y \mid X = x]$ — except has been adapted to include treatment assignments $A_1, \ldots, A_n$ as inputs to the estimator and to account for the fact that CATE estimators should be invariant, rather than equivariant, to shifts in the outcome. Algorithm 3 relies on four modules. Each module m_k, $k \in \{1, 2, 3, 4\}$, can be represented by a function m_k belonging to a collection $\mathcal{M}_k$ of functions mapping from $\mathbb{R}^{b_k}$ to $\mathbb{R}^{c_k}$, where the values of b_k and c_k can be deduced from the statement of the algorithm. For given data $\boldsymbol{d}$, a prediction at a covariate x_0 can be obtained sequentially calling the modules and, between calls, either mean pooling across one of the dimensions of the output or concatenating the evaluation point as a new column in the output matrix.

We let $\mathcal{T}_{\mathcal{M}}$ represent the collection of all CATE estimators described by Algorithm 3, where here $(m_k)_{k=1}^4$ varies over $\prod_{k=1}^4 \mathcal{M}_k$. Estimators in $\mathcal{T}_{\mathcal{M}}$ can be seen to automatically satisfy T3-T5. Indeed, by virtue of the standardization of the covariates on line 1 of Algorithm 3, conditions T3 and T4 are satisfied. Furthermore, by this standardization along with the rescaling of the output by the empirical standard deviation on line 9, T5 is also satisfied. Provided the module classes $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ satisfy certain sufficient conditions, estimators in $\mathcal{T}_{\mathcal{M}}$ also satisfy T1 and T2. These conditions are as follows:

M1) $m_1(BvC)_{**\ell} = B[m_1(v)_{**\ell}]C$ for all $m_1 \in \mathcal{M}_1$, $B \in \mathcal{B}_n$, $C \in \mathcal{B}_p$, $v \in \mathbb{R}^{n \times p \times 3}$, and $\ell \in \{1, \ldots, o_1\}$.

M2) $m_2(Bv) = Bm_2(v)$ for all $m_2 \in \mathcal{M}_2$, $B \in \mathcal{B}_p$, and $v \in \mathbb{R}^{p \times o_1}$.

Algorithm 3 Use data $\boldsymbol{d}$ to obtain CATE estimate at x_0.

Preprocess: Let $x_0^0 := \frac{x_0 - \bar{\boldsymbol{x}}}{s(\boldsymbol{x})}$ and define $\boldsymbol{d}^0 \in \mathbb{R}^{n \times p \times 2}$ so that $\boldsymbol{d}^0_{i*1} = \frac{x_i - \bar{\boldsymbol{x}}}{s(\boldsymbol{x})}$ for all $i = 1, \ldots, n$ and, for all $j = 1, \ldots, p$, $\boldsymbol{d}^0_{*j2} = \boldsymbol{a}$ and $\boldsymbol{d}^0_{*j3} = \frac{\boldsymbol{y} - \bar{\boldsymbol{y}}}{s(\boldsymbol{y})}$.

▷ Process observed dataset $\boldsymbol{d}$.

Module 1: $\boldsymbol{d}^1 := m_1(\boldsymbol{d}^0)$. $\quad \boldsymbol{d}^1 \in \mathbb{R}^{n \times p \times o_1}$

Mean Pool: $\bar{\boldsymbol{d}}^1 := n^{-1} \sum_{i=1}^n \boldsymbol{d}^1_{i**}$.

Module 2: $\boldsymbol{d}^2 := m_2(\bar{\boldsymbol{d}}^1)$. $\quad \boldsymbol{d}^2 \in \mathbb{R}^{p \times o_2}$

▷ Generate and return a CATE prediction at a test point x_0.

Augment: $\tilde{\boldsymbol{d}}^2 := [\boldsymbol{d}^2 \mid x_0^0]$. $\quad \tilde{\boldsymbol{d}}^2 \in \mathbb{R}^{p \times (o_2+1)}$

Module 3: $\boldsymbol{d}^3 := m_3(\tilde{\boldsymbol{d}}^2)$. $\quad \boldsymbol{d}^3 \in \mathbb{R}^{p \times o_3}$

Mean Pool: $\bar{\boldsymbol{d}}^3 := p^{-1} \sum_{j=1}^p \boldsymbol{d}^3_{j*}$. $\quad \bar{\boldsymbol{d}}^3 \in \mathbb{R}^{o_3}$

Module 4: $\boldsymbol{d}^4 := m_4(\bar{\boldsymbol{d}}^3)$. $\quad \boldsymbol{d}^4 \in \mathbb{R}$

return $s(\boldsymbol{y})\boldsymbol{d}^4$.

M3) $m_3(Bv) = Bm_3(v)$ for all $m_3 \in \mathcal{M}_3$, $B \in \mathcal{B}_p$, and $v \in \mathbb{R}^{p \times (o_2+1)}$.

Many different choices of the module classes $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ satisfy the above conditions. Here, we focus on particular neural network classes that satisfy these properties.

There are several reasons that neural network classes provide appealing choices for these module classes. First, readily available software frameworks (e.g., Paszke et al., 2019; Abadi et al., 2015) make it easy to implement Algorithms 1 and 2 when the estimators are parameterized as a neural network class. Second, in a variety of settings, sufficiently wide or deep neural networks have been shown to be able to approximate any smooth function arbitrarily well (e.g., Cybenko, 1989; Hornik, 1991; Maron et al., 2019). Working over a rich class of estimators should, in principle, yield some estimators with better Γ-maximal risk than would be available within a smaller class. Third, there are well-established neural network architectures that can be used to parameterize the module classes $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M}_3$ so that they satisfy the invariance properties M1-M3. One possible drawback to making these module classes neural network classes is that the resulting minimax optimization problems that Algorithms 1 and 2 seek to solve are not convex-concave. Nevertheless, in our experiments, we show that the proposed approach appears to yield high-performing estimators despite this difficulty. This is consistent with findings in other minimax optimization problems where neural networks are employed, such as those arising in the optimization of generative adversarial networks (Goodfellow et al., 2014; Arjovsky et al., 2017).

The neural network classes used for the modules are parameterized in the same way as was done for regression problems in Luedtke et al. (2021). To orient readers, their presentation is organized in the same way here. For each k, $\mathcal{M}_k$ contains the neural networks consisting of h_k hidden layers of widths $w_k^1, w_k^2, \ldots, w_k^{h_k}$, where the types of layers used depends on the module k. When $k = 1$, multi-input-output channel equivariant layers as defined in Hartford et al. (2018) are used. For $j = 1, \ldots, h_1 + 1$, $\mathcal{L}_1^j$ denotes the collection of all such layers that map from $\mathbb{R}^{n \times p \times w_1^{j-1}}$ to $\mathbb{R}^{n \times p \times w_1^j}$, where $w_1^0 = 3$ and $w_1^{h_1+1} = o_1$. For each j, each member L_1^j of $\mathcal{L}_1^j$ is equivariant in the sense that, for all $B \in \mathcal{B}_n$, $C \in \mathcal{B}_p$, and $v \in \mathbb{R}^{n \times p \times w_1^{j-1}}$, $L_1^j(BvC)_{**\ell} = BL_1^j(v)_{**\ell}C$ for

all $\ell = 1, \ldots, o_1$. When $k = 2, 3$, multi-input-output channel equivariant layers as described in Eq. 22 of Zaheer et al. (2017) are used, except that the sum-pool term in that equation is replaced with a mean-pool term. For $j = 1, \ldots, h_k + 1$, the collection of all such equivariant layers that map from $\mathbb{R}^{p \times w_k^{j-1}}$ to $\mathbb{R}^{p \times w_k^j}$ is denoted by $\mathcal{L}_k^j$. For each j, each member L_k^j of $\mathcal{L}_k^j$ is equivariant in the sense that, for all $B \in \mathcal{B}_p$ and $v \in \mathbb{R}^{p \times w_k^{j-1}}$, $L_k^j(Bv) = BL_k^j(v)$. When $k = 4$, standard linear layers mapping from $\mathbb{R}^{w_4^{j-1}}$ to $\mathbb{R}^{w_4^j}$ are used for each $j = 1, \ldots, h_4 + 1$, where $w_4^0 = o_3$ and $w_4^{h_4+1} = 1$. For each j, the collection of all such layers is denoted by $\mathcal{L}_4^j$. For a user-specified activation function q, the module classes are defined as follows for $k = 1, 2, 3, 4$:

$$\mathcal{M}_k := \{v \mapsto q \circ L_k^{h_k+1} \circ q \circ L_k^{h_k} \circ \ldots \circ q \circ L_k^1(v) : L_k^j \in \mathcal{L}_k^j, j = 1, 2, \ldots, h_k + 1\}.$$

Notably, $\mathcal{M}_1$ satisfies M1 (Ravanbakhsh et al., 2017; Hartford et al., 2018), and $\mathcal{M}_2$ and $\mathcal{M}_3$ satisfy M2 and M3, respectively (Ravanbakhsh et al., 2016; Zaheer et al., 2017). Each element of $\mathcal{M}_4$ is a multilayer perceptron.

The proposed class of estimators satisfies several nice properties. The first is computational in nature. In particular, if the estimated CATE function will be evaluated at several different values of the test point x_0, then the computational cost can be amortized across these runs by only processing the data on lines 2-4 of Algorithm 3 a single time. A prediction at a new test point x_0 can then be evaluated by evaluating lines 5-9. When there are many test points at which to obtain CATE estimates, this can lead to meaningful savings in computational time relative to rerunning the entirety of Algorithm 3 for every new test point. Another advantage of the proposed class of estimators is that its members can be evaluated on datasets that have a different sample size than did the datasets used during meta-training. In the notation of Eq. 4 from Hartford et al., this corresponds to noting that the weights from an $\mathbb{R}^{N \times M \times k} \to \mathbb{R}^{N \times M \times o}$ multi-input-output channel layer can be used to define an $\mathbb{R}^{N' \times M \times k} \to \mathbb{R}^{N' \times M \times o}$ layer for which the output $Y_{n,m}^{\langle o \rangle}$ is given by the same symbolic expression as that displayed in Eq. 4 from that work, but now with n ranging over $1, \ldots, N'$. In the context of regression, Luedtke et al. (2021) showed that estimators trained using 500 observations can perform well even when evaluated on datasets containing only 100 observations. It is similarly possible to evaluate the proposed architecture on datasets containing a different number of covariates than did the datasets used during meta-training — again see Eq. 4 in Hartford et al. (2018), and also see Eq. 22 in Zaheer et al. (2017), but with the sum-pool term replaced by a mean-pool term. Taken together, these observations make it so that an estimator learned for a setting where n observations and p covariates are observed can be evaluated without modification on datasets with $n' \neq n$ observations and $p' \neq p$ covariates. When $n' \approx n$ and $p' \approx p$, it is reasonable to expect that the learned estimator will perform about as well as would have an estimator constructed explicitly for the case where the sample size is n' and p' covariates are observed.

4 Numerical Experiments

The performance of the proposed AMC approach was evaluated via a numerical experiment similar to experiments that Luedtke et al. (2021) used to evaluate AMC in the context of regression problems. In these experiments, the model $\mathcal{P}$ consisted of distributions for which there exists a vector $\upsilon \in \mathbb{R}^p$, a positive definite matrix $\Sigma \in \mathbb{R}^{p \times p}$, a p-variate distribution $Q_{\boldsymbol{\upsilon}, \Sigma}$

with the same copula as a $N(\boldsymbol{v}, \Sigma)$ distribution, a variance $\sigma^2 > 0$, and an outcome regression $\mu : \mathcal{A} \times \mathcal{X} \to \mathbb{R}$ belonging to a class $\mathcal{R}_{m,s}$ that will be defined shortly such that

$$X \sim Q_{\boldsymbol{v},\Sigma}, \qquad A \mid X \sim \text{Bernoulli}(1/2), \qquad Y \mid A, X \sim N[\sigma \cdot \mu(A, X), \sigma^2].$$

The class $\mathcal{R}_{m,s}$ is indexed by $m \in (0, \infty)$ and $s \in \{1, 2, \ldots, p\}$. The definition of this class relies on the notion of the total variation a function. Recall that the total variation of $f : \mathbb{R} \to \mathbb{R}$, denoted by $V(f)$, is equal to the supremum of $\sum_{\ell=1}^{k} |f(b_{\ell+1}) - f(b_\ell)|$ over all $(b_\ell)_{\ell=1}^{k+1}$ such that $k \in \mathbb{N}$ and $b_1 < b_2 < \ldots < b_{k+1}$ (Cohn, 2013). Writing x_j to denote the j-th covariate, the model we considered imposes that μ falls in

$$\mathcal{R} := \left\{ (a, x) \mapsto \sum_{j=1}^{p} \mu_j(a, x_j) \ : \ \max_{a' \in \mathcal{A}} \|v_{a'}(\mu)\|_1 \leq m, \ \|v_0(\mu) \odot v_1(\mu)\|_0 \leq s \right\},$$

where $v_{a'}(\mu) := (V(\mu_j(a', \cdot)))_{j=1}^{p}$, $\|\cdot\|_1$ denotes the ℓ^1 norm, and $\|\cdot\|_0$ returns the number of non-zero entries of its $\mathbb{R}^p$-valued argument. The above class $\mathcal{R}$ enforces that the treatment-specific outcome regressions $x \mapsto \sigma \cdot \mu(a, x)$, $a \in \mathcal{A}$, should belong to a variant of the fused lasso additive regression model (FLAM) and, together, these functions should rely on no more than s of the covariates (Petersen et al., 2016).

In our experiments, we fix $(n, p, m, s) = (500, 10, 10, 4)$. The collection of priors Γ considered was the same as that presented in Appendices D.2.1 and D.2.3 of Luedtke et al. (2021) in the context of regression, but with two modifications to adapt to our CATE estimation setting. First, a draw from a distribution P in the support of Γ corresponded to a covariate-treatment-outcome tuple (X, A, Y), rather than a covariate-outcome tuple (X, Y). As specified by the model, all $P \in \mathcal{P}$ are such that $A \mid X \sim \text{Bernoulli}(1/2)$ for all $P \in \mathcal{P}$. Second, the generator network used to sample from a given prior returns two regression functions, corresponding to $x \mapsto E_P[Y|A = 0, X = x]$ and $x \mapsto E_P[Y|A = 1, X = x]$, from a given prior rather than only the single regression function $x \mapsto E_P[Y|X = x]$. These two regression functions were sampled independently of one another from the same generator network as described in Appendix D.2.3 of Luedtke et al. (2021). By setting $s = 4$, we imitate the "dense" FLAM setting reported in Luedtke et al. (2021). Following Luedtke et al. (2021), we preprocessed the covariates before supplying them to the estimator. In particular, we replaced each entry with its rank statistic among the n observations so that, for each $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, p\}$, we replaced $\boldsymbol{x}_{ij}$ by $\sum_{k=1}^{n} I\{\boldsymbol{x}_{ij} \geq \boldsymbol{x}_{kj}\}$ and x_{0j} by $\sum_{k=1}^{n} I\{x_{0j} \geq \boldsymbol{x}_{kj}\}$. As each prior in Γ is parameterized via a generator function, Algorithm 2 was used to construct the estimator. This estimator was constructed over approximately 3 million iterations, where each stochastic gradient update to the prior and estimator was based on 100 datasets. Consequently, the final constructed estimator, hereafter referred to as the AMC estimator, was iteratively updated based on its performance on approximately 300 million datasets. Further details of the implementation used in these experiments, along with the neural network weights used by the constructed estimator, can be found at the following GitHub repository: `https://github.com/alexluedtke12/amc-meta-learning-of-optimal-cate-estimators`.

Performance of the AMC estimator was evaluated using a simulation study. Four scenarios were considered. In all of these scenarios, X consists of covariates $X_1, X_2, \ldots, X_{10}$ that are independently drawn from a Uniform$[-1, 1]$ distribution. The treatment A is generated independently of X according to a Bernoulli$(1/2)$ distribution, and, conditionally on A and X, the

	Scenario			
	1	2	3	4
Linear Model	**0.092**	**0.117**	0.655	4.640
FLAM	0.403	0.423	0.545	**3.601**
Causal Forests	0.169	0.144	0.435	4.208
AMC (ours)	0.156	0.148	**0.316**	3.823

Table 1: MSEs of four estimators of the CATE across the four simulation scenarios.

outcome Y is normally distributed with mean $(2A-1)\theta(X)/2$ and variance 1, where the value of the CATE θ varies across the four scenarios. In particular, $\theta(x) = 0.884(1 - x_1 - x_2)$ in scenario 1, $\theta(x) = 2(x_2 - 0.25x_1^2 - 1)$ in scenario 2, $\theta(x) = 2(0.5 - x_1^2 - x_2^2)(x_1^2 + x_2^2 - 0.3)$ in scenario 3, and $\theta(x) = 2(1 - x_1^3 + \exp[x_3^2 + x_5] + 0.6x_6 - [x_7 + x_8]^2)$ in scenario 4. Performance of AMC was evaluated via Monte Carlo simulation and compared to that of three existing methods. The first comparator corresponds to causal forests (Athey et al., 2019) as implemented in the `grf` package in R (Tibshirani et al., 2022), where all tuning parameters were selected via cross-validation. The latter two comparators correspond to simple plug-in estimators of the treatment-specific outcome regressions $x \mapsto E_P[Y|A = 1, X = x]$ and $x \mapsto E_P[Y|A = 0, X = x]$ whose difference defines the CATE. Such plug-in approaches were referred to as T-learners in Künzel et al. (2019). The first of these approaches estimates each treatment-specific outcome regression with a main terms linear regression, whereas the second estimates each of them using the fused lasso additive model as implemented in the `flam` package in R with tuning parameters selected via cross-validation (Petersen, 2018). The performance of all approaches was evaluated using 1000 Monte Carlo repetitions.

Table 1 reports the performance of the four approaches. The linear-model-based T-learner outperformed all other approaches for scenario 1, which is unsurprising given that the outcome regressions were truly linear in that setting. The AMC estimator outperformed the remaining two comparators in this setting. In the second scenario, the AMC estimator was again outperformed by the linear model, was slightly outperformed by causal forests, and outperformed FLAM. The strong performance of the linear model in scenario 2 may be due to the fact that the CATE is nearly a linear function of the covariates in that setting. In scenario 3, the AMC estimator outperformed all comparators, with the next best estimator being causal forests and the worst-performing estimator being the linear model. Finally, in scenario 4, the AMC estimator was only outperformed by FLAM.

Overall, AMC demonstrated a reasonable level of adaptivity in these scenarios, being able to perform relatively well both in the simpler scenarios 1 and 2 and the more complex scenarios 3 and 4. In future work, it would be interesting to explore this adaptivity further by using cross-validation to select between several AMC estimators, each constructed to be Γ-minimax against a different collection of priors Γ. For example, these different collections of priors may be indexed by different bound m on the total variation of the outcome regression μ or sparsity levels s. Alternatively, some of these collections of priors may enforce parsimony of the CATE function (e.g., linearity) while others may allow for a much more complex form (e.g., via a collection of Bayesian additive regression tree priors; Chipman et al., 2010; Hill, 2011).

5 Notes and Further Reading

This chapter belongs to a recent body of literature introducing AMC schemes for general decision problems (Luedtke et al., 2020; Qiu and Luedtke, 2020) and in a special case, namely regression settings (Luedtke et al., 2021). Owing to the similarity between CATE estimation and regression estimation, the AMC approach proposed described in this chapter is most closely related to the approach described in Luedtke et al. (2021). Therefore, to facilitate comparisons, we have organized the presentation of the method and results similarly.

More broadly, the proposed approach is related to several prior works from the statistics, econometrics, and machine learning communities. We begin by describing those in the statistics and econometrics literatures. In finite-dimensional models, early works showed that it is possible to numerically learn minimax rules (Nelson, 1966; Kempthorne, 1987) and, in settings where Γ consists of all priors that satisfy a finite number of generalized moment conditions, Γ-minimax rules (Noubiap and Seidel, 2001). Other works have studied the Γ-minimax case where Γ consists of priors that only place mass on a pre-specified finite set of distributions or priors, both for general decision problems (Chamberlain, 2000) and for constructing confidence intervals (Schafer and Stark, 2009). The collections Γ that we consider in Algorithm 1 are of this same form. In contrast to the earlier works that considered collections Γ of this type, we explicitly consider the equivariance properties that are desirable for an estimator to have in the context of CATE estimation. We moreover present a neural network class of estimators that has these properties and construct our estimator via a form of stochastic gradient descent ascent that is easy to implement efficiently using standard software.

The approach proposed in this work is also related to the meta-learning literature from the machine learning community (Schmidhuber, 1987; Thrun and Pratt, 1998; Hochreiter et al., 2001; Vilalta and Drissi, 2002). As noted in Luedtke et al. (2021), most existing works in that literature focus on the special case where $\Gamma = \{\Pi\}$ for some fixed, user-specified prior Π, so that the goal is to construct a Bayes estimator. To our knowledge, these works have not previously been developed in the context of CATE estimation. Nevertheless, there are a number of interesting meta-learning approaches that have been used in supervised learning settings (e.g., Vinyals et al., 2016; Ravi and Larochelle, 2017; Finn et al., 2017; Garnelo et al., 2018). Adapting these approaches to estimate the CATE for cases where $\Gamma = \{\Pi\}$ would be an interesting area for future work.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

S. Athey, J. Tibshirani, and S. Wager. Generalized random forests. *The Annals of Statistics*, 47 (2):1148–1178, 2019.

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

P. J. Bickel, C. A. Klaassen, P. J. Bickel, Y. Ritov, J. Klaassen, J. A. Wellner, and Y. Ritov. *Efficient and adaptive estimation for semiparametric models*, volume 4. Johns Hopkins University Press Baltimore, 1993.

G. Chamberlain. Econometric applications of maxmin expected utility. *Journal of Applied Econometrics*, 15(6):625–644, 2000.

H. A. Chipman, E. I. George, and R. E. McCulloch. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

D. L. Cohn. *Measure theory*. Springer, 2013.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

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*, pages 1126–1135. JMLR. org, 2017.

M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. A. Eslami. Conditional neural processes. In *International Conference on Machine Learning*, pages 1704–1713. PMLR, 2018.

P. W. Glynn. Likelilood ratio gradient estimation: an overview. In *Proceedings of the 19th conference on Winter simulation*, pages 366–375, 1987.

I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

J. Hartford, D. R. Graham, K. Leyton-Brown, and S. Ravanbakhsh. Deep models of interactions across sets. *arXiv preprint arXiv:1803.02879*, 2018.

M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in neural information processing systems*, pages 6626–6637, 2017.

J. L. Hill. Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, 2011.

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

K. Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4 (2):251–257, 1991.

G. Hunt and C. Stein. Most stringent tests of statistical hypotheses. *Unpublished manuscript*, 1946.

P. J. Kempthorne. Numerical specification of discrete least favorable prior distributions. *SIAM Journal on Scientific and Statistical Computing*, 8(2):171–184, 1987.

S. R. Künzel, J. S. Sekhon, P. J. Bickel, and B. Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the national academy of sciences*, 116 (10):4156–4165, 2019.

T. Lin, C. Jin, and M. I. Jordan. On gradient descent ascent for nonconvex-concave minimax problems. *arXiv preprint arXiv:1906.00331v6*, 2019.

A. Luedtke, M. Carone, N. R. Simon, and O. Sofrygin. Learning to learn from data: using deep adversarial learning to construct optimal statistical procedures. *Science Advances* (in press; available online late Feb or Mar 2020), 2020.

A. Luedtke, I. Chung, and O. Sofrygin. Adversarial monte carlo meta-learning of optimal prediction procedures. *Journal of Machine Learning Research*, 22(255):1–67, 2021.

A. R. Luedtke and M. J. van der Laan. Super-learning of an optimal dynamic treatment rule. *The international journal of biostatistics*, 12(1):305–332, 2016.

H. Maron, E. Fetaya, N. Segol, and Y. Lipman. On the universality of invariant networks. *arXiv preprint arXiv:1901.09342*, 2019.

K. Mishchenko, D. Kovalev, E. Shulgin, P. Richtárik, and Y. Malitsky. Revisiting stochastic extragradient. In *International Conference on Artificial Intelligence and Statistics*, pages 4573–4582. PMLR, 2020.

W. Nelson. Minimax solution of statistical decision problems by iteration. *The Annals of Mathematical Statistics*, pages 1643–1657, 1966.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

R. F. Noubiap and W. Seidel. An algorithm for calculating γ-minimax decision rules under generalized moment conditions. *The Annals of Statistics*, 29(4):1094–1116, 2001.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035, 2019.

A. Petersen. *flam: Fits Piecewise Constant Models with Data-Adaptive Knots*, 2018. URL `https://CRAN.R-project.org/package=flam`. R package version 3.2.

A. Petersen, D. Witten, and N. Simon. Fused lasso additive model. *Journal of Computational and Graphical Statistics*, 25(4):1005–1025, 2016.

H. Qiu and A. Luedtke. Leveraging vague prior information in general models via iteratively constructed gamma-minimax estimators. *arXiv preprint arXiv:2012.05465*, 2020.

S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016.

S. Ravanbakhsh, J. Schneider, and B. Poczos. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2892–2901. JMLR. org, 2017.

S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

C. M. Schafer and P. B. Stark. Constructing confidence regions of optimal expected size. *Journal of the American Statistical Association*, 104(487):1080–1089, 2009.

J. Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

S. Thrun and L. Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

J. Tibshirani, S. Athey, E. Sverdrup, and S. Wager. *grf: Generalized Random Forests*, 2022. URL `https://CRAN.R-project.org/package=grf`. R package version 2.1.0.

R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

O. Vinyals, C. Blundell, T. Lillicrap, and D. Wierstra. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.