

Appendix A Synthetic Data Study and Results

In this section, we are interested in understanding the performance of estimators under various sets of assumptions (A.5(a) to A.5(c)). In particular, we are interested in understanding the potential gains as (i) total number of units, n , increases, (ii) the dimensionality of X , denoted as p , increases and (iii) the log of the ratio of number of units in the auxiliary to primary dataset, $\log\left(\frac{P(S=1)}{P(S=0)}\right)$, increases. First, we discuss our data generative procedures, and then we present and discuss our results.

Data Generative Procedure. The data generation procedure (DGP) in this study is designed to simulate a complex causal structure. We begin by generating covariates $X = (X_1, X_2, \dots, X_p)$ from a multivariate normal distribution with zero mean and identity covariance matrix where p is the number of covariates. The binary study indicator S is then generated as a Bernoulli random variable, where the probability of assignment to the auxiliary study (i.e., $S = 1$) is $\Pr(S = 1|X) = \text{expit}(a_0 + a_1X_1 + a_2X_2)$, where $\text{expit}(x) = \frac{1}{1+e^{-x}}$. The treatment assignment T is also generated as a study and covariate-dependent Bernoulli variable $\Pr(T = 1|X, S) = (1 - S) \times 0.5 + S \times \text{expit}(\zeta_1X_1)$. The auxiliary outcome W , observed only in the auxiliary study ($S = 1$), is defined as follows:

$$W = \mu_W(X, T, S) + \delta = (\gamma_0 + \gamma_1X_1 + \gamma_2X_2) \cdot T + \beta_1X_1 + \beta_2X_2 + \beta_3X_3 + \beta_0 + \delta,$$

where vectors γ and β define the treatment and baseline effects on W . This equation includes both linear and interaction terms, capturing treatment-covariate dependencies. In the primary study (where $S = 0$), the primary outcome Y is modeled as: $Y = \alpha(X) \cdot \mu_W(X, T, S) + \gamma$ where $\alpha(X) = \rho_1X_1 + \rho_0$. This outcome depends on the treatment effect modulated by covariate-driven heterogeneity in $\alpha(X)$, capturing treatment-mediated effects of covariates on Y .

Here the true ATE in primary is given as $\theta_0 = \mathbb{E}[(\rho_0 + \rho_1X_1) \cdot (\gamma_0 + \gamma_1X_1 + \gamma_2X_2) | S = 0]$.

Analysis and Results. We use mean-squared error (MSE) to compare the performance of the following three estimators: (i) efficient estimator only using primary data ($\hat{\theta}_0$), (ii) efficient estimator augmented with auxiliary score ($\hat{\theta}_b$) and (iii) efficient estimator with known α integrating auxiliary data ($\hat{\theta}_a$). The simulation results are compiled in Figure 2. As expected, the performance of all three estimators improves as n increases and deteriorates as p increases. Further, $\hat{\theta}_a$ dominates $\hat{\theta}_0$ and $\hat{\theta}_b$ especially for scenarios with large p and/or large $\log\left(\frac{P(S=1)}{P(S=0)}\right)$ – indicating that in scenarios where the primary study is relatively small and the problem is high-dimensional leveraging auxiliary data yields more benefits. This aligns with our theoretical results showing that knowing α can yield efficiency gains. For, $\hat{\theta}_b$ (which uses auxiliary data), we observe that it yields benefits relative to $\hat{\theta}_0$ in small n scenarios especially when p and $\log\left(\frac{P(S=1)}{P(S=0)}\right)$ are large. However, these benefits diminish relative to $\hat{\theta}_0$ as n grows. This is consistent with our theoretical result showing that there are no asymptotic benefits if α is unknown. However, there are some finite sample benefits of using the auxiliary score even when α is unknown.

Appendix B Efficiency Score Functions Derivation (Theorems 1–4)

Following the above mentioned procedure we derive the EIFs and corresponding efficiency bounds under the three sets of assumptions. As $\psi^*(O; \theta_0, \eta_0) = \{\mathbb{E}[R^*(O; \theta_0, \eta_0)R^*(O; \theta_0, \eta_0)^T]\}^{-1} R^*(O; \theta_0, \eta_0)$, and the semiparametrically efficient asymptotic variance (i.e., efficiency bound) is equal to $\{\mathbb{E}[R^*(O; \theta_0, \eta_0)R^*(O; \theta_0, \eta_0)^T]\}^{-1}$, we only present the efficient score function R^* instead of the EIF ψ^* . However, note that deriving the EIF from R^* is straightforward in our context.

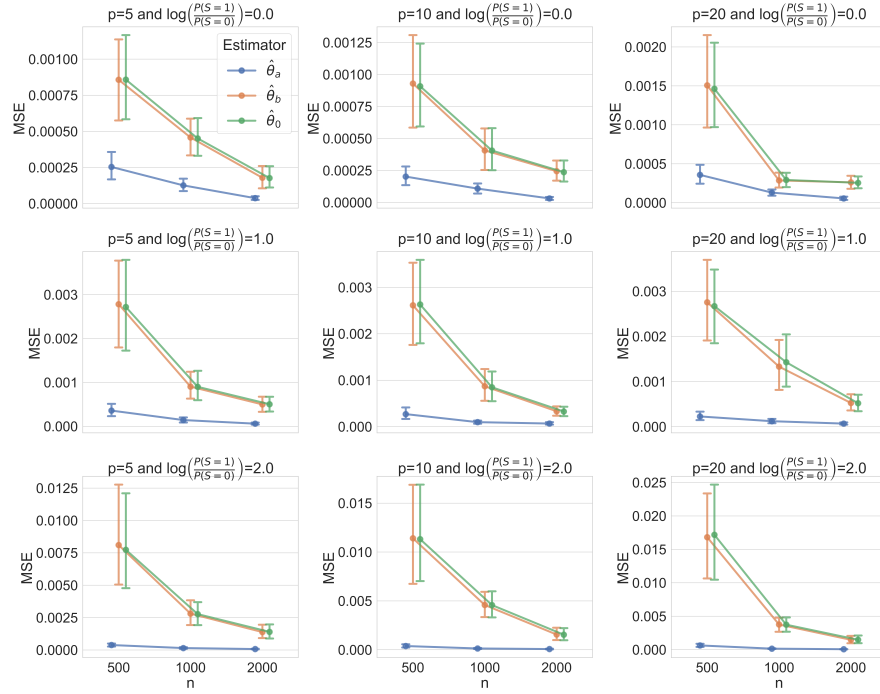


Figure 2: *Simulation Study Results.* Mean squared error rates for three different estimators $\hat{\theta}_0$, $\hat{\theta}_a$ and $\hat{\theta}_b$ based on R_0^* , R_a^* , and R_b^*

655 In our case, $O = (X, S, T, V)$, $P(O = o; \theta, \eta) = P(X = x)P(S = s | X = x)P(T = t | X =$
 656 $x, S = s)P(V = v | T = t, X = x, S = s)$ and $\mathcal{L}(O; \theta, \eta) = \log P(O = o; \theta, \eta)$. Thus,

$$\begin{aligned} \mathcal{L}(O; \theta, \eta) &= \log P(X = x) + \log P(S = s | X = x) + \log P(T = t | S = s, X = x) \\ &\quad + \log P(V = v | T = t, S = s, X = x) \\ &= \log P(X = x) + \log(S\mu_S(x) + (1 - S)(1 - \mu_S(x))) \\ &\quad + \log(T\mu_T(x, s) + (1 - T)(1 - \mu_T(x, s))) \\ &\quad + \log(P(V = v | T = t, S = s, X = x)), \end{aligned}$$

We know that $V = SW + (1 - S)Y$ and $Y = \alpha(X)W + \beta(X) + \varepsilon$.

Thus,

$$P(V = v | T = t, S = s, X = x) = P(S(Y - \beta(X) - \varepsilon) + \alpha(X)(1 - S)Y = \alpha(X)v | T = t, S = s, X = x).$$

. Simplifying it further,

$$P(V = v | T = t, S = s, X = x) = P((S + \alpha(X)(1 - S))Y - S\varepsilon = \alpha(X)v + \beta(X)S | T = t, S = s, X = x).$$

657 Substituting Y with $\theta(X)T + g(X) + \gamma$:

$$\begin{aligned} P(V = v | T = t, S = s, X = x) &= P((S + \alpha(X)(1 - S))(\theta(X)T + g(X) + \gamma) - S\varepsilon = \alpha(X)v + \beta(X)S | T = t, S = s, X = x) \\ &= sP(\gamma - \varepsilon = \alpha(X)(v - \mu_W(X, 1)) - \theta(X)(T - \mu_T(X, 1)) | T = t, S = s, X = x) \\ &\quad + (1 - s)P(\gamma = (v - \mu_Y(X, 0)) - \theta(X)(T - \mu_T(X, 0)) | T = t, S = s, X = x) \\ &= sP(\alpha(X)\delta = \alpha(X)(v - \mu_W(X, 1)) - \theta(X)(T - \mu_T(X, 1)) | T = t, S = s, X = x) \\ &\quad + (1 - s)P(\gamma = (v - \mu_Y(X, 0)) - \theta(X)(T - \mu_T(X, 0)) | T = t, S = s, X = x) \\ &= sP(\delta = (v - \mu_W(X, 1)) - \frac{\theta(X)}{\alpha(X)}(T - \mu_T(X, 1)) | T = t, S = s, X = x) \\ &\quad + (1 - s)P(\gamma = (v - \mu_Y(X, 0)) - \theta(X)(T - \mu_T(X, 0)) | T = t, S = s, X = x). \end{aligned}$$

Assuming γ and δ are normally distributed with mean 0 and homoskedastic variances σ_γ^2 and σ_δ^2 respectively,

$$\begin{aligned} & \log P(V = v \mid T = t, S = s, X = x) \\ &= \log \left(\frac{s \exp \left(-\frac{((v - \mu_W(x, 1)) - \frac{\theta(x)}{\alpha(x)}(t - \mu_T(x, 1)))^2}{2\sigma_\delta^2} \right)}{(1 - s) \exp \left(-\frac{((v - \mu_Y(x, 0)) - \theta(x)(t - \mu_T(X, 0)))^2}{2\sigma_\gamma^2} \right)} \right) \end{aligned}$$

Efficient score function using only primary data (Result of Theorem 1). Now, we first show the efficient score function for the case that only uses primary data. This works as a baseline case for us and the subsequent efficiency bounds are compared with this case. The efficient score function under assumptions A.2. and A.4. is given as:

$$R_0^*(O; \theta_0, \eta_0) = (1 - S) \cdot \left(((V - \mu_Y(X, 0)) - \theta(X)(T - \mu_T(X, 0))) \cdot \frac{(T - \mu_T(X, 0))}{\sigma_\gamma^2} \right).$$

Let $\Delta_0 = \left(((V - \mu_Y(X, 0)) - \theta(X)(T - \mu_T(X, 0))) \cdot \frac{(T - \mu_T(X, 0))}{\sigma_\gamma^2} \right)$ then $\mathbb{E}[(R_0^*(O; \theta_0, \eta_0))(R_0^*(O; \theta_0, \eta_0))^T] = \mathbb{E}[(1 - S)^2 \Delta_0^2]$, and the asymptotic variance

$$\mathbb{V}_0^\theta(X) := (\mathbb{E}[(R_0^*(O; \theta_0, \eta_0))(R_0^*(O; \theta_0, \eta_0))^T])^{-1} = \frac{1}{\mathbb{E}[\Delta_0^2 \mid S = 0, X] p(S = 0 \mid X)}.$$

Efficient score function under assumptions A.1.–A.4. and A.5(a) (Result of Theorem 2). Under this assumption, only θ is unknown and would be estimated using the data while α and β are known a priori. Thus, the efficient score function under A.5(a) is:

$$R_a^*(O; \theta_0, \eta_0) = \left(\begin{aligned} & S \cdot \left((\alpha(X)(V - \mu_W(X, 1)) - \theta(X)(T - \mu_T(X, 1))) \cdot \frac{(T - \mu_T(X, 1))}{\alpha^2(X)\sigma_\delta^2} \right) \\ & + (1 - S) \cdot \left(((V - \mu_Y(X, 0)) - \theta(X)(T - \mu_T(X, 0))) \cdot \frac{(T - \mu_T(X, 0))}{\sigma_\gamma^2} \right) \end{aligned} \right)$$

Let $\Delta_1 = \left((\alpha(X)(V - \mu_W(X, 1)) - \theta(X)(T - \mu_T(X, 1))) \cdot \frac{(T - \mu_T(X, 1))}{\alpha^2(X)\sigma_\delta^2} \right)$. Then, the asymptotic variance

$$\mathbb{V}_a^\theta(X) := (\mathbb{E}[R_a^*(O; \theta_0, \eta_0)(R_a^*(O; \theta_0, \eta_0))^T \mid X])^{-1} = \left(\begin{aligned} & \mathbb{E}[\Delta_0^2 \mid S = 0, X] p(S = 0 \mid X) \\ & + \mathbb{E}[\Delta_1^2 \mid S = 1, X] p(S = 1 \mid X) \end{aligned} \right)^{-1}.$$

$\mathbb{V}_a^\theta(X)$ is always smaller than or equal to $\mathbb{V}_0^\theta(X)$ because $\mathbb{E}[\Delta_1^2 \mid S = 1, X] p(S = 1 \mid X)$ is non-negative.

Efficient score function under assumptions A.1.–A.4. and A.5(b) (Result of Theorem 3). Here, along with θ , α is an unknown parameter. Thus, the efficient score function is given as:

$$R_b^*(O; \{\theta_0, \alpha_0\}, \eta_0) = \left(\begin{aligned} & S \cdot \left((\alpha(X)(V - \mu_W(X, 1)) - \theta(X)(T - \mu_T(X, 1))) \cdot \frac{(T - \mu_T(X, 1))}{\alpha^2(X)\sigma_\delta^2} \right) \\ & + (1 - S) \cdot \left(((V - \mu_Y(X, 0)) - \theta(X)(T - \mu_T(X, 0))) \cdot \frac{(T - \mu_T(X, 0))}{\sigma_\gamma^2} \right) \\ & S \cdot \left(\frac{(\theta(X)(T - \mu_T(X, 1)) - \alpha(X)(V - \mu_W(X, 1)))}{\alpha^2(X)\sigma_\delta^2} \cdot \frac{\theta(X)(t - \mu_T(X, 1))}{\alpha(X)} \right) \end{aligned} \right).$$

$$\begin{aligned} \mathbb{E}[(R_b^*(R_b^*)^T) \mid X] &= \mathbb{E} \left[\left(\begin{aligned} & S\Delta_1 + (1 - S)\Delta_0 \\ & \frac{\theta(X)}{\alpha(X)} S\Delta_1 \end{aligned} \right) \left(\begin{aligned} & S\Delta_1 + (1 - S)\Delta_0 \\ & \frac{\theta(X)}{\alpha(X)} S\Delta_1 \end{aligned} \right) \mid X \right] \\ &= \left(\begin{aligned} & \mathbb{E}[\Delta_1^2 \mid X, S = 1] P(S = 1 \mid X) + \mathbb{E}[\Delta_0^2 \mid X, S = 0] P(S = 0 \mid X) & \frac{\theta(X)}{\alpha(X)} \mathbb{E}[\Delta_1^2 \mid X, S = 1] P(S = 1 \mid X) \\ & \frac{\theta(X)}{\alpha(X)} \mathbb{E}[\Delta_1^2 \mid X, S = 1] P(S = 1 \mid X) & \frac{\theta^2(X)}{\alpha^2(X)} \mathbb{E}[\Delta_1^2 \mid X, S = 1] P(S = 1 \mid X) \end{aligned} \right) \end{aligned}$$

The asymptotic variance-covariance is then

$$\begin{aligned} \Sigma_b(X) &:= \begin{pmatrix} \mathbb{V}_b^\theta(X) & Cov_b^{\theta, \alpha}(X) \\ Cov_b^{\theta, \alpha}(X) & \mathbb{V}_b^\alpha(X) \end{pmatrix} := (\mathbb{E}[(R_b^*(R_b^*)^T) \mid X])^{-1} \\ &= \begin{pmatrix} \frac{1}{\mathbb{E}[\Delta_0^2 \mid X, S = 0] P(S = 0 \mid X)} & -\frac{\alpha_0(X)}{\theta_0(X)} \frac{1}{\mathbb{E}[\Delta_0^2 \mid X, S = 0] P(S = 0 \mid X)} \\ -\frac{\alpha_0(X)}{\theta_0(X)} \frac{1}{\mathbb{E}[\Delta_0^2 \mid X, S = 0] P(S = 0 \mid X)} & \left(\frac{\alpha_0(X)}{\theta_0(X)} \right)^2 \left(\frac{1}{\mathbb{E}[\Delta_0^2 \mid X, S = 0] P(S = 0 \mid X)} + \frac{1}{\mathbb{E}[\Delta_1^2 \mid X, S = 1] P(S = 1 \mid X)} \right) \end{pmatrix} \end{aligned}$$

From this, we see that the asymptotic variance for the efficient estimator of θ is $\mathbb{V}_b^\theta(X) = \frac{1}{\mathbb{E}[\Delta_0^2 | X, S=0] P(S=0 | X)}$. Note, that this asymptotic variance $\mathbb{V}_b^\theta(X) = \mathbb{V}_0^\theta(X)$. This highlights that under assumption [A.5\(b\)](#) there are no efficiency gains from leveraging auxiliary data compared to the baseline which only uses the primary study.

Efficient score function under assumptions [A.1–A.4](#) and [A.5\(c\)](#) (Result of Theorem 4). As the likelihood is agnostic of β , the efficient score function under [A.5\(c\)](#) is identical to that of [A.5\(b\)](#), i.e.,

$$R_c^*(O; \{\theta_0, \alpha_0\}, \eta_0) = R_b^*(O; \{\theta_0, \alpha_0\}, \eta_0).$$

As the score functions are identical under assumptions [A.5\(b\)](#) and [A.5\(c\)](#), the asymptotic variance is also identical. This indicates that there are no efficiency gains from leveraging auxiliary data compared to the baseline that uses only the primary study.

Appendix C Proof of Theorem 5 (Misspecification Bias)

Proof of Theorem 5. We begin by defining $\hat{\theta}_a$ as the estimator solving the empirical moment condition $\mathcal{P}_n R_a^*(O; \hat{\theta}_a, \hat{\eta}) = 0$. In the population, θ_0 solves $\mathbb{E}[R_a^*(O; \theta_0, \eta_0)] = 0$ only under the correct specification of $\alpha = \alpha^*$. We now investigate what happens when the analyst assumes $\alpha = \alpha_{\text{mis}}$, where $\alpha_{\text{mis}} \neq \alpha^*$. Recall, $\hat{\theta}_a$ is

$$\frac{\sum_i \left((1 - S_i) \frac{\hat{r}_Y(X_i, 0) \hat{r}_T(X_i, 0)}{\hat{\sigma}_Y^2} + S_i \frac{\hat{r}_W(X_i, 1) \hat{r}_T(X_i, 1)}{\alpha(X_i) \hat{\sigma}_W^2} \right)}{\sum_i \left((1 - S_i) \frac{\hat{r}_T^2(X_i, 0)}{\hat{\sigma}_Y^2} + S_i \frac{\hat{r}_T^2(X_i, 1)}{\alpha^2(X_i) \hat{\sigma}_W^2} \right)}$$

Thus, $\mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*)] = \mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*) | S = 0] P(S = 0) + \mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*) | S = 1] P(S = 1)$. In the estimator, terms with $(1 - S)$ do not interact with α . Thus, $\mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*) | S = 0] P(S = 0) = 0$. Now, consider $\mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*) | S = 1] P(S = 1)$.

$$\begin{aligned} \mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*) | S = 1] &= \mathbb{E}[\mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*) | X, S = 1] | S = 1] \\ \mathbb{E}[\hat{\theta}_a(\alpha_{\text{mis}}) - \hat{\theta}_a(\alpha^*) | X, S = 1] &= \mathbb{E} \left[\frac{(\alpha_{\text{mis}}(X) - \alpha^*(X)) \mathbb{E}[\hat{r}_W(X, 1) \hat{r}_T(X, 1)]}{(\mathbb{E}[\hat{r}_T^2(X, 1)])} | X, S = 1 \right] \\ &= \mathbb{E} \left[\frac{(\alpha_{\text{mis}}(X) - \alpha^*(X)) \mathbb{E}[\hat{r}_Y(X, 1) \hat{r}_T(X, 1)]}{\alpha^*(X) (\mathbb{E}[\hat{r}_T^2(X, 1)])} | X, S = 1 \right] \\ &= \mathbb{E} \left[\frac{(\alpha_{\text{mis}}(X) - \alpha^*(X))}{\alpha^*(X)} \theta(X) | X, S = 1 \right] \end{aligned}$$

□

Appendix D Proof of Theorem 6 (Error bound for $\hat{\mu}_Y$)

Proof. We analyze the estimation error for both $\hat{\mu}_{Y,0}$ and $\hat{\mu}_{Y,b}$ under the given metric entropy assumptions.

(i) **One-stage estimator $\hat{\mu}_{Y,0}$.** By assumption [A.6.](#), $\mu_Y \in \mathcal{M}$, and the metric entropy of \mathcal{M} satisfies

$$\log N(\varepsilon, \mathcal{M}, \|\cdot\|) \leq C\varepsilon^{-\omega}.$$

From standard results in empirical process theory and nonparametric regression (e.g., [Györfi et al. \(2006\)](#) and [Tsybakov and Tsybakov \(2009\)](#)), it follows that the least-squares estimator $\hat{\mu}_{Y,0}$ satisfies

$$\|\hat{\mu}_{Y,0} - \mu_Y\| = o_p \left(n_0^{-1/(2+\omega)} \right).$$

697 **(ii) Two-stage estimator $\hat{\mu}_{Y,b}$.** By assumption A.5., $\mu_Y(X) = \alpha(X)\mu_W(X) + \beta(X)$. We estimate
698 $\hat{\mu}_W(X)$ from n_1 auxiliary samples. Let $\hat{\mu}_W$ be an estimator satisfying

$$\|\hat{\mu}_W - \mu_W\| = o_p\left(n_1^{-1/(2+\omega)}\right),$$

699 under the assumption that $\mu_W \in \mathcal{M}$ and satisfies the same entropy bound as μ_Y .

700 The two-stage estimator is defined as:

$$\hat{\mu}_{Y,b}(X) = \hat{\alpha}(X) \cdot \hat{\mu}_W(X, 0) + \hat{\beta}(X),$$

701 where $(\hat{\alpha}, \hat{\beta})$ minimize the squared error loss over the primary sample:

$$(\hat{\alpha}, \hat{\beta}) = \arg \min_{\alpha \in \mathcal{A}, \beta \in \mathcal{B}} \frac{1}{n_0} \sum_{i=1}^{n_0} (Y_i - \alpha(X_i)\hat{\mu}_W(X_i, 0) - \beta(X_i))^2.$$

702 We now decompose the error:

$$\|\hat{\mu}_{Y,b} - \mu_Y\| = \|\hat{\alpha}\hat{\mu}_W + \hat{\beta} - \alpha\mu_W - \beta\|.$$

703 Adding and subtracting intermediate terms:

$$= \|(\hat{\alpha} - \alpha)\hat{\mu}_W + \alpha(\hat{\mu}_W - \mu_W) + (\hat{\beta} - \beta)\|.$$

704 Applying triangle inequality:

$$\|\hat{\mu}_{Y,b} - \mu_Y\| \leq \|(\hat{\alpha} - \alpha)\| \|\hat{\mu}_W\|_\infty + \|\alpha\|_\infty \|\hat{\mu}_W - \mu_W\| + \|\hat{\beta} - \beta\|.$$

705 Under the assumption that $\hat{\mu}_W$ is uniformly bounded (which holds if μ_W and $\hat{\mu}_W$ are bounded and
706 consistent), and using the entropy conditions on \mathcal{A} and \mathcal{B} :

$$\|\hat{\alpha} - \alpha\| = o_p(n_0^{-1/(2+\omega_\alpha)}) \quad (\text{given A.6.}), \text{ and, } \|\hat{\beta} - \beta\| = 0 \quad (\text{given A.5(b)}).$$

707 Combining all the pieces, we obtain:

$$\|\hat{\mu}_{Y,b} - \mu_Y\| = o_p\left(n_0^{-1/(2+\omega)} \left(n_0^{\frac{1}{2+\omega} - \frac{1}{2+\omega_\alpha}} + \left(\frac{n_1}{n_0}\right)^{-1/(2+\omega)}\right)\right),$$

708 where the first term reflects the complexity reduction from modeling μ_Y via $\alpha(X)$, and the second
709 term reflects the error propagated from estimating μ_W using auxiliary data. \square

710 Appendix E Implementational Details

711 We implement the methods and case studies in this paper using Python 3.10. The implementa-
712 tion relies on following key libraries/packages: PyTorch for tensor operations and gradient-based
713 optimization, scikit-learn for machine learning models and cross-validation procedures, pandas
714 for data manipulation, and NumPy for numerical computations. Here is the python code of our
715 implementation:

```
716 1 import torch
717 2 import torch.optim as optim
718 3 from sklearn.linear_model import Ridge
719 4 from sklearn.ensemble import AdaBoostRegressor,
720   GradientBoostingRegressor
721 5 from sklearn.model_selection import KFold
722 6 import pandas as pd
723 7 import numpy as np
724 8
725 9 class ThetaAlphaEstimator:
726 10     def __init__(self, data, T_col, S_col, V_col, model_type='ridge',
727   n_splits=5, predefined_alpha=None):
728     # Store column names and data
729     self.T_col = T_col
```

```

73113         self.S_col = S_col
73214         self.V_col = V_col
73315         self.X_cols = [col for col in data.columns if col not in [
734         T_col, S_col, V_col]]
73516
73617         # Extract columns from DataFrame
73718         self.data = data
73819         self.X = data[self.X_cols].values # Covariates (numpy array
739         for compatibility with sklearn)
74020         self.T = data[T_col].values      # Treatment
74121         self.V = data[V_col].values      # Observed outcome
74222         self.S = data[S_col].values      # Study indicator
74323
74424         # Store model type and number of splits for cross-fitting
74525         self.model_type = model_type.lower()
74626         self.n_splits = n_splits
74727
74828         # Define constants
74929         self.sigma_delta2 = data[V_col].loc[data[S_col]==0].var()
75030         self.sigma_epsilon2 = data[V_col].loc[data[S_col]==1].var()
75131
75232         self.theta_0 = torch.tensor(0.1, dtype=torch.float32,
requires_grad=True)
753
75433         if predefined_alpha is None:
75534             self.alpha_0 = torch.tensor(0.1, dtype=torch.float32,
requires_grad=True)
756
75735             self.has_predefined_alpha = False
75836         else:
75937             self.alpha_0 = torch.tensor(predefined_alpha, dtype=torch.
float32, requires_grad=False)
760
76138             self.has_predefined_alpha = True
762

```

Listing 1: Imports and Class Initialization

```

763
7641         def _select_model(self):
7652             # Helper method to select and initialize the model
7663             if self.model_type == 'ridge':
7674                 return Ridge(alpha=1.0)
7685             elif self.model_type == 'adaboost':
7696                 return AdaBoostRegressor(n_estimators=50)
7707             elif self.model_type == 'gradientboosting':
7718                 return GradientBoostingRegressor(n_estimators=100)
7729             else:
77310                 raise ValueError("Unsupported model type. Choose 'ridge',
774                 'adaboost', or 'gradientboosting'.")
7751
77612         def cross_fit_models(self):
77713             # Initialize KFold for cross-fitting
77814             kf = KFold(n_splits=self.n_splits, shuffle=True, random_state
779             =42)
78015
78116             # Initialize empty arrays for predictions
78217             mu_Y_pred = np.zeros(len(self.data))
78318             mu_W_pred = np.zeros(len(self.data))
78419             mu_T_pred = np.zeros(len(self.data))
78520
78621             # Perform K-fold cross-fitting
78722             for train_index, test_index in kf.split(self.X):
78823                 # Split data into training and test sets for this fold
78924                 X_train, X_test = self.X[train_index], self.X[test_index]
79025                 T_train, T_test = self.T[train_index], self.T[test_index]
79126                 V_train, V_test = self.V[train_index], self.V[test_index]
79227                 S_train, S_test = self.S[train_index], self.S[test_index]
79328

```

```

79429         # Fit models for each conditional expectation on the
795         training set
79630         model_mu_Y = self._select_model()
79731         model_mu_W = self._select_model()
79832         model_mu_T_0 = self._select_model()
79933         model_mu_T_1 = self._select_model()
80034
80135         # Fit mu_Y(X,0) on data with S=0
80236         model_mu_Y.fit(X_train[S_train == 0], V_train[S_train ==
803         0])
80437         mu_Y_pred[test_index] = model_mu_Y.predict(X_test)
80538
80639         # Fit mu_W(X,1) on data with S=1
80740         model_mu_W.fit(X_train[S_train == 1], V_train[S_train ==
808         1])
80941         mu_W_pred[test_index] = model_mu_W.predict(X_test)
81042
81143         # Fit mu_T(X,0) and mu_T(X,1) for T given S
81244         model_mu_T_0.fit(X_train[S_train == 0], T_train[S_train ==
813         0])
81445         model_mu_T_1.fit(X_train[S_train == 1], T_train[S_train ==
815         1])
81646         mu_T_pred[test_index] = np.where(S_test == 0, model_mu_T_0
817         .predict(X_test), model_mu_T_1.predict(X_test))
81847
81948         # Convert predictions to PyTorch tensors
82049         self.mu_Y_pred = torch.tensor(mu_Y_pred, dtype=torch.float32)
82150         self.mu_W_pred = torch.tensor(mu_W_pred, dtype=torch.float32)
82251         self.mu_T_pred = torch.tensor(mu_T_pred, dtype=torch.float32)
823

```

Listing 2: Model Selection and Cross-Fitting

```

824
8251     def R_b_star(self):
8262         # Vectorized computation of  $R_b^*$  using cross-fitted
827         predictions
8283         term1 = (torch.tensor(self.T, dtype=torch.float32) - self.
829         mu_T_pred) * \
8304         (self.alpha_0 * (torch.tensor(self.V, dtype=torch.
831         float32) - self.mu_W_pred) - self.theta_0 * (torch.tensor(self.T,
832         dtype=torch.float32) - self.mu_T_pred)) * \
8335         torch.tensor(self.S, dtype=torch.float32)
8346
8357         term2 = (torch.tensor(self.T, dtype=torch.float32) - self.
836         mu_T_pred) * \
8378         ((torch.tensor(self.V, dtype=torch.float32) - self.
838         mu_Y_pred) - self.theta_0 * (torch.tensor(self.T, dtype=torch.
839         float32) - self.mu_T_pred)) * \
8409         (1 - torch.tensor(self.S, dtype=torch.float32))
84110
84211         R_0 = term1 + term2
84312
84413         R_1 = torch.tensor(self.S, dtype=torch.float32) * \
84514         (torch.tensor(self.T, dtype=torch.float32) - self.
846         mu_T_pred) * (self.theta_0 / self.alpha_0) * \
84715         (self.theta_0 * (torch.tensor(self.T, dtype=torch.
848         float32) - self.mu_T_pred) - self.alpha_0 * (torch.tensor(self.V,
849         dtype=torch.float32) - self.mu_W_pred))
85016
85117         # Stack R_0 and R_1 to form the 2D  $R_b^*$  matrix
85218         R = torch.stack((R_0, R_1), dim=1)
85319         return R
85420
85521     def R_a_star(self):
85622         term1 = (torch.tensor(self.T, dtype=torch.float32) - self.
857         mu_T_pred) * \

```

```

85823         (self.alpha_0 * (torch.tensor(self.V, dtype=torch.
859 float32) - self.mu_W_pred) - self.theta_0 * (torch.tensor(self.T,
860 dtype=torch.float32) - self.mu_T_pred)) * \
86124         torch.tensor(self.S, dtype=torch.float32)
86225
86326         term2 = (torch.tensor(self.T, dtype=torch.float32) - self.
864 mu_T_pred) * \
86527         ((torch.tensor(self.V, dtype=torch.float32) - self.
866 mu_Y_pred) - self.theta_0 * (torch.tensor(self.T, dtype=torch.
867 float32) - self.mu_T_pred)) * \
86828         (1 - torch.tensor(self.S, dtype=torch.float32))
86929
87030         return term1 + term2
87131
87232     def R_0_star(self):
87333         term2 = (torch.tensor(self.T, dtype=torch.float32) - self.
874 mu_T_pred) * \
87534         ((torch.tensor(self.V, dtype=torch.float32) - self.
876 mu_Y_pred) - self.theta_0 * (torch.tensor(self.T, dtype=torch.
877 float32) - self.mu_T_pred)) * \
87835         (1 - torch.tensor(self.S, dtype=torch.float32))
87936
88037         return term2
881

```

Listing 3: R-star Functions Implementation

```

882
883 1     def objective(self, R_star_fn):
884 2         # Computes the objective function for any R_star function
885 passed as an argument
886 3         R = R_star_fn()
887 4         expectation = torch.mean(R, dim=0) # Approximate expectation
888 5         return torch.sum(expectation**2) # Minimize this to get as
889 close to zero as possible
890 6
891 7     def optimize(self, R_star_fn, lr=0.01, num_epochs=1000):
892 8         # Set up the optimizer; only optimize for theta_0 if alpha_0
893 is predefined
894 9         params = [self.theta_0] if not hasattr(self, 'alpha_0', '
895 requires_grad') else [self.theta_0, self.alpha_0]
896 10        optimizer = optim.Adam(params, lr=lr)
897 11
898 12        # Training loop to minimize the objective
899 13        for epoch in range(num_epochs):
900 14            optimizer.zero_grad() # Zero out the gradients
901 15            loss = self.objective(R_star_fn) # Calculate the loss
902 16            loss.backward()
903 17            optimizer.step() # Update theta_0 and/or alpha_0
904

```

Listing 4: Objective and Optimization Methods