A kernel balancing approach that scales to big data

Kwangho Kim
Department of Health Care Policy
Harvard Medical School
Boston, MA 02115
kkim@hcp.med.harvard.edu

Bijan A. Niknam *
Department of Health Care Policy
Harvard Medical School
Boston, MA 02115
niknam@g.harvard.edu

Jose R. Zubizarreta †
Department of Health Care Policy
Harvard Medical School
Boston, MA 02115
zubizarreta@hcp.med.harvard.edu

Abstract

In causal inference, weighting is commonly used for covariate adjustment. Procedurally, weighting can be accomplished either through methods that model the propensity score, or methods that use convex optimization to find the weights that balance the covariates directly. However, the computational demand of the balancing approach has to date precluded it from including broad classes of functions of the covariates in large datasets. To address this problem, we outline a scalable approach to balancing that incorporates a kernel representation of a broad class of basis functions. First, we use the Nyström method to rapidly generate a kernel basis in a reproducing kernel Hilbert space containing a broad class of basis functions of the covariates. Then, we integrate these basis functions as constraints in a state-of-the-art implementation of the alternating direction method of multipliers, which rapidly finds the optimal weights that balance the general basis functions in the kernel. Using this kernel balancing approach, we conduct a national observational study of the relationship between hospital profit status and treatment and outcomes of heart attack care in a large dataset containing 1.27 million patients and over 3,500 hospitals. After weighting, we observe that for-profit hospitals perform percutaneous coronary intervention at similar rates as other hospitals; however, their patients have slightly worse mortality and higher readmission rates.

1 Introduction

Weighting is commonly used to remove overt biases in observational studies and has useful properties, such as its avoidance of explicit outcome modeling [26] and hence division between study design and outcome analysis [29], as well as its ability in principle to examine multiple outcomes [22]. Typically, weighting approaches rely on regression modeling of the propensity score for probability of treatment assignment [27]. However, this approach relies on strong modeling assumptions, and can yield unstable estimators [18]. A recent balancing approach addresses both of these problems by forgoing propensity score modeling and instead using convex optimization to directly find the

*CAUSALab, Harvard T.H. Chan School of Public Health

†Department of Statistics, Harvard University; Department of Biostatistics and CAUSALab, Harvard T.H. Chan School of Public Health

NeurIPS 2022Workshop on Causal Machine Learning for Real-World Impact (CML4Impact 2022).

weights of minimum dispersion that approximately balance the covariate distributions [45]. Despite these appealing features, current algorithms for the balancing approach are impractical in the large and ever-growing datasets commonly found in health sciences.

We propose a weighting approach blending an efficient kernel balancing method with a state-of-the-art convex optimization algorithm. We draw on kernel balancing approaches from [39], [43], and [16], which posit that the outcome regressions are located within a kernel representation of a flexible function space. We use the Nyström approximation to efficiently compute the kernel basis in linear time and space. Then, we add the basis functions as linear constraints in a quadratic program (QP), which is efficiently solved via a recent specialized first-order alternating direction method of multipliers [33]. The resulting weights are the stable balancing weights that approximately balance functions of the covariates from the kernel. We show that the proposed method is computationally efficient, has excellent estimator accuracy, and is practical in large-scale observational studies.

2 Estimation framework

2.1 Setup

Our setting is an observational study with n triplets $(X, A, Y) \overset{iid}{\sim} \mathbb{P}$, where $X \in \mathbb{R}^d$ is a set of observed baseline covariates, A is a binary indicator for treatment assignment, and $Y \in \mathbb{R}$ is a real-valued outcome. Applying the potential outcomes framework for causal inference [23, 28], we write $Y_i = A_i Y_i^1 + (1 - A_i)Y_i^0$, where Y_i^a is the potential outcome of unit i under treatment $A = a$; $a = 0, 1$. In our observational study of hospital profit status and heart attack care, $A = 1$ if the patient was admitted to a for-profit hospital, and $A = 0$ if admitted to another type of hospital. Our goal is to estimate the *average treatment effect on the treated* (ATT),

$$\mathbb{E}(Y^1 - Y^0 \mid A = 1).$$

Identification of the ATT relies on some standard assumptions: (i) *consistency*, $Y = Y^a$ if $A = a$, (ii) *exchangeability*, $A \perp\!\!\!\perp Y^0 \mid X$, and (iii) *positivity*, $\mathbb{P}(A = 0 \mid X = x) > 0$ a.s. $[\mathbb{P}]$ (see, e.g., Chapter 12 in Imbens and Rubin 17). Given these assumptions,

$$\begin{aligned}\mathbb{E}(Y^1 - Y^0 \mid A = 1) &= \mathbb{E}(Y \mid A = 1) - \mathbb{E}\{\mathbb{E}(Y \mid X, A = 0) \mid A = 1\} \\ &\equiv \mathbb{E}(Y \mid A = 1) - \psi.\end{aligned}$$

The first term is simply the expectation of the observed outcomes of the treated sample. However, ψ represents the mean conditional potential outcomes of the treated sample under control, which was not observed, and thus its estimation relies on the aforementioned assumptions as well as adjustment for differences in the covariates X. More formally, let $f_{X|A=a}(x)$ be the density of X given $A = a$, and let $\mu_a(X) = \mathbb{E}[Y \mid X, A = a]$ represent the outcome regression function. Then, $\psi = \int \mu_0(x) f_{X|A=1}(x)dx$. Our goal is to estimate the mean of Y in a control population ($A = 0$) that has covariate distributions similar to the treated population ($A = 1$).

In order to estimate the ATT given the propensity score $\pi(X)$, weighting leverages the equality

$$\mathbb{E}(Y^0 \mid A = 1) = \frac{\mathbb{E}\left\{\mathbb{E}(AY^0 \mid X)\right\}}{\mathbb{P}(A = 1)} = \frac{1}{\mathbb{P}(A = 1)} \mathbb{E}\left\{(1 - A)\frac{\pi(X)}{1 - \pi(X)} Y\right\}, \tag{1}$$

which holds under the assumptions of consistency and exchangeability. Mechanically, weighting removes covariate imbalances by giving more or less emphasis to each unit's Y_i via the function $\frac{\pi(X)}{1-\pi(X)}$. A simple yet general weighting estimator for $\mathbb{E}(Y^0 \mid A = 1) = \psi$ is given by

$$\widehat{\psi} = \sum_{\{i|A=0\}} \hat{w}_i(X_i) Y_i \quad \text{for} \quad \hat{w}_i \geq 0. \tag{2}$$

2.2 Two classes of weighting approaches

In the modeling approach to weighting, we explicitly model the unknown propensity score π, typically with logistic regression, and then construct the weights using the estimated $\widehat{\pi}$. We define a normalized

version of the classic Horvitz-Thompson estimator as

$$\widehat{\psi}_{\text{mod}} = \mathbb{P}_n \left\{ \frac{\widehat{\pi}(X)(1-A)}{1-\widehat{\pi}(X)} Y \right\} \Bigg/ \mathbb{P}_n \left\{ \frac{\widehat{\pi}(X)(1-A)}{1-\widehat{\pi}(X)} \right\}. \tag{3}$$

Model-based weighting estimator performance depends on how accurately $\widehat{\pi}$ is estimated. If a parametric model captures the true π, asymptotic normality is attained with fast $\sqrt{n}$ rates. However, this requires strong functional form assumptions. Nonparametric models are a useful alternative requiring weaker assumptions; however, they do not scale well to large datasets [44]. The more recently introduced balancing approach [5, 15, 46] avoids explicit modeling of π and instead directly solves for the weights w_i that meet prespecified balance and dispersion criteria, guaranteeing balance and yielding a more stable estimator. Zubizarreta [46] recently introduced stable balancing weights (SBW), which are the solution to the QP

$$\begin{aligned} \underset{w \in \mathbb{R}^{n_c}}{\text{minimize}} \quad & \sum_{\{i|A=0\}} (w_i - \bar{w})^2 \\ \text{subject to} \quad & \left| \sum_{\{i|A=0\}} w_i B_b(X_i) - \frac{1}{n_t} \sum_{\{i|A=1\}} B_b(X_i) \right| \leq \delta_b, \quad b = 1, 2, ..., M, \\ & \sum_{\{i|A=0\}} w_i = 1, \quad w \geq 0, \end{aligned} \tag{4}$$

where $\bar{w} = \frac{1}{n_c} \sum_{\{i|A=0\}} w_i$, $B_1, ..., B_M$ are M real-valued basis functions of the covariates spanning the model space containing the unknown true outcome model, and δ_b is the tolerance for the maximum imbalance. By balancing approximately rather than exactly, the approach is more robust to data with limited covariate overlap. Importantly, the degree of balance is customizable via modifying δ_b based on subject matter expertise or a tuning algorithm [3, 6, 37].

2.3 Kernel balancing

There are many possible models of the relationship between covariates and outcomes, but the true model is unknown. Thus, it is desirable to adjust for the largest possible set of basis functions $\{B_1, ..., B_M\}$ spanning a general model space for μ_a [3], such as the Reproducing Kernel Hilbert Space (RKHS) [2, 32]. For a given Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, there exists a corresponding feature map $\Phi : \mathcal{X} \to \mathcal{Q}$ given by $\Phi(x) = [\sqrt{\lambda_1}\varphi_1(x), \sqrt{\lambda_2}\varphi_2(x), \cdots]$, where $\{\lambda_j\}$ and $\{\varphi_j\}$ are the eigenvalues and eigenfunctions of the kernel operator, respectively such that $\int K(x, y)\varphi_j(y)dy = \lambda_j \varphi_j(x)$. Here, $\mathcal{Q}$ is an expanded, typically infinite-dimensional feature space. We assume that

$$\mu_a(x) \in \mathcal{H}_K \coloneqq \left\{ g : g(x) = \sum_{j=1}^{\infty} \beta_j \Phi_j(x) \right\},$$

so the true μ_a lies within the expanded feature space $\mathcal{Q}$. By the "kernel trick" $K(x, y) = \Phi(x)^\top \Phi(y)$, there is no need to compute $\Phi(\cdot)$ explicitly, since balancing only requires the dot product of the transformed feature vectors $\Phi_j(\cdot)$s. Further, for any loss function $\mathcal{L} : \mathbb{R}^2 \to \mathbb{R}$, monotone increasing function Ω, and some $\tau > 0$, by the representer theorem [35] the solution of the regularized problem

$$\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} \mathcal{L}(f(X_i), Y_i) + \tau \Omega(\|f\|) \tag{5}$$

has the form

$$\widehat{f}(x) = \sum_{i=1}^{n} \alpha_i K(X_i, x),$$

for some $\alpha_i \in R$. Let $\boldsymbol{K} \in \mathbb{R}^{n \times n}$ be a kernel matrix such that $\boldsymbol{K}_{ij} = K(X_i, X_j)$. Then, since $\mu_a \in \mathcal{H}_K$, the regularized problem (5) (with any nonzero degree of regularization) will produce $\widehat{\mu}_a(X_i) = \boldsymbol{K}_{i\cdot}\alpha$, where $\boldsymbol{K}_{i\cdot}$ is the i-th row of $\boldsymbol{K}$.

Admitting the eigenvalue (spectral) decomposition $\boldsymbol{K} = U\Lambda U^\top$ where Λ is the diagonal matrix with an ordered set of decreasing eigenvalues, [16] showed that the linear estimator (2)'s empirical bias we wish to minimize through kernel balancing is given by

$$\text{Bias}(\widehat{\psi}; \boldsymbol{K}) := \left(\widehat{w}^\top U_c - \frac{1}{n_t}\mathbb{1}_{n_t}^\top U_t\right)\Lambda U^\top \alpha, \tag{6}$$

where $\widehat{w} \in \mathbb{R}^{n_c}$ in (2), U_c (U_t) are rows of U corresponding to the control (treated) units, and $\mathbb{1}_{n_t}$ is a vector of ones of length n_t. As a result, rather than balancing all n columns of $\boldsymbol{K}$, Wong and Chan [39] and [16] proposed balancing the first $r \ll n$ eigenvectors of $\boldsymbol{K}$; i.e., the first r columns of U. While easier than balancing all n columns, this may still be prohibitive for large n due to the cost of computing the eigenvalue decomposition. Thus, we need an effective approximation method for $\boldsymbol{K}$.

3 Scalable and flexible weighting

3.1 Streamlining the kernel basis calculation

While the kernel approach is a flexible method of balancing covariates, the eigenvalue decomposition of a dense $n \times n$ matrix is typically solved with algorithms requiring $O(n^2)$-space and have $O(n^3)$-time complexity, making them impractical for large samples However, this can be surmounted via low-rank matrix approximations like the Nyström method [9, 13, 38].

By using only a subset of columns of a large kernel matrix, the Nyström method can efficiently find its low-rank approximation. First, we sample a set of column indices $\mathcal{I}_m$ with $m \ll n$ for a matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$. While the Nyström method's standard form selects the indices in $\mathcal{I}_m$ are via uniform random sampling, it is possible to use more sophisticated sampling approaches [e.g., 20]. Next, we form a matrix $C \in \mathbb{R}^{n\times c}$ with $C_{ij} = K(X_i, X_j)$, $i \in \{1, ..., n\}$, $j \in \mathcal{I}_m$, and a matrix $W \in \mathbb{R}^{m\times m}$ with $W_{ij} = K(X_i, X_j)$, $i, j \in \mathcal{I}_m$. Then, the standard Nyström approximation for $\boldsymbol{K}$ is

$$\widetilde{\boldsymbol{K}} = CW^+C^\top,$$

where W^+ is the Moore-Penrose (pseudo) inverse of W.

Rank-restricted Nyström approximation. Small singular values in W may lead to unstable W^+, so usually one chooses $l < m$ with the rank-restricted Nyström approximation [e.g., 13]. Assume W admits the singular value decomposition (SVD) as $W = U_W\Lambda_W U_W^\top$. Let $W_l = U_{W,l}\Lambda_{W,l}U_{W,l}^\top$ be the rank-l truncated SVD of W. Then,

$$\widetilde{\boldsymbol{K}}_l = CW_l^{-1}C = CU_{W,l}\Lambda_{W,l}^{-1}U_{W,l}^\top C^\top \tag{7}$$

as the rank-l Nyström approximation for $\boldsymbol{K}$.

Extra dimensionality reduction. Further dimensionality reduction is useful for improving computational speed with large n. Drawing from Wang et al. [36], let $R = CU_{W,l}\Lambda_{W,l}^{-1/2}$ which admits the truncated SVD as $R = \tilde{U}_{R,s}\tilde{\Lambda}_{R,s}\tilde{V}_{R,s}^\top$ for $s < l$. Here, $\tilde{V}_{R,s} \in \mathbb{R}^{l\times s}$ contains the dominant s right singular vectors of R. Then, we get

$$\widetilde{\boldsymbol{K}}_s := DD^\top \quad \text{where} \quad D = R\tilde{V}_{R,s} \in \mathbb{R}^{n\times s} \tag{8}$$

as our rank-s Nyström approximation for $\boldsymbol{K}$.

Given that $s < l < m \ll n$, the time and space complexities of computing $\widetilde{\boldsymbol{K}}_l$ are $O(m^3 + nml)$ and $O(nm + m^2)$, respectively. Crucial for scalability, these complexities are now linear in n. Thus, (7) approximates the eigenvector-based kernel bases much more efficiently.

While optimal settings for s, l, and m are not certain, we draw from Wang et al. [36] by setting $l = \lceil m/2 \rceil$ and $s = l$ and gradually decreasing s until the problem becomes feasible.

3.2 Fast optimization of balancing weights

We now consider solving for the balancing weights in large samples. The *alternating direction method of multipliers* (ADMM) is a popular convex optimization solution method thanks to its

computational efficiency [4]. Despite its strengths, it cannot detect infeasibility, has convergence rates that are sensitive to parameter and data settings, and loses speed gains on high-accuracy settings. Recently, [33] introduced the operator splitting solver for quadratic programs (OSQP), a state-of-the-art ADMM-based general QP solver that overcomes these limitations and is a convenient method for finding the stable balancing weights in (4). In our setting, since the coefficient matrix is symmetric quasi-definite and sparse, the algorithm can more efficiently solve the linear system of equations, and since the linear system's coefficient matrix is always non-singular regardless of step-size parameter values, the algorithm maintains numerical stability without losing speed (see Appendix B for details). Finally, since the relevant linear system's coefficient matrix system is fixed across iterations and not dependent on δ, the latter can be tuned much more efficiently via factorization caching. Moreover, because the weighting solutions do not vary with small changes in δ, warm starting can provide even more efficiency (see Appendix D.5 for experimental results).

3.3 Implementation and bounds on bias

Assume we have completed the rank-l Nyström approximation for $\boldsymbol{K}$ as in (7). For the matrix C let $C_t \in \mathbb{R}^{n_t \times m}$ and $C_c \in \mathbb{R}^{n_c \times m}$ be rows of C for the treated and control units, respectively. Given these, the stable kernel balancing weights can be found by solving the following QP:

$$\begin{aligned} \underset{\substack{w \in \mathbb{R}^{n_c} \\ z \in R^{1+n_c+l}}}{\text{minimize}} \quad & w^\top \mathbb{I}_{n_c} w - \frac{1}{n_c} w^\top \mathbb{1}_{n_c} \\ \text{subject to} \quad & Qw = z, z \in \mathcal{C}(\delta), \end{aligned} \tag{9}$$

for $Q = \begin{bmatrix} \mathbb{1}_{n_c} & \mathbb{I}_{n_c} & C_c U_{W,l} \end{bmatrix}^\top$ and a set $\mathcal{C}(\delta) = \{z \mid l(\delta) \leq z \leq u(\delta)\}$ where

$l(\delta) = \begin{bmatrix} 1 & \mathbb{0}_{n_c} & \overline{C_t U_{W,l}} - \delta \end{bmatrix}^\top$, $u(\delta) = \begin{bmatrix} 1 & \mathbb{1}_{n_c} & \overline{C_t U_{W,l}} + \delta \end{bmatrix}^\top$, $\delta = (\delta_1 \sqrt{\sigma_1}, ..., \delta_l \sqrt{\sigma_l})$.

Here, $\mathbb{I}_r \in \mathbb{R}^{r \times r}$ is the $r \times r$ identity matrix, and $\mathbb{1}_r, \mathbb{0}_r \in \mathbb{R}^{1 \times r}$ are vectors of ones and zeros of length r, respectively; $\delta = (\delta_1, ..., \delta_l)$ is a vector of investigator-specified maximum tolerable imbalances; $\overline{C_t U_{W,l}} \in \mathbb{R}^{1 \times l}$ is a row vector of means of each column of $C_t U_{W,l}$; and $\text{diag}(\sigma_1, ..., \sigma_l) = \Lambda_{W,l} \in \mathbb{R}^{l \times l}$. In (9), $x \in \mathbb{R}^{n_c}$, $z, l, u \in \mathbb{R}^{1+n_c+l}$, and $Q \in \mathbb{R}^{(1+n_c+l) \times n_c}$. In Appendix A, we describe computation of the proposed estimator in detail.

While in ADMM form, (9) corresponds to the balancing weights problem in (4) with the objective function being the approximated kernel basis $(CU_{W,l})_{ij}$ as $B_j(X_i)$ and sample variance of w $\zeta(w) = \frac{1}{n_c} \sum_{\{i|A=0\}} \left(w_i - \frac{1}{n_c} \sum_{\{i|A=0\}} w_i \right)$. we have $s < l$ balancing constraints in the QP.

In the following proposition, we show the worst-case bound of the bias (6) for the weighting estimator resulting from from the optimal solution to (9) (see Appendix C for the proof).

Proposition 3.1. *If $\|\alpha\|_2 < \infty$, then we have the deterministic bound*

$$\textit{Bias}(\widehat{\psi}; \boldsymbol{K}) \lesssim \sum_{b=1}^{l} \delta_b + \|\boldsymbol{K} - \widetilde{\boldsymbol{K}}\|_2 + \|W^+ - W_l\|_F,$$

where $\|\cdot\|_2$ and $\|\cdot\|_F$ are the 2-norm and the Frobenius norm, respectively. Further, if we use the extra dimensionality reduction, then, with high probability, the following bound

$$\textit{Bias}(\widehat{\psi}; \boldsymbol{K}) \lesssim \sum_{b=1}^{s} \delta_b + \|\boldsymbol{K} - \boldsymbol{K}_s\|_*$$

holds, where $\boldsymbol{K}_s$ is a rank-s truncated SVD of $\boldsymbol{K}$ and $\|\cdot\|_*$ is the trace norm.

By this proposition, the upper bound on bias is the sum of error due to residual covariate imbalance after weighting and error due to kernel matrix approximation, with the former controllable via the balancing constraints. Note that W_l and $\boldsymbol{K}_s$ are the best low-rank approximations for W^+ and $\boldsymbol{K}$, respectively, with the corresponding ranks. The kernel matrix approximation error generally declines with larger s, l, and m, and the bound of the residual error $\|\boldsymbol{K} - \widetilde{\boldsymbol{K}}\|_2$ can be found; see, e.g., [13].

In Appendix D, we provide extensive simulation studies showing that the proposed approach has excellent estimation accuracy and also demonstrates superior computational efficiency at sample sizes up to 1 million and under a variety of covariate specifications, even relative to logistic regression.

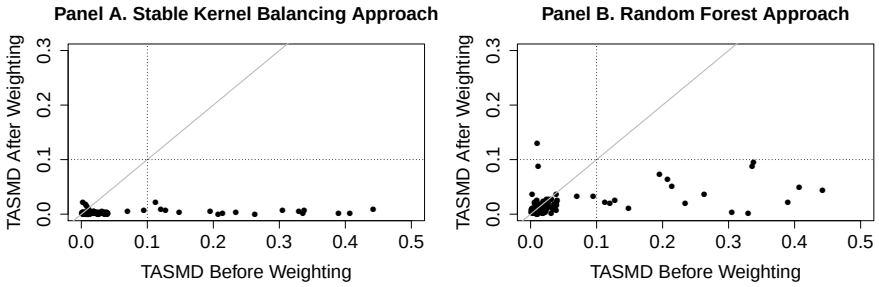

Figure 1: Here, we plot the TASMDs between for-profit and control hospitals. The coordinates of each dot represent the TASMD of a covariate before and after weighting.

4 A national study of heart attack care by hospital profit status

The effect of for-profit orientation on medical care is a central question in health economics. Medicare pays hospitals more if heart attack patients receive percutaneous coronary intervention (PCI), which may incentivize excessive and potentially unneccessary use. However, a reasonable alternative hypothesis is that profit motives may improve clinical efficiency and spur technical and human capital investments that lead to better outcomes and potentially reduced need for PCI. In fact, two studies from the 1990s and early 2000s using small data samples reported opposing results [30, 31]; however, these studies had fewer than 160,000 patients from small numbers of hospitals that were participating in voluntary cardiac reporting systems, which may have altered provider behavior.

In contrast, we studied a larger sample of 1,273,636 Medicare beneficiaries admitted to hospitals for heart attack between October 1, 2010 and November 30, 2019. There were 206,948 patients admitted to 775 for-profit hospitals, while 1,066,780 patients were admitted to 2,763 control hospitals. The kernel balancing incorporated 87 adjustment covariates describing the patients and the hospitals in their county of residence, with the latter partially addressing selection bias due to patients choosing where to live based on their medical needs. After weighting, we considered the samples balanced if all target absolute standardized mean differences (TASMD) [6] were below 0.1.

4.1 Kernel balance results

Panel A of Figure 1 shows that the weighting algorithm successfully balanced all covariates, with all TASMDs after weighting far below our standard of 0.1, and none above 0.03. The kernel basis computation required only 32.3 seconds, while the OSQP algorithm solved for the stable kernel balancing weights in just 132 seconds. Other methods encountered difficulties- the balancing problem was too large for `quadprog`, while the kernel ridge regression implementation of the modeling approach could not obtain a solution. While fast random forests via `ranger` did find a solution in the modeling approach, it required much more time than kernel balancing, and as can be seen in Panel B of Figure 1, the random forest approach was less successful at balancing even just simple covariate means.

Table 1: PCI Use and Outcomes by Hospital Profit Status

Measure	For-Profit	Control	Difference	95% CI
Received PCI	33.2	33.1	0.1	(-0.1, 0.3)
30-day Mortality	14.3	13.9	0.4	(0.2, 0.6)
30-day Readmission/Inpatient Death	19.2	18.2	0.9	(0.7, 1.1)
30-day Readmission Only	11.5	10.6	0.8	(0.6, 0.9)

4.2 Heart attack treatment and outcomes

Table 1 shows treatment and outcomes by hospital profit status after weighting. For-profit hospitals and control hospitals used PCI at almost the same rate, which aligns with [30], but conflicts with Sloan et al. [31]. However, the latter used data from 1994-1995, and guideline-based care has become more prominent since, perhaps leading to converging treatment practices. 30-day mortality was slightly but significantly higher, while 30-day readmission was about 1% higher at for-profit hospitals, a significant difference.

5 Summary and future work

We described a stable kernel balancing approach to weighting that scales to large datasets without sacrificing estimator accuracy. We showed that the Nyström approximation makes it computationally feasible to find a set of general functions of the covariates in RKHS, and described how the kernel basis from the Nyström approximation can be efficiently balanced via a modern ADMM-based convex optimization method. Using the method to study the relationship between hospital profit status and heart attack care, we found for-profit hospitals have similar PCI rates, but worse patient outcomes. Scalable and stable kernel balancing affords new opportunities to leverage the advantages of the balancing approach to weighting for observational studies using big data.

There are several avenues for future work. First, the algorithm requires a choice of kernel and parameters s, l, and m to form the kernel basis. While our simulations used heuristics from [36], data-adaptive selection methods could be explored, possibly using balance quality after weighting to adjust the parameters to improve balance. The impact of kernel approximation could be studied by re-examining the large-scale results in [39] and [37] after adding the approximation step, giving conditions for the validity of the bootstrap to conduct inference. Finally, while we used simple random sampling for the Nyström approximation, future work could study how more complex sampling schemes affect the kernel approximation bias of the estimator.

References

[1] Patrick R Amestoy, Timothy A Davis, and Iain S Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[2] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[3] Eli Ben-Michael, Avi Feller, David A Hirshberg, and José R Zubizarreta. The balancing act in causal inference. *arXiv preprint arXiv:2110.14831*, 2021.

[4] Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers, Inc., 2011.

[5] Kwun Chuen Gary Chan, Sheung Chi Phillip Yam, and Zheng Zhang. Globally efficient nonparametric inference of average treatment effects by empirical balancing calibration weighting. *Journal of the Royal Statistical Society. Series B*, 78(3):673, 2016.

[6] Ambarish Chattopadhyay, Christopher H Hase, and José R Zubizarreta. Balancing vs modeling approaches to weighting in practice. *Statistics in Medicine*, 39(24):3227–3254, 2020.

[7] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 01 2018.

[8] Timothy A Davis. Algorithm 849: A concise sparse cholesky factorization package. *ACM Transactions on Mathematical Software (TOMS)*, 31(4):587–591, 2005.

[9] Petros Drineas, Michael W Mahoney, and Nello Cristianini. On the nyström method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research*, 6(12), 2005.

[10] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6(4):327–363, 2014.

[11] Christopher Fougner and Stephen Boyd. Parameter selection and preconditioning for a graph form solver. In *Emerging Applications of Control and Systems Theory*, pages 41–61. Springer, 2018.

[12] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[13] Alex Gittens and Michael W Mahoney. Revisiting the nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research*, 17(1):3977–4041, 2016.

[14] Donald Goldfarb and Ashok Idnani. A numerically stable dual method for solving strictly convex quadratic programs. *Mathematical Programming*, 27(1):1–33, 1983.

[15] Jens Hainmueller. Entropy balancing for causal effects: A multivariate reweighting method to produce balanced samples in observational studies. *Political Analysis*, pages 25–46, 2012.

[16] Chad Hazlett. Kernel balancing. *Statistica Sinica*, 30(3):1155–1189, 2020.

[17] Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[18] J. D. Y. Kang and J. L. Schafer. Demystifying double robustness: a comparison of alternative strategies for estimating a population mean from incomplete data (with discussion). *Statistical Science*, 22(4):523–539, 2007.

[19] Edward H Kennedy, Sivaraman Balakrishnan, and Max G'Sell. Sharp instruments for classifying compliers and generalizing causal effects. *The Annals of Statistics*, 48(4):2008–2030, 2020.

[20] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 13(1):981–1006, 2012.

[21] Brian K Lee, Justin Lessler, and Elizabeth A Stuart. Improving propensity score weighting using machine learning. *Statistics in Medicine*, 29(3):337–346, 2010.

[22] Roderick JA Little and Donald B Rubin. *Single imputation methods*. Wiley Online Library, 2019.

[23] J. Neyman. On the application of probability theory to agricultural experiments. *Statistical Science*, 5(5):463–480, 1923, 1990.

[24] Brendan O'donoghue, Eric Chu, Neal Parikh, and Stephen Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, 2016.

[25] James M Robins and Andrea Rotnitzky. Semiparametric efficiency in multivariate regression models with missing data. *Journal of the American Statistical Association*, 90(429):122–129, 1995.

[26] Paul R. Rosenbaum. Model-based direct adjustment. *Journal of the American Statistical Association*, 82(398):387–394, 1987.

[27] Paul R. Rosenbaum and Donald B. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

[28] Donald B Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66(5):688, 1974.

[29] Donald B Rubin. The design versus the analysis of observational studies for causal effects: Parallels with the design of randomized trials. *Statistics in Medicine*, 2006.

[30] Bimal R Shah, Seth W Glickman, Li Liang, W Brian Gibler, E Magnus Ohman, Charles V Pollack, Matthew T Roe, and Eric D Peterson. The impact of for-profit hospital status on the care and outcomes of patients with non–st-segment elevation myocardial infarction: results from the crusade initiative. *Journal of the American College of Cardiology*, 50(15):1462–1468, 2007.

[31] Frank A Sloan, Justin G Trogdon, Lesley H Curtis, and Kevin A Schulman. Does the ownership of the admitting hospital make a difference? outcomes and process of care of medicare beneficiaries admitted with acute myocardial infarction. *Medical Care*, pages 1193–1205, 2003.

[32] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, 2004.

[33] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[34] Dustin Tran, Panos Toulis, and Edoardo M Airoldi. Stochastic gradient descent methods for estimation with large data sets. *arXiv preprint arXiv:1509.06459*, 2015.

[35] Grace Wahba. *Spline models for observational data*. SIAM, 1990.

[36] Shusen Wang, Alex Gittens, and Michael W Mahoney. Scalable kernel k-means clustering with nyström approximation: relative-error bounds. *The Journal of Machine Learning Research*, 20 (1):431–479, 2019.

[37] Yixin Wang and Jose R Zubizarreta. Minimal dispersion approximately balancing weights: asymptotic properties and practical considerations. *Biometrika*, 107(1):93–105, 2020.

[38] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in Neural Information Processing Systems*, 13, 2000.

[39] Raymond KW Wong and Kwun Chuen Gary Chan. Kernel-based covariate functional balancing for observational studies. *Biometrika*, 105(1):199–213, 2018.

[40] Simon N Wood, Yannig Goude, and Simon Shaw. Generalized additive models for large data sets. *Journal of the Royal Statistical Society: Series C*, pages 139–155, 2015.

[41] Shifeng Xiong, Bin Dai, Jared Huling, and Peter ZG Qian. Orthogonalizing em: A design-based least squares algorithm. *Technometrics*, 58(3):285–293, 2016.

[42] Yaohui Zeng and Patrick Breheny. The biglasso package: A memory-and computation-efficient solver for lasso model fitting with big data in r. *arXiv preprint arXiv:1701.05936*, 2017.

[43] Qingyuan Zhao. Covariate balancing propensity score by tailored loss functions. *Annals of Statistics*, page in press, 2019.

[44] Lina Zhou, Shimei Pan, Jianwu Wang, and Athanasios V Vasilakos. Machine learning on big data: Opportunities and challenges. *Neurocomputing*, 237:350–361, 2017.

[45] J. R. Zubizarreta, C. E. Reinke, R. R. Kelz, J. H. Silber, and P. R. Rosenbaum. Matching for several sparse nominal variables in a case-control study of readmission following surgery. *The American Statistician*, 65(4):229–238, 2011.

[46] José R Zubizarreta. Stable weights that balance covariates for estimation with incomplete outcome data. *Journal of the American Statistical Association*, 110(511):910–922, 2015.

APPENDIX

A Algorithm

Algorithm 1: Fast stable kernel balancing weights

input: Sample $Z_1, ..., Z_n$, vector δ, integers $s < l < m \ll n$, and kernel $K(\cdot, \cdot)$

***Step 1: Form kernel bases**

1 Sample a set $\mathcal{I}_m$ of $m \ll n$ indices in $\{1, ..., n\}$

2 Form $C \in \mathbb{R}^{n \times m}$ with $C_{ij} = K(X_i, X_j)$, $i \in \{1, ..., n\}$, $j \in \mathcal{I}_m$, and $W \in \mathbb{R}^{m \times m}$ with $W_{ij} = K(X_i, X_j)$, $i, j \in \mathcal{I}_m$

3 Compute the rank-l truncated SVD $W_l = U_{W,l} \Lambda_{W,l} U_{W,l}^\top$

4 **if** *dimensionality reduction* **then**

5 Compute the rank-s truncated SVD of $R = C U_{W,l} \Lambda_{W,l}^{-1/2}$ as $\tilde{U}_{R,s} \tilde{\Lambda}_{R,s} \tilde{V}_{R,s}^\top$

6 Compute $D = R \tilde{V}_{R,s}$

7 Compute $Q, l(\delta), u(\delta)$ using (9)

***Step 2: Optimize weights with OSQP**

8 Choose parameters $\rho > 0$, $\sigma > 0$, and $\alpha \in (0, 2)$; initialize w^0, y^0, z^0, and ν^0; and set $k = 0$

9 **repeat**

10 $(\widetilde{w}^{k+1}, \nu^{k+1}) \leftarrow$ solve $\begin{bmatrix} -(1+\sigma)\mathbb{I}_{n_c} & -Q^\top \\ -Q & \rho^{-1}\mathbb{I} \end{bmatrix} \begin{bmatrix} \widetilde{w}^{k+1} \\ \nu^{k+1} \end{bmatrix} = - \begin{bmatrix} \sigma w^k - 1/n_c \mathbb{1}_{n_c} \\ z^k - \rho^{-1} y^k \end{bmatrix}$;

11 $\widetilde{z}^{k+1} \leftarrow z^k + \rho^{-1}(\nu^{k+1} - y^k)$

12 $w^{k+1} \leftarrow \alpha \widetilde{w}^{k+1} + (1 - \alpha) w^k$

13 $z^{k+1} \leftarrow \Pi_{\mathcal{C}(\delta)} \left(\alpha \widetilde{z}^{k+1} + (1 - \alpha) z^k + \rho^{-1} y^k \right)$

14 $y^{k+1} \leftarrow y^k + \rho \left(\alpha \widetilde{z}^{k+1} + (1 - \alpha) z^k - z^{k+1} \right)$

15 $w \leftarrow w_k$, $k \leftarrow k + 1$

16 **until** *termination criterion satisfied*;

output: $\sum_{i=1}^{n_c} w_i Y_{c,i}$, where Y_c is an outcome vector for the control units

B OSQP for SBW

The above OSQP algorithm for obtaining the SBW is particularly well-suited to large-scale problems. First, the most computationally expensive step of Algorithm 1 is solving the linear system in Step 10; however, since the matrix $\begin{bmatrix} -(1+\sigma)\mathbb{I}_{n_c} & -Q^\top \\ -Q & \rho^{-1} I \end{bmatrix}$ is symmetric quasi-definite and sparse (with sparsity $\frac{2n_c - 1 + (Kd+2)(2n_c+1)}{n_c^2 + (n_c + Kd + 1)(3n_c + Kd + 1)} \approx \frac{1}{n_c}$), a number of efficient algorithms can be applied to this step. One option is the QDLDL factorization procedure [8], which is the default linear system solver in OSQP, while an alternative option is approximate minimum degree ordering [1]. Second, although the investigator should carefully tune δ in the SBW algorithm to obtain the desired degree of balance, for example using the bootstrap [6, 37], there are methods to execute this step with greater efficiency. Although by necessity the bootstrap requires a number of iterations, the fact that the coefficient matrix used in Step 3 of Algorithm 1 remains unchanged across iterations and does not depend on δ permits the use of factorization caching; i.e., the factorization needs to only be performed once and the resulting information can be stored for use in subsequent iterations. Also, small changes in δ tend not to substantially affect SBW solutions, so warm starting can be used for added efficiency. Together, the techniques of factorization caching and warm starting make OSQP a particularly efficient approach to completing the SBW procedure. Finally, since the coefficient matrix in Step 10 of Algorithm 1 is always non-singular regardless of the value of σ, some arbitrarily small σ can be chosen without a detrimental effect on computation time.

We remark that there exist alternative ADMM-based methods that could be applied; however, these approaches require reformulation of our QP, which introduces many extra coefficients into the optimization procedure and therefore may not be computationally efficient [33].

C Proof of Proposition 3.1

Proof. **i)** $\widetilde{\boldsymbol{K}}_l = CU_{W,l}\Lambda_{W,l}^{-1}U_{W,l}^\top C^\top$.

Let $w_t = \frac{1}{n_t}\mathbb{1}_{n_t}$ and $\widetilde{U}\Lambda_{W,l}^{-1}\widetilde{U}^\top \equiv CU_{W,l}\Lambda_{W,l}^{-1}U_{W,l}^\top C^\top = \widetilde{\boldsymbol{K}}_l$. Then essentially by the Cauchy–Schwarz and triangle inequalities, we have

$$
\begin{aligned}
&\left|\left(w_c^\top U_c - w_t^\top U_t\right)\Lambda U^\top \alpha\right| \\
&\leq \left|\left(w_c^\top U_c - w_t^\top U_t\right)\Lambda U^\top \alpha - \left(w_c^\top \widetilde{U}_c - w_t^\top \widetilde{U}_t\right)\Lambda_{W,l}^{-1}\widetilde{U}^\top\alpha + \left(w_c^\top \widetilde{U}_c - w_t^\top \widetilde{U}_t\right)\Lambda_{W,l}^{-1}\widetilde{U}^\top\alpha\right| \\
&\leq \|w_c\|_2 \left\|U_c\Lambda U^\top - \widetilde{U}_c\Lambda_{W,l}^{-1}\widetilde{U}^\top\right\|_2 \|\alpha\|_2 + \left\|U_t\Lambda U^\top - \widetilde{U}_t\Lambda_{W,l}^{-1}\widetilde{U}^\top\right\|_2 \|\alpha\|_2 \\
&\quad + \left|\left(w_c^\top \widetilde{U}_c - w_t^\top \widetilde{U}_t\right)\Lambda_{W,l}^{-1}\widetilde{U}^\top\alpha\right| \\
&\lesssim \left\|\boldsymbol{K} - \widetilde{\boldsymbol{K}}_l\right\|_2 \|\alpha\|_2 + \left|\left(w_c^\top \widetilde{U}_c - w_t^\top \widetilde{U}_t\right)\Lambda_{W,l}^{-1}\widetilde{U}^\top\alpha\right|.
\end{aligned}
$$

For the second term in the last display, it follows that

$$
\begin{aligned}
\sup_\alpha \left|\left(w_c^\top \widetilde{U}_c - w_t^\top \widetilde{U}_t\right)\Lambda_{W,l}^{-1}\widetilde{U}^\top\alpha\right| &\leq \left\|\left(w_c^\top \widetilde{U}_c - w_t^\top \widetilde{U}_t\right)\Lambda_{W,l}^{-1}\right\|_2 \sup_\alpha \left\|\widetilde{U}^\top\alpha\right\|_2 \\
&\lesssim \left\|\left(w_c^\top \widetilde{U}_c - w_t^\top \widetilde{U}_t\right)\Lambda_{W,l}^{-1}\right\|_2 \\
&\leq \sum_{k=1}^c \delta_b,
\end{aligned}
$$

where the first inequality follows by the Cauchy–Schwarz inequality and the second by the balancing conditions and the fact that $\sup_\alpha \|\alpha\|_2 < \infty$. Also for the first term it follows that

$$
\begin{aligned}
\sup_\alpha \left\|\boldsymbol{K} - \widetilde{\boldsymbol{K}}_l\right\|_2 \|\alpha\|_2 &\lesssim \left\|\boldsymbol{K} - \widetilde{\boldsymbol{K}}_l\right\|_2 \\
&\leq \|\boldsymbol{K} - \widetilde{\boldsymbol{K}}\|_2 + \|CW^+C^\top - CW_lC^\top\|_2 \\
&\lesssim \|\boldsymbol{K} - \widetilde{\boldsymbol{K}}\|_2 + \|W^+ - W_l\|_F.
\end{aligned}
$$

Putting the two pieces together, we have the desired bound

$$
\sup_\alpha \left|\left(w_c^\top U_c - w_t^\top U_t\right)\Lambda U^\top \alpha\right| \lesssim \sum_{k=1}^c \delta_b + \|\boldsymbol{K} - \widetilde{\boldsymbol{K}}\|_2 + \|W^+ - W_l\|_F.
$$

ii) $\widetilde{\boldsymbol{K}}_s = DD^\top$.

It can be shown that $DD^\top$ is the rank-s truncated SVD of $\widetilde{\boldsymbol{K}}_l$ [36]. Hence

$$
\begin{aligned}
\left\|\boldsymbol{K} - DD^\top\right\|_2 &\leq \left\|\boldsymbol{K} - DD^\top\right\|_* \\
&\lesssim \|\boldsymbol{K} - \boldsymbol{K}_s\|_*
\end{aligned}
$$

holds with probability at least 0.9 [36, Theorem 1]. Then result follows by the same logic as part i.

□

D Numerical Experiment

We study the computational performance and estimation accuracy of several modeling and balancing weighting methods. We extend the simulation design by Hainmueller [15] to encompass nonlinear treatment assignment processes and sample sizes up to ten million.

D.1 Study design

Consider an observational study with continuous and binary covariates generated as follows

$$\begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 2 & 1 & -1 \\ 1 & 1 & -0.5 \\ -1 & -0.5 & 1 \end{bmatrix} \right), \quad X_4 \sim \text{Unif}[-3, 3], \quad X_5 \sim \chi^2_1, \quad X_6 \sim \text{Bern}[0.5].$$

Using these six covariates, we generate the outcome and treatment variables as

$$Y = (X_1 + X_2 + X_5)^2 + \eta, \quad \eta \sim N(0, 1), \tag{10}$$

$$A = \mathbb{1}\left\{ X_1^2 + 2X_2^2 - 2X_3^2 - (X_4 + 1)^3 - 0.5 \log(X_5 + 10) + X_6 - 1.5 + \varepsilon > 0 \right\}, \tag{11}$$

where $\varepsilon \sim N(0, \sigma^2_\varepsilon)$. By adding more nonlinearity, (11) is extended from the treatment assignment mechanism in Hainmueller [15]. We use $\sigma^2_\varepsilon = 30$ in our setup.

We consider two covariate specifications with $X_{\text{cor}} = (X_1, X_2, X_3, X_4, X_5, X_6)$ corresponding to the setting where all covariates are correctly specified, and $X_{\text{mis}} = (X_1 X_3, X_2^2, X_4, X_5, X_6)$ corresponding to the setting where some covariates are misspecified. The latter case is intended to yield a misspecified propensity score in the modeling approach, and misspecified covariate balance in the balancing approach. For each σ^2_ε, we consider each interaction of treatment assignment mechanism and accuracy of covariate specification. We consider five sample sizes: $n \in \{2 \cdot 10^3, 5 \cdot 10^3, 10^4, 10^5, 10^6\}$. In this design, the ATT is equal to zero by construction. To separate design from analysis and focus on computing scalability and speed, we based the weighting estimator (2) solely on $\widehat{\pi}$; however, it can be augmented by an outcome model to form a doubly-robust estimator as in [25].

D.2 Methods considered

Estimation of the ATT by weighting requires explicit modeling of π for $\widehat{\psi}_{\text{mod}}$ in the modeling approach, or solving the QP (9) to obtain $\widehat{\psi}_{\text{bal}}$ in the balancing approach. We consider various common methods, including some considered useful for large-scale modeling or optimization. For practicality, we focus on methods that can be executed using readily available packages in R. For consistent comparisons of balancing estimators, we fix $\delta = 0.0005$ across all methods, although in Section D.5 we evaluate the computational performance of the methods when selecting δ in a data-driven fashion using the algorithm in [6].

We consider nine methods for $\widehat{\psi}_{\text{mod}}$ and seven methods for $\widehat{\psi}_{\text{bal}}$ totaling 16 methods classified into six groups (see Table 2. For the modeling approach, Group M1 methods use logistic regression (or GLM) to model the propensity score. Group M2 contains lasso algorithms where the tuning parameter is selected via 10-fold cross validation. In Groups M1 and M2, the regression models for π add quadratic terms and all pairwise products of the observed covariates. Group M3 includes three common non-parametric algorithms: random forests, kernel ridge regression, and generalized additive models using splines [21].[3] For the balancing approach, Group B1 includes the widely-used open-source QP solvers in R. Group B2 studies two well-known commercial solvers. Finally, Group B3 includes three state-of-the-art ADMM-based solvers aimed at reducing processing time and used with the kernel balancing approach. The kernel bases are constructed using the Gaussian kernel, which projects into an infinite-dimensional extended feature space $\mathcal{Q}$. For the Nyström approximation, we used $m = 100$ and set other parameters via the heuristic described in Section 3.1. For OSQP, we use the same parameters as Stellato et al. [33].

[3] When non-parametric methods are employed in the modeling approach, we must rely on either empirical process conditions or cross fitting to estimate $\widehat{\pi}$. Here, we use the two-fold cross fitting in the same spirit of [see, e.g., 7, 19].

Table 2: Description of modeling and solution methods used in the simulation study.

	Method	R package	Description
Modeling Approach	M1. Logistic regression	`glm`	Commonly used method to fit logistic regression models in R
		`sgdglm`	Stochastic gradient descent methods for estimation with big data [34]
	M2. Regularized regression	`glmnet`	Lasso via coordinate descent [12]
		`biglasso`	Scalable lasso for big data [42]
		`oem`	Lasso for tall (large n) data [41]
		`penreg`	ADMM-based lasso implementation [4] (`admm.lasso` in results)
	M3. Non-parametric regression	`ranger`	Fast implementation for Random Forests
		`DRR`	Fast implementation for Kernel Ridge Regression (`kernel.ridge` in results)
		`bam`	Generalized additive model using splines for big data [40]
Balancing Approach	B1. Open-source solvers	`quadprog`	Commonly-used QP solver based on the dual method [14]
		`qpoases`	QP solver based on parametric active-set algorithm [10]
	B2. Commercial solvers	`gurobi`	Large scale commercial optimizer by *Gurobi Optimization*
		`Rmosek`	Large scale commercial optimizer by *Mosek ApS*
	B3. ADMM-based solvers	`osqp`	Operator splitting solver for QP (OSQP) [33]
		`pogs`	ADMM-based graph form solver (POGS) [11]
		`scs`	Operator splitting conic solver (SCS) [24]

D.3 Measures and settings

We assess estimator performance using the root-mean-squared error (RMSE) defined by $\left\{\frac{1}{S}\sum_{j=1}^{S}(\widehat{\psi}_j - \psi)^2\right\}^{1/2}$ averaged across $S = 100$ simulations, and quantify the computational cost in average central processing unit (CPU) time across simulations, which we report in seconds unless otherwise noted. Unless otherwise mentioned, each method is implemented in `R` using each package's internally available solvers and default settings for ease of performance comparisons across methods. However, some algorithms may perform better under non-default settings. All simulations were completed using the Harvard Faculty of Arts and Sciences Research Computing (FASRC) clusters,[4] with available memory fixed at 128GB. In the results tables, we write 'F' when the optimization process failed in more than half the simulations, and 'M' when the system ran out of memory in more than half the simulations.

[4] `https://www.rc.fas.harvard.edu/about/cluster-architecture/`

Table 3: RMSE under weak overlap

		X correctly specified (X)					X transformed (X^*)				
	Method \ n	2k	5k	10k	100k	1M	2k	5k	10k	100k	1M
$\widehat{\psi}_{\text{mod}}$	`glm`	1.72	1.29	1.20	1.19	1.18	1.85	1.37	1.30	1.18	1.18
	`sgdglm`	1.72	1.29	1.20	1.19	1.18	1.86	1.37	1.31	1.18	1.19
	`glmnet`	2.20	1.18	0.68	0.23	0.12	4.57	1.79	0.67	0.39	0.35
	`biglasso`	2.57	1.08	0.65	0.22	0.12	5.23	1.68	0.68	0.39	0.35
	`oem`	3.37	1.27	0.79	0.26	0.12	6.47	2.97	0.79	0.40	0.35
	`admm.lasso`	3.83	1.59	0.84	0.29	0.15	7.29	3.04	1.07	0.70	0.51
	`ranger`	0.71	0.33	0.22	0.15	0.09	0.97	0.41	0.36	0.27	0.22
	`kernel.ridge`	1.87	1.19	1.01	0.89	M	6.26	6.07	5.59	4.87	M
	`bam`	2.84	1.31	1.20	1.19	1.18	3.37	1.95	1.25	1.19	1.18
$\widehat{\psi}_{\text{bal}}$	`quadprog`	0.31	0.20	0.15	F	F	0.51	0.34	0.33	F	F
	`qpoases`	0.32	0.22	0.15	F	F	0.65	0.34	0.24	F	F
	`gurobi`	0.19	0.13	0.09	0.04	0.02	0.31	0.19	0.15	0.08	0.05
	`Rmosek`	0.19	0.13	0.10	0.04	0.02	0.31	0.20	0.15	0.08	0.05
	`osqp`	0.19	0.03	0.10	0.04	0.02	0.31	0.19	0.16	0.08	0.05
	`pogs`	0.31	0.20	0.15	M	M	0.59	0.36	0.29	M	M
	`scs`	0.25	0.22	0.19	0.15	0.10	0.55	0.38	0.25	0.18	0.15

D.4 Empirical results

RMSE. Table 3 shows results under weak overlap. The balancing estimators $\widehat{\psi}_{\text{bal}}$ substantially outperform the modeling estimators $\widehat{\psi}_{\text{mod}}$ at all considered sample sizes regardless of covariate specification; however, gaps between $\widehat{\psi}_{\text{bal}}$ and $\widehat{\psi}_{\text{mod}}$ are somewhat smaller when the covariates are misspecified. We also note that the RMSE for kernel balancing is the same as or only slightly different from the commercial solvers.

CPU time. Because the average CPU time did not appreciably vary across simulation settings, we present only their averages in Table 4. The GLM and OSQP methods were fastest, taking only seconds to find the optimal weights even in the largest sample size of one million. OSQP was faster than the commercial solvers MOSEK and Gurobi, by about an order of magnitude compared to the latter. In contrast, the lasso and non-parametric methods required much more time at that sample size, in some cases hundreds of times more than GLM and OSQP. Among the ADMM-based solvers, OSQP was much faster than SCS at the largest sample size. In our approach, forming the kernel basis requires less computation time than solving the QP, demonstrating the efficiency of the Nyström approximation.

Convergence and Memory. Several methods had computational difficulties at larger sample sizes, particularly $\widehat{\psi}_{\text{bal}}$ approaches. At samples of 100,000 or more, the two open-source solvers failed to converge, while the system ran out of memory when attempting to execute POGS, perhaps due to its inability to accommodate sparse matrices. Among $\widehat{\psi}_{\text{mod}}$ approaches, the kernel ridge regression ran out of memory at 1 million observations.

D.5 Time reduction via factorization caching and warm starting

Since the vectors l, u are the only parameters that vary in δ in Algorithm 1, we can speed up repeated OSQP calls via factorization caching and warm starting. To illustrate this computational gain, we solve a sequence of OSQP-based SBW problems using 100 values of δ equally spaced between 0.001 and 0.1 on the same data, and compare its computing time against two popular commercial solvers `gurobi` and `Rmosek` (with the default setting of interior-point methods). We measure the cumulative time for each solver at $m = 1, 20, 40, 60, 80, 100$ problem instances across different values of δ. We repeat the simulation 100 times and present their average values in Web Figure 2.

Table 4: Average CPU time across simulation settings

		Time (in seconds unless otherwise noted)				
	Method \ n	2k	5k	10k	100k	1M
$\widehat{\psi}_{\text{mod}}$	`glm`	<0.1	<0.1	<0.1	0.4	8.6
	`sgdglm`	<0.1	<0.1	<0.1	0.3	5.0
	`glmnet`	1.0	1.8	4.4	18	362
	`biglasso`	9.2	12	20	266	0.8hr
	`oem`	38	81	224	0.6hr	5hr
	`admm.lasso`	1.3	2.7	6.3	15	311
	`ranger`	8.7	30	128	0.4hr	3hr
	`kernel.ridge`	1.1	3.6	29	0.3hr	M
	`bam`	0.7	1.6	3.8	21	210
$\widehat{\psi}_{\text{bal}}$	`quadprog`	0.9	15	119	F	F
	`qpoases`	48	127	392	F	F
	`gurobi`	<0.1	<0.1	0.5	6.3	94
	`Rmosek`	<0.1	<0.1	<0.1	4.2	37
	`osqp`	<0.1	<0.1	<0.1	3.2	9.6
	`pogs`	3.7	46	305	M	M
	`scs`	<0.1	5.3	14.5	27	169

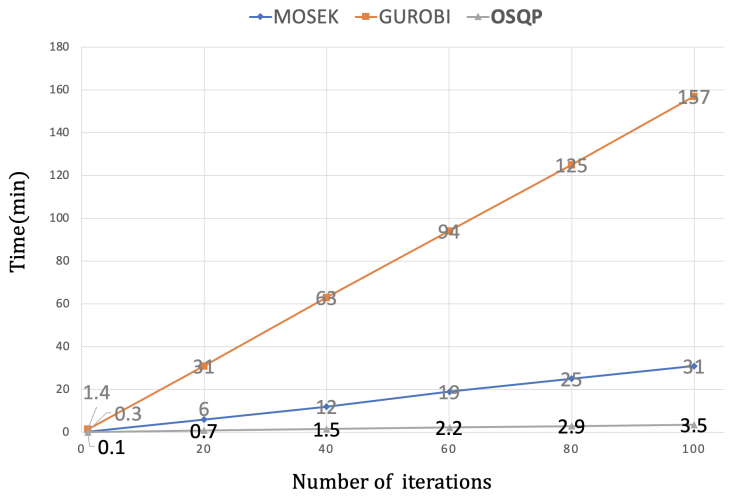

Figure 2: Average cumulative CPU time over multiple iterations of SBW calls

For `gurobi` and `Rmosek`, the cumulative time increases linearly with the number of iterations as would be expected; i.e., m iterations took roughly m times as long as a single execution. However, m iterations of the OSQP algorithm took much less time than $m \times$ (the execution time for a single run); in fact, it is on average 2.4 times faster. This 2.4-fold time improvement demonstrates the benefits of factorization caching and warm starting via OSQP when implementing SBW.