Algorithm of HOIF estimators for ATE

Xingyu Chen[1]

[1]School of Mathematical Sciencei, Shanghai Jiao Tong University, China

January 27, 2026

Contents

1 Input data

Let the observed data be denoted by $(X_i, A_i, Y_i)_{\{i=1\}}^{\{n\}}$, where $A_i \in \{0, 1\}$ is the binary treatment indicator, $Y_i \in \mathbb{R}$ is the observed outcome, and $X_i \in \mathbb{R}^p$ represents the vector of covariates.

We assume the availability of pre-computed nuisance function estimators: the conditional mean outcomes $(\hat{\mu}(1, X_i), \hat{\mu}(0, X_i))$ and the propensity score $\hat{\pi}(X_i)$. All these estimators map to the real line $\mathbb{R}$.

2 The Integrated Algorithm (Main Interface)

The whole function combines all the following steps to get the HOIF estiamtors for ATE. It serves as the primary entry point, orchestrating the data transformation, residual calculation, and the choice of cross-fitting strategy.

Function Inputs:

- Full observed data $(X_i, A_i, Y_i)_{i=1}^n$.
- Nuisance function estimators $(\hat{\mu}(1, X_i), \hat{\mu}(0, X_i), \hat{\pi}(X_i))$.
- Transformation method and its tuning parameters.
- Inverse method of weighted Gram matrix.
- Maximum HOIF order m and ustat backend.
- Switch: A boolean flag for sample splitting and the number of splits K.

Procedure:

(i) Global Pre-processing:

- Transform the covariates X_i on the whole data to obtain basis functions $(Z_i)_{i=1}^n$.
- Compute the global residuals $((R_i^1, r_i^1), (R_i^0, r_i^0))_{i=1}^n$.

(ii) Branching Logic:

- **If not using sample splitting**: Proceed with the entire dataset using the steps described in the following sections. Output $(\text{ATE}_l, \text{HOIF}_l^a, \text{IIFF}_l^a)$ for $l = 2, ..., m$ and $a \in \{0, 1\}$.
- **If using sample splitting (Cross-fitting)**: a. Split the indices $\{1, ..., n\}$ into K disjoint parts $(I_1, I_2, ..., I_K)$. b. For each fold $j = 1, ..., K$:
  - Training: Use data with indices not in I_j $(i \notin I_j)$ to compute the inverse Gram matrices $\left(\Omega_{1,j}, \Omega_{0,j}\right)$.
  - Estimation: Use data with indices in I_j $(i \in I_j)$ and the pre-computed $\left(\Omega_{1,j}, \Omega_{0,j}\right)$ to compute:
    - Local projection matrices $\left(B_{1,j}, B_{0,j}\right)$.
    - Local HOIF estimators $\left(\text{ATE}_{l,j}, \text{HOIF}_{l,j}^a, \text{IIFF}_{l,j}^a\right)$ for $l = 2, ..., m$.

  c. Aggregation: Average the results across all K folds:

$$\text{ATE}_l = \frac{1}{K}\sum_{j=1}^{K} \text{ATE}_{l,j}, \quad \text{HOIF}_l^a = \frac{1}{K}\sum_{j=1}^{K} \text{HOIF}_{l,j}^a, \quad \text{IIFF}_l^a = \frac{1}{K}\sum_{j=1}^{K} \text{IIFF}_{l,j}^a$$

Output: Final averaged estimators $(\text{ATE}_l, \text{HOIF}_l^a, \text{IIFF}_l^a)$ for $l = 2, ..., m$.

3 Step-by-Step Technical Notes

3.1 Transformation of Covariates

First, we transform the covariates X_i into a set of basis functions $Z_i \in \mathbb{R}^k$. Common choices include B-splines or Fourier basis functions.

Function Requirement:

- **Input**: Covariate matrix X, the transformation method (e.g., Fourier or B-splines), and relevant tuning parameters (including the basis dimension k).
- **Output**: The transformed basis matrix $Z \in \mathbb{R}^{n \times k}$, where each row Z_i corresponds to the i-th observation.

3.2 Compute the Residuals

Next, we compute two pairs of residuals, indexed by the treatment assignment $a \in \{0, 1\}$.

For the treatment group $(a = 1)$:

$$R_i^1 = A_i(Y_i - \hat{\mu}(1, X_i))$$

$$r_i^1 = 1 - \frac{A_i}{\hat{\pi}}(X_i)$$

For the control group $(a = 0)$:

$$R_i^0 = (1 - A_i)(Y_i - \hat{\mu}(0, X_i))$$

$$r_i^0 = 1 - \frac{1 - A_i}{1 - \hat{\pi}(X_i)}$$

Function Requirement:

- **Input**: Observed data (A, Y) and estimated nuisance functions $(\hat{\mu}(1, X), \hat{\mu}(0, X), \hat{\pi}(X))$.
- **Output**: Two residual pairs $\left((R_i^1, r_i^1)\right)_{i=1}^n$ and $\left((R_i^0, r_i^0)\right)_{i=1}^n$.

3.3 Compute the Inverse of the Weighted Gram Matrix

Compute the inverse of the weighted Gram matrix G_a for each $a \in \{0, 1\}$:

$$G_a = \frac{1}{n}\sum_{i=1}^{n} s_i^a Z_i Z_i^T,$$

where $s_i^a = A_i^a(1 - A_i)^{1-a}$ serves as the indicator for the a-th group. Let $\Omega_a = G_a^{-1}$ denote the corresponding inverse matrix.

Several estimation methods can be employed for Ω_a, such as direct inversion (e.g., using `chol2inv()` in R for efficiency) or shrinkage estimators (e.g., via the `nlshrink` or `corpcor` R packages) to improve numerical stability in high-dimensional settings.

Function Requirement:

- **Input**: Basis functions Z, treatment indicators A, and the inversion method (`direct`, `nlshrink`, or `corpcor`).
- **Output**: A pair of inverse matrices (Ω_1, Ω_0).

3.4 Compute the Projection Matrix

Compute the projection-like basis matrix B^a for each $a \in \{0, 1\}$:

$$B^a = Z\Omega_a Z^T,$$

where $Z \in \mathbb{R}^{n \times k}$ is the matrix of basis functions. Note that B^a is an $n \times n$ matrix.

Function Requirement:

- **Input**: Basis matrix Z and the inverse matrices (Ω_1, Ω_0).
- **Output**: Basis matrices (B^1, B^0).

3.5 Compute the HOIF Estimators

Finally, we compute the HOIF estimators for the ATE.

Function Requirement:

- **Input**:
  - Maximum order m.
  - Residual pairs $\left((R_i^a, r_i^a)\right)_{a\in\{0,1\}}$.
  - Projection matrices (B^1, B^0).
  - Backend for ustat (`numpy` or `torch`).
- **Procedure**: For each order j from 2 to m, and for each treatment assignment $a \in \{0, 1\}$, calculate the U-statistics:

  $$\mathrm{U}_j^a = (-1)^j \text{ ustat } \left(\text{tensors} = T_j^a, \text{expression} = E_j^a, \text{backend} = \text{backend}, \text{average} = 1\right)$$

  The tensors T_j^a and index expressions E_j^a are defined as:

  $$T_j^a = \left(R^a, \underbrace{B^a}_{j-1 \text{ times}}, r^a\right)$$

  $$E_j^a = (1, (1, 2), ..., (j-1, j), j)$$

  Then, for each order $l \in \{2, ..., m\}$, compute:

  $$\mathrm{IIFF}_l^a = \sum_{j=2}^{l} \binom{l-2}{l-j} \mathrm{U}_j^a$$

  $$\mathrm{HOIF}_l^a = \sum_{j=2}^{l} \mathrm{IIFF}_j^a$$

  $$\mathrm{ATE}_l = \mathrm{HOIF}_l^1 - \mathrm{HOIF}_l^0$$

- **Output**: $(\mathrm{ATE}_l, \mathrm{HOIF}_l^a, \mathrm{IIFF}_l^a)$ for $l = 2, ..., m$.

4 Implementation Notes for Developer (LLM Guidance)

Developer Instruction: When implementing this algorithm in R (or Python), please pay strict attention to the following technical details to ensure the statistical validity of the HOIF estimators.

4.1 Basis Transformation Strategy

- **Input Scaling**: Always scale covariates X to $[0, 1]$ before applying B-splines or Fourier transformations.
- **Additive Construction**: For $X \in \mathbb{R}^p$, generate bases for each dimension separately and concatenate them: $\boldsymbol{Z} = \big[\mathbf{1}, \varphi(X_1), ..., \varphi\big(X_p\big)\big]$. Do not include interaction terms unless specified.
- **Intercept**: Ensure a constant term (column of 1s) is included in $\boldsymbol{Z}$ to maintain the centering property of the Gram matrix.

4.2 Numerical Stability in Matrix Inversion

- **High-Dimensional Case**: If $k \approx n$, the Gram matrix $\boldsymbol{G}_a$ will be ill-conditioned.
- **Efficiency**: In R, prefer `chol2inv(chol(Ga))` for speed, but wrap it in a `tryCatch` to handle non-positive-definite cases.

4.3 Sample Splitting (Cross-Fitting) Logic

This is the most critical part of the implementation. For each fold $k \in \{1, ..., K\}$:

- **Training Set ($\boldsymbol{I_{-k}}$)**: Used **only** to compute $\boldsymbol{\Omega}_{\{a,(-k)\}}$.
- **Estimation Set ($\boldsymbol{I_k}$)**: Used to compute the local projection matrix $\boldsymbol{B}_{\{a,k\}}$ and the U -statistics.
- **Index Alignment**: Ensure that the residuals R_i^a and r_i^a are indexed by I_k and their order matches the rows/columns of $\boldsymbol{B}_{\{a,k\}}$.
  - $\boldsymbol{B}_{\{a,k\}}$ should be of size $|I_k| \times |I_k|$.
  - R_{loc} and r_{loc} should be vectors of length $|I_k|$.

4.4 Data Structure for Output

Return a nested list or a named list containing:

- `ATE`: A vector of length $m - 1$ containing estimators from order 2 to m.
- `IIFF_components`: The raw increment at each order to monitor convergence.
- `Convergence_Plot`: (Optional) A plot of ATE_l vs. l to verify if the estimator stabilizes as order increases.