Comparative Analysis of Propensity Score Methods and Double Machine Learning in Estimating Treatment Effects

Jiazheng Li

May 27, 2024

1 Introduction

Accurately estimating treatment effects is vital for decision-making and policy formulation across healthcare, education, and social sciences. While randomized controlled trials (RCTs) are the gold standard for mitigating confounding biases, they are often impractical or unethical in real-world scenarios. Consequently, researchers rely on observational data, where treated and untreated subjects often differ systematically, making unbiased estimation of average treatment effects challenging and prone to biases.

To address these challenges, advanced causal inference methods like Propensity Score and Double Machine Learning have been developed. These methods enhance the accuracy of treatment effect estimations from observational data. Comparing these approaches is crucial for the robustness of policy recommendations and intervention efficacy across various fields. Through a simplified simulation, this paper reviews their theoretical foundations, strengths and weaknesses, providing insights into their applicability in empirical research.

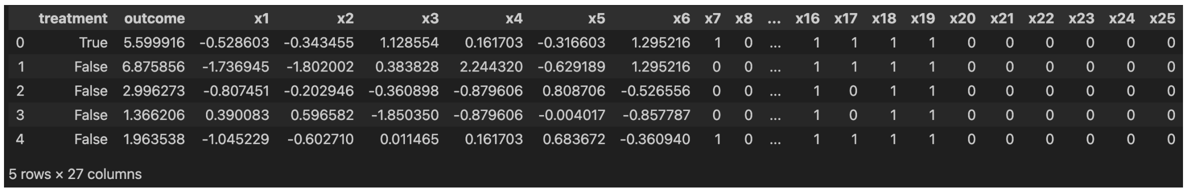
2 Simulation

In order to compare the effectiveness of Propensity Score Methods(PSM) and Double Machine Learning (DML) in estimating treatment effects, I will conduct a simplified simulation study using the Infant Health and Development Program (IHDP) dataset. This dataset is well-suited for such a comparison as it mimics observational study conditions through induced selection bias, providing a robust platform for testing causal inference methodologies. It initially designed as a randomized controlled trial (RCT) to assess the impact of specialist home visits on the cognitive test scores of premature infants. The intervention aimed to improve developmental outcomes by providing timely medical and educational support. The IHDP dataset includes 747 subjects and 25 variables, covering treatment assignment, relevant covariates, and the cognitive test score outcome. This setup allows for comprehensive analysis of the intervention's causal impact.
Firstly, we need to load libraries for calculating propensity socre and the dataset

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import pairwise_distances_argmin_min
from sklearn.neighbors import NearestNeighbors

df = pd.read_excel("..Your path to/ihdp_data.xlsx")
df = df.drop(['y_cfactual','mu0','mu1'], axis = 1)
df.rename(columns={'y_factual': 'outcome'}, inplace=True)
```

And here is how the head five rows of dataset looks like:



	treatment	outcome	x1	x2	x3	x4	x5	x6	x7	x8	...	x16	x17	x18	x19	x20	x21	x22	x23	x24	x25
0	True	5.599916	-0.528603	-0.343455	1.128554	0.161703	-0.316603	1.295216	1	0	...	1	1	1	1	0	0	0	0	0	0
1	False	6.875856	-1.736945	-1.802002	0.383828	2.244320	-0.629189	1.295216	0	0	...	1	1	1	1	0	0	0	0	0	0
2	False	2.996273	-0.807451	-0.202946	-0.360898	-0.879606	0.808706	-0.526556	0	0	...	1	0	1	1	0	0	0	0	0	0
3	False	1.366206	0.390083	0.596582	-1.850350	-0.879606	-0.004017	-0.857787	0	0	...	1	0	1	1	0	0	0	0	0	0
4	False	1.963538	-1.045229	-0.602710	0.011465	0.161703	0.683672	-0.360940	1	0	...	1	1	1	1	0	0	0	0	0	0

5 rows × 27 columns

2.1 Propensity Score Methods

Propensity score methods are used to reduce bias in the estimation of treatment effects from observational data by balancing the distribution of observed covariates between treated and untreated groups. The propensity score, as defined by Rosenbaum and Rubin, is the conditional probability of receiving a treatment given a vector of observed covariates [RR84]. This score helps design and analyze observational studies to mimic some characteristics of randomized controlled trials (RCTs).

Several propensity score methods will be reviewed: matching on propensity score, stratification on propensity score and inverse probability of treatment weighting using propensity score.

2.1.1 Propensity Score Matching

Propensity Score Matching involves pairing treated subjects with untreated subjects who have similar propensity scores. The goal is to create matched sets where the distribution of covariates is similar across treated and untreated subjects, thus reducing bias.

The propensity score $e(X)$ is the probability of receiving treatment given observed covariates X, and matching on it helps estimate the average treatment effect on the treated (ATT) [Imb04]:

$$e(X) = \Pr(T = 1 \mid X)$$

The propensity score is estimated using logistic regression on the dataset:

```
# Estimate propensity scores
model = LogisticRegression()
model.fit(df.drop(['treatment', 'outcome'], axis=1), df['treatment'])
# Calculate the propensity scores for the training data
df['propensity_score'] = \
model.predict_proba(df.drop(['treatment', 'outcome'], axis=1))[:,1]
```

A widely used method is one-to-one matching, the goal is to create pairs of treated and untreated units (individuals, subjects, etc.) that are similar in terms of their propensity scores, which are the probabilities of receiving the treatment given their observed covariates:

```
# Separate the treated and untreated subjects in the training data
treated = df[df['treatment'] == 1]
untreated = df[df['treatment'] == 0]
# Find the closest untreated subjects to the treated subjects based on propensity scores
indices, _ = pairwise_distances_argmin_min(treated[['propensity_score']], \
untreated[['propensity_score']])
# Create a matched dataset where each treated subject is paired with
# the closest untreated subject
matched = treated.copy()
matched['matched_outcome'] = untreated.iloc[indices]['outcome'].values
# Calculate the Average Treatment Effect on the Treated (ATT) using the matched pairs
att_psm = (matched['outcome'] - matched['matched_outcome']).mean()
```

Alternative matching methods include:

Many-to-One Matching: Involves matching multiple untreated subjects to each treated subject, which can increase the efficiency and robustness by utilizing more of the available control data [MR00].The treated subject i is matched with M untreated subjects:

$$\{j_1, j_2, \ldots, j_M\} = \arg \min_{j \in \text{Untreated}} |e(X_i) - e(X_j)|$$

To implement this in python, we need to create a function that pairs each treated subject with multiple nearest untreated subjects based on propensity scores, a good way is using the NearestNeighbors model to find the specified number of closest untreated subjects for each treated subject:

```python
def many_to_one_matching(treated, untreated, n_neighbors=5):
    neighbors = NearestNeighbors(n_neighbors=n_neighbors)
    neighbors.fit(untreated[['propensity_score']])
    distances, indices = neighbors.kneighbors(treated[['propensity_score']])
    return [(treated.index[i], untreated.index[j]) for i in range(len(treated)) \
    for j in indices[i]]

matched_indices = many_to_one_matching(treated, untreated)
matched = pd.DataFrame([(i, j, treated.at[i, 'outcome'], untreated.at[j, 'outcome']) \
for i, j in matched_indices],
    columns=['treated_index', 'untreated_index', 'outcome_treated', 'outcome_untreated'])

att_many_to_one = (matched['outcome_treated'] - matched['outcome_untreated']).mean()
```

Matching with Replacement: matching with replacement allows an untreated subject to be matched to multiple treated subjects, which requires special variance estimation [HR06]

- **Greedy Matching:** Selects the nearest untreated subject iteratively [GR93]). For a treated subject i, it finds the untreated subject j such that:

$$j = \arg\min_{k \in \text{Untreated}} |e(X_i) - e(X_k)|$$

- **Optimal Matching:** Minimizes the total within-pair difference in propensity scores [Aus11]. The objective is to minimize:

$$\sum_{(i,j) \in \text{Matched Pairs}} |e(X_i) - e(X_j)|$$

Full Matching(typically done with specialized packages in R): Forms matched sets with either one treated subject and at least one untreated subject or one untreated subject and at least one treated subject, offering a flexible approach to matching [Aus11]. This method aims to use all available data by creating the most balanced groups possible.

In summary, propensity score matching is a versatile method that can be adapted using various approaches to improve balance and reduce bias in observational studies. By carefully selecting the matching method and adjusting for remaining imbalances, researchers can effectively estimate causal treatment effects from non-randomized data.

2.1.2 Stratification on the Propensity Score

Stratification on the propensity score is a method used in causal inference to control for confounding variables in observational studies. This technique involves dividing subjects into distinct, non-overlapping groups based on their estimated propensity scores, which represent the probability of receiving the treatment given a set of observed covariates. A common approach is to divide subjects into five equal-sized groups using the quintiles of the estimated propensity score [Aus11].

Mathematically, let $e(X)$ be the estimated propensity score for a subject with covariates X. Subjects are stratified into K strata based on the quantiles of $e(X)$. For example, for quintiles, $K = 5$, and the strata are defined by the quintile thresholds $Q_1, Q_2, \ldots, Q_{K-1}$.

$$S_i = \begin{cases} 1 & \text{if } e(X_i) \leq Q_1 \\ 2 & \text{if } Q_1 < e(X_i) \leq Q_2 \\ \vdots & \\ K & \text{if } Q_{K-1} < e(X_i) \end{cases}$$

Within each stratum S_k, the treatment effect on outcomes Y can be estimated by directly comparing treated ($T = 1$) and untreated ($T = 0$) subjects:

$$\hat{\Delta}_k = \bar{Y}_{T=1,S_k} - \bar{Y}_{T=0,S_k}$$

where $\bar{Y}_{T=1}$ and $\bar{Y}_{T=0}$ are the mean outcomes for treated and untreated subjects in the matched sample, respectively.

```
df['stratum'] = pd.qcut(df['propensity_score'], q=5, labels=False)
weighted_mean_treated_list = []
weighted_mean_untreated_list = []

for stratum in df['stratum'].unique():
    stratum_data = df[df['stratum'] == stratum]
    treated = stratum_data[stratum_data['treatment'] == 1]
    untreated = stratum_data[stratum_data['treatment'] == 0]
    treated_mean = treated['outcome'].mean()
    untreated_mean = untreated['outcome'].mean()
    weighted_mean_treated_list.append(treated_mean)
    weighted_mean_untreated_list.append(untreated_mean)
```

Then we may calculate the overall treatment effect by averaging the treatment effects across strata:

```
overall_treated_mean = np.mean(weighted_mean_treated_list)
overall_untreated_mean = np.mean(weighted_mean_untreated_list)
overall_att = overall_treated_mean - overall_untreated_mean
```

2.1.3 Inverse Probability of Treatment Weighting Using the Propensity Score

In our previous propensity score matching code, we were estimating the ATT by matching treated individuals with similar untreated individuals. In contrast, inverse probability of treatment weighting (IPTW) uses the propensity score to create a synthetic sample in which the distribution of measured baseline covariates is independent of treatment [MT08]. It weights individuals to create a pseudo-population in which treatment assignment is independent of observed covariates, allowing for the estimation of the ATE. The weight for each subject is defined as follows [Aus11]:

$$w_i = \frac{Z_i}{e_i} + \frac{(1 - Z_i)}{1 - e_i}$$

where Z_i is an indicator variable denoting whether the ith subject was treated, and e_i is the propensity score for the ith subject.
To implement this method in python, we firstly need to split the original dataset into training and testing set:

```
train_data, test_data = train_test_split(df, test_size=0.3, random_state=1)
```

Then we define the features (X), treatment (T), and outcome (Y) for training and testing sets:

```
X_train = train_data.drop(['treatment', 'outcome'], axis=1)
T_train = train_data['treatment']
Y_train = train_data['outcome']

X_test = test_data.drop(['treatment', 'outcome'], axis=1)
T_test = test_data['treatment']
Y_test = test_data['outcome']
```

Then we estimated the propensity scores using logistic regression, the same way we did for propensity score matching:

```
ps_model = LogisticRegression()
ps_model.fit(X_train, T_train)
propensity_scores_train = ps_model.predict_proba(X_train)[:, 1]
```

Then we may calculate IPTW weights:

```
weights_train = T_train / propensity_scores_train + (1 - T_train) / (1 - propensity_scores_train)
```

And calculate the weighted mean outcome for treated and untreated subjects in the training set:

```
treated_weights_train = weights_train[T_train == 1]
untreated_weights_train = weights_train[T_train == 0]

treated_outcomes_train = Y_train[T_train == 1]
untreated_outcomes_train = Y_train[T_train == 0]

weighted_mean_treated = np.sum(treated_weights_train * treated_outcomes_train) /
np.sum(treated_weights_train)
weighted_mean_untreated = np.sum(untreated_weights_train * untreated_outcomes_train) /
np.sum(untreated_weights_train)
```

Finally, we are able to estimate the Average Treatment Effect (ATE):

```
ate_iptw = weighted_mean_treated - weighted_mean_untreated
```

We may following the exact same steps to validate the estimation using the test set.

2.2 Double Machine Learning

While effective, PSM relies on correctly specifying the propensity score model and can struggle with high-dimensional data where the number of covariates is large relative to the sample size. Double or Debiased Machine Learning (DML) represents a significant advancement over traditional methods like Propensity Score Methods(PSM) by relaxing stringent assumptions about model specification and the functional forms of covariates [FBP24]. Machine learning algorithms can control for confounding variables and provide robust estimates of treatment effects. DML allows kernel regressions to be replaced with modern machine learning (ML) methods and uses a combination of sample splitting and orthogonalization to account for and mitigate the effects of confounding variables, leading to more accurate and unbiased estimates of treatment effects even in complex scenarios [CCD+17]. This adaptation enables the application of DML in high-dimensional contexts, where traditional methods struggle.

The DML framework typically involves the following steps [CCD+17]:

1. **Sample Splitting**: Divide the sample into K folds to mitigate overfitting.
2. **Nuisance Parameter Estimation**: Use ML methods to estimate the nuisance parameters $\hat{m}(X_i)$ and $\hat{g}(X_i)$ in each fold.
3. **Orthogonalization**: Construct orthogonalized scores to remove the bias due to nuisance parameters. For instance, the orthogonalized score for the treatment effect can be written as:

$$\psi(W_i, \hat{\eta}) = \left(\hat{g}(X_i) - \hat{g}(X_i^{(-k)})\right)\left(Y_i - \hat{m}(X_i^{(-k)})\right)$$

   where $\hat{g}(X_i^{(-k)})$ and $\hat{m}(X_i^{(-k)})$ are estimates of $g(X_i)$ and $m(X_i)$ obtained without using the k-th fold.
4. **Final Estimation**: Combine the orthogonalized scores across all folds to obtain the final estimate of the treatment effect $\hat{\tau}$.

In summary, DML represents a robust and flexible framework for causal inference, particularly in high-dimensional settings. By leveraging modern ML methods and incorporating sample splitting, DML addresses many limitations of traditional parametric and semi-parametric methods, providing more accurate and unbiased estimates of treatment effects.

For the implementation, I employ the DoubleML library, which offers a sophisticated approach that accommodates high-dimensional settings and leverages modern machine learning techniques. This library allows for efficient implementation of DML methods by integrating machine learning models to estimate nuisance parameters and then using orthogonalization techniques to debias the treatment effect estimates.

```python
from doubleml import DoubleMLData, DoubleMLPLR
```

Then, we need to import the following machine learning libraries for estimation:

```python
import numpy as np
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier
from xgboost import XGBRegressor, XGBClassifier
from sklearn.neural_network import MLPRegressor, MLPClassifier
```

The DML process involves several steps. First, I split the dataset into five folds to prevent overfitting and facilitate cross-validation. This step ensures that the model's performance is evaluated on unseen data, enhancing its generalizability. In each fold, I estimate the nuisance parameters, including the expected outcome given covariates and the propensity score model, using machine learning methods such as Random Forest, Gradient Boosting and Neural Network. Consider this is a relatively small dataset, for Random Forest, I set the number of estimators to 100, the maximum depth of each tree to 10 and a fixed random state for reproducibility; for Gradient Boosting (XGBoost), the number of estimators is also set to 100, with a maximum depth of 10 for each tree; for Neural Network, I use a multi-layer perceptron with a hidden layer size of 100 neurons and a maximum of 500 iterations. These estimates are crucial for constructing orthogonalized scores. I then proceed with orthogonalization, which involves calculating residuals from the nuisance parameter models and using these residuals to adjust the treatment effect estimates, thereby removing bias due to nuisance parameters. The final estimate of the treatment effect is obtained by combining the orthogonalized scores from all folds, leveraging the entire dataset while mitigating overfitting. Last but not least, I used bootstrapping methods to calcualte the mean, variance, and confidence interval (CI), which are statistical measures used to assess the performance and reliability of treatment effect estimates in PSM and DML methods, respectively.

```python
dml_data = DoubleMLData.from_arrays(X.values, Y.values, T.values)

# Define the DML model using different ML algorithms
ml_g_rf = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=1)
ml_m_rf = RandomForestClassifier(n_estimators=100, max_depth=10, random_state=1)
ml_g_xgb = XGBRegressor(n_estimators=100, max_depth=10, random_state=1)
ml_m_xgb = XGBClassifier(n_estimators=100, max_depth=10, random_state=1)
ml_g_nn = MLPRegressor(hidden_layer_sizes=(100,), max_iter=500, random_state=1)
ml_m_nn = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500, random_state=1)

# Initialize DML models
dml_plr_rf = DoubleMLPLR(dml_data, ml_g_rf, ml_m_rf, n_folds=5)
dml_plr_xgb = DoubleMLPLR(dml_data, ml_g_xgb, ml_m_xgb, n_folds=5)
dml_plr_nn = DoubleMLPLR(dml_data, ml_g_nn, ml_m_nn, n_folds=5)

# Fit the models and estimate treatment effects
dml_plr_rf.fit()
treatment_effect_rf = dml_plr_rf.coef
dml_plr_xgb.fit()
treatment_effect_xgb = dml_plr_xgb.coef
dml_plr_nn.fit()
treatment_effect_nn = dml_plr_nn.coef
```

3 Results and Discussions

The table 1 presents the results of the simulation comparing Propensity Score(IPTW) and Double Machine Learning (DML) using different machine learning models (Random Forest, XGBoost, and Neural Network). The metrics reported include the mean estimate of the treatment effect, the average treatment effect(ATE), the variance estimate, and the 95% confidence interval for each method.

Method	**Mean Estimate**	**ATE**	**Variance Estimate**	**95% CI**
Propensity Score(IPTW)	4.070	4.122	0.053	[3.463, 4.088]
DML Random Forest	3.848	4.012	0.021	[3.597, 4.113]
DML XGBoost	3.827	3.325	0.055	[2.828, 3.650]
DML Neural Network	3.850	3.626	0.045	[3.495, 4.324]

Table 1: Comparison of Methods

The simulation results indicate that Double Machine Learning (DML) methods, particularly Random Forest, offer more precise and stable estimates of treatment effects compared to Propensity Score Methods (IPTW). This conclusion is supported by the lower variance and narrower confidence intervals observed in the DML estimates. Propensity Score methods, on the other hand, takes much less time to run. This advantage might be amplified when dealing with large scale of data.

The theoretical foundation underlying this conclusion stems from the strengths of DML in addressing high-dimensional settings and model misspecification. DML leverages modern machine learning techniques to flexibly model complex relationships between covariates and outcomes. By orthogonalizing the estimation process, DML effectively mitigates biases arising from nuisance parameter estimation, leading to more robust causal inferences. Specifically, the use of machine learning models like Random Forest in DML provides an advantage due to their ability to capture non-linear relationships and interactions between covariates, thereby improving the accuracy and stability of the treatment effect estimates. The superior performance of Random Forest in estimating the ATE compared to Neural Networks and XGBoost in this simulation can be attributed to the dataset size and complexity. Random Forest, known for its robustness and ability to handle small to medium-sized datasets effectively, reduces overfitting and captures essential patterns without requiring large datasets. In contrast, Neural Networks and XGBoost, while more advanced, require larger datasets to optimize their complex architectures and fully leverage their capabilities. The IHDP dataset, with its limited size, is insufficient to support these complex models, leading to higher variance and less stable estimates. Therefore, Random Forest's simplicity and robustness make it better suited for smaller dataset, providing more precise and stable ATE estimates.

PSM, on the other hand, relies on the assumption that all confounding variables are adequately controlled through the propensity score model. This method can be sensitive to model misspecification and may result in higher variability when the propensity score model does not fully capture the underlying data structure. This sensitivity is reflected in the higher variance and wider confidence intervals observed in the PSM estimates in our simulation.

It is important to note that these results were drawn using the Infant Health and Development Program (IHDP) dataset. While the IHDP dataset is well-suited for benchmarking causal inference methods, it has limitations that must be considered. The dataset size and the specific covariate structures may influence the generalizability of the findings. In real-world applications, datasets may vary in size and complexity, potentially affecting the performance of both PSM and DML methods. Therefore, while the conclusions from this study are informative, further validation with other datasets and in different contexts is necessary to fully understand the robustness and applicability of these methods.

4 Appendix

For the code of the simulation in this study, please see the following GitHub repository: Causal Inference Repository.

References

[Aus11] Peter C Austin. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate behavioral research*, 46(3):399–424, 2011.

[CCD$^+$17] Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and causal parameters, 2017.

[FBP24] Jonathan Fuhr, Philipp Berens, and Dominik Papies. Estimating causal effects with double machine learning – a method evaluation, 2024.

[GR93] Xing Sam Gu and Paul R Rosenbaum. Comparison of multivariate matching methods: Structures, distances, and algorithms. *Journal of Computational and Graphical Statistics*, 2(4):405–420, 1993.

[HR06] Jennifer Hill and Jerome P Reiter. Interval estimation for treatment effects using propensity score matching. *Statistics in medicine*, 25(13):2230–2256, 2006.

[Imb04] Guido W. Imbens. Nonparametric Estimation of Average Treatment Effects Under Exogeneity: A Review. *The Review of Economics and Statistics*, 86(1):4–29, February 2004. _eprint: https://direct.mit.edu/rest/article-pdf/86/1/4/1613802/003465304323023651.pdf.

[MR00] Kewei Ming and Paul R Rosenbaum. Substantial gains in bias reduction from matching with a variable number of controls. *Biometrics*, 56(1):118–124, 2000.

[MT08] Stephen L Morgan and Jennifer J Todd. A diagnostic routine for the detection of consequential heterogeneity of causal effects. *Sociological Methodology*, 38(1):231–281, 2008.

[RR84] Paul R. Rosenbaum and Donald B. Rubin. Reducing bias in observational studies using subclassification on the propensity score. *Journal of the American Statistical Association*, 79(387):516–524, 1984.