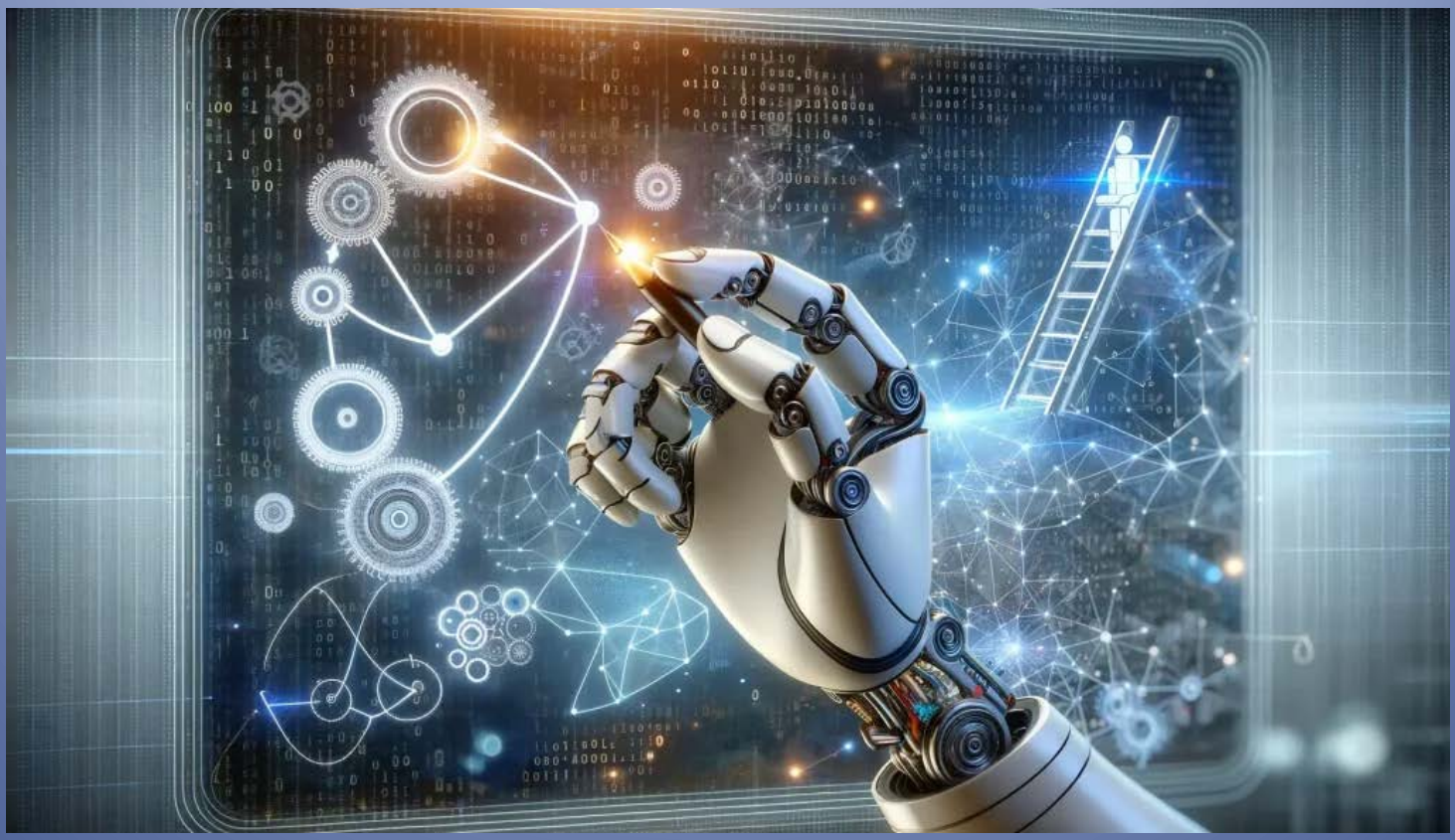


Machine Learning for Causal Inference with Applications in Health Care

Mandana Tabrizi

Winter 2024

Contents

Chapter 1: Defining Data, Model and Parameter

There are multiple ways to conduct research and collect data, and understanding the different ways to do so is important in interpreting the results of the research. The two main types of research are observational studies and experiments

Observational vs Experimental Studies

Observational studies: An observational study is when the researcher observes the effect of a specific variable as it occurs naturally, without making any attempt to intervene. These studies can identify associations or correlations between variables but cannot definitively establish causation. This is because researchers do not manipulate variables and therefore cannot control for all potential confounding factors.

Experimental studies: By actively manipulating one or more variables and controlling for other factors through random assignment, experimental studies can establish causality. Researchers can confidently conclude that changes in the independent variable directly cause changes in the dependent variable, as they have control over potential confounding variables.

Traditional method to find causal effect

The traditional approach is to fit several regression models and select the favorite one. Report point estimate of coefficient in front of treatment, confidence intervals and p-value, as if the parametric model was priori.

There are several problems with traditional approach in observational data. The first one is that the parametric model is usually misspecified, but parameter estimates are interpreted as if the model is correct. This challenge becomes more pronounced if the true data generating distribution is complex and we have many covariates.

The second problem is that traditional approach does not typically make causal assumptions which allow us to define the causal effect, and often don't include other assumptions such as positivity assumptions that are necessary for causal model. Also, estimates of variance don't account for model selection, and

therefore confidence intervals and p-values are wrong, even if the final model is somehow correct.

Data, model and target parameter

This section aims to provide formal definitions about data, model, and target parameter using notation. We denote *O* as the random variable with P_0 representing the corresponding probability distribution. In a simple scenario, our random variable, observed n times, could be represented as $O = (W, A, Y) \sim P_0$, assuming no common issues such as missing data or censoring. However, real-world data structures are often more complex, as seen in cases like right-censored data, where certain variables may not be observed until the study's end. For example, if we track subjects for 5 years and some drop out before completion, resulting in censoring, our data structure becomes more intricate. This type of censoring, related to a desired full-data structure, is known as right censoring, as timelines are typically numbered from left to right, with the right portion being censored.

The model

The data generating distribution P_0 is included as a part of a statistical model *M*, denoted as $P_0 \in M$. In other words, *M* is the set of possible probability distributions for P_0.

How do we find a model that represents the true distribution of data P_0? One way is to choose parametric algorithm which simplify the distribution to a known form. A parametric model summarizes data with a set of parameters of fixed size (independent of the number of training examples).

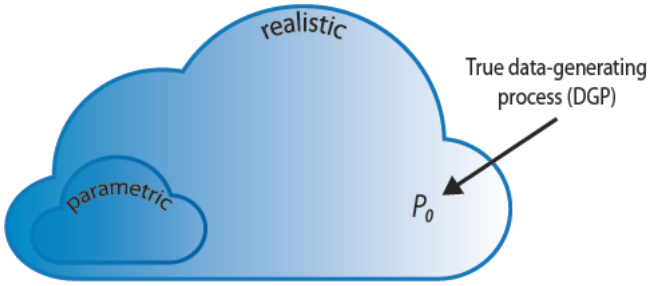
Algorithms that do not make strong assumptions about the form of the mapping function are called nonparametric algorithms. By not making assumptions, they are free to learn any functional form from the training data. Nonparametric methods seek to best fit the training data in constructing the mapping function, whilst maintaining some ability to generalize to unseen data.

If we possess supplementary information regarding how our data were generated, imposing constraints on the data-generating distribution P_0, we may

also contemplate a semiparametric statistical model. For instance, we might know that the effect of exposure A on the average outcome follows a linear function.

Statistical models vs Causal Models

Statistical models focus on describing patterns and making predictions based on observed data, causal models aim to uncover the underlying causal mechanisms that drive those patterns and relationships. Causal models require more stringent assumptions and are typically used when the goal is to understand and intervene on causal relationships between variables.



Target parameter

Our primary aim when analyzing our data often revolves around quantifying differences in the probability distribution of a particular outcome between groups subjected to different treatments or exposures. We seek to comprehend how treatment or exposure influences the probability distribution of the outcome, whether on an additive or multiplicative scale, such as relative risk or odds ratio. Once a decision is reached on the parameter of interest, we can explicitly define the target parameter,$\psi(P_0)$, as a function of the probability distribution P_0.

Numerous practitioners are accustomed to conceptualizing their parameter within the framework of a regression coefficient. However, such an approach is frequently unfeasible within practical nonparametric and semiparametric statistical models. Our focus lies in defining the parameter based on the knowledge about true distribution P_0, which is included in a nonparametric or semiparametric statistical model M, $P_0 \in M$.

Given the data structure $O = (W, A, Y) \sim P_0$, the parameter of interest is called Average Treatment Effect (ATE) and is the following function of the distribution P_0:

$$\psi(P_0) = E_{W,0}\left[E_0(Y|\, A = 1,\, W\right) - E_0(Y|\, A = 0,\, W)]\ (*)$$

After obtaining $\psi(P_0)$, we can interpret this parameter in two ways. One as purely statistical parameter of P_0, and one as a causal parameter under additional causal assumptions. Here are the assumptions we need to add to the statistical model in order to be able to interpret it as a causal model.

- No unmeasured confounding (randomization assumption)
- $P_0(A = a|W = w) > 0$ (positivity assumption)

Now we can substitute $\bar{Q}_n\left(A_i\ ,\ W_i\right) =\ E_0(Y_i|A_i,\ W_i)$ and the empirical distribution of $W = 1/n$ in the formula (*) to get the *substitution estimator* for ATE

$$\psi_n = \Psi(Q_n) = \frac{1}{n}\sum_{i=1}^{n}\{\bar{Q}_n(1, W_i) - \bar{Q}_n(0, W_i)\},$$

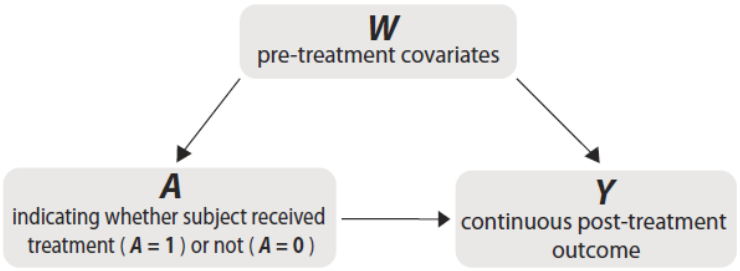


Figure: *Under the causal model, Average Treatment Effect answers the question: What is the average difference in outcomes between treatment groups when adjusting for covariates?*

Chapter 2 Super Learning

Super Learner Framework

For any given data application, we do not know which algorithms will work well. Iteratively examining the data and making modeling decisions can lead to overfitting.

The Super Learner framework is ideal in that you can lock down a large and diverse set of algorithms prior to any investigation of the data and still control for overfitting

Three components of Super Learning:

1. Loss function
2. Cross validation
3. Library

Loss function

A loss function assigns a measure of performance to a candidate function $\bar{Q}$ when applied to an observation O. That is, a loss function is a function L of the random variable O and parameter value $\bar{Q}$ given by

$$L : (O, \bar{Q}) \rightarrow L(O, \bar{Q}) \in \mathbb{R}.$$

Here are a few common loss functions used in the literature. The first one is the L_1 absolute error loss function

$$L(O, \bar{Q}) = |Y - \bar{Q}(A, W)|,$$

The L_2 squared error (or quadratic) loss function

$$L(O, \bar{Q}) = (Y - \bar{Q}(A, W))^2,$$

And the negative log loss function for a binary Y

$$L(O, \bar{Q}) = -\log(\bar{Q}(A, W)^Y (1 - \bar{Q}(A, W))^{1-Y}).$$

Now we define risk as expected loss which we want to be small. If we choose squared error loss, then we have

$$Risk \ = \ Expected \ loss \ = \ Mean \ squared \ error \ (MSE) =$$

$$E_0\big(L(O, \bar{Q})\big) = E_0(Y - \bar{Q}(A, W))^2$$

Risk (or MSE) quantifies the degree of variability between the algorithm's predictions and the true outcomes, providing insight into the algorithm's performance and predictive accuracy.

Cross Validation

Training a prediction function and testing it on the same data is a methodological mistake which causes overfitting. In this scenario, the model may perfectly reproduce seen data but fail to predict unseen data accurately.

The most common strategy to address this issue is k-fold cross-validation. In this approach the dataset is divided into k equal-sized folds. The model is trained k times, each time using a different fold as the validation set and the remaining folds as the training set. The risk from each iteration are then averaged to obtain an overall assessment of the model's performance.

Cross-validation helps to ensure that the model's performance is not overly dependent on a particular subset of the data and provides a more robust estimate of its generalization performance. It is especially useful when the dataset is limited in size or when the model is prone to overfitting.

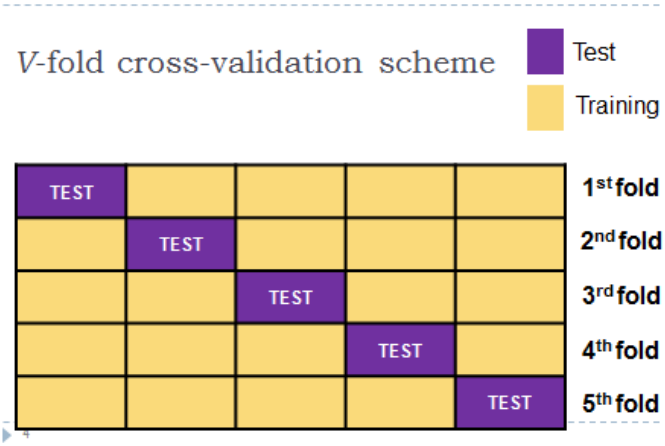


Illustration: V-fold cross-validation. A total of 5 folds will be trained and tested.

Library

The last component of super learner is the library which is a collection of algorithms. This collection can be large and includes other algorithms besides parametric statistical models, for example, it may include random forest algorithms and neural networks.

By incorporating a rich collection of algorithms that vary in bias and degree of data-fitting, the cross-validation within the machine learning prevents overfitting and it also prevents selecting a fit that is too biased.

Super (Machine) Learning

Suppose an example with data structure: $O = (W, A, Y) \sim P_0$. The outcome *Y* is binary, indicating death within 5 years of baseline, and *A* is also binary, indicating if the subject meets suggested levels of physical activity. Let us consider only the covariates $W = \{W1, W2, W3\}$. Age *(W1)* is a continuous variable, gender *(W2)* is binary, and chronic health *(W3)* is a binary variable indicating if the subject has a chronic health condition.

Now consider three different specifications of logistic regression of outcome *Y* on treatment *A* and covariates *W*. The first regression have only main terms. The second regression includes interaction of age and gender. The third statistical model uses main terms and age^2.

We would like to run all three of these statistical models and find out the best estimator. So we need to decide on how can we responsibly select the optimal estimator from a collection of algorithms?

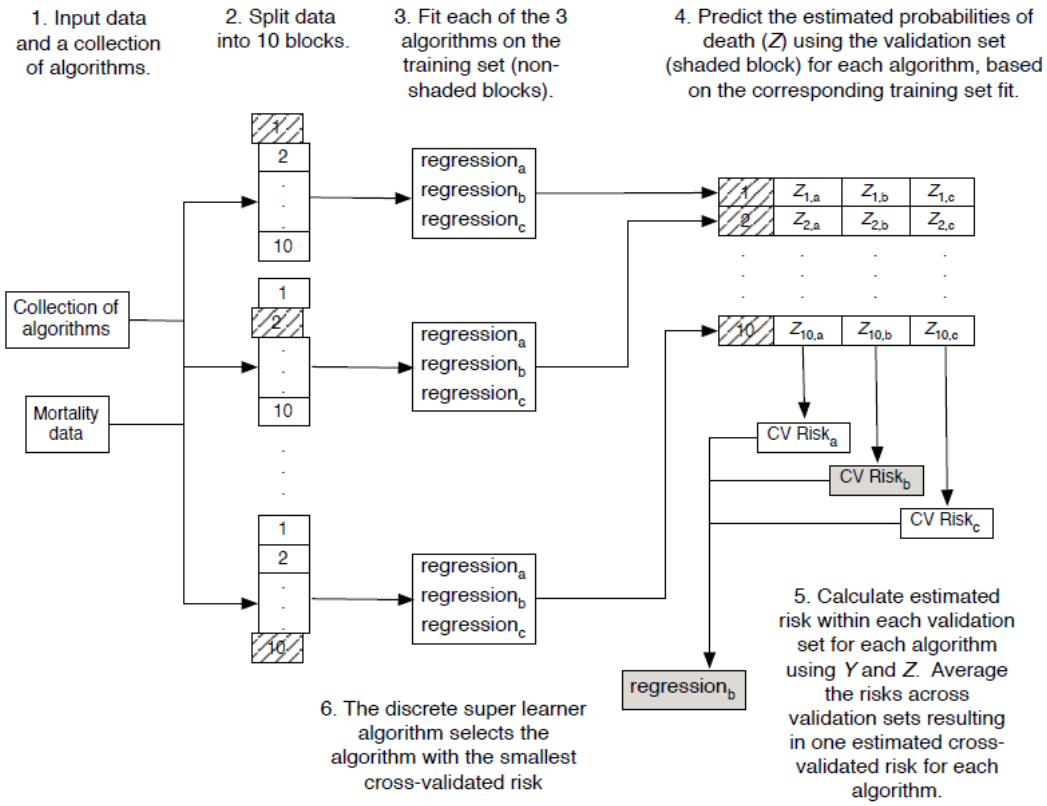
Discrete Super Learner

We introduce discrete super learner which provides us with an optimal estimator from a collection of algorithms by using cross-validated risk of each algorithm.

First we split the whole dataset into 10 groups (10 fold cross validation). Group 1 observations are set aside, and the first regression is trained on the remaining nine groups, known as the training set. Subsequently, the observations from group 1, known as the validation set, are used to derive the predicted probabilities of deaths using the regression fit obtained on the training set.

This process is executed for each algorithm within the collection, resulting in predicted probabilities for each of the three regressions at the end of the first fold. Additionally, we calculate a risk estimate within the validation set (group 1) for each of the three regressions, computed based on their respective predicted probabilities.

This procedure needs to be repeated nine more times to complete the process. So, the procedure continues until we have predicted probabilities of death for all observations for each algorithm, and also estimated risk within each validation set for each algorithm. Therefore, we obtain 10 estimated risks for each of the three algorithms, which are then averaged across validation sets, yielding one estimated cross-validated risk for each algorithm. The discrete super learner algorithm then chooses the algorithm with the lowest cross-validated risk as the "best algorithm."



Super Learner

It is reasonable to expect that the combination of three regressions may outperform any single regression alone. This basic concept enables us to compose a set of algorithms (here, our three regressions) into a library of weighted averages of these algorithms. We can then utilize the same cross-validation selector on this collection of potential algorithms, leading to the creation of the super learner.

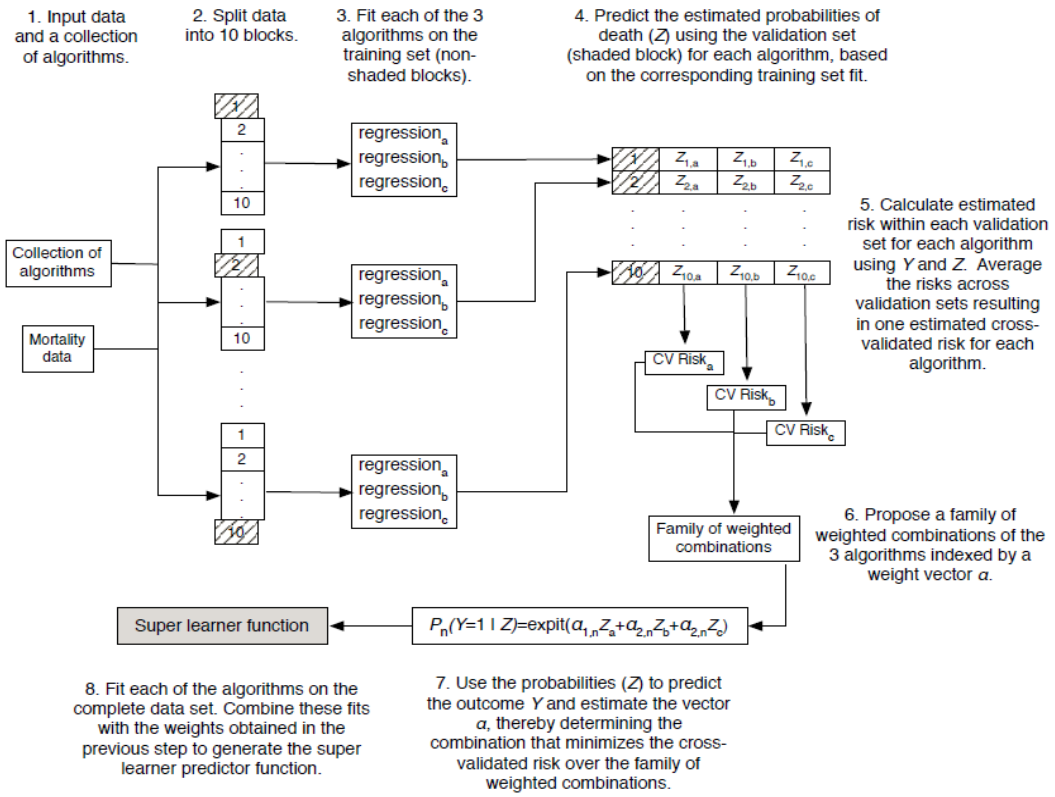
Given that the discrete super learner has been developed as outlined in the previous Section, we introduce a family of weighted combinations of the three regression algorithms, each identified by the weight vector α. Our objective is to identify the combination that minimizes the cross-validated risk.

This minimization problem can be formulated as a regression of the outcomes Y on the predicted values of the algorithms Z. Solving the regression will minimize the estimated expected squared error loss function and provides us with coefficients for the combination with the lowest cross-validated risk.

This selected weighted combination is a new estimator with the lowest cross-validated risk and is considered the "best" estimator. Now, we can utilize input data, such as our entire learning dataset or a new dataset, to estimate predicted probabilities. The picture below shows the full implementation of the super learner algorithm. To compute an honest risk for the super learner, the super learner must undergo external cross-validation once the described procedure has been executed.

Evaluation of Super Learner in R

The SuperLearner package in R includes a convenient function, CV.SuperLearner(), for evaluation of a super learner method utilizing nested V-fold cross-validation to estimate the performance of the super learner using the data, while the function SuperLearner() is the primary function for the final predictor.



Simulation

To test the performance of super learner we did a simulation on four different types of functions defined by

- *Simulation 1*

$$Y = -2 * I(X < -3) + 2.55 * I(X > -2) - 2 * I(X > 0) + 4 * I(X > 2) - I(X > 3) + U$$

- *Simulation2* $Y = 6 + 0.4X - 0.36X^2 + 0.005X^3 + U$
- *Simulation3* $Y = 2.83 * \sin\left(\frac{\pi}{2}X\right) + U$
- *Simulation4* $Y = 4 * \sin(3\pi X) * I(X > 0)$

Where $U \sim N(0,1)$, $X \sim unif(-4,4)$ *and* $I(.)$ is the indicator function.

For each simulation we produced a scatterplot from a sample of the simulation. The true curve for the simulation is represented by the solid black line. The red

dots represent the simulated data and blue dots represent the predictions from super learner fit. Also, for each simulation, we produced a plot for V-fold cross-validated risk estimate for the super learner and all algorithms in SL.library for comparison.

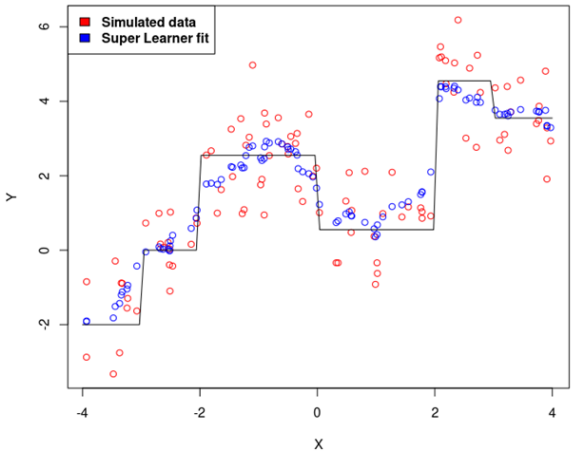
Set of Algorithms for Super Learner

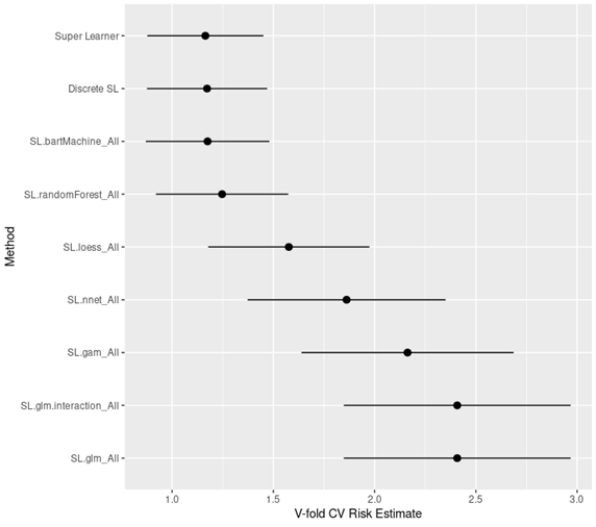
The set of algorithms in the collection should be a diverse set to adapt to different types of data generating models. Here we have regression, regression trees, random forest and neural networks.

R Algorithm	Description	Source
glm	linear models	R Development Core Team (2010)
interaction	polynomial linear models	R Development Core Team (2010)
randomForest	random forest	Liaw and Wiener (2002), Breiman (2001b)
gam	generalized additive models	Hastie (1992), Hastie and Tibshirani (1990)
nnet	neural network	Venables and Ripley (2002)
bart	baysian additive regression trees	Chipman and McCulloch (2009), Chipman et al. (2010)
loess	local polynomial regression	Cleveland et al. (1992)

The first simulation function is

$$Y = -2 * I(X < -3) + 2.55 * I(X > -2) - 2 * I(X > 0) + 4 * I(X > 2) - I(X > 3) + U$$



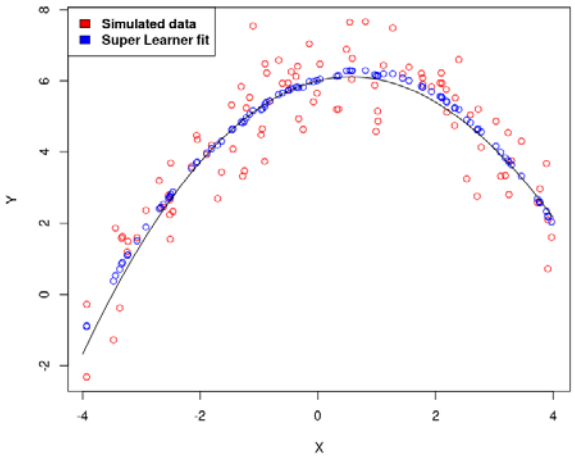


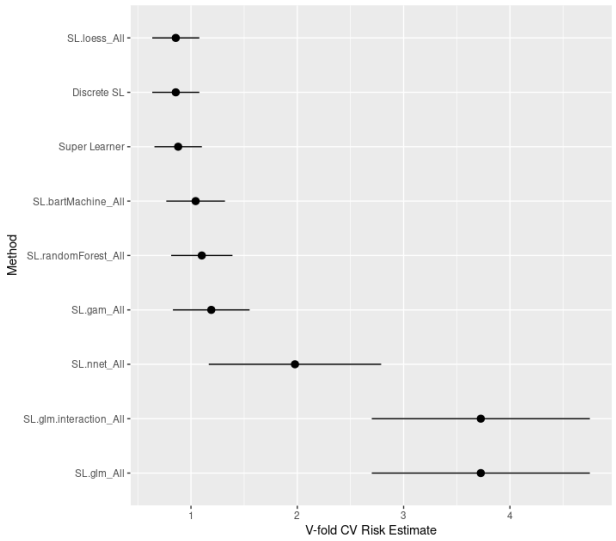
In this simulation, the best individual algorithm is bartMachine while glm has the largest risk. With the given collection of algorithms, the super learner is able to adapt to the underlying structure of the data-generating function and has a risk close to the best algorithm.

The second simulation function is

$$Y = 6 + 0.4X - 0.36X^2 + 0.005X^3 + U$$

In this simulation loess is beating the super learner by a small margin and perfectly fitting the data. Loess (locally weighted regression and scatterplot smoothing), is a weighted polynomial regression method that fits the data locally, iteratively within neighborhoods.

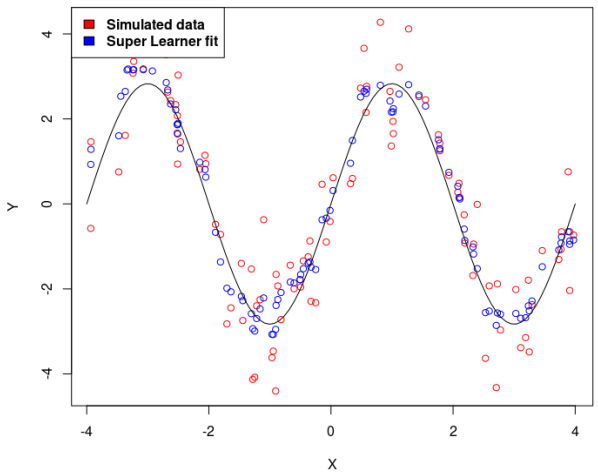


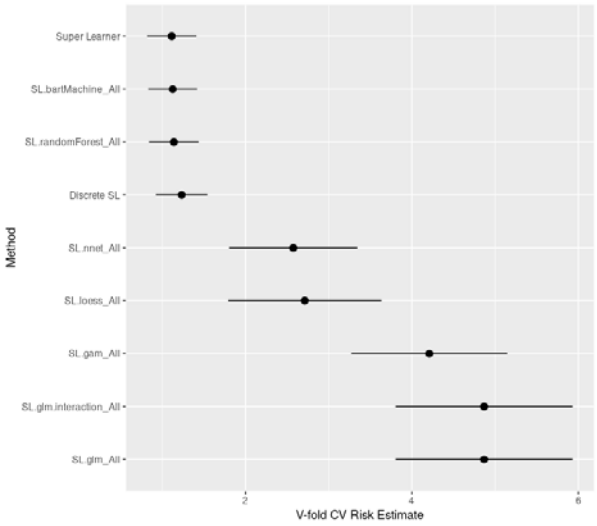


The third simulation function is

$$Y = 2.83 * \sin\left(\frac{\pi}{2}X\right) + U$$

In this simulation we have a periodic function and we can see that aggregated trees such as random forest and bartMachine produce an accurate prediction. Also we can see that loess which outperform other algorithms in the previous simulation, performs poorly here. So one algorithm that performs well in for one data set, may not perform well for a different dataset and we don't know which algorithm is best a priori. However, similar to the two simulations before, the super learner produces the result close to the best algorithm.

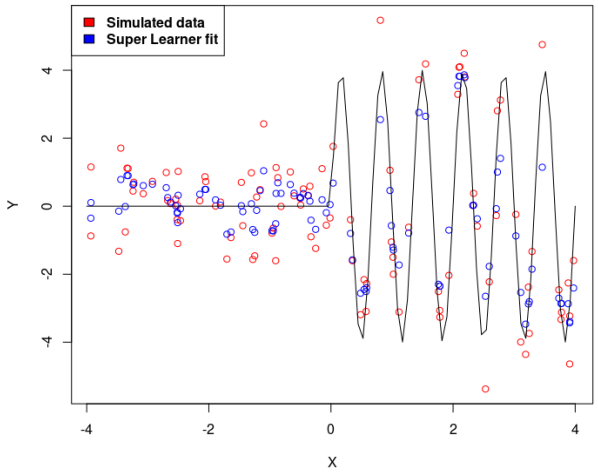


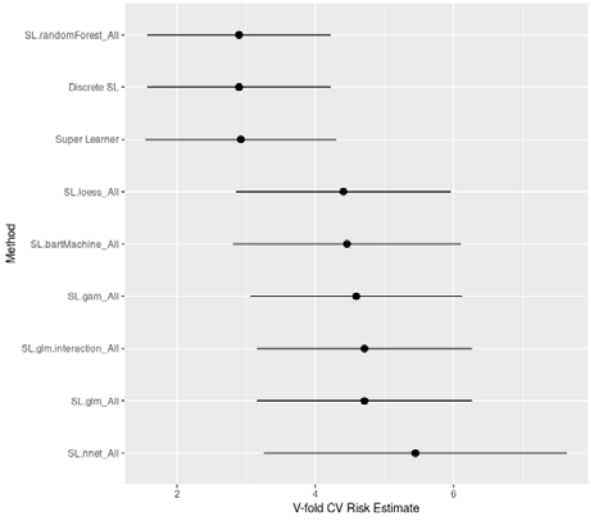


The fourth simulation function is

$$Y = 4 * \sin(3\pi X) * I(X > 0)$$

In this simulation, none of the algorithms in the collection are able to get a good prediction. The super learner does as well as the best algorithms in the library but does not attain the optimal value





Summary

The super learner provides a flexible but robust procedure for estimating an ensemble prediction model. It allows the researcher to evaluate a large library of prediction algorithms, but controls over-fitting with cross-validation

The framework easily extends to the high dimensional settings where p >> n by only changing the algorithms in the library. Large p examples also increases the number of algorithms in the library because various dimension reduction (screening) algorithms are possible to be used in conjunction with prediction algorithms

Chapter 3: Targeted Maximum Likelihood Estimation

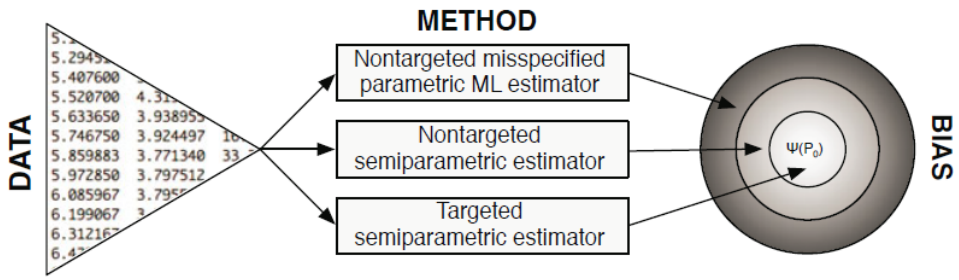
Several methods exist in the literature for estimating causal effects:

- Outcome regression methods
    - Model expected value of outcome given treatment and covariates
- Propensity score methods
    - Model conditional distribution of treatment given covariates
    - Matching, stratification
    - Inverse probability weighting (IPW)
- Double-Robust methods
    - consistent if either outcome regression or treatment model are correct
    - Targeted maximum likelihood estimation (TMLE)

TMLE

TMLE tailors estimation specifically for the parameter of interest, ψ_0. TMLE is a two stage procedure:

- Stage 1: obtain an estimate of the data-generating distribution P_0, or the relevant portion Q_0 of P_0.
- Stage 2: update this initial fit in a step targeted toward making an optimal bias–variance trade-off for the parameter of interest $\psi(Q_0)$, instead of the overall density P_0



TMLE Methodology

Step1: Estimate the Outcome

Estimate the expected value of the outcome using treatment and confounders

$$Q_n^0\left(A_i\ ,\ W_i\right) =\ E_0(Y_i|A_i,\ W_i)$$

Estimate the outcome for each observation where we set $a = 1\ and\ a = 0$

$$Q_n^0\ (1,\ W_i)\ and Q_n^0\ (0,\ W_i)$$

Substitute into the ATE estimator

$$\psi_{MLE,n} = \Psi(Q_n) = \frac{1}{n}\sum_{i=1}^{n}\{\bar{Q}_n^0(1, W_i) - \bar{Q}_n^0(0, W_i)\}.$$

Step2: Estimate the probability of Treatment

Estimate probability of treatment, given confounders (propensity score)

$$P_0(A|W)\sim \mathrm{g}_0$$

Estimate the propensity score for each observation where $a = 1\ and\ a = 0$

$$\mathrm{g}_n\ (1,\ W_i)\ and\ \mathrm{g}_n\ (0,\ W_i)$$

Step3: Estimate the fluctuation parameter

Use propensity score to create a clever covariate to define a parametric model for fluctuations of the initial estimator

$$H_n^*(A, W) \equiv \left(\frac{I(A = 1)}{g_n(1 \mid W)} - \frac{I(A = 0)}{g_n(0 \mid W)}\right).$$

Run a logistic regression of our outcome *Y* on the clever covariate using as intercept the offset $logit(Q_n^0\ (A,\ W))$to obtain the estimate ϵ_n

$$logit(Y) = logit\left(\bar{Q}_n^0\ (A, W)\right) + \epsilon_n H_n^*\ (A,\ W)$$

Step 4: Update the Initial Estimates of the Expected Outcome

Update the estimate Q_n^0 into a new estimate Q_n^1

$$\text{logit}\, \bar{Q}_n^1(A, W) = \text{logit}\, \bar{Q}_n^0(A, W) + \epsilon_n H_n^*(A, W).$$

Update the prediction at $a = 1$ for all observations

$$\text{logit}\, \bar{Q}_n^1(1, W) = \text{logit} \bar{Q}_n^0(1, W) + \epsilon_n H_n^*(1, W),$$

Update the prediction at $a = 0$ for all observations

$$\text{logit}\, \bar{Q}_n^1(0, W) = \text{logit} \bar{Q}_n^0(0, W) + \epsilon_n H_n^*(0, W)$$

Step5: Compute the statistical estimand of Interest

Using updated estimates, the formula from step1 becomes

$$\psi_{TMLE,n} = \Psi(Q_n^*) = \frac{1}{n} \sum_{i=1}^{n} \{\bar{Q}_n^1(1, W_i) - \bar{Q}_n^1(0, W_i)\}.$$

Influence Curve

In this section we provide some theory behind the targeting step. First we define influence curve (IC) for TMLE. IC will be used in calculating statistical inferences and also explains where does clever covariate $H(A, W)$ comes from.

Influence curve is a function that describes estimator behavior under slight perturbations of the empirical distribution of the data. Empirical mean of IC for regular asymptotically linear (RAL) estimator provides linear approximation of estimator. Among all ICs for all RAL estimators of the same statistical parameter, the one with minimum variance is the Efficient Influence Curve.

$$IC_n(O_i) = \overbrace{\left(\frac{I(A_i = 1)}{g_n(1 \mid W_i)} - \frac{I(A_i = 0)}{g_n(0 \mid W_i)}\right)(Y - \bar{Q}_n^1(A_i, W_i))}^{a} + \underbrace{\bar{Q}_n^1(1, W_i) - \bar{Q}_n^1(0, W_i) - \psi_{TMLE,n}}_{b},$$

Now we provide a heuristic proof that targeting ensures the empirical mean of the efficient influence curve is 0. Remember the TMLE is a substitution estimator

$$\psi_{TMLE,n} = \Psi(Q_n^*) = \frac{1}{n}\sum_{i=1}^{n}\{\bar{Q}_n^1(1, W_i) - \bar{Q}_n^1(0, W_i)\}.$$

So empirical mean of *b* equals 0.

We show that targeting step arranges for *a* to also have mean 0. Define a parametric submodel

$$logit(Y) = logit\big(\bar{Q}_n^0\ (A, W)\big) + \epsilon_n H_n^*\ (A,\ W)$$

Now use maximum likelihood to fit ϵ. MLE solves the score equation

$$\sum_{i=1}^{n} H(A_i, W_i)[logit(Y_i) - logit(\bar{Q}_n^*(A_i, W_i))] = 0$$

Where

$$logit(\bar{Q}_n^*(A, W) = logit\big(\bar{Q}_n^0\ (A, W)\big) + \epsilon_n H_n^*\ (A,\ W)$$

An obvious choice for $H_n^*\ (A,\ W)$ would be

$$H_n^*(A, W) \equiv \left(\frac{I(A = 1)}{g_n(1 \mid W)} - \frac{I(A = 0)}{g_n(0 \mid W)}\right).$$

By choosing $H_n^*\ (A,\ W)$ as above we will make sure that mean of *a* = 0 and since we have already seen mean of *b*=0, therefore we have mean of IC = 0 at the updated outcome.

An important property of IC is that the empirical mean of the influence curve equals TMLE estimator minus its estimand. Or in other words, TMLE is asymptotically linear

$$\sqrt{n}\big(\psi_{TMLE,n} - \psi_0\big) \xrightarrow{D} N(0, \sigma^2)$$

Statistical Inference

In Machine learning models sampling distribution is not asymptotically linear, and so inference is not possible. But TMLE is asymptotically linear (and thus normally distributed) estimator that accommodates inference via asymptotically consistent Wald-style confidence intervals.

Since TMLE yields an asymptotically linear estimator, obtaining statistical inference is very convenient. As we have seen in the previous section, each TML estimator has a corresponding (efficient) influence curve that describes the asymptotic distribution of the estimator. By using the estimated IC, Wald-style inference (asymptotically correct confidence intervals) can be constructed simply by plugging into the form of the IC our initial estimates $\bar{Q}_n$ and g_n, then computing the sample standard error.

Sample mean

$$\bar{IC}_n = \frac{1}{n}\sum_{i=1}^{n} IC_n(o_i),$$

Sample Variance

$$S^2(IC_n) = \frac{1}{n}\sum_{i=1}^{n}\left(IC_n(o_i) - \bar{IC}_n\right)^2.$$

Standard error

$$\sigma_n = \sqrt{\frac{S^2(IC_n)}{n}}.$$

95% Wald-type confidence interval

$$\psi_{TMLE,n} \pm z_{0.975}\frac{\sigma_n}{\sqrt{n}},$$

Finally p-value is calculated by

$$2\left[1 - \Phi\left(\left|\frac{\psi_{TMLE,n}}{\sigma_n/\sqrt{n}}\right|\right)\right],$$

Where Φ(.) is the cumulative distribution function of standard normal.

TMLE Properties

- Double robustness: Consistent estimator of ψ_0 when either the outcome or the propensity score mechanism are estimated consistently
- Asymptotically linear: Allows for construction of Wald-type confidence intervals, even when using machine learning
- Efficient: Achieves the Cramer-Rao efficiency bound when both models are consistent
- Substitution estimator: Can be expressed in terms of the portion of likelihood relevant for evaluating ψ_0

Simulation 1

Double robustness property of TMLE is achieved by adding the targeting step. In other words, targeting gives us a second chance to get it right. If the outcome regression model is wrong, use info in propensity score to reduce bias in *ATE*. Even if outcome model is right, in a high dimensional setting we may be able to make a better bias/variance trade-off for *ATE*. In this simulation we explore the double robustness property.

Dataset

Data has the structure O=(A, W1, W2, W3,Y)

The confounders W = {W1, W2, W3} are binary with corresponding mean values of (0.55, 0.30, 0.25)

The exposure A is binary with logit(P(A = 1|W1, W2, W3)) = −0.5 + 0.75W1 + W2 +1.5W3.

The outcome Y is normal with standard deviation of 4.5 and the following mean value: E (Y |A, W1, W2, W3) = 24 − 3A + 3W1 – 4W2 – 6W3−1.5A*W3.

The true marginal effect in our simulated data (or true population ATE) is −3.38.

Simulation Design:

Since the goal of this simulation is mainly to check the double robustness property of TMLE, we tried different types of model misspecifications as described below:

Outcome/Exposure misspecification

- the exposure was correctly specified but the outcome regression was misspecified
- the exposure regression was misspecified but the outcome was correctly specified.
- both outcome and exposure were misspecified.

Variable misspecification

- The "main-terms" misspecification included only main-effect terms for A and W1, W2, W3, omitting the interaction term A × W3.
- The "omitted variable" misspecification included main terms for A and tW1 and W2, omitting W3 entirely

Unmeasured confounder

- Both outcome and exposure were misspecified by taking out W3 from confounder matrix

set of Algorithms for Super Learner

R Algorithm	Description	Source
glm	linear models	R Development Core Team (2010)
interaction	polynomial linear models	R Development Core Team (2010)
randomForest	random forest	Liaw and Wiener (2002), Breiman (2001b)
gam	generalized additive models	Hastie (1992), Hastie and Tibshirani (1990)
nnet	neural network	Venables and Ripley (2002)
bart	baysian additive regression trees	Chipman and McCulloch (2009), Chipman et al. (2010)
loess	local polynomial regression	Cleveland et al. (1992)

Results

The table below shows the results. The bias is small when one of the outcome/exposure is misspecified but for the cases of double misspecification and unmeasured confounder the bias is about 50%. The variance is almost the same for all models, around 0.3.

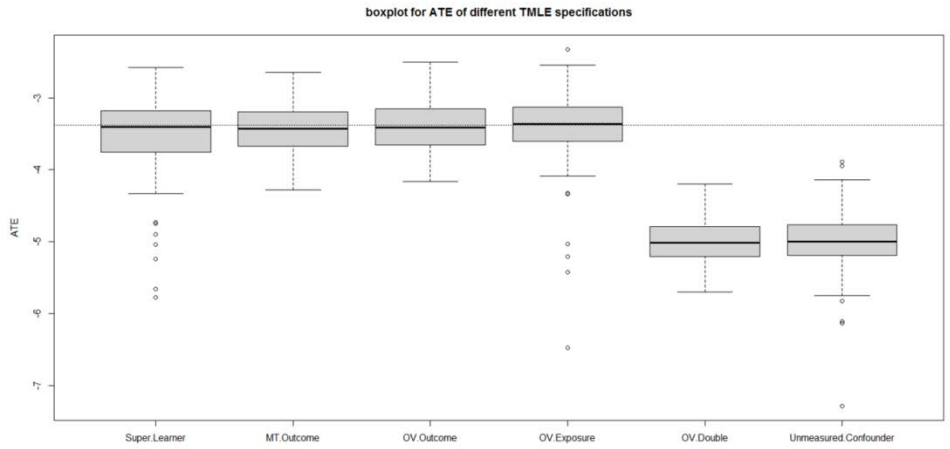
TMLE					
Super learner		Avg ATE	SD_ATE	Bias	95% CI
	Outcome variables: A, W1, W2, W3	-3.36	0.30	0.02	(-3.89,-2.80)
	exposure variables: W1, W2, W3				
Misspecified parametric regression					
	Main-terms misspecification	-3.41	0.34	-0.03	(-4.03,-2.70)
	Outcome variables: A, W1, W2, W3				
	Omitted-variable misspecification	-3.41	0.34	-0.03	(-4.06,-2.83)
	Outcome variables: A, W1, W2				
	Omitted-variable misspecification	-3.30	0.36	0.08	(-4.03,-2.66)
	exposure variables: W1, W2				
	Double misspecification	-5.00	0.32	-1.62	(-5.62,-4.41)
	Outcome variables: A, W1, W2				
	exposure variables: W1, W2				
Unmeasured Confounder					
	Outcome variables: A, W1, W2	-4.94	0.31	-1.56	(-5.56,-4.36)
	exposure variables: W1, W2				
G-Computation					
Super learner					
	Outcome variables: A, W1, W2, W3	-3.23	0.33	0.15	(-3.96, -2.58)
Misspecified parametric regression					
	Main-terms misspecification	-3.26	0.32	0.12	(-3.93, -2.61)
	Outcome variables: A, W1, W2, W3				
	Omitted-variable misspecification	-4.89	0.36	-1.51	(-5.62, -4.24)
	Outcome variables: A, W1, W2				

Table. Estimates of ATE, Bias and 95% CI in a Simulation Study of TMLE and G-Computation

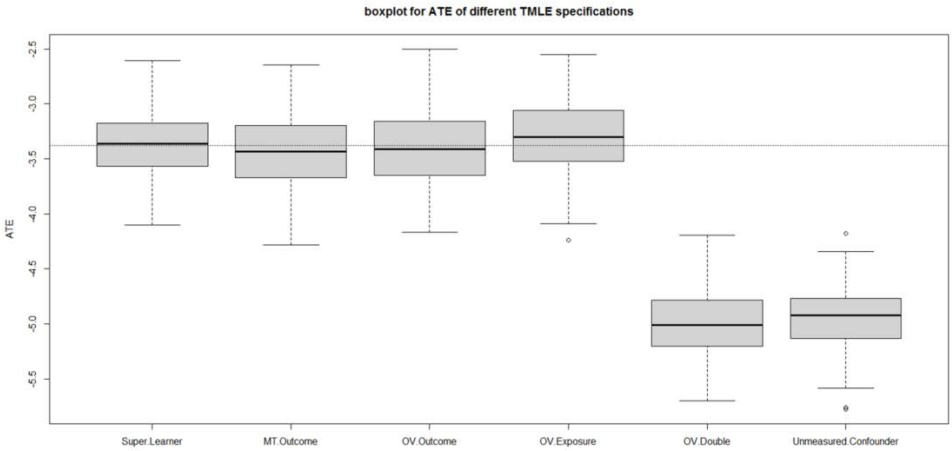
Abbreviations: ATE, average treatment effect; CI, confidence interval; SD, standard deviation. Each method was implemented with both the super learner and misspecified parametric regression. The true Y was generated with the terms A, W1, W2, W3, and A × W3, and A was generated with W1, W2 and W3. Misspecified parametric regressions included outcome main-terms misspecification, outcome omitted-variable misspecification, exposure omitted-variable misspecification and finally both outcome and exposure misspecification.

Boxplot below captures the results for different specifications of TMLE. We noticed that the models which are using super learner for outcome estimation are producing some outliers. These models include Super Learner, OV Exposure, Unmeasured Confounder (the other three models are using misspecified regression for outcome).

We examined how these outliers are generated and discovered that neural network has the largest coefficient, thus exerting the greatest impact in creating the outliers.



The plot below illustrates the results when we removed neural networks from the library. As we can see, the outliers that were skewing our results have disappeared and the variance improved.



Summary

In summary, TMLE+ SL produces the best estimate. TMLE's double robustness ensures unbiasedness of the ATE if either the exposure or the outcome mechanism is consistently estimated. If both outcome and exposure are misspecified or we have unmeasured confounder then TMLE cant produce accurate results.

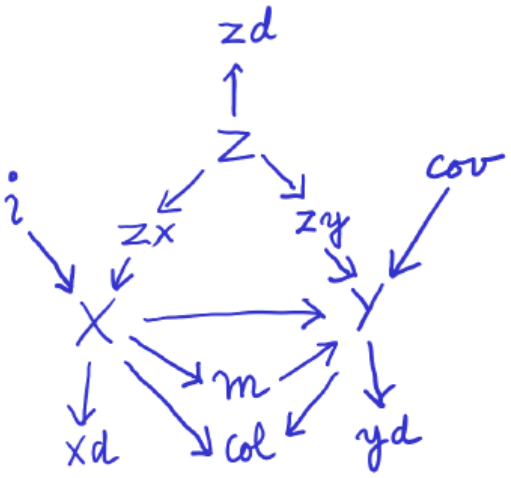
We can add powerful algorithms such as nnet and xgboost to super learner but they need precise hyperparameter tuning (which is time consuming). And if they aren't tuned properly they can create inaccurate estimates

Simulation 2

Data

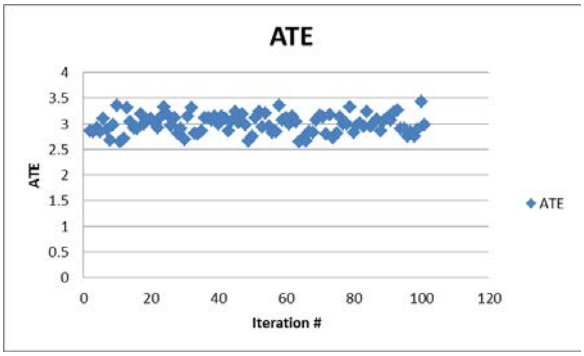
In this simulation we consider data generated by the below causal graph. According to the adjacency matrix, the true ATE = 3 for direct causal effect.

	z	zx	zy	cov	x	y	m	i	xd	yd	zd	col
z	1	0	0	0	0	0	0	0	0	0	0	0
zx	3	1	0	0	0	0	0	0	0	0	0	0
zy	3	0	1	0	0	0	0	0	0	0	0	0
cov	0	0	0	1	0	0	0	0	0	0	0	0
x	0	1	0	0	1	0	0	1	0	0	0	0
y	0	0	2	2	3	4	1	0	0	0	0	0
m	0	0	0	0	1	0	1	0	0	0	0	0
i	0	0	0	0	0	0	0	2.4	0	0	0	0
xd	0	0	0	0	1	0	0	0	1	0	0	0
yd	0	0	0	0	0	1	0	0	0	1	0	0
zd	2	0	0	0	0	0	0	0	0	0	1	0
col	0	0	0	0	1	1	0	0	0	0	0	1



Results

For simplicity we only considered the data $O = (Z, ZX, ZY, X, Y)$, then generated 100 samples of size 250. Average ATE as well as standard deviation and bias are summarized in the table below. The graph also shows the ATE for 100 samples . As we can see, TMLE produces accurate results with small standard deviation and bias.



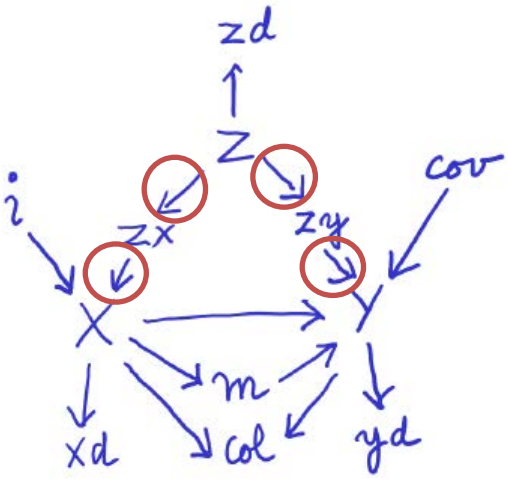
Avg_ATE	SD_ATE	Avg_Bias
2.997	0.182	-0.003

Simulation 3

Data

Now we add (squared and cubic) curvature to the edges of the DAG marked by a red circle. Therefore we have 8 possible curves:

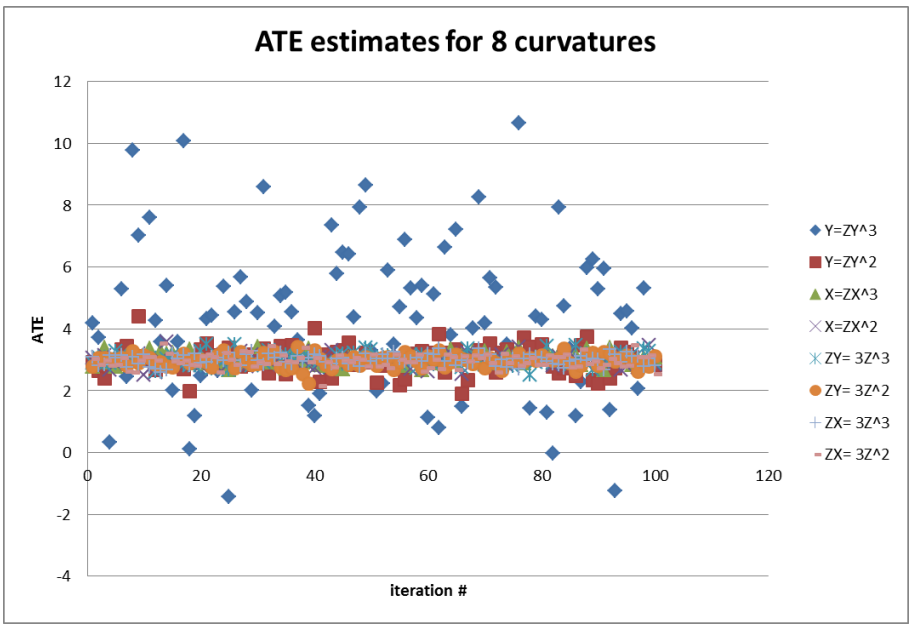
- $ZX = 3Z^2 + U$
- $ZX = 3Z^3 + U$
- $ZY = 3Z^2 + U$
- $ZY = 3Z^3 + U$
- $P(X = 1) = 1/(1 + e^{-ZX^2})$
- $P(X = 1) = 1/(1 + e^{-ZX^3})$
- $Y = 3X + 2(ZY^2) + U$
- $Y = 3X + 2(ZY^3) + U$



Results

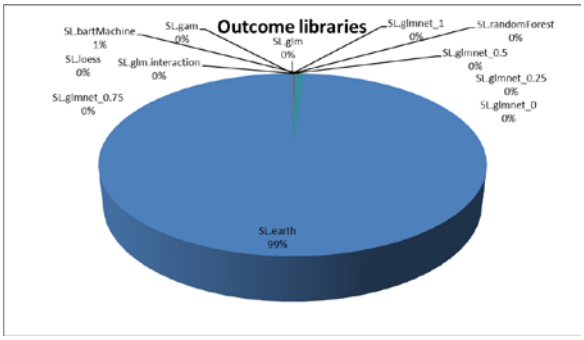
The ATE results for 8 curvatures are summarized in the scatterplot below

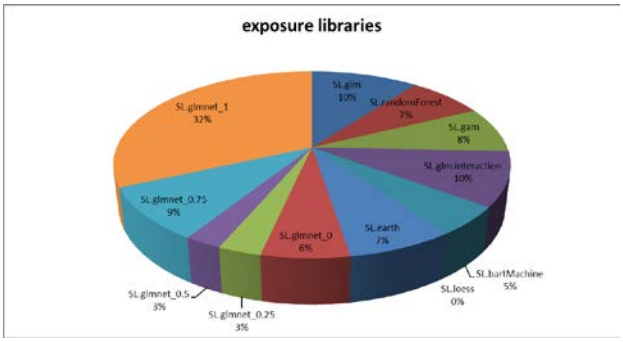
.



As we can see, all the 8 curvatures produce close estimates to the true ATE except for $Y = 3X + 2(ZY^3) + U$ (blue points). ($Y = 3X + 2(ZY^2) + U$ (red points) also has higher variance, albeit not to the extent observed in the blue points.)

To delve into the generation of estimates for the case of $Y = 3X + 2(ZY^3) + U$, we examine the coefficients of algorithms within the super learner library. The pie charts below illustrate the weights assigned to algorithms for estimating exposure and outcome.





For outcome estimation, the "earth" algorithm is assighned a 99% weighting on average, reflecting its expected significance in handling polynomial outcomes. Additionally, we incorporated glmnet into the library, followed by hyperparameter tuning of glmnet with with 5 different alpha = {0, 0.2, 0.4, 0.6, 0.8, 1.0} where 0 corresponds to ridge and 1 to lasso. Although these adjustments improved results, the standard deviation remains prohibitively high.

For exposure, the distribution of algorithms' weights appears to be highly varied. It seems that super learner cant decide which algorithm to choose, this could be because exposure is binary. Hence, we opted for discrete super learner, which selects the best algorithm, as opposed to ensemble super learner, which calculates a weighted average of algorithms. Surprisingly, this adjustment led to a deterioration in results.

Column1	earth	glmnet	glmnet with tuning
Avg_ATE	4.85	4.82	4.25
SD_ATE	3.61	3.07	2.42

Column1	Super Learner	Discrete Super Learner
Avg_ATE	4.25	4.49
SD_ATE	2.42	3.33

Future work

As demonstrated in simulation 3, the presence of high correlations among certain covariates poses a challenge, making it difficult to obtain stable, low-variance estimates of the association between treatment and outcome.

To address this challenge, we may consider employing C-TMLE. This method has shown effectiveness in handling complex scenarios. C-TMLE incorporates variable and model selection for the propensity score in a collaborative manner, directly optimizing the empirical metric on the causal estimator.

Also, we may explore alternative loss functions, such as employing the negative log loss function for binary treatment, or using the absolute error loss for Gaussian outcomes. The latter, whose expectation equals the conditional median, is more robust to outliers.

Conclusion

The super learner provides a flexible but robust procedure for estimating an ensemble prediction model. By incorporating a rich collection of algorithms that vary in bias and degree of data-fitting, the cross-validation within the super learning prevents overfitting and it also prevents selecting a fit that is too biased.

Targeted Learning integrates causal inference, machine learning and statistical theory. TMLE provides doubly robust estimation for complex observations, particularly high dimensional data. The estimate is accompanied with confidence interval and p-value.