

Statistical Methods for Causal Inference in Observational and Randomized Studies

Mark J. van der Laan¹, Maya L. Petersen¹, Sherri Rose²

¹University of California, Berkeley School of Public Health

²Johns Hopkins Bloomberg School of Public Health

laan@berkeley.edu · mayaliv@berkeley.edu · srose@jhsph.edu
stat.berkeley.edu/~laan/
works.bepress.com/maya-petersen/
drsherrirose.com

targetedlearningbook.com

September 27, 2011

DAY TWO: LECTURE ONE

Cross-validation and super learner

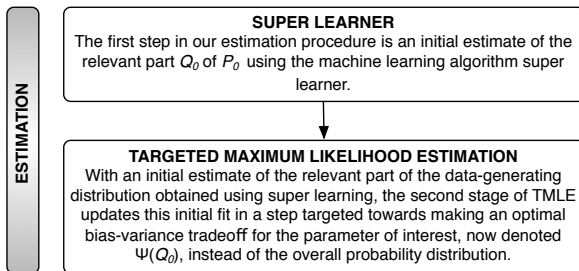
Review

Data, Model, Parameter.

We can explicitly define $\Psi()$ as a mapping from the statistical model to the parameter space $\Psi : \mathcal{M} \rightarrow \mathbb{R}$.

How does one translate the results from studies, how do we take the information in the data, and draw effective conclusions?

Road map - Estimation



Estimation

An **estimator** is an algorithm that can be applied to any empirical distribution to provide a mapping from the empirical distribution to the parameter space.

Effect Estimation vs. Prediction

Both **effect** and **prediction** research questions are inherently *estimation* questions, but they are distinct in their goals.

Effect: Interested in estimating the effect of exposure on outcome adjusted for covariates.

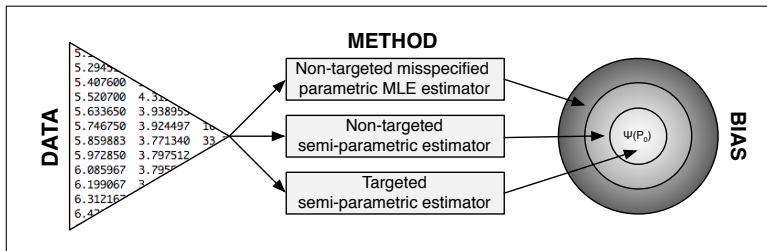
Prediction: Interested in generating a function to input covariates and predict a value for the outcome.

Effect parameters where no causal assumptions are made may be referred to as variable importance measures (VIMs).

Estimation in Misspecified Parametric Models

In most observational studies, standard practice for prediction and effect estimation involves assuming a parametric statistical model and using maximum likelihood estimation (MLE) to estimate the parameters in that statistical model.

Traditional Practice



Estimation in Misspecified Parametric Models

We pose a particular parametric regression, a so called linear regression model for the conditional mean of Y given A and W , $E_0(Y \mid A, W)$. However, we leave the distributions of A and W unspecified.

Parametric regression statistical models have varying levels of complexity, and what variables one includes impacts this complexity.

Estimation in Misspecified Parametric Models

High Dimensional Data

High dimensional data has become increasingly common, and researchers often have dozens, hundreds, or even thousands of potential confounders to include in their parametric regression.

Estimation in Misspecified Parametric Models

High Dimensional Data

Not only does this provide an impossible challenge to correctly specify the parametric regression, but the complexity of the statistical model may also increase to the point that there are more unknown parameters than observations.

Let's discuss an example from your research...

Estimation in Misspecified Parametric Models

High Dimensional Data

The true functional for $E_0(Y | A, W)$ might be described by a complex function not easily approximated by main terms or interaction terms.

The Complications of Human Art in Traditional Practice

The moment we use **post-hoc arbitrary criteria** and **human judgment** to select the parametric statistical model after looking at the data, the analysis becomes prone to additional bias. This bias manifests in both the effect estimate and the assessment of uncertainty for that estimate (i.e., standard errors).

So why not simply use a purely non-parametric model with high dimensional data?

- $p > n!$
- data sparsity

Example: Local Averaging

Local averaging of the outcome Y within covariate “neighborhoods.” Neighborhoods are bins for observations that are close in value. The number of neighborhoods will determine the smoothness of our regression function. How do you choose the size of these neighborhoods?

Example: Local Averaging

This becomes a **bias-variance** trade-off question.

- If we have many neighborhoods with small size, the estimate will not be smooth and will have high variance since some neighborhoods will be empty or contain only a small number of observations.
- On the other hand, if we have very few neighborhoods with large size, the estimate is much smoother, but it will be biased since the neighborhoods fail to capture the complexity of the data.

Example: LOESS

Locally weighted regression and scatterplot smoothing (LOESS) is a weighted polynomial regression method that fits the data locally, iteratively within neighborhoods.

Data-adaptive/Machine Learning

In the computer science literature, these methods are called machine learning. In statistics they are often referred to as data-adaptive.

A problem

If the true data-generating distribution is very smooth, a misspecified parametric regression might beat the semi-parametric estimator.

This is frustrating!

We want to create a smart semi-parametric estimator that is consistent, but in some cases it may “lose” to a misspecified parametric estimator because it is more variable.

The Dangers of Favoritism

- Relative Mean Squared Error (compared to main terms least squares regression) based on the validation sample

Method	Study 1	Study 2	Study 3	Study 4
Least Squares	1.00	1.00	1.00	1.00
LARS	0.91	0.95	1.00	0.91
D/S/A	0.22	0.95	1.04	0.43
Ridge	0.96	0.9	1.02	0.98
Random Forest	0.39	0.72	1.18	0.71
MARS	0.02	0.82	0.17	0.61

Super Learning in Prediction

Method	Study 1	Study 2	Study 3	Study 4	Overall
Least Squares	1.00	1.00	1.00	1.00	1.00
LARS	0.91	0.95	1.00	0.91	0.95
D/S/A	0.22	0.95	1.04	0.43	0.71
Ridge	0.96	0.9	1.02	0.98	1.00
Random Forest	0.39	0.72	1.18	0.71	0.91
MARS	0.02	0.82	0.17	0.61	0.38
Super Learner	0.02	0.67	0.16	0.22	0.19

Goals for Lecture

Loss Based Estimation

We will use **loss functions** to define the best estimator of $E_0(Y | A, W)$, and the evaluate it.

Cross Validation

Our available data is partitioned to **train** and **validate** our estimators.

Semi-Parametric Estimation

Allow the **data** to drive your **estimates**, but in an honest (cross validated) way.

These are detailed rigorous topics. We will cover core concepts.

The Question

What we want is an automated algorithm to semi-parametrically estimate $E_0(Y \mid A, W)$.

Loss-Based Estimation

What we want is an automated algorithm to semi-parametrically estimate $E_0(Y \mid A, W)$.

In the computer science literature, this is called machine learning.

In statistics these methods are often referred to as data-adaptive.

However, the essential point is that there are semi-parametric methods that also aim to “smooth” the data and estimate this regression function.

Loss-Based Estimation

There are many different potential algorithms we can implement to estimate $E_0(Y \mid A, W)$.

However, how are we to know which one to use a priori?

We cannot bet on a misspecified parametric regression, but we have a problem that one particular algorithm is going to do better than the other candidate estimators.

Loss-Based Estimation

Now that we know what our problem is, we can formally define our estimation problem and introduce cross validation, a method that will allow us to choose the best algorithm from a library of algorithms or produce a weighted combination of these algorithms.

Loss-Based Estimation

In our example, we wish to estimate: $\bar{Q}_0 = E_0(Y \mid A, W)$.

Before we can choose a “best” algorithm to estimate this regression function, we must have a way to define what “best” means. We do this in terms of a loss function.

Loss-Based Estimation

Our data structure is $O = (W, A, Y) \sim P_0$, with empirical distribution P_n which places probability $1/n$ on each observed O_i , $i = 1, \dots, n$. We choose the best algorithm in terms of a loss function which 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 :

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

It is a function of the random variable O and parameter value \bar{Q} .

Loss-Based Estimation

Examples of loss functions include 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:

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

Loss-Based Estimation

We define our parameter of interest, $\bar{Q}_0 = E_0(Y \mid A, W)$, as the minimizer of the expected squared error loss:

$$\bar{Q}_0 = \arg \min_{\bar{Q}} E_0 L(O, \bar{Q}),$$

where $L(O, \bar{Q}) = (Y - \bar{Q}(A, W))^2$. $E_0 L(O, \bar{Q})$, which we want to be small, evaluates the candidate \bar{Q} , and it is minimized at the optimal choice of \bar{Q}_0 . We refer to expected loss as the risk.

Loss-Based Estimation

We now have a way to define the “best” algorithm. We want the estimator of the regression function \bar{Q}_0 that minimizes the expectation of the squared error loss function.

Loss-Based Estimation

This makes a lot of sense intuitively. We want an estimator that has small bias and variance. How do we find out which algorithm meets our criteria? Now we need to understand the concept of cross-validation in order to obtain an accurate estimate of risk, so that we can select the algorithm that yields the minimal estimated risk.

Cross-Validation

Cross-validation involves partitioning the sample of n observations O_1, \dots, O_n in training and corresponding validation sets.

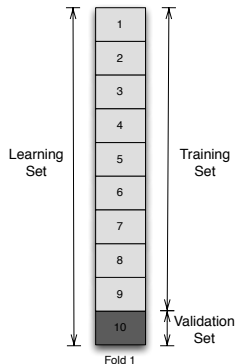
We will use cross-validation to select our “best” algorithm among a library of algorithms, which then defines our discrete super learner in terms of the library. In addition, we also use cross-validation to evaluate the overall performance of the super learner itself.

Cross-Validation

Cross-validation solves the problem of having many algorithms, and not knowing which one to use and helps us avoid overfitting.

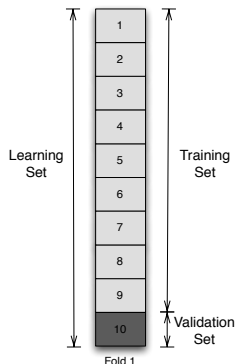
There are many forms of cross-validation, and here we will discuss V -fold cross-validation due to its desirable finite sample and asymptotic optimality property, which will be discussed later, and its computational ease.

Cross-Validation



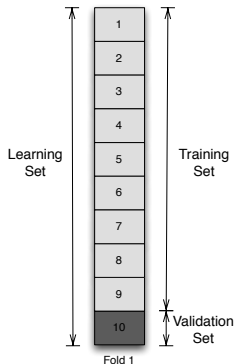
In V -fold cross-validation, our observed data O_1, \dots, O_n is referred to as the learning set.

Cross-Validation



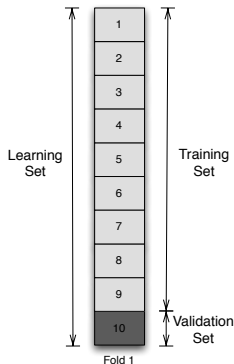
We wish to use the learning set to estimate our parameter of interest \bar{Q}_0 by partitioning into V sets of size $\approx \frac{n}{V}$.

Cross-Validation



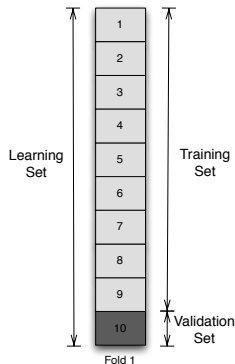
For any given fold, $V - 1$ sets will comprise the training set and the remaining 1 set is the validation set.

Cross-Validation



The observations in the training set are used to construct (or train) the candidate estimators.

Cross-Validation



The observations in the validation set are used to assess the performance (i.e., risk) of the candidate algorithms.

Cross-Validation

1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9
10	10	10	10	10	10	10	10	10	10
Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Fold 6	Fold 7	Fold 8	Fold 9	Fold 10

The validation set rotates V times such that each set is used as the validation set once.

Semi-Parametric Estimation

We've described, very generally, V-fold cross-validation.

Now we want to use cross-validation to determine the best algorithm based on estimated risk.

Discrete Super Learner

Discrete Super Learner

Suppose a researcher is interested in using three different parametric statistical models to estimate $E_0(Y | A, W)$.

We can use these algorithms to build a library of algorithms and select the one with the smallest (honest) cross-validated risk.

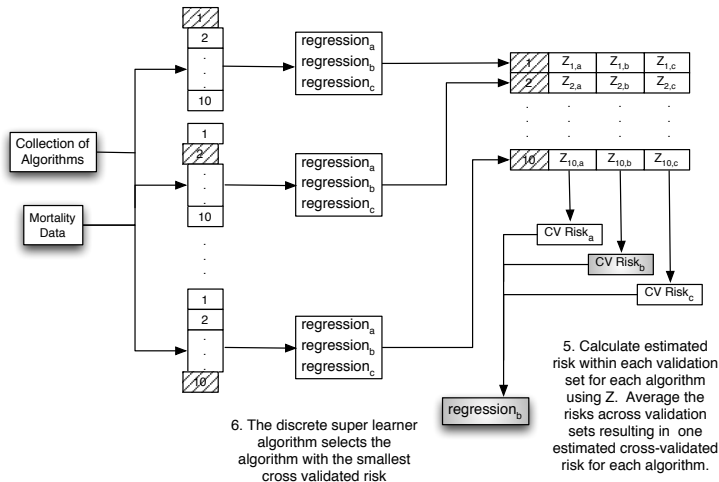
Discrete Super Learner

1. Input data and a collection of algorithms.

2. Split data into 10 blocks.

3. Fit each of the 3 algorithms on the training set (non-shaded blocks).

4. Predict the estimated probabilities of death (Z) using the validation set (shaded block) for each algorithm, based on the corresponding training set fit.



Discrete Super Learner

Method	Risk
MLE	0.30
DSA	0.04
Random Forest	0.23

Which one does the discrete super learner pick?

Discrete Super Learner

Method	Risk
MLE	0.90
DSA	0.63
Random Forest	0.50

Which one does the discrete super learner pick?

Oracle Properties

We now introduce the concept of the oracle selector, which is the best estimator given the K algorithms in the library of algorithms. The oracle selector chooses the algorithm with the smallest risk under P_0 , the true probability distribution of the random variable O . However, the oracle selector is unknown since it depends on both the observed data and P_0 .

Oracle Properties

Theory shows that that the discrete super learner performs as well as the oracle selector, up to a second order term. The loss function must be bounded, and then we will perform asymptotically as well as the algorithm selected by the oracle selector. The number of algorithms in the library can grow with samples size.

Super Learner

Super Learner (van der Laan, Polley, and Hubbard; 2007)

Allows researchers to use multiple algorithms to outperform a single algorithm in realistic non-parametric and semi-parametric statistical models that are based on actual knowledge.

The term algorithm is used very loosely to describe any mapping from the data into a predictor. This can range from a simple logistic regression to more complex algorithms such as neural nets.

Method

We define our parameter of interest, $\bar{Q}_0 = E_0(Y \mid A, W)$, as the minimizer of the expected squared error loss:

$$\bar{Q}_0 = \arg \min_{\bar{Q}} E_0 L(O, \bar{Q}),$$

where $L(O, \bar{Q}) = (Y - \bar{Q}(A, W))^2$. $E_0 L(O, \bar{Q})$, which we want to be small, evaluates the candidate \bar{Q} , and it is minimized at the optimal choice of \bar{Q}_0 . We refer to expected loss as the risk.

Can we improve upon the discrete super learner?

Yes!

We can use our algorithms to **build a library of algorithms** consisting of all weighted averages of the algorithms.

It is reasonable to expect that one of these weighted averages might perform better than one of the algorithms alone.

Methods

This simple principle allows us to *map a collection of candidate algorithms* into a *library of weighted averages of these algorithms*.

- Each weighted average is a unique candidate algorithm in this augmented library.
- We can then apply the same cross-validation selector to this augmented set of candidate algorithms, resulting in the super learner.

It might seem that the implementation of such an estimator is problematic, since it requires **minimizing the cross-validated risk over an infinite set of candidate algorithms** (the weighted averages).

The contrary is true.

Super learner is not more computer intensive than discrete super learner.

- If the discrete super learner has been implemented, then all the work has been done!
- Only the relatively trivial calculation of the optimal weight vector needs to be completed.

Super Learner: How it works

Consider that the discrete super learner has already been completed.

Weight vector

We propose a family of weighted combinations of the algorithms in our collection, which we index by the weight vector α . The family of weighted combinations includes only those α -vectors that have a sum equal to one, and where each weight is positive or zero.

We want to determine which combination minimizes the cross-validated risk over the family of weighted combinations.

Super Learner: How it works

Weight vector

$$P_n(Y = 1 | Z) = \text{expit}(\alpha_{a,n}Z_a + \alpha_{b,n}Z_b + \dots + \alpha_{p,n}Z_p)$$

The (cross-validated) probabilities of death (Z) for each algorithm are used as inputs in a working (statistical) model to predict the outcome Y .

Super Learner: How it works

Weight vector

$$P_n(Y = 1 \mid Z) = \text{expit}(\alpha_{a,n}Z_a + \alpha_{b,n}Z_b + \dots + \alpha_{p,n}Z_p)$$

Therefore, we have a working model with multiple coefficients $\alpha = \{\alpha_a, \alpha_b, \dots, \alpha_p\}$ that need to be estimated, one for each of the algorithms.

Super Learner: How it works

Weight vector

$$P_n(Y = 1 \mid Z) = \text{expit}(\alpha_{a,n}Z_a + \alpha_{b,n}Z_b + \dots + \alpha_{p,n}Z_p)$$

Selecting the weights that minimize the cross-validated risk is a simple minimization problem, formulated as a regression of the outcomes Y on the predicted values of the algorithms (Z) according to the user supplied parametric family of weighted combinations.

Super Learner: How it works

Weight vector

$$P_n(Y = 1 \mid Z) = \text{expit}(\alpha_{a,n}Z_a + \alpha_{b,n}Z_b + \dots + \alpha_{p,n}Z_p)$$

The weighted combination with the smallest cross-validated risk is the “best” estimator according to our criteria: minimizing the estimated expected squared error loss function.

Super Learner: How it works

The super learner improves asymptotically on the discrete super learner by **working with a larger library.**

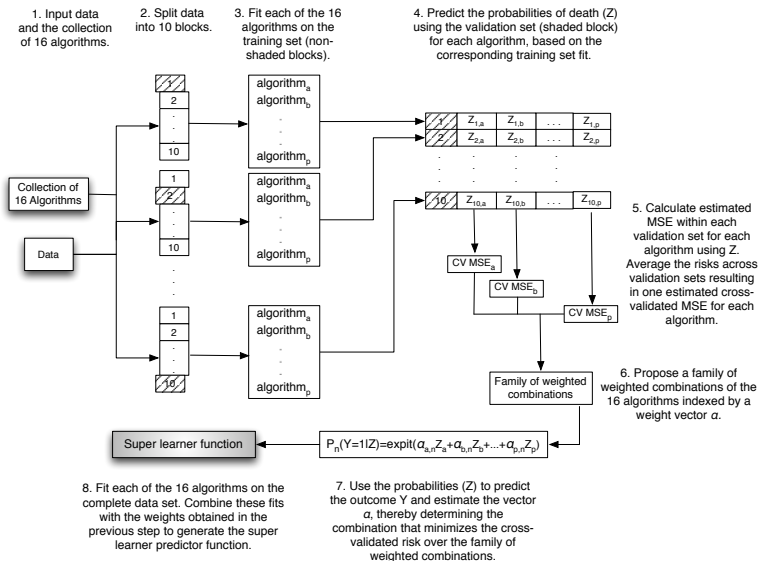
Super Learner: How it works

Consequently, super learner performs asymptotically as well as the best choice among the family of weighted combinations of estimators.

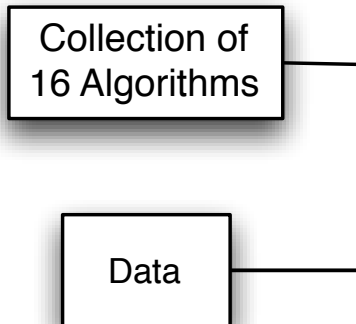
Thus, *by adding more competitors, we only improve the performance of the super learner*. The asymptotic equivalence remains true if the number of algorithms in the library grows very quickly with sample size.

Even when the collection of algorithms contains a correctly specified parametric statistical model, the super learner will approximate the truth as fast as the parametric statistical model, although it will be more variable.

Super Learner



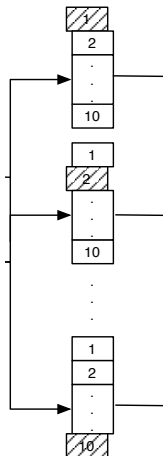
Super Learner



Inputs: collection of algorithms and the data.

Super Learner

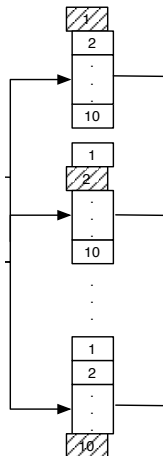
2. Split data
into 10 blocks.



The entire data set (learning set, size n) was divided into 10 groups of size $\sim n/10$. These groups were mutually exclusive and exhaustive sets.

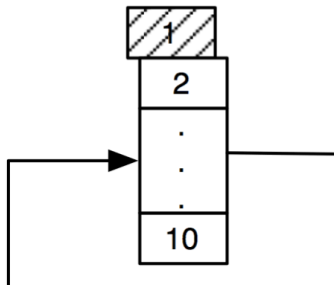
Super Learner

2. Split data
into 10 blocks.



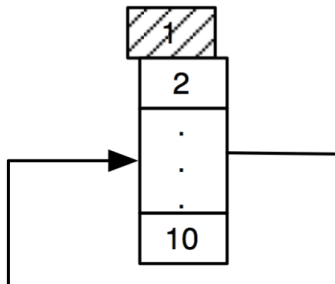
We will rotate fitting the data on the 9 training sets and evaluating the fit on the 1 validation set in the 10 folds.

Super Learner



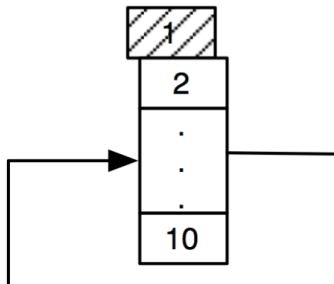
Let's focus on understanding the procedure for just 1 fold and 1 algorithm.

Super Learner



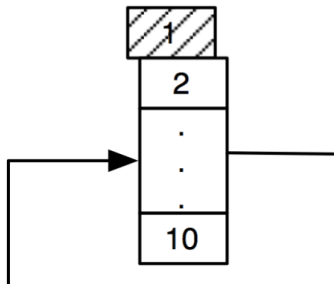
The observations in group 1 are set aside, and the first regression (algorithm_a) is fit on the remaining nine groups (called the training set).

Super Learner



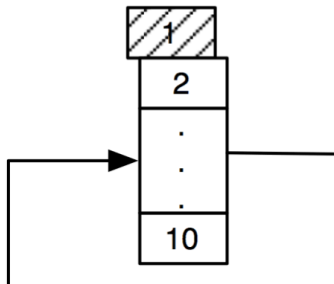
Then, we took the observations in group 1 (called the validation set) and obtained predicted probabilities of death for the $\sim n/10$ observations using the regression fit on the training set.

Super Learner



It is important to note that the observations in group 1 *were not included in the fitting process* and will only be used to evaluate the performance of the predictor that was obtained on the training sample.

Super Learner



We have succeeded in obtaining predicted probabilities for approximately 10% of our data, where the prediction function used to obtain these predicted probabilities was fit based on the remaining 90% of data.

Super Learner

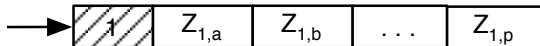
At this stage, we calculated the estimated risk/MSE:

$$\text{MSE}_{\text{validation set 1, algorithm}_a} = \frac{\sum_{i=1}^{(n/10)} (Y_i - Z_{1,a,i})^2}{(n/10)},$$

within the validation set using their predicted probabilities ($Z_{1,a}$).

Super Learner

4. Predict the probabilities of death (Z) using the validation set (shaded block) for each algorithm, based on the corresponding training set fit.

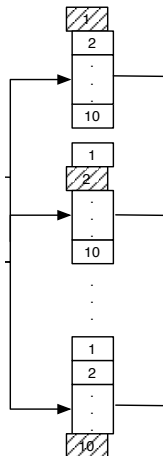


This procedure was performed for all of the algorithms in the collection of algorithms, so that we had, at the end of the first fold, predicted probabilities for each of the algorithms.

We also had an estimate of MSE within the validation set (group 1) for each of the regressions, calculated using their corresponding predicted probabilities.

Super Learner

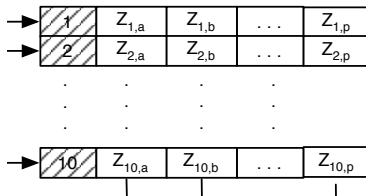
2. Split data
into 10 blocks.



This cycle was repeated nine more times, such that each group had the opportunity to take on the role of the validation set and obtain predicted probabilities for each algorithm fit on the corresponding training set.

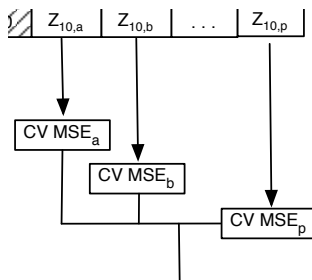
Super Learner

4. Predict the probabilities of death (Z) using the validation set (shaded block) for each algorithm, based on the corresponding training set fit.



Thus, we had predicted probabilities of death for all n subjects for each algorithm, and also estimated MSE within each validation set for each algorithm.

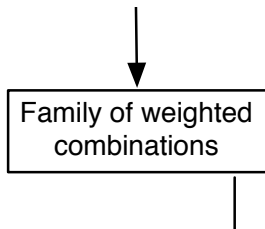
Super Learner



5. Calculate estimated MSE within each validation set for each algorithm using Z . Average the risks across validation sets resulting in one estimated cross-validated MSE for each algorithm.

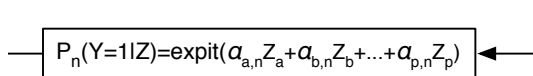
We had 10 estimated MSEs for each of the algorithms, and these MSEs were averaged across validation sets resulting in one estimated cross-validated MSE for each algorithm.

Super Learner



6. Propose a family of weighted combinations of the 16 algorithms indexed by a weight vector α .

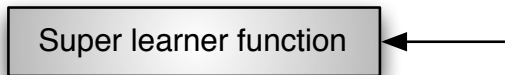
Super Learner


$$P_n(Y=1|Z)=\text{expit}(\alpha_{a,n}Z_a+\alpha_{b,n}Z_b+\dots+\alpha_{p,n}Z_p)$$

7. Use the probabilities (Z) to predict the outcome Y and estimate the vector α , thereby determining the combination that minimizes the cross-validated risk over the family of weighted combinations.

The selected weighted combination is a new estimator we can now use to input data (e.g., our complete mortality data set) to estimate predicted probabilities.

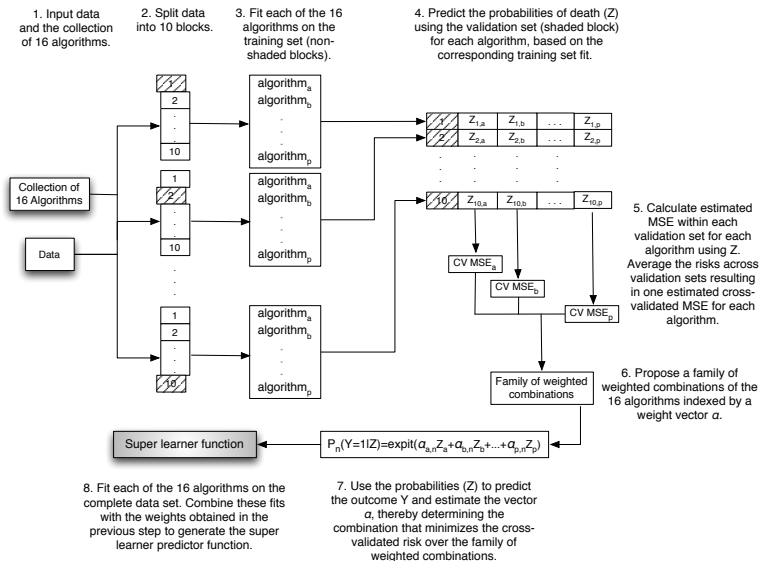
Super Learner



8. Fit each of the 16 algorithms on the complete data set. Combine these fits with the weights obtained in the previous step to generate the super learner predictor function.

Thus, we fit each of the algorithms on our complete data (learning set). Combining these algorithm fits with our new estimator generates the super learner prediction function. This prediction function is the weighted combination of the candidate algorithms applied to the whole data set.

Super Learner



Finite sample performance

To study the super learner in real data examples, we collected a number of publicly available data sets.

- sample sizes ranged from 200 to 654 observations
- number of covariates ranged from 3 to 18
- all 13 data sets have a continuous outcome and no missing values

Finite sample performance

Table 3.3 Description of data sets, where n is the sample size and p is the number of covariates. All examples have a continuous outcome.

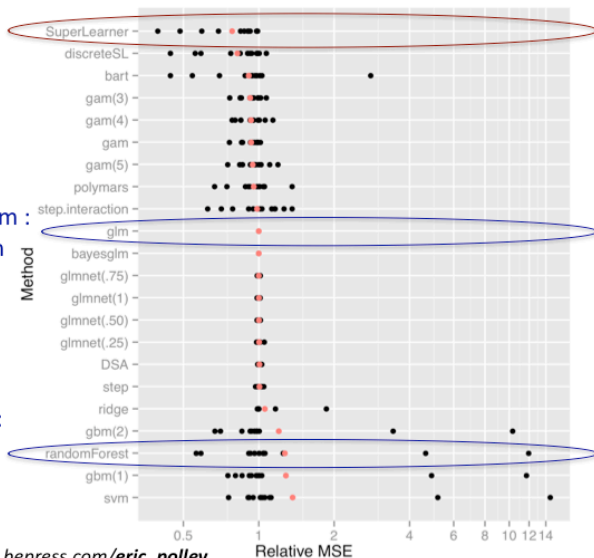
Name	n	p	Source
ais	202	10	Cook and Weisberg (1994)
diamond	308	17	Chu (2001)
cps78	550	18	Berndt (1991)
cps85	534	17	Berndt (1991)
cpu	209	6	Kibler et al. (1989)
FEV	654	4	Rosner (1999)
Pima	392	7	Newman et al. (1998)
laheart	200	10	Afifi and Azen (1979)
mussels	201	3	Cook (1998)
enroll	258	6	Liu and Stengos (1999)
fat	252	14	Penrose et al. (1985)
diabetes	366	15	Harrell (2001)
house	506	13	Newman et al. (1998)

Finite sample performance

Super Learner
Best weighted
combination of
algorithms for a
given prediction
problem

Example algorithm :
Linear Main Term
Regression

Example algorithm:
Random Forest



Technical Report: works.bepress.com/eric_polley

Finite sample performance

The super learner **outperforms the discrete super learner**, and **both outperform any individual algorithm**.

Among the individual algorithms, the Bayesian additive regression trees performs the best, but over-fits one of the data sets with a relative mean squared error of almost 3.0.

Screening

In high dimensional data, it is often beneficial to screen the variables before running prediction algorithms.

Screening algorithms can be coupled with prediction algorithms to create new algorithms in the library.

Screening

For example:

- We may consider an algorithm using all features and an algorithm on the subset of only clinical variables. These two algorithms are considered unique algorithms.
- Another screening algorithm involves testing the pairwise correlations of each variable with the outcome and ranking the variables by the corresponding p-value.
- An additional screening algorithm involves running the glmnet algorithm and selecting the variables with non-zero coefficients.

The free lunch

- There is no point in painstakingly trying to decide what estimators to enter in the collection; **instead add them all.**
- The theory supports this approach and finite sample simulations and data analyses only confirm that **it is very hard to over-fit the super learner by augmenting the collection**, but benefits are obtained.
- Indeed, for large data sets, we simply do not have enough algorithms available to build the desired collection that would fully utilize the power of the super learning.

The Library in Super Learning: The Richer the Better

- The key is a vast library of machine learning algorithms to build your estimator
- Currently 40+ R packages for machine learning/prediction
- If we combine dimension-reduction algorithms with these prediction algorithms, we quickly generate a large library