

Super Learning

Eric Polley

Joint work with Mark van der Laan and Alan Hubbard
e-mail: ecpolley@berkeley.edu

PH 246C



- 1 Cross-Validation
 - Definitions
 - Oracle Inequalities
- 2 Super Learning
 - Definitions
 - Simulations and Data examples



We have the usual setting with:

- Data observed: $O_i = (X_i, Y_i) \sim P_0, i = 1, \dots, n$
- Parameter of interest, $\psi_0(X)$, defined as minimizer of a loss function, $L(O, \psi)$, over a parameter space Ψ
- $\psi_0 = \arg \min_{\psi \in \Psi} E_0 L(O, \psi)$
- for example, the regression problem:
 - $\psi_0(X) = E_0(Y|X)$
 - $L(O, \psi) = (Y - \psi(X))^2$



We have the usual setting with:

- Data observed: $O_i = (X_i, Y_i) \sim P_0, i = 1, \dots, n$
- Parameter of interest, $\psi_0(X)$, defined as minimizer of a loss function, $L(O, \psi)$, over a parameter space Ψ
- $\psi_0 = \arg \min_{\psi \in \Psi} E_0 L(O, \psi)$
- for example, the regression problem:
 - $\psi_0(X) = E_0(Y|X)$
 - $L(O, \psi) = (Y - \psi(X))^2$



We have the usual setting with:

- Data observed: $O_i = (X_i, Y_i) \sim P_0, i = 1, \dots, n$
- Parameter of interest, $\psi_0(X)$, defined as minimizer of a loss function, $L(O, \psi)$, over a parameter space Ψ
- $\psi_0 = \arg \min_{\psi \in \Psi} E_0 L(O, \psi)$
- for example, the regression problem:
 - $\psi_0(X) = E_0(Y|X)$
 - $L(O, \psi) = (Y - \psi(X))^2$



We have the usual setting with:

- Data observed: $O_i = (X_i, Y_i) \sim P_0, i = 1, \dots, n$
- Parameter of interest, $\psi_0(X)$, defined as minimizer of a loss function, $L(O, \psi)$, over a parameter space Ψ
- $\psi_0 = \arg \min_{\psi \in \Psi} E_0 L(O, \psi) \equiv \arg \min_{\psi \in \Psi} \int L(O, \psi) dP_0$
- for example, the regression problem:
 - $\psi_0(X) = E_0(Y|X)$
 - $L(O, \psi) = (Y - \psi(X))^2$



We have the usual setting with:

- Data observed: $O_i = (X_i, Y_i) \sim P_0, i = 1, \dots, n$
- Parameter of interest, $\psi_0(X)$, defined as minimizer of a loss function, $L(O, \psi)$, over a parameter space Ψ
- $\psi_0 = \arg \min_{\psi \in \Psi} E_0 L(O, \psi) \equiv \arg \min_{\psi \in \Psi} \int L(O, \psi) dP_0$
- for example, the regression problem:
 - $\psi_0(X) = E_0(Y|X)$
 - $L(O, \psi) = (Y - \psi(X))^2$



Cross-Validation

- We denote the entire observation set as the **learning** data set,
- split the data into training and validation sets.
- Since the Super Learner is built on V-fold Cross-Validation I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



Cross-Validation

- We denote the entire observation set as the learning data set,
- split the data into **training** and **validation** sets.
- Since the Super Learner is built on V-fold Cross-Validation I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



Cross-Validation

- We denote the entire observation set as the learning data set,
- split the data into training and validation sets.
- Since the Super Learner is built on **V-fold Cross-Validation** I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



Cross-Validation

- We denote the entire observation set as the learning data set,
- split the data into training and validation sets.
- Since the Super Learner is built on **V-fold Cross-Validation** I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



Cross-Validation

- We denote the entire observation set as the learning data set,
- split the data into training and validation sets.
- Since the Super Learner is built on **V-fold Cross-Validation** I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



Cross-Validation

- We denote the entire observation set as the learning data set,
- split the data into training and validation sets.
- Since the Super Learner is built on **V-fold Cross-Validation** I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



Cross-Validation

- We denote the entire observation set as the learning data set,
- split the data into training and validation sets.
- Since the Super Learner is built on **V-fold Cross-Validation** I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



Cross-Validation

- We denote the entire observation set as the learning data set,
- split the data into training and validation sets.
- Since the Super Learner is built on **V-fold Cross-Validation** I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



- We denote the entire observation set as the learning data set,
- split the data into training and validation sets.
- Since the Super Learner is built on **V-fold Cross-Validation** I will only discuss V-fold (although most results work for other forms of cross validation, like monte carlo or bootstrap)
- Let $\nu \in \{1, \dots, V\}$ index the data splits such that:
 - $T(\nu) \subset \{1, \dots, n\}$ and $V(\nu) \subset \{1, \dots, n\}$
 - $T(\nu) \cup V(\nu) = \{1, \dots, n\}$
 - $T(\nu) \cap V(\nu) = \emptyset$
 - $\bigcup_{\nu=1}^V V(\nu) = \{1, \dots, n\}$
 - $V(\nu_1) \cap V(\nu_2) = \emptyset$, for $\nu_1 \neq \nu_2$



V-Fold Cross Validation

- For the data splits, define B_n^ν as the binary split n-vector for the ν -fold :

$$B_n^\nu(i) = \begin{cases} 0 & \text{if } i \in T(\nu) \\ 1 & \text{if } i \in V(\nu) \end{cases}$$

- Let $n_V^\nu = \sum_i B_n^\nu(i)$ be the size of the ν -fold Validation set, and
- $n_T^\nu = \sum_i (1 - B_n^\nu(i))$ be the size of the ν -fold Training set



V-Fold Cross Validation

- For the data splits, define B_n^ν as the binary split n-vector for the ν -fold :

$$B_n^\nu(i) = \begin{cases} 0 & \text{if } i \in T(\nu) \\ 1 & \text{if } i \in V(\nu) \end{cases}$$

- Let $n_V^\nu = \sum_i B_n^\nu(i)$ be the size of the ν -fold Validation set, and
- $n_T^\nu = \sum_i (1 - B_n^\nu(i))$ be the size of the ν -fold Training set



V-Fold Cross Validation

- For the data splits, define B_n^ν as the binary split n-vector for the ν -fold :

$$B_n^\nu(i) = \begin{cases} 0 & \text{if } i \in T(\nu) \\ 1 & \text{if } i \in V(\nu) \end{cases}$$

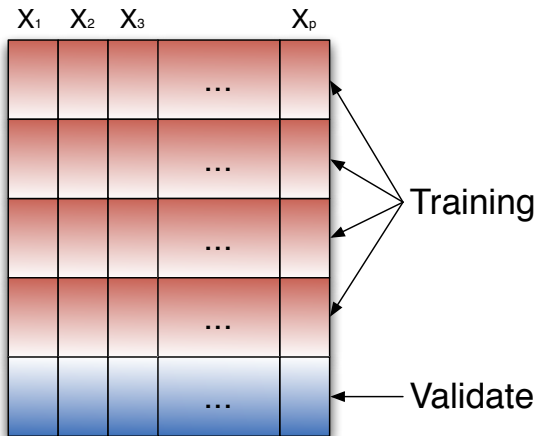
- Let $n_V^\nu = \sum_i B_n^\nu(i)$ be the size of the ν -fold Validation set, and
- $n_T^\nu = \sum_i (1 - B_n^\nu(i))$ be the size of the ν -fold Training set



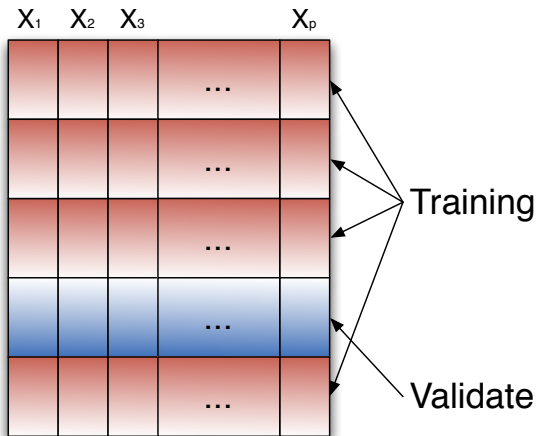
Data



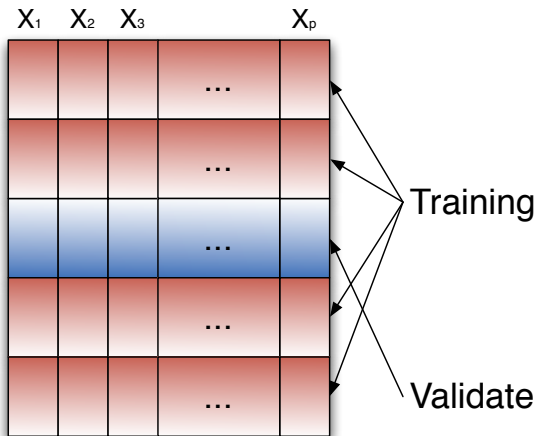
Cross-Validation



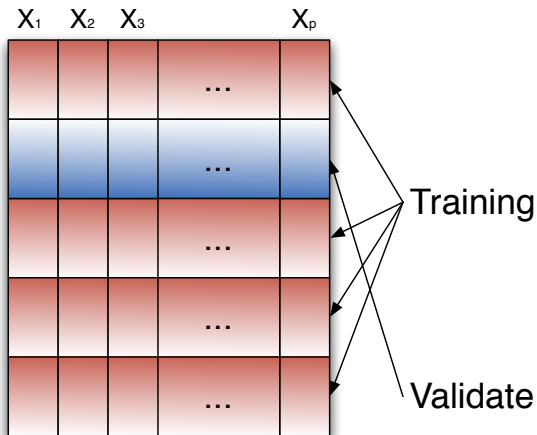
Cross-Validation



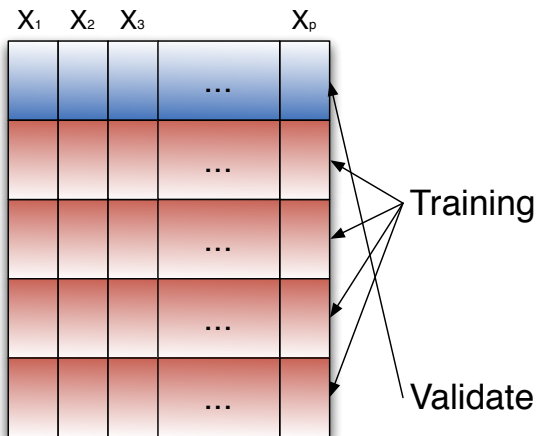
Cross-Validation



Cross-Validation



Cross-Validation



V-Fold Cross Validation

- Let P_n be the empirical distribution for the learning data, which places probability $1/n$ on each O_i , $i = 1, \dots, n$.
- Then, $P_{n,T(\nu)}$ and $P_{n,V(\nu)}$ be the empirical distributions for the training and validation data splits
- An estimate of the parameter of interest can then be seen with the mapping $\hat{\Psi} : \mathcal{M}_n \rightarrow \Psi$
- $\psi_n = \hat{\Psi}(P_n)$ is the estimate based on the entire learning sample, and
- $\psi_{n,\nu} = \hat{\Psi}(P_{n,T(\nu)})$ is the estimate based on the ν -fold training sample



V-Fold Cross Validation

- Let P_n be the empirical distribution for the learning data, which places probability $1/n$ on each O_i , $i = 1, \dots, n$.
- Then, $P_{n,T(\nu)}$ and $P_{n,V(\nu)}$ be the empirical distributions for the training and validation data splits
- An estimate of the parameter of interest can then be seen with the mapping $\hat{\Psi} : \mathcal{M}_n \rightarrow \Psi$
- $\psi_n = \hat{\Psi}(P_n)$ is the estimate based on the entire learning sample, and
- $\psi_{n,\nu} = \hat{\Psi}(P_{n,T(\nu)})$ is the estimate based on the ν -fold training sample



V-Fold Cross Validation

- Let P_n be the empirical distribution for the learning data, which places probability $1/n$ on each O_i , $i = 1, \dots, n$.
- Then, $P_{n,T(\nu)}$ and $P_{n,V(\nu)}$ be the empirical distributions for the training and validation data splits
- An estimate of the parameter of interest can then be seen with the mapping $\hat{\Psi} : \mathcal{M}_n \rightarrow \Psi$
- $\psi_n = \hat{\Psi}(P_n)$ is the estimate based on the entire learning sample, and
- $\psi_{n,\nu} = \hat{\Psi}(P_{n,T(\nu)})$ is the estimate based on the ν -fold training sample



V-Fold Cross Validation

- Let P_n be the empirical distribution for the learning data, which places probability $1/n$ on each O_i , $i = 1, \dots, n$.
- Then, $P_{n,T(\nu)}$ and $P_{n,V(\nu)}$ be the empirical distributions for the training and validation data splits
- An estimate of the parameter of interest can then be seen with the mapping $\hat{\Psi} : \mathcal{M}_n \rightarrow \Psi$
- $\psi_n = \hat{\Psi}(P_n)$ is the estimate based on the entire learning sample, and
- $\psi_{n,\nu} = \hat{\Psi}(P_{n,T(\nu)})$ is the estimate based on the ν -fold training sample



V-Fold Cross Validation

- Let P_n be the empirical distribution for the learning data, which places probability $1/n$ on each O_i , $i = 1, \dots, n$.
- Then, $P_{n,T(\nu)}$ and $P_{n,V(\nu)}$ be the empirical distributions for the training and validation data splits
- An estimate of the parameter of interest can then be seen with the mapping $\hat{\Psi} : \mathcal{M}_n \rightarrow \Psi$
- $\psi_n = \hat{\Psi}(P_n)$ is the estimate based on the entire learning sample, and
- $\psi_{n,\nu} = \hat{\Psi}(P_{n,T(\nu)})$ is the estimate based on the ν -fold training sample



- The Risk of an estimate ψ_n is then:

$$R(\psi_n, P) = \int L(O, \hat{\Psi}(P_n)) dP,$$

which depends on the true data generating distribution, P .

- Plug-in estimate: use the empirical distribution, P_n , in place of P :

$$R(\psi_n, P_n) = \int L(O, \hat{\Psi}(P_n)) dP_n \equiv \frac{1}{n} \sum L(O_i, \psi_n),$$

but concerns about over-fitting with large Ψ .



- The Risk of an estimate ψ_n is then:

$$R(\psi_n, P) = \int L(O, \hat{\Psi}(P_n)) dP,$$

which depends on the true data generating distribution, P .

- Plug-in estimate: use the empirical distribution, P_n , in place of P :

$$R(\psi_n, P_n) = \int L(O, \hat{\Psi}(P_n)) dP_n \equiv \frac{1}{n} \sum L(O_i, \psi_n),$$

but concerns about over-fitting with large Ψ .



- Use the V-fold CV to estimate risk, where parameter is estimated on training set, and risk is estimated on corresponding validation set:

$$E_{B_n} R(\psi_{n,\nu}, P_{n,V(\nu)}) = E_{B_n} \int L(O, \hat{\Psi}(P_{n,T(\nu)})) dP_{n,V(\nu)}$$



V-fold Cross Validation

- Also concerns with searching over entire parameter space to estimate ψ_0
- might propose a collection of candidate estimators which search over different subspaces of Ψ
- In the regression setting, possible candidates: ridge regression, lasso, random forests, D/S/A, etc.
- consider a collection of K_n candidate learners (estimators), then we have $\psi_{n,k}$, $k = 1, \dots, K_n$ estimates.



V-fold Cross Validation

- Also concerns with searching over entire parameter space to estimate ψ_0
- might propose a collection of candidate estimators which search over different subspaces of Ψ
- In the regression setting, possible candidates: ridge regression, lasso, random forests, D/S/A, etc.
- consider a collection of K_n candidate learners (estimators), then we have $\psi_{n,k}$, $k = 1, \dots, K_n$ estimates.



V-fold Cross Validation

- Also concerns with searching over entire parameter space to estimate ψ_0
- might propose a collection of candidate estimators which search over different subspaces of Ψ
- In the regression setting, possible candidates: ridge regression, lasso, random forests, D/S/A, etc.
- consider a collection of K_n candidate learners (estimators), then we have $\psi_{n,k}$, $k = 1, \dots, K_n$ estimates.



- Also concerns with searching over entire parameter space to estimate ψ_0
- might propose a collection of candidate estimators which search over different subspaces of Ψ
- In the regression setting, possible candidates: ridge regression, lasso, random forests, D/S/A, etc.
- consider a collection of K_n candidate learners (estimators), then we have $\psi_{n,k}$, $k = 1, \dots, K_n$ estimates.



- The original Super Learner (Sinisi, et al. SAGMB 2007) proposes the following minimum cross-validated risk selector:

$$\hat{k}_{n,B_n} = \hat{K}(P_n) \equiv \arg \min_{k \in \{1, \dots, K_n\}} E_{B_n} \int L(O, \hat{\psi}_k(P_{n,T(\nu)})) dP_{n,V(\nu)},$$

where, $\hat{\psi}_k$ is the k^{th} estimator.

- then the V-fold CV selected estimate is $\psi_{n, \hat{k}_{n,B_n}}$



- The original Super Learner (Sinisi, et al. SAGMB 2007) proposes the following minimum cross-validated risk selector:

$$\hat{k}_{n,B_n} = \hat{K}(P_n) \equiv \arg \min_{k \in \{1, \dots, K_n\}} E_{B_n} \int L(O, \hat{\psi}_k(P_{n,T(\nu)})) dP_{n,V(\nu)},$$

where, $\hat{\psi}_k$ is the k^{th} estimator.

- then the V-fold CV selected estimate is $\psi_{n,\hat{k}_{n,B_n}}$



- Let $\tilde{k}_{B_n, n}$ be the oracle selector based on the V -fold training sets:

$$\tilde{k}_{B_n, n} = \tilde{K}_{B_n}(P_n) \equiv \arg \min_k E_{B_n} R(\hat{\Psi}_k(P_{n, T(\nu)}), P)$$

- compare the oracle to:

$$\hat{k}_{n, B_n} = \hat{K}(P_n) \equiv \arg \min_{k \in \{1, \dots, K_n\}} E_{B_n} R(\hat{\Psi}_k(P_{n, T(\nu)}), P_{n, V(\nu)})$$



- Let $\tilde{k}_{B_n, n}$ be the oracle selector based on the V -fold training sets:

$$\tilde{k}_{B_n, n} = \tilde{K}_{B_n}(P_n) \equiv \arg \min_k E_{B_n} R(\hat{\Psi}_k(P_{n, T(\nu)}), P)$$

- compare the oracle to:

$$\hat{k}_{n, B_n} = \hat{K}(P_n) \equiv \arg \min_{k \in \{1, \dots, K_n\}} E_{B_n} R(\hat{\Psi}_k(P_{n, T(\nu)}), P_{n, V(\nu)})$$



- Let $\tilde{k}_{B_n, n}$ be the oracle selector based on the V -fold training sets:

$$\tilde{k}_{B_n, n} = \tilde{K}_{B_n}(P_n) \equiv \arg \min_k E_{B_n} R(\hat{\Psi}_k(P_{n, T(\nu)}), P)$$

- compare the oracle to:

$$\hat{k}_{n, B_n} = \hat{K}(P_n) \equiv \arg \min_{k \in \{1, \dots, K_n\}} E_{B_n} R(\hat{\Psi}_k(P_{n, T(\nu)}), P_{n, V(\nu)})$$



- Let $d_0(\psi, \psi_0) = E_{P_0}\{L(O, \psi) - L(O, \psi_0)\}$, be the risk difference between the candidate estimate ψ and the true parameter value ψ_0 .
- Suppose $\Pr\left\{(\hat{\Psi}_k(P_n) \in \Psi) : \forall k\right\} = 1$.
- Assumptions:
 - A1: $L(O, \psi)$ must be uniformly bounded
 - A2: variance of the ψ_0 -centered loss function $(L(O, \psi) - L(O, \psi_0))$ can be bounded by its expectation uniformly in ψ
- then...



- Let $d_0(\psi, \psi_0) = E_{P_0}\{L(O, \psi) - L(O, \psi_0)\}$, be the risk difference between the candidate estimate ψ and the true parameter value ψ_0 .
- Suppose $\Pr\{(\hat{\Psi}_k(P_n) \in \Psi) : \forall k\} = 1$.
- Assumptions:
 - A1: $L(O, \psi)$ must be uniformly bounded
 - A2: variance of the ψ_0 -centered loss function ($L(O, \psi) - L(O, \psi_0)$) can be bounded by its expectation uniformly in ψ
- then...



- Let $d_0(\psi, \psi_0) = E_{P_0} \{L(O, \psi) - L(O, \psi_0)\}$, be the risk difference between the candidate estimate ψ and the true parameter value ψ_0 .
- Suppose $\Pr \left\{ (\hat{\Psi}_k(P_n) \in \Psi) : \forall k \right\} = 1$.
- Assumptions:
 - A1: $L(O, \psi)$ must be uniformly bounded
 - A2: variance of the ψ_0 -centered loss function $(L(O, \psi) - L(O, \psi_0))$ can be bounded by its expectation uniformly in ψ
- then...



- Let $d_0(\psi, \psi_0) = E_{P_0} \{L(O, \psi) - L(O, \psi_0)\}$, be the risk difference between the candidate estimate ψ and the true parameter value ψ_0 .
- Suppose $\Pr \left\{ (\hat{\Psi}_k(P_n) \in \Psi) : \forall k \right\} = 1$.
- Assumptions:
 - A1: $L(O, \psi)$ must be uniformly bounded
 - A2: variance of the ψ_0 -centered loss function $(L(O, \psi) - L(O, \psi_0))$ can be bounded by its expectation uniformly in ψ
- then...



- for any $\lambda > 0$:

$$Ed_0(\hat{\Psi}_{\hat{K}(P_n)}(P_{n,T(\nu)}), \psi_0) \leq (1 + 2\lambda)Ed_0(\hat{\Psi}_{\tilde{K}(P_n)}(P_{n,T(\nu)}), \psi_0) + 2C(\lambda)\frac{1 + \log(K(n))}{np},$$

where p is the proportion of the observations in the validation sample, and $C(\lambda)$ is a constant defined in van der Laan et al. (2006).



Oracle Inequalities

- basically, the super learner performs as well (in terms of expected risk difference) as the oracle selector, up to a typically second order term.
- Thus, as long as the number of candidate learners considered ($K(n)$) is polynomial in sample size, the super learner is the optimal learner
- If, as is typical, none of the candidate learners (nor, as a result, the oracle selector) converge at a parametric rate, the super learner performs asymptotically as well (in the risk difference sense) as the oracle selector, which chooses the best of the candidate learners
- If one of the candidate learners searches within a parametric model and that parametric model contains the truth, and thus achieves a parametric rate of convergence, then the super learner achieves the almost parametric rate of convergence $\log n/n$.



Oracle Inequalities

- basically, the super learner performs as well (in terms of expected risk difference) as the oracle selector, up to a typically second order term.
- Thus, as long as the number of candidate learners considered ($K(n)$) is polynomial in sample size, the super learner is the optimal learner
- If, as is typical, none of the candidate learners (nor, as a result, the oracle selector) converge at a parametric rate, the super learner performs asymptotically as well (in the risk difference sense) as the oracle selector, which chooses the best of the candidate learners
- If one of the candidate learners searches within a parametric model and that parametric model contains the truth, and thus achieves a parametric rate of convergence, then the super learner achieves the almost parametric rate of convergence $\log n/n$.



Oracle Inequalities

- basically, the super learner performs as well (in terms of expected risk difference) as the oracle selector, up to a typically second order term.
- Thus, as long as the number of candidate learners considered ($K(n)$) is polynomial in sample size, the super learner is the optimal learner
- If, as is typical, none of the candidate learners (nor, as a result, the oracle selector) converge at a parametric rate, the super learner performs asymptotically as well (in the risk difference sense) as the oracle selector, which chooses the best of the candidate learners
- If one of the candidate learners searches within a parametric model and that parametric model contains the truth, and thus achieves a parametric rate of convergence, then the super learner achieves the almost parametric rate of convergence $\log n/n$.



Oracle Inequalities

- basically, the super learner performs as well (in terms of expected risk difference) as the oracle selector, up to a typically second order term.
- Thus, as long as the number of candidate learners considered ($K(n)$) is polynomial in sample size, the super learner is the optimal learner
- If, as is typical, none of the candidate learners (nor, as a result, the oracle selector) converge at a parametric rate, the super learner performs asymptotically as well (in the risk difference sense) as the oracle selector, which chooses the best of the candidate learners
- If one of the candidate learners searches within a parametric model and that parametric model contains the truth, and thus achieves a parametric rate of convergence, then the super learner achieves the almost parametric rate of convergence $\log n/n$.

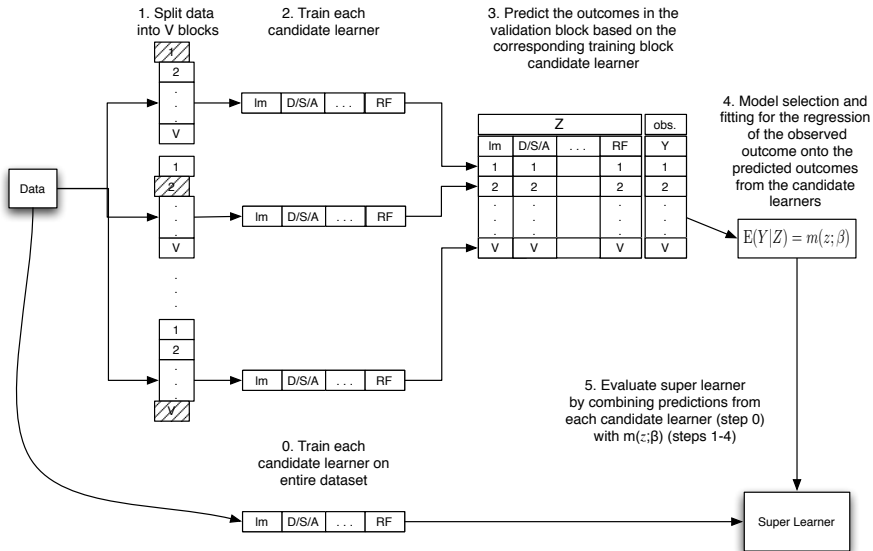


- with the cross validation theorem above as the motivation, we looked to extend the candidate learner selector to include weighted averages of the candidate learners.
- easier to show a diagram to explain algorithm:



- with the cross validation theorem above as the motivation, we looked to extend the candidate learner selector to include weighted averages of the candidate learners.
- easier to show a diagram to explain algorithm:





- A similar oracle results holds for this new proposed super learner: For each $\delta > 0$ we have that there exists a $C(\delta) < \infty$ such that

$$\frac{1}{V} \sum_{v=1}^V E d(\hat{\Psi}_{\alpha_n}(P_{nT(v)}), \psi_0) \leq (1 + \delta) E \min_{\alpha \in \mathcal{A}_n} \frac{1}{V} \sum_{v=1}^V d(\hat{\Psi}_{\alpha}(P_{nT(v)}), \psi_0) + C(\delta) \frac{V \log n}{n}.$$



Thus, if

$$\frac{E \min_{\alpha \in \mathcal{A}_n} \frac{1}{V} \sum_{v=1}^V d(\hat{\Psi}_{\alpha}(P_{nT(v)}), \psi_0)}{\frac{\log n}{n}} \rightarrow 0 \text{ as } n \rightarrow \infty, \quad (1)$$

then it follows that the estimator $\hat{\Psi}_{\alpha_n}$ is asymptotically equivalent with the oracle estimator $\hat{\Psi}_{\tilde{\alpha}_n}$ when applied to samples of size $(1 - 1/V)n$:

$$\frac{\frac{1}{V} \sum_{v=1}^V E d(\hat{\Psi}_{\alpha_n}(P_{nT(v)}), \psi_0)}{E \min_{\alpha \in \mathcal{A}_n} \frac{1}{V} \sum_{v=1}^V d(\hat{\Psi}_{\alpha}(P_{nT(v)}), \psi_0)} \rightarrow 1 \text{ as } n \rightarrow \infty.$$



If (1) does not hold, then it follows that $\hat{\Psi}_{\alpha_n}$ achieves the $(\log n)/n$ rate:

$$\frac{1}{V} \sum_{v=1}^V Ed(\hat{\Psi}_{\alpha_n}(P_{nT(v)}), \psi_0) = O\left(\frac{\log n}{n}\right).$$



- The theorem implies that the proposed super learner will perform asymptotically as well (up to a constant) as the best estimator among the family of estimators $\{\hat{\Psi}_\alpha : \alpha \in \mathcal{A}\}$ when applied to samples of size $n(1 - 1/V)$, or achieve a rate $\log(n)/n$.



- simulated from:

$$Y_i = 2w_1w_{10} + 4w_2w_7 + 3w_4w_5 - 5w_6w_{10} + 3w_8w_9 + w_1w_2w_4 - 2w_7(1 - w_6)w_2w_9 - 4(1 - w_{10})w_1(1 - w_4) + \varepsilon,$$

where $W_j \sim \text{Binom}(p = 0.4)$ and $\varepsilon \sim N(0, 1)$.

- simulated a learning sample of 500 observations.



- simulated from:

$$Y_i = 2w_1 w_{10} + 4w_2 w_7 + 3w_4 w_5 - 5w_6 w_{10} + 3w_8 w_9 + w_1 w_2 w_4 - 2w_7(1 - w_6)w_2 w_9 - 4(1 - w_{10})w_1(1 - w_4) + \varepsilon,$$

where $W_j \sim \text{Binom}(p = 0.4)$ and $\varepsilon \sim N(0, 1)$.

- simulated a learning sample of 500 observations.



Simulation Results

Candidate learners considered:

Method	R Package	Authors
Least Angle Regression	lars	Hastie and Efron
Logic Regression	LogicReg	Kooperberg and Ruczinski
D/S/A	DSA	Neugebauer and Bullard
Regression Trees	rpart	Therneau and Atkinson
Ridge Regression	MASS	Venables and Ripley
Random Forests	randomForest	Liaw and Wiener
Adapt Regression Splines	pol spline	Kooperberg

Table: R Packages for Candidate Learners. R is available at <http://www.r-project.org>



- candidates for first simulation: least squares, LARS, D/S/A, Logic, random forests.
- We applied the super learner with 10-fold cross-validation on the learning set. Applying the prediction to all 10 folds of the learning set gives us the predicted values $Z_i \equiv (\hat{\Psi}_{j\nu(i)}(W_i) : j = 1, \dots, 5)$
- then used least squares to fit the linear model $E(Y|Z) = \alpha + \beta Z$
- each of the candidate learners was fit on the entire learning set to obtain $\hat{\Psi}_j(P_n)(W)$, which gives the super learner $\hat{\Psi}(P_n)(W) = \alpha_n + \beta_n(\hat{\Psi}_j(P_n)(W) : j = 1, \dots, 5)$



Simulation results

- candidates for first simulation: least squares, LARS, D/S/A, Logic, random forests.
- We applied the super learner with 10-fold cross-validation on the learning set. Applying the prediction to all 10 folds of the learning set gives us the predicted values $Z_i \equiv (\hat{\Psi}_{j\nu(i)}(W_i) : j = 1, \dots, 5)$
- then used least squares to fit the linear model $E(Y|Z) = \alpha + \beta Z$
- each of the candidate learners was fit on the entire learning set to obtain $\hat{\Psi}_j(P_n)(W)$, which gives the super learner $\hat{\Psi}(P_n)(W) = \alpha_n + \beta_n(\hat{\Psi}_j(P_n)(W) : j = 1, \dots, 5)$



Simulation results

- candidates for first simulation: least squares, LARS, D/S/A, Logic, random forests.
- We applied the super learner with 10-fold cross-validation on the learning set. Applying the prediction to all 10 folds of the learning set gives us the predicted values $Z_i \equiv (\hat{\Psi}_{j\nu(i)}(W_i) : j = 1, \dots, 5)$
- then used least squares to fit the linear model $E(Y|Z) = \alpha + \beta Z$
- each of the candidate learners was fit on the entire learning set to obtain $\hat{\Psi}_j(P_n)(W)$, which gives the super learner $\hat{\Psi}(P_n)(W) = \alpha_n + \beta_n(\hat{\Psi}_j(P_n)(W) : j = 1, \dots, 5)$



Simulation results

- candidates for first simulation: least squares, LARS, D/S/A, Logic, random forests.
- We applied the super learner with 10-fold cross-validation on the learning set. Applying the prediction to all 10 folds of the learning set gives us the predicted values $Z_i \equiv (\hat{\Psi}_{j\nu(i)}(W_i) : j = 1, \dots, 5)$
- then used least squares to fit the linear model $E(Y|Z) = \alpha + \beta Z$
- each of the candidate learners was fit on the entire learning set to obtain $\hat{\Psi}_j(P_n)(W)$, which gives the super learner $\hat{\Psi}(P_n)(W) = \alpha_n + \beta_n(\hat{\Psi}_j(P_n)(W) : j = 1, \dots, 5)$



- results

method	RMSPE	β_n
Least Squares	1.00	0.038
LARS	1.15	-0.171
D/S/A	0.22	0.535
Logic	0.32	0.274
Random Forest	0.42	0.398
Super Learner	0.20	

Table: Simulation Example 1: Estimates of the relative mean squared prediction error (compared to least squares) based on a learning sample of 500 observations and the evaluation sample $M=10,000$. The estimates for β in the super learner are also reported in the right column ($\alpha_n = -0.018$).



Simulation results

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	0.99	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

Table: Simulation Example 3: Estimates of the relative mean squared prediction error (compared to least squares) based on the validation sample. The 3 new studies are combined with the second simulation example and the relative mean squared prediction error is reported in the overall column.

