UNIVERSITY OF COPENHAGEN
FACULTY OF SCIENCE

A Bachelor of Science thesis

Super Learners

and their oracle properties

Jinyang Liu

Supervised by Prof. Thomas Gerds
Co-supervised by Prof. Niels Richard Hansen
Department of Mathematical Sciences
University of Copenhagen, Denmark

Submitted: May 19, 2023

Contents

1 Introduction

In the context of regression, where $O = (Y, X)$ is an observation or outcome-covariate pair for $Y \in \mathbb{R}$ and $X \in \mathbb{R}^d$. A natural goal is to estimate a regression function θ such that the L^2-risk or mean squared error $E(Y - \theta(X))^2$ is minimized. It turns out that the conditional mean – called the regression function or regression $x \mapsto E(Y \mid X = x)$ minimizes the squared error, but consistent estimation of the regression requires a statistical model $\mathcal{P}$ for the data-generating distribution $P \in \mathcal{P}$ for which $O \sim P$. We typically make certain assumptions about the statistical model, $\mathcal{P}$, in which we believe P resides. For instance, we might assume that $\mathcal{P}$ is a parametric family of distributions, for example a curved exponential family (Lauritzen 2022). In doing so we are able to identify through maximum likelihood techniques, the parameters of the distribution P and compute the regression from the estimated parameters.

However, if we are dealing with complex data, there is a risk of misspecifying the model by identifying it as an exponential family. Our assumptions may be wrong. In such situations, it is tempting to utilize non-parametric and data-driven regression methods, such as tree-based algorithms like XGBoost or random forests. Machine learning methods seek to estimate regression function directly, in contrast to parametric statistics where the parameters of the underlying model are estimated first, then a regression function derived as some analytical expression of these parameters and the covariates. The assumptions of these machine learning methods regarding the data-generating distribution are not explicitly specified, but it does not pose a problem for us in achieving our goal. Indeed, it is not necessary for us to identify P completely if P can be factored into the conditional distribution $P_{Y|X}$ and the distribution over our covariates P_X, here our goal of estimating the regression function is akin to estimating $P_{Y|X}$.

Given an abundance of different ways we can tackle the problem of estimating the regression, we would like to be able to compare the different methods select the best one. In a practical scenario we might have set of different learning algorithms from which we can choose from. Cross-validation, for example, can help in determining the risk of each algorithm by splitting the data into training and validation sets, each algorithm would then be fitted on the training set and an empirical risk for each fitted model is then calculated by evaluating the model on the validation set. We would run cross-validation using a predefined splitting mechanism for each learning algorithm that we have. A popular choice in machine learning and statistics is K-fold cross-validation. K-fold cross-validation divides the data into K disjoint and exhaustive sets referred to as validation sets. For every $k = 1, \ldots, K$ the learning algorithm is trained using all data excluding the k'th validation set. Subsequently, the risk of the fitted model is computed by applying it on the validation set that was held out from training. We obtain an estimate of the true risk of the model by repeating this procedure K times and then averaging the risks. The model with the lowest risk is then selected as our candidate estimate of the regression function.

This is the idea behind the cross-validation selector (Laan and Dudoit 2003). The discrete super learner (Van der Laan, Polley, and Hubbard 2007) is simply the learner (fitted model) that is created by applying the result of the cross-validation selector to our data. The cross-validation selector, given a library of learning algorithms and our data, applies cross-validation to each algorithm in the library.

Another method, called the ensemble super learner (Van der Laan, Polley, and Hubbard 2007) seeks to combine the learning algorithms into a single learner by taking a linear combination of algorithms in the library. One way to fit the ensemble super learner is by taking a weighted average of the each learner, where the weights are chosen to minimize

the risk of the ensemble. A prediction made by an ensemble super learner will then be a weighted sum of the predictions made by the individual learners.
As we will demonstrate in this thesis, given a library of learning algorithms, the super learner is effectively the best estimate of the data-generating regression that we can obtain. It is the best in the sense that it can not perform worse than the best learner created from our library in terms of risk.
To formalize the notion of the best learner in the library, we will define the oracle selector. The oracle selector knows P – hence it is called the oracle – and selects the learner with the lowest risk relative to P. In this thesis we demonstrate that the cross-validation selector is asymptotically equivalent to the oracle selector.
Our setup and notation is similar to Vaart, Dudoit, and Laan 2006 and Laan and Dudoit 2003. The focus in the thesis will to demonstrate the effectiveness of the super learner applied to binary regression. More specifically, we focus on the case where we regress on a binary outcome $Y \in \{0, 1\}$. The conditional expectation of Y given X exactly becomes the conditional probability $P(Y = 1 \mid X = x)$. The simulation results in section 5 shows that the super learner is able to achieve the minimum risk as the number of training samples increases. The simulation is conducted by first defining the data-generating distribution explicitly, from which we can sample from using standard sampling methods that are available in R. We evaluate a collection of learning algorithms, including logistic regression and XGBoost. From this library, we create the super learner and compare its performance against the individual algorithms.
The choice to focus on binary regression stems from its significance in various fields. For instance, in biomedicine, researchers might want to predict patient mortality upon administering a specific drug. The survival indicator for the patient is a binary outcome, and the regression $P(Y = 1 \mid X = x)$ could represent the probability of the patient's survival.

2 Background

Our setup and notation is similar to Vaart, Dudoit, and Laan 2006 and Laan and Dudoit 2003: Let a statistical model $\mathcal{P}$ be given on the measurable space $(\mathcal{O}, \mathcal{A})$ where $\mathcal{O} = \{0, 1\} \times \mathcal{X}$ is our sample space for some $\mathcal{X} \subseteq \mathbb{R}^d$. We will consider the parameter set $\Theta = \{\theta \mid \theta : \mathcal{X} \to [0, 1]\}$, which represents the set of *regression functions* that map from our covariates to the probability interval. We define the quadratic loss and the corresponding risk that we wish to minimize

Definition 1 (Quadratic loss)**.** The quadratic loss or L^2-loss, $L : \mathcal{O} \times \Theta \to [0, \infty)$, for an observation $o \in \mathcal{O}$ and a regression function $\theta \in \Theta$ is defined as

$$L(o, \theta) = L((y, x), \theta) = (y - \theta(x))^2.$$

A natural aim would be to find the optimal parameter value $\theta^* \in \Theta$ that minimizes the expected L^2-loss, or conditional risk $R : \theta \to \mathbb{R}$ given by

$$R(\theta, P) := \int L(o, \theta) dP(o). \tag{1}$$

Theorem 2 shows that the minimum risk is achieved by the conditional probability $x \mapsto P(Y = 1 \mid X = x)$.

Theorem 2. *Let $(\mathcal{O}, \mathcal{A}, P)$ be a probability space for some probability measure $P \in \mathcal{P}$. Let Θ be the set of regression functions of the form $\theta : \mathcal{X} \to [0, 1]$. Let the loss function be the L^2-loss $L(o, \theta) = (y - \theta(x))^2$, then for the optimum θ^* defined as*

$$\theta^* := \arg\min_{\theta \in \Theta} R(\theta, P) = \arg\min_{\theta \in \Theta} \int L(o, \theta) dP(o),$$

it holds for an observation $O = (Y, X) \sim P$ that

$$\theta^*(x) = E(Y \mid X = x)$$

Proof. Proof emulated from (Györfi et al. 2002)[ch. 1].
Let $\theta \in \Theta$ be arbitrary and $m(x) = E(Y \mid X = x)$, we have

$$\begin{aligned} E|\theta(X) - Y|^2 &= E|\theta(X) - m(X) + m(X) - Y|^2 \\ &= E|\theta(X) - m(X)|^2 + E|m(X) - Y|^2 + 2E[(\theta(X) - m(X))(m(X) - Y)]. \end{aligned}$$

We see that the last term is zero, see that by using the tower rule we have

$$\begin{aligned} E[(\theta(X) - m(X))(m(X) - Y)] &= E[E[(\theta(X) - m(X))(m(X) - Y) \mid X]] \\ &= E[(\theta(X) - m(X))E((m(X) - Y) \mid X)] \\ &= E[(\theta(X) - m(X))(m(X) - E(Y \mid X))] \\ &= E[(\theta(X) - m(X))(m(X) - m(X))] \\ &= 0. \end{aligned}$$

We conclude that

$$\int L(o, \theta) dP(o) = E|\theta(X) - m(X)|^2 + E|m(X) - Y|^2.$$

The first term after the equality is always positive and is 0 only when $\theta = m$, this proves that m minimizes the expression above. □

It follows immediately that if Y is binary, then $E(Y \mid X = x) = P(Y = 1 \mid X = x)$. As we do not have access to the data-generating distribution P, our goal is to estimate θ^*, this means to *learn* the true regression function from our data. We will therefore introduce the terminology *learning algorithm* and *learner* in the context of learning from our data.

2.1 Learning algorithms and learners

We will denote $O_1, \ldots, O_n \in \mathcal{O}$ and $D_n = (O_1, \ldots, O_n)$ as our observations and data respectively.

Definition 3 (Learning algorithm)**.** A learning algorithm is a measurable map $\psi : \mathcal{O}^n \to \Theta$ for $n \in \mathbb{N}$.

We will throughout assume that the learning algorithm is well defined for each $n \in \mathbb{N}$, and that permuting the observations have no effect on the outcome, i.e., the algorithm is symmetric in the observations.

Definition 4 (Learner)**.** Given a learning algorithm ψ. A learner is a stochastic variable in Θ representing the outcome of applying the learning algorithm to our data $\theta = \psi(D_n)$.

Formally $\psi(D_n)$ is a stochastic variable since D_n is stochastic. In practice, we would have observed $O_3 = o_1, \ldots, O_n = o_n$, and can subsequently apply our learning algorithm on $d_n = (o_1, \ldots, o_n)$, which is a particular instance of a dataset. We will refer to the quantity, $\psi(d_n)$, as a *fitted learner*.

Definition 5 (Fitted learner)**.** Let ψ be a learning algorithm. Upon observing the data $d_n = (o_1, \ldots, o_n)$, a fitted learner is a regression function in Θ, obtained by applying the learning algorithm to the observed data $\theta = \psi(d_n)$.

Example 1 (Parametric and nonparametric learning algorithms)*.* An example of a parametric learning algorithm is logistic regression. In logistic regression we assume that the conditional probability, $P(Y = 1 \mid X = x)$, can be expressed as $\theta(x) = \text{expit}(\beta x)$ for some $\beta \in \mathbb{R}^d$. The parameter β can be estimated via maximum likelihood.
Nonparametric learning algorithms such as gradient boosting, for example XGBoost, can also be used to estimate the regression function. The gradient boosting algorithm, XGBoost, has a number of hyperparameters that can be tuned. These include, number of boosted trees, depth of each tree, learning rates, etc., but most importantly the internal loss objective which could for example be log-loss or mean squared error. XGBoost aims to iteratively refine the fitted learner by approximating the data $x \mapsto f_m(x)$ at each step m. It does so by introducing a new tree $h_m(x)$, which is trained on the error of $f_m(x)$, such that $f_{m+1}(x) = f_m(x) + h_m(x)$. The internal loss of the updated learner, f_{m+1}, evaluated on the training data, is lower than that of the previous learner due to the inclusion of the new tree (Chen and Guestrin 2016).
The parameters of the resulting fit are not directly interpretable. Despite this, XGBoost has demonstrated its ability to model very complex datasets (Chen and Guestrin 2016).

We will denote the empirical measure obtained from the data D_n as

$$P_n(A) = \frac{1}{n} \sum_{i=1}^{n} \delta_{O_i}(A) \qquad \text{for } A \in \mathcal{A}.$$

When the observations are independent and identically distributed, then there is a one-to-one correspondence between the empirical measures obtained from n observations and D_n. We can, therefore, write $\psi(P_n)$ as an alternative representation of the learner $\psi(D_n)$, by adjusting the notation slightly without introducing ambiguity. The motivation for using this notation will become clearer in the subsequent section, where we introduce the discrete super learner.

3 The Discrete Super Learner

3.1 Library of learners

We would now like to consider the scenario where we have a set of learning algorithms, $\psi_1, \ldots, \psi_k$. From these algorithms, we can define the *library of learning algorithms*

$$\Psi = \{\psi_q \mid 1 \leq q \leq k\}$$

of size k. Our goal is to find $\tilde{\psi} \in \Psi$ such that $\tilde{\psi}(P_n)$ that achieves the lowest risk among our learners

$$\tilde{\psi} = \arg\min_{\psi \in \Psi} R(\psi(P_n), P)$$

We can only provide an estimate for $\tilde{\psi}$ since knowing it would require knowledge of the data-generating distribution. We will denote the estimate as $\hat{\psi}$.

3.2 Cross-validation methodology

To provide $\hat{\psi}$ we have proceed via cross validation. In cross validation, we randomly split our data into a *training set* and a *test set*. Let the random binary vector – from now on referred to as the *split variable* or just *split* – $S^n = (S_1^n, \ldots, S_n^n) \in \{0,1\}^n$ be independent of $O_1, \ldots, O_n$ such that $S_i^n = 0$ indicates that O_i should be in the training set and $S_i^n = 1$ indicates that O_i belongs to the test set. The split S^n depends on n as it is formally a tuple in $\{0,1\}^n$, here the superscript n is used to indicate that. We can define the empirical distributions over these two subsets, P^0_{n,S^n} and P^1_{n,S^n} as

$$P^0_{n,S^n} = \frac{1}{n_0} \sum_{i:S_i^n=0} \delta_{O_i}$$
$$P^1_{n,S^n} = \frac{1}{n_1} \sum_{i:S_i^n=1} \delta_{O_i},$$

where $n_1 = \sum_{i=1}^n S_i^n, n_0 = 1 - n_1$ identifies the number of observations in the test and training set respectively.

Example 2 (Random splits). For n observations we have 2^n ways of choosing which observations should be in the training set and in the test set. It might not be desirable to define the discrete probabilities for S^n over $\{0,1\}^n$ simply as $\frac{1}{2^n}$ for each possible combination of training/test data, since that would also include the combination where $n_1 = 0$. To ensure that there is always a certain amount of observations in our test set, let $n_1 > 0$ be given, we see that there are $\binom{n}{n_1}$ ways of choosing both the test and training set. We can therefore define the distribution of S^n as

$$P\left(S^n = s^n\right) = \binom{n}{n_1}^{-1} \qquad \text{for each } s^n \in \{0,1\}^n \text{ where } \sum_i s_i^n = n_1,$$

this procedure is also known as Monte Carlo cross-validation.

Example 3 (Cross-validation loss). The test size n_1 is formally a random variable as it depends on S^n. Hold one out cross validation

$$E_{S^n} R(\theta(P^0_{n,S^n}, P^1_{n,S^n})) = E_{S^n} \frac{1}{n_1} \sum_{i:S^n_i=1} L(o_i, \theta) = E_{S^n} L(o_{S_t}, \theta) = \frac{1}{n} \sum_{i=1}^{n} L(o_i, \theta)$$

K-fold cross validation

$$E_S R(\theta(P^0_{n,S}, P^1_{n,S})) = E_S \frac{1}{n_1} \sum_{i:S_i=1} L(o_i, \theta) = \frac{1}{K} \sum_{k=1}^{K} \sum_{i:s_{k,i}=1} L(o_i, \theta)$$

3.3 Risks and selectors

We now provide the formal definitions for the expected loss associated with our learners. Recall that the expected L^2-loss (1) was the integral of the loss with respect to data-generating distribution P. Consider a learner or regression function θ. Upon observing our data D_n, we can define the empirical risk as the integral of the loss function with respect to P_n, as follows

$$R(\theta, P_n) = \int L(o, \theta) dP_n(o)$$

For example, if S^n is a split for our data D_n, the risk of our learner on the cross-validation test data can be expressed as

$$R(\theta, P^1_{n,S^n}) = \int L(o, \theta) dP^1_{n,S^n}(o).$$

The following definitions are analogous to what is stated in section 1 and 2 of (Laan and Dudoit 2003).

Definition 6 (Expected loss averaged over splits (Vaart, Dudoit, and Laan 2006))**.** Given the data D_n, a split-variable S^n and a library Ψ. Let $\psi \in \Psi$ be a learning algorithm. The expected loss for the learner, $\psi(P^0_{n,s})$, created from applying ψ on the training data, averaged over S^n is

$$E_{S^n} R(\psi(P^0_{n,S}), P),$$

where P is the data-generating distribution.

The expectation E_{S^n} is a simple average since S^n is discrete. Therefore, for a given ψ we have

$$E_{S^n} R(\psi(P^0_{n,S^n}), P) = \sum_{s^n \in \{0,1\}^n} R(\psi(P^0_{n,s}), P) \cdot P(S^n = s^n).$$

Definition 7 (Oracle selector (Laan and Dudoit 2003))**.** Given the data D_n, a split variable S^n and a library Ψ, the learning algorithm $\psi \in \Psi$ with the lowest averaged expected loss is the *oracle selected algorithm*

$$\tilde{\psi} := \arg\min_{\psi \in \Psi} E_{S^n} R(\psi(P^0_{n,S^n}), P).$$

As we are never able to know what the oracle chooses, we must proceed via cross-validation to estimate $\tilde{\psi}$. Cross-validation replaces P with P^1_{n,S^n} in the second argument of R.

Definition 8 (Cross-validation expected loss)**.** Given the data D_n, a split-variable S^n and a library Ψ. Let $\psi \in \Psi$ be a learning algorithm. The cross-validation expected loss for the learner, $\psi(P^0_{n,s})$, created from applying ψ on the training data, averaged over S^n is

$$E_{S^n} R(\psi(P^0_{n,S^n}), P^1_{n,S^n}),$$

where P^1_{n,S^n} is the test data as specified by the split-variable.

Definition 9 (Cross-validation selector (Laan and Dudoit 2003))**.** Given the data D_n, a split variable S^n and a library Ψ, the learning algorithm $\psi \in \Psi$ with the lowest cross-validation loss is the *cross-validation selected algorithm*

$$\hat{\psi} := \arg\min_{\psi \in \Psi} E_{S^n} R(\psi(P^0_{n,S^n}), P^1_{n,S^n}).$$

The oracle selector and cross-validation selector are simply procedures for selecting a learning algorithm from the library. Alternatively, one can first define the oracle selector as the index, $\tilde{q}$, of the algorithm in the library with the lowest risk, then the oracle selected algorithm would naturally be $\psi_{\tilde{q}}$.
We will now give the definition of the *discrete super learner*

Definition 10 (Discrete super learner)**.** The *discrete super learner*, $\psi(P_n)$, created from a library Ψ is the cross-validation selected learning algorithm fitted to the entire dataset

$$\mathcal{X} \ni x \mapsto \hat{\psi}(P_n)(x).$$

Formally, the map above is a random map as P_n is stochastic. The discrete super learner is not a specific learner among the learners in the library, but the result of after applying the cross-validation selector to the library.

3.4 Oracle inequalities

We introduce the notation Pf for the integral $\int f dP$ of an integrable function f with respect to P. Additionally, if P_n represents the empirical measure of $O_1, \ldots, O_n$, we denote the empirical process indexed over an appropriate class of functions $\mathcal{F}$ as $G_n f = \sqrt{n}(P_n f - Pf)$. Furthermore, we extend this notation to $G^i_{n,S^n} f = \sqrt{n}(P^i_{n,S^n} - Pf)$ for the empirical processes that correspond to applying the empirical measure over either the training data or test data, $i = 0$ or $i = 1$.
In the following results we assume that a proper loss function $L : \mathcal{O} \times \Theta \to \mathbb{R}$ has been given.

Lemma 11 (Lemma 2.1 in (Vaart, Dudoit, and Laan 2006))**.** *Let G_n be the empirical process of an i.i.d. sample of size n from the distribution P and Ψ a library of learning algorithms. Furthermore, let $\hat{\psi}$ and $\tilde{\psi}$ denote the cross-validation- and oracle selected algorithm from Ψ of size k respectively. For $\delta > 0$ it holds that*

$$\begin{aligned} E_{S^n} \int L(o, \hat{\psi}(P^0_{n,S^n}))dP(o) \leq (1+2\delta) E_{S^n} \int L(o, \tilde{\psi}(P^0_{n,S^n}))dP(o) \\ + E_{S^n} \frac{1}{\sqrt{n_1}} \max_{\psi \in \Psi} \int L(o, \psi(P^0_{n,S^n}))d((1+\delta)G^1_{n,S^n} - \delta\sqrt{n_1}P)(o) \\ + E_{S^n} \frac{1}{\sqrt{n_1}} \max_{\psi \in \Psi} \int -L(o, \psi(P^0_{n,S^n}))d((1+\delta)G^1_{n,S^n} + \delta\sqrt{n_1}P)(o) \end{aligned}$$

Proof. See appendix □

To control the bounds for the expected loss we introduce Bernstein pairs

Definition 12 (Bernstein pair (Vaart, Dudoit, and Laan 2006))**.** Given a measurable function $f : \mathcal{O} \to \mathbb{R}$, the tuple $(M(f), v(f))$ is a Bernstein pair if

$$M(f)^2 P\left(e^{|f|/M(f)} - 1 - \frac{|f|}{|M(f)|}\right) \leq \frac{1}{2} v(f) \tag{2}$$

Proposition 13. *If f is uniformly bounded, then $(\|f\|_\infty, \frac{3}{2}Pf^2)$ is a Bernstein pair.*

Proof. Following proof is due to (Vaart, Dudoit, and Laan 2006)[ch. 8.1].

$$\begin{aligned}
\|f\|_\infty^2 P\left(e^{|f|/\|f\|_\infty} - 1 - \frac{|f|}{\|f\|_\infty}\right) &= \|f\|_\infty^2 \sum_{k\geq 2}^{\infty} P\frac{|f|^k}{\|f\|_\infty^k k!} = Pf^2 \sum_{k\geq 2}^{\infty} P\frac{|f|^{k-2}}{\|f\|_\infty^{k-2} k!} \\
&\leq Pf^2 \sum_{k\geq 2}^{\infty} \frac{\|f\|_\infty^{k-2}}{\|f\|_\infty^{k-2} k!} = Pf^2 \sum_{k\geq 2}^{\infty} \frac{1}{k!} \\
&= Pf^2(e-2) \leq \frac{3}{4} Pf^2 = \frac{1}{2}\left(\frac{3}{2} Pf^2\right).
\end{aligned}$$

In the first inequality we replace the absolute value of f with the uniform norm, which is larger. □

Example 4 (Binary regression). Consider binary regression with quadratic loss. Let $\theta \in \Theta$ be arbitrary, then a Bernstein pair for the function $f(o) = L(o, \theta) = (Y - \theta(X))^2$ can be found by applying proposition 13. By requiring that $\theta(x) \in [0, 1]$, then it is clear that f is bounded between 0 and 1 for all $o \in \mathcal{O}$ since $Y \in \{0, 1\}$.

Lemma 14 (Lemma 2.2 in (Vaart, Dudoit, and Laan 2006))**.** *Let G_n be the empirical process of an i.i.d. sample of size n from the distribution P and assume that $Pf \geq 0$ for every $f \in \mathcal{F}$ in some set of measurable functions $\mathcal{F}$ on $\mathcal{O}$. Then, for any Bernstein pair $(M(f), v(f))$ and for any $\delta > 0$ and $1 \leq p \leq 2$,*

$$E_{D_n} \max_{f\in\mathcal{F}} (G_n - \delta\sqrt{n}P)f \leq \frac{8}{n^{1/p-1/2}} \log(1 + \#\mathcal{F}) \max_{f\in\mathcal{F}} \left[\frac{M(f)}{n^{1-1/p}} + \left(\frac{v(f)}{(\delta Pf)^{2-p}}\right)^{1/p}\right].$$

The same upper bound is valid for $E_{D_n} \max_{f\in\mathcal{F}} (G_n + \delta\sqrt{n}P)(-f)$

The expectation E_{D_n} is taken wrt. the joint probability measure over our observations, here we have that $D_n \sim P_O^n = P_{O_1} \otimes P_{O_2} \otimes \cdots \otimes P_{O_n}$.

Theorem 15 (Theorem 2.3 in (Vaart, Dudoit, and Laan 2006))**.** *Let Ψ be a library of learning algorithms of size k. For $\theta \in \Theta$ let the numbers $(M(\theta), v(\theta))$ be a Bernstein pair for the function $o \mapsto L(o, \theta)$ and assume that $R(\theta, P) \geq 0$ for every $\theta \in \Theta$. Then for $\delta > 0$ and $1 \leq p \leq 2$ it holds that*

$$\begin{aligned}
E_{D_n} E_{S^n} R(\hat{\psi}(P^0_{n,S^n}), P) \leq (1 + 2\delta) E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P) + \\
(1 + \delta) E_{S^n}\left(\frac{16}{n_1^{1/p}} \log(1 + k) \sup_{\theta\in\Theta}\left[\frac{M(\theta)}{n_1^{1-1/p}} + \left(\frac{v(\theta)}{R(\theta, P)^{2-p}}\right)^{1/p}\left(\frac{1+\delta}{\delta}\right)^{2/p-1}\right]\right).
\end{aligned}$$

Where $\hat{\psi}$ and $\tilde{\psi}$ are the cross-validation- and the oracle selected algorithm from Ψ.

In the expectations above, we are taking the expectation wrt. the random split-variable S^n as well as the joint distribution of our observations. In a more verbose manner one would write

$$E_{D_n} E_{S^n} R(\hat{\psi}(P^0_{n,S^n}), P) = \int R(\hat{\psi}(P^0_{n,S^n}), P) d(P^n_S \otimes P^n_O).$$

Proof. We will apply lemma 14 to the second and third terms on the left hand side of the inequality in lemma 11. Let $\mathcal{F} = \{o \mapsto L(o, \psi(P^0_{n,S^n})) \mid \psi \in \Psi\}$, be the collection of functions obtained by applying the loss L to each algorithm in our library Ψ. Note that $\mathcal{F} \subseteq \{o \mapsto L(o, \theta) \mid \theta \in \Theta\}$, and since $R(\theta, P) \geq 0$ for every $\theta \in \Theta$ it follows that $Pf \geq 0$ for every $f \in \mathcal{F}$.
First, we take the expectation wrt. D_n on both sides in lemma 11. For the second term we have

$$\begin{aligned}
&E_{D_n} E_{S^n} \frac{1}{\sqrt{n_1}} \max_{\psi \in \Psi} \int L(o, \psi(P^0_{n,S^n})) d((1+\delta) G^1_{n,S^n} - \delta \sqrt{n_1} P)(o) \\
&= E_{D_n} E_{S^n} \frac{1+\delta}{\sqrt{n_1}} \max_{\psi \in \Psi} \int L(o, \psi(P^0_{n,S^n})) d(G^1_{n,S^n} - \frac{\delta}{1+\delta} \sqrt{n_1} P)(o) \\
&= E_{S^n} \frac{1+\delta}{\sqrt{n_1}} E_{D_n} \max_{\psi \in \Psi} \int L(o, \psi(P^0_{n,S^n})) d(G^1_{n,S^n} - \frac{\delta}{1+\delta} \sqrt{n_1} P)(o).
\end{aligned}$$

Where we use Fubini in the last equality. Recall that $S^n \perp\!\!\!\perp D_n$, so we can always consider n_1 as fixed given D_n, now applying lemma 14 to the expression above with $n = n_1$ yields

$$\begin{aligned}
&E^n_S \frac{1+\delta}{\sqrt{n_1}} E_{D_n} \max_{\psi \in \Psi} \int L(o, \psi(P^0_{n,S^n})) d(G^1_{n,S^n} - \frac{\delta}{1+\delta} \sqrt{n_1} P)(o) \\
&\leq E_{S^n} \frac{1+\delta}{\sqrt{n_1}} \frac{8}{n_1^{1/p-1/2}} \log(1+k) \max_{\psi \in \Psi} \left[\frac{M(\psi(P^0_{n,S^n}))}{n_1^{1-1/p}} + \left(\frac{v(\psi(P^0_{n,S^n}))}{(\frac{\delta}{1+\delta})^{2-p} R(\psi(P^0_{n,S}), P)^{2-p}} \right)^{1/p} \right] \\
&\leq E_{S^n} \frac{1+\delta}{\sqrt{n_1}} \frac{8}{n_1^{1/p-1/2}} \log(1+k) \sup_{\theta \in \Theta} \left[\frac{M(\theta)}{n_1^{1-1/p}} + \left(\frac{v(\theta)}{(\frac{\delta}{1+\delta})^{2-p} R(\theta, P)^{2-p}} \right)^{1/p} \right] \\
&= (1+\delta) E_{S^n} \frac{8}{n_1^{1/p}} \log(1+k) \sup_{\theta \in \Theta} \left[\frac{M(\theta)}{n_1^{1-1/p}} + \left(\frac{v(\theta)}{R(\theta, P)^{2-p}} \right)^{1/p} \left(\frac{1+\delta}{\delta} \right)^{2/p-1} \right]
\end{aligned}$$

Where for the third inequality we take the sup over Θ. We can also bound the third term with the same expression above as lemma 14 is also valid for $-L$. It is now immediate from lemma 11 that

$$\begin{aligned}
E_{D_n} E_{S^n} \int L(o, \hat{\psi}(P^0_{n,S^n})) dP(o) &\leq (1+2\delta) E_{D_n} E_{S^n} \int L(o, \tilde{\psi}(P^0_{n,S^n})) dP(o) \\
&+ (1+\delta) E_{S^n} \frac{8}{n_1^{1/p}} \log(1+k) \sup_{\theta \in \Theta} \left[\frac{M(\theta)}{n_1^{1-1/p}} + \left(\frac{v(\theta)}{R(\theta)^{2-p}} \right)^{1/p} \left(\frac{1+\delta}{\delta} \right)^{2/p-1} \right] \\
&+ (1+\delta) E_{S^n} \frac{8}{n_1^{1/p}} \log(1+k) \sup_{\theta \in \Theta} \left[\frac{M(\theta)}{n_1^{1-1/p}} + \left(\frac{v(\theta)}{R(\theta)^{2-p}} \right)^{1/p} \left(\frac{1+\delta}{\delta} \right)^{2/p-1} \right],
\end{aligned}$$

The second and third terms above are identical, meaning that they can be combined into one term where the numerator in the first fraction is 16 instead of 8. □

Remark: One might notice that S^n still appears in the remainder of the bound above, but the only variable in the remainder that depends on S^n is n_1. It is, therefore, completely possible to drop the expectation wrt. S^n if n_1 is deterministic, for example, if it is a fraction of n.

Corollary 16 (Asymptotic equivalence)**.** *If there exists an $\varepsilon > 0$ such that*

$$E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P) > \varepsilon \qquad \text{for all } n \in \mathbb{N},$$

and if $n_1 \to \infty$ as $n \to \infty$, then the risk of the super learner is asymptotically equivalent to the risk of the oracle selected learner, that is

$$\lim_{n\to\infty} \frac{E_{D_n} E_{S^n} R(\hat{\psi}(P^0_{n,S^n}), P)}{E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P)} = 1.$$

For the sake of simplicity and the previous remark we will consider n_1 as a sequence of numbers that increases as n increases, this will allow us to remove E_{S^n} in the remainder term.

Proof. By choosing $p = 2$ in theorem 15, we obtain

$$\begin{aligned} E_{D_n} E_{S^n} R(\hat{\psi}(P^0_{n,S^n}), P) \leq (1 + 2\delta) E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P) \\ + (1 + \delta) \frac{16}{\sqrt{n_1}} \log(1 + k) \sup_{\theta \in \Theta} \left[\frac{M(\theta)}{\sqrt{n_1}} + \sqrt{v(\theta)} \right]. \end{aligned}$$

In the above inequality the remainder term goes to 0 as n_1 increases. Now by dividing the oracle risk on both sides, we obtain

$$\frac{E_{D_n} E_{S^n} R(\hat{\psi}(P^0_{n,S^n}), P)}{E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P)} \leq 1 + 2\delta + \frac{(1 + \delta) \frac{16}{\sqrt{n_1}} \log(1 + k) \sup_{\theta \in \Theta} \left[\frac{M(\theta)}{\sqrt{n_1}} + \sqrt{v(\theta)} \right]}{E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P)}.$$

As the inequality holds for all $\delta > 0$, we can just set it to 0 (formally one would perhaps choose δ_n decreasing in n). See that by setting $\delta = 0$ the inequality becomes

$$\frac{E_{D_n} E_{S^n} R(\hat{\psi}(P^0_{n,S^n}), P)}{E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P)} \leq 1 + \frac{\frac{16}{\sqrt{n_1}} \log(1 + k) \sup_{\theta \in \Theta} \left[\frac{M(\theta)}{\sqrt{n_1}} + \sqrt{v(\theta)} \right]}{E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P)}.$$

The fraction on the right-hand side will converge to 0 because the numerator converges to 0 and the oracle risk is bounded away from 0. We also note that by the definition of the oracle, we have

$$E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P) \leq E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P),$$

implying that

$$1 \leq \frac{E_{D_n} E_{S^n} R(\hat{\psi}(P^0_{n,S^n}), P)}{E_{D_n} E_{S^n} R(\tilde{\psi}(P^0_{n,S^n}), P)},$$

applying the squeeze theorem thus yields the desired result. □

For the asymptotic equivalence to hold, we must have that $n_1 \to \infty$ as $n \to \infty$. This is a reasonable assumption for cross-validation schemes such as K-fold cross-validation or if $n_1 = np$ for $p \in (0, 1)$ where the test size increases with the amount of available data. Note that the condition is not satisfied for leave-one-out cross-validation since n_1 is equal to 1 for every n.

Example 5 (Regression). In a regression context with quadratic loss, the risk for a learner can never be zero unless the binary outcome is deterministic given x. We may begin by noting that the risk of $\theta^*(x) = E(Y \mid X = x)$ is always positive,

$$\int (Y - E(Y \mid X))^2 dP \geq 0.$$

Since the integrand is positive, the integral is zero if and only if $Y = E(Y \mid X)$ almost surely. That is only the case when Y is deterministic given X, i.e. Y is X-measurable. Assuming, therefore, that Y is not deterministic given X, then we note that by the fact that the quadratic loss is strictly proper, then we have for any $\theta \in \Theta$

$$R(\theta, P) \geq R(\theta^*, P) > 0,$$

thus showing that the risk of any learner is strictly positive.

4 The Ensemble Super Learner

The idea behind the ensemble super learner is to fit a weighted combination of candidate learners, $\psi_1(P_n), \ldots, \psi_k(P_n)$, where the weights are chosen to minimize the cross-validated risk. We will first introduce

4.1 K-fold cross validation

The idea behind cross validation is to split the data into K equally sized folds, where the candidate learners are fitted on $K - 1$ folds and their performance is evaluated on the remaining fold. We index the folds by $s \in \{1, \ldots, K\}$ and let $s(i)$ indicate the fold that observation i belongs to. Furthermore, we will let $P^0_{n,s}$ denote the empirical measure over the observations that are not in fold s, and let $P^1_{n,s}$ denote the empirical measure over the observations that are in the fold s.

4.2 Level 1 data

Let D_n be our data and let Ψ be a library of learning algorithms. The *level 1 covariates* of Ψ applied to D_n as specified by a K-fold cross validation procedure is

$$\mathcal{Z} = \{Z_i = (\psi_1(P^0_{n,s(i)})(X_i), \ldots, \psi_k(P^0_{n,s(i)})(X_i))\}_{i=1}^n \subseteq [0,1]^k,$$

which is the set of possible outcomes by applying the learners on the observed X_i's.

Definition 17 (Level 1 data)**.** The *level 1 data* is given by concatenating our observed Y_i's with the level 1 covariates, i.e.

$$\mathcal{L}_n = \{(Y_i; Z_i) = (Y_i; \psi_1(P^0_{n,s(i)})(X_i), \ldots, \psi_k(P_{n,s(i)})(X_i))\}_{i=1}^n \subseteq \{0,1\} \times \mathcal{Z}$$

The level 1 data is exactly the predictions made by the learners on the observations that they were not trained on. We now define $\mathcal{M}$ as the class of all measurable functions that map from $\mathcal{Z}$ to $[0,1]$ which we will refer to as *meta learners*.

4.3 Meta learners

Definition 18 (Meta learner)**.** The *meta learner* is a function $\phi : \mathcal{Z} \to [0,1]$ in $\mathcal{M}$ that maps the output of the candidate learners to a prediction.

The goal is to estimate the regression of Y given the predictions made by our learners, $\mathcal{Z} \ni z \mapsto E(Y \mid Z = z)$, which is an element in $\mathcal{M}$. We accomplish this by applying a *meta learning algorithm* to the level 1 data.

Definition 19 (Meta learning algorithm)**.** A *meta learning algorithm* is a measurable map that creates a meta learner from our level 1 data $\mathcal{L}_n \mapsto \Phi(\mathcal{L}_n) \in \mathcal{M}$.

A meta learning algorithm seeks to estimate $E(Y \mid Z)$. It can for example be a parametric learning algorithm such as logistic regression.

4.4 Ensemble super learner

Definition 20 (Ensemble super learner (Van der Laan, Polley, and Hubbard 2007))**.** Let a library of learning algorithms Ψ be given and let $\mathcal{L}_n$ be the level 1 data obtained by fitting each learning algorithm according to some K-fold cross validation procedure. Let $\phi = \Phi(\mathcal{L}_n)$ be the outcome of applying a meta learning algorithm to the level 1 data, then the map

$$x \mapsto \phi(\psi_1(P_n)(x), \ldots, \psi_k(P_n)(x))$$

is called the *ensemble super learner* and we will denote it by $\Sigma(P_n)$.

Here the P_n indicates that we fit each learner in the library on the entire data set, then a meta learner, ϕ, is used to combine the predictions made by each learner.

4.5 Oracle inequalities

The meta learning algorithm fits the learner that most accurately predicts Y given Z, a way of formalizing the notion of 'fitting' is that the meta learning algorithm seeks to select the meta learner with the lowest risk among a indexed set of meta learners. Let $\mathcal{A}$ be a finite set, for each $a \in \mathcal{A}$ let $\phi(\cdot \mid a)$ be a meta learner. Intuitively, one might consider $\phi(\cdot \mid a)$ as the learner obtained by instantiating it with the parameter a. Given our level 1 data $\mathcal{L}_n$, define

$$\hat{a} := \arg\min_{a \in \mathcal{A}} \frac{1}{n} \sum_{i=1}^{n} L((Y_i, Z_i), \phi(\cdot \mid a)) = \arg\min_{a \in \mathcal{A}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \phi(Z_i \mid a))^2$$

The meta learning algorithm applied to $\mathcal{L}_n$ returns $\phi(\cdot \mid \hat{a})$, which is the meta learner with the empirical lowest risk on the level 1 data. We will now let the finite parameter set to depend on n, such that the number elements in $\mathcal{A}_n$ grows at most at a polynomial rate in n. It is possible to write

$$\phi_a := \phi(\cdot \mid a),$$
$$\hat{a}_n := \arg\min_{a \in \mathcal{A}_n} \frac{1}{n} \sum_{i=1}^{n} (Y_i - \phi_a(Z_i))^2.$$

We can denote the ensemble super learner that applies the meta learner ϕ_a as

$$\Sigma_a(P_n) := \phi_a(\psi_1(P_n), \ldots, \psi_k(P_n)).$$

Denote the oracle selector of a as

$$\tilde{a}_n := \arg\min_{a \in \mathcal{A}_n} \frac{1}{K} \sum_{s=1}^{K} R(\Sigma_a(P^0_{n,s}), P),$$

where P is the distribution of our level 0 data O.

Theorem 21 (Finite sample bound)**.**

$$\frac{1}{K} \sum_{s=1}^{K} E_{D_n} R(\Sigma_{\hat{a}_n}(P_n), P) \leq (1 + 2\delta) E_{D_n} R(\Sigma_{\tilde{a}_n}(P_n), P) + (1 + \delta) \frac{C \log(1 + K(n))}{\sqrt{n}}$$

5 Simulation results

In the following section we show the results of applying the discrete super learner to a simulated dataset. Our dataset consists of a binary outcome, Y, which depends on two covariates X_1 and X_2. Our setup is as follows

$$\begin{aligned} X_1 &\sim \text{Unif}(0.5, 15), \\ X_2 \mid X_1 = x_1 &\sim \mathcal{N}(3.5 - 0.03x_1, 1), \\ Y \mid X_1 = x_1, X_2 = x_2 &\sim \text{Ber}(\theta_0(x_1, x_2)), \end{aligned}$$

for $\theta_0(x_1, x_2) = \text{expit}(-3.5 - 0.3x_1 + 0.85x_2 + 0.35x_1x_2)$ which is the data-generating regression function. It is in fact possible to visualize the regression function explicitly as a 2-dimensional heat map in the covariates. In figure 1 we have applied the true regression across the grid of (x_1, x_2) covariate pairs in $(0, 15) \times (0, 7)$ where the spacing is 0.5 horizontally and vertically between each pair. In the plot the probabilities are colored from 0 to 1.

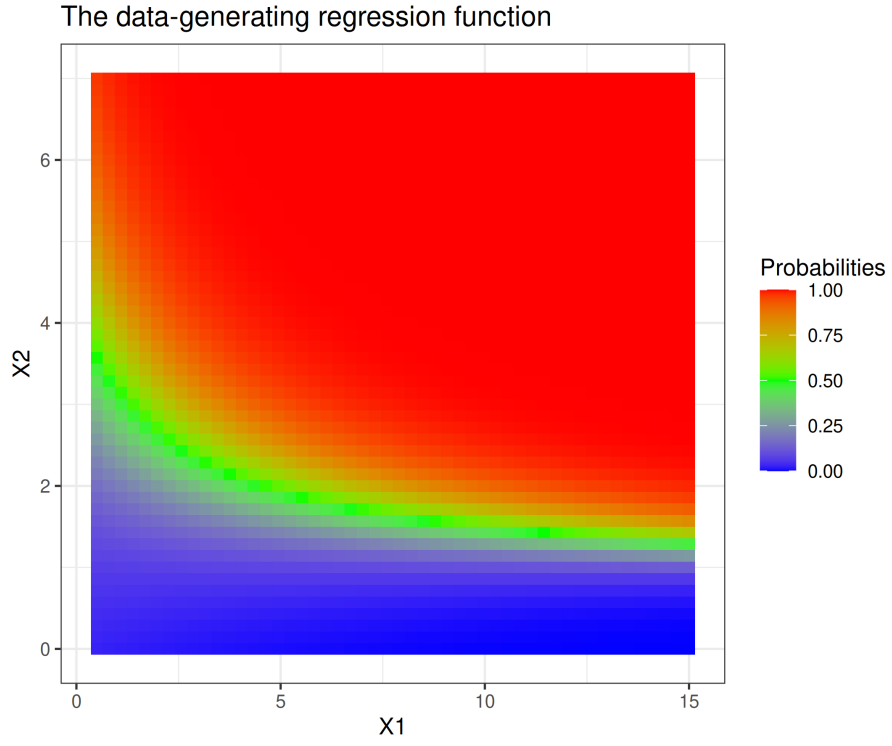

Figure 1: The data-generating regression plotted as a heat map

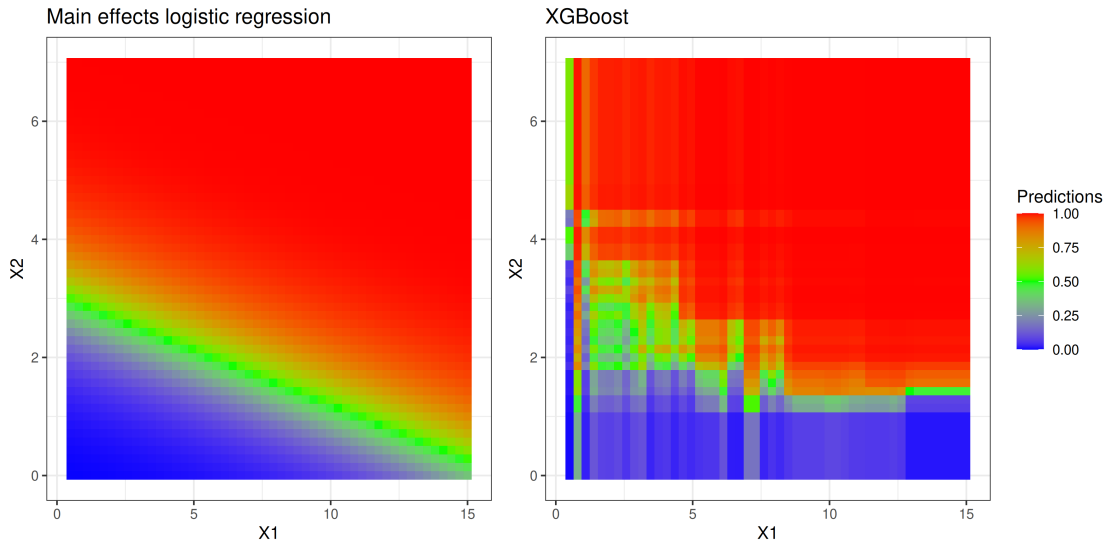

Figure 2: The predictions of the main effects logistic regression and XGBoost plotted as a heat map

The regression is captured by the logistic regression model with interaction terms. We will use the following library of learning algorithms as an illustrative example:

1. Intercept only logistic regression: $E[Y \mid X_1, X_2] = \text{expit}(\beta_0)$
2. Logistic regression with main effects: $E[Y \mid X_1, X_2] = \text{expit}(\beta_0 + \beta_1 X_1 + \beta_2 X_2)$
3. XGBoost with hyperparameters: `max_depth=3, eta=0.3, n_rounds=100, objective='binary:logistic', booster='dart', nthread=5`

As we have have been able to plot the data-generating regression, we can also visualize the predictions of the learning algorithms in the library similarly. In figure 2 we visualize the predictions of the main effects logistic regression and XGBoost fitted using 1000 observations sampled from the distribution. The plot for the intercept only logistic regression is omitted, as its appearance is as one would expect – the plot is simply an orange square. From figure 2 we can observe a clear difference in the predicted probabilities between the logistic regression and the tree-based XGBoost. The main effects logistic regression is a parametric model that assumes that the regression function is a smooth transformation of the linear predictor $X\beta$. XGBoost, in contrast, is made up of decision trees, which explains the patchwork pattern in its prediction plot. For small samples and as we will see in the simulations, XGBoost has a high risk in comparison to the main effects logistic regression despite the fact that it is misspecified. However, XGBoost becomes increasingly better at approximating the true regression when the number of observations becomes large as seen in figure 3. By applying a super learning algorithm to the library, we will see that the cross-validation selector is able to qualitatively assess and select the best learning algorithm to apply given the amount of data at hand. For example the cross-validation selector might select the main effects logistic regression in the beginning with few training samples, but as the predictions of XGBoost become more stable with more samples, the selector is likely to shift its preference towards XGBoost.

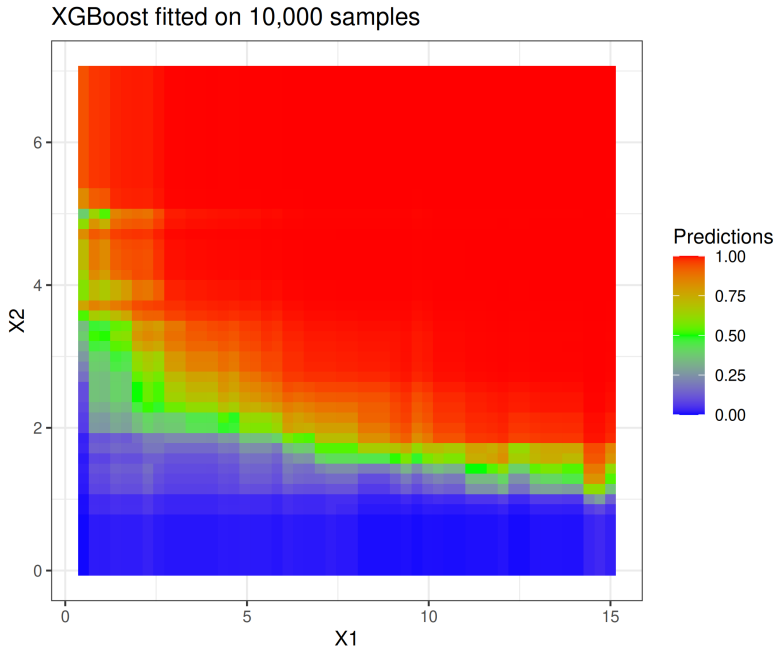

Figure 3: XGBoost becoming better at approximating the true regression as the sample size increases

In our setup, we will consider a discrete super learner that uses 10-fold cross validation and the internal loss function will be the quadratic loss. The performance of the discrete super learner will be compared to each individual learner in the library. We show that

1. As the sample size increases, the discrete super learner achieves the minimum risk
2. For a single new observation, the prediction of the discrete super learner on the outcome has the lowest variance

5.1 Simulation results for the discrete super learner

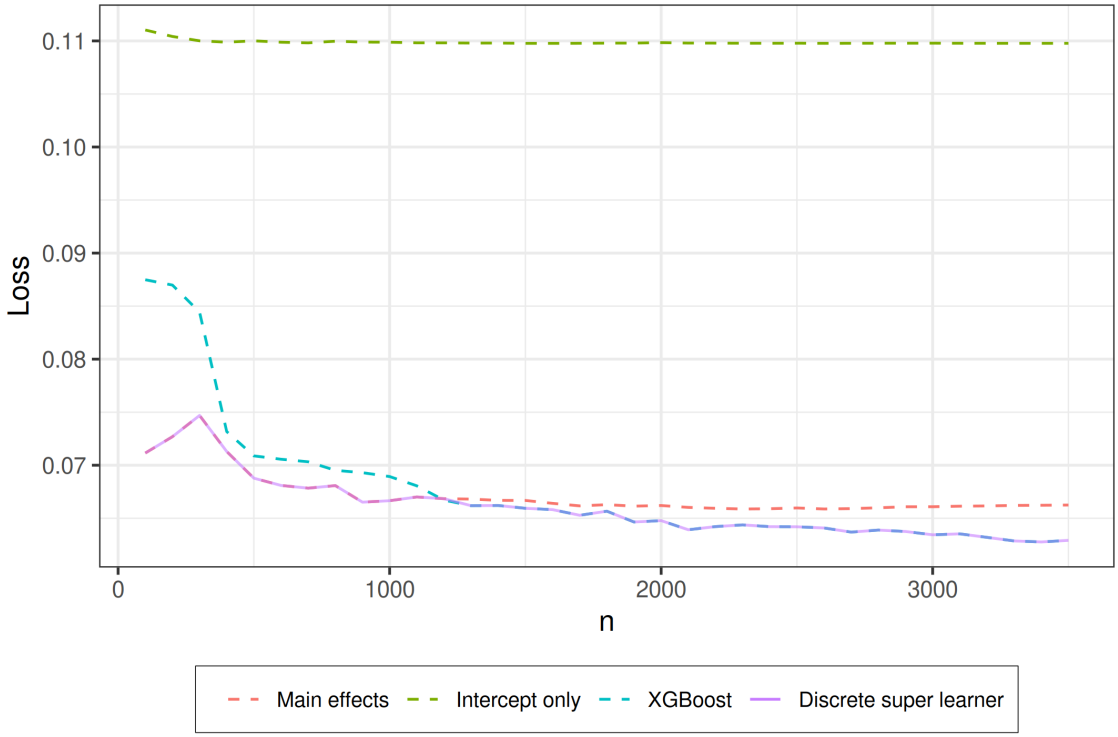

Figure 4: The risk of the discrete super learner compared to other learners. $N = 3500$

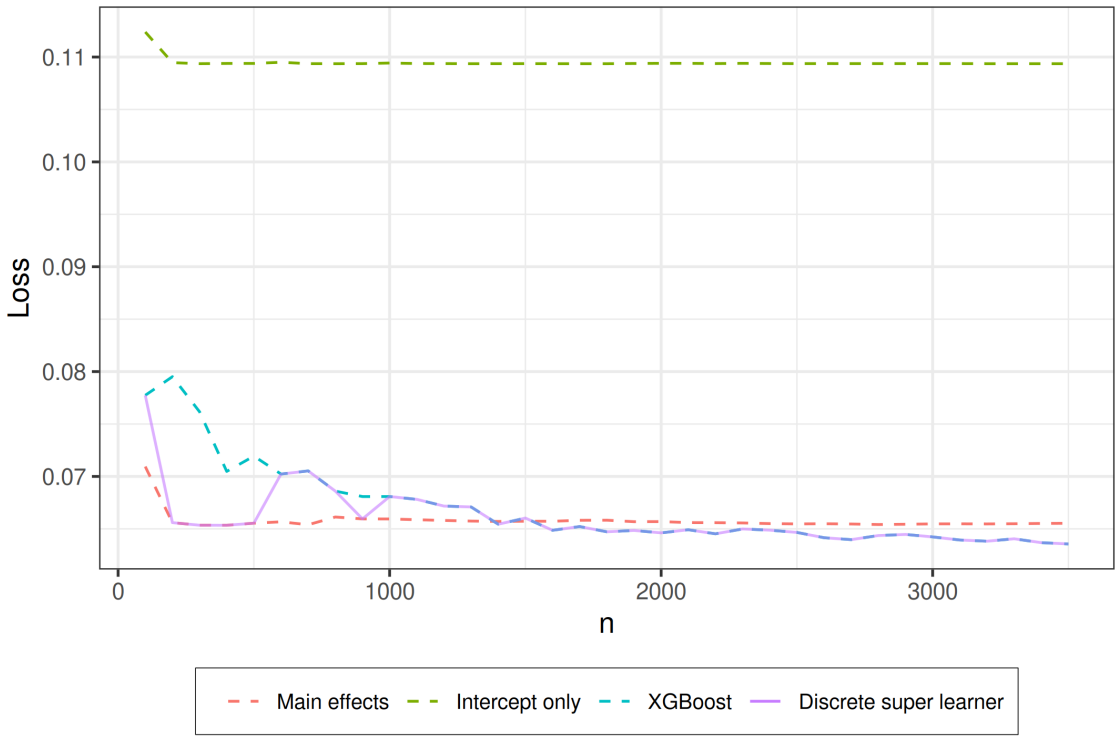

Figure 5: The risk of the discrete super learner compared to other learners. $N = 3500$

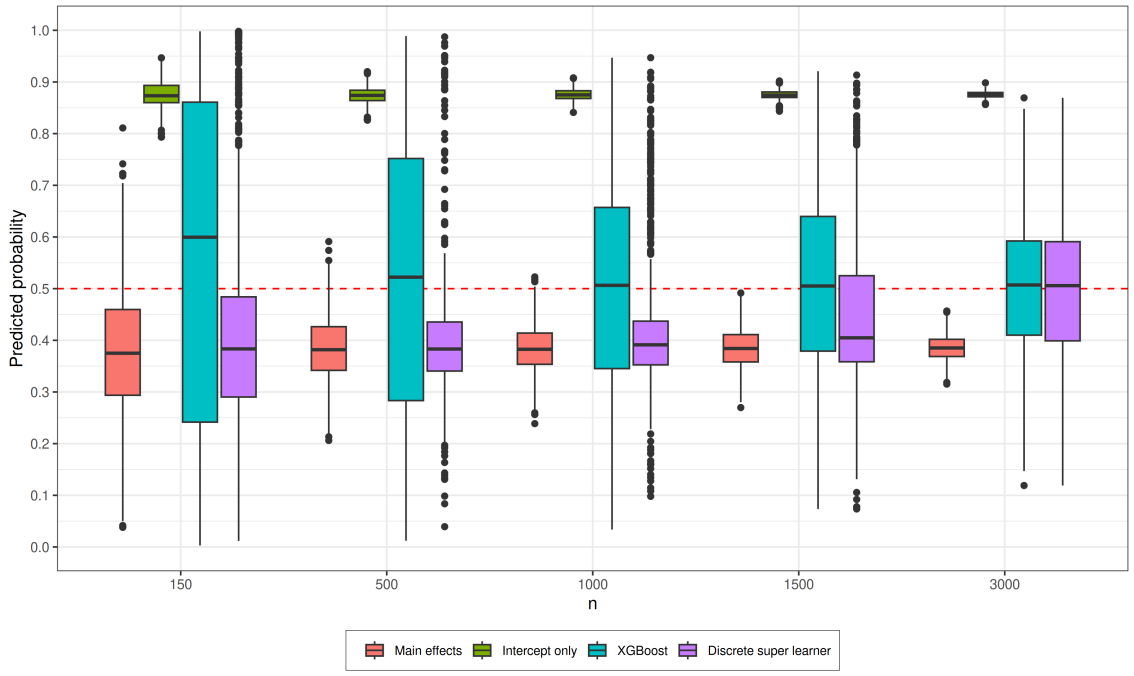

Figure 6: Variances of learner predictions for a single observation, each trained on n samples and evaluated $K = 1000$ times on a single observation

5.2 Discussion of results

Figures 4 and 5 illustrate how the discrete super learner performs in comparison to the learners in terms of loss over number of training samples. The plots are generated for two runs where the model is fitted on $n = 100, 200, \ldots, 3500$ observations, as indicated on the x-axis, then the empirical risk is calculated by evaluating each fitted learner on a fixed test sample of size of 5000. The test data is sampled from the data-generating distribution.
The first run perfectly illustrates how the discrete super learner is able to achieve the minimum risk. For small training sample sizes, the machine learning method XGBoost has a higher risk than the main effects logistic regression, and it is therefore more desirable for the discrete super learner to choose logistic regression despite the fact that it is misspecified. The discrete super learner consequently achieves the same risk as the logistic regression in the beginning, but for $n > 1200$ the risk of the discrete super learner becomes less than the logistic regression. Here XGBoost begins to achieve a lower risk than the misspecified logistic regression, and so the discrete super learner chooses XGBoost instead.
The second run shows that the discrete super learner might be unable to determine the learner with the lowest risk when the training sample size is small, which results in it moving in a zig-zag pattern between two learners that have quite similar risks. However, we see that the discrete super learner eventually chooses XGBoost as the training sample size becomes larger.
Figure 6 shows the variance of the predictions of each learner for a single observation, whose true probability is indicated by the red dashed line. Each learner have been trained $K = 1000$ times on $n = 150, 500, \ldots 3000$ samples and is used to predict K times after each training. The box plots are created from the K predictions.
We observe that the machine learning model, XGBoost, has the highest prediction variance across all training sample sizes. Recall that we only had two covariates, here having 1500 observations limits the range of predictions of our main effects model to be between 0.27

and 0.46. Whereas for XGBoost the predictions can vary from below 0.1 to above 0.9. While XGBoost is extremely efficient at minimizing loss, its predictions are unreliable unless one has an absurd amount of training data.
The discrete super learner attempts to b

References

[1] László Györfi et al. *A distribution-free theory of nonparametric regression*. Vol. 1. Springer, 2002.

[2] Mark J. van der Laan and Sandrine Dudoit. "Unified Cross-Validation Methodology For Selection Among Estimators and a General Cross-Validated Adaptive Epsilon-Net Estimator: Finite Sample Oracle Inequalities and Examples". In: *UC Berkeley Division of Biostatistics Working Paper Series* (Jan. 2003).

[3] Aad W. van der Vaart, Sandrine Dudoit, and Mark J. van der Laan. In: *Statistics & Decisions* 24.3 (2006), pp. 351–371. DOI: doi:10.1524/stnd.2006.24.3.351. URL: https://doi.org/10.1524/stnd.2006.24.3.351.

[4] Mark J Van der Laan, Eric C Polley, and Alan E Hubbard. "Super learner". In: *Statistical applications in genetics and molecular biology* 6.1 (2007).

[5] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[6] Steffen Lauritzen. *Mathematical Statistics: A Concise Course*. 2022.