Super Learners

and their oracle properties

Jinyang Liu (sqf320)

Department of Mathematical Sciences
University of Copenhagen

June 2023

Overview

Introduction

Binary regression

Let $O = (Y, X)$ be an observation for $Y \in \{0, 1\}$ and $X \in \mathcal{X}$ for $\mathcal{X} \subseteq \mathbb{R}^d$. We assume that $O \sim P$ for some $P \in \mathcal{P}$.

Let $\Theta = \{\theta \mid \theta : \mathcal{X} \to [0, 1] \text{ measurable}\}$ be the set of **regression functions**. We would like to **estimate** a function $\theta \in \Theta$ such that the mean squared error (MSE) or risk

$$R(\theta, P) = \int L(O, \theta) dP = \int (Y - \theta(X))^2 dP$$

is minimized. It turns out that the conditional expectation

$$x \mapsto E(Y \mid X = x) = P(Y \mid X = x)$$

is what minimizes the MSE. We refer to it as the **regression**.

Example of a regression function

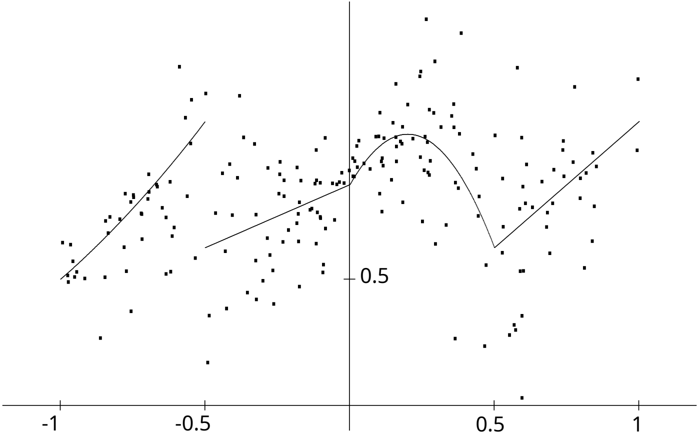

Figure: Example of a pathological regression that can be difficult to learn using parametric techniques. Here a continuous outcome Y is plotted against a single continuous covariate X (Györfi et al., 2002).

Linear approximation

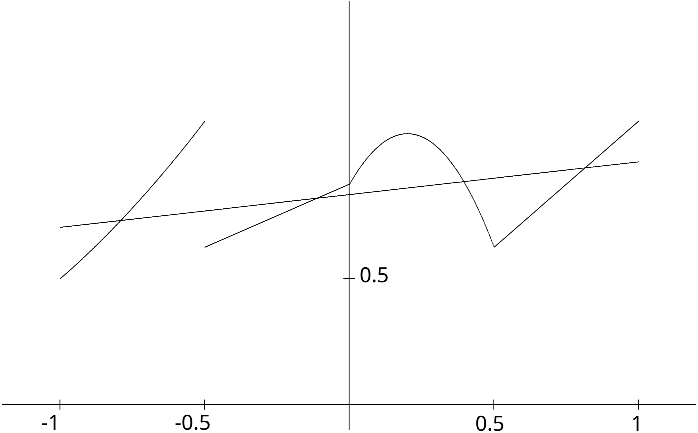

Figure: Approximating the regression using linear regression, which is very biased (Györfi et al., 2002).

Terminology

We observe $D_n = (O_1, \ldots, O_n)$, upon which we apply our **learning algorithms**

Definition (Learning algorithm)

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

We assume that the ψ is well-defined for all $n \in \mathbb{N}$ and that the ordering of the observations does not matter.

Definition (Learner or fitted learner)

Let ψ be a learning algorithm, a learner is the outcome of applying ψ to our data D_n denoted as $\psi(D_n)$, which is a map in Θ.

We usually have a **library of learning algorithms**,

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

for which we can use to estimate the regression.

K-fold Cross-validation

There is a one-to-one correspondance between our data D_n and the empirical measures over n observations

$$P_n = \sum_{i=1}^{n} \delta_{O_i},$$

K-fold cross-validation splits D_n into **validation** and **training** sets. The validation sets are indexed by $s \in \{1, \ldots, K\}$ and we denote the empirical measure over the validation set s as

$$P_{n,s}^1 := \frac{1}{n_1} \sum_{i:s(i)=s} \delta_{O_i}, \qquad P_{n,s}^0 := \frac{1}{n_0} \sum_{i:s(i)\neq s} \delta_{O_i}.$$

Here $s(i)$ denotes whether O_i is in the validation set s, and n_1, n_0 are the number of observations in the validation and training sets respectively.

K-fold cross-validation procedure

Cross-validation is used to evaluate each algorithm, and is the central idea of the **super learner**:

1. Randomly split D_n into K disjoint and exhaustive validation sets
2. For each $s \in \{1, \ldots, K\}$ fit each $\psi \in \Psi$ on the training data $P^0_{n,s}$ and obtain $\psi(P^0_{n,s})$
3. For each ψ, use $\psi(P^0_{n,s})$ to predict on the validation set to obtain **level-1 covariates**:

$$Z_i = \left(\psi_1(P^0_{n,s(i)})(X_i), \ldots, \psi_k(P^0_{n,s(i)})(X_i)\right)$$

4. Calculate the MSE of ψ on the validation set for $s \in \{1, \ldots, K\}$

$$R(\psi(P^0_{n,s}), P^1_{n,s}) = \frac{1}{n_1} \sum_{i:s(i)=s} \left(Y_i - \psi(P^0_{n,s})(X_i)\right)^2$$

Overview

Discrete Super Learner

Cross-validation allows us to select the algorithm with the lowest **empirical risk**

$$\hat{\psi}_n := \arg\min_{\psi \in \Psi} \frac{1}{K} \sum_{s=1}^{K} R(\psi(P_{n,s}^0), P_{n,s}^1),$$

also known as the **cross-validation selected algorihm**. The **discrete super learner** is simply the cross-validation selected algorithm fitted on the entire dataset

$$X \mapsto \hat{\psi}_n(P_n)(X).$$

It is compared to the **oracle selected learning algorithm** that has the true minimum risk

$$\tilde{\psi}_n := \arg\min_{\psi \in \Psi} \frac{1}{K} \sum_{s=1}^{K} R(\psi(P_{n,s}^0), P).$$

Discrete Super Learner: Asymptotic Equivalence

We let E_{D_n} denote the expectation wrt. the product measure of $O_1, \ldots, O_n$.

Theorem (Asymptotic equivalence)

If there exists an $\varepsilon > 0$ such that

$$E_{D_n} \frac{1}{K} \sum_{s=1}^{K} R(\tilde{\psi}_n(P_{n,s}^0), P) > \varepsilon \qquad \textit{for all } n \in \mathbb{N},$$

and if $n_1 = f(n)$ for some polynomial function f, then the risk of the super learner is asymptotically equivalent with the risk of the oracle selected learner, that is

$$\lim_{n \to \infty} \frac{E_{D_n} \frac{1}{K} \sum_{s=1}^{K} R(\hat{\psi}_n(P_{n,s}^0), P)}{E_{D_n} \frac{1}{K} \sum_{s=1}^{K} R(\tilde{\psi}_n(P_{n,s}^0), P)} = 1.$$

Overview

Ensemble Super Learner

Let $\mathcal{Z} \subseteq [0,1]^k$ be the learners' out-of-fold predictions from doing K-fold cross-validation (level 1 covariates).

Definition (Level 1 data)

The *level 1 data*, $\mathcal{L}_n \subseteq \{0,1\} \times \mathcal{Z}$, is the observed Y_i's concatenated with the level 1 covariates:

$$\begin{aligned}\mathcal{L}_n &= \{(Y_i; Z_i)\}_{i=1}^n \\ &= \{(Y_i; \psi_1(P^0_{n,s(i)})(X_i), \dots, \psi_k(P^0_{n,s(i)})(X_i))\}_{i=1}^n.\end{aligned}$$

Let $\mathcal{M}$ be the set of measurable functions, $\phi : \mathcal{Z} \to [0,1]$, known as the set of **meta learners**

Definition (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.

We estimate $E(Y \mid Z)$ by applying a **meta learning algorithm** to the level 1 data.

Definition (Meta learning algorithm)

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

Definition (Ensemble super learner)

Let $\phi = \Phi(\mathcal{L}_n)$ be the outcome of applying a meta learning algorithm Φ 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 $\Sigma(P_n)$.

Each learning algorithm is fitted on the entire dataset, and a meta learner is used to combine the predictions.

Ensemble Super Learner: Constrained Regression

There are many choices of the meta learning algorithm, one can fit a weighted linear combination of the learning algorithms where

$$\phi_a(z) = a \cdot z, \qquad \sum_{q=1}^{k} a_q = 1, a_q \geq 0 \text{ for all } q,$$

such that

$$\begin{aligned}\Sigma(P_n)(x) &= \phi_a(\psi_1(P_n)(x), \ldots, \psi_k(P_n)(x)) \\ &= \sum_{q=1}^{k} a_q \psi_q(P_n)(x) \in [0, 1].\end{aligned}$$

The optimal weighting, a, can be found by solving a **constrained least squares** on the level 1 data using quadratic programming techniques.

Overview

Simulations

We consider a simulated dataset consisting of two covariates X_1, X_2 and a binary outcome $Y \in \{0, 1\}$

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

where $\theta_0(x_1, x_2) = \mathrm{expit}(-3.5 - 0.3x_1 + 0.85x_2 + 0.35x_1x_2)$.

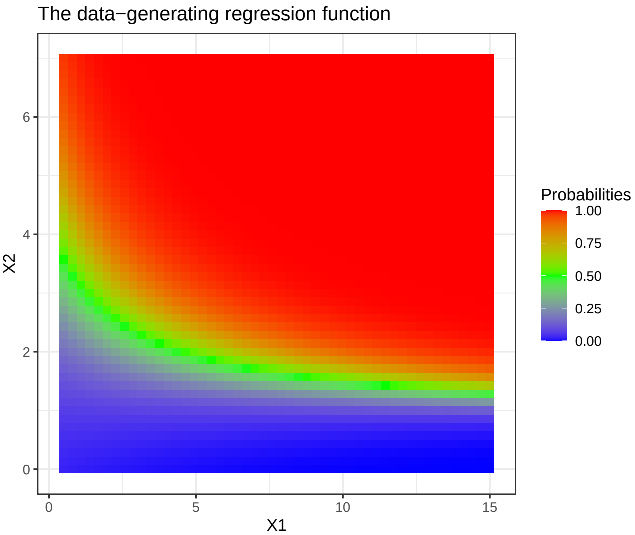

Figure: The data-generating regression plotted as a heat map. The two covariates X_1 and X_2 are mapped by the regression function θ_0 to a probability as indicated by the colors.

Simulations: Library of Algorithms

The regression is captured by using logistic regression with interaction terms. We use the following library of learning algorithms:

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`

The intercept model is included as a **baseline**, no learning algorithm should perform worse than the baseline.

Simulations: Logistic Regression and XGBoost

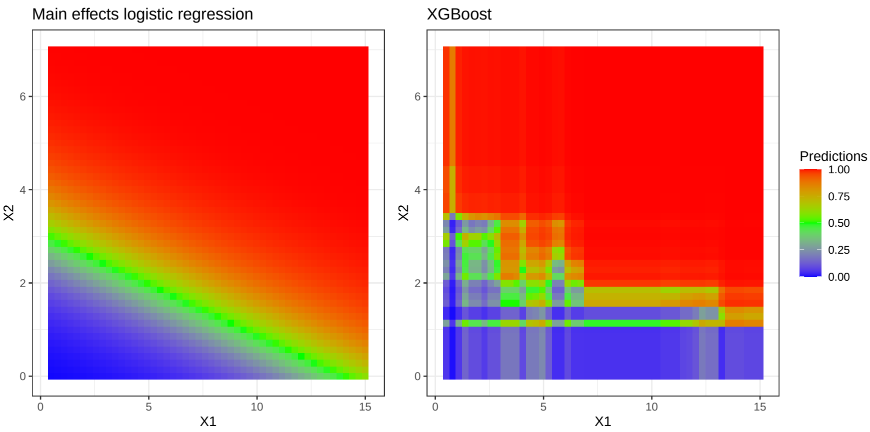

Figure: The predictions of the main effects logistic regression and XGBoost fitted on $1,000$ observations.

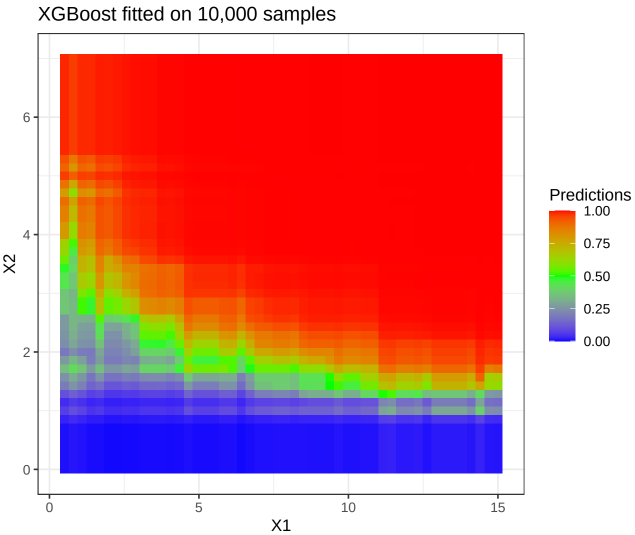

Figure: XGBoost becoming better at approximating the regression as the sample size increases. Here the predictions of XGBoost are visualized for a training sample size of $10,000$.

Simulations: Risk and Variance over Training Samples

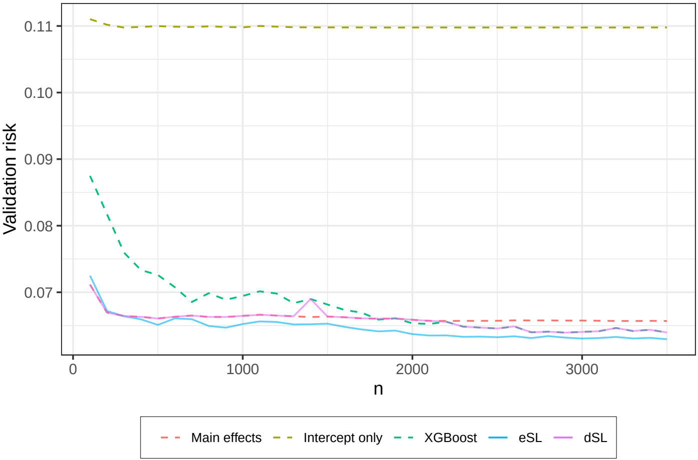

Figure: The validation risk of the super learners compared to other algorithms where the number of training samples are $n = 100, 200, \ldots, N = 3,500$.

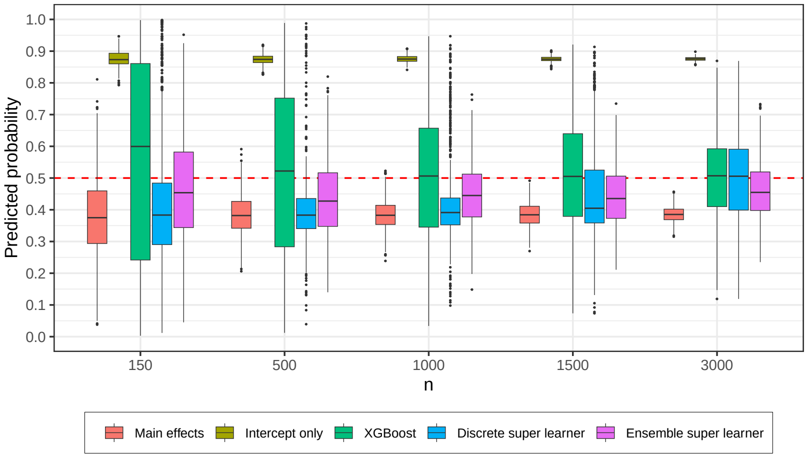

Figure: The variance of the super learners compared to other algorithms. Each algorithm is fitted $K = 1,000$ times on n samples and used to predict K times on a single observation.

Locally Weighted Ensemble Super Learner

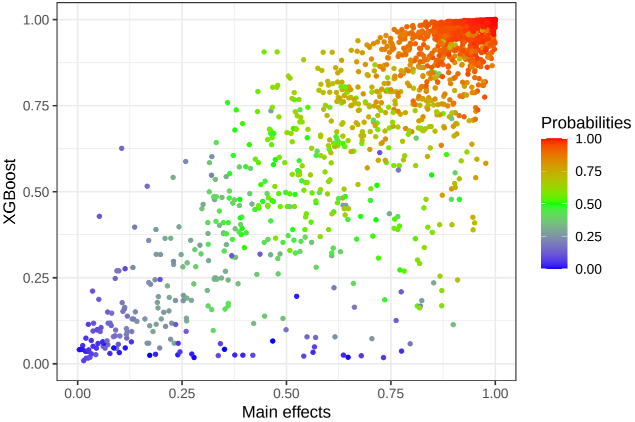

Figure: Predictions of XGBoost vs main effects model. The learners' prediction on a single observation represent a point in the unit square, the point is colored by the probability obtained from applying the true regression function on that observation.

Clustering level 1 covariates using k-means

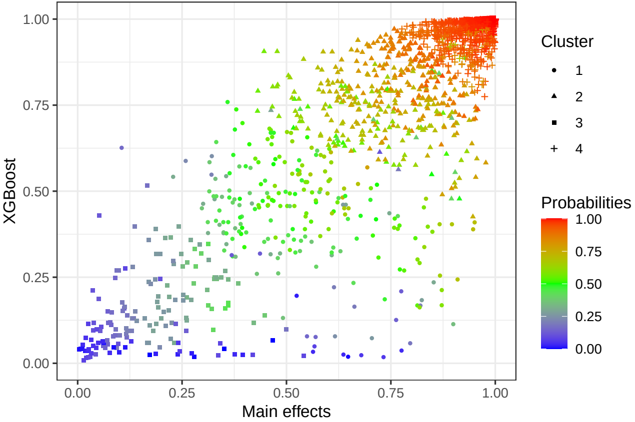

Figure: The predictions are clustered into 4 groups using k-means clustering.