Super Learners

and their oracle properties

Jinyang Liu (sqf320)

Department of Mathematical Sciences
University of Copenhagen

June 2023

Overview

1. Introduction
   - Terminology
   - Cross-validation
2. Discrete Super Learner
   - Oracle Property
3. Ensemble Super Learner
   - Constrained Regression
4. Simulations
   - Validation risk and variance
   - Locally weighted eSL
5. Concluding Remarks

Binary regression

Let $O = (Y, X) \sim P \in \mathcal{P}$ be an observation where $Y \in \{0,1\}$ is our **outcome** and $X \in \mathcal{X} \subseteq \mathbb{R}^d$ is our **covariates**.

The set of **regression functions** is

$$\Theta = \{\theta \mid \theta : \mathcal{X} \to [0,1] \text{ measurable}\}.$$

The mean squared error (MSE) or **risk** of $\theta \in \Theta$

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

is minimized by the conditional expectation referred to as the **true regression**

$$x \mapsto E(Y \mid X = x) = P(Y = 1 \mid X = x).$$

We will consider the case of binary regression, where an observation consists of a binary outcome Y and covariates X. The observation is distributed according to P which comes from statistical model
Binary regression estimates the probability of Y being equal to 1 given the covariates X. It differs from binary classification in that the prediction is continuous.
We consider a set of regression functions that map from our covariates to the probability interval, which are the functions we use to predict the probability.
A way to evaluate the regression functions is to consider the MSE or risk, which we denote as R theta P, and is the integral of squared difference between Y and theta applied on X wrt. to P.
However, we can never calculate the true MSE in the form above directly, since it depends on P, which is unknown.
It turns out, that the MSE is minimized by the conditional expectation, which we refer to as the true regression.
our goal is therefore to estimate the true regression, one way is to assume... problem is that our assumptions are frequently incorrect, and we risk misspecifying the model so it is tempting to use nonparametric machine learning methods which are less biased.
We now introduce some terminology

Terminology

Let $D_n = (O_1, \ldots, O_n) \in \mathcal{O}^n$ be the **data**.

Learning algorithm

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

Learner or fitted learner

A learner $\psi(D_n) \in \Theta$ is the outcome of applying ψ to the data D_n.

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

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

which we can use to estimate $P(Y = 1 \mid X = x)$.

1. Let Dn be the data which consists of the observations from 1 up to n
2. A learning algorithm, denoted as psi, is a map from the data to the set of regression functions
3. A learner is then simply the outcome of applying the learning algorithm to the data
4. A learning algorithm could for example be logistic regression, or a neural network
5. In machine learning, the algorithm is fitted or trained on the observed data, and the outcome is a a fitted model, which is used to predict on new covariates
6. we usually have a library of learning algorithms at our disposoal, which we can use to estimate the true regression

Two learners: Logistic Regression and XGBoost

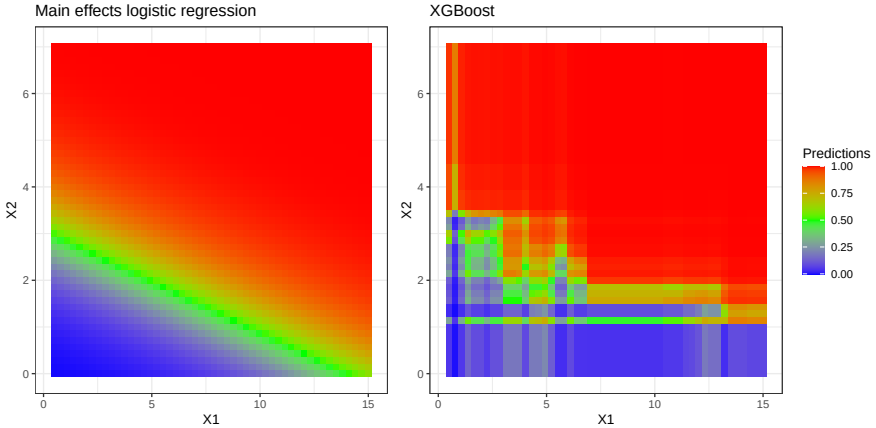

Figure: The predictions of the main effects logistic regression and XGBoost fitted on 1,000 observations.

1. As an example, we can consider logistic regression and the gradient-boosting tree-based algorithm XGBoost.
2. Here we have two covariates X1 and X2 and a binary outcome, the two algorithms are fitted on 1000 observations and are used to predict on the covariates in a grid
3. the predicted probabilities are colored from 0 up to 1
4. We see that there's a huge difference in the pattern of predicted probabilities by the two algorithms
5. Logistic regression, which assumes that the statistical model is binomial has a much smoother pattern than XGBoost, which since it is based on decision trees, predicts the probability in a patchy pattern
6. A natual question arises which is how do we determine the best learning algorithm from looking at the predictions?
7. Given that we do not know the data-generating distribution P, it is not possible to say if any of the algorithms has correctly estimated the regression
8. It might be the case that the true regression is pathological, and assumes the irregular pattern that XGBoost has estimated
9. But this seems improbable in a real-world scenario, and one might believe that it is more smooth as assumed by the logistic regression
10. A way to evaluate the algorithms is to do K-fold cross-validation

K-fold Cross-validation

There is a one-to-one correspondence 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 K **validation** and **training** sets. The splits are indexed by $s \in \{1, \ldots, K\}$ and we denote the empirical measures over the validation and training sets of split s as

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

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

1. We first note that there is a one-to-one correspondence between the data and the empirical measures over n observations
2. Where the delta O i is the dirac measure over O i
3. Indeed, from the observations we can construct the empirical measures, and by knowing the empirical measures we also know our observations
4. The learning algorithms was a function of the data, but can just as well a function of the empirical measures due to this correspondence
5. K-fold cross-validation splits the data in validation and training sets ..
6. s of i denotes the split s where observation i is in the validation set s. The variable s really just indexes the validation sets, since for each s you have a validation set that is disjoint from all other validation sets, here the training set is the observations complement of the validation set

The **super learner** K-fold cross-validation procedure

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 s and 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. For $s \in \{1, \ldots, K\}$ calculate the MSE of ψ on the validation set

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

1. K-fold cross-validation is integral in the super learner procedure, the super learneris in fact just cross-validation.
2. The level 1 covariates are obtained by fitting each learner to the training data pertaining to the split s, and then using the fitted learners for that split to predict on the observations in the validation set
3. The mean squared error for an algorithm in a specific split, can be calculated by evaluating the fitted learner wrt. the empirical measure of the validation set of that split.
4. If we would calculate the true MSE then p n s one would be replaced with P in the second argument of the risk function

Overview

1. Introduction
   - Terminology
   - Cross-validation
2. Discrete Super Learner
   - Oracle Property
3. Ensemble Super Learner
   - Constrained Regression
4. Simulations
   - Validation risk and variance
   - Locally weighted eSL
5. Concluding Remarks

Discrete Super Learner

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

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

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

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

We compare it to the **oracle selected learning algorithm** that has the true minimum risk

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

1. Cross validation allows us to select the aglorithm with the lowest empirical risk
2. We do this by averaging all the MSEs calculated on the validation sets and then take the algorithm with the minimum risk.
3. We compare it to the oracle selected algorithm, which selects the learning algorithm that has the true minimum risk over all K splits, the reason it is called the oracle is because we evaluate the risk wrt. the data-generating distribution P as seen in the second argument of the risk function
4. The discrete super learner is simply the cross-validated selected algorithm fitted on the entire dataset

Discrete Super Learner

Theorem (Asymptotic equivalence)

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

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

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

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

1. The following result holds for the discrete super learner
2. If the risk of the oracle selected learner is strictly non-zero for all n, and if the number of validation samples is polynomial in n, the we have that the risk of the discrete super learner is asymptotically equivalent with the risk of the oracle selected learner
3. Here E D n means to integrate out the observations O 1 up to n which were stochastic
4. The assumption that the true risk is strictly non-zero holds when we are using MSE in regression

Overview

1. Introduction
   - Terminology
   - Cross-validation
2. Discrete Super Learner
   - Oracle Property
3. Ensemble Super Learner
   - Constrained Regression
4. Simulations
   - Validation risk and variance
   - Locally weighted eSL
5. Concluding Remarks

Ensemble Super Learner

The **ensemble super learner** does not select one specific learner but instead creates a combination of all the learners

Level 1 data

The *level 1 data*, combines the observed outcomes Y_i with the level 1 covariates:

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

Meta learning algorithm

A *meta learning algorithm* Φ is applied to the level 1 data to create a *meta learner*: $\mathcal{L}_n \mapsto \Phi(\mathcal{L}_n)$.

1. The ensemble super learner is an extension of the discrete super learner in that it does not select one specific learner from the library but instead creates a combination of all the learners
2. We first define the level 1 data, which is the observed outcomes Y i combined with the level one covariates, which were the learners out-of-fold predictions on the validation data
3. A meta learning algorithm, is applied to the level 1 data to create a meta learner

Ensemble super learner

Ensemble super learner

Let $\phi = \Phi(\mathcal{L}_n)$ be the meta learner from 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* which we denote as $\Sigma(P_n)$.

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

1. The ensemble super learner is obtained by fitting a meta learning algorithm to the level 1 data to obtain a meta learner, then each learning algorithm in the library is fitted on the entire dataset
2. When we wish to predict on new covariates x, we use each fitted learner to evaluate on x and use the meta learner to combine the predictions

Ensemble Super Learner: Constrained Regression

There are many meta learning algorithms. One can for example fit a weighted linear combination of the learners where for $a \in \mathbb{R}^k$

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

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

1. Many possible meta learning algorithms can exist, one could for example fit a weighted linear combination of the learning algorithms where the parameter a is a weight vector in R k whose entries are positive and sum to 1
2. The ensemble super learner applied to new covariates x is to take a weighted linear combination of the learners with the weights specified by the weight parameter a
3. These weights can be found by solving a constrained least squares using quadratic programming

Overview

1. Introduction
   - Terminology
   - Cross-validation
2. Discrete Super Learner
   - Oracle Property
3. Ensemble Super Learner
   - Constrained Regression
4. Simulations
   - Validation risk and variance
   - Locally weighted eSL
5. Concluding Remarks

Simulations

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

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

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

1. We consider a simulated dataset consisting of two covariates and a binary outcome Y which is specified by the following conditional distributions
2. Where theta 0 which is the true regression is given explicitly by the following fomula
3. One can visualize the true regression as a heatmap by mapping the grid of the two covariates to a probability as indicated by the colors

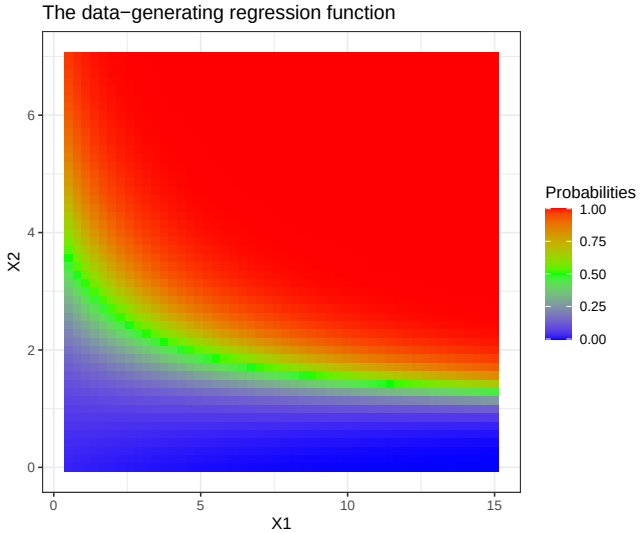

Figure: The true regression plotted as a heat map. The two covariates X_1 and X_2 are mapped by the regression function θ_0 to a probability as indicated by the colors.

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 **benchmark**, no learning algorithm should perform worse than the benchmark.

1. The true regression is captured by logistic regression with interaction terms, and we will consider the following library as an example
2. We have an intercept logistic regression which is included as a benchmark, the logistic regression with only main effects terms and no interaction, and XGBoost with hyperparameters
3. The intercept model is included as a benchmark, so no learning algorithm should perform worse than the benchmark

Simulations: Logistic Regression and XGBoost

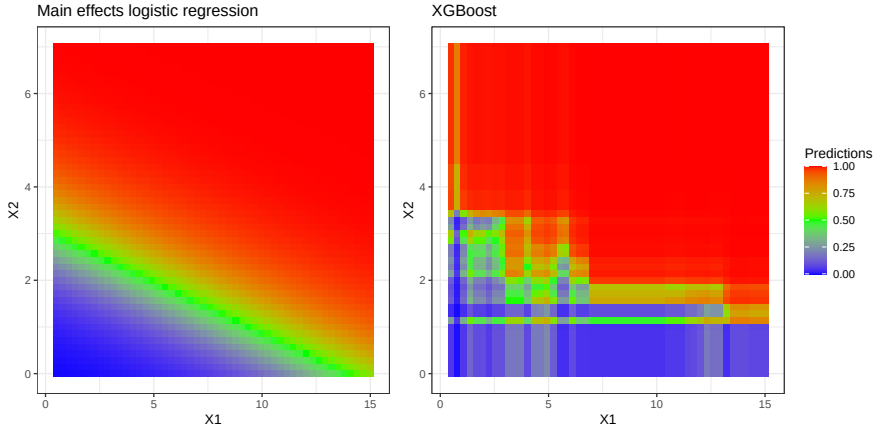

Figure: The predictions of the main effects logistic regression and XGBoost fitted on 1,000 observations.

1. These plots are from the third slide from before, and we see that the main effects model is very biased and cannot model the interaction effect between two covariates
2. XGBoost predicts a patchy patterm, which becomes smoother as we fit with more samples, fitting on 10,000 observations yields the following plot

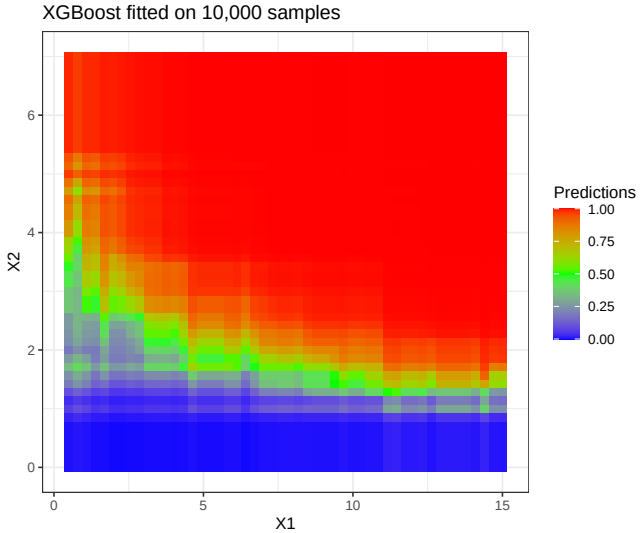

Figure: XGBoost becoming better at approximating the regression as the sample size increases. Here the predictions of XGBoost are visualized for a training sample size of 10,000.

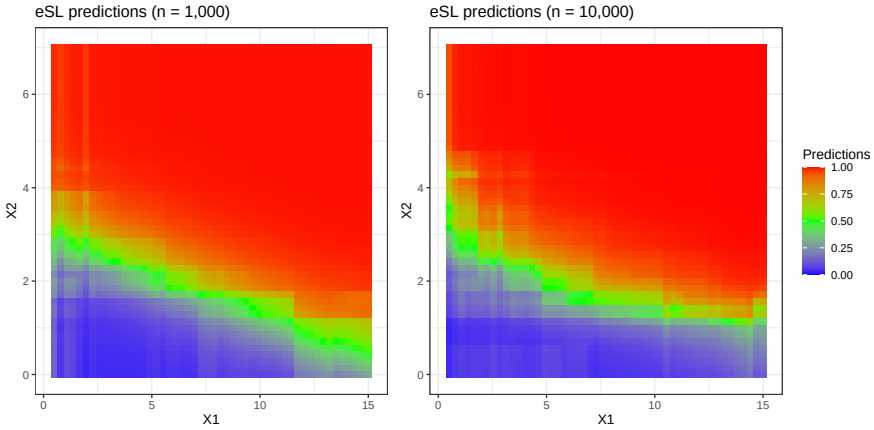

Figure: The predictions by the ensemble super learner using the constrained regression meta learning algorithm fitted on 1,000 and 10,000 observations.

1. The ensemble super learner combines the learner predictions, and as we see in the following plots, it seems that it is indeed a blend of the smooth parametric logistic regression and the patchy pattern of xgboost
2. We see that there is some gradient where the predictions are around 0.5, intertwined with the patchiness characteristic of XGBoost

Simulations: Risk and Variance over Training Samples

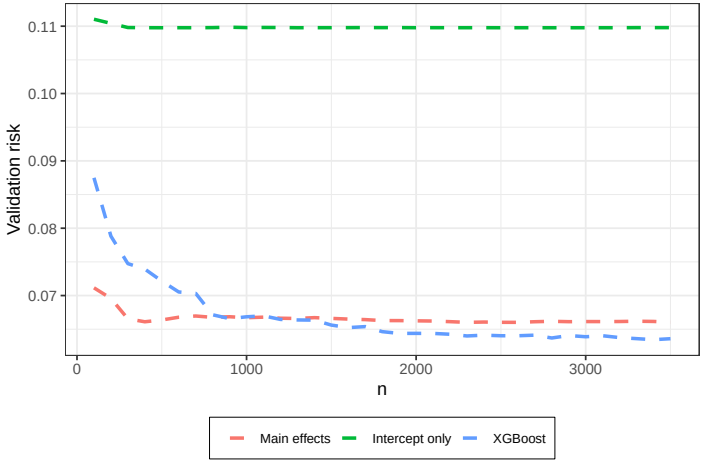

Figure: The validation risk of the learning algorithms where training samples are $n = 100, 200, \ldots, N = 3{,}500$.

1. Here we have plotted validation risks over the number of training samples, the validation risks are calculated by sampling 10000 observations from the true distribution, each algorithm is fitted on n observations and an empirical risk is calculated on the validation set
2. We see that the validation risk for XGBoost steadily decreases, whereas for the the logistic regression it appears quite stable
3. We also notice that XGBoost has a high validation risk for low samples, which is characteristic of machine learning methods that are very flexible, and can fit all sorts of distributions
4. It makes sense that since XGBoost has not been trained on as much data, then the validation risk is necessarily higher
5. XGboost eventually beats the main effects logistic regression when the number of training samples increase

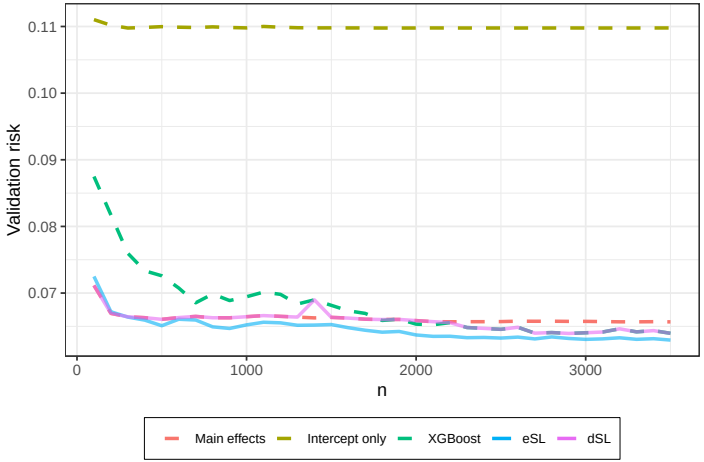

Figure: The validation risk of the super learners compared to the library of algorithms where the number of training samples are $n = 100, 200, \ldots, N = 3{,}500$.

1. Here is a plot where the validation risks of the super learners are included
2. The pink line, representing the discrete super learner, has the same risk to the main effects model for small samples. However, as the sample size grows and the risk of XGBoost becomes smaller, the risk of the discrete super learner aligns with that of XGBoost.
3. This is in line with the fact that the discrete super learner selects a learner from the library, and hence the minimum risk that it can achieve must be the risk of one of the learners
4. The ensemble super learner in blue, is a combination of different learners achieves a risk that is strictly lower than all learners including the discrete super learner

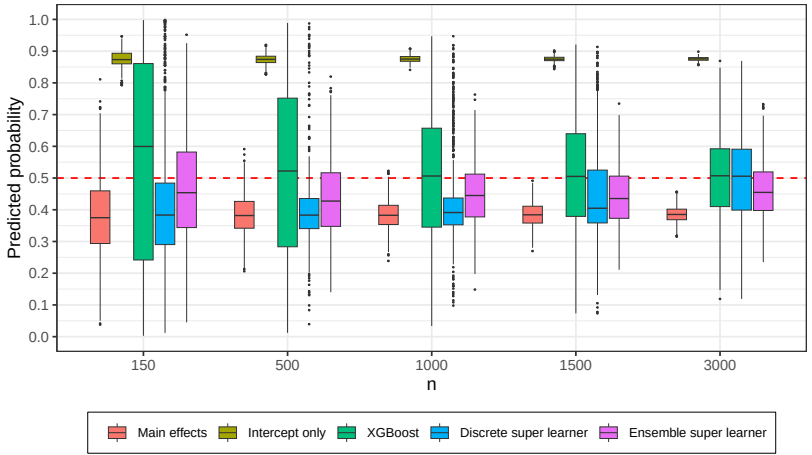

Figure: The variances of the super learners compared to other algorithms. Each algorithm is fitted $K = 1{,}000$ times on n samples and used to predict K times on a single observation.

1. In this plot we have fitted each algorithm 1000 times on n samples as indicated on the horizontal axis, and each time they are fitted, they are used to predict on a single fixed observation whose true probability is 0.5
2. We see that the variance of XGBoost is extremely high, ranging from 0.1 to 0.9 for a single observation
3. In comparison, the logistic regression models achieve a much lower variance, but are in turn heavily biased
4. which illustrates the bias-variance tradeoff between using data-adaptive machine learning methods and parametric methods such as logistic regression
5. An interesting observation here is that the ensemble super learner, which is a combination of the learners, is not as biased as the logistic regression, but has a higher variance, though less than XGBoost

Locally Weighted Ensemble Super Learner

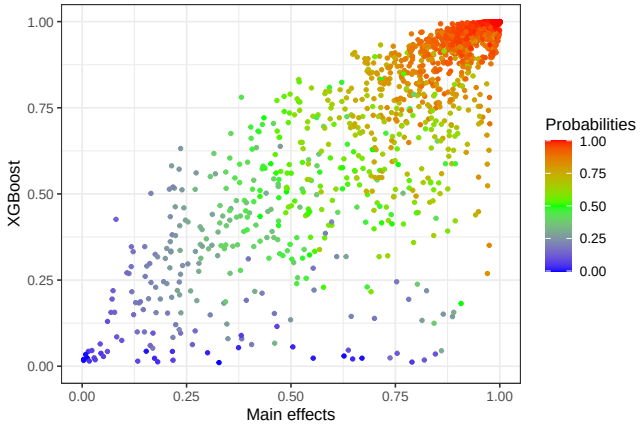

Figure: Predictions of XGBoost vs main effects model in level 1 covariates. The learners' prediction on a single observation represent a point in the unit square, the point is colored by the probability obtained from applying the true regression function on that observation.

1. i will now introduce the locally weighted ensemble super learner is my own extension
2. Here i have visualized the predictions of XGBoost vs Main effects, each point represents an observation, and is colored by their true probabilities, the axes indicate the predictions of the two learning algorithms
3. We see that XGBoost and the main effects model disagree significantly on some observations
4. For example as seen in the lower right corner, the main effects sometimes predicts the probability to be around 0.7 whereas XGBoost predicts to be around 0.1
5. One idea would be to weigh certain learners more than others in certain regions, for example we might weigh XGBoost a lot more in the lower right corner as it seems to be more accurate

Clustering level 1 covariates using *k*-means

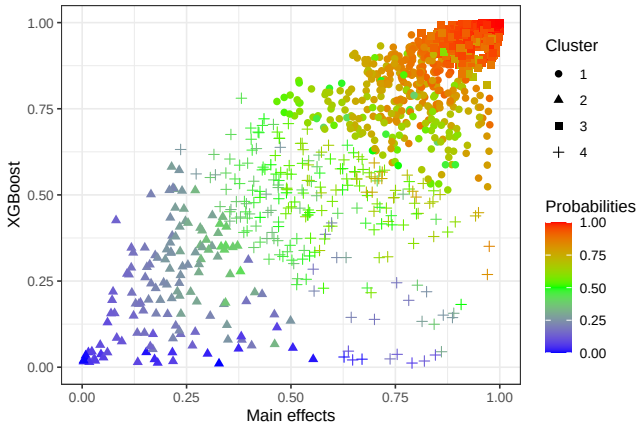

Figure: The predictions are clustered into 4 groups using *k*-means clustering.

1. Here I have used k-means clustering to cluster the level-1 covariates into 4 groups, for each group, an ensemble super learner is used combines the predictions using weights specifically fitted for that group

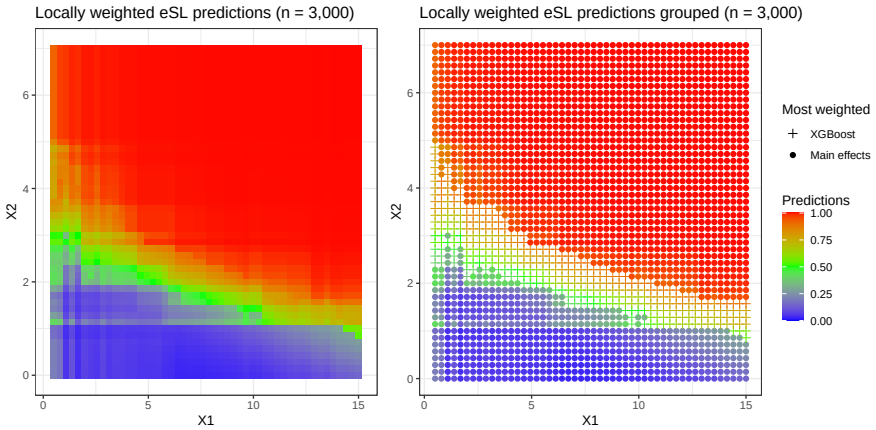

Figure: Predictions of locally weighted ensemble super learner, where the predictions of XGBoost have highest weight when $p \approx 0.5$

1. One can then stratify the predicted probabilities according to the learner that was weighed the most
2. Here XGBoost has the highest weight when the predicted probability is around 0.5 which is quite interesting to see
3. It seems also that there is a clearer gradient when going from 0.5 to 0.7

Overview

1. Introduction
   - Terminology
   - Cross-validation
2. Discrete Super Learner
   - Oracle Property
3. Ensemble Super Learner
   - Constrained Regression
4. Simulations
   - Validation risk and variance
   - Locally weighted eSL
5. Concluding Remarks

Concluding Remarks

1. Oracle property also holds for the ensemble super learner

1. Similar oracle property holds for the ensemble super learner which is a minor extension to the oracle result of the discrete super learner

Concluding Remarks

1. Oracle property also holds for the ensemble super learner
2. How can we make machine learning methods more stable?

1. As we have seen with the machine learning method XGBoost, it has an extremely high variance unless one has a lot of observations. The high variance results in undesirable irregular patchy predictions which does not model the data-generating distribution
2. The locally weighted ensemble super learner was an attempt at smoothening the irregular predictions of XGBoost by combining it with a parametric method such as logistic regression
3. My hope was that by weighing the parametric regression more in certain regions, one can selectively ignore the noisy predictions from XGBoost, but still retain its more accurate predictions that the parametric regression could not capture

Concluding Remarks

1. Oracle property also holds for the ensemble super learner
2. How can we make machine learning methods more stable?
3. The ensemble super learner seeks to balance the bias-variance tradeoff

$$R(\theta, P) = E(\theta(X) - m(X))^2 + E(m(X) - Y)^2$$

$$E(\theta(X) - m(X))^2 = \text{Var}(\theta(X)) + \text{Bias}(\theta(X))^2$$

where m is the true regression

1. Finally, if we consider the ensemble super learner with the quadratic risk function, then one can argue that it attempts to select the most optimal weighting scheme that balances the bias-variance tradeoff
2. It is well-known that the quadratic risk has the following decomposition
3. So by minimizing the quadratic risk over theta, we are minimizing the first the term which can decomposed into the variance of theta plus its squared bias