Machine Learning for policy evaluation: supervised learning

Jérémy Do Nascimento Miguel*

*BSE, Univ. Bordeaux, jdnmiguel@u-bordeaux.fr

Master APP - EADD; Univ. Bordeaux - Fall 2023

Plan for this lesson

We will focus on some basics supervised machine learning algorithms

1. Linear regression models and their variants
   - 1.1 Ridge
   - 1.2 LASSO
   - 1.3 Post-Lasso
2. Tree-based methods
   - 2.1 Regression trees
   - 2.2 Random forests
   - 2.3 Boosting

Overview

Regularized Linear regression models
- Ridge
- LASSO
- Post-LASSO

Tree-based methods
- Regression trees
- Random forests
- Boosting

Linear models

Recall the standard linear model is defined as follow:

$$y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

Why we might want to use alternatives to OLS? Let's consider we want to run this model with n observations and p covariates

1. Prediction accuracy:
   - If $n >> p \Rightarrow$ OLS estimates have low variance
   - If n is close to p, then OLS has high variability yielding to poor predictions and overfitting issue. Why? Many of the potential covariates will be correlated with each other $\Rightarrow$ not statistically significant. Should we keep them in or leave them out?
   - If $p > n$, there is not a single linear solution but several resulting in poor Test MSE

2. Model interpretability:
   - When $\times$ covariates are irrelevant, keeping them in the model increases the complexity while removing them increases the interpretability

Solutions? (i) *Subset selection* using theory to pick p; (ii) *Shrinkage*: use all p but shrunk the estimated coefficient; (iii) *dimension-reduction* : e.g., pca

Shrinkage method: Ridge Regression

Shrinkage methods consists in fititing a model with all p covariates using a penalized estimate. Similar to least square, except that the coefficient are estimated by minimizing:

$$\hat{\beta}^R = argmin_\beta \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2 = RSS + \lambda \sum_{j=1}^{p}\beta_j^2$$

with $\lambda > 0$: tuning parameter. Two different criteria

- Coefficient estimates fit the data well: min. the RSS
- The second term is a shrinkage penalty is small when $\beta_1, \ldots, \beta_p$ are close to zero

λ controls the relative impact of these two on the reg. coeff. estimates

- λ=0, ridge regression = least square estimates
- $\lambda \to \infty$: shrinkage penalty $\uparrow$ and $\hat{\beta}^R \to 0$

Key is to select the good value of λ. Will see in Lab.

Advantages of Ridge over OLS

Rooted in the bias-variance trade-off:

- As $\lambda \uparrow$, the flexibility of the ridge regression fit $\downarrow$, leading to $\downarrow$ variance but $\uparrow$ bias

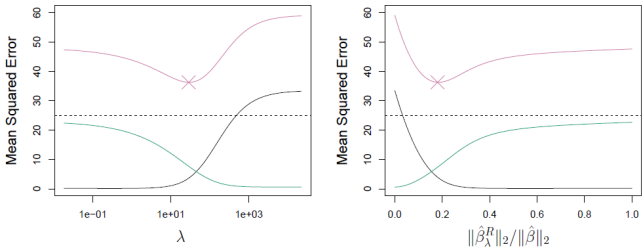


FIGURE 6.5. *Squared bias (black), variance (green), and test mean squared error (purple) for the ridge regression predictions on a simulated data set, as a function of* λ *and* $\|\hat{\beta}^R_\lambda\|_2/\|\hat{\beta}\|_2$. *The horizontal dashed lines indicate the minimum possible MSE. The purple crosses indicate the ridge regression models for which the MSE is smallest.*

LASSO

Ridge drawback: includes all p predictors in the final model. Not a problem in prediction accuracy but it is for model interpretability. The most common used in economics is the **LASSO**.

LASSO = Least Absolute Shrinkage and Selection Operator

$$\beta_L = argmin_\beta \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\beta_j|$$

- $\hat{\beta}_\lambda^L$: LASSO coefficient which minimizes $RSS + \lambda \sum_{j=1}^{p}|\beta_j|$
- A linear regression with a penalty term, and a tuning parameter λ
- The penalty term shrinks some coefficients towards zero when λ is large enough
- It performs variable selection by shrunking some coeff. estimates to be exactly zero
- Ease model interpretation resulting in a **sparse model** (model with a subset of covariates)

An example of Lasso application

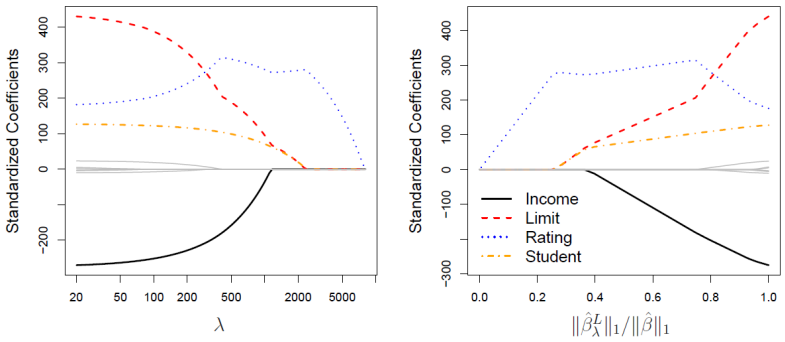


FIGURE 6.6. *The standardized lasso coefficients on the* `Credit` *data set are shown as a function of* λ *and* $\|\hat{\beta}_\lambda^L\|_1/\|\hat{\beta}\|_1$.

Variable Selection Property of the Lasso

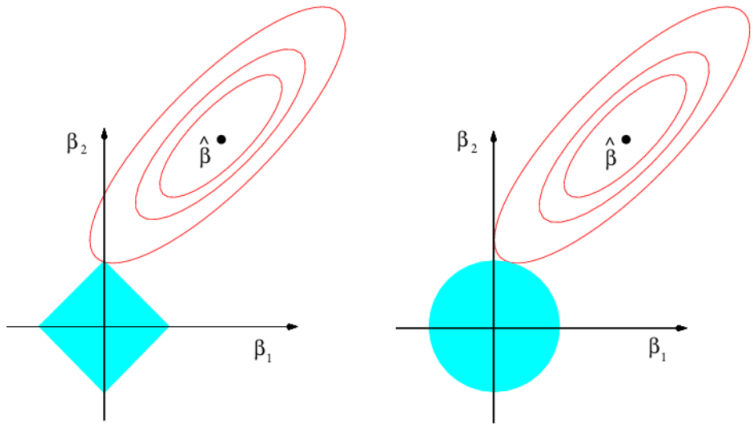


FIGURE 6.7. *Contours of the error and constraint functions for the lasso* (left) *and ridge regression* (right). *The solid blue areas are the constraint regions,* $|\beta_1| + |\beta_2| \leq s$ *and* $\beta_1^2 + \beta_2^2 \leq s$, *while the red ellipses are the contours of the RSS.*

Lasso vs Ridge?

Neither Lasso nor the ridge universally dominates the other, **but** some rules/patterns are well known.

- Lasso performs better in setting where a relativelly small number of predictors have substantial coefficients and the remaining have very small or zero
- Ridge regression will perform better when the response is a function of many predictors, all with roughly same size

Issue: number of predictors that is related to the response is never known *a priori* for real data sets $\Rightarrow$ CV can be used to determine which one is the best

Selecting the tuning parameter

How to choose λ? **Cross-validation** provides an easy approach:

- Choose a grid of λ values
- Compute the cross-validation error for each value of λ
- Select λ for wich the CVE is smallest
- Re-fit the model using all of the available observations and the selected

Selecting the tuning parameter

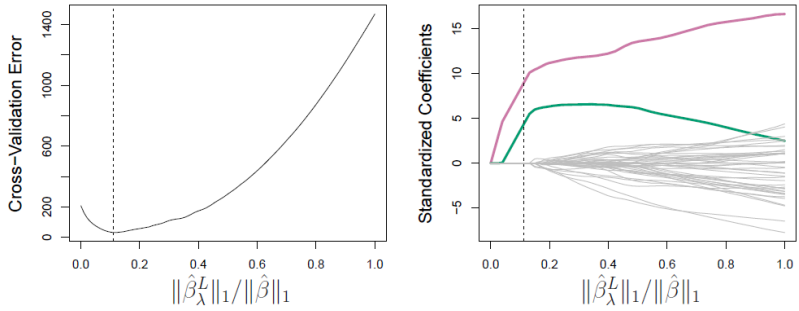
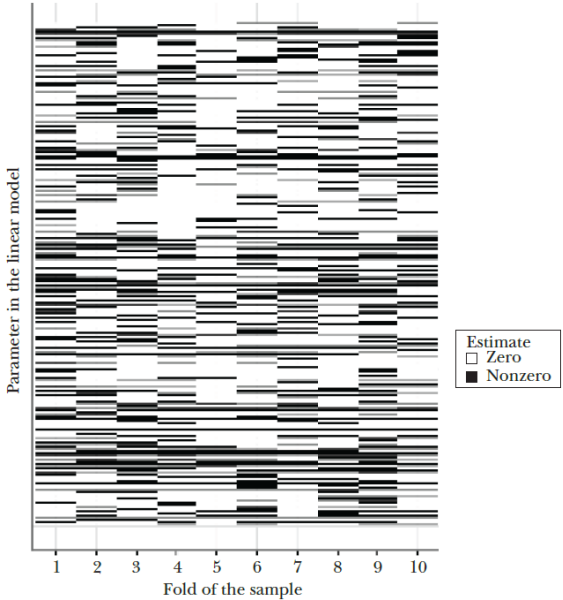


FIGURE 6.13. Left*: Ten-fold cross-validation MSE for the lasso, applied to the sparse simulated data set from Figure 6.9.* Right: *The corresponding lasso coefficient estimates are displayed. The two signal variables are shown in color, and the noise variables are in gray. The vertical dashed lines indicate the lasso fit for which the cross-validation error is smallest.*

LASSO: few elements to have in mind

- **LASSO is not scale invariant**! Changing the scale of the covariates changes the estimator. We should standardize any variable x to be mean-zero unit variance
- Machine learning has different goals than microeconometrics. Even when a ML looks like a regression, the interpretation of its coefficient estimates are different than OLS ones.
  - The coefficients are in general biased
  - Whether a covariate is included or excluded in the regression post-LASSO is meaningless

Interpreting the variables from LASSO? Mullainathan and Spiess (2017)



Post-LASSO

Getting around the inintepretability of the variables selected by LASSO by shifting our objective: we pick which covariates we are interested in.

A solution is the Belloni et al. (2014)'s post-LASSO estimator. The intuition is the following:

1. They use a LASSO to shrink a model
2. Use the selected covariates in a "regular" OLS model to predict y
3. Provide theoretical and use simulation to show that post-LASSO outperform LASSO under certain conditions
4. Why? Post-LASSO reduces the coefficient bias associated with shrinkage
5. Note: this relies on the "approximately sparse" assumption (only a subset of covariates matter)

Issues in high-dimension settings

What can go wrong when one applies techniques no intended for high-dimensional settings?

- When $p >> n$ or even p slightly larger small than $n \rightarrow$ Cannot perform OLS
  - Reason: OLS yields to a set of coefficient estimates that result in a perfect fit to the data, such that the residuals are zero
  - Possible to perfectly fit the training data in the high-dimensional setting, the resulting linear model will perform extremely poorly on an independent test set
  - Too flexible and overfit the data
- Adding more variables $\uparrow R^2$ and $\downarrow$ Train MSE even though the new variables are unrelated to y (same way $\uparrow$ Test MSE because $\uparrow$ variance of coeff.)

LASSO in Stata

LASSO is implemented using:

```
[LASSO] lasso — Lasso for prediction and model selection
                (View complete PDF manual entry)
```

post-LASSO is implemented using:

```
pdslasso and ivlasso —
             Programs for post-selection and post-regularization OLS or IV estimation and inference
```

Overview

Regularized Linear regression models
- Ridge
- LASSO
- Post-LASSO

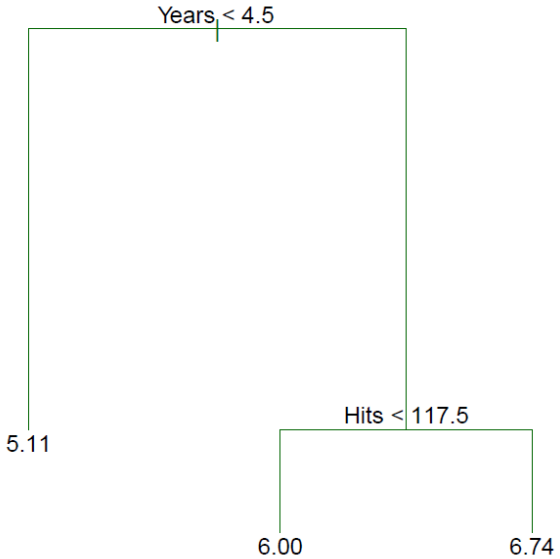
Tree-based methods
- Regression trees
- Random forests
- Boosting

Regression trees

Intuition:

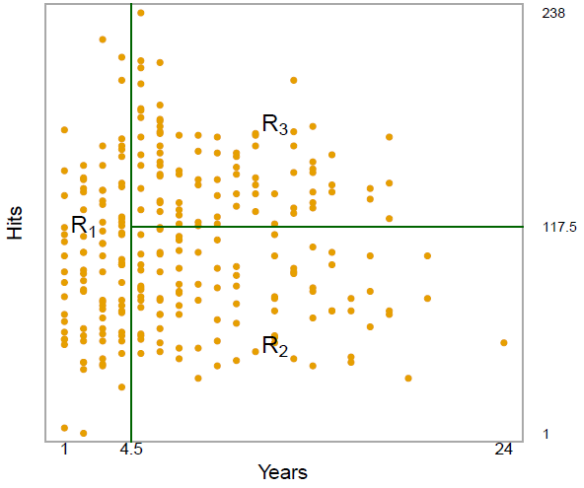
- Segment the set of possible values for $X_1, \ldots, X_p$ (i.e., predictor space) into J distinct and not overlapping regions $R_1, \ldots, R_J$
- Assign the region average outcome predicted value (region mean) for every observation in a region

A regression tree and some terminology



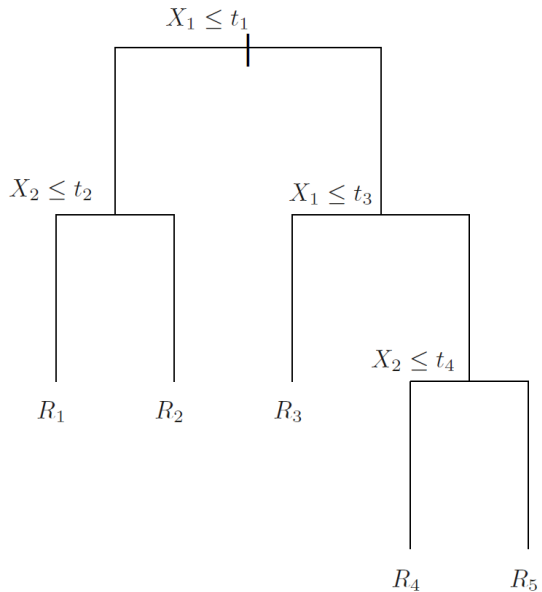
From (Hastie et al., 2009)

Representing a tree formally



From (Hastie et al., 2009)

Basic approach to grow a tree



Growing a regression tree

1. Find boxes $R_1, \ldots, R_J$ minimizing RSS $= \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$
   - Problem: infeasible to consider all possible partitions. Solution is to implement a recursive binary splitting to grow the largest tree with the training data
   - Top-down: starts at the top of the tree, then successively splits into new branches
   - Greddy: best split is made at each step.

2. Repeat the procedure at each node until a stopping criterion is reached:
   - Minimum # of observations in each terminal node R_m or maximum tree depth

3. Once all regions have been created, $\hat{f}(x)$ is equal to the average outcome within each region R_m that the x of an observation fall into

Practice: start with all the data, select the covariates X_j and cutpoint s to split the regions leading to highest possible reduction in RSS.

Regression trees: when to stop??

Similarly to previous ML algorithms, we are facing the bias-variance trade-off

- Larger trees (= $\times$ regions): minimize bias but have large variance $\Rightarrow$ overfitting and interpretation complexity
- Smaller trees: higher bias but have lower variance $\Rightarrow$ easier to interprete.

Strategy is to grow a very large tree T_0 and **pruning** it in subtrees:

- Sequence of trees indexed by a non negative tuning parameter α
- For each α there corresponds a subtree $T \subset T_0$ such that the following is minimized:

$$\sum_{m=1}^{|T|} \sum_{x_1 \in R_m} (y_i - \hat{y}_{Rm})^2 + \alpha |T|$$

with Rm is the rectangle links to the m^{th} terminal node, α controls the tradeoff between subtree complexity and its fit to the training data, and $|T|$ is the number of terminal node in tree T

Regression trees: when to stop??

$$\sum_{m=1}^{|T|} \sum_{x_1 \in R_m} (y_i - \hat{y}_{Rm})^2 + \alpha |T| \tag{1}$$

with Rm is the rectangle links to the m^{th} terminal node, α controls the tradeoff between subtree complexity and its fit to the training data, and $|T|$ is the number of terminal node in tree T

- $\alpha = 0$: subtree = tree, why? 1 = training error
- α min. 1 with small subtrees and branches get prunned in a nested and predictable fashion
- $\approx$ to Lasso

Advantages of growing trees: interpretable and handle qualitative variables

Linear model or trees?

- Linear model:
  $f(X) = \beta_0 + \sum X_j \beta_j$
- Trees:
  $f(X) = \beta_0 + \sum c_m . 1_{(X \in R_m)}$

Assess relative performance estimating test error by CV

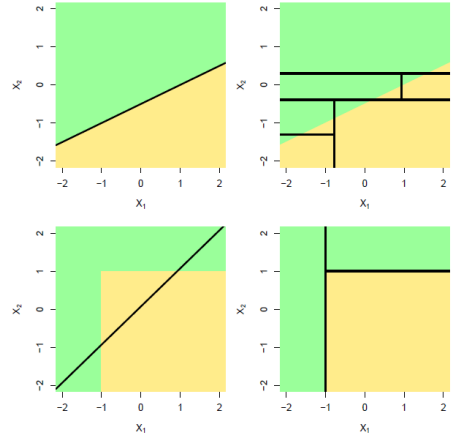


FIGURE 8.7. Top Row: *A two-dimensional classification example in which the true decision boundary is linear, and is indicated by the shaded regions. A classical approach that assumes a linear boundary (left) will outperform a decision tree that performs splits parallel to the axes (right).* Bottom Row: *Here the true decision boundary is non-linear. Here a linear model is unable to capture the true decision boundary (left), whereas a decision tree is successful (right).*

From growing trees to planting forests

Growing a single tree has some drawbacks:

- Not as accurate as other methods
- Not as robust (high variance): if we split the training data at random and fit tree to both halves, the results that we get could be quite different

Bagging intuition: the average of random variables has lower variance than any single of them.

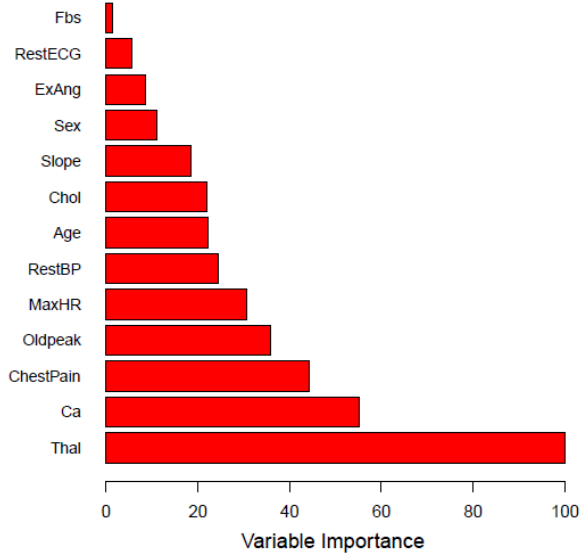
- It takes multiple bootstrap samples from the training set, build a separate prediction model on each of them and average them: $\hat{f_{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\hat{f}^{*b}(x)$
- Assess performance using the **out-of-bag observations** ($\approx$ 1/3 of the sample) to predict y for the i^th observation using each of the trees in which it was OOB, yields to $B/3$ predictions for the i^th obs.
- Average each of them to compute the the out-of-bag error (i.e., test MSE).

Bagging

Drawbacks:

- More complex and less interpretable than trees
  solution: record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.
- $\hat{f}^b$ might be highly correlated in practice $\Rightarrow$ use random forests to decorrelates the trees

Variables importance plot



Random forest

Random Forest: randomly restrict which splits of the tree are allowed at each node. At each split, only a random subsample of covariates m is considered. Then, one covariate is choose among m. A fresh sample is drawn at each split.

- RF is not even allowed to consider a majority of the available predictors
- Why? Suppose 1 very strong p and a large number of not strong p. Then in the collection of bagged- trees, most of them will use the strong at the top $\Rightarrow$ all of the bagged trees will look quite similar

$\Rightarrow \hat{f}^b$ will be less correlated which reduces variance

- Usually, we pick $m \approx \sqrt{p}$ to avoid strong covariates to be picked at the top of each tree.
- On average $\frac{p-m}{m}$ of splits not consider a covariate
- Main difference with bagging: choice of subset m
- Note that RF=bagging if $p = m$
- As with bagging,it will not overfit if we increase $B \Rightarrow$ use large B

Boosting

While **bagging** involves creating *multiple copies* of the original training data, **boosting** *grows each tree squentially* using information from previously grown trees. Each tree is fit on a modified version of the original data set.

1. Fit a decision tree to the residual from the model. It fits a tree using the current residuals and not the outcome as dependent variable

2. Add this new decision tree into the fitted function and update the residuals

3. Grows a new tree using the updated residual from the previous fitted function.

Small trees are grown successively such that it improves slowly the fitted function in poor performance area.

Boosting for regression trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.

2. For $b = 1, 2, \ldots, B$, repeat:

   (a) Fit a tree $\hat{f}^b$ with d splits ($d+1$ terminal nodes) to the training data (X, r).

   (b) Update $\hat{f}$ by adding in a shrunken version of the new tree:

   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x). \tag{8.10}$$

   (c) Update the residuals,

   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i). \tag{8.11}$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x). \tag{8.12}$$

Parameters to consider

Three tuning parameters:

- Number of trees B: overfit if too large. Use cross-validation to pick B
- Shrinkage parameter λ: small positive number which controls the boosting learning rate. The smaller it is, the larger B should be
- Number of splits d in each tree: controls the complexity. Works well with $d = 1$

References

Alexandre Belloni, Victor Chernozhukov, and Christian Hansen. High-dimensional methods and inference on structural and treatment effects. *Journal of Economic Perspectives*, 28(2):29–50, 2014.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Sendhil Mullainathan and Jann Spiess. Machine learning: an applied econometric approach. *Journal of Economic Perspectives*, 31(2):87–106, 2017.