Machine Learning for Economists

Prediction

Anthony Strittmatter

Outline

1. General Machine Learning Framework
2. Regularised Regressions
3. Trees and Random Forests
4. Other Machine Learning Methods (Backup)

Literature

- Mullainathan and Spiess (2017): ”Machine Learning: An Applied Econometric Approach”, Journal of Economic Perspectives, 31 (2), pp. 87-106, download.
- Hastie, Tibshirani, and Friedman (2009): ”Elements of Statistical Learning”, 2nd ed., Springer, download.
- James, Witten, Hastie, and Tibshirani (2013): ”An Introduction to Statistical Learning”, Springer, download.

General Machine Learning Framework

Supervised Machine Learning in One Expression

Objective function:

$$\min \sum \underbrace{L(Y,f(X))}_{\text{loss function}} \text{ over } \underbrace{f \in F}_{\text{function class}} \text{ s.t. } \underbrace{R(f) \leq c}_{\text{complexity restriction}}$$

- Loss function $L(\cdot)$: quadratic, absolute, etc.
- Function class F: linear (e.g., LASSO), non-linear (e.g., Logit-LASSO), non-parametric (e.g., tree, neural net)
- Complexity restriction $R(f) \leq c$: depth of tree, number of hidden layers, LASSO penalty, etc.
- Complexity level c: tuning parameter for max. complexity

Source: Francis Diebold's "No Hesitations" blog

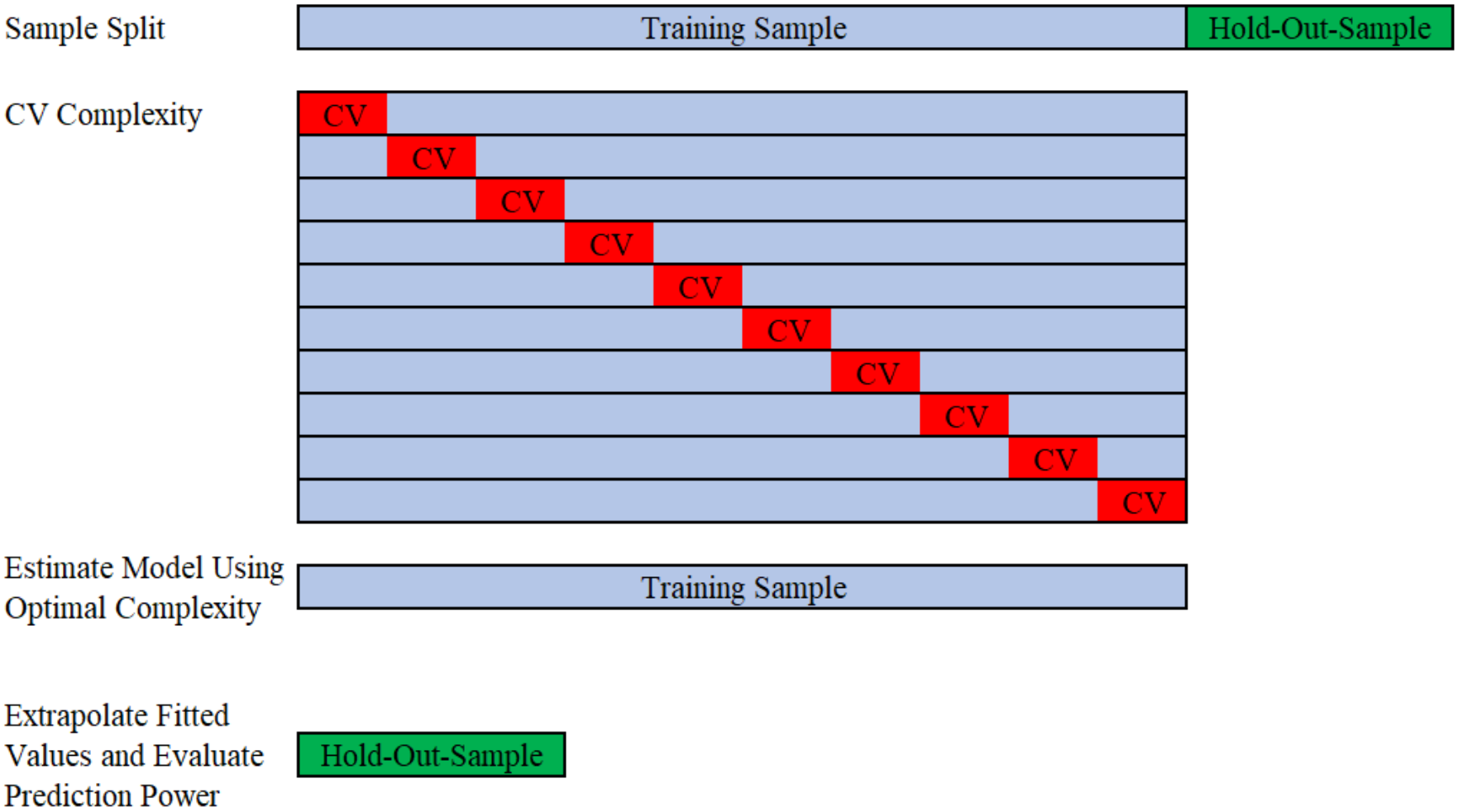
General ML Procedure

1. Select a ML method (e.g., LASSO).
2. Draw randomly a hold-out-sample from the data.
3. Estimate the machine learning model using different complexity levels c.
4. Select the optimal complexity level c^*.
5. Predict $\widehat{Y}$ using c^* and extrapolate the fitted values to the retarded hold-out-sample.
6. Evaluate the prediction power of the ML method in the hold-out-sample.

Selection of the Optimal Complexity

- Van der Vaart, Dudoit, and van der Laan (2006) show that cross-validation (CV) tuning approximates the optimal complexity.
- Concrete implementation of CV:
  - Number of CV-folds: 5-20?
  - Performance measure: MSE, Variance, Gini-index?
  - Ad-hoc extensions: ”one standard-error rule” for tuning the LASSO.
- Alternatives to CV:
  - Some alternatives (e.g., AIC, BIC) rely on assumptions about the sampling process.
  - Rare theoretical guidance applies only to specific methods (see, e.g., Belloni, Chen, Chernozhukov, and Hansen, 2012, for the LASSO).

Cross-Validation (CV) Algorithm



The Firewall Principle

Why do we use the hold-out-sample to evaluate the prediction power?

- If we try many tuning parameter values, we may end up overfitting even in cross-validation samples.
- The cross-validation samples are smaller than the hold-out-sample, which may lead to underestimation of the prediction power.
- The cross-validation performance is an aggregation over multiple different prediction functions, which differs from the single prediction function we finally estimate.

Prediction Performance in Housing Price Example

	In-Sample		Hold-Out-Sample	
	MSE	R^2	MSE	R^2
OLS	0.589	47.30%	0.674	41.70%
LASSO	0.603	46.00%	0.656	43.30%
Shallow Tree	0.675	39.60%	0.758	34.50%

Source: Mullainathan and Spiess (2017)

Honest Inference

- Using the same data to train and estimate the ML model might lead to overfitting.
- For example, there is a risk that extreme outliers are grouped in the same leave of a tree which may increase the variance.
- To avoid this, we split the data in training and estimation samples.
- The training sample is used to select the ML model, e.g., with a CV procedure.
- The selected ML model is used for predicting Y in the estimation sample.
- The fitted values $\widehat{Y}$ are extrapolated to the hold-out-sample for an assessment of the prediction power.
- **Potential gain:** Better coverage of confidence intervals.
- **Price to pay:** Smaller samples, less precise predictions.

Honest Algorithm

Cross-Fitting

- To improve the finite sample performance Chernozhukov et al. (2017) propose different cross-fitting procedures.
- For example, we could partition the data in two samples A and B (and a hold-out-sample).
- Train the ML model in each sample separately.
- Fit $\widehat{Y}_A$ in sample A using the model trained in sample B (and extrapolate fitted values to hold-out-sample).
- Fit $\widehat{Y}_B$ in sample B using the model trained in sample A (and extrapolate fitted values to hold-out-sample).
- Finally, $\widehat{Y} = \frac{1}{2}(\widehat{Y}_A + \widehat{Y}_B)$.
- Potentially more than two cross-fitting samples can be used.

Data transformations

- The way how we encode and transform our data can affect the performance of the ML method
- Transformations of outcome variable Y:
  - ML methods do not guide us regarding the appropriate transformation (e.g., levels, logarithm).
- Transformations of covariates X:
  - Different measures for same variable: We can include many different measures of the same variable (e.g., level, logarithm, squared, cubic) and the ML algorithm will disregard them if not needed.
  - Same variable on different aggregation levels: ML methods select an appropriate aggregation level.
  - Standardisation: Usually, we standardise all X variables by the means and standard deviations in the training sample.

Regularised Regressions

Regularised Regressions

$$\min_{\beta} \left\{ \sum_{i=1}^{N} \left(Y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j \right)^2 + \lambda \sum_{j=1}^{p} p(\beta_j) \right\}$$

where $\lambda \geq 0$ is the tuning parameter and the number of covariates p can be high-dimensional ($p \gg N$).

Choice of penalty function $p(\cdot)$:

- Ridge: $p(\beta_j) = \beta_j^2$
- LASSO: $p(\beta_j) = |\beta_j|$ (Least Absolute Shrinkage and Selection Operator)
- Elastic Net: $p(\beta_j) = \alpha|\beta_j| + (1-\alpha)\beta_j^2$
- Best Subset Selection: $p(\beta_j) = 1\{\beta_j \neq 0\}$

$\rightarrow$ Note that coefficient size depends on the scaling of X_j. It is best practice to standardise X_j.

Summation Notation

- OLS residual sum of squares (RSS):

$$RSS = \sum_{i=1}^{N} (Y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2$$

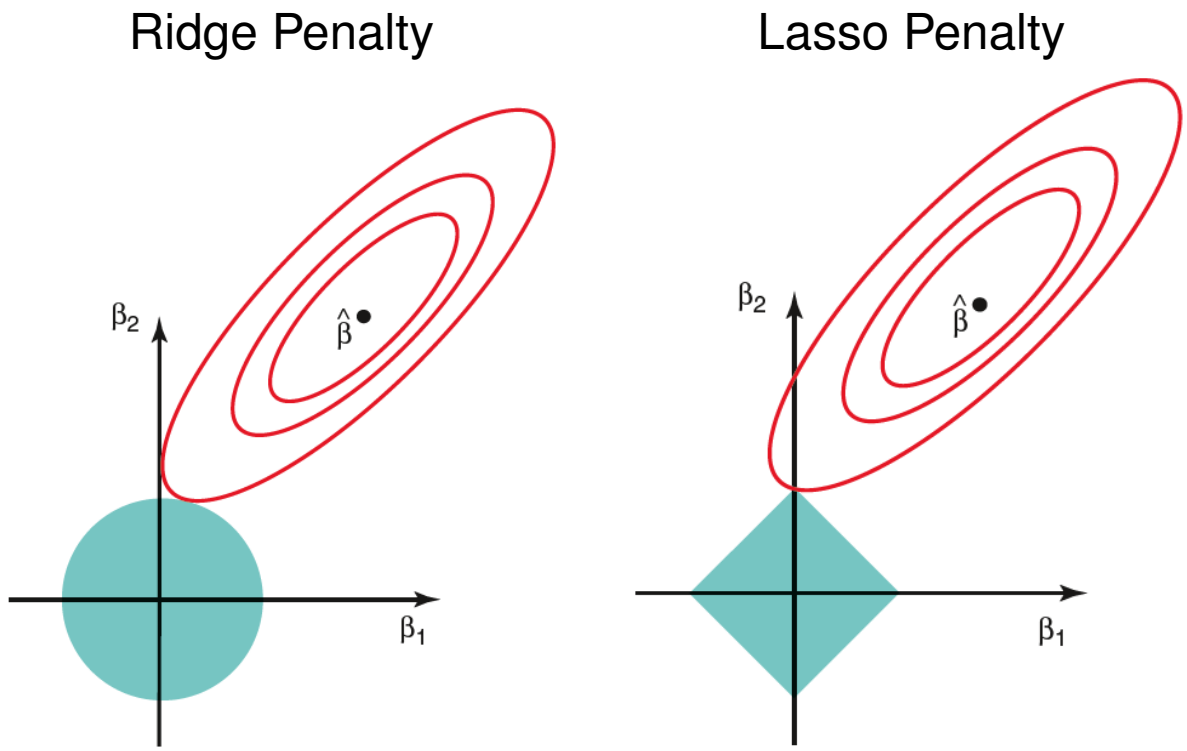
- Penalized regression:
  - Lagrangian operator

$$\min_{\beta} \{RSS + \lambda \sum_{j=1}^{p} p(\beta_j)\}$$

  - Constrained regression

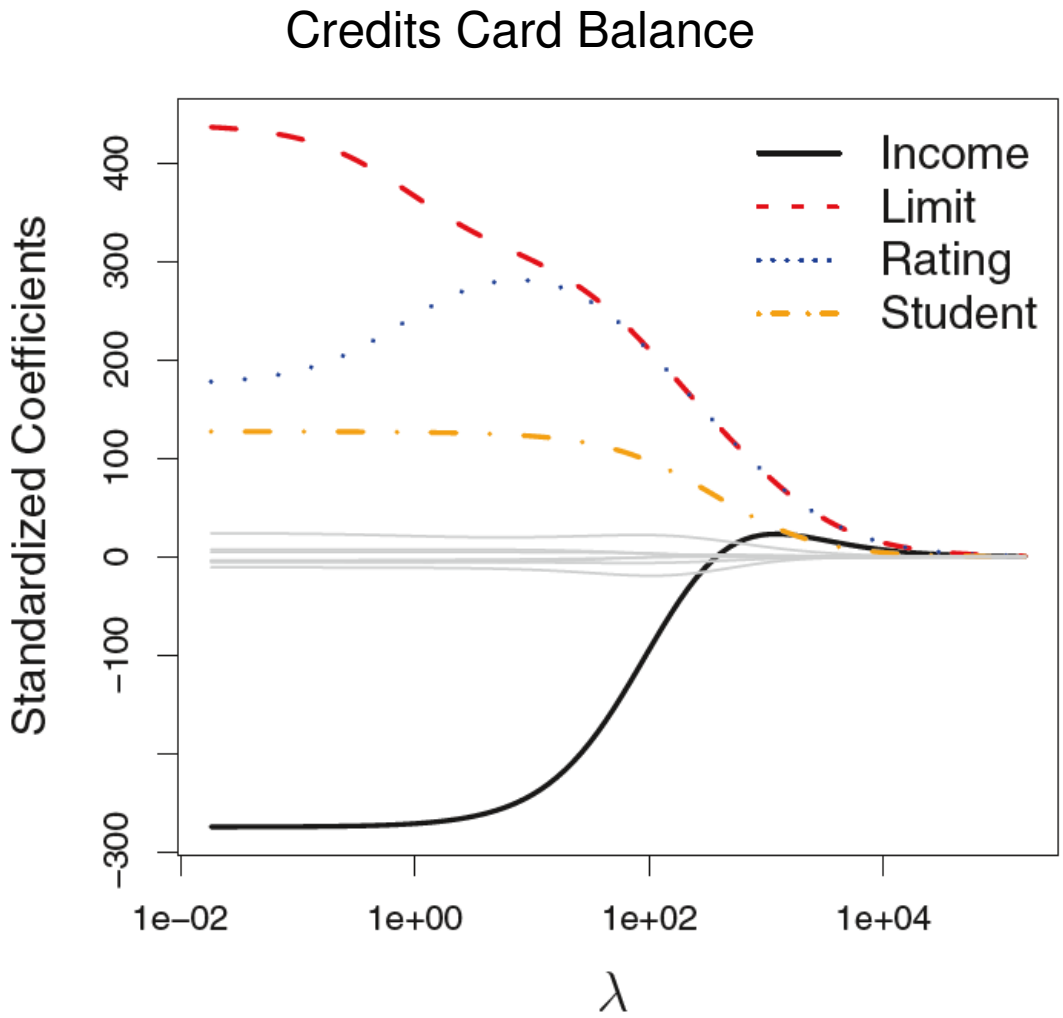
$$\min_{\beta} \{RSS\} \text{ s.t. } \sum_{j=1}^{p} p(\beta_j) \leq c$$

Constraint Regions



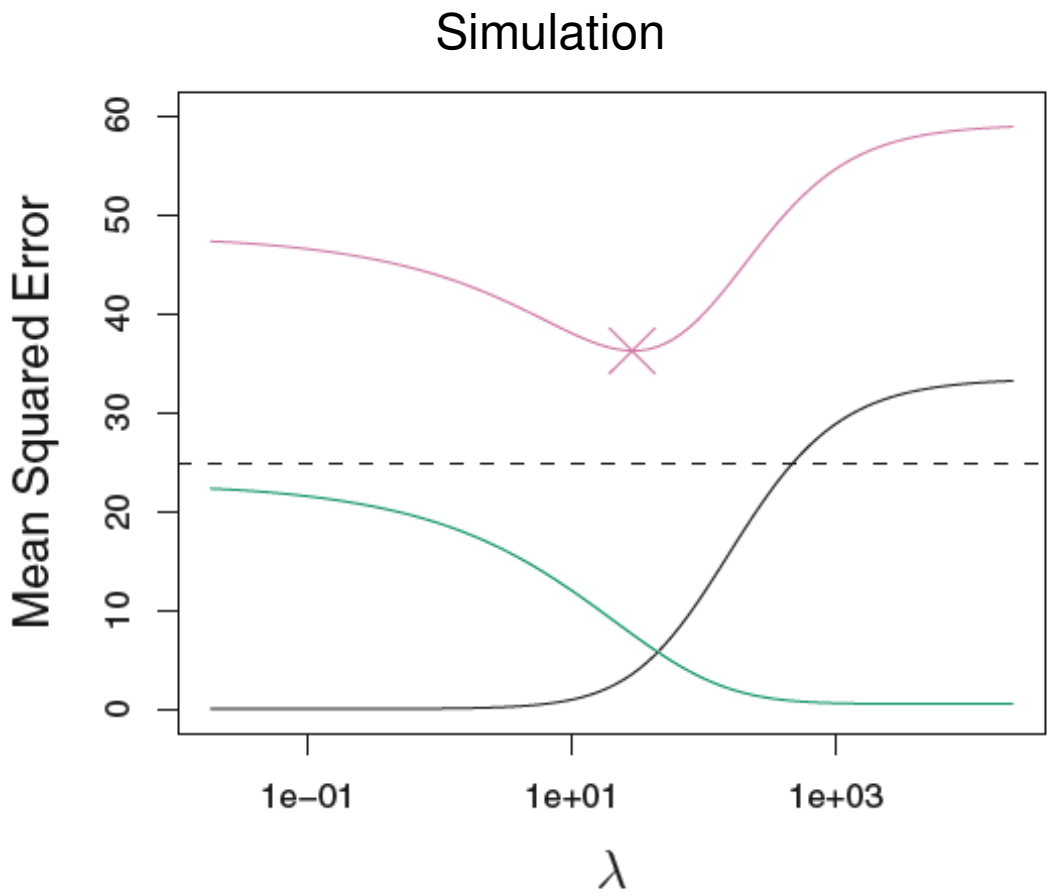
Source: James, Witten, Hastie, Tibshirani (2013)

Ridge Coefficients



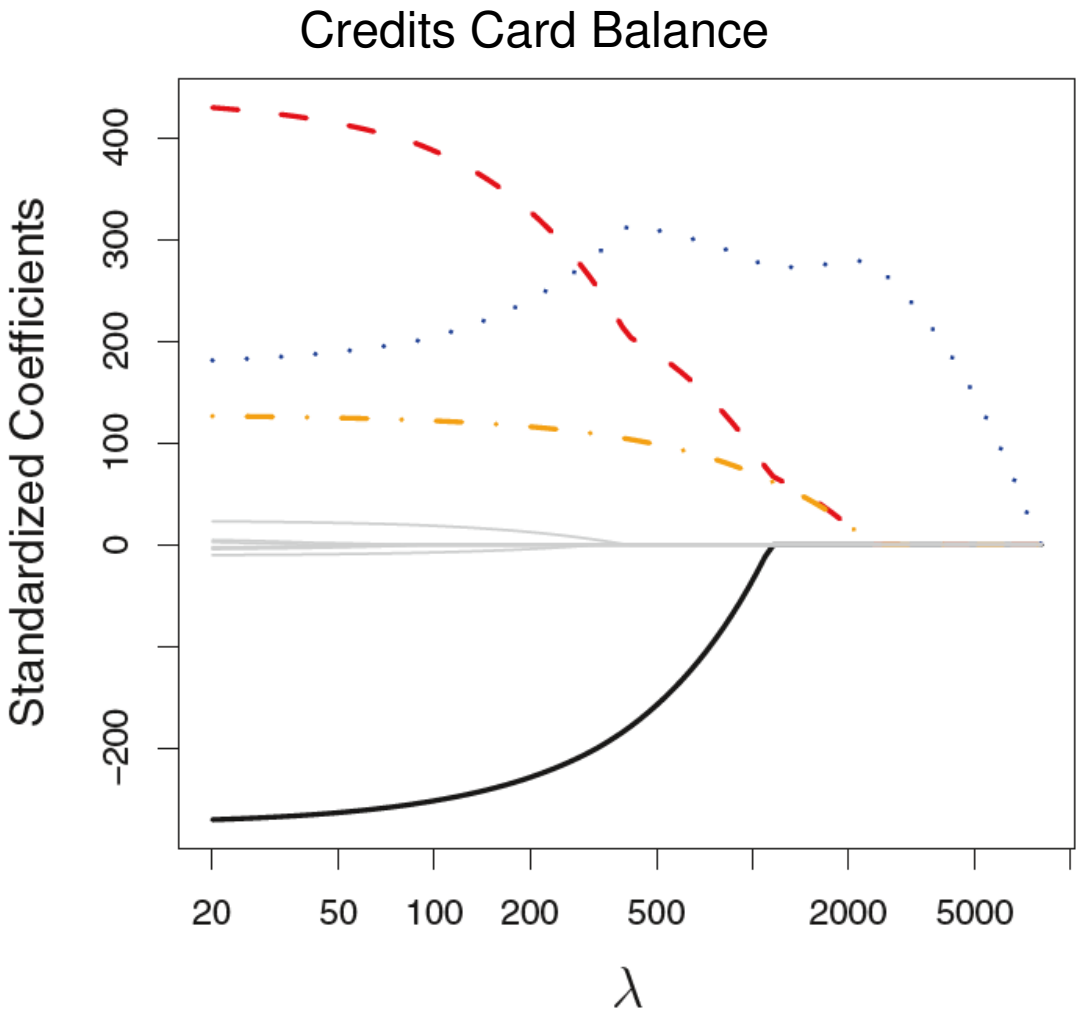
Source: James, Witten, Hastie, Tibshirani (2013)

Ridge: Variance-Bias Trade-Off



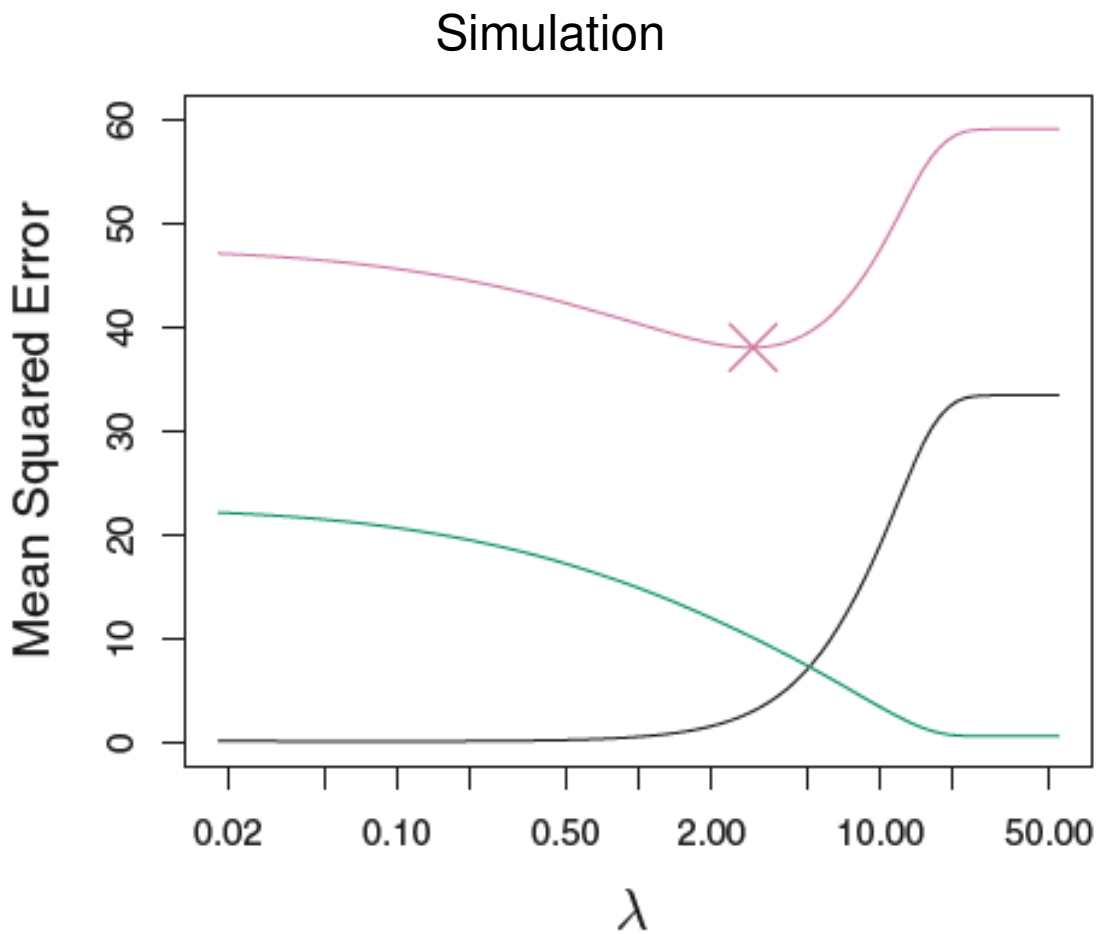
Source: James, Witten, Hastie, Tibshirani (2013)

LASSO Coefficients



Source: James, Witten, Hastie, Tibshirani (2013)

LASSO: Variance-Bias Trade-Off



Source: James, Witten, Hastie, Tibshirani (2013)

Post-LASSO

- Coefficients of LASSO $\widehat{\beta}_j$ are inconsistent when $\lambda_N > 0$ (asymptotically)
- Post-LASSO:

$$\min_{\alpha} \sum_{i=1}^{N} \left(Y_i - \alpha_0 - \sum_{j=1}^{p} 1\{\widehat{\beta}_j \neq 0\} X_{ij} \alpha_j \right)^2$$

- Coefficients of Post-LASSO are consistent.
- Model selection consistency depends on the first-step LASSO estimates.
- Alternatives:
  - Adaptive LASSO: $\lambda_j^* = \lambda / |\widehat{\beta}_j|^{\gamma}$ (Zou, 2006).
  - Conservative LASSO: $\lambda_j^* = \lambda / \max(|\widehat{\beta}_j|, \lambda)$ (Caner and Kock, 2018).

Simple Example

- Consider $X = I$ with dimension $p = N$.
- OLS model

$$\min \sum_{j=1}^{p} (Y_j - \beta_j)^2,$$

such that the estimated OLS coefficients are $\widehat{\beta}_j = Y_j$.

- Ridge model

$$\min \sum_{j=1}^{p} (Y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} \beta^2,$$

such that the estimated Ridge coefficients are $\widehat{\beta}_j^R = \widehat{\beta}_j/(1+\lambda)$.

Simple Example (cont.)

- LASSO model

$$\min \sum_{j=1}^{p} (Y_j - \beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta|,$$

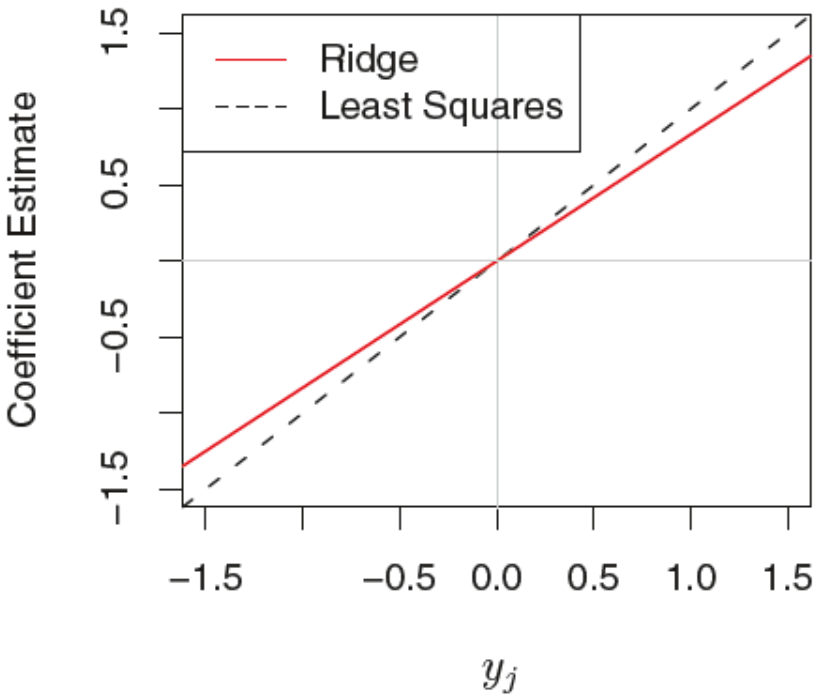
such that the estimated LASSO coefficients are

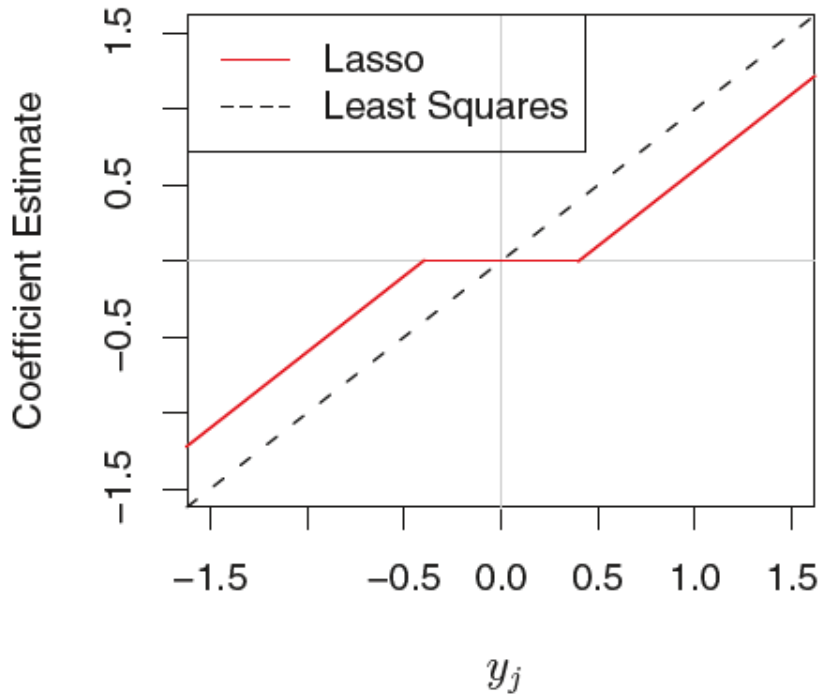
$$\widehat{\beta}_j^L = \begin{cases} \widehat{\beta}_j - \lambda/2 & \text{if } \widehat{\beta}_j > \lambda/2, \\ \widehat{\beta}_j + \lambda/2 & \text{if } \widehat{\beta}_j < -\lambda/2, \\ 0 & \text{if } |\widehat{\beta}_j| < \lambda/2, \end{cases}$$

which corresponds to the soft-thresholding operator

$$\widehat{\beta}_j^L = sign(\widehat{\beta}_j)(|\widehat{\beta}_j| - \lambda/2)_+$$

Simple Example (cont.)





Source: James, Witten, Hastie, Tibshirani (2013)

Oracle Property

Setup:

- Structural model: $Y_i = \beta_0 + \sum_{j=1}^{K} X_{ij}\beta_j + U_i$
- The vector X_i has $p > K$ dimensions
- Estimated LASSO model: $\widehat{Y}_i = \widehat{\beta}_0 + \sum_{j=1}^{p} X_{ij}\widehat{\beta}_j + \lambda \sum_{j=1}^{p} |\widehat{\beta}_j|$

Oracle property:

- Model selection consistency: When $N \to \infty$,

$$Pr(\widehat{\beta}_j = 0) \xrightarrow{p} 1$$

  for all $\widehat{\beta}_j \in \{\widehat{\beta}_{K+1}, \ldots, \widehat{\beta}_p\}$ (irrelevant covariates)

- Coefficient estimation consistency: When $N \to \infty$,

$$Pr(|\widehat{\beta}_j - \beta_j| > \varepsilon) \xrightarrow{p} 0$$

  for any $\varepsilon > 0$ and all $\widehat{\beta}_j \in \{\widehat{\beta}_0, \ldots, \widehat{\beta}_K\}$ (relevant covariates)

Minimum assumptions required:

- Sparsity ($K \ll N$) and relevant covariates have to be roughly orthogonal to the irrelevant ones (”irrepresentability condition”)

Matrix Notation

- **Ridge:**

$$\min_{\beta} \left\{ (Y - X\beta)'(Y - X\beta) + \lambda \|\beta\|_2^2 \right\}$$

with $\|\beta\|_2^2 = \beta'\beta = \sum_{j=1}^{p} \beta_j^2$ (squared l_2-norm), $\widehat{\beta} = (X'X + \lambda I)^{-1} X'Y$, and I being the identity matrix

- **Lasso:**

$$\min_{\beta} \left\{ (Y - X\beta)'(Y - X\beta) + \lambda \|\beta\|_1 \right\}$$

with $\|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$ (l_1-norm)

Coordinate Descent Algorithm for LASSO

$$\min_{\beta} \left\{ \frac{1}{2N} \sum_{i=1}^{N} (Y_i - \beta_0 - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda_s \sum_{j=1}^{p} |\beta_j| \right\}$$

(1) Specify a grid of $s = 1, ...S$ tuning parameters $\lambda_s \in \{\lambda_1, \lambda_2, ..., \lambda_S\}$

(2) Take residuals $Y_i^* = Y_i - \frac{1}{N}\sum_{i=1}^{N} Y_i$ and initialise $\beta_j = 0$

(3) Circulate repeatedly over all $j = 1, ..., p$ until convergence:

  (a) Compute the partial residuals by $r_{ij} = Y_i^* - \sum_{k \neq j} X_{ik}\beta_k$

  (b) Calculate the simple univariate OLS coefficient $\tilde{\beta}_j = \frac{1}{N}\sum_{i=1}^{N} X_{ij} r_{ij}$

  (c) Update β_j with the soft-thresholding operator:

$$\beta_j = sign(\tilde{\beta}_j)(|\tilde{\beta}_j| - \lambda_s)_+$$

(4) Repeat (3) for $s = 1, ..., S$

Note: Standardisation of X is required

Logit-Lasso (Backup)

- Logistic log-likelihood function:

$$L = \sum_{i=1}^{N} (Y_i X_i \beta - log(1 + exp(X_i \beta)))$$

- Logit-Lasso:

$$\min_{\beta} \left\{ -L + \lambda \sum_{j=1}^{p} |\beta| \right\}$$

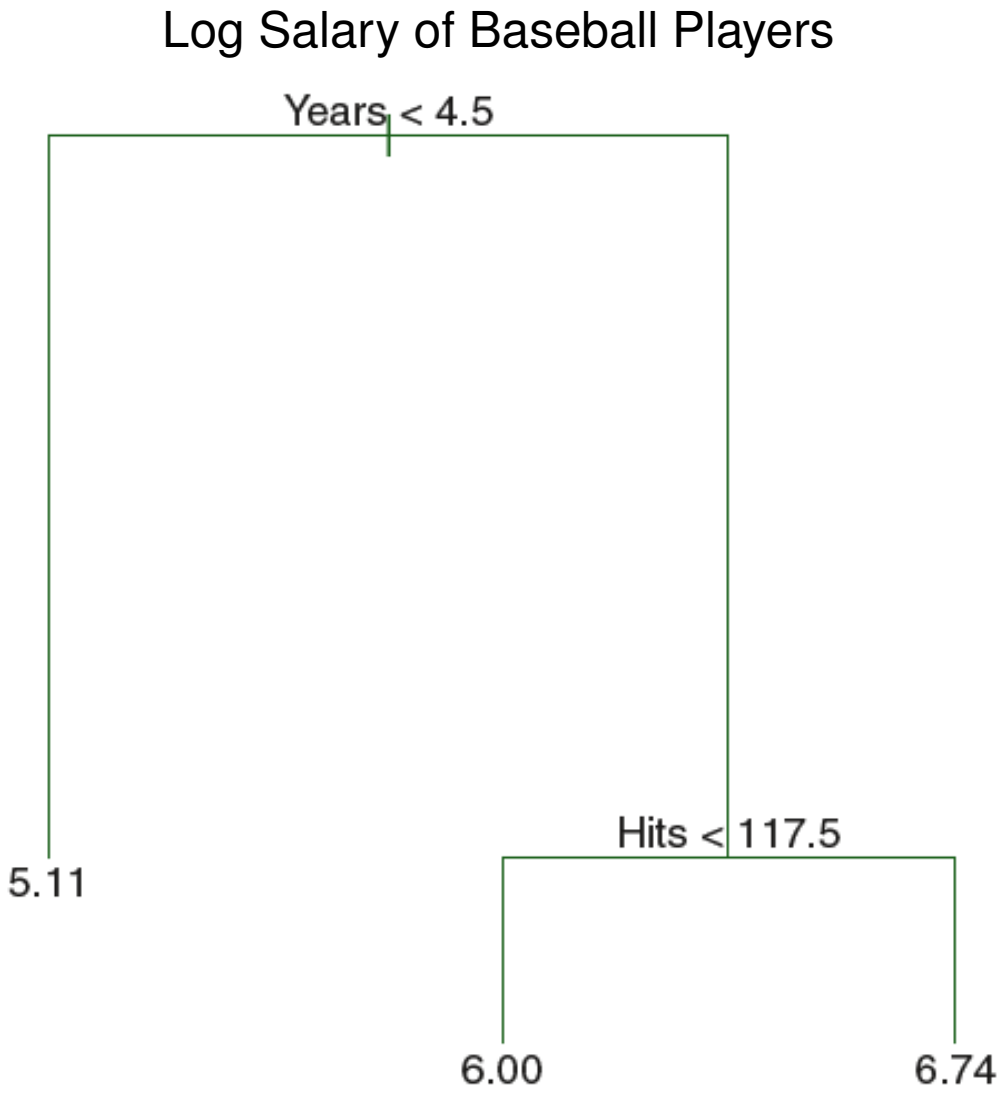
- Log likelihood function can be approximated by repeated application of weighted LASSO (proximal Newton iterations, Lee, Sun, and Saunders, 2014)
- Can be estimated with a coordinate descend algorithm with weighted soft-thresholding
- If $p \gg N$ we cannot let λ go down to zero
- See Hastie, Tibshirani and Wainwright (2016) for a comprehensive summary of logit-Lasso

Trees and Random Forests

Tree

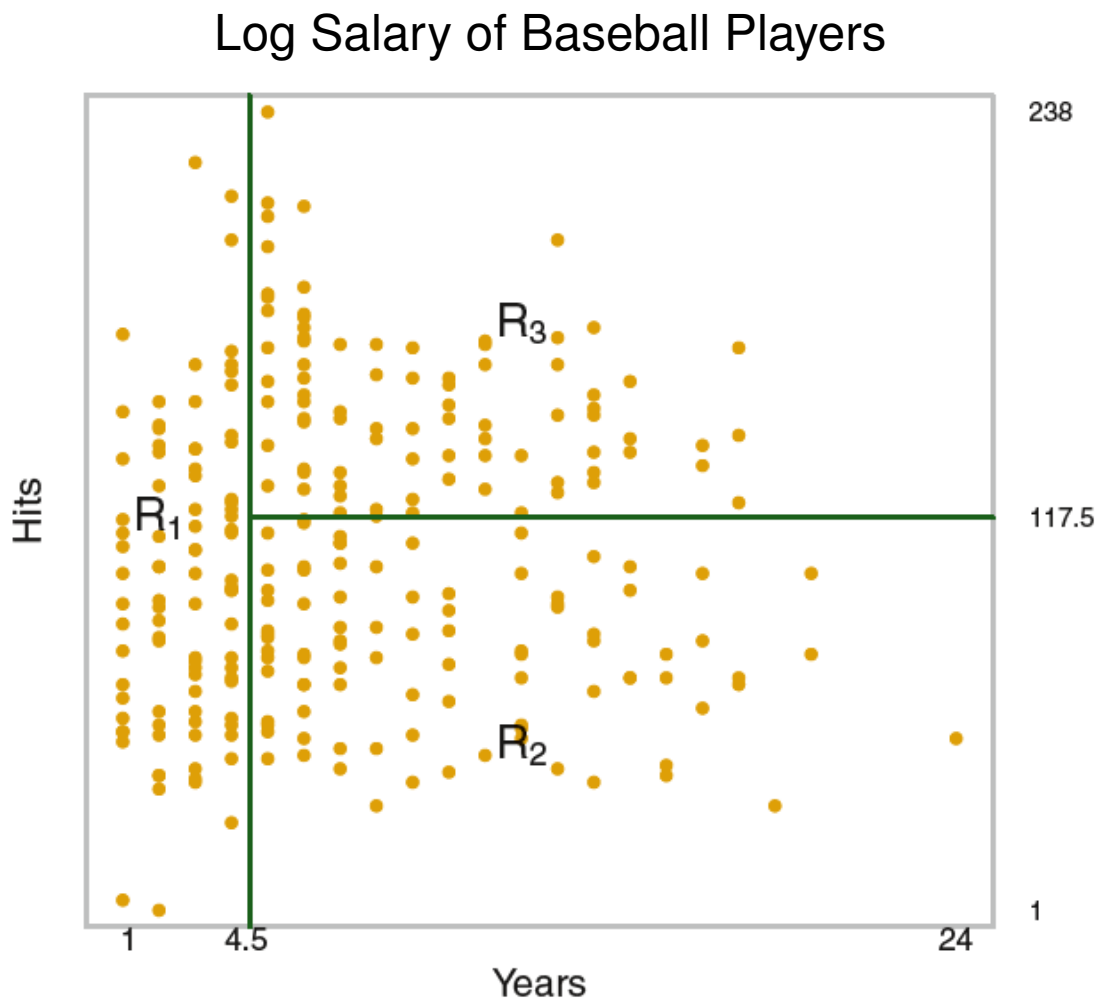
- Trees partition the sample into mutually exclusive groups l_j, which are called leaves.
- Let $\pi = \{l_1, ..., l_{\#(\pi)}\}$ be the terminal leaves of a specific tree or sample partition.
- Let $l_j \equiv l_j(x, \pi)$ be the respective leaf (for $j = 1, ..., \#(\pi)$).
- The leaf $l_j(x, \pi)$ of tree π is a function of the covariates X such that $x \in l_j$.
- Let $\#(\pi)$ be the number of terminal leaves in tree π.

Example: Shallow Tree



Source: James, Witten, Hastie, Tibshirani (2013)

Example: Shallow Tree (cont.)



Source: James, Witten, Hastie, Tibshirani (2013)

Recursive Partitioning

- Trees select the leaves with a top-down, greedy algorithm, which is called *recursive partitioning*.
- *Top-down* because we start with a root (tree without leaves) and successively add splits.
- *Greedy* because at each step of the tree building we add the split that improves the prediction power best (instead of looking ahead).

Tree Building Algorithm

(1) Start with the entire sample (root).

(2) For each predictor X_j and cut-point s define the pair of half-planes

$$l_1^{(j,s)} = \{X|X_j < s\} \text{ and } l_2^{(j,s)} = \{X|X_j \geq s\}.$$

- Calculate the mean outcomes $\bar{Y}_1$ and $\bar{Y}_2$ in each half-plane, respectively.
- Seek the covariate X_{j1}^* and the cut-point s_1^* that minimise

$$\sum_{i:X_i \in l_1^{(j,s)}} (Y_i - \bar{Y}_1)^2 + \sum_{i:X_i \in l_2^{(j,s)}} (Y_i - \bar{Y}_2)^2.$$

Tree Building Algorithm (cont.)

(2) For each predictor X_j and cut-point s define the triple of half-planes

$$l_1^{(j,s)} = \{X | X_{j1}^* < s_1^*, X_j < s\},\ l_2^{(j,s)} = \{X | X_{j1}^* < s_1^*, X_j \geq s\}, \text{ and}$$
$$l_3^{(j,s)} = \{X | X_{j1}^* \geq s_1^*\}$$

and

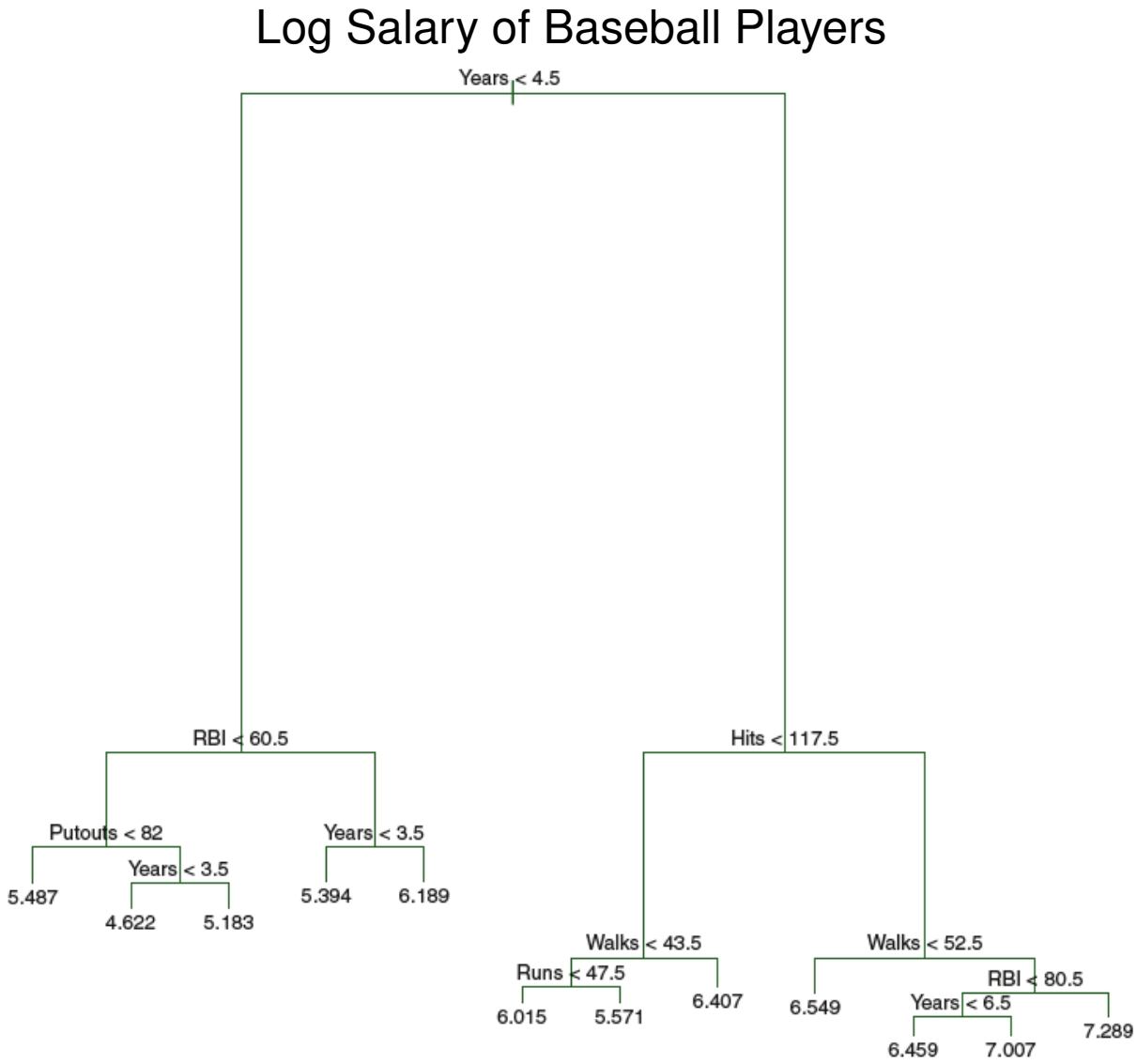
$$l_1^{(j,s)} = \{X | X_{j1}^* \geq s_1^*, X_j < s\},\ l_2^{(j,s)} = \{X | X_{j1}^* \geq s_1^*, X_j \geq s\}, \text{ and}$$
$$l_3^{(j,s)} = \{X | X_{j1}^* < s_1^*\}.$$

- Calculate the mean outcomes $\bar{Y}_1$, $\bar{Y}_2$, and $\bar{Y}_3$ in each half-plane, respectively.
- Seek the covariate X_{j2}^* and the cut-point s_2^* that minimise

$$\sum_{i:X_i \in l_1^{(j,s)}} (Y_i - \bar{Y}_1)^2 + \sum_{i:X_i \in l_2^{(j,s)}} (Y_i - \bar{Y}_2)^2 + \sum_{i:X_i \in l_3^{(j,s)}} (Y_i - \bar{Y}_3)^2.$$

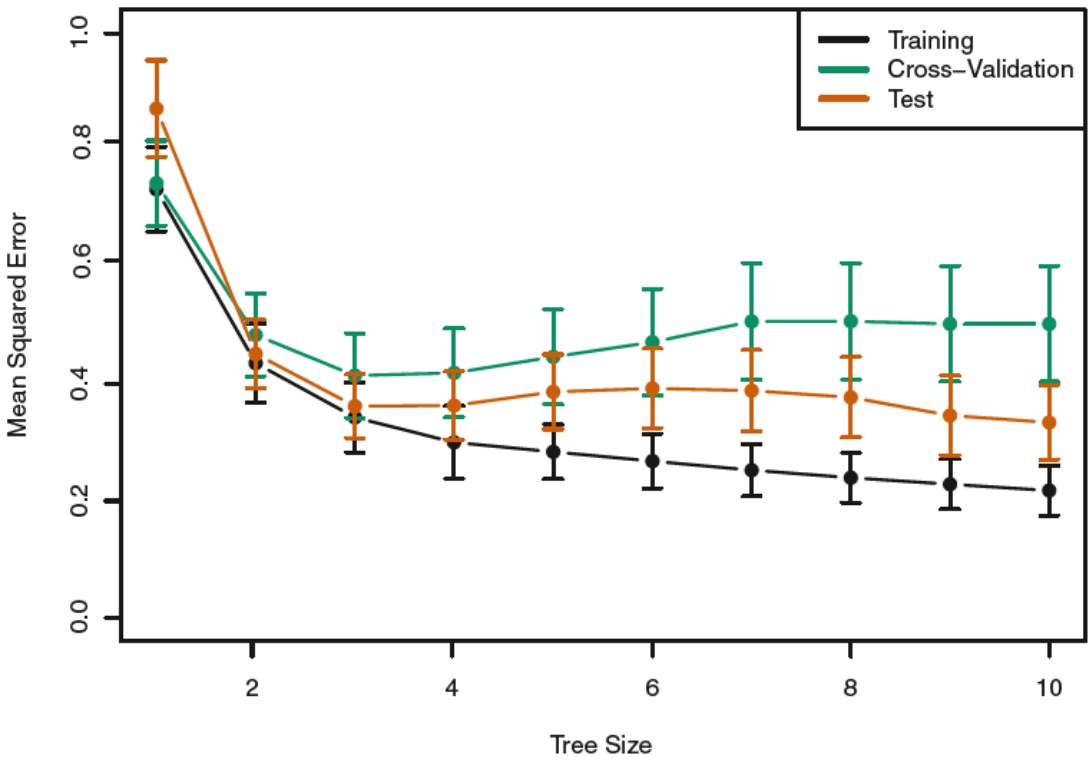
(3) Continue until some stopping rule is reached (e.g., max. tree size, min. terminal leave size, min. MSE gain) .

”Deep” Tree



Source: James, Witten, Hastie, Tibshirani (2013)

Selecting Optimal Tree Size



→ Pruning the complexity of trees can improve out-of-sample prediction power.

→ Select optimal tree size π with cross-validation.

Source: James, Witten, Hastie, Tibshirani (2013)

Complexity Pruning

(A) Use recursive partitioning to grow the deep tree π_0 in the training data.

(B) For each value of α a subtree $\pi \subseteq \pi_0$ minimises

$$\sum_{j=1}^{\#(\pi)} \sum_{i:X_i \in l_j} (Y_i - \bar{Y}_j)^2 + \alpha \#(\pi). \tag{1}$$

Obtain a sequence of best subtrees.

(C) Use cross-validation to choose α. Partition the sample in k folds. For each fold:

- (a) Repeat steps (A) and (B) in the kth-fold.
- (b) Evaluate the MSE using equation (1).
- (c) Average the MSE across the k folds for each value of α and select the α that minimises the average MSE.

(D) Return to the subtree from (B) with the selected value of α.

Prediction

- For the selected tree π^* use the estimation sample to predict

$$\widehat{Y}_i = \sum_{j=1}^{\#(\pi^*)} \frac{1}{\sum_{i=1}^{N} 1\{X_i \in l_j(x, \pi^*)\}} \sum_{i=1}^{N} 1\{X_i \in l_j(x, \pi^*)\} \cdot Y_i.$$

- This is equivalent to the linear regression

$$\min_{\beta} \sum_{i=1}^{N} \left(Y_i - \sum_{j=1}^{\#(\pi^*)} 1\{X_i \in l_j(x, \pi^*)\} \beta_j \right)^2.$$

Advantages and Disadvantages of Trees

Advantages:

- Shallow trees are very easy to understand.
- Shallow trees can be displayed graphically in a nice way.
- Trees automatically handle interactions between covariates.
- It is not necessary to transform covariates as long as they have an order.

Disadvantages:

- Often trees are unstable and have a high variance.

Random Forests

- Build G deep trees π_g on different subsets of the data (subsampling or bootstrapping) and covariates.

  $\rightarrow$ decorrelated trees

- These trees are overfitted in the sample and will have a high out-of-sample variance.
- To overcome this problem, we aggregate the trees

$$\widehat{Y}_i^{RF} = \frac{1}{G} \sum_{g=1}^{G} \widehat{Y}_i^{\pi_g}.$$

- This procedure is often called bootstrap-aggregation (”bagging”).
- We loose interoperability but gain prediction power compared to (shallow) trees.
- Tuning parameters: Forest size, subsample selection, covariate selection, tree size, honest inference, etc.

Random Forests: Weighted Representation

- Tree weights:

$$\alpha_{ig}(x) = \frac{1\{X_i \in l_j(x, \pi_g)\}}{\sum_{i=1}^{N} 1\{X_i \in l_j(x, \pi_g)\}}$$

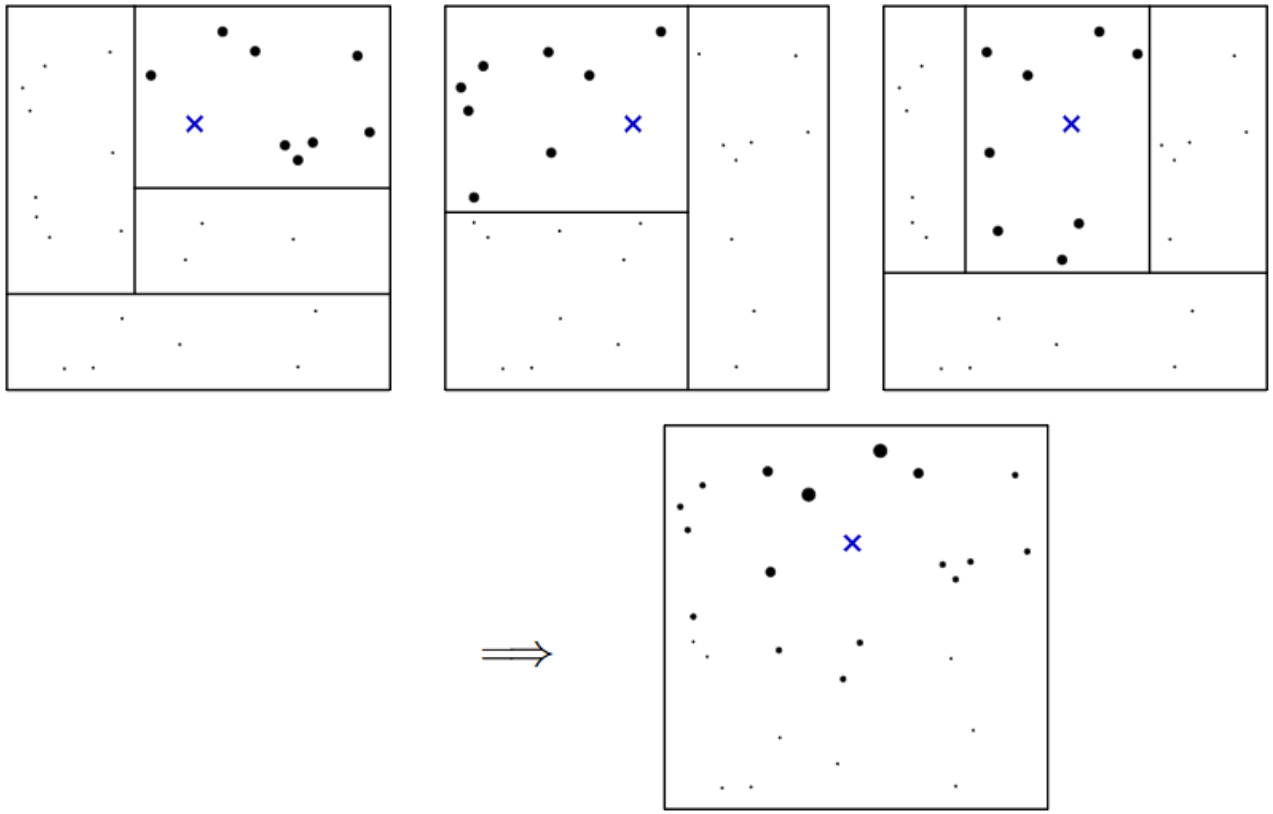
- Forest weights:

$$\alpha_i(x) = \frac{1}{G} \sum_{i=1}^{G} \alpha_{ig}(x)$$

- Predicted outcome:

$$\widehat{\mu}(x) = \sum_{i=1}^{N} \alpha_i(x) Y_i \text{ and } \widehat{Y}_i = \widehat{\mu}(X_i)$$

Random Forest: Weighted Representation (cont.)

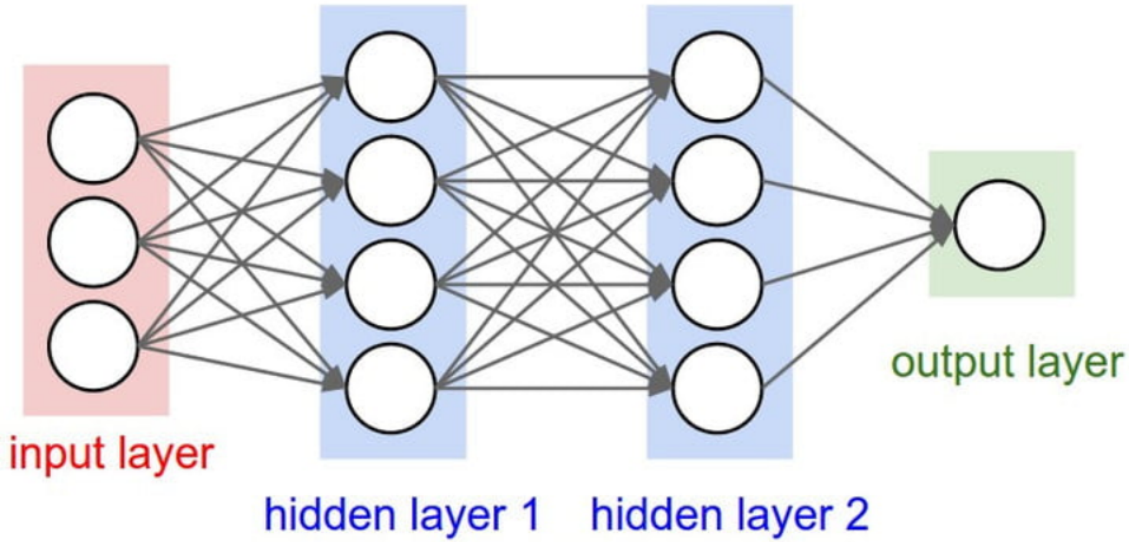


Source: Athey, Tibshirani, Wager (2018)

Other Machine Learning Methods (Backup)

Neural Networks

- Input layer, output layer, hidden layer, neurons



- Neurons network function: $f(X) = F(\sum_j w_j g(x_j))$
- Types of neural nets: Feedforward, Recurrent, Autoencoder, etc.
- Econometric references: Farell, Liang, and Misra (2018), Hartford, Lewis, Leyton-Brown, and Taddy (2016)

Other (Supervised) Machine Learning Methods

- Best Subset Selection, Forward Selection
- Boosting (Luo and Spindler, 2017)
- k-Nearest Neighbours, Kernel, Splines
- Ensemble Methods (Künzel, Sekhon, Bickel, and Yu, 2018)
- etc.