Machine Learning Methods in Empirical Finance

Marcelo C. Medeiros

Departamento de Economia
Pontifícia Universidade Católica do Rio de Janeiro

Lecture 2
XVIII Encontro Brasileiro de Finanças

Regression Trees

Regression Trees

Introduction

- Flexible non-parametric predictive model.

Regression Trees

Introduction

- Flexible non-parametric predictive model.
- Recursive partition of the input space $\mathbb{X}$ (set of explanatory variables).

Regression Trees

Introduction

- Flexible non-parametric predictive model.
- Recursive partition of the input space $\mathbb{X}$ (set of explanatory variables).
- Hierarchical nature

Regression Trees

Introduction

- Flexible non-parametric predictive model.
- Recursive partition of the input space $\mathbb{X}$ (set of explanatory variables).
- Hierarchical nature
- Interpretability

Regression Trees

Introduction

- Flexible non-parametric predictive model.
- Recursive partition of the input space $\mathbb{X}$ (set of explanatory variables).
- Hierarchical nature
- Interpretability
- Categorical variables and missing data (treated as a category) easily handled.

Regression Trees

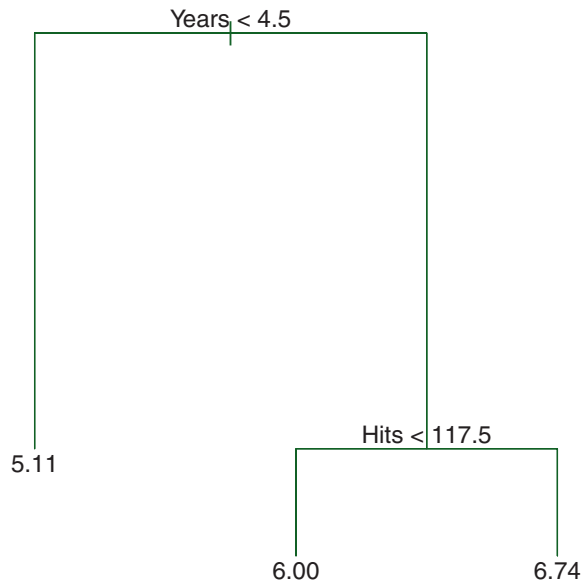
Introduction

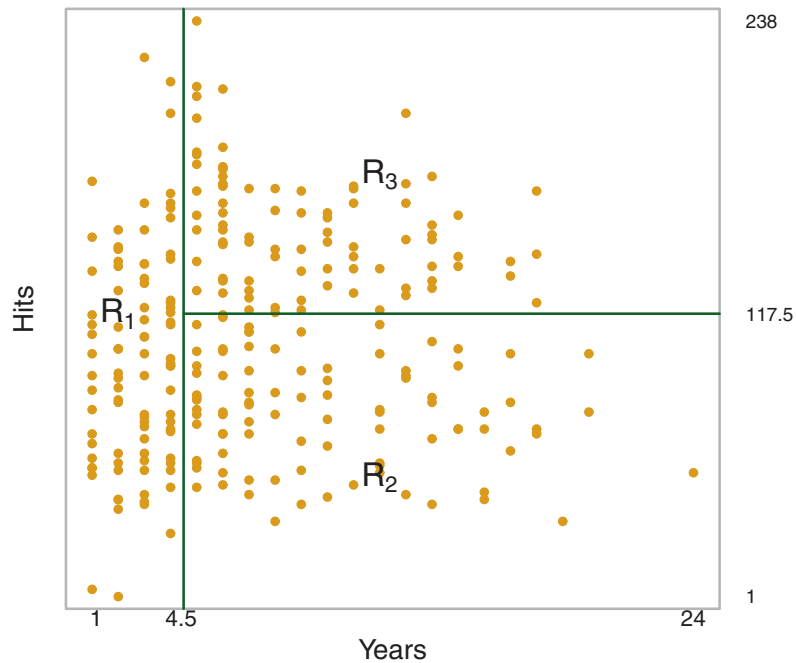
- Flexible non-parametric predictive model.
- Recursive partition of the input space $\mathbb{X}$ (set of explanatory variables).
- Hierarchical nature
- Interpretability
- Categorical variables and missing data (treated as a category) easily handled.
- Main disadvantage: **highly unstable**.
  - Easy fixes: *bagging* and *boosting*
  - Drawback: interpretability lost.

Regression Trees

Example

- Predict log wages of baseball players based on the length of the career (*years*) and number of *hits*.





Regression Trees

Mathematical Representation

Notation:

- number of terminal nodes (regions, *leaves*) K and N parent nodes;

Regression Trees

Mathematical Representation

Notation:

- number of terminal nodes (regions, *leaves*) K and N parent nodes;
- different regions denoted as $\mathcal{R}_1, \ldots, \mathcal{R}_K$;

Regression Trees

Mathematical Representation

Notation:

- number of terminal nodes (regions, *leaves*) K and N parent nodes;
- different regions denoted as $\mathcal{R}_1, \ldots, \mathcal{R}_K$;
- root node at position 0;

Regression Trees

Mathematical Representation

Notation:

- number of terminal nodes (regions, *leaves*) K and N parent nodes;
- different regions denoted as $\mathcal{R}_1, \ldots, \mathcal{R}_K$;
- root node at position 0;
- parent node at position j has two split (child) nodes at positions $2j + 1$ and $2j + 2$;

Regression Trees

Mathematical Representation

Notation:

- number of terminal nodes (regions, *leaves*) K and N parent nodes;
- different regions denoted as $\mathcal{R}_1, \ldots, \mathcal{R}_K$;
- root node at position 0;
- parent node at position j has two split (child) nodes at positions $2j + 1$ and $2j + 2$;
- each parent node has a threshold (split) variable associated, $x_{s_j t} \in \boldsymbol{x}_t$, where $s_j \in \mathbb{S} = \{1, 2, \ldots, p\}$; and

Regression Trees

Mathematical Representation

Notation:

- number of terminal nodes (regions, *leaves*) K and N parent nodes;
- different regions denoted as $\mathcal{R}_1, \ldots, \mathcal{R}_K$;
- root node at position 0;
- parent node at position j has two split (child) nodes at positions $2j + 1$ and $2j + 2$;
- each parent node has a threshold (split) variable associated, $x_{s_j t} \in \boldsymbol{x}_t$, where $s_j \in \mathbb{S} = \{1, 2, \ldots, p\}$; and
- $\mathbb{J}$ and $\mathbb{T}$ are the sets of parent and terminal nodes, respectively.

Regression Trees

Mathematical Representation

$$y_t = H_{\mathbb{J}\mathbb{T}}(\boldsymbol{x}_t; \boldsymbol{\psi}) + u_t = \sum_{i=1}^{K} \beta_i I(\boldsymbol{x}_t \in \mathcal{R}_i) + u_t = \sum_{i \in \mathbb{T}} \beta_i B_{\mathbb{J}i}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_i\right) + u_t$$

$$B_{\mathbb{J}i}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_i\right) = \prod_{j \in \mathbb{J}} I(x_{s_j,t}; c_j)^{\frac{n_{i,j}(1+n_{i,j})}{2}} \left[1 - I(x_{s_j,t}; c_j)\right]^{(1-n_{i,j})(1+n_{i,j})},$$

$$I(x_{s_j,t}; c_j) = \begin{cases} 1 & \text{if } x_{s_j,t} \leq c_j \\ 0 & \text{otherwise,} \end{cases}$$

$$n_{i,j} = \begin{cases} -1 & \text{if the path to leaf } i \text{ does not include parent node } j; \\ 0 & \text{if the path to leaf } i \text{ include the right-hand child of parent node } j; \\ 1 & \text{if the path to leaf } i \text{ include the left-hand child of parent node } j. \end{cases}$$

- $\mathbb{J}_i$: set of indexes of parent nodes included in the path to leaf i.
- $\boldsymbol{\theta}_i = \{c_k\}$ such that $k \in \mathbb{J}_i$, $i \in \mathbb{T}$ e $\sum_{j \in \mathbb{J}} B_{\mathbb{J}i}\left(\boldsymbol{x}_t; \boldsymbol{\theta}_j\right) = 1$.

Regression Trees

Recursive Partitioning

Idea:

- Choose the split variable x_j and the threshold c in order to reduce the squared errors.

Regression Trees

Recursive Partitioning

Idea:

- Choose the split variable x_j and the threshold c in order to reduce the squared errors.
- For each split, two new regions:

$$\mathcal{R}_1 = \{\boldsymbol{x} | x_j < x\} \quad \text{and} \quad \mathcal{R}_2 = \{\boldsymbol{x} | x_j \geq x\}$$

Regression Trees

Recursive Partitioning

Idea:

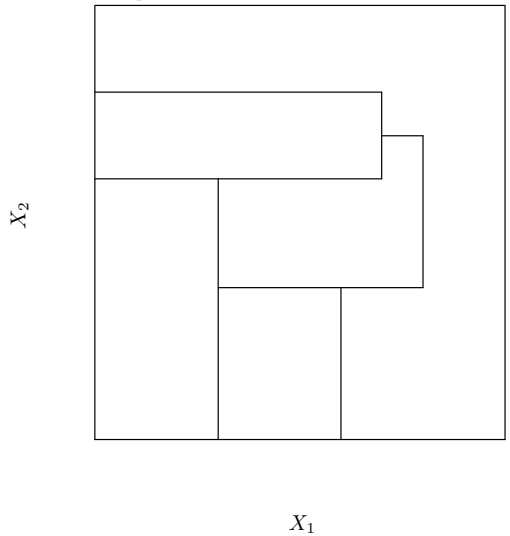
- Choose the split variable x_j and the threshold c in order to reduce the squared errors.
- For each split, two new regions:

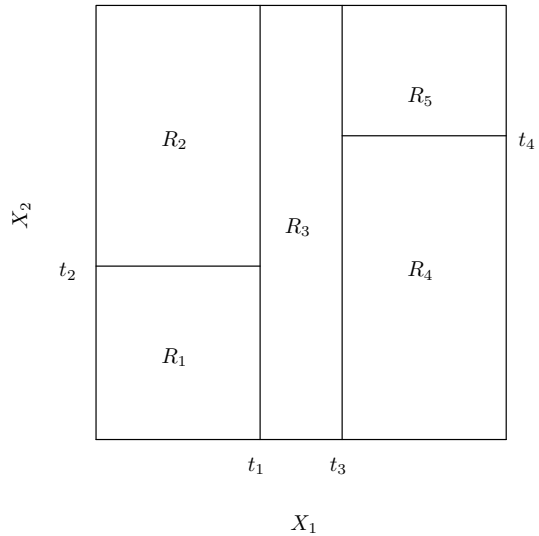
$$\mathcal{R}_1 = \{\boldsymbol{x}|x_j < x\} \quad \text{and} \quad \mathcal{R}_2 = \{\boldsymbol{x}|x_j \geq x\}$$

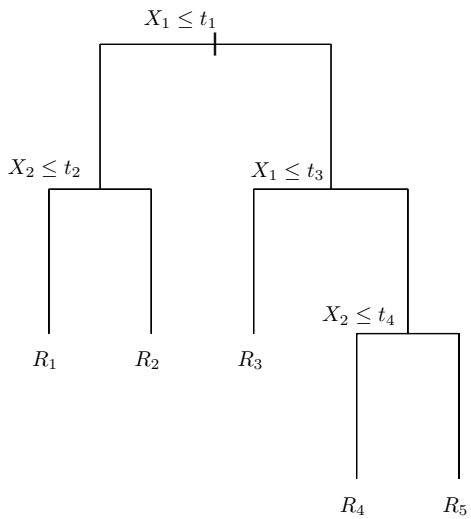
- Recursive partitions are created until the total sum of squares is bellow a certain pre-specified value or the number of observations in each region reached a minimum value.

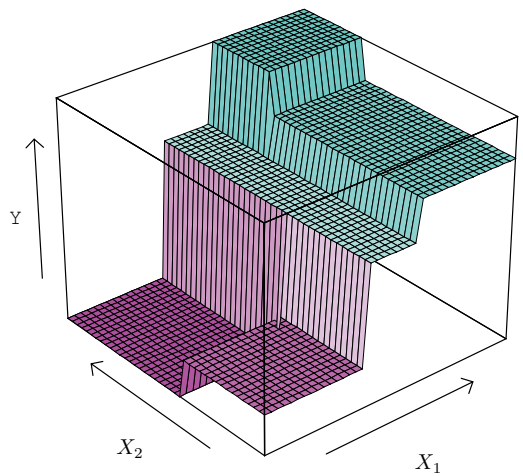
Regression Trees

Recursive Partitioning: Example









Regression Trees

Pruning

Idea:

- Grow a large tree $\mathcal{T}_0$ and prune the leaves in order to reduce complexity.

Regression Trees

Pruning

Idea:

- Grow a large tree $\mathcal{T}_0$ and prune the leaves in order to reduce complexity.
- The degree of pruning is controlled by an user-defined parameter α.

Regression Trees

Pruning

Idea:

- Grow a large tree $\mathcal{T}_0$ and prune the leaves in order to reduce complexity.
- The degree of pruning is controlled by an user-defined parameter α.
- For each tree $\mathcal{T} \subset \mathcal{T}_0$, define the loss function

$$C_\alpha(\mathcal{T}) = \text{sum of squared errors} + \alpha K,$$

where K is the number of leaves (regions) of $\mathcal{T}$.

Regression Trees

Pruning

Idea:

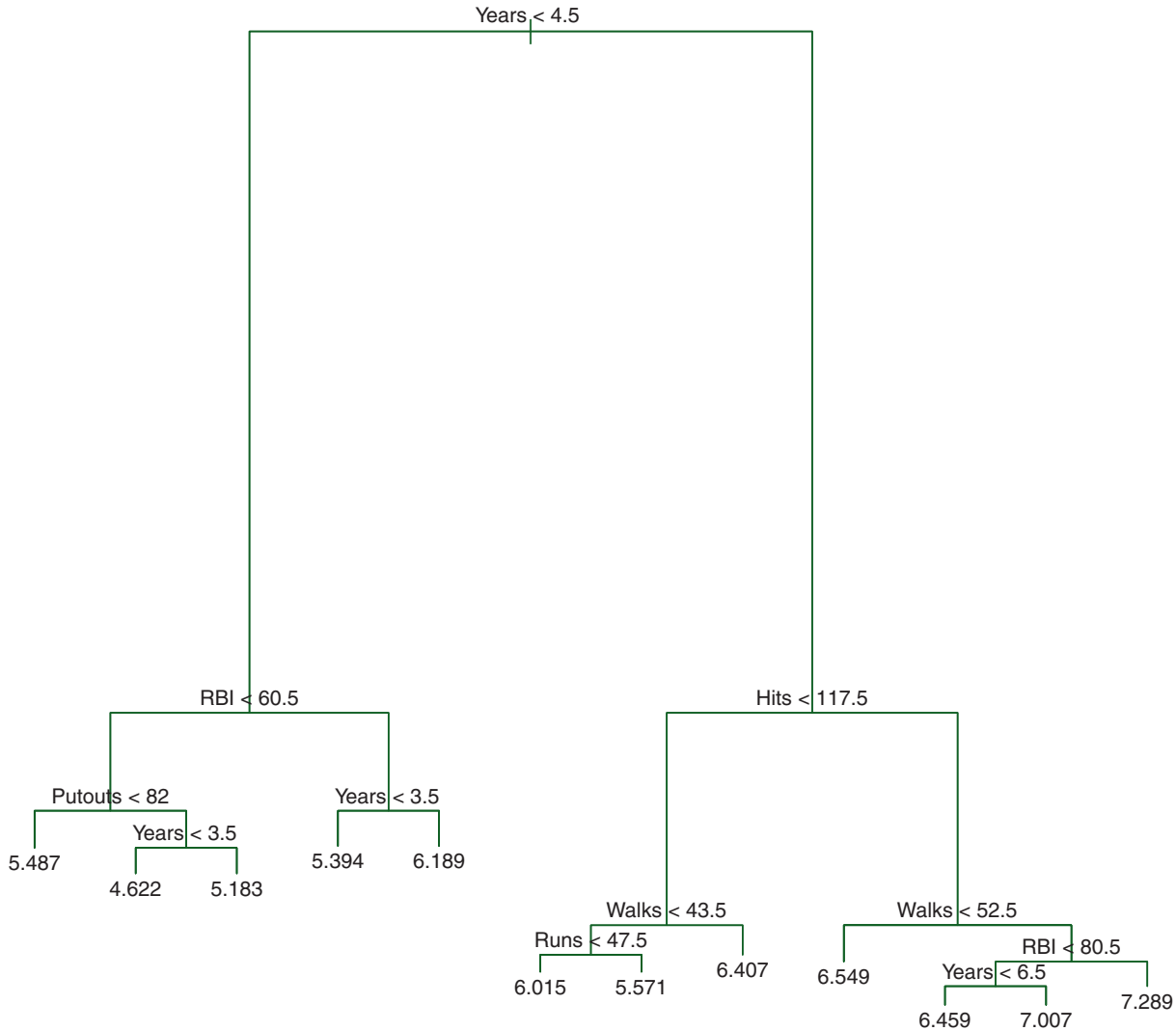
- Grow a large tree $\mathcal{T}_0$ and prune the leaves in order to reduce complexity.
- The degree of pruning is controlled by an user-defined parameter α.
- For each tree $\mathcal{T} \subset \mathcal{T}_0$, define the loss function

$$C_\alpha(\mathcal{T}) = \text{sum of squared errors} + \alpha K,$$

  where K is the number of leaves (regions) of $\mathcal{T}$.
- α is selected by cross-validation.

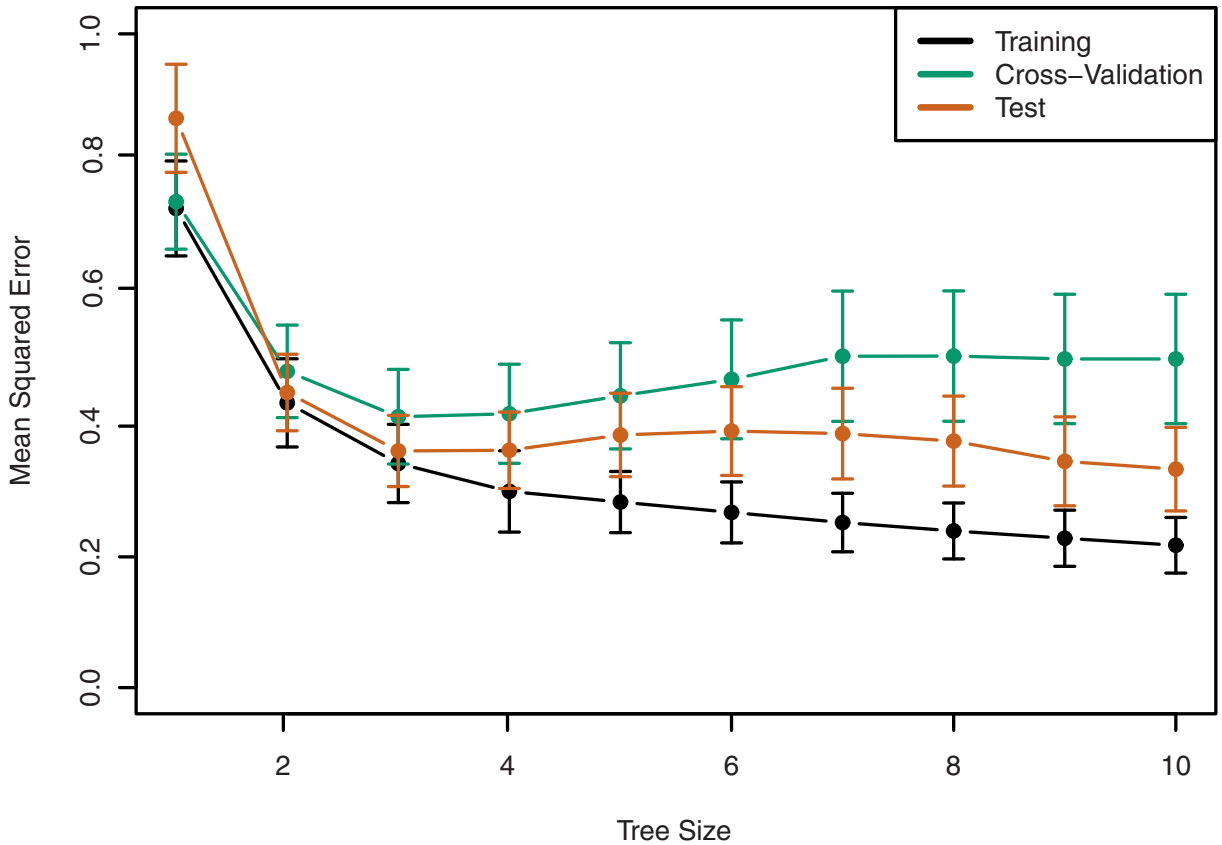
Regression Trees

Pruning: Example



Regression Trees

Pruning: Example



Boosting Regression Trees

Introduction to Boosting

- *Boosting* is a greedy algorithm for **additive models**.

Introduction to Boosting

- *Boosting* is a greedy algorithm for **additive models**.
  - A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.

Introduction to Boosting

- *Boosting* is a greedy algorithm for **additive models**.
  - A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.
  - In many problems, a greedy strategy does not produce an optimal solution, but may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.

Introduction to Boosting

- *Boosting* is a greedy algorithm for **additive models**.
  - A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.
  - In many problems, a greedy strategy does not produce an optimal solution, but may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.
  - Additive models:

$$y_t = f(\boldsymbol{x}_t) + u_t = \sum_{m=1}^{M} \beta_m f_m(\boldsymbol{x}_t) + u_t,$$

where:

- y_t: dependent variable;
- $\boldsymbol{x}_t \in \mathbb{R}^p$: vector of explanatory variables;
- u_t: random error;
- $f_m(\boldsymbol{x}_t)$: basis function or *weak learner*.

Introduction to Boosting

- *Boosting* is a greedy algorithm for **additive models**.
  - A greedy algorithm is an algorithmic paradigm that follows the problem solving heuristic of making the locally optimal choice at each stage with the hope of finding a global optimum.
  - In many problems, a greedy strategy does not produce an optimal solution, but may yield locally optimal solutions that approximate a global optimal solution in a reasonable time.
  - Additive models:

$$y_t = f(\boldsymbol{x}_t) + u_t = \sum_{m=1}^{M} \beta_m f_m(\boldsymbol{x}_t) + u_t,$$

where:

- y_t: dependent variable;
- $\boldsymbol{x}_t \in \mathbb{R}^p$: vector of explanatory variables;
- u_t: random error;
- $f_m(\boldsymbol{x}_t)$: basis function or *weak learner*.

- Origin of the method: classification problems.

Introduction to Boosting

- The goal is to estimate the optimal function f^*, defined as

$$\boxed{f^* := \arg\min_f \mathbb{E}\left[L(\boldsymbol{Y}, f(\boldsymbol{X}))\right],}$$

where:

- $\boldsymbol{Y} = (y_1, \ldots, y_t)'$
- $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iT})'$ and
- $L(\cdot, \cdot)$ is a **loss function**.

Introduction to Boosting

- The goal is to estimate the optimal function f^*, defined as

$$\boxed{f^* := \arg\min_f \mathbb{E}\left[L(\boldsymbol{Y}, f(\boldsymbol{X}))\right],}$$

  where:
  - $\boldsymbol{Y} = (y_1, \ldots, y_t)'$
  - $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iT})'$ and
  - $L(\cdot, \cdot)$ is a **loss function**.
- We consider just the **quadratic loss**: L_2-Boosting.

Introduction to Boosting

- The goal is to estimate the optimal function f^*, defined as

$$\boxed{f^* := \arg\min_f \mathbb{E}\left[L(\boldsymbol{Y}, f(\boldsymbol{X}))\right],}$$

  where:
  - $\boldsymbol{Y} = (y_1, \ldots, y_t)'$
  - $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_p)$ and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{iT})'$ and
  - $L(\cdot, \cdot)$ is a **loss function**.
- We consider just the **quadratic loss**: L_2-Boosting.
- Therefore, the goal is to estimate $\mathbb{E}(\boldsymbol{Y}|\boldsymbol{X})$.

Introduction to Boosting

- In practice, the optimization problem has the following form:

$$\widehat{f} = \arg\min_{f} \underbrace{\frac{1}{T}\sum_{t=1}^{T} L(y_t, f(\boldsymbol{x}_t))}_{:=\mathcal{R}},$$

Introduction to Boosting

- In practice, the optimization problem has the following form:

$$\widehat{f} = \arg\min_{f} \underbrace{\frac{1}{T}\sum_{t=1}^{T} L(y_t, f(\boldsymbol{x}_t))}_{:=\mathcal{R}},$$

- For quadratic loss:

$$\mathcal{R} = \frac{1}{T}\sum_{t=1}^{T} \left[y_t - f(\boldsymbol{x}_t)\right]^2.$$

Properties

- Easy to adjust for different choices of loss functions.

Properties

- Easy to adjust for different choices of loss functions.
- Variable selection.

Properties

- Easy to adjust for different choices of loss functions.
- Variable selection.
- Useful when $p >> T$. In this case,

$$f^* := \sum_{i=1}^{p} f_i(x_{it}).$$

Properties

- Easy to adjust for different choices of loss functions.
- Variable selection.
- Useful when $p >> T$. In this case,

$$f^* := \sum_{i=1}^{p} f_i(x_{it}).$$

- Robust to multi-collinearity.

Properties

- Easy to adjust for different choices of loss functions.
- Variable selection.
- Useful when $p >> T$. In this case,

$$f^* := \sum_{i=1}^{p} f_i(x_{it}).$$

- Robust to multi-collinearity.
- Optimizes the predictive power.

Gradient Boosting

Algorithm:

1. Initialize a T-dimensional vector $\widehat{\boldsymbol{f}}^{[0]}$ with some value. Set $j = 0$ and choose the weak learners. Set the number of learners to p.
2. Set $j = j + 1$ and compute

$$-\frac{\partial}{\partial \boldsymbol{f}} L(\boldsymbol{Y}, \boldsymbol{f}) \quad \text{and} \quad \widehat{\boldsymbol{f}}^{[j-1]}(\boldsymbol{x}_t),\ t = 1, \ldots, T.$$

Write

$$\begin{aligned} \boldsymbol{U}^{[j-1]} &= \left[U_t^{[j-1]}\right]_{t=1,\ldots,T} \\ &:= \left[-\frac{\partial}{\partial \boldsymbol{f}} L(\boldsymbol{Y}, \boldsymbol{f}) \Big|_{\boldsymbol{Y}=y_t, \boldsymbol{f}=\widehat{f}^{[j-1]}(\boldsymbol{x}_t)} \right]_{t=1,\ldots,T} \end{aligned}$$

Gradient Boosting

Algorithm (continuation):

3. Compute $\boldsymbol{U}^{[j-1]}$ and choose the learner that best fits $\boldsymbol{U}^{[j-1]}$. Set $\widehat{\boldsymbol{U}}^{[j-1]}$ as the fitted values of the best model.
4. Update

$$\widehat{\boldsymbol{f}}^{[j]} = \widehat{\boldsymbol{f}}^{[j-1]} + \nu \widehat{\boldsymbol{U}}^{[j-1]}, \quad 0 < \nu \leq 1.$$

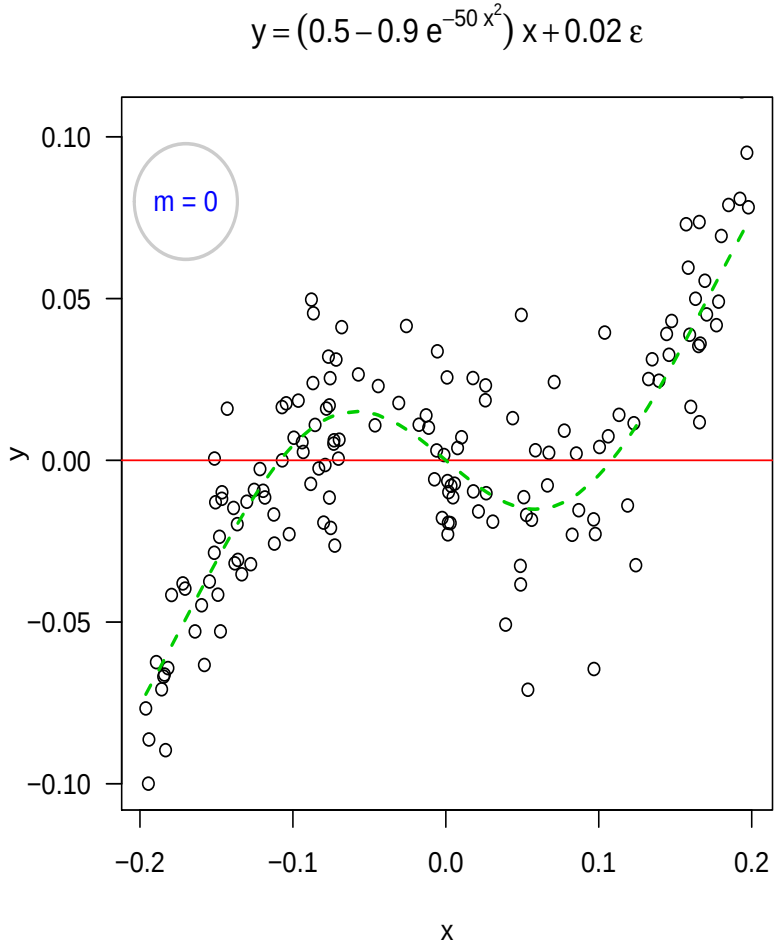
5. Repeat steps 2-4 until the maximum number of iterations is reached.

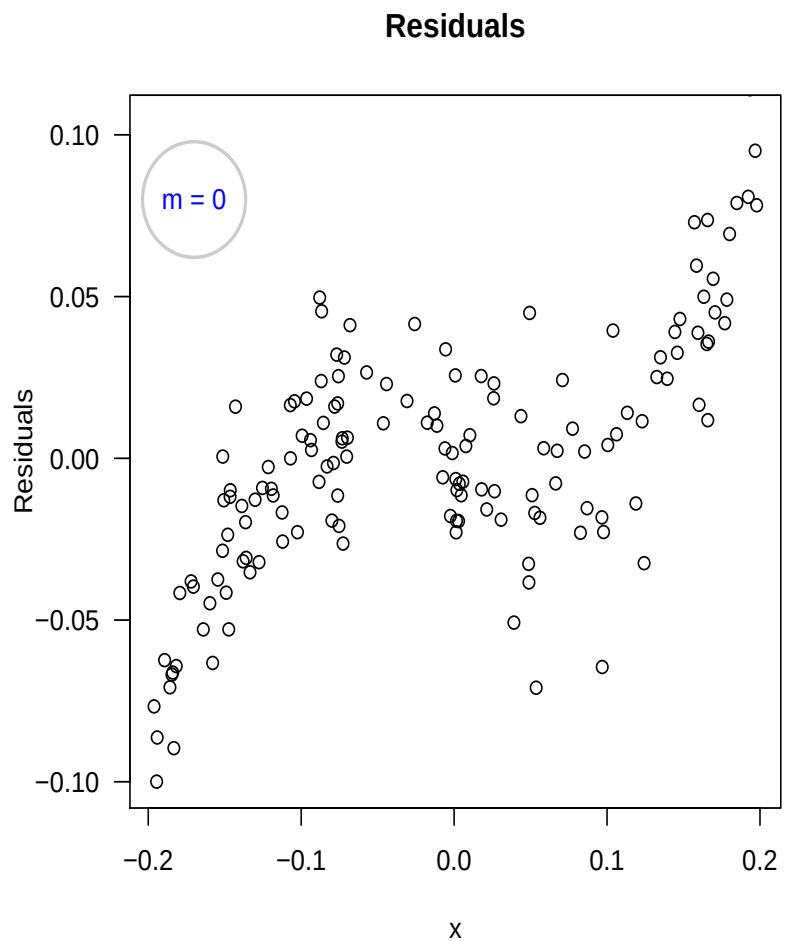
Componentwise Algorithm

- One learner for each covariate.

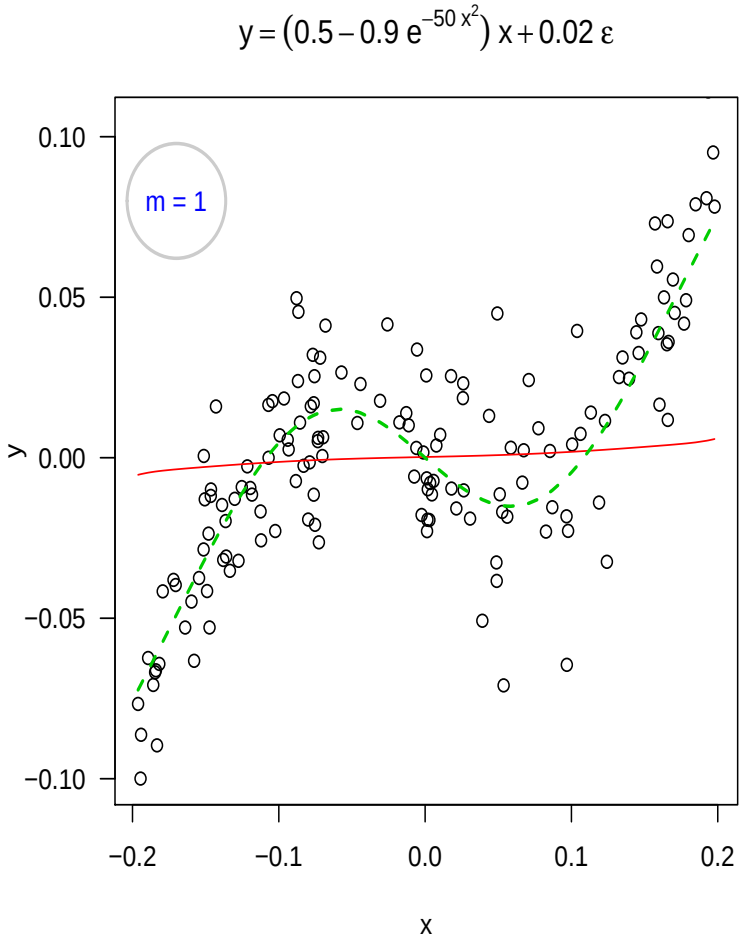
$$\begin{aligned}
\boldsymbol{U}^{[j-1]} &\sim \boldsymbol{x}_1 \\
\boldsymbol{U}^{[j-1]} &\sim \boldsymbol{x}_2 \\
\boldsymbol{U}^{[j-1]} &\sim \boldsymbol{x}_3 \\
&\vdots \\
\boldsymbol{U}^{[j-1]} &\sim \boldsymbol{x}_k \longrightarrow \text{Best Model} \longrightarrow \widehat{\boldsymbol{U}}^{[j-1]} \\
&\vdots \\
\boldsymbol{U}^{[j-1]} &\sim \boldsymbol{x}_p
\end{aligned}$$

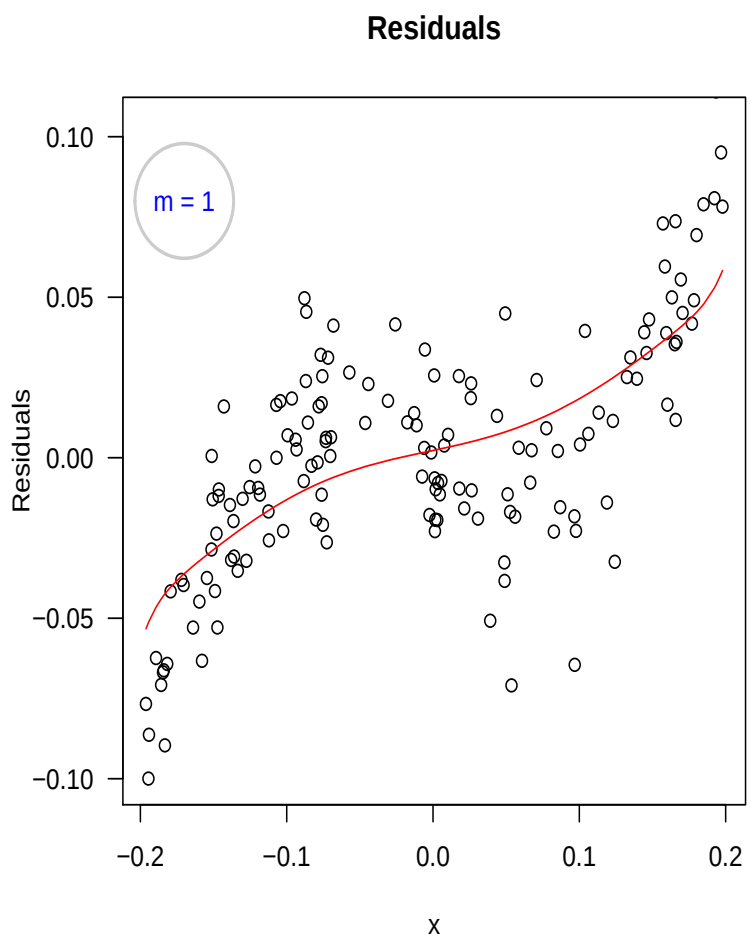
Example


$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 0
0.10
0.05
0.00
−0.05
−0.10
y
−0.2
−0.1
0.0
0.1
0.2
x

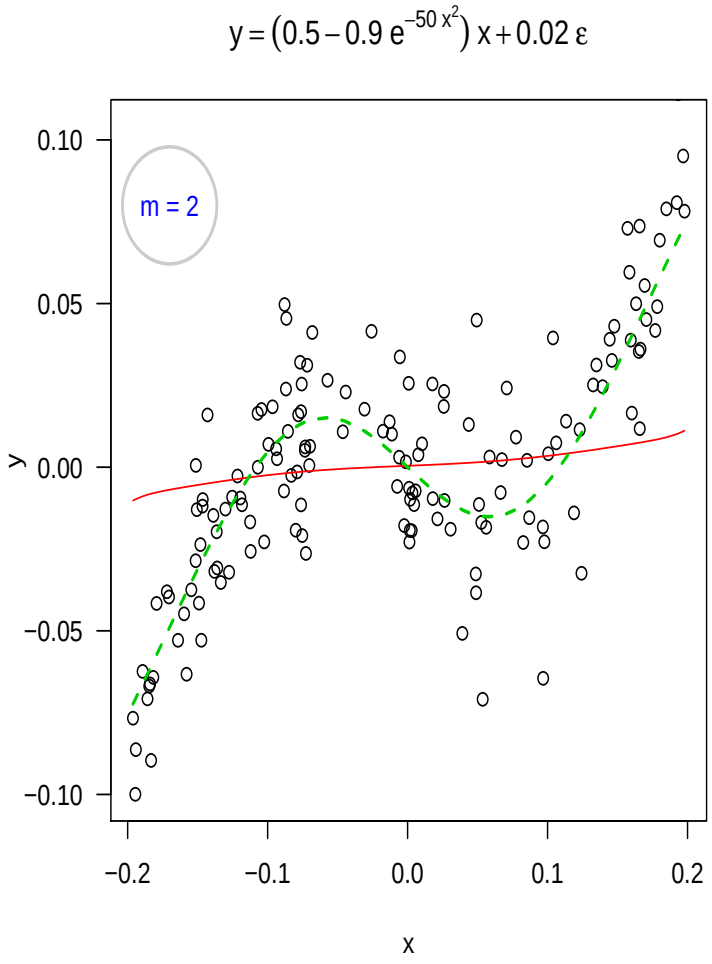

Residuals
m = 0
0.10
0.05
0.00
−0.05
−0.10
Residuals
−0.2
−0.1
0.0
0.1
0.2
x

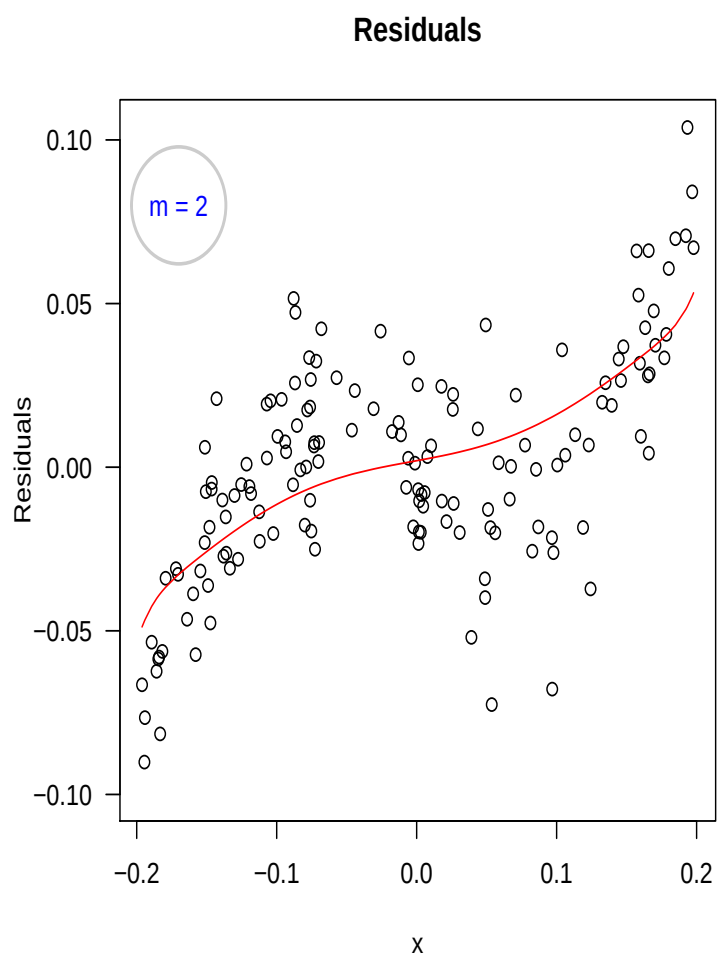
Example


$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 1
0.10
0.05
0.00
−0.05
−0.10
y
−0.2
−0.1
0.0
0.1
0.2
x

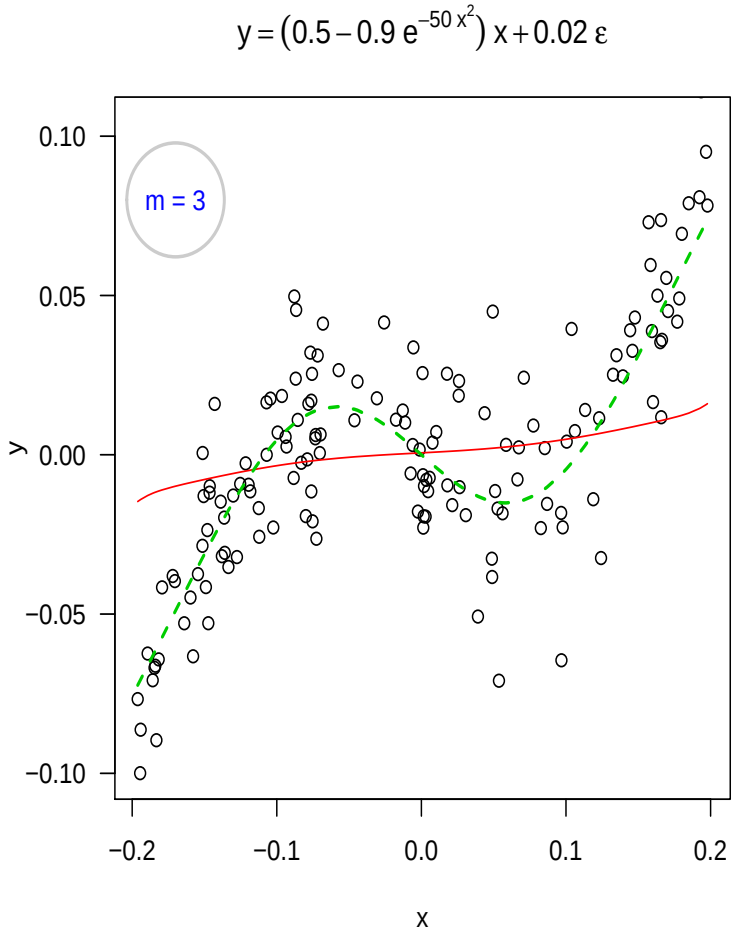

Residuals
m = 1
0.10
0.05
0.00
−0.05
−0.10
Residuals
−0.2
−0.1
0.0
0.1
0.2
x

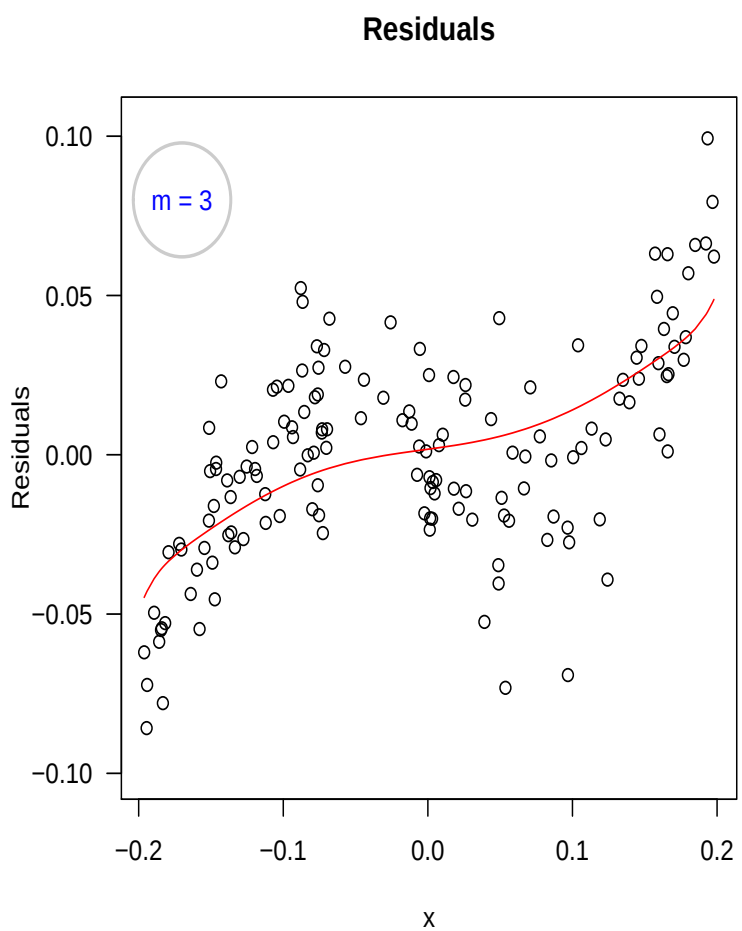
Example


$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 2
0.10
0.05
0.00
−0.05
−0.10
y
−0.2
−0.1
0.0
0.1
0.2
x

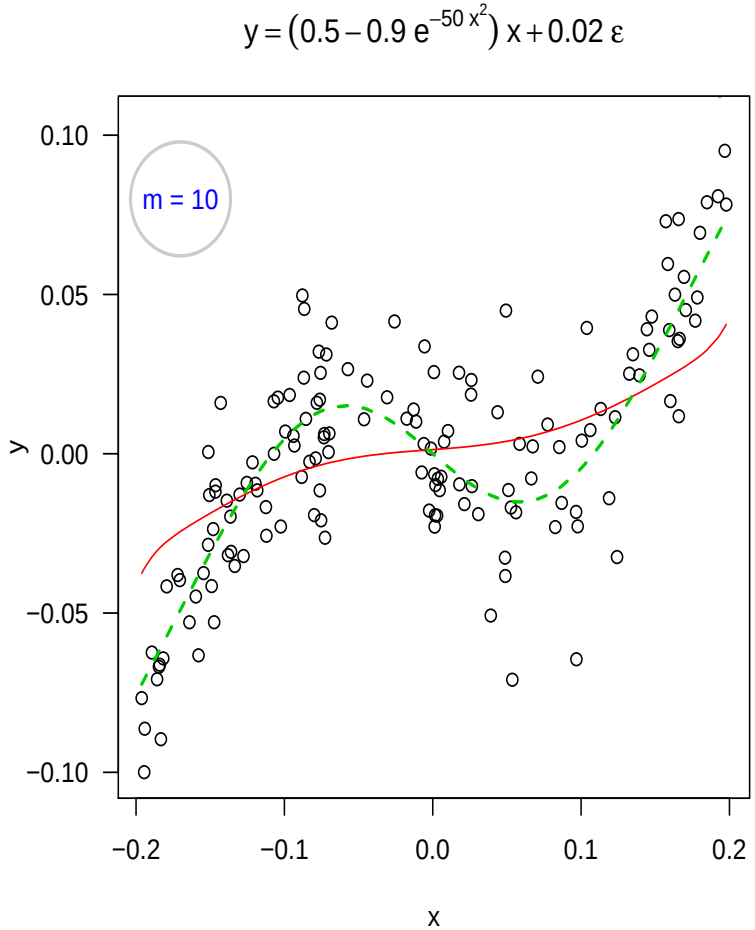

Residuals
m = 2
0.10
0.05
0.00
−0.05
−0.10
Residuals
−0.2
−0.1
0.0
0.1
0.2
x

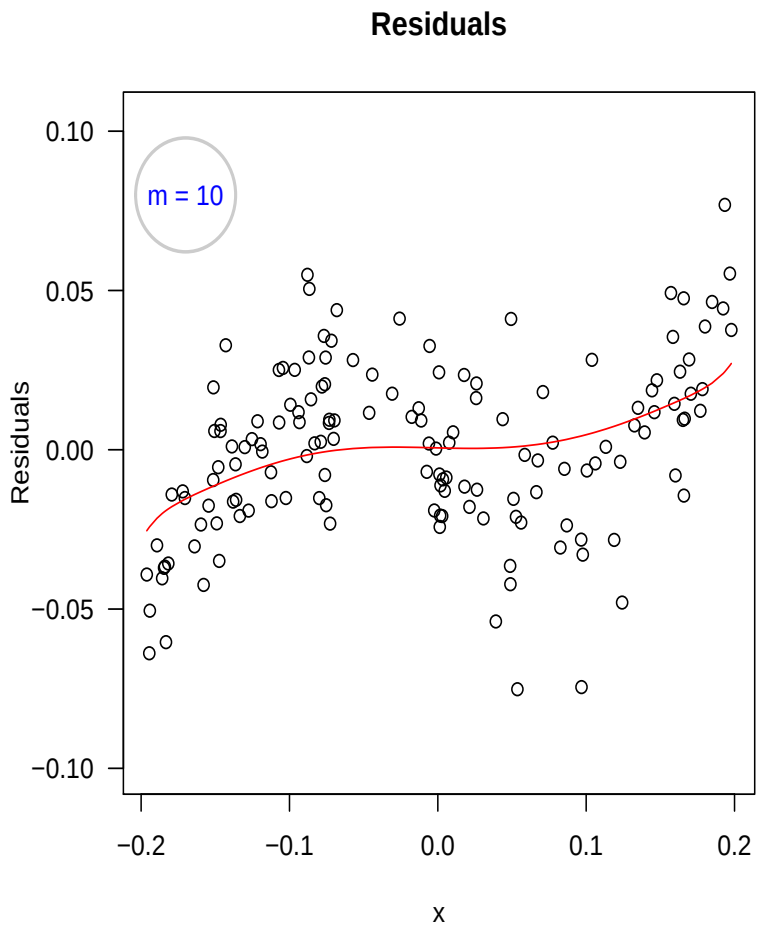
Example


$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 3
0.10
0.05
0.00
−0.05
−0.10
y
−0.2
−0.1
0.0
0.1
0.2
x

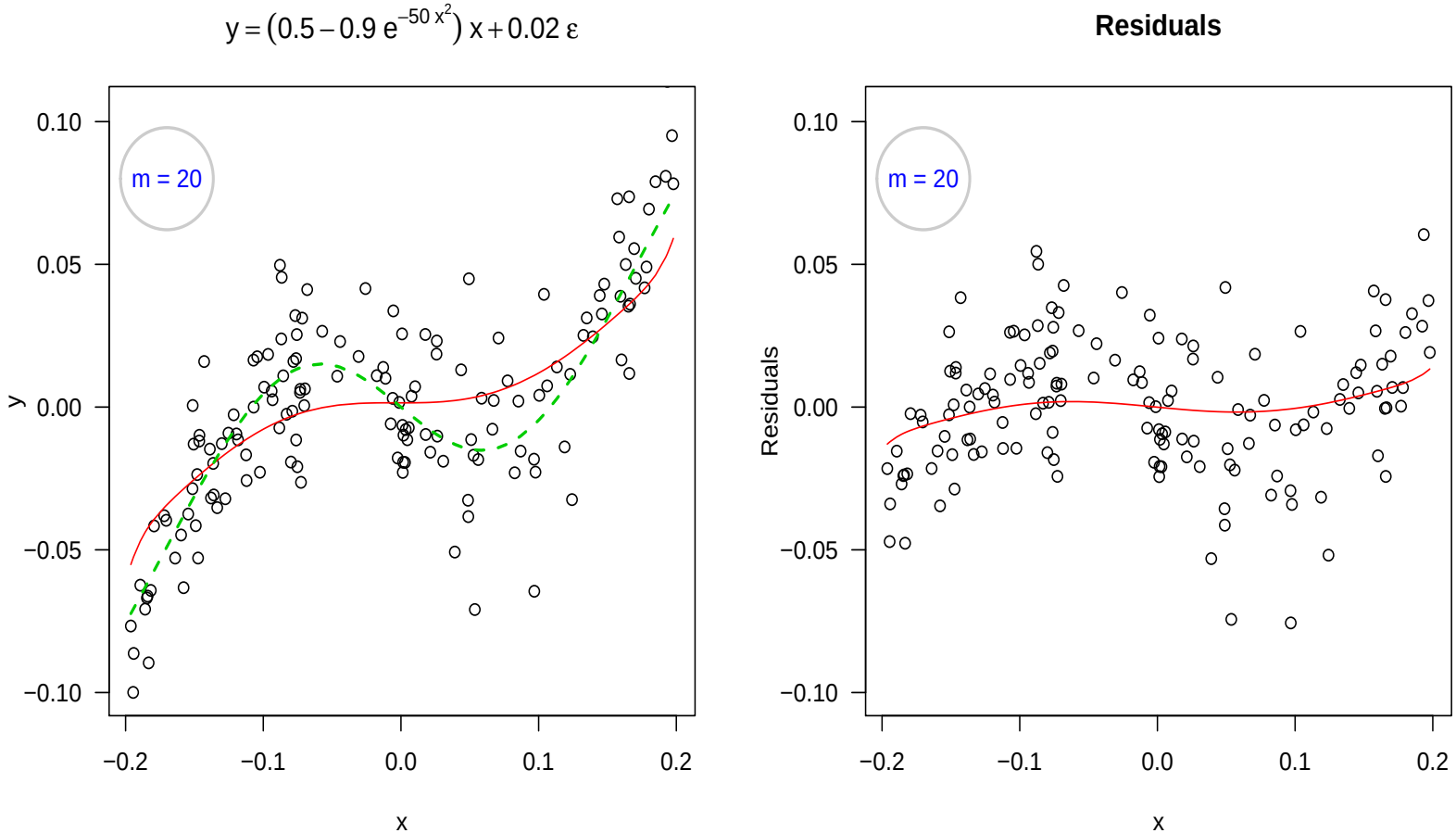

Residuals
m = 3
0.10
0.05
0.00
−0.05
−0.10
Residuals
−0.2
−0.1
0.0
0.1
0.2
x

Example

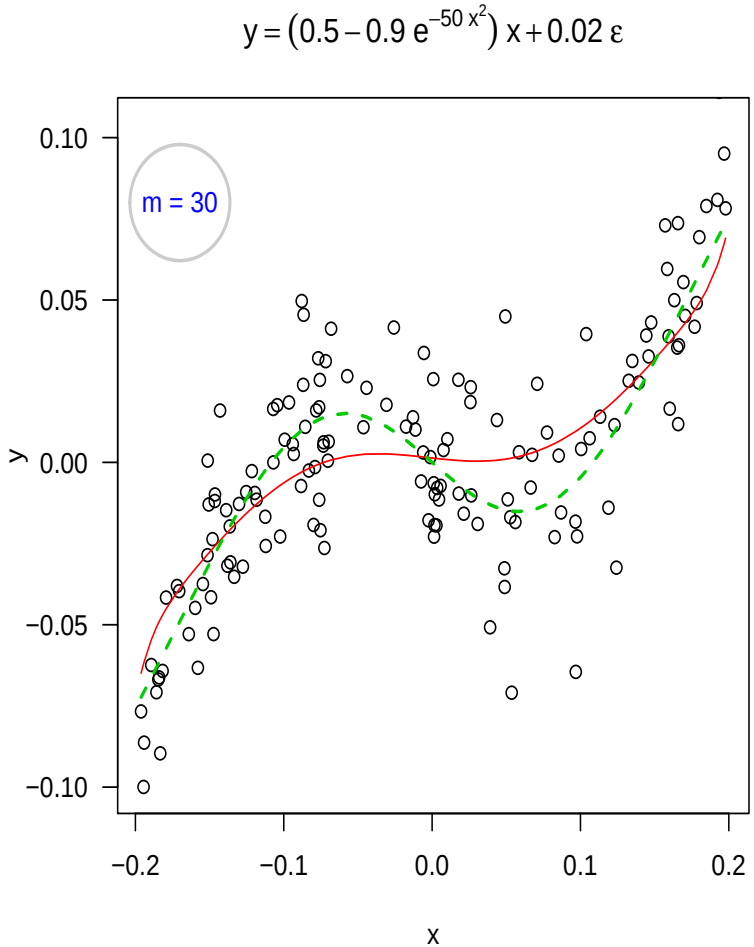

$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 10
0.10
0.05
0.00
−0.05
−0.10
y
−0.2
−0.1
0.0
0.1
0.2
x

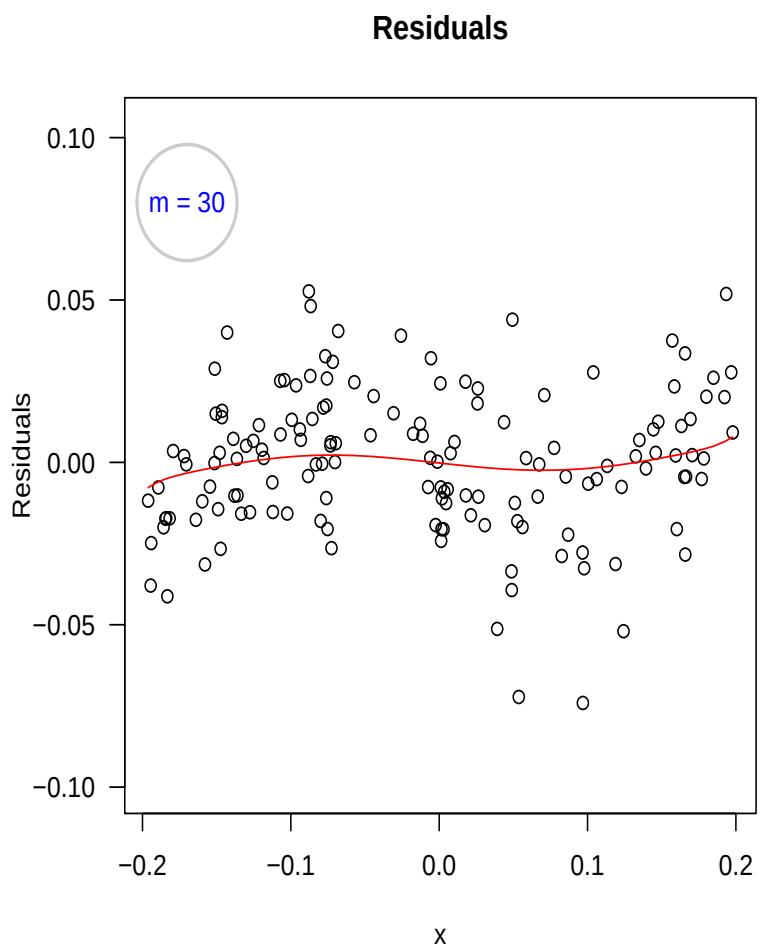

Residuals
m = 10
0.10
0.05
0.00
−0.05
−0.10
Residuals
−0.2
−0.1
0.0
0.1
0.2
x

Example

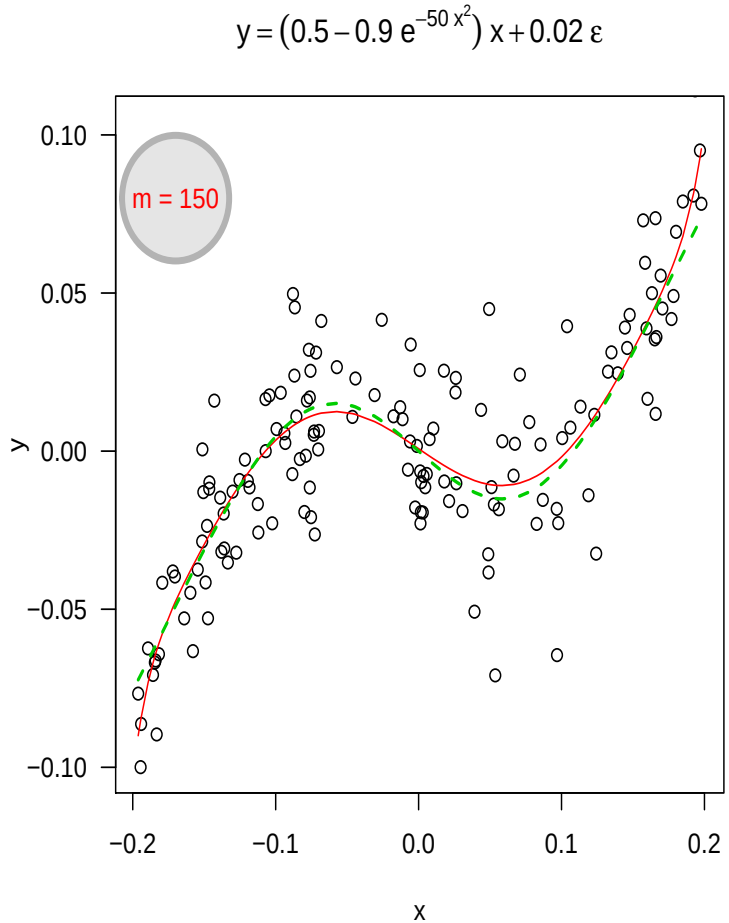

$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 20
y
x
Residuals
m = 20
Residuals
x

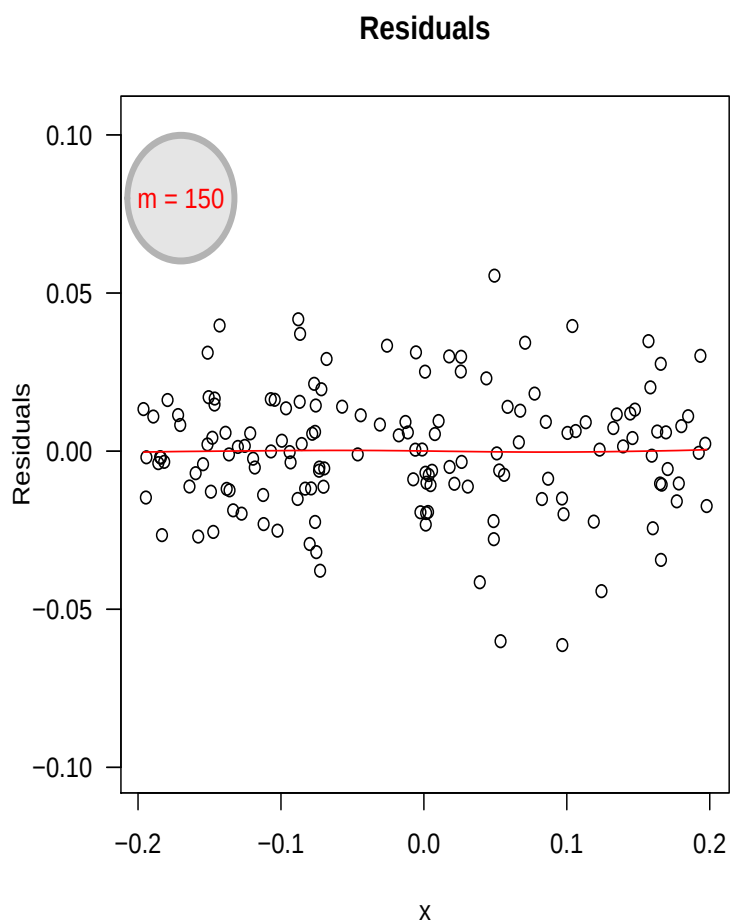
Example


$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 30
0.10
0.05
0.00
−0.05
−0.10
y
−0.2
−0.1
0.0
0.1
0.2
x


Residuals
m = 30
0.10
0.05
0.00
−0.05
−0.10
Residuals
−0.2
−0.1
0.0
0.1
0.2
x

Example


$y = (0.5 - 0.9\, e^{-50\, x^2})\, x + 0.02\, \varepsilon$
m = 150
0.10
0.05
0.00
−0.05
−0.10
y
−0.2
−0.1
0.0
0.1
0.2
x


Residuals
m = 150
0.10
0.05
0.00
−0.05
−0.10
Residuals
−0.2
−0.1
0.0
0.1
0.2
x

Gradient Boosting Properties

- It is clear from step 4 that

$$\boxed{\widehat{\boldsymbol{f}}^{[J]} = \widehat{\boldsymbol{f}}^{[0]} + \nu\widehat{\boldsymbol{U}}^{[0]} + \cdots + \nu\widehat{\boldsymbol{U}}^{[J-1]}.}$$

- The structure of the prediction function will depend on the choice of the weak learners.
  - Linear learners $\longrightarrow$ linear function
  - Smooth learners $\longrightarrow$ smooth function
- Variable selection
- Useful when $p >> T$.
- If $J \longrightarrow \infty$, the algorithm will select irrelevant regressors.
  - Solution: *early stopping* with cross-validation or information criteria.

Example: Linear Learners

Set-up:

- Three regressors: $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$.
- Three linear learners.
- $J = 5$
- Hypothetical solution: $\boldsymbol{x}_1$ selected at iterations 1, 2 and 5 and $\boldsymbol{x}_2$ selected at iterations 3 and 4.

$$
\begin{aligned}
\widehat{\boldsymbol{f}}^{[J]} &= \widehat{\boldsymbol{f}}^{[0]} + \nu\widehat{\boldsymbol{U}}^{[0]} + \nu\widehat{\boldsymbol{U}}^{[1]} + \nu\widehat{\boldsymbol{U}}^{[2]} + \nu\widehat{\boldsymbol{U}}^{[3]} + \nu\widehat{\boldsymbol{U}}^{[4]} \\
&= \widehat{\boldsymbol{\beta}}^{[0]} + \nu\widehat{\boldsymbol{\beta}}_1^{[0]}\boldsymbol{x}_1 + \nu\widehat{\boldsymbol{\beta}}_1^{[1]}\boldsymbol{x}_1 + \nu\widehat{\boldsymbol{\beta}}_3^{[2]}\boldsymbol{x}_3 + \nu\widehat{\boldsymbol{\beta}}_3^{[3]}\boldsymbol{x}_3 + \nu\widehat{\boldsymbol{\beta}}_1^{[4]}\boldsymbol{x}_1 \\
&= \widehat{\boldsymbol{\beta}}^{[0]} + \nu\left(\widehat{\boldsymbol{\beta}}_1^{[0]} + \widehat{\boldsymbol{\beta}}_1^{[1]} + \widehat{\boldsymbol{\beta}}_1^{[4]}\right)\boldsymbol{x}_1 + \nu\left(\widehat{\boldsymbol{\beta}}_3^{[2]} + \widehat{\boldsymbol{\beta}}_3^{[3]}\right)\boldsymbol{x}_3 \\
&= \widehat{\boldsymbol{\beta}}^{[0]} + \widehat{\boldsymbol{\beta}}_1\boldsymbol{x}_1 + \widehat{\boldsymbol{\beta}}_3\boldsymbol{x}_3.
\end{aligned}
$$

Boosting Regression Trees

Algorithm:

1. Initialize $f_0(\boldsymbol{x}) = \arg\min_c \sum_{t=1}^{T} L(y_t, c)$.
2. For $m = 1, \ldots, M$:
   - 2.1 For $t = 1, \ldots, T$, compute:

$$r_{tm} = -\left[\frac{\partial L(y_t, f(\boldsymbol{x}_t))}{\partial f(\boldsymbol{x}_t)}\right]_{f=f_{m-1}}$$

   - 2.2 Fit a regression tree for r_{tm} giving terminal regions (leaves) R_{jm}, $j = 1, \ldots, K_m$.
   - 2.3 For $j = 1, 2, \ldots, K_m$, compute

$$c_{jm} = \arg\min_c \sum_{x_t \in R_{jm}} L(y_t, f_{m-1}(x_t) + c)$$

   - 2.4 Update $f_m(\boldsymbol{x}) = f_{m-1}(\boldsymbol{x}) + \nu \sum_{j=1}^{K_m} c_{jm} I(\boldsymbol{x} \in R_{jm})$
3. Output: $\widehat{f}(\boldsymbol{x}) = f_M(\boldsymbol{x})$

Boosting Regression Trees

- Note that for square loss:

$$\frac{\partial L(y_t, f(\boldsymbol{x}_t))}{\partial f(\boldsymbol{x}_t)} = y_t - f(\boldsymbol{x}_t)$$

- What is the right size of the trees (K_m)?
  - Simple strategy: $K_m = K, \forall\, m$, K small.
  - The larger is K, the larger is the interaction among different variables. For example, if $K = 2$ there is no interaction.
  - Typically, in empirical works $4 \geq K \leq 8$.
- What are the values of M and ν?
  - ν $(0 < \nu < 1)$ is the learning rate.
  - Smaller values of ν (more shrinkage) result in larger training risk for the same number of iterations M.
  - Both ν and M control the prediction risk in the training data and do not operate independently.
  - Typically, ν is small $(\nu < 0.1)$ and M is chosen by *early stopping*.

Regression Trees

Boosting and Relative Importance of Predictor Variables

- Measure of relative importance of the ith covariate for a single tree $\mathcal{T}$:

$$\mathcal{I}_i(\mathcal{T}) = \sum_{j=1}^{N} \widehat{i}^2(s_j = i),$$

  where $N = K - 1$ is the number of internal nodes and $\widehat{i}^2(\cdot)$ is the estimated improvement in squared error risk over that for a constant fit over the entire region.

- For boosted trees:

$$\mathcal{I}_i = \sum_{m=1}^{M} \mathcal{I}_i(\mathcal{T}_m).$$

Boosting Regression Trees

Example: California Housing

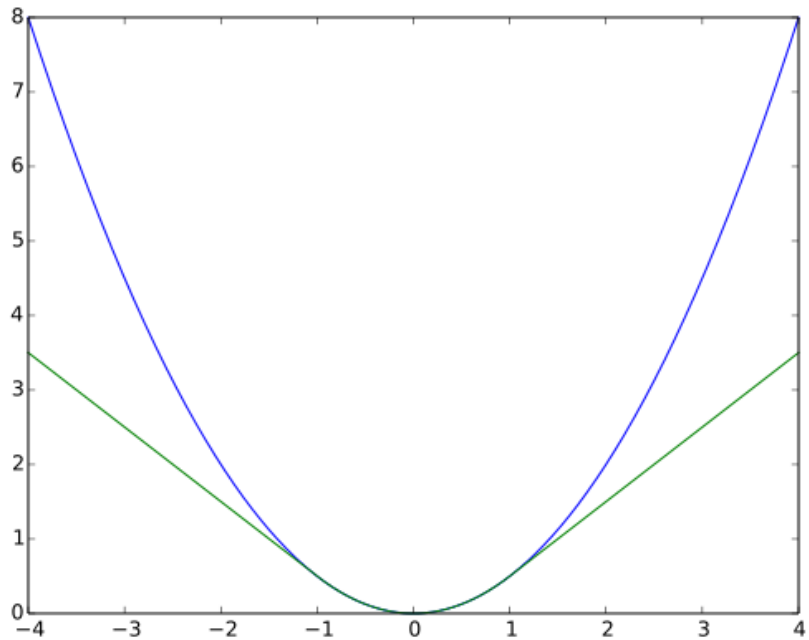
- Reference: Hastie, Tibshirani and Friedman (2009), Section 10.14.1 and Pace and Barry (1997,Stat&Prob Letters)
- Data is available form Carnegie-Mellon *Statlib* repository.
- The dataset consists of aggregated data from $20,460$ neighborhood (1990 census block groups) in California.
- The dependent variable y is the median house value in each neighborhood measures in units of $\$100,000$.
- Predictors are demographic variables such as median income (MedInc), housing density as reflected by the number of houses (House), and average occupancy in each house (AveOccup); location of each neighborhood (longitude and latitute); and several quantities reflecting the properties of the houses in the neighborhood such as average number of room AveRooms and bedrooms (AvgBedrms). Total of eight predictors, all numeric.

Boosting Regression Trees

Example: California Housing

- $K = 6$ and $\nu = 0.1$.
- Huber loss to control for outliers:

$$L(y_t, f(\boldsymbol{x}_t)) = \begin{cases} \frac{1}{2}\left[y_t - f(\boldsymbol{x}_t)\right]^2 & \text{if } |y_t - f(\boldsymbol{x}_t)| \leq \delta \\ \delta|y_t - f(\boldsymbol{x}_t)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$



Boosting Regression Trees

Example: California Housing

- For the Huber loss:

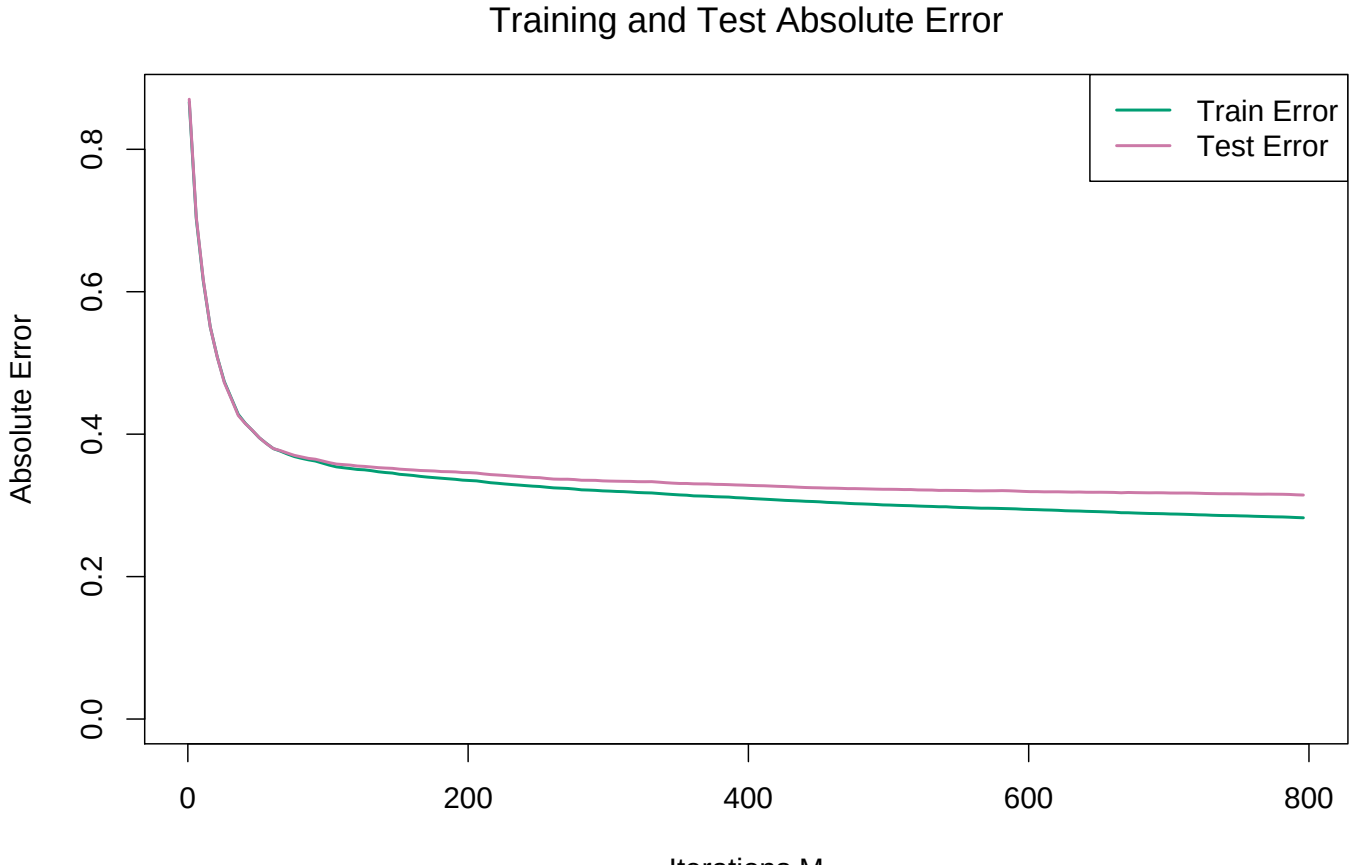
$$\frac{\partial L(y_t, f(\boldsymbol{x}_t))}{\partial f(\boldsymbol{x}_t)} = \begin{cases} y_t - f(\boldsymbol{x}_t) & \text{for } |y_i - f(\boldsymbol{x}_t)| \leq \delta \\ \delta \mathsf{sign}[y_t - f(\boldsymbol{x}_t)] & \text{otherwise,} \end{cases}$$

  where $\delta = \alpha - \text{quantile}[|y_i - f(\boldsymbol{x}_t)|]$.

Boosting Regression Trees

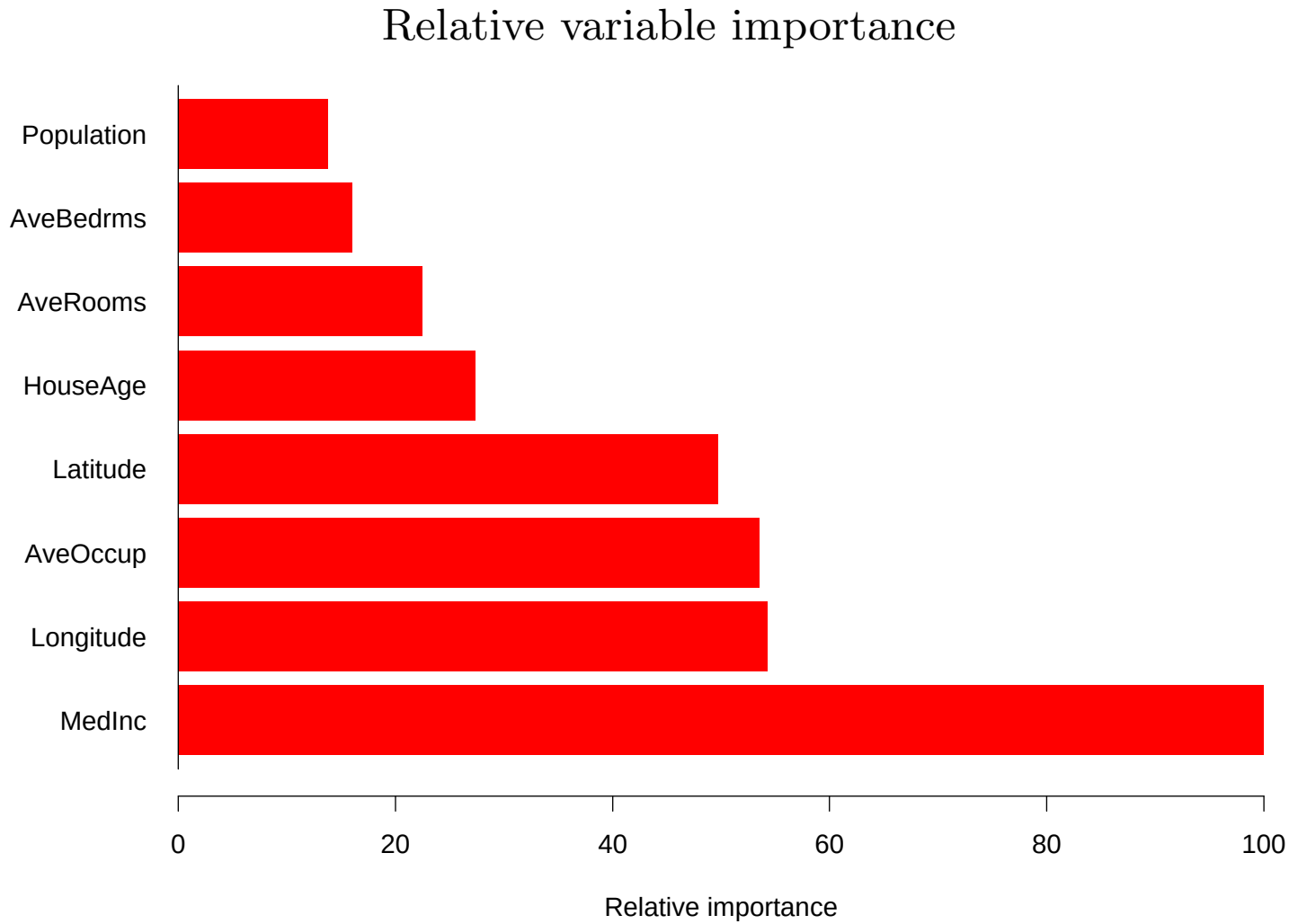
Example: California Housing

Average-absolute error as a function of iterations



Boosting Regression Trees

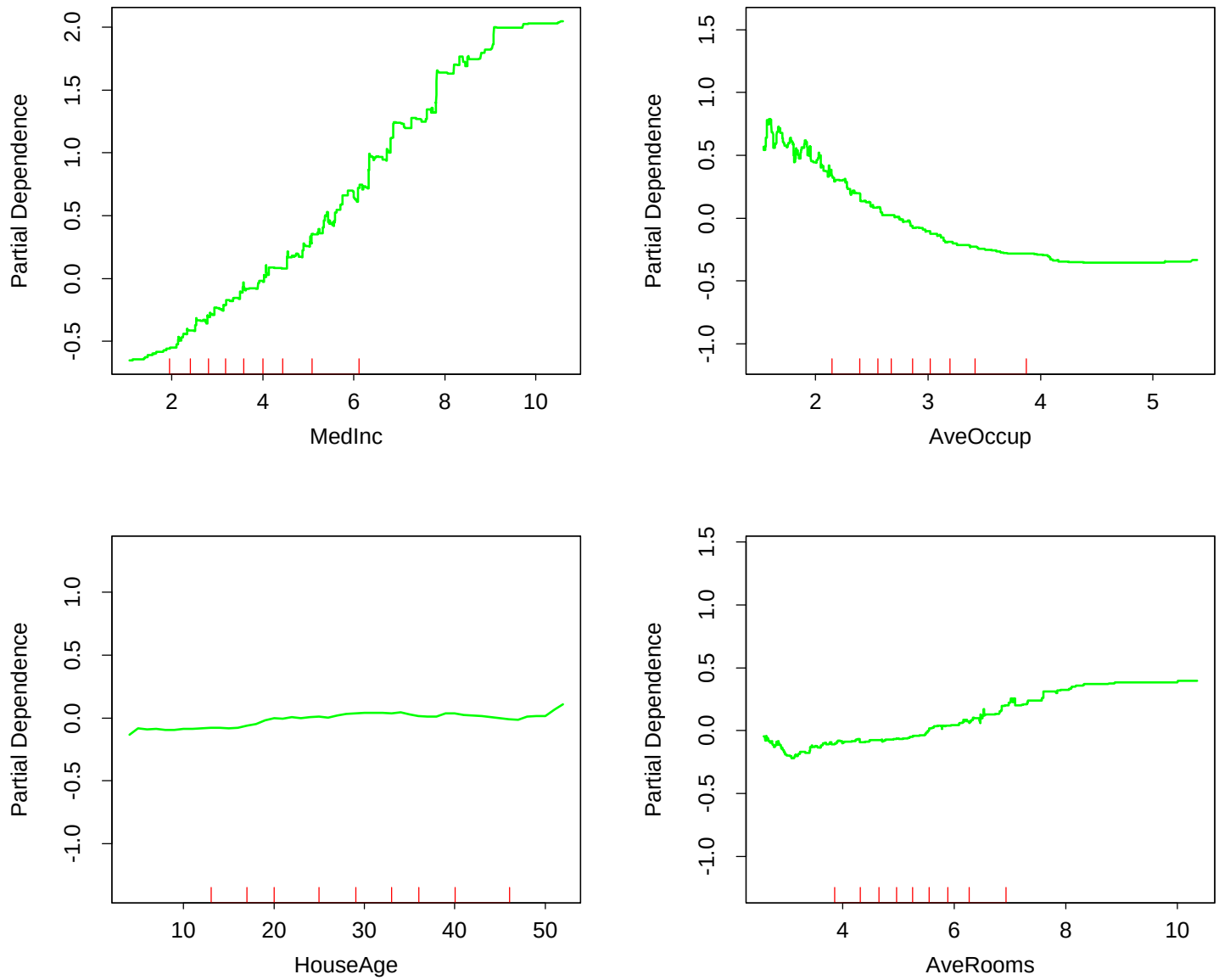
Example: California Housing



Boosting Regression Trees

Example: California Housing

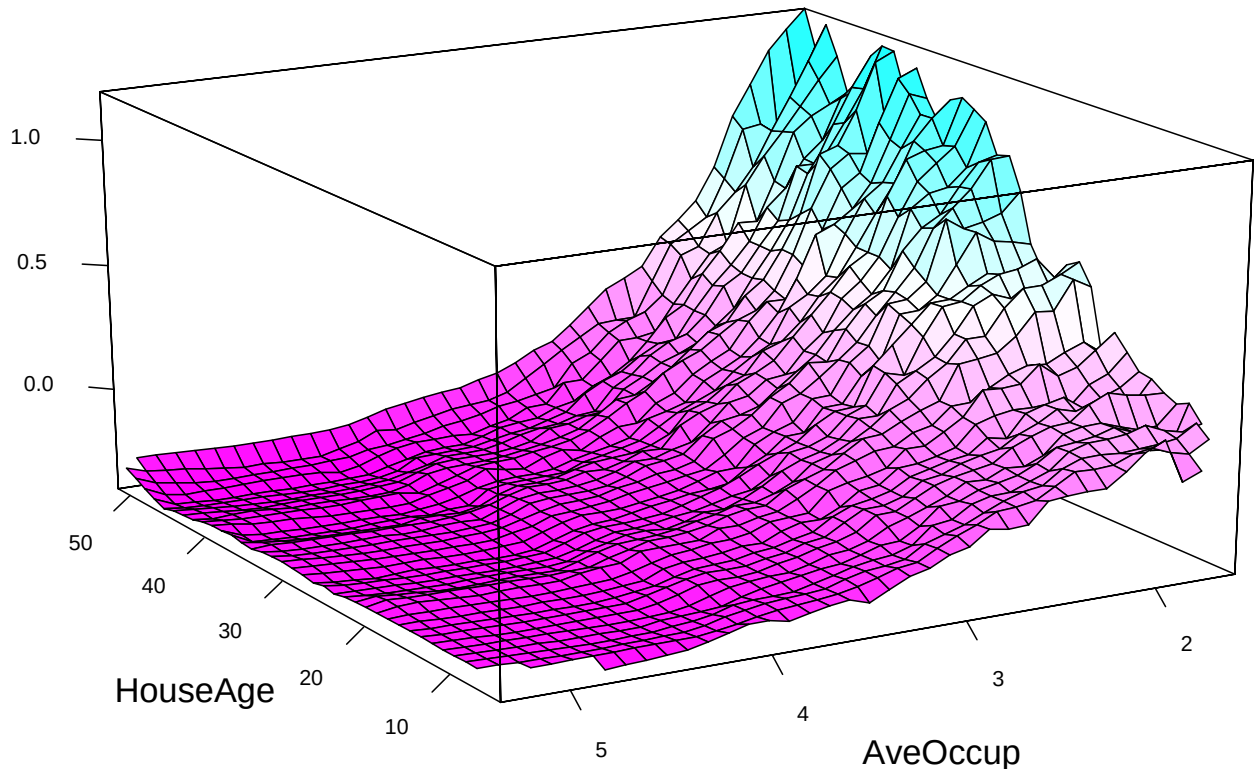
Partial dependence of housing value on nonlocation variables



Boosting Regression Trees

Example: California Housing

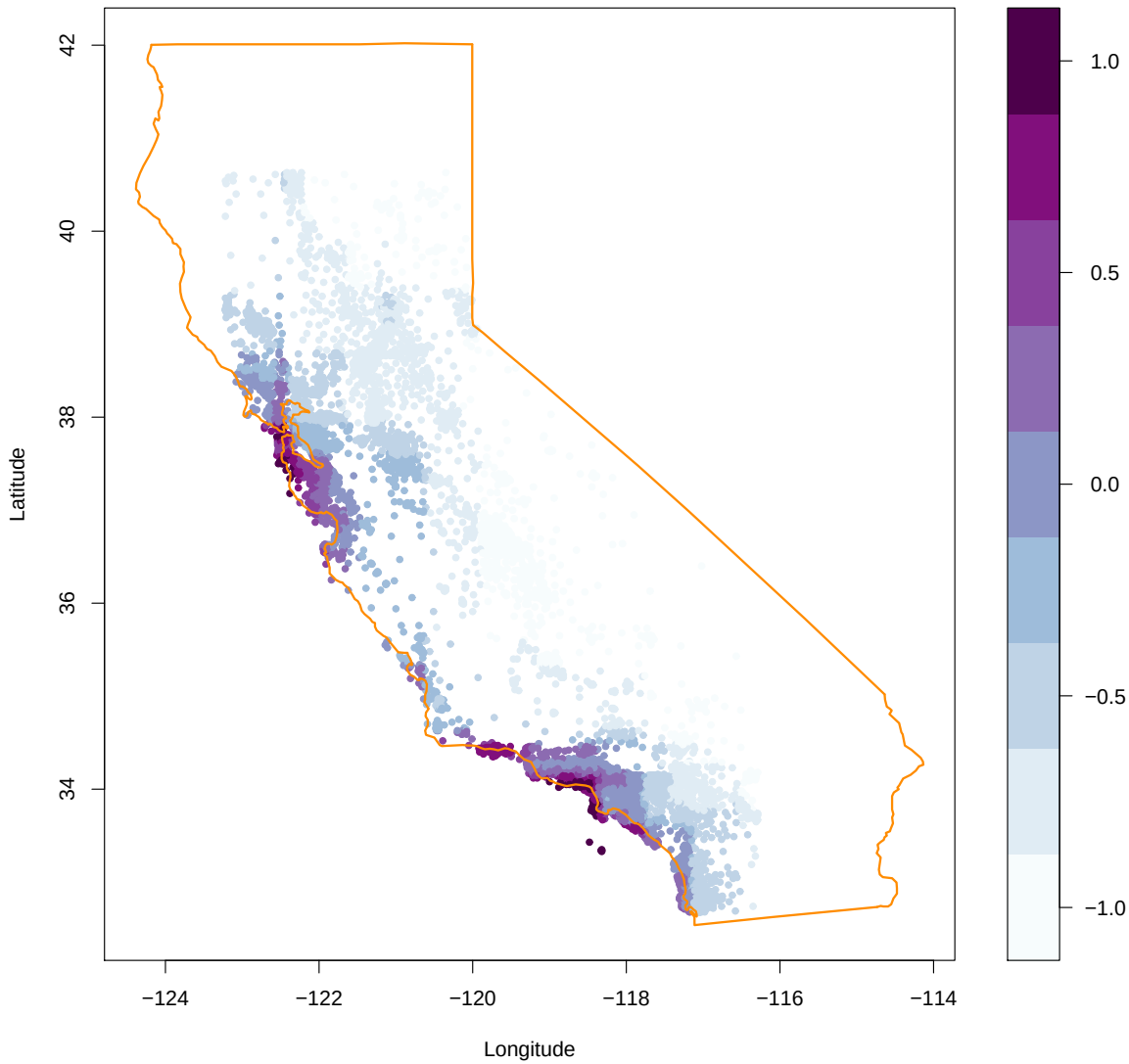
Partial dependence of house value on median age and average occupancy



Boosting Regression Trees

Example: California Housing

Partial dependence of house value on location



Bagging Regression Trees: Random Forests

What is Bagging?

- Bagging = Bootstrap Aggregating

What is Bagging?

- Bagging = Bootstrap Aggregating
- Introduced by Breiman(1996, ML) to reduce the variance of a predictor.

What is Bagging?

- Bagging = Bootstrap Aggregating
- Introduced by Breiman(1996, ML) to reduce the variance of a predictor.
- Let's consider a regression setup where data is collected as

$$\boldsymbol{Z}_t = (y_t, \boldsymbol{x}_t'), \quad t = 1, \ldots, T.$$

What is Bagging?

- Bagging = Bootstrap Aggregating
- Introduced by Breiman(1996, ML) to reduce the variance of a predictor.
- Let's consider a regression setup where data is collected as

$$\boldsymbol{Z}_t = (y_t, \boldsymbol{x}_t'), \quad t = 1, \ldots, T.$$

- Let

$$\widehat{\theta}_T(\boldsymbol{x}) = h_T(\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_T)(\boldsymbol{x})$$

be an estimator of $\mathbb{E}(y|\boldsymbol{x})$.

What is Bagging?

The Bagging Algorithm

1. Construct a bootstrap sample

$$\boldsymbol{Z}_t^* = (y_t^*, \boldsymbol{x}_t^{*'}), \quad t = 1, \ldots, T.$$

according to the empirical distribution of the pairs $(y_t, \boldsymbol{x}_t)$, $t = 1, \ldots, T$.

What is Bagging?

The Bagging Algorithm

1. Construct a bootstrap sample

$$\boldsymbol{Z}_t^* = (y_t^*, \boldsymbol{x}_t^{*'}), \quad t = 1, \ldots, T.$$

   according to the empirical distribution of the pairs $(y_t, \boldsymbol{x}_t)$, $t = 1, \ldots, T$.

2. Compute the bootstrapped predictor $\widehat{\theta}_T^*(\boldsymbol{x})$ by the plug-in principle; that is,

$$\widehat{\theta}_T^*(\boldsymbol{x}) = h_T(\boldsymbol{Z}_1^*, \ldots, \boldsymbol{Z}_T^*)(\boldsymbol{x}).$$

What is Bagging?

The Bagging Algorithm

1. Construct a bootstrap sample

$$\boldsymbol{Z}_t^* = (y_t^*, \boldsymbol{x}_t^{*'}), \quad t = 1, \ldots, T.$$

   according to the empirical distribution of the pairs $(y_t, \boldsymbol{x}_t)$, $t = 1, \ldots, T$.

2. Compute the bootstrapped predictor $\widehat{\theta}_T^*(\boldsymbol{x})$ by the plug-in principle; that is,

$$\widehat{\theta}_T^*(\boldsymbol{x}) = h_T(\boldsymbol{Z}_1^*, \ldots, \boldsymbol{Z}_T^*)(\boldsymbol{x}).$$

3. The bagged predictor is

$$\widehat{\theta}_T^B(\boldsymbol{x}) = \mathbb{E}^* \left[\widehat{\theta}_T^*(\boldsymbol{x}) \right]$$

What is Bagging?

- In practice the expectation $\mathbb{E}^*\left[\widehat{\theta}^*_T(\boldsymbol{x})\right]$ is computed by Monte Carlo:

What is Bagging?

- In practice the expectation $\mathbb{E}^*\left[\widehat{\theta}^*_T(\boldsymbol{x})\right]$ is computed by Monte Carlo:
  - For each bootstrap simulation j, $j = 1, \ldots, B$

$$\widehat{\theta}^*_{T,(j)}(\boldsymbol{x}) = h_n(\boldsymbol{Z}^*_{1,(j)}, \ldots, \boldsymbol{Z}^*_{T,(j)})(\boldsymbol{x}).$$

What is Bagging?

- In practice the expectation $\mathbb{E}^*\left[\widehat{\theta}^*_T(\boldsymbol{x})\right]$ is computed by Monte Carlo:
  - For each bootstrap simulation j, $j = 1, \ldots, B$

$$\widehat{\theta}^*_{T,(j)}(\boldsymbol{x}) = h_n(\boldsymbol{Z}^*_{1,(j)}, \ldots, \boldsymbol{Z}^*_{T,(j)})(\boldsymbol{x}).$$

  - Therefore,

$$\widehat{\theta}^*_T(\boldsymbol{x}) \approx \frac{1}{B}\sum_{j=1}^{B} \widehat{\theta}^*_{T,(j)}(\boldsymbol{x}).$$

What is Bagging?

- In practice the expectation $\mathbb{E}^*\left[\widehat{\theta}^*_T(\boldsymbol{x})\right]$ is computed by Monte Carlo:
  - For each bootstrap simulation j, $j = 1, \ldots, B$

$$\widehat{\theta}^*_{T,(j)}(\boldsymbol{x}) = h_n(\boldsymbol{Z}^*_{1,(j)}, \ldots, \boldsymbol{Z}^*_{T,(j)})(\boldsymbol{x}).$$

  - Therefore,

$$\widehat{\theta}^*_T(\boldsymbol{x}) \approx \frac{1}{B}\sum_{j=1}^{B} \widehat{\theta}^*_{T,(j)}(\boldsymbol{x}).$$

- There can be a drastic variance reduction if the predictor is unstable.

What is Bagging?

Stability of a predictor

A statistic

$$\widehat{\theta}_T(\boldsymbol{x}) = h_T(\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_T)(\boldsymbol{x})$$

is called stable at $\boldsymbol{x}$ if

$$\widehat{\theta}_n(\boldsymbol{x}) = \theta(\boldsymbol{x}) + o_p(1)$$

as $n \longrightarrow \infty$ for some fixed value $\theta(\boldsymbol{x})$.

- Different from consistency.

What is Bagging?

Stability of a predictor

A statistic

$$\widehat{\theta}_T(\boldsymbol{x}) = h_T(\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_T)(\boldsymbol{x})$$

is called stable at $\boldsymbol{x}$ if

$$\widehat{\theta}_n(\boldsymbol{x}) = \theta(\boldsymbol{x}) + o_p(1)$$

as $n \longrightarrow \infty$ for some fixed value $\theta(\boldsymbol{x})$.

- Different from consistency.
- $\theta(\boldsymbol{x})$ is just a stable limit and not necessarily the parameter of interest.

What is Bagging?

Stability of a predictor

A statistic

$$\widehat{\theta}_T(\boldsymbol{x}) = h_T(\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_T)(\boldsymbol{x})$$

is called stable at $\boldsymbol{x}$ if

$$\widehat{\theta}_n(\boldsymbol{x}) = \theta(\boldsymbol{x}) + o_p(1)$$

as $n \longrightarrow \infty$ for some fixed value $\theta(\boldsymbol{x})$.

- Different from consistency.
- $\theta(\boldsymbol{x})$ is just a stable limit and not necessarily the parameter of interest.
- Subset model selection via testing creates unstable predictors.

What is Bagging?

Stability of a predictor

A statistic

$$\widehat{\theta}_T(\boldsymbol{x}) = h_T(\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_T)(\boldsymbol{x})$$

is called stable at $\boldsymbol{x}$ if

$$\widehat{\theta}_n(\boldsymbol{x}) = \theta(\boldsymbol{x}) + o_p(1)$$

as $n \longrightarrow \infty$ for some fixed value $\theta(\boldsymbol{x})$.

- Different from consistency.
- $\theta(\boldsymbol{x})$ is just a stable limit and not necessarily the parameter of interest.
- Subset model selection via testing creates unstable predictors.
- **Bagging reduces the variance of unstable predictors**.

Random Forests

Bagging and Random Forests

Algorithm:

1. For $b = 1, \ldots, B$:
   - 1.1 Draw a bootstrap sample of size T from the original data (sampling with replacement).
   - 1.2 For each sample, estimate a tree $\mathcal{T}_b$, using as potential split variables a subset of q out of the p original variables randomly chosen. The tree should grow until the minimum number of observations in each leaf is reached. **No pruning**.
2. The final prediction is given as:

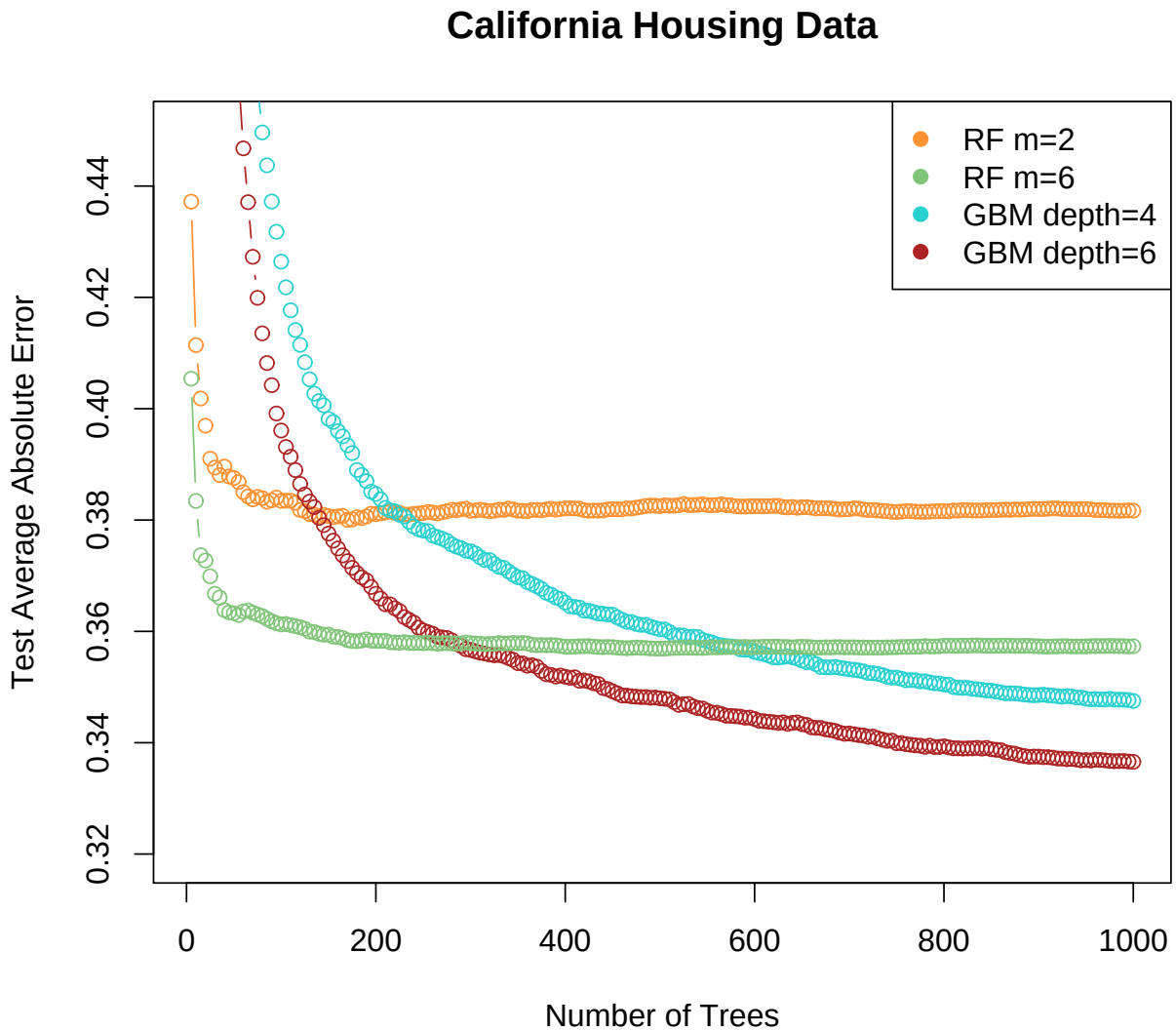
$$\frac{1}{B}\sum_{b=1}^{B}\mathcal{T}_b(\boldsymbol{x}),$$

3. B can be monitored by the *Out-of-the-Bag* error: for each observation $\boldsymbol{z}_t = (y_t, \boldsymbol{x}_t')'$, construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which $\boldsymbol{z}_t$ did not appear.

Random Forests

Example: California Housing

Test average absolute error as a function of the number of trees



Sieves and Neural Networks

Sieve Spaces

Motivation

- Approximation of nonlinear unknown functions by the method of sieves of Grenander (1981).

Sieve Spaces

Motivation

- Approximation of nonlinear unknown functions by the method of sieves of Grenander (1981).
- Unlike the kernel estimator which is a local estimator, the sieve is a global estimator in the sense that it estimates the function of interest over its entire domain in a single step.

Sieve Spaces

Motivation

- Approximation of nonlinear unknown functions by the method of sieves of Grenander (1981).
- Unlike the kernel estimator which is a local estimator, the sieve is a global estimator in the sense that it estimates the function of interest over its entire domain in a single step.
- Sieve estimators have several advantages:

Sieve Spaces

Motivation

- Approximation of nonlinear unknown functions by the method of sieves of Grenander (1981).
- Unlike the kernel estimator which is a local estimator, the sieve is a global estimator in the sense that it estimates the function of interest over its entire domain in a single step.
- Sieve estimators have several advantages:
  1. Computational easiness

Sieve Spaces

Motivation

- Approximation of nonlinear unknown functions by the method of sieves of Grenander (1981).
- Unlike the kernel estimator which is a local estimator, the sieve is a global estimator in the sense that it estimates the function of interest over its entire domain in a single step.
- Sieve estimators have several advantages:
  1. Computational easiness
  2. Easy to impose restrictions

Sieve Spaces

Motivation

- Approximation of nonlinear unknown functions by the method of sieves of Grenander (1981).
- Unlike the kernel estimator which is a local estimator, the sieve is a global estimator in the sense that it estimates the function of interest over its entire domain in a single step.
- Sieve estimators have several advantages:
  1. Computational easiness
  2. Easy to impose restrictions
  3. Much easier solution under endogeneity

Sieve Spaces

Definition

- Consider the general nonlinear, semi-parametric model:

$$\begin{aligned} y_i &= m_0(\boldsymbol{x}_i) + u_i \\ &= \underbrace{\boldsymbol{\beta}_0' \boldsymbol{x}_i}_{\text{parametric component}} + \underbrace{h_0(\boldsymbol{x}_i)}_{\text{nonparametric component}} + u_i, \end{aligned}$$

where $\{u_i\}$, $i = 1, \ldots, n$ is a sequence of random disturbances and $\boldsymbol{x}_i \in \mathbb{R}^p$ is a vector of covariates.

Sieve Spaces

Definition

- Consider the general nonlinear, semi-parametric model:

$$\begin{aligned} y_i &= m_0(\boldsymbol{x}_i) + u_i \\ &= \underbrace{\boldsymbol{\beta}_0' \boldsymbol{x}_i}_{\text{parametric component}} + \underbrace{h_0(\boldsymbol{x}_i)}_{\text{nonparametric component}} + u_i, \end{aligned}$$

  where $\{u_i\}$, $i = 1, \ldots, n$ is a sequence of random disturbances and $\boldsymbol{x}_i \in \mathbb{R}^p$ is a vector of covariates.

- Let's assume for now that $\mathbb{E}(u_i|\boldsymbol{x}_i) = 0$ and $\mathbb{E}(u_i^2|\boldsymbol{x_i}) = \sigma^2(\boldsymbol{x}_i) < \infty$, for all $i = 1, \ldots n$.

Sieve Spaces

Definition

- Consider the general nonlinear, semi-parametric model:

$$\begin{aligned} y_i &= m_0(\boldsymbol{x}_i) + u_i \\ &= \underbrace{\boldsymbol{\beta}_0' \boldsymbol{x}_i}_{\text{parametric component}} + \underbrace{h_0(\boldsymbol{x}_i)}_{\text{nonparametric component}} + u_i, \end{aligned}$$

  where $\{u_i\}$, $i = 1, \ldots, n$ is a sequence of random disturbances and $\boldsymbol{x}_i \in \mathbb{R}^p$ is a vector of covariates.

- Let's assume for now that $\mathbb{E}(u_i|\boldsymbol{x}_i) = 0$ and $\mathbb{E}(u_i^2|\boldsymbol{x_i}) = \sigma^2(\boldsymbol{x}_i) < \infty$, for all $i = 1, \ldots n$.
- The goal is to estimate the vector of parameters $\boldsymbol{\theta}_0 = (\boldsymbol{\beta}_0', h_0)' \in \mathcal{B} \times \mathcal{H} \equiv \Theta$, where $\mathcal{B}$ denotes a finite-dimensional compact parameter space and $\mathcal{H}$ is an infinite-dimensional parameter space.

Sieve Spaces

Definition

- Suppose that the infinite-dimensional space Θ is endowed with a (pseudo-)metric d.

Sieve Spaces

Definition

- Suppose that the infinite-dimensional space Θ is endowed with a (pseudo-)metric d.
- A typical semi-nonparametric econometric model specifies that there is a population criterion function $Q : \Theta \to \mathbb{R}$ which is uniquely maximized (or minimized) at a (pseudo-)true parameter $\boldsymbol{\theta}_0$.

Sieve Spaces

Definition

- Suppose that the infinite-dimensional space Θ is endowed with a (pseudo-)metric d.
- A typical semi-nonparametric econometric model specifies that there is a population criterion function $Q : \Theta \to \mathbb{R}$ which is uniquely maximized (or minimized) at a (pseudo-)true parameter $\boldsymbol{\theta}_0$.
- When Θ is infinite-dimensional and possibly not compact with respect to the (pseudo-)metric d, maximizing and empirical criterion function $\widehat{Q}_n$ over Θ may not be well-defined.

Sieve Spaces

Definition

- We say that the optimization problem is well-posed if for sequences $\{\boldsymbol{\theta}_k\} \in \Theta$ such that $Q(\boldsymbol{\theta}_0) - Q(\theta_k) \to 0$, then $d(\boldsymbol{\theta}_0, \boldsymbol{\theta}_k) \to 0$.

Sieve Spaces

Definition

- We say that the optimization problem is well-posed if for sequences $\{\boldsymbol{\theta}_k\} \in \Theta$ such that $Q(\boldsymbol{\theta}_0) - Q(\theta_k) \to 0$, then $d(\boldsymbol{\theta}_0, \boldsymbol{\theta}_k) \to 0$.
- The problem is ill-posed if for sequences $\{\boldsymbol{\theta}_k\} \in \Theta$ such that $Q(\boldsymbol{\theta}_0) - Q(\theta_k) \to 0$, but $d(\boldsymbol{\theta}_0, \boldsymbol{\theta}_k) \nrightarrow 0$.

Sieve Spaces

Definition

Key Idea

- The method of sieves provides a general approach to resolve difficulties associated with maximizing $\widehat{Q}_n$ over Θ by maximizing $\widehat{Q}_n$ over a sequence of approximating spaces Θ_n, called *Sieves*.

Sieve Spaces

Definition

Key Idea

- The method of sieves provides a general approach to resolve difficulties associated with maximizing $\widehat{Q}_n$ over Θ by maximizing $\widehat{Q}_n$ over a sequence of approximating spaces Θ_n, called *Sieves*.
- These spaces are less complex than Θ but are **dense** in Θ.

Sieve Spaces

Definition

Key Idea

- The method of sieves provides a general approach to resolve difficulties associated with maximizing $\widehat{Q}_n$ over Θ by maximizing $\widehat{Q}_n$ over a sequence of approximating spaces Θ_n, called *Sieves*.
- These spaces are less complex than Θ but are **dense** in Θ.
- Popular sieves are typically compact, nondecreasing ($\Theta_n \subseteq \Theta_{n+1} \subseteq \cdots \subseteq \Theta$) and are such that for any element $\boldsymbol{\theta} \in \Theta$ there exists as element $\pi_n \boldsymbol{\theta} \in \Theta_n$ satisfying $d(\boldsymbol{\theta}, \pi_n \boldsymbol{\theta}) \to 0$ as $n \to \infty$, where the notation π_n can be regarded as a projection mapping from Θ to Θ_n.

Sieve Spaces

Definition

- The *approximate sieve extremum estimate*, denoted by $\widehat{\boldsymbol{\theta}}_n$, is defined as an approximate maximizer of $\widehat{Q}_n(\boldsymbol{\theta})$ over the sieve space Θ_n:

$$\widehat{Q}_n(\widehat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta} \in \Theta_n} \widehat{Q}_n(\boldsymbol{\theta}) - O_p(\eta_n),$$

with $\eta_n \to 0$ as $n \to \infty$.

Sieve Spaces

Definition

- The *approximate sieve extremum estimate*, denoted by $\widehat{\boldsymbol{\theta}}_n$, is defined as an approximate maximizer of $\widehat{Q}_n(\boldsymbol{\theta})$ over the sieve space Θ_n:

$$\widehat{Q}_n(\widehat{\boldsymbol{\theta}}_n) \geq \sup_{\boldsymbol{\theta} \in \Theta_n} \widehat{Q}_n(\boldsymbol{\theta}) - O_p(\eta_n),$$

  with $\eta_n \to 0$ as $n \to \infty$.
- When $\eta_n = 0$ we call the estimator the *exact sieve extremum estimate*.

Sigmoid Artificial Neural Network

The single hidden layer Sigmoid Artificial Neural Network (sANN) sieve is defined as

$$\mathsf{sANN}(J_n) = \left\{ \gamma_0 + \sum_{j=1}^{J_n} \alpha_j S(\boldsymbol{\gamma}_j' \boldsymbol{x} + \gamma_{0,j}) : \boldsymbol{\gamma}_j \in \mathbb{R}^p, \alpha_j, \gamma_0, \gamma_{0,j} \in \mathbb{R} \right\}.$$

- $S : \mathbb{R} \to R$ is a sigmoid "activation" function, i.e., a bounded nondecreasing function such that $\lim_{u \to -\infty} S(u) = 0$ and $\lim_{u \to \infty} S(u) = 1$.

Nonlinear Sieve Spaces

Single Hidden Layer Neural Networks

- Some popular sigmoid functions:
  - Heaviside: $S(u) = 1(u \geq 0)$;
  - Logistic: $S(u) = 1/[1 + \exp(-u)]$;
  - Hyperbolic tangent:
    $S(u) = [\exp(u) - \exp(-u)]/[\exp(u) + \exp(-u)]$;
  - Gaussian sigmoid: $S(u) = (2\pi)^{-1/2} \int_{-\infty}^{u} \exp(-y^2/2)dy$;
  - Cosine squasher:
    $S(u) = [1 + \cos(u + 3\pi/2)]/21(|u| \leq \pi/2) + 1(u > \pi/2)$.
  - ReLU: $S(u) = u\mathsf{I}(u > 0)$

Empirical Example: Equity Premium Forecasting with ML Methods

Gu, Shihao, Bryan Kelly and Dacheng Xiu (2018). *Empirical Asset Pricing via Machine Learning*. Working paper available at SSRN id 3159577.

Main Idea

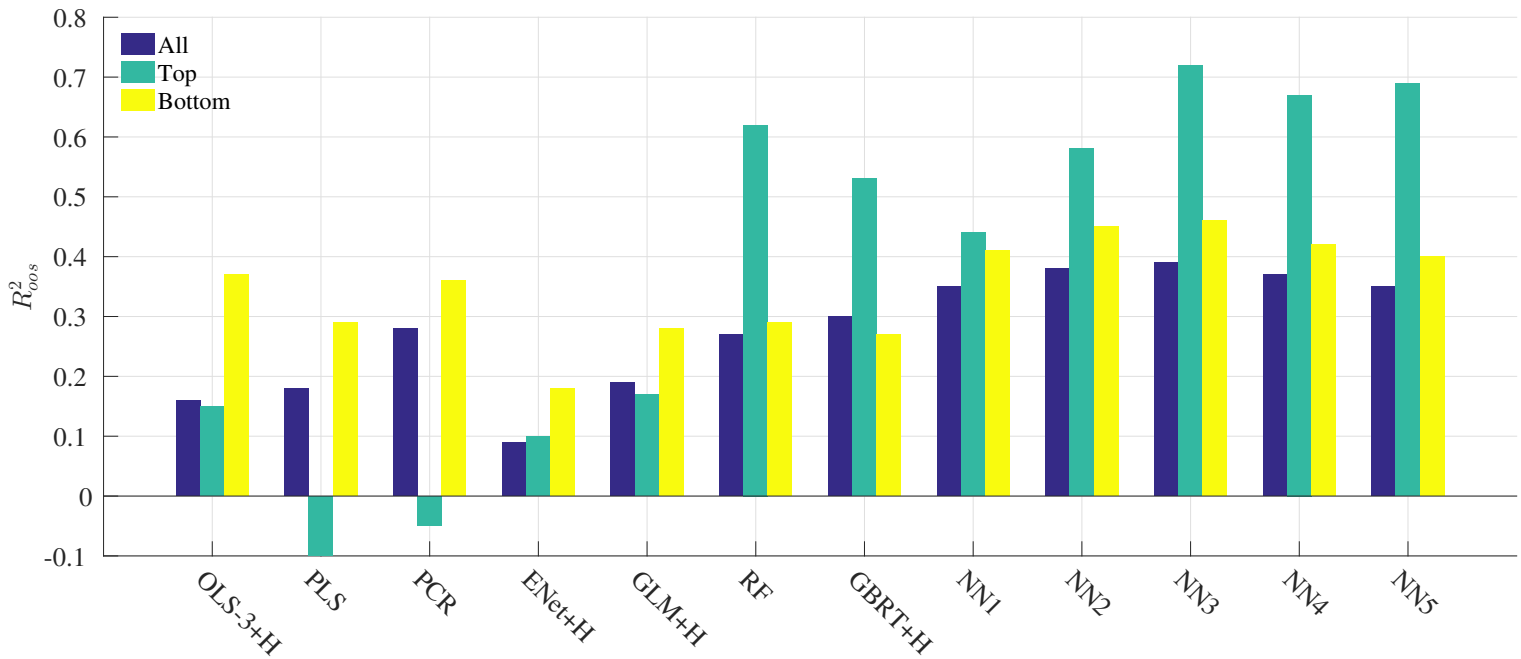
- Model:

$$\boxed{\begin{aligned} r_{t,t+1} &= \mathbb{E}_t(r_{i,t+1}) + \epsilon_{i,t+1} \\ \mathbb{E}_t(r_{i,t+1}) &= g(\boldsymbol{z}_{i,t}) \end{aligned}}$$

  where $\boldsymbol{z}_{it}$ is a large vector of predictors:
  - 91 firm characteristics (61 of which are updated annually, 13 updated quarterly and 20 updated monthly);
  - 74 industry dummies
  - 8 macroeconomic predictors
  - Interactions between macro factors and firm characteristics.
- 30,000 stocks over a sample starting in March 1957 and ending in December 2016 (60 years).
- ML methods: linear regression, restricted linear regression (FF factors), partial least squares (PLS), principal component regression (PCR), generalized linear model (GLM), random forest (RF), Boosted Trees (GBRT), neural networks with 1 to 5 layers (NN1–NN5).

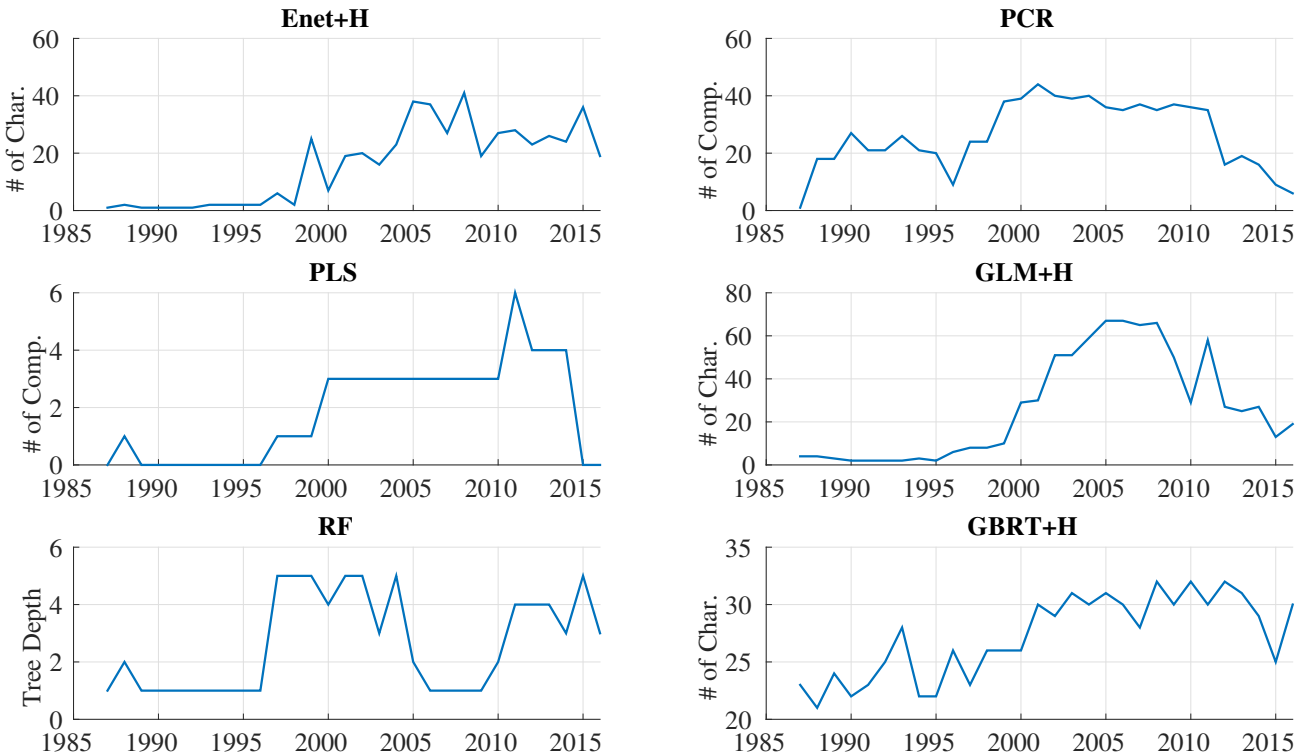
Results

	OLS +H	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
All	-4.60	0.16	0.18	0.28	0.09	0.19	0.27	0.30	0.35	0.38	0.39	0.37	0.35
Top 1000	-14.21	0.15	-0.10	-0.05	0.10	0.17	0.62	0.53	0.44	0.58	0.72	0.67	0.69
Bottom 1000	-2.13	0.37	0.29	0.36	0.18	0.28	0.29	0.27	0.41	0.45	0.46	0.42	0.40



Note: In this table, we report monthly R^2_{oos} for the entire panel of stocks using OLS with all variables (OLS), OLS using only size, book-to-market, and momentum (OLS-3), PLS, PCR, elastic net (ENet), generalize linear model (GLM), random forest (RF), gradient boosted regression trees (GBRT), and neural networks with one to five layers (NN1–NN5). "+H" indicates the use of Huber loss instead of the l_2 loss. We also report these R^2_{oos} within subsamples that include only the top 1,000 stocks or bottom 1,000 stocks by market value. The lower panel provides a visual comparison of the R^2_{oos} statistics in the table (omitting OLS due to its large negative values).

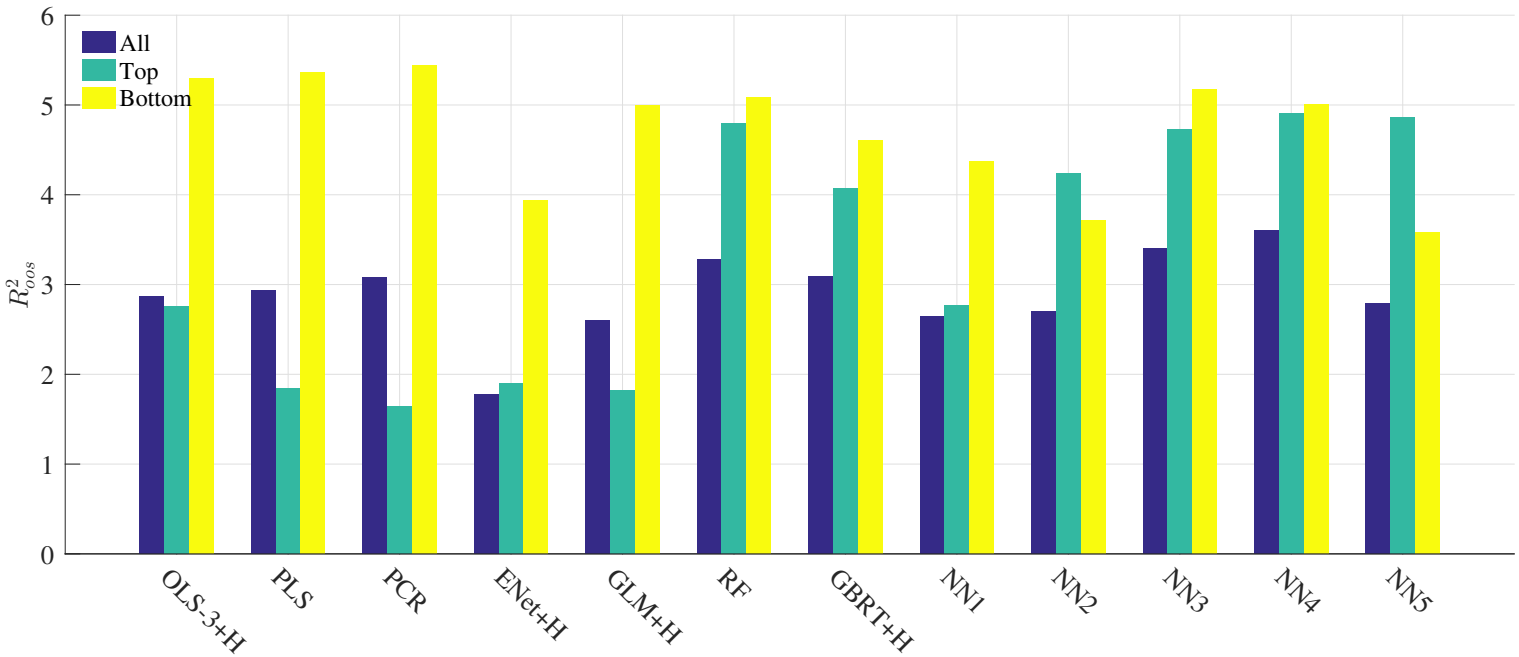
Results



Note: This figure demonstrates the model complexity for elastic net (ENet), PCR, PLS, generalized linear model with group lasso (GLM), random forest (RF) and gradient boosted regression trees (GBRT) in each training sample of our 30-year recursive out-of-sample analysis. For ENet and GLM we report the number of features selected to have non-zero coefficients; for PCR and PLS we report the number of selected components; for RF we report the average tree depth; and for GBRT we report the number of distinct characteristics entering into the trees.

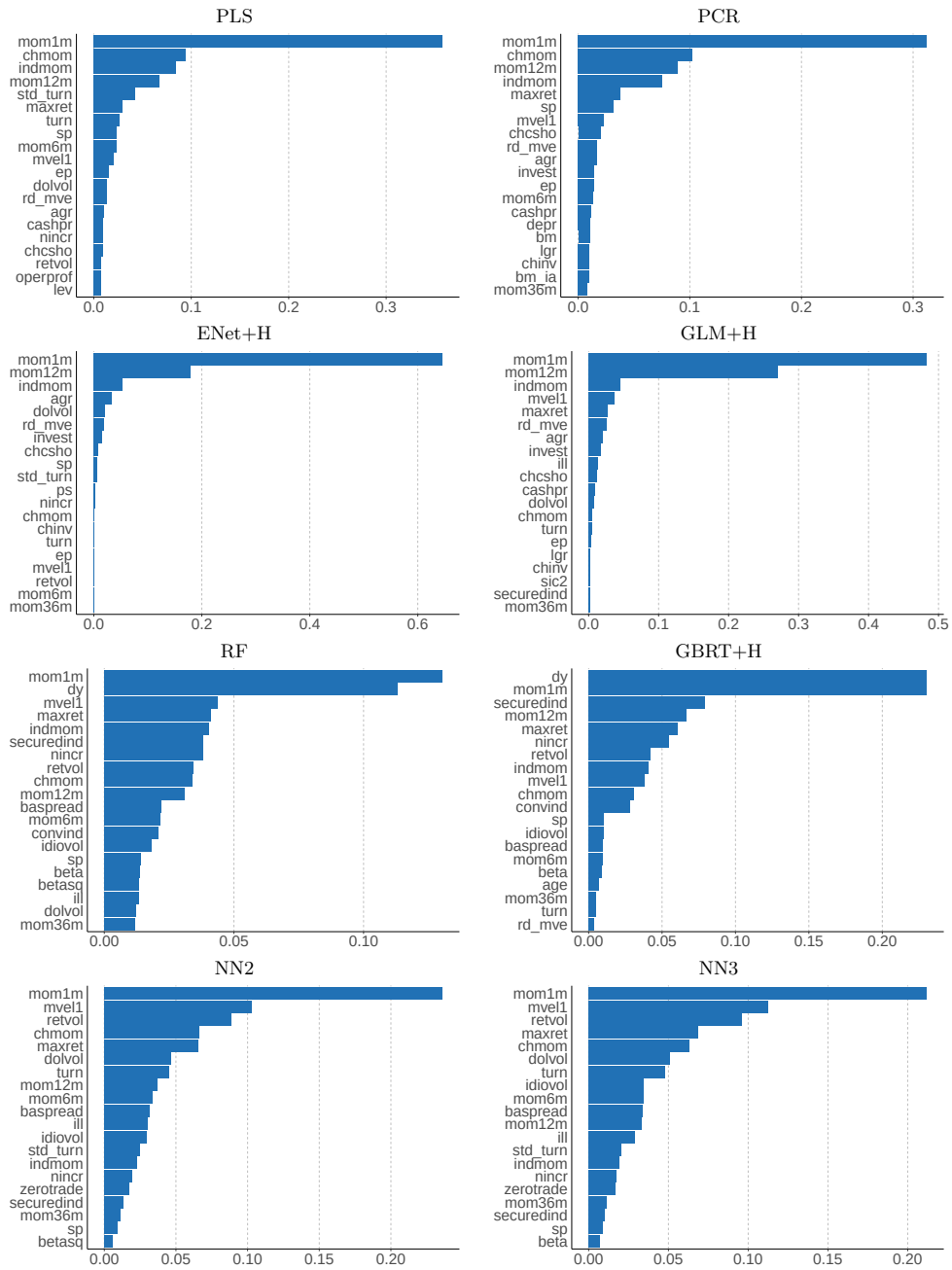
Results

	OLS +H	OLS-3 +H	PLS	PCR	ENet +H	GLM +H	RF	GBRT +H	NN1	NN2	NN3	NN4	NN5
All	-34.86	2.87	2.93	3.08	1.78	2.60	3.28	3.09	2.64	2.70	3.40	3.60	2.79
Top	-54.86	2.77	1.84	1.64	1.90	1.82	4.80	4.07	2.77	4.24	4.73	4.91	4.86
Bottom	-19.22	5.30	5.36	5.44	3.94	5.00	5.08	4.61	4.37	3.72	5.17	5.01	3.58



Note: Annual return forecasting R^2_{oos} (see Table 1 notes).

Results


PLS
mom1m
chmom
indmom
mom12m
std_turn
maxret
turn
sp
mom6m
mvel1
ep
dolvol
rd_mve
agr
cashpr
nincr
chcsho
retvol
operprof
lev
0.0
0.1
0.2
0.3
PCR
mom1m
chmom
mom12m
indmom
maxret
sp
mvel1
chcsho
rd_mve
agr
invest
ep
mom6m
cashpr
depr
bm
lgr
chinv
bm_ia
mom36m
0.0
0.1
0.2
0.3
ENet+H
mom1m
mom12m
indmom
agr
dolvol
rd_mve
invest
chcsho
sp
std_turn
ps
nincr
chmom
chinv
turn
ep
mvel1
retvol
mom6m
mom36m
0.0
0.2
0.4
0.6
GLM+H
mom1m
mom12m
indmom
mvel1
maxret
rd_mve
agr
invest
ill
chcsho
cashpr
dolvol
chmom
turn
ep
lgr
chinv
sic2
securedind
mom36m
0.0
0.1
0.2
0.3
0.4
0.5
RF
mom1m
dy
mvel1
maxret
indmom
securedind
nincr
retvol
chmom
mom12m
baspread
mom6m
convind
idiovol
sp
beta
betasq
ill
dolvol
mom36m
0.00
0.05
0.10
GBRT+H
dy
mom1m
securedind
mom12m
maxret
nincr
retvol
indmom
mvel1
chmom
convind
sp
idiovol
baspread
mom6m
beta
age
mom36m
turn
rd_mve
0.00
0.05
0.10
0.15
0.20
NN2
mom1m
mvel1
retvol
chmom
maxret
dolvol
turn
mom12m
mom6m
baspread
ill
idiovol
std_turn
indmom
nincr
zerotrade
securedind
mom36m
sp
betasq
0.00
0.05
0.10
0.15
0.20
NN3
mom1m
mvel1
retvol
maxret
chmom
dolvol
turn
idiovol
mom6m
baspread
mom12m
ill
std_turn
indmom
nincr
zerotrade
mom36m
securedind
sp
beta
0.00
0.05
0.10
0.15
0.20