Lecture 13: Decision Trees and Random Forests

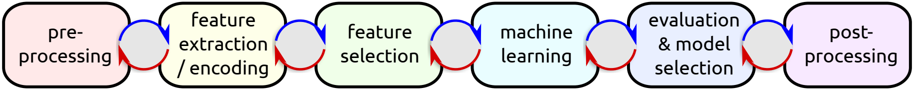
Machine Learning, Summer Term 2019

July 1, 2019

Michael Tangermann Frank Hutter Marius Lindauer

University of Freiburg

The Big Picture



- Lecture 1: overview
- Lecture 2-6: linear methods
- Lecture 7-9: algorithm-independent principles
- Lectures 10-15: nonlinear methods
  - Lecture 10-12: kernel-based methods
  - Lectures 13-14: tree-based methods and ensembles
  - Lecture 15: neural networks

Lecture Overview

1. Decision and Regression Trees
   - Regression Trees
   - Classification Trees (= Decision Trees)

2. Bagging

3. Random Forests

Lecture Overview

1. Decision and Regression Trees
   - Regression Trees
   - Classification Trees (= Decision Trees)

2. Bagging

3. Random Forests

Motivation for Trees and Forests

For many applications, random forests are the best off-the-shelf model

Motivation for Trees and Forests

For many applications, random forests are the best off-the-shelf model

- Trees: easy to interpret

Motivation for Trees and Forests

For many applications, random forests are the best off-the-shelf model

- Trees: easy to interpret
- Directly handle categorical features

Motivation for Trees and Forests

For many applications, random forests are the best off-the-shelf model

- Trees: easy to interpret
- Directly handle categorical features
- Scalable to many data points (they are fast)

Motivation for Trees and Forests

For many applications, random forests are the best off-the-shelf model

- Trees: easy to interpret
- Directly handle categorical features
- Scalable to many data points (they are fast)
- Scalable to many features (they are automated feature selectors)

Motivation for Trees and Forests

For many applications, random forests are the best off-the-shelf model

- Trees: easy to interpret
- Directly handle categorical features
- Scalable to many data points (they are fast)
- Scalable to many features (they are automated feature selectors)
- Random forests: robust performance even for small datasets

Motivation for Trees and Forests

For many applications, random forests are the best off-the-shelf model

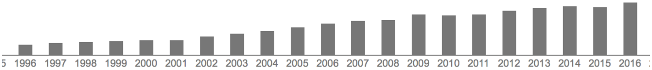
- Trees: easy to interpret
- Directly handle categorical features
- Scalable to many data points (they are fast)
- Scalable to many features (they are automated feature selectors)
- Random forests: robust performance even for small datasets
- Random forests: robust to their hyperparameter settings
  - In contrast to, e.g., SVMs or neural networks

Trees and Forests are Extremely Popular Models

Leo Breiman 1928-2005

Classification and Regression Trees

Authors	Leo Breiman, Jerome H Friedman, Richard A Olshen, Charles J Stone
Publication date	1999/5
Publisher	CRC Press, New York
Total citations	Cited by 34504



Trees and Forests are Extremely Popular Models

Leo Breiman 1928-2005

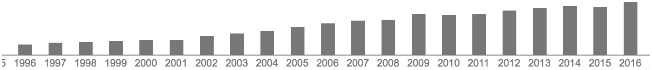
Classification and Regression Trees

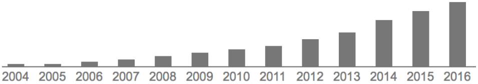
Authors Leo Breiman, Jerome H Friedman, Richard A Olshen, Charles J Stone

Publication date 1999/5

Publisher CRC Press, New York

Total citations Cited by 34504

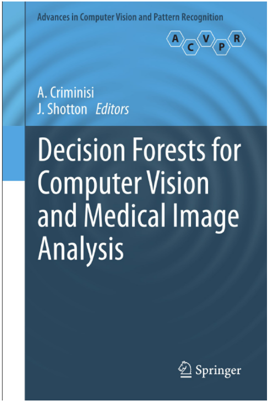




Scholar articles Random forests
L Breiman - Machine learning, 2001
Cited by 29053 - Related articles - All 68 versions

Acknowledgement

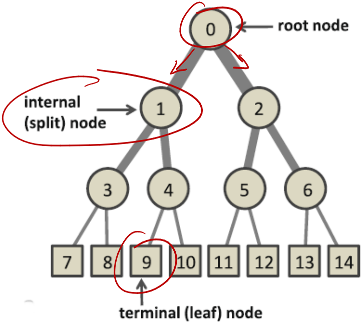
Most visualizations in this lecture are taken from this excellent book by Criminisi et. al (2013):



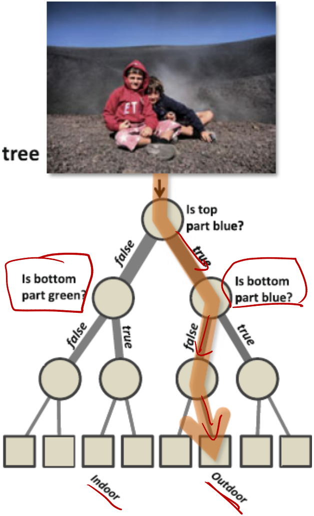
PDF of entire book available from university machines:
`http://link.springer.com/book/10.1007/978-1-4471-4929-3`

Decision and Regression Trees – General Idea

A general tree structure



A decision tree



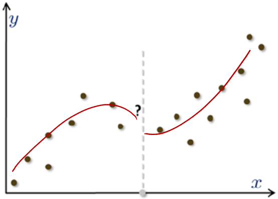
Lecture Overview

1. Decision and Regression Trees
   - Regression Trees
   - Classification Trees (= Decision Trees)

2. Bagging

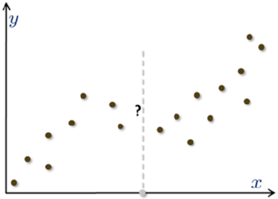
3. Random Forests

Regression Trees

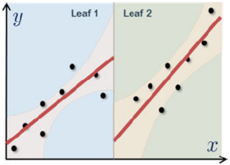


Idea: fit simple model to subset of the data

Regression Trees

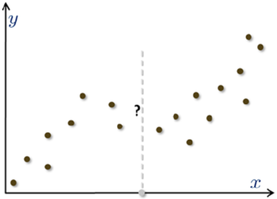


Idea: fit simple model to subset of the data

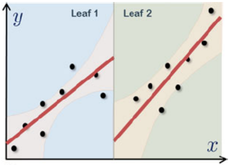


probab. model

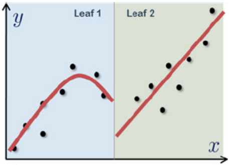
Regression Trees



Idea: fit simple model to subset of the data

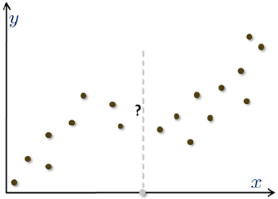


probab. model

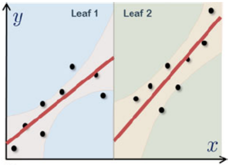


polynomial

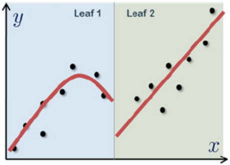
Regression Trees



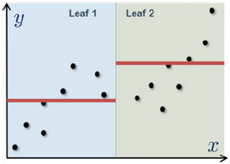
Idea: fit simple model to subset of the data



probab. model

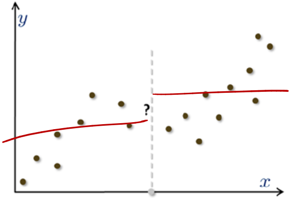


polynomial

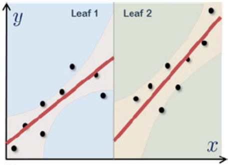


constant (standard)

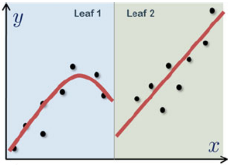
Regression Trees



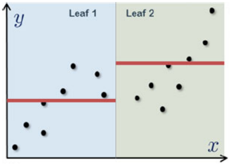
Idea: fit simple model to subset of the data



probab. model



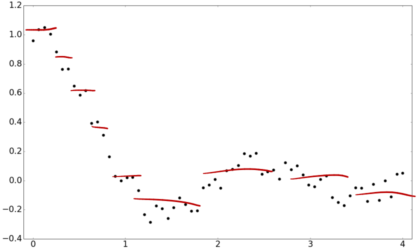
polynomial



constant (standard)

We will only cover the standard case of constant leaf predictions

How to Split the Data

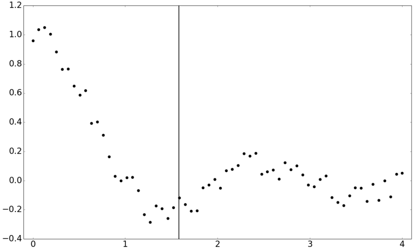


- Constant models in the leafs: $\hat{f}_{left}$ and $\hat{f}_{right}$
- Greedily minimize sum of squared errors in the two children

$$x_{\mathsf{split}} \in \arg\min_{x} \left(\sum_{x_i \leq x} \left(y_i - \hat{f}_{left} \right)^2 + \sum_{x_i > x} \left(y_i - \hat{f}_{right} \right)^2 \right)$$

- Recursively split subsets

How to Split the Data

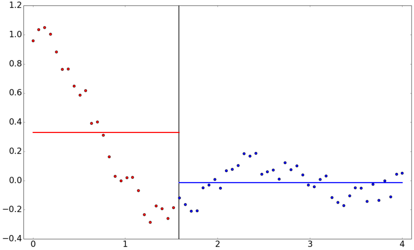


- Constant models in the leafs: $\hat{f}_{left}$ and $\hat{f}_{right}$
- Greedily minimize sum of squared errors in the two children

$$x_{\text{split}} \in \arg\min_{x} \left(\sum_{x_i \leq x} \left(y_i - \hat{f}_{left} \right)^2 + \sum_{x_i > x} \left(y_i - \hat{f}_{right} \right)^2 \right)$$

- Recursively split subsets

How to Split the Data

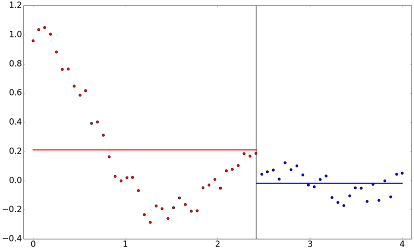


- Constant models in the leafs: $\hat{f}_{left}$ and $\hat{f}_{right}$
- Greedily minimize sum of squared errors in the two children

$$x_{\text{split}} \in \arg\min_{x} \left(\sum_{x_i \le x} \left(y_i - \hat{f}_{left} \right)^2 + \sum_{x_i > x} \left(y_i - \hat{f}_{right} \right)^2 \right)$$

- Recursively split subsets

How to Split the Data

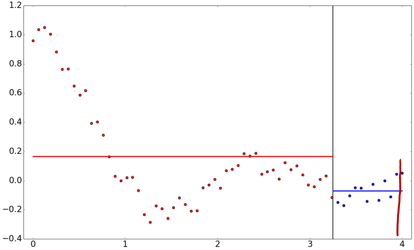


- Constant models in the leafs: $\hat{f}_{left}$ and $\hat{f}_{right}$
- Greedily minimize sum of squared errors in the two children

$$x_{\text{split}} \in \arg\min_{x} \left(\sum_{x_i \leq x} \left(y_i - \hat{f}_{left}\right)^2 + \sum_{x_i > x} \left(y_i - \hat{f}_{right}\right)^2 \right)$$

- Recursively split subsets

How to Split the Data

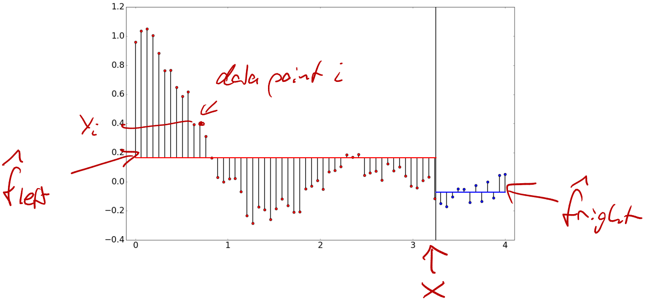


- Constant models in the leafs: $\hat{f}_{left}$ and $\hat{f}_{right}$
- Greedily minimize sum of squared errors in the two children

$$x_{\text{split}} \in \arg\min_{x} \left(\sum_{x_i \leq x} \left(y_i - \hat{f}_{left} \right)^2 + \sum_{x_i > x} \left(y_i - \hat{f}_{right} \right)^2 \right)$$

- Recursively split subsets

How to Split the Data

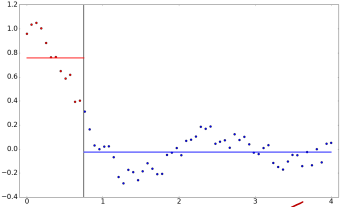


- Constant models in the leafs: $\hat{f}_{left}$ and $\hat{f}_{right}$
- Greedily minimize sum of squared errors in the two children

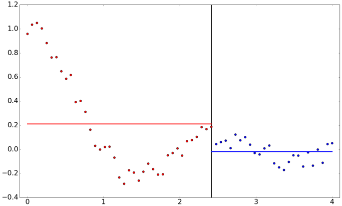
$$x_{\text{split}} \in \arg\min_{x} \boxed{\left(\sum_{x_i \leq x} \left(y_i - \hat{f}_{left} \right)^2 + \sum_{x_i > x} \left(y_i - \hat{f}_{right} \right)^2 \right)}$$

- Recursively split subsets

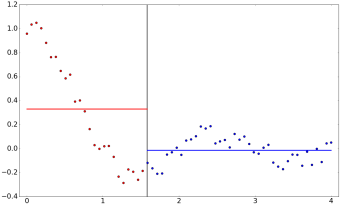
Let’s vote: Which of These Splits is the Best?



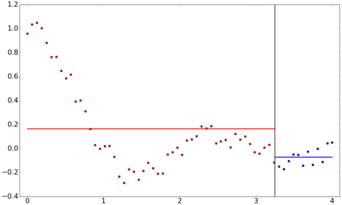
★ (black)



★ (red)

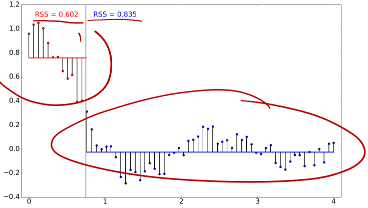


★ (yellow)

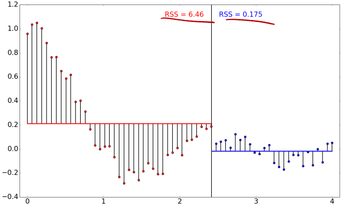


★ (blue)

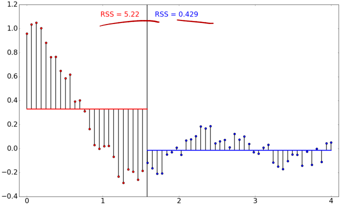
Let's vote: Which of These Splits is the Best?



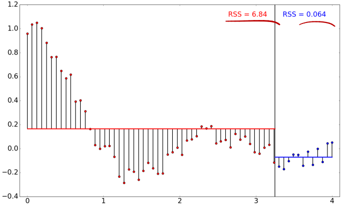
★ (black)



★ (red)



★ (yellow)



★ (blue)

CART algorithm [Breiman et. al 1984]

CART = Classification And Regression Trees

- Input: $\mathbf{X}$, $\mathbf{y}$ (and hyperparameters `max_depth`, `min_leaf`)
- Check whether data should be split further; otherwise return leaf node
- Find best split value for each feature
- Choose best combination of split feature and split value
- Split data into left and right accordingly: $(\mathbf{X}_l, \mathbf{y}_l)$ and $(\mathbf{X}_r, \mathbf{y}_r)$
- Save split feature and value, and pointer to two new subtrees to be built recursively:

(CART($\mathbf{X}_l$,$\mathbf{y}_l$,`max_depth`-1, `min_leaf`),
CART($\mathbf{X}_r$,$\mathbf{y}_r$,`max_depth`-1, `min_leaf`))

Visualization of CART Algorithm For Regression: Training

feature1	feature2	feature3	target
false	2	red	3.7
false	2.5	blue	20
true	5.5	red	2.1
false	5.5	blue	25
false	5	red	1.2
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

Visualization of CART Algorithm For Regression: Training

feature1	feature2	feature3	target
false	2	red	3.7
false	2.5	blue	20
true	5.5	red	2.1
false	5.5	blue	25
false	5	red	1.2
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

$feature_3 \in \{red\}$

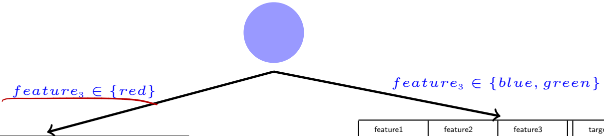
feature1	feature2	feature3	target
false	2	red	3.7
true	5.5	red	2.1
false	5	red	1.2

$feature_3 \in \{blue, green\}$

feature1	feature2	feature3	target
false	2.5	blue	20
false	5.5	blue	25
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

Visualization of CART Algorithm For Regression: Training

- In each internal node: only store split criterion used



feature1	feature2	feature3	target
false	2	red	3.7
true	5.5	red	2.1
false	5	red	1.2

feature1	feature2	feature3	target
false	2.5	blue	20
false	5.5	blue	25
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

Visualization of CART Algorithm For Regression: Training

- In each internal node: only store split criterion used

$feature_3 \in \{red\}$

feature1	feature2	feature3	target
false	2	red	3.7
true	5.5	red	2.1
false	5	red	1.2

$feature_3 \in \{blue, green\}$

feature1	feature2	feature3	target
false	2.5	blue	20
false	5.5	blue	25
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

$feature_2 \leq 3.5$

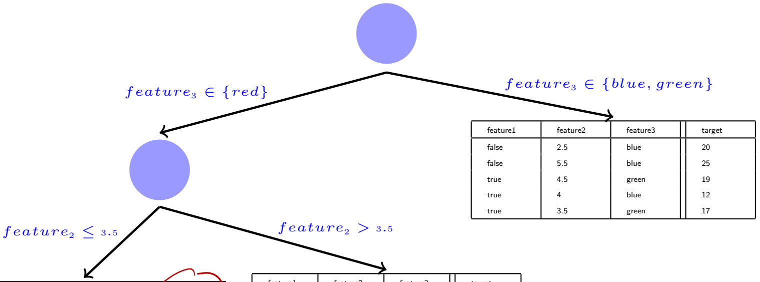
feature1	feature2	feature3	target
false	2	red	3.7

$feature_2 > 3.5$

feature1	feature2	feature3	target
true	5.5	red	2.1
false	5	red	1.2

Visualization of CART Algorithm For Regression: Training

- In each internal node: only store split criterion used



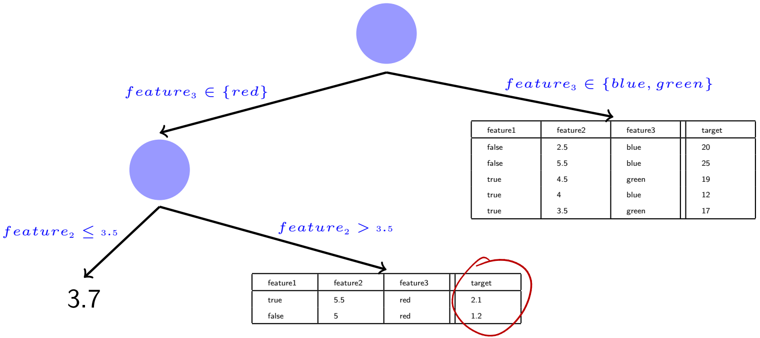
feature1	feature2	feature3	target
false	2.5	blue	20
false	5.5	blue	25
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

feature1	feature2	feature3	target
false	2	red	3.7

feature1	feature2	feature3	target
true	5.5	red	2.1
false	5	red	1.2

Visualization of CART Algorithm For Regression: Training

- In each internal node: only store split criterion used
- In each leaf: store mean of targets

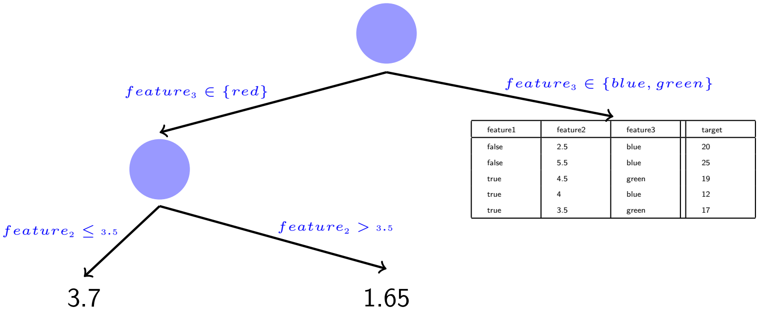


feature1	feature2	feature3	target
false	2.5	blue	20
false	5.5	blue	25
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

feature1	feature2	feature3	target
true	5.5	red	2.1
false	5	red	1.2

Visualization of CART Algorithm For Regression: Training

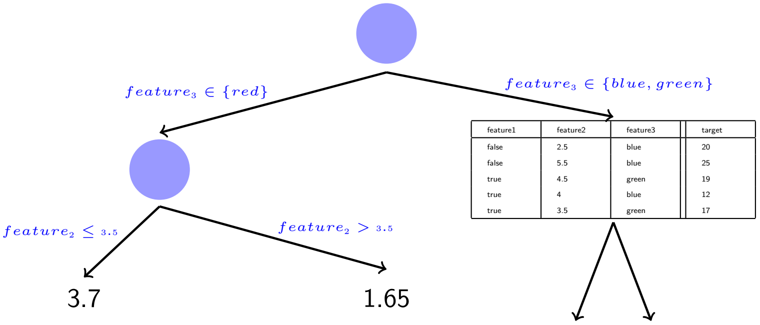
- In each internal node: only store split criterion used
- In each leaf: store mean of targets



feature1	feature2	feature3	target
false	2.5	blue	20
false	5.5	blue	25
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

Visualization of CART Algorithm For Regression: Training

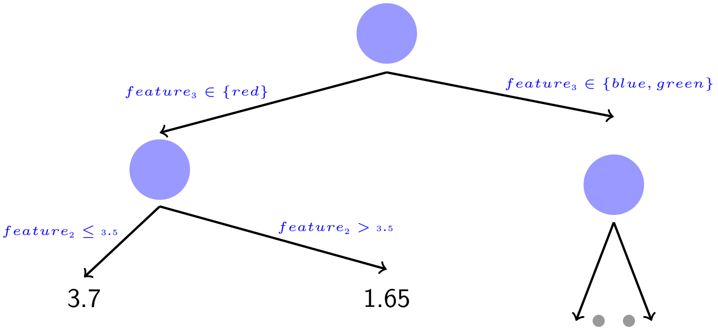
- In each internal node: only store split criterion used
- In each leaf: store mean of targets



feature1	feature2	feature3	target
false	2.5	blue	20
false	5.5	blue	25
true	4.5	green	19
true	4	blue	12
true	3.5	green	17

Visualization of CART Algorithm For Regression: Training

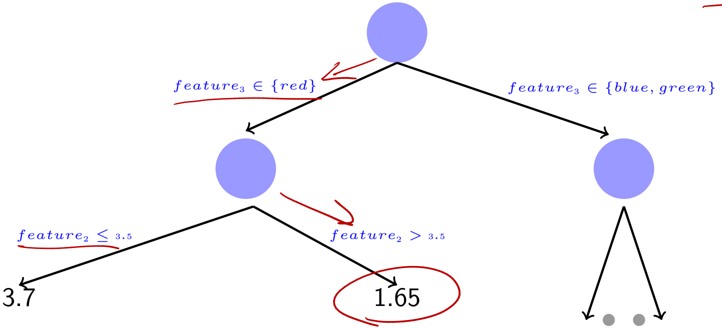
- In each internal node: only store split criterion used
- In each leaf: store mean of targets



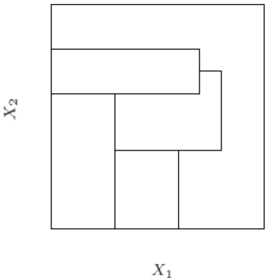
Visualization of CART Algorithm For Regression: Prediction for New Inputs

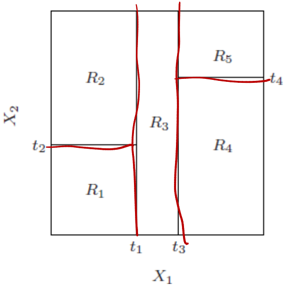
E.g $x_{n+1} = (true, 4.7, red)$

- Walk down tree, return mean target stored in leaf $\Rightarrow 1.65$



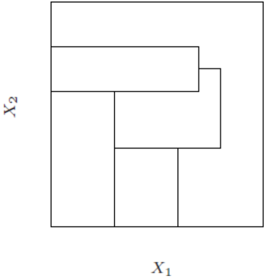
Hierarchichal Binary Splits

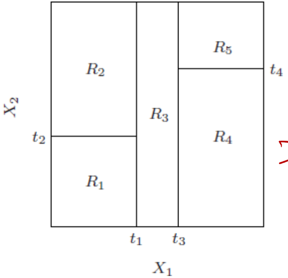




- Hierarchical splits (as on the right side) are easy to represent
  - Splits as in the left figure would be harder

Hierarchichal Binary Splits

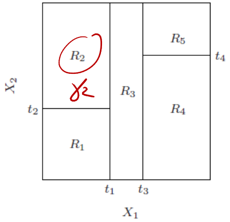




- Hierarchical splits (as on the right side) are easy to represent
  - Splits as in the left figure would be harder
- We could also use k-ary splits
  - But every k-ary split can be seen as a sequence of binary splits
  - Binary splits are faster and often yield better predictions

Formal Notation for Tree Predictions (1/2)

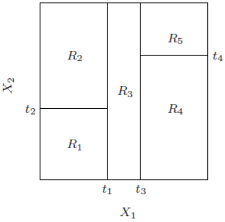
After the intuitive treatment so far, we now write down tree predictions formally.



- Trees partition the input space $\mathcal{X}$ into regions $R_1, \ldots, R_J$ associated with their leaves
- Each leaf j has a simple model; here the constant γ_j

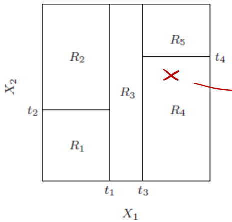
Formal Notation for Tree Predictions (1/2)

After the intuitive treatment so far, we now write down tree predictions formally.



- Trees partition the input space $\mathcal{X}$ into regions $R_1, \ldots, R_J$ associated with their leaves
- Each leaf j has a simple model; here the constant γ_j
- Mathematically, a decision/regression tree T with constant leaf predictions is fully specified by $\langle R_1, \ldots, R_J, \gamma_1, \ldots, \gamma_J \rangle$

Formal Notation for Tree Predictions (2/2)



Tree Prediction

The prediction of a decision/regression tree with parameters $\Theta = \langle R_1, \ldots, R_J, \gamma_1, \ldots, \gamma_J \rangle$ is

$$T(x_i, \Theta) = \sum_{j=1}^{J} \gamma_j \mathbb{I}(x \in R_j) = \gamma_4 \cdot 1 = \gamma_4$$

Here and throughout, $\mathbb{I}$ is the indicator function

$$\mathbb{I}(a) := \begin{cases} 1 & , \quad \text{if } a = true \\ 0 & , \quad \text{otherwise} \end{cases}$$

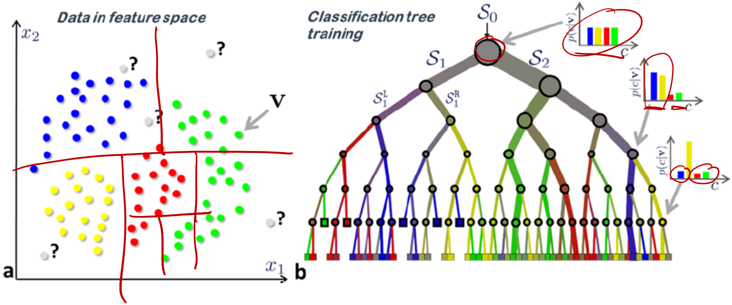
Lecture Overview

1. Decision and Regression Trees
   - Regression Trees
   - Classification Trees (= Decision Trees)

2. Bagging

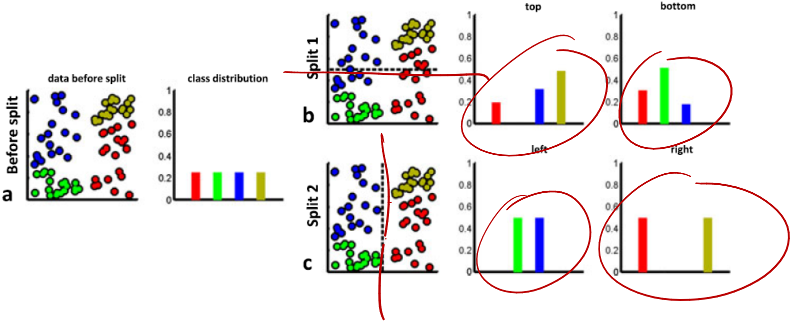
3. Random Forests

Classification Trees (= Decision Trees)



- Conceptually the same as for regression
- Leaf model: majority vote; probability of data in the leaf
- Split criterion: Gini index; variance reduction; information gain

Classification Splitting Criteria



Intuitively, which of these splits is better?

★ Split 1 (into top and bottom)

★ Split 2 (into left and right)

Information Entropy

Definition: Entropy

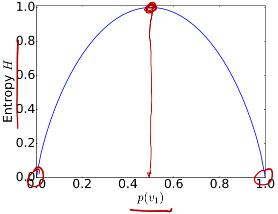
The entropy of a discrete random variable V with K possible outcomes v_k of respective probability $p(v_k)$ (for $k = 1, \ldots, K$) is

$$H(V) = -\sum_{k=1}^{K} p(v_k) \log_2 p(v_k)$$

Definition: Entropy

The entropy of a discrete random variable V with K possible outcomes v_k of respective probability $p(v_k)$ (for $k = 1, \ldots, K$) is

$$H(V) = -\sum_{k=1}^{K} p(v_k) \log_2 p(v_k)$$



Examples for a Boolean random variable V with outcomes v_1 and v_2:

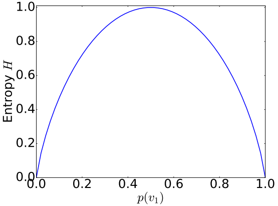
- $p(v_1) = 0.5$, $p(v_2) = 0.5$. Then,
  $H(V) = - 0.5 \cdot \log_2(0.5) - 0.5 \cdot \log_2(0.5)$
  $= 0.5 + 0.5 = 1$

Information Entropy

Definition: Entropy

The entropy of a discrete random variable V with K possible outcomes v_k of respective probability $p(v_k)$ (for $k = 1, \ldots, K$) is

$$H(V) = -\sum_{k=1}^{K} p(v_k) \log_2 p(v_k)$$



Examples for a Boolean random variable V with outcomes v_1 and v_2:

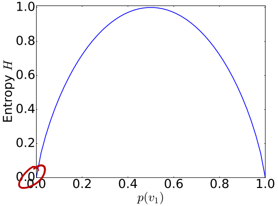
- $p(v_1) = 0.5$, $p(v_2) = 0.5$. Then, $H(V) = -0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$

Information Entropy

Definition: Entropy

The entropy of a discrete random variable V with K possible outcomes v_k of respective probability $p(v_k)$ (for $k = 1, \dots, K$) is

$$H(V) = -\sum_{k=1}^{K} p(v_k) \log_2 p(v_k)$$



Examples for a Boolean random variable V with outcomes v_1 and v_2:

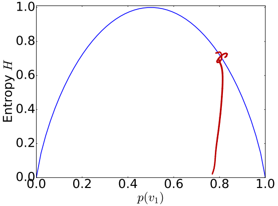
- $p(v_1) = 0.5$, $p(v_2) = 0.5$. Then, $H(V) = -0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$
- $p(v_1) = 0$, $p(v_2) = 1$. Then, $H(V) =$

Information Entropy

Definition: Entropy

The entropy of a discrete random variable V with K possible outcomes v_k of respective probability $p(v_k)$ (for $k = 1, \ldots, K$) is

$$H(V) = -\sum_{k=1}^{K} p(v_k) \log_2 p(v_k)$$



Examples for a Boolean random variable V with outcomes v_1 and v_2:

- $p(v_1) = 0.5$, $p(v_2) = 0.5$. Then, $H(V) = -0.5 \log_2(0.5) - 0.5 \log_2(0.5) = 1$
- $p(v_1) = 0$, $p(v_2) = 1$. Then, $H(V) = -\lim_{x \to 0} x \log_2(x) - 1 \log_2(1) = 0$

Prominent Splitting Criterion: Information Gain

Definition: Information Gain

Let V denote a random variable with the empirical class distribution p of the N data points at the current node.

Prominent Splitting Criterion: Information Gain

Definition: Information Gain

Let V denote a random variable with the empirical class distribution p of the N data points at the current node. Let a split s result in two child nodes with N_l and N_r data points and empirical class distributions p_l and p_r, and let V_l and V_r denote random variables with these probability distributions, respectively.

Prominent Splitting Criterion: Information Gain

Definition: Information Gain

Let V denote a random variable with the empirical class distribution p of the N data points at the current node. Let a split s result in two child nodes with N_l and N_r data points and empirical class distributions p_l and p_r, and let V_l and V_r denote random variables with these probability distributions, respectively. Then, the information gain achieved by split s is:

$$I = \underline{N} \cdot \underline{H(V)} - \underline{N_l} \cdot \underline{H(V_l)} - \underline{N_r} \cdot \underline{H(V_r)}$$

Prominent Splitting Criterion: Information Gain

Definition: Information Gain

Let V denote a random variable with the empirical class distribution p of the N data points at the current node. Let a split s result in two child nodes with N_l and N_r data points and empirical class distributions p_l and p_r, and let V_l and V_r denote random variables with these probability distributions, respectively. Then, the information gain achieved by split s is:

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

Prominent Splitting Criterion: Information Gain

Definition: Information Gain

Let V denote a random variable with the empirical class distribution p of the N data points at the current node. Let a split s result in two child nodes with N_l and N_r data points and empirical class distributions p_l and p_r, and let V_l and V_r denote random variables with these probability distributions, respectively. Then, the information gain achieved by split s is:

$$I = N \cdot H(V) - N_l \cdot \underline{H(V_l)} - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

We want to maximize this information gain.

Information Gain: Example

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

- Example: Splitting 8 data points ($4 \times$ false, $4 \times$ true).
  - Split 1: left: $4 \times$ true; right: $4 \times$ false
    - $N \cdot H(V) =$ $8 \cdot 1 = 8$
    - $N_l \cdot \underline{H(V_l)} =$ $4 \cdot 0 = 0$
    - $N_r \cdot \underline{H(V_r)}) =$ $4 \cdot 0 = 0$
  - Split 2: left: $2 \times$ true, $2 \times$ false; right: $2 \times$ true, $2 \times$ false
    - $N \cdot H(V) =$
    - $N_l \cdot H(V_l) =$
    - $N_r \cdot H(V_r)) =$

Information Gain: Example

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

- Example: Splitting 8 data points ($4 \times$ false, $4 \times$ true).
  - Split 1: left: $4 \times$ true; right: $4 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) =$
    - $N_r \cdot H(V_r)) =$
  - Split 2: left: $2 \times$ true, $2 \times$ false; right: $2 \times$ true, $2 \times$ false
    - $N \cdot H(V) =$
    - $N_l \cdot H(V_l) =$
    - $N_r \cdot H(V_r)) =$

Information Gain: Example

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

- Example: Splitting 8 data points ($4 \times$ false, $4 \times$ true).
  - Split 1: left: $4 \times$ true; right: $4 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) = 4 \cdot 0 = 0$
    - $N_r \cdot H(V_r)) =$
  - Split 2: left: $2 \times$ true, $2 \times$ false; right: $2 \times$ true, $2 \times$ false
    - $N \cdot H(V) =$
    - $N_l \cdot H(V_l) =$
    - $N_r \cdot H(V_r)) =$

Information Gain: Example

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

- Example: Splitting 8 data points ($4 \times$ false, $4 \times$ true).
  - Split 1: left: $4 \times$ true; right: $4 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) = 4 \cdot 0 = 0$
    - $N_r \cdot H(V_r)) = 4 \cdot 0 = 0$
  - Split 2: left: $2 \times$ true, $2 \times$ false; right: $2 \times$ true, $2 \times$ false
    - $N \cdot H(V) =$ $8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) =$ $4 \cdot 1 = 4$
    - $N_r \cdot H(V_r)) =$ $4 \cdot 1 = 4$

Information Gain: Example

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

- Example: Splitting 8 data points ($4 \times$ false, $4 \times$ true).
  - Split 1: left: $4 \times$ true; right: $4 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) = 4 \cdot 0 = 0$
    - $N_r \cdot H(V_r)) = 4 \cdot 0 = 0$
  - Split 2: left: $2 \times$ true, $2 \times$ false; right: $2 \times$ true, $2 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) =$
    - $N_r \cdot H(V_r)) =$

Information Gain: Example

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

- Example: Splitting 8 data points ($4 \times$ false, $4 \times$ true).
  - Split 1: left: $4 \times$ true; right: $4 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) = 4 \cdot 0 = 0$
    - $N_r \cdot H(V_r)) = 4 \cdot 0 = 0$
  - Split 2: left: $2 \times$ true, $2 \times$ false; right: $2 \times$ true, $2 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) = 4 \cdot 1 = 4$
    - $N_r \cdot H(V_r)) =$

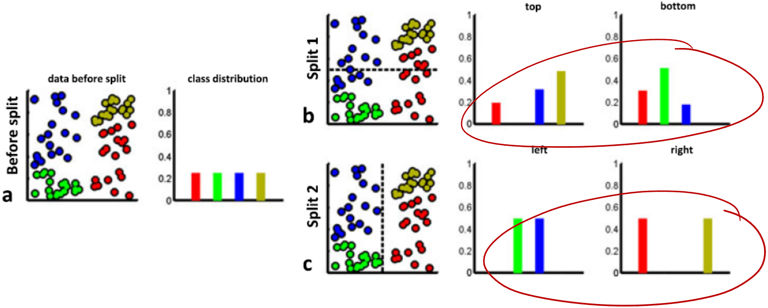
Information Gain: Example

$$I = N \cdot H(V) - N_l \cdot H(V_l) - N_r \cdot H(V_r)$$

$$= -N \sum_{k=1}^{K} p(v_k) \log_2(p(v_k)) + N_l \sum_{k=1}^{K} p_l(v_k) \log_2(p_l(v_k)) + N_r \sum_{k=1}^{K} p_r(v_k) \log_2(p_r(v_k))$$

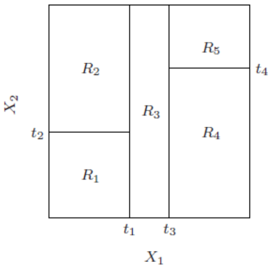
- Example: Splitting 8 data points ($4 \times$ false, $4 \times$ true).
  - Split 1: left: $4 \times$ true; right: $4 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) = 4 \cdot 0 = 0$
    - $N_r \cdot H(V_r)) = 4 \cdot 0 = 0$
  - Split 2: left: $2 \times$ true, $2 \times$ false; right: $2 \times$ true, $2 \times$ false
    - $N \cdot H(V) = 8 \cdot 1 = 8$
    - $N_l \cdot H(V_l) = 4 \cdot 1 = 4$
    - $N_r \cdot H(V_r)) = 4 \cdot 1 = 4$

Information Gain: Example



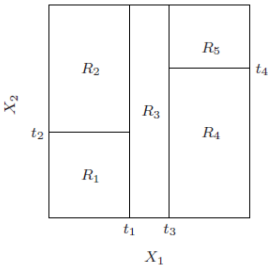
- We previously said split 2 (into left and right) is intuitively better than split 1 (into top and bottom)
- Information gain quantifies this

Splits Along Several Dimensions

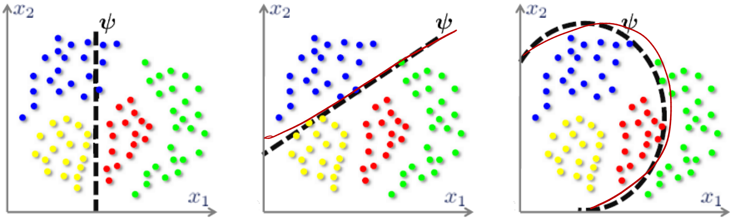


Axis-aligned split are standard, but more complex splits are possible

Splits Along Several Dimensions



Axis-aligned split are standard, but more complex splits are possible



Pros and Cons of Decision and Regression Trees

+ Flexible framework with exchangeable components: splitting criterion, leaf model, type of split
+ Interpretability
+ Handle categorical input values natively
+ Handle unimportant features well
+ Scalable for large datasets
- Tend to overfit
- Deterministic, i.e. not suitable for some ensemble methods

Hyperparameters of Decision and Regression Trees

Regression and decision trees have several hyperparameters:

- Minimum number of samples in a leaf (`min_leaf`)
- Maximal depth of the tree (`max_depth`)
- Total number of nodes
- Leaf model (weak learner; here constant)
- Split criterion

Assignment 8 explores some aspects of their influence on the predictive quality.

Bias and Variance of Trees

- Facts about tree-based models
  - Trees are very expressive models
  - When you slightly change the data, you might get a very different tree

Bias and Variance of Trees

- Facts about tree-based models
  - Trees are very expressive models
  - When you slightly change the data, you might get a very different tree
- Using these facts, please choose the right answer:
  - ★ Trees are a high-variance model. ✓
  - ★ Trees are a low-variance model.

Bias and Variance of Trees

- Facts about tree-based models
  - Trees are very expressive models
  - When you slightly change the data, you might get a very different tree
- Using these facts, please choose the right answer:
  - ★ Trees are a high-variance model.
  - ★ Trees are a low-variance model.
- Using these facts, please choose the right answer:
  - ★ Trees are a high-bias model.
  - ★ Trees are a low-bias model.

Reminder: Bias and Variance

$$\mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - h^*(\mathbf{x})\}^2] = \underbrace{\{\bar{h}(\mathbf{x}) - h^*(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - \bar{h}(\mathbf{x})\}^2]}_{\text{variance}}$$

expected squared deviation from best prediction $=$ $(\text{bias})^2 +$ variance

- $\hat{h}(\mathbf{x};\mathcal{D})$ is the prediction of the model that fits data $\mathcal{D}$ best
- $h^*(\mathbf{x})$ is the true best (unknown) prediction
- $\bar{h}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}[\hat{h}(\mathbf{x};\mathcal{D}]$ is the average model prediction (average of models trained on data sets $\mathcal{D}$ from a data distribution)

Computational Complexity of Regression and Decision Trees

- N data points of dimensionality D, axis-aligned splits
- Finding the best split value for a given feature (pre-sorted): $O(N)$
  - Amortized analysis: $O(N)$ to initialize all data points to the left child, $O(1)$ for moving one data point from left to right at a time

Computational Complexity of Regression and Decision Trees

- N data points of dimensionality D, axis-aligned splits
- Finding the best split value for a given feature (pre-sorted): $O(N)$
  - Amortized analysis: $O(N)$ to initialize all data points to the left child, $O(1)$ for moving one data point from left to right at a time
- Finding best split point at root: $O(DN)$
- Let $T(N)$ denote the time we pay for a complete subtree with N data points; this has a part due to the cost at the root and parts due to the 2 smaller child subtrees

Computational Complexity of Regression and Decision Trees

- N data points of dimensionality D, axis-aligned splits
- Finding the best split value for a given feature (pre-sorted): $\underline{O(N)}$
  - Amortized analysis: $O(N)$ to initialize all data points to the left child, $O(1)$ for moving one data point from left to right at a time

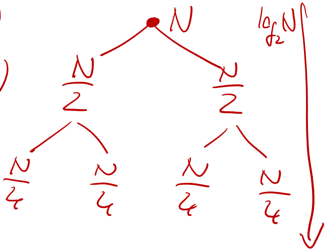
- Finding best split point at root: $O(DN)$
- Let $T(N)$ denote the time we pay for a complete subtree with N data points; this has a part due to the cost at the root and parts due to the 2 smaller child subtrees

- Best case: balanced trees
  - Fitting: $T(N) = \underline{O(DN)} + 2\underline{T(N/2)}$

$O(DN)$

$2 \cdot O(D \cdot \frac{N}{2})$

$4 \cdot O(D \cdot \frac{N}{4})$



Computational Complexity of Regression and Decision Trees

- N data points of dimensionality D, axis-aligned splits
- Finding the best split value for a given feature (pre-sorted): $O(N)$
  - Amortized analysis: $O(N)$ to initialize all data points to the left child, $O(1)$ for moving one data point from left to right at a time

- Finding best split point at root: $O(DN)$
- Let $T(N)$ denote the time we pay for a complete subtree with N data points; this has a part due to the cost at the root and parts due to the 2 smaller child subtrees

- Best case: balanced trees
  - Fitting: $T(N) = O(DN) + 2T(N/2)$

  $\rightsquigarrow$ This leads to $O(DN \log N)$ since we pay $O(D \cdot N)$ at each of $\log N$ levels

Computational Complexity of Regression and Decision Trees

- N data points of dimensionality D, axis-aligned splits
- Finding the best split value for a given feature (pre-sorted): $O(N)$
  - Amortized analysis: $O(N)$ to initialize all data points to the left child, $O(1)$ for moving one data point from left to right at a time

- Finding best split point at root: $O(DN)$
- Let $T(N)$ denote the time we pay for a complete subtree with N data points; this has a part due to the cost at the root and parts due to the 2 smaller child subtrees

- Best case: balanced trees
  - Fitting: $T(N) = O(DN) + 2T(N/2)$

  $\rightsquigarrow$ This leads to $O(DN \log N)$ since we pay $O(D \cdot N)$ at each of $\log N$ levels
  - Prediction: $O(\log N)$

Computational Complexity of Regression and Decision Trees

- N data points of dimensionality D, axis-aligned splits
- Finding the best split value for a given feature (pre-sorted): $O(N)$
  - Amortized analysis: $O(N)$ to initialize all data points left to the left child, $O(1)$ for moving one data point from left to right at a time
- Finding best split point at root: $O(DN)$
- Let $T(N)$ denote the time we pay for a complete subtree with N data points; this has a part due to the cost at the root and parts due to the 2 smaller child subtrees
- Worst case: splitting off one data point at a time
  - Fitting: $T(N) = O(DN) + T(N-1)$; this leads to $O(DN^2)$
  - Prediction: $O(N)$

Lecture Overview

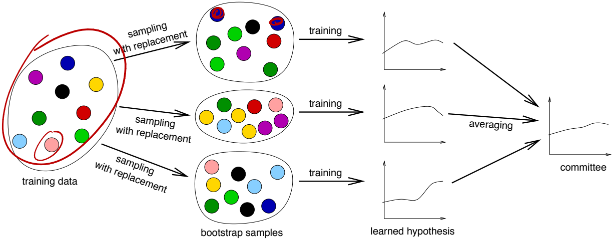
1. Decision and Regression Trees
   - Regression Trees
   - Classification Trees (= Decision Trees)
2. Bagging
3. Random Forests

Bagging is a Committee / Ensemble Approach

- A single expert may fail
- A committee of experts is more likely to get it right (if experts are experienced and diverse)

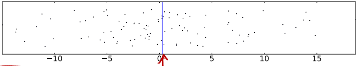
Bagging [Breimann, 1996]

- Train N models on bootstrap samples of the training data
- For each model, data is drawn randomly with replacements
- Average output of all models (bagging = bootstrap aggregation)

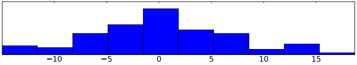


Boostrapping

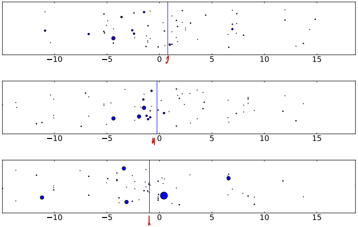
original data



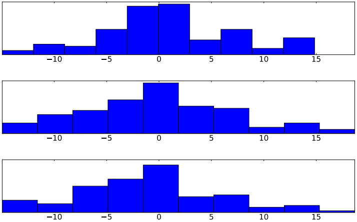
histogram



resampled data

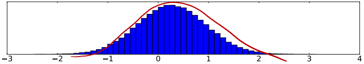


associated histograms



⋮ ⋮

histogram of the means: bootstrap distribution of the mean



Why Does Bagging Work?

- Bagging
  - Train N models on bootstrap samples of the training data
  - For each model, data is drawn randomly with replacements
  - Average output of all models (bagging = bootstrap aggregation)

Discuss with your neighbor first. (2 minutes)

- Voting question 1: A bagged estimator has
  - ★ higher variance
  - ★ lower variance

  than the individual models it bags.

- Voting question 2: A bagged estimator has
  - ★ higher bias
  - ★ lower bias

  than the individual models it bags.

Reminder: Bias and Variance

$$\mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - h^*(\mathbf{x})\}^2] = \underbrace{\{\bar{h}(\mathbf{x}) - h^*(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}[\{\hat{h}(\mathbf{x};\mathcal{D}) - \bar{h}(\mathbf{x})\}^2]}_{\text{variance}}$$

expected squared deviation from best prediction $= (\text{bias})^2 + \text{variance}$

- $\hat{h}(\mathbf{x};\mathcal{D})$ is the prediction of the model that fits data $\mathcal{D}$ best
- $h^*(\mathbf{x})$ is the true best (unknown) prediction
- $\bar{h}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}[\hat{h}(\mathbf{x};\mathcal{D}]$ is the average model prediction
  (average of models trained on data sets $\mathcal{D}$ from a data distribution)

Lecture Overview

1. Decision and Regression Trees
   - Regression Trees
   - Classification Trees (= Decision Trees)
2. Bagging
3. Random Forests

Why Does Bagging Work Well for Trees?

- Theoretical result [Breiman 2001]:
  - Ensemble error depends on (1) strength of individual models and (2) the degree to which the models' errors are uncorrelated.

Why Does Bagging Work Well for Trees?

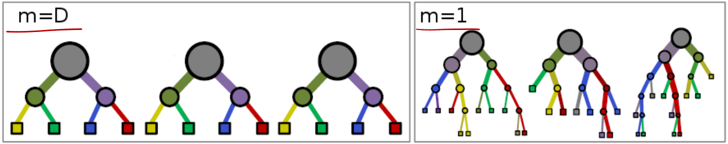
- Theoretical result [Breiman 2001]:
  - Ensemble error depends on (1) strength of individual models and (2) the degree to which the models' errors are uncorrelated.
- We thus want:
  - Individual models that still work (quite) well
  - Quite different models

Why Does Bagging Work Well for Trees?

- Theoretical result [Breiman 2001]:
  - Ensemble error depends on (1) strength of individual models and (2) the degree to which the models' errors are uncorrelated.
- We thus want:
  - Individual models that still work (quite) well
  - Quite different models
- Randomized trees give us these properties. We can randomize in many ways:
  - best split using a random subset of $m \leq D$ features $m \sim \log_2 D$
  - splitting using best out of a fixed number of random splits
  - training on bootstrap-samples of the data

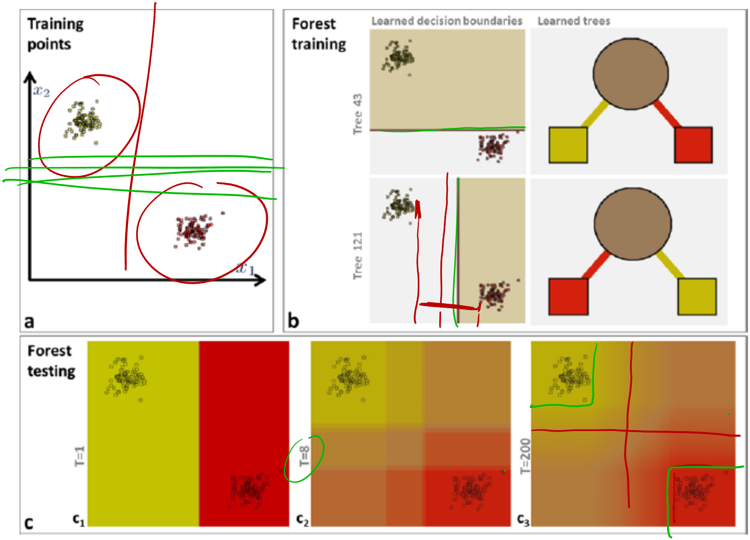
Why Does Bagging Work Well for Trees?

- Theoretical result [Breiman 2001]:
  - Ensemble error depends on (1) strength of individual models and (2) the degree to which the models' errors are uncorrelated.
- We thus want:
  - Individual models that still work (quite) well
  - Quite different models
- Randomized trees give us these properties. We can randomize in many ways:
  - best split using a random subset of $m \leq D$ features
  - splitting using best out of a fixed number of random splits
  - training on bootstrap-samples of the data



a) Low randomness, high tree correlation   **b)** High randomness, low tree correlation

Visualization of Random Forests and Their Predictions



Algorithm to Grow a Random Forest

Algorithm 15.1 *Random Forest for Regression or Classification.*

1. For $b = 1$ to B:
   (a) Draw a bootstrap sample $\mathbf{Z}^*$ of size N from the training data.
   (b) Grow a random-forest tree T_b to the bootstrapped data, by recursively repeating the following steps for each terminal node of the tree, until the minimum node size n_{min} is reached.
      i. Select m variables at random from the p variables.
      ii. Pick the best variable/split-point among the m.
      iii. Split the node into two daughter nodes.
2. Output the ensemble of trees $\{T_b\}_1^B$.

To make a prediction at a new point x:

Regression: $\hat{f}_{\text{rf}}^B(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classification: Let $\hat{C}_b(x)$ be the class prediction of the bth random-forest tree. Then $\hat{C}_{\text{rf}}^B(x) = \textit{majority vote } \{\hat{C}_b(x)\}_1^B$.

from [Hastie, Tibshirani and Friedman]

Computational Complexity of Random Forests

- N data points of dimensionality D, axis-aligned splits
- Picking m variables at random at each split; B trees

Computational Complexity of Random Forests

- N data points of dimensionality D, axis-aligned splits
- Picking m variables at random at each split; B trees

- Best case: balanced trees
  - Fitting: $O(BmN \log N)$

Computational Complexity of Random Forests

- N data points of dimensionality D, axis-aligned splits
- Picking m variables at random at each split; B trees

- Best case: balanced trees
  - Fitting: $O(BmN \log N)$
  - Prediction: $O(B \log N)$

Computational Complexity of Random Forests

- N data points of dimensionality D, axis-aligned splits
- Picking m variables at random at each split; B trees

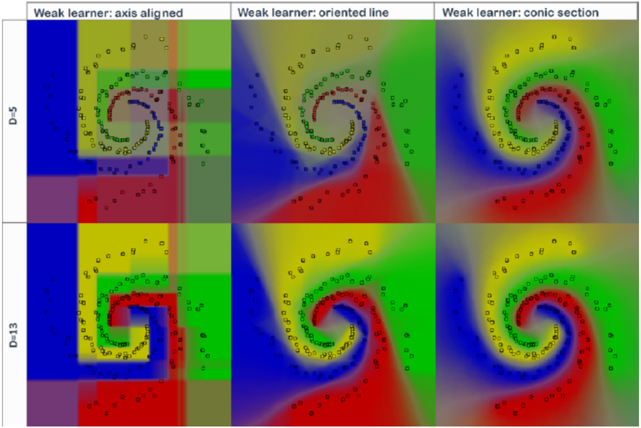
- Best case: balanced trees
  - Fitting: $O(BmN \log N)$
  - Prediction: $O(B \log N)$

- Worst case: splitting off one data point at a time
  - Fitting: $O(BmN^2)$
  - Prediction: $O(BN)$

Some Ways of Ensembling Trees

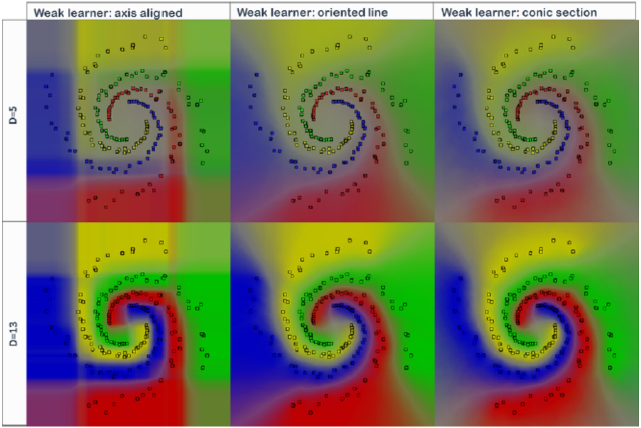
- Decision Trees + Bagging = Bagged trees
- Decision Trees + random feature subsets + Bagging = Random forest
- Decision Trees with almost random splits + Bagging = ExtraTrees
  - Extremely randomized trees

Examples: Influence of split randomization



- 400 trees with `max_depth` 5 (top) and 13 (bottom)
- Different split types: axis aligned (left), oriented lines (middle), and conic sections (right)
- Splitting: best out of 500 random proposed splits for each splits

Examples: Influence of split randomization



- 400 trees with `max_depth` 5 (top) and 13 (bottom)
- Different split types: axis aligned (left), oriented lines (middle), and conic sections (right)
- Splitting: best out of 5 random proposed splits for each splits

Advantages of Boostrapping

- Can help detect outliers
- Decorrelation of the trees in the ensemble
- Out-of-bag error:
  - not every data point is used to fit every single tree
  - in fact, almost 37% are not used in each tree (see assignment)
  - predict unused points for each tree to obtain unbiased estimate of the generalization error

Pros and Cons of Random Forests (and co.)

+ All pros from decision trees remain (except interpretability)
+ Better generalization
+ Out-of-bag error with little overhead
+ Scalability to large data sets and high dimensions
+ Require little tuning
- Relatively weak performance for smooth functions without noise

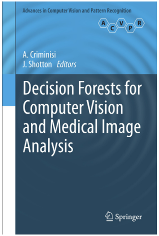
Summary by learning goals

Having heard this lecture, you can now . . .

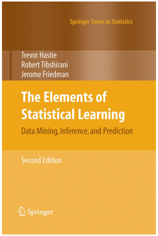
- determine good splits in regression and classification trees
- describe the steps of the CART algorithm
- describe the steps of the random forest algorithm
- formally describe entropy and information gain
- derive the complexity of decision trees and random forests
- explain why bootstrapping works well for trees
- describe some ways to randomize trees and their effects

Further Reading

- Criminisi et. al (2013)
  - Chapter 3: Introduction: The Abstract Forest Model
  - Chapter 4: Classification Forests
  - Chapter 5: Regression Forests
  - PDF of entire book available from university machines: `http://link.springer.com/book/10.1007/978-1-4471-4929-3`



- Hastie, Tibshirani and Friedman
  - Section 9.2: Tree-Based Methods
  - Chapter 15: Random Forests



Preview of Assignment 8

In assignement 8, you will . . .

- implement a simple regression tree and forest
- study hyperparameter influence on toy data
- calculate entropy and information gain by hand
- build a small decision tree by hand