Tree-Based Methods

Maria Jose Medina

Universidad de Santiago de Chile

Outline

Introduction

- Tree-based methods involve *stratifying* or *segmenting* the predictor space into a number of simple regions.

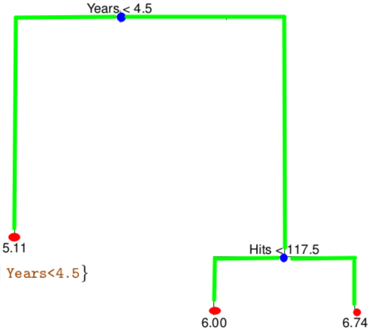
Introduction

- Tree-based methods involve *stratifying* or *segmenting* the predictor space into a number of simple regions.
- To make a prediction, we use the mean or the mode for the training observations in the region to which it belongs.

Introduction

- Tree-based methods involve *stratifying* or *segmenting* the predictor space into a number of simple regions.
- To make a prediction, we use the mean or the mode for the training observations in the region to which it belongs.
- Let's begin with an example...
- Let's predict a baseball player's **Salary** based on the **years** that he has played in the major leagues and the **hits** that he made in the previous year.

Introduction



$R_1 = \{X \mid$ `Years<4.5`$\}$

$R_2 = \{X \mid$ `Years>=4.5`, `Hits<117.5`$\}$ $R_3 = \{X \mid$ `Years>=4.5`, `Hits>=117.5`$\}$

- Terminal nodes or leaves.
- Internal nodes.
- Branches.

Table of Contents

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

Roughly speaking, there are two steps for building a regression tree:

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

Roughly speaking, there are two steps for building a regression tree:

1. **Divide the predictor space**

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

Roughly speaking, there are two steps for building a regression tree:

1. **Divide the predictor space**
   Divide the set of possible values of $X_1, \cdots, X_p$ into J disticnt and non-overlapping regions, $R_1, \cdots, R_J$

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

Roughly speaking, there are two steps for building a regression tree:

1. **Divide the predictor space**
   Divide the set of possible values of $X_1, \cdots, X_p$ into J disticnt and non-overlapping regions, $R_1, \cdots, R_J$
2. **Make prediction for every** R_j

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

Roughly speaking, there are two steps for building a regression tree:

1. **Divide the predictor space**
   Divide the set of possible values of $X_1, \cdots, X_p$ into J disticnt and non-overlapping regions, $R_1, \cdots, R_J$
2. **Make prediction for every R_j**
   For every observation that falls into the region R_j , we make the same prediction.

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

Roughly speaking, there are two steps for building a regression tree:

1. **Divide the predictor space**
   Divide the set of possible values of $X_1, \cdots, X_p$ into J disticnt and non-overlapping regions, $R_1, \cdots, R_J$
2. **Make prediction for every R_j**
   For every observation that falls into the region R_j , we make the same prediction.

The goal is to find $R_1, \cdots, R_J$ that minimize the RSS, given by

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

Roughly speaking, there are two steps for building a regression tree:

1. **Divide the predictor space**
   Divide the set of possible values of $X_1, \cdots, X_p$ into J disticnt and non-overlapping regions, $R_1, \cdots, R_J$
2. **Make prediction for every R_j**
   For every observation that falls into the region R_j , we make the same prediction.

The goal is to find $R_1, \cdots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$$

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

To construct $R_1, \cdots, R_J$ we use the **recursive binary splitting** approach:

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

To construct $R_1, \cdots, R_J$ we use the **recursive binary splitting** approach:

1. **Define the region of predictor space**

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

To construct $R_1, \cdots, R_J$ we use the **recursive binary splitting** approach:

1. **Define the region of predictor space**
   It's the space in which X_j takes on a value less than the cutpoint s.
   For example, for $J = 2$

$$R_1(j,s) = \{X | X_j < s\}, \qquad R_2(j,s) = \{X | X_j \geq s\}$$

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

To construct $R_1, \cdots, R_J$ we use the **recursive binary splitting** approach:

1. **Define the region of predictor space**
   It's the space in which X_j takes on a value less than the cutpoint s.
   For example, for $J = 2$

$$R_1(j,s) = \{X|X_j < s\}, \qquad R_2(j,s) = \{X|X_j \geq s\}$$

2. **Minimize RSS**

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

To construct $R_1, \cdots, R_J$ we use the **recursive binary splitting** approach:

1. **Define the region of predictor space**
   It's the space in which X_j takes on a value less than the cutpoint s.
   For example, for $J = 2$

$$R_1(j,s) = \{X|X_j < s\}, \qquad R_2(j,s) = \{X|X_j \geq s\}$$

2. **Minimize RSS**
   We seek the value j and s that minimize the equation,

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

To construct $R_1, \cdots, R_J$ we use the **recursive binary splitting** approach:

1. **Define the region of predictor space**
   It's the space in which X_j takes on a value less than the cutpoint s.
   For example, for $J = 2$

$$R_1(j,s) = \{X | X_j < s\}, \qquad R_2(j,s) = \{X | X_j \geq s\}$$

2. **Minimize RSS**
   We seek the value j and s that minimize the equation,

$$\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

3. **Repeat 1 and 2 in one of the previous identified regions**

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

To construct $R_1, \cdots, R_J$ we use the **recursive binary splitting** approach:

1. **Define the region of predictor space**
   It's the space in which X_j takes on a value less than the cutpoint s.
   For example, for $J = 2$

$$R_1(j,s) = \{X | X_j < s\}, \qquad R_2(j,s) = \{X | X_j \geq s\}$$

2. **Minimize RSS**
   We seek the value j and s that minimize the equation,

$$\sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

3. **Repeat 1 and 2 in one of the previous identified regions**
4. **Continue until a stopping criterion is reached**

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**
   y is the *mean of the training observations* in the region to which x belongs.

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**
   y is the *mean of the training observations* in the region to which x belongs.

For instance,

- Suppose that in Step 1 we obtain two regions, R_1 and R_2.

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**
   y is the *mean of the training observations* in the region to which x belongs.

For instance,

- Suppose that in Step 1 we obtain two regions, R_1 and R_2.
- The *response means* of the training observations in R_1 is 10, and in R_2 is 20.

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**
   y is the *mean of the training observations* in the region to which x belongs.

For instance,

- Suppose that in Step 1 we obtain two regions, R_1 and R_2.
- The *response means* of the training observations in R_1 is 10, and in R_2 is 20.
- For a given observation $X = x$,

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**
   y is the *mean of the training observations* in the region to which x belongs.

For instance,

- Suppose that in Step 1 we obtain two regions, R_1 and R_2.
- The *response means* of the training observations in R_1 is 10, and in R_2 is 20.
- For a given observation $X = x$,
  $\rightarrow$ If $x \in R_1$, we will predict a value of 10.

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**
   y is the *mean of the training observations* in the region to which x belongs.

For instance,

- Suppose that in Step 1 we obtain two regions, R_1 and R_2.
- The *response means* of the training observations in R_1 is 10, and in R_2 is 20.
- For a given observation $X = x$,
  $\rightarrow$ If $x \in R_1$, we will predict a value of 10.
  $\rightarrow$ If $x \in R_2$, we will predict a value of 20.

The Basics of Decision Trees

Prediction via Stratification of the Feature Space

6. **Predict the response y for x**
   y is the *mean of the training observations* in the region to which x belongs.

For instance,

- Suppose that in Step 1 we obtain two regions, R_1 and R_2.
- The *response means* of the training observations in R_1 is 10, and in R_2 is 20.
- For a given observation $X = x$,
  $\rightarrow$ If $x \in R_1$, we will predict a value of 10.
  $\rightarrow$ If $x \in R_2$, we will predict a value of 20.

Table of Contents

The Basics of Decision Trees

Tree Pruning

- The process described is likely to overfit the data, leading to poor test set performance.

The Basics of Decision Trees

Tree Pruning

- The process described is likely to overfit the data, leading to poor test set performance.
- This is because the resulting tree might be too complex.

The Basics of Decision Trees

Tree Pruning

- The process described is likely to overfit the data, leading to poor test set performance.
- This is because the resulting tree might be too complex.
- One way to solve this is consider a very large tree T_0 , and then prune it back in order to obtain a subtree that leads to the **lowest test error rate**.

The Basics of Decision Trees

Tree Pruning

- The process described is likely to overfit the data, leading to poor test set performance.
- This is because the resulting tree might be too complex.
- One way to solve this is consider a very large tree T_0 , and then prune it back in order to obtain a subtree that leads to the **lowest test error rate**.
- **Cost complexity pruning** —also known as *weakest link pruning*—gives us a way to do just this.

The Basics of Decision Trees

Tree Pruning

- We consider a sequence of trees indexed by a tuning parameter α.

The Basics of Decision Trees

Tree Pruning

- We consider a sequence of trees indexed by a tuning parameter α.
- For each value of α there corresponds a subtree $T \subset T_0$ such that

The Basics of Decision Trees

Tree Pruning

- We consider a sequence of trees indexed by a tuning parameter α.
- For each value of α there corresponds a subtree $T \subset T_0$ such that

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

is as small as possible.

The Basics of Decision Trees

Tree Pruning

- We consider a sequence of trees indexed by a tuning parameter α.
- For each value of α there corresponds a subtree $T \subset T_0$ such that

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

is as small as possible. Where,

  - $|T|$ indicates the number of terminal nodes of the tree T.

The Basics of Decision Trees

Tree Pruning

- We consider a sequence of trees indexed by a tuning parameter α.
- For each value of α there corresponds a subtree $T \subset T_0$ such that

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

is as small as possible. Where,

  - $|T|$ indicates the number of terminal nodes of the tree T.
  - R_m is the subset of predictor space corresponding to the mth terminal node.

The Basics of Decision Trees

Tree Pruning

- We consider a sequence of trees indexed by a tuning parameter α.
- For each value of α there corresponds a subtree $T \subset T_0$ such that

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

  is as small as possible. Where,
  - $|T|$ indicates the number of terminal nodes of the tree T.
  - R_m is the subset of predictor space corresponding to the mth terminal node.
  - $\hat{y}_{R_m}$ is the predicted response associated with R_m.

The Basics of Decision Trees

Tree Pruning

- We consider a sequence of trees indexed by a tuning parameter α.
- For each value of α there corresponds a subtree $T \subset T_0$ such that

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

is as small as possible. Where,

  - $|T|$ indicates the number of terminal nodes of the tree T.
  - R_m is the subset of predictor space corresponding to the mth terminal node.
  - $\hat{y}_{R_m}$ is the predicted response associated with R_m.
  - α controls a trade-off between the subtree's complexity and its fit to the training data.

The Basics of Decision Trees

Tree Pruning

For each value of α there corresponds a subtree $T \subset T_0$ such that

The Basics of Decision Trees

Tree Pruning

For each value of α there corresponds a subtree $T \subset T_0$ such that

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

is as small as possible.

The Basics of Decision Trees

Tree Pruning

For each value of α there corresponds a subtree $T \subset T_0$ such that

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

is as small as possible.

- When $\alpha = 0$, then $T = T_0$.

The Basics of Decision Trees

Tree Pruning

For each value of α there corresponds a subtree $T \subset T_0$ such that

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \qquad (1)$$

is as small as possible.

- When $\alpha = 0$, then $T = T_0$.
- As α increases, the quantity (1) will tend to be minimized for a smaller subtree.
- We can select a value of α using cross-validation as is described next.

The Basics of Decision Trees

Tree Pruning

Algorithm for Building a Regression Tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when a threshold is reached.

The Basics of Decision Trees

Tree Pruning

Algorithm for Building a Regression Tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when a threshold is reached.
2. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \cdots, K$:

The Basics of Decision Trees

Tree Pruning

Algorithm for Building a Regression Tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when a threshold is reached.
2. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \cdots, K$:
    a. Repeat step 1 on all but the kth fold of the training data.

The Basics of Decision Trees

Tree Pruning

Algorithm for Building a Regression Tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when a threshold is reached.
2. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \cdots, K$:
   a. Repeat step 1 on all but the kth fold of the training data.
   b. Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

The Basics of Decision Trees

Tree Pruning

Algorithm for Building a Regression Tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when a threshold is reached.
2. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \cdots, K$:
    a. Repeat step 1 on all but the kth fold of the training data.
    b. Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

    Average the results for each value of α, and pick α to minimize the average error.

The Basics of Decision Trees

Tree Pruning

Algorithm for Building a Regression Tree

1. Use recursive binary splitting to grow a large tree on the training data, stopping only when a threshold is reached.
2. Use K-fold cross-validation to choose α. That is, divide the training observations into K folds. For each $k = 1, \cdots, K$:
   a. Repeat step 1 on all but the kth fold of the training data.
   b. Evaluate the mean squared prediction error on the data in the left-out kth fold, as a function of α.

   Average the results for each value of α, and pick α to minimize the average error.
3. Return the subtree that corresponds to the chosen value of α.

Table of Contents

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

1. **Classification error rate (CER)**

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

1. **Classification error rate (CER)**
    - The plan is to assign and observation in a region R_i to the *most commonly occurring class*

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

1. **Classification error rate (CER)**
   - The plan is to assign and observation in a region R_i to the *most commonly occurring class*
   - Then, CER is the fraction of observations in the region R_i that do not belong to the *most common class*.

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

1. **Classification error rate (CER)**
    - The plan is to assign and observation in a region R_i to the *most commonly occurring class*
    - Then, CER is the fraction of observations in the region R_i that do not belong to the *most common class*.

$$E = 1 - \max_k(\hat{p}_{mk}).$$

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

1. **Classification error rate (CER)**
   - The plan is to assign and observation in a region R_i to the *most commonly occurring class*
   - Then, CER is the fraction of observations in the region R_i that do not belong to the *most common class*.

$$E = 1 - \max_k(\hat{p}_{mk}).$$

$\rightarrow \hat{p}_{mk}$ is the proportion of observations in the mth region that are from the kth class.

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

1. **Classification error rate (CER)**
   - The plan is to assign and observation in a region R_i to the *most commonly occurring class*
   - Then, CER is the fraction of observations in the region R_i that do not belong to the *most common class*.

$$E = 1 - \max_k(\hat{p}_{mk}).$$

   $\rightarrow \hat{p}_{mk}$ is the proportion of observations in the mth region that are from the kth class.
   - But, CER is not sufficiently sensitive for tree-growing.

The Basics of Decision Trees

Classification Trees

- We also use binary splitting to grow a classification tree.
- We can use different criterias for making the binary splits

1. **Classification error rate (CER)**
    - The plan is to assign and observation in a region R_i to the *most commonly occurring class*
    - Then, CER is the fraction of observations in the region R_i that do not belong to the *most common class*.

    $$E = 1 - \max_k(\hat{p}_{mk}).$$

    $\rightarrow \hat{p}_{mk}$ is the proportion of observations in the mth region that are from the kth class.
    - But, CER is not sufficiently sensitive for tree-growing.
    - In practice two other measures are preferable.

The Basics of Decision Trees

Classification Trees

2. **Gini index**

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

   - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

   - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.
   - G is referred to as a measure of *node purity*:

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

   - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.
   - G is referred to as a measure of *node purity*: a small value indicates that a node contains predominantly observations from a single class.

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

   - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.
   - G is referred to as a measure of *node purity*: a small value indicates that a node contains predominantly observations from a single class.
3. **Entropy**

The Basics of Decision Trees

Classification Trees

2. **Gini index**
    - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

    - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.
    - G is referred to as a measure of *node purity*: a small value indicates that a node contains predominantly observations from a single class.
3. **Entropy**
    - Measures the impurity or uncertainty in a group of observations.

The Basics of Decision Trees

Classification Trees

2. **Gini index**
    - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

    - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.
    - G is referred to as a measure of *node purity*: a small value indicates that a node contains predominantly observations from a single class.
3. **Entropy**
    - Measures the impurity or uncertainty in a group of observations.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

   - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.
   - G is referred to as a measure of *node purity*: a small value indicates that a node contains predominantly observations from a single class.

3. **Entropy**
   - Measures the impurity or uncertainty in a group of observations.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

   - It can be shown that $D \to 0$ if all $\hat{p}_{mk} \to 0$ or $\hat{p}_{mk} \to 1$.

The Basics of Decision Trees

Classification Trees

2. **Gini index**
   - Is a measure of total variance across the K classes.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}).$$

   - G is small if $\hat{p}_{mk} \to 1$ or $\hat{p}_{mk} \to 0$.
   - G is referred to as a measure of *node purity*: a small value indicates that a node contains predominantly observations from a single class.
3. **Entropy**
   - Measures the impurity or uncertainty in a group of observations.

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

   - It can be shown that $D \to 0$ if all $\hat{p}_{mk} \to 0$ or $\hat{p}_{mk} \to 1$.
   - In this sense, Entropy is also referred as a measure of *node purity*.

The Basics of Decision Trees

Classification Trees

- When building a classification tree, either the Gini index or the entropy are typically used to evaluate the **quality** of a particular split.

The Basics of Decision Trees

Classification Trees

- When building a classification tree, either the Gini index or the entropy are typically used to evaluate the **quality** of a particular split.
- Any of these three approaches might be used when **pruning** the tree, but CER is preferable if **prediction accuracy** of the final pruned tree is the goal.

Table of Contents

The Basics of Decision Trees

Advantages and Disadvantages of Trees

Advantages

- ✓ Trees can be displayed graphically, and are easily interpreted.

The Basics of Decision Trees

Advantages and Disadvantages of Trees

Advantages

- ✓ Trees can be displayed graphically, and are easily interpreted.
- ✓ Can handle qualitative predictors without the need to create dummy variables.

The Basics of Decision Trees

Advantages and Disadvantages of Trees

Advantages

- ✓ Trees can be displayed graphically, and are easily interpreted.
- ✓ Can handle qualitative predictors without the need to create dummy variables.

Disadvantages

- × Do not have the same level of predictive accuracy as some methods seen before.

The Basics of Decision Trees

Advantages and Disadvantages of Trees

Advantages

- ✓ Trees can be displayed graphically, and are easily interpreted.
- ✓ Can handle qualitative predictors without the need to create dummy variables.

Disadvantages

- × Do not have the same level of predictive accuracy as some methods seen before.
- × Suffers for high variance: a small change in the data can cause a large change in the final estimated tree.

The Basics of Decision Trees

Advantages and Disadvantages of Trees

Advantages

- ✓ Trees can be displayed graphically, and are easily interpreted.
- ✓ Can handle qualitative predictors without the need to create dummy variables.

Disadvantages

- × Do not have the same level of predictive accuracy as some methods seen before.
- × Suffers for high variance: a small change in the data can cause a large change in the final estimated tree.

Note

By aggregating many decision trees, using methods like bagging, random forests, and boosting, the predictive performance of trees can be substantially improved.

Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

- We'll begin with define an *ensemble method*.

Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

- We'll begin with define an *ensemble method*.
- An ensemble method is an approach that combines many simple *building ensemble block* models to obtain a single and potentially very powerful model.

Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

- We'll begin with define an *ensemble method*.
- An ensemble method is an approach that combines many simple *building ensemble block* models to obtain a single and potentially very powerful model.
- These simple building block models are sometimes known as *weak learners*, since they may lead to mediocre predictions on their own.

Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

- We'll begin with define an *ensemble method*.
- An ensemble method is an approach that combines many simple *building ensemble block* models to obtain a single and potentially very powerful model.
- These simple building block models are sometimes known as *weak learners*, since they may lead to mediocre predictions on their own.
- We will now discuss bagging, random forests, boosting, and Bayesian learners additive regression trees.

Bagging, Random Forests, Boosting, and Bayesian Additive Regression Trees

- We'll begin with define an *ensemble method*.
- An ensemble method is an approach that combines many simple *building ensemble block* models to obtain a single and potentially very powerful model.
- These simple building block models are sometimes known as *weak learners*, since they may lead to mediocre predictions on their own.
- We will now discuss bagging, random forests, boosting, and Bayesian learners additive regression trees.
- These are ensemble methods for which the simple building block is a regression or a classification tree.

Table of Contents

Bagging

- Recall that given a set of n independent observations $Z_1, \cdots, Z_n$, each with $Var(Z_i) = \sigma^2$, then $Var(\bar{Z}) = \sigma^2/n$.

Bagging

- Recall that given a set of n independent observations $Z_1, \cdots, Z_n$, each with $Var(Z_i) = \sigma^2$, then $Var(\bar{Z}) = \sigma^2/n$.
  $\rightarrow$ Averaging a set of observations reduces variance.

Bagging

- Recall that given a set of n independent observations $Z_1, \cdots, Z_n$, each with $Var(Z_i) = \sigma^2$, then $Var(\bar{Z}) = \sigma^2/n$.
  $\rightarrow$ Averaging a set of observations reduces variance.
- We use this idea to do resampling with replacement (bootstrap sample) for the training dataset.

Bagging

- Recall that given a set of n independent observations $Z_1, \cdots, Z_n$, each with $Var(Z_i) = \sigma^2$, then $Var(\bar{Z}) = \sigma^2/n$.
  $\rightarrow$ Averaging a set of observations reduces variance.
- We use this idea to do resampling with replacement (bootstrap sample) for the training dataset.

Bagging

For regression:

Bagging

For regression:

- We generate B different bootstrapped training sets.

Bagging

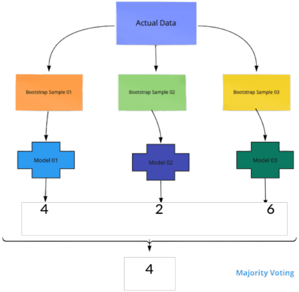
For regression:

- We generate B different bootstrapped training sets.
- Train the model on the bth bootstrapped training set
- Average all the predictions.

Bagging

For regression:

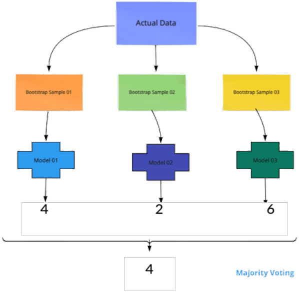
- We generate B different bootstrapped training sets.
- Train the model on the bth bootstrapped training set
- Average all the predictions.



Bagging

For regression:

- We generate B different bootstrapped training sets.
- Train the model on the bth bootstrapped training set
- Average all the predictions.

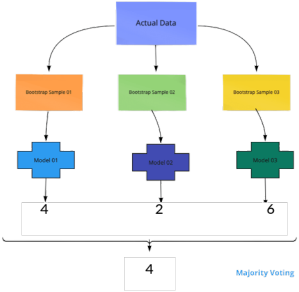


For classification:

Bagging

For regression:

- We generate B different bootstrapped training sets.
- Train the model on the bth bootstrapped training set
- Average all the predictions.



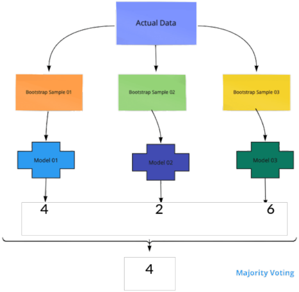
For classification:

- We generate B different bootstrapped training sets.

Bagging

For regression:

- We generate B different bootstrapped training sets.
- Train the model on the bth bootstrapped training set
- Average all the predictions.



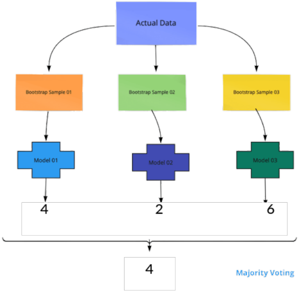
For classification:

- We generate B different bootstrapped training sets.
- Record the class predicted by each of the B trees.

Bagging

For regression:

- We generate B different bootstrapped training sets.
- Train the model on the bth bootstrapped training set
- Average all the predictions.



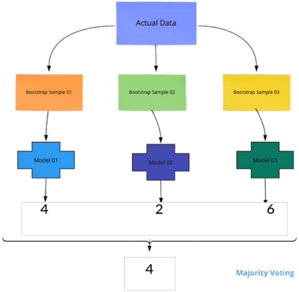
For classification:

- We generate B different bootstrapped training sets.
- Record the class predicted by each of the B trees.
- Take the *majority vote* as the prediction.

Bagging

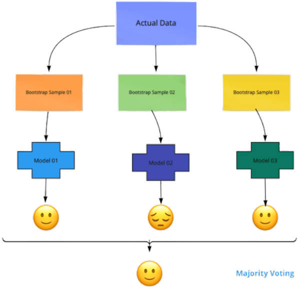
For regression:

- We generate B different bootstrapped training sets.
- Train the model on the bth bootstrapped training set
- Average all the predictions.



For classification:

- We generate B different bootstrapped training sets.
- Record the class predicted by each of the B trees.
- Take the *majority vote* as the prediction.



Bagging

Out-of-Bag Error Estimation

- On average, each bagged tree makes use of around $2/3$ of the observations.

Bagging

Out-of-Bag Error Estimation

- On average, each bagged tree makes use of around $2/3$ of the observations.
- The remaining $1/3$ of the observations not used to fit a given bagged tree are referred to as the **out-of-bag (OOB) observations**.

Bagging

Out-of-Bag Error Estimation

- On average, each bagged tree makes use of around $2/3$ of the observations.
- The remaining $1/3$ of the observations not used to fit a given bagged tree are referred to as the **out-of-bag (OOB) observations**.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB.

Bagging

Out-of-Bag Error Estimation

- On average, each bagged tree makes use of around $2/3$ of the observations.
- The remaining $1/3$ of the observations not used to fit a given bagged tree are referred to as the **out-of-bag (OOB) observations**.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB.
- The final prediction is then the average predicted responses (for regression) or the majority vote (for classification).

Bagging

Out-of-Bag Error Estimation

- On average, each bagged tree makes use of around $2/3$ of the observations.
- The remaining $1/3$ of the observations not used to fit a given bagged tree are referred to as the **out-of-bag (OOB) observations**.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB.
- The final prediction is then the average predicted responses (for regression) or the majority vote (for classification).
- This leads to a single OOB prediction for the ith observation.

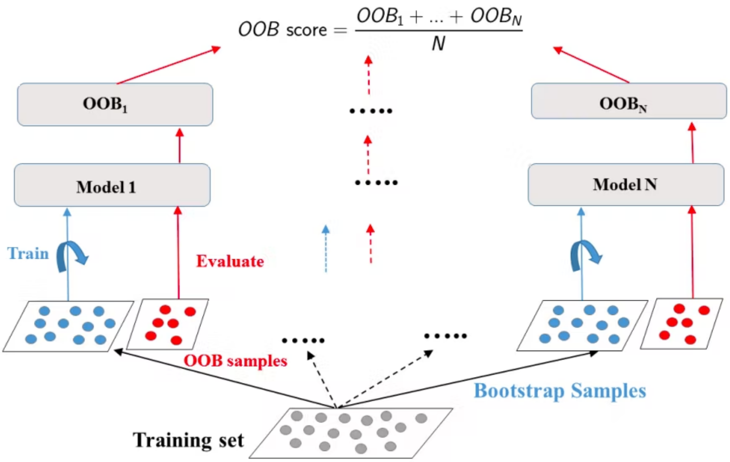
Bagging

Out-of-Bag Error Estimation

- On average, each bagged tree makes use of around $2/3$ of the observations.
- The remaining $1/3$ of the observations not used to fit a given bagged tree are referred to as the **out-of-bag (OOB) observations**.
- We can predict the response for the ith observation using each of the trees in which that observation was OOB.
- The final prediction is then the average predicted responses (for regression) or the majority vote (for classification).
- This leads to a single OOB prediction for the ith observation.
- The resulting OOB error is a valid estimate of the test error for the bagged model.

Bagging

Out-of-Bag Error Estimation



Bagging

Variable importance measures

One can obtain an overall summary of the importance of each predictor.

Bagging

Variable importance measures

One can obtain an overall summary of the importance of each predictor.

For regression:

Bagging

Variable importance measures

One can obtain an overall summary of the importance of each predictor.

For regression:

- We record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.

Bagging

Variable importance measures

One can obtain an overall summary of the importance of each predictor.

For regression:

- We record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.
- A large value indicates an important predictor.

Bagging

Variable importance measures

One can obtain an overall summary of the importance of each predictor.

For regression:

- We record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.
- A large value indicates an important predictor.

For classification:

Bagging

Variable importance measures

One can obtain an overall summary of the importance of each predictor.

For regression:

- We record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.
- A large value indicates an important predictor.

For classification:

- We add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.

Bagging

Variable importance measures

One can obtain an overall summary of the importance of each predictor.

For regression:

- We record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees.
- A large value indicates an important predictor.

For classification:

- We add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.
- A small value indicates an important predictor.

Table of Contents

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

- Suppose that there is one strong predictor in the data set, along with others moderately strong predictors.

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

- Suppose that there is one strong predictor in the data set, along with others moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

- Suppose that there is one strong predictor in the data set, along with others moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.
- Consequently, all of the bagged trees will look quite similar to each other.

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

- Suppose that there is one strong predictor in the data set, along with others moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.
- Consequently, all of the bagged trees will look quite similar to each other.
- Then the predictions will be highly correlated.

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

- Suppose that there is one strong predictor in the data set, along with others moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.
- Consequently, all of the bagged trees will look quite similar to each other.
- Then the predictions will be highly correlated.

The solution

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

- Suppose that there is one strong predictor in the data set, along with others moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.
- Consequently, all of the bagged trees will look quite similar to each other.
- Then the predictions will be highly correlated.

The solution

- Random forests overcome this problem by forcing each split to consider only a subset of the predictors.

Random Forests

- As in bagging, we build a number of decision trees on bootstrapped samples.
- But here, a random sample of m predictors is chosen as split candidates from the full set of p predictors.
- Typically we choose $m \approx \sqrt{p}$

Why?

- Suppose that there is one strong predictor in the data set, along with others moderately strong predictors.
- In the collection of bagged trees, most or all of the trees will use this strong predictor in the top split.
- Consequently, all of the bagged trees will look quite similar to each other.
- Then the predictions will be highly correlated.

The solution

- Random forests overcome this problem by forcing each split to consider only a subset of the predictors.
- On average $(p - m)/p$ of the splits will not even consider the strong predictor, and so other predictors will have more of a chance.

Table of Contents

Boosting

- Boosting refers to an ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessor.

Boosting

- Boosting refers to an ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessor.
- More formally, in boosting many **weak learners** are combined to form a **strong learner**.

Boosting

- Boosting refers to an ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessor.
- More formally, in boosting many **weak learners** are combined to form a **strong learner**.
  $\rightarrow$A weak learner is a model doing slightly better than random guessing.

Boosting

- Boosting refers to an ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessor.
- More formally, in boosting many **weak learners** are combined to form a **strong learner**.
  →A weak learner is a model doing slightly better than random guessing.
- Here, an ensemble of predictors are trained *sequentially* and each predictor tries to correct the errors made by its predecessor.

Boosting

- Boosting refers to an ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessor.
- More formally, in boosting many **weak learners** are combined to form a **strong learner**.
  $\rightarrow$A weak learner is a model doing slightly better than random guessing.
- Here, an ensemble of predictors are trained *sequentially* and each predictor tries to correct the errors made by its predecessor.
- Boosting has three tuning parameters:
  1. **The number of trees** B: selected by cross-validation.

Boosting

- Boosting refers to an ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessor.
- More formally, in boosting many **weak learners** are combined to form a **strong learner**.
  →A weak learner is a model doing slightly better than random guessing.
- Here, an ensemble of predictors are trained *sequentially* and each predictor tries to correct the errors made by its predecessor.
- Boosting has three tuning parameters:
  1. **The number of trees** B: selected by cross-validation.
  2. **The shrinkage parameter** η: this controls the rate at which boosting learns.

Boosting

- Boosting refers to an ensemble method in which many predictors are trained and each predictor learns from the errors of its predecessor.
- More formally, in boosting many **weak learners** are combined to form a **strong learner**.
  →A weak learner is a model doing slightly better than random guessing.
- Here, an ensemble of predictors are trained *sequentially* and each predictor tries to correct the errors made by its predecessor.
- Boosting has three tuning parameters:
  1. **The number of trees** B: selected by cross-validation.
  2. **The shrinkage parameter** η: this controls the rate at which boosting learns.
  3. **The number** d **of splits in each tree**: which controls the complexity of the boosted ensemble.

Boosting

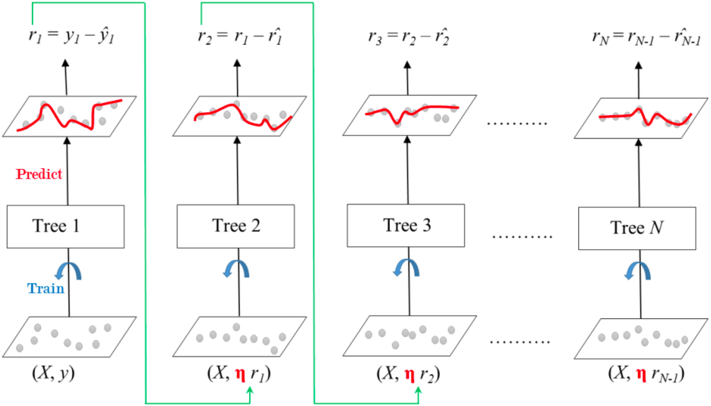

$r_1 = y_1 - \hat{y}_1$
$r_2 = r_1 - \hat{r}_1$
$r_3 = r_2 - \hat{r}_2$
$r_N = r_{N-1} - \hat{r}_{N-1}$
Predict
Tree 1
Tree 2
Tree 3
Tree N
Train
(X, y)
$(X, \eta\, r_1)$
$(X, \eta\, r_2)$
$(X, \eta\, r_{N-1})$

Table of Contents

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.
- Each tree tries to capture signal not yet accounted for by the current model.

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.
- Each tree tries to capture signal not yet accounted for by the current model.
- The main novelty in BART is the way in which new trees are generated.

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.
- Each tree tries to capture signal not yet accounted for by the current model.
- The main novelty in BART is the way in which new trees are generated.

But first, some notation:

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.
- Each tree tries to capture signal not yet accounted for by the current model.
- The main novelty in BART is the way in which new trees are generated.

But first, some notation:

- Let K denote the number of regression trees.

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.
- Each tree tries to capture signal not yet accounted for by the current model.
- The main novelty in BART is the way in which new trees are generated.

But first, some notation:

- Let K denote the number of regression trees.
- B is the number of iterations for which the BART algorithm will be run.

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.
- Each tree tries to capture signal not yet accounted for by the current model.
- The main novelty in BART is the way in which new trees are generated.

But first, some notation:

- Let K denote the number of regression trees.
- B is the number of iterations for which the BART algorithm will be run.
- $\hat{f}_k^b(x)$ is the prediction at x for the kth regression tree used in the bth iteration.

Bayesian Additive Regression Trees (BART)

BART is related to both approaches seen before:

- Each tree is constructed in a random manner.
- Each tree tries to capture signal not yet accounted for by the current model.
- The main novelty in BART is the way in which new trees are generated.

But first, some notation:

- Let K denote the number of regression trees.
- B is the number of iterations for which the BART algorithm will be run.
- $\hat{f}_k^b(x)$ is the prediction at x for the kth regression tree used in the bth iteration.
- At the end of each iteration, all the K trees from b will be summed,

$$\hat{f}^b(x) = \sum_{k=1}^{K} \hat{f}_k^b(x) \text{ for } b = 1, , B.$$

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK} \sum_{i=1}^{n} y_i.$$

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$

2. Compute the prediction in for all the K trees as,

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$

2. Compute the prediction in for all the K trees as,

$$\hat{f}^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$

2. Compute the prediction in for all the K trees as,

$$\hat{f}^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

3. Update each of the K trees in each iteration.

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,
$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$
2. Compute the prediction in for all the K trees as,
$$\hat{f}^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i.$$
3. Update each of the K trees in each iteration.
For $b = 2, \cdots, B$:

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$

2. Compute the prediction in for all the K trees as,

$$\hat{f}^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

3. Update each of the K trees in each iteration.
   For $b = 2, \cdots, B$:
   a. For $k = 1, \cdots, K$: compute the **current partial residual**,

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$

2. Compute the prediction in for all the K trees as,

$$\hat{f}^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

3. Update each of the K trees in each iteration.
   For $b = 2, \cdots, B$:
   a. For $k = 1, \cdots, K$: compute the **current partial residual**,

$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i) \; i = 1, \cdots, n.$$

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$

2. Compute the prediction in for all the K trees as,

$$\hat{f}^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

3. Update each of the K trees in each iteration.
   For $b = 2, \cdots, B$:
   a. For $k = 1, \cdots, K$: compute the **current partial residual**,

   $$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i) \; i = 1, \cdots, n.$$

   b. Fit a new tree $\hat{f}_{k'}^b(x_i)$ with r_i, by randomly perturbing $\hat{f}_{k'}^{b-1}(x_i)$.

Bayesian Additive Regression Trees (BART)

The algorithm is described as follows,

1. Initialized all trees to have a single root node, with a prediction given by,

$$\hat{f}_k^1(x) = \frac{1}{nK}\sum_{i=1}^{n} y_i.$$

2. Compute the prediction in for all the K trees as,

$$\hat{f}^1(x) = \frac{1}{n}\sum_{i=1}^{n} y_i.$$

3. Update each of the K trees in each iteration.
   For $b = 2, \cdots, B$:
   a. For $k = 1, \cdots, K$: compute the **current partial residual**,

$$r_i = y_i - \sum_{k'<k} \hat{f}_{k'}^b(x_i) - \sum_{k'>k} \hat{f}_{k'}^{b-1}(x_i) \;\; i = 1, \cdots, n.$$

   b. Fit a new tree $\hat{f}_{k'}^b(x_i)$ with r_i, by randomly perturbing $\hat{f}_{k'}^{b-1}(x_i)$.
   c. Compute $\hat{f}_k^b(x_i)$.

Bayesian Additive Regression Trees (BART)

5. Typically models obtained in the earlier iterations — known as the **burn-in** period— tend not to provide very good results. So we throw away the L burn-in iterations and to obtain a single prediction, we compute:

Bayesian Additive Regression Trees (BART)

5. Typically models obtained in the earlier iterations — known as the **burn-in period**— tend not to provide very good results. So we throw away the L burn-in iterations and to obtain a single prediction, we compute:

$$\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \hat{f}^b(x).$$

Bayesian Additive Regression Trees (BART)

5. Typically models obtained in the earlier iterations — known as the **burn-in** period— tend not to provide very good results. So we throw away the L burn-in iterations and to obtain a single prediction, we compute:

$$\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \hat{f}^b(x).$$

What is the random perturbation?

Bayesian Additive Regression Trees (BART)

5. Typically models obtained in the earlier iterations — known as the **burn-in period**— tend not to provide very good results. So we throw away the L burn-in iterations and to obtain a single prediction, we compute:

$$\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \hat{f}^b(x).$$

What is the random perturbation?

1. We may change the structure of the tree by adding or pruning branches.

Bayesian Additive Regression Trees (BART)

5. Typically models obtained in the earlier iterations — known as the **burn-in** period— tend not to provide very good results. So we throw away the L burn-in iterations and to obtain a single prediction, we compute:

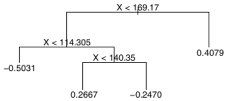
$$\hat{f}(x) = \frac{1}{B-L} \sum_{b=L+1}^{B} \hat{f}^b(x).$$

What is the random perturbation?

1. We may change the structure of the tree by adding or pruning branches.
2. We may change the prediction in each terminal node of the tree.

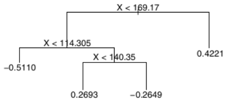
Bayesian Additive Regression Trees (BART)

(a): $\hat{f}_k^{b-1}(X)$



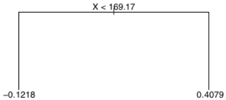
The kth tree at the $(b-1)$*st iteration,*

(b): Possibility #1 for $\hat{f}_k^{b}(X)$



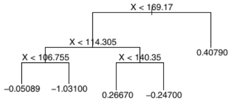
same structure as $\hat{f}_k^{b-1}(X)$*, but with different predictions at the terminal nodes.*

(c): Possibility #2 for $\hat{f}_k^{b}(X)$



pruning $\hat{f}_k^{b-1}(X)$.

(d): Possibility #3 for $\hat{f}_k^{b}(X)$



$\hat{f}_k^{b}(X)$ *may have more terminal nodes than* $\hat{f}_k^{b-1}(X)$.

Table of Contents

Summary

- The idea of this ensemble methods is to use many *weak learners* to construct a *strong learner*.

Summary

- The idea of this ensemble methods is to use many *weak learners* to construct a *strong learner*.
- They have the flexibility and the ability to handle predictors of mixed types.

Summary

- The idea of this ensemble methods is to use many *weak learners* to construct a *strong learner*.
- They have the flexibility and the ability to handle predictors of mixed types.
- We have now seen four approaches for fitting an ensemble of trees:
  1. **Bagging**: the trees are grown independently on *random samples* of the observations and the trees tend to be quite similar to each other.

Summary

- The idea of this ensemble methods is to use many *weak learners* to construct a *strong learner*.
- They have the flexibility and the ability to handle predictors of mixed types.
- We have now seen four approaches for fitting an ensemble of trees:
  1. **Bagging**: the trees are grown independently on *random samples* of the observations and the trees tend to be quite similar to each other.
  2. **Random forests**: overcomes the problem with *bagging* by using a *random subset* of the features on each split.

Summary

- The idea of this ensemble methods is to use many *weak learners* to construct a *strong learner*.
- They have the flexibility and the ability to handle predictors of mixed types.
- We have now seen four approaches for fitting an ensemble of trees:
  1. **Bagging**: the trees are grown independently on *random samples* of the observations and the trees tend to be quite similar to each other.
  2. **Random forests**: overcomes the problem with *bagging* by using a *random subset* of the features on each split.
  3. **Boosting**: we do not draw any random samples, and each new tree is fit to the signal that is left over from the earlier trees.

Summary

- The idea of this ensemble methods is to use many *weak learners* to construct a *strong learner*.
- They have the flexibility and the ability to handle predictors of mixed types.
- We have now seen four approaches for fitting an ensemble of trees:
  1. **Bagging**: the trees are grown independently on *random samples* of the observations and the trees tend to be quite similar to each other.
  2. **Random forests**: overcomes the problem with *bagging* by using a *random subset* of the features on each split.
  3. **Boosting**: we do not draw any random samples, and each new tree is fit to the signal that is left over from the earlier trees.
  4. **BART**: we once again only make use of the original data, and we grow the trees successively. However, each tree is perturbed.

Thank you!

Any question?