Summer Program FGV/EMAp 2019

Introduction to Machine Learning with Python

Decision Trees and Random Forest

Prof. Luis Gustavo Nonato

University of São Paulo - São Carlos - SP

Classification and Regression Trees - CART

Given a training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$,

tree-based methods aim to

Classification and Regression Trees - CART

Given a training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$,

tree-based methods aim to

- partitioning the input space into axis aligned sub-domains;

Classification and Regression Trees - CART

Given a training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$,

tree-based methods aim to

- partitioning the input space into axis aligned sub-domains;
- "fit" a very simple model in each sub-domain;

Classification and Regression Trees - CART

Given a training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$,

tree-based methods aim to

- partitioning the input space into axis aligned sub-domains;
- "fit" a very simple model in each sub-domain;
- use each simple model to make predictions within its corresponding sub-domain.

Classification and Regression Trees - CART

Given a training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$,

tree-based methods aim to

- partitioning the input space into axis aligned sub-domains;
- "fit" a very simple model in each sub-domain;
- use each simple model to make predictions within its corresponding sub-domain.

The simplest model to be fitted in each sub-domain is a constant predictor, that is

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i I(\mathbf{x} \in R_i)$$

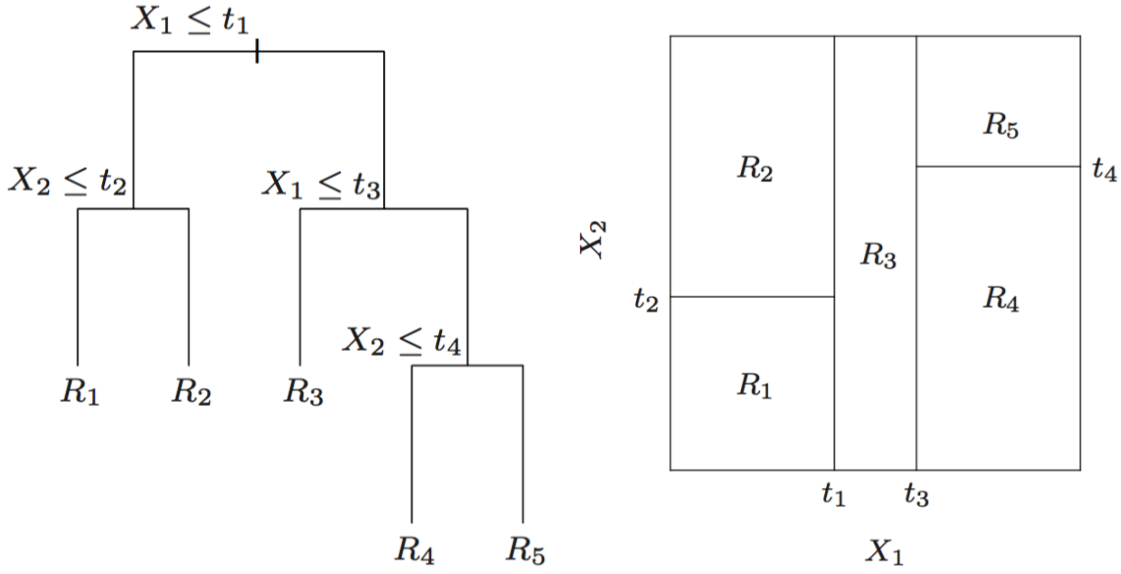
where m is the number of sub-domains, c_i is the constant predictor, and I is the indicator function, that is, $I(\mathbf{x} \in R_i) = 1$ if $\mathbf{x}$ lies on the sub-domain R_i and $I(\mathbf{x} \in R_i) = 0$ otherwise.

Classification and Regression Trees - CART

Example when $\mathbf{x} \in \mathbb{R}^2$

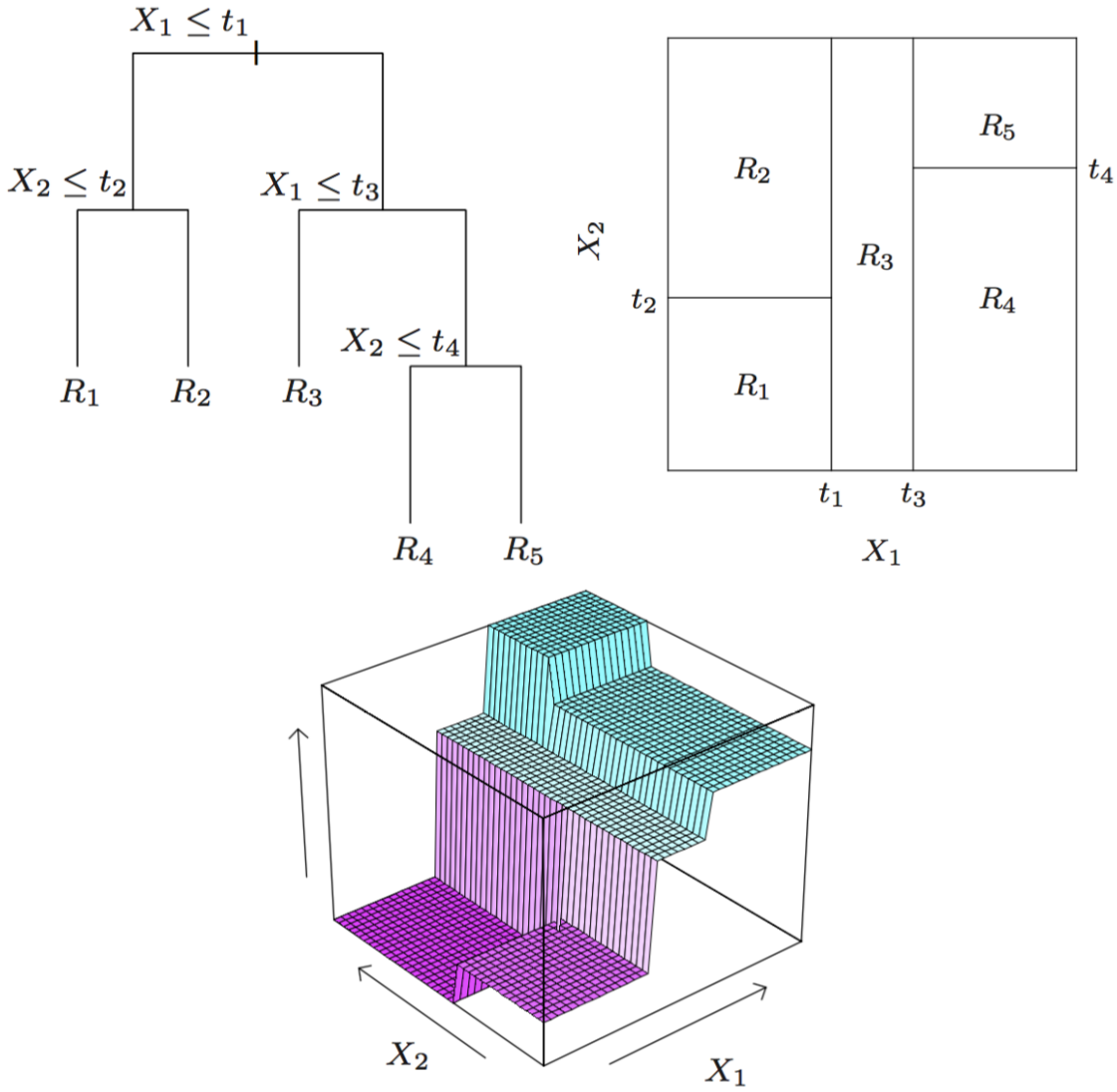
Classification and Regression Trees - CART

Example when $\mathbf{x} \in \mathbb{R}^2$



Classification and Regression Trees - CART

Example when $\mathbf{x} \in \mathbb{R}^2$



Classification and Regression Trees - CART

There are three parameters involved in the process of "growing" a tree:

Classification and Regression Trees - CART

There are three parameters involved in the process of "growing" a tree:

- How to define the constant predictors c_i in each sub-domain.

Classification and Regression Trees - CART

There are three parameters involved in the process of "growing" a tree:

- How to define the constant predictors c_i in each sub-domain.
- Which attribute to split in each step.

Classification and Regression Trees - CART

There are three parameters involved in the process of "growing" a tree:

- How to define the constant predictors c_i in each sub-domain.
- Which attribute to split in each step.
- How to find the splitting value.

Finding the Predictor

Finding the Predictor

Suppose the input domain is already partitioned in m sub-domains $R_1, \ldots, R_m$ and that the prediction model in the sub-domain R_i is a constant c_i, that is,

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i I(\mathbf{x} \in R_i)$$

Finding the Predictor

Suppose the input domain is already partitioned in m sub-domains $R_1, \ldots, R_m$ and that the prediction model in the sub-domain R_i is a constant c_i, that is,

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i I(\mathbf{x} \in R_i)$$

If the prediction is supposed to minimize the residual sum of squares

$$\arg\min_{f} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2$$

Finding the Predictor

Suppose the input domain is already partitioned in m sub-domains $R_1, \ldots, R_m$ and that the prediction model in the sub-domain R_i is a constant c_i, that is,

$$f(\mathbf{x}) = \sum_{i=1}^{m} c_i I(\mathbf{x} \in R_i)$$

If the prediction is supposed to minimize the residual sum of squares

$$\arg\min_{f} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2$$

It is easy to see that the minimizer is given by:

$$c_j = \texttt{ave}(y_i | \mathbf{x}_i \in R_j)$$

(regression trees)

Optimal Splitting Attribute and Value

Finding, among all possibilities, the binary partition that minimizes the sum of squares is typically unfeasible.

Optimal Splitting Attribute and Value

Finding, among all possibilities, the binary partition that minimizes the sum of squares is typically unfeasible.

The alternative is a greedy mechanism.

Optimal Splitting Attribute and Value

Finding, among all possibilities, the binary partition that minimizes the sum of squares is typically unfeasible.

The alternative is a greedy mechanism.

For each sub-domain R, we look for a partition $R = R'(j,s) \cup R''(j,s)$, where $R'(j,s) = \{\mathbf{x}|x_j \leq s\}$, $R''(j,s) = \{\mathbf{x}|x_j > s\}$, such that:

Optimal Splitting Attribute and Value

Finding, among all possibilities, the binary partition that minimizes the sum of squares is typically unfeasible.

The alternative is a greedy mechanism.

For each sub-domain R, we look for a partition $R = R'(j,s) \cup R''(j,s)$, where $R'(j,s) = \{\mathbf{x}|x_j \leq s\}$, $R''(j,s) = \{\mathbf{x}|x_j > s\}$, such that:

$$\min_{j,s}(\min_{c'} \sum_{\mathbf{x}_i \in R'(j,s)} (y_i - c')^2 + \min_{c''} \sum_{\mathbf{x}_i \in R''(j,s)} (y_i - c'')^2)$$

Optimal Splitting Attribute and Value

Finding, among all possibilities, the binary partition that minimizes the sum of squares is typically unfeasible.

The alternative is a greedy mechanism.

For each sub-domain R, we look for a partition $R = R'(j,s) \cup R''(j,s)$, where $R'(j,s) = \{\mathbf{x}|x_j \leq s\}$, $R''(j,s) = \{\mathbf{x}|x_j > s\}$, such that:

$$\min_{j,s}(\underbrace{\min_{c'} \sum_{\mathbf{x}_i \in R'(j,s)} (y_i - c')^2}_{c' = \mathtt{ave}\{y_i|\mathbf{x}_i \in R'(j,s)\}} + \underbrace{\min_{c''} \sum_{\mathbf{x}_i \in R''(j,s)} (y_i - c'')^2}_{c'' = \mathtt{ave}\{y_i|\mathbf{x}_i \in R''(j,s)\}}) \qquad (1)$$

Optimal Splitting Attribute and Value

Finding, among all possibilities, the binary partition that minimizes the sum of squares is typically unfeasible.

The alternative is a greedy mechanism.

For each sub-domain R, we look for a partition $R = R'(j,s) \cup R''(j,s)$, where $R'(j,s) = \{\mathbf{x}|x_j \leq s\}$, $R''(j,s) = \{\mathbf{x}|x_j > s\}$, such that:

$$\min_{j,s}(\underbrace{\min_{c'} \sum_{\mathbf{x}_i \in R'(j,s)} (y_i - c')^2}_{c' = \texttt{ave}\{y_i|\mathbf{x}_i \in R'(j,s)\}} + \underbrace{\min_{c''} \sum_{\mathbf{x}_i \in R''(j,s)} (y_i - c'')^2}_{c'' = \texttt{ave}\{y_i|\mathbf{x}_i \in R''(j,s)\}}) \qquad (1)$$

Fixing j, the optimal s can be found by scanning through all the inputs.

Optimal Splitting Attribute and Value

Finding, among all possibilities, the binary partition that minimizes the sum of squares is typically unfeasible.

The alternative is a greedy mechanism.

For each sub-domain R, we look for a partition $R = R'(j,s) \cup R''(j,s)$, where $R'(j,s) = \{\mathbf{x}|x_j \leq s\}$, $R''(j,s) = \{\mathbf{x}|x_j > s\}$, such that:

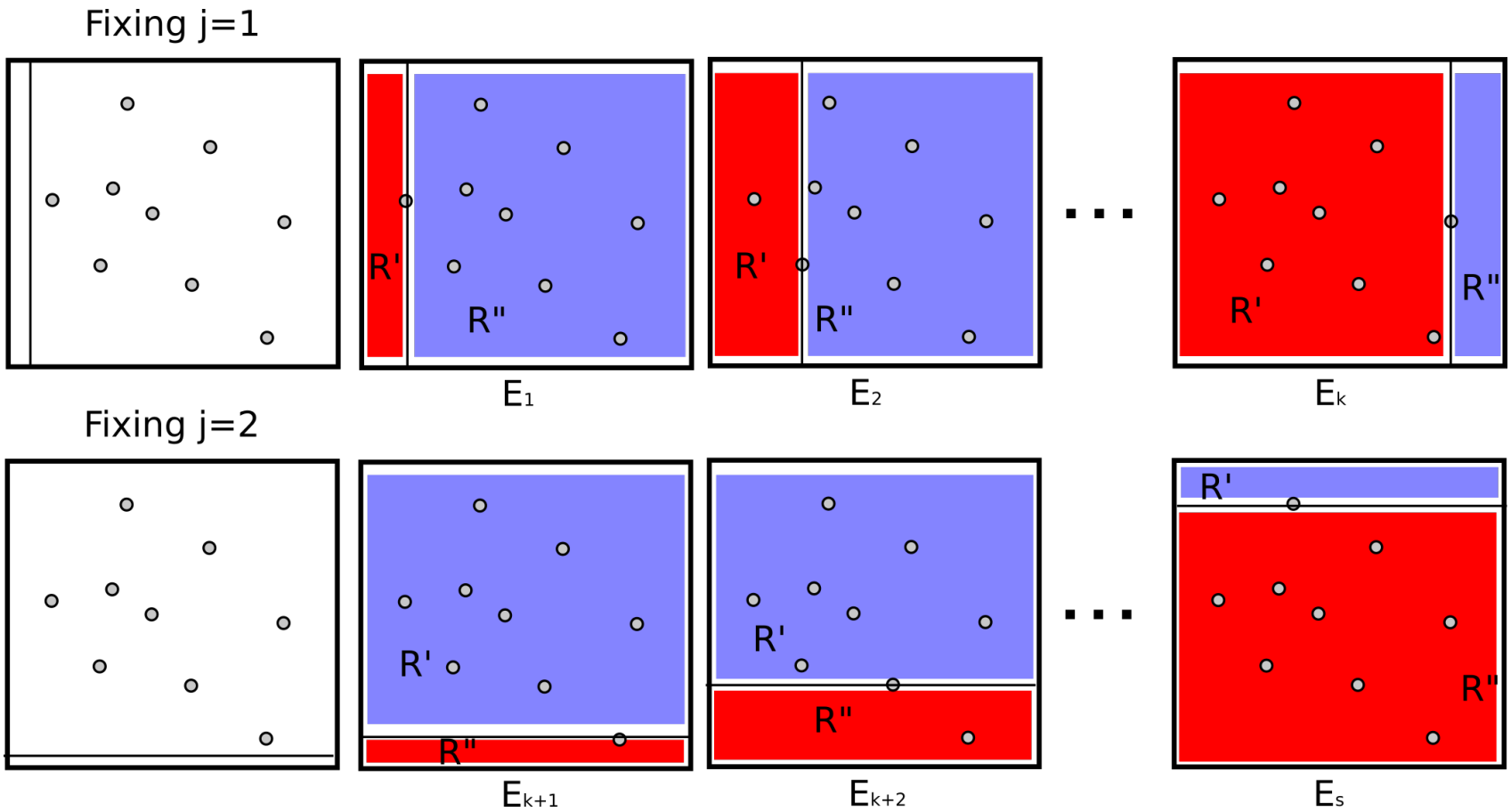
$$\min_{j,s}(\underbrace{\min_{c'} \sum_{\mathbf{x}_i \in R'(j,s)} (y_i - c')^2}_{c' = \texttt{ave}\{y_i|\mathbf{x}_i \in R'(j,s)\}} + \underbrace{\min_{c''} \sum_{\mathbf{x}_i \in R''(j,s)} (y_i - c'')^2}_{c'' = \texttt{ave}\{y_i|\mathbf{x}_i \in R''(j,s)\}}) \qquad (1)$$

Fixing j, the optimal s can be found by scanning through all the inputs.

The pair (j,s) minimizing (1) is chosen as the optimal one for R.

Optimal Splitting Attribute and Value

$$E_i = \Big(\sum_{\mathbf{x}_i \in R'(j,s)} (y_i - c')^2 + \sum_{\mathbf{x}_i \in R''(j,s)} (y_i - c'')^2 \Big)$$



Tree Size

Starting from the whole training data, the splitting process is repeated for all sub-domains.

Tree Size

Starting from the whole training data, the splitting process is repeated for all sub-domains.

How far should we grow the tree?

Tree Size

Starting from the whole training data, the splitting process is repeated for all sub-domains.

How far should we grow the tree?

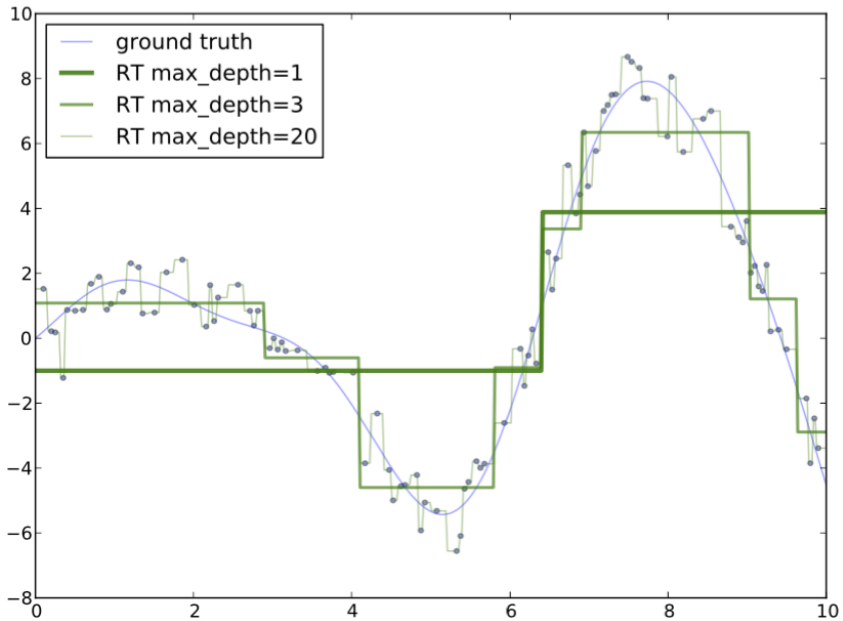
Tree size is a tuning parameter governing the model's complexity.

Tree Size

Starting from the whole training data, the splitting process is repeated for all sub-domains.

How far should we grow the tree?

Tree size is a tuning parameter governing the model's complexity.

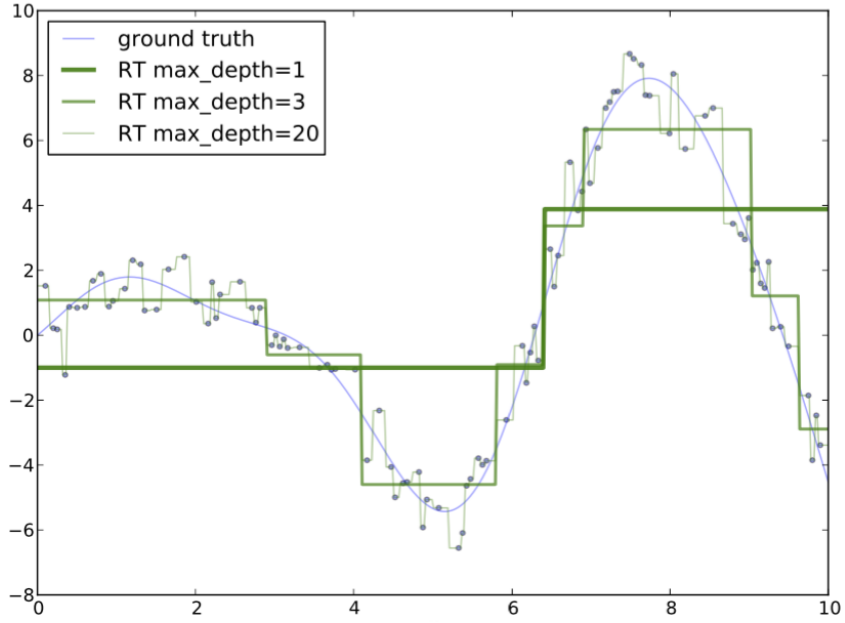


Tree Size

Starting from the whole training data, the splitting process is repeated for all sub-domains.

How far should we grow the tree?

Tree size is a tuning parameter governing the model's complexity.



A common procedure is to grow the tree until a maximum node size (say 5) is reached in each sub-domain and then prune the tree using a *cost-complexity pruning*.

Classification Tree

Classification trees can be build similarly to regression trees.

Classification Tree

Classification trees can be build similarly to regression trees.

Supposing k classes, the classification of a point $\mathbf{x} \in R_j$ is given by majority voting, that is,

$$k(j) = \arg\max_k p_{jk} = \arg\max_k \frac{1}{n_j} \sum_{\mathbf{x}_i \in R_j} I(y_i = k)$$

where p_{jk} is the proportion of class k in the region R_j.

Bagging, Random Forest, and Boosting

CART are seldom used alone!! They are employed with:

CART are seldom used alone!! They are employed with:

Bagging

Random Forest

Boosting

Bootstrap

Most committee-based approaches, such as Bagging and Random Forest rely on the *Bootstrap* mechanism.

Bootstrap

Most committee-based approaches, such as Bagging and Random Forest rely on the *Bootstrap* mechanism.

The bootstrap is a re-sampling scheme employed to assess statistical accuracy.

Bootstrap

Most committee-based approaches, such as Bagging and Random Forest rely on the *Bootstrap* mechanism.

The bootstrap is a re-sampling scheme employed to assess statistical accuracy.

Basic idea:

Bootstrap

Most committee-based approaches, such as Bagging and Random Forest rely on the *Bootstrap* mechanism.

The bootstrap is a re-sampling scheme employed to assess statistical accuracy.

Basic idea:

- Randomly draw *bootstrap* datasets with replacement from the training data;

Bootstrap

Most committee-based approaches, such as Bagging and Random Forest rely on the *Bootstrap* mechanism.

The bootstrap is a re-sampling scheme employed to assess statistical accuracy.

Basic idea:

- Randomly draw *bootstrap* datasets with replacement from the training data;
- Each bootstrap dataset the same size as the original training set;

Bootstrap

Most committee-based approaches, such as Bagging and Random Forest rely on the *Bootstrap* mechanism.

The bootstrap is a re-sampling scheme employed to assess statistical accuracy.

Basic idea:

- Randomly draw *bootstrap* datasets with replacement from the training data;
- Each bootstrap dataset the same size as the original training set;
- A number B of bootstrap datasets are generated;

Bootstrap

Most committee-based approaches, such as Bagging and Random Forest rely on the *Bootstrap* mechanism.

The bootstrap is a re-sampling scheme employed to assess statistical accuracy.

Basic idea:

- Randomly draw *bootstrap* datasets with replacement from the training data;
- Each bootstrap dataset the same size as the original training set;
- A number B of bootstrap datasets are generated;
- The model is fitted in each dataset and the behaviour examined over the B replications.

Bagging

Decision trees typically suffer from high variance.

Bagging

Decision trees typically suffer from high variance.

Suppose we have $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$, each with variance σ^2.

Bagging

Decision trees typically suffer from high variance.

Suppose we have $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$, each with variance σ^2.

It is known that the variance of the mean $\overline{\mathbf{Z}}$ is $\frac{\sigma^2}{n}$.

Bagging

Decision trees typically suffer from high variance.

Suppose we have $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$, each with variance σ^2.

It is known that the variance of the mean $\overline{\mathbf{Z}}$ is $\frac{\sigma^2}{n}$.

In other words, averaging a set of observations reduces variance.

Bagging

Decision trees typically suffer from high variance.

Suppose we have $\mathbf{Z}_1, \ldots, \mathbf{Z}_n$, each with variance σ^2.

It is known that the variance of the mean $\overline{\mathbf{Z}}$ is $\frac{\sigma^2}{n}$.

In other words, averaging a set of observations reduces variance.

This reasoning applies to any regression or classification procedure, not only for trees.

Bagging

The *Bagging* procedure:

1. Compute B bootstrapped training sets $b_1, \ldots, b_B$
2. Train a model f^i (grow the tree) in each b_i
3. Given a new input $\mathbf{x}$
   - Regression:
     Predicts $\hat{f}(\mathbf{x}) = \frac{1}{B} \sum_{i=1}^{B} f^{b_i}(\mathbf{x})$
   - Classification:
     Predicts $\hat{f}(\mathbf{x})$ by majority voting among the f^{b_i} predictions

The *Bagging* procedure:

1. Compute B bootstrapped training sets $b_1, \ldots, b_B$
2. Train a model f^i (grow the tree) in each b_i
3. Given a new input $\mathbf{x}$
   - Regression:
     Predicts $\hat{f}(\mathbf{x}) = \frac{1}{B} \sum_{i=1}^{B} f^{b_i}(\mathbf{x})$
   - Classification:
     Predicts $\hat{f}(\mathbf{x})$ by majority voting among the f^{b_i} predictions

Out-of-Bag Error Estimation

Bagging

The *Bagging* procedure:

1. Compute B bootstrapped training sets $b_1, \ldots, b_B$
2. Train a model f^i (grow the tree) in each b_i
3. Given a new input $\mathbf{x}$
   - Regression:
     Predicts $\hat{f}(\mathbf{x}) = \frac{1}{B} \sum_{i=1}^{B} f^{b_i}(\mathbf{x})$
   - Classification:
     Predicts $\hat{f}(\mathbf{x})$ by majority voting among the f^{b_i} predictions

Out-of-Bag Error Estimation

- On average, each bootstrapped training data use about two-thirds of the observations.

The *Bagging* procedure:

1. Compute B bootstrapped training sets $b_1, \ldots, b_B$
2. Train a model f^i (grow the tree) in each b_i
3. Given a new input $\mathbf{x}$
   - Regression:
     Predicts $\hat{f}(\mathbf{x}) = \frac{1}{B}\sum_{i=1}^{B} f^{b_i}(\mathbf{x})$
   - Classification:
     Predicts $\hat{f}(\mathbf{x})$ by majority voting among the f^{b_i} predictions

Out-of-Bag Error Estimation

- On average, each bootstrapped training data use about two-thirds of the observations.
- Testing in a particular observation is accomplished by predicting using only the models (trees) trained on bootstraptraining data in which the particular observation does not appear.

Bagging

The *Bagging* procedure:

1. Compute B bootstrapped training sets $b_1, \ldots, b_B$
2. Train a model f^i (grow the tree) in each b_i
3. Given a new input $\mathbf{x}$
   - Regression:
     Predicts $\hat{f}(\mathbf{x}) = \frac{1}{B}\sum_{i=1}^{B} f^{b_i}(\mathbf{x})$
   - Classification:
     Predicts $\hat{f}(\mathbf{x})$ by majority voting among the f^{b_i} predictions

Out-of-Bag Error Estimation

- On average, each bootstrapped training data use about two-thirds of the observations.
- Testing in a particular observation is accomplished by predicting using only the models (trees) trained on bootstraptraining data in which the particular observation does not appear.
- This avoid the use of cross-validation.

Random Forest

Random forests provide an improvement over bagged trees.

Random Forest

Random forests provide an improvement over bagged trees.

- Bootstrapped training data is used to grow a set of trees.

Random Forest

Random forests provide an improvement over bagged trees.

- Bootstrapped training data is used to grow a set of trees.
- However, only a random sample of m attributes are considered as split candidates, making remaining $d - m$ attributes "unseen".

Random Forest

Random forests provide an improvement over bagged trees.

- Bootstrapped training data is used to grow a set of trees.
- However, only a random sample of m attributes are considered as split candidates, making remaining $d - m$ attributes "unseen".
- Predictions are made as in the bagging case.

Random Forest

Random forests provide an improvement over bagged trees.

- Bootstrapped training data is used to grow a set of trees.
- However, only a random sample of m attributes are considered as split candidates, making remaining $d - m$ attributes "unseen".
- Predictions are made as in the bagging case.

The reasoning is to reduce the impact of strong attributes in the tree construction.

Random Forest

Random forests provide an improvement over bagged trees.

- Bootstrapped training data is used to grow a set of trees.
- However, only a random sample of m attributes are considered as split candidates, making remaining $d - m$ attributes "unseen".
- Predictions are made as in the bagging case.

The reasoning is to reduce the impact of strong attributes in the tree construction.

- Strong attributes tend to be used by most or all of the bagged trees, making then similar

Random Forest

Random forests provide an improvement over bagged trees.

- Bootstrapped training data is used to grow a set of trees.
- However, only a random sample of m attributes are considered as split candidates, making remaining $d - m$ attributes "unseen".
- Predictions are made as in the bagging case.

The reasoning is to reduce the impact of strong attributes in the tree construction.

- Strong attributes tend to be used by most or all of the bagged trees, making then similar
- Therefore, predictions from distinct trees tend to be highly correlated

Random Forest

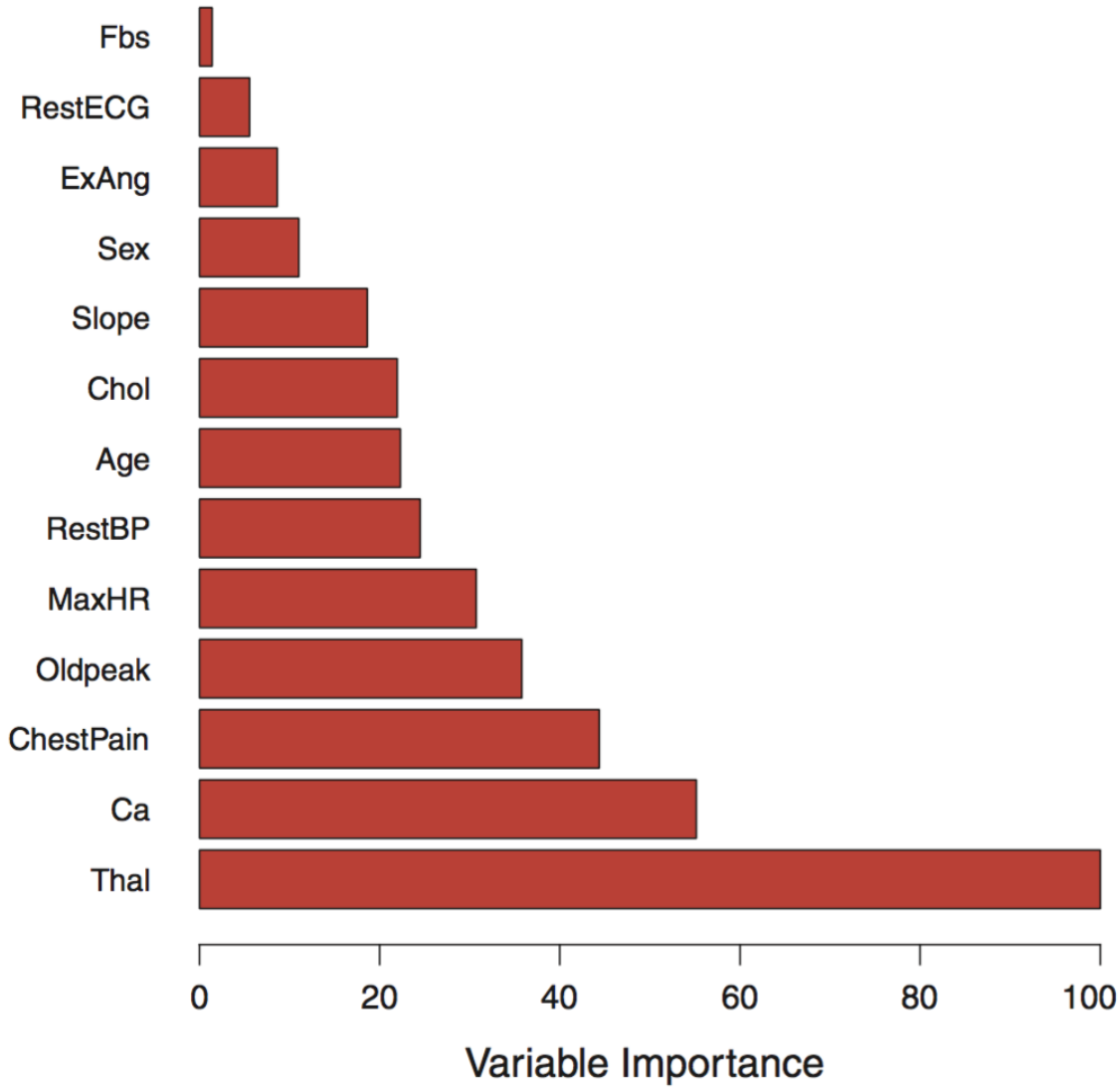
Random forests provide an improvement over bagged trees.

- Bootstrapped training data is used to grow a set of trees.
- However, only a random sample of m attributes are considered as split candidates, making remaining $d - m$ attributes "unseen".
- Predictions are made as in the bagging case.

The reasoning is to reduce the impact of strong attributes in the tree construction.

- Strong attributes tend to be used by most or all of the bagged trees, making then similar
- Therefore, predictions from distinct trees tend to be highly correlated
- The effect of bagging in correlated variables is not so good (does not substantially reduce variance)

Random Forest



Boosting

The idea of *Boosting* is to combining the outputs of many "weak" classifiers to produce a powerful committee.

Boosting

The idea of *Boosting* is to combining the outputs of many "weak" classifiers to produce a powerful committee.

In contrast to bagging, boosting grows trees sequentially, using information from previously grown trees to build a new one.

Boosting

The idea of *Boosting* is to combining the outputs of many "weak" classifiers to produce a powerful committee.

In contrast to bagging, boosting grows trees sequentially, using information from previously grown trees to build a new one.

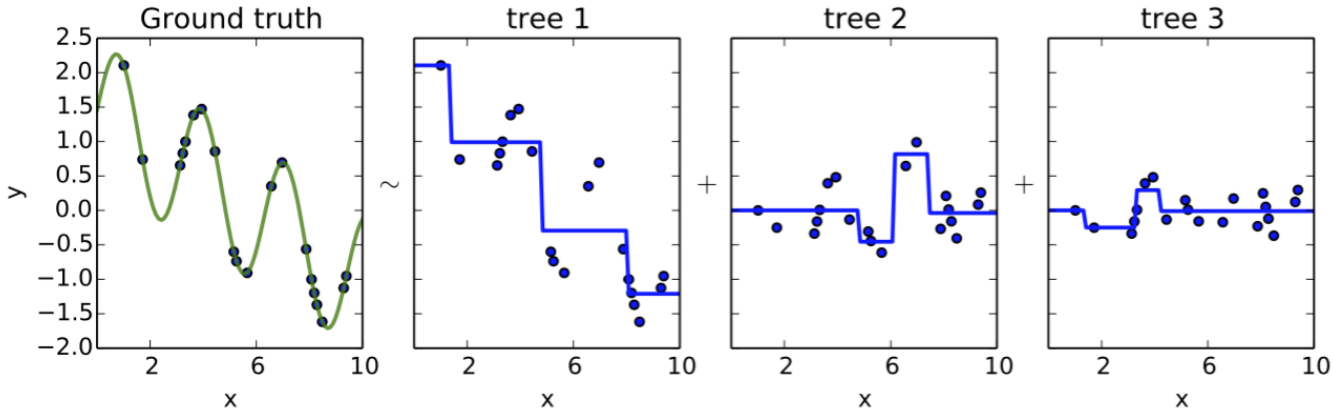
The idea is to start with a fairly simple model (initial tree) and fit new trees based on the residual of that model, updating the model based on the residuals.

Boosting

The idea of *Boosting* is to combining the outputs of many "weak" classifiers to produce a powerful committee.

In contrast to bagging, boosting grows trees sequentially, using information from previously grown trees to build a new one.

The idea is to start with a fairly simple model (initial tree) and fit new trees based on the residual of that model, updating the model based on the residuals.



There are many variations of the boosting idea.

Boosting

There are many variations of the boosting idea.

AdaBoost is a well known version for classification.

Boosting

There are many variations of the boosting idea.

AdaBoost is a well known version for classification.

The idea is to weight training samples according to their relevance to reduce prediction error.

Boosting

Boosted trees tend to present good results in practice.