Introduction to Machine Learning

Regression and Classification Trees

Andres Mendez-Vazquez

July 8, 2018

Outline

Cinvestav

Outline

Cinvestav

Decision Trees

Powerful/popular

For classification and prediction.

Decision Trees

Powerful/popular

For classification and prediction.

Represent rules

- Rules can be expressed in English.
  - **IF** $Age \leq 43$ & $Sex == Male$ **AND** $Credit\ Card\ Insurance == No$ **THEN**

$$Life\ Insurance\ Promotion = No$$

- Rules can be expressed using SQL for query.

Decision Trees

Powerful/popular

For classification and prediction.

Represent rules

- Rules can be expressed in English.
  - **IF** $Age \leq 43$ & $Sex == Male$ **AND** $Credit\ Card\ Insurance == No$ **THEN**

$$Life\ Insurance\ Promotion = No$$

- Rules can be expressed using SQL for query.

Useful to explore data to gain insight into relationships

Of a large number of candidate input variables to a target (output) variable.

Decision Trees

Powerful/popular

For classification and prediction.

Represent rules

- Rules can be expressed in English.
  - **IF** $Age \leq 43$ & $Sex == Male$ **AND** $Credit\ Card\ Insurance == No$ **THEN**

$$Life\ Insurance\ Promotion = No$$

- Rules can be expressed using SQL for query.

Useful to explore data to gain insight into relationships

Of a large number of candidate input variables to a target (output) variable.

Decision Trees

Powerful/popular

For classification and prediction.

Represent rules

- Rules can be expressed in English.
  - **IF** $Age \leq 43$ & $Sex == Male$ **AND** $Credit\ Card\ Insurance == No$ **THEN**

$$Life\ Insurance\ Promotion = No$$

- Rules can be expressed using SQL for query.

Useful to explore data to gain insight into relationships

Of a large number of candidate input variables to a target (output) variable.

What are They?

Decision Tree

A structure that can be used to divide up a large collection of records into successively smaller sets of records by applying a sequence of simple decision rules.

A decision tree model

Consists of a set of rules for dividing a large heterogeneous population into smaller, more homogeneous groups with respect to a particular target variable.

What are They?

Decision Tree

A structure that can be used to divide up a large collection of records into successively smaller sets of records by applying a sequence of simple decision rules.

A decision tree model

Consists of a set of rules for dividing a large heterogeneous population into smaller, more homogeneous groups with respect to a particular target variable.

Decision Tree Types

Binary trees

- Only two choices in each split. Can be non-uniform (uneven) in depth.

N-way trees or Ternary trees

- Three or more choices in at least one of its splits (3-way, 4-way, etc.).

Decision Tree Types

Binary trees

- Only two choices in each split. Can be non-uniform (uneven) in depth.

N-way trees or Ternary trees

- Three or more choices in at least one of its splits (3-way, 4-way, etc.).

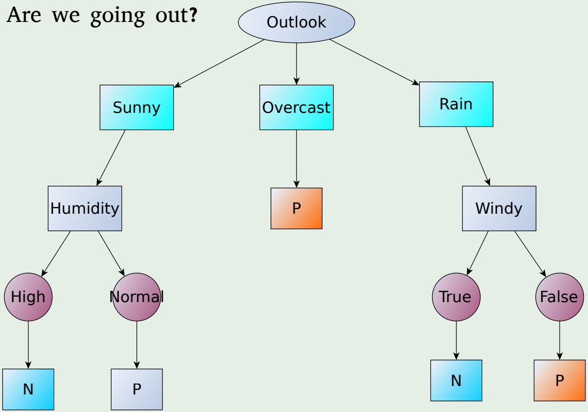
Outline

Cinvestav

An Example

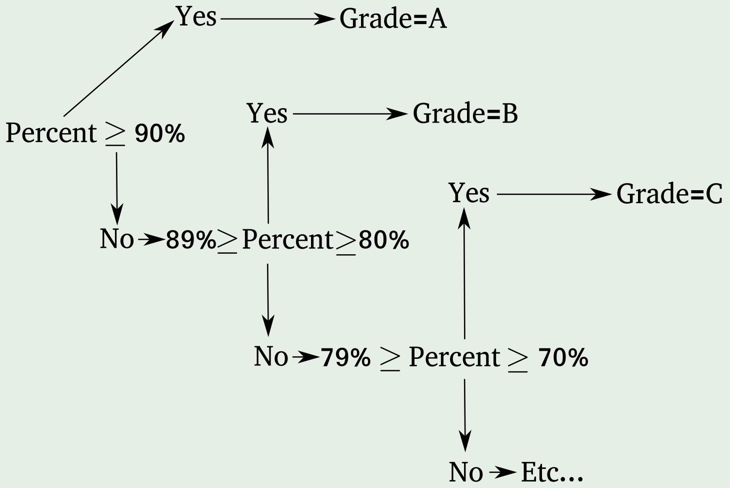
We have

Are we going out?



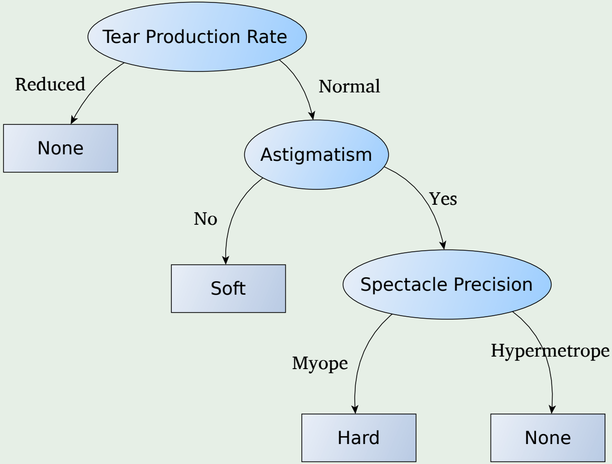
Another Example - Grades

Deciding the grades



Yet Another Example

Decision About Needing Glasses



Outline

Cinvestav

First than anything

Assume

Consider a Regression Problem with:

1. Continuous Response y.
2. Inputs x_1 and x_2 taking values in $[0, 1]$.
3. We have only recursive binary decisions/partitions.

First than anything

Assume

Consider a Regression Problem with:

1. Continuous Response y.
2. Inputs x_1 and x_2 taking values in $[0, 1]$.
3. We have only recursive binary decisions/partitions.

Example of a partition

First than anything

Assume

Consider a Regression Problem with:

1. Continuous Response y.
2. Inputs x_1 and x_2 taking values in $[0, 1]$.
3. We have only recursive binary decisions/partitions.

Example of a partition

First than anything

Assume

Consider a Regression Problem with:

1. Continuous Response y.
2. Inputs x_1 and x_2 taking values in $[0, 1]$.
3. We have only recursive binary decisions/partitions.

Example of a partition

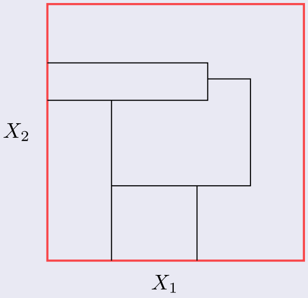
First than anything

Assume

Consider a Regression Problem with:

1. Continuous Response y.
2. Inputs x_1 and x_2 taking values in $[0, 1]$.
3. We have only recursive binary decisions/partitions.

Example of a partition



Everything is fine, but!!!

Although

- In each partition element we can model Y with a different constant.

- The Resulting Regions are difficult to describe!!!

Cinvestav

Everything is fine, but!!!

Although

- In each partition element we can model Y with a different constant.

There is a problem

- Each partitioning line has a simple description like $x_1 = c$!!!
- The Resulting Regions are difficult to describe!!!

Everything is fine, but!!!

Although

- In each partition element we can model Y with a different constant.

There is a problem

- Each partitioning line has a simple description like $x_1 = c$!!!
- The Resulting Regions are difficult to describe!!!

Everything is fine, but!!!

Although

- In each partition element we can model Y with a different constant.

There is a problem

- Each partitioning line has a simple description like $x_1 = c$!!!
- The Resulting Regions are difficult to describe!!!

Solving the Issue

We do the following

- Chose a variable and split the space using $x_i = c$

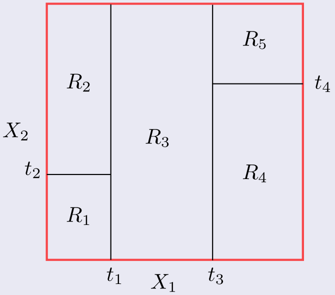
Keep doing that using one of the variables until a rules stops the process

Solving the Issue

We do the following

- Chose a variable and split the space using $x_i = c$

Keep doing that using one of the variables until a rules stops the process



The corresponding Regression Tree

We have

$$\widehat{y} = f(\boldsymbol{x}) = \sum_{m=1}^{5} c_m I\left\{(x_1, x_2) \in R_m\right\}$$

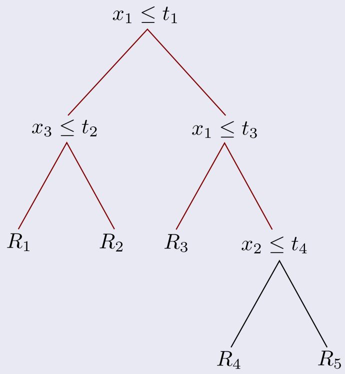
This regression can be interpreted as

The corresponding Regression Tree

We have

$$\widehat{y} = f\left(\boldsymbol{x}\right) = \sum_{m=1}^{5} c_m I\left\{(x_1, x_2) \in R_m\right\}$$

This regression can be interpreted as



Outline

Cinvestav

Structure

Structure

- Nodes
  - Appear as rectangles or circles
  - Represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes.

- Internal nodes are used for decisions
- Terminal Nodes or Leaves are the final results

Structure

Structure

- Nodes
  - Appear as rectangles or circles
  - Represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes.

Nodes

- Top or starting node is root node
- Internal nodes are used for decisions
- Terminal Nodes or Leaves are the final results

Cinvestav

Structure

Structure

- Nodes
  - Appear as rectangles or circles
  - Represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes.

Nodes

- Top or starting node is root node
- Internal nodes are used for decisions
- Terminal Nodes or Leaves are the final results

Structure

Structure

- Nodes
  - Appear as rectangles or circles
  - Represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes.

Nodes

- Top or starting node is root node
- Internal nodes are used for decisions
- Terminal Nodes or Leaves are the final results

Cinvestav

Structure

Structure

- Nodes
  - Appear as rectangles or circles
  - Represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes.

Nodes

- Top or starting node is root node
- Internal nodes are used for decisions
- Terminal Nodes or Leaves are the final results

Cinvestav

Structure

Structure

- Nodes
  - Appear as rectangles or circles
  - Represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes.

Nodes

- Top or starting node is root node
- Internal nodes are used for decisions
- Terminal Nodes or Leaves are the final results

Structure

Structure

- Nodes
  - Appear as rectangles or circles
  - Represent test or decision
- Lines or branches - represent outcome of a test
- Circles - terminal (leaf) nodes.

Nodes

- Top or starting node is root node
- Internal nodes are used for decisions
- Terminal Nodes or Leaves are the final results

Outline

Cinvestav

Types of Decision Trees

Regression Trees

The predicted outcome can be considered a number.

Classification Trees

- The predicted outcome is the class to which the data belongs.

Types of Decision Trees

Regression Trees

The predicted outcome can be considered a number.

Classification Trees

- The predicted outcome is the class to which the data belongs.

Classification and Regression Trees (CART)

CART

- The term CART is an umbrella term used to refer to both of the above procedures.

Introduced by

- It was introduced by Breiman et. al in the book
  - **"Classification and Regression Trees"**

Similarities

- Regression and Classification trees have some similarities – nevertheless they differ in the way the splitting at each node is done.

Classification and Regression Trees (CART)

CART

- The term CART is an umbrella term used to refer to both of the above procedures.

Introduced by

- It was introduced by Breiman et. al in the book
  - **“Classification and Regression Trees”**

Similarities

- Regression and Classification trees have some similarities – nevertheless they differ in the way the splitting at each node is done.

Cinvestav

Classification and Regression Trees (CART)

CART

- The term CART is an umbrella term used to refer to both of the above procedures.

Introduced by

- It was introduced by Breiman et. al in the book
  - **"Classification and Regression Trees"**

Similarities

- Regression and Classification trees have some similarities – nevertheless they differ in the way the splitting at each node is done.

Cinvestav

Outline

Cinvestav

Setup

Data Consists on inputs of dimensionality d

$$\left\{(x_i, y_i)_{i=1}^{N}\right\}$$

Where $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$.

Here, we want an algorithm

- To do the splitting automatically

Thus, assume a initial M partition $R_1, R_2, ..., R_M$

- We model the response as a constant c_m in each region

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m I(\boldsymbol{x} \in R_m)$$

Setup

Data Consists on inputs of dimensionality d

$$\left\{ (x_i, y_i)_{i=1}^{N} \right\}$$

Where $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$.

Here, we want an algorithm

- To do the splitting automatically

Thus, assume a initial M partition $R_1, R_2, ..., R_M$

- We model the response as a constant c_m in each region

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

Setup

Data Consists on inputs of dimensionality d

$$\left\{ (x_i, y_i)_{i=1}^{N} \right\}$$

Where $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$.

Here, we want an algorithm

- To do the splitting automatically

Thus, assume a initial M partition $R_1, R_2, ..., R_M$

- We model the response as a constant c_m in each region

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m I(\boldsymbol{x} \in R_m)$$

Cinvestav

Outline

We have then

We adopt as our criterion minimization

$$L\left(c_1, c_2, ..., c_M\right) = \sum_{i=1}^{N} \sum_{i=1}^{M} \left(y_i - f\left(\boldsymbol{x}_i\right)\right)^2$$

Then using a classic derivative with respect to c_m

$$\frac{\partial L\left(c_1, c_2, ..., c_M\right)}{\partial c_m} = -2 \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m I\left(\boldsymbol{x}_i \in R_m\right)\right) I\left(\boldsymbol{x}_i \in R_m\right)$$

Then

$$\sum_{y_i | \boldsymbol{x}_i \in R_m} y_i - \sum_{i=1}^{N} I\left(\boldsymbol{x}_i \in R_m\right) \sum_{m=1}^{M} c_m I\left(\boldsymbol{x}_i \in R_m\right) = 0$$

We have then

We adopt as our criterion minimization

$$L\left(c_1, c_2, ..., c_M\right) = \sum_{i=1}^{N} \sum_{i=1}^{M} \left(y_i - f\left(\boldsymbol{x}_i\right)\right)^2$$

Then using a classic derivative with respect to c_m

$$\frac{\partial L\left(c_1, c_2, ..., c_M\right)}{\partial c_m} = -2 \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m I\left(\boldsymbol{x}_i \in R_m\right)\right) I\left(\boldsymbol{x}_i \in R_m\right)$$

Then

$$\sum_{y_i|\boldsymbol{x}_i \in R_m} y_i - \sum_{i=1}^{N} I\left(\boldsymbol{x}_i \in R_m\right) \sum_{m=1}^{M} c_m I\left(\boldsymbol{x}_i \in R_m\right) = 0$$

We have then

We adopt as our criterion minimization

$$L\left(c_1, c_2, \ldots, c_M\right) = \sum_{i=1}^{N} \sum_{i=1}^{M} \left(y_i - f\left(\boldsymbol{x}_i\right)\right)^2$$

Then using a classic derivative with respect to c_m

$$\frac{\partial L\left(c_1, c_2, \ldots, c_M\right)}{\partial c_m} = -2 \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M} c_m I\left(\boldsymbol{x}_i \in R_m\right)\right) I\left(\boldsymbol{x}_i \in R_m\right)$$

Then

$$\sum_{y_i | \boldsymbol{x}_i \in R_m} y_i - \sum_{i=1}^{N} I\left(\boldsymbol{x}_i \in R_m\right) \sum_{m=1}^{M} c_m I\left(\boldsymbol{x}_i \in R_m\right) = 0$$

The simplest function for c_m

Something Notable

$$\sum_{\boldsymbol{x}_i \in R_m} c_m = \sum_{y_i | \boldsymbol{x}_i \in R_m} y_i$$

Then

$$c_m = \frac{1}{N_m} \sum_{y_i | \boldsymbol{x}_i \in R_m} y_i$$

Problem

- Finding the best binary partition in terms of minimum sum of squares is generally $O\left(2^N\right)$ a NP Problem!!!

The simplest function for c_m

Something Notable

$$\sum_{\boldsymbol{x}_i \in R_m} c_m = \sum_{y_i | \boldsymbol{x}_i \in R_m} y_i$$

Then

$$c_m = \frac{1}{N_m} \sum_{y_i | \boldsymbol{x}_i \in R_m} y_i$$

Problem

- Finding the best binary partition in terms of minimum sum of squares is generally $O\left(2^N\right)$ a NP Problem!!!

The simplest function for c_m

Something Notable

$$\sum_{\boldsymbol{x}_i \in R_m} c_m = \sum_{y_i | \boldsymbol{x}_i \in R_m} y_i$$

Then

$$c_m = \frac{1}{N_m} \sum_{y_i | \boldsymbol{x}_i \in R_m} y_i$$

Problem

- Finding the best binary partition in terms of minimum sum of squares is generally $O\left(2^N\right)$ a NP Problem!!!

Cinvestav

What to do?

Consider a splitting variable j and split point s

- Define the pair of half-planes

$$R_1(j,s) = \{\boldsymbol{x} | x_j \leq s\} \text{ and } R_2(j,s) = \{\boldsymbol{x} | x_j > s\}$$

Using an Optimization Problem

$$\min_{j,s} \left\{ \min_{c_1} \sum_{\boldsymbol{x}_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\boldsymbol{x}_i \in R_2(j,s)} (y_i - c_2)^2 \right\}$$

The nice part of this

- For any choice j and s, the inner minimization is solved by

$$\widehat{c}_1 = \frac{1}{N_1} \sum_{y_i | \boldsymbol{x}_i \in R_1(j,s)} y_i \text{ and } \widehat{c}_2 = \frac{1}{N_1} \sum_{y_i | \boldsymbol{x}_i \in R_2(j,s)} y_i$$

What to do?

Consider a splitting variable j and split point s

- Define the pair of half-planes

$$R_1(j,s) = \{\boldsymbol{x} | x_j \leq s\} \text{ and } R_2(j,s) = \{\boldsymbol{x} | x_j > s\}$$

Using an Optimization Problem

$$\min_{j,s} \left\{ \min_{c_1} \sum_{\boldsymbol{x}_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\boldsymbol{x}_i \in R_2(j,s)} (y_i - c_2)^2 \right\}$$

The nice part of this

- For any choice j and s, the inner minimization is solved by

$$\widehat{c}_1 = \frac{1}{N_1} \sum_{y_i | \boldsymbol{x}_i \in R_1(j,s)} y_i \text{ and } \widehat{c}_2 = \frac{1}{N_1} \sum_{y_i | \boldsymbol{x}_i \in R_2(j,s)} y_i$$

What to do?

Consider a splitting variable j and split point s

- Define the pair of half-planes

$$R_1(j,s) = \{\boldsymbol{x} | x_j \leq s\} \text{ and } R_2(j,s) = \{\boldsymbol{x} | x_j > s\}$$

Using an Optimization Problem

$$\min_{j,s}\left\{\min_{c_1} \sum_{\boldsymbol{x}_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{\boldsymbol{x}_i \in R_2(j,s)} (y_i - c_2)^2\right\}$$

The nice part of this

- For any choice j and s, the inner minimization is solved by

$$\widehat{c}_1 = \frac{1}{N_1} \sum_{y_i | \boldsymbol{x}_i \in R_1(j,s)} y_i \text{ and } \widehat{c}_2 = \frac{1}{N_1} \sum_{y_i | \boldsymbol{x}_i \in R_2(j,s)} y_i$$

Therefore

For each splitting variable j

- Finding s is done quickly!!!

We can repeat this process

- Problem, we can finish with an over-fitting tree/a very large tree.

How do we solve?

- Tree size is an hyper-parameter governing the model's complexity.

Cinvestav

Therefore

For each splitting variable j

- Finding s is done quickly!!!

We can repeat this process

- Problem, we can finish with an over-fitting tree/a very large tree.

How do we solve?

- Tree size is an hyper-parameter governing the model's complexity.

Therefore

For each splitting variable j

- Finding s is done quickly!!!

We can repeat this process

- Problem, we can finish with an over-fitting tree/a very large tree.

How do we solve?

- Tree size is an hyper-parameter governing the model's complexity.

Therefore

We have that

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

A preferred strategy

- Grow the tree until some minimum size node is done.

Then

- This large tree is pruned using cost-complexity pruning.

Therefore

We have that

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

A preferred strategy

- Grow the tree until some minimum size node is done.

Then

- This large tree is pruned using cost-complexity pruning.

Therefore

We have that

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

A preferred strategy

- Grow the tree until some minimum size node is done.

Then

- This large tree is pruned using cost-complexity pruning.

Outline

Cinvestav

We need to define something

Definition

- We define a subtree $T \subseteq T_0$ to be any tree that can be obtained by pruning T_0:
  - By collapsing any number of its internal (non-terminal) nodes.

Given that each R_m is indexed by m

- Let $|T|$ denote the number of terminal nodes in T:

$$N_m = |R_m|,\ \hat{c}_m = \frac{1}{N_m}\sum_{y_i|x_i \in R_m} y_i \text{ and } Q_m(T) = \frac{1}{N_m}\left(\hat{c}_m - y_i\right)^2$$

We need to define something

Definition

- We define a subtree $T \subseteq T_0$ to be any tree that can be obtained by pruning T_0:
  - By collapsing any number of its internal (non-terminal) nodes.

Given that each R_m is indexed by m

- Let $|T|$ denote the number of terminal nodes in T:

$$N_m = |R_m|\,,\ \widehat{c}_m = \frac{1}{N_m} \sum_{y_i | \boldsymbol{x}_i \in R_m} y_i \text{ and } Q_m(T) = \frac{1}{N_m} \left(\widehat{c}_m - y_i\right)^2$$

Thus

Define the cost complexity criterion with $\alpha \geq 0$

$$C_{\alpha}(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|$$

Finally

- The idea is to find, for each α, the subtree $T_{\alpha} \subseteq T_0$ to minimize $C_{\alpha}(T)$

Properties of α

- Large values of α result in smaller T_{α}
- Small values of α result in larger T_{α}

Thus

Define the cost complexity criterion with $\alpha \geq 0$

$$C_\alpha (T) = \sum_{m=1}^{|T|} N_m Q_m (T) + \alpha |T|$$

Finally

- The idea is to find, for each α, the subtree $T_\alpha \subseteq T_0$ to minimize $C_\alpha (T)$

Properties of α

- Large values of α result in smaller T_α
- Small values of α result in larger T_α

Thus

Define the cost complexity criterion with $\alpha \geq 0$

$$C_{\alpha}(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|$$

Finally

- The idea is to find, for each α, the subtree $T_{\alpha} \subseteq T_0$ to minimize $C_{\alpha}(T)$

Properties of α

- Large values of α result in smaller T_{α}
- Small values of α result in larger T_{α}

Furthermore

For each α one can show the existence of unique smallest subtree T_α

- How do we find T_α?

Using weakest link pruning

- We successively collapse the internal node that produces the smallest per-node increase in

$$\sum_{m=1}^{|T|} N_m Q_m(T)$$

Until you get a single-node (root) and a sequence

$$T \supseteq T_1 \supseteq T_2 \supseteq \cdots \supseteq T_N$$

Furthermore

For each α one can show the existence of unique smallest subtree T_α

- How do we find T_α?

Using weakest link pruning

- We successively collapse the internal node that produces the smallest per-node increase in

$$\sum_{m=1}^{|T|} N_m Q_m(T)$$

Until you get a single-node (root) and a sequence

$$T \supset T_1 \supset T_2 \supset \cdots \supset T_N$$

Furthermore

For each α one can show the existence of unique smallest subtree T_α

- How do we find T_α?

Using weakest link pruning

- We successively collapse the internal node that produces the smallest per-node increase in

$$\sum_{m=1}^{|T|} N_m Q_m(T)$$

Until you get a single-node (root) and a sequence

$$T \supseteq T_1 \supseteq T_2 \supseteq \cdots \supseteq T_N$$

Then

We get that

- T_α is one of the threes in the in the sequence.

Estimation of α is achieved by cross-validation

- We choose the value $\widehat{\alpha}$ to minimize the cross-validated sum of squares.
  - This is the final $T_{\widehat{\alpha}}$

For Details

- "Pattern Recognition and Neural Networks" by Brian D. Ripley

Then

We get that

- T_α is one of the threes in the in the sequence.

Estimation of α is achieved by cross-validation

- We choose the value $\widehat{\alpha}$ to minimize the cross-validated sum of squares.
  - This is the final $T_{\widehat{\alpha}}$

For Details

- "Pattern Recognition and Neural Networks" by Brian D. Ripley

Then

We get that

- T_α is one of the threes in the in the sequence.

Estimation of α is achieved by cross-validation

- We choose the value $\widehat{\alpha}$ to minimize the cross-validated sum of squares.
  - This is the final $T_{\widehat{\alpha}}$

For Details

- "Pattern Recognition and Neural Networks" by Brian D. Ripley

Outline

Cinvestav

Important

Most of the work

It focuses on deciding which property test or query should be performed at the node!!!

If the data test is numerical in nature

There is a way to visualize the decision boundaries produced by the decision trees.

Important

Most of the work

It focuses on deciding which property test or query should be performed at the node!!!

If the data test is numerical in nature

There is a way to visualize the decision boundaries produced by the decision trees.

Definition OBCT

Definition

They are binary decision trees where the basic question is $x_i \leq a_i$?

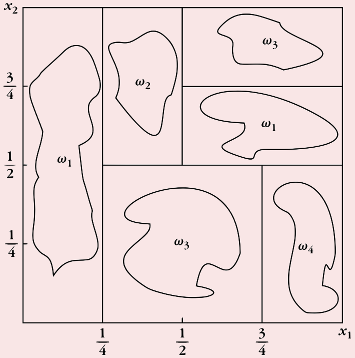
Example

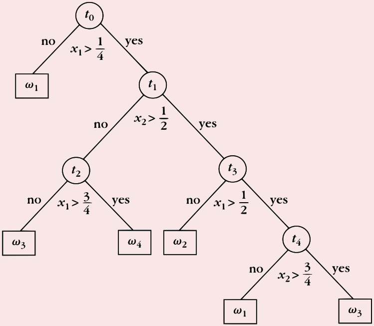
Definition OBCT

Definition

They are binary decision trees where the basic question is $x_i \leq a_i$?

Example





Outline

Training of a OBCT

We need first

- At each node, the set of candidate questions to be asked has to be decided.
- Each question corresponds to a specific binary split into two descendant nodes.
- Each node, t, is associated with a specific subset X_t of the training set X.

Training of a OBCT

We need first

- At each node, the set of candidate questions to be asked has to be decided.
- Each question corresponds to a specific binary split into two descendant nodes.
- Each node, t, is associated with a specific subset X_t of the training set X.

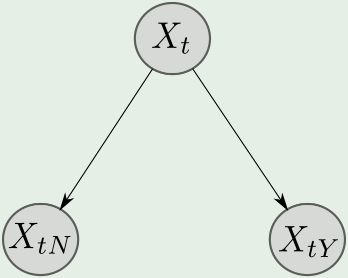
Training of a OBCT

We need first

- At each node, the set of candidate questions to be asked has to be decided.
- Each question corresponds to a specific binary split into two descendant nodes.
- Each node, t, is associated with a specific subset X_t of the training set X.

Splitting the Node X_t

Basically, we want to split the node into two groups with questions $t_Y == "YES"$ and $t_N = "NO"$

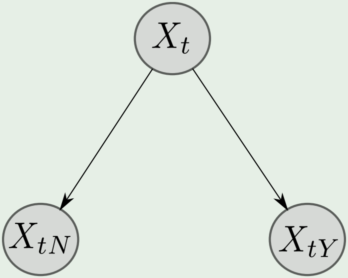


With Properties

- $X_{tY} \cap X_{tN} = \emptyset$.
- $X_{tY} \cup X_{tN} = X_t$

Splitting the Node X_t

Basically, we want to split the node into two groups with questions $t_Y == "YES"$ and $t_N = "NO"$



With Properties

- $X_{tY} \cap X_{tN} = \emptyset$.
- $X_{tY} \cup X_{tN} = X_t$

Important

Given the question for each feature k "Is $x_k \leq \alpha$"

For each feature, every possible value of the threshold α defines a specific split of the subset X_t.

Thus in theory

An infinite set of questions has to be asked if α is an interval $Y_\alpha \subseteq \mathbb{R}$.

In practice

only a finite set of questions can be considered.

Important

Given the question for each feature k "Is $x_k \leq \alpha$"

For each feature, every possible value of the threshold α defines a specific split of the subset X_t.

Thus in theory

An infinite set of questions has to be asked if α is an interval $Y_\alpha \subseteq \mathbb{R}$.

In practice

only a finite set of questions can be considered.

Important

Given the question for each feature k "Is $x_k \leq \alpha$"

For each feature, every possible value of the threshold α defines a specific split of the subset X_t.

Thus in theory

An infinite set of questions has to be asked if α is an interval $Y_\alpha \subseteq \mathbb{R}$.

In practice

only a finite set of questions can be considered.

Cinvestav

For example

Since the number, N, of training points in X is finite

Any of the features x_k with $k = 1, ..., l$ can take at most $N_t \leq N$ different values

Where

$N_t = |X_t|$ with $X_t \subset X$

Then

For feature x_k, one can use α_{kn} with $n = 1, 2, ..., N_{tk}$ and $N_{tk} \leq N_t$ where α_{kn} are taken halfway between consecutive distinct values of x_k in the training subset X_t.

For example

Since the number, N, of training points in X is finite

Any of the features x_k with $k = 1, ..., l$ can take at most $N_t \leq N$ different values

Where

$N_t = |X_t|$ with $X_t \subset X$

Then

For feature x_k, one can use α_{kn} with $n = 1, 2, ..., N_{tk}$ and $N_{tk} \leq N_t$ where α_{kn} are taken halfway between consecutive distinct values of x_k in the training subset X_t.

For example

Since the number, N, of training points in X is finite

Any of the features x_k with $k = 1, ..., l$ can take at most $N_t \leq N$ different values

Where

$N_t = |X_t|$ with $X_t \subset X$

Then

For feature x_k, one can use α_{kn} with $n = 1, 2, ..., N_{tk}$ and $N_{tk} \leq N_t$ where α_{kn} are taken halfway between consecutive distinct values of x_k in the training subset X_t.

Then

We repeat this with all features

In such a case, the total number of candidate questions is

$$\sum_{k=1}^{l} N_{tk} \tag{1}$$

Then

We repeat this with all features

In such a case, the total number of candidate questions is

$$\sum_{k=1}^{l} N_{tk} \tag{1}$$

However

Only one of them has to be chosen to provide the binary split at the current node, t, of the tree.

- The best split is decided according to a splitting criterion

Then

We repeat this with all features

In such a case, the total number of candidate questions is

$$\sum_{k=1}^{l} N_{tk} \tag{1}$$

However

Only one of them has to be chosen to provide the binary split at the current node, t, of the tree.

Thus

- This is selected to be the one that leads to the best split of the associated subset X_t.
- The best split is decided according to a splitting criterion.

Then

We repeat this with all features

In such a case, the total number of candidate questions is

$$\sum_{k=1}^{l} N_{tk} \tag{1}$$

However

Only one of them has to be chosen to provide the binary split at the current node, t, of the tree.

Thus

- This is selected to be the one that leads to the best split of the associated subset X_t.
- The best split is decided according to a **splitting criterion**.

Outline

Criterion's to be Found

Splitting criterion

- A splitting criterion must be adopted according to which the best split from the set of candidate ones is chosen.

Stop-splitting rule

A stop-splitting rule is required that controls the growth of the tree,and a node is declared as a terminal one (leaf).

Rule

A rule is required that assigns each leaf to a specific class.

Criterion's to be Found

Splitting criterion

- A splitting criterion must be adopted according to which the best split from the set of candidate ones is chosen.

Stop-splitting rule

A stop-splitting rule is required that controls the growth of the tree,and a node is declared as a terminal one (leaf).

Rule

A rule is required that assigns each leaf to a specific class.

Cinvestav

Criterion's to be Found

Splitting criterion

- A splitting criterion must be adopted according to which the best split from the set of candidate ones is chosen.

Stop-splitting rule

A stop-splitting rule is required that controls the growth of the tree,and a node is declared as a terminal one (leaf).

Rule

A rule is required that assigns each leaf to a specific class.

Looking for Homogeneity!!!

In order for the tree growing methodology

From the root node down to the leaves every split must generate a subsets that are more homogeneous compared to the ancestor's subset X_t.

Meaning

The training feature vectors in each one of the new subsets show, whereas data in X_t are more equally distributed among the classes.

For example

Consider the task of classifying four classes $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ and assume that the vectors in subset X_t are distributed among the classes with equal probability.

Looking for Homogeneity!!!

In order for the tree growing methodology

From the root node down to the leaves every split must generate a subsets that are more homogeneous compared to the ancestor's subset X_t.

Meaning

The training feature vectors in each one of the new subsets show, whereas data in X_t are more equally distributed among the classes.

For example

Consider the task of classifying four classes $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ and assume that the vectors in subset X_t are distributed among the classes with equal probability.

Looking for Homogeneity!!!

In order for the tree growing methodology

From the root node down to the leaves every split must generate a subsets that are more homogeneous compared to the ancestor's subset X_t.

Meaning

The training feature vectors in each one of the new subsets show, whereas data in X_t are more equally distributed among the classes.

For example

Consider the task of classifying four classes $\{\omega_1, \omega_2, \omega_3, \omega_4\}$ and assume that the vectors in subset X_t are distributed among the classes with equal probability.

Thus

If we split the node so

- ω_1 and ω_2 form X_{tY}
- ω_3 and ω_4 form X_{tN}

Then

X_{tY} and X_{tN} are more homogeneous compared to X_t.

In other words

"Purer" in the decision tree terminology.

Cinvestav

Thus

If we split the node so

- ω_1 and ω_2 form X_{tY}
- ω_3 and ω_4 form X_{tN}

Then

X_{tY} and X_{tN} are more homogeneous compared to X_t.

In other words

"Purer" in the decision tree terminology.

Thus

If we split the node so

- ω_1 and ω_2 form X_{tY}
- ω_3 and ω_4 form X_{tN}

Then

X_{tY} and X_{tN} are more homogeneous compared to X_t.

In other words

"Purer" in the decision tree terminology.

Our Goal

We need

To define a measure that quantifies node impurity.

Thus

The Overall Impurity of the descendant nodes is optimally decreased with respect to the ancestor node's impurity.

Our Goal

We need

To define a measure that quantifies node impurity.

Thus

The Overall Impurity of the descendant nodes is optimally decreased with respect to the ancestor node's impurity.

Outline

Cinvestav

Probabilistic Impurity

Assume the following probability of a vector in X_t belongs to class ω_i

$$P(\omega_i|t) \text{ for } i = 1, \cdots, M \tag{2}$$

Probabilistic Impurity

Assume the following probability of a vector in X_t belongs to class ω_i

$$P(\omega_i|t) \text{ for } i = 1, \cdots, M \quad (2)$$

A Common Impurity

We define one of the most common impurities

$$I(t) = -\sum_{i=1}^{M} P(\omega_i|t) \log_2 P(\omega_i|t)$$

This is nothing more than the Shannon's Entropy!!!

- Facts:
  - $I(t)$reaches its maximum when

$$P(\omega_i|t) = \frac{1}{M}$$

A Common Impurity

We define one of the most common impurities

$$I(t) = -\sum_{i=1}^{M} P(\omega_i|t) \log_2 P(\omega_i|t)$$

This is nothing more than the Shannon's Entropy!!!

- Facts:
  - $I(t)$reaches its maximum when
  $$P(\omega_i|t) = \frac{1}{M}$$
  - $I(t) = 0$ if all data belongs to a single class i.e.
  $P(\omega_i|t) = 1$ for only one class, and $P(\omega_j|t, j \neq i) = 0$ for everybody else.

A Common Impurity

We define one of the most common impurities

$$I(t) = -\sum_{i=1}^{M} P(\omega_i|t) \log_2 P(\omega_i|t)$$

This is nothing more than the Shannon's Entropy!!!

- Facts:
  - $I(t)$reaches its maximum when

$$P(\omega_i|t) = \frac{1}{M}$$

  - $I(t) = 0$ if all data belongs to a single class i.e.

    $P(\omega_i|t) = 1$ for only one class, and $P(\omega_j|t, j \neq i) = 0$ for everybody else.

In reality...

We estimate

$$P\left(\omega_i|t\right) = \frac{N_t^i}{N_t}$$

Where $|\omega_i| = N_t^i$ as the number of points in $X_t$that belongs to class ω_i.

Assume now

If we perform a split, N_{tY} points are sent into the "YES" node X_{tY} and N_{tN} into the "NO" node X_{tN}

Cinvestav

In reality...

We estimate

$$P\left(\omega_i|t\right) = \frac{N_t^i}{N_t}$$

Where $|\omega_i| = N_t^i$ as the number of points in $X_t$that belongs to class ω_i.

Assume now

If we perform a split, N_{tY} points are sent into the "YES" node X_{tY} and N_{tN} into the "NO" node X_{tN}

Decrease in node impurity

Then

In a recursive way we define the term decrease in node impurity as:

$$\Delta I(t) = I(t) - \frac{N_{tY}}{N_t} I(t_Y) - \frac{N_{tN}}{N_t} I(t_N) \tag{3}$$

where $I(t_Y)$ and $I(t_N)$ are the impurities of the t_Y and t_N nodes.

The Final Goal

The Final Goal

To adopt from the set of candidate questions the one that performs the split with the highest decrease of impurity.

Stop-Splitting Rule

Now

The natural question that now arises is when one decides to stop splitting a node and declares it as a leaf of the tree.

Stop-Splitting Rule

Now

The natural question that now arises is when one decides to stop splitting a node and declares it as a leaf of the tree.

For example you can adopt

A threshold T and stop splitting if the maximum value of $\Delta I(t)$ over all possible splits is less than T.

- If the subset X_t is pure, in the sense that all points in it belong to a single class.

Stop-Splitting Rule

Now

The natural question that now arises is when one decides to stop splitting a node and declares it as a leaf of the tree.

For example you can adopt

A threshold T and stop splitting if the maximum value of $\Delta I(t)$ over all possible splits is less than T.

Other posibilities

- If the subset X_t is small enough.
- If the subset X_t is pure, in the sense that all points in it belong to a single class.

Cinvestav

Stop-Splitting Rule

Now

The natural question that now arises is when one decides to stop splitting a node and declares it as a leaf of the tree.

For example you can adopt

A threshold T and stop splitting if the maximum value of $\Delta I(t)$ over all possible splits is less than T.

Other posibilities

- If the subset X_t is small enough.
- If the subset X_t is pure, in the sense that all points in it belong to a single class.

Cinvestav

Once a node is declared to be a leaf

Class Assignment Rule

Once a node is declared a leaf, we assign the leaf to a class using the rule:

$$j = \arg\max_{i} P\left(\omega_i | t\right).$$

Outline

Cinvestav

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN} ,
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN}
.
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Cinvestav

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r.t. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN} ,
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN} ,
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r.t. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN} ,
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN}
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r.t. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN} ,
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN}
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN}
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN}
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Final Algorithm

Algorithm

1. Begin with the root node, that is, $X_t = X$.
2. For each new node t
3. For every feature x_k, $k = 1, 2, ..., l$:
4. For every value α_{kn}, $n = 1, 2, ..., N_{tk}$
5. Generate X_{tY} and X_{tN} according to the answer in the question:
6. "Is $x_k(i) \leq \alpha_{kn}$," $i = 1, 2, ..., N_t$
7. Compute the impurity decrease
8. Choose α_{kn_0} leading to the maximum decrease w.r. to x_k.
9. Choose x_{k_0} and associated $\alpha_{k_0 n_0}$ for overall maximum decrease of impurity.
10. If the stop-splitting rule is met, declare node t as a leaf and label a class
11. If not, generate two descendant nodes t_Y and t_N with subsets X_{tY} and X_{tN} ,
12. depending on the answer to the question: is $x_{k_0} \leq \alpha$?

Cinvestav

Outline

Cinvestav

Remarks

Popular Classification Methods

- Decision trees have emerged as one of the most popular methods of classification.

More Impurity Measures

- A variety of node impurity measures can be defined.

The size of the three need to be controlled

- The threshold T leads incorrect sizes.

Remarks

Popular Classification Methods

- Decision trees have emerged as one of the most popular methods of classification.

More Impurity Measures

- A variety of node impurity measures can be defined.

The size of the three need to be controlled

- The threshold T leads incorrect sizes.

Remarks

Popular Classification Methods

- Decision trees have emerged as one of the most popular methods of classification.

More Impurity Measures

- A variety of node impurity measures can be defined.

The size of the three need to be controlled

- The threshold T leads incorrect sizes.

Remarks

Why Binary Splits?

- We could consider a Multi-way split

However

- That will fragment the data too fast.

We would rather do only split when necessary

- After all a Multi-way split can be achieved with multiple binary split.

Remarks

Why Binary Splits?

- We could consider a Multi-way split

However

- That will fragment the data too fast.

We would rather do only split when necessary

- After all a Multi-way split can be achieved with multiple binary split.

Remarks

Why Binary Splits?

- We could consider a Multi-way split

However

- That will fragment the data too fast.

We would rather do only split when necessary

- After all a Multi-way split can be achieved with multiple binary split.

Remark

Linear Combination Splits

- Instead of doing simple splittings, we could use

$$\sum_{j=1}^{d} a_i x_i < s$$

This improve the predictive power of the tree

- It can hurts interpretability

Better use

- Hierarchical Mixture of Experts (HME).

Remark

Linear Combination Splits

- Instead of doing simple splittings, we could use

$$\sum_{j=1}^{d} a_i x_i < s$$

This improve the predictive power of the tree

- It can hurts interpretability

Better use

- Hierarchical Mixture of Experts (HME).

Remark

Linear Combination Splits

- Instead of doing simple splittings, we could use

$$\sum_{j=1}^{d} a_i x_i < s$$

This improve the predictive power of the tree

- It can hurts interpretability

Better use

- Hierarchical Mixture of Experts (HME).

Outline

Cinvestav

Issues

One of the biggest issues

- One major problem with trees is their high variance.

A small change in the data can result in a very different series of splits

- Making interpretability precarious!!!

Issues

One of the biggest issues

- One major problem with trees is their high variance.

A small change in the data can result in a very different series of splits

- Making interpretability precarious!!!

Issues

Lack of Smoothness

- Another limitation of trees is the lack of smoothness of the prediction surface

Thus strategies to alleviate this problem are necessary

- Multivariate Adaptive Regression Splines (MARS) procedure

Issues

Lack of Smoothness

- Another limitation of trees is the lack of smoothness of the prediction surface

Thus strategies to alleviate this problem are necessary

- Multivariate Adaptive Regression Splines (MARS) procedure

Issue

The CART trees are bad at modeling additive structures

- For Example

$$y = c_1 I\left(x_1 < t_1\right) + c_2 I\left(x_2 < t_2\right) + \epsilon \text{ with } \epsilon \sim N\left(0, \sigma^2\right)$$

Problem, CART has no special encouragement to capture this model

- Again MARS can help for this given its no dependency to the binary tree structure

Issue

The CART trees are bad at modeling additive structures

- For Example

$$y = c_1 I\left(x_1 < t_1\right) + c_2 I\left(x_2 < t_2\right) + \epsilon \text{ with } \epsilon \sim N\left(0, \sigma^2\right)$$

Problem, CART has no special encouragement to capture this model

- Again MARS can help for this given its no dependency to the binary tree structure