Interpretable Machine Learning

INSA-Toulouse & LAAS-CNRS

December 14, 2023

Motivation

Black-Box Machine Learning

`https://www.youtube.com/watch?v=6Kf3I_OyD1I&ab_channel=SouthChinaMorningPost`

The COMPAS Tool

MIT Technology Review

Featured Topics Newsletters Events Podcasts

Sign in Subscribe

TECH POLICY

AI is sending people to jail—and getting it wrong

Using historical data to train risk assessment tools could mean that machines are copying the mistakes of the past.

By Karen Hao

January 21, 2019

IAN WALDIE/GETTY IMAGES

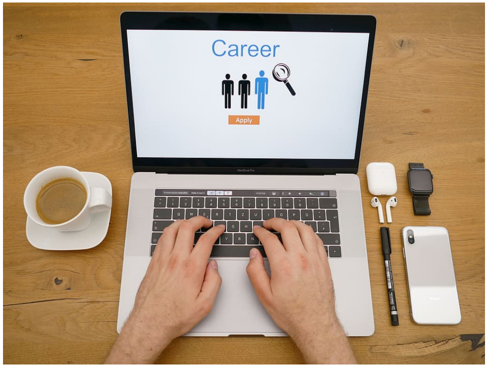
COMPASS data and Rule-based Predictions

Sex	Age	Priors	Juvenile Felonies	Juvenile Crimes	Ethnicity
Male	15	1	0	1	Caucasian
Male	15	1	0	1	African-American
Female	33	1	0	1	African-American
Female	27	0	1	0	Caucasian
Male	41	0	1	0	Caucasian
...	...	...	...	...	...

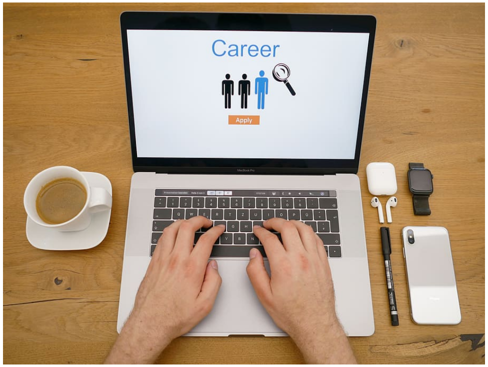
The problem is to predict recidivism. That is, the tendency of a convicted criminal to re-offend.

Increasing Number of Real Life and Social AI Applications

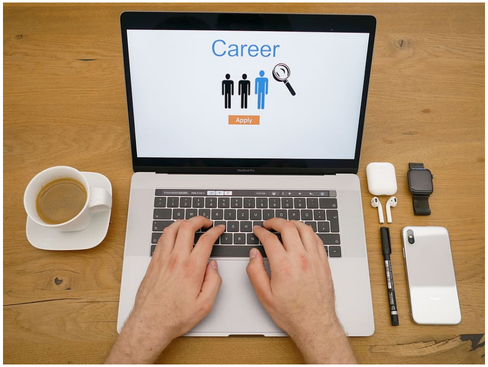
Increasing Number of Real Life and Social AI Applications



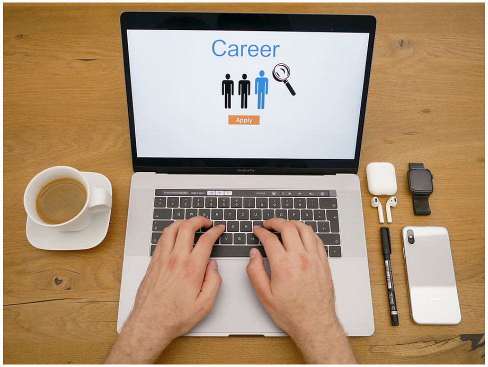
Increasing Number of Real Life and Social AI Applications



Increasing Number of Real Life and Social AI Applications



Increasing Number of Real Life and Social AI Applications



AI: Increasing Number of Real Life Applications Of Machine Learning

- The diverse applications of AI raised many ethical issues and questions

AI: Increasing Number of Real Life Applications Of Machine Learning

- The diverse applications of AI raised many ethical issues and questions
  - Job applications: AI that parses CVs for software engineers and recommends to hire mostly men

AI: Increasing Number of Real Life Applications Of Machine Learning

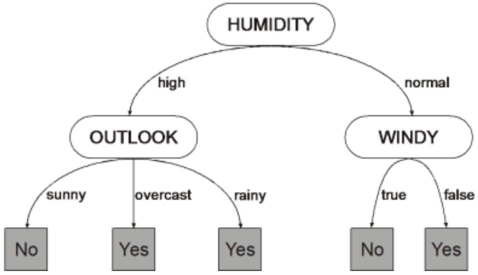
- The diverse applications of AI raised many ethical issues and questions
  - Job applications: AI that parses CVs for software engineers and recommends to hire mostly men
  - Credit scoring: AI that gives a credit score (for bank loans and credit applications) that recommends people from a particular geographical region, specific gender, social class, etc

AI: Increasing Number of Real Life Applications Of Machine Learning

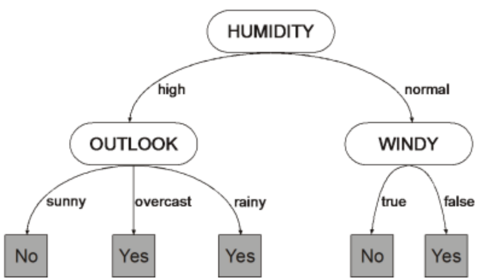
- The diverse applications of AI raised many ethical issues and questions
  - Job applications: AI that parses CVs for software engineers and recommends to hire mostly men
  - Credit scoring: AI that gives a credit score (for bank loans and credit applications) that recommends people from a particular geographical region, specific gender, social class, etc
  - Compass tool: (2016) used by judges in the US to predict which criminals are likely to re-offend is found to be biased by the ethnicity (African-American/Caucasian).

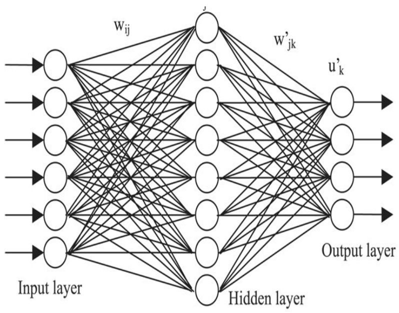
Black-Box vs Interpretable Models

Black-Box vs Interpretable Models



Black-Box vs Interpretable Models





Background

Supervised/Unsupervised/Reinforcement Learning

Supervised/Unsupervised/Reinforcement Learning

- Supervised Learning (Labelled data): Predict a function that associates inputs to outputs based on historical data

Supervised/Unsupervised/Reinforcement Learning

- Supervised Learning (Labelled data): Predict a function that associates inputs to outputs based on historical data
    - Categorical labels (discrete values): Classification

Supervised/Unsupervised/Reinforcement Learning

- Supervised Learning (Labelled data): Predict a function that associates inputs to outputs based on historical data
  - Categorical labels (discrete values): Classification
  - Non-categorical labels (real numbers): Regression

Supervised/Unsupervised/Reinforcement Learning

- Supervised Learning (Labelled data): Predict a function that associates inputs to outputs based on historical data
    - Categorical labels (discrete values): Classification
    - Non-categorical labels (real numbers): Regression
- Unsupervised Learning: The task is to figure out patterns presented in the data (unlabelled data)

Supervised/Unsupervised/Reinforcement Learning

- Supervised Learning (Labelled data): Predict a function that associates inputs to outputs based on historical data
  - Categorical labels (discrete values): Classification
  - Non-categorical labels (real numbers): Regression
- Unsupervised Learning: The task is to figure out patterns presented in the data (unlabelled data)
- Reinforcement learning: learning from a series of rewards /punishments
- But also, depending on the problem, data could be both labelled/non labelled, etc.. (semi-supervised learning)

[1] Image from https://en.wikipedia.org/wiki/Global_biodiversity

Figure 1: How to teach a child animal recognition? [1]

[1] Image from https://en.wikipedia.org/wiki/Global_biodiversity

Figure 1: How to teach a child animal recognition? [1]

Classification task

[1] Image from https://en.wikipedia.org/wiki/Global_biodiversity

[2] Image from https://en.wikipedia.org/wiki

Figure 2: How to predict a player's performance? [2]

[2] Image from `https://en.wikipedia.org/wiki`

Figure 2: How to predict a player's performance? [2]

Regression task

[2] Image from https://en.wikipedia.org/wiki

[3] Image from https://en.wikipedia.org/wiki/DNA

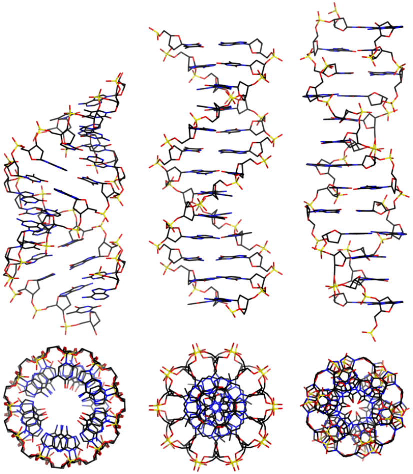


Figure 3: Analysis of evolutionary biology based on DNA patterns [3]

[3] Image from https://en.wikipedia.org/wiki/DNA

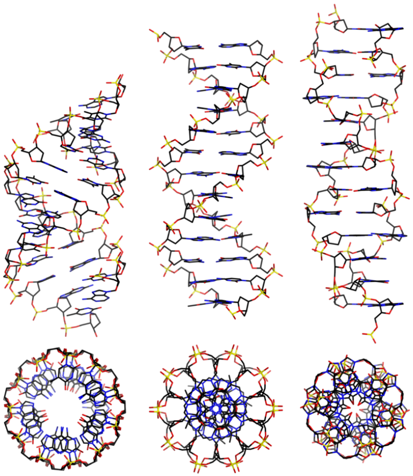


Figure 3: Analysis of evolutionary biology based on DNA patterns [3]

Unsupervised learning (clustering) task

[3] Image from `https://en.wikipedia.org/wiki/DNA`

[4] Image from https://en.wikipedia.org/wiki/Cycling

Figure 4: How to cycle? [4]

[4] Image from https://en.wikipedia.org/wiki/Cycling

Figure 4: How to cycle? [4]

Reinforcement learning

[4] Image from https://en.wikipedia.org/wiki/Cycling

Problem Definition

Problem Definition

- Input: data (**training set**) in the form of input-output examples: $\{(x_1, y_1), \ldots (x_n, y_n)\}$ where x_i is an input, y_i is the output of x_i drawn from an unknown distribution

Problem Definition

- Input: data (**training set**) in the form of input-output examples: $\{(x_1, y_1), \ldots (x_n, y_n)\}$ where x_i is an input, y_i is the output of x_i drawn from an unknown distribution
- Find a function f_h (called a hypothesis or model) that approximates the true data distribution

Problem Definition

- Input: data (**training set**) in the form of input-output examples: $\{(x_1, y_1), \ldots (x_n, y_n)\}$ where x_i is an input, y_i is the output of x_i drawn from an unknown distribution
- Find a function f_h (called a hypothesis or model) that approximates the true data distribution
- The approximation criterion can be defined in different ways. We can consider it as a function minimizing some error.

Training Phase

Training Phase

- The function f_h is constructed via a **training algorithm**

Training Phase

- The function f_h is constructed via a **training algorithm**
- The training algorithm depends on the hypothesis space

Training Phase

- The function f_h is constructed via a **training algorithm**
- The training algorithm depends on the hypothesis space
- Examples of hypothesis space (family of functions) include polynomial functions, trigonometry functions, decision trees, decision lists, neural networks, ...

Testing Phase

Testing Phase

- The evaluation of the constructed hypothesis (or model) is done via a set of unseen examples called **testing set**

Testing Phase

- The evaluation of the constructed hypothesis (or model) is done via a set of unseen examples called **testing set**
- The testing set is usually taken randomly from the initial data. However, it shouldn't be part of the training set

Testing Phase

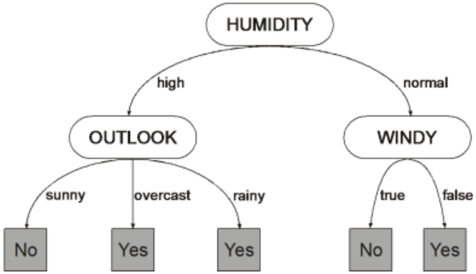
- The evaluation of the constructed hypothesis (or model) is done via a set of unseen examples called **testing set**
- The testing set is usually taken randomly from the initial data. However, it shouldn't be part of the training set
- The common way is to split the initial data into a training and testing sets randomly

Testing Phase

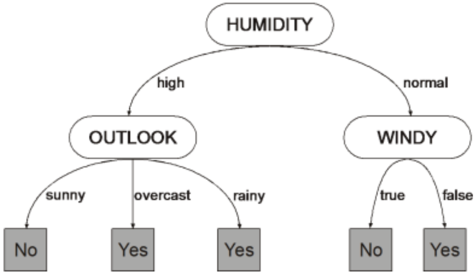
- The evaluation of the constructed hypothesis (or model) is done via a set of unseen examples called **testing set**
- The testing set is usually taken randomly from the initial data. However, it shouldn't be part of the training set
- The common way is to split the initial data into a training and testing sets randomly

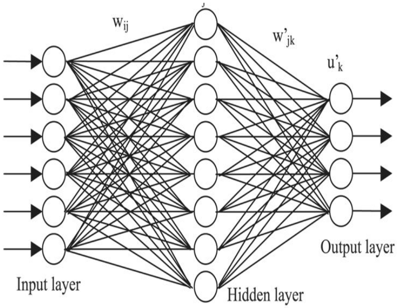
Black-Box vs Interpretable Models

Black-Box vs Interpretable Models



Black-Box vs Interpretable Models





Black-Box vs Interpretable Models: Definitions

- **Black-box model** [1]: A formula that is either too complicated for any human to understand, or proprietary, so that one cannot understand its inner workings
- **Interpretable model** [1] obeys a domain-specific set of constraints to allow it (or its predictions, or the data) to be more easily understood by humans. These constraints can differ dramatically depending on the domain.

Why Interpretable Models?

Why Interpretable Models?

- Transparent

Why Interpretable Models?

- Transparent
- Trustworthy

Why Interpretable Models?

- Transparent
- Trustworthy
- Inherently Explainable
- Well adapted for troubleshooting and diagnosis

Why Interpretable Models?

- Transparent
- Trustworthy
- Inherently Explainable
- Well adapted for troubleshooting and diagnosis
- **Mandatory criteria in high-stake decision making**

Interpretable Models

Case Study: Decision Trees

Case Study: Decision Trees

- Restrictions & notations: Tabular data with binary features and output

Case Study: Decision Trees

- Restrictions & notations: Tabular data with binary features and output
- Let $\mathbb{F} = \{f_1, \ldots f_k\}$ be a set of binary features (or attributes).

Case Study: Decision Trees

- Restrictions & notations: Tabular data with binary features and output
- Let $\mathbb{F} = \{f_1, \dots f_k\}$ be a set of binary features (or attributes).
- An example e_i is represented as $(x_1, \dots x_k, y_i)$ where x_i are the values associated to the different features and $y_i \in \{0, 1\}$ is the class of e_i

Case Study: Decision Trees

- Restrictions & notations: Tabular data with binary features and output
- Let $\mathbb{F} = \{f_1, \ldots f_k\}$ be a set of binary features (or attributes).
- An example e_i is represented as $(x_1, \ldots x_k, y_i)$ where x_i are the values associated to the different features and $y_i \in \{0, 1\}$ is the class of e_i
- Without loss of generality, we use the term 'positive' for the class 1 and 'negative' for the class 0
- The data is a collection of examples $\{e_1, \ldots, e_n\}$

Toy Example: The Likelihood of Animal Extinction

Toy Example: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

Toy Example: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

- $k = 4$ binary features, $n = 7$ examples

Toy Example: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

- $k = 4$ binary features, $n = 7$ examples
- Features $\mathbb{F} = \{$ Big size, Carnivore, Seasonal Reproduction, Solitary $\}$

Toy Example: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

- $k = 4$ binary features, $n = 7$ examples
- Features $\mathbb{F} = \{$ Big size, Carnivore, Seasonal Reproduction, Solitary $\}$
- Binary output: Extinct (the animal is extinct)

Toy Example: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

- $k = 4$ binary features, $n = 7$ examples
- Features $\mathbb{F} = \{$ Big size, Carnivore, Seasonal Reproduction, Solitary $\}$
- Binary output: Extinct (the animal is extinct)
- Example $e_6 = (0, 1, 1, 0, 1)$

Toy Example: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

- $k = 4$ binary features, $n = 7$ examples
- Features $\mathbb{F} = \{$ Big size, Carnivore, Seasonal Reproduction, Solitary $\}$
- Binary output: Extinct (the animal is extinct)
- Example $e_6 = (0, 1, 1, 0, 1)$
- Data= $\{e_1, e_2, \dots e_7\}$

Definition of Decision Tree (in the case of binary classification)

- A decision tree is a binary tree where each leaf node corresponds to a binary value (positive/negative class) and each internal node j is associated to a feature $feature(j) \in \mathbb{F}$
- Let DT be a decision tree. Denote by $feature(j)$ the feature associated to node j in DT. We name the children of an internal node j as right and left. We also use $(j, r(j))$ ($(j, l(j))$ respectively) to denote the arc from a node j to its right (respectively left) child .

Definition of Decision Tree (in the case of binary classification)

- A decision tree is a binary tree where each leaf node corresponds to a binary value (positive/negative class) and each internal node j is associated to a feature $feature(j) \in \mathbb{F}$
- Let DT be a decision tree. Denote by $feature(j)$ the feature associated to node j in DT. We name the children of an internal node j as right and left. We also use $(j, r(j))$ ($(j, l(j))$ respectively) to denote the arc from a node j to its right (respectively left) child .
- Classifying an example e_i by a decision DT is done by following the path $P(e_i)$ from the root to a leaf node where $(j, r(j)) \in P(e_i)$ if $x(feature(j)) = 1$, otherwise $(j, l(j)) \in P(e_i)$. The leaf node of $P(e_i)$ is the class of e_i decided by DT

Definition of Decision Tree (in the case of binary classification)

- A decision tree is a binary tree where each leaf node corresponds to a binary value (positive/negative class) and each internal node j is associated to a feature $feature(j) \in \mathbb{F}$
- Let DT be a decision tree. Denote by $feature(j)$ the feature associated to node j in DT. We name the children of an internal node j as right and left. We also use $(j, r(j))$ ($(j, l(j))$ respectively) to denote the arc from a node j to its right (respectively left) child .
- Classifying an example e_i by a decision DT is done by following the path $P(e_i)$ from the root to a leaf node where $(j, r(j)) \in P(e_i)$ if $x(feature(j)) = 1$, otherwise $(j, l(j)) \in P(e_i)$. The leaf node of $P(e_i)$ is the class of e_i decided by DT
- This definition can be extended to multiple classification and regression (by adapting the leaf values)

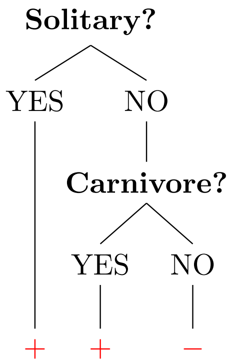
Toy Example: DTs to Predict The Likelihood of Animal Extinction

Toy Example: DTs to Predict The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

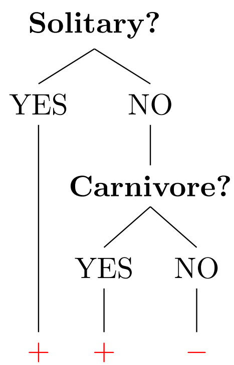
Toy Example: DTs to Predict The Likelihood of Animal Extinction

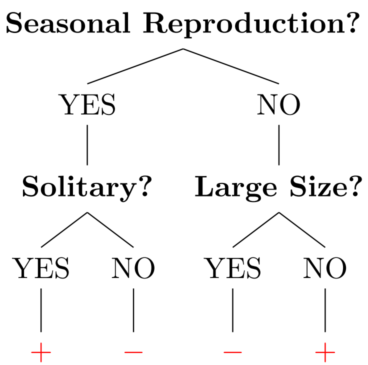
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes



Toy Example: DTs to Predict The Likelihood of Animal Extinction

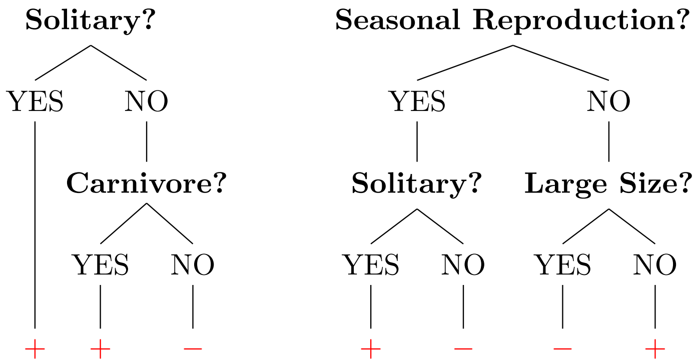
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes





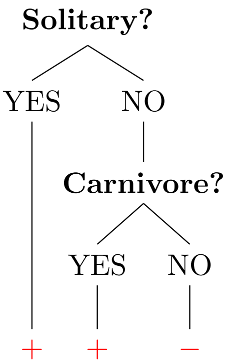
Toy Example: DTs to Predict The Likelihood of Animal Extinction

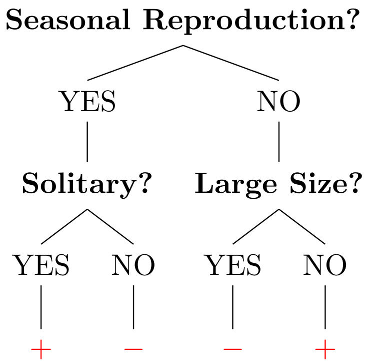
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes



Toy Example: DTs to Predict The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

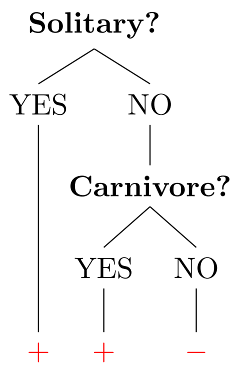


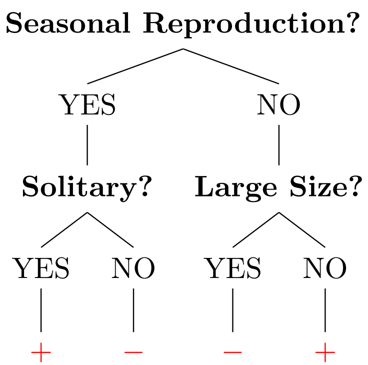


- Tabular data

Toy Example: DTs to Predict The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

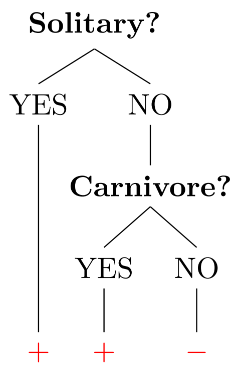


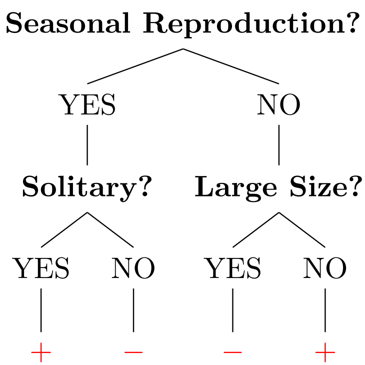


- Tabular data
- Hypothesis space: Decision trees

Toy Example: DTs to Predict The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

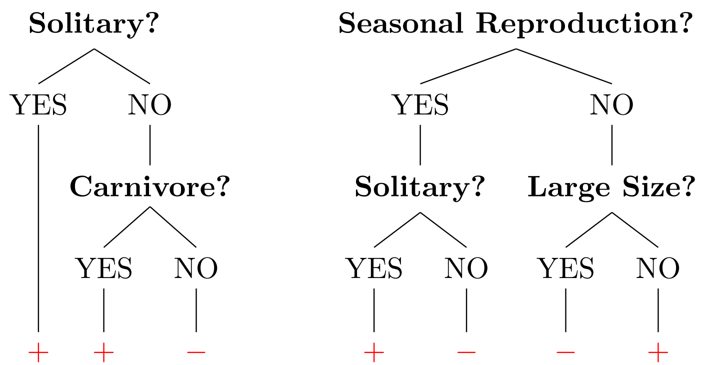




- Tabular data
- Hypothesis space: Decision trees
- Left tree: accuracy:

Toy Example: DTs to Predict The Likelihood of Animal Extinction

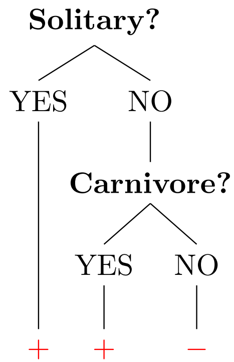
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

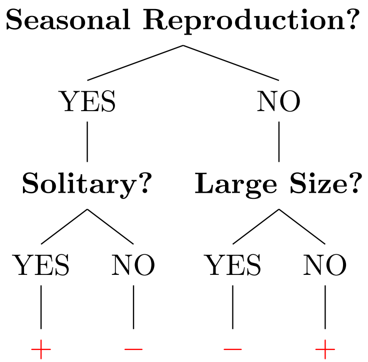


- Tabular data
- Hypothesis space: Decision trees
- Left tree: accuracy: $2/5 = 40\%$

Toy Example: DTs to Predict The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

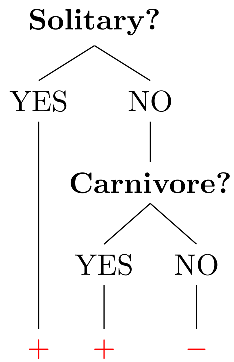


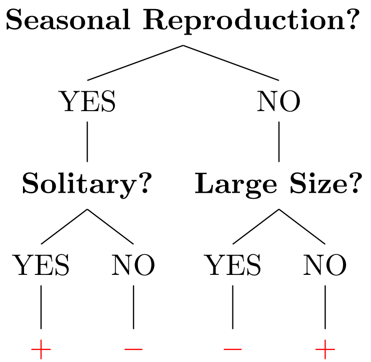


- Tabular data
- Hypothesis space: Decision trees
- Left tree: accuracy: $2/5 = 40\%$
- Right tree: accuracy:

Toy Example: DTs to Predict The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

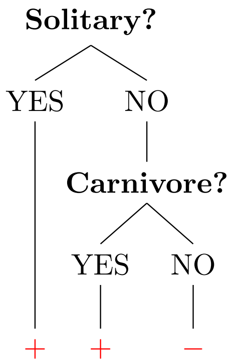


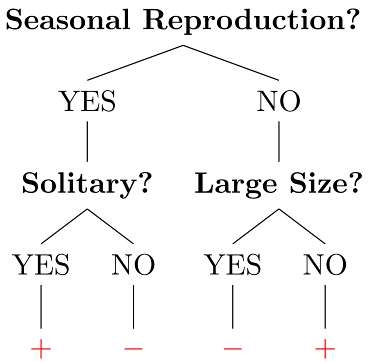


- Tabular data
- Hypothesis space: Decision trees
- Left tree: accuracy: $2/5 = 40\%$
- Right tree: accuracy: $2/5 = 40\%$

Toy Example: DTs to Predict The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes





- Tabular data
- Hypothesis space: Decision trees
- Left tree: accuracy: $2/5 = 40\%$
- Right tree: accuracy: $2/5 = 40\%$

Building a decision tree: The search space

- What is the search space with n examples and k features? That is, how many potential trees are there?

Building a decision tree: The search space

- What is the search space with n examples and k features? That is, how many potential trees are there?
- Since $n \leq 2^k$, a decision tree in this case is a partial Boolean function defined over k features

Building a decision tree: The search space

- What is the search space with n examples and k features? That is, how many potential trees are there?
- Since $n \leq 2^k$, a decision tree in this case is a partial Boolean function defined over k features
- We are looking for a partial Boolean function g over the set of possible partial Boolean functions $\mathbb{S}$ defined over k features that meet the criteria of the decision tree. In this case $\mathbb{S}$ is the search space
- The size of the search space is $|\mathbb{S}|$
- With k features, there are 2^k possible Boolean function (outputs of the associated truth table). This is because a truth table is determined by the binary string corresponding to the output and because there are 2^k possible strings
- Out of $z = 2^k$ Boolean function we are looking for a partial Boolean function that meet the requirements.

Building a Decision Tree: The Search Space

- Let g be a Boolean function and $Compatible(g)$ is a Boolean function that returns true if g is compatible with the requirements of the decision tree

Building a Decision Tree: The Search Space

- Let g be a Boolean function and $Compatible(g)$ is a Boolean function that returns true if g is compatible with the requirements of the decision tree
- $\mathbb{S}$ is the set containing all the possible $Compatible$ functions
- Let $z = 2^k$ be the size of the input space of $Compatible$. There exists 2^z possible instantiation of the $Compatible$ function because $Compatible(g) \in \{0, 1s\}$
- Therefore, $|\mathbb{S}| = 2^z = 2^{2^k}$

Building a Decision Tree: The Search Space

- Let g be a Boolean function and $Compatible(g)$ is a Boolean function that returns true if g is compatible with the requirements of the decision tree
- $\mathbb{S}$ is the set containing all the possible $Compatible$ functions
- Let $z = 2^k$ be the size of the input space of $Compatible$. There exists 2^z possible instantiation of the $Compatible$ function because $Compatible(g) \in \{0, 1s\}$
- Therefore, $|\mathbb{S}| = 2^z = 2^{2^k}$
- And since a partial Boolean function can be represented by several decision trees, then the search space for decision trees is bigger than 2^{2^k}

Building a Decision Tree: The Search Space

- Let g be a Boolean function and $Compatible(g)$ is a Boolean function that returns true if g is compatible with the requirements of the decision tree
- $\mathbb{S}$ is the set containing all the possible $Compatible$ functions
- Let $z = 2^k$ be the size of the input space of $Compatible$. There exists 2^z possible instantiation of the $Compatible$ function because $Compatible(g) \in \{0, 1s\}$
- Therefore, $|\mathbb{S}| = 2^z = 2^{2^k}$
- And since a partial Boolean function can be represented by several decision trees, then the search space for decision trees is bigger than 2^{2^k}
- This is a gigantic number!

Building a Decision Tree: The Search Space

- Let g be a Boolean function and $Compatible(g)$ is a Boolean function that returns true if g is compatible with the requirements of the decision tree
- $\mathbb{S}$ is the set containing all the possible $Compatible$ functions
- Let $z = 2^k$ be the size of the input space of $Compatible$. There exists 2^z possible instantiation of the $Compatible$ function because $Compatible(g) \in \{0, 1s\}$
- Therefore, $|\mathbb{S}| = 2^z = 2^{2^k}$
- And since a partial Boolean function can be represented by several decision trees, then the search space for decision trees is bigger than 2^{2^k}
- This is a gigantic number! For $k = 5$, $2^{2^k} = 4294967296$

Dealing with Intractability

Dealing with Intractability

Dealing with Intractability

- Enormous search space

Dealing with Intractability

- Enormous search space
- Building short trees under specific constraints is usually intractable

Dealing with Intractability

- Enormous search space
- Building short trees under specific constraints is usually intractable
- Exact algorithms hardly scale up

Dealing with Intractability

- Enormous search space
- Building short trees under specific constraints is usually intractable
- Exact algorithms hardly scale up
- Most of the approaches are greedy (heuristic) approaches

Dealing with Intractability

- Enormous search space
- Building short trees under specific constraints is usually intractable
- Exact algorithms hardly scale up
- Most of the approaches are greedy (heuristic) approaches
- Greedy algorithms follow a top-down approach: at each step, choose the best feature (to split the data) then recursively apply the same for the children until a certain stopping criterion

Building a decision Tree

- Decision trees can be represented as follows $(f, right, left)$ where f is a feature and $right$ (respectively $left$) are either decision trees or binary values (an outcome)
- We use the following oracles (functions):
  - $SelectBestFeature(data)$: select the best splitting feature according to some criterion
  - $UpdateInformation(Tree, Node)$: update information related to a given stopping requirement
  - $SelectClass(\mathbb{E})$: returns a class according to a selection criterion
  - $Explore(\mathbb{E}, info))$ a Boolean that indicates if the algorithm should develop more the tree
- The following is a high level greedy algorithm:

Building a Decision Tree: A Greedy Algorithm

Algorithm 1 GREEDY

Require: $\mathbb{F} = \{f_1, \ldots f_k\}$, $\mathbb{E} = \{e_1, \ldots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

Building a Decision Tree: A Greedy Algorithm

Algorithm 2 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

Building a Decision Tree: A Greedy Algorithm

Algorithm 3 GREEDY

Require: $\mathbb{F} = \{f_1, \ldots f_k\}$, $\mathbb{E} = \{e_1, \ldots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

Building a Decision Tree: A Greedy Algorithm

Algorithm 4 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

Building a Decision Tree: A Greedy Algorithm

Algorithm 5 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

Building a Decision Tree: A Greedy Algorithm

Algorithm 6 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

$LeftTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, L, f_j, LeftInformation)$;

Building a Decision Tree: A Greedy Algorithm

Algorithm 7 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

$LeftTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, L, f_j, LeftInformation)$;

$RightTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, R, f_j, RightInformation)$;

Building a Decision Tree: A Greedy Algorithm

Algorithm 8 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

$LeftTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, L, f_j, LeftInformation)$;

$RightTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, R, f_j, RightInformation)$;

return $(f_j, LeftTree, RightTree)$

else

return $SelectClass(\mathbb{E})$

end if;

Decision Trees for Regression? Yes!

Decision Trees for Regression? Yes!

Algorithm 10 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

Decision Trees for Regression? Yes!

Algorithm 11 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

Decision Trees for Regression? Yes!

Algorithm 12 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

Decision Trees for Regression? Yes!

Algorithm 13 GREEDY

Require: $\mathbb{F} = \{f_1, \ldots f_k\}$, $\mathbb{E} = \{e_1, \ldots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

Decision Trees for Regression? Yes!

Algorithm 14 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

Decision Trees for Regression? Yes!

Algorithm 15 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information *info* regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

$LeftTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, L, f_j, LeftInformation)$;

Decision Trees for Regression? Yes!

Algorithm 16 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

$LeftTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, L, f_j, LeftInformation)$;

$RightTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, R, f_j, RightInformation)$;

Decision Trees for Regression? Yes!

Algorithm 17 GREEDY

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

if $Explore(\mathbb{E}, info))$ **then**

$f_j \leftarrow SelectBestFeature(data)$

$L \leftarrow \{x \in E | f_j = 0\}$; $R \leftarrow \{x \in E | f_j = 1\}$;

$LeftInfo \leftarrow UpdateInformation(L, parent)$;

$RightInfo \leftarrow UpdateInformation(R, parent)$;

$LeftTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, L, f_j, LeftInformation)$;

$RightTree \leftarrow GREEDY(\mathbb{F} \setminus f_j, R, f_j, RightInformation)$;

return $(f_j, LeftTree, RightTree)$

else

return $SelecValue(\mathbb{E})$

end if;

Information Gain

- There are several ways to choose a 'good' feature
- The Information Gain is one of the most used criterion
- It uses the notion of Entropy that evaluates data uncertainty (initially proposed in the context of information theory by Shanon and Weaver)

Entropy

Entropy

- Entropy is a statistical measure (proposed by Clause Shanon and Weaver) as the number of bits needed to represent uncertainty

Entropy

- Entropy is a statistical measure (proposed by Clause Shanon and Weaver) as the number of bits needed to represent uncertainty
- Imagine you toss a normal coin. Both heads and tails have a 50% chance to occur

Entropy

- Entropy is a statistical measure (proposed by Clause Shanon and Weaver) as the number of bits needed to represent uncertainty
- Imagine you toss a normal coin. Both heads and tails have a 50% chance to occur
- Guessing the outcome of the toss is highly uncertain because of the equal chances

Entropy

- Entropy is a statistical measure (proposed by Clause Shanon and Weaver) as the number of bits needed to represent uncertainty
- Imagine you toss a normal coin. Both heads and tails have a 50% chance to occur
- Guessing the outcome of the toss is highly uncertain because of the equal chances
- In this case, $Entropy = 1$

Entropy

- Entropy is a statistical measure (proposed by Clause Shanon and Weaver) as the number of bits needed to represent uncertainty
- Imagine you toss a normal coin. Both heads and tails have a 50% chance to occur
- Guessing the outcome of the toss is highly uncertain because of the equal chances
- In this case, $Entropy = 1$
- In the other extreme, a coin with heads on both sides has no uncertainty because of the constant outcome (always heads)
- In this case, $Entropy = 0$

Entropy

Let Y be a discrete random variable taking values y_j, the entropy of Y is defined as follows:

$$H(Y) = \sum_j P(y_j) \times log_2(1/P(y_j))$$

where $P(y_j)$ is the probability of the value y_j
Example: For a fair coin:

$$H(Y) = 0.5 \times log_2(2) + 0.5 \times log_2(2) = 1$$

For a coin with 90% with heads chance:

$$H(V) = 0.9 \times log_2(10/9) + 0.1 \times log_2(10) = 0.46$$

Entropy in the case of binary decision trees

Entropy in the case of binary decision trees

- Back to binary classification with a set E of n examples containing a positive examples and b negative examples. Consider the classification outcome as a random variable. We denote the entropy of this Boolean random variable as $H(data)$
- For a feature f_j, we define $n_1 = |E_1|$ where $E_1 = E \setminus \{x|x_j = 1\}$ and $n_0 = |E_0|$ where $E_0 = E \setminus \{x|x_j = 0\}$. We also denote by a_1 (respectively a_0) the number of positive examples in E_1 (respectively E_0) and by b_1 (respectively b_0) the number of negative examples in E_1 (respectively E_0)

Entropy in the case of binary decision trees

The expected entropy after splitting the data with the f_j is

$$Remaining(f_j) = n_1/n \times H(E_1) + n_0/n \times H(E_0)$$

We are looking for a feature that has a low level of uncertainty when splitting the data. A good splitter f_j is a feature with a minimum value of $Remaining(f_j)$ (this measures how much uncertainty is removed from the data).
This is equivalent to maximizing the *information gain (IG)*:

$$IG(f_j) = 1 - Remaining(f_j)$$

Back to Greedy Algorithms

Back to Greedy Algorithms

- Different algorithms use different criteria

Back to Greedy Algorithms

- Different algorithms use different criteria
- For instance, Information gain is used in C4.5 [2] and Gini Index is used in in CART (Classification and regression trees [3])

Back to Greedy Algorithms

- Different algorithms use different criteria
- For instance, Information gain is used in C4.5 [2] and Gini Index is used in in CART (Classification and regression trees [3])
- Information gain reflects uncertainty and the purpose is to favor the feature that minimize uncertainty

Back to Greedy Algorithms

- Different algorithms use different criteria
- For instance, Information gain is used in C4.5 [2] and Gini Index is used in in CART (Classification and regression trees [3])
- Information gain reflects uncertainty and the purpose is to favor the feature that minimize uncertainty
- *Gini Index (GI)* reflects the purity of the data.

Back to Greedy Algorithms

- Different algorithms use different criteria
- For instance, Information gain is used in C4.5 [2] and Gini Index is used in in CART (Classification and regression trees [3])
- Information gain reflects uncertainty and the purpose is to favor the feature that minimize uncertainty
- *Gini Index (GI)* reflects the purity of the data. The values range from 0 to 1 where 0 represents a pure data (with one class), 1 represents a random distribution, and 0.5 represents a completely equal distribution.

Back to Greedy Algorithms

- Different algorithms use different criteria
- For instance, Information gain is used in C4.5 [2] and Gini Index is used in in CART (Classification and regression trees [3])
- Information gain reflects uncertainty and the purpose is to favor the feature that minimize uncertainty
- *Gini Index (GI)* reflects the purity of the data. The values range from 0 to 1 where 0 represents a pure data (with one class), 1 represents a random distribution, and 0.5 represents a completely equal distribution. The chosen split is to the one that minimzes the GI

Back to Greedy Algorithms

- Different algorithms use different criteria
- For instance, Information gain is used in C4.5 [2] and Gini Index is used in in CART (Classification and regression trees [3])
- Information gain reflects uncertainty and the purpose is to favor the feature that minimize uncertainty
- *Gini Index (GI)* reflects the purity of the data. The values range from 0 to 1 where 0 represents a pure data (with one class), 1 represents a random distribution, and 0.5 represents a completely equal distribution. The chosen split is to the one that minimzes the GI
- Both algorithms are efficient in practice, however without guarantee of optimality
- A trend is observed recently to build optimal DTs (for instance [4, 5])

Exercise: The Likelihood of Animal Extinction

Exercise: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

Exercise: The Likelihood of Animal Extinction

Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes
0	1	1	0	yes
1	1	1	0	no

- What is the value of information gain of the original dataset ?
- Which feature is better according to the information gain value (between Big Size and Solitary)?
- Build a decision tree with the previous approach where the the height is at most 3, and the classification follows a majority rule

Pruning as a post processing

Pruning as a post processing

- Any training algorithm might build dense and long trees. This might induce overfitting

Pruning as a post processing

- Any training algorithm might build dense and long trees. This might induce overfitting
- A simple way to overcome this issue is to ‘trim’ the tree as a post-processing step by removing useless branches or nodes (the ones causing overfitting)

Pruning as a post processing

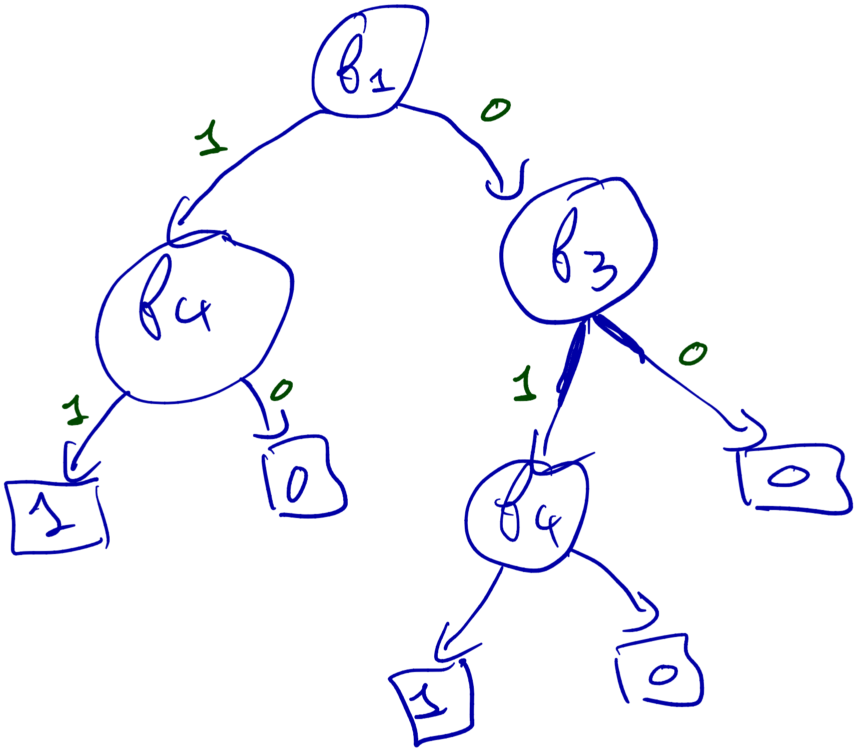
- Any training algorithm might build dense and long trees. This might induce overfitting
- A simple way to overcome this issue is to 'trim' the tree as a post-processing step by removing useless branches or nodes (the ones causing overfitting)
- Useless branches are typically long and are used to classify a limited number of examples

Pruning as a post processing

- Any training algorithm might build dense and long trees. This might induce overfitting
- A simple way to overcome this issue is to 'trim' the tree as a post-processing step by removing useless branches or nodes (the ones causing overfitting)
- Useless branches are typically long and are used to classify a limited number of examples
- The post processing might include other operations such as removing redundant sub-trees and useless splits

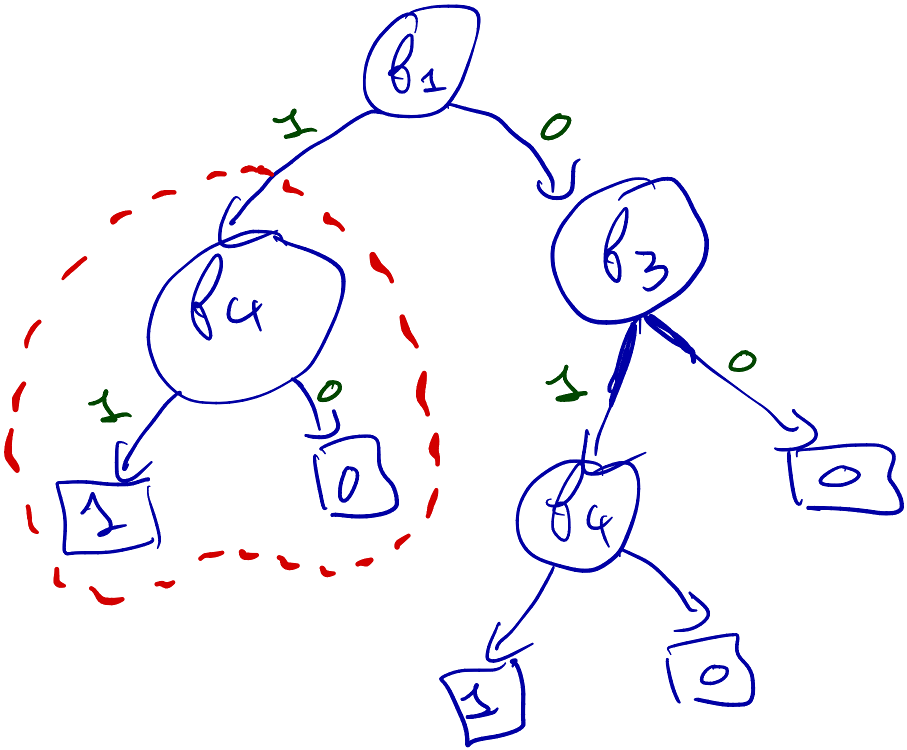
Something is weird, can you find it ?

Something is weird, can you find it ?



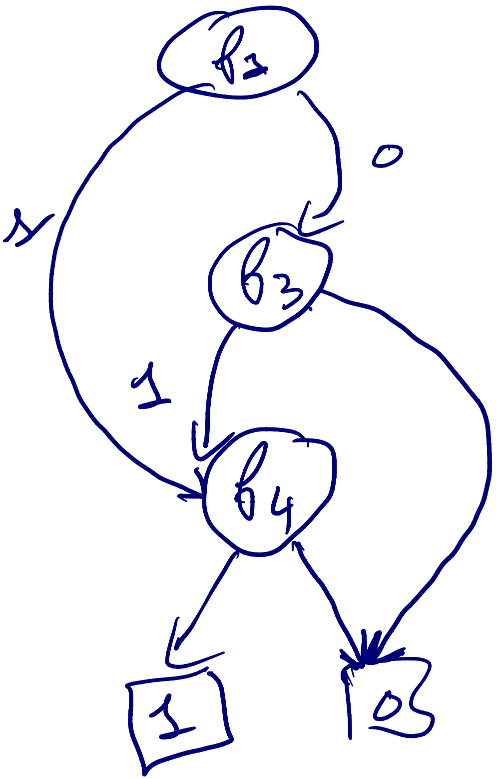
Something is weird, can you guess it ?

Something is weird, can you guess it ?



Binary Decision Diagram as an Alternative Model

Binary Decision Diagram as an Alternative Model



Ensemble Learning, Random Forest, and Boosted Trees

- Ensemble Learning is a learning methodology that relies on training a number of prediction models. The idea is to improve a constructed model by using instead a set of models

Ensemble Learning, Random Forest, and Boosted Trees

- Ensemble Learning is a learning methodology that relies on training a number of prediction models. The idea is to improve a constructed model by using instead a set of models
- There are two types of ensemble learning: Boosting and Bagging

Ensemble Learning, Random Forest, and Boosted Trees

- Ensemble Learning is a learning methodology that relies on training a number of prediction models. The idea is to improve a constructed model by using instead a set of models
- There are two types of ensemble learning: Boosting and Bagging
- Bagging is a technique that learns several models by randomly selecting a subset of the data for each model. The predictions are made based on majority vote (in case of binary classification). In the case of bagging with decision trees, the model is called random forest.

Ensemble Learning, Random Forest, and Boosted Trees

- Ensemble Learning is a learning methodology that relies on training a number of prediction models. The idea is to improve a constructed model by using instead a set of models
- There are two types of ensemble learning: Boosting and Bagging
- Bagging is a technique that learns several models by randomly selecting a subset of the data for each model. The predictions are made based on majority vote (in case of binary classification). In the case of bagging with decision trees, the model is called random forest.
- Boosting is a technique that learns several models in a sequence where each model relies on the mistakes of the previous ones to improve the quality of the learning. Usually, when boosting a model, each example is weighted by how many times it is badly classified in order to give it an advantage.

Decision Rules & Decision Sets

Decision Rules & Decision Sets

- They are defined as If-Condition-Then-Prediction rules

Decision Rules & Decision Sets

- They are defined as If-Condition-Then-Prediction rules
- **Decision sets**: no specific order is given between the rules. Ties are broken by majority votes

Decision Rules & Decision Sets

- They are defined as If-Condition-Then-Prediction rules
- **Decision sets**: no specific order is given between the rules. Ties are broken by majority votes
- **Decision rules**: rules are ordered by priority

The COMPASS Example

Sex	Age	Priors	Juvenile Felonies	Juvenile Crimes	Ethnicity
Male	15	1	0	1	Caucasian
Male	15	1	0	1	African-American
Female	33	1	0	1	African-American
Female	27	0	1	0	Caucasian
Male	41	0	1	0	Caucasian
...	...	...	...	...	...

The problem is to predict recidivism. That is, the tendency of a convicted criminal to re-offend.

The COMPASS Example

Sex	Age	Priors	Juvenile Felonies	Juvenile Crimes	Ethnicity
Male	15	1	0	1	Caucasian
Male	15	1	0	1	African-American
Female	33	1	0	1	African-American
Female	27	0	1	0	Caucasian
Male	41	0	1	0	Caucasian
...	...	...	...	...	...

The problem is to predict recidivism. That is, the tendency of a convicted criminal to re-offend.

- Data : `https://www.kaggle.com/danofer/compass`
- FairCORELS: An open source tool to learn fair rule lists `https://github.com/ferryjul/fairCORELS`

Example

Rule List found by FairCORELS on the COMPASS Sataset

Example

Rule List found by FairCORELS on the COMPASS Sataset

```
if [priors:>3] then [recidivism]
else if [age:21-22 && gender:Male] then [recidivism]
else if [age:18-20] then [recidivism]
else if [age:23-25 && priors:2-3] then [recidivism]
else [no recidivism]
```

Rule list 5. Example of an unconstrained rule list found by `FairCORELS` on COMPAS dataset, with Accuracy $= 0.681$, $\text{UNF}_{\text{EOdds}} = 0.217$ and $\text{UNF}_{\text{CUAE}} = 0.046$

Explanations

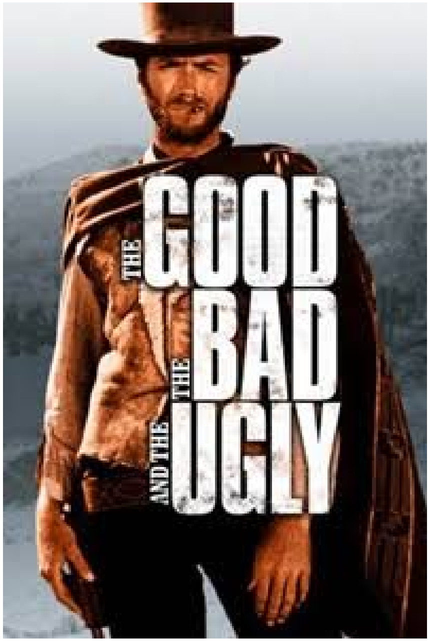
```
if [priors:>3] then [recidivism]
else if [age:21-22 && gender:Male] then [recidivism]
else if [age:18-20] then [recidivism]
else if [age:23-25 && priors:2-3] then [recidivism]
else [no recidivism]
```

Rule list 5. Example of an unconstrained rule list found by FairCORELS on COMPAS dataset, with Accuracy $= 0.681$, $\mathsf{UNF}_{\mathsf{EOdds}} = 0.217$ and $\mathsf{UNF}_{\mathsf{CUAE}} = 0.046$

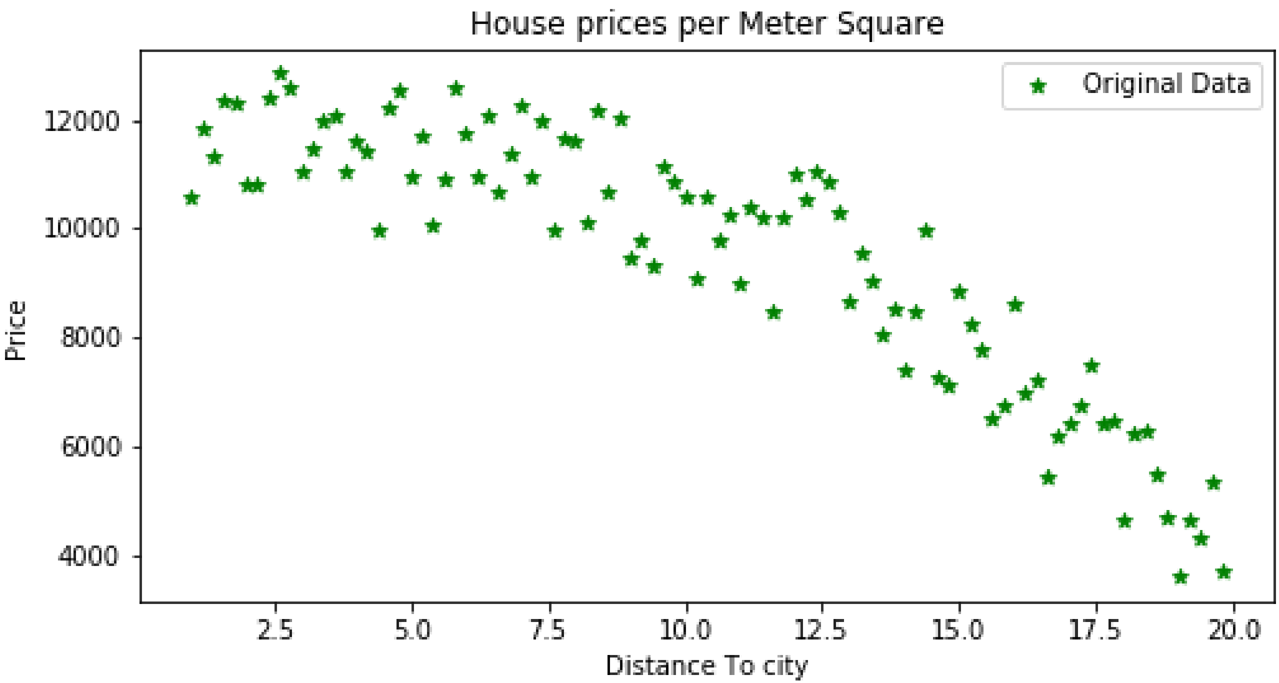
Sex	Age	Priors	Juvenile Felonies	Juvenile Crimes	Ethnicity
Male	21	1	0	1	Caucasian
Female	33	1	0	1	African-American
Female	27	0	1	0	Caucasian

- Example 1 is predicted positively. **Explanation** : $priors < 3$ and $age = 21$ and $gender = male$
- Example 2 is predicted negatively. **Explanation** :$priors = 1$ and $gender = female$ and $age : 33$

Overfitting, Underfitting, and Goodfitting

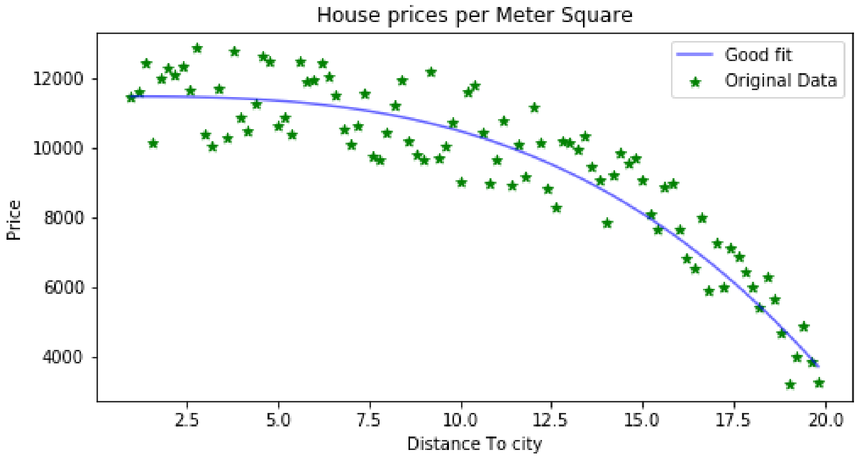


The Housing Prices Example



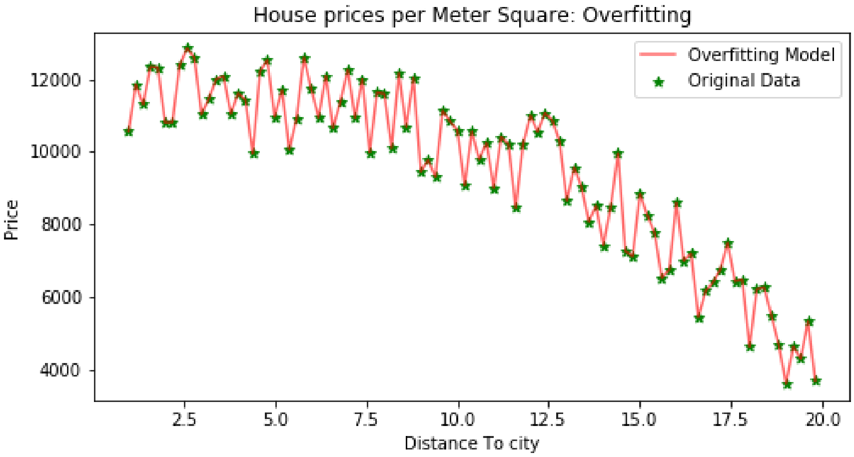
This data includes some **noise**. That is, points that are not correctly collected (which is often the case in real applications)

The Housing Prices Example: The Good

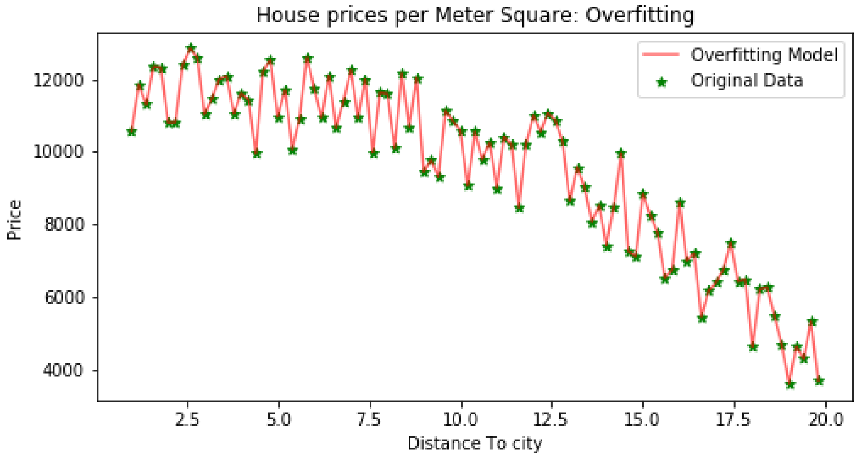


We can make an analogy to a smart student who has a good understanding of a lecture

The Housing Prices Example: The Bad (Overfitting)

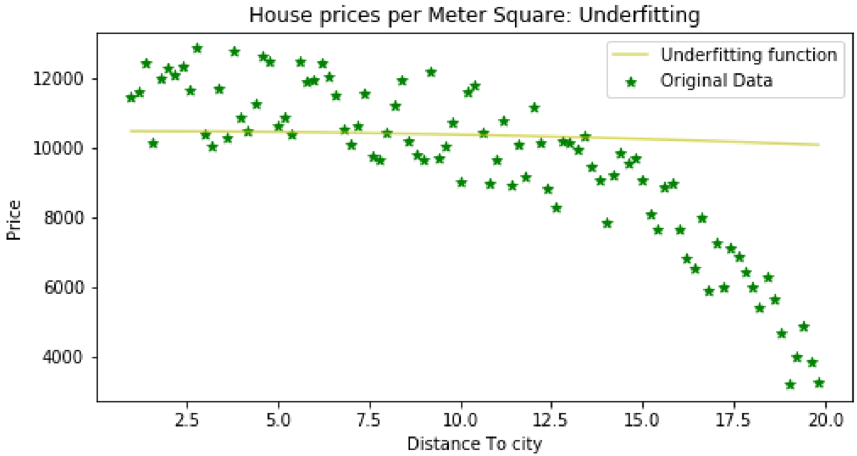


The Housing Prices Example: The Bad (Overfitting)

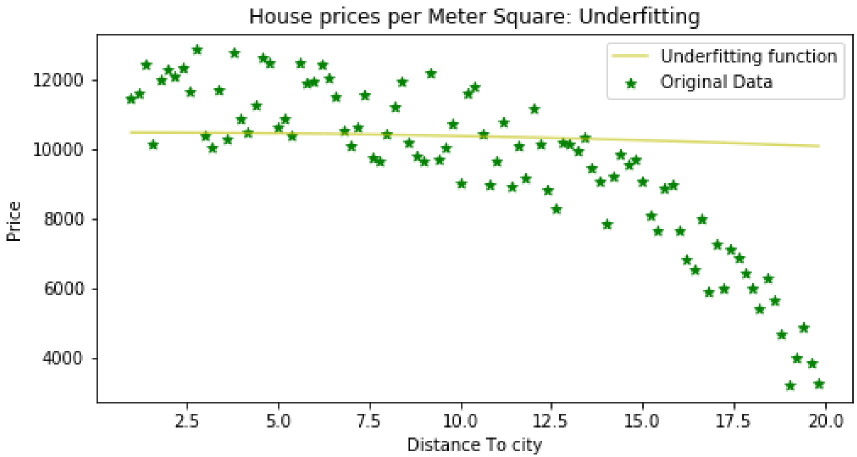


We can make an analogy to the student who "learns" the lecture mechanically without a real understanding.

The Housing Prices Example: The Ugly (Underfitting)

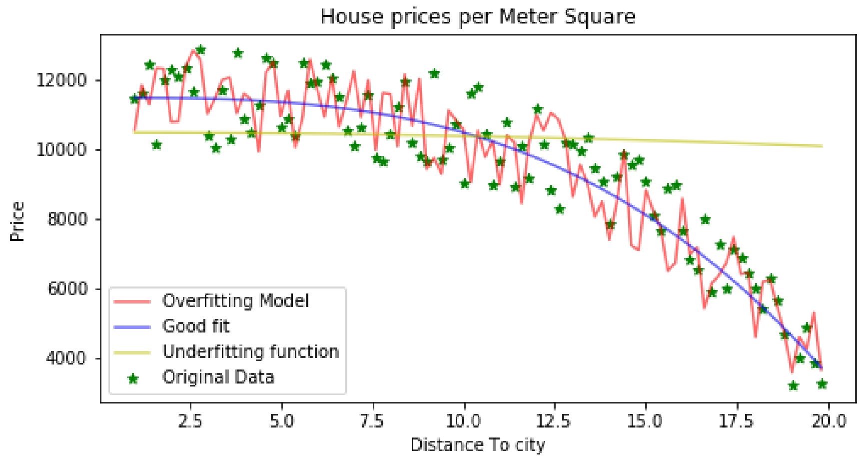


The Housing Prices Example: The Ugly (Underfitting)



We can make an analogy to a lazy student who barely remember the lecture without any understanding

The Housing Prices Example: All Together



Overfitting, Underfitting, and a Good Fit

Overfitting, Underfitting, and a Good Fit

- Overfitting happens when the model tries to squeeze everything in including noise without an "intuitive understanding of the data"

Overfitting, Underfitting, and a Good Fit

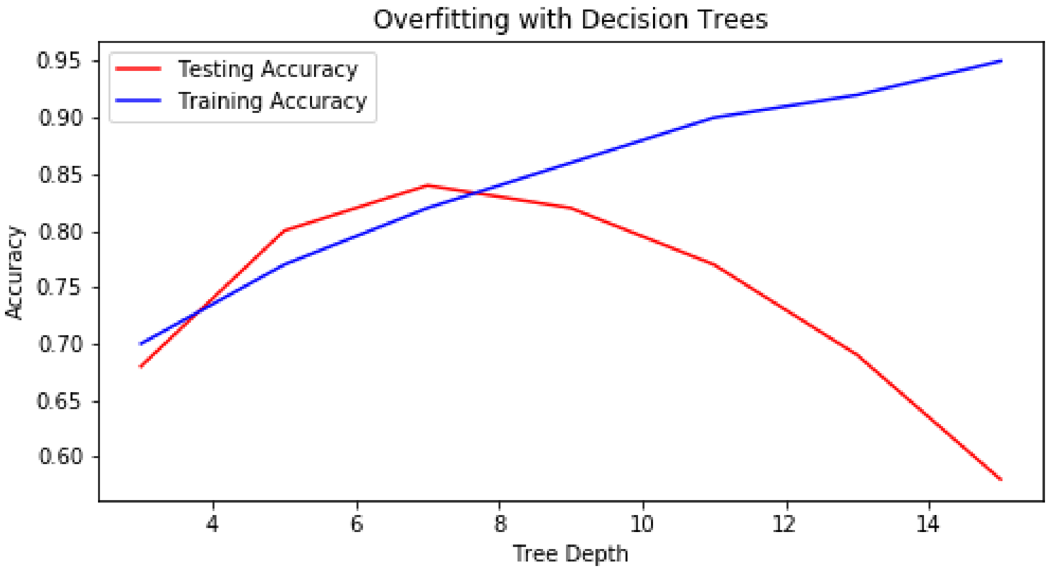
- Overfitting happens when the model tries to squeeze everything in including noise without an "intuitive understanding of the data"
- Underfitting happens when the model performs badly on the training and testing data (no real learning).

Overfitting, Underfitting, and a Good Fit

- Overfitting happens when the model tries to squeeze everything in including noise without an "intuitive understanding of the data"
- Underfitting happens when the model performs badly on the training and testing data (no real learning).
- A good fit happens when the model approximates well the true distribution without being disturbed by noise (good generalisation)

Overfitting with Decision Trees as an Example

Overfitting with Decision Trees as an Example



Overfitting with Decision Trees as an Example

- The longer the tree, the better the training accuracy gets, however, this is not necessarily the case for the testing accuracy

Overfitting with Decision Trees as an Example

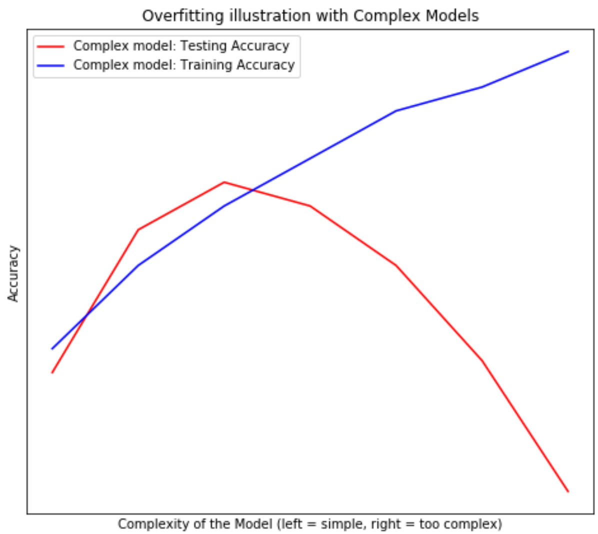
- The longer the tree, the better the training accuracy gets, however, this is not necessarily the case for the testing accuracy
- Testing accuracy increases at the beginning until a certain value (depth $= 7$), the it decreases afterwards

Overfitting with Decision Trees as an Example

- The longer the tree, the better the training accuracy gets, however, this is not necessarily the case for the testing accuracy
- Testing accuracy increases at the beginning until a certain value (depth $= 7$), the it decreases afterwards
- This happens because with longer trees, the model can classify correctly more examples in the training set, however, this includes noise.

Overfitting Based on the Complexity of the Model

Overfitting Based on the Complexity of the Model



- When the model is too simple, there is a risque of underfitting
- When the model is too complex, there is a risque of overfitting
- We need a Model that is somehow in between
- ML libraries offer parameters for regulation to avoid overfitting/underfitting

Complexity/Quality Tradeoff

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime
- For instance, training quadratic functions is much harder (computationally) than training linear functions

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime
- For instance, training quadratic functions is much harder (computationally) than training linear functions
- However, may be a quadratic function is a better fit for the data at hand

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime
- For instance, training quadratic functions is much harder (computationally) than training linear functions
- However, may be a quadratic function is a better fit for the data at hand
- There is a trade-off between the quality of predictions and the model complexity

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime
- For instance, training quadratic functions is much harder (computationally) than training linear functions
- However, may be a quadratic function is a better fit for the data at hand
- There is a trade-off between the quality of predictions and the model complexity
- For example training a tree with depth 5 is much faster than training a tree of depth 9, but in terms of training quality, trees of depth 9 are better. However, trees with depth 9 might overfit

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime
- For instance, training quadratic functions is much harder (computationally) than training linear functions
- However, may be a quadratic function is a better fit for the data at hand
- There is a trade-off between the quality of predictions and the model complexity
- For example training a tree with depth 5 is much faster than training a tree of depth 9, but in terms of training quality, trees of depth 9 are better. However, trees with depth 9 might overfit
- Most ML libraries offer the possibility to control the complexity with a regularization parameter

Ockham's Razor Principle

[5] Philosopher `https://en.wikipedia.org/wiki/William_of_Ockham`

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy

[5] Philosopher `https://en.wikipedia.org/wiki/William_of_Ockham`

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy
- Which tree to choose?

[5] Philosopher `https://en.wikipedia.org/wiki/William_of_Ockham`

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements

[5] Philosopher https://en.wikipedia.org/wiki/William_of_Ockham

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[5]: pick the simplest!

[5] Philosopher https://en.wikipedia.org/wiki/William_of_Ockham

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[5]: pick the simplest!
- Simplicity is also hard to define

[5] Philosopher `https://en.wikipedia.org/wiki/William_of_Ockham`

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[5]: pick the simplest!
- Simplicity is also hard to define
- In decision trees, simplicity could be the depth, the number of features, a combination of both, etc

[5] Philosopher `https://en.wikipedia.org/wiki/William_of_Ockham`

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[5]: pick the simplest!
- Simplicity is also hard to define
- In decision trees, simplicity could be the depth, the number of features, a combination of both, etc
- When using polynomials (as a hypothesis space), lower degrees seem to be simpler

[5] Philosopher `https://en.wikipedia.org/wiki/William_of_Ockham`

Ockham's Razor Principle

- In the animal extinction example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[5]: pick the simplest!
- Simplicity is also hard to define
- In decision trees, simplicity could be the depth, the number of features, a combination of both, etc
- When using polynomials (as a hypothesis space), lower degrees seem to be simpler
- In other cases it is very hard to define simplicity

[5] Philosopher `https://en.wikipedia.org/wiki/William_of_Ockham`

Complexity/Quality/Overfitting Tradeoff

The bottom line

Complexity/Quality/Overfitting Tradeoff

The bottom line

- There are fine lines between:

Complexity/Quality/Overfitting Tradeoff

The bottom line

- There are fine lines between:
  - overfitting/underfitting

Complexity/Quality/Overfitting Tradeoff

The bottom line

- There are fine lines between:
  - overfitting/underfitting
  - hard/easy training algorithms

Complexity/Quality/Overfitting Tradeoff

The bottom line

- There are fine lines between:
  - overfitting/underfitting
  - hard/easy training algorithms
  - complex/simple models
- Complex models can be computationally hard, however have better flexibility (some parameters can be turned off) and might have better quality

Complexity/Quality/Overfitting Tradeoff

The bottom line

- There are fine lines between:
  - overfitting/underfitting
  - hard/easy training algorithms
  - complex/simple models
- Complex models can be computationally hard, however have better flexibility (some parameters can be turned off) and might have better quality
- Complex models might overfit

Complexity/Quality/Overfitting Tradeoff

The bottom line

- There are fine lines between:
  - overfitting/underfitting
  - hard/easy training algorithms
  - complex/simple models
- Complex models can be computationally hard, however have better flexibility (some parameters can be turned off) and might have better quality
- Complex models might overfit
- Simple models might underfit

Complexity/Quality/Overfitting Tradeoff

The bottom line

- There are fine lines between:
  - overfitting/underfitting
  - hard/easy training algorithms
  - complex/simple models
- Complex models can be computationally hard, however have better flexibility (some parameters can be turned off) and might have better quality
- Complex models might overfit
- Simple models might underfit
- Ideally, we look for a hypothesis that is 'easy' to compute and simple enough to be a good fit

Interpretability vs. Explanability

The Debate & The 1 Million Dollars Reward

nature machine intelligence

Explore content ⌄ About the journal ⌄ Publish with us ⌄ Subscribe

nature > nature machine intelligence > perspectives > article

Perspective | Published: 13 May 2019

Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead

Cynthia Rudin

Nature Machine Intelligence **1**, 206–215 (2019) | Cite this article

49k Accesses | **1020** Citations | **402** Altmetric | Metrics

A preprint version of the article is available at arXiv.

`https://www.youtube.com/watch?v=4oXFEDoEcAk`

Back to Interpretable Models: Examples

Back to Interpretable Models: Examples

- Decision trees, Linear Models, but also:

Back to Interpretable Models: Examples

- Decision trees, Linear Models, but also:
- Rule lists

Back to Interpretable Models: Examples

- Decision trees, Linear Models, but also:
- Rule lists
- Decision list

Back to Interpretable Models: Examples

- Decision trees, Linear Models, but also:
- Rule lists
- Decision list
- Binary Decision Diagram (very useful to handle redundancy with decision trees)
- ...

Back to Interpretable Models: Rule lists

Back to Interpretable Models: Rule lists

- Rule lists: an ordered list of if-then-else rules with a default prediction.

Back to Interpretable Models: Rule lists

- Rule lists: an ordered list of if-then-else rules with a default prediction.
  1. **If 'Carnivore' then Extinct**
  2. **Else If 'Solitary' and not 'Big Size' then Not Extinct**
  3. **Else Extinct**

Back to Interpretable Models: Decision Lists

Back to Interpretable Models: Decision Lists

- Decision Lists: a set of if-then-else rules without any specific order. The prediction is made following a majority rule or a random choice if needed (e.g., if an example satisfies two different rules).
- For example: **{If 'Carnivore' then Extinct; If 'Solitary' and not 'Big Size' then Not Extinct ; If 'Seasonal Reproduction' then Extinct }**
- Consider an example that is 'Carnivore', follows a 'Seasonal Reproduction', 'Solitary', and does not have a 'Big Size'. Two rules classify the example positively (Extinct) and one rule classifies it negatively (Not Extinct). In this case the majority vote is used and the prediction is positive (Extinct).

Back to Interpretable Models: Why Interpretable Models?

Back to Interpretable Models: Why Interpretable Models?

- Transparent

Back to Interpretable Models: Why Interpretable Models?

- Transparent
- Coherent with trustworthy AI (See for instance 'GDPR' (the European General Data Protection Regulation)

Back to Interpretable Models: Why Interpretable Models?

- Transparent
- Coherent with trustworthy AI (See for instance 'GDPR' (the European General Data Protection Regulation)
- Inherently Explainable
- Well adapted for troubleshooting and diagnosis

Back to Interpretable Models: Why Interpretable Models?

- Transparent
- Coherent with trustworthy AI (See for instance ‘GDPR’ (the European General Data Protection Regulation)
- Inherently Explainable
- Well adapted for troubleshooting and diagnosis
- **Mandatory criteria in high-stake decision making**

Explainability

- Very complex (and philosophical) notion (see for instance the interview with Richard Feynman on the 'Why' question `https://www.youtube.com/watch?v=36GT2zI8lVA`)

Explainability

- Very complex (and philosophical) notion (see for instance the interview with Richard Feynman on the 'Why' question `https://www.youtube.com/watch?v=36GT2zI8lVA`)
- To explain predictions one needs a clear context to define explanations (user defined)

Explainability

- Very complex (and philosophical) notion (see for instance the interview with Richard Feynman on the 'Why' question `https://www.youtube.com/watch?v=36GT2zI8lVA`)
- To explain predictions one needs a clear context to define explanations (user defined)
- In machine learning, we usually use a subset of the example that are 'responsible' for the prediction. That is, changing their values would change the prediction
- Explainability can be applied to black box models as a post processing step

Explainability

Explainability

- Explaining black box models is usually done by probing the model few times

Explainability

- Explaining black box models is usually done by probing the model few times
- Sometime it is not even possible to explain black box models

Explainability

- Explaining black box models is usually done by probing the model few times
- Sometime it is not even possible to explain black box models
- No theoretical guarantees

Explainability

- Explaining black box models is usually done by probing the model few times
- Sometime it is not even possible to explain black box models
- No theoretical guarantees
- It gets worse! Since different explanations can be used, one might pick a particular explanation to hide model biases (this is observed with many commercial tools!)

Explainability

- Explaining black box models is usually done by probing the model few times
- Sometime it is not even possible to explain black box models
- No theoretical guarantees
- It gets worse! Since different explanations can be used, one might pick a particular explanation to hide model biases (this is observed with many commercial tools!)
- Imagine a credit score black box model where a client might have several explanations regarding the refusal. The company might pick an explanation that doesn't show certain bias (such as predictions based on the gender)

Back to Interpretability

- Interpretability guarantees the transparency of the explanations
- No post-processing (in the sense of probing the model) is necessary for explanations. It is enough to look at the model
- However sometimes the explanations are not optimal (in the size of set inclusion). In this case, a user might ask for minimal explanations. This task can be done as a post-processing step
- Unfortunately, interpretable models (so far) are not adapted to all applications (for instance in tumor detection and computer vision). Such applications depend heavily on recent advances of black box models

Think about it..

Geoffrey Hinton
@geoffreyhinton

Suppose you have cancer and you have to choose between a black box AI surgeon that cannot explain how it works but has a 90% cure rate and a human surgeon with an 80% cure rate. Do you want the AI surgeon to be illegal?

8:37 PM · Feb 20, 2020 · Twitter Web App

Part 4: Training Algorithms

Notation

Notation

- The objective function is to minimise the error of mis-classification in the context of binary Classification

Notation

- The objective function is to minimise the error of mis-classification in the context of binary Classification
- $\mathbb{F} = \{f_1, \ldots f_k\}$ is a set of binary features (or attributes).

Notation

- The objective function is to minimise the error of mis-classification in the context of binary Classification
- $\mathbb{F} = \{f_1, \ldots f_k\}$ is a set of binary features (or attributes).
- The data is a collection of examples $\{e_1, \ldots, e_n\}$

Notation

- The objective function is to minimise the error of mis-classification in the context of binary Classification
- $\mathbb{F} = \{f_1, \dots f_k\}$ is a set of binary features (or attributes).
- The data is a collection of examples $\{e_1, \dots, e_n\}$
- An example e_i is represented as $(x_1, \dots x_k, y_i)$ where x_i are the values associated to the different features and $y_i \in \{0, 1\}$ is the class of e_i

Optimal Decision Trees

Optimal Decision Trees

- Propose a recursive algorithm to find an optimal decision tree
- You can use the following oracles:
- $SelectClass(\mathbb{E})$: returns the most probable class in the dataset $\mathbb{E}$
- $Explore(\mathbb{E}, info))$ a Boolean that indicates if the algorithm should develop more the tree

A Recursive Exact Algorithm To Find an Optimal Tree

Algorithm 18 OptimalTree

Require: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: $(Decisiontree, Error)$

if $Explore(\mathbb{E}, info))$ **then**

  for $j \in [1, K]$ **do**

    $Left_j \leftarrow \{x \in E | f_j = 0\}$

    $Right_j \leftarrow \{x \in E | f_j = 1\}$

    $Info_j \leftarrow$ Stopping information as if f_j is selected

    $(RightTree_j, RightError_j) \leftarrow OptimalTree(F \setminus \{f_j\}, Right_j, f_j, Info_j)$

    $(LeftTree_j, LeftError_j) \leftarrow OptimalTree(F \setminus \{f_j\}, Left_j, f_j, Info_j)$

    $Error_j \leftarrow RightError_j + LeftError_j$

  end for

  $j^* \leftarrow ArgMin\{Error_j\}$

  return $(f_{j^*}, LeftTree_{j^*}, RightTree_{j^*})$

else

  $C \leftarrow SelectClass(\mathbb{E})$

  $FixedError \leftarrow$ Error when choosing C

  return $(SelectClass(\mathbb{E}), FixedError)$

end if;

Search Symmetry

- Any two branches that share the exact same features (in different order) have the exact same best solutions.

Search Symmetry

- Any two branches that share the exact same features (in different order) have the exact same best solutions.
- In the recursive algorithm, one can avoid this redundant calls by chashing the results of each branch.
- The cashing contains strictly different branches

Optimal Rule Lists

Optimal Rule Lists

- A prediction rule is defined as an If-Condition-Then-Prediction

Optimal Rule Lists

- A prediction rule is defined as an If-Condition-Then-Prediction
- The "condition" is called antecedent
- A **rule list** model is an ordered list of m rules. Rule i is checked only when all previous rules do not apply
- If all rules do not apply, a majority vote prediction is used
- We want to find an algorithm that builds an optimal rule list for binary classification using a given set of pre-mined antecedents

Optimal Rule Lists

- A prediction rule is defined as an If-Condition-Then-Prediction
- The "condition" is called antecedent
- A **rule list** model is an ordered list of m rules. Rule i is checked only when all previous rules do not apply
- If all rules do not apply, a majority vote prediction is used
- We want to find an algorithm that builds an optimal rule list for binary classification using a given set of pre-mined antecedents
- The set of pre-mined antecedents is denoted by $A = \{a_1, \ldots a_l\}$

Optimal Rule Lists

- A prediction rule is defined as an If-Condition-Then-Prediction
- The "condition" is called antecedent
- A **rule list** model is an ordered list of m rules. Rule i is checked only when all previous rules do not apply
- If all rules do not apply, a majority vote prediction is used
- We want to find an algorithm that builds an optimal rule list for binary classification using a given set of pre-mined antecedents
- The set of pre-mined antecedents is denoted by $A = \{a_1, \ldots a_l\}$

Example

```
if [priors:>3] then [recidivism]
else if [age:21-22 && gender:Male] then [recidivism]
else if [age:18-20] then [recidivism]
else if [age:23-25 && priors:2-3] then [recidivism]
else [no recidivism]
```

Rule list 5. Example of an unconstrained rule list found by FairCORELS on COMPAS dataset, with Accuracy $= 0.681$, $\mathrm{UNF}_{\mathrm{EOdds}} = 0.217$ and

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 19 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
Allowed length H
Current rule list R
A set of antecedents A
Result: $(RuleList, Error)$

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 20 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
Allowed length H
Current rule list R
A set of antecedents A
Result: $(RuleList, Error)$
$BestError \leftarrow |\mathbb{E}|$
$Best \leftarrow \emptyset$

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 21 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
  Allowed length H
  Current rule list R
  A set of antecedents A
  Result: $(RuleList, Error)$
  $BestError \leftarrow |\mathbb{E}|$
  $Best \leftarrow \emptyset$
  if $Length < H$ **then**

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 22 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
  Allowed length H
  Current rule list R
  A set of antecedents A
  Result: $(RuleList, Error)$
  $BestError \leftarrow |\mathbb{E}|$
  $Best \leftarrow \emptyset$
  if $Length < H$ **then**
    for $a \in A$ **do**

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 23 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
Allowed length H
Current rule list R
A set of antecedents A
Result: $(RuleList, Error)$
$BestError \leftarrow |\mathbb{E}|$
$Best \leftarrow \emptyset$
if $Length < H$ **then**
  for $a \in A$ **do**
    $Prediction \leftarrow$ Majority class if a is applied
    $ErrorP \leftarrow Error(a \implies Prediction)$ on $\mathbb{E}$
    $E_a \leftarrow$ The dataset after applying a

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 24 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
Allowed length H
Current rule list R
A set of antecedents A
Result: $(RuleList, Error)$
$BestError \leftarrow |\mathbb{E}|$
$Best \leftarrow \emptyset$
if $Length < H$ **then**
  for $a \in A$ **do**
    $Prediction \leftarrow$ Majority class if a is applied
    $ErrorP \leftarrow Error(a \implies Prediction)$ on $\mathbb{E}$
    $E_a \leftarrow$ The dataset after applying a
    if $ErrorP + ErrorLowerBound(E_a, A \setminus \{a\}) <$ BestError) **then**
      $(RL, error) \leftarrow OptimalRuleList(E_a, A \setminus \{a\}, H - 1)$

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 25 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
Allowed length H
Current rule list R
A set of antecedents A
Result: $(RuleList, Error)$
$BestError \leftarrow |\mathbb{E}|$
$Best \leftarrow \emptyset$
if $Length < H$ **then**
    for $a \in A$ **do**
        $Prediction \leftarrow$ Majority class if a is applied
        $ErrorP \leftarrow Error(a \implies Prediction)$ on $\mathbb{E}$
        $E_a \leftarrow$ The dataset after applying a
        if $ErrorP + ErrorLowerBound(E_a, A \setminus \{a\}) <$ BestError) **then**
            $(RL, error) \leftarrow OptimalRuleList(E_a, A \setminus \{a\}, H - 1)$
            if $error + ErrorP < BestError$ **then**
                $BestError \leftarrow error + ErrorP$
                $Best \leftarrow (R + RL)$
            end if

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 26 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
  Allowed length H
  Current rule list R
  A set of antecedents A
  Result: $(RuleList, Error)$
  $BestError \leftarrow |\mathbb{E}|$
  $Best \leftarrow \emptyset$
  if $Length < H$ **then**
    for $a \in A$ **do**
      $Prediction \leftarrow$ Majority class if a is applied
      $ErrorP \leftarrow Error(a \implies Prediction)$ on $\mathbb{E}$
      $E_a \leftarrow$ The dataset after applying a
      if $ErrorP + ErrorLowerBound(E_a, A \setminus \{a\}) <$ BestError) **then**
        $(RL, error) \leftarrow OptimalRuleList(E_a, A \setminus \{a\}, H - 1)$
        if $error + ErrorP < BestError$ **then**
          $BestError \leftarrow error + ErrorP$
          $Best \leftarrow (R + RL)$
        end if
      end if
    end for
  else

A Branch and Bound Algorithm for Optimal Rule Lists

Algorithm 27 OptimalRuleList

Require: $\mathbb{E} = \{e_1, \ldots e_n\}$
  Allowed length H
  Current rule list R
  A set of antecedents A
  Result: $(RuleList, Error)$
  $BestError \leftarrow |\mathbb{E}|$
  $Best \leftarrow \emptyset$
  if $Length < H$ **then**
    for $a \in A$ **do**
      $Prediction \leftarrow$ Majority class if a is applied
      $ErrorP \leftarrow Error(a \implies Prediction)$ on $\mathbb{E}$
      $E_a \leftarrow$ The dataset after applying a
      if $ErrorP + ErrorLowerBound(E_a, A \setminus \{a\}) <$ BestError) **then**
        $(RL, error) \leftarrow OptimalRuleList(E_a, A \setminus \{a\}, H - 1)$
        if $error + ErrorP < BestError$ **then**
          $BestError \leftarrow error + ErrorP$
          $Best \leftarrow (R + RL)$
        end if
      end if
    end for
  else
    $Best \leftarrow (R, Else)$
    $BestError \leftarrow Error(R) + Error$ of majority prediction on $\mathbb{E}$;
  end if
  return $(Best, BestError)$

- How to Find Lower Bounds of the objective funtion at each node ?
    - By considering a best hypothetical senario at each node, one can compute its objective function and use its value (that we denote by *bound*) as a bound for the rest of the search tree
    - If *bound* is worse that the best objective function found, one can safely prune the current note and not exploring its children
- Symmetry Breaking ?
    - If two nodes share the same set of antecedents then necessarily they share the best error value. Therefore, one can use caching to save the objective value for each set of antecedents that are already explored. By doing so, each time a sequence of antecedents is about to be explored, one can check whether it's correspondent set is already explored. If it is the case, then the search space can be pruned.

References

C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *CoRR*, vol. abs/2103.11251, 2021.

J. R. Quinlan, *C4.5: Programs for Machine Learning.*
Morgan Kaufmann, 1993.

L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees.*
Wadsworth, 1984.

H. Hu, M. Siala, E. Hebrard, and M. Huguet, "Learning optimal decision trees with maxsat and its integration in adaboost," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020* (C. Bessiere, ed.), pp. 1170–1176, ijcai.org, 2020.

E. Demirovic, A. Lukina, E. Hebrard, J. Chan, J. Bailey, C. Leckie, K. Ramamohanarao, and P. J. Stuckey, "Murtree: Optimal decision trees via dynamic programming and search," *J. Mach. Learn. Res.*, vol. 23, pp. 26:1–26:47, 2022.