An Introduction to Supervised Machine Learning

Mohamed Siala
https://siala.github.io/

INSA-Toulouse & LAAS-CNRS

March 23, 2022

Context

Context

- Exponential increase of real-life applications of machine learning

Context

- Exponential increase of real-life applications of machine learning
- This is an introductory course. Next year you will follow more advanced courses (depending on your orientation)

Context

- Exponential increase of real-life applications of machine learning
- This is an introductory course. Next year you will follow more advanced courses (depending on your orientation)
- You need basic knowledge regarding linear algebra, algorithms and probability

Context

- Exponential increase of real-life applications of machine learning
- This is an introductory course. Next year you will follow more advanced courses (depending on your orientation)
- You need basic knowledge regarding linear algebra, algorithms and probability
- I ask questions very often, so please be interactive

Context

- Exponential increase of real-life applications of machine learning
- This is an introductory course. Next year you will follow more advanced courses (depending on your orientation)
- You need basic knowledge regarding linear algebra, algorithms and probability
- I ask questions very often, so please be interactive
- Feel free to stop me anytime. There is no stupid question!

Context

- Exponential increase of real-life applications of machine learning
- This is an introductory course. Next year you will follow more advanced courses (depending on your orientation)
- You need basic knowledge regarding linear algebra, algorithms and probability
- I ask questions very often, so please be interactive
- Feel free to stop me anytime. There is no stupid question!
- Speak up if you spot a typo or an error. You might get bonus points!!

Context

- Exponential increase of real-life applications of machine learning
- This is an introductory course. Next year you will follow more advanced courses (depending on your orientation)
- You need basic knowledge regarding linear algebra, algorithms and probability
- I ask questions very often, so please be interactive
- Feel free to stop me anytime. There is no stupid question!
- Speak up if you spot a typo or an error. You might get bonus points!!
- The evaluation will be decided later. Most likely it will be based on lecture questions and practical sessions

Context

- Exponential increase of real-life applications of machine learning
- This is an introductory course. Next year you will follow more advanced courses (depending on your orientation)
- You need basic knowledge regarding linear algebra, algorithms and probability
- I ask questions very often, so please be interactive
- Feel free to stop me anytime. There is no stupid question!
- Speak up if you spot a typo or an error. You might get bonus points!!
- The evaluation will be decided later. Most likely it will be based on lecture questions and practical sessions
- The course is articulated around three parts: introduction, interpretable machine learning (myself), and neural networks (Arthur Bit Monnot)

References

T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition.*
Springer Series in Statistics, Springer, 2009.

S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach, Third International Edition.*
Pearson Education, 2010.

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,

Course Syllabus

1. Part 1: Introduction
   - Context
   - Supervised Machine Learning
   - Deeper Evaluations
2. Interpretability
   - Motivation
   - Interpretable Models
   - Interpretability vs. Explanability

Part 1: Introduction

Context

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

Figure 1: How to cycle? [1]

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

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

Figure 2: How to teach a child animal recognition? [2]

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

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

Figure 3: How to predict a player's performance? [3]

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

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

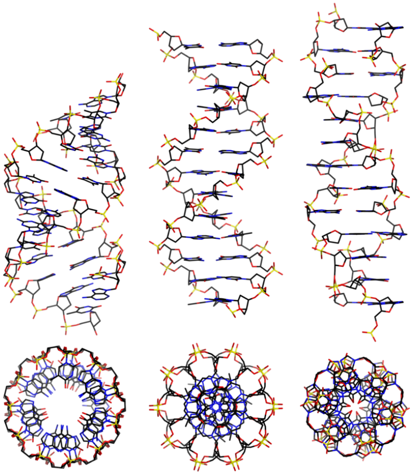


Figure 4: Analysis of evolutionary biology based on DNA patterns [4]

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

- Cycling: It needs a sequence of successful/unsuccessful trials!

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

  $\Longrightarrow$ Show some pictures per animal and let the child learn

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

  $\Longrightarrow$ Show some pictures per animal and let the child learn
- Player performance: No straightforward formulae. Keep track of its latest performances and predict accordingly

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

  $\Longrightarrow$ Show some pictures per animal and let the child learn
- Player performance: No straightforward formulae. Keep track of its latest performances and predict accordingly
- DNA clustering: Let the machine discover by itself the different patterns.

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

  $\Longrightarrow$ Show some pictures per animal and let the child learn
- Player performance: No straightforward formulae. Keep track of its latest performances and predict accordingly
- DNA clustering: Let the machine discover by itself the different patterns.
- That's pretty much the philosophy of machine learning: feed the computer some date, and let it learn by itself

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

  $\Longrightarrow$ Show some pictures per animal and let the child learn
- Player performance: No straightforward formulae. Keep track of its latest performances and predict accordingly
- DNA clustering: Let the machine discover by itself the different patterns.
- That's pretty much the philosophy of machine learning: feed the computer some date, and let it learn by itself
- Note 1: Some computational problems are simply not solvable in a traditionally way and need machine learning

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

  $\Longrightarrow$ Show some pictures per animal and let the child learn
- Player performance: No straightforward formulae. Keep track of its latest performances and predict accordingly
- DNA clustering: Let the machine discover by itself the different patterns.
- That's pretty much the philosophy of machine learning: feed the computer some date, and let it learn by itself
- Note 1: Some computational problems are simply not solvable in a traditionally way and need machine learning
- Note 2: Machine learning is not always the solution! Consider for instance basic arithmetic operations

- Cycling: It needs a sequence of successful/unsuccessful trials!
- Animal recognition: It does not make sense to show the picture of every animal!

  $\Longrightarrow$ Show some pictures per animal and let the child learn
- Player performance: No straightforward formulae. Keep track of its latest performances and predict accordingly
- DNA clustering: Let the machine discover by itself the different patterns.
- That's pretty much the philosophy of machine learning: feed the computer some date, and let it learn by itself
- Note 1: Some computational problems are simply not solvable in a traditionally way and need machine learning
- Note 2: Machine learning is not always the solution! Consider for instance basic arithmetic operations

Examples of Machine Learning Applications [1]

- Autonomous cars
- Flying drones
- Face recognition
- Computer vision
- Natural language processing
- Music/movie recommendation
- Dating apps
- Friends recommendation
- Weather prediction
- Trading
- ...

The Big Picture

It depends on the problem at hand and on the nature of data!

The Big Picture

It depends on the problem at hand and on the nature of data!

1. **Labelled data**:
    - The task of animal recognition is to predict the animal in a given picture. The data (used to "train" the computer) is a set of pictures and each picture is associated to an animal. In this case, the **label** is a **category**

The Big Picture

It depends on the problem at hand and on the nature of data!

1. **Labelled data**:
   - The task of animal recognition is to predict the animal in a given picture. The data (used to “train” the computer) is a set of pictures and each picture is associated to an animal. In this case, the **label** is a **category**
   - In trading, the task is to predict the price evolution of a given share and the data is simply historical evolution of the share. In this case the data is **labelled** with **real numbers**.

The Big Picture

It depends on the problem at hand and on the nature of data!

1. **Labelled data**:
   - The task of animal recognition is to predict the animal in a given picture. The data (used to "train" the computer) is a set of pictures and each picture is associated to an animal. In this case, the **label** is a **category**
   - In trading, the task is to predict the price evolution of a given share and the data is simply historical evolution of the share. In this case the data is **labelled** with **real numbers**.
2. **Unlabelled data**: For instance, when using clustering to evaluate the density of a population. The data can simply be a set of coordinates with no labels.

The Big Picture

It depends on the problem at hand and on the nature of data!

1. **Labelled data**:
    - The task of animal recognition is to predict the animal in a given picture. The data (used to "train" the computer) is a set of pictures and each picture is associated to an animal. In this case, the **label** is a **category**
    - In trading, the task is to predict the price evolution of a given share and the data is simply historical evolution of the share. In this case the data is **labelled** with **real numbers**.
2. **Unlabelled data**: For instance, when using clustering to evaluate the density of a population. The data can simply be a set of coordinates with no labels.
3. **Continuously updated data**: The data is continuously updated according to previous experiences: For instance, a robot that tries to ride a bicycle learns how to bike by a sequence of trials

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)

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

Figure 5: How to teach a child animal recognition? [5]

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

Figure 5: How to teach a child animal recognition? [5]

Classification task

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

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

Figure 6: How to predict a player's performance? [6]

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

Figure 6: How to predict a player's performance? [6]

Regression task

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

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

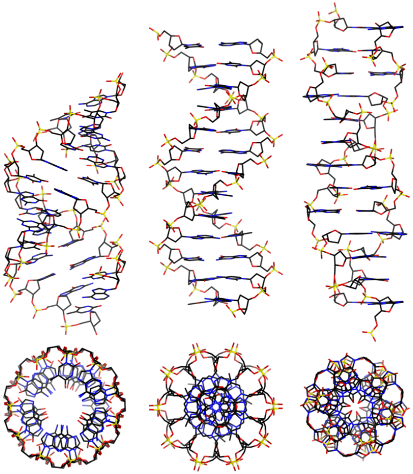


Figure 7: Analysis of evolutionary biology based on DNA patterns [7]

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

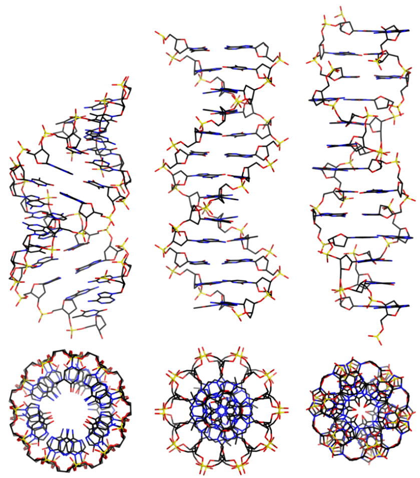


Figure 7: Analysis of evolutionary biology based on DNA patterns [7]

Unsupervised learning (clustering) task

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

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

Figure 8: How to cycle? [8]

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

Figure 8: How to cycle? [8]

Reinforcement learning

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

Supervised Machine Learning

Supervised Machine Learning: The Basics

- We focus in this course on Supervised ML
- Examples of applications
  - Tumor detection:

Supervised Machine Learning: The Basics

- We focus in this course on Supervised ML
- Examples of applications
  - Tumor detection: The data is a collection of brain scans. Each scan is associated with a label indicating the type of tumor
    $\Longrightarrow$ Classification
  - Credit score:

Supervised Machine Learning: The Basics

- We focus in this course on Supervised ML
- Examples of applications
  - Tumor detection: The data is a collection of brain scans. Each scan is associated with a label indicating the type of tumor
    $\Longrightarrow$ Classification
  - Credit score: The data is a collection of clients profiles (age, salary, genre (?), job, ...) with a positive or negative feedback
    $\Longrightarrow$ Binary classification
  - Precipitation prediction:

Supervised Machine Learning: The Basics

- We focus in this course on Supervised ML
- Examples of applications
  - Tumor detection: The data is a collection of brain scans. Each scan is associated with a label indicating the type of tumor
    $\Longrightarrow$ Classification
  - Credit score: The data is a collection of clients profiles (age, salary, genre (?), job, ...) with a positive or negative feedback
    $\Longrightarrow$ Binary classification
  - Precipitation prediction: (loosely speaking) the data is a collection of sequential weather conditions and the purpose is to predict the Precipitation chance (real value)
    $\Longrightarrow$ Regression

Problem Definition [2]

Problem Definition [2]

- Given a historical 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 function f

Problem Definition [2]

- Given a historical 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 function f
- Find a function f_h (called a hypothesis or model) that approximates the true function f
- 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

Model Evaluation: Classification

Model Evaluation: Classification

- Training accuracy: percentage of training examples that are well predicted

Model Evaluation: Classification

- Training accuracy: percentage of training examples that are well predicted
- Testing accuracy: percentage of testing examples that are well predicted

Model Evaluation: Classification

- Training accuracy: percentage of training examples that are well predicted
- Testing accuracy: percentage of testing examples that are well predicted
- The concept of **generalisation** is precisely the quality of testing accuracy

Model Evaluation: Classification

- Training accuracy: percentage of training examples that are well predicted
- Testing accuracy: percentage of testing examples that are well predicted
- The concept of **generalisation** is precisely the quality of testing accuracy

Classification Evaluation: The Confusion Matrix

Classification Evaluation: The Confusion Matrix

- Accuracy alone hinders the way predictions are made. Model evaluation needs more refinement

Classification Evaluation: The Confusion Matrix

- Accuracy alone hinders the way predictions are made. Model evaluation needs more refinement
- Consider binary classification (positive/negative classes) with 80% testing accuracy

Classification Evaluation: The Confusion Matrix

- Accuracy alone hinders the way predictions are made. Model evaluation needs more refinement
- Consider binary classification (positive/negative classes) with 80% testing accuracy
- 80% seems good, however, a deeper investigation shows that most of negative examples are wrongly predicted! This shows a biases in the model (biased towards positive examples)

Classification Evaluation: The Confusion Matrix

- Accuracy alone hinders the way predictions are made. Model evaluation needs more refinement
- Consider binary classification (positive/negative classes) with 80% testing accuracy
- 80% seems good, however, a deeper investigation shows that most of negative examples are wrongly predicted! This shows a biases in the model (biased towards positive examples)
- The purpose of the confusion matrix is precisely to have a refined evaluation

Classification Evaluation: The Confusion Matrix

- Accuracy alone hinders the way predictions are made. Model evaluation needs more refinement
- Consider binary classification (positive/negative classes) with 80% testing accuracy
- 80% seems good, however, a deeper investigation shows that most of negative examples are wrongly predicted! This shows a biases in the model (biased towards positive examples)
- The purpose of the confusion matrix is precisely to have a refined evaluation
- In the case on m classes, the matrix is of dimension $m \times m$ where $M[i][j]$ is the number of examples of class i that are predicted to be as the class b

Classification Evaluation: The Confusion Matrix

- Accuracy alone hinders the way predictions are made. Model evaluation needs more refinement
- Consider binary classification (positive/negative classes) with 80% testing accuracy
- 80% seems good, however, a deeper investigation shows that most of negative examples are wrongly predicted! This shows a biases in the model (biased towards positive examples)
- The purpose of the confusion matrix is precisely to have a refined evaluation
- In the case on m classes, the matrix is of dimension $m \times m$ where $M[i][j]$ is the number of examples of class i that are predicted to be as the class b
- The model bias can be easily seen: For instance, one can answer statistical queries such as: is the error evenly distributed? to which class the model is likely to predict? ...

The Confusion Matrix in the Case of Binary Classification

The Confusion Matrix in the Case of Binary Classification

- A positive class and a negative class

The Confusion Matrix in the Case of Binary Classification

- A positive class and a negative class
- The confusion matrix is of dimension 2×2

The Confusion Matrix in the Case of Binary Classification

- A positive class and a negative class
- The confusion matrix is of dimension 2×2
- True Positive Rate (TP rate): the likelihood that a positive example is well classified

The Confusion Matrix in the Case of Binary Classification

- A positive class and a negative class
- The confusion matrix is of dimension 2×2
- True Positive Rate (TP rate): the likelihood that a positive example is well classified
- False Positive Rate (FP rate): the likelihood that a positive example is wrongly classified

The Confusion Matrix in the Case of Binary Classification

- A positive class and a negative class
- The confusion matrix is of dimension 2×2
- True Positive Rate (TP rate): the likelihood that a positive example is well classified
- False Positive Rate (FP rate): the likelihood that a positive example is wrongly classified
- True Negative Rate (TN rate): the likelihood that a negative example is well classified

The Confusion Matrix in the Case of Binary Classification

- A positive class and a negative class
- The confusion matrix is of dimension 2×2
- True Positive Rate (TP rate): the likelihood that a positive example is well classified
- False Positive Rate (FP rate): the likelihood that a positive example is wrongly classified
- True Negative Rate (TN rate): the likelihood that a negative example is well classified
- False Negative Rate (FN rate): the likelihood that a negative example is wrongly classified

The Covid Example (1)

The Covid Example (1)

- Assume that we have a prediction model with 85% accuracy

The Covid Example (1)

- Assume that we have a prediction model with 85% accuracy
- 100 individuals: 70 positives and 30 negatives

The Covid Example (1)

- Assume that we have a prediction model with 85% accuracy
- 100 individuals: 70 positives and 30 negatives
- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

The Covid Example (1)

- Assume that we have a prediction model with 85% accuracy
- 100 individuals: 70 positives and 30 negatives
- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

- TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$

The Covid Example (1)

- Assume that we have a prediction model with 85% accuracy
- 100 individuals: 70 positives and 30 negatives
- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

- TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
- TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$

The Covid Example (1)

- Assume that we have a prediction model with 85% accuracy
- 100 individuals: 70 positives and 30 negatives
- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

- TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
- TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
- Positive individuals has 93% chance to be correctly predicted

The Covid Example (1)

- Assume that we have a prediction model with 85% accuracy
- 100 individuals: 70 positives and 30 negatives
- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

- TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
- TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
- Positive individuals has 93% chance to be correctly predicted
- Negative individuals has 66% chance to be correctly predicted

The Covid Example (2)

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
  - TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
  - TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
  - Positive individuals has 93% chance to be correctly predicted

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
  - TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
  - Positive individuals has 93% chance to be correctly predicted
  - Negative individuals has 66% chance to be correctly predicted

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
  - TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
  - Positive individuals has 93% chance to be correctly predicted
  - Negative individuals has 66% chance to be correctly predicted
- What is the probability for someone who tested positive to be truly positive?

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
  - TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
  - Positive individuals has 93% chance to be correctly predicted
  - Negative individuals has 66% chance to be correctly predicted
- What is the probability for someone who tested positive to be truly positive? $= 65/(65+10) = 86\%$

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
  - TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
  - Positive individuals has 93% chance to be correctly predicted
  - Negative individuals has 66% chance to be correctly predicted
- What is the probability for someone who tested positive to be truly positive? $= 65/(65 + 10) = 86\%$
- What is the probability for someone who tested negative to be truly negative?

The Covid Example (2)

- The confusion matrix:

	Positive	Negative
Positive	65	5
Negative	10	20

  - TP rate is $65/70 = 93\%$; FP rate is $5/70 = 7\%$
  - TN rate is $20/30 = 66\%$; FN rate is $10/30 = 33\%$
  - Positive individuals has 93% chance to be correctly predicted
  - Negative individuals has 66% chance to be correctly predicted
- What is the probability for someone who tested positive to be truly positive? $= 65/(65 + 10) = 86\%$
- What is the probability for someone who tested negative to be truly negative? $= 20/(5 + 20) = 80\%$

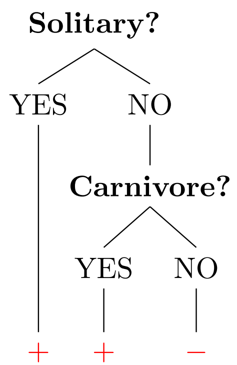
Toy Example: DTs to Predict The Likelihood of Animal Extinction

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

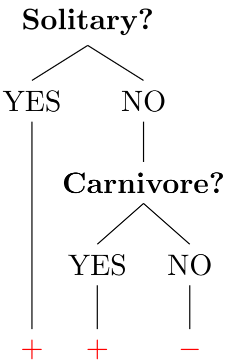
Toy Example: DTs to Predict The Likelihood of Animal Extinction

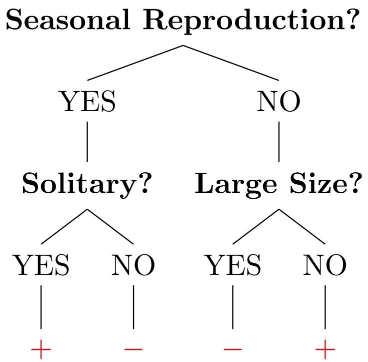
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes



Toy Example: DTs to Predict The Likelihood of Animal Extinction

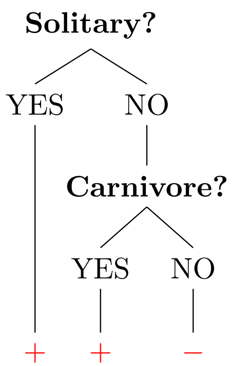
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

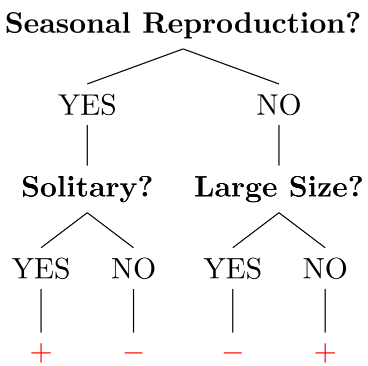




Toy Example: DTs to Predict The Likelihood of Animal Extinction

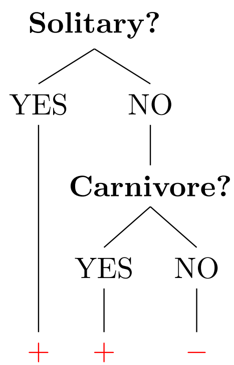
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

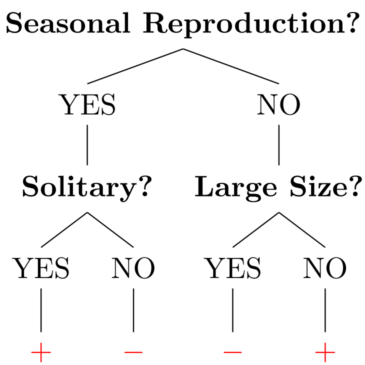




Toy Example: DTs to Predict The Likelihood of Animal Extinction

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

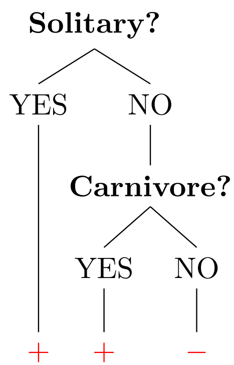


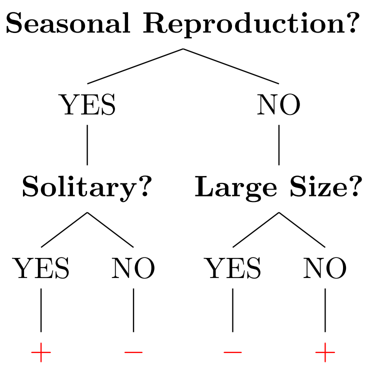


- Tabular data

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

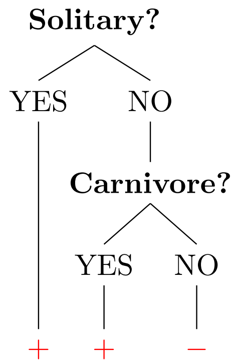


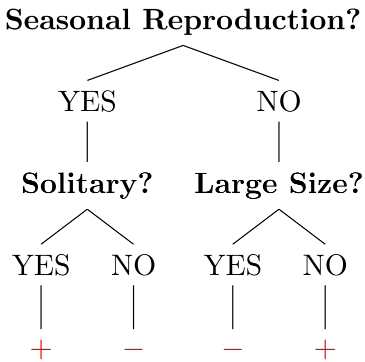


- Tabular data
- Hypothesis space: Decision trees

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

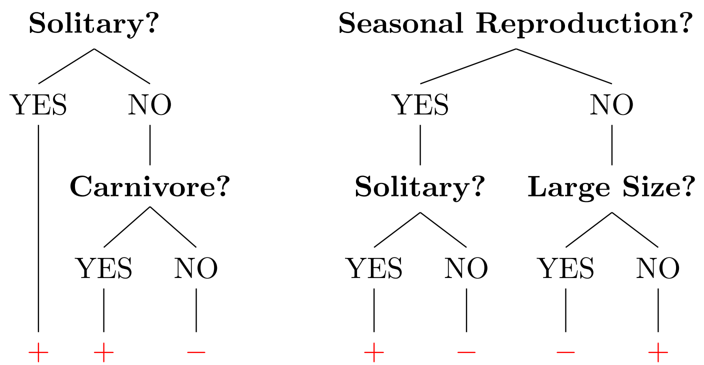




- Tabular data
- Hypothesis space: Decision trees
- Left tree: accuracy 2/5

Toy Example: DTs to Predict The Likelihood of Animal Extinction

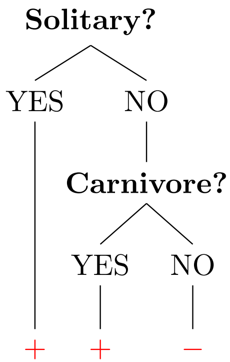
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

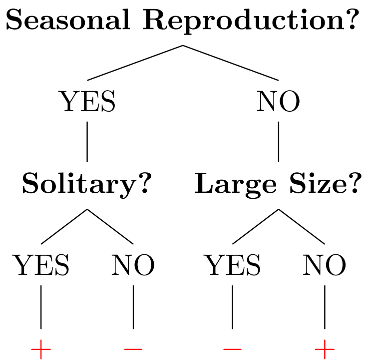


- Tabular data
- Hypothesis space: Decision trees
- Left tree: accuracy 2/5
- Right tree: accuracy 2/5

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
- Right tree: accuracy 2/5

Model Evaluation: Regression

Model Evaluation: Regression

- In the case of classification, a simple way to define error is to count the number of examples wrongly predicted

Model Evaluation: Regression

- In the case of classification, a simple way to define error is to count the number of examples wrongly predicted
- How about regression?

Model Evaluation: Regression

- In the case of classification, a simple way to define error is to count the number of examples wrongly predicted
- How about regression?
- Take the example of estimating the price of a house based on the surface and the distance to the city.

Model Evaluation: Regression

- In the case of classification, a simple way to define error is to count the number of examples wrongly predicted
- How about regression?
- Take the example of estimating the price of a house based on the surface and the distance to the city.
- Suppose that the real price of a house is 1 Million and the model predicted $999k$.

Model Evaluation: Regression

- In the case of classification, a simple way to define error is to count the number of examples wrongly predicted
- How about regression?
- Take the example of estimating the price of a house based on the surface and the distance to the city.
- Suppose that the real price of a house is 1 Million and the model predicted $999k$.
- Obviously it's not a correct prediction, however, it seems close enough! But how close?

Model Evaluation: Regression

- In the case of classification, a simple way to define error is to count the number of examples wrongly predicted
- How about regression?
- Take the example of estimating the price of a house based on the surface and the distance to the city.
- Suppose that the real price of a house is 1 Million and the model predicted $999k$.
- Obviously it's not a correct prediction, however, it seems close enough! But how close?
- We need an error function that take into account a notion of distance between true values and predicted values

Model Evaluation: Regression

Model Evaluation: Regression

- Two well known functions: Mean Absolute Error(MAE) and Mean Square Error(MSE) (or Root Mean Square Error($RMSE$))

Model Evaluation: Regression

- Two well known functions: Mean Absolute Error(MAE) and Mean Square Error(MSE) (or Root Mean Square Error($RMSE$))
- Consider a dataset with n examples where y is the vector of the true values and $\hat{y}$ is the vector of predicted values:

$$MAE = \frac{1}{n}\sum |y_i - \hat{y}_i|$$

$$MSE = \frac{1}{n}\sum (y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{MSE}$$

Beyond Traditional Evaluations: Crucial Predictions

Beyond Traditional Evaluations: Crucial Predictions

- Take the example of tumor detection

Beyond Traditional Evaluations: Crucial Predictions

- Take the example of tumor detection
- Predicting the non-presence of a tumor for a patient that has a tumor is a serious problem

Beyond Traditional Evaluations: Crucial Predictions

- Take the example of tumor detection
- Predicting the non-presence of a tumor for a patient that has a tumor is a serious problem
- How to deal with such situation?

Beyond Traditional Evaluations: Crucial Predictions

- Take the example of tumor detection
- Predicting the non-presence of a tumor for a patient that has a tumor is a serious problem
- How to deal with such situation?
- For instance, one can add a weight on the mis-classification error as follows: the cost of mis-classifying a patient with a tumor is 5 times the cost of mis-classifying a patient without a tumor

Beyond Traditional Evaluations: Crucial Predictions

- Take the example of tumor detection
- Predicting the non-presence of a tumor for a patient that has a tumor is a serious problem
- How to deal with such situation?
- For instance, one can add a weight on the mis-classification error as follows: the cost of mis-classifying a patient with a tumor is 5 times the cost of mis-classifying a patient without a tumor
- In this case the objective function is slightly different from the standard accuracy (weighted sum)

Computational Hardness

Out of these three questions, which one is the hardest and which one is the easiest (computationally)?

1. Build a perfect decision tree (a tree that classifies every example correctly) 100% accuracy ?
2. Build the best decision tree within a height h ?
3. Build a perfect decision tree with a minimum height ?

Computational Hardness

Out of these three questions, which one is the hardest and which one is the easiest (computationally)?

1. Build a perfect decision tree (a tree that classifies every example correctly) 100% accuracy ?
2. Build the best decision tree within a height h ?
3. Build a perfect decision tree with a minimum height ?

- Question 1 is the easiest: we can always develop children to classify every example
- Let k be the number of features; $f(n)$ be the complexity of Question 2 and $g(n)$ be the complexity of Question 3
- $g(n) = O(k \times f(n))$: We can solve Question 3 by solving Question 2 iteratively by increasing/decreasing the height

Computational Hardness

Out of these three questions, which one is the hardest and which one is the easiest (computationally)?

1. Build a perfect decision tree (a tree that classifies every example correctly) 100% accuracy ?
2. Build the best decision tree within a height h ?
3. Build a perfect decision tree with a minimum height ?

- Question 1 is the easiest: we can always develop children to classify every example
- Let k be the number of features; $f(n)$ be the complexity of Question 2 and $g(n)$ be the complexity of Question 3
- $g(n) = O(k \times f(n))$: We can solve Question 3 by solving Question 2 iteratively by increasing/decreasing the height
- Different objective functions can be defined (i.e., the training problem itsetf can have different definitions)
- The definition of the objective function with the hypothesis space has an impact on the complexity of training

To summarize

- Supervised Learning: The data is labelled

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)
- **Generalisation:** when the model generalizes well on unseen data

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)
- **Generalisation:** when the model generalizes well on unseen data
- Regression: Supervised Learning with real values

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)
- **Generalisation:** when the model generalizes well on unseen data
- Regression: Supervised Learning with real values
- Classification: Supervised Learning with discrete values (classes)

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)
- **Generalisation:** when the model generalizes well on unseen data
- Regression: Supervised Learning with real values
- Classification: Supervised Learning with discrete values (classes)
- Regression evaluation: accuracy in terms of minimum error (Mean Absolute Error, Root Mean Square Error)

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)
- **Generalisation:** when the model generalizes well on unseen data
- Regression: Supervised Learning with real values
- Classification: Supervised Learning with discrete values (classes)
- Regression evaluation: accuracy in terms of minimum error (Mean Absolute Error, Root Mean Square Error)
- Classification evaluation: accuracy in terms of examples well classified but also the confusion matrix (it can show some bias)

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)
- **Generalisation:** when the model generalizes well on unseen data
- Regression: Supervised Learning with real values
- Classification: Supervised Learning with discrete values (classes)
- Regression evaluation: accuracy in terms of minimum error (Mean Absolute Error, Root Mean Square Error)
- Classification evaluation: accuracy in terms of examples well classified but also the confusion matrix (it can show some bias)
- Depending on the problem at hand, the objective function can have have different forms

To summarize

- Supervised Learning: The data is labelled
- Each example is defined by a sequence of values (of the different attributes/features) with an output
- Training vs. Testing sets (disjoints)
- **Generalisation:** when the model generalizes well on unseen data
- Regression: Supervised Learning with real values
- Classification: Supervised Learning with discrete values (classes)
- Regression evaluation: accuracy in terms of minimum error (Mean Absolute Error, Root Mean Square Error)
- Classification evaluation: accuracy in terms of examples well classified but also the confusion matrix (it can show some bias)
- Depending on the problem at hand, the objective function can have have different forms
- The way the optimisation problem is defined impacts its computational complexity

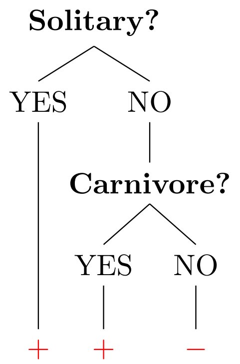
Questions & Exercises

Questions & Exercises

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

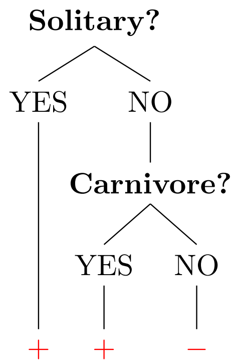
Questions & Exercises

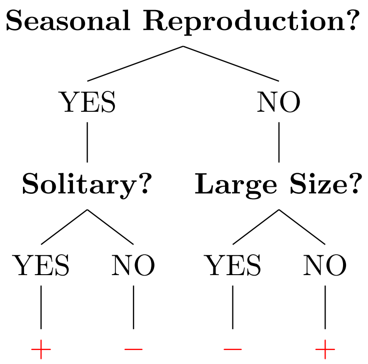
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes



Questions & Exercises

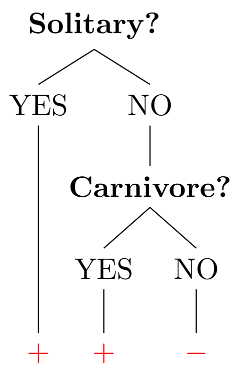
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

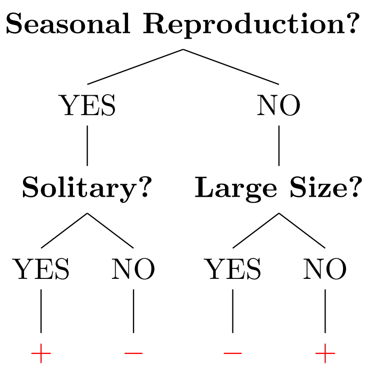




Questions & Exercises

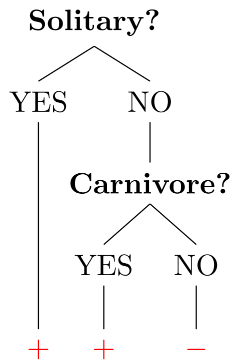
Big Size	Carnivore	Seasonal Reproduction	Solitary	Extinct
0	1	0	1	yes
1	0	0	1	yes
0	0	0	1	no
1	1	1	0	no
0	0	1	0	yes

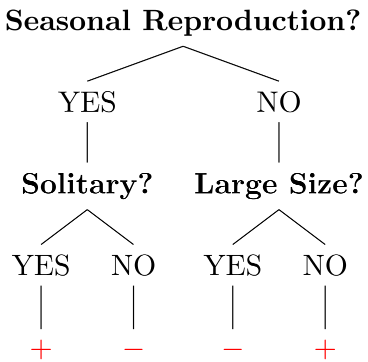




Questions & Exercises

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

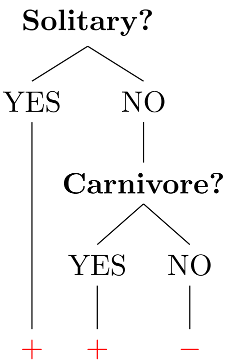


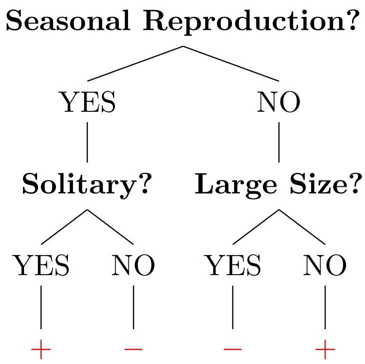


- Find a perfect tree

Questions & Exercises

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

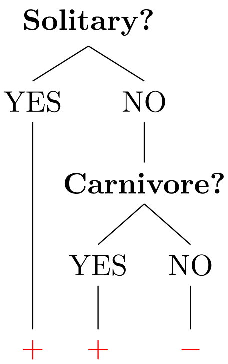


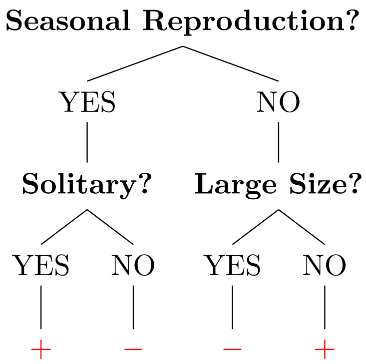


- Find a perfect tree
- Find a perfect tree with a minimum weight

Questions & Exercises

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

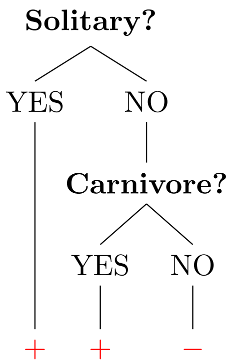


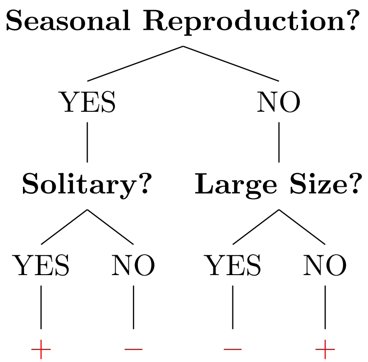


- Find a perfect tree
- Find a perfect tree with a minimum weight
- Find a best tree with height 2

Questions & Exercises

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

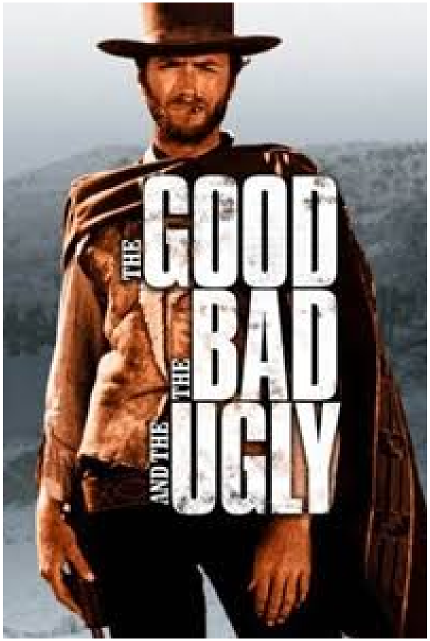




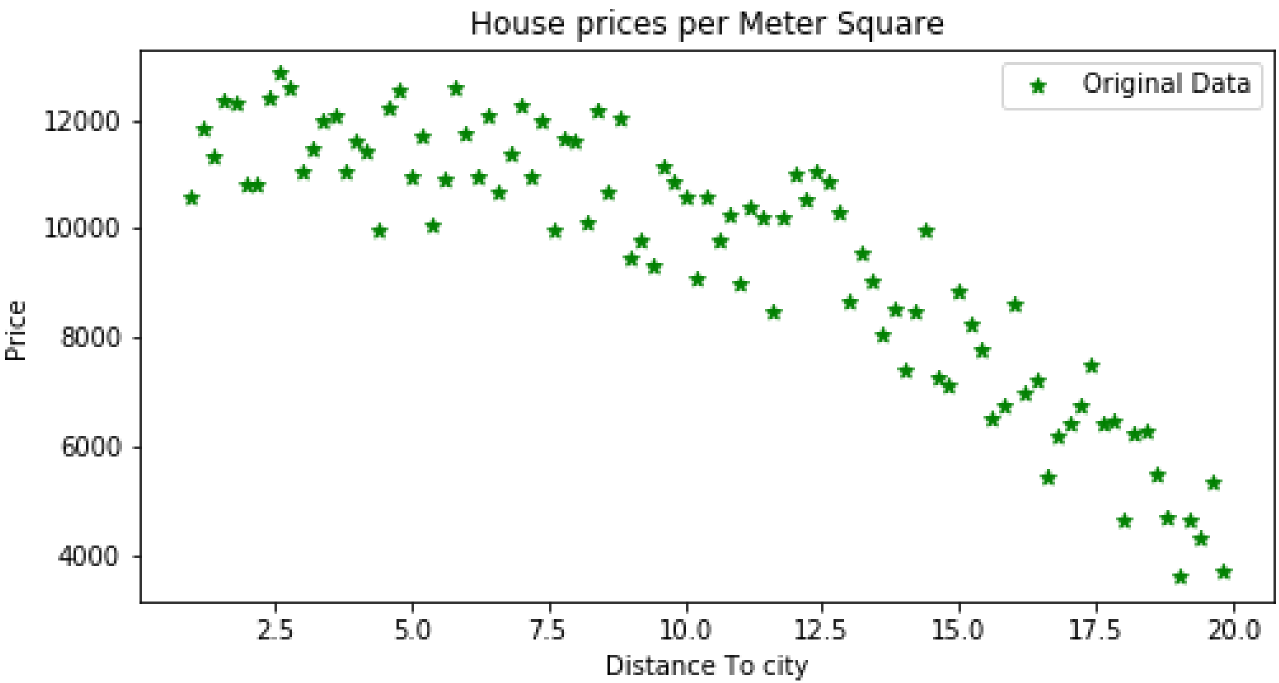
- Find a perfect tree
- Find a perfect tree with a minimum weight
- Find a best tree with height 2

Deeper Evaluations

Overfitting, Underfitting, and Goodfitting

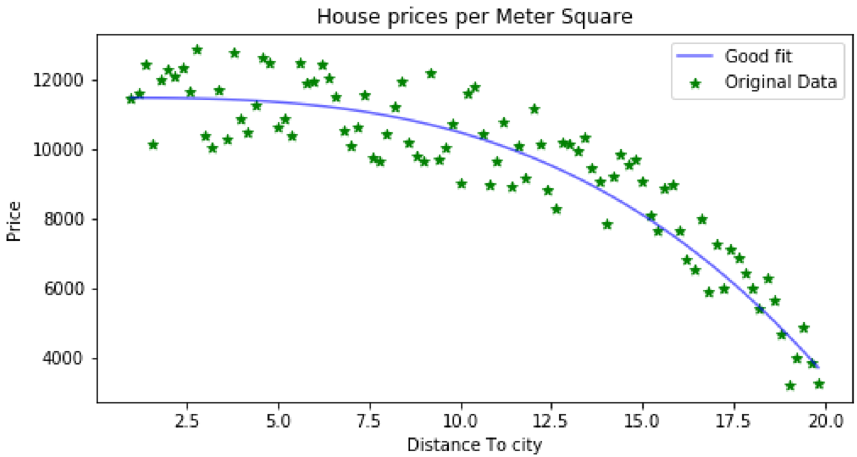


The Housing Prices Example



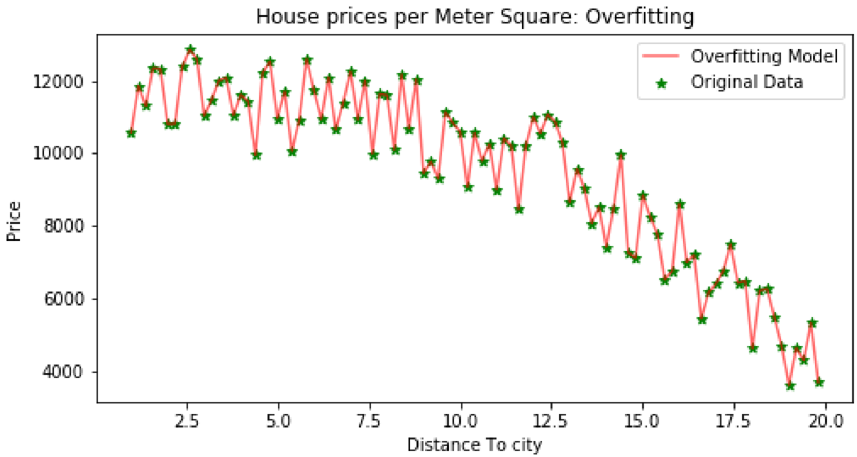
This data includes some **noise**. That is, points that are not correctly collected (which is often the case in real applications)

The Housing Prices Example: The Good

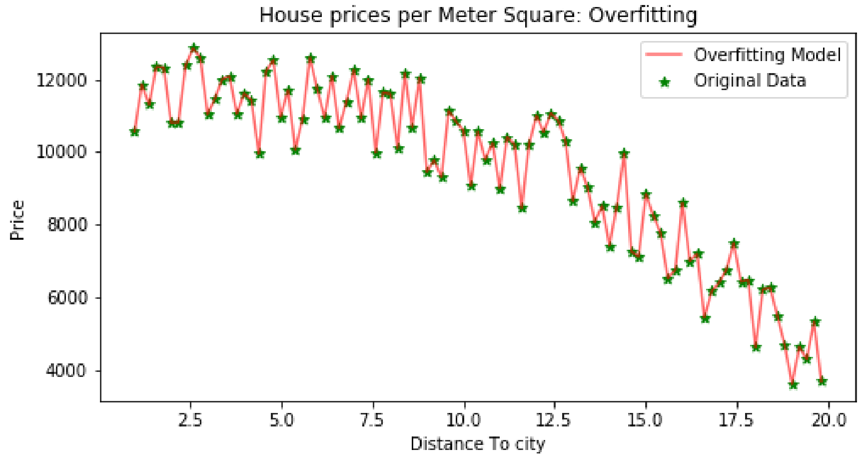


We can make an analogy to a smart student who has a good understanding of a lecture

The Housing Prices Example: The Bad (Overfitting)

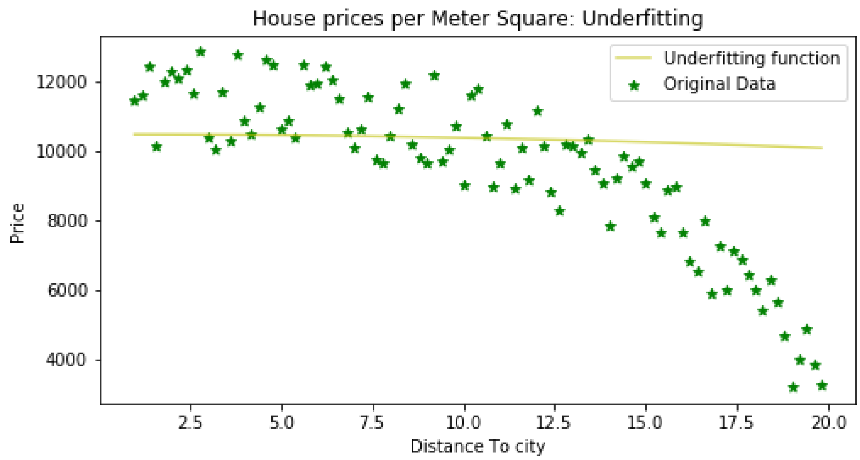


The Housing Prices Example: The Bad (Overfitting)

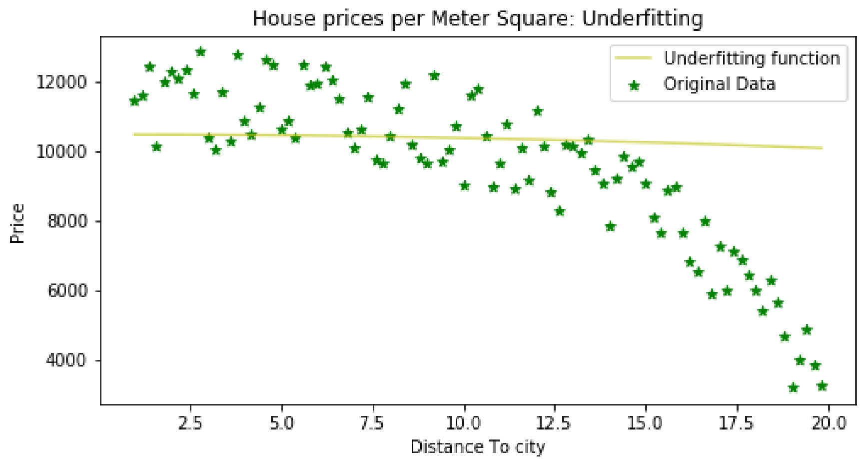


We can make an analogy to the student who "learns" the lecture mechanically without a real understanding.

The Housing Prices Example: The Ugly (Underfitting)

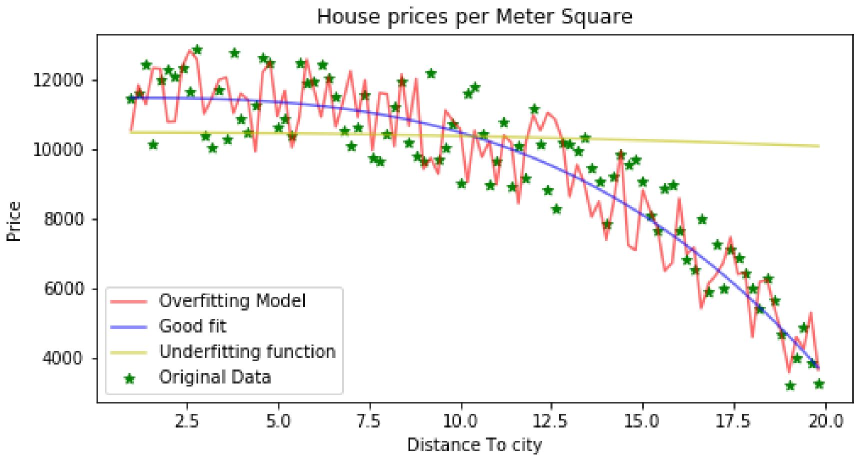


The Housing Prices Example: The Ugly (Underfitting)



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

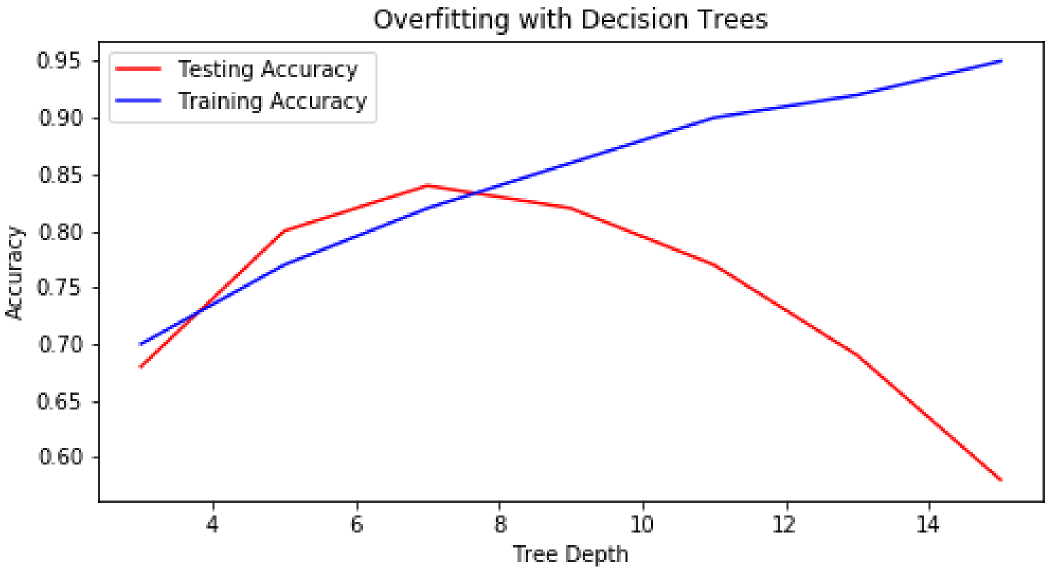
- Overfitting happens when the model tries to squeeze everything in including noise without an "intuitive understanding of the data"
- Underfitting happens when the model performs badly on the training and testing data (no real learning).

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

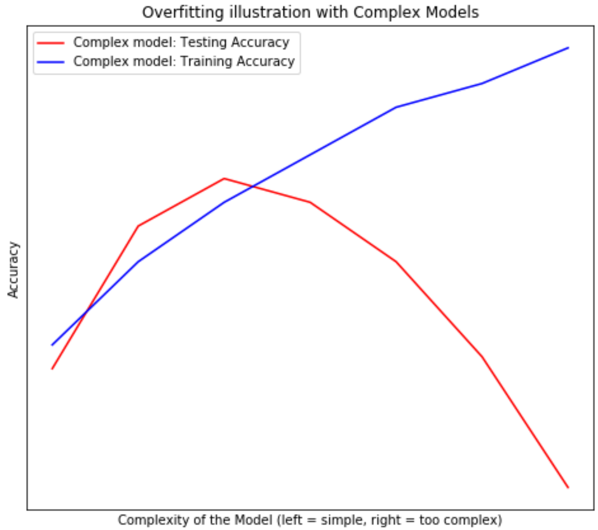
- The longer the tree, the better the training accuracy gets, however, this is not necessarily the case for the testing accuracy
- Testing accuracy increases at the beginning until a certain value (depth $= 7$), the it decreases afterwards

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

Training Algorithms Evaluation

- Suppose that we have a number of training algorithms for a given hypothesis space

Training Algorithms Evaluation

- Suppose that we have a number of training algorithms for a given hypothesis space
- How to choose the best?

Training Algorithms Evaluation

- Suppose that we have a number of training algorithms for a given hypothesis space
- How to choose the best?
- Generalisation should be evaluated a number of random splits

Training Algorithms Evaluation

- Suppose that we have a number of training algorithms for a given hypothesis space
- How to choose the best?
- Generalisation should be evaluated a number of random splits
- Also the confusion matrices can be used for evaluation

Training Algorithms Evaluation

- Suppose that we have a number of training algorithms for a given hypothesis space
- How to choose the best?
- Generalisation should be evaluated a number of random splits
- Also the confusion matrices can be used for evaluation
- A common way is to use the $k-$fold cross validation:

Training Algorithms Evaluation

- Suppose that we have a number of training algorithms for a given hypothesis space
- How to choose the best?
- Generalisation should be evaluated a number of random splits
- Also the confusion matrices can be used for evaluation
- A common way is to use the $k-$fold cross validation:
  1. Split the data into k folds

Training Algorithms Evaluation

- Suppose that we have a number of training algorithms for a given hypothesis space
- How to choose the best?
- Generalisation should be evaluated a number of random splits
- Also the confusion matrices can be used for evaluation
- A common way is to use the $k-$fold cross validation:
  1. Split the data into k folds
  2. Perform the training k times. At each iteration, a different fold is chosen as a testing set and the rest is used for training
  3. Typical values for k are 5 and 10

Overcome Overfitting

- How to avoid overfitting?

Overcome Overfitting

- How to avoid overfitting?
- The testing set is inaccessible at the moment of training
- We can sacrifice a part of the training set as a 'validation set' to evaluate the generalisation of the model.
- Basically, the training set has a subset for training and a subset for validation (evaluation)
- A common way is to use $k-$cross validation on the training set to overcome overfitting
- Also, we can restrict the hypothesis space with simple models

Complexity/Quality Tradeoff

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime (we will see later how to train decision trees/linear functions)

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime (we will see later how to train decision trees/linear functions)
- For instance, training quadratic functions is much harder (computationally) than training linear functions

Complexity/Quality Tradeoff

- Training algorithms have resources costs: memory and runtime (we will see later how to train decision trees/linear functions)
- 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 (we will see later how to train decision trees/linear functions)
- 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 (we will see later how to train decision trees/linear functions)
- 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 (we will see later how to train decision trees/linear functions)
- 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

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

Ockham's Razor Principle

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

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

Ockham's Razor Principle

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

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

Ockham's Razor Principle

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

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

Ockham's Razor Principle

- In the previous example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[9]: pick the simplest!

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

Ockham's Razor Principle

- In the previous example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[9]: pick the simplest!
- Simplicity is also hard to define

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

Ockham's Razor Principle

- In the previous example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[9]: 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

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

Ockham's Razor Principle

- In the previous example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[9]: 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 seems to be simpler

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

Ockham's Razor Principle

- In the previous example, we have two different trees with the same accuracy
- Which tree to choose?
- Hard to answer without specific requirements
- Ockham's Razor Principle[9]: 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 seems to be simpler
- In other cases it is very hard to define simplicity

[9] 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

Part 2: Interpretability

Motivation

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

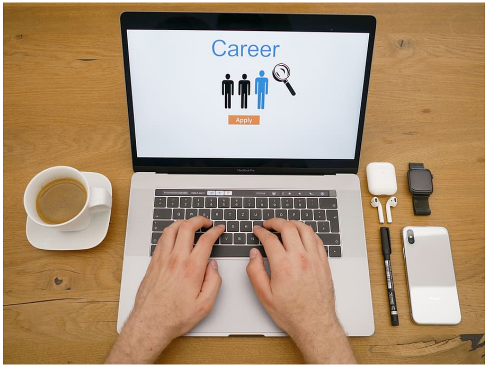
Using historical data to train risk assessment tools could mean that machines are copying the mistakes of the past.

By Karen Hao January 21, 2019

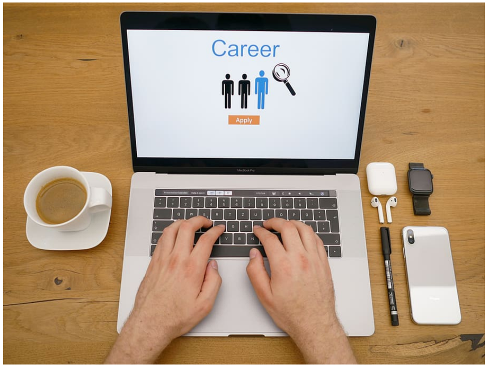
IAN WALDIE/GETTY IMAGES

Increasing Number of Real Life and Social AI Applications

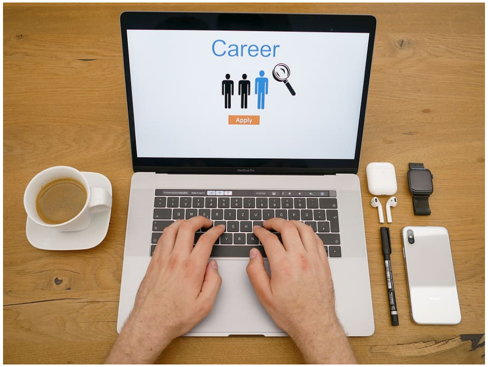
Increasing Number of Real Life and Social AI Applications



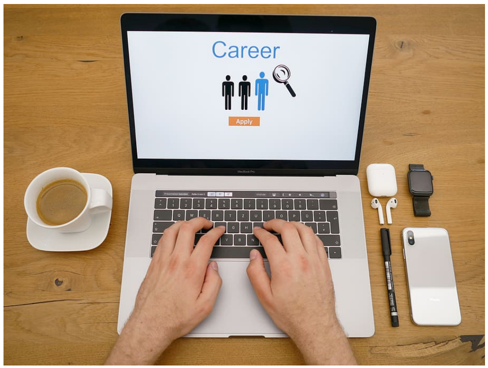
Increasing Number of Real Life and Social AI Applications

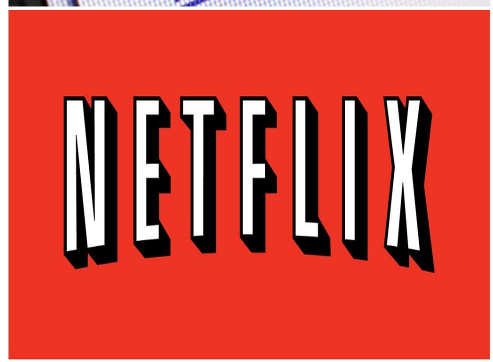


Increasing Number of Real Life and Social AI Applications



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).

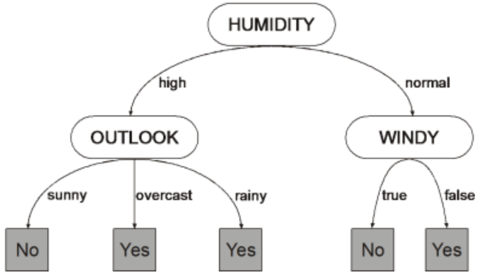
COMPASS data and Rule-based Predictions

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
...	...	...	...	...	...

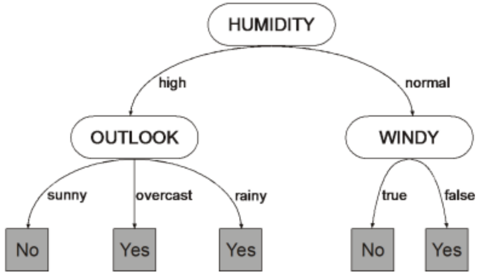
The problem is to predict recidivism. That is, the tendency of a convicted criminal to re-offend.

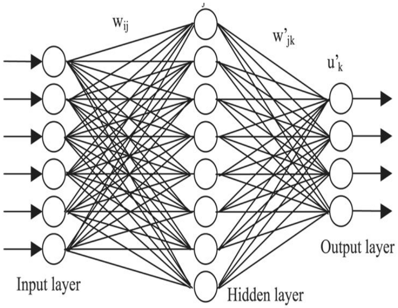
Black-Box vs Interpretable Models

Black-Box vs Interpretable Models



Black-Box vs Interpretable Models





Definitions [3]

- **Black-box model** : A formula that is either too complicated for any human to understand, or proprietary, so that one cannot understand its inner workings
- **Interpretable model** 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**

- We consider in this course tabular data (but extensions to other types is possible)
- Models:Decision trees, decision lists, decision rules, and linear functionns . . .

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

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

Example of Rule List found by FairCORELS

- Data : `https://www.kaggle.com/danofer/compass`
- FairCORELS: `https://github.com/ferryjul/fairCORELS`

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

Example of Rule List found by FairCORELS

- Data : `https://www.kaggle.com/danofer/compass`
- FairCORELS: `https://github.com/ferryjul/fairCORELS`

```
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 $\mathrm{UNF}_{\mathrm{CUAE}} = 0.046$

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 $\mathbb{F} = \{f_1, \ldots f_k\}$ 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

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
- 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 $\{e_1, \dots, e_n\}$

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= $\{e_1, e_2, \ldots e_7\}$

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)

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
- Since there are z values in the input space of $Compatible$, then there exists 2^z possible instantiation of the $Compatible$ function
- 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
- Since there are z values in the input space of *Compatible*, then there exists 2^z possible instantiation of the *Compatible* function
- 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
- Since there are z values in the input space of $Compatible$, then there exists 2^z possible instantiation of the $Compatible$ function
- 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!

Dealing with Intractability

Dealing with Intractability

Dealing with Intractability

- Enormous search space

Dealing with Intractability

- Enormous search space
- Building Short trees is usually intractable

Dealing with Intractability

- Enormous search space
- Building Short trees is usually intractable
- Exact algorithms hardly scale up

Dealing with Intractability

- Enormous search space
- Building Short trees 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 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: $\mathbb{F} = \{f_1, \dots f_k\}$, $\mathbb{E} = \{e_1, \dots e_n\}$, a parent node *parent*, and an information $info$ regarding the stopping conditions

Result: A decision tree

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

$f_j \leftarrow SelectBestFeature(data)$

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

Building a decision Tree: A Greedy Algorithm

Algorithm 4 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**

$f_j \leftarrow SelectBestFeature(data)$

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

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

Building a decision Tree: A Greedy Algorithm

Algorithm 5 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**

$f_j \leftarrow SelectBestFeature(data)$

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

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

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

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

Result: A decision tree

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

$f_j \leftarrow SelectBestFeature(data)$

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

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

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

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

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

Building a decision Tree: A Greedy Algorithm

Algorithm 8 GREEDY

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

Result: A decision tree

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

$f_j \leftarrow SelectBestFeature(data)$

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

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

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

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

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

return $(f_j, LeftTree, RightTree)$

else

return $SelectClass(\mathbb{E})$

end if;

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)*:

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

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

- Build a decision tree with the previous approach where the the height is at most 3, and the classification follows a majority rule

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 [4] and Gini Index is used in in CART (Classification and regression trees [5])

Back to Greedy Algorithms

- Different algorithms use different criteria
- For instance, Information gain is used in C4.5 [4] and Gini Index is used in in CART (Classification and regression trees [5])
- 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 [4] and Gini Index is used in in CART (Classification and regression trees [5])
- 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 [4] and Gini Index is used in in CART (Classification and regression trees [5])
- 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 [4] and Gini Index is used in in CART (Classification and regression trees [5])
- 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 [4] and Gini Index is used in in CART (Classification and regression trees [5])
- 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 [6, 7])

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

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.

Regression

Regression

Regression

- Restriction: an example is $e_i = (x_i, y_i)$ where $x_i, y_i \in \mathbb{R}$

Regression

- Restriction: an example is $e_i = (x_i, y_i)$ where $x_i, y_i \in \mathbb{R}$
- A linear regression task in this case is to learn a linear function $h_{a,b}(x) = a \times x + b$ that approximates best the data distribution

Regression

- Restriction: an example is $e_i = (x_i, y_i)$ where $x_i, y_i \in \mathbb{R}$
- A linear regression task in this case is to learn a linear function $h_{a,b}(x) = a \times x + b$ that approximates best the data distribution
- The linear model is a line on a two dimensional plane

Regression

- Restriction: an example is $e_i = (x_i, y_i)$ where $x_i, y_i \in \mathbb{R}$
- A linear regression task in this case is to learn a linear function $h_{a,b}(x) = a \times x + b$ that approximates best the data distribution
- The linear model is a line on a two dimensional plane
- Let $E = \{(x_1, y1), \dots (x_n, y_n)\}$. The optimisation problem (considered here) is to find a function $h_{a,b} = a \times x + b$ that minimizes the following loss function (Sum of Square Error):

$$Loss_{h_{a,b}} = \sum_i (y_i - h_{a,b}(x_i))^2$$

Linear Regression as an Interpretable Model

Linear Regression as an Interpretable Model

- The objective is to find a, b that minimize

$$Loss_{h_{a,b}} = \sum_{i} (y_i - (a \times x_i + b))^2$$

Linear Regression as an Interpretable Model

- The objective is to find a, b that minimize

$$Loss_{h_{a,b}} = \sum_i (y_i - (a \times x_i + b))^2$$

- This function is minimized when the two partial derivatives of $h_{a,b}$ with respect to a and with respect to b are 0

Linear Regression as an Interpretable Model

- The objective is to find a, b that minimize

$$Loss_{h_{a,b}} = \sum_i (y_i - (a \times x_i + b))^2$$

- This function is minimized when the two partial derivatives of $h_{a,b}$ with respect to a and with respect to b are 0
- There are two unique solutions:

$$a = \frac{n \times (\sum x_i \times y_j) - \sum x_i \times \sum y_i}{n \times \sum x_i^2 - (\sum x_i)^2}$$

$$b = \frac{\sum y_i - a \times \sum x_i}{n}$$

Regression with other hypothesis space

- Assume that the hypothesis space is a non linear function

Regression with other hypothesis space

- Assume that the hypothesis space is a non linear function
- In the general case, the objective function depends on some parameters $p_1, \ldots p_m$.

Regression with other hypothesis space

- Assume that the hypothesis space is a non linear function
- In the general case, the objective function depends on some parameters $p_1, \ldots p_m$. That is, the purpose is to find the parameters that minimize $Loss_{p_1,\ldots p_m}$

Regression with other hypothesis space

- Assume that the hypothesis space is a non linear function
- In the general case, the objective function depends on some parameters $p_1, \dots p_m$. That is, the purpose is to find the parameters that minimize $Loss_{p_1,\dots p_m}$
- A common way to minimize/maximize a function is to look for the points where the gradient (assuming that the function is differentiable) is zero.

Regression with other hypothesis space

- Assume that the hypothesis space is a non linear function
- In the general case, the objective function depends on some parameters $p_1, \dots p_m$. That is, the purpose is to find the parameters that minimize $Loss_{p_1, \dots p_m}$
- A common way to minimize/maximize a function is to look for the points where the gradient (assuming that the function is differentiable) is zero.
- In the general case, it is hard to figure out the closed expression of the gradient (and the function)
- Therefore, computational approaches are proposed as an approximation method to look for the minimum values (and the correspondent parameters). The approach is called the *Gradient descent*.

Regression with other hypothesis space

- Assume that the hypothesis space is a non linear function
- In the general case, the objective function depends on some parameters $p_1, \ldots p_m$. That is, the purpose is to find the parameters that minimize $Loss_{p_1,\ldots p_m}$
- A common way to minimize/maximize a function is to look for the points where the gradient (assuming that the function is differentiable) is zero.
- In the general case, it is hard to figure out the closed expression of the gradient (and the function)
- Therefore, computational approaches are proposed as an approximation method to look for the minimum values (and the correspondent parameters). The approach is called the *Gradient descent*.
- The gradient descent method can be seen as a local a search approach

Local Search

Local Search

- The principle of a local search algorithm is to start somewhere in the search space with a (partial) solution

Local Search

- The principle of a local search algorithm is to start somewhere in the search space with a (partial) solution
- Then in a sequence of steps, replace the current (partial) solution with a better solution in its neighborhood

Local Search

- The principle of a local search algorithm is to start somewhere in the search space with a (partial) solution
- Then in a sequence of steps, replace the current (partial) solution with a better solution in its neighborhood
- The algorithm stops when a certain stopping criterion is met

Local Search

- The principle of a local search algorithm is to start somewhere in the search space with a (partial) solution
- Then in a sequence of steps, replace the current (partial) solution with a better solution in its neighborhood
- The algorithm stops when a certain stopping criterion is met
- Note that local search suffer from local minimums

Local Search

- The principle of a local search algorithm is to start somewhere in the search space with a (partial) solution
- Then in a sequence of steps, replace the current (partial) solution with a better solution in its neighborhood
- The algorithm stops when a certain stopping criterion is met
- Note that local search suffer from local minimums
- A common way to handle this issue is to restart search from time to time

Gradient Descent: A Simple Algorithm

Gradient Descent: A Simple Algorithm

- Gradient Descent is a local search approach

Gradient Descent: A Simple Algorithm

- Gradient Descent is a local search approach
- Start somewhere in the search space (of finding the best parameters) by initializing $p = [p_1, \ldots p_m]$

Gradient Descent: A Simple Algorithm

- Gradient Descent is a local search approach
- Start somewhere in the search space (of finding the best parameters) by initializing $p = [p_1, \ldots p_m]$
- The idea is to move to a point where the gradient is closer to zero

Gradient Descent: A Simple Algorithm

- Gradient Descent is a local search approach
- Start somewhere in the search space (of finding the best parameters) by initializing $p = [p_1, \dots p_m]$
- The idea is to move to a point where the gradient is closer to zero
- Replace p_i by $p_i - \frac{\partial p}{\partial p_i}$ (i.e., the current solution is replaced with a better solution in its neighborhood). This is because $\frac{\partial p}{\partial p_i}$ gives the slope of the function with respect to p_i

Gradient Descent: A Simple Algorithm

- Gradient Descent is a local search approach
- Start somewhere in the search space (of finding the best parameters) by initializing $p = [p_1, \dots p_m]$
- The idea is to move to a point where the gradient is closer to zero
- Replace p_i by $p_i - \frac{\partial p}{\partial p_i}$ (i.e., the current solution is replaced with a better solution in its neighborhood). This is because $\frac{\partial p}{\partial p_i}$ gives the slope of the function with respect to p_i
- Stop until a certain criterion is met
- The slope $\frac{\partial p}{\partial p_i}$ is usually amplified by a regulator α that is called a step size

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 need 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 need 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

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

Explainability

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

Explainability

- Explainaning 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

- Explainaning 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

- Explainaning 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