Introduction to Artificial Intelligence (AI)

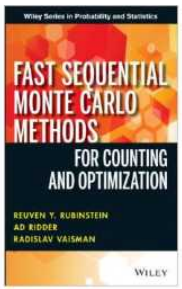
Slava Vaisman

The University of Queensland

r.vaisman@uq.edu.au

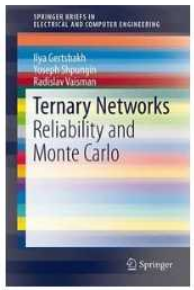
Introduction

- I am a Lecturer at the School of Mathematics and Physics (SMP) at the University of Queensland.
- My research interests lie at the intersection of mathematics and machine learning.
- `https://researchers.uq.edu.au/researcher/11107`



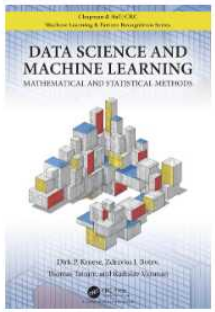
R.Y. Rubinstein, A. Ridder and **R. Vaisman**, *Fast Sequential Monte Carlo Methods for Counting and Optimization*, John Wiley & Sons, 2013.

Order Information:
[Wiley | Amazon]



I. Gertsbakh, Y. Shpungin and **R. Vaisman**, *Ternary Networks: Reliability and Monte Carlo*, Springer Briefs in Electrical and Computer Engineering, 2014.

Order Information:
[Springer | Amazon]



D.P. Kroese, Z.I. Botev, T. Taimre, and **R. Vaisman**. *Data Science and Machine Learning: Mathematical and Statistical Methods*, Chapman and Hall/CRC, 2019

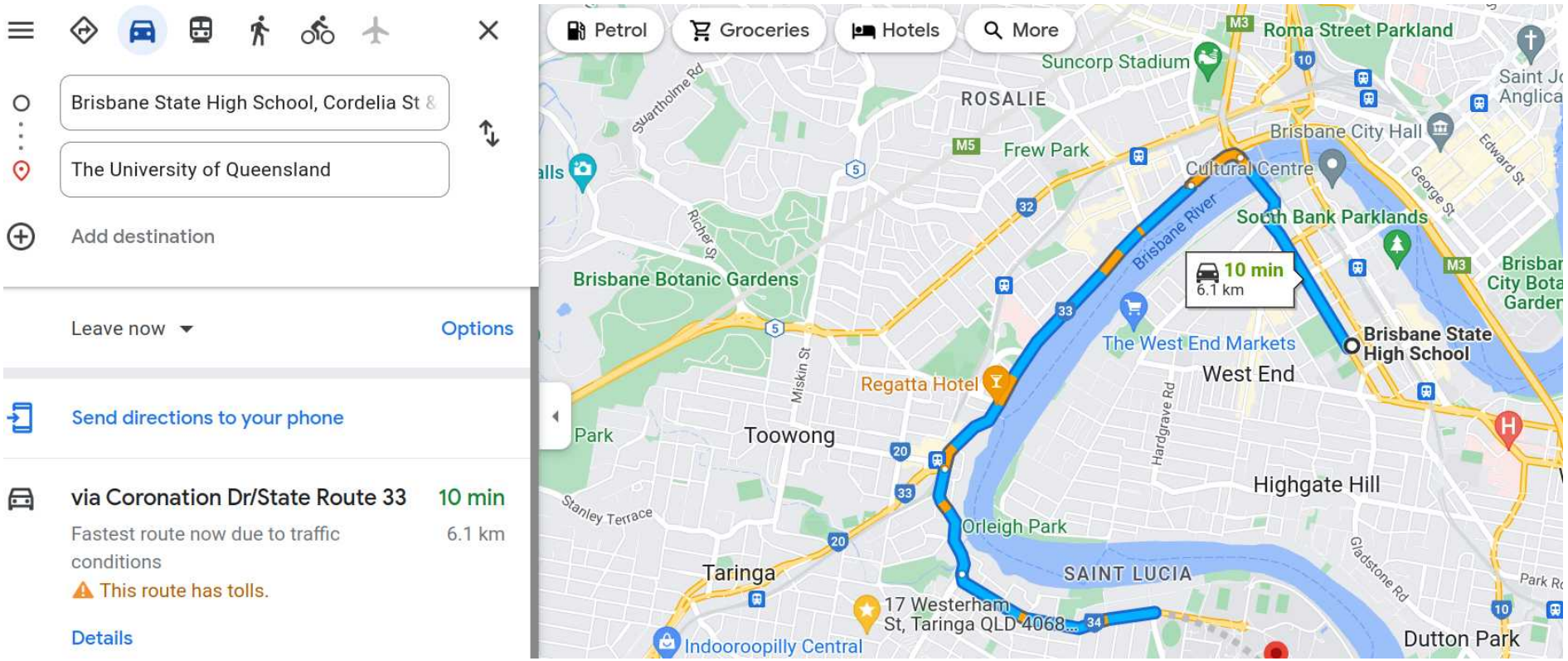
Order Information:
[CRC Press]

Problems we care about (some examples)

1. Calculate shortest path between point A and point B (navigation).
2. Calculate current in electric circuit.
3. Speech and image recognition (many patterns, accents, etc.).
4. Predicting the future (stock/housing prices).
5. Big data processing and pattern recognition.
6. Adaptive computer programs.

An "easy" problem

Calculate shortest path between point A and point B for navigation — can be solved using Dijkstra's Algorithm using a relatively small computation time.



Hard problems

1. Speech and image recognition (many patterns, accents, etc.).
2. Predicting the future (stock/housing prices).
3. Big data processing and pattern recognition.
4. Adaptive computer programs.

For these problems, there is no algorithm that can be simply applied and that can deliver an exact result.

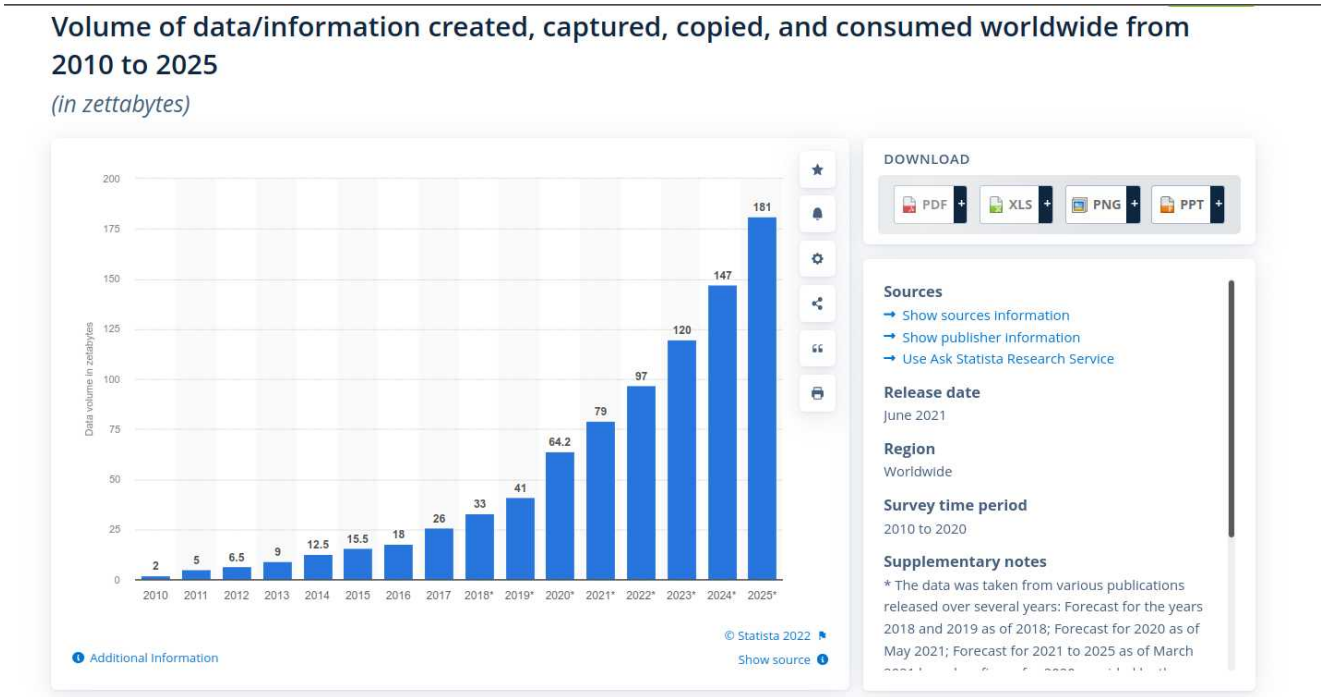
So, when do we need AI?

The problem can be solved by a human, but the corresponding computer program is very complex. Distinguish between a cat and a dog.

So, when do we need AI?

The problem cannot be solved by a human.

- For example, the data that should be processed to derive a meaningful insight is too large for a human to handle.
- Specifically, and especially in today's big-data world, we need to learn to detect meaningful patterns in large and complex data sets. These tasks appear in commerce, search engines, etc. . .



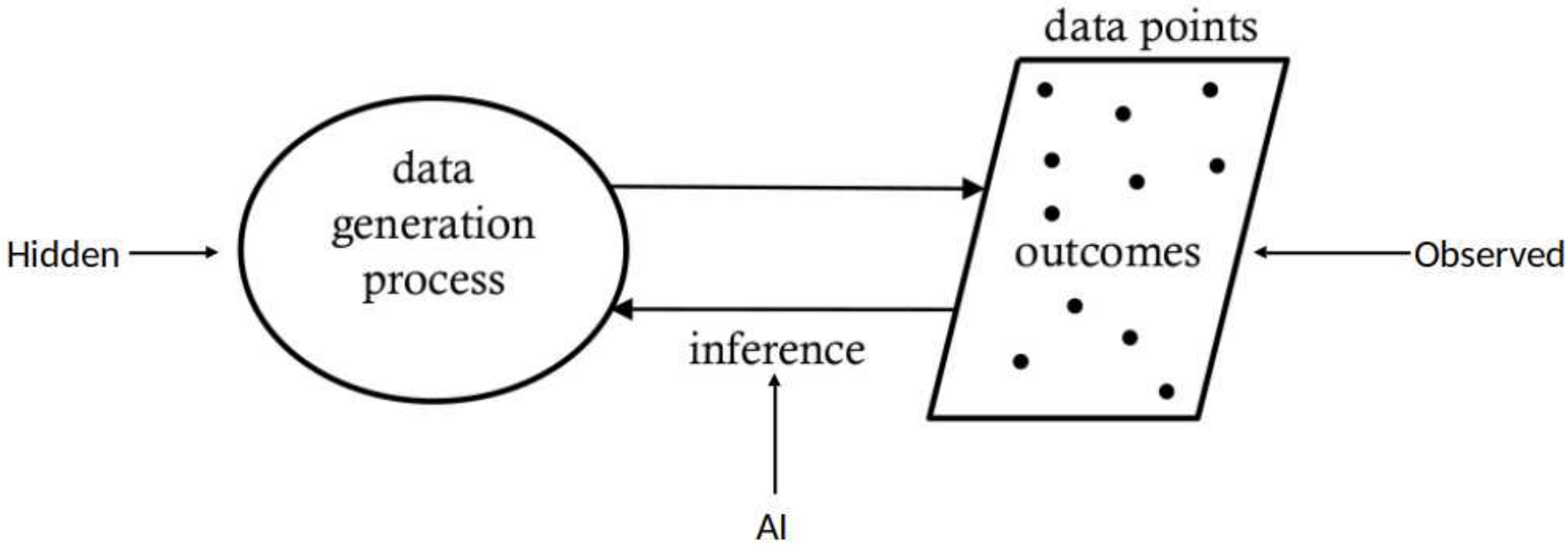
So, when do we need AI?

Program's adaptability.

- A normal computer program does not change its behavior over time.
- However, a learning algorithm should be able to adapt to a new input.
- For example, a voice or hand-writing recognition system in your smart-phone may introduce a better performance as soon as it learns the owner's patterns.

The general setting for AI problems

The AI objective is to develop a (simple) model for the unobserved data generation process based in the data points (observed outcomes).



Today, we are going to develop a general AI algorithm that works really well in many practical situations.

The mnist dataset

In the mnist dataset, each number is represented by $28 \times 28 = 784$ pixels. Given a 28×28 figure, we need to guess the number.

The mnist dataset — the numerical data

```
label pixel0 pixel1 ...  pixel783
2       0      1    ...      1
5       0      1    ...      1
.
.
.
4       1      0    ...      0
```

Our objective is as follows.

- Create a function that receives a 784 dimensional vector $\mathbf{x}$ (here $\mathbf{x}$ represents the pixels), and outputs $y \in \{0, 1, \ldots, 9\}$ — the label.
- Please note that the setting is quite general. That is, many hard problems can be solved using such function f. For example, distinguish between cats and dogs, or, predict a stock market movement.
- Clearly, for every problem, we will require to think about a different function $f(\mathbf{x}) = y$.

Introduction to decision trees

- Statistical learning methods based on decision trees have gained tremendous popularity due to their simplicity, intuitive representation, and predictive accuracy.
- Today, we give an introduction to the usage of such trees.
- We also discuss how a number of trees can be combined to further improve the efficiency of decision trees and other learning methods.
- The main idea is to divide a (potentially complicated) feature space $\mathcal{X}$ into smaller regions and fit a simple prediction function to each region.

Decision trees example (1)

The left panel of Figure 1 shows a training set of 15 two-dimensional points (features) falling into two classes (red and blue). How should the new feature vector (black point) be classified?

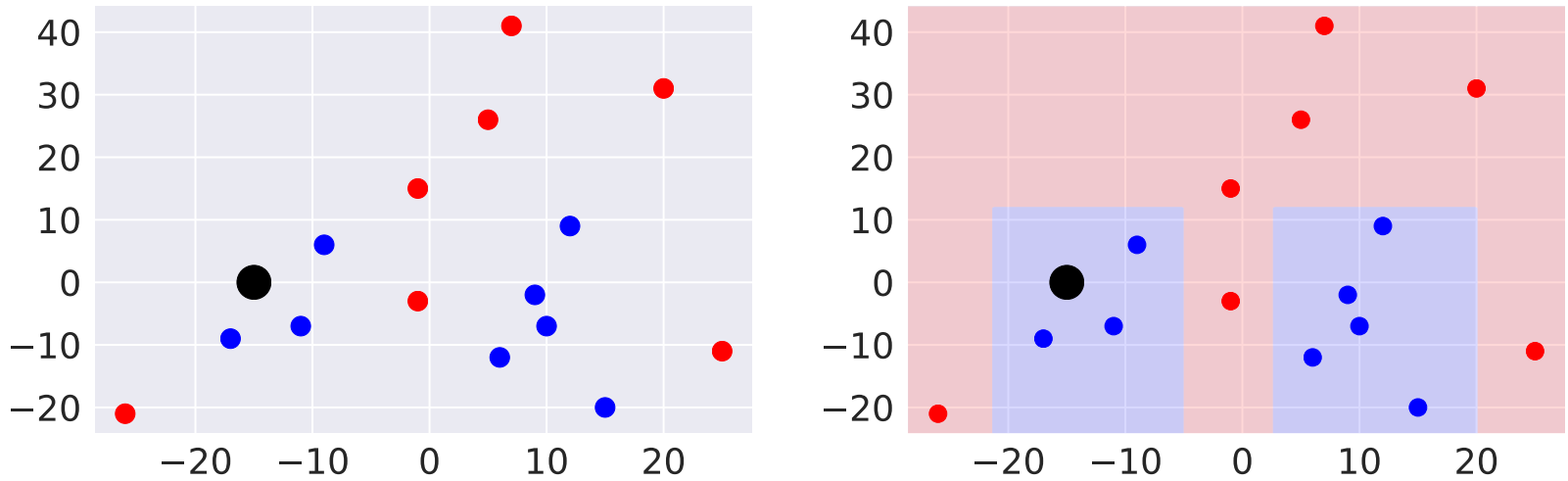
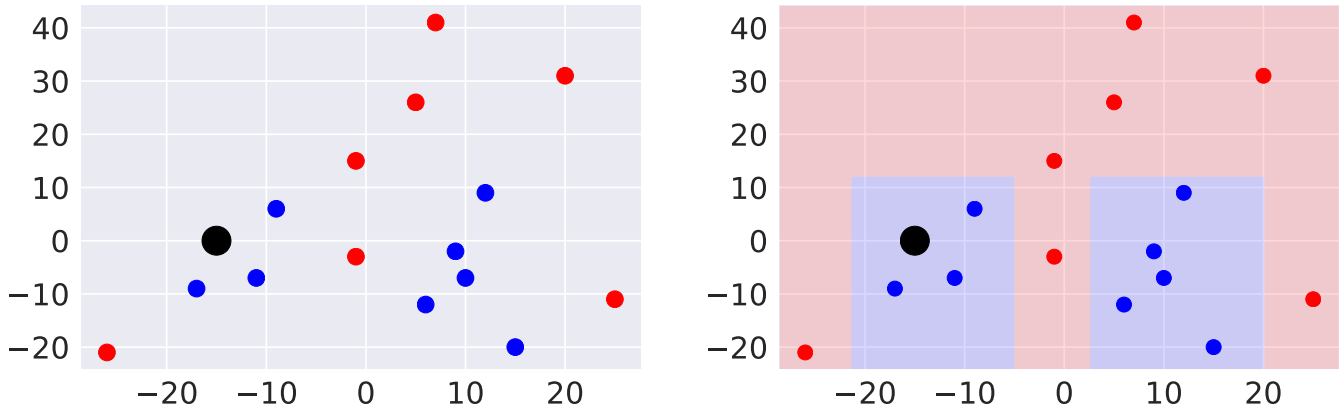


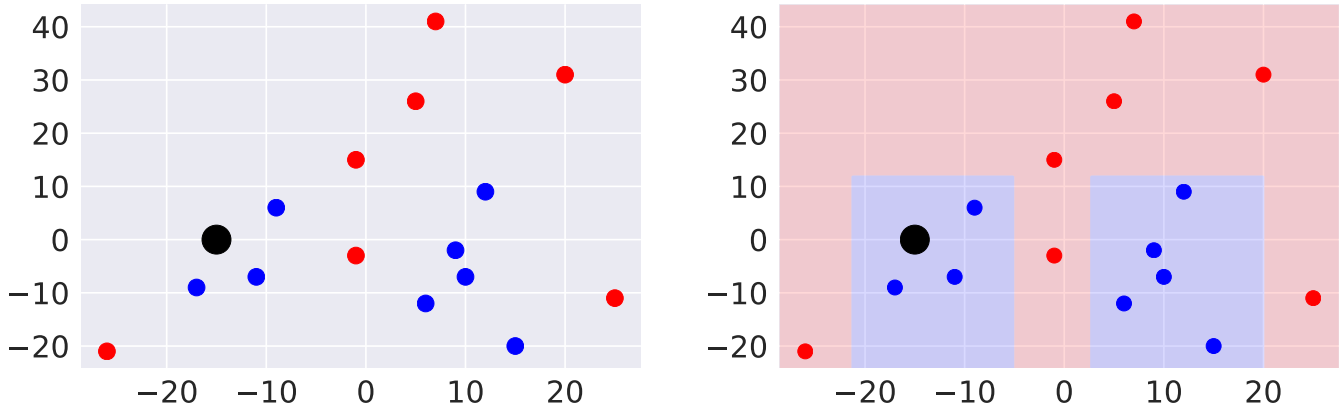
Figure 1: Left: training data and a new feature. Right: a partition of the feature space.

Decision trees example (2)

- It is not possible to linearly separate the training set, but we can partition the feature space $\mathcal{X} = \mathbb{R}^2$ into rectangular regions and assign a class (color) to each region, as shown in the right panel of Figure 1.
- Points in these regions are classified accordingly as blue or red.
- The partition thus defines a classifier (prediction function) g (in this unit we will use g instead of h), that assigns to each feature vector $\mathbf{x}$ a class "red" or "blue".



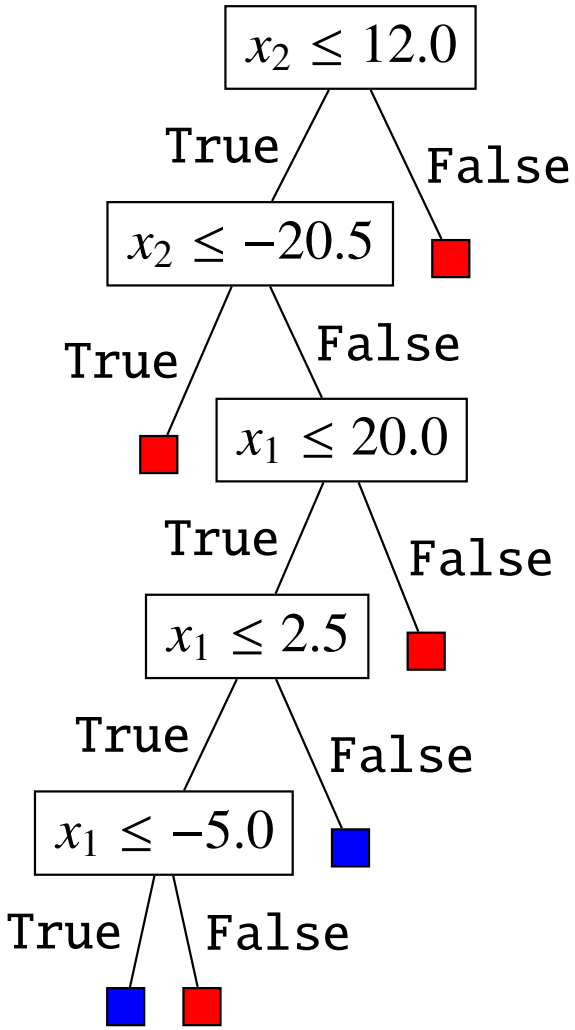
Decision trees example (3)



- For example, for $\mathbf{x} = [-15, 0]^\top$ (solid black point), $g(\mathbf{x}) =$ "blue", since it belongs to a blue region of the feature space.
- Both the classification procedure and the partitioning of the feature space can be conveniently represented by a binary decision tree. This is a tree where each node v corresponds to a region (subset) R_v of the feature space X — the root node corresponding to the feature space itself.

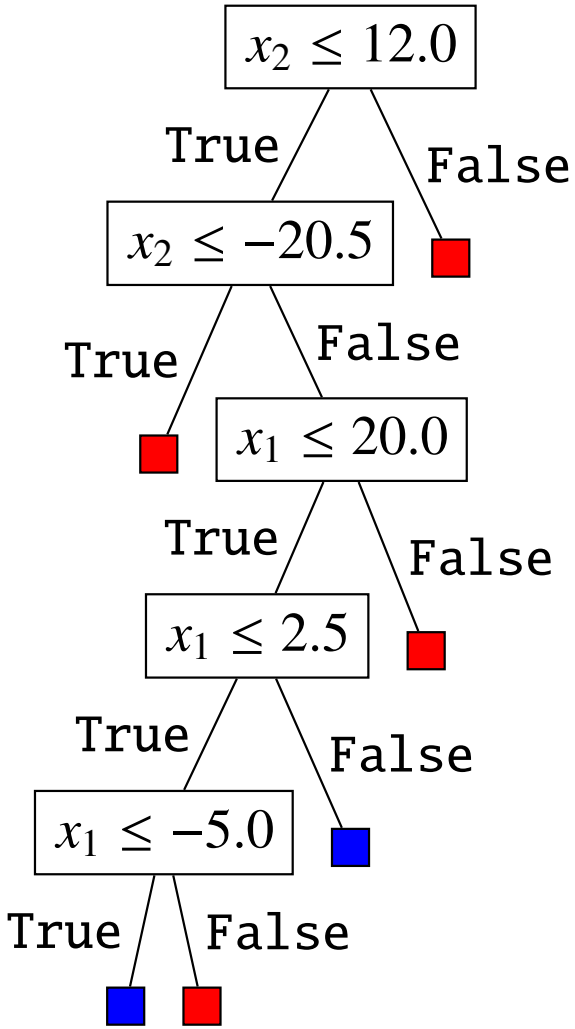
Decision trees example (4)

- Each internal node v contains a logical condition that divides R_v into two disjoint subregions.
- The leaf nodes (the terminal nodes of the tree) are not subdivided, and their corresponding regions form a partition of X, as they are disjoint and their union is X.
- Associated with each leaf node w is also a regional prediction function g^w on R_w.
- The partitioning of the 2D space, was obtained from the decision tree shown on the right.



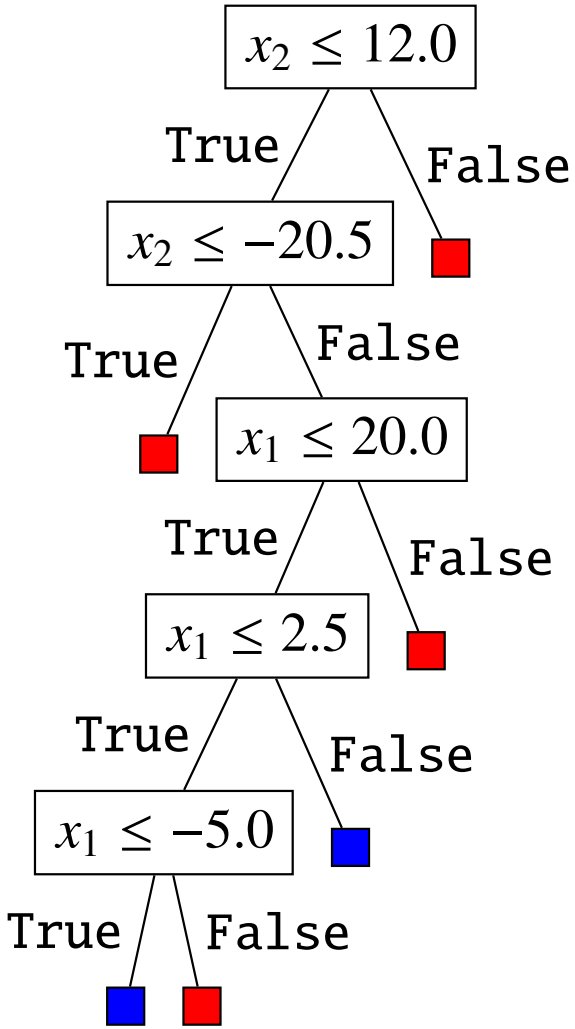
Decision trees example (5)

- As an illustration of the decision procedure, consider again the input $\mathbf{x} = [x_1, x_2]^\top = [-15, 0]^\top$.
- The classification process starts from the tree root, which contains the condition $x_2 \leq 12.0$.
- As the second component of $\mathbf{x}$ is 0, the root condition is satisfied and we proceed to the left child, which contains the condition $x_2 \leq -20.5$.



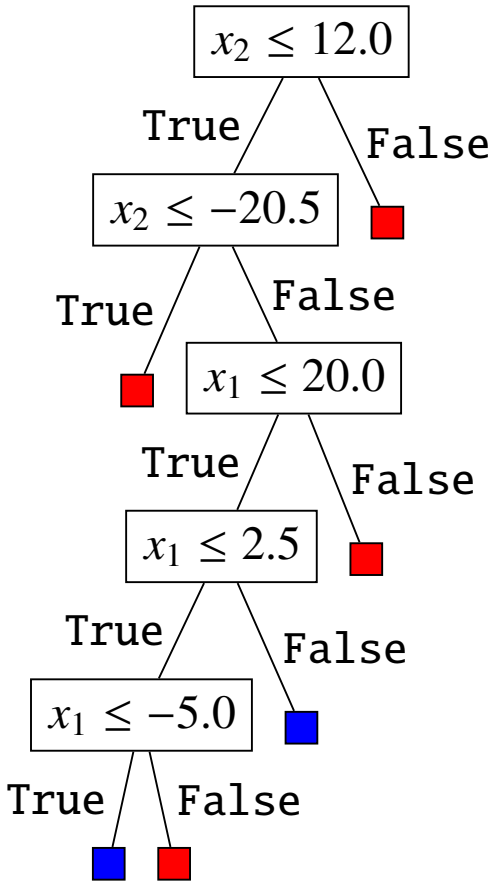
Decision trees example (6)

- The next step is similar. As $0 > -20.5$, the condition is not satisfied and we proceed to the right child.
- Such an evaluation of logical conditions along the tree path will eventually bring us to a leaf node and its associated region.
- In this case the process terminates in a leaf that corresponds to the left blue region in the right-hand panel of the Figure.



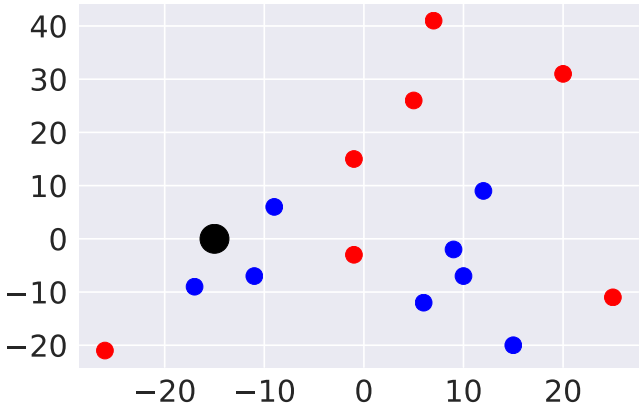
Decision trees example (7)

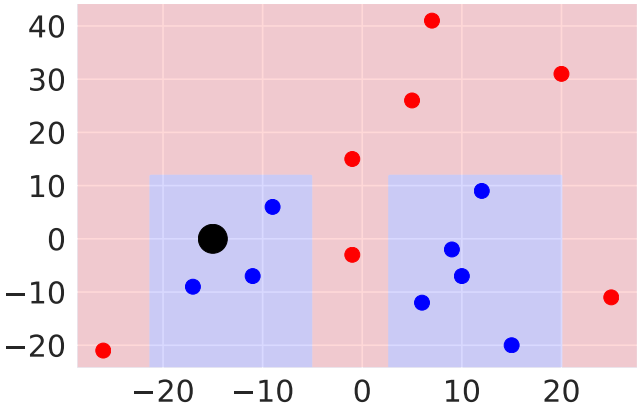
For example, the below Figure partitions the feature space into six rectangles: two blue and four red rectangles.



Decision trees for classification

In a classification setting: the regional prediction function g^w corresponding to a leaf node w takes values in the set of possible class labels.





Constructing a tree

Constructing a tree with a training set $\tau = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ amounts to minimizing the error on the training set. The main idea is as follows.

- The tree root contains all points in the $\tau = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ set.
- Go over all decision rules of the form of the type

$$s(\mathbf{x}) = \mathbb{1}\{x_j \leq \xi\},$$

  where $j \in \{1, \ldots, p\}$ and $\xi \in \mathbb{R}$.
- We can continue in this fashion for each tree node until some termination criteria is satisfied.
- Many times, we stop the tree's growth when the predictive performance of the tree stops to improve. To do so, we need towo datasets:
  1. the train set $\tau_{train} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_1}$, and
  2. the test set $\tau_{test} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_2}$.

Advantages of decision trees

- The tree structure can handle both categorical and numerical features in a natural and straightforward way. Specifically, there is no need to pre-process categorical features, say via the introduction of dummy variables.
- The final tree obtained after the training phase can be compactly stored for the purpose of making predictions for new feature vectors. The prediction process only involves a single tree traversal from the tree root to a leaf.
- The tree structure can be easily understood and interpreted by domain experts with little statistical knowledge, since it is essentially a logical decision flow diagram.

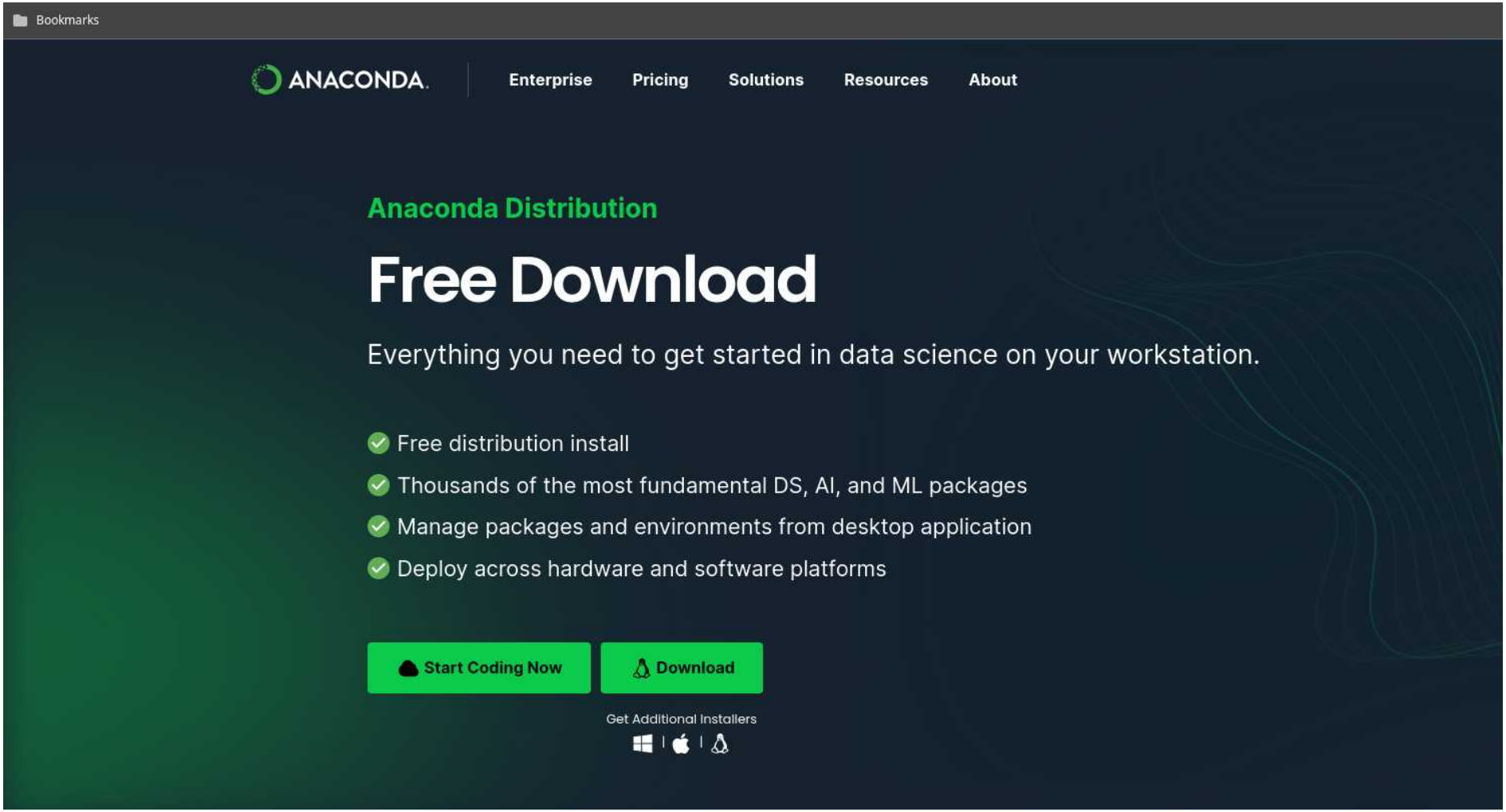
Limitations of decision trees

- Despite the fact that the decision trees are extremely interpretable, the predictive accuracy is generally inferior to other established statistical learning methods.
- In addition, decision trees, and in particular very deep trees that were not subject to pruning, are heavily reliant on their training set.
- A small change in the training set can result in a dramatic change of the resulting decision tree.
- To overcome the above limitations, several promising approaches exist. Today, we consider one of the simplest and most powerful methods called a *random forest*.

Random Forests

- The major idea of the *random forest* method is to combine prediction functions learned from multiple data sets, with a view to improving overall prediction accuracy.
- Random Forests are especially beneficial when dealing with predictors that tend to overfit the data, such as in decision trees, where the (unpruned) tree structure is very sensitive to small changes in the training set.
- Consider an idealized setting for a decision tree, where we have access to B iid copies $\mathcal{T}_1, \ldots, \mathcal{T}_B$ of a training set $\mathcal{T}$.
- Then, we can train B different decision trees, giving learners $g_{\mathcal{T}_1}, \ldots, g_{\mathcal{T}_B}$, and deliver g_{rf} the majority vote among $\{g_{\mathcal{T}_b^*}\}, b = 1, \ldots, B$; that is, to accept the most frequent class among B predictors.

Python


Bookmarks
ANACONDA.
Enterprise
Pricing
Solutions
Resources
About
Anaconda Distribution
Free Download
Everything you need to get started in data science on your workstation.
Free distribution install
Thousands of the most fundamental DS, AI, and ML packages
Manage packages and environments from desktop application
Deploy across hardware and software platforms
Start Coding Now
Download
Get Additional Installers