P02 Decision Trees

Luis Sa-Couto[1] and Andreas Wichert[2]

INESC-ID, Instituto Superior Tecnico, Universidade de Lisboa
{luis.sa.couto,andreas.wichert}@tecnico.ulisboa.pt

1 Decision Trees

A decision tree represents a data table where each data point has features and a class. A node in the tree represents a test on a feature that partitions the original table in one table for each possible result of that test (see figure 1).

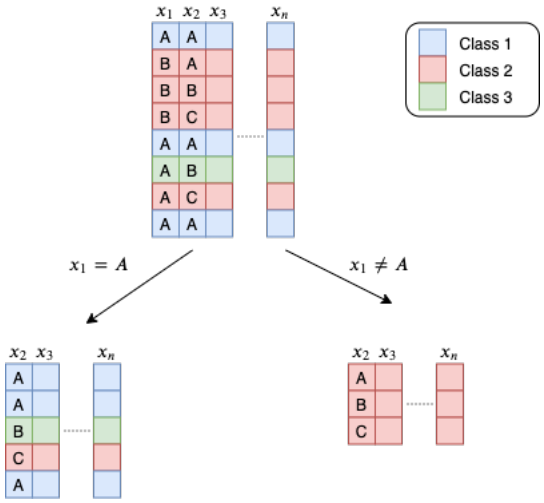


Fig. 1. A test node that creates two partitions.

The tree is built by inserting these test nodes until we reach all leafs. Leafs are nodes where the table partition has either all elements of the same class or no more tests can be applied, for instance the rightmost node in the aforementioned figure. Once the full tree is built, to perform classification of a given point, we traverse the tree by applying each node's test until a leaf is reached (see figure 2).

The size of the tree depends on the order we choose for the tests. Different orders will result in different decision trees, all of which can be correct. So, how do we decide which one we want? On the one hand, we want a tree that correctly captures the information present in the data table (i.e. classifies correctly its

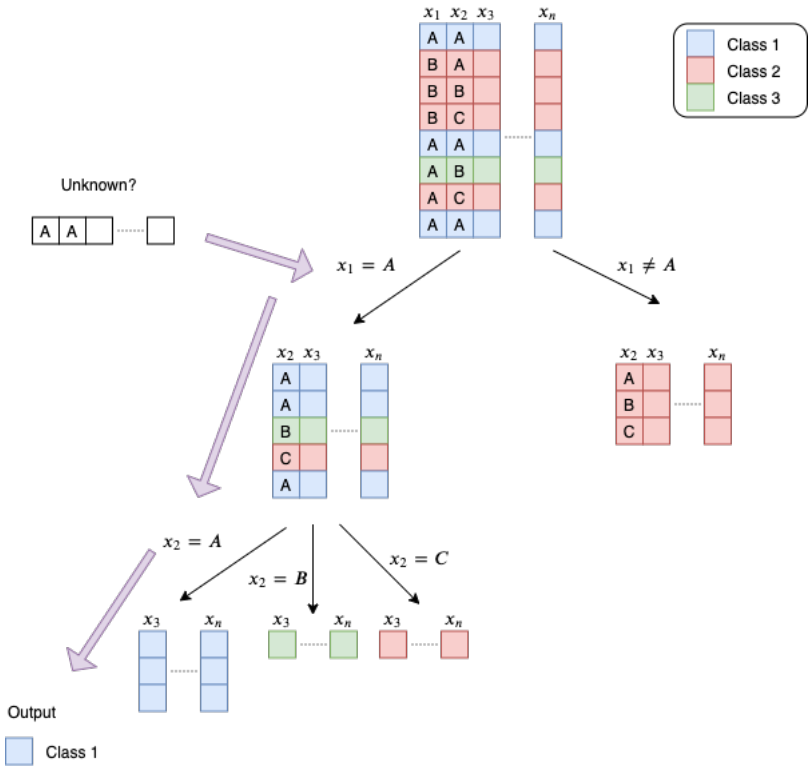


Fig. 2. Classifying a point with a decision tree.

instances). On the other hand, we want a tree that classifies unseen examples correctly too. In that sense, a simple tree is preferrable because it is less likely to include unnecessary constraints that only appear true in this training set but are not true in general.

We could test all possible orders and choose the best tree but it grows extremely fast ($\Theta(n!)$). Since blind search won't do, we need a heuristic. A common procedure known as ID3 uses entropy as the heuristic.

Entropy is a function that receives a distribution and returns a value:

$$E(p_1, p_2, p_3, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

The higher the value the less deterministic the distribution is. More specifically, a high entropy tells us that the outcome of sampling that distribution is very hard to predict. Entropy of a given table is given by the entropy in the distribution of classes. How much entropy we need to get rid of to have perfect classification.

In ID3, a test node is a test on a specific feature, where one partition is created for each possible value. When building a tree, at a given moment, the entropy of that tree is the weighted average of the entropies of all partitions (see figure 3).

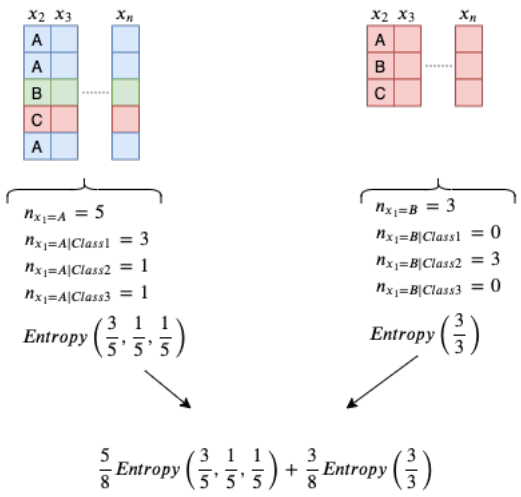


Fig. 3. Computing entropy from multiple partitions.

So, every time we want to choose the next attribute to test, we do:

- Compute the original entropy of the tree
- For each possible feature test we can add to the tree

  - compute the partition tables that are created each possible feature assignment
  - compute the total entropy of each partition
  - compute the new entropy by taking the weighted average of entropies of all partitions
  - compute the test's information gain by computing the difference between the original entropy and the new entropy
- Choose the feature that provides the highest information gain and add it to the tree

An alternative method we will see is CART where all tests have binary outputs. Concretely, instead of testing a feature and creating a branch for all its possible values, we test a feature value assignment $x_i = a$ and create two branches:

- A branch for instances where $x_i == a$;
- A branch for all reamining instances.

1) Consider the following data table:

F_1	F_2	F_3	O
a	a	a	$+$
c	b	c	$+$
c	a	c	$+$
b	a	a	$-$
a	b	c	$-$
b	b	c	$-$

a) Determine the whole decision tree using ID3 (**information gain**), taking "O" as the target. Show all steps.

Solution:

Before anything, we need to compute the entropy we start with:

$$E_{start} = E\left(\frac{3}{3+3}, \frac{3}{3+3}\right) = 1bit$$

Let us test attribute F_1:

$$\begin{array}{ccc} F_1 = a & F_1 = b & F_1 = c \\ \downarrow & \downarrow & \downarrow \\ \#\{O=+\} = 1 & \#\{O=+\} = 0 & \#\{O=+\} = 2 \\ \#\{O=-\} = 1 & \#\{O=-\} = 2 & \#\{O=-\} = 0 \\ \downarrow & \downarrow & \downarrow \\ E\left(\frac{1}{1+1}, \frac{1}{1+1}\right) & E\left(\frac{2}{2+0}\right) & E\left(\frac{2}{2+0}\right) \end{array}$$

Total entropy after partitioning by F_1 is equal to the weighted average of each partition's entropy: $E_{F_1} = \frac{2}{6}E\left(\frac{1}{1+1}, \frac{1}{1+1}\right) + \frac{2}{6}E\left(\frac{2}{2+0}\right) + \frac{2}{6}E\left(\frac{2}{2+0}\right) \approx 0.33bit$.

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_1: $G(F_1) = E_{start} - E_{F_1} \approx 0.66bit$.

Let us test the next attribute, namely F_2:

$$\begin{array}{cc} F_2 = a & F_2 = b \\ \downarrow & \downarrow \\ \#\{O=+\} = 2 & \#\{O=+\} = 1 \\ \#\{O=-\} = 1 & \#\{O=-\} = 2 \\ \downarrow & \downarrow \\ E\left(\frac{2}{2+1}, \frac{1}{2+1}\right) & E\left(\frac{1}{1+2}, \frac{2}{1+2}\right) \end{array}$$

Total entropy after partitioning by F_2 is equal to the weighted average of each partition's entropy: $E_{F_2} = \frac{3}{6}E\left(\frac{2}{2+1}, \frac{1}{2+1}\right) + \frac{3}{6}E\left(\frac{1}{1+2}, \frac{2}{1+2}\right) = 0.9183bit$.

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_2: $G(F_2) = E_{start} - E_{F_2} = 0.0817bit$

For F_3:

$$\begin{array}{cc} F_3 = a & F_3 = c \\ \downarrow & \downarrow \\ \#\{O=+\} = 1 & \#\{O=+\} = 2 \\ \#\{O=-\} = 1 & \#\{O=-\} = 2 \\ \downarrow & \downarrow \\ E\left(\frac{1}{1+1}, \frac{1}{1+1}\right) & E\left(\frac{2}{2+2}, \frac{2}{2+2}\right) \end{array}$$

Total entropy after partitioning by F_3 is equal to the weighted average of each partition's entropy: $E_{F_3} = \frac{2}{6}E\left(\frac{1}{1+1}, \frac{1}{1+1}\right) + \frac{4}{6}E\left(\frac{2}{2+2}, \frac{2}{2+2}\right) = 1.0bit$.

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_3: $G(F_3) = E_{start} - E_{F_3} = 0bit$

Since F_1 provides the highest gain, we use it as the first node in the tree and get this break down of the dataset:

$F_1 = a$			$F_1 = b$			$F_1 = c$		
F_2	F_3	O	F_2	F_3	O	F_2	F_3	O
a	a	$+$	a	a	$-$	b	c	$+$
b	c	$-$	b	c	$-$	a	c	$+$
			Done!			*Done*!		

The first partition ($F_1 = a$) still has uncertainty. For that reason, we will repeat the same process to decide the next attribute to test. Before anything, we need to compute the entropy we start with:

$$E_{start} = E\left(\frac{1}{1+1}, \frac{1}{1+1}\right) = 1bit$$

Let us test the next attribute, namely F_2:

$$\begin{array}{cc} F_2 = a & F_2 = b \\ \downarrow & \downarrow \\ \#\{O=+\}=1 & \#\{O=+\}=0 \\ \#\{O=-\}=0 & \#\{O=-\}=1 \\ \downarrow & \downarrow \\ E\left(\frac{1}{1+0}\right) & E\left(\frac{1}{0+1}\right) \end{array}$$

Total entropy after partitioning by F_2 is equal to the weighted average of each partition's entropy: $E_{F_2} = \frac{1}{2}E\left(\frac{1}{1+0}\right) + \frac{1}{2}E\left(\frac{1}{0+1}\right) = 0bit.$

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_2: $G(F_2) = E_{start} - E_{F_2} = 1bit$

For F_3:

$$\begin{array}{cc} F_3 = a & F_3 = c \\ \downarrow & \downarrow \\ \#\{O=+\}=1 & \#\{O=+\}=0 \\ \#\{O=-\}=0 & \#\{O=-\}=1 \\ \downarrow & \downarrow \\ E\left(\frac{1}{1+0}\right) & E\left(\frac{1}{0+1}\right) \end{array}$$

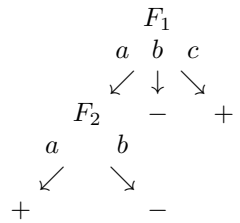
Total entropy after partitioning by F_3 is equal to the weighted average of each partition's entropy: $E_{F_3} = \frac{1}{2}E\left(\frac{1}{1+0}\right) + \frac{1}{2}E\left(\frac{1}{0+1}\right) = 0.0bit.$

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_3: $G(F_3) = E_{start} - E_{F_3} = 1bit$

Since both attributes have the same gain. we can choose either. Let us go with F_2 and get the following partioning:

$F_1 = a$				$F_1 = b$			$F_1 = c$		
$F_2 = a$		$F_2 = b$		F_2	F_3	O	F_2	F_3	O
F_3	O	F_3	O	a	a	$-$	b	c	$+$
a	$+$	c	$-$	b	c	$-$	a	c	$+$
Done!		*Done*!		*Done*!			*Done*!		

Since there is no more uncertainty, we needn't explore any further. Thus, we reach the final tree:



2) Consider the following data table:

F_1	F_2	F_3	O
d	a	b	m
c	a	b	n
c	a	a	y
d	a	a	y
c	b	a	f
c	b	b	f

a) Compute the first attribute to be tested using ID3.

Solution:

Before anything, we need to compute the entropy we start with:

$$E_{start} = E\left(\frac{\#\{O=m\}}{\sum_o \#\{O=o\}}, \frac{\#\{O=n\}}{\sum_o \#\{O=o\}}, \frac{\#\{O=y\}}{\sum_o \#\{O=o\}}, \frac{\#\{O=f\}}{\sum_o \#\{O=o\}}\right) =$$

$$= E\left(\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3}\right) = 1.9183bit$$

Let us test attribute F_1:

$$\begin{array}{cc} F_1 = d & F_1 = c \\ \downarrow & \downarrow \\ \#\{O=m\}=1 & \#\{O=m\}=0 \\ \#\{O=n\}=0 & \#\{O=n\}=1 \\ \#\{O=y\}=1 & \#\{O=y\}=1 \\ \#\{O=f\}=0 & \#\{O=f\}=2 \\ \downarrow & \downarrow \\ E\left(\frac{1}{1+1}, \frac{1}{1+1}\right) & E\left(\frac{1}{1+1+2}, \frac{1}{1+1+2}, \frac{2}{1+1+2}\right) \end{array}$$

Total entropy after partitioning by F_1 is equal to the weighted average of each partition's entropy: $E_{F_1} = \frac{2}{6}E\left(\frac{1}{1+1}, \frac{1}{1+1}\right) + \frac{4}{6}E\left(\frac{1}{1+1+2}, \frac{1}{1+1+2}, \frac{2}{1+1+2}\right) \approx 1.3333bit$.

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_1: $G(F_1) = E_{start} - E_{F_1} \approx 0.5850bit$.

Let us test the next attribute, namely F_2:

$$\begin{array}{cc} F_2 = a & F_2 = b \\ \downarrow & \downarrow \\ \#\{O=m\}=1 & \#\{O=m\}=0 \\ \#\{O=n\}=1 & \#\{O=n\}=0 \\ \#\{O=y\}=2 & \#\{O=y\}=0 \\ \#\{O=f\}=0 & \#\{O=f\}=2 \\ \downarrow & \downarrow \\ E\left(\frac{1}{1+1+2}, \frac{1}{1+1+2}, \frac{2}{1+1+2}\right) & E\left(\frac{2}{2}\right) \end{array}$$

Total entropy after partitioning by F_2 is equal to the weighted average of each partition's entropy: $E_{F_2} = \frac{4}{6}E\left(\frac{1}{1+1+2}, \frac{1}{1+1+2}, \frac{2}{1+1+2}\right) + \frac{2}{6}E\left(\frac{2}{2}\right) \approx 1.0bit$.

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_2: $G(F_2) = E_{start} - E_{F_2} \approx 0.9183bit$

For F_3:

$$\begin{array}{cc} F_3 = b & F_3 = a \\ \downarrow & \downarrow \\ \#\{O = m\} = 1 & \#\{O = m\} = 0 \\ \#\{O = n\} = 1 & \#\{O = n\} = 0 \\ \#\{O = y\} = 0 & \#\{O = y\} = 2 \\ \#\{O = f\} = 1 & \#\{O = f\} = 1 \\ \downarrow & \downarrow \\ E\left(\frac{1}{1+1+1}, \frac{1}{1+1+1}, \frac{1}{1+1+1}\right) & E\left(\frac{2}{2+1}, \frac{1}{2+1}\right) \end{array}$$

Total entropy after partitioning by F_3 is equal to the weighted average of each partition's entropy: $E_{F_3} = \frac{3}{6}E\left(\frac{1}{1+1+1}, \frac{1}{1+1+1}, \frac{1}{1+1+1}\right) + \frac{3}{6}E\left(\frac{2}{2+1}, \frac{1}{2+1}\right) \approx 1.2516bit$.

So, the information gain is given by subtracting the entropy at begining from the remaining entropy after F_3: $G(F_3) = E_{start} - E_{F_3} \approx 0.6667bit$

Since F_2 provides the highest gain, we would use it as the first node in the tree.

---

b) Complete the tree started in the previous question. There is no need to perform all computations. What do you need to take into account?

Solution:

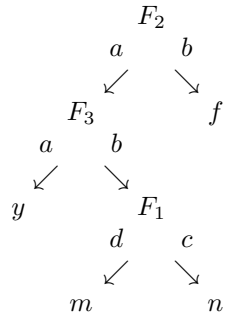
From last question we got that the root node should be a test on F_2, this yields the following partition:

$F_2 = a$			$F_2 = b$		
F_1	F_3	O	F_1	F_3	O
d	b	m	c	a	f
c	b	n	c	b	f
c	a	y	*Done*!		
d	a	y			

Analyzing the second table, we see that no more uncertainty exists, so that branch is completed. The first table still provides uncertainty, so we would have to decide between testing F_1 or F_3. Testing F_1 would give no advantage since both branche would end up undecided. Whereas testing F_3 allows us to separate the y's from the rest. So, we would get the follwoing partition:

$F_2 = a$				$F_2 = b$		
$F_3 = b$		$F_3 = a$		F_1	F_3	O
F_1	O	F_1	O	c	a	f
d	m	c	f	c	b	f
c	n	d	f	$Done!$		
		$Done!$				

With this test, two branches appear. On the one hand, for $F_3 = a$, uncertainty disappears, so it is done. On the other hand, for $F_3 = a$, we require a final test on F_1 to get rid of the final bit of uncertainty. With that test, we get the final tree:



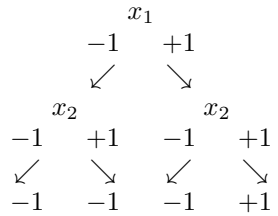
---

3) Show if a decision tree can learn the following logical functions and if so plot the corresponding decision boundaries.

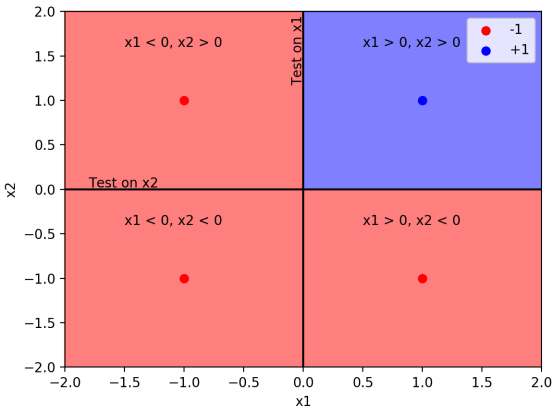
a) AND

Solution:

To show it is possible we only need to create a decision tree that solves the problem. The following does:



The corresponding decision boundaries are shown below.

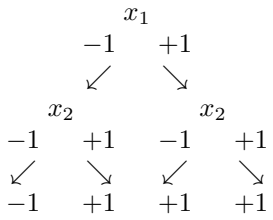


---

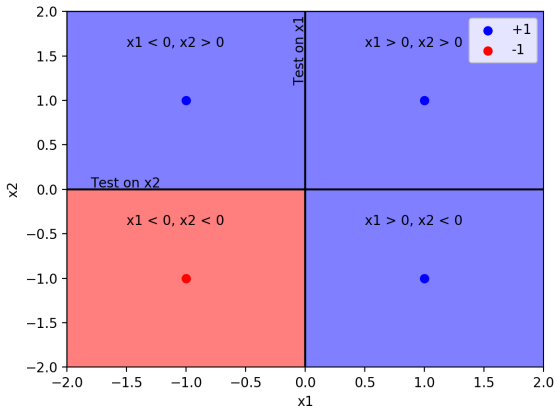
b) *OR*

Solution:

To show it is possible we only need to create a decision tree that solves the problem. The following does:



The corresponding decision boundaries are shown below.

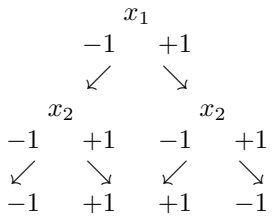


---

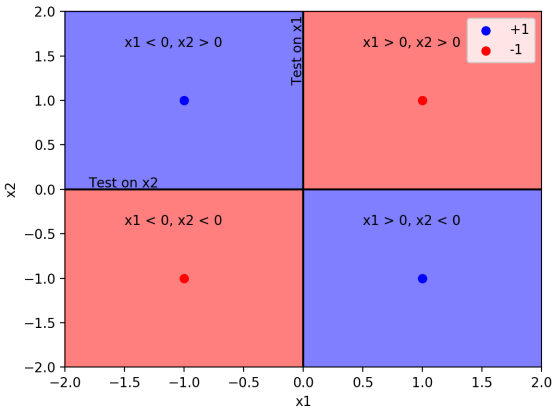
c) XOR

Solution:

To show it is possible we only need to create a decision tree that solves the problem. The following does:



The corresponding decision boundaries are shown below.



---

2 Thinking Questions

Can you think why we want a good heuristic when building the tree? Why are usually smaller trees preferred?