Chapter 2

Decision Trees

(following Chapter 3 of Mitchell)

Definition

Learning in decision trees (DT) is a method of approximating discrete-value target function represented in a decision tree.

The learned trees may also be represented by a set of if-then rules to improve human readability and interpretability.

These learning methods are among the most used and have been applied to a large number of fields (since medial diagnosis to credit risk management in banking).

Representation of concepts in DT

Concept: “good day to play tennis”

Meteorology

Outlook (Céu):
- Sunny (Soalheiro) ,
- Overcast (Nublado),
- Rain (Chuva).

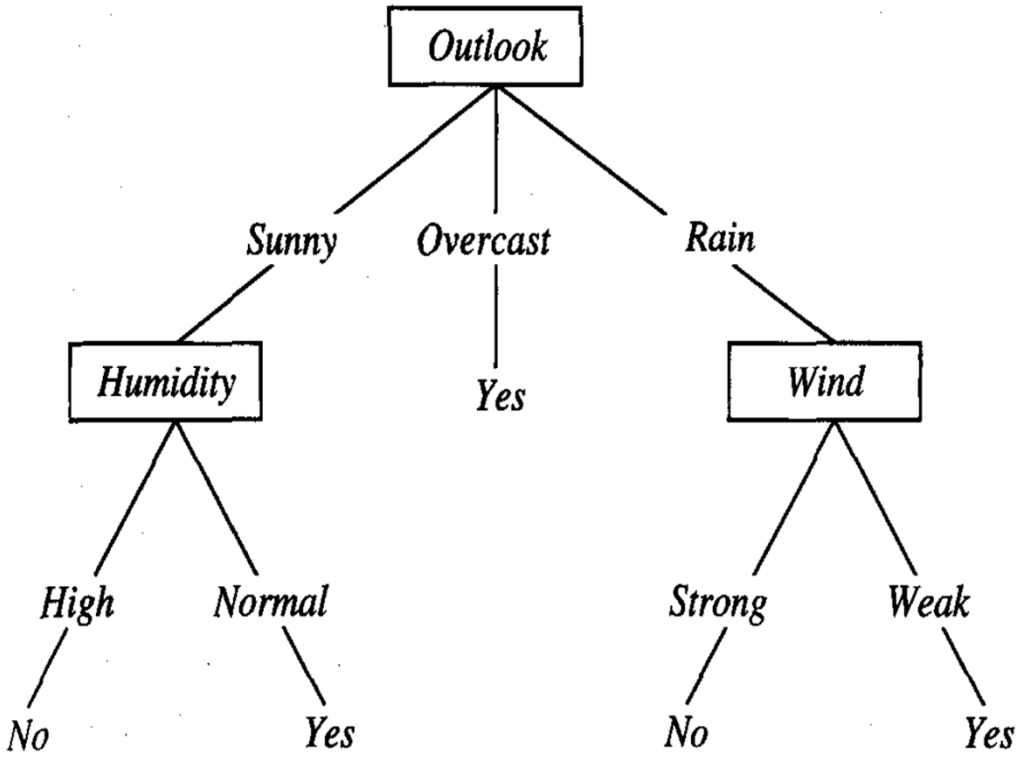
Humidity (Humidade):
- High (Alta),
- Normal.

Wind (Vento):
- Strong (Forte),
- Weak (Fraco).

Temperature :
- Hot (Quente),
- Mild (Amena),
- Cool (Fria).

Representation of concepts in DT

Concept: “good day to play tennis”



Root

Branches

Nodes

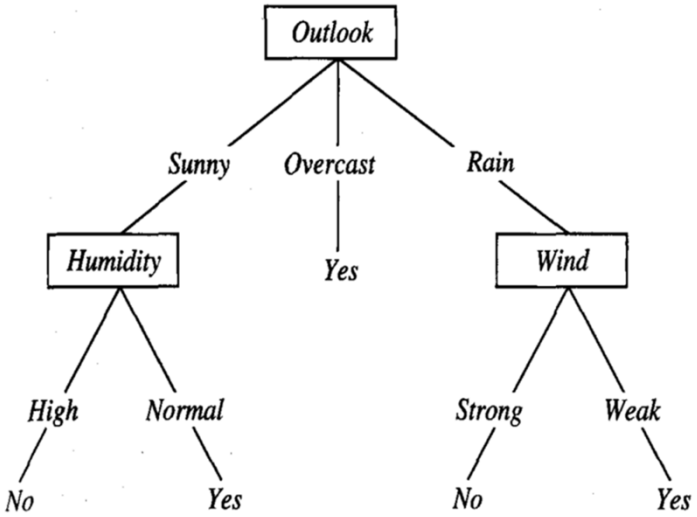
Leafs

Binary decision function: Yes / No

(Mitchell)

Representation of concepts in DT

The decision tree classifies instances of the problem by descending the tree from the root to some leaf that gives the classification of the instance.



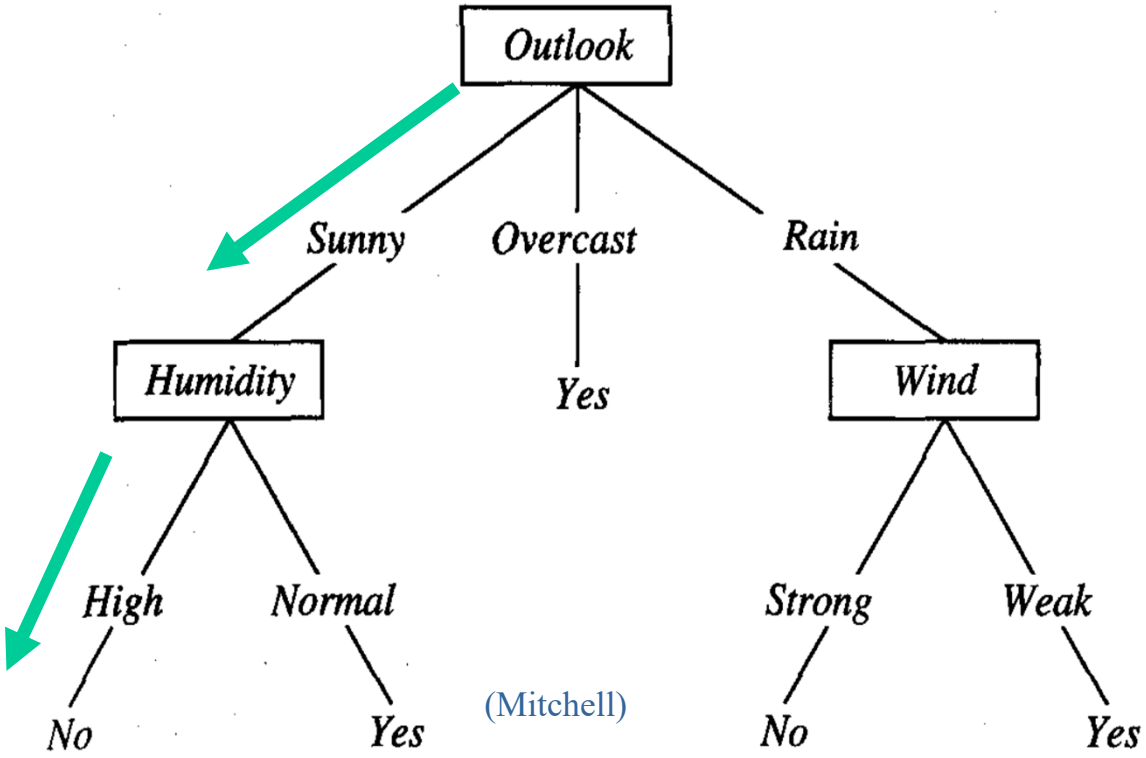
Each node executes a test over some attribute of the instance, and each descending branch from that node corresponds to one of the possible values of that attribute.

The classification process of an instance starts at the root, testing the attribute specified by this node, descending the branch of the tree corresponding to the value of the attribute in the example, and going to the extreme node of this branch, where the test of its attribute is made, and the process continues until a leaf of the tree is reached.

Representation of concepts in DT

Given the instance

<Outlook=Sunny, Temperature=Hot, Humidity=High, Wind=Strong>



(Mitchell)

Remark that the instance has the attribute temperature that does not exist in the tree, but this is no problem, since with the other attributes it is possible to classify the instance, i.e., to make a decision. This is common in DT: **it can be possible to classify one instance only with a part of its attributes**.

Representation of concepts in DT

A DT represents a disjunction of conjunctions of constrains over the values of the attributes of one instance: each path from the root until one leaf represents a conjunction of test attributes, and the tree itself is a disjunction of these conjunctions.

IF (Outlook=Sunny $\wedge$ Humidity=Normal)

$\vee$ (Outlook=Overcast)

$\vee$ (Outlook=Rain $\wedge$ Wind=Weak)

THEN

Yes

Definition of the concept "PlayTennis"

Problems appropriated for decision trees

- *The instances are represented by attribute-value pairs; there is a finite number of attributes with a finite number of values.*
- *The target function has discrete values in its output.*
- *Disjunctive descriptions may be necessary*
- *The training data may contain errors*
- *The training data may contain missing attribute values*

Classify examples in one of a finite number of possible categories:

→ *classification problems*

The basic algorithm for classification in DTs

ID3 (**Iterative Dichotomiser 3**): Quinlan 1986

a top-down search greedy technique, through the space of all possible decision trees.

The initial question:

which attribute should be tested in the root ?

A statistical test is used to determine how well each attribute, by its own, classifies the training example, and we chose the one that best classifies it, the one that is more “succulent”, and this is because the algorithm is called “greedy”.

The basic algorithm for classification in DTs

Witch of the attributes is the best classifier (the most succulent)?

How to define a quantitative measure of the discriminant capability of an attribute ?

Or, by other way, which is the informative capability of an attribute to separate the training examples in accordance with the classification target ?

This is to say, which is the *gain of information* that an attribute can provide if we chose it ??

ID3 uses this *information gain* to select the candidate attribute in each step as the tree grows.

The tree is built using a subset of examples that are part of a larger set of all possible instances.

The general classification problem :

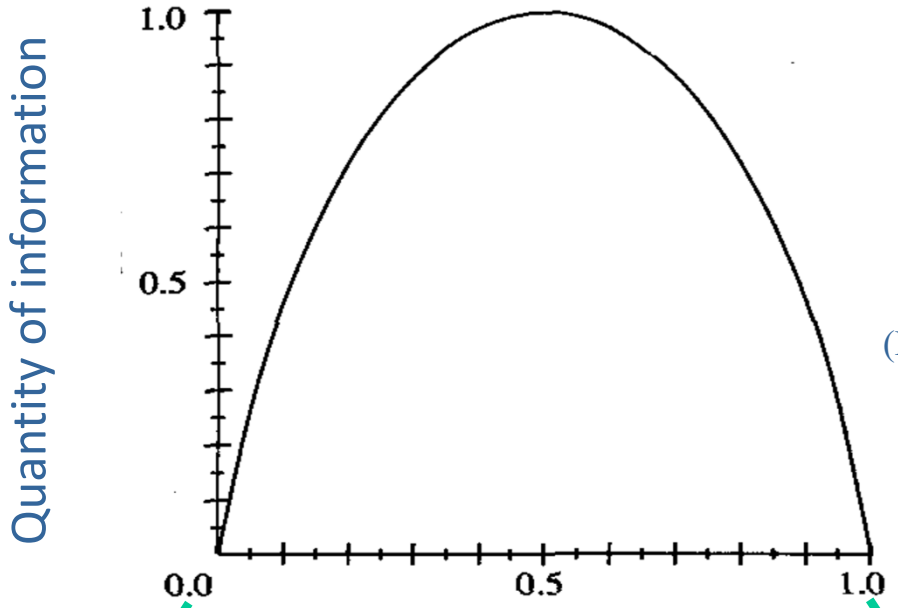
The subsets of testing and training must be statistically representative of the total set.

- If all training examples give positive (Yes), it is not possible to classify, there is not enough information, there is too homogeneity in the data.

- If all training examples give negative (No), the situation is similar.

- If there are as much positives as negatives, the information is high, there is low data homogeneity, there is high data heterogeneity; this is the best situation.

We need a function of the type:



Given a collection S , containing positive and negative examples of a two values target-concept, the Entropy of S relative to that Boolean classification is defined by

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\odot} \log_2 p_{\odot}$$

where

$p_{\oplus}$ is the proportion of positive examples in S

$p_{\odot}$ is the proportion of negative examples in S

In all calculations it is considered that $0 \log 0 = 0$

The Entropy classifies the degree of homogeneity of the collection S.

For a collection *S* with 14 examples of a boolean concept, being 9 positives and 5 negatives (notation 9+ and 5-), the entropy S relative to the boolean classification will be :

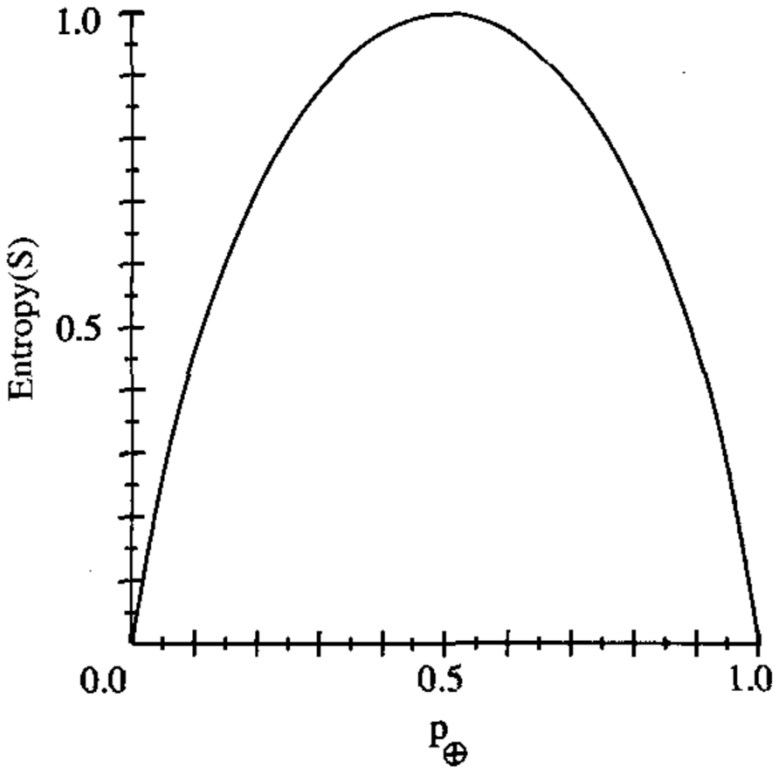
$$Entropy[9+,5-] = \\ = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0,940$$

and

$$Entropy[14+,0-] = \\ = -(14/14)\log_2(14/14) - (0/14)\log_2(0/14) = \\ = -1\times 0 - 0\log_2 0 = 0$$

and

$$Entropy[7+,7-] = -(7/14)\log_2(7/14) - (7/14)\log_2(7/14) = \\ = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = \\ = -\log_2(1/2) = -(-1) = 1$$



(Mitchell)

FIGURE 3.2
The entropy function relative to a boolean classification, as the proportion, $p_{\oplus}$, of positive examples varies between 0 and 1.

If the classification contains more than two classes (not boolean), *c* classes, the entropy is defined by

$$Entropy(\mathrm{S}) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

where p_i is the proportion of S belonging to class i

Admitting that the collection *S* has the same proportion 1/*c* of each class, the entropy will be maximum, and with value

$$Entropy(S) = -(1/c)\log_2(1/c) - (1/c)\log_2(1/c)\ldots. - (1/c)\log_2(1/c) =$$
$$= (1/c)\log_2 c + (1/c)\log_2 c + \ldots + (1/c)\log_2 c =$$
$$= (1/c + 1/c + \ldots. + 1/c)\log_2 c = 1\log_2 c = \log_2 c$$

The entropy measures the level of “impurity” (high entropy, high impurity) or of homogeneity (low entropy, high homogeneity) of a training data set.

Descending the three, from the root to the leafs, the level of entropy must diminish, such that the homogeneity will increase until one well defined class is reached (entropy equal to zero).

This objective allows to define a measure of how much appropriate is an attribute to classify the training data set.

This measure is the **information gain** , that is exactly the expected reduction of the entropy obtained by the partition of that data with that attribute.

The information gain *Gain (S,A)* of an attribute *A*, in relation to a collection *S* of examples, is defined by

$$Gain(S,A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Initial entropy of *S*

Expected values of the entropy after partitioning *S* using the attribute A value *v*

Where *Values(A)* is the set of all possible values of the attribute *A*, and S_v is the subset of *S* for which the attribute *A* has the value *v*, that is

$$S_v = \{ s \in S \,|\, A(s) = v \}$$

Example: Target-function PlayTennis, 14 training examples

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

(Mitchell)

In these conditions the total number of possible instances where all the four attributes assume a concrete value are 3x3x2x2=36. Note however that semantic valid instances can be defined (to which we give some meaning), in which some attributes may not assume a specific value. For example

D15 = (*, Mild, Normal, Weak, Yes)

Means : whatever be the Outlook, if temperature is Mild, Humidity is Normal, Wind is Weak, the day is good to play tennis. The asterisk means precisely that: whatever be. So the attribute Outlook may assume 4 values, Outlook={Sunny, Rain, Overcast, *). And in the same way for the other attributes.

We have still the situation where the attributes are represented by empty sets,

$$D16=(\varnothing,\varnothing,\varnothing,\varnothing, \text{Yes})$$

meaning that there is not any day appropriate to play tennis.

So, the total number of instances with semantic meaning, in this example, is

$$4x4x3x3 + 1=145 \text{ (the 1 corresponds to D16).}$$

And we will train the tree only with 14 examples. That is why it is an inductive learning.

Consider the attribute *Wind* assuming the values *Weak* and *Strong*.

In the set S with the 14 training examples , we have [9+,5-].

Wind= Weak in 6 positivess and 2 negatives , S_{Wweak}: [6+, 2-]
Wind=Strong in 3 positives e 3 negatives, $S_{Wstrong}$: [3+,3-]

$$Entropy(S) = Entropy[9+,5-] = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0,940$$

$$Entropy(S_{Wstrong}) = -(3/6)\log_2(3/6) - (3/6)\log_2(3/6) = (1/2)\log_2 2 + (1/2)\log_2 2 = 1/2 + 1/2 = 1,00$$

$$Entropy(S_{Wweak}) = -(6/8)\log_2(6/8) - (2/8)\log_2(2/8) = 0,811$$

Gain of information chosing *Wind* to separate the data:

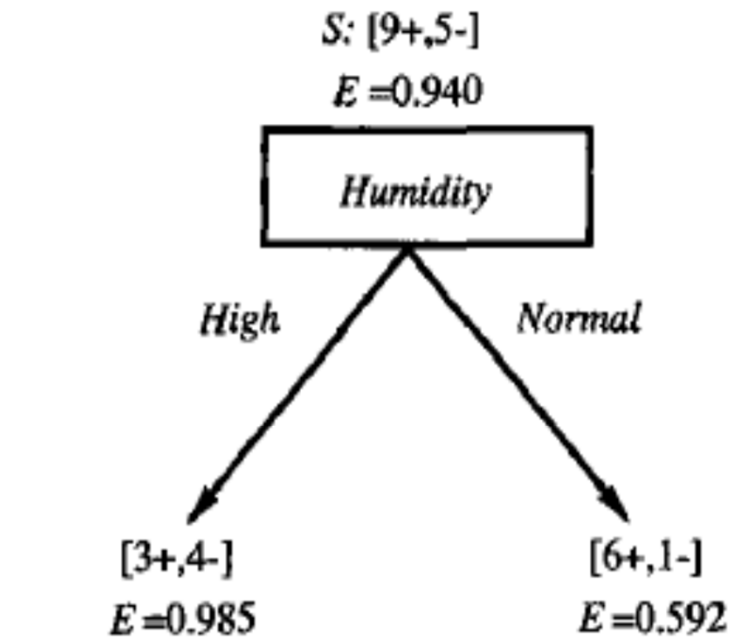
$$Gain(S,Wind) = Entropy(S) - \frac{|S_{Wweak}|}{|S|} Entropy(S_{Wweak}) - \frac{|S_{Wstrong}|}{|S|} Entropy(S_{Wstrong}) =$$

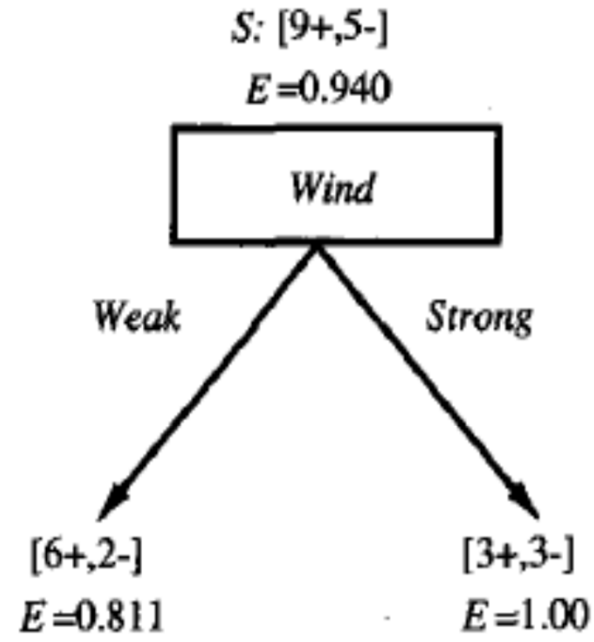
$$=0.940\text{-}(8/14)0.811\text{-}(6/14)1.00=0.048$$

Consider now the attribute *Humidity* for the concept *PlayTennis,* with values *High* and *Normal*.

It is *High* in 3 positives and 4 negatives S_{Hhigh} : [3+, 4-]

it is normal *Normal* in 6 positives 1 negative $S_{Hnormal}$: [6+,1-]





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

(Mitchell)

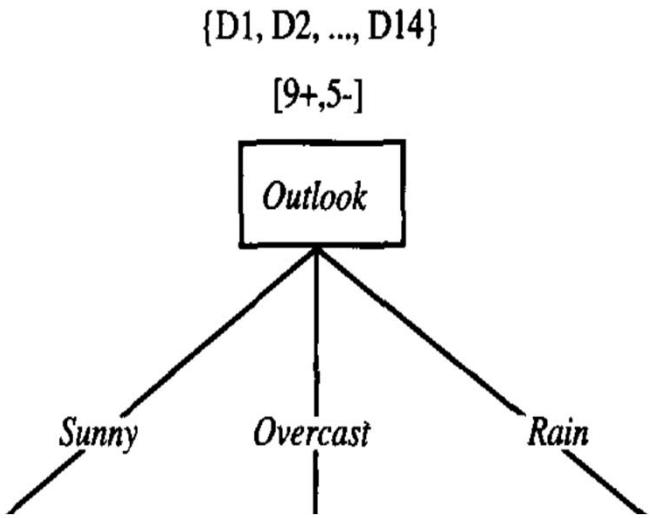
Considering the attributes *Outlook* and *Temperature* and making similar computations, we find :

$$Gain(S, Outlook) = 0.246$$ ⟹ Best, root

$$Gain(S, Humidity) = 0.151$$

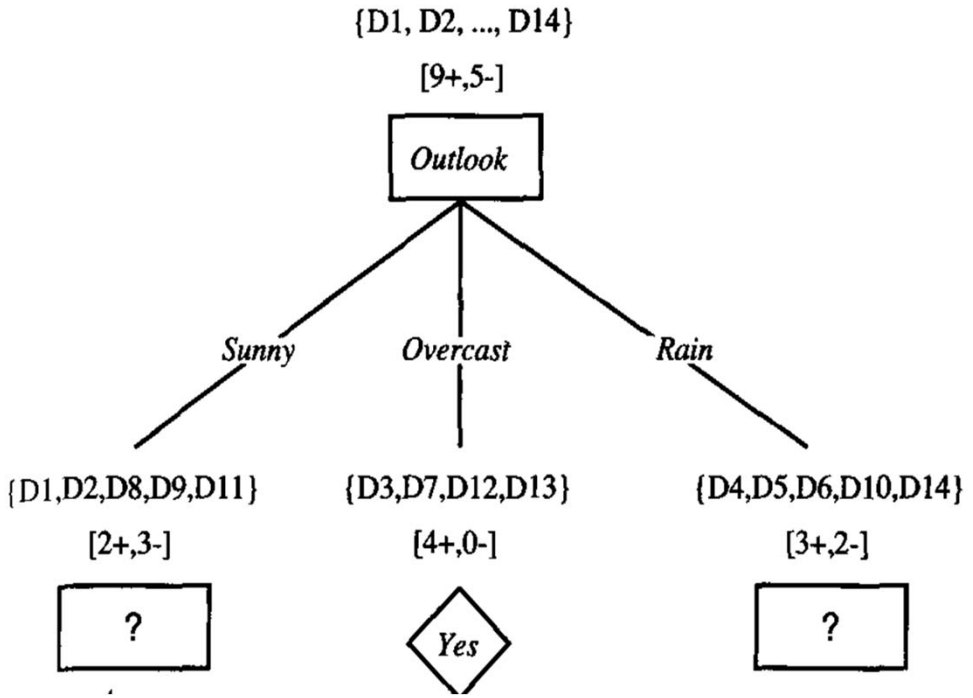
$$Gain(S, Wind) = 0.048$$

$$Gain(S, Temperature) = 0.029$$



(Mitchell)

Now how to proceed ?
We count the positives and the negatives in each branch



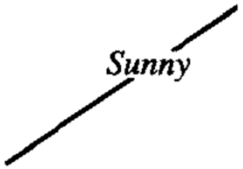
Mitchell)

Terminates here

$$Entropy([2+,3-)] = -2/5.\log_2(2/5) - 3/5.\log_2(3/5) = 0.970$$

$$Entropy([4+,0-)] = -4/4.\log_2(4/4) - 0/4.\log_2(0/4) = 0.000$$

$$Entropy([3+,2-)] = -3/5.\log_2(3/5) - 2/5.\log_2(2/5) = 0.970$$



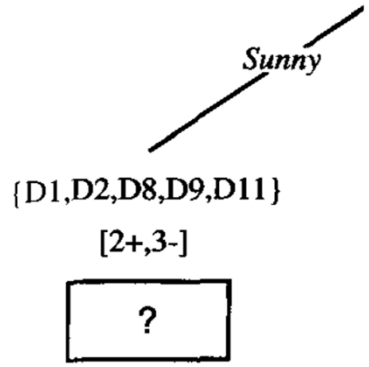
{D1,D2,8,D9,D11}

[2+,3-]

(Mitchell)

One repeats for Sunny:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes



(Mitchell)

We can chose: *Humidity, Temperature* ou *Wind*.
Counting in the table (in the 5 lines *sunny*) :

Humidity: High =[0+,3-], *Normal*=[2+,0-]
Temp: Hot =[0+,2-], *Mild*=[1+,1-], *Cool*=[1+,0-]
Wind: Weak=[1+,2-] , *Strong*=[1+,1-]

$$S_{Sunny} = \{D1, D2, D8, D9, D11\}$$

$$Gain(S_{Sunny}, Humidity) = 0.970 - (3/5)0.0 - (2/5)0.0 = 0.970$$

$$Gain(S_{Sunny}, Temperature) = 0.970 - (2/5)0.0 - (2/5)1.0 - (1/5)0.0 = 0.570$$

$$Gain(S_{Sunny}, Wind) = 0.970 - (3/5)0.918 - (2/5)1.0 = 0.19$$

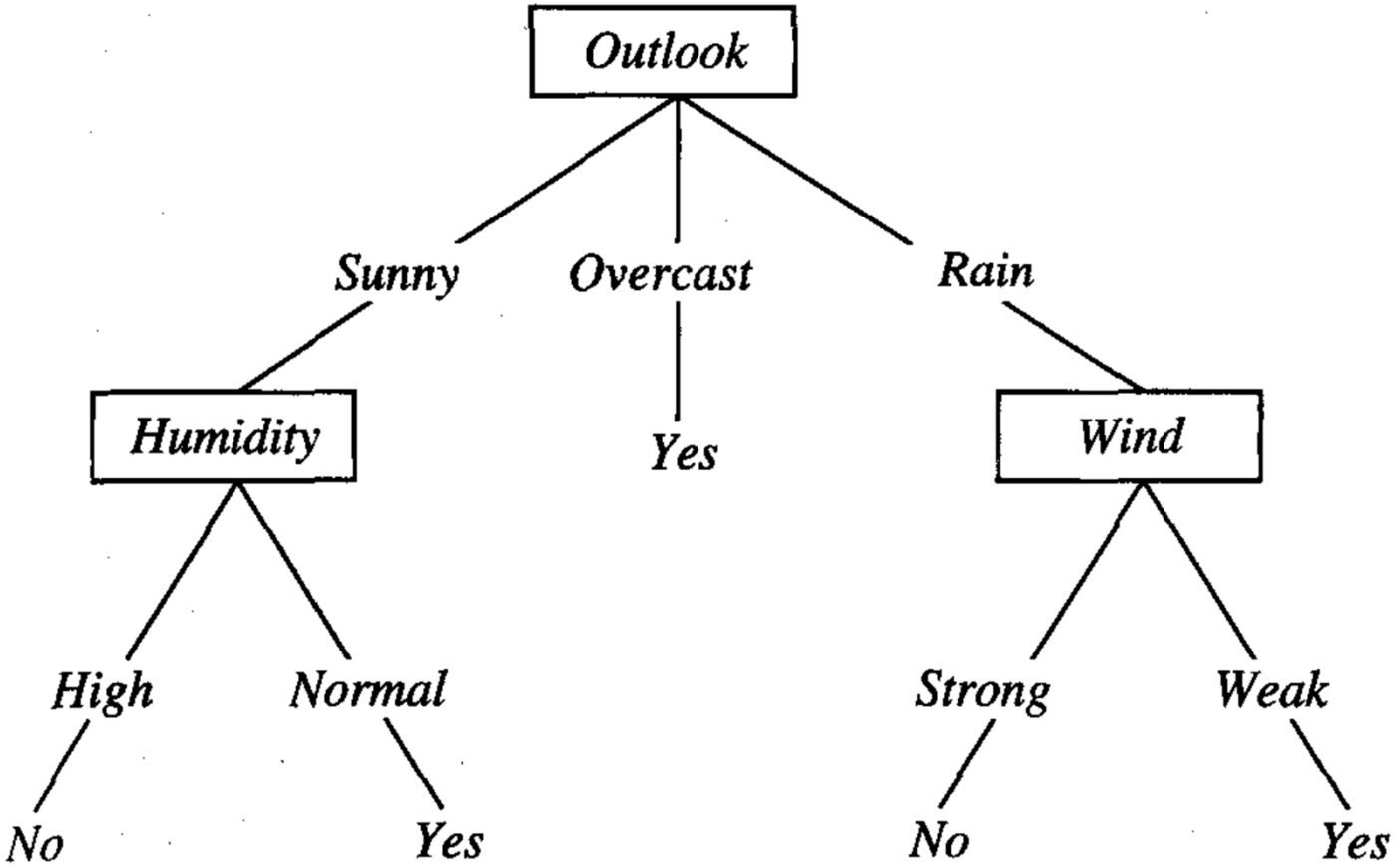
Continuing by the branch *Rain*,

We repeat for *rain*:

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D10	Rain	Mild	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

(Mitchell)

Continuing by the branch *Rain*, the tree is completed.



(Mitchell)

5 leafs

The tree can be used for instances not contained in the training set. For example:

(Outlook, Temperature, Humidity, Wind) PlayTennis

(Overcast, Cool, High, Weak) gives Yes
(Sunny, Cool, High, Strong) gives No

In each path from the root to the leafs, it is not mandatory that one finds all the attributes. If some is missing, it means that in accordance with the inductive learning from a training set, that missing attribute is irrelevant for the classification in that path.

There are many decision trees for the same training set. The search strategy of ID3 (in the decision trees space):

(a) prefers the shortest trees with respect to the longer ones and
(b) selects the trees that place the attributes with more information gain closer to the root.

However, it is not guaranteed that the ID3 finds always the smallest tree consistent with the training set, because it favors the trees that place the attributes with higher information gain closer to the root.

Why to prefer shorter hypothesis ?

Will favoring the shorter trees be a solid approach for the generalization capability of the tree (for other than the training examples)? This generic question (favoring the more complex or the simpler) has been debated by philosophes through the centuries, and that debate still goes on without solution. William de Occam (or Ockham) was one of the pioneers to discuss the question, circa 1320, and this principle is frequently called the "Occam's razor (apparently, he has formulated it while shaving himself):

Occam's razor principle: Prefer the hypothesis simpler that adjust to the data.

Lex parsimonia:"*entia non sunt multiplicanda praeter necessitatem*" (the entities should not be multiplied more that it is necessary).

Challenges of learning in decision trees: from ID3 to C4.5

ID3 has been the first algorithm to build trees from data, by Quinlan (1986).

It poses however several question:

- How to prevent overfitting the data ?
- How deep should the tree grow ?
- How to treat continuous attributes ?
- How to select an adequate measure for the selection of attributes ?
- How to process training sets with missing data ?
- How to treat attributes with different costs ?
- How to improve the computational performance ?

Answers to these questions lead to the algorithm C4.5.

Preventing overfitting:

- **post-pruning nodes**. Pruning a node is to remove it from the tree as well as the subtree rooted at that node. The pruned node becomes a leaf assigned to the most common classification of the training examples affiliated with that node. Then the new tree is validated in a validation set and if it performs no worse than the original tree in that set the node is confirmed removed.
- nodes are pruned iteratively, always choosing the node whose removal most increases the decision tree performance over the validation set.
- see more on page 70 Mitchel.

Incorporating continuous-valued attributes:

- Defining new discrete-valued attributes that partition the continuous attribute value into a discrete set of intervals, defining a set of thresholds. The thresholds must produce the greatest information gain. More on page 73 Mitchel.

Alternative measurements for selecting attributes

- The information gain measure is naturally biased, because it favors attributes with many values over those with few values. An attribute with many values divides the training set into very small subsets. As a consequence, it will have a very high information gain relative to the training examples, despite being a very poor predictor of the target function over unseen instances (because of the many possible values).
- The gain ratio is an alternative to information gain. It incorporates the term *split information* that is sensitive to how broadly and uniformly the attribute splits the data. It is the entropy of the train set with respect to the attribute.

Split information:

$$SplitInformation(S,A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Gain ratio:

$$GaiRatio(S,A) \equiv \frac{Gain(S,A)}{SplitInformation(S,A)}$$

where Gain (S,A) is the previous information gain.

The exist other alternatives to Gain ratio. See page 74 Mitchel.

Handling training examples with missing attribute values

It happens frequently that in practical cases some attributes do not have all values for some instances.

To compute the Gain (S,A) in a node *n*, the attribute A must have values for all instances in S. Possible solutions:

- consider the value that is most common among the training examples at node *n*.
- assign a probability to each of the possible values of A, looking at the observed frequencies of the various values for A in the examples at node *n*. Then these probabilities are used to compute the information Gain.
- there are other solutions. For more see page 75 Mitchel.

Handling attributes with different costs

When the attributes have different costs (to obtain, social cost, etc.), trees with low-cost attributes may be preferred where possible, relying on high-cost attributes only when needed to produce reliable classifications.
This may be done by introducing a cost into the attribute selection measure.
One possible solution: divide the information Gain by the cost of the attribute, leading to the preference of lower-costs when building the tree

There are other solutions proposed in several works. For more see page 76 Mitchel.

Bibliography

Mitchell, T.M., Machine Learning, McGraw-Hill, 1997 (Caps. 2 e 3).

Quinlan, J.R. Induction of decision trees, Machine learning 1(1), 81-106, 1986

Quinlan, J.R. C4.5: Programs for Machine Leaning, San Mateo, CA: Morhan Kaufmann, 1993.

Marsland, S., Machine Learning, An Algorithmic Perspective (Cap.6), 2nd Ed, Chapman and Hall/CRC, 2014.

Murphy, Kevin P., Machine Learning, A Probabilistic Perspective. MIT Press 2012. (p. 544- 563.)