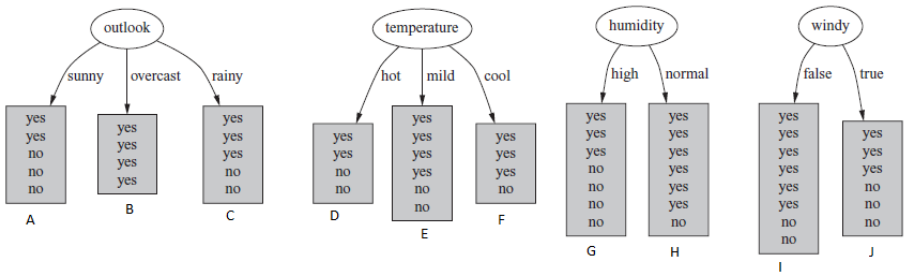
A **decision tree** is a rooted tree used to classify instances based on their attributes. Each node in the tree corresponds to an attribute, and each branch from the node indicates possible values for the attribute. For nominal attributes, a node will have a child for each possible value. For numeric attributes, some other scheme is used (e.g., splitting based on cutoff values). The idea is that as one travels down a branch from the root, the set of classes an instance matches gets smaller and smaller.



Many trees can be constructed for a given data set. The basic process is to select some attribute, construct a node for it, and expand the node based on possible values. This process is repeated until all instances represented by the node are all of the same class or the node otherwise cannot be expanded. In the ideal case, a finished tree will correctly classify any given instance (that is, at a leaf, only one class remains). This ideal can't always be reached, however. For instance, a data set might put two distinct instances with identical values for input attribute in distinct classes. In this case, there's no way to deterministically classify them both correctly.

Since many trees are possible, the problem of determining which is best naturally arises. A common scheme for tree construction uses concepts from information theory. The basic idea is that the instances represented at a given node will possibly belong to several classes. If so, the node is in some sense "impure". When picking an attribute to divide the instances, we chose one that results in child nodes that most reduce this impurity. that is, we pick the attribute that results in the highest **information gain**.

The measure of impurity is called **entropy**. Let S be a set of examples and $\{c_1, \ldots, c_m\}$ a partition of S into classes. Suppose when we randomly select an instance from S, the probability of being in class $c_1, \ldots, c_m$, respectively, is $pc_1, \ldots, p_m$. The entropy of S, written $entropy(S)$, or $H(S)$, is

$$entropy(S) = -\sum_{i=1}^{m} p_i log_2(p_i)$$

Using $\frac{|c_i|}{|S|}$ for p_i yields $entropy(S) = -\sum_{i=1}^{m} \frac{|c_i|}{|S|} log_2(\frac{|c_i|}{|S|})$.

If S consists all of the same class (as in B from the figure above), then its entropy is $-1 * log_2(1) = 0$. A little thought reveals that if the probability of being in one of classes $c_1, \ldots, c_m$ is uniformly distributed, then the entropy of the set is $log_2(m)$:

$$entropy(S) = -\sum_{i=1}^{m} \frac{1}{m} log_2(\frac{1}{m}) = -\frac{1}{m}\sum_{i=1}^{m}(log_2(1) - log_2(m)) = -\frac{1}{m}\sum_{i=1}^{m} -log_2(m) = -\frac{1}{m}m(-log_2(m)) = log_2(m).$$

Entropy is typically measured in "bits". In a given problem, if there are m classes and there is an equal probability of an example from S being in any class, then it takes, on average, $log_2(m)$ yes/no questions (bits) to determine which class the instance belongs to, and $log_2(m)$ is the entropy of S. If $Pr(c) = 1$, then no bits are needed (the answer is known), and $entropy(S) = 0$.

Example 1. The entropy values of sets $A - J$ in the partial decision trees are shown below (note that entropy can be a real number ≥ 0):

A	$-\frac{2}{5}log_2(\frac{2}{5}) - \frac{3}{5}log_2(\frac{3}{5}) \approx 0.971$	F	$-\frac{3}{4}log_2(\frac{3}{4}) - \frac{1}{4}log_2(\frac{1}{4}) \approx 0.811$
B	$-\frac{4}{4}log_2(\frac{4}{4}) = 0$	G	$-\frac{3}{7}log_2(\frac{3}{7}) - \frac{4}{7}log_2(\frac{4}{7}) \approx 0.985$
C	$-\frac{3}{5}log_2(\frac{3}{5}) - \frac{2}{5}log_2(\frac{2}{5}) \approx 0.971$	H	$-\frac{6}{7}log_2(\frac{6}{7}) - \frac{1}{7}log_2(\frac{1}{7}) \approx 0.592$
D	$-\frac{2}{4}log_2(\frac{2}{4}) - \frac{2}{4}log_2(\frac{2}{4}) = 1$	I	$-\frac{6}{8}log_2(\frac{6}{8}) - \frac{2}{8}log_2(\frac{2}{8}) \approx 0.811$
E	$-\frac{4}{6}log_2(\frac{4}{6}) - \frac{2}{6}log_2(\frac{2}{6}) \approx 0.918$	J	$-\frac{3}{6}log_2(\frac{3}{6}) - \frac{3}{6}log_2(\frac{3}{6}) = 1$

The path from the root of a decision tree to a node a will filter out possible classes. For a given set of examples, only a subset will match a. Let $|a|$ stand for the size of that set. If a has children $b_1, \ldots, b_n$, we can calculate the weighted average entropy of the children:

$$\sum_{i=1}^{n} \frac{|b_i|}{|a|} entropy(b_i)$$

Example 2. The weighted average entropies for the leaves of each tree above is shown below.

A,B,C: $\frac{5}{14} * 0.971 + \frac{4}{14} * 0 + \frac{5}{14} * 0.971 \approx 0.693536139$	G,H: $\frac{7}{14} * 0.985 + \frac{7}{14} * 592 \approx 0.788450457$
D,E,F: $\frac{4}{14} * 1 + \frac{6}{14} * 0.918 + \frac{4}{14} * 0.811 \approx 0.911063393$	I,J: $\frac{8}{14} * 0.811 + \frac{6}{14} * 1 \approx 0.892158928$

We use this to calculate the **information gain** for a (that is, how much entropy is reduced): $gain(a) = entropy(a) - \sum_{i=1}^{n} \frac{|b_i|}{|a|} entropy(b_i)$.

To determine which attribute to split on, we choose the one yielding the highest information gain.

Example 3. In the weather data set, there are 9 "yes" and 5 "no" examples, which yields an entropy of $-\frac{9}{14} log_2(\frac{9}{14}) - \frac{5}{14} log_2(\frac{5}{14} \approx 0.9403$. Given this, the information gain associated with each of the attributes *outlook*, *temperature*, *humidity*, and *windy* is:

$gain(outlook) \approx 0.9403 - 0.693536139 \approx 0.247$	$gain(humidity) \approx 0.9403 - 0.788450457 \approx 0.152$
$gain(temperature) \approx 0.9403 - 0.911063393 \approx 0.029$	$gain(windy) \approx 0.9403 - 0.892158928 \approx 0.048$

Information gain is biased towards attributes with large numbers of possible values. E.g., if an ID attribute uniquely identifies each instance, then each $entropy(b_i)$ is 0, and so $gain(a) = entropy(a)$ (i.e., the gain is as good as it can get). However, while each leaf of the tree would correctly classify each element from the training set, it would be of no help in classifying new instances.

To counteract this, we can use the **gain ratio** instead of the raw information gain.

$$gain\_ratio(a) = \frac{gain(a)}{intrinsic\_information(a)} \text{ where } intrinsic\_information(a) = -\sum_{i=1}^{n} (\frac{|b_i|}{|a|} * log_2(\frac{|b_i|}{|a|})).$$

Observe that the intrinsic information calculation is another entropy calculation, this time with the attribute values serving as the classes. As the number of b's increases, the ratio decreases. If there are a great many b's partitioning a, the ratio $\frac{|b_i|}{|a|}$ will be very small for each b_i. And so the magnitude of $log_2(\frac{|b_i|}{|a|})$ will be large and $gain\_ratio(a)$ will be small. In contrast, if a is partitioned into only a few sets, each $\frac{|b_i|}{|a|}$ will be closer to 1, and so the magnitude of $log_2(\frac{|b_i|}{|a|})$ and hence $gain\_ratio(a)$ will be larger.

Example 4. Using the weather data, the gain ratio for each attribute is shown below.

attribute	*intrinsic information*	*gain ratio*
outlook	$-\frac{5}{14} log_2(\frac{5}{14}) - \frac{4}{14} log_2(\frac{4}{14}) - \frac{5}{14} log_2(\frac{5}{14}) \approx 1.577406283$	0.156427562
temperature	$-\frac{4}{14} log_2(\frac{4}{14}) - \frac{6}{14} log_2(\frac{6}{14}) - \frac{4}{14} log_2(\frac{4}{14}) \approx 1.556656707$	0.018772646
humidity	$-\frac{7}{14} log_2(\frac{7}{14}) - \frac{7}{14} log_2(\frac{7}{14}) = 1$	0.151835501
windy	$-\frac{8}{14} log_2(\frac{8}{14}) - \frac{6}{14} log_2(\frac{6}{14}) \approx 0.985228136$	0.048848616

In this case, information gain and gain ratio both recommend *outlook*. However, in other cases, the two may diverge.

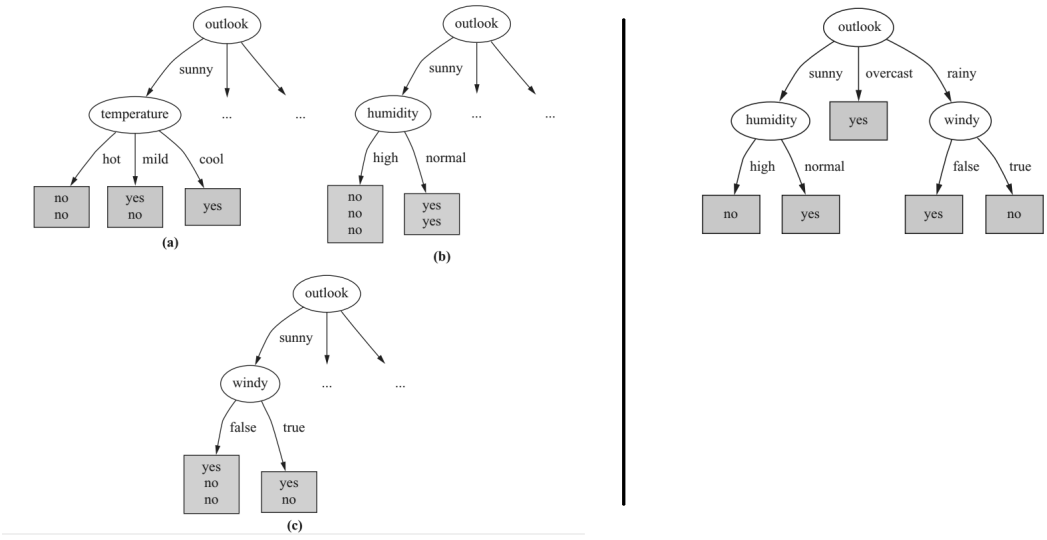
Algorithms

The basic scheme described above forms the basis for a few decision trees algorithms. These are greedy algorithms, and they typically aren't optimal. They are considered to be **divide and conquer** algorithms. **ID3** applies the scheme using information gain. C4.5 (J48 in Weka) improved upon ID3, incorporating gain ratio. Both algorithms were developed by Ross Quinlan. Also, numerical attributes are split using a threshold which maximizes the information gate ratio.

References

[1] Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning* 1(1):81-106.

[2] Witten, I; Frank, E.; Hall, M. Data Mining: Practical Machine Learning Tools and Techniques (3rd ed). Morgan Kaufmann 2011.



Further expanding the tree (Left); a completed tree (Right).

Algorithm 1 ID3($Examples, TargetAttribute, Attributes$).

// $Examples$, the set of examples;
// $TargetAttribute$, the target attribute to use;
// $Attributes$, the set of attributes remaining
Create a node $Root$ for the tree
if each element of $Examples$ is positive **then**
  return $Root$ with label $+$
else if each element of $Examples$ is negative **then**
  return $Root$ with label $-$
else if Attributes is empty **then**
  return $Root$ with label = most common value of $TargetAttribute$ in $Examples$
end if
Set A to the attribute from $Attributes$ that best classifies $Examples$ (e.g., using information gain)
Set $Root$ to A
for each possible value, v_i of A **do**
  Add a new tree branch below $Root$, corresponding to the test $A = v_i$
  Let $Examples(v_i)$ be the subset of $Examples$ that have value v_i for A
  if $Examples(v_i)$ is empty **then**
    Then below this new branch add a leaf node with label = most common value of Target attribute in Examples
  else
    Else below this new branch add the subtree $ID3(Examples(v_i), Targetattribute, Attributes - (A))$
  end if
end for
return $Root$