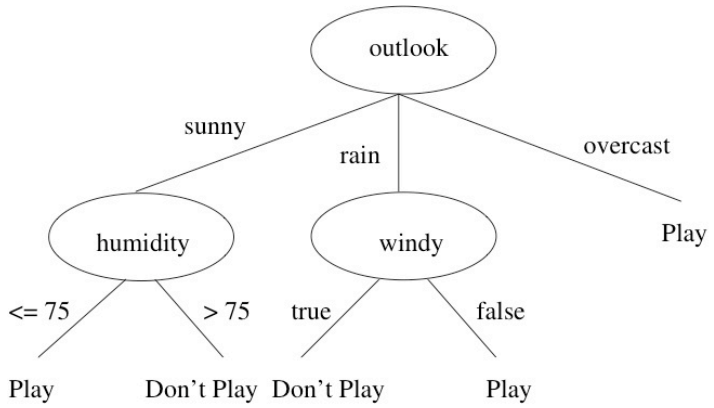
Learning Decision Trees

What is a decision tree?
- Specifies tests to perform on (or questions to ask about) attributes of an example
- Depending on the outcomes/answers, specifies the class to which an example belongs

The following is a simple example of a decision tree for determining whether today is a good golf day. There are two classes: Good day to play and Bad day to play.



Given an example, you determine the class to which it belongs by tracing through the decision tree from the top ("root") to a classification ("leaf") That is, for a particular day, you can determine whether it is a good or bad golf day by tracing through the decision tree.

Objective: learn a tree that is

Correct (both on training examples and on unseen examples)

Compact (for efficiency, comprehensibility, generality)

When constructing the tree, select tests on attributes that will best differentiate examples from each other.

Why construct a tree? Our goal is to construct a classifier. A decision tree is clearly a classifier.

<u>Consider this</u>: What's the relationship between knowledge represented in a decision tree and knowledge represented in logic? How might you describe the information in the tree above as logic sentences?

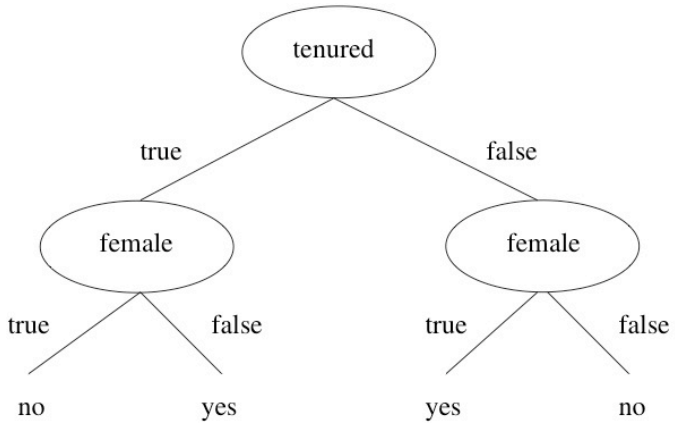
Using our intuition to build a tree

Before looking at an algorithm for building a decision tree, let's first use our intuition to build a good decision tree from data.

Consider the following data set:

	Tenured	Female	Has Kids	Has Pets	Teaches "Systems" Courses
Tom	True	False	True	False	True
Duane	True	False	True	True	True
Andrea	True	True	True	True	False
Steve	True	False	False	False	True
Brent	False	False	False	False	False
Morgan	False	False	True	True	False
Jeannie	False	True	False	True	True

What might lead you to derive a tree from this data set that looks like:



High-level view of the learning algorithm

If all examples are from the same class
    The tree is a leaf with that class name
Else
    Pick an attribute for the decision node (i.e., pick a test to make)
    Construct one edge for each possible value of that attribute (i.e., each possible test outcome)
    Partition the examples by attribute value (test outcome)
    Now do the same for each subgroup just created.

What makes one test node better than another?

Going back to the "golf" example:

Let P be the set of examples with class "Play."
Let D be the set of examples with class "Don't Play."

Say that $|P| = 10$. That is, say there are 10 examples in the set "Play".
And say that $|D| = 10$. That is, there are 10 examples in the set "Don't Play."

Now say that an attribute test splits the data into two subsets, T_1 and T_2.

The best such test is one where $T_1 = P$ and $T_2 = D$.

The worst test is one where $T_1 = 1/2\ P + 1/2\ D$ and $T_2 = 1/2\ P + 1/2\ D$

Selecting Tests: Using ideas from Information Theory

Suppose that we have a set of training examples. We'll call the set **T**.

Let $\mathbf{C_i}$ be the name of a class, where $C_1, C_2, \ldots, C_n$ are the class labels assigned to examples in T.

Let $\mathbf{freq(C_i, T)}$ be the number of examples in the training set that belong to class C_i.

Now let $\mathbf{|T|}$ be the number of examples in the training set.

Suppose you were to select one example at random from T and announce that the example belonged to class C_i. This announcement would have probability

$$\frac{\mathbf{freq(C_i, T)}}{\mathbf{|T|}}$$

and the information it conveys is

$\mathbf{-\log_2 (freq(C_i, T)/|T|)}$ bits

As the probability goes up, the information conveyed goes to 0. It's highest for low probabilities.

Here's one way you might think about this: Say that I have a bag of 100 marbles. 99 of them are blue and one is red. If I pull out a blue one and announce to you that it's blue, that's not terribly interesting. On the other hand, if I tell you it's red, then that's more interesting.

We can think of the information term as representing the amount of information we need to identify an example as being a member of a particular class.

Entropy

The following measures the average amount of information needed to identify the class of an example in T:

info(T) = $- \Sigma(\text{freq}(Ci, T) / |T|) * \log_2 (\text{freq}(C_i, T)/|T|)$,

where the weighted sum is computed over the number of classes in T.

This is called the entropy of T. You can think of it as a measure of the disorder (or impurity) of the set of examples.

[Think about what happens when there are multiple classes and all of the examples in the training set belong to just one of those classes.]

So if we compute info(T), we get a measure of the disorder of the examples.

Now, how does this relate to the selection of a test? (i.e., the selection of the attribute on which we will branch in the decision tree).

We will select a split of the examples that lessens the disorder. That is, we will select a test that produces the most "pure" subsets.

Information Gain

Let X be an attribute of the examples. Now say that X has n possible values.

If X were selected as a test, we would create a decision tree node with n branches.

Now say that j is a possible value of X. Let T_j be the examples that have the value j for attribute X.

We can compute the average entropy (disorder) that results from making this split:

info_X(T) = $\Sigma (|T_i| / |T|) * \text{info}(T_i)$,

Where the sum is taken over the n possible values of the attribute X.

We can compute this for every attribute.

Once we have done so, we select the attribute that maximizes the value of

$\text{info}(T) - \text{info}_X(T)$

Information Gain measures the expected reduction in entropy caused by partitioning the examples according to attribute X.

Gain(T, X) = info(T) – $info_X(T)$

An Example. Building the Golf Tree

Say that we would like to build a "golf" decision tree from data. The training examples describe days on which it is either good to play golf – or not. The days are described by four attributes: outlook, temperature, humidity, and wind.

Say the available training examples are as follows:

	Outlook	Humidity	Wind	Class
Example 1	Sunny	>75	False	Don't Play
Example 2	Sunny	>75	True	Don't Play
Example 3	Overcast	>75	False	Play
Example 4	Rainy	>75	False	Play
Example 5	Rainy	>75	False	Play
Example 6	Rainy	≤75	True	Don't Play
Example7	Overcast	≤75	True	Play
Example8	Sunny	>75	False	Don't Play
Example9	Sunny	≤75	False	Play
Example10	Rainy	>75	False	Play
Example11	Sunny	≤75	True	Play
Example12	Overcast	>75	True	Play
Example13	Overcast	≤75	False	Play
Example14	Rainy	>75	True	Don't Play

So |T| , the number of examples in the training set, = 14.

Now, freq(Play, T) = 9 and freq(Don'tPlay, T) = 5.

So info(T) = -((9/14)($\log_2$ 9/14) + (5/14)($\log_2$ 5/14)) = 0.94

Now let's consider splitting this set of examples on the *outlook* attribute.

First, we divide the data into three sets. One contains all the examples for which the outlook is sunny; the next contains all the examples for which it is raining; the third contains all the examples for which it is overcast.

Say that the 14 original examples divide as follows:

Sunny: 5 examples, Play = 2, Don't Play = 3

Rainy: 5 examples, Play = 3, Don't Play = 2
Overcast: 4 examples, Play = 4, Don't Play = 0.

info(Sunny Set) = - ((2/5 $\log_2$ 2/5) + (3/5 $\log_2$ 3/5)) = 0.97

info(Rainy Set) = 0.97

info(Overcast Set) = 0

Now let's use these to compute the information needed to classify examples that have been divided on the outlook attribute:

(5/14)(0.97) + (5/14)(0.97) + (4/14)(0) = 0.69

This gives us an information gain of 0.94 – 0.69 = 0.25

Now we do the same type of analysis for wind and humidity. We choose the attribute that yields the greatest information gain. It turns out to be "outlook".

Now we do the same at each newly created tree node. That is, we treat the sunny group as if it is an entire set of examples. We determine the attribute that best splits those. We then do the same for the rainy group. We might find that different tests are selected at each of these nodes!