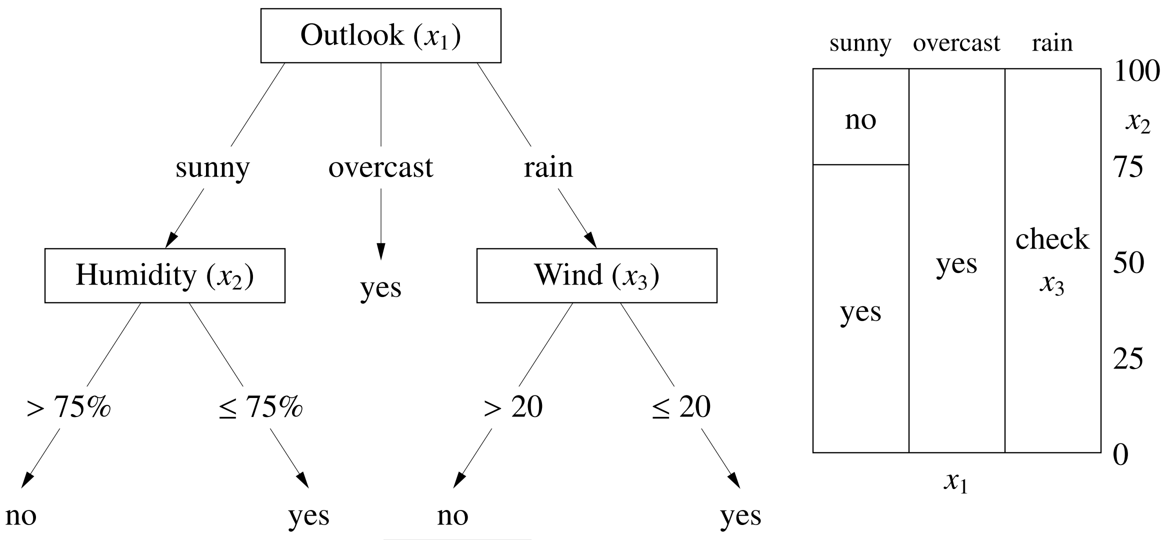
15 Decision Trees

DECISION TREES

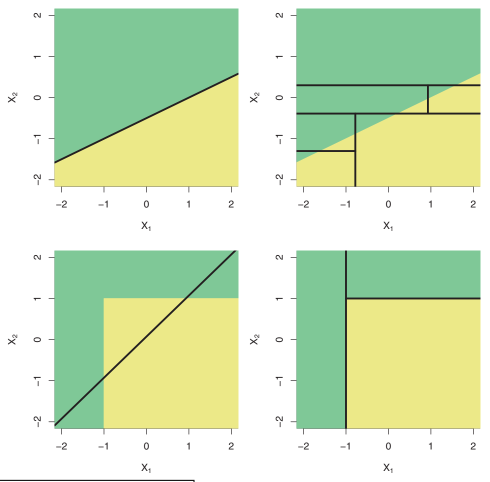
Nonlinear method for classification and regression.

Uses tree with 2 node types:
- internal nodes test feature values (usually just one) & branch accordingly
- leaf nodes specify class $h(x)$



[Draw this by hand. $\boxed{\text{dectree.pdf}}$ Deciding whether to go out for a picnic.]

- Cuts x-space into rectangular cells
- Works well with both categorical and quantitative features
- Interpretable result (inference)
- Decision boundary can be arbitrarily complicated



$\boxed{\text{treelinearcompare.pdf (ISL, Figure 8.7)}}$ [Comparison of linear classifiers (left) vs. decision trees (right) on 2 examples.]

Consider classification first. Greedy, top-down learning heuristic:
[This algorithm is more or less obvious, and has been rediscovered many times. It's naturally recursive. I'll show how it works for classification first; later I'll talk about how it works for regression.]

Let $S \subseteq \{1, 2, \ldots, n\}$ be list of sample point indices.
Top-level call: $S = \{1, 2, \ldots, n\}$.

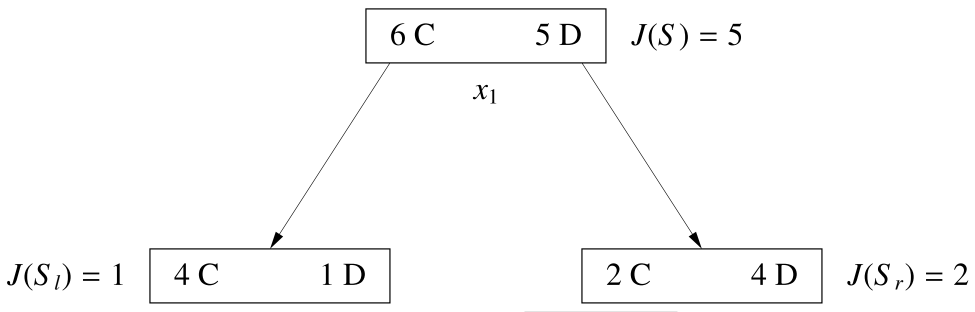
```
GrowTree($S$)
    if ($y_i = C$ for all $i \in S$ and some class C) then {
        return new leaf($C$)                    [We say the leaves are pure]
    } else {
        choose best splitting feature $j$ and splitting value $\beta$    (*)
        $S_l = \{i : X_{ij} < \beta\}$                [Or you could use $\leq$ and $>$]
        $S_r = \{i : X_{ij} \geq \beta\}$
        return new node($j, \beta$, GrowTree($S_l$), GrowTree($S_r$))
    }
```

(*) How to choose best split?
- Try all splits. [All features, and all splits within a feature.]
- For a set S, let $J(S)$ be the cost of S.
- Choose the split that minimizes $J(S_l) + J(S_r)$; or,
  the split that minimizes weighted average $\frac{|S_l| J(S_l) + |S_r| J(S_r)}{|S_l| + |S_r|}$.

[Here, I'm using the vertical brackets $|\cdot|$ to denote set cardinality.]

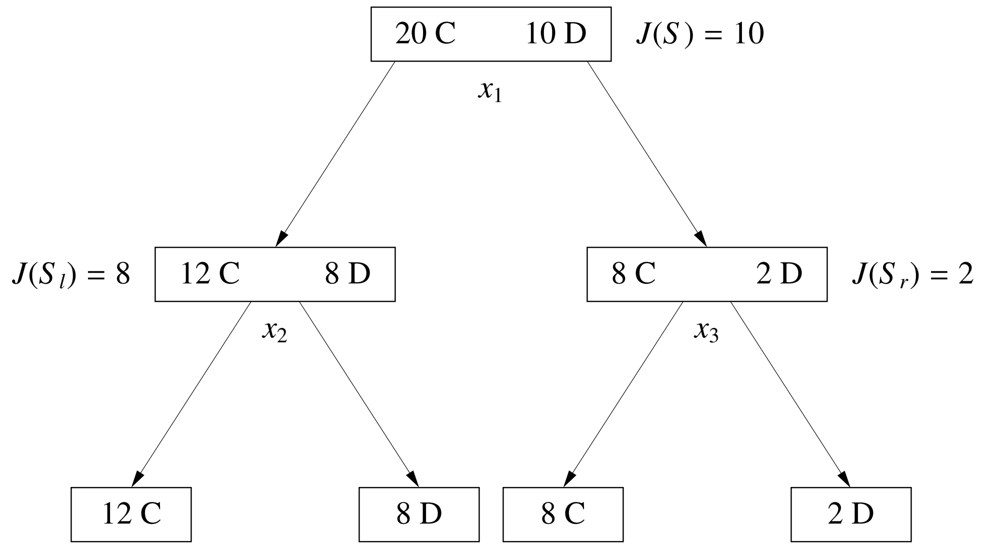
How to choose cost $J(S)$?
[I'm going to start by suggesting a mediocre cost function, so you can see why it's mediocre.]

Idea 1 (bad): Label S with the class C that labels the most points in S.
$J(S) \leftarrow$ # of points in S not in class C.



[Draw this by hand. $\boxed{\text{badcost.pdf}}$]

Problem: Sometimes we make "progress," yet $J(S_l) + J(S_r) = J(S)$.



[Draw this by hand. $\boxed{\text{delayed.pdf}}$]

[In this example, notice that even though the first split doesn't reduce the total cost at all, we're still making progress, because after one more level of splits, we're done!]

Idea 2 (good): Measure the entropy. [An idea from information theory.]
Let Y be a random class variable, and suppose $P(Y = \text{C}) = p_\text{C}$.
The surprise of Y being class C is $-\log_2 p_\text{C}$. [Always nonnegative.]
- event w/prob. 1 gives us zero surprise.
- event w/prob. 0 gives us infinite surprise!

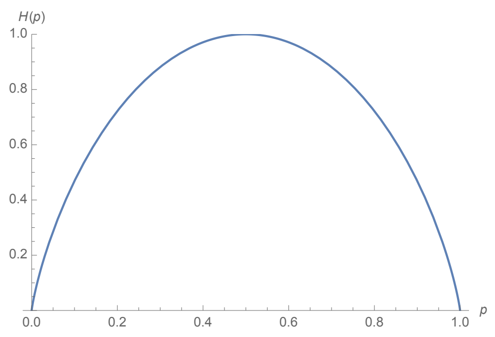
[In information theory, the surprise is equal to the expected number of bits of information we need to transmit which events happened to a recipient who knows the probabilities of the events. Often this means using fractional bits, which may sound crazy, but it makes sense when you're compiling lots of events into a single message; e.g. a sequence of biased coin flips.]

The entropy of an index set S is the average surprise

$$H(S) = -\sum_{\text{C}} p_\text{C} \log_2 p_\text{C}, \quad \text{where } p_\text{C} = \frac{|\{i \in S : y_i = \text{C}\}|}{|S|}.$$

[The proportion of points in S that are in class C.]

If all points in S belong to same class? $H(S) = -1 \log_2 1 = 0$.
Half class C, half class D? $H(S) = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$.
n points, all different classes? $H(S) = -\log_2 \frac{1}{n} = \log_2 n$.



$\boxed{\text{entropy.pdf}}$ [Plot of entropy $H(p_C)$ when there are only two classes.]

Weighted avg entropy after split is $H_{\text{after}} = \frac{|S_l|\,H(S_l)+|S_r|\,H(S_r)}{|S_l|+|S_r|}$.

Choose split that maximizes information gain $H(S) - H_{\text{after}}$. [Which is just the same as minimizing H_{after}.]

$$\begin{tikzcd}
 & \boxed{14\text{ C} \quad 16\text{ D}} \arrow[dl] \arrow[dr] \arrow[d, phantom, "x_1"] & \\
\boxed{13\text{ C} \quad 4\text{ D}} & & \boxed{1\text{ C} \quad 12\text{ D}}
\end{tikzcd}$$

$H(S) = -\frac{14}{30} \lg \frac{14}{30} - \frac{16}{30} \lg \frac{16}{30} \doteq 0.997$

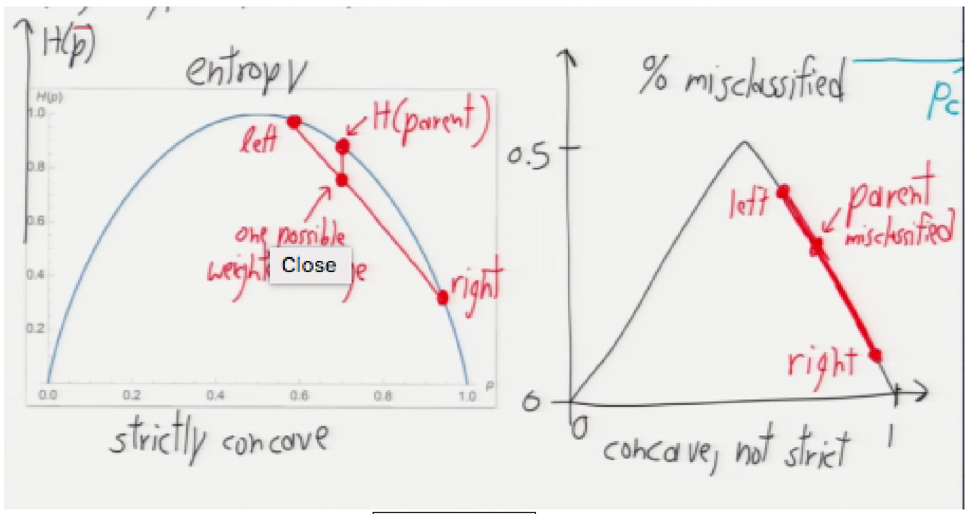
$H(S_l) = -\frac{13}{17} \lg \frac{13}{17} - \frac{4}{17} \lg \frac{4}{17} \doteq 0.787$ $\qquad$ $H(S_r) = -\frac{1}{13} \lg \frac{1}{13} - \frac{12}{13} \lg \frac{12}{13} \doteq 0.391$

[Draw this by hand. $\boxed{\text{infogain.pdf}}$]

$H_{\text{after}} = 0.615$; $\qquad$ info gain $= 0.382$

Info gain always positive *except* when one child is empty or
for all C, $P(y_i = \text{C}|i \in S_l) = P(y_i = \text{C}|i \in S_r)$.

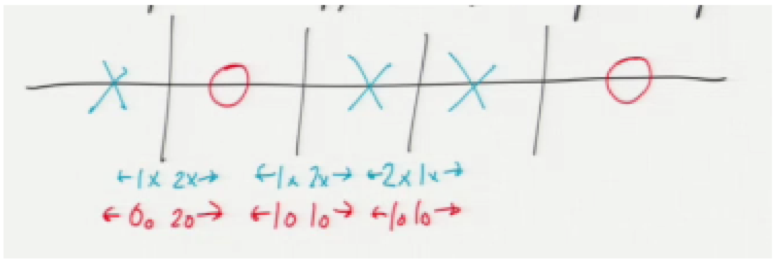
[Recall graph of entropy.]



[Draw this by hand on entropy.pdf. $\boxed{\text{concave.png}}$ If you have > 2 classes, you would need a multidimensional chart to graph the entropy, but the entropy is still strictly concave.]

[Suppose we pick two points on the entropy curve, then draw a line segment connecting them. Because the entropy curve is strictly concave, the interior of the line segment is strictly below the curve. Any point on that segment represents a weighted average of the two entropies for suitable weights. Whereas the point directly above that point represents the entropy if you unite the two sets into one. So the union of two nonempty sets has greater entropy than the weighted average of the entropies of the two sets, unless the two sets both have exactly the same p.]
[On the other hand, for the graph on the right, showing the % misclassified, if we draw a line segment connecting two points on the curve, the segment might lie entirely on the curve. In that case, uniting the two sets into one, or splitting one into two, changes neither the total misclassified sample points nor the % misclassified. The real issue, though, is that many different splits will get the same score; this test doesn't distinguish the quality of different splits well.]
[By the way, the entropy is not the only function that works well. Many concave functions work fine, including the simple polynomial $p(1-p)$.]

More on choosing a split:
- For binary feature x_i: children are $x_i = 0$ & $x_i = 1$.
- If x_i has 3+ discrete values: split depends on application.
  [Sometimes it makes sense to use multiway splits; sometimes binary splits.]
- If x_i is quantitative: sort x_i values in S; remove duplicate x_i values; [remove values, not points!]
  try splitting between each pair of consecutive values.
  [We can radix sort the values in linear time, and if n is huge we should.]
  Clever Bit: As you scan sorted list from left to right, you can update entropy in $O(1)$ time per point!
  [This is important for obtaining a fast tree-building time.]
  [Draw a row of X's and O's; show how we update the # of X's and # of O's in each of S_l and S_r as we scan from left to right.]



scan.png

Algs & running times:
- Test point: Walk down tree until leaf. Return its label.
  Worst-case time is O(tree depth).
  For binary features, that's $\leq d$. [Quantitative features may go deeper.]
  Usually (not always) $\leq O(\log n)$.
- Training: For binary features, try $O(d)$ splits at each node.
  For quantitative features, try $O(n'd)$ splits; n' = points in node
  Either way $\Rightarrow O(n'd)$ time at this node
  [Quantitative features are asymptotically just as fast as binary features because of our clever way of computing the entropy for each split.]
  Each point participates in O(depth) nodes, costs $O(d)$ time in each node.
  Running time $\leq O(nd$ depth$)$.
  [As nd is the size of the design matrix X, and the depth is often logarithmic, this is a surprisingly reasonable running time.]