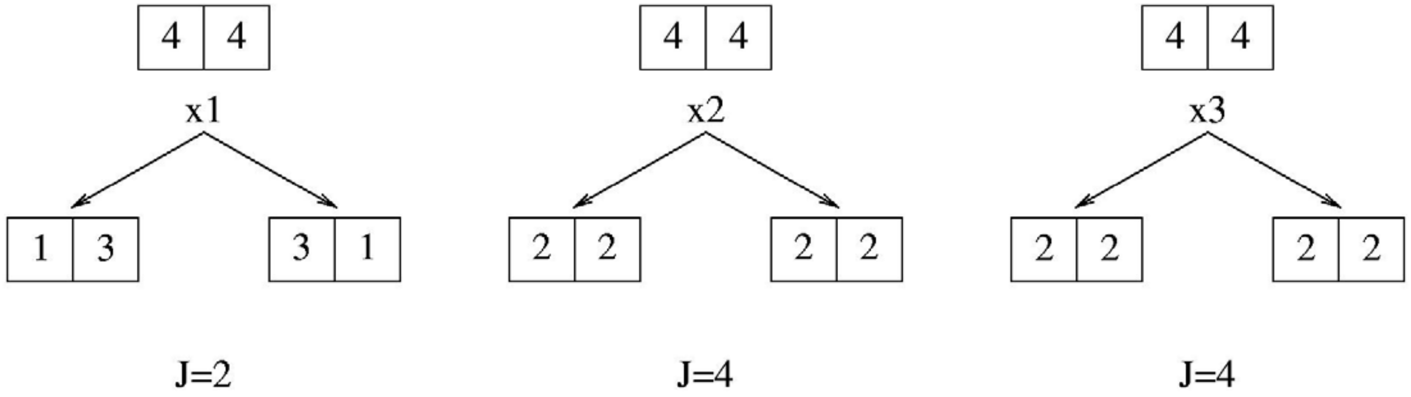
Choosing the Best Attribute

One way to choose the best attribute is to perform a 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data.

```
CHOOSEBESTATTRIBUTE(S)
choose $j$ to minimize $J_j$, computed as follows:
    $S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
    $S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
    $y_0 =$ the most common value of $y$ in $S_0$
    $y_1 =$ the most common value of $y$ in $S_1$
    $J_0 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_0$ with $y \neq y_0$
    $J_1 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$
    $J_j = J_0 + J_1$ (total errors if we split on this feature)
return $j$
```

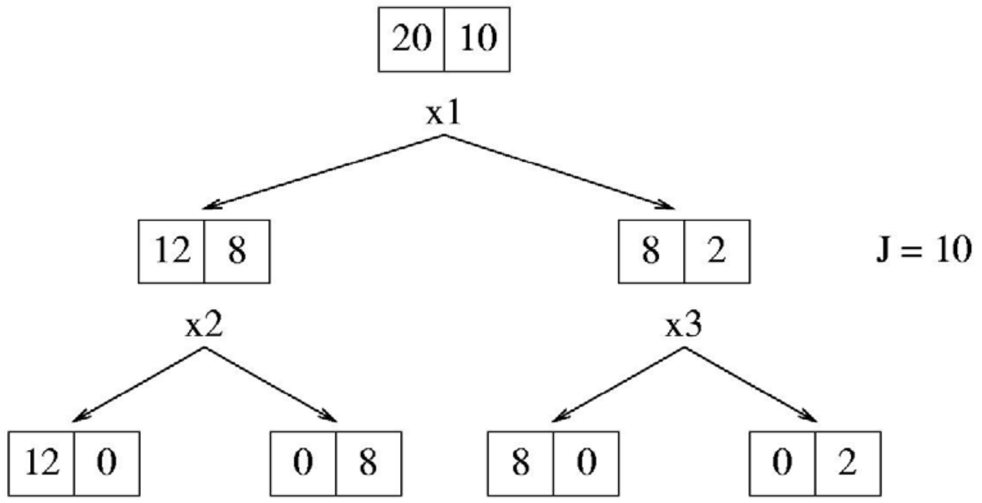
Choosing the Best Attribute—An Example

x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Choosing the Best Attribute (3)

Unfortunately, this measure does not always work well, because it does not detect cases where we are making "progress" toward a good tree.



A Better Heuristic From Information Theory

Let V be a random variable with the following probability distribution:

$P(V=0)$	$P(V=1)$
0.2	0.8

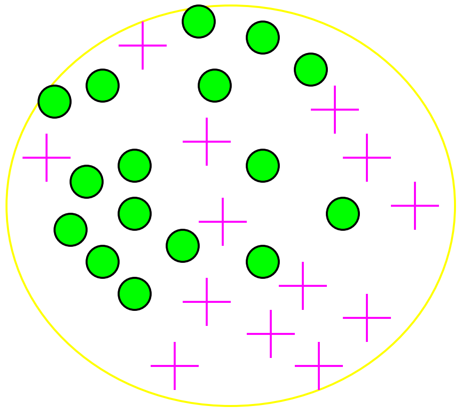
The ***surprise***, $S(V=v)$ of each value of V is defined to be

$$S(V=v) = -\lg P(V=v).$$

An event with probability 1 gives us zero surprise.
An event with probability 0 gives us infinite surprise!
It turns out that the surprise is equal to the number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results.
This is also called the *description length* of $V=v$.
Fractional bits only make sense if they are part of a longer message (e.g., describe a whole sequence of coin tosses).

Entropy: Average Surprise

- Entropy H = $\sum_i -p_i \log_2 p_i$



p_i is the probability of class i

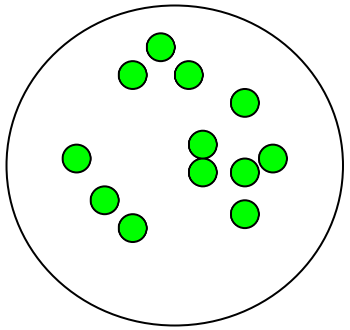
Compute it as the proportion of class i in the set.

in 2-Class case:

- What is the entropy of a group in which all examples belong to the same class?
  - entropy = $- 1 \log_2 1 = 0$

  We are done!

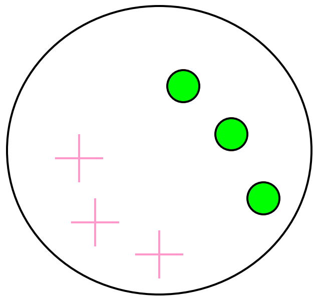
Minimum entropy



- What is the entropy of a group with 50% in either class?
  - entropy = $-0.5 \ \log_2 0.5 - 0.5 \ \log_2 0.5 = 1$

  Better start splitting

Maximum entropy

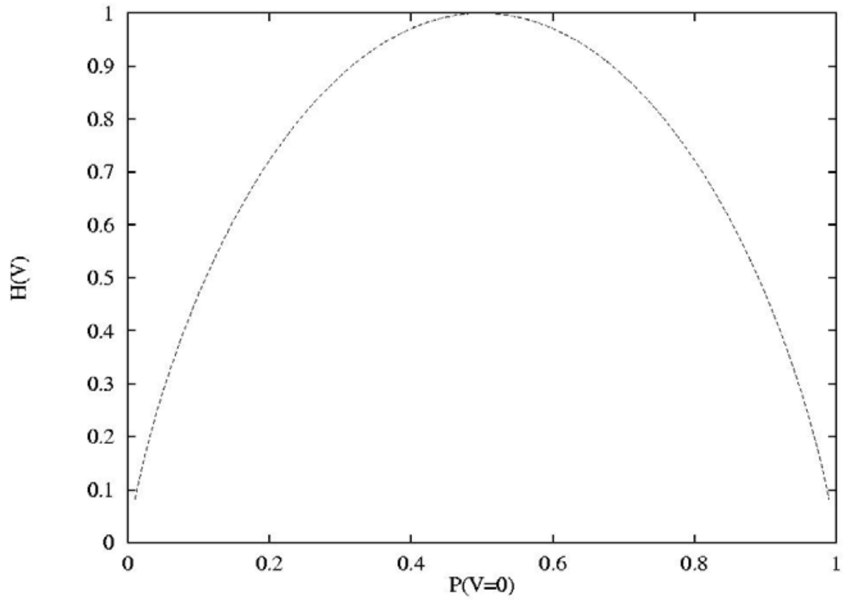


Entropy

The *entropy* of V, denoted $H(V)$ is defined as follows:

$$H(V) = \sum_{v=0}^{1} -P(H = v) \lg P(H = v).$$

This is the average surprise of describing the result of one "trial" of V (one coin toss).



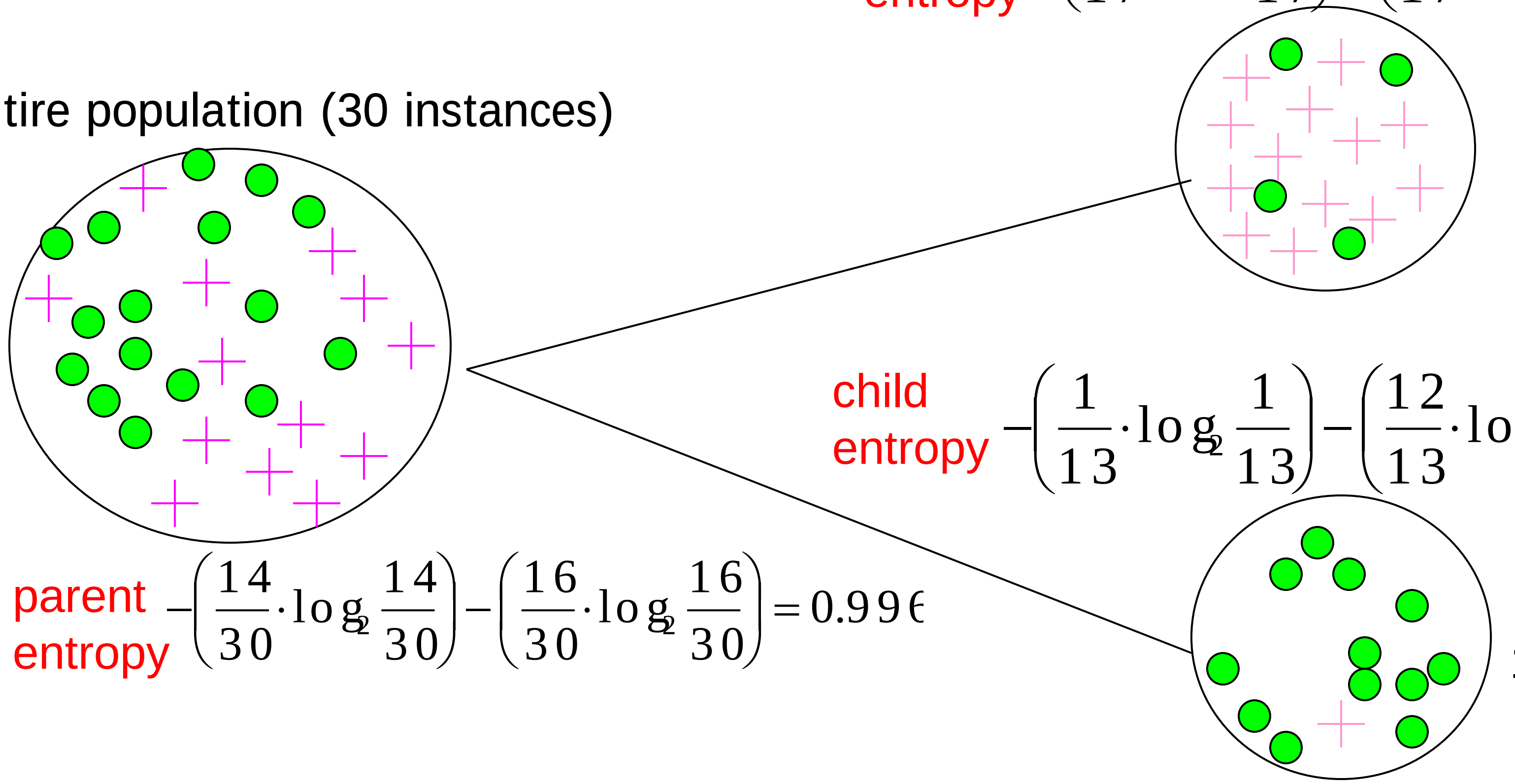
Entropy can be viewed as a measure of uncertainty.

How much can we gain from splitting?

Information Gain = entropy(parent) – [average entropy(children)]

child entropy $-\left(\frac{13}{17}\cdot\log_2\frac{13}{17}\right)-\left(\frac{4}{17}\cdot\log_2\frac{4}{17}\right)=0.787$

Entire population (30 instances)



child entropy $-\left(\frac{1}{13}\cdot\log_2\frac{1}{13}\right)-\left(\frac{12}{13}\cdot\log_2\frac{12}{13}\right)=0.391$

parent entropy $-\left(\frac{14}{30}\cdot\log_2\frac{14}{30}\right)-\left(\frac{16}{30}\cdot\log_2\frac{16}{30}\right)=0.996$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30}\cdot 0.787\right)+\left(\frac{13}{30}\cdot 0.391\right)=0.615$

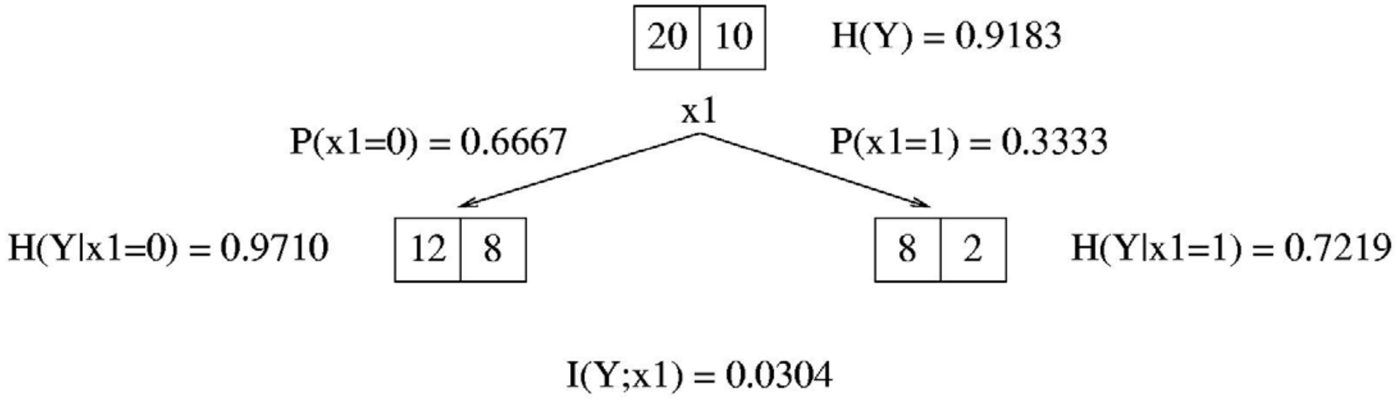
Information Gain= 0.996 - 0.615 = 0.38

Information Gain

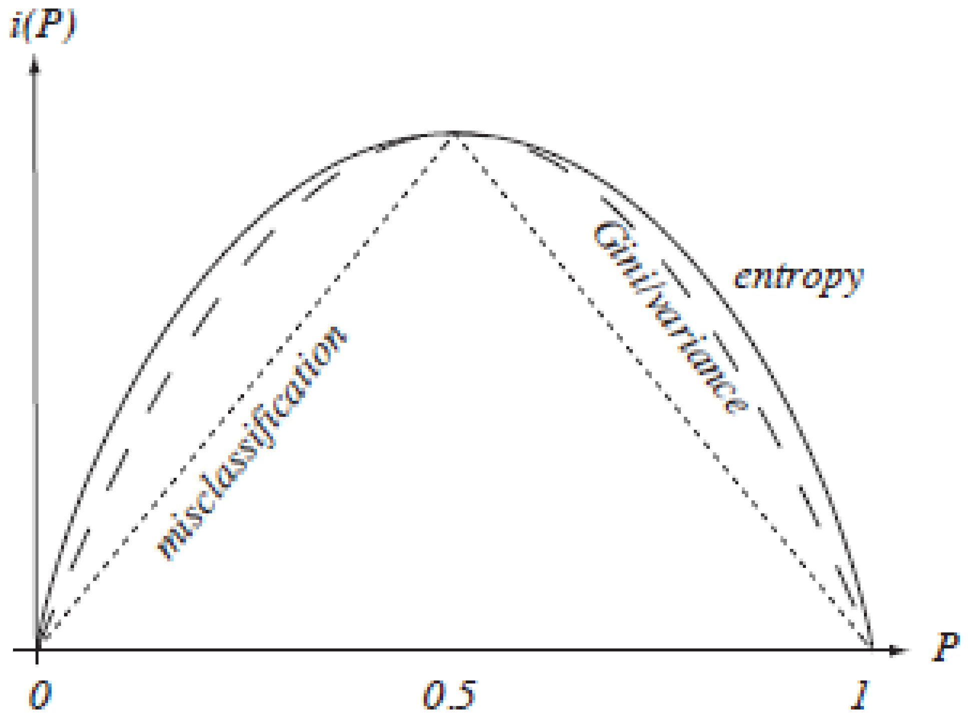
Now consider two random variables A and B that are not necessarily independent. The *mutual information* between A and B is the amount of information we learn about B by knowning the value of A (and vice versa—it is symmetric). It is computed as follows:

$$I(A;B) = H(A) - \sum_b P(B=b) \cdot H(A|B=b)$$

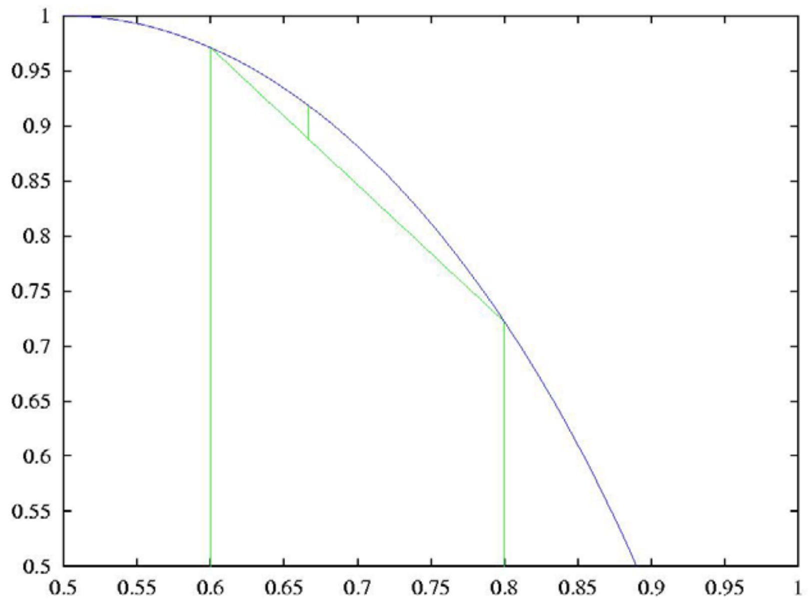
In particular, consider the class Y of each training example and the value of feature x_1 to be random variables. Then the mutual information quantifies how much x_1 tells us about the value of the class Y.



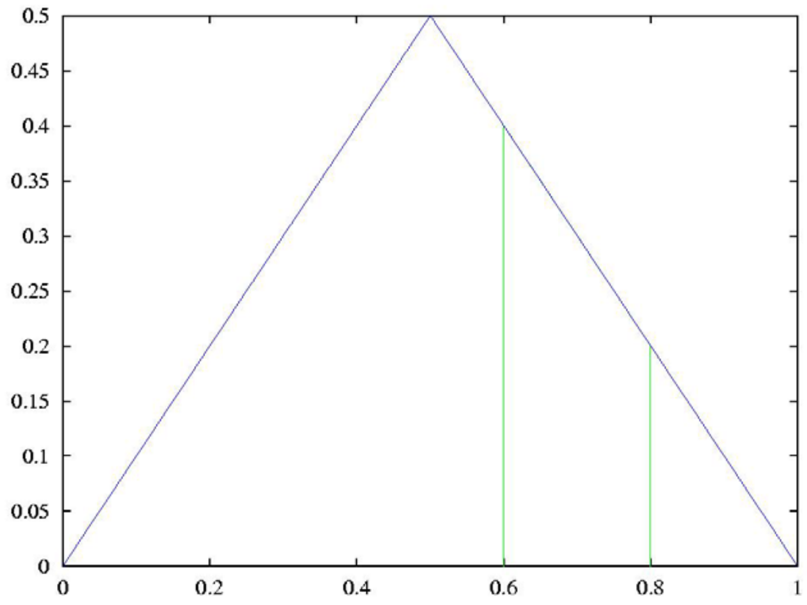
Misclassification Error vs. Entropy



Visualizing Heuristics



Entropy



Absolute Error

Mutual information works because it is a convex measure.

Non-binary Features

- Features with multiple discrete values
  - Multiway split
    - Need to normalize Information Gain, else it will always prefer multiway splits
  - One-vs-all split
    - Makes a cascade of binary splits
  - Group values into two disjoint subsets
- Real-valued Features
  - Consider a threshold split at each observed value of the feature
    - Reduces to previous problem

Multivariate Splits

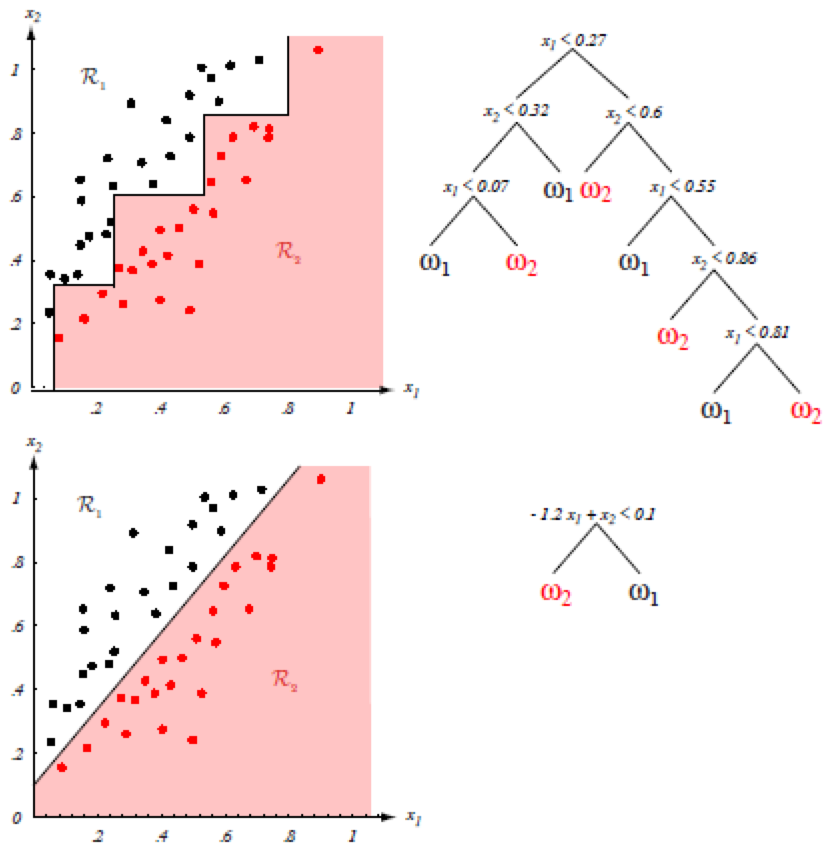


Figure 8.5: If the class of node decisions does not match the form of the training data, a very complicated decision tree will result, as shown at the top. Here decisions are parallel to the axes while in fact the data is better split by boundaries along another direction. If however "proper" decision forms are used (here, linear combinations of the features), the tree can be quite simple, as shown at the bottom.

Unknown Attribute Values

What if some examples are missing values of A?

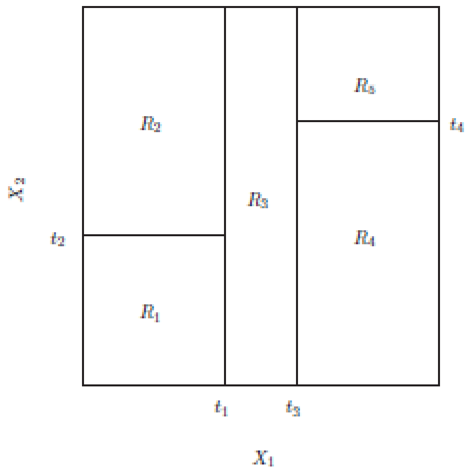
Use training example anyway, sort through tree

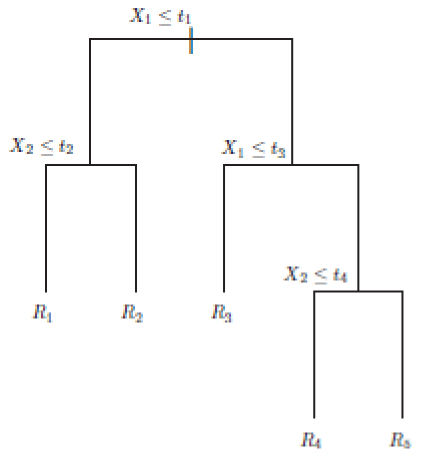
- If node n tests A, assign most common value of A among other examples sorted to node n
- Assign most common value of A among other examples with same target value
- Assign probability p_i to each possible value v_i of A
  Assign fraction p_i of example to each descendant in tree

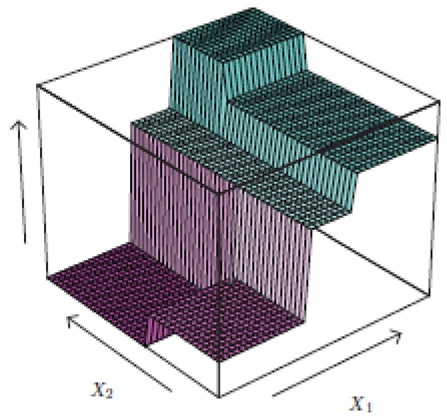
Classify new examples in same fashion

Classification vs. Regression Trees

- Classification tree: $x \rightarrow \{0,1\}$
- Regression tree: $x \rightarrow y \ \in R$



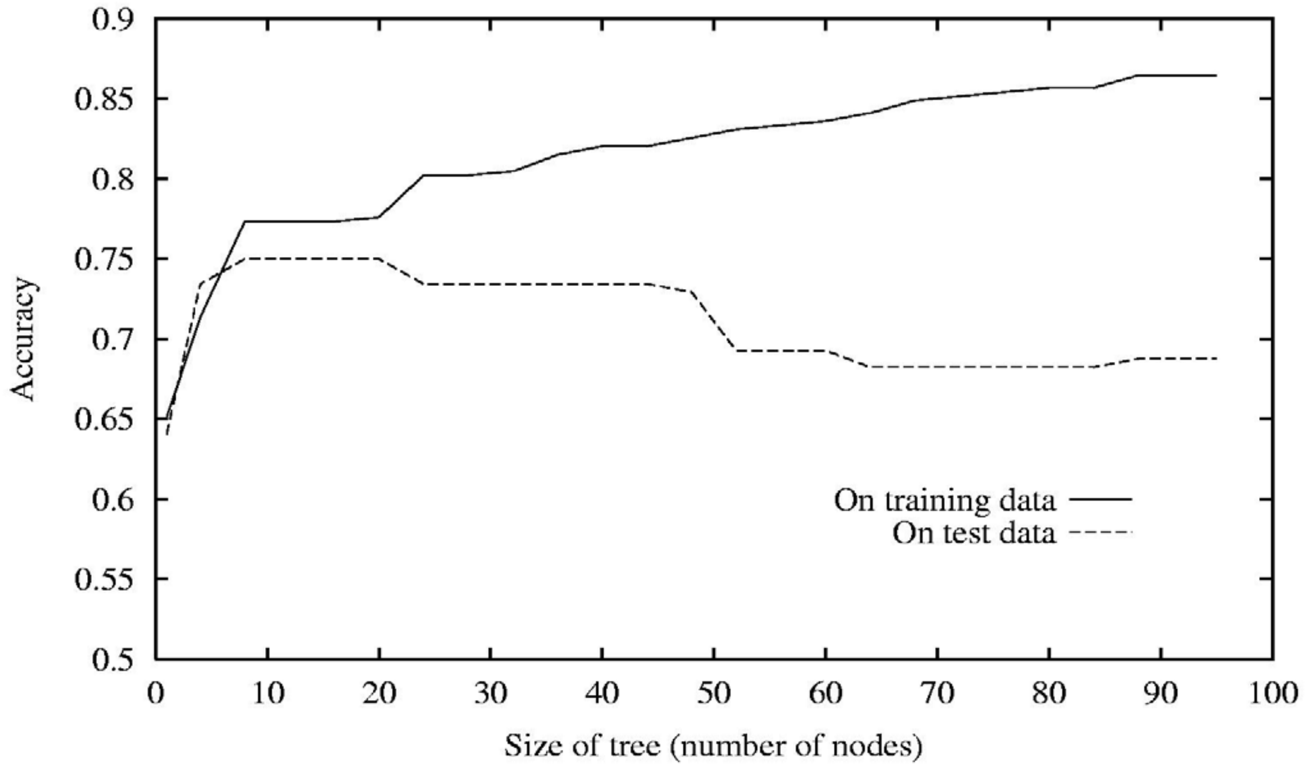




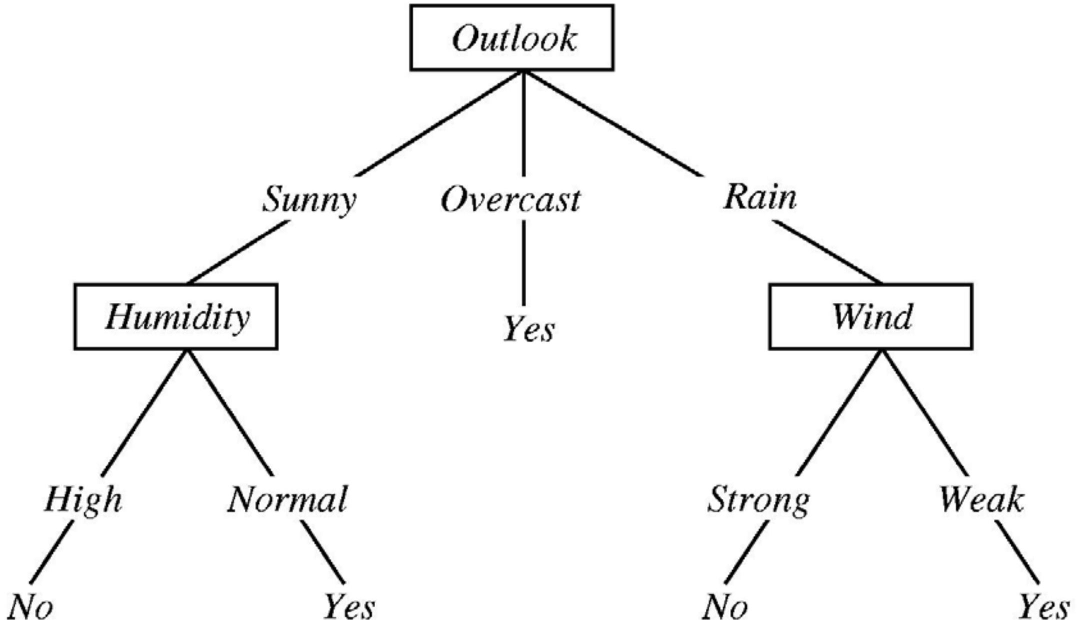
– Instead of minimizing entropy, we minimize sum of variances after the split:

$$\sum_{i:\ x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:\ x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$$

Overfitting in Decision Tree Learning



Overfitting in Decision Trees



Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, No!

What effect on tree?

Limiting Tree Size

- Shouldn’t wait until absolute purity:
  - It might never happen (label noise)
  - Data-starvation: as tree becomes deeper, each branch gets very few data points
  - Deep trees are slow and huge
- We can stop splitting earlier and make leaf node:
  - Average of *y*’s at the leaf (for regression trees)
  - Majority class or posterior distribution at the leaf (for classification trees)

How do we know when to stop growing?

- Stopping criteria:
  - Max tree depth
  - Min # of points at a node
  - Tree complexity penalty
  - Validation error monitoring
- Pruning (post-pruning, reduced-error pruning)
  - Avoid horizon effect (biasing trees to be "early deciders")
  - Still greedy but works much better

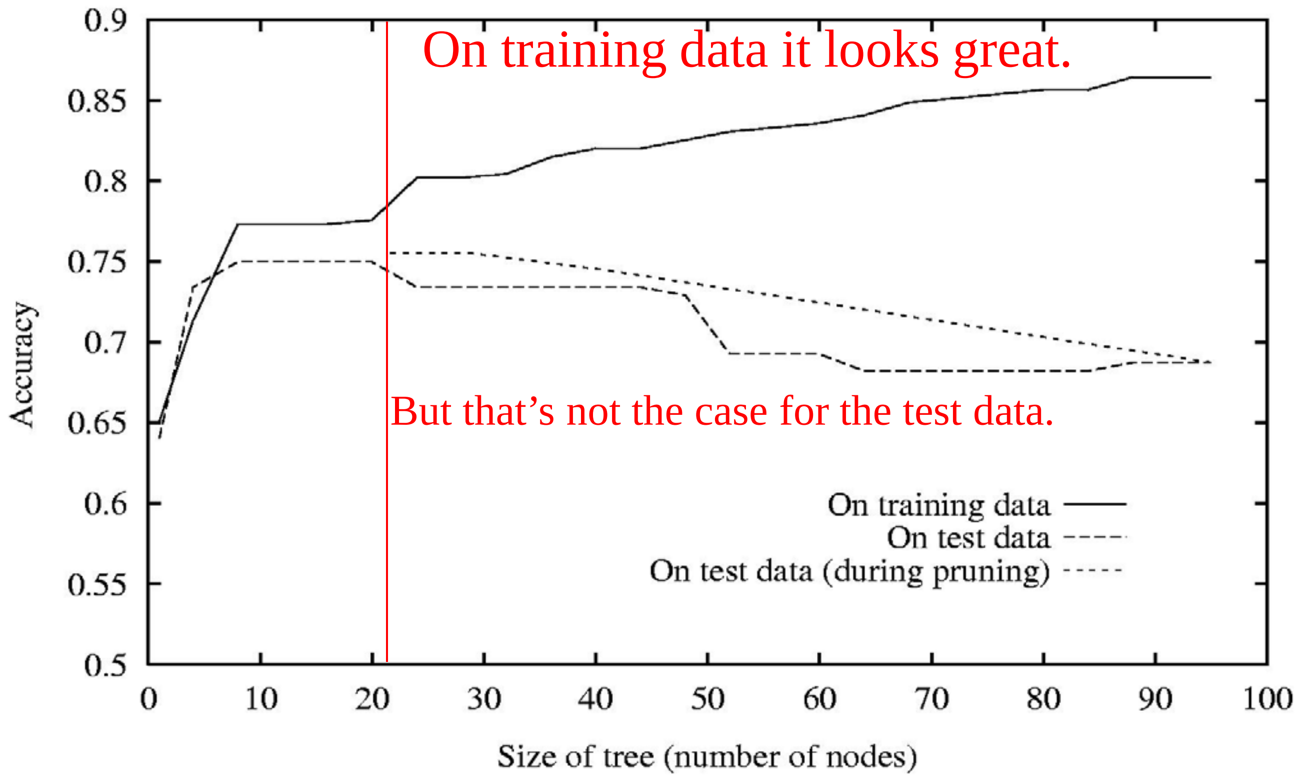
Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

Effect of Reduced-Error Pruning



The tree is pruned back to the red line where it gives more accurate results on the test data.