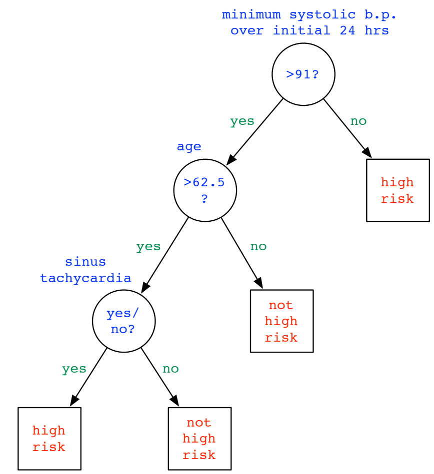
Decision trees

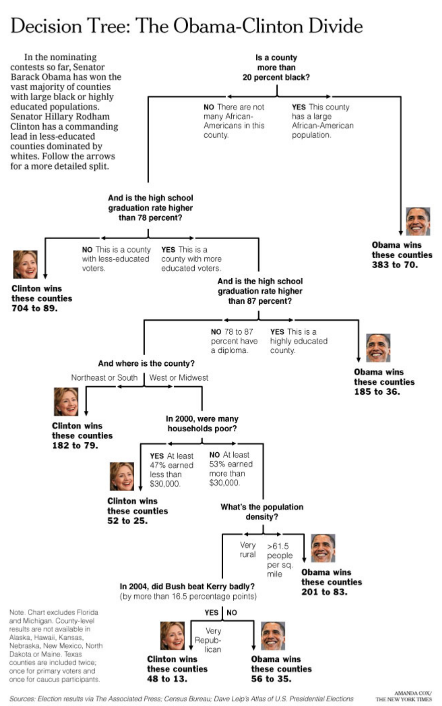
CSE 250B

Decision trees

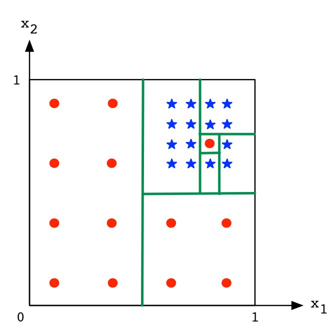
Study at UCSD Medical Center, late 1970s.
Goal: identify patients at risk of dying within 30 days after heart attack.

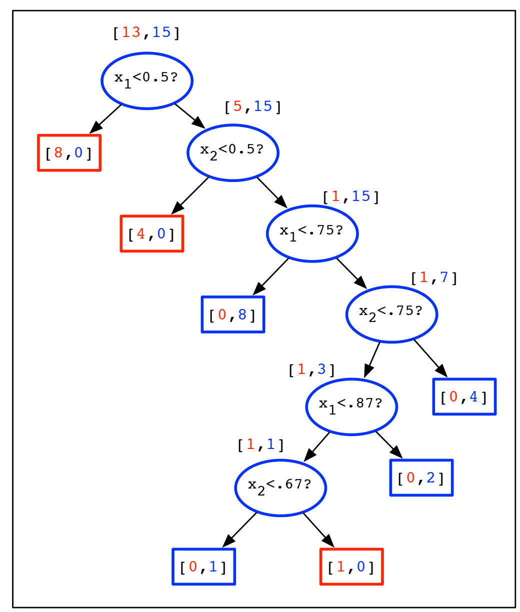
Data set: 215 patients, 37 (=20%) died. 19 relevant variables.





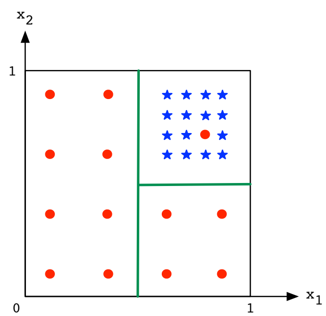
Example: building a decision tree

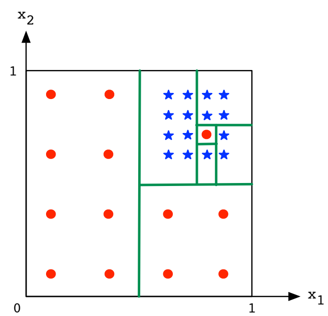




Overfitting?

Go back a few steps...





The final partition does better on the training data, but is more complex.
That one point might have been an outlier anyway.

We have probably ended up **overfitting** the data.

Decision tree issues

A very expressive family of classifiers:

- Can accommodate any type of data: real, Boolean, categorical, ...
- Can accommodate any number of classes
- Can fit any data set
- Statistically consistent

But this also means that there is serious danger of overfitting.

Building a decision tree

Greedy algorithm: build tree top-down.

- Start with a single node containing all data points
- Repeat:
  - Look at all current leaves and all possible splits
  - Choose the split that most decreases the uncertainty in prediction

We need a measure of **uncertainty in prediction**.

Uncertainty in prediction

Say there are two labels:

$+$ p fraction of the points
$-$ $1-p$ fraction of the points

What uncertainty score should we give to this?

1. Misclassification rate

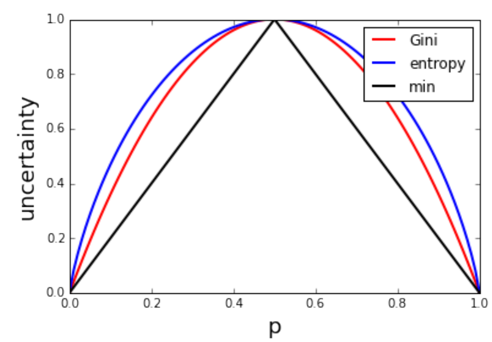
   $\min\{p, 1-p\}$

2. Gini index

   $2p(1-p)$

3. Entropy

   $p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$



Uncertainty: k classes

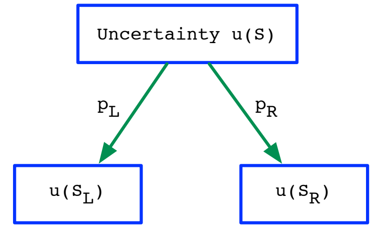
Suppose there are k classes, with probabilities $p_1, p_2, \ldots, p_k$.

	$k = 2$	General k
Misclassification rate	$\min\{p, 1-p\}$	$1 - \max_i p_i = 1 - \\|p\\|_\infty$
Gini index	$2p(1-p)$	$\sum_{i \neq j} p_i p_j = 1 - \\|p\\|^2$
Entropy	$p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$	$\sum_i p_i \log \frac{1}{p_i}$

Benefit of a split

Let $u(S)$ be the uncertainty score for a set of labeled points S.

Consider a particular split:



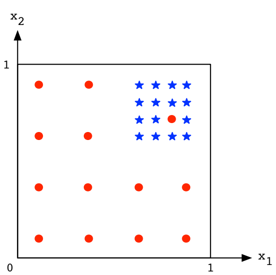
Of the points in S:

- p_L fraction go to S_L
- p_R fraction go to S_R

Benefit of split = reduction in uncertainty:

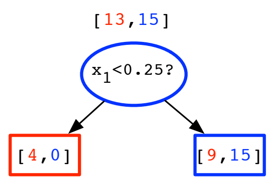
$$\Big(u(S) - \underbrace{(p_L\, u(S_L) + p_R\, u(S_R))}_{\text{expected uncertainty after split}} \Big) \times |S|$$

Benefit of a split: example

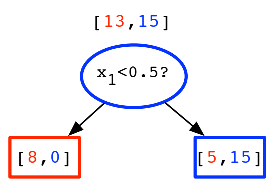


Initial uncertainty (Gini):

$$2 \times \frac{13}{28} \times \frac{15}{28}$$



$$p_L u_L + p_R u_R = \frac{4}{28} \cdot 0 + \frac{24}{28} \cdot 2 \cdot \frac{9}{24} \cdot \frac{15}{24} = \frac{45}{112}$$



$$p_L u_L + p_R u_R = \frac{8}{28} \cdot 0 + \frac{20}{28} \cdot 2 \cdot \frac{5}{20} \cdot \frac{15}{20} = \frac{30}{112}$$

Building a decision tree

- Start with a single node containing all data points
- Repeat:
  - Look at all current leaves and all possible splits
  - Choose the split with the greatest benefit

When to stop?

- When each leaf is pure?
- When the tree is already pretty big?
- When each leaf has uncertainty below some threshold?

Common strategy: keep going until leaves are pure.
Then, shorten the tree by **pruning**, to correct for overfitting.

What is overfitting?

Data comes from an unknown, underlying distribution D on $\mathcal{X} \times \mathcal{Y}$.
All we ever see are samples from D.

For a data set $(x_1, y_1), \ldots, (x_n, y_n)$, the **training error** of a classifier $h : \mathcal{X} \to \mathcal{Y}$ is

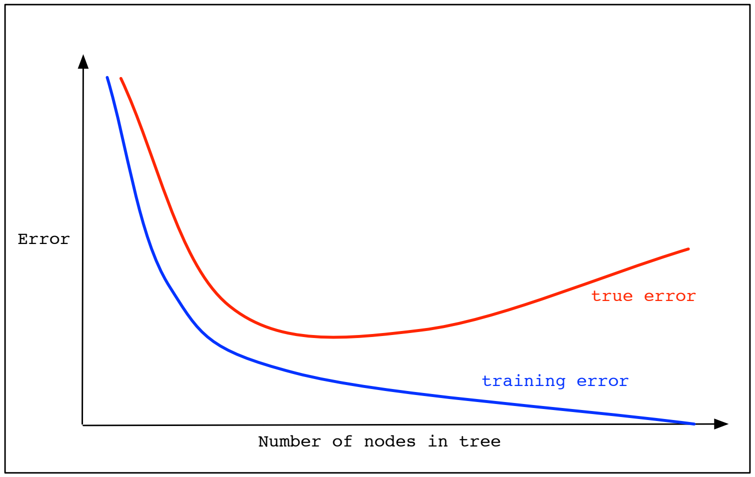
$$\widehat{\text{err}}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}(h(x_i) \neq y_i).$$

What we really care about is the **true error** of h with respect to D:

$$\text{err}(h) = \Pr_{(x,y)\sim D}(h(x) \neq y).$$

How are these two quantities related?

Overfitting: picture

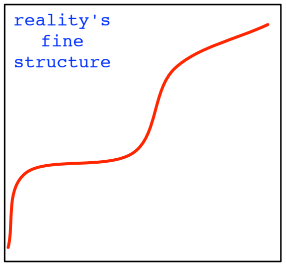


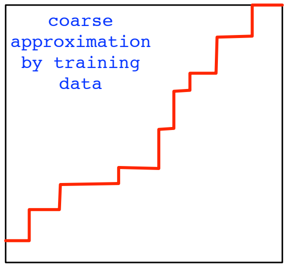
As we make our tree more and more complicated:

- training error keeps going down
- but, at some point, true error starts increasing!

Overfitting: perspectives

- The true underlying distribution D is the one whose structure we would like to capture.
- The training data reflects the structure of D, so it helps us.
- But it also has chance structure of its own – we must avoid modeling this.





Another perspective: it is absurd to fit a line to a point.
More generally, it is not good to use a model that is so complex that there isn't enough data to reliably estimate its parameters.

Decision tree construction and pruning

1. Split the training set into two parts
   - A smaller training set S
   - A validation set V (surrogate test set, model of reality)
2. Build a full decision tree using S
3. Then prune using V
   Use dynamic programming to find the pruning that does best on V

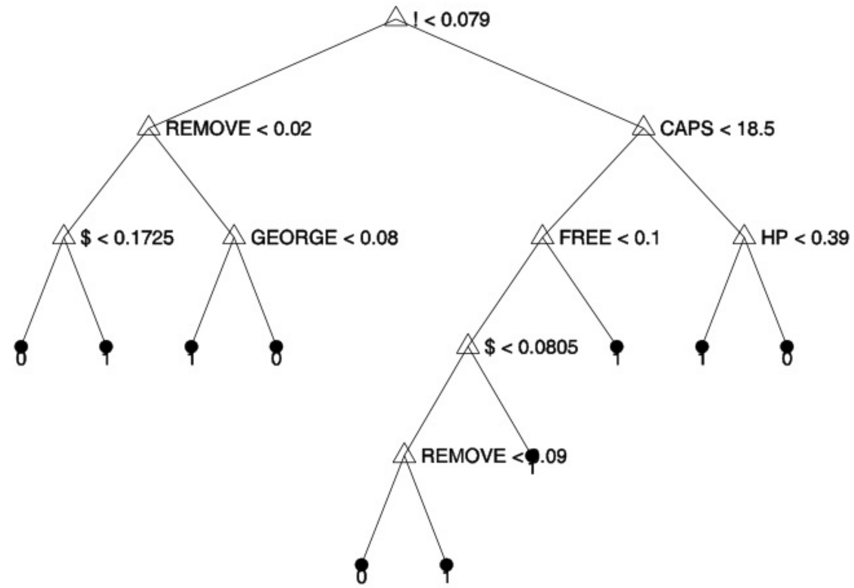
Of course, V can have chance structure too — but its chance structure is unlikely to coincide with that of S.

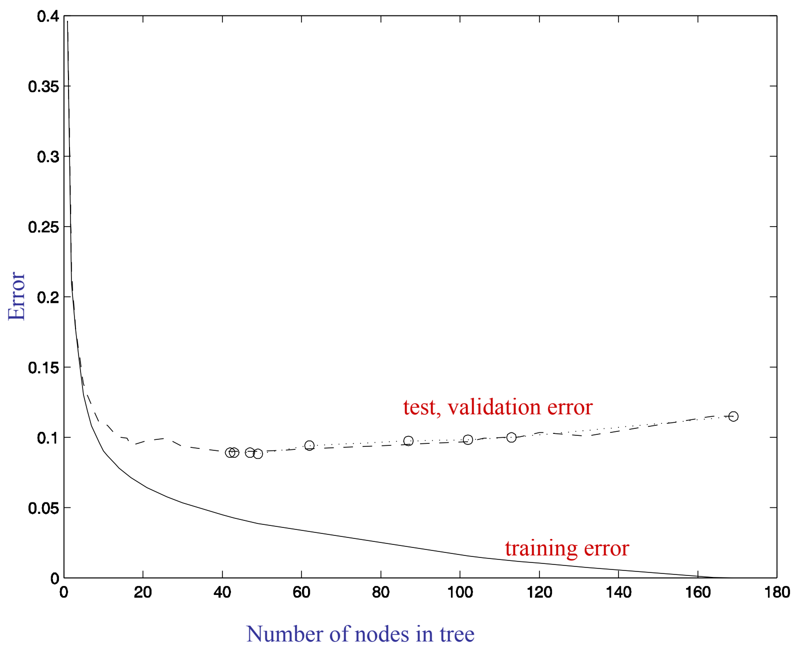
SPAMbase data set

- 4601 email messages, labeled SPAM or NOT SPAM.
- 39.4% are SPAM.
- Each email is represented by 57 features.
  - 48 check for specific words, e.g. FREE
  - 6 check for specific characters, e.g. !
  - 3 others, e.g. longest run of capitals

Randomly divide into three parts:

50% training data
25% validation set
25% test set





How accurate is the validation error?

For any classifier h and underlying distribution D on $\mathcal{X} \times \mathcal{Y}$:

true error $\quad \text{err}(h) = \Pr_{(x,y)\sim D}(h(x) \neq y)$

error on set A $\quad \text{err}(h, A) = \frac{1}{|A|} \sum_{(x,y)\in A} \mathbf{1}(h(x) \neq y)$

Suppose A is chosen i.i.d. (independent, identically distributed) from D. Then (over the random choice of A),

$$\mathbb{E}[\text{err}(h, A)] = \text{err}(h)$$

and the standard deviation of $\text{err}(h, A)$ is roughly $1/\sqrt{|A|}$.

Caution! In this scenario:

- Set A is used to assess a single, prespecified classifier h.
- If h was created using A as a training set, the result does not apply. In such situations, $\text{err}(h, A)$ could be a very poor estimate of $\text{err}(h)$.