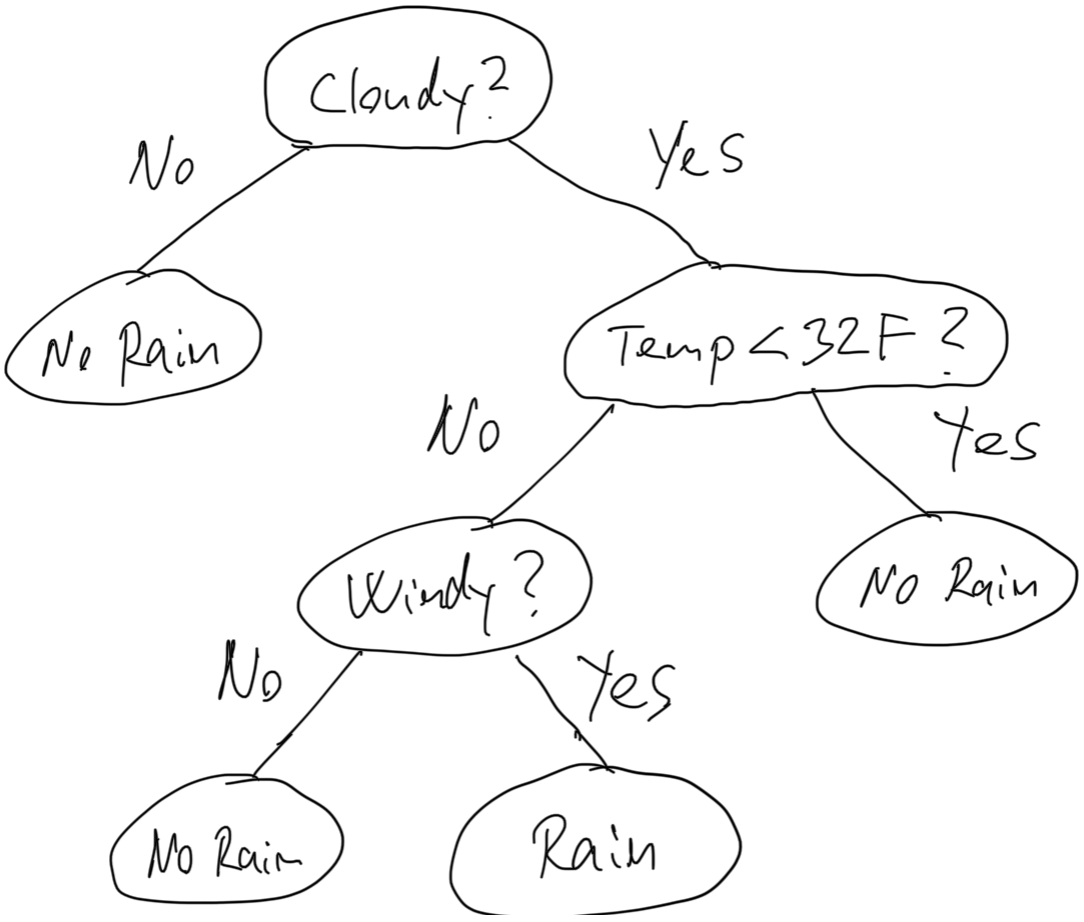
Decision trees

- Decision trees are popular types of predictors
- They are often used as weak learners in conjuction with a boosting algorithm

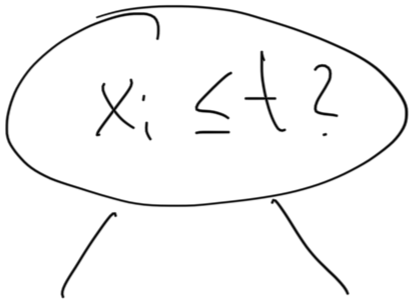
Example: Is it going to rain today?



- Decision tree is a rooted binary tree:
  1) Internal nodes are predicates (i.e. questions with yes/No answers)
  2) Edge to a left subtree corresponds to "No"
  3) Edge to a right subtree correspond to "Yes"
  4) Leaves are predictions (i.e. contant classifiers)

- In principle the predicates in internal nodes can be anything.
- Actual implementations use

predicates of the form



where an example lies in $\mathbb{R}^d$

$$X = (x_1, \ldots, x_d)$$

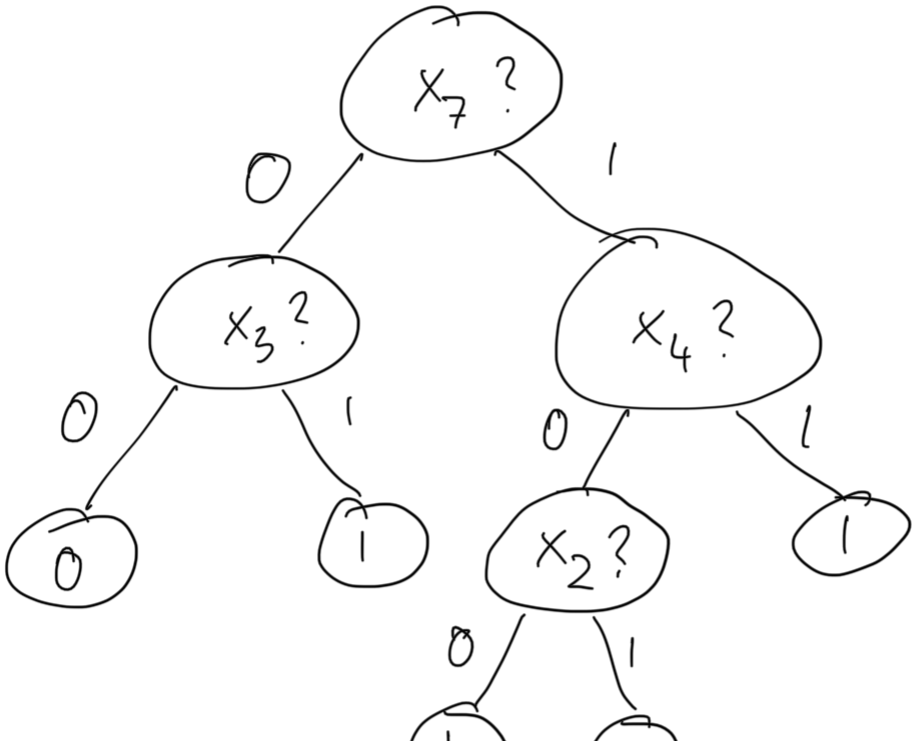
and $i \in \{1, 2, \ldots, d\}$ and $a \in \mathbb{R}$.

- Suppose S is a labeled sample

$$S = \{(x_1, y_1), \ldots, (x_1, y_m)\}$$

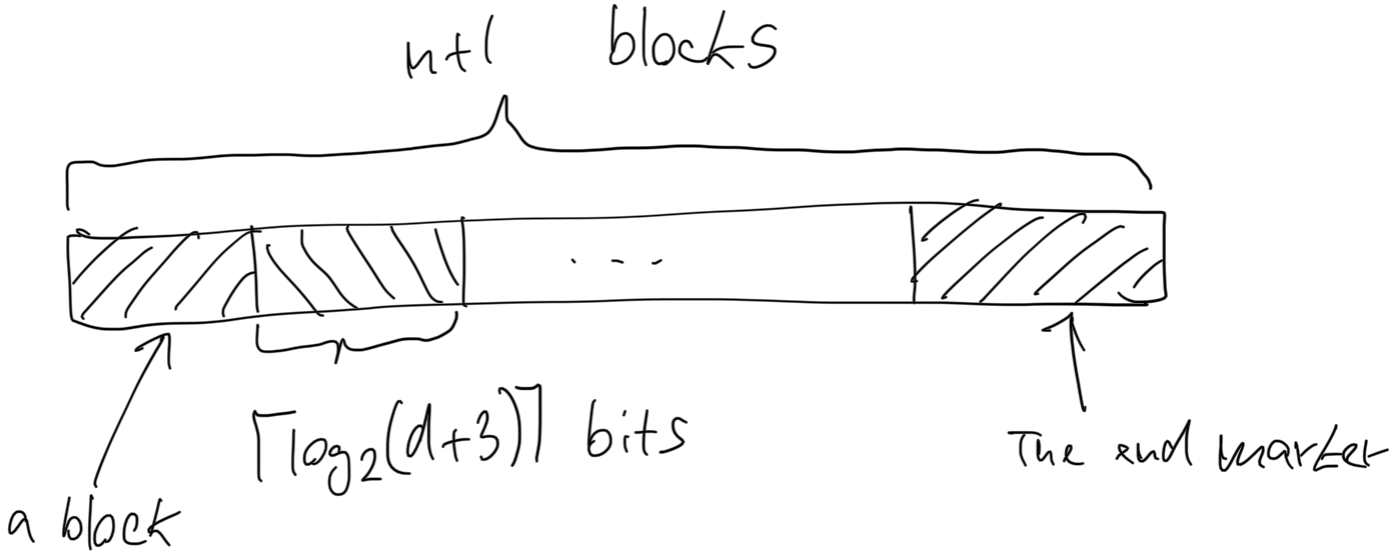
- If not both $(x, 1), (x, 0) \in S$ for any x, then there exists a decision tree with 0 empirical error.

- This tree might be very large. It can have as many as m leaves. Its generalization error will likely be very large.
- To avoid overfitting, we need to construct "small" trees.
- For simplicity assume $X = \{0,1\}^d$ i.e. features are binary

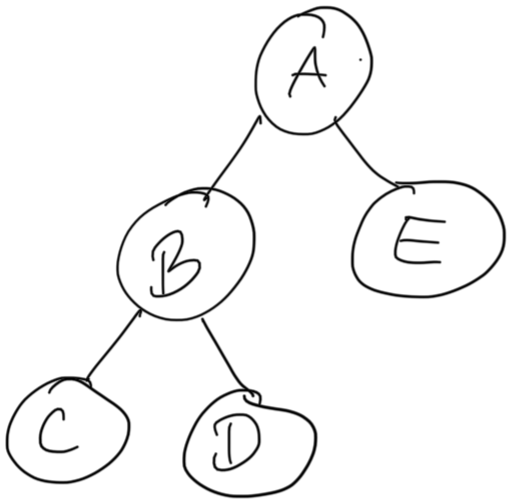


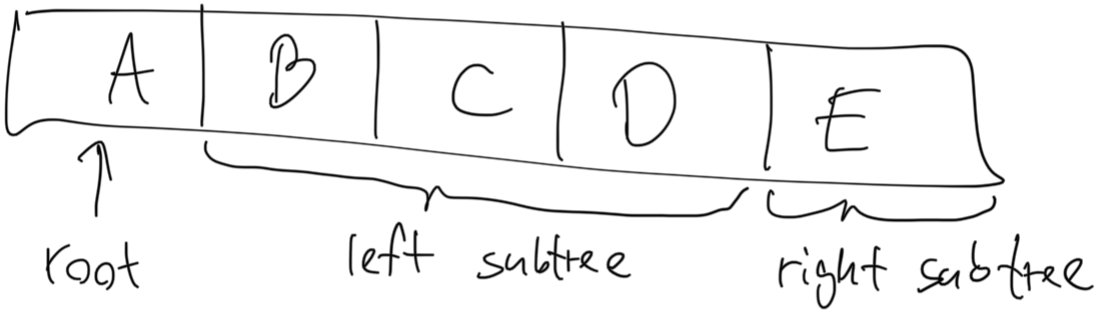
(1) (0)

- We describe such decision trees with prefix-free code.
- Suppose we have tree with n nodes (internal + leaves)
- We will use code consisting of $n+1$ blocks, each consisting of $\lceil \log_2(d+3) \rceil$ bits



- First n blocks correspond to nodes of the tree in pre-order:





- Each block encodes a number $i \in \{1, 2, \ldots, d, d+1, d+2, d+3\}$ in binary:

1) if $i \in \{1, \ldots, d\}$, it encodes internal node (x_i?).

2) $d+1$ encodes leaf with value 0

3) $d+2$ encodes leaf with value 1

4) $d+3$ is the end marker

Generalization bound:

Let $X = \{0,1\}^d$ and $Y = \{0,1\}$.

Let D be a distribution over $X \times Y$.

Let S be an i.i.d. sample from D of size m.

Let $\delta \in (0,1)$. With probability at least $1-\delta$, for any decision tree h

$$err_D(h) \leq \widehat{err_S}(h) + \sqrt{\frac{(m+1)\lceil \log(d+3) \rceil + \log(1/\delta)}{2m}}.$$

Proof:

- Follows from result from lecture about non-uniform learning.

- Given a labeled sample S, finding a decision tree h that minimizes

$$\widehat{err}_S(h) + \sqrt{\frac{(n+1)\lceil \log_2(d+3) \rceil + \ln\left(\frac{1}{\delta}\right)}{2m}}$$

is NP-hard.

- Instead people use various fast heuristic algorithms.
- These algorithms are fast, but they are not guaranteed to find the optimal solution.

- Many heuristics depend on a function

$$Gain(S, i)$$

labeled sample

feature to split on

that "evaluates" how good is split on feature i

Algorithm ID3 (Iterative Dichotomizer 3)

Inputs

- Set of attributes $A \subseteq \{1, 2, \ldots, d\}$
- Labeled sample $S \in (X \times Y)^*$

$(X = \{0,1\}^d, \; Y = \{0,1\})$

Algorithm

- If all examples in S are labeled 0, output leaf (0)
- If all examples in S are labeled 1, output leaf (1)
- If $A = \emptyset$ output leaf (m) where $m \in \{0,1\}$ is the majority label in S.
- Find $j = \underset{i \in A}{\text{argmax}}\ \text{Gain}(S, i)$
- Split S into

$$S_0 = \{(x,y) \in S : x_j = 0\}$$

$$S_1 = \{(x,y) \in S : x_j = 1\}$$

- Remove j from A:

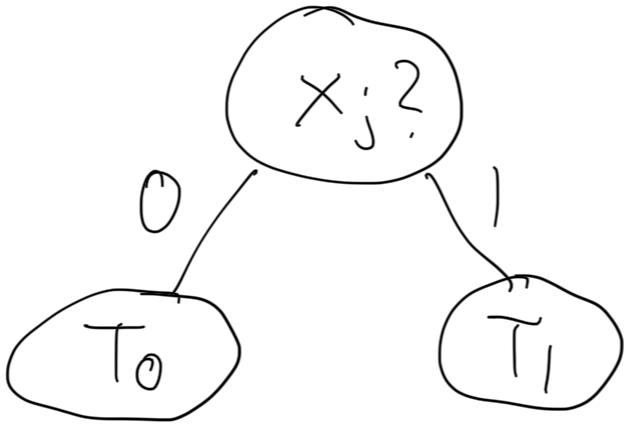
$$A' = A \setminus \{j\}$$

- Recursively call ID3:

trees → $T_0 = ID3(A', S_0)$

trees → $T_1 = ID3(A', S_1)$

- Return



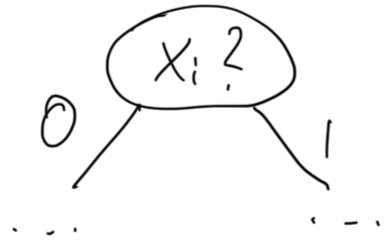
Gain Functions

- Let P_S be the uniform distribution over S

- Train error:
  - Let $C(a) = \min(a, 1-a)$ for any $a \in [0,1]$.
  - With a single leaf node we can achieve empirical error:

$$C\left(\underbrace{P_S[Y=1]}_{\text{fraction of positive examples}}\right)$$

  - After split on X_i, we can achieve empirical error



$$P_S[X_i=1] \cdot C\left(P_S[Y=1 \mid X_i=1]\right)$$

$$+ P_S[X_i=0] \cdot C\left(P_S[Y=1 \mid X_i=0]\right)$$

- $\text{Gain}(S, i)$ = difference of the two

$$= C(P_S[Y=1]) - P_S[X_i=1] \cdot C(P_S[Y=1 \mid X_i=1]) - P_S[X_i=0] \cdot C(P_S[Y=1 \mid X_i=0])$$

Information gain:

- Use $C(a) = -a \log a - (1-a)\log(1-a)$

entropy of Bernoulli variable with parameter a

Gini index:

- Use $C(a) = 2a(1-a)$

Pruning

- Trees produced by ID3 can be still fairly large

- The trees are pruned using a function that estimates generalization error based on empirical error, sample size and complexity of the tree.

Pruning algorithm:

- Parameters: $f(T)$ estimate of generalization error, decision tree T

- Algorithm:

- Enumerate nodes in reverse order of their depth.
- For each node x:

  1) Consider subtree T_x rooted at x

  2) Consider replacing T_x with

     a) Leaf (0)

     b) Leaf (1)

     c) Left subtree of T_x

     d) Right subtree of T_x

     e) T_x itself

  3) Pick a, b, c, d, e based on f

Decision trees for $\mathbb{R}^d$

- Suppose $X = \mathbb{R}^d$ and $Y = \{0, 1\}$

- Suppose $S = \left((x_1, y_1) \dots (x_m, y_m) \right)$

  $S \in (X \times Y)^m$

- For each feature $i \in \{1, 2, \dots, d\}$ consider m thresholds

$t \in \{ x_{i,1}, x_{i,2}, \dots, x_{i,m} \}$