Decision Trees

Review

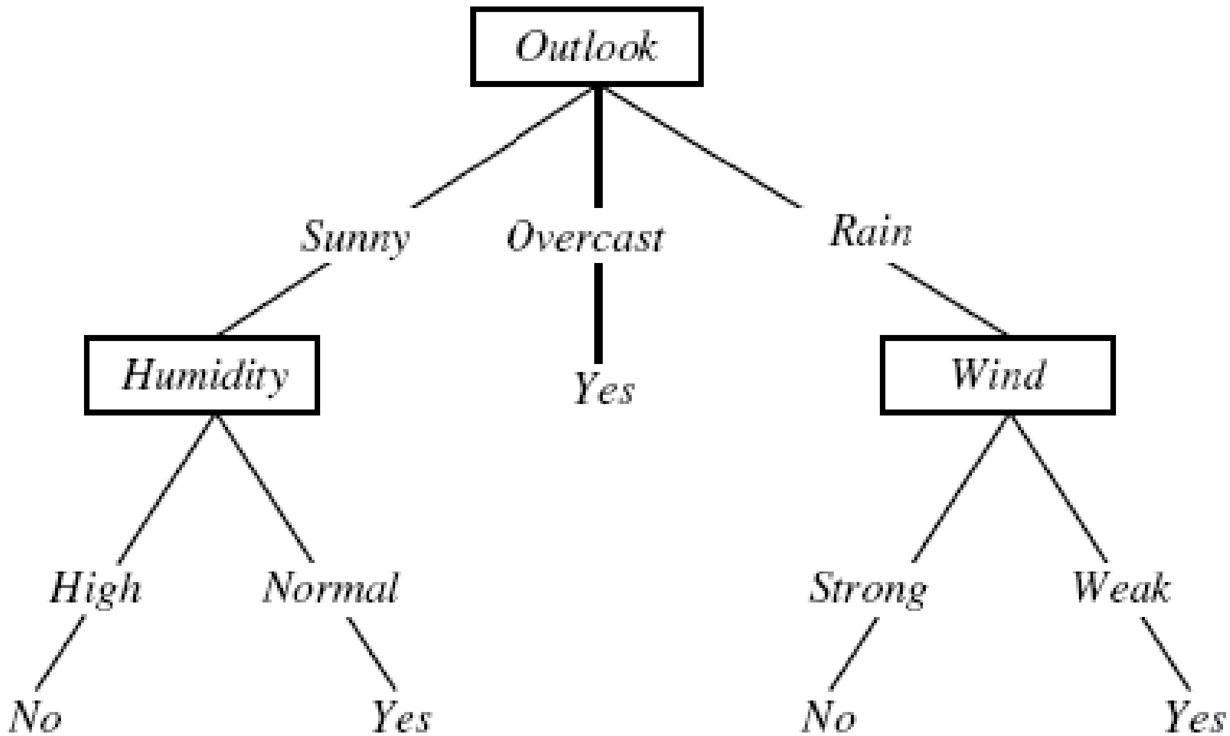
- We introduced three styles of algorithms for linear classifiers:
  - 1) Perceptron – learn classifier direction
  - 2) Logistic Regression – learn P(Y | X)
  - 3) Linear Discriminant Analysis – learn P(X,Y)

- Linear separability
  - A data set is linearly separable if there exists a linear hyperplane that separates positive examples from negative examples

Not linearly separable data

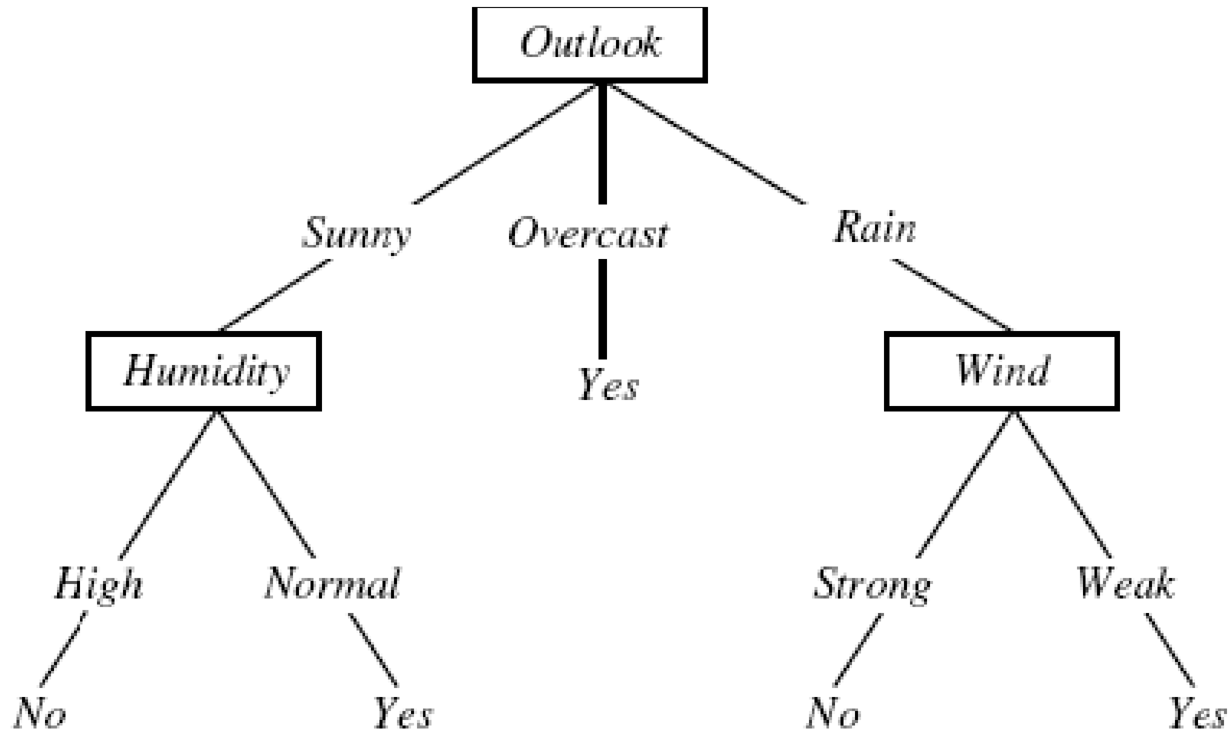
- Some data sets are not linearly separable!
- Option 1
  - Use non-linear features, e.g., polynomial basis functions
  - Learn linear classifiers in the non-linear feature space
  - Will discuss more later
- Option 2
  - Use non-linear classifiers (decision trees, neural networks, nearest neighbors etc.)

Decision Tree for Playing Tennis



Prediction is done by sending the example down the tree till a class assignment is reached

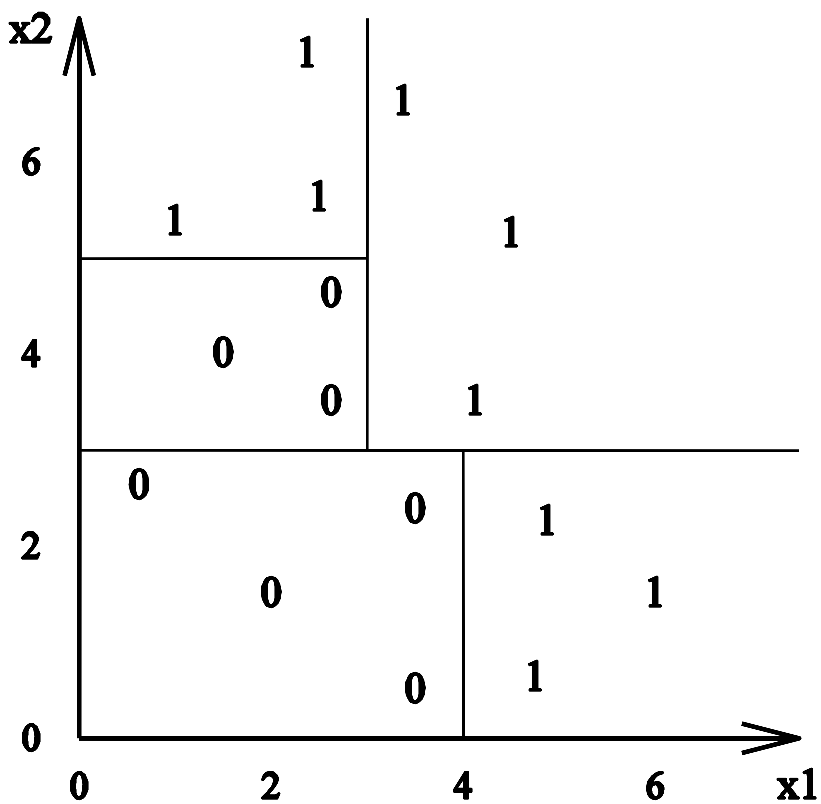
Definitions

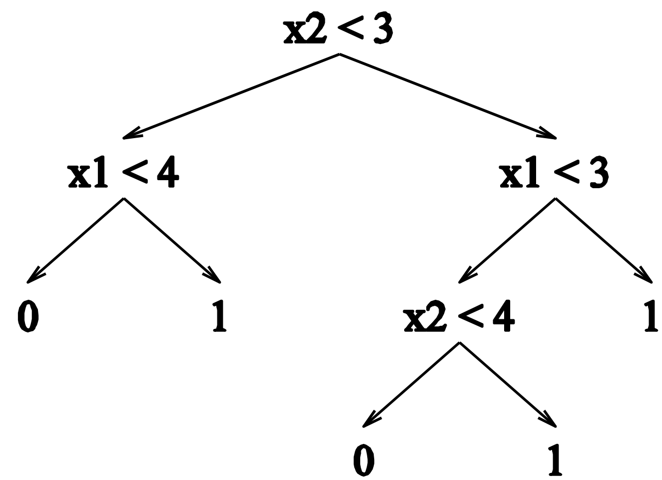


- **Internal nodes**
  - Each test a feature
  - Branch according to feature values
  - Discrete features – branching is naturally defined
  - Continuous features – branching by comparing to a threshold
- **Leaf nodes**
  - Each assign a classification

Decision Tree Decision Boundaries

- Decision Trees divide the feature space into axis-parallel rectangles and label each rectangle with one of the K classes



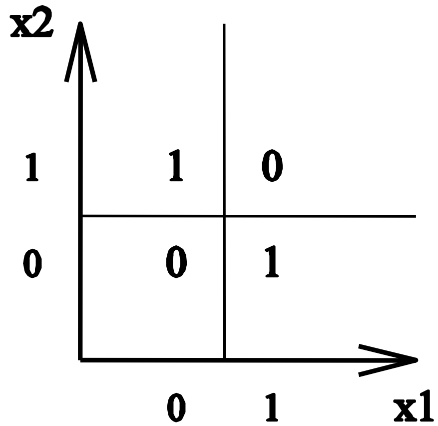


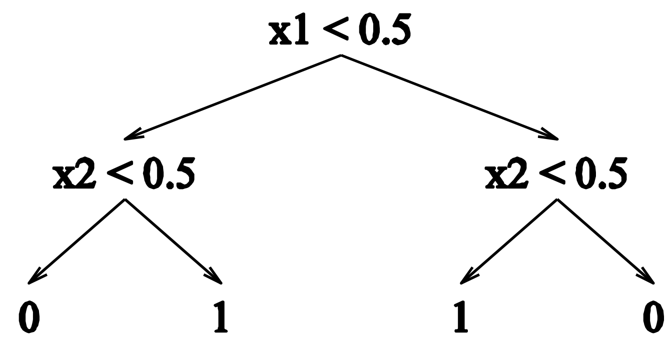
Hypothesis Space of Decision Trees

- Decision trees provide a very popular and efficient hypothesis space
  - Deal with both **Discrete and Continuous** features
  - **Variable size**: as the # of nodes (or depth) of tree increases, the hypothesis space grows
    - Depth 1 (“decision stump”) can represent any Boolean function of one feature
    - Depth 2: Any Boolean function of two features and some Boolean functions involving three features:
    - In general, can represent any Boolean functions

Decision Trees Can Represent Any Boolean Function

XOR





- If a target Boolean function has *n* inputs, there always exists a decision tree representing that target function.
- However, in the worst case, exponentially many nodes will be needed (why?)
  - 2^n possible inputs to the function
  - In the worst case, we need to use one leaf node to represent each possible input

Learning Decision Trees

- Goal: Find a decision tree **h** that achieves minimum misclassification errors on the training data
- A trivial solution: just create a decision tree with one path from root to leaf for each training example
  - Bug: Such a tree would just memorize the training data. It would not generalize to new data points
- Solution 2: Find the <u>smallest</u> tree *h* that minimizes error
  - Bug: This is NP-Hard

Top-down Induction of Decision Trees

There are different ways to construct trees from data. We will focus on the top-down, greedy search approach:

Basic idea:

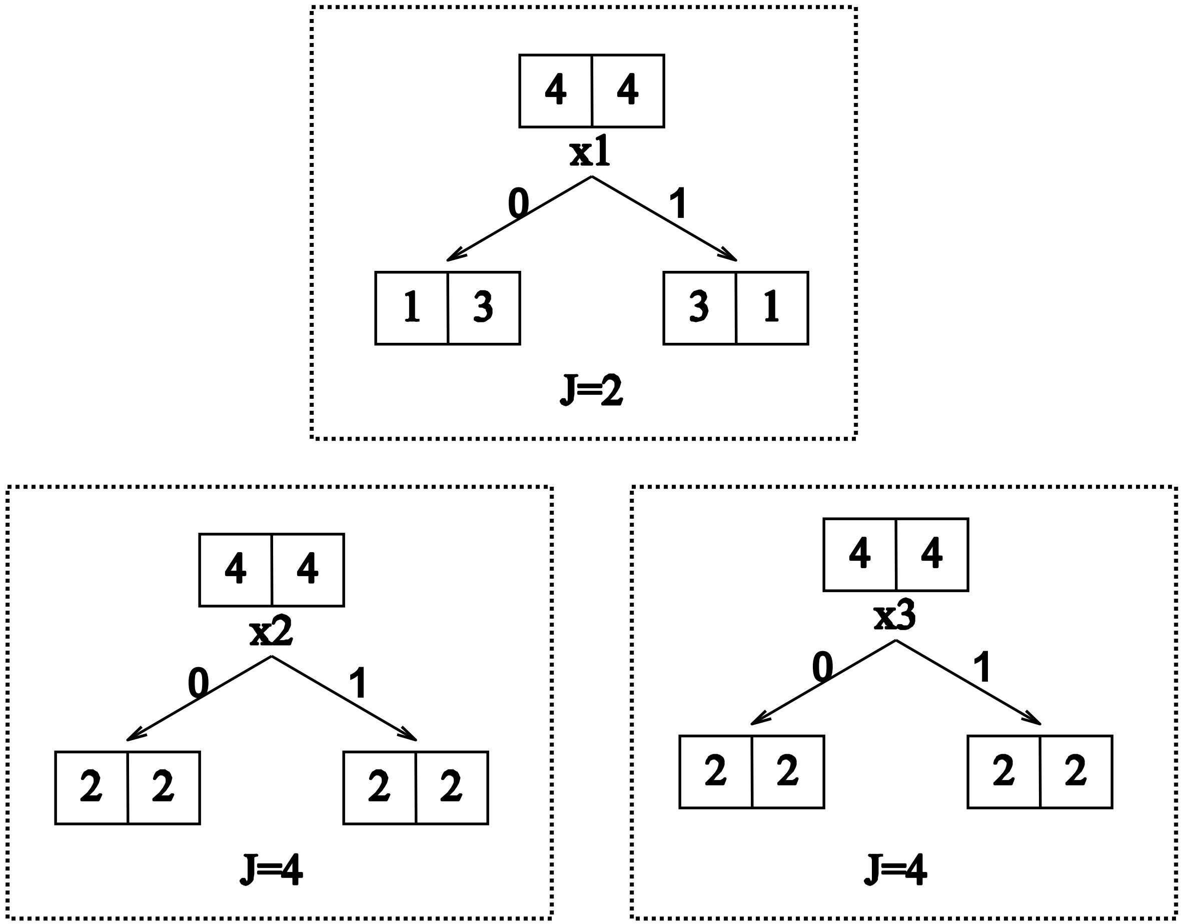
1. Choose the **best** feature a* for the root of the tree.
2. Separate training set **S** into subsets {$\mathbf{S}_1$, $\mathbf{S}_2$, .., $\mathbf{S}_k$} where each subset $\mathbf{S}_i$ contains examples having the same value for a*.
3. Recursively apply the algorithm on each new subset until all examples have the same class label.

Choosing Feature Based on Classification Error

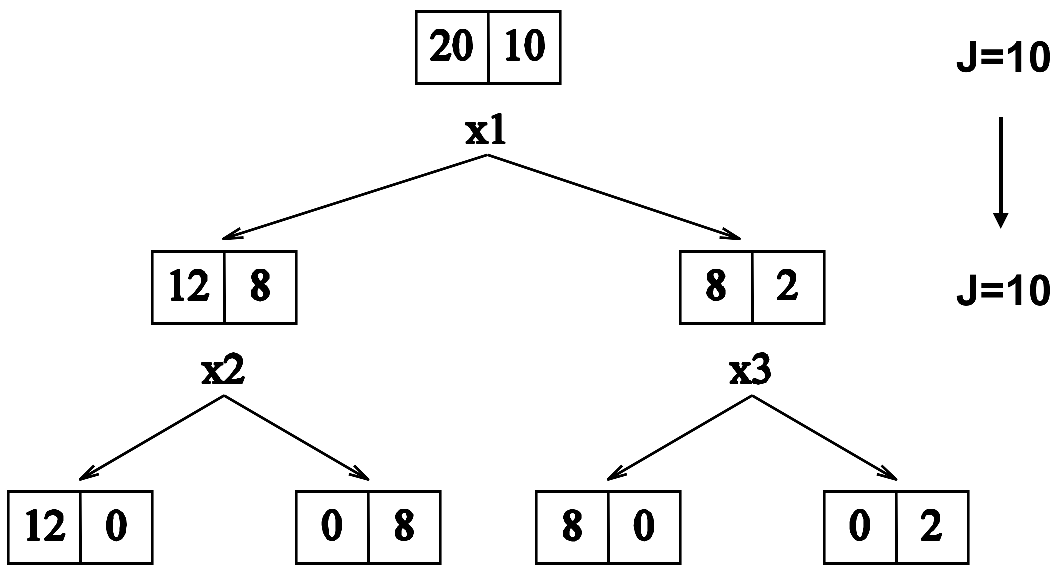
- Perform 1-step look-ahead search and choose the attribute that gives the lowest error rate on the training data

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

Training examples



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



Entropy

- Let *X* be a random variable with the following probability distribution

P(X = 0)	*P(X = 1)*
0.2	0.8

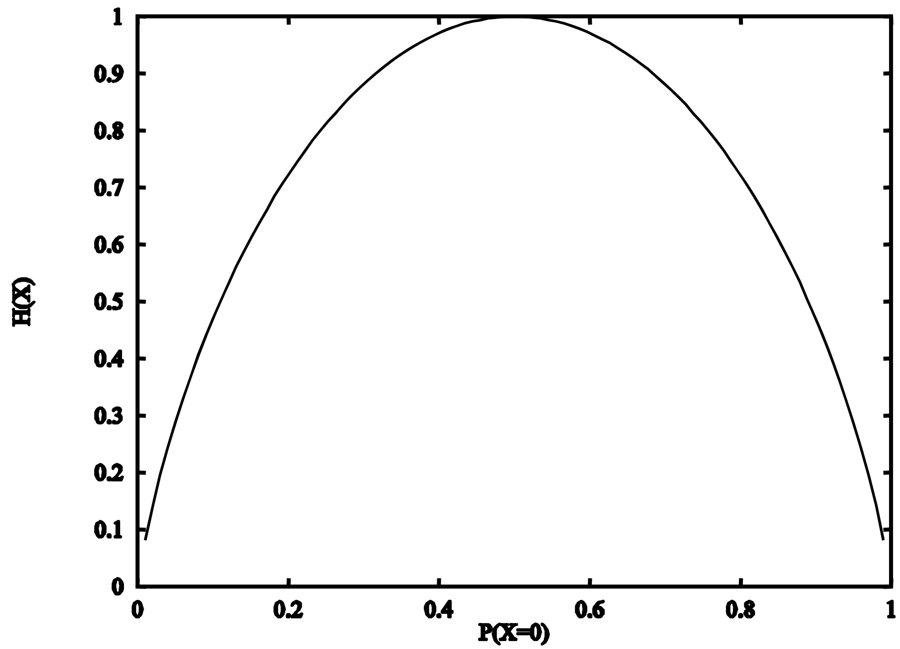
- The <u>entropy</u> of ***X***, denoted *H(**X**)*, is defined as

$$H(X) = -\sum_{x} P_X(x) \log_2 P_X(x)$$

- **Entropy** measures the uncertainty of a random variable
- The larger the entropy, the more uncertain we are about the value of *X*
- If P(X=0)=0 (or 1), there is no uncertainty about the value of X, entropy = 0
- If P(X=0)=P(X=1)=0.5, the uncertainty is maximized, entropy = 1

Entropy

- Entropy is a concave function downward



More About Entropy

- Joint Entropy

$$H(X,Y) = -\sum_x \sum_y P(X=x, Y=y) \log P(X=x, Y=y)$$

- Conditional Entropy is defined as

$$H(Y \mid X) = \sum_x P(X=x) H(Y \mid X=x)$$
$$= -\sum_x P(X=x) \sum_y P(Y=y \mid X=x) \log P(Y=y \mid X=x)$$

  – The average surprise of Y when we know the value of X

- Entropy is additive

$$H(X,Y) = H(X) + H(Y \mid X)$$

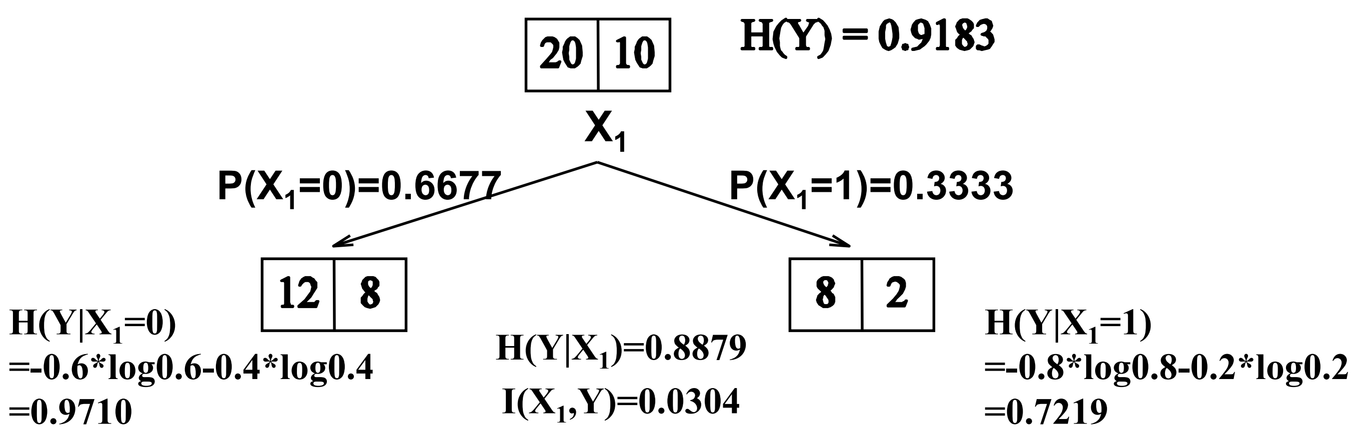
Mutual Information

- The <u>mutual information</u> between two random variables *X* and *Y* is defined as:

$$I(X,Y) = H(Y) - H(Y \mid X)$$

  – the amount of information we learn about *Y* by knowing the value of *X* (and vice versa – it is symmetric).

- Consider the class *Y* of each training example and the value of feature X_1 to be random variables. The mutual information quantifies how much X_1 tells us about *Y*.



Choosing the Best Feature

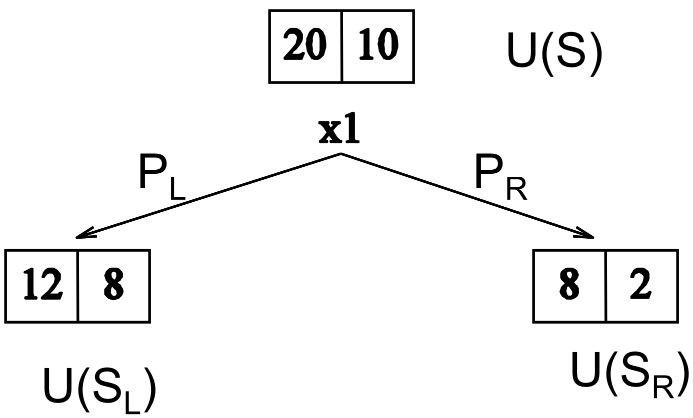
- Choose the feature X_j that has the highest mutual information with Y - often referred to as the **information gain** criterion

$$\begin{aligned}\arg\max_j I(X_j;Y) &= \arg\max_j H(Y) - H(Y \mid X_j) \\ &= \arg\min_j H(Y \mid X_j)\end{aligned}$$

- Define $\widetilde{J}(j)$ to be the expected remaining uncertainty about y after testing x_j

$$\widetilde{J}(j) = H(Y \mid X_j) = \sum_x P(X_j = x) H(Y \mid X_j = x)$$

Choosing the Best Feature: A General View



Benefit of split =

$$U(S) - [\underbrace{P_L{*}U(S_L)+P_R{*}U(S_R)}_{\text{Expected Remaining Uncertainty (Impurity)}}]$$

Measures of Uncertainty	
Error	$\min\{p, 1-p\}$
Entropy	$-p\log p - (1-p)\log 1 - p$
Gini Index	$2p(1-p)$

Multi-nomial Features

- Multiple discrete values
  - Method 1: Construct multi-way split
    - Information Gain will tend to prefer multi-way split
    - To avoid this bias, we rescale the information gain:

Correction: the correct objective should be argmax I(X_j, Y)/H(X_j)

$$\arg\min_j \frac{H(Y \mid X_j)}{H(X_j)} = \arg\min_j \frac{\sum_x P(X_j = x) H(Y \mid X_j = x)}{-\sum_x P(X_j = x) \log P(X_j = x)}$$

  - Method 2: Test for one value versus all of the others
  - Method 3: Group the values into two disjoint sets and test one set against the other

Continuous Features

- Test against a threshold
- How to compute the best threshold θ_j for X_j?
  - Sort the examples according to X_j.
  - Move the threshold θ from the smallest to the largest value
  - Select θ that gives the best information gain
  - Only need to compute information gain when class label changes

Continuous Features

- Information gain for θ = 1.2 is 0.2294

Y	0	0	1	0	1	1	0	1	1
X_j	0.2	0.4	0.7	1.1	1.3	1.7	1.9	2.4	2.9

$n_{0,L} = 3$	$n_{0,R} = 1$
$n_{1,L} = 1$	$n_{1,R} = 4$

- Information gain only needs to be computer when class label changes

Y	0	0	1	0	1	1	0	1	1
X_j	0.2	0.4	0.7	1.1	1.3	1.7	1.9	2.4	2.9

Top-down Induction of Decision Trees

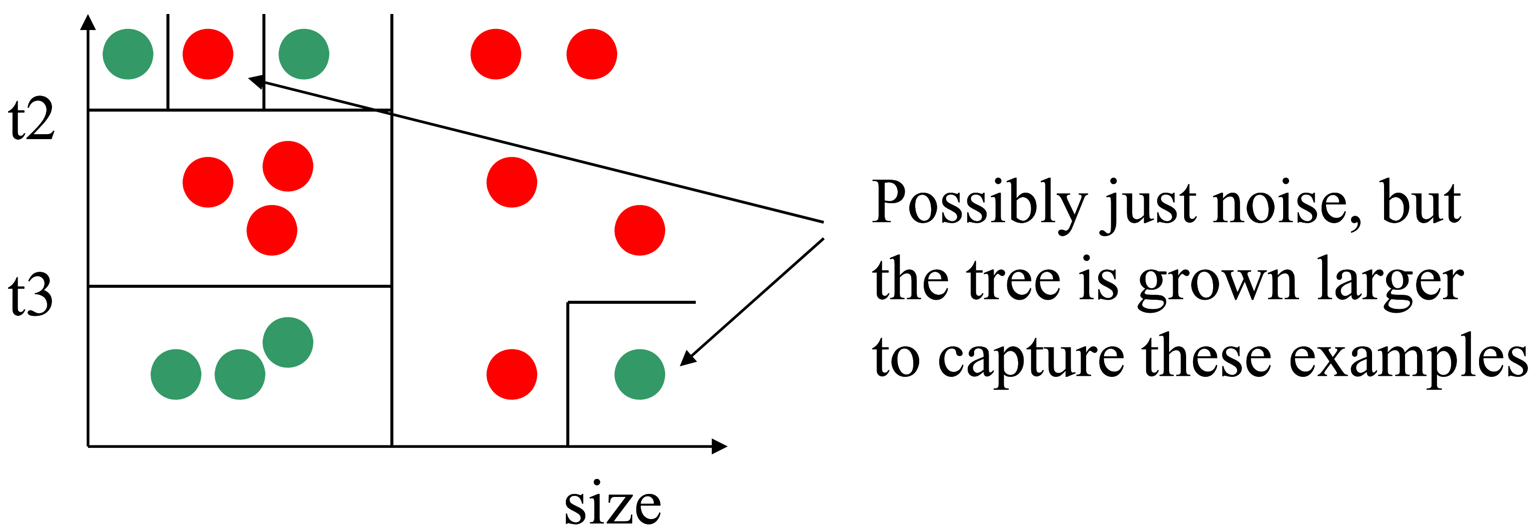
There are different ways to construct trees from data. We will focus on the top-down, greedy search approach:

Basic idea:

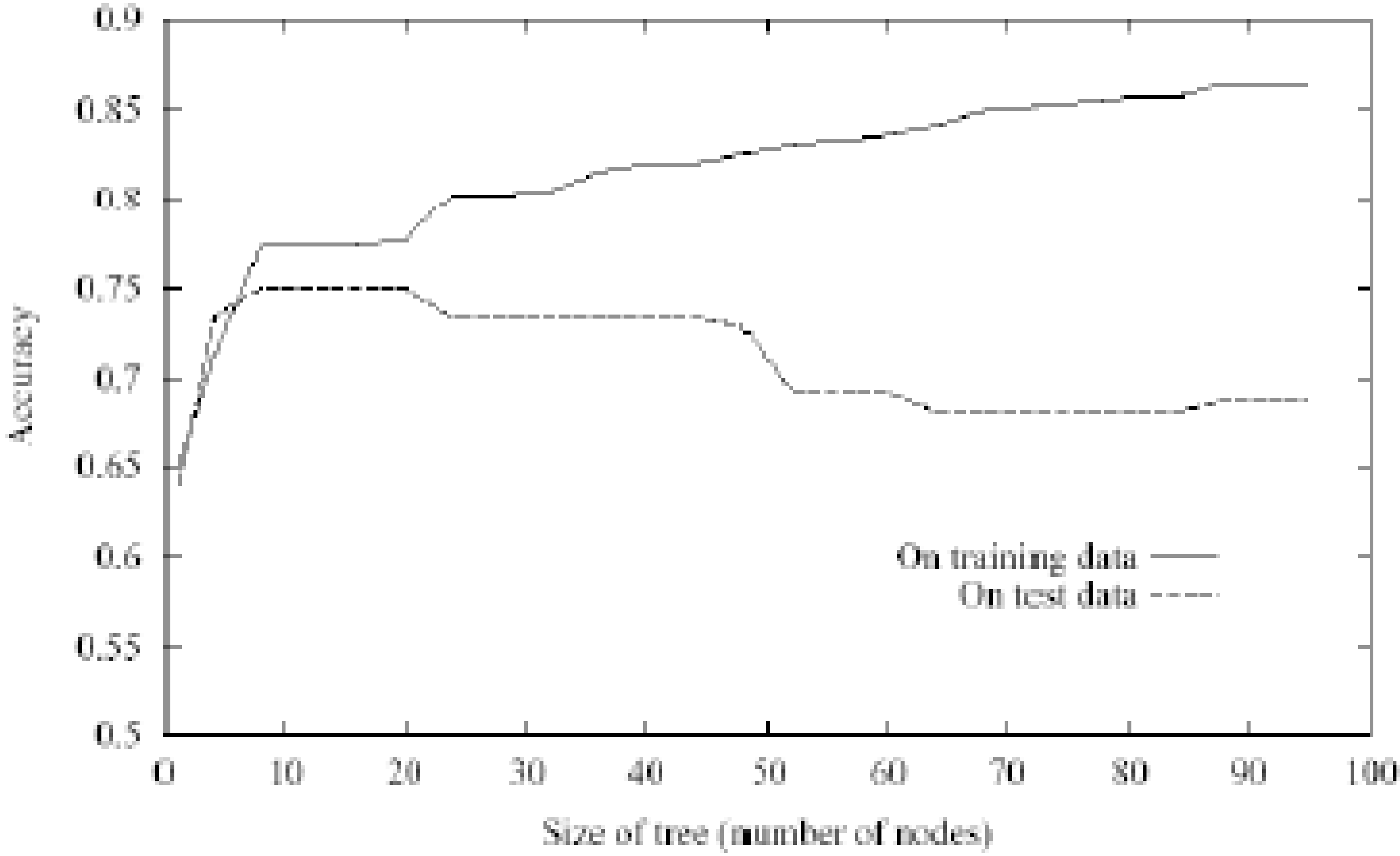
1. Choose the **best** feature a* for the root of the tree.
2. Separate training set **S** into subsets {$\mathbf{S}_1$, $\mathbf{S}_2$, .., $\mathbf{S}_k$} where each subset $\mathbf{S}_i$ contains examples having the same value for a*.
3. Recursively apply the algorithm on each new subset until all examples have the same class label.

Over-fitting

- Decision tree has a very flexible hypothesis space
- As the nodes increase, we can represent arbitrarily complex decision boundaries – training set error is always zero
- This can lead to over-fitting



Over-fitting

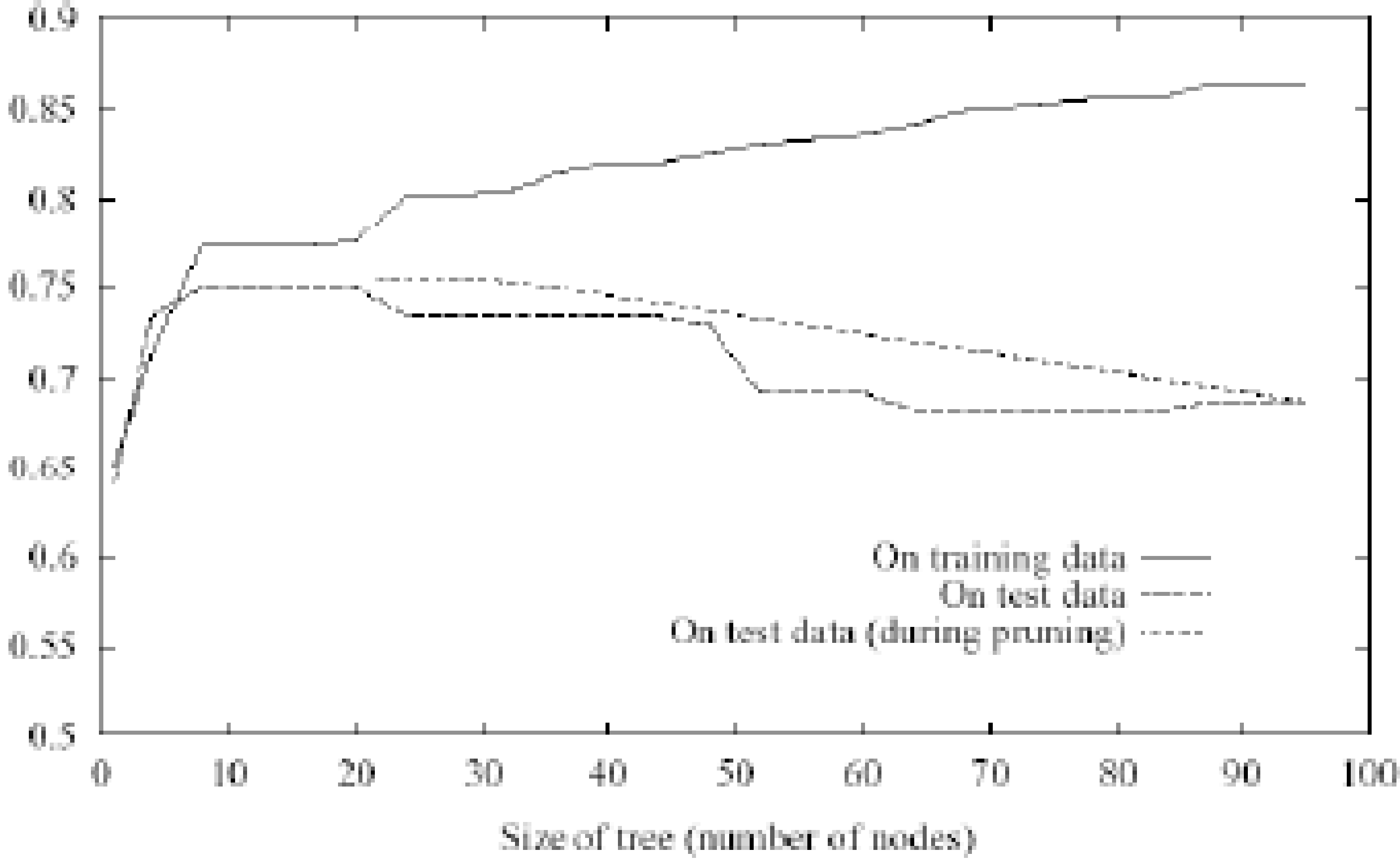


Avoid Overfitting

- Early stop
  - Stop growing the tree when data split does not differ significantly different from random split

- Post pruning
  - Separate training data into **training set** and **validating set**
  - Evaluate impact on validation set when pruning each possible node
  - Greedily remove the one that most improve the validation set performance

Effect of Pruning



Decision Tree Summary

- DT - one of the most popular machine learning tools
  – Easy to understand
  – Easy to implement
  – Easy to use
  – Computationally cheap
- Information gain to select features (ID3, C4.5 …)
- DT over-fits
  – Training error can always reach zero
- To avoid overfitting:
  – Early stopping
  – Pruning