Decision Trees

Off the shelf classifier

- A method that can be applied directly to the data without the need for preprocessing or tuning of learning algorithm
- So far
  - Perceptron – Directly learns the classifier
  - Logistic Regression – Discriminative
  - LDA - Generative

Criteria

- “Mixed” data types
- Missing values
  - Missing at random
  - Missing for a cause
- Robustness to outliers
- Insensitive to Monotone transformation of features
- Scalability
- Irrelevant inputs
- Interpretability
- Predictive power

Summary

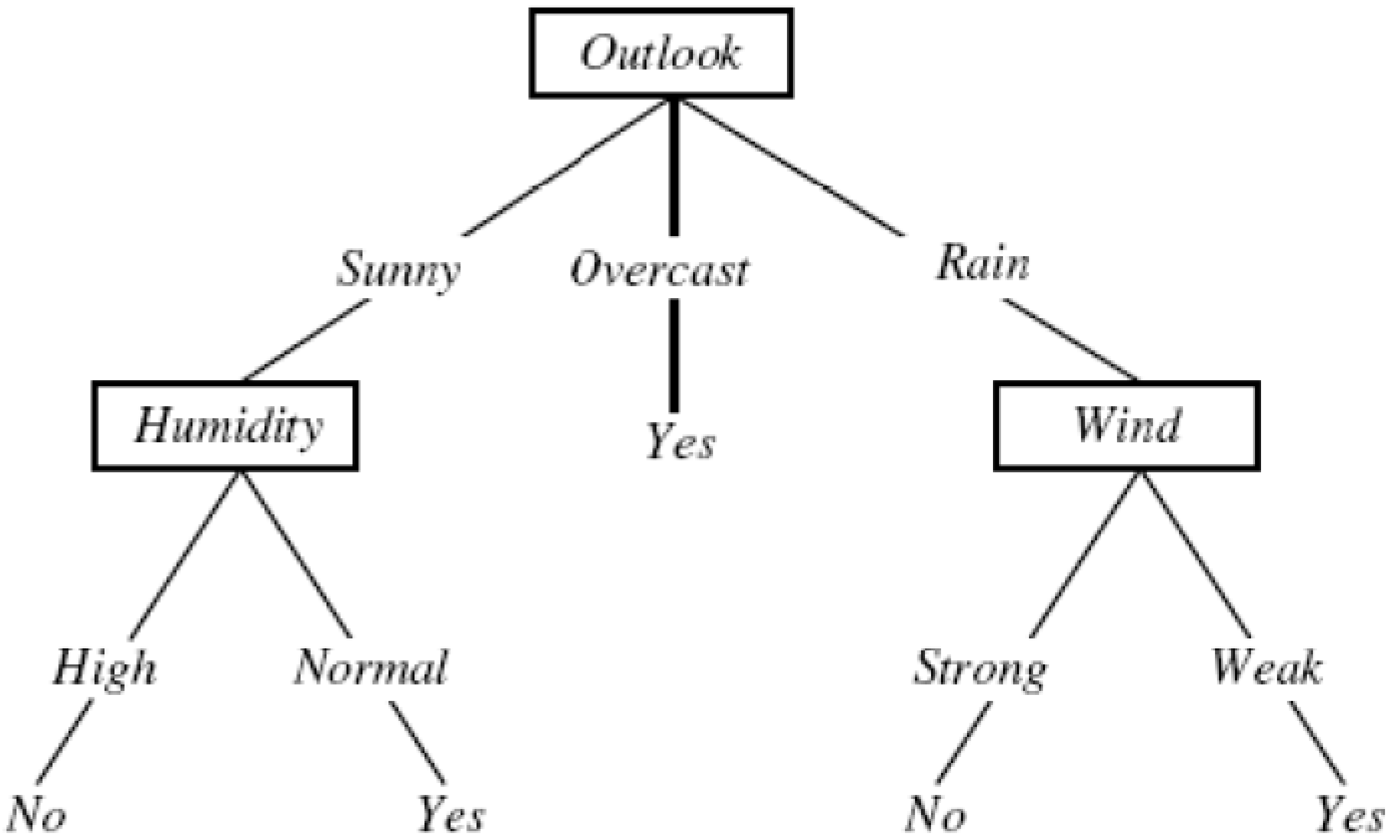
(This table will grow with the course)

Criterion	**Perceptron**	**Logistic**	**LDA**
Mixed data	N	N	N
Missing values	N	N	Y
Outliers	N	Y	N
Monotone	N	N	N
Scalability	Y	Y	Y
Irrelevant i/p	N	N	N
Interpretable	Y	Y	Y
Accurate	Y	Y	Y

Linear Separability

- A data set is linearly separable if there exists a hyperplane that separates pos examples from neg examples
- Many data sets in real world are not linearly separable!
- Two options
  - Use non-linear features and learn a linear classifier on this non-linear feature space. We will see a few such methods later
  - Use non-linear classifiers (decision trees, neural nets, nearest neighbors etc)

Decision Tree for Playing Tennis

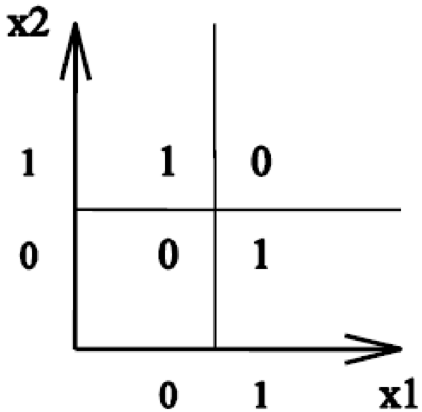


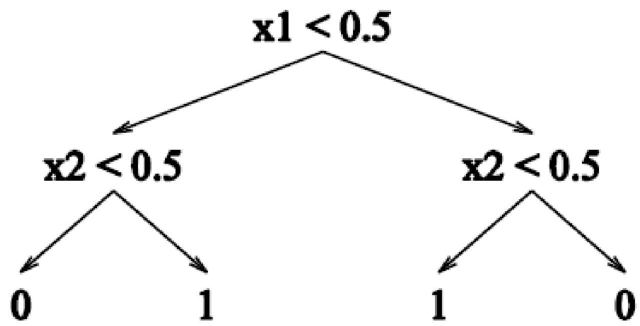
Decision Trees

- Decision tree representation:
  - Each internal node tests an attribute
  - Each branch corresponds to attribute value
  - Each leaf node assigns a classification
- How would we represent:
  - $\wedge$, $\vee$, XOR
  - $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
  - M of N

Decision Trees Can Represent Any Boolean Function

XOR

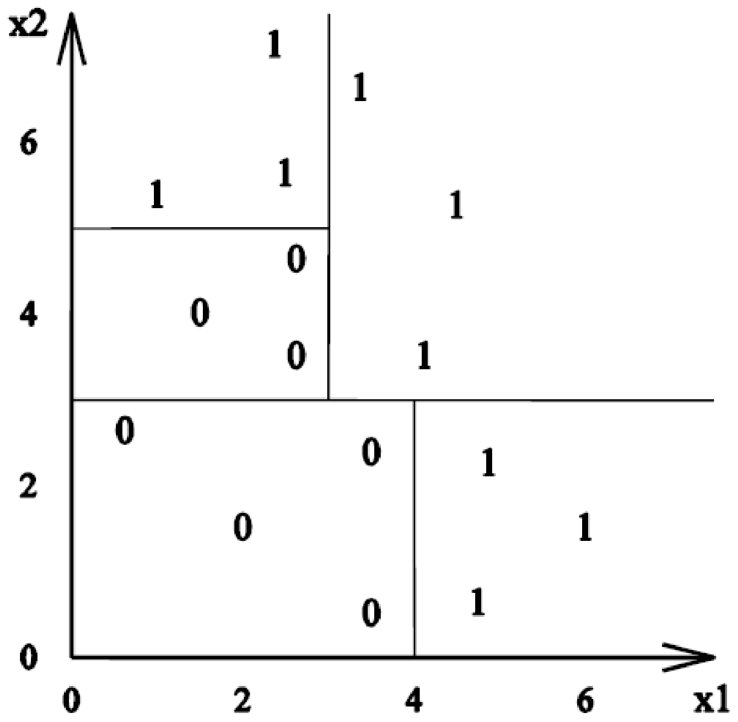


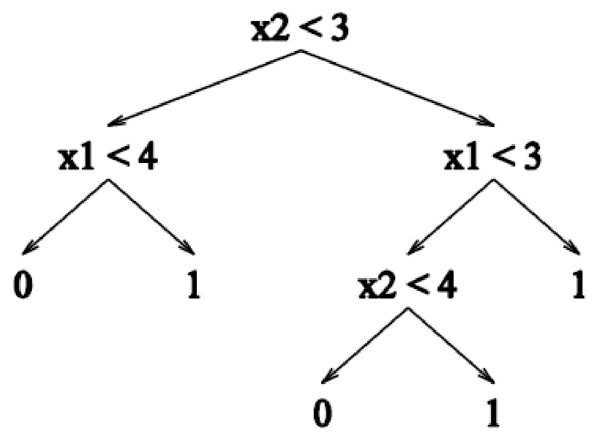


- If a target Boolean function has *n* inputs, there always exists a decision tree representing that target function.
- However, in the worst case, exponentially many nodes will be needed (why?)
  - 2^n possible inputs to the function
  - In the worst case, we need to use one leaf node to represent each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

Decision Tree Decision Boundaries

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





When do you want Decision trees?

- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data
- Need for interpretable model

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

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

There are different ways to construct these trees. We will now look at a top-down, greedy search approach

High-level Idea:

1. Choose the best feature f^* for the root of the tree.
2. Separate the training set into subsets $\{S_1, S_2, \ldots, S_k\}$ where each subset S_i contains examples that have the same value for f^*
3. Recursively apply the algorithm on each new subset until all examples have the same class label

Growing Trees

How to choose next feature to place in decision tree?

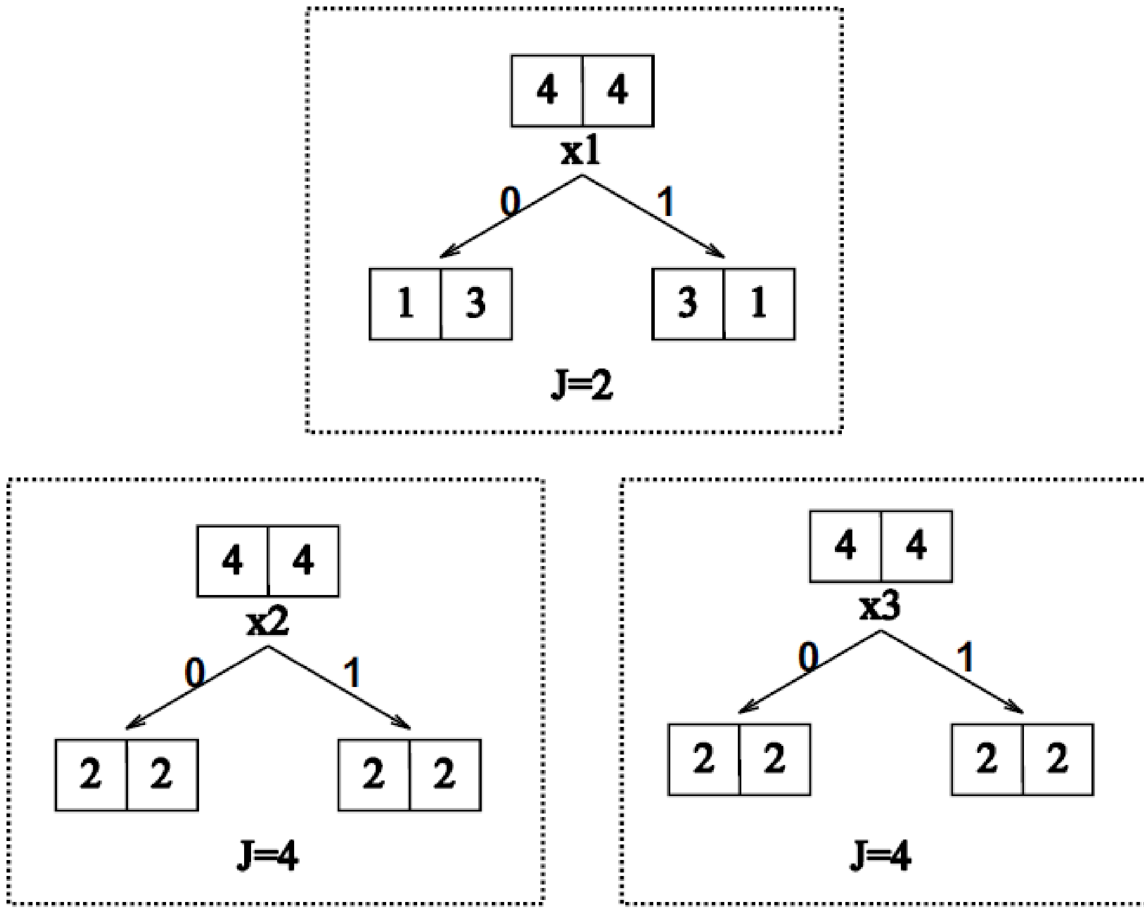
- Random choice?
- Feature with largest number of values?
- Feature with fewest?
- Lowest classification error?
- Information theoretic measure (Quinlan's approach)

Criteria: Classification Error

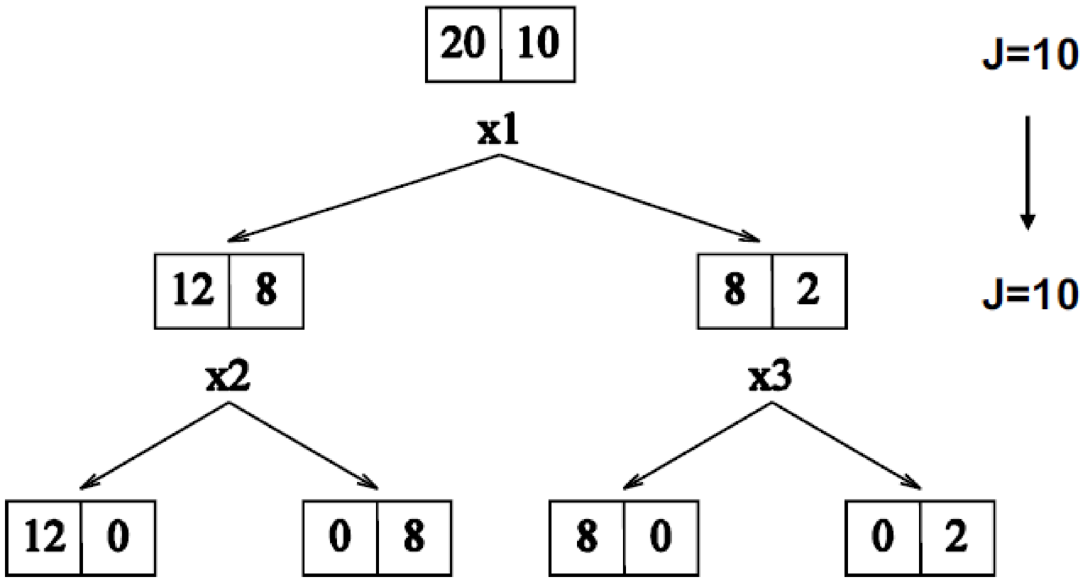
- Choose the feature for split as the one that has the lowest error on the training data

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



Information Theory to the rescue

- Let X be a random variable with the distribution

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

- The surprise S(X=x) of each example of V is defined to be $S(V = v) = -logP(V = v)$
- An event with probability 1 has zero surprise
- An event with probability 0 has infinite surprise
- The surprise is the asymptotic number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results. This is also called as description length of X.

Entropy

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

$$H(X) = -\sum_x P_X(x) \log_2 P_X(x)$$

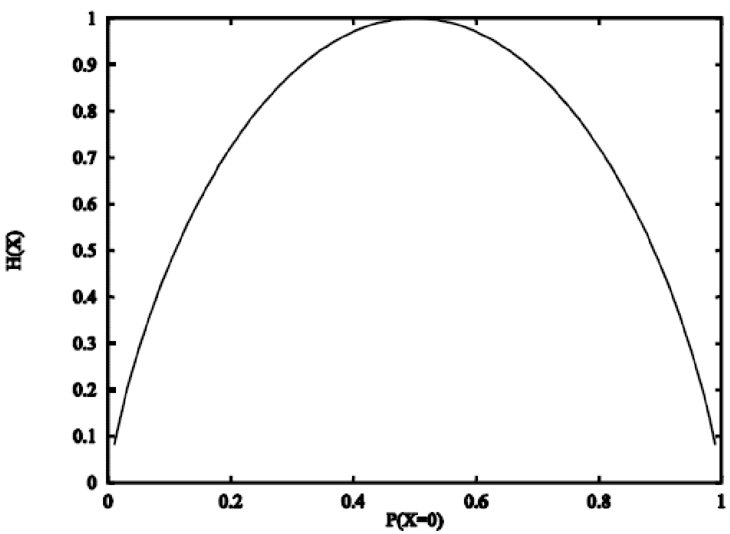
- **Entropy** measures the uncertainty of a random variable
- The larger the entropy, the more uncertain we are about the value of *X*
- If P(X=0)=0 (or 1), there is no uncertainty about the value of X, entropy = 0
- If P(X=0)=P(X=1)=0.5, the uncertainty is maximized, entropy = 1

Measuring Entropy

- S is a sample of training examples
- $p_\oplus$ is the proportion of positive examples in S
- $p_\otimes$ is the proportion of negative examples in S

Entropy measures the impurity of S

$$Entropy(S) = -p_\oplus \log p_\oplus - p_\otimes \log p_\otimes$$



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

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

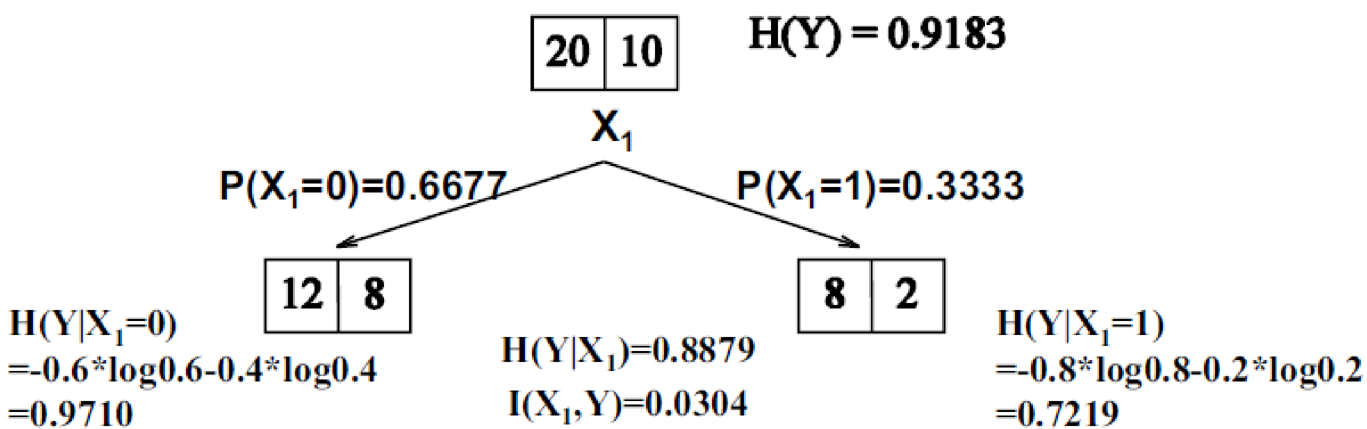
Mutual Information

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

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

Non-Boolean Features

- Multiple discrete values
  - Method 1: Construct multiway split
  - 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
- Real-valued variables
  - Test the variable against a threshold

- In all the cases, mutual information can be computed to choose the split

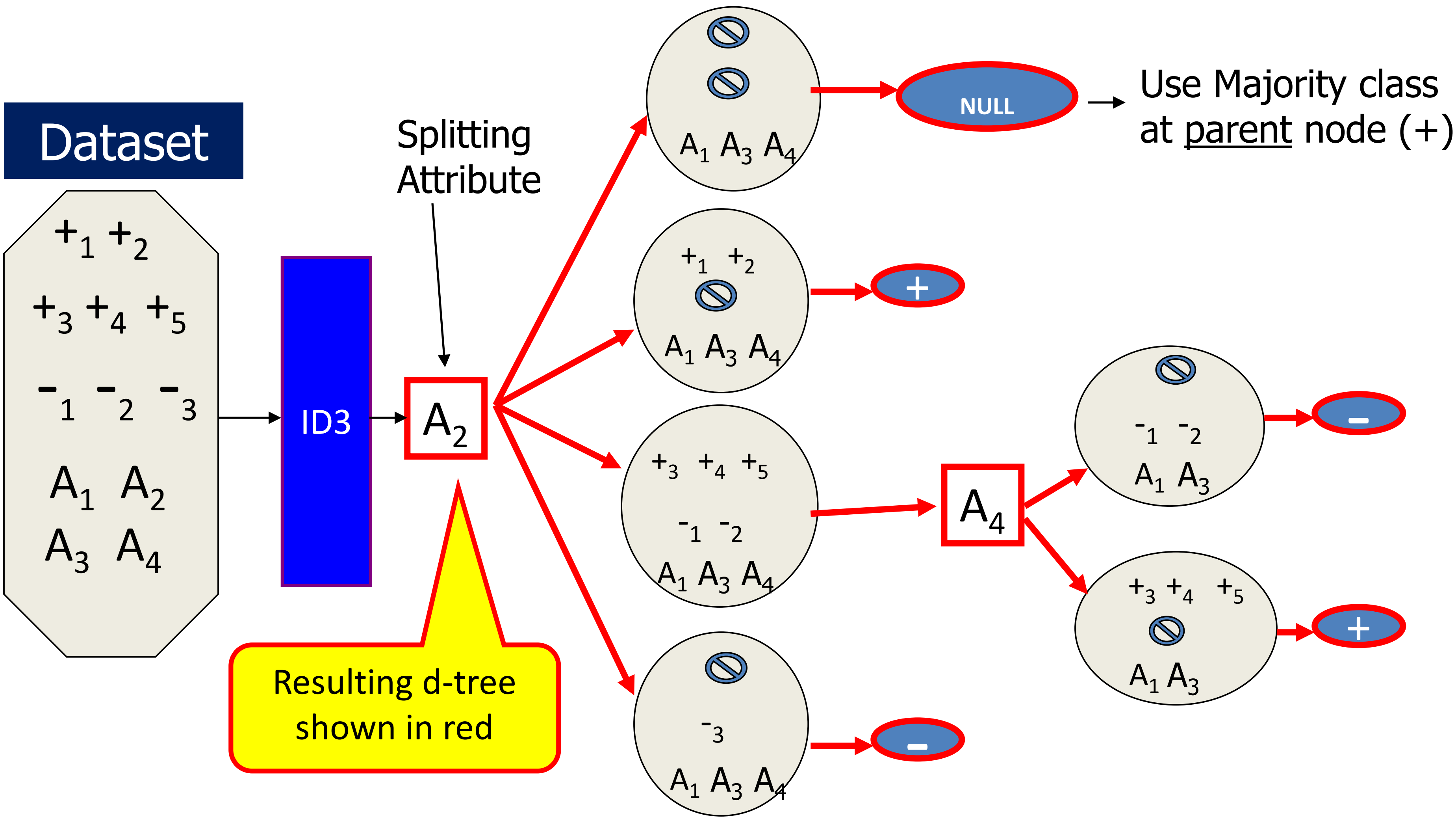
Top Down Induction

There are different ways to construct these trees. We will now look at a top-down, greedy search approach

High-level Idea:

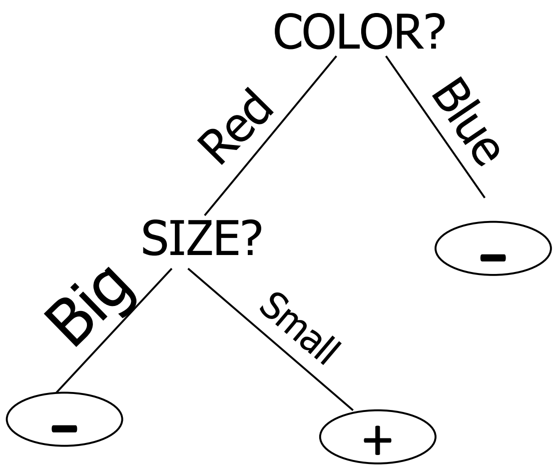
1. Choose the best feature f^* for the root of the tree.
2. Separate the training set into subsets $\{S_1, S_2, \ldots, S_k\}$ where each subset S_i contains examples that have the same value for f^*
3. Recursively apply the algorithm on each new subset until all examples have the same class label

Overview of ID3

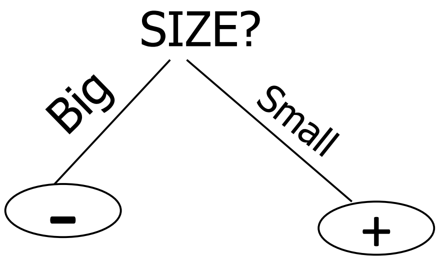


Main Hypothesis of ID3

The <u>simplest tree</u> that classifies training examples will work best on future examples (Occam's Razor)[1]



VS.



NP-Hard to find the smallest tree
(Hyafil +Rivest, 1976)

Some empirical evidence calls this assumption into question
(Mingers MLJ, Murphy+Pazzani JAIR)

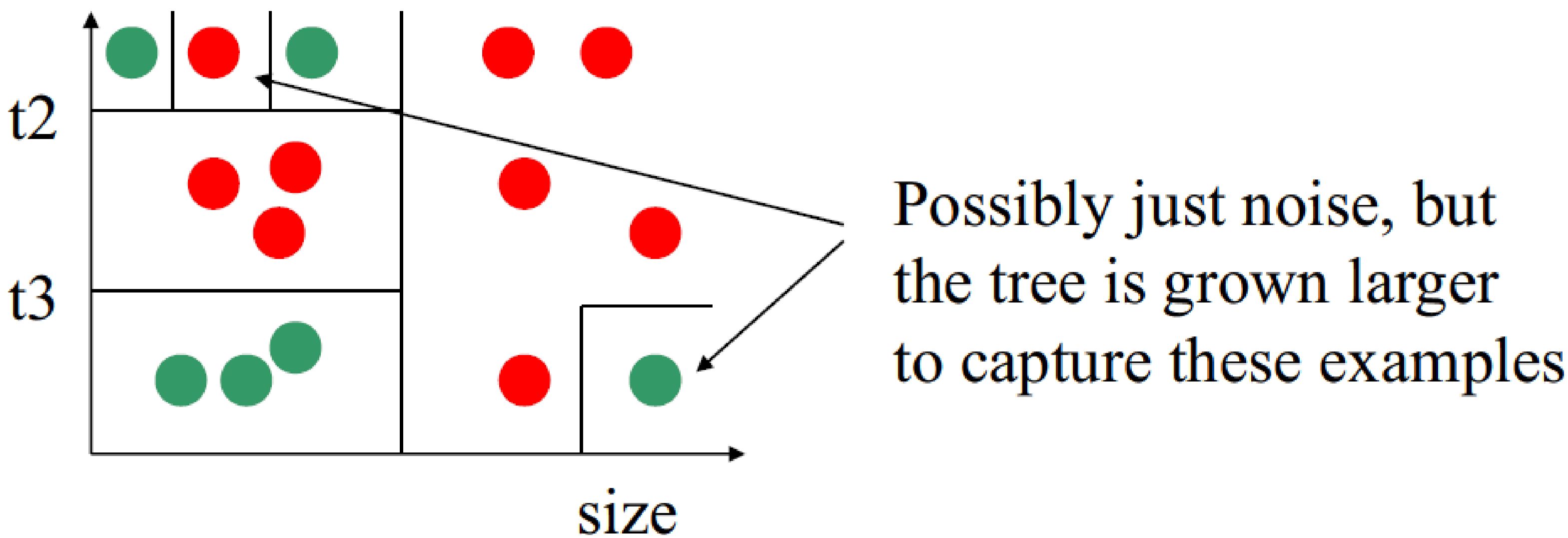
Why Occam's Razor?

(Occam lived 1285 – 1349)

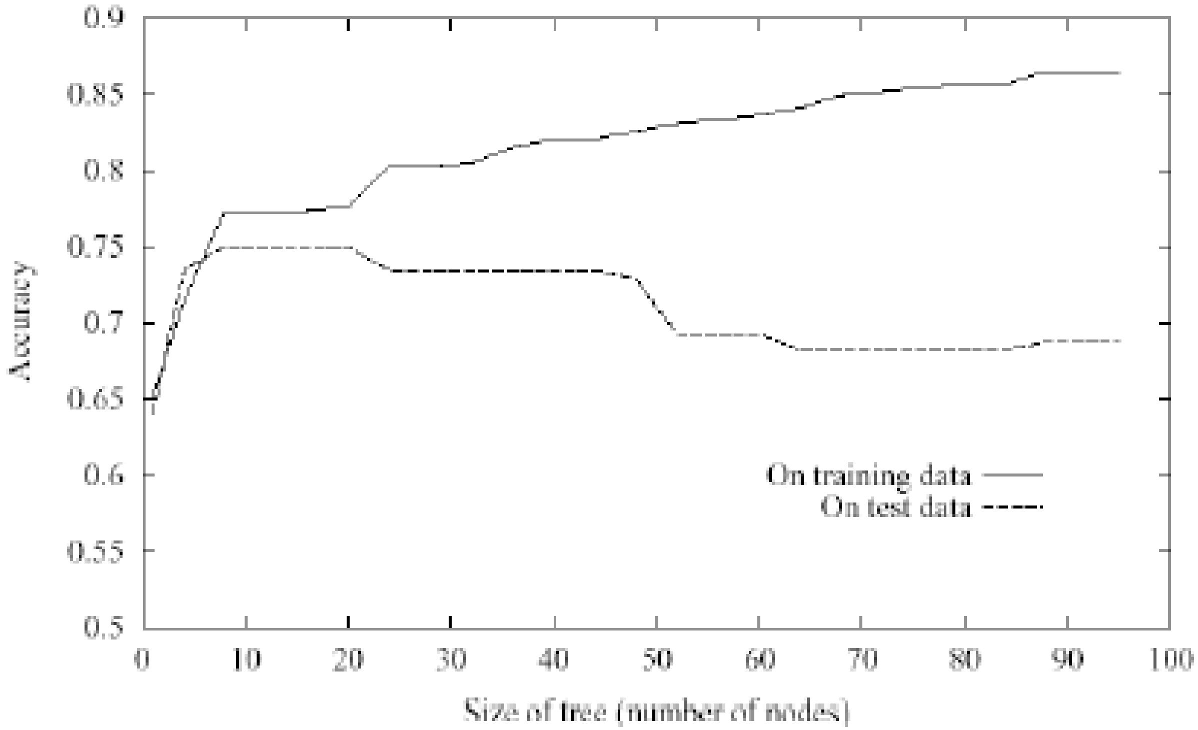
- There are fewer short hypotheses (trees in ID3) than long ones
- Short hypothesis that fits training data unlikely to be coincidence
- Long hypothesis that fits training data might be (since many more possibilities)
- COLT community formally addresses these issues (see Chapter 7 of Mitchell)
- **Arguments against**
  - There are many different ways to define small sets of hypotheses
  - E.g, All trees with a prime number of nodes that use attributes beginning with "Z"
  - What is so special about small sets based on size of hypothesis?

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

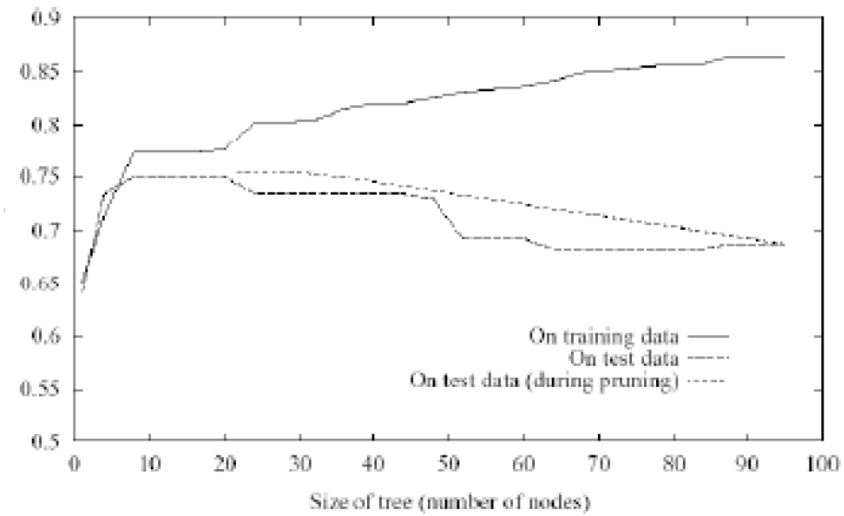


Over-fitting in Learning



Avoiding Overfitting

- Stop growing when data split is not statistically significant
- Grow a full tree, then post-prune
  - Separate training data into training set and validation set
  - Evaluate impact on validation set when a node is “pruned”
  - Greedily remove the one that improves the performance the most
  - Produces the smallest version of most accurate subtree
  - What if the data is limited?



Attributes with Costs

- Consider
  - Medical diagnosis, BloodTest costs $150
  - Robotics, Width_from_1ft has cost 23 sec
- How to learn a consistent tree with low expected cost? Find min cost tree.
- Another approach – Replace gain when you split by
- $Gain^2(S, A)/Cost(A)$ Tan and Schimmer (1990)
- $(2^{Gain(S,A)} - 1)/(Cost(A) + 1)^w$ where w in [0,1] reflects the importance – Nunez (1998)

Decision Trees

- Decision Trees – Popular and a very efficient hypothesis space
  - Variable size: Any boolean function can be represented
  - Deterministic (can be extended)
  - Discrete and continuous paramters
- Constructive search: Built by adding nodes
- Eager
- Batch (now online algorithms are popular as well)

Criterion	**Perceptron**	**Logistic**	**LDA**	**DT**
Mixed data	N	N	N	Y
Missing values	N	N	Y	Y
Outliers	N	Y	N	Y
Monotone	N	N	N	Y
Scalability	Y	Y	Y	Y
Irrelevant i/p	N	N	N	Some what
Interpretable	Y	Y	Y	Y
Accurate	Y	Y	Y	N