Decision Trees

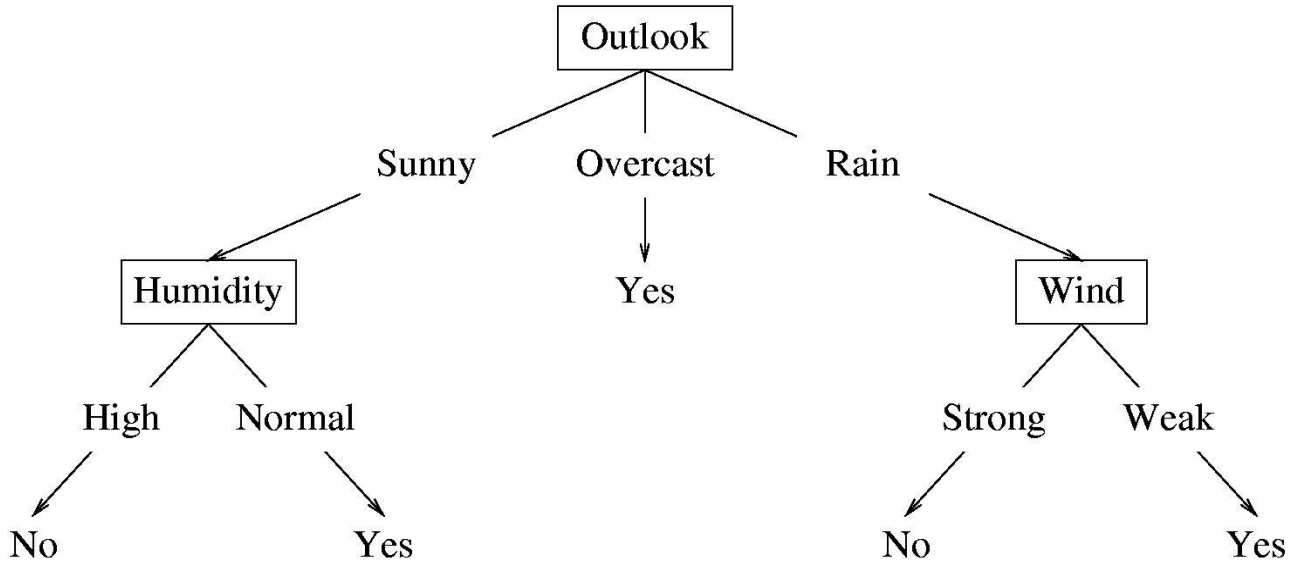
Vibhav Gogate
The University of Texas at Dallas

Recap

- Supervised learning
  - Given: Training data with desired output
  - Assumption: There exists a function f which transforms input “x” into output f(x).
  - To do: find an approximation to f
- Classification: Output, i.e., f(x) is discrete
- What makes learning hard?
- Issues.

Decision Tree Hypothesis Space

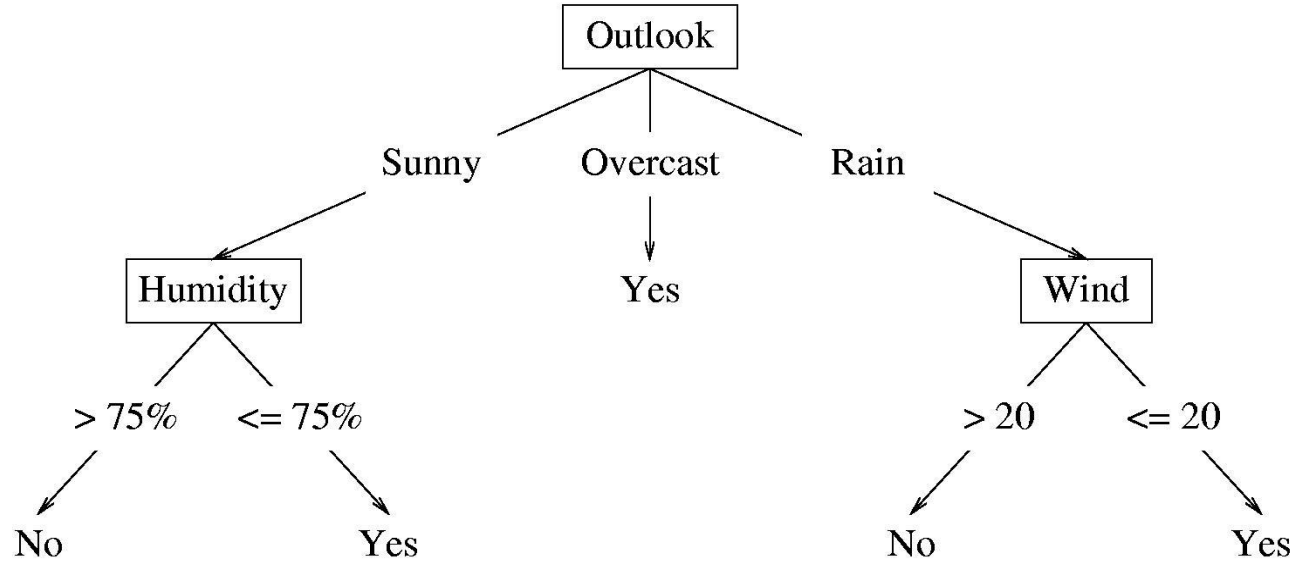
- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

Decision Tree Hypothesis Space

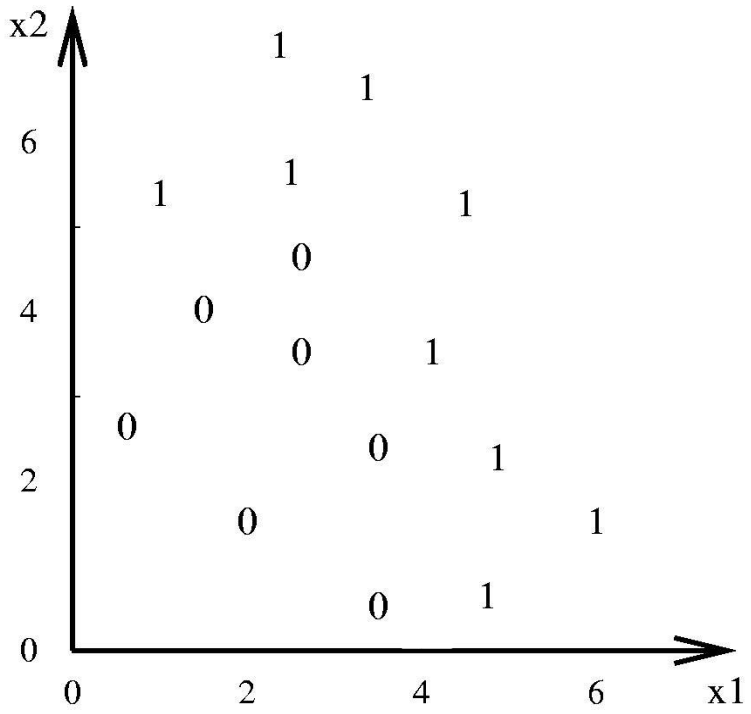
If the features are continuous, internal nodes may test the value of a feature against a threshold.

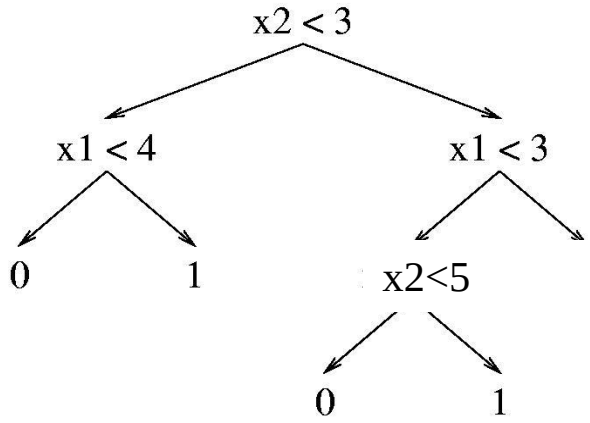


- **Notes:**
  - **A discrete feature can appear only once (or not appear at all) along the unique path from the root to a leaf.**
  - **Question: Can I test on Humidity with a threshold of 95?**
    - **YES (it is a different discrete feature).**

Decision Tree Decision Boundaries

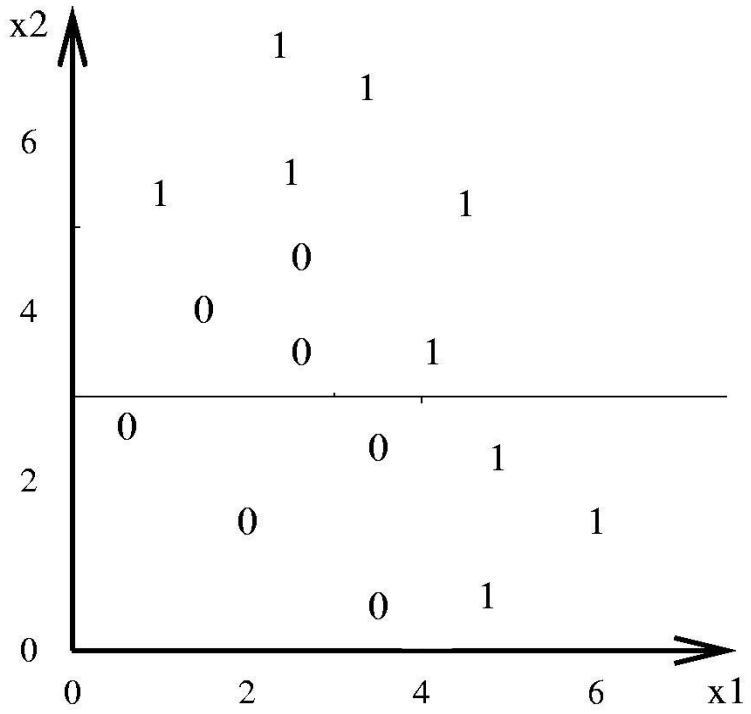
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

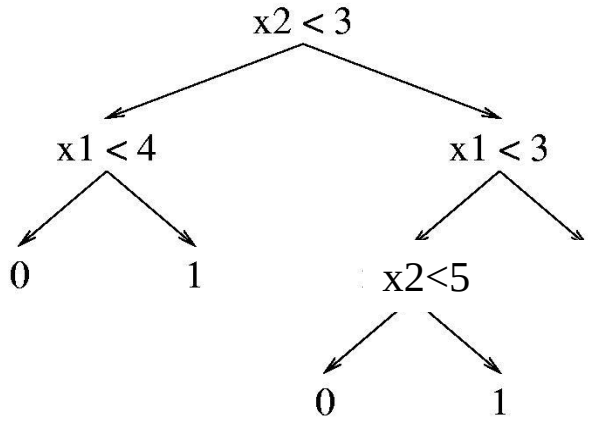




Decision Tree Decision Boundaries

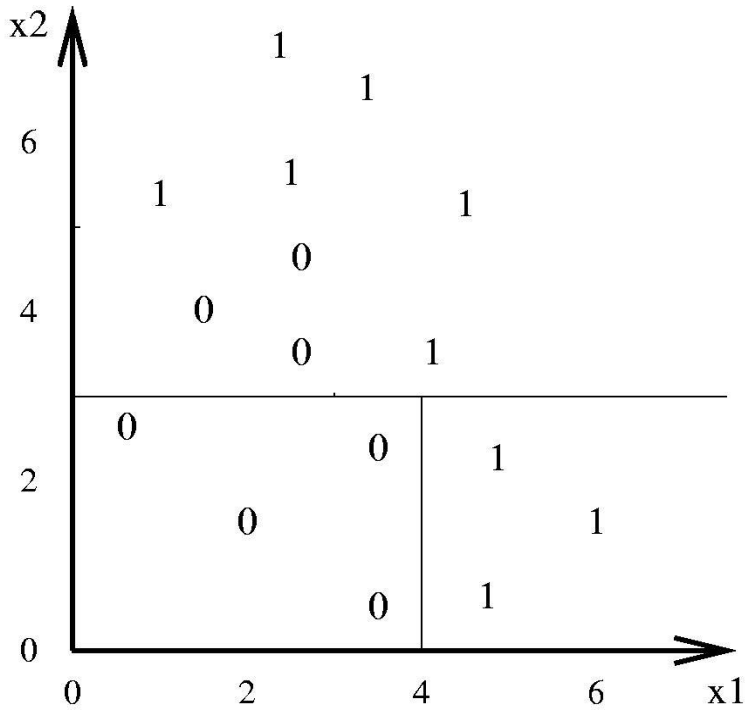
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

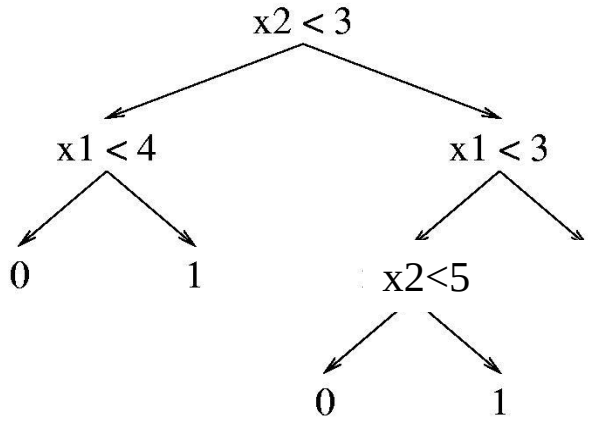




Decision Tree Decision Boundaries

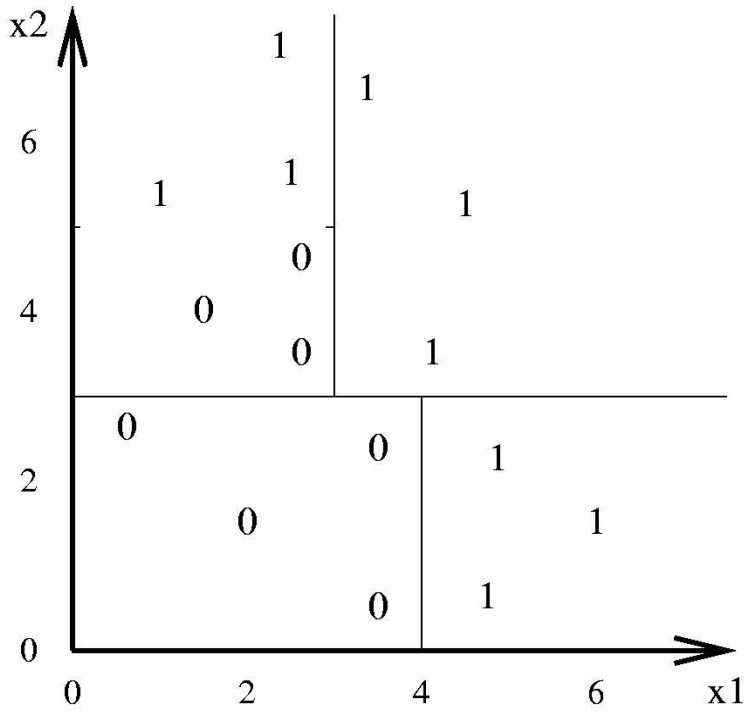
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

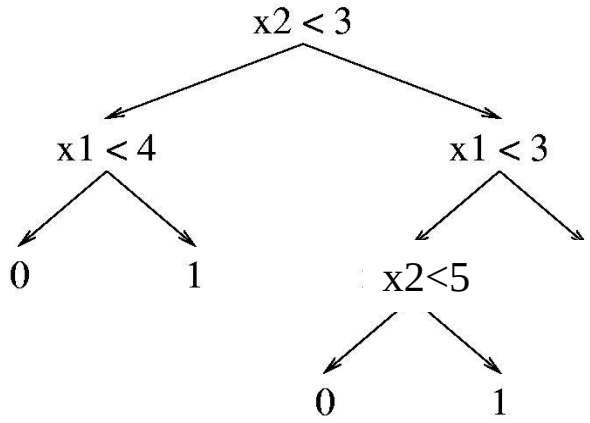




Decision Tree Decision Boundaries

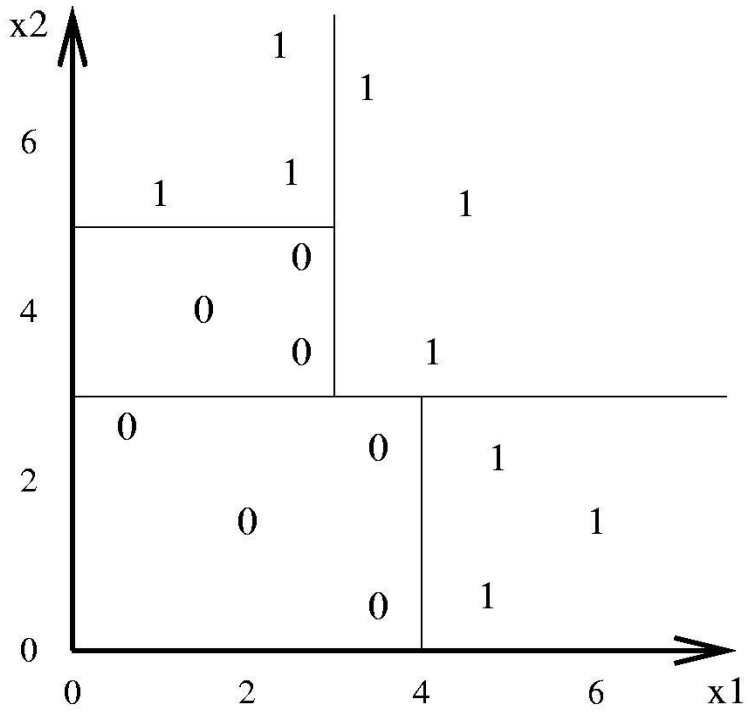
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

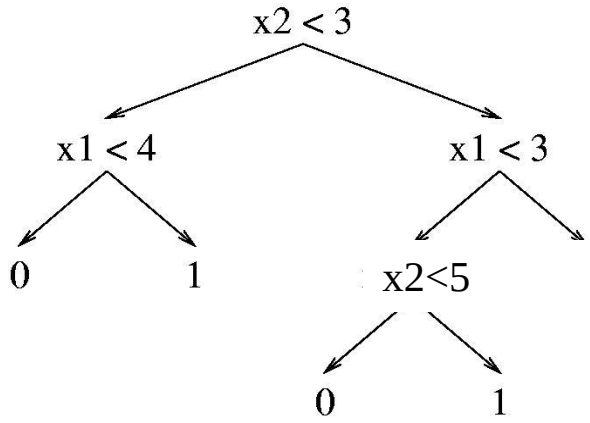




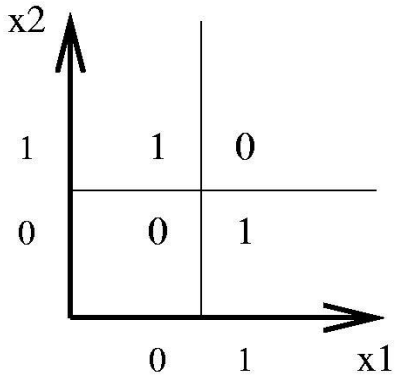
Decision Tree Decision Boundaries

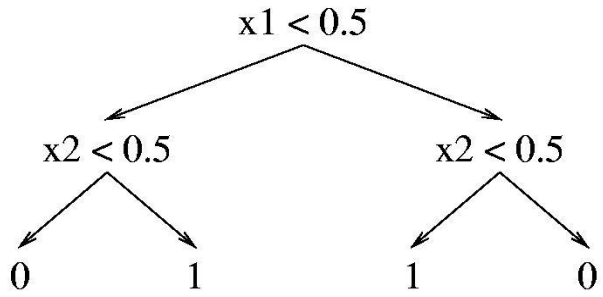
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision Trees Can Represent Any Boolean Function





The tree will in the worst case require exponentially many nodes, however.

Can you put a bound on the number of leaf nodes?

Decision Trees Provide Variable-Size Hypothesis Space

As the number 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; some boolean functions involving three features (e.g., $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_3)$
- **etc.**

Learning Algorithm for Decision Trees

The same basic learning algorithm has been discovered by many people independently:

GrowTree(S)
if ($y = 0$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(0)
else if ($y = 1$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(1)
else
    choose best attribute x_j
    $S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
    $S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
    return new node(x_j, GrowTree(S_0), GrowTree(S_1))

The following questions may arise in your mind!

- How to choose the best attribute?
  - Which property to test at a node
- When to declare a particular node as leaf?
- What types of trees should we prefer, smaller, larger, balanced, etc?
- If a leaf node is impure (has both positive and negative classes), what should we do?
- What if some attribute value is missing?

Choosing the best Attribute?

- Fundamental principle underlying tree creation
  - Simplicity (prefer smaller trees)
  - Occam's Razor: Simplest model that explains the data should be preferred
- Each node divides the data into subsets
  - Heuristic: Make each subset as **pure as possible**.

Choosing the best Attribute: Information Gain Heuristic

$$Gain(S, A) = H(S) - \sum_{v \in Values(A)} P(v) H(S_v)$$

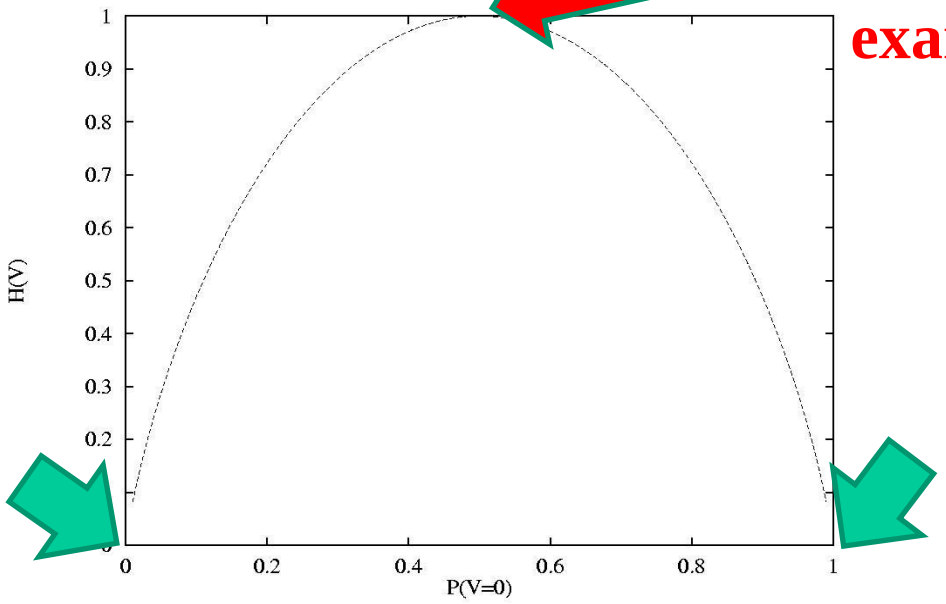
- Entropy, denoted by H is a measure of impurity
- Gain = Current impurity – New impurity
  - Reduction in impurity
  - Maximize gain
- Second term actually gives expected entropy (weigh each bin by the amount of data in it)

Entropy

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

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

50% positive and 50% negative examples



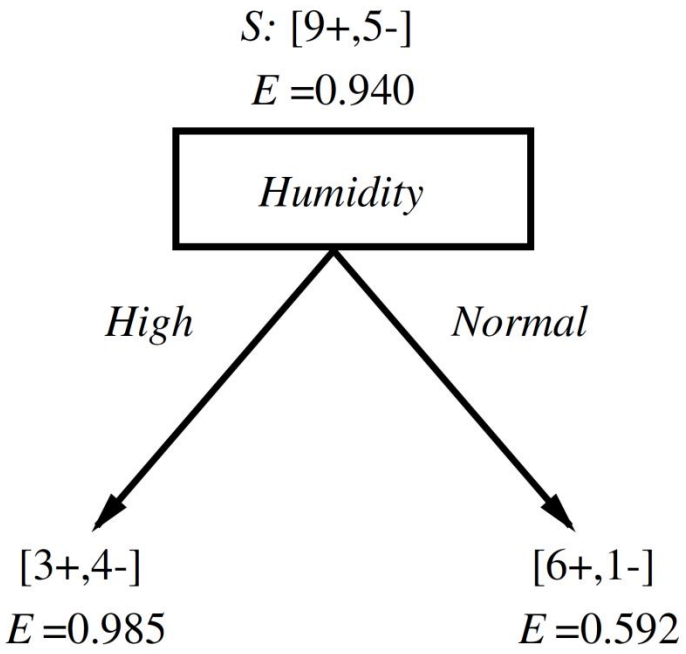
All examples are negative

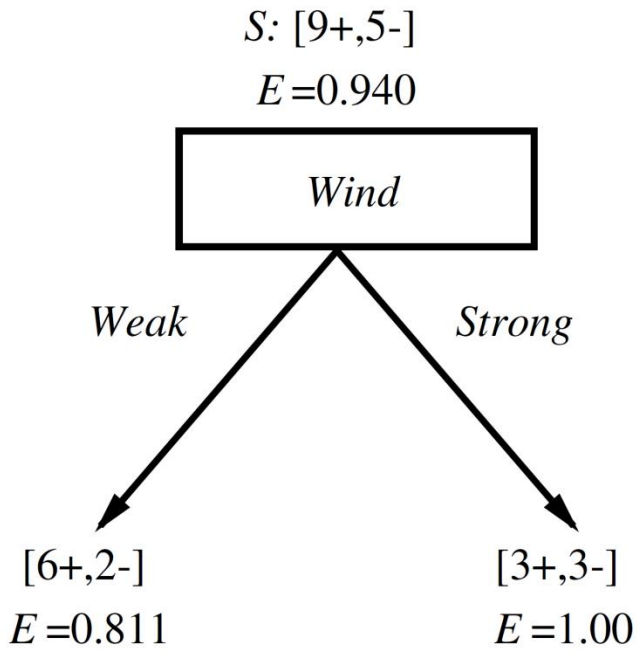
All examples are positive

Entropy can be viewed as a measure of uncertainty.

When do I play tennis?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

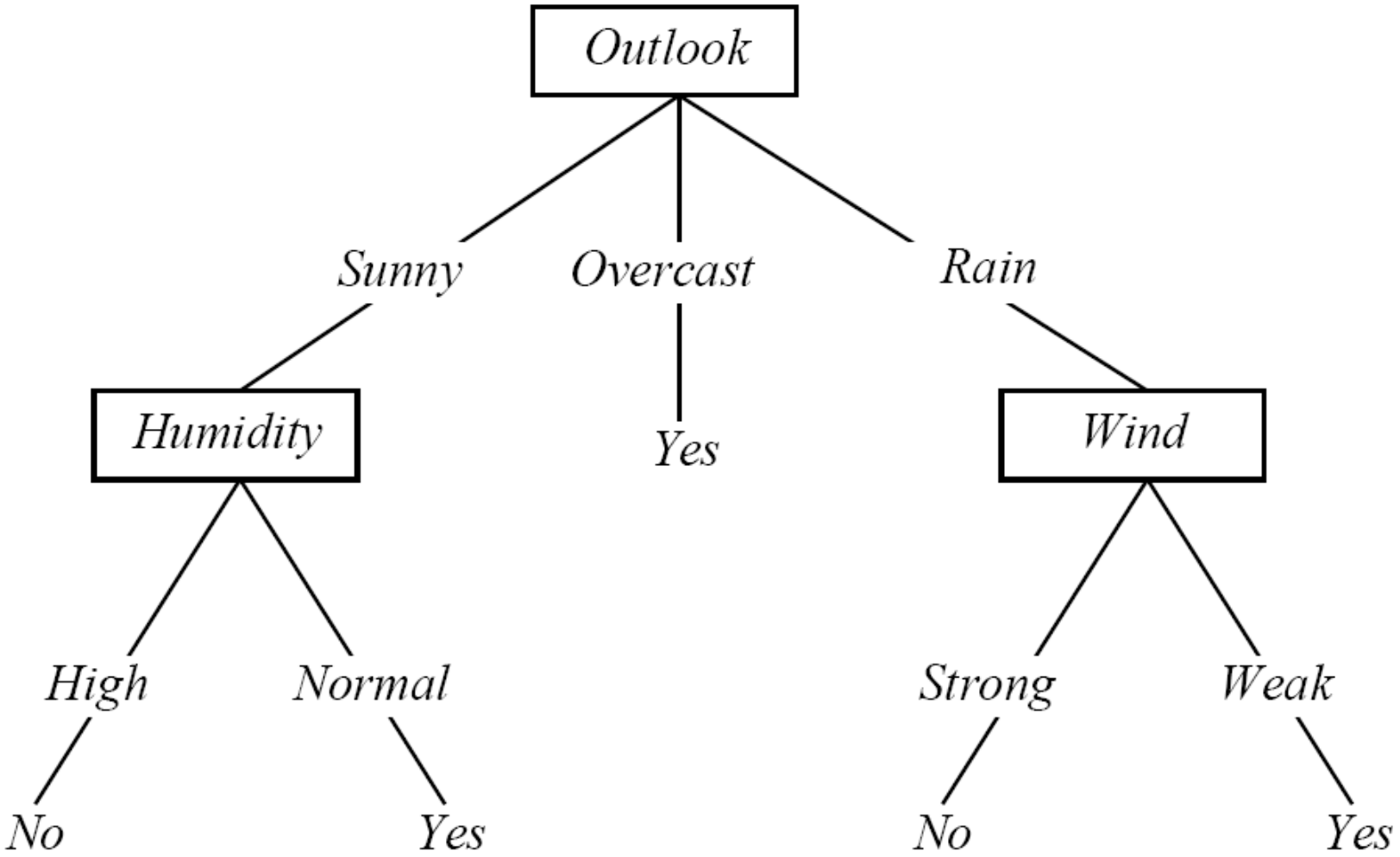
Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

When do I play tennis?

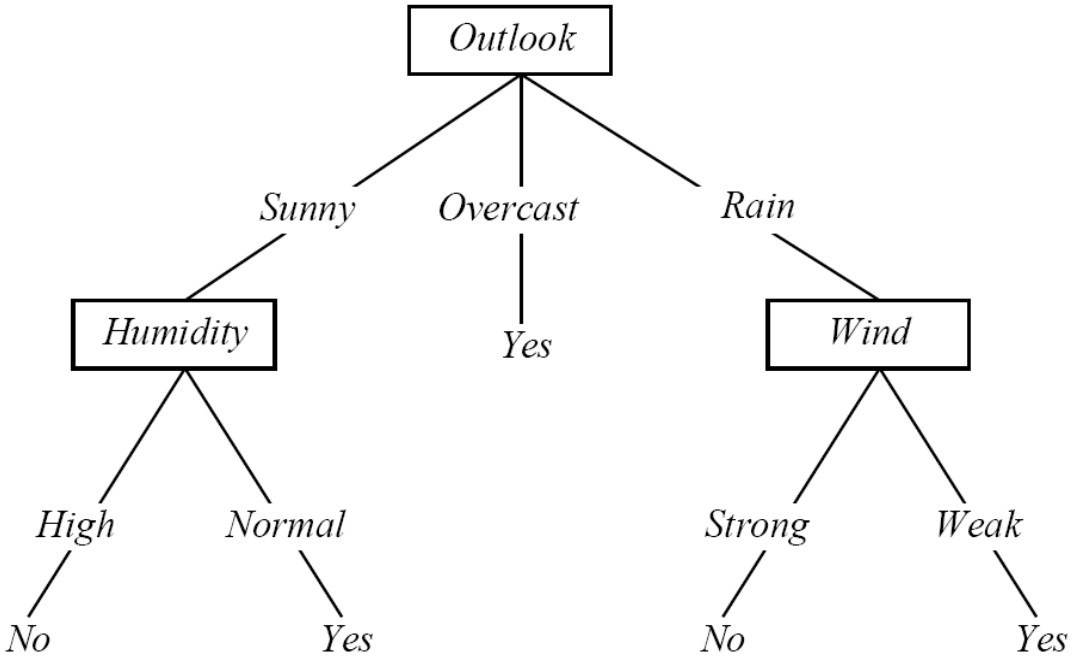
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Decision Tree



Is the decision tree correct?

- Let's check whether the split on Wind attribute is correct.
- We need to show that Wind attribute has the highest information gain.



When do I play tennis?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Wind attribute – 5 records match

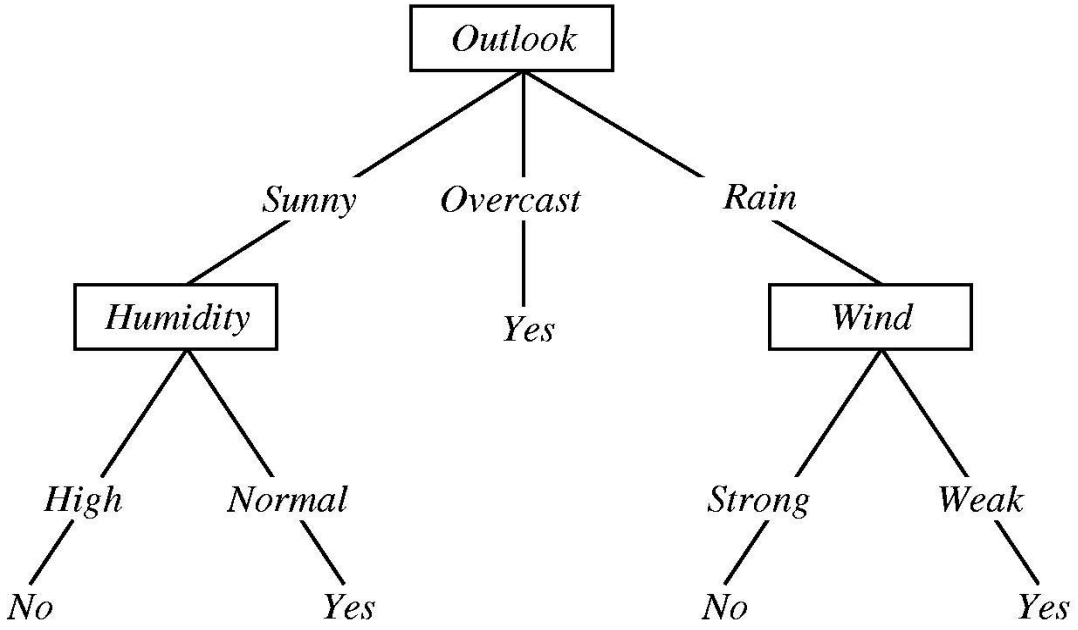
Note: calculate the entropy only on examples that got "routed" in our branch of the tree (Outlook=Rain)

Day					PlayTennis
D1					No
D2					No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Practical Issues in Decision Tree Learning

- Overfitting
- When to stop growing a tree?
- Handling non-Boolean attributes
- Handling missing attribute values

Overfitting in Decision Trees



Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, PlayTennis=No

What effect on tree?

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

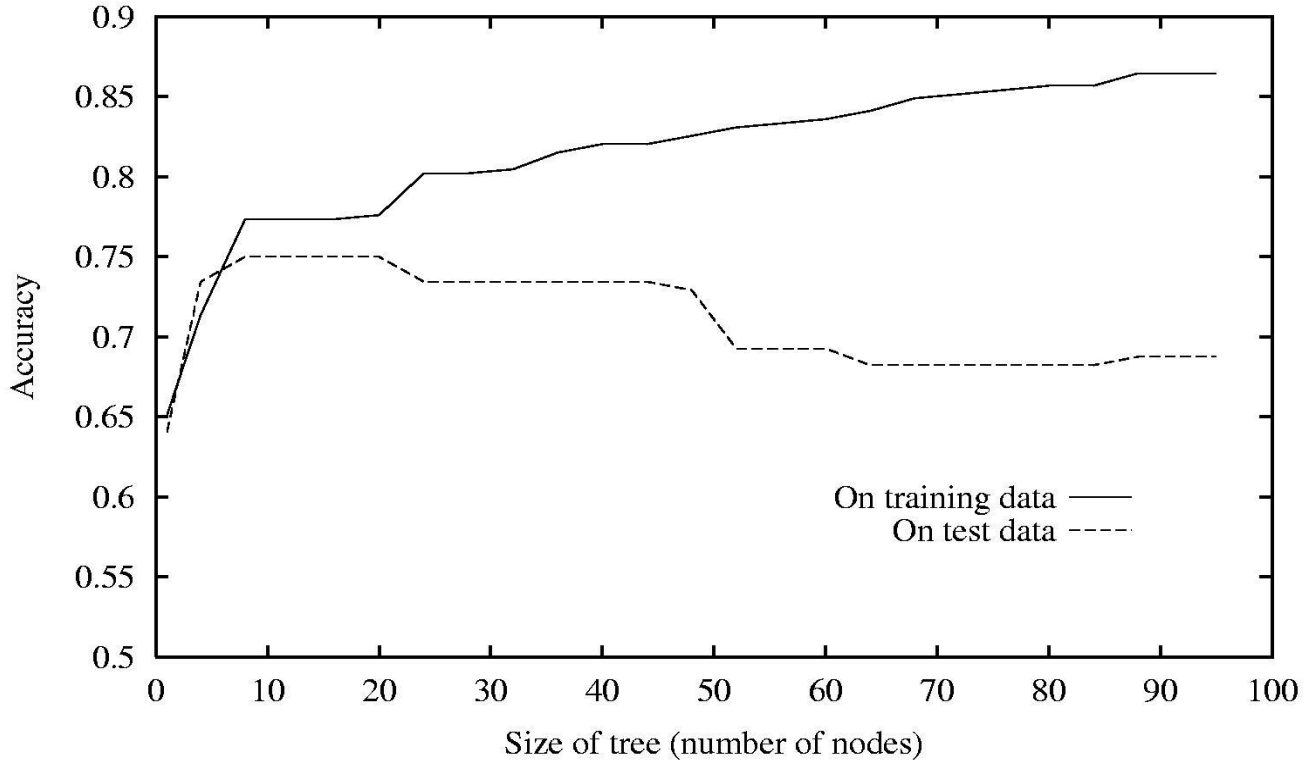
Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting in Decision Tree Learning



Sources of Overfitting

- Noise
- Small number of examples associated with each leaf
  - What if only one example is associated with a leaf. Can you believe it?
  - Coincidental regularities
- Generalization is the most important criteria
  - Your method should work well on examples which you have not seen before.

Avoiding Overfitting

- Two approaches
  - Stop growing the tree when data split is not statistically significant
  - Grow tree fully, then post-prune
- Key Issue: What is the correct tree size?
  - Divide data into training and validation set
    - Random noise in two sets might be different
  - Apply statistical test to estimate whether expanding a particular node is likely to produce an improvement beyond the training set
  - Add a complexity penalty

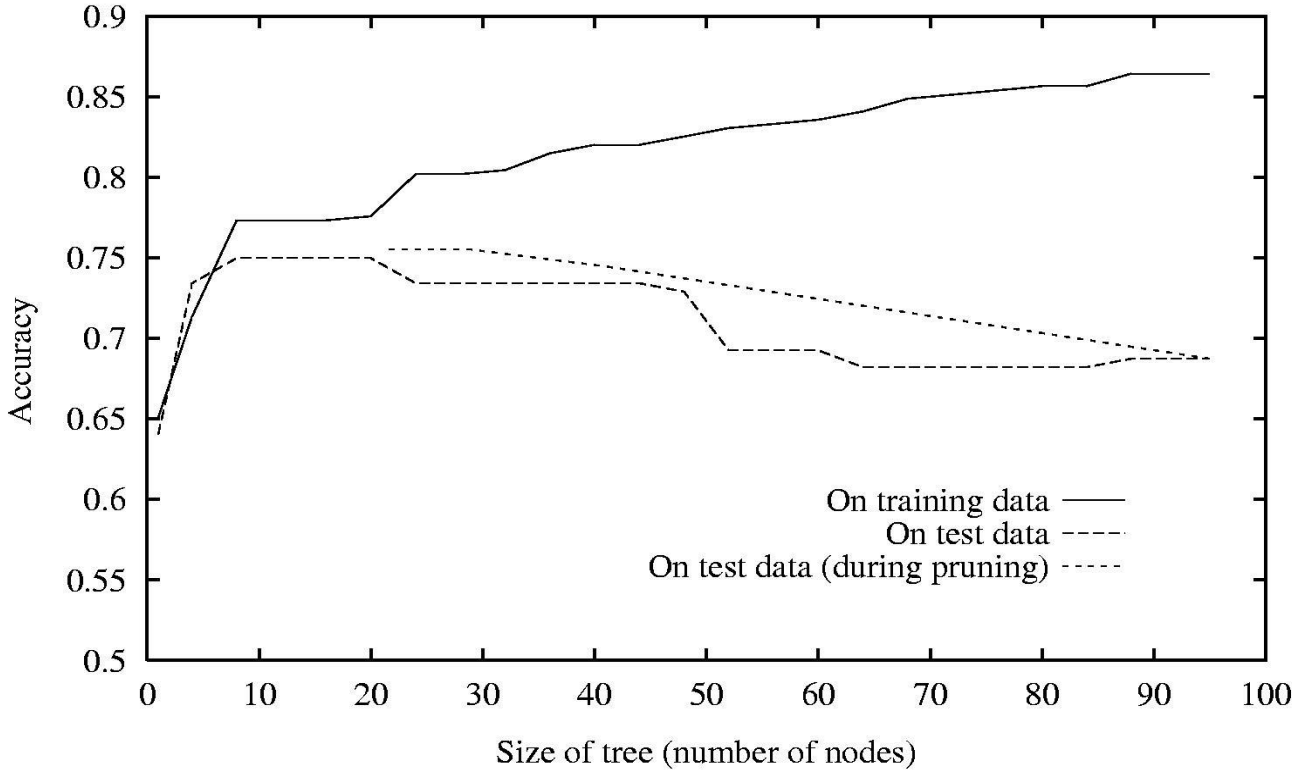
Reduced-Error Pruning

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



Rule Post Pruning

- Induce the decision tree using the full training set (allowing it to overfit)
- Convert the decision tree to a set of rules
- Prune each rule by removing pre-conditions that improve the estimated accuracy
  - Estimate accuracy using a validation set
- Sort the rules using their estimated accuracy
- Classify new instances using the sorted sequence

Non-Boolean Features

- **Features with multiple discrete values**

  Construct a multiway split?

  Test for one value versus all of the others?

  Group the values into two disjoint subsets?

- **Real-valued features**

  Consider a threshold split using each observed value of the feature.

Whichever method is used, the mutual information can be computed to choose the best split.

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

Handling Missing Values

- Some attribute-values are missing
  - Example: patient data. You don't expect blood test results for everyone.
- Treat the missing value as another value
- Ignore instances having missing values
  - Problematic because throwing away data
- Assign it the most common value
- Assign it the most common value based on the class that the example belongs to.

Handling Missing Values: Probabilistic approach

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

Summary: Decision Trees

- Representation
- Tree growth
- Choosing the best attribute
- Overfitting and pruning
- Special cases: Missing Attributes and Continuous Attributes
- Many forms in practice: CART, ID3, C4.5