FIT 3080: Intelligent Systems

Mathematical Principles of Machine Learning

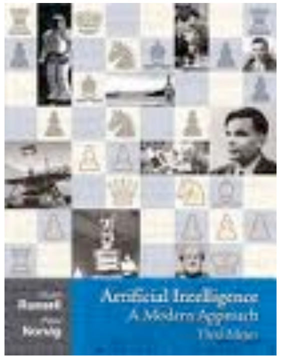
A Case Study: Decision Trees

Gholamreza Haffari – Monash University

Some slides are adapted from Dieter Fox or Ingrid Zukerman

Announcements

- Readings:
  - Sections 18.1-3



Outline

- Mathematical Principles of Learning
  - Inductive Learning
  - Hypothesis Space
  - Hypothesis complexity
  - Overfitting & Generalization

- Decision Trees
  - Model
  - Entropy and Information Gain
  - DT Learning Algorithm
  - Preventing Overfitting

Why Learning?

- Learning is essential for unknown environments
  - e.g., when designer lacks omniscience
- Learning is necessary in dynamic environments
  - Agent can adapt to changes in environment not foreseen at design time
- Learning is useful as a system construction method
  - Expose the agent to reality rather than trying to approximate it through equations etc.
- Learning modifies the agent's decision mechanisms to improve performance

Types of learning

- Supervised learning: correct answers for each input is provided
  - E.g., decision trees, Perceptron, Naïve Bayes, K-NN
- Unsupervised learning: correct answers not given, must discover patterns in input data
  - E.g., K-Means
- Reinforcement learning: occasional rewards (or punishments) given
  - E.g., Q learning, MDPs

Inductive learning

- A form of Supervised Learning:

  Learn a function from examples

Picked from a hypothesis space H

$$h \approx f$$

$$x \longrightarrow y$$

Training Data

(x_1,y_1)

(x_2,y_2)

…

(x_n,y_n)

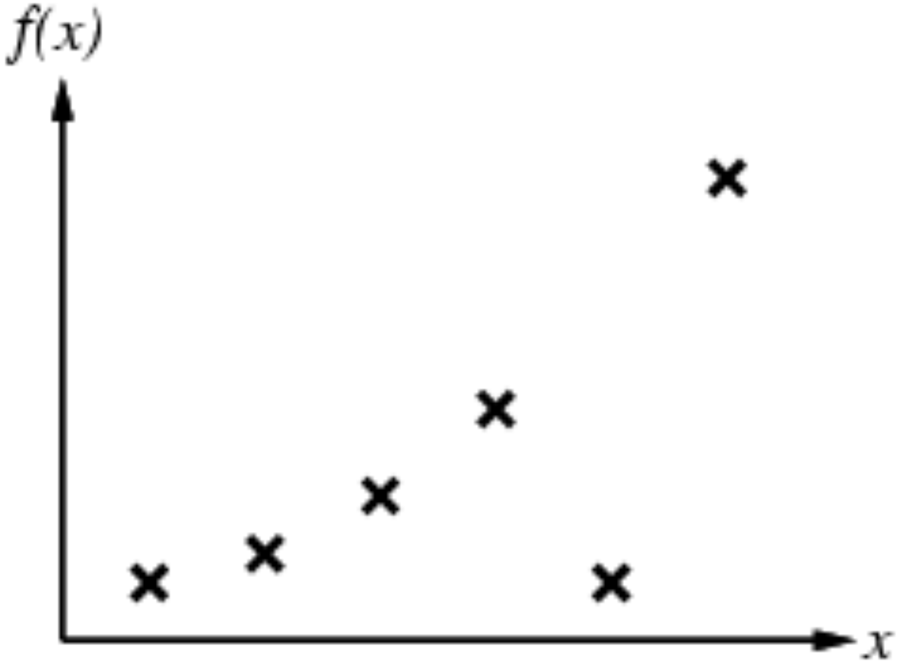
Inductive learning

- Setup:
  - f is the unknown target function
  - Given some examples pairs from it (x, f(x))

- Problem: learn a function ("hypothesis") h
  - Based on the training set of examples
  - Such that $h \approx f$ (h approximates f as best as possible)
  - Meaning h must generalize well on unseen examples

Picking the best hypothesis h

- Big Idea 1: Pick h from the space H which agrees with f on training set
  - h is consistent if it agrees with f on all training examples
- Example: curve fitting (regression):



Training Data

(x_1,y_1)

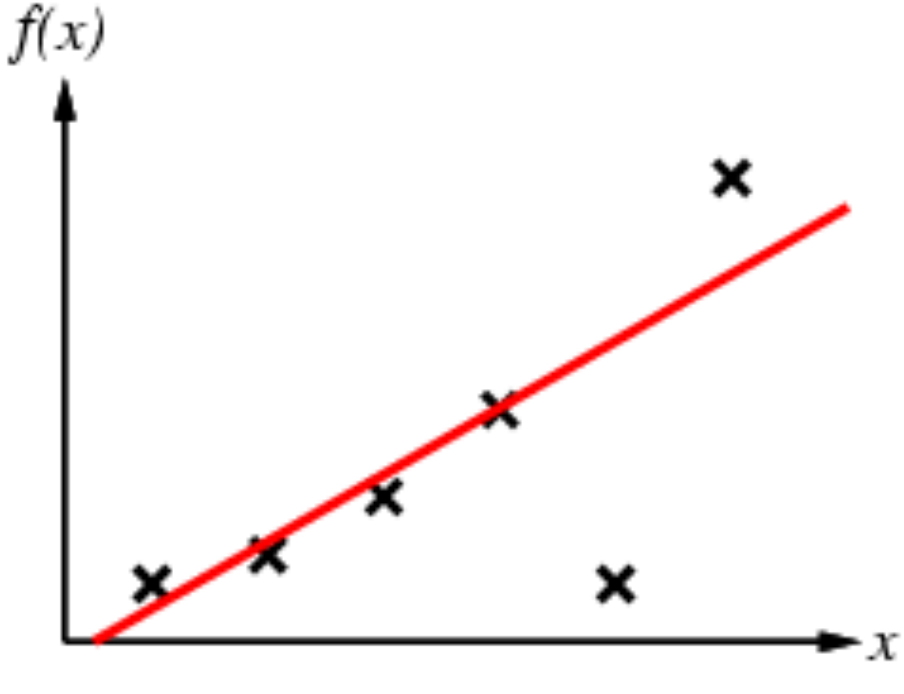
(x_2,y_2)

…

(x_n,y_n)

Inductive learning example

- h = Straight line?



Training Data

(x_1, y_1)

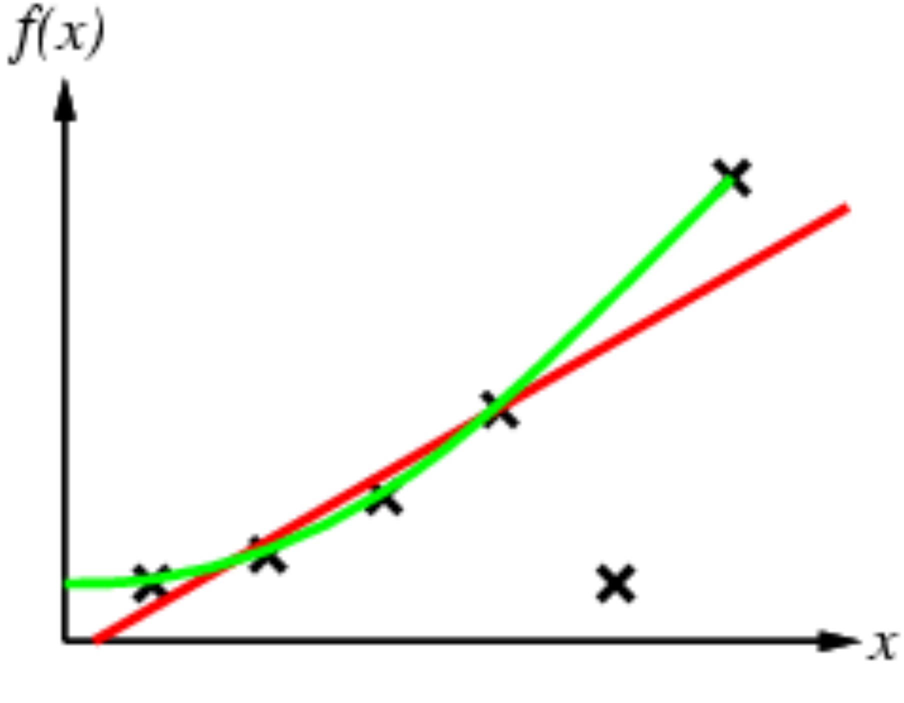
(x_2, y_2)

…

(x_n, y_n)

Inductive learning example

- What about a quadratic function?



Training Data

(x_1,y_1)

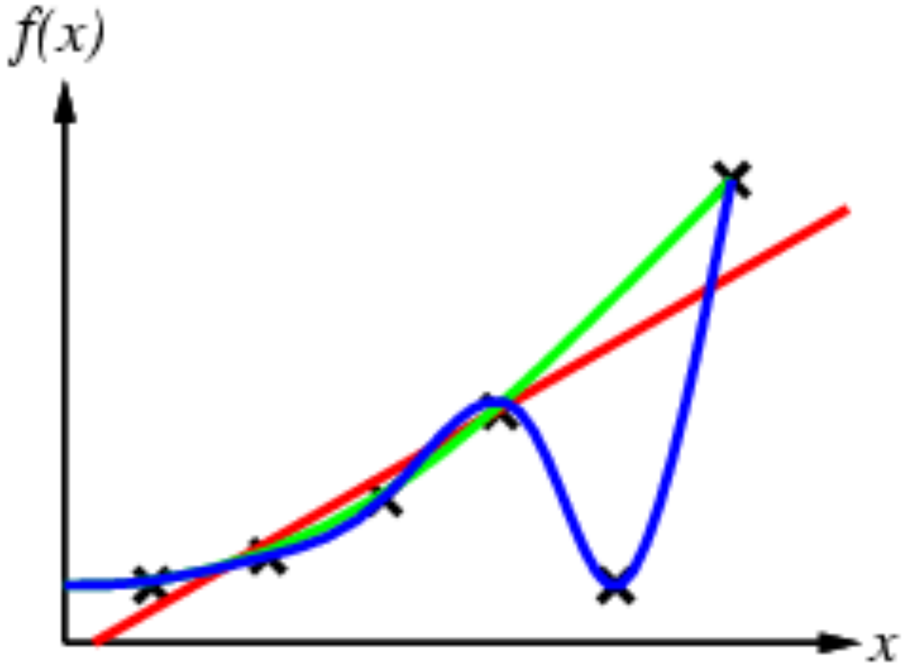
(x_2,y_2)

…

(x_n,y_n)

Inductive learning example

- Finally, a function that satisfies all



Training Data

(x_1,y_1)

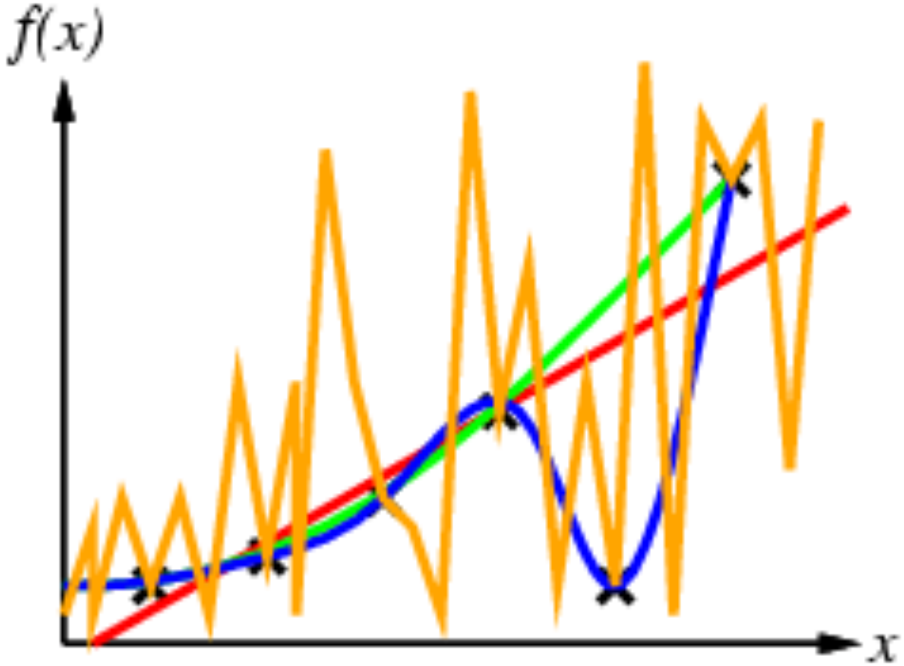
(x_2,y_2)

…

(x_n,y_n)

Inductive learning example

- But so does this one…



Training Data

(x_1,y_1)

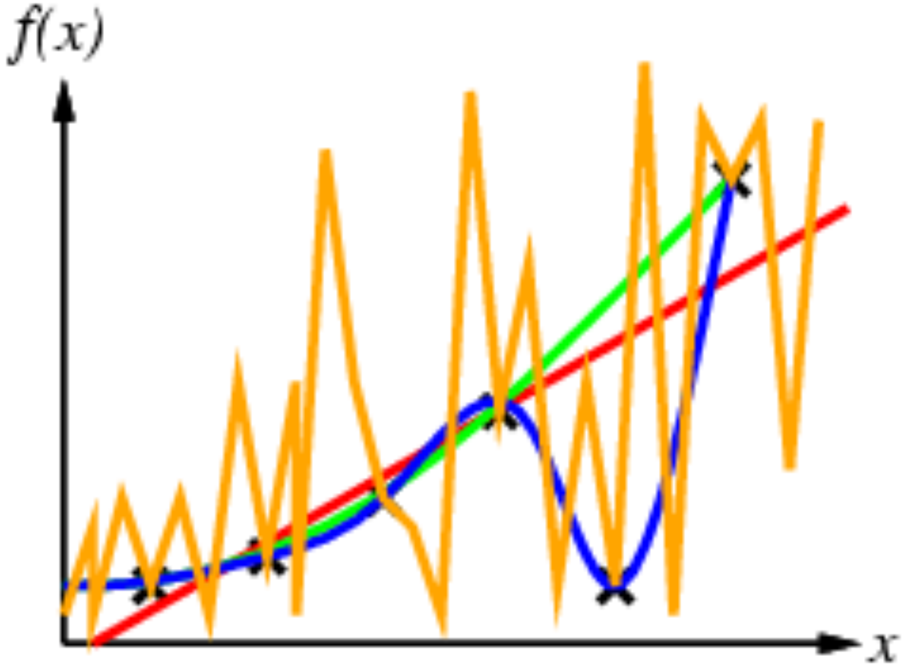
(x_2,y_2)

…

(x_n,y_n)

Ockham’s razor principle

- Big Idea 2: Prefer the simpler hypothesis vs complex ones
  - Smooth blue function preferable over wiggly yellow one
  - The wiggly one is perfect on training data but most probably will be very bad on unseen data: Overfitting



Training Data

(x_1,y_1)

(x_2,y_2)

…

(x_n,y_n)

Mathematical Principles of Learning

Idea 2: To improve generalizability and prevent overfitting

Bias

Choose the **simplest** hypothesis **from H** which is **consistent** with the training data

Idea 1: To be similar to the unknown true underlying function

Mathematical Principles of Learning

Best learned hypothesis

Idea 2penalize complexity

$$h^* = \operatorname{argmin}_{h \in H} \left[\sum_{(x,y) \in D} error(h(x), y) \right] + \lambda Complexity(h)$$

Bias

Idea 1: Error on the training data

Learning is indeed Search/Optimization

Outline

- Mathematical Principles of Learning
  - Inductive Learning
  - Hypothesis Space
  - Hypothesis complexity
  - Overfitting & Generalization

- Decision Trees
  - Model
  - Entropy and Information Gain
  - DT Learning Algorithm
  - Preventing Overfitting

Decision Trees (DTs)

- **Input:** Description of an object or a situation through a set of **attributes**.

- **Output:** a **decision**, that is the predicted output value for the input.

DT Example: Training Dataset

Attributes

Day	Outlook	Temperature	Humidity	Wind	Play ball
D1	Sunny	Hot	High	Weak	No

Input x (vector of attribute values)

Output y

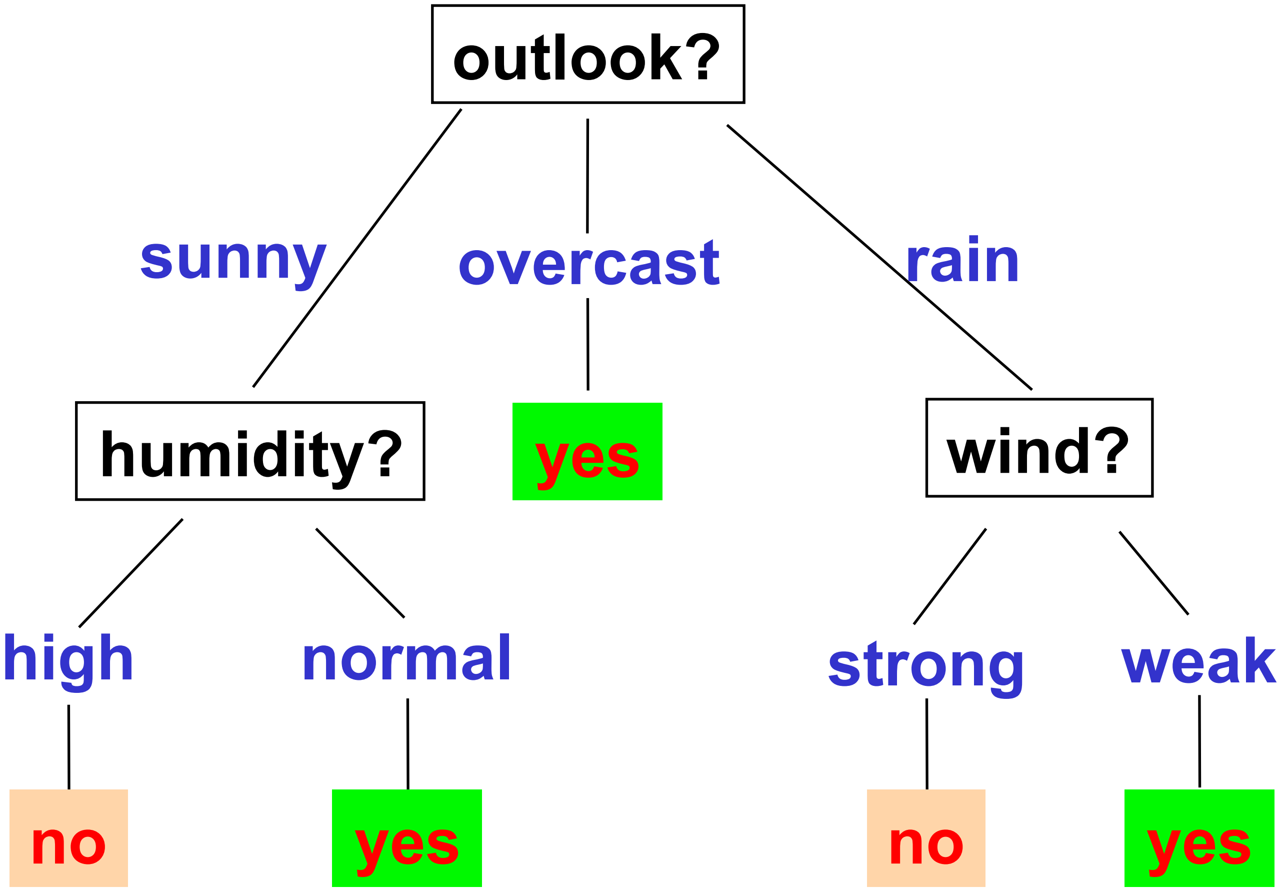
DT Example: Training Dataset

Input x (vector of **attributes**)

Output y

Day	Outlook	Temperature	Humidity	Wind	Play ball
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

DT Example: Learned Hypothesis h



Decision tree is equivalent to logic in disjunctive normal form

G-Day $\Leftrightarrow$ (Sunny $\wedge$ Normal) $\vee$ Overcast $\vee$ (Rain $\wedge$ Weak)

Classification by Decision Tree Induction

- Training Data: Records of items that have:
  - Input x: Represented by a vector of attribute values
  - Output y: The corresponding target value
- Learning/Constructing the tree:
  - Based on a “greedy” algorithm
  - Builds a decision tree in a top-down, recursive, divide-and-conquer manner

Decision Tree Learning Algorithm

1. Start with all training examples at the root

2. Partition examples recursively based on selected attributes
   - attributes are categorical
     - if continuous-valued, they are broken up into ranges
   - attributes are selected using heuristics or a statistical measure
     - e.g., ***information gain***

3. Stop partitioning when
   - there is no further gain in partitioning

Employ majority voting for classifying the leafs

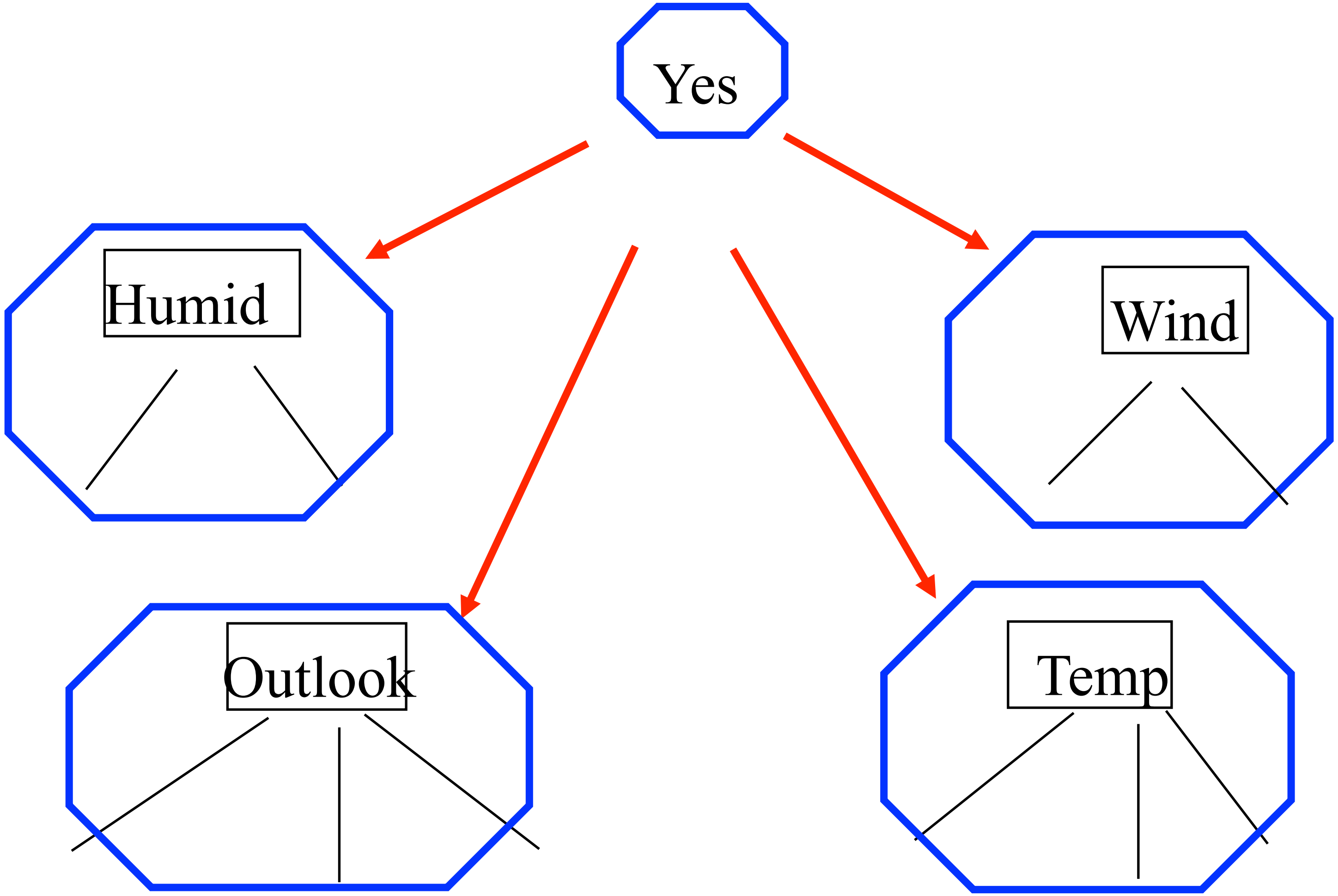
What is the “simplest” Tree?

- Always predict “yes”
  - A tree with one node

- How good it is?
  - Correct on 10 examples
  - Incorrect on 4 examples
  - Notation: [10+,4-]

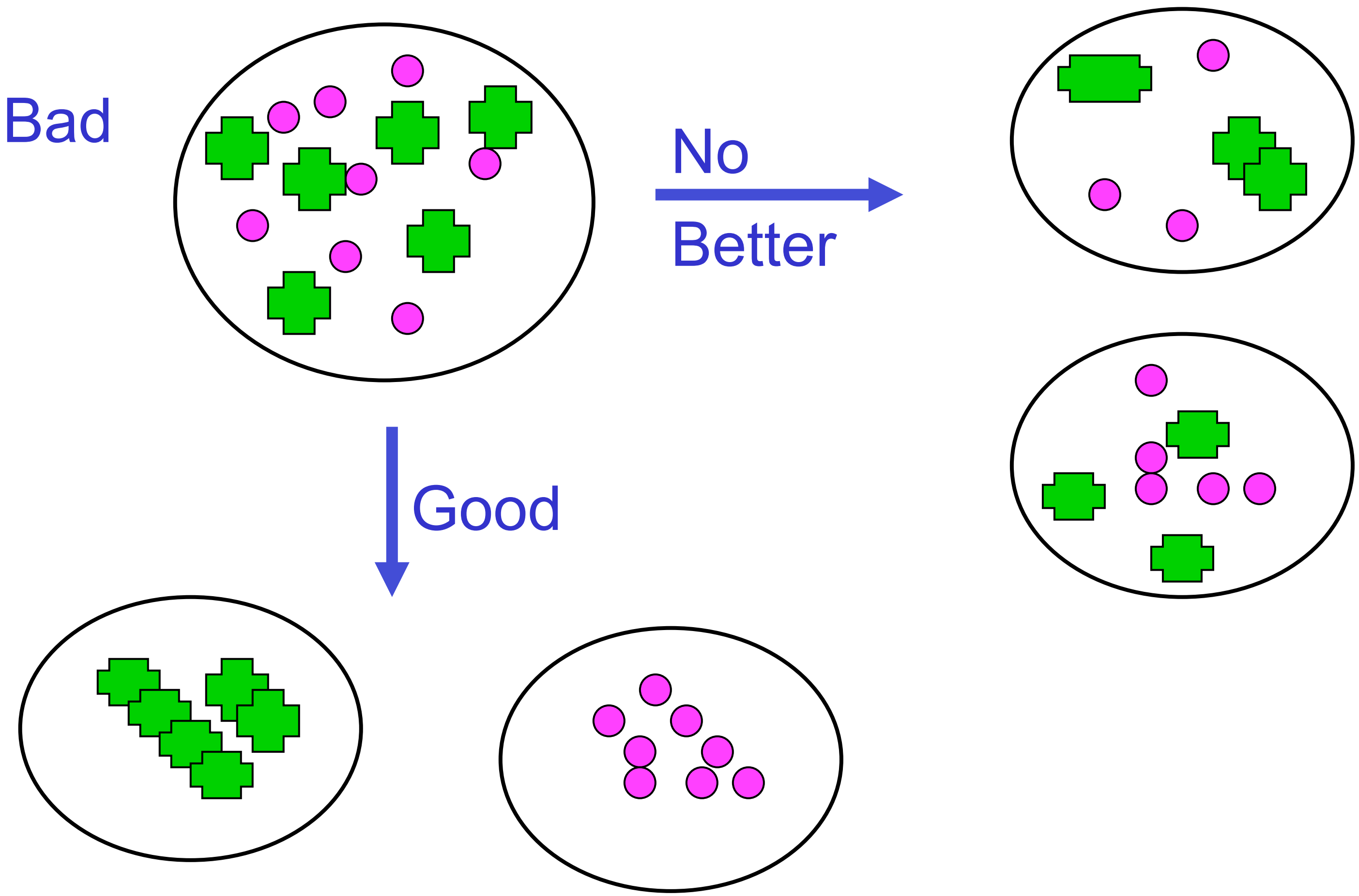
Day	Outlook	Temperature	Humidity	Wind	Play ball
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

Successors



Which attribute should we use to split?

Disorder is bad
Homogeneity is good



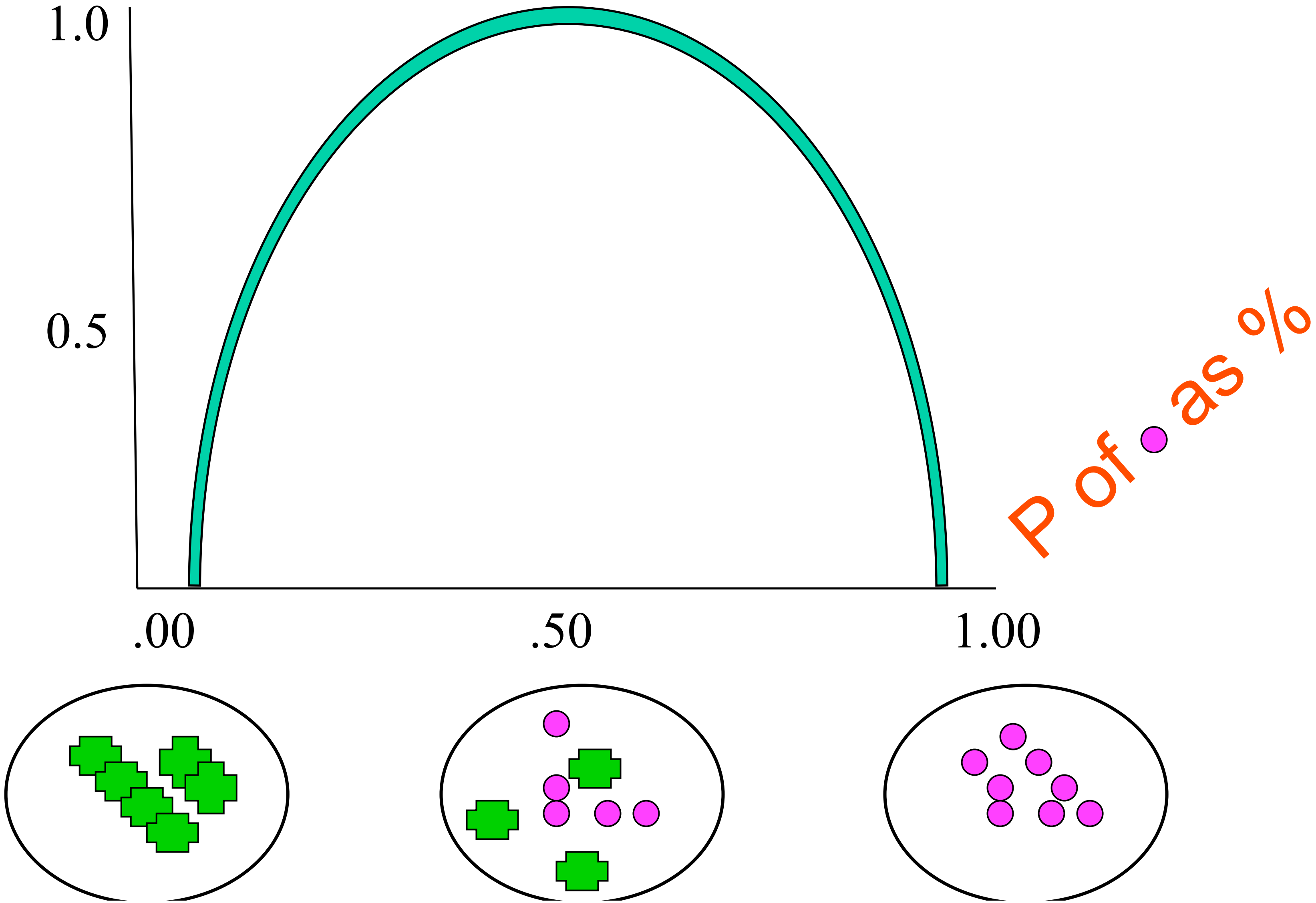
Using information theory to quantify uncertainty

- Entropy measures the amount of uncertainty in a probability distribution

- Entropy (or Information Content) of an answer to a question with possible answers $v_1, \ldots, v_n$:

  $$I(P(v_1), \ldots, P(v_n)) = \Sigma_i -P(v_i) \log 2\, P(v_i)$$

Entropy



Entropy (disorder) is bad
Homogeneity is good

- Let S be a set of examples
  - Labeled positive or negative
- Entropy(S) = -P log2(P) - N log2(N)
  - P is proportion of pos example
  - N is proportion of neg examples
  - and 0 log 0 == 0
- Example: S has 10 pos and 4 neg
  - Entropy([10+, 4-])

    = -(10/14) log2(10/14) - (4/14)log2(4/14)

    = 0.863

Information Gain

- Measure of expected reduction in entropy

$$\text{Gain}(S,A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} (|S_v| / |S|)\ \text{Entropy}(S_v)$$

  - Resulting from splitting along an attribute
  - $\text{Entropy}(S) = -P \log_2(P) - N \log_2(N)$

Gain of Splitting on Wind

Values(wind)=weak, strong

$S = [10+, 4-]$

$S_{weak} = [6+, 2-]$

$S_s = [3+, 3-]$

Gain(S, wind)

$$= \text{Entropy}(S) - \sum_{v \in \{weak, s\}} (|S_v| / |S|) \text{ Entropy}(S_v)$$

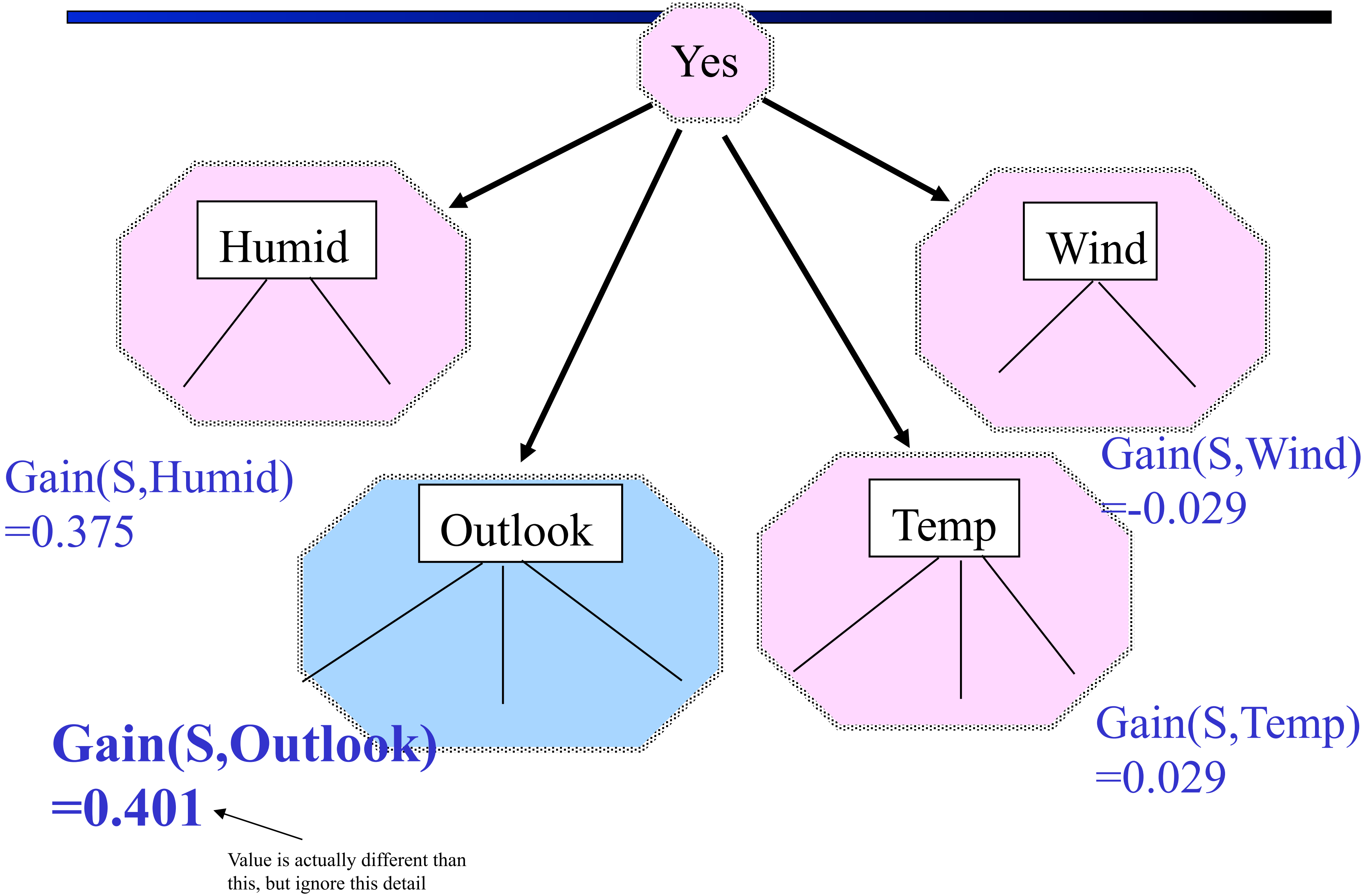
$$= \text{Entropy}(S) - 8/14 \text{ Entropy}(S_{weak}) - 6/14 \text{ Entropy}(S_s)$$

$= 0.863 - (8/14)\ 0.811 - (6/14)\ 1.00$

$= -0.029$

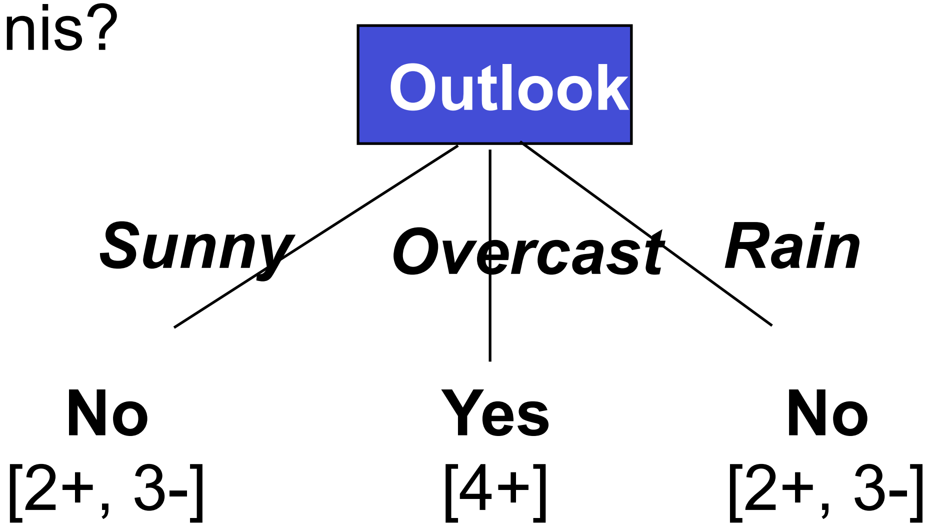
Day	Wind	Tennis?
d1	weak	n
d2	s	n
d3	weak	yes
d4	weak	yes
d5	weak	yes
d6	s	yes
d7	s	yes
d8	weak	n
d9	weak	yes
d10	weak	yes
d11	s	yes
d12	s	yes
d13	weak	yes
d14	s	n

Evaluating Attributes

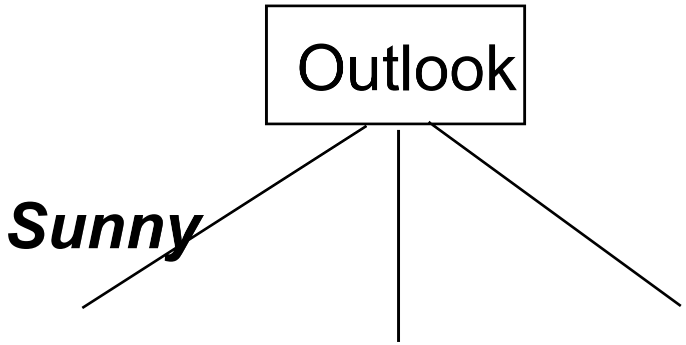


Resulting Tree ….

Good day for tennis?

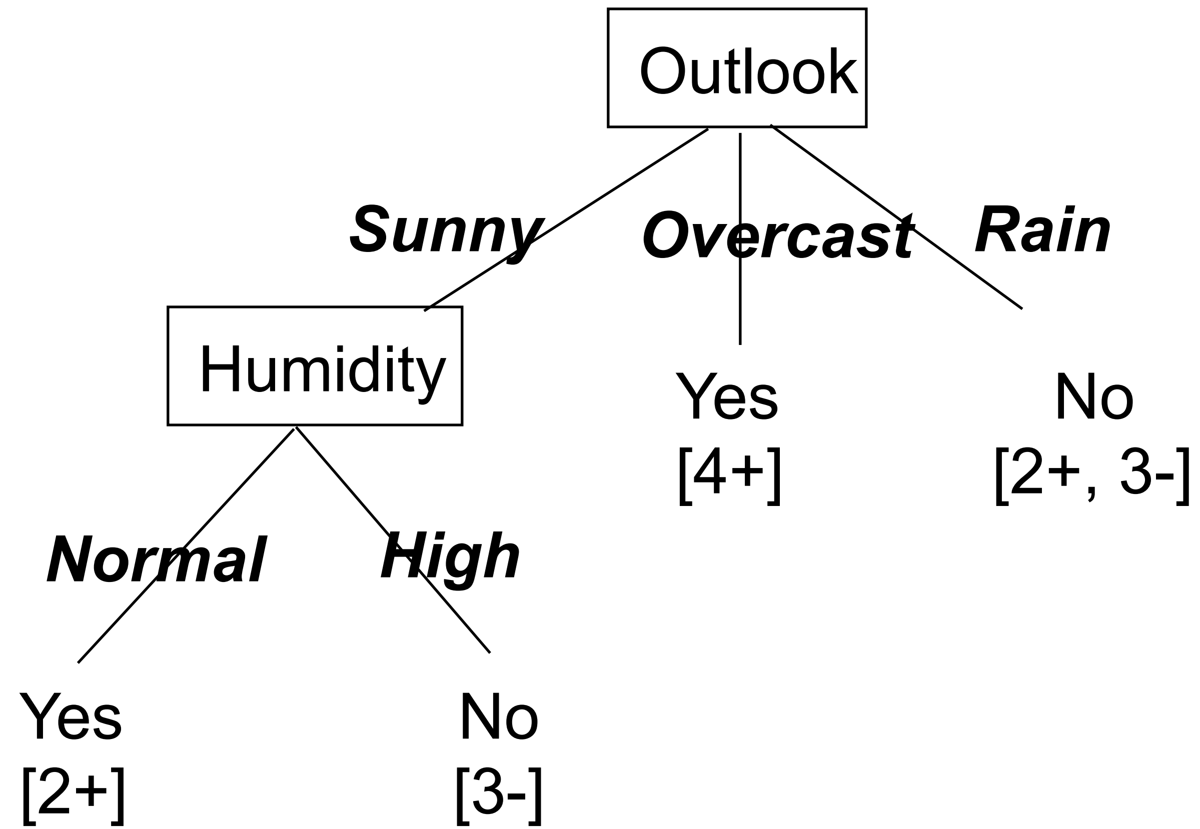


Recurse!



Day	Temp	Humid	Wind	Tennis?
d1	h	h	weak	n
d2	h	h	s	n
d8	m	h	weak	n
d9	c	n	weak	yes
d11	m	n	s	yes

One Step Later…

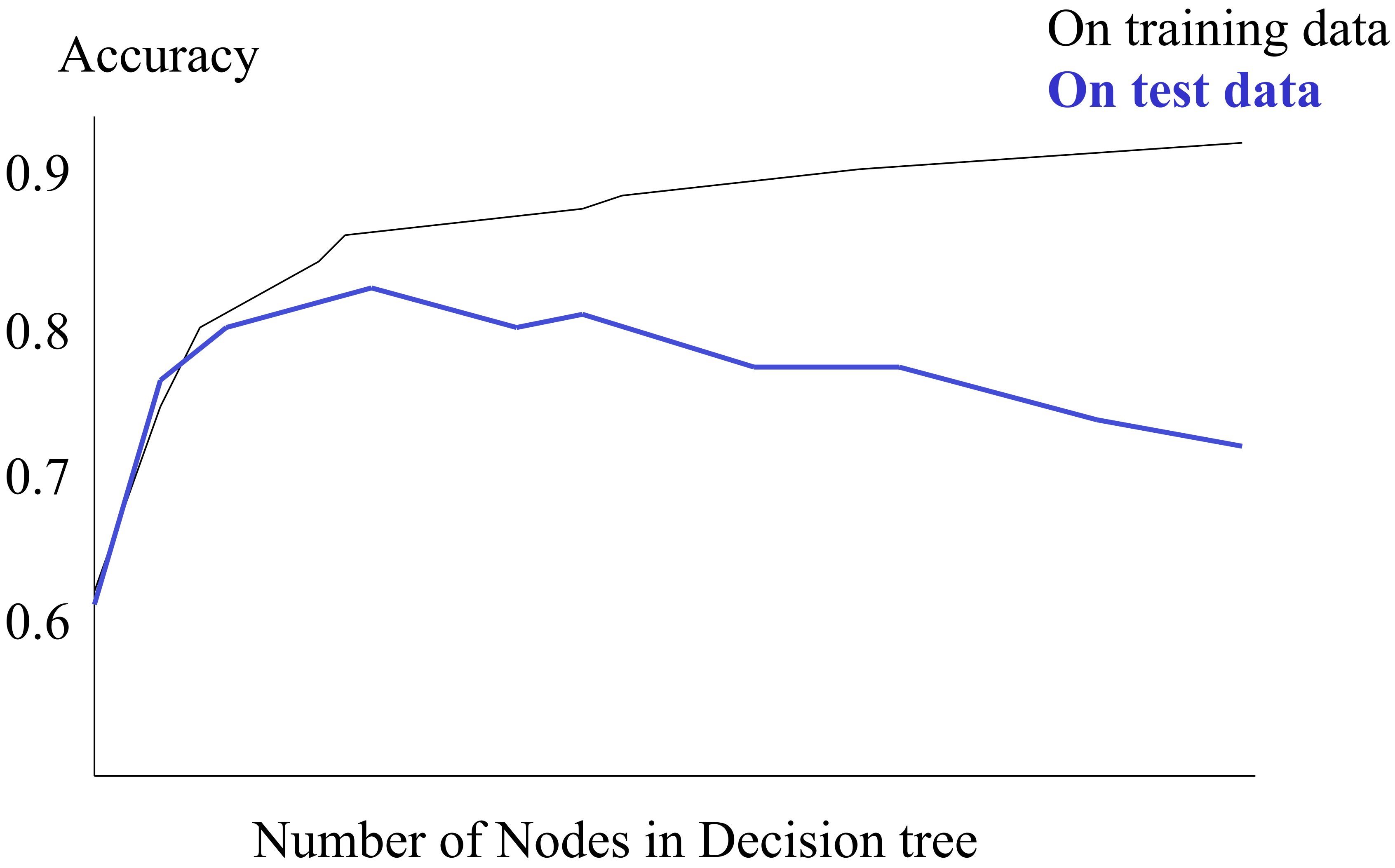


Decision Tree Algorithm

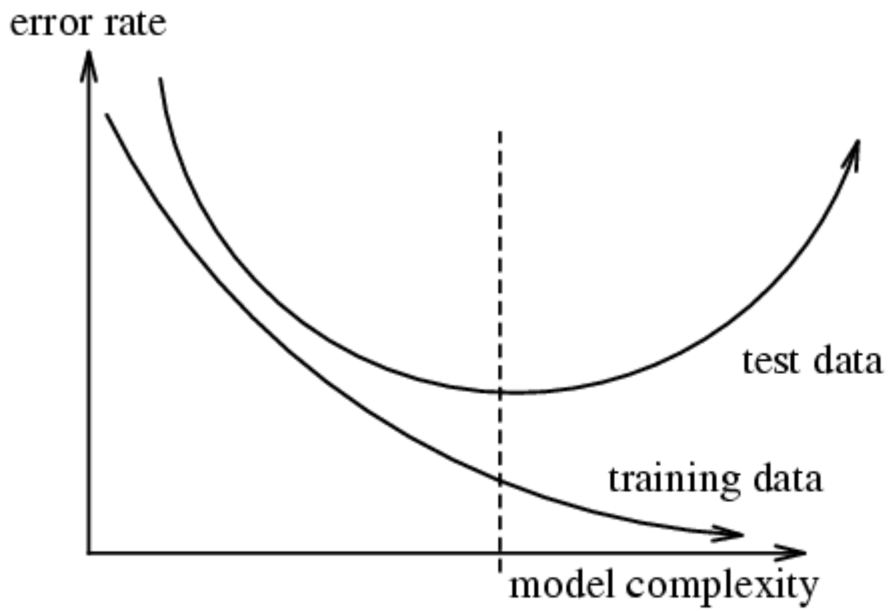
```
BuildTree(TrainingData)
        - Split(TrainingData)


Split(D)
        - If (all points in D are of the same class)
                Then Return
        - For each attribute A
                Evaluate splits on attribute A
        - Use best split to partition D into D1, D2
        - Split (D1)
        - Split (D2)
```

Overfitting



Overfitting



Overfitting

- DT is overfitting when there exists another DT ‘ and:
  - DT has smaller error on training examples, but
  - DT has bigger error on test examples

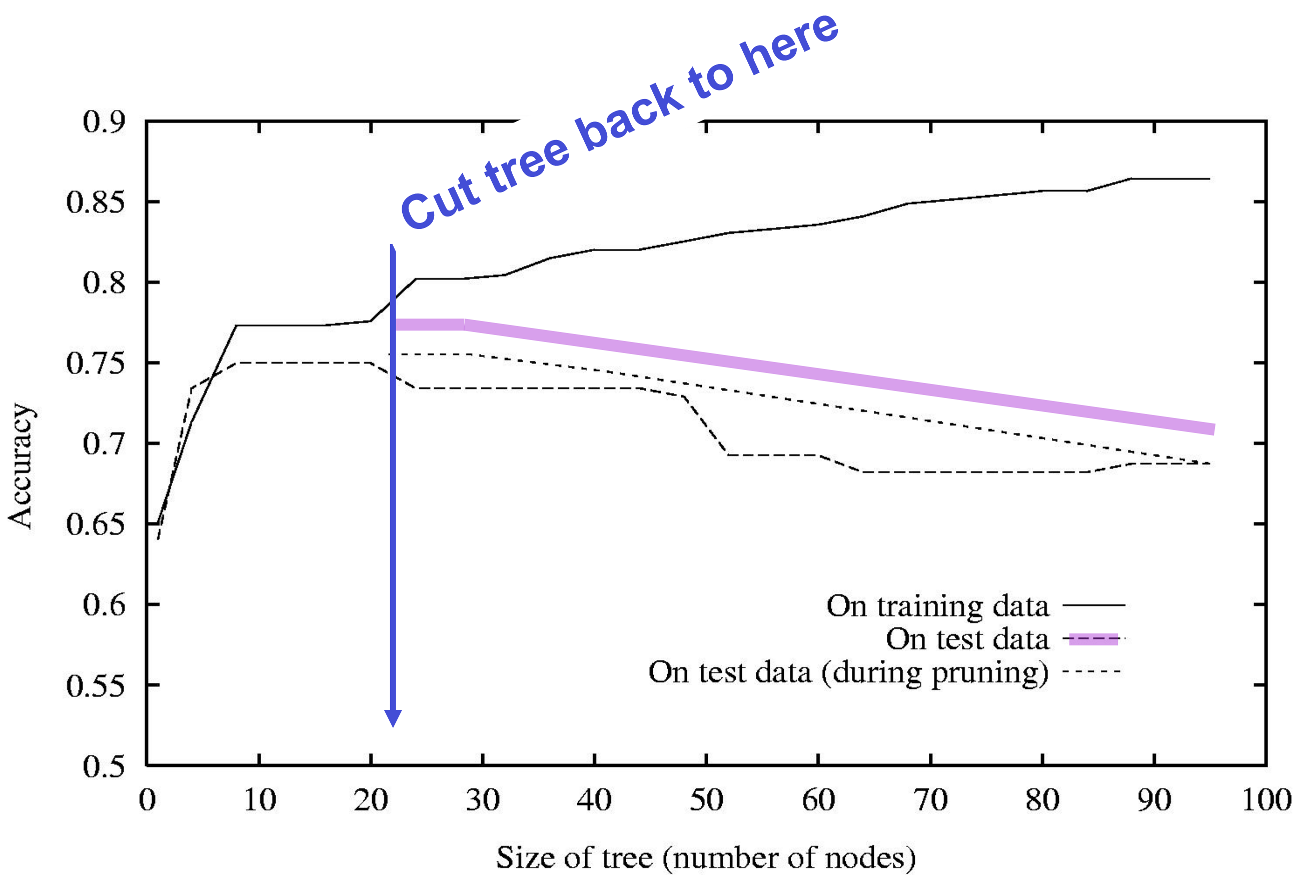
- Causes of overfitting
  - Noisy data, or
  - Training set is too small

Avoiding Overfitting

- How to prevent overfitting:
  - Stop growing the tree when data split is not statistically significant
  - Grow full tree, then post prune

- How to select best tree?
  - Measure performance on training data
  - Measure performance on a separate validation set
  - Add complexity penalty to the performance measure
  - Complexity: Number of nodes in the tree

Effect of Post Pruning



Other Features of Decision Tree

- Can handle continuous data
  - Input: Use threshold to split
  - Output: Estimate linear function at each leaf

- Can handle missing values
  - Use expectation taken from other samples

Other Classification Methods

- In the next lecture, we cover the following classifiers
  - Naïve Bayes Classifiers
  - Perceptrons
  - K-nearest neighbor

WEKA

- www.cs.waikato.ac.nz/ml/weka
- Tool with several classifiers
  - weka.classifiers.
  - bayes.NaiveBayes – Naïve Bayes
  - trees.DecisionStump – decision trees with one split only