Machine Learning with Python

Supervised Learning (Decision Trees)

Dr. Aeshah Alsughayyir

Collage of Computer Science and Engineering

Taibah University

2021-2022

Outline:

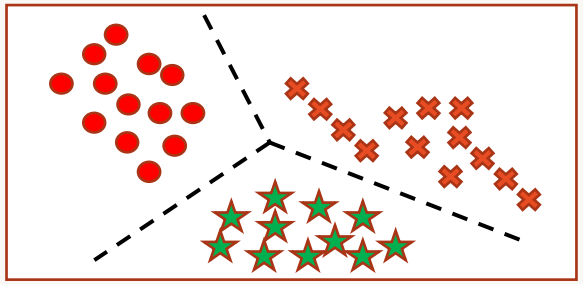
- Some recalls
- Supervised Learning Categories and Techniques
- Parametric and Nonparametric ML Algorithms
- **Inference with decision trees**
  - Decision Tree for *PlayTennis*
  - Decision Tree for Conjunction
  - Decision Tree for Disjunction
  - Decision Tree representing disjunctions of conjunctions
  - Converting a Tree to Rules
  - Example of a Decision Trees Representation
- When to use Decision Trees
- Top-down approach for creating Decision Trees
  - ID3 Algorithm
  - Search Space in Decision Tree learning
  - Entropy
  - Information Gain
- Training Examples
- Selecting the Best Attribute
- Issues in decision trees learning
- Avoid Overfitting in Decision Trees

Recall

(1) Supervised learning

Data is labeled and comes in pairs *(input, output)*$=(\mathbf{x}, y)$. We need to find a mapping $f(\mathbf{x}) = y$

- If y is a discrete label: classification
- If y is a number: regression

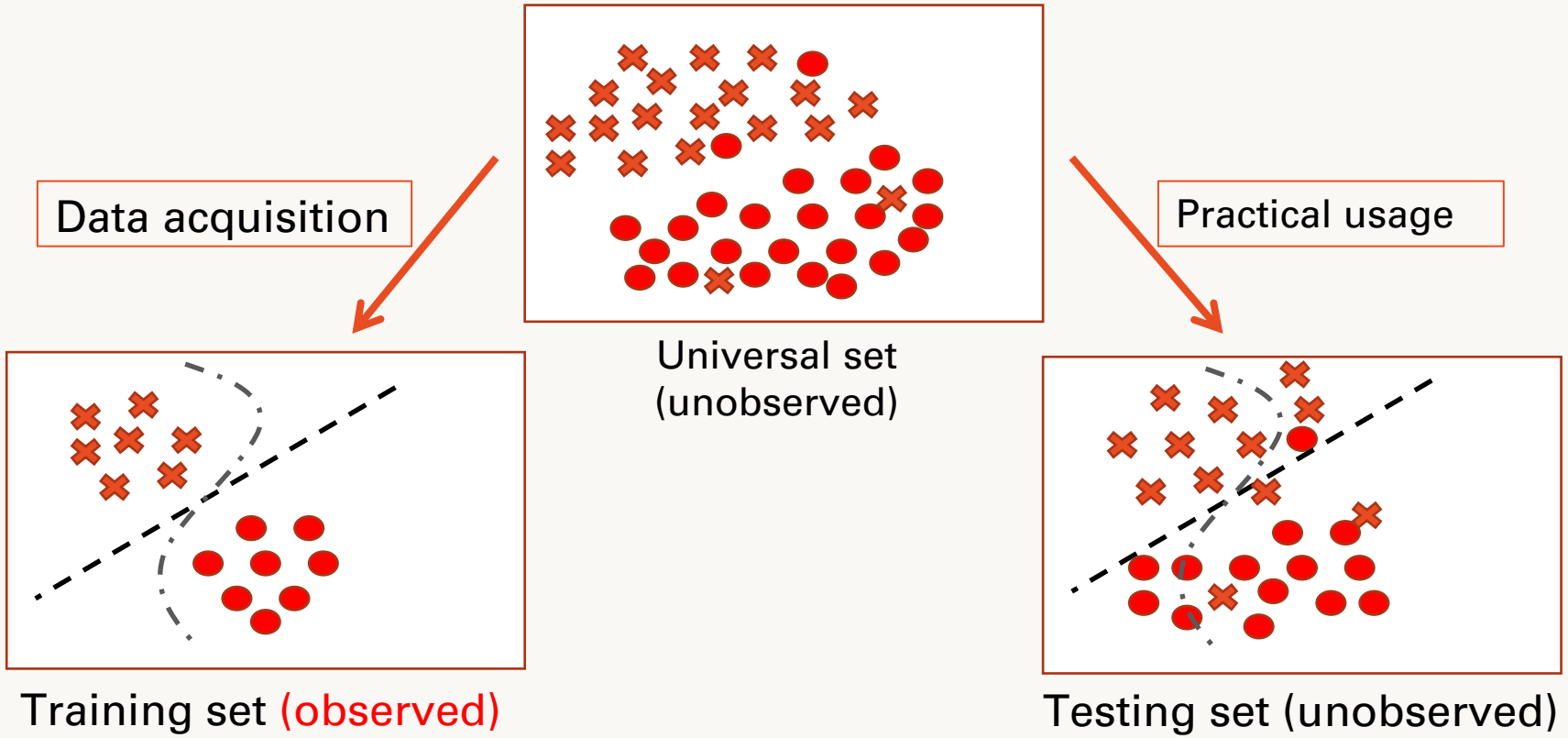


Supervised learning

Recall (cont.)

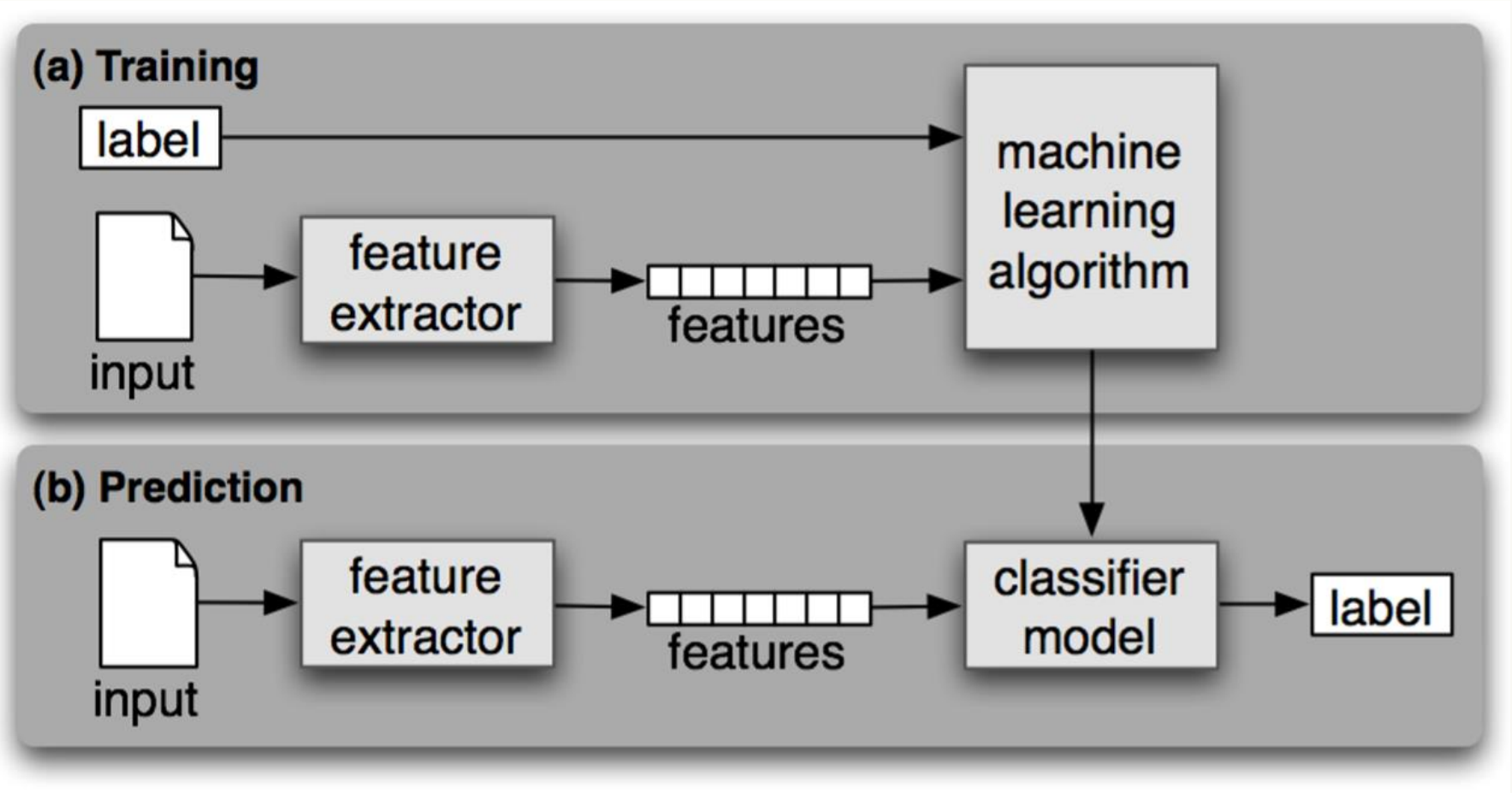
Supervised learning: *Training and prediction*

- **Training** is the process of making the system able to learn.
- **Training set** and **testing set** come from the same distribution



Recall (cont.)

Supervised learning: *Training and prediction* (cont.)



Supervised Learning Categories and Techniques

- **Linear (and non-linear) classifier** (numerical functions)
  - Logistic regression
  - Support vector machine (SVM)
  - Multi-layer perceptron (MLP)
- **Parametric** (Probabilistic functions)
  - Naïve Bayes
  - Hidden Markov models (HMM)
  - Probabilistic graphical models
- **Non-parametric** (Instance-based functions)
  - *K*-nearest neighbors.
- **Non-metric** (Symbolic functions)
  - Classification and regression tree (CART), decision tree
- **Aggregation**
  - Bagging (bootstrap + aggregation), Random forest, Adaboost

Parametric and Nonparametric ML Algorithms

Parametric ML Algorithms

A learning model that summarizes data with a set of parameters of fixed size (independent of the number of training examples) is called a parametric model. No matter how much data you throw at a parametric model, it won't change its mind about how many parameters it needs.
— Artificial Intelligence: A Modern Approach, page 737

Examples of parametric machine learning algorithms include:

- Logistic Regression
- Perceptron
- Naive Bayes
- Simple Neural Networks

Benefits:

- Simpler: These methods are easier to understand and interpret results.
- Speed: Parametric models are very fast to learn from data.
- Less Data: They do not require as much training data and can work well even if the fit to the data is not perfect.

Limitations:

- Constrained: By choosing a functional form these methods are highly constrained to the specified form.
- Limited Complexity: The methods are more suited to simpler problems.
- Poor Fit: In practice the methods are unlikely to match the underlying mapping function.

Non-Parametric ML Algorithms

Nonparametric methods are good when you have a lot of data and no prior knowledge, and when you don't want to worry too much about choosing just the right features.
— Artificial Intelligence: A Modern Approach, page 757

Examples of popular nonparametric machine learning algorithms are:

- k-Nearest Neighbors
- Decision Trees like CART and C4.5
- Support Vector Machines

Benefits:

- **Flexibility**: Capable of fitting a large number of functional forms.
- **Power**: No assumptions (or weak assumptions) about the underlying function.
- **Performance**: Can result in higher performance models for prediction.

Limitations:

- **More data**: Require a lot more training data to estimate the mapping function.
- **Slower**: A lot slower to train as they often have far more parameters to train.
- **Overfitting**: More of a risk to overfit the training data and it is harder to explain why specific predictions are made.

Inference with decision trees

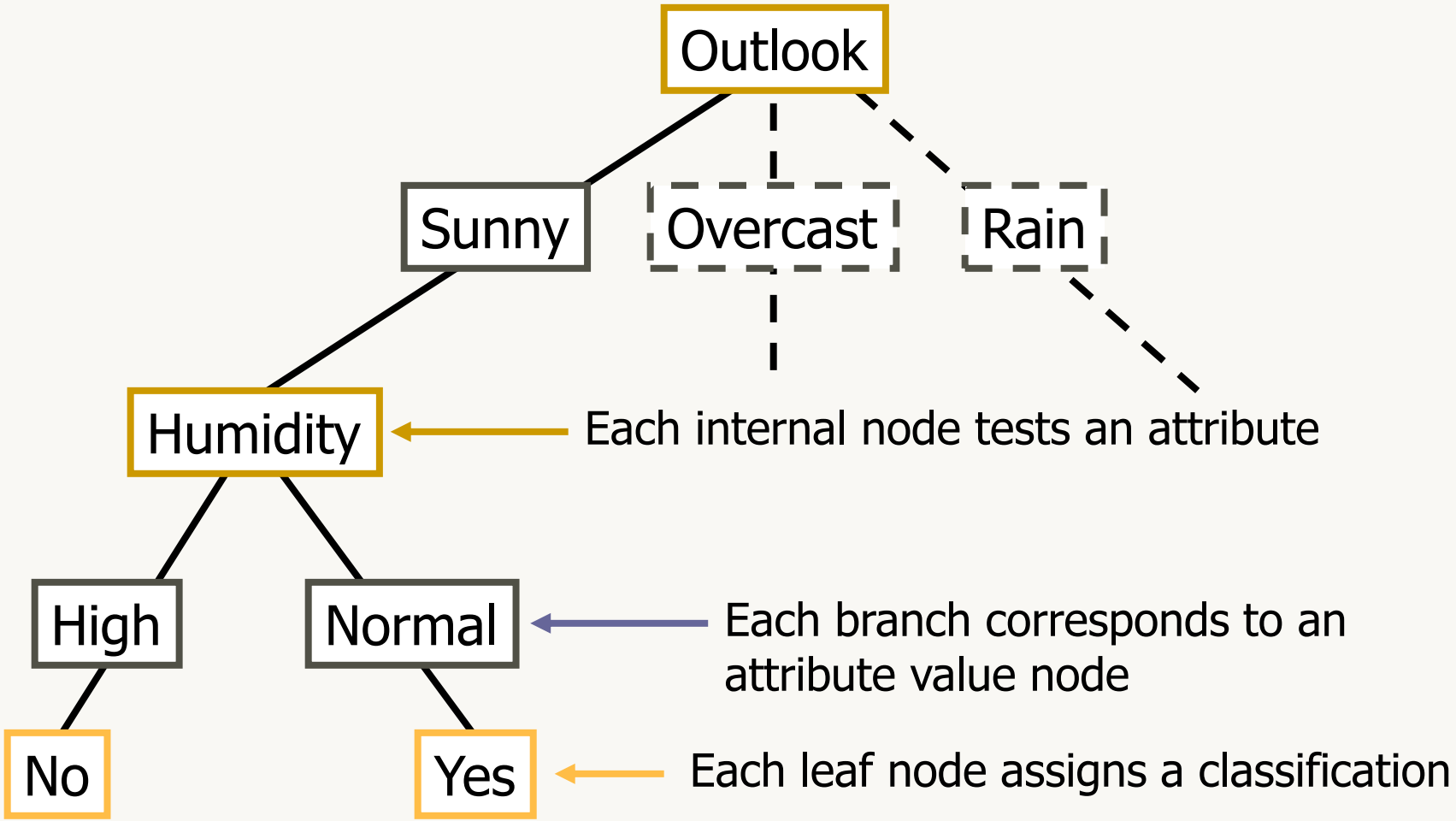
Decision Trees

is one of the most widely used and practical methods of *inference*

Features

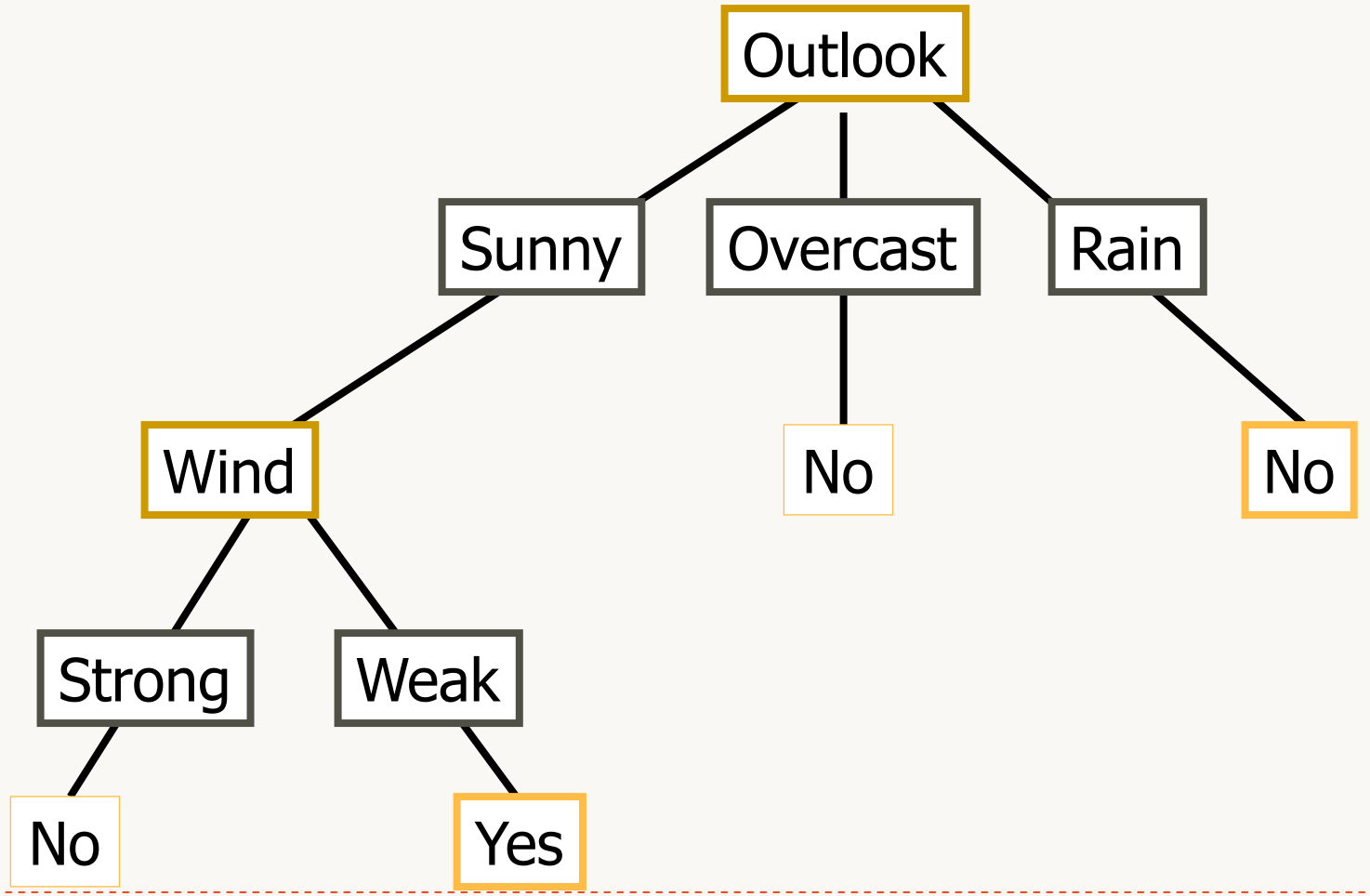
- Method for approximating *discrete-valued* functions (including Boolean)
- Learned functions are represented as *decision trees* (or *if-then-else* rules)
- Expressive hypotheses space, including disjunction
- Robust to noisy data

Decision Tree for *PlayTennis*



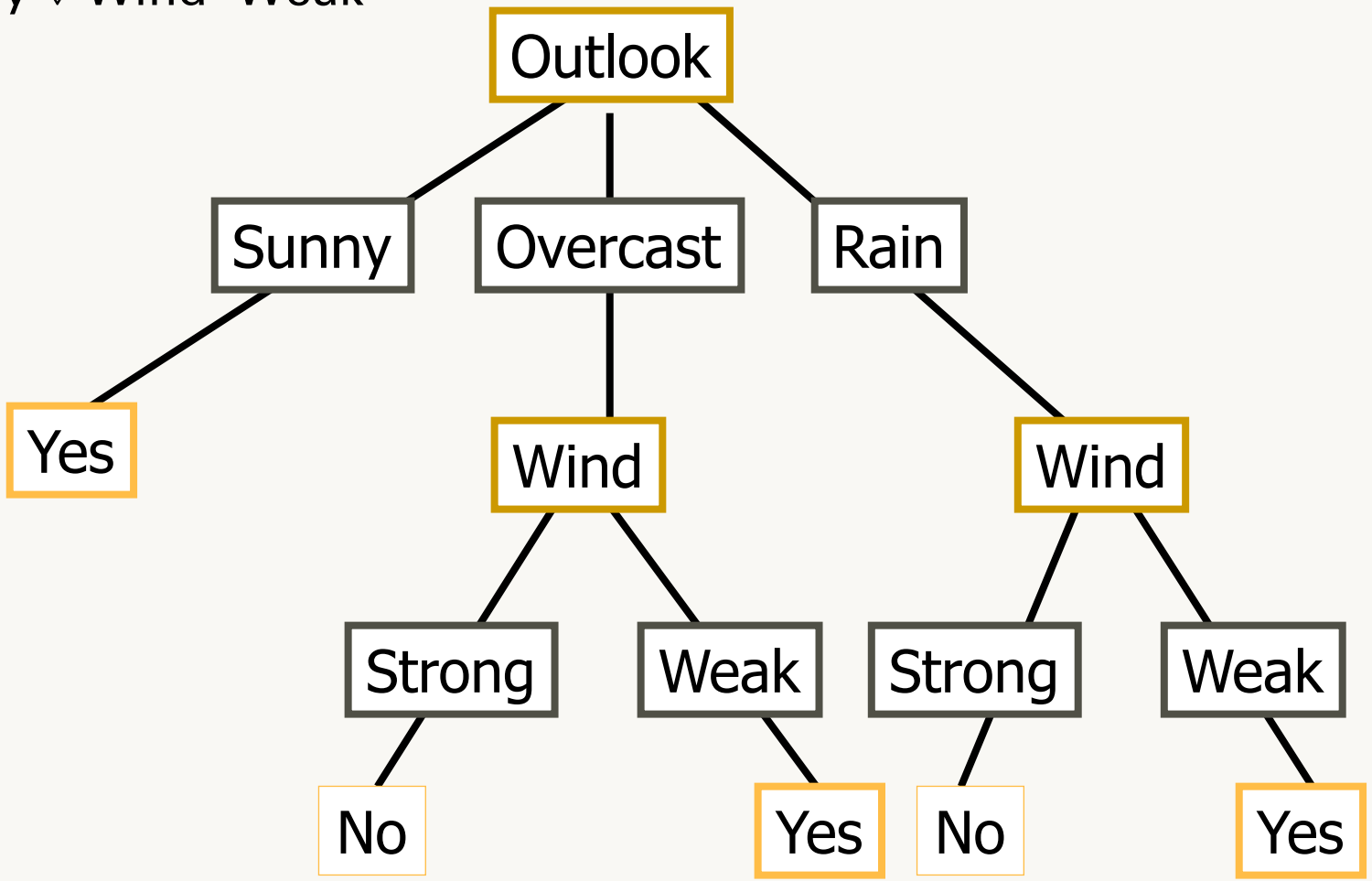
Decision Tree for Conjunction

Outlook=Sunny $\wedge$ Wind=Weak

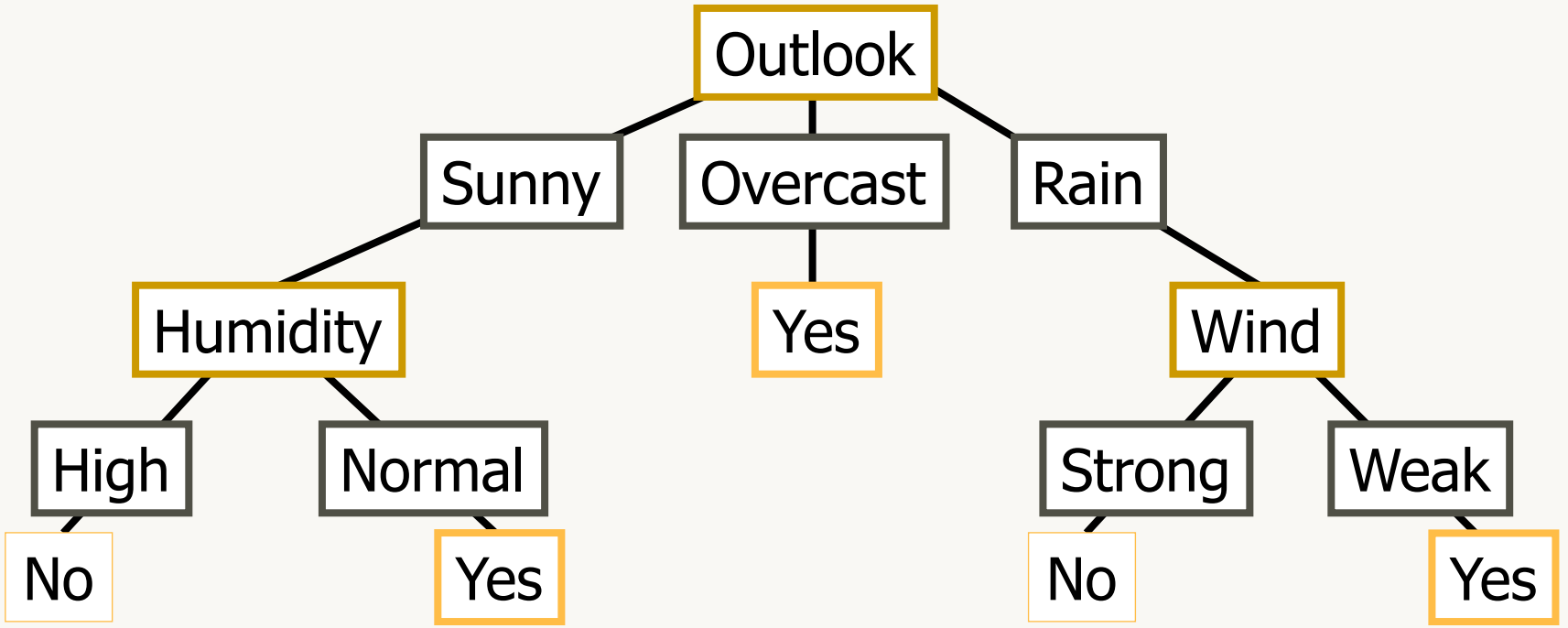


Decision Tree for Disjunction

Outlook=Sunny $\vee$ Wind=Weak

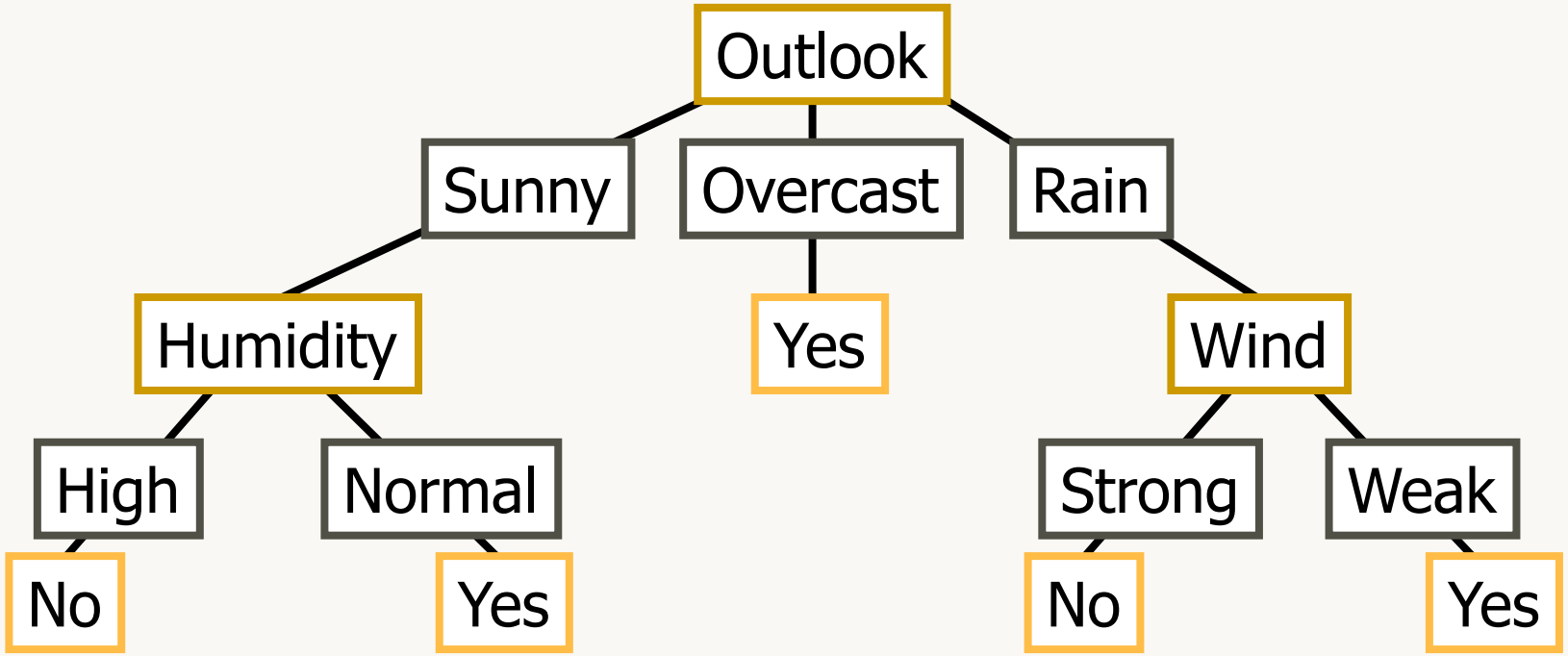


Decision Tree representing disjunctions of conjunctions



(Outlook=Sunny $\wedge$ Humidity=Normal)
$\vee$ (Outlook=Overcast)
$\vee$ (Outlook=Rain $\wedge$ Wind=Weak)

Converting a Tree to Rules



R_1: If (Outlook=Sunny) $\wedge$ (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) $\wedge$ (Humidity=Normal) Then PlayTennis=Yes
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) $\wedge$ (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) $\wedge$ (Wind=Weak) Then PlayTennis=Yes

Example: Decision Trees Representation

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .2
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

The notation [n+, m-] means n positive examples and m negative examples

When to use Decision Trees

- Problem characteristics:
  - Instances can be described by attribute value pairs
  - Target function is discrete valued
  - Disjunctive hypothesis may be required
  - Possibly noisy training data samples
    - Robust to errors in training data
    - Missing attribute values

- Different classification problems:
  - Equipment or medical diagnosis
  - Credit risk analysis
  - Several tasks in natural language processing
  - *And many others …*

Top-down approach for creating Decision Trees

- ID3 (Quinlan, 1986) is a basic algorithm for learning DT's
- Given a training set of examples, the algorithms for building DT performs search in the space of decision trees
- The construction of the tree is top-down. The algorithm is greedy.
- The fundamental question is "which attribute should be tested next? Which question gives us more information?"
- Select the ***best*** attribute
- A descendent node is then created for each possible value of this attribute and examples are partitioned according to this value
- The process is repeated for each successor node until all the examples are classified correctly or there are no attributes left.

C4.5 algorithm: used for continues (*numeric*) values .. read it yourself

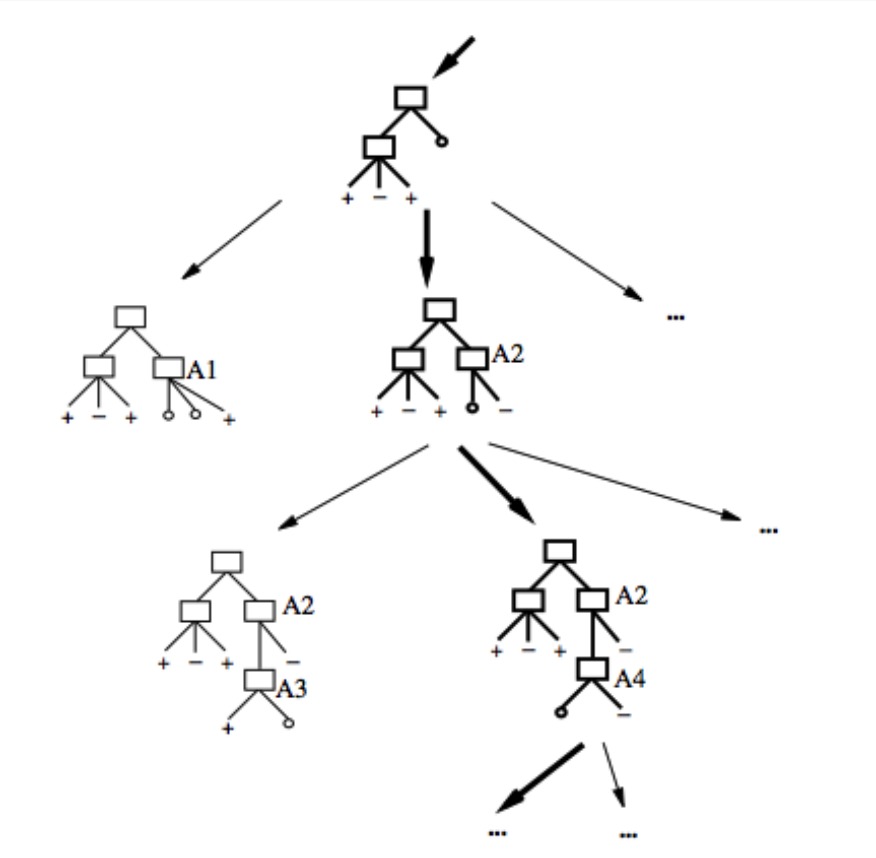
ID3: Algorithm

ID3(*X, T, Attrs*) *X*: training examples:
T: target attribute (e.g. *PlayTennis*),
Attrs: other attributes, initially all attributes

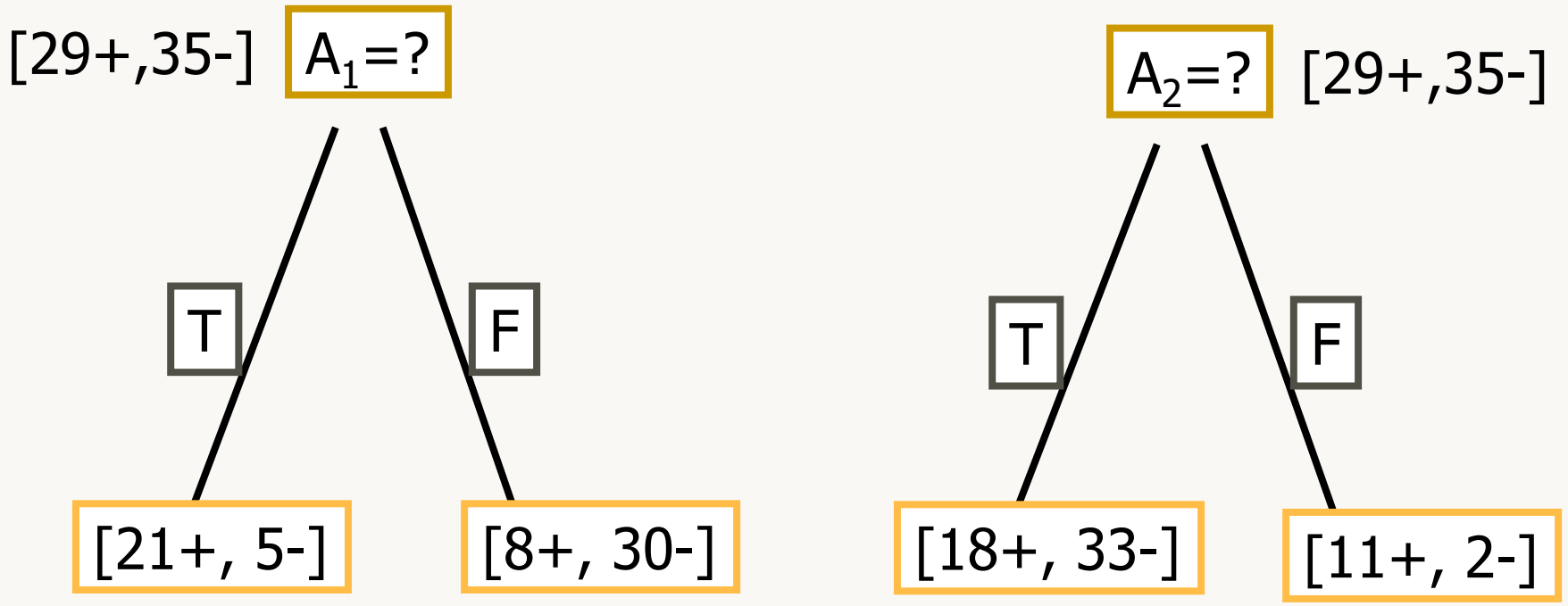
Create *Root* node
If all X's are +, *return Root* with class +
If all X's are –, *return Root* with class –
If Attrs is empty *return Root* with class most common value of *T* in *X*
else
  $A \leftarrow$ best attribute; decision attribute for Root $\leftarrow A$
  For each possible value v_i of *A:*
   - add a new branch below *Root,* for test $A = v_i$
   - $X_i \leftarrow$ subset of *X* with $A = v_i$
   - *If* X_i is empty *then* add a new leaf with class the most common value of *T* in *X*
      else add the subtree generated by ID3(X_i, *T, Attrs* – {*A*})
return Root

Search Space in Decision Tree learning

- The search space is made by partial decision trees
- The algorithm is *hill-climbing*
- The evaluation function is *information gain*
- The hypotheses space is complete (*represents all discrete-valued function*s)
- The search maintains a single current hypothesis
- No backtracking; no guarantee of optimality
- It uses all the available examples (not incremental)
- May terminate earlier, accepting noisy classes



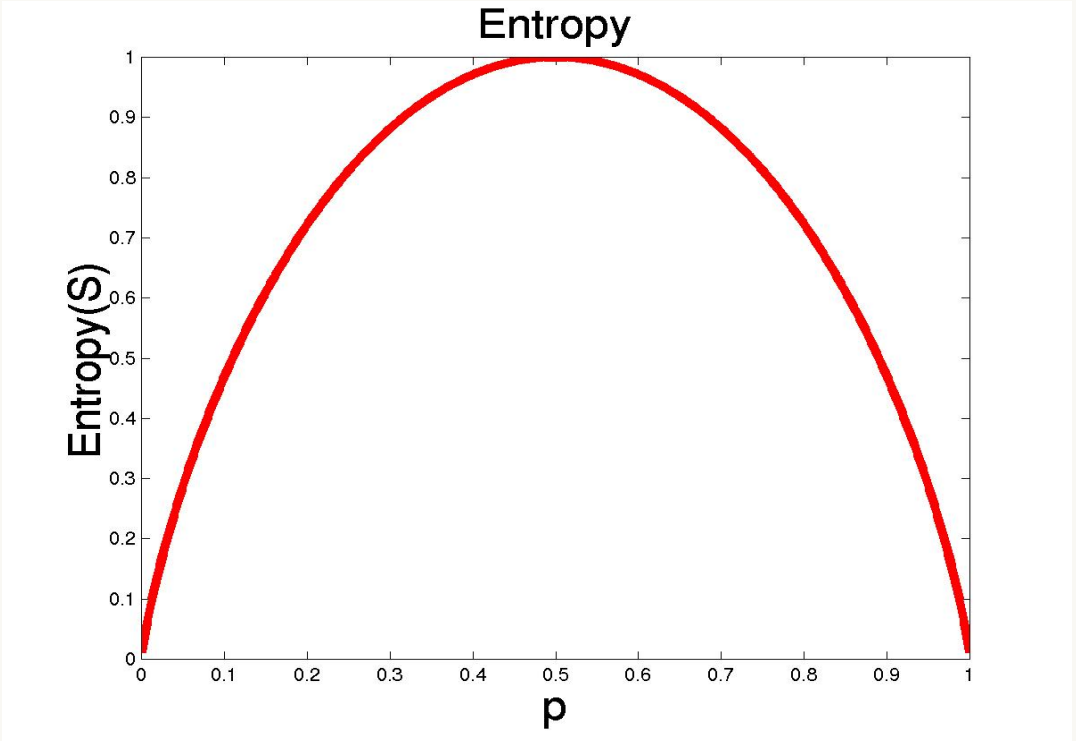
Which attribute is best?



- A statistical property called ***information gain***, measures how well a given attribute separates the training examples
- Information gain uses the notion of ***entropy***, commonly used in information theory
- *Information gain = expected reduction of entropy*

Entropy

- **S** is a sample of training examples
- p_+ is the proportion of positive examples
- p_- is the proportion of negative examples
- Entropy measures the impurity of **S**



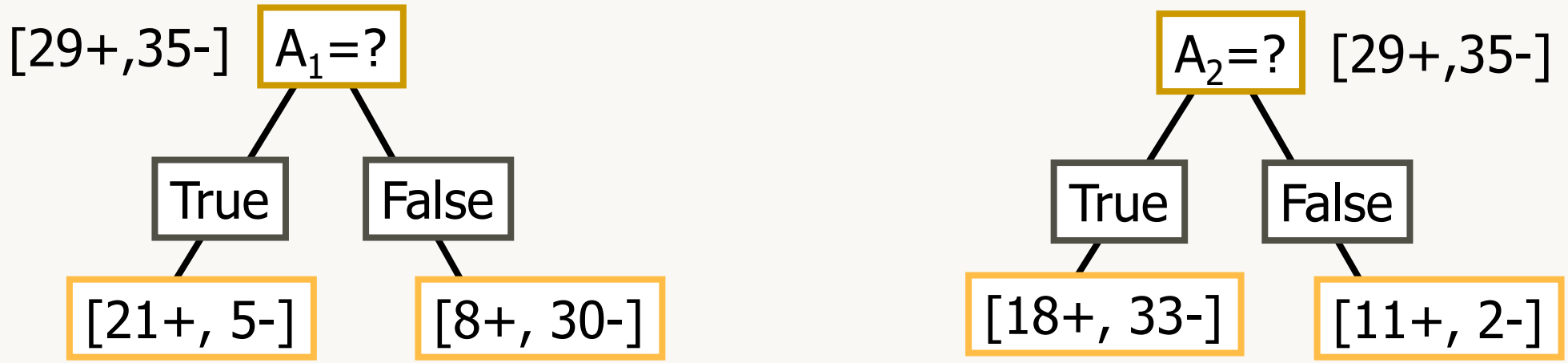
$$\text{Entropy}(S) = -(p_+)\log_2(p_+) - (p_-)\log_2(p_-)$$

If entropy == 0 ➔ the best classification
If entropy == 1 ➔ the worse classification

Information Gain (S=E)

- Gain(S, A): expected reduction in entropy due to sorting S on attribute A
- The higher the information gain the more effective the attribute in classifying training data.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in D_A} \frac{|S_v|}{|S|} Entropy(S_v)$$



Entropy ([29+,35-]) = -29/64 log2 29/64 – 35/64 log2 35/64
= 0.99

Information Gain

Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

Gain(S, A_1)= Entropy(S)

- 26/64 * Entropy([21+,5-])
- 38/64 * Entropy([8+,30-])

=0.27

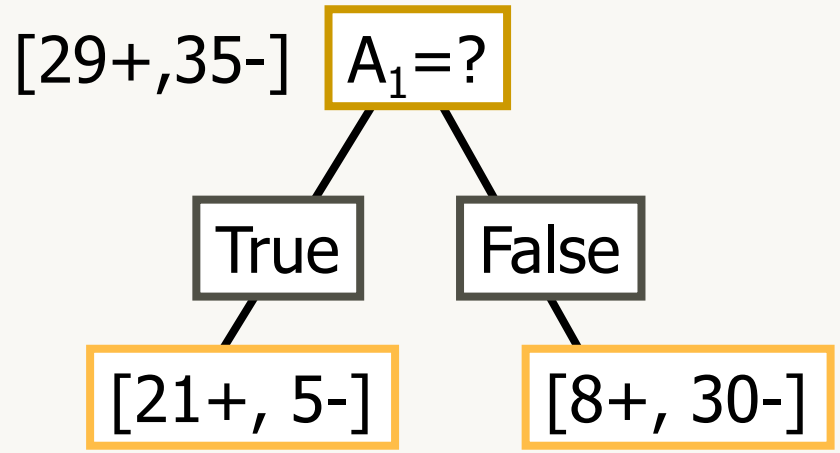
Entropy([18+,33-]) = 0.94

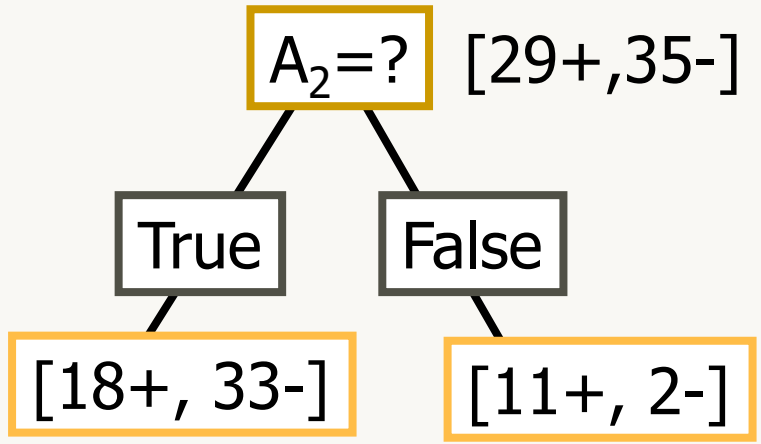
Entropy([11+,2-]) = 0.62

Gain(S,A2)= Entropy(S)

- 51/64 * Entropy([18+,33-])
- 13/64 * Entropy([11+,2-])

=0.12

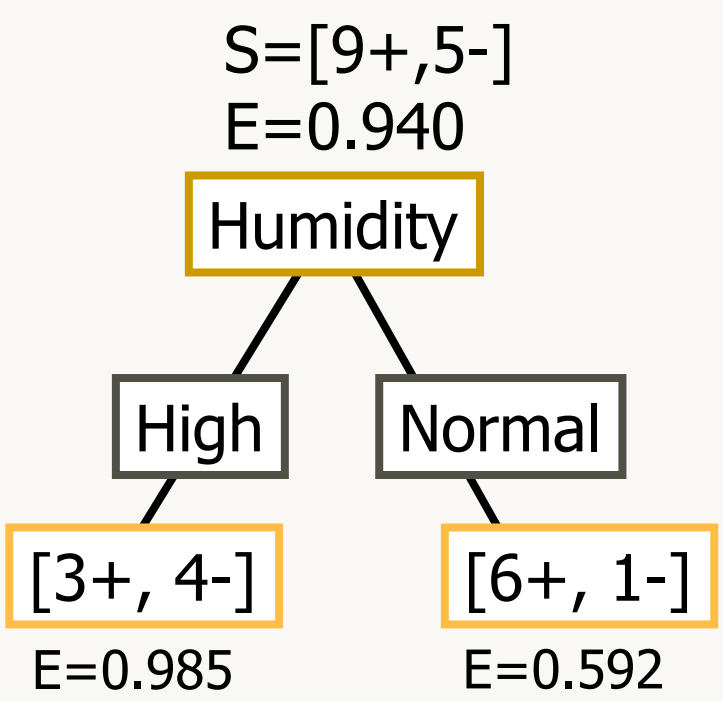




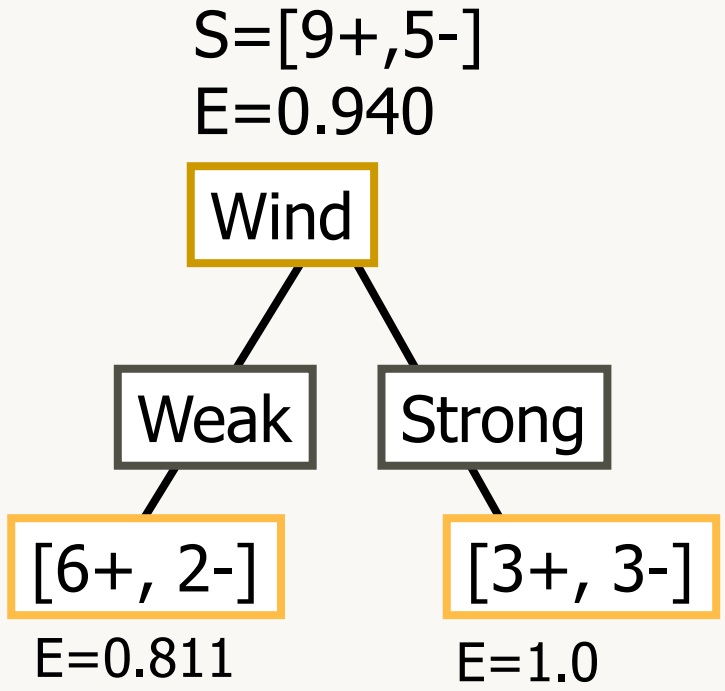
Training Examples

Day	Outlook	Temp.	Humidity	Wind	Play Tennis
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	Weak	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cold	Normal	Weak	Yes
D10	Rain	Mild	Normal	Strong	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

Selecting the Best Attribute



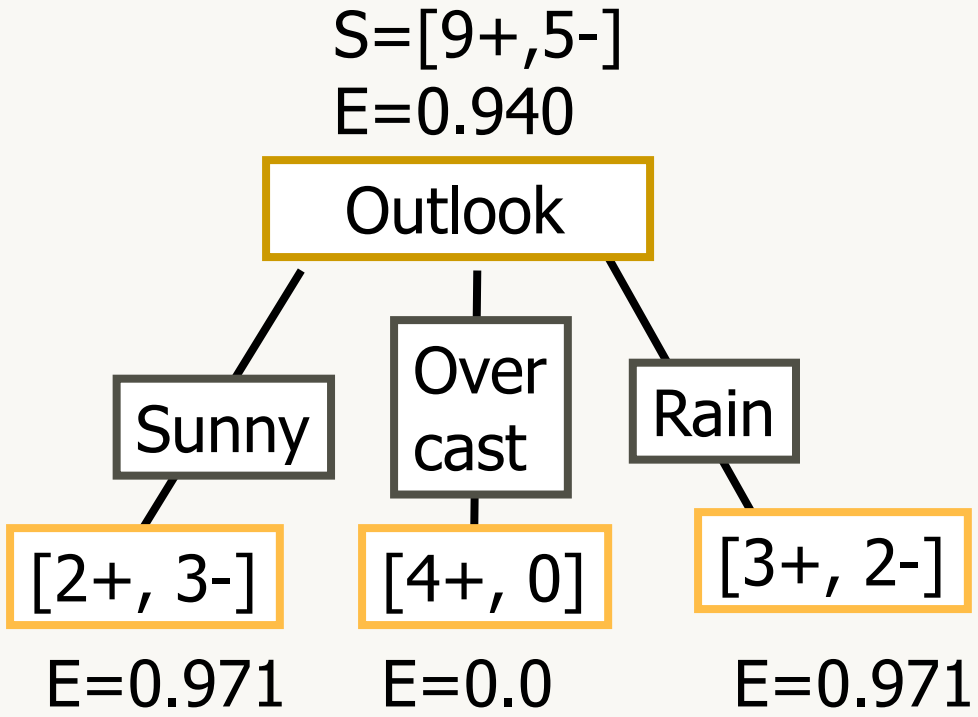
Gain(S,Humidity)
=0.940-(7/14)*0.985
– (7/14)*0.592
=0.151



Gain(S,Wind)
=0.940-(8/14)*0.811
– (6/14)*1.0
=0.048

Humidity provides greater info. gain than Wind, w.r.t target classification.

Selecting the Best Attribute



Gain(S,Outlook)
=0.940-(5/14)*0.971
-(4/14)*0.0 – (5/14)*0.0971
=0.247

Temperature

Do it yourself !

First step: which attribute to test at the root?

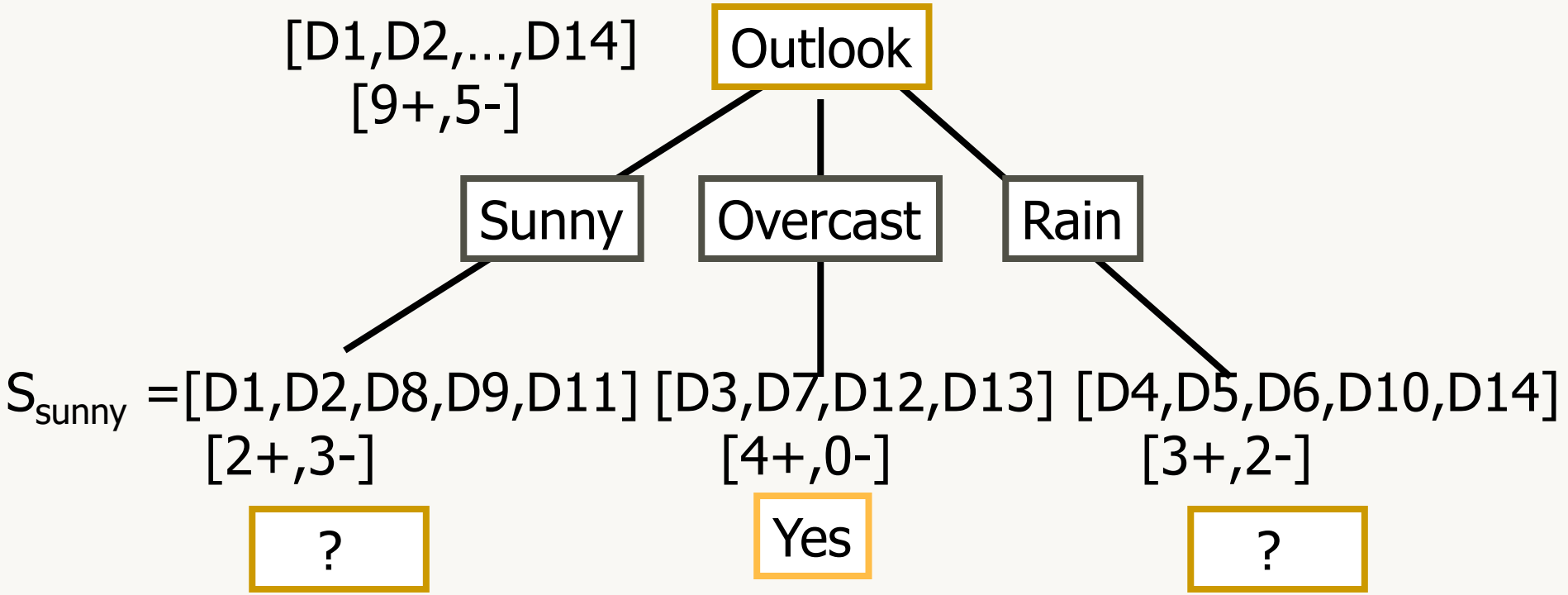
The information gain values for the 4 attributes are:

- Gain(S, Outlook) =0.247
- Gain(S, Humidity) =0.151
- Gain(S, Wind) =0.048
- Gain(S, Temperature) =0.029

where S denotes the collection of training examples

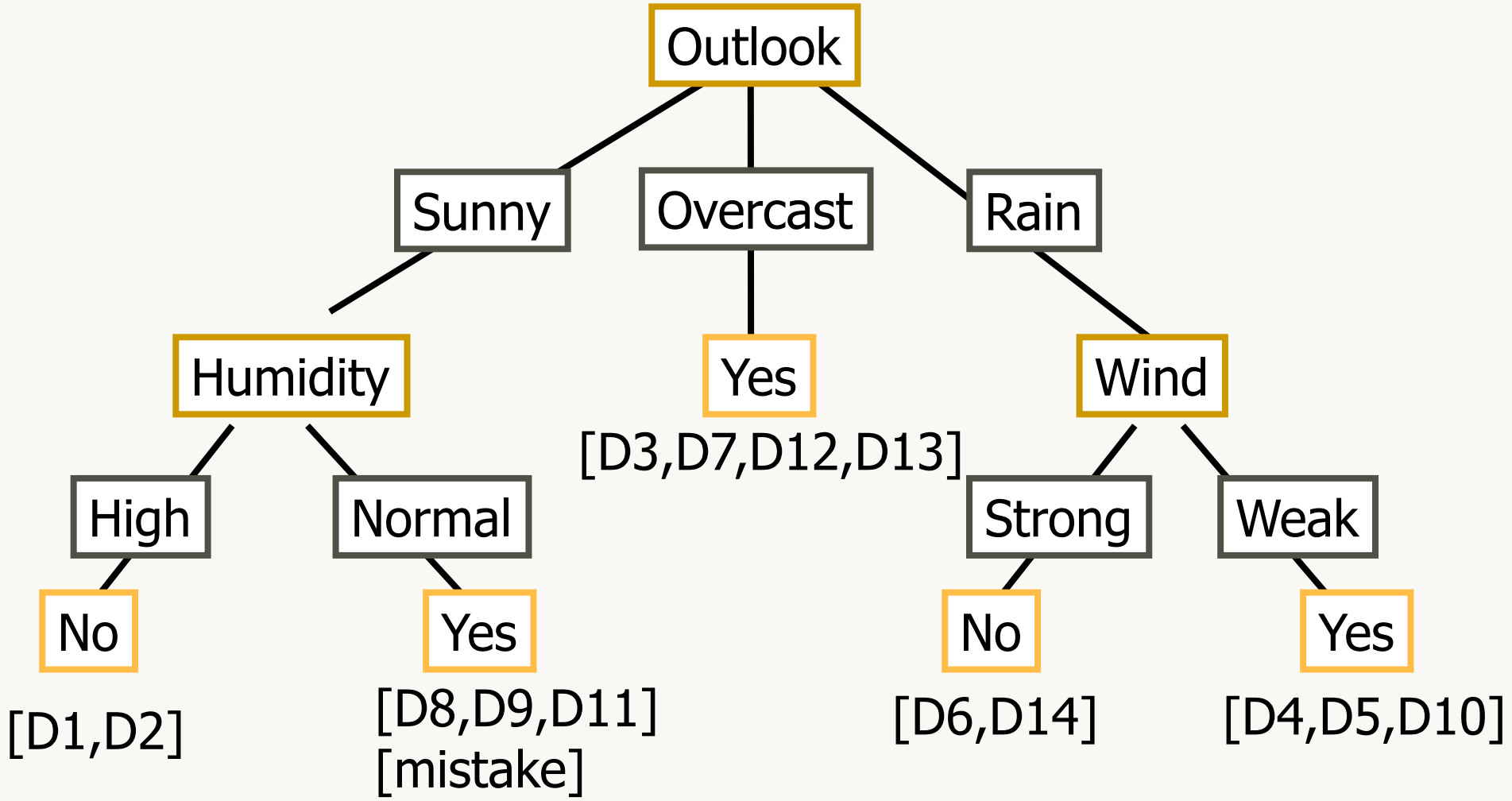
- Outlook provides the best prediction for the target
- Lets grow the tree:
  - add to the tree a successor for each possible value of Outlook
  - partition the training samples according to the value of Outlook

Next step



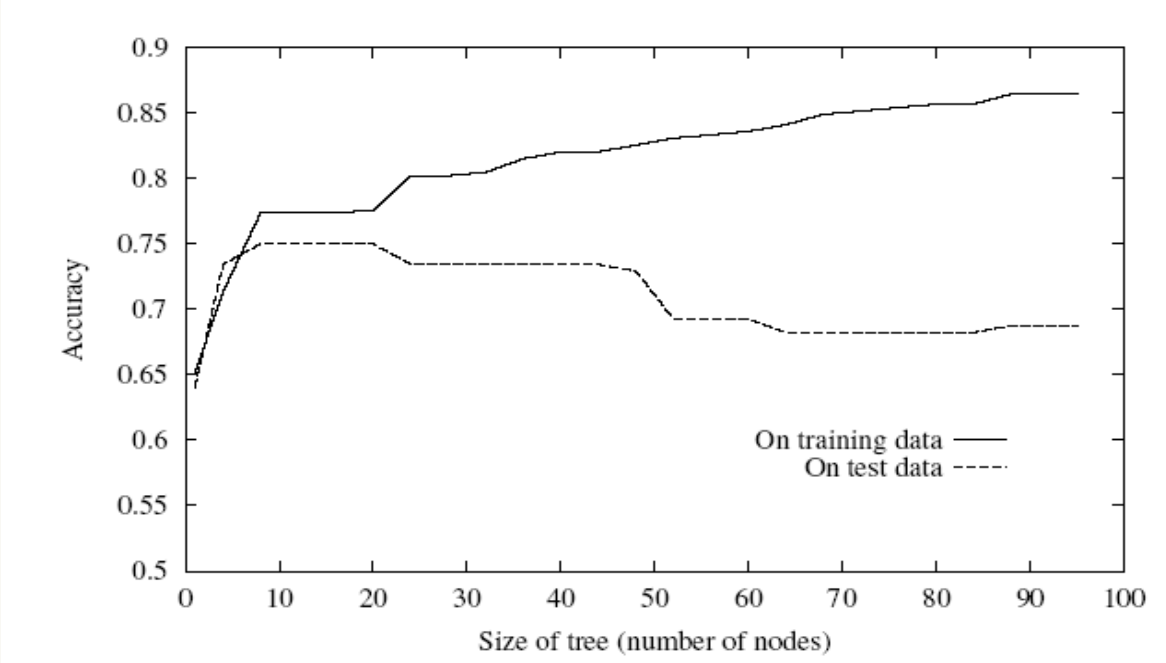
Gain(S_{sunny}, Humidity)=0.970-(3/5)0.0 – 2/5(0.0) = 0.970
Gain(S_{sunny}, Temp.)=0.970-(2/5)0.0 –2/5(1.0)-(1/5)0.0 = 0.570
Gain(S_{sunny}, Wind)=0.970= -(2/5)1.0 – 3/5(0.918) = 0.019

Next step


Outlook
Sunny
Overcast
Rain
Humidity
Yes
[D3,D7,D12,D13]
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes
[D1,D2]
[D8,D9,D11]
[mistake]
[D6,D14]
[D4,D5,D10]

Issues in decision trees learning

- Overfitting
  - Reduced error pruning
  - Rule post-pruning



- Extensions
  - Continuous valued attributes
  - Alternative measures for selecting attributes
  - Handling training examples with missing attribute values
  - Improving computational efficiency, etc.

One of the biggest problems with decision trees is **Overfitting**

Avoid overfitting in Decision Trees

- Two strategies:
  - Stop growing the tree earlier, before perfect classification
  - Allow the tree to *overfit* the data, and then *post-prune* the tree
- Training and validation set
  - Split the training in two parts (training and validation) and use validation to assess the utility of *post-pruning*
    - *Reduced error pruning*
    - *Rule pruning*
- Other approaches
  - Use a statistical test to estimate effect of expanding or pruning
  - *Minimum description length principle*: uses a measure of complexity of encoding the DT and the examples, and cut growing the tree when this encoding size is minimal

Any questions?