# Outline

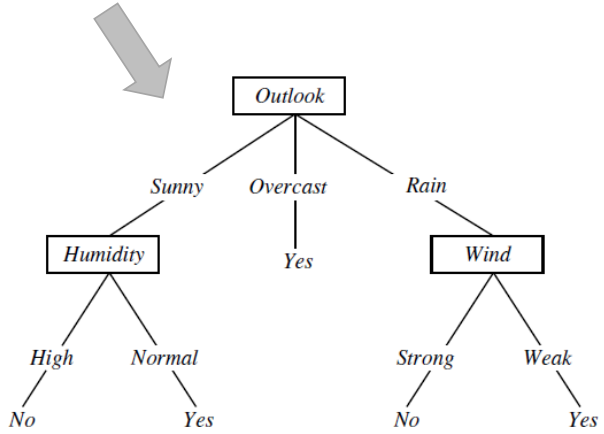
- Classification basics
- Decision tree classifiers
- Overfitting
- Lazy vs Eager Learners
- k-Nearest Neighbors (or learning from your neighbors)
- Evaluation of classifiers

# Decision tree (DTs) classifiers

- One of the most popular classification methods
- DTs are included in many commercial systems nowadays
- Easy to interpret, human readable, intuitive
- Simple and fast methods

*Training set*

| 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 |



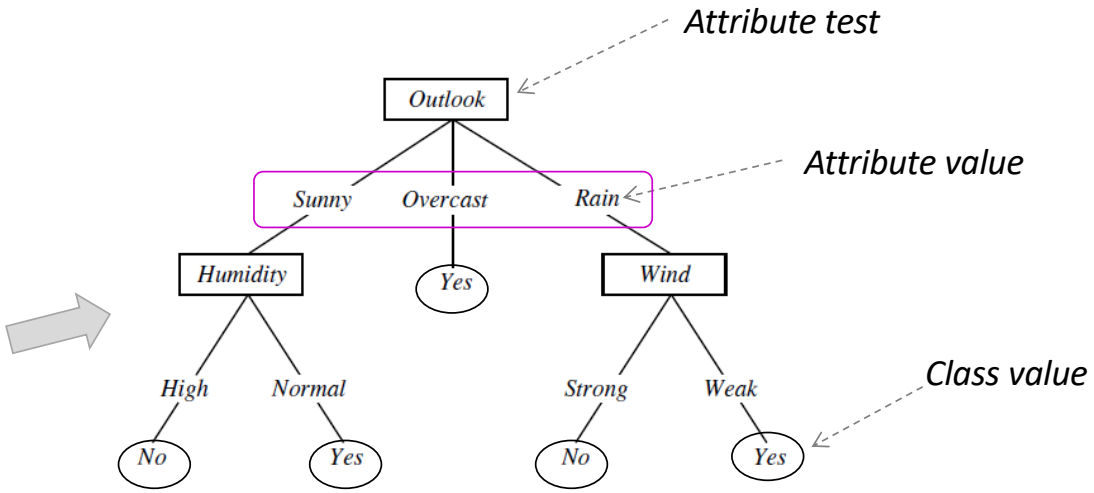

- Many algorithms have been proposed
  - ID3 (Quinlan 1986)
  - C4.5 (Quinlan 1993)
  - CART (Breiman et al 1984)
  - …

# Representation 1/2

- Representation
  - Each *internal node* specifies a test of some predictor attribute
  - Each *branch* descending from a node corresponds to one of the possible values for this attribute
  - Each *leaf node* assigns a class label
- Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance.

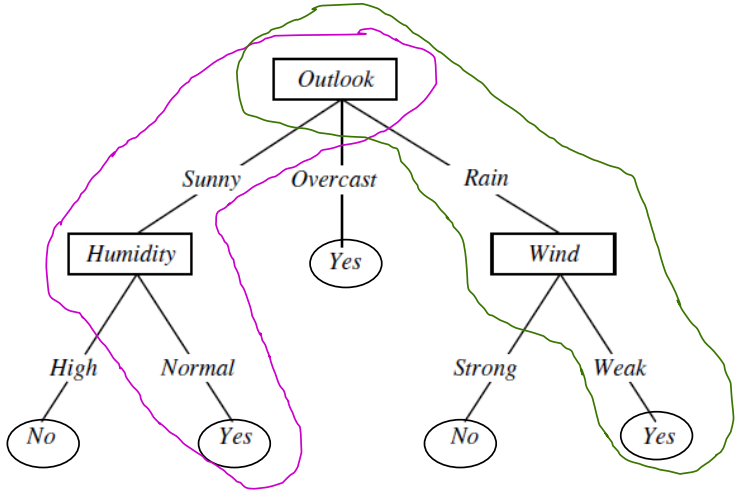
*Training set*

| 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 |



# Representation 2/2

- Decision trees represent a disjunction of conjunctions of constraints on the attribute values of the instances
  - Each path from the root to a leaf node, corresponds to a conjunction of attribute tests
  - The tree corresponds to a disjunction of these conjunctions
- We can “translate” each path into IF-THEN rules (human readable)



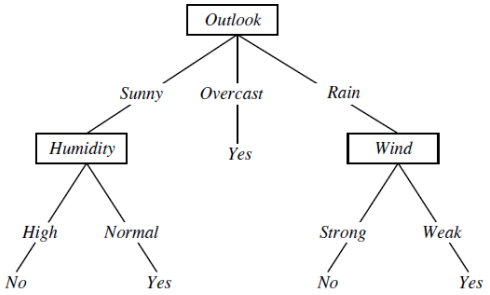

*IF ((Outlook = Sunny) ^ (Humidity = Normal)),*
*THEN (Play tennis=Yes)*

*IF ((Outlook = Rain) ^ (Wind = Weak)),*
*THEN (Play tennis=Yes)*

# The basic decision tree learning algorithm

Basic algorithm (ID3, Quinlan 1986)

- The tree is constructed in a top-down recursive divide-and-conquer manner
- At start, all the training examples are at the root node
- The question is "*which attribute should be tested at the root*?"
  - Attributes are evaluated using some statistical measure, which determines how well each attribute alone classifies the training examples.
  - The *best splitting attribute* is selected and used as the *test attribute* at the root.
- For each possible value of the test attribute, a descendant of the root node is created and the instances are mapped to the appropriate descendant node.
- The procedure is repeated for each descendant node, so instances are partitioned recursively.




*Training set*

| 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 |

# The basic decision tree learning algorithm

- Pseudocode

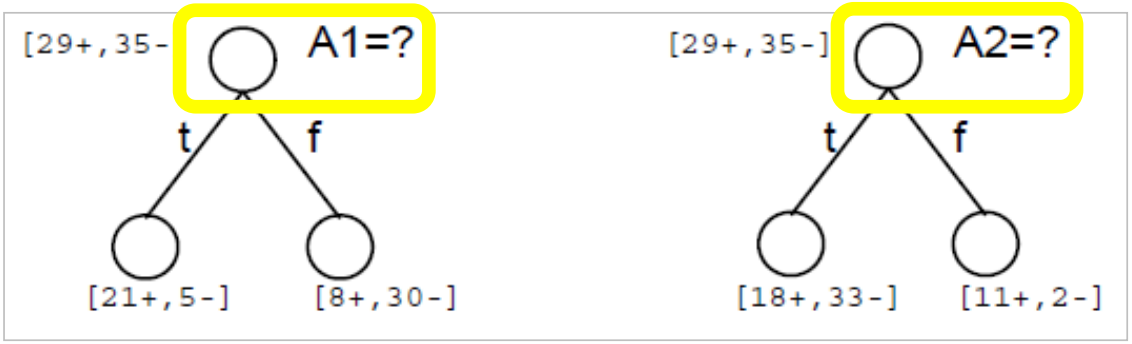
Main loop:

1. $A \leftarrow$ the "best" decision attribute for next *node*
2. Assign $A$ as decision attribute for *node*
3. For each value of $A$, create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

- When do we stop partitioning?
  - All samples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – *majority voting* for classifying the leaf

# Which attribute is the best?

- Which attribute to choose for splitting? A1 or A2?

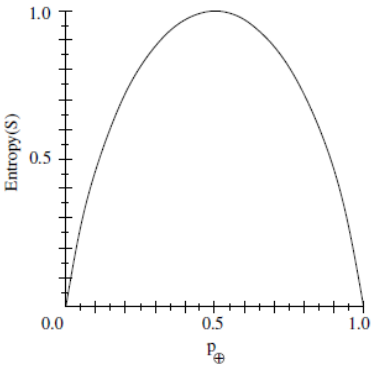



- The goal is to select the attribute that is most useful for classifying examples.
- By useful we mean that the resulting partitioning is as *pure* as possible
  - A partition is pure if all its instances belong to the same class.
- Different attribute selection measures
  - Information gain, gain ratio, gini index, …
  - all based on the degree of impurity of the parent (before splitting) vs the children nodes (after splitting)

# Entropy for measuring impurity of a set of instances

- Let S be a collection of positive and negative examples for a binary classification problem, C={+, -}.
  - $p_+$: the percentage of positive examples in S
  - $p_-$: the percentage of negative examples in S
- Entropy measures the impurity of S:

$$Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$




  - Entropy = 0, when all members belong to the same class
  - Entropy = 1, when there is an equal number of positive and negative examples
- Examples :
  - Let S: [9+,5-] $Entropy(S) = -\frac{9}{14}\log_2(\frac{9}{14}) - \frac{5}{14}\log_2(\frac{5}{14}) = 0.940$
  - Let S: [7+,7-] $Entropy(S) = -\frac{7}{14}\log_2(\frac{7}{14}) - \frac{7}{14}\log_2(\frac{7}{14}) = 1$
  - Let S: [14+,0-] $Entropy(S) = -\frac{14}{14}\log_2(\frac{14}{14}) - \frac{0}{14}\log_2(\frac{0}{14}) = 0$

in the general case (*k-classification problem)*

$$Entropy(S) = \sum_{i=1}^{k} - p_i \log_2(p_i)$$

# Attribute selection measure: Information gain

- Used in ID3
- It uses entropy, a measure of pureness of the data
- The information gain *Gain(S, A)* of an attribute *A* relative to a collection of examples *S* measures the entropy reduction in *S* due to splitting on *A*:

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

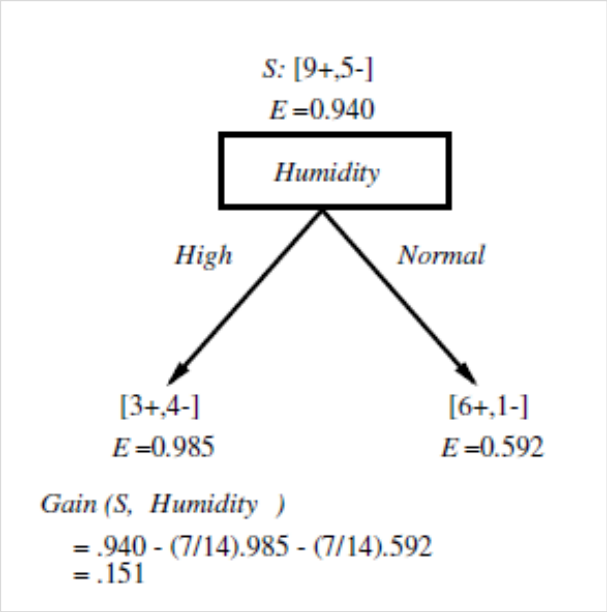
Before splitting

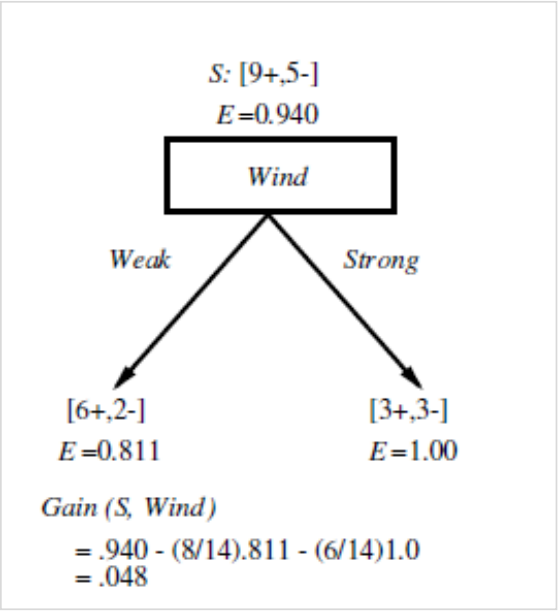
After splitting on A

- Gain measures the expected reduction in entropy due to splitting on *A*
- The attribute with the higher entropy reduction is chosen for splitting

# Information Gain example 1

- "Humidity" or "Wind"? Which attribute to choose for splitting?



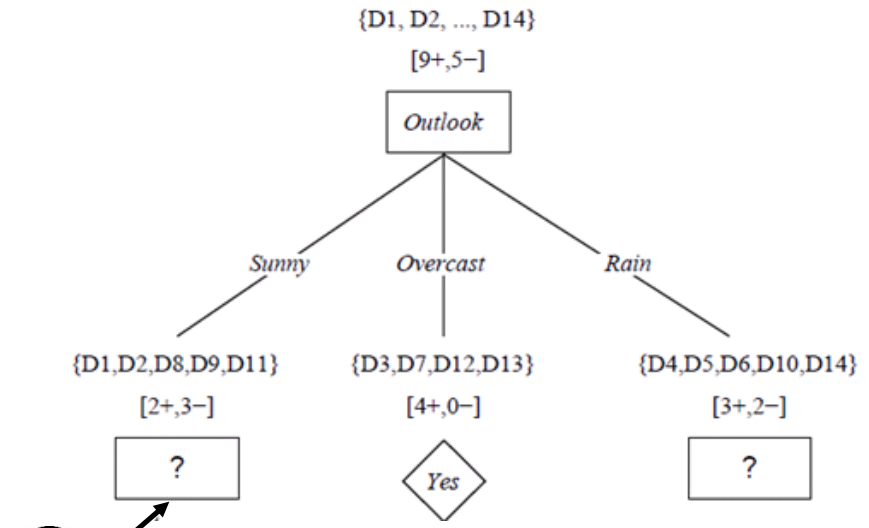




Which attribute is chosen?

# Information Gain example 2

Training *set*

| 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 |




Which attribute should we choose for splitting here?

$S_{sunny}$ = {D1,D2,D8,D9,D11}

*Gain* ($S_{sunny}$, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

*Gain* ($S_{sunny}$, *Temperature*) = .970 − (2/5) 0.0 − (2/5) 1.0 − (1/5) 0.0 = .570

*Gain* ($S_{sunny}$, *Wind*) = .970 − (2/5) 1.0 − (3/5) .918 = .019

Which attribute is chosen?