Introduction to Machine Learning

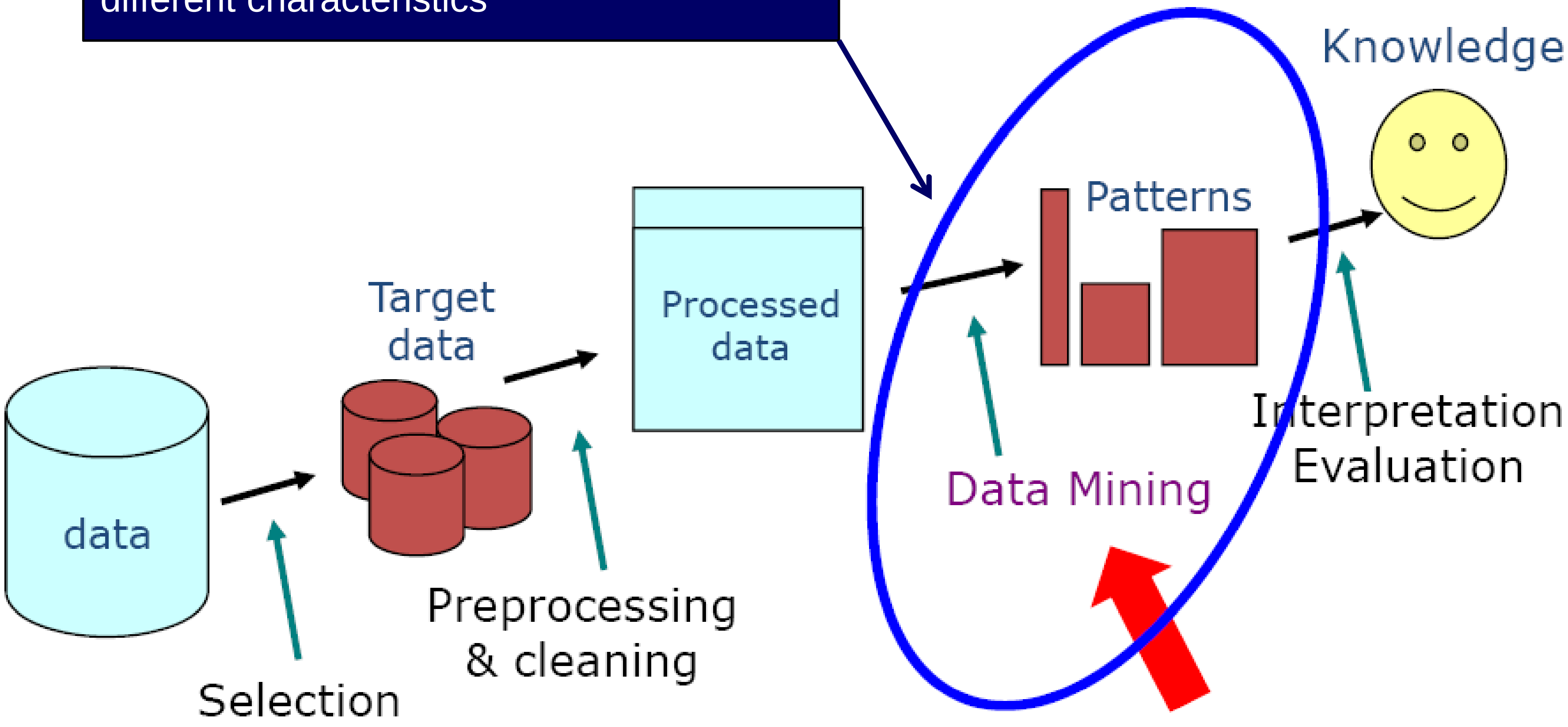
Lecture 5

Albert Orriols i Puig
aorriols@salle.url.edu

Artificial Intelligence – Machine Learning
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull

Recap of Lecture 4

The input consists of examples featured by different characteristics



Recap of Lecture 4

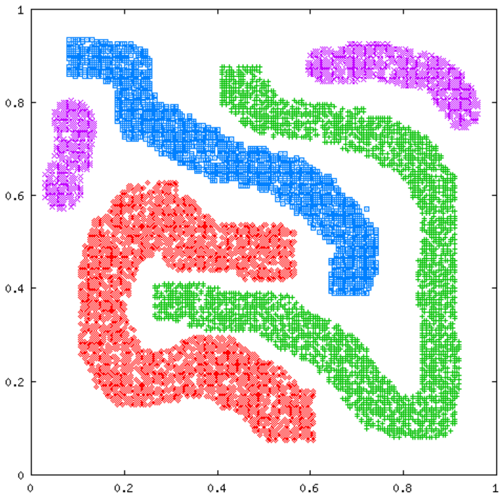
- **Different problems in machine learning**
  - **Classification:** Find the class to which a new instance belongs to
    - **E.g.: Find whether a new patient has cancer or not**
  - Numeric prediction: A variation of classification in which the output consists of numeric classes
    - **E.g.: Find the frequency of cancerous cell found**
  - Regression: Find a function that fits your examples
    - **E.g.: Find a function that controls your chain process**
  - Association: Find association among your problem attributes or variables
    - **E.g.: Find relations such as a patient with high-blood-pressure is more likely to have heart-attack disease**
  - Clustering: Process to cluster/group the instances into classes
    - **E.g.: Group clients whose purchases are similar**

Today's Agenda

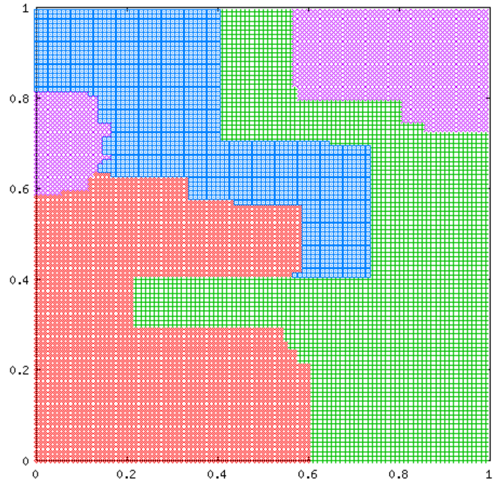
- Reviewing the Goal of Data Classification
- What's a Decision Tree
- How to build Decision Trees:
  - ID3
  - From ID3 to C4.5
- Run C4.5 on Weka

The Goal of Data Classification

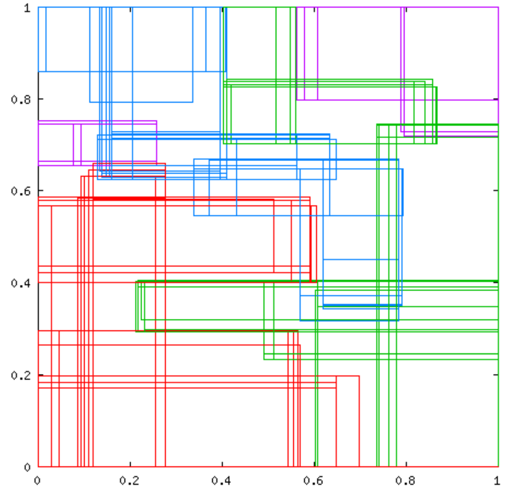
Data Set



Classification Model



How?



The *classification model* can be implemented in *several ways*:

- Rules
- **Decision trees**
- Mathematical formulae

The Goal of Data Classification

- **Data can have complex structures**
- **We will accept the following type of data:**
  - Data described by features which have a single measurement
  - Features can be
    - **Nominal: @attribute color {green, orange, red}**
    - **Continuous: @attribute length real [0,10]**
  - I can have unknown values
    - **I could have lost – or never have measured – the attribute of a particular example**

Classifying Plants

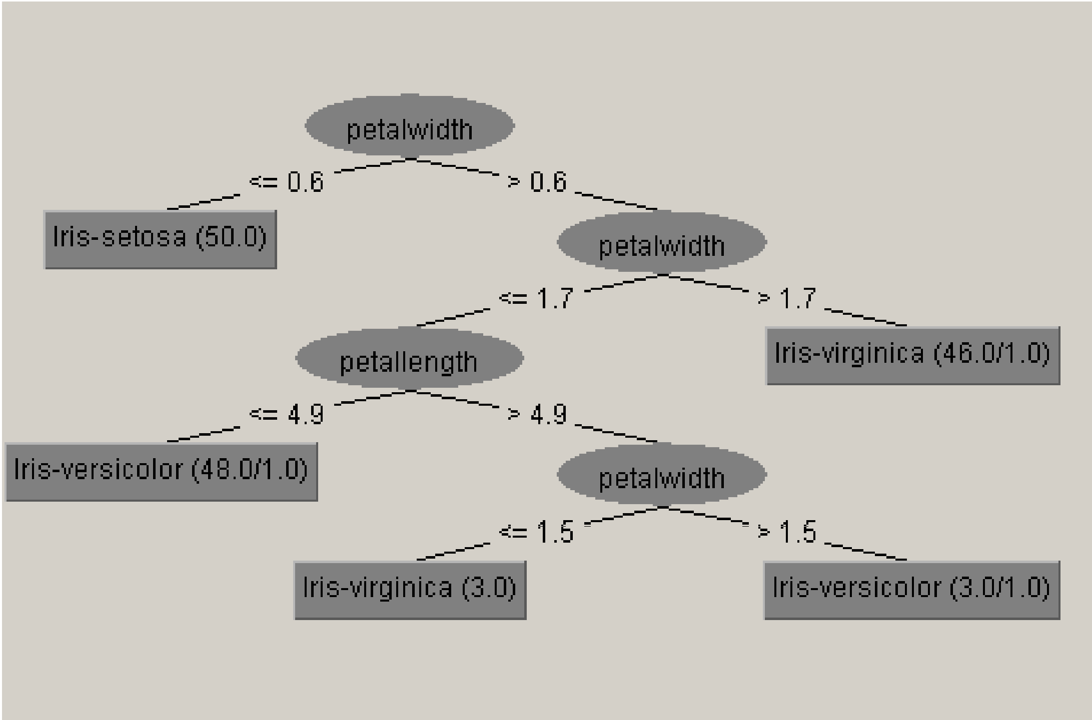
- **Let's classify different plants in three classes:**
  - Iris-setosa, iris-virginica, and iris-versicolor

```
@RELATION iris

@ATTRIBUTE sepallength  REAL
@ATTRIBUTE sepalwidth   REAL
@ATTRIBUTE petallength  REAL
@ATTRIBUTE petalwidth   REAL
@ATTRIBUTE class    {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
```

Weka format
Dataset publically available at UCI repository

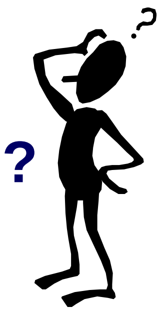
Classifying Plants

- Decision tree form this output



- **Internal node:** test of the value of a given attribute
- **Branch:** value of an attribute
- **Leaf:** predicted class

- How could I automatically generate these types of trees?



Types of DT Builders

- **So, where is the trick?**
  - Chose the attribute in each internal node and chose the best partition
- **Several algorithms able to generate decision trees:**
  - Naïve Bayes Tree
  - Random forests
  - CART (classification and regression tree)
  - ID3
  - C45
- **We are going to start from ID3 and finish with C4.5**
  - C4.5 is the most influential decision tree builder algorithm

ID3

- **Ross Quinlan started with**
  - ID3 (Quinlan, 1979)
  - C4.5 (Quinlan, 1993)

- **Some assumptions in the basic algorithm**
  - All attributes are nominal
  - We do not have unknown values

- **With these assumptions, Quinlan designed a heuristic approach to *infer* decision trees from labeled data**

Description of ID3

ID3(D, Target, Atts) **(Mitchell,1997)**
returns: **a decision tree that correctly classifies the given examples**

variables
D: **Training set of examples**
Target: **Attribute whose value is to be predicted by the tree**
Atts: **List of other attributes that may be tested by the learned decision tree**

create a *Root* node for the tree
if D are all positive then *Root* ← +
else if *D* are all negative then *Root* ← –
else if *Atts* = ∅ then *Root* ← most common value of target in D
else

> **A← the best decision attribute from Atts**
> **root ← A**
> **for each possible value v_i of A**
>> **add a new tree branch with A=v_i**
>> ***Dvi* ← subset of *D* that have value v_i for A**
>> **if *Dvi* = ∅ add then leaf ← most common value of *Target* in *D***
>> **else add the subtree ID3(*Dvi*, *Target, Atts-{A}*)**

Description of ID3

ID3(D, Target, Atts) (Mitchell,1997)

returns: a decision tree that correctly classifies the given examples

$$D = \{d_1, d_2, ..., d_L\}$$

$$Atts = \{a_1, a_2, ..., a_k\}$$

variables

D: Training set of examples

Target: Attribute whose value is to be predicted by the tree

Atts: List of other attributes that may be tested by the learned decision tree

create a *Root* node for the tree

if D are all positive then *Root* ← +

else if *D* are all negative then *Root* ← –

else if *Attrs* = ∅ then *Root* ← most common value of target in D

else

  A← the best decision attribute from Atts

  root ← A

  for each possible value v_i of A

    add a new tree branch with A=v_i

    Dvi ← subset of *D* that have value v_i for A

    if *Dvi* = ∅ add then leaf ← most common value of *Target* in *D*

    else add the subtree ID3(*Dvi*, *Target, Atts-{A}*)

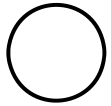
Description of ID3

ID3(D, Target, Atts) **(Mitchell,1997)**

returns: a decision tree that correctly classifies the given examples

$D = \{d_1, d_2, \ldots, d_L\}$

$Atts = \{a_1, a_2, \ldots, a_k\}$



variables

D: Training set of examples

Target: Attribute whose value is to be predicted by the tree

Atts: List of other attributes that may be tested by the learned decision tree

create a *Root* node for the tree

if D are all positive **then** ***Root*** ← +

else if ***D*** are all negative **then** ***Root*** ← –

else if ***Attrs*** = ∅ **then** ***Root*** ← most common value of target in D

else

  A← the best decision attribute from Atts

  root ← A

  for each possible value v_i of A

    add a new tree branch with A=v_i

    Dvi ← subset of ***D*** that have value v_i for A

    if ***Dvi*** = ∅ add **then** leaf ← most common value of ***Target*** in ***D***

    else add the subtree ID3(***Dvi***, ***Target, Atts-{A}***)

Description of ID3

ID3(D, Target, Atts) (Mitchell,1997)
returns: a decision tree that correctly classifies the given examples

$$D = \{d_1, d_2, \ldots, d_L\}$$
$$Atts = \{a_1, a_2, \ldots, a_k\}$$

a_i

variables
D: Training set of examples
Target: Attribute whose value is to be predicted by the tree
Atts: List of other attributes that may be tested by the learned decision tree

create a *Root* node for the tree
if D are all positive **then** *Root* ← +
else if *D* are all negative **then** *Root* ← –
else if *Attrs* = ∅ **then** *Root* ← most common value of target in D
else
    A← the best decision attribute from Atts
    root ← A
    for each possible value v_i of A
        add a new tree branch with A=v_i
        Dvi ← subset of *D* that have value v_i for A
        if *Dvi* = ∅ add **then** leaf ← most common value of *Target* in *D*

        else add the subtree ID3(*Dvi*, *Target, Atts-{A}*)

Description of ID3

ID3(D, Target, Atts) **(Mitchell,1997)**
returns: a decision tree that correctly classifies the given examples

$$D = \{d_1, d_2, \ldots, d_L\}$$
$$Atts = \{a_1, a_2, \ldots, a_k\}$$

a_i

$a_i = v_1$ $a_i = v_2$ … $a_i = v_n$

variables
D: Training set of examples
Target: Attribute whose value is to be predicted by the tree
Atts: List of other attributes that may be tested by the learned decision tree

create a *Root* node for the tree
if D are all positive **then** *Root* ← +
else if *D* are all negative **then** *Root* ← –
else if *Attrs* = ∅ **then** *Root* ← most common value of target in D
else
  A← the best decision attribute from Atts
  root ← A
  for each possible value v_i of A
    add a new tree branch with A=v_i
    Dvi ← subset of *D* that have value v_i for A
    if *Dvi* = ∅ add **then** leaf ← most common value of *Target* in *D*

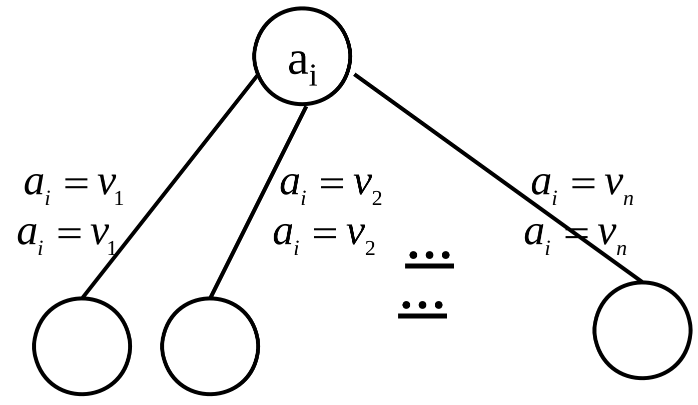
    else add the subtree ID3(*Dvi*, *Target, Atts-{A}*)

Description of ID3

ID3(D, Target, Atts) **(Mitchell,1997)**

returns: **a decision tree that correctly classifies the given examples**

$$D=\{d_1,d_2,\ldots,d_L\}$$
$$Atts=\{a_1,a_2,\ldots,a_k\}$$



$$D=\{d_1^{'},d_2^{'},\ldots,d_L^{'}\}$$
$$Atts=\{a_1,..a_{i-1},a_{i+1},..,a_k\}$$

$$D=\{d_1^{'''},d_2^{'''},\ldots,d_L^{'''}\}$$
$$Atts=\{a_1,..a_{i-1},a_{i+1},..,a_k\}$$

variables

D: **Training set of examples**

Target: **Attribute whose value is to be predicted by the tree**

Atts: **List of other attributes that may be tested by the learned decision tree**

create a *Root* node for the tree

if D are all positive then *Root* ← +

else if *D* are all negative then *Root* ← –

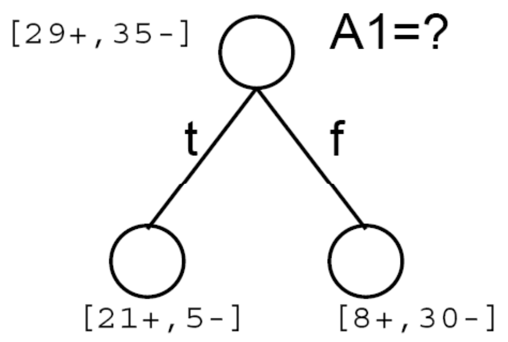
else if *Attrs* = ∅ then *Root* ← most common value of target in D

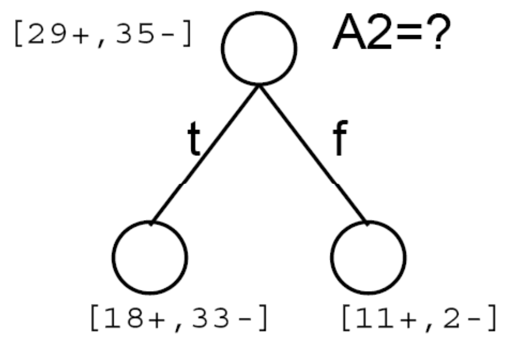
else

- **A← the best decision attribute from Atts**
- **root ← A**
- **for each possible value v_i of A**
  - **add a new tree branch with A=v_i**
  - ***Dvi* ← subset of *D* that have value v_i for A**
  - **if *Dvi* = ∅ add then leaf ← most common value of *Target* in *D***
  - **else add the subtree ID3(*Dvi*, *Target*, *Atts-{A}*)**

Which Attribute Should I Select First?

- **Which is the best choice?**
  - We have 29 positive examples and 35 negative ones
  - Should I use attribute 1 or attribute 2 in this iteration of the node?





Let’s Rely on Information Theory

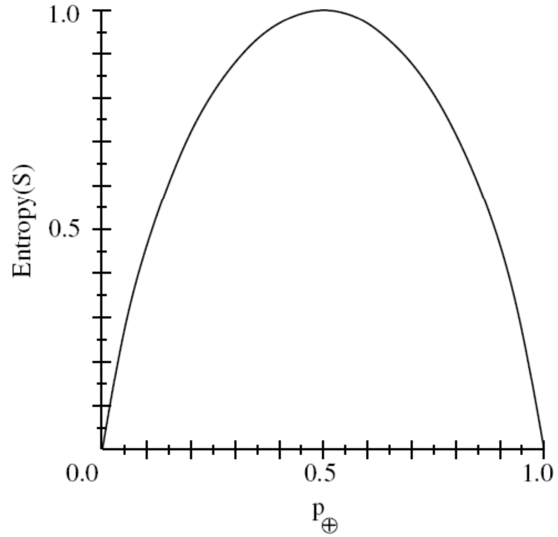
- **Use the concept of Entropy**
  - Characterizes the impurity of an arbitrary collection of examples
  - Given S:

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

where p+ and p- are the proportion of positive/negative examples in S.

  - Extension to c classes:

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



- Examples:
  - p+=0 $\rightarrow$ entropy=0
  - p+=1 $\rightarrow$ entropy=0
  - p+=0.5 p-=0.5 $\rightarrow$ entropy=1 (maximum)
  - P+=9 p-=5 $\rightarrow$ entropy=-(9/14)$\log_2$(9/14)-(5/14)$\log_2$(5/14)=0.940

Entropy

- **What does this measure mean?**
  - Entropy is the minimum number of bits needed to encode the classification of a member of S randomly drawn.
    - **p+=1, the receiver knows the class, no message sent, Entropy=0.**
    - **p+=0.5, 1 bit needed.**
  - Optimal length code assigns $-\log_2 p$ to message having probability p
  - The idea behind is to assign shorter codes to the more probable messages and longer codes to less likely examples.
  - Thus, the expected number of bits to encode + or – of random member of S is:

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

Information Gain

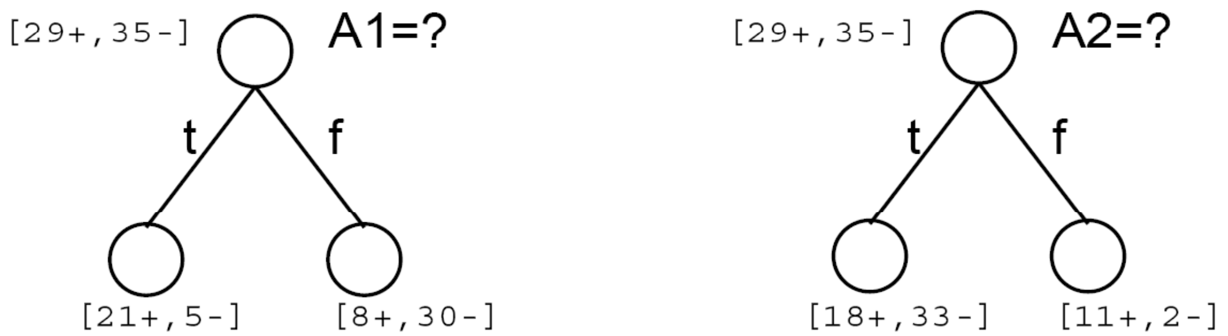
- Measures the expected reduction in entropy caused by partitioning the examples according to the given attribute
- Gain(S,A): the number of bits saved when encoding the target value of an arbitrary member of S, knowing the value of attribute A.
- Expected reduction in entropy caused by knowing the value of A.

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

where values(A) is the set of all possible values for A, and Sv is the subset of S for which attribute A has value v

Remember the Example?

- **Which is the best choice?**
  - We have 29 positive examples and 35 negative ones
  - Should I use attribute 1 or attribute 2 in this iteration of the node?



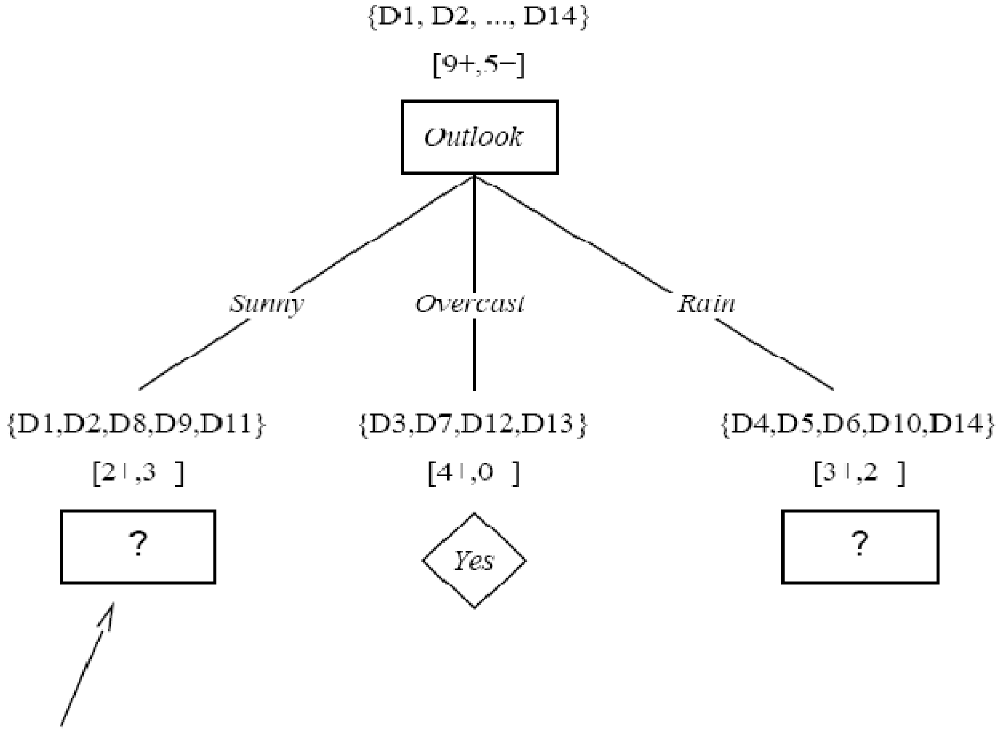
Gain(A1)=0.993 - 26/64 0.70 - 36/64 0.74=0.292
Gain(A2)=0.993 - 51/64 0.93 -13/64 0.61 = 0.128

Yet Another Example

- The textbook example

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

Yet Another Example



Which attribute should be tested here?

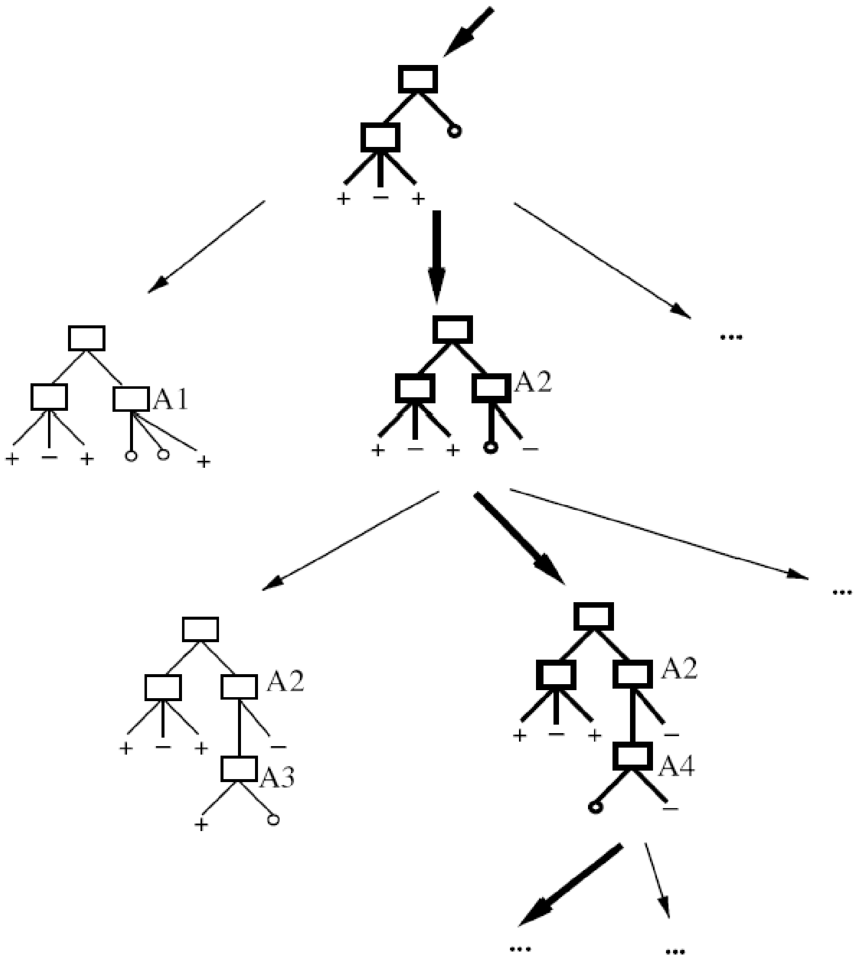
S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

Hypothesis Search Space



To Sum Up

- **ID3 is a strong system that**
  - Uses hill-climbing search based on the information gain measure to search through the space of decision trees
  - Outputs a single hypothesis.
  - Never backtracks. It converges to locally optimal solutions.
  - Uses all training examples at each step, contrary to methods that make decisions incrementally.
  - Uses statistical properties of all examples: the search is less sensitive to errors in individual training examples.
  - Can handle noisy data by modifying its termination criterion to accept hypotheses that imperfectly fit the data.

Next Class

- **From ID3 to C4.5. C4.5 extends ID3 and enables the system to:**
  - Be more robust in the presence of noise. Avoiding overfitting
  - Deal with continuous attributes
  - Deal with missing data
  - Convert trees to rules

Introduction to Machine Learning

Lecture 5

Albert Orriols i Puig
aorriols@salle.url.edu

Artificial Intelligence – Machine Learning
Enginyeria i Arquitectura La Salle
Universitat Ramon Llull