Unit 3: Inductive machine learning

M. A. Gutiérrez Naranjo
F. J. Martín Mateos
A. Riscos Núñez
J. L. Ruiz Reina

Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla

Contents

Section 1

Introduction

Learning

- Definitions of *learning*:
  - Learning is any process by which a system *improves performance* from experience (*H. Simon*)
  - Modify the representation of perceived world (*R. Michalski*)
  - Making useful changes in our minds (*M. Minsky*)
- Machine learning: build programs that automatically improve with experience
- Tasks examples:
  - Build knowledge bases from experience
  - Classification and diagnosis
  - Data mining, uncover unknownd patterns in big databases
  - Resolutions of problems, planning and acting

Types of problems in machine learning and paradigms

- Types of learning
  - Supervised learning
  - Unsupervised learning
  - Reinforcement learning
- Paradigms
  - Rote learning
  - Classification (*Clustering*)
  - Inductive learning
  - Learning by analogy
  - Discover
  - Genetic algorithms, neural networks

Example of learning

- Training set
  - Examples: days appropriate (or not) for playing tennis
  - Data represented as a list of pairs attribute-value

Ej.	Outlook	Temperature	Humidity	Wind	PlayTennis
D_1	Sunny	High	High	Weak	-
D_2	Sunny	High	High	Strong	-
D_3	Overcast	High	High	Weak	+
D_4	Rainy	Mild	High	Weak	+
...					

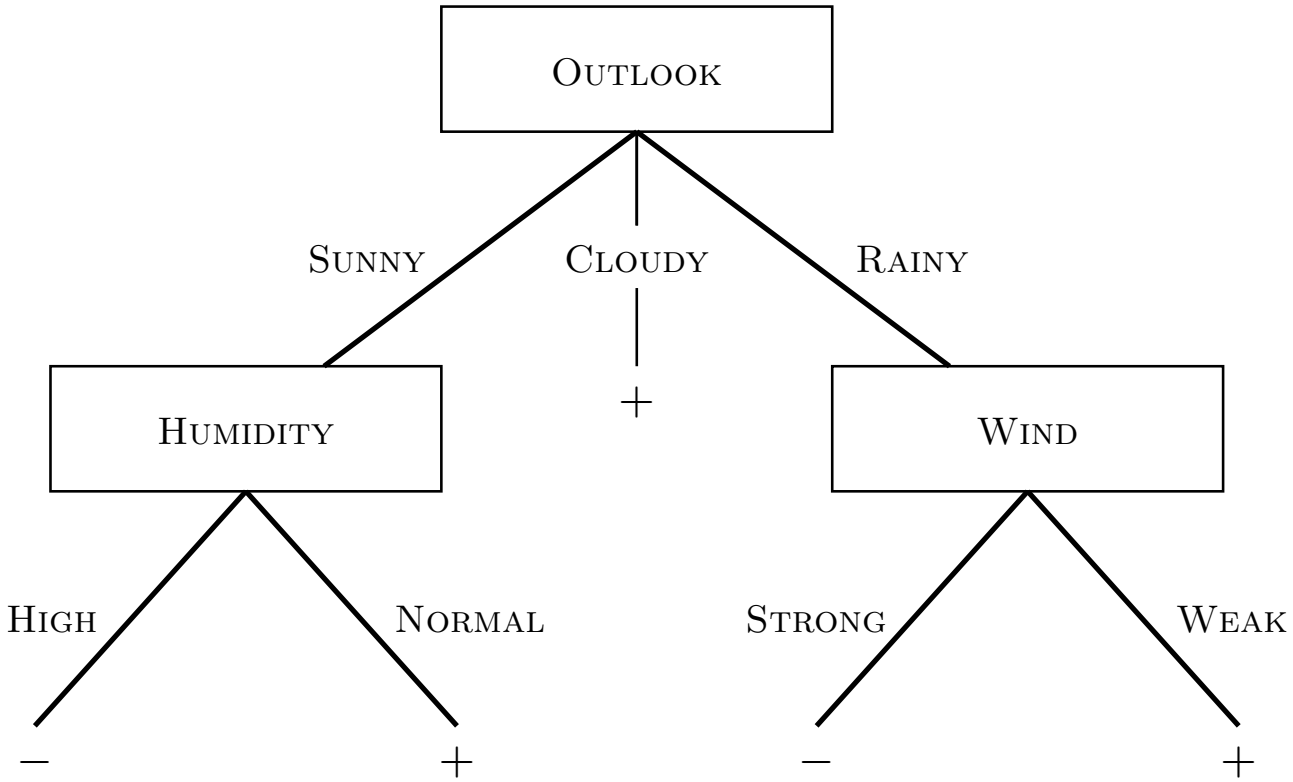
- Goal: given the *training set*, *learn* the concept "Day apropriate for playing tennis".
  - This is *supervised learning*
- Problem: ¿How to express what has been learned?
  - In this unit, we will see algorithms for learning *decision trees*, *rules*, *neural networks*,...

Section 2

Learning decision trees

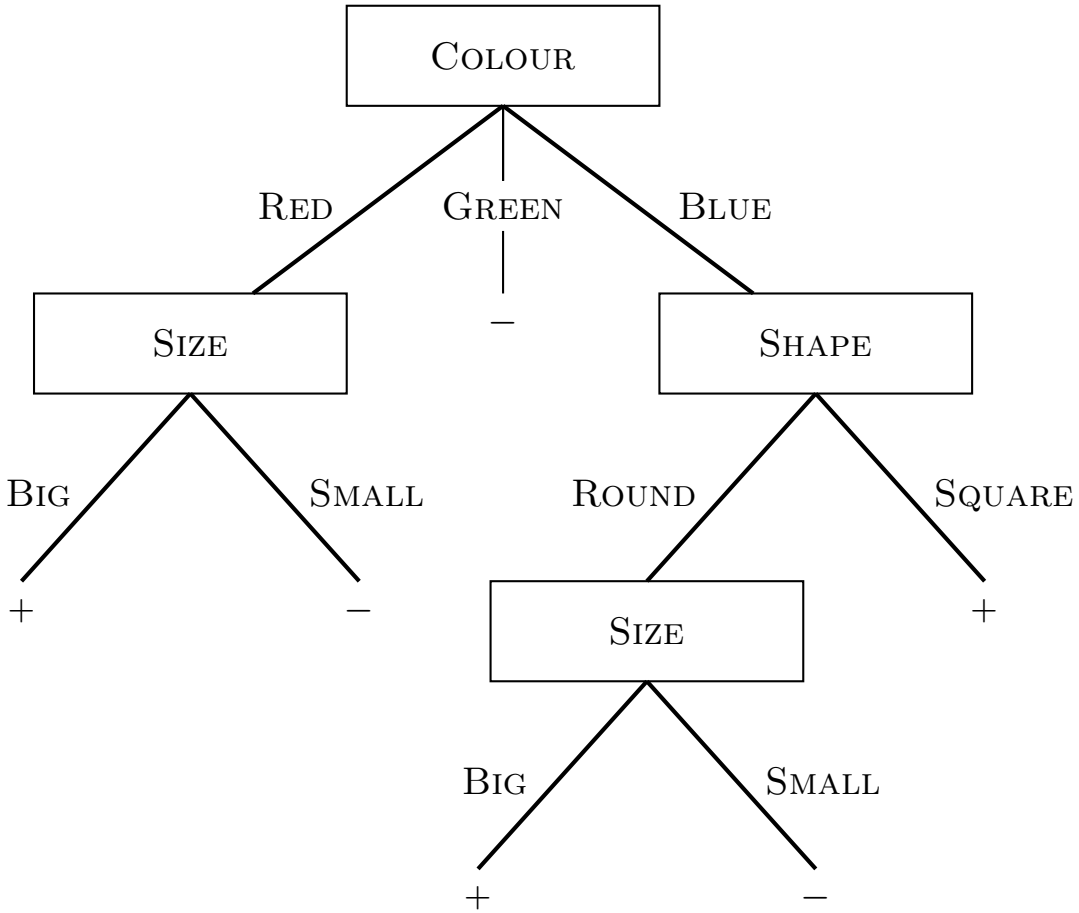
Decision trees

- Decision trees examples



Decisioon trees

- Decision trees examples



Decision trees

- Decison trees
  - Internal nodes: attributes
  - Outgoing edges labeled with values of the node
  - Leaves: classification value (usually $+$ or $-$, although it could be any set of classes, not necessarily binary)
  - It representes a classification function
- Disjunction of propositional rules

$$
\begin{aligned}
&(\textsc{Outlook}{=}\textsc{Sunny} \wedge \textsc{Humidity}{=}\textsc{High} \rightarrow \textsc{PlayTennis}{=} -)\\
\vee\ &(\textsc{Outlook}{=}\textsc{Sunny} \wedge \textsc{Humidity}{=}\textsc{Normal} \rightarrow \textsc{PlayTennis}{=} +)\\
\vee\ &(\textsc{Outlook}{=}\textsc{Overcast} \rightarrow \textsc{PlayTennis}{=} +)\\
\vee\ &(\textsc{Outlook}{=}\textsc{Rainy} \wedge \textsc{Wind}{=}\textsc{Strong} \rightarrow \textsc{PlayTennis}{=} -)\\
\vee\ &(\textsc{Outlook}{=}\textsc{Rainy} \wedge \textsc{Wind}{=}\textsc{Weak} \rightarrow \textsc{PlayTennis}{=} +)
\end{aligned}
$$

- Powerful enough to represent any subset of positive instances

Learning decsion trees

- Goal: Learn a decision tree, consistent with the examples in the training set, and then use it to classify new examples
- Training set example:

Ej.	Outlook	Temperature	Humidity	Wind	PlayTennis
D_1	Sunny	High	High	Weak	-
D_2	Sunny	High	High	Strong	-
D_3	Overcast	High	High	Weak	+
D_4	Rainy	Mild	High	Weak	+
...					

ID3 algorithm

```
ID3(Examples, Class, Attributes)
1. If all the Examples are positive, return a leaf node labeld with +
2. If all the Examples are negative, return a leaf node labeled with -
3. If Attributes is empty, return a node labeled with the most
   frequent class in Examples
4. Otherwise:
4.1. Let A be the BEST atribute of Attributes, as for the classifying
     Examples
4.2. Create Tree, with a node labeled with A
4.3. For every possible value v of A:
   * Add an outgoing edge to Tree, labeled with v
   * Let Examples(v) the subset of Examples with value v in atribute A
   * If Examples(v) is empty:
     - Then put a leaf labeled with the most frequent class
       in Examples
     - Else, extend the outgoing edge with the subtree
       ID3(Examples(v), Class, Attributes-{A}).
4.4  Return Tree
```

Which is the BEST attribute for an intenal node?

- Entropy of a set of examples D (w.r.t. a classification):
$$Ent(D) = -\frac{|P|}{|D|} \cdot log_2 \frac{|P|}{|D|} - \frac{|N|}{|D|} \cdot log_2 \frac{|N|}{|D|}$$
where P and N are, respectively, the subsets of positive and negative examples in D
  - Notation: $Ent([p^+, n^-])$, where $p = |P|$ and $n = |N|$
- Intuition:
  - Quantify the "heterogeneity" of a classification
  - Information Theory: average amount of information (in bits) needed to codify the class of an example
- Examples:
  - $Ent([9^+, 5^-]) = -\frac{9}{14} \cdot log_2 \frac{9}{14} - \frac{5}{14} \cdot log_2 \frac{5}{14} = 0{,}94$
  - $Ent([k^+, k^-]) = 1$ (no homogeneity at all)
  - $Ent([p^+, 0^-]) = Ent([0^+, n^-]) = 0$ (total homogeneity)

Information gain

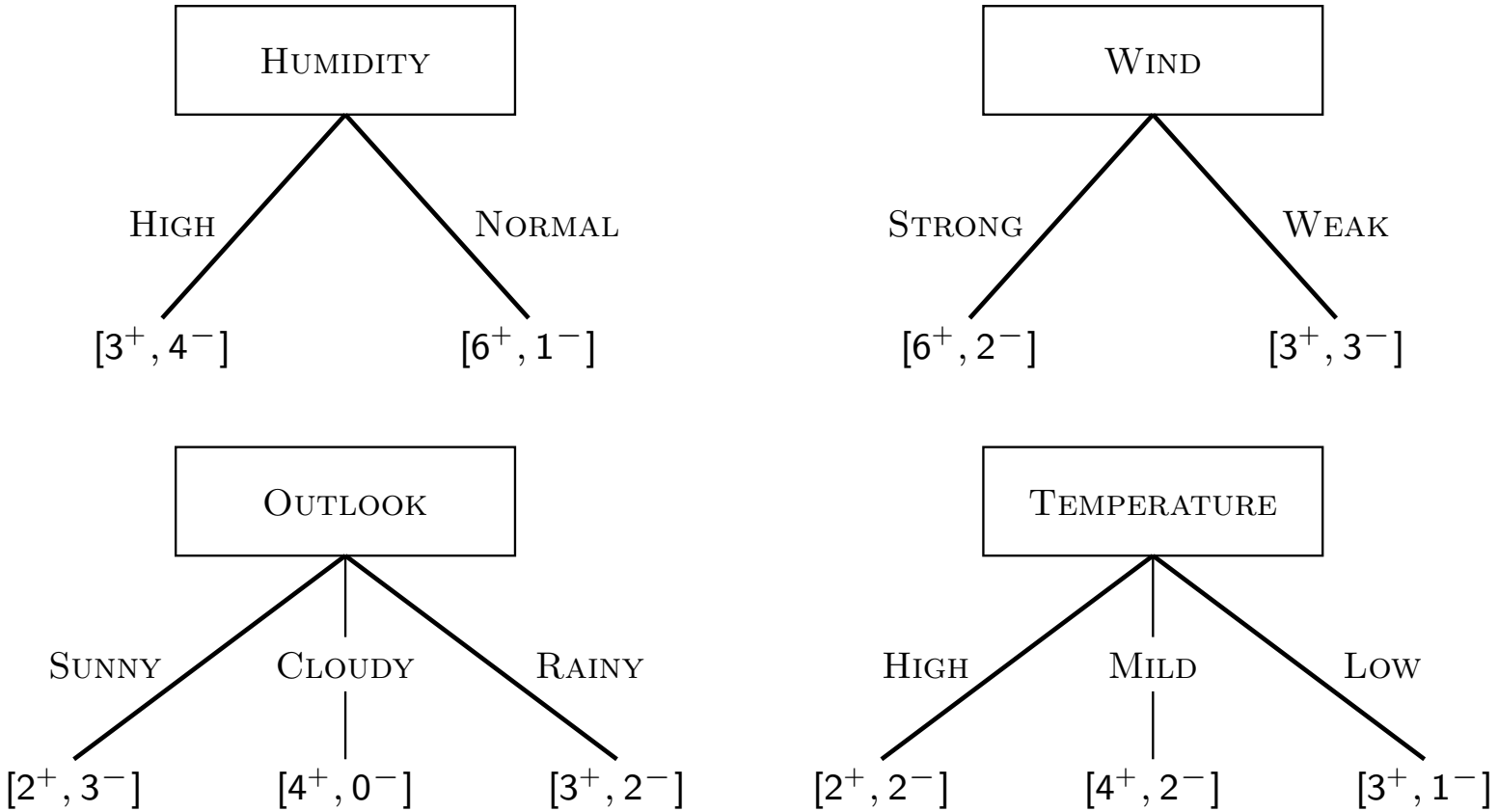
- Nodes with less entropy are better (small trees)
- *Expected* entropy after using attribute A in the tree:
$$\sum_{v \in Valores(A)} \frac{|D_v|}{|D|} \cdot Ent(D_v)$$
where D_v es the subset of D with value v in attribute A
- *Expected* information gain después after using attribute A:
$$Gain(D, A) = Ent(D) - \sum_{v \in Valores(A)} \frac{|D_v|}{|D|} \cdot Ent(D_v)$$
- In ID3, we use *information gain* as the criteria for selecting the best attribute in every internal node (takinh into account the examples at that node)

Algorithm ID3 (example 1)

- Training set:

Ej.	Outlook	Temperature	Humidity	Wind	PlayTennis
D_1	Sunny	High	High	Weak	-
D_2	Sunny	High	High	Strong	-
D_3	Overcast	High	High	Weak	+
D_4	Rainy	Mild	High	Weak	+
D_5	Rainy	Low	Normal	Weak	+
D_6	Rainy	Low	Normal	Strong	-
D_7	Overcast	Low	Normal	Strong	+
D_8	Sunny	Mild	High	Weak	-
D_9	Sunny	Low	Normal	Weak	+
D_{10}	Rainy	Mild	Normal	Weak	+
D_{11}	Sunny	Mild	Normal	Strong	+
D_{12}	Overcast	Mild	High	Strong	+
D_{13}	Overcast	High	Normal	Weak	+
D_{14}	Rainy	Mild	High	Strong	-

Algorithm ID3 (example 1)

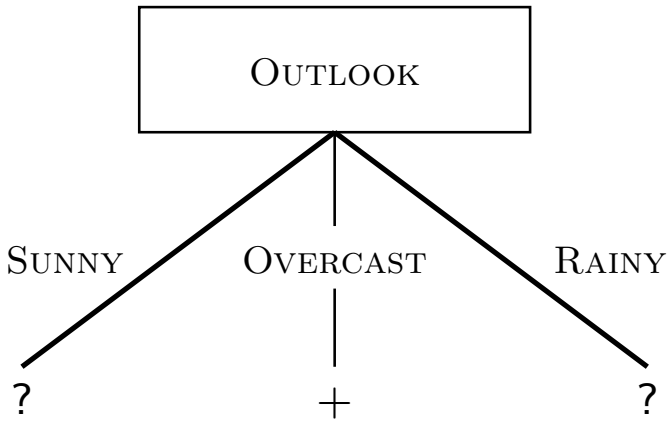


Algorithm ID3 (example 1)

- Initial entropy: $Ent([9^+, 5^-]) = 0{,}94$
- Selecting the attribute for the root node:
  - $Gain(D, \text{HUMIDITY}) =$
    $0{,}94 - \frac{7}{14} \cdot Ent([3^+, 4^-]) - \frac{7}{14} \cdot Ent([6^+, 1^-]) = 0{,}151$
  - $Gain(D, \text{WIND}) =$
    $0{,}94 - \frac{8}{14} \cdot Ent([6^+, 2^-]) - \frac{6}{14} \cdot Ent([3^+, 3^-]) = 0{,}048$
  - $Gain(D, \text{OUTLOOK}) =$
    $0{,}94 - \frac{5}{14} \cdot Ent([2^+, 3^-]) - \frac{4}{14} \cdot Ent([4^+, 0^-])$
    $- \frac{5}{14} \cdot Ent([3^+, 2^-]) = 0{,}246$ (best attribute)
  - $Gain(D, \text{TEMPERATURE}) =$
    $0{,}94 - \frac{4}{14} \cdot Ent([2^+, 2^-]) - \frac{6}{14} \cdot Ent([4^+, 2^-])$
    $- \frac{4}{14} \cdot Ent([3^+, 1^-]) = 0{,}02$
- The selected attribute is OUTLOOK

Algorithm ID3 (example 1)

- Partial tree:



Algorithm ID3 (example 1)

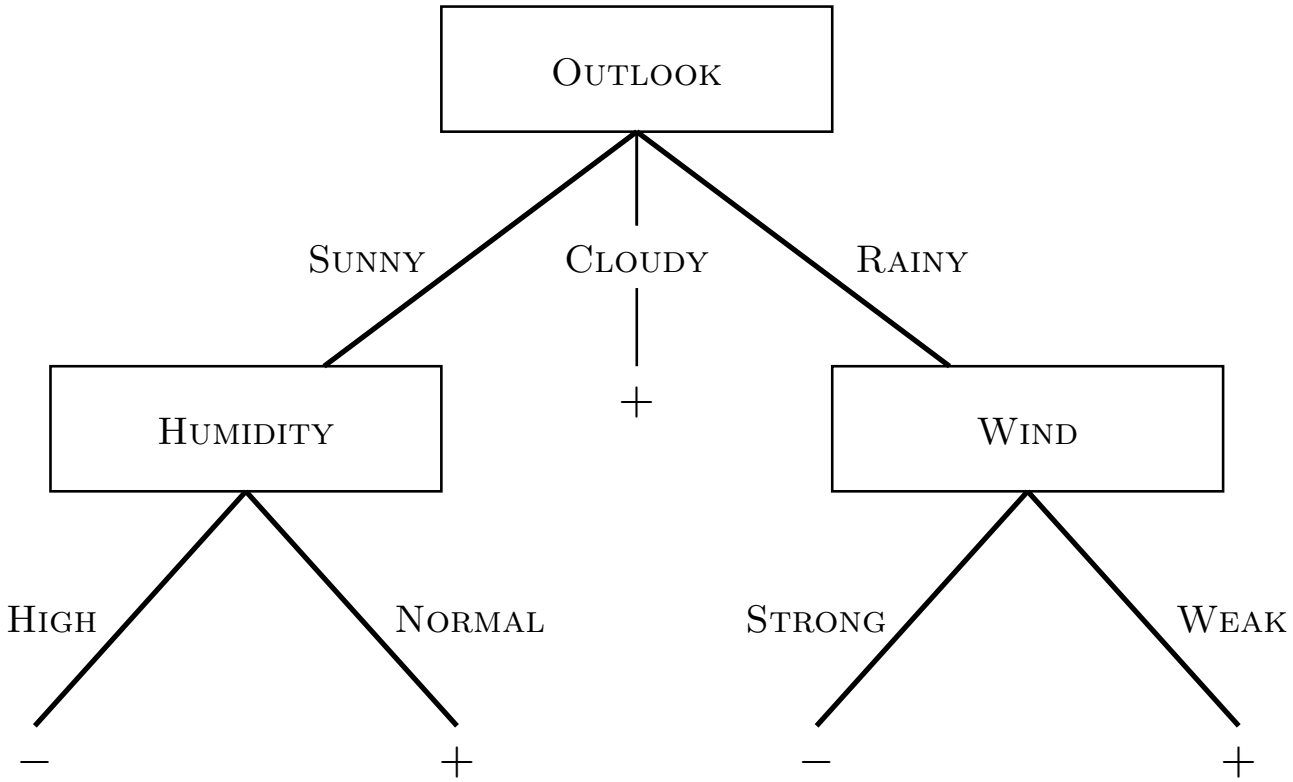
- Selecting the attribute for node $\textsc{Outlook}{=}\textsc{Sunny}$
- $D_{\textsc{Sunny}} = \{D_1, D_2, D_8, D_9, D_{11}\}$ with entropy $Ent([2^+, 3^-]) = 0{,}971$
  - $Gain(D_{\textsc{Sunny}}, \textsc{Humidity}) = 0{,}971 - \frac{3}{5} \cdot 0 - \frac{2}{5} \cdot 0 = 0{,}971$ (best attribute)
  - $Gain(D_{\textsc{Sunny}}, \textsc{Temperature}) = 0{,}971 - \frac{2}{5} \cdot 0 - \frac{2}{5} \cdot 1 - \frac{1}{5} \cdot 0 = 0{,}570$
  - $Gain(D_{\textsc{Sunny}}, \textsc{Wind}) = 0{,}971 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0{,}918 = 0{,}019$
- The selected attribute is $\textsc{Humidity}$

Algorithm ID3 (example 1)

- Selecting the attribute for node $\textsc{Outlook}{=}\textsc{Rainy}$:
- $D_{\textsc{Rainy}} = \{D_4, D_5, D_6, D_{10}, D_{14}\}$ with entropy $Ent([3^+, 2^-]) = 0{,}971$
  - $Gain(D_{\textsc{Rainy}}, \textsc{Humidity}) = 0{,}971 - \frac{2}{5} \cdot 1 - \frac{3}{5} \cdot 0{,}918 = 0{,}020$
  - $Gain(D_{\textsc{Rainy}}, \textsc{Temperatura}) = 0{,}971 - \frac{3}{5} \cdot 0{,}918 - \frac{2}{5} \cdot 1 = 0{,}020$
  - $Gain(D_{\textsc{Rainy}}, \textsc{Wind}) = 0{,}971 - \frac{3}{5} \cdot 0 - \frac{2}{5} \cdot 0 = 0{,}971$ (best attribute)
- The selected attribute is $\textsc{Wind}$

Algoritmo ID3 (ejemplo 1)

- The tree finally learned:



Algoritmo ID3 (ejemplo 2)

- Conjunto de entrenamiento:

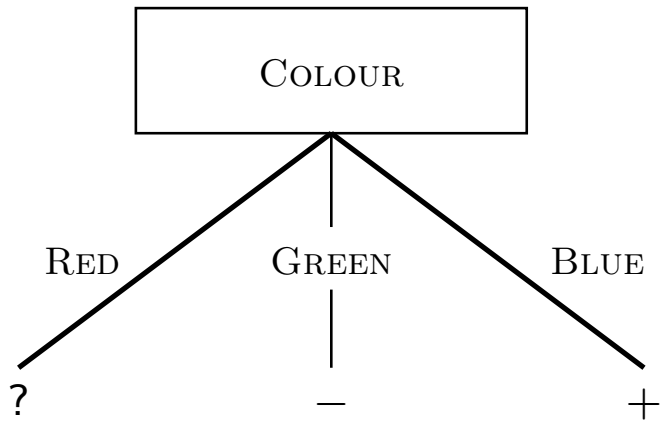
Ej.	Colour	Shape	Size	Class
O_1	Red	Square	Big	+
O_2	Blue	Square	Big	+
O_3	Red	Round	Small	-
O_4	Green	Square	Small	-
O_5	Red	Round	Big	+
O_6	Green	Square	Big	-

Algorithm ID3 (example 2)

- Initial entropy, $Ent([3^+, 3^-]) = 1$
- Selecting the attribute for the root node:
  - $Gain(D, \textsc{Colour}) = 1 - \frac{3}{6} \cdot Ent([2^+, 1^-]) - \frac{1}{6} \cdot Ent([1^+, 0^-]) - \frac{2}{6} \cdot Ent([0^+, 2^-]) = 0{,}543$
  - $Gain(D, \textsc{Shape}) = 1 - \frac{4}{6} \cdot Ent([2^+, 2^-]) - \frac{2}{6} \cdot Ent([1^+, 1^-]) = 0$
  - $Gain(D, \textsc{Size}) = 1 - \frac{4}{6} \cdot Ent([3^+, 1^-]) - \frac{2}{6} \cdot Ent([0^+, 2^-]) = 0{,}459$
- The selected attribute is $\textsc{Colour}$

Algorithm ID3 (example 2)

- Partial tree:

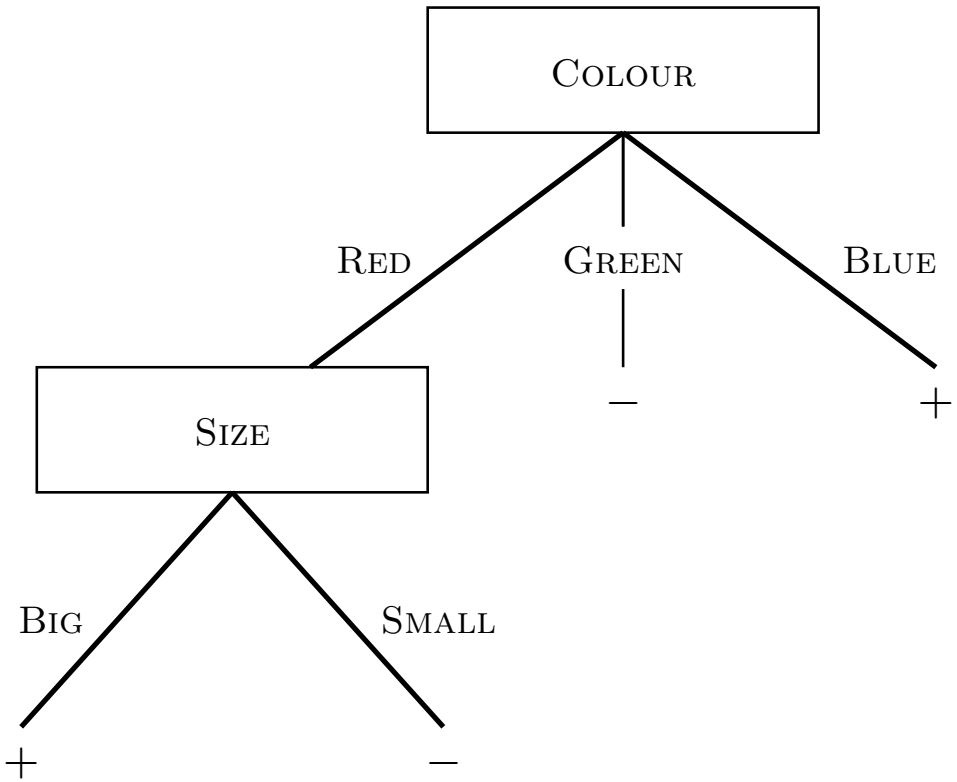


Algorithm ID3 (example 2)

- Selecting attribute for the node $\textsc{Colour}{=}\textsc{Red}$:
- $D_{\textsc{Red}} = \{O_1, O_3, O_5\}$ with entropy $Ent([2^+, 1^-]) = 0{,}914$
  - $Gain(D_{\textsc{Red}}, \textsc{Shape}) =$
    $0{,}914 - \frac{1}{3} \cdot Ent([1^+, 0^-]) - \frac{2}{3} \cdot Ent([1^+, 1^-]) = 0{,}247$
  - $Gain(D_{\textsc{Red}}, \textsc{Size}) =$
    $0{,}914 - \frac{2}{3} \cdot Ent([2^+, 0^-]) - \frac{1}{3} \cdot Ent([0^+, 1^-]) = 0{,}914$
- The selected attribute is $\textsc{Size}$

Algorithm ID3 (example 2)

- Final tree learned:



Searching and inductive bias

- This is a searching problem (local search)
  - In the space of all the decision trees
  - In each step we refine the current tree
  - No backtracking (local optima), hill-climbing search
  - Decisons made by analyzing sets of examples
- Inductive Bias
  - We prefer smaller trees
  - Preference bias, implicit in the search
  - Occam's razor principle

Some practicalquestions in machine learning

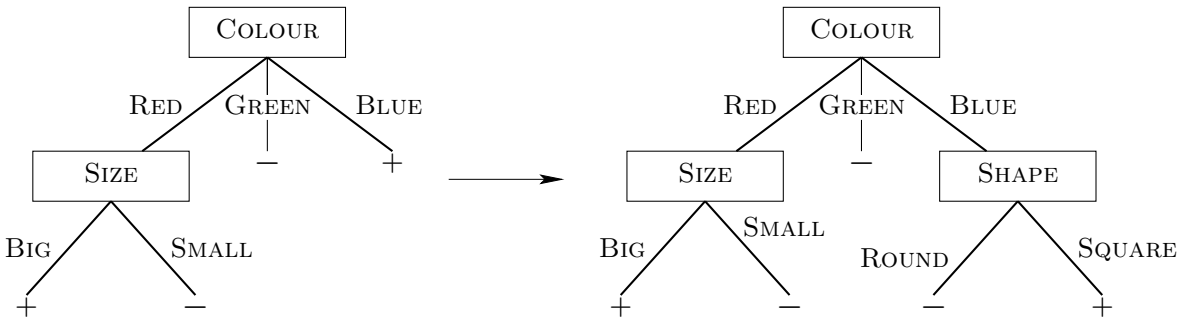
- Validate the learned hypothesis
  - Can we *quantify* how good is what we have learned, compared to the real explanation?
- Overfitting
  - Is it *too close* to the training examples?

Assesing whta we have learned

- Traning and test sets (*test*)
  - Learn with the training set
  - Measure the performance using the test set:
    - *Classification accuracy*: percent of correctly classified examples in the test set
- Repeat this proccess
  - Learning curve
  - Startification: each class has to be properly represented both in the traing set and in the test set
- Sometimes, it is not possible to "waste" the examples in the test set and we do *cross-validation*:
  - Partition the set of examples in k equal sized parts. Apply the learning algorithm k times, each one taking as test set one of the parts and learning with the rest. Return the average of the k clasification accuracies.
  - In practice: cross-validation, with $k = 10$ and stratification

Overfitting and noise

- We say that a learned model is *overfitted* if it is so fitted to the training set that it has a poor predictive performance on the rest of unseen examples
- *Noise:* examples in the training set, incorrectly classifed. This causes overfitting
- Example: assume that by mistake, we include the folloewing example
  <BLUE,ROUND,SMAL> as a negative example
- The learned model in this case would be (overfitted to the data):



Overfittinbg and noise

- Other sources of overfitting:
  - Attributes presenting an apparent regularity in the training set, but actually irrelevant
  - Small training sets
- Ways to avoid overfitting in decision trees:
  - Early stopping
  - *A posteriori* tree pruning
- *A posteriori* pruning, two approaches:
  - Transformation to rules, prune rule conditions
  - Prune the tree
  - Only prune when improving classification accuracy on a test set

Tree pruning

Reduced error pruning

```
1. Partition the set of examples into Training and Test sets
2. Tree=tree obtained using ID3 on the Trainig set
3. Continue=True
4. While Continue:
 * Measure = classification accuracy of the leraned tree on Test
 * For every internal node N of Tree:
   - Temporarily prune the subtree of Tree at node N, replacing it
     by a leaf labeled with the majority clase at that node
   - Compute the classification accuracy of the pruned tree,
     on the test set
 * Let K the node with best accuracy
 * If its accuracy is better than Measure, then
   Tree = permanently prune Tree at node K
 * Otherwise, Continue=Falso
5. Return Tree
```

More practical questions for tree learning

- ID3 extensions:
  - Real-valued attributes
  - Other measures for selecting the bet node
  - Error estimates
  - Missing values in attributes
  - Attributes with cost
- Algorithms C4.5 and C5.0 (Quinlan)

Section 3

Learning rules

Changing the formalism

- Classification rules:
  - R1: If Outlook=Sunny $\wedge$ Humidity=High
    Then PlayTennis= $-$
  - R2: If Astigmatism=+ $\wedge$ Tear_Rate=Normal
    Then Lenses=Hard
- Advantages of using rules:
  - Clarity
  - Modularity
  - Expresiveness: rules can represent every subset of instances
  - Methods can naturally be extended to first–order rules,
  - Formalism used in knowledge based systems
- Rules and decison trees
  - Easy translation from trees to rules, but not in the other way

Rule learning

- Goal: learn a set of rules consistent with the exaples
  - A rule *covers* an example if the example satisfies its conditions
  - It *correctly covers* the example if the class of the example is equal to the class in the the rule conclusion
- Measuring the accuracy of a rule R with respect to a set of examples D:
  - *Relative frequency*: p/t (where $t =$ examples in D covered by R, $p =$ examples correctly covered). Notation: $RF(R, D)$
- Rule learning algorithms:
  - ID3 + traslation to rules
  - Covering
  - Genetic algorithms

Training set

Ex.	Age	Prescription	Astigmatism	Tear_rate	Lenses
E_1	young	myope	-	reduced	none
E_2	young	myope	-	normal	soft
E_3	young	myope	+	reduced	none
E_4	young	myope	+	normal	hard
E_5	young	hypermetrope	-	reduced	none
E_6	young	hypermetrope	-	normal	soft
E_7	young	hypermetrope	+	reduced	none
E_8	young	hypermetrope	+	normal	hard
E_9	pre-presbyopic	myope	-	reduced	none
E_{10}	pre-presbyopic	myope	-	normal	soft
E_{11}	pre-presbyopic	myope	+	reduced	none
E_{12}	pre-presbyopic	myope	+	normal	hard
E_{13}	pre-presbyopic	hypermetrope	-	reduced	none
E_{14}	pre-presbyopic	hypermetrope	-	normal	soft
E_{15}	pre-presbyopic	hypermetrope	+	reduced	none
E_{16}	pre-presbyopic	hypermetrope	+	normal	none

Un conjunto de entrenamiento

Ej.	Age	Prescription	Astigmatism	Tear_Rate	Lenses
E_{17}	presbyopic	myope	-	reduced	none
E_{18}	presbyopic	myope	-	normal	none
E_{19}	presbyopic	myope	+	reduced	none
E_{20}	presbyopic	myope	+	normal	hard
E_{21}	presbyopic	hypermetrope	-	reduced	none
E_{22}	presbyopic	hypermetrope	-	normal	soft
E_{23}	presbyopic	hypermetrope	+	reduced	none
E_{24}	presbyopic	hypermetrope		normal	none

- R2: If Astigmatism=+ $\wedge$ Tear_Rate=Normal
  Then Lenses=Hard
- R2 covers E_4, E_8, E_{12}, E_{16}, E_{20} y E_{24}, but only E_4, E_8, E_{12} y E_{20} are correctly covered

Learning rules to cover examples

- Learn a rule to classify Lenses=Hard
  - If ?
    Then Lenses=Hard
  - Different choices for ?, and the relative frequency of the resulting rule:

Age=Young	2/8
Age=PrePresbyopic	1/8
Age=Presbyopic	1/8
Prescription=Myope	3/12
Prescription=Hypermetrope	1/12
Astigmatism=−	0/12
Astigmatism=+	4/12 *
Tear_Rate=Reduced	0/12
Tear_Rate=Normal	4/12 *

- Partially learned rule:
  - If Astigmatism=+
    Then Lenses=Hard

Learnig rules to cover examples

- We continue, to exclude incorrectly covered examples:
  - If Astigmatism=+ $\wedge$?
    Then Lenses=Hard
  - Choices for ?, and relative frequency of the resulting rule:

Age=Young	2/4
Age=PrePresbyopic	1/4
Age=Presbyopic	1/4
Prescription=Myope	3/6
Prescription=Hypermetrope	1/6
Tear_Rate=Reduced	0/6
Tear_Rate=Normal	4/6 *

- Partially learned rule:
  - If Astigmatism=+ $\wedge$ Tear_Rate=Normal
    Then Lenses=Hard

Learning rules to cover examples

- We continue, to exclude incorrectly covered examples:
  - If Astigmatism=+ ∧ Tear_Rate=Normal ∧ ?
    Then Lenses=Hard
  - Choices for ?, and relative frequency of the resulting rule:

Age=Young	2/2 *
Age=PrePresbyopic	1/2
Age=Presbyopic	1/2
Prescription=Myope	3/3 *
Prescription=Hypermetrope	1/3

- Rule finally learned (no incorrectly covered examples)
  - If Astigmatism=+ ∧ Tear_Rate=Normal ∧
    Prescription=Myope
    Then Lenses=Hard

Learning rules to cover examples

- Still an uncovered example with LENSES=HARD
  - We start again "If ? Then LENSES=HARD", but now with $D' = D \setminus \{E_4, E_{12}, E_{20}\}$
- Rules finally learned for LENSES=HARD:
  - R1: If ASTIGMATISM= + $\wedge$ TEAR_RATE=NORMAL $\wedge$ PRESCRIPTION=MYOPE Then LENSES=HARD
  - R2: If AGE=YOUNG $\wedge$ ASTIGMATISM= + $\wedge$ TEAR_RATE=NORMAL Then LENSES=HARD
  - Thr rules correctly cover the 4 examples of LENSES=HARD (actually, they overlap)
- We now could continue learning rules for:
  - LENSES=SOFT
  - LENSES=NONE

Sequential covering algorithm for learning rules

Sequential covering algorithm

```
Sequential-Covering(D,Class,c)
1. Initialize Learned-Rules to the empty set
2. Initialize E to D
3. While E contains examples with Class c, do:
3.1 Create a rule R with no conditions and conclusion Class=c
3.2 While there are examples in E incorrectly covered by R and
    there are attributres to use, do:
3.2.1 Choose the BEST condition A=v to add to R, where A is
      an attribute not appearing in R and v is a value of A
3.2.2 Update R adding condition A=v to R
3.3 Add R to Learned-Rules
3.4 Uodate E, removing the examples covered by R
4. Return Learned-Rules
```

- Algorithm to learn rules from a training set D
  - Rules predicting situations where the class is a given value c

Sequential covering algorithm: control

- Outer loop:
  - Adds rules (the model is *generalized*)
  - Each rule covers some examples, all of them correctly
  - In every step, the covered examples are removed
  - And rules are added while there are examples not covered
- Inner loop:
  - Adds conditions to the rule (the rule is specialized)
  - Each new condition excludes examples incorrectly covered
  - And this is done while there are examples incorrectly covered
- Sequential covering *vs* ID3
  - It learns one rule at a time; in contrast, ID3 learns globally
  - ID3: choose attributes
  - Sequential covering: choose pairs attribute-value

Sequential covering (properties)

- There are several criteria to choose the "best" condition in the inner loop
  - Choose the condition with *the greatest relative frequency* (as shown in the example)
  - Choose the one with *greatest information gain*:
    $$p \cdot (log_2 \frac{p'}{t'} - log_2 \frac{p}{t})$$
    wher p'/t' is the relative frequency *after* adding the condition and p/t is the relative frequency *before* adding the condition
- The set of rules learned by the sequential covering algorithm *perfectly* fit the training set (*danger of overfitting*)
  - *A posteriori* rule pruning: incrementally remove conditions, until no *improvement* is produced
  - *A priori* rule pruning: early stopping rule generation

Sección

Instance–based learning: kNN

Classification by means of nearest neighbour

- An alternative to building probabilistic models is to find a way to calculate the classification for a new instance directly from previous examples (*instance-based learning*)
- Idea: assign the classification of a new example by observing the categories of the "closest" examples.
  - We thus need a notion of "distance" between examples.
  - Most of the time we will handle examples belonging to R^n, and by default we may assume the Euclidean distance.
  - But of course any other particular distance function might be defined.
- Example of application: documents classification

k-NN algorithm

- k-NN algorithm (stands for *"k nearest neighbours"*):
  - Given a training set (numerical vectors with an associated category) and a new example
  - Return the category of the majority within the k examples of the training set which are the closest to the new example that we want to classify

Distances for k-NN

- Possible definitions of distance functions:
  - Euclidean: $d_e(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$
  - Manhattan: $d_m(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{n} |x_i - y_i|$
  - Hamming: number of components which differ.
- Euclidean is suitable when every dimension refers to similar properties, and Mahattan otherwise; Hamming distance can be used even when handling non-numerical values.
- Normalization: when not all dimensions fall within the same order of magnitude, components should be normalized (by substracting the average value and dividing by the standard deviation)

Some observations on k-NN

- Choice of k:
  - Usually, based on some specific knowledge about the classification problem
  - May be tuned after tests over smaller sets (validation sets)
  - When classification is binary, odd values avoid ties (k=5, for example)
- Variant of k-NN: for each class c, sum the similarity of every example of this class belonging to the k closests to the new example. Return the class which gets the highest sum.
  - In this way, the closer an example is, the more its classification is taken into account.

Section 5

Clustering

Clustering

- As the last application of statistical learning, let us now overview *clustering* techniques
- The goal is to divide an input data set into several subsets (*clusters*), in such a way that elements within the same subset share some kind of pattern or some characteristics which are previously unknown
- We shall focus on the case when data are expressed as numbers or as numerical vectors, and *the number of clusters is given in advance*
- *Unsupervised* learning: we have no prior information about which is the "correct" cluster that corresponds to each data.
- Application of clustering:
  - Data mining
  - Digital image processing
  - Bioinformatics

Two examples

- *Color quantization*:
  - A digital image stored with a resolution of 24 bits/pixel (aprox. 16 million colors) is to be displayed on a screen that allows only 8 bits/pixel (256 colors)
  - What is the best correspondence between the colors of the original image and the colors that can be displayed on the screen?
- Merge of distributions:
  - We have some data collection about the weight of people; data are not labelled so for each value we do not know if they refer to men or women. However, we know (or we assume) that there are two different weight distributions for men and women, and both of them follow a normal distribution
  - By just looking at the data, is it possible to *learn* which are the aforementioned two distributions?

Clustering based on distance

- Idea: given the number k of *clusters*, find k points or *centers* which will represent each of the clusters, in such a way that each example will be assigned to the cluster corresponding to the "nearest" center.
- Just like before, distance can be specific for each problem:
  - Expressing the degree of similarity
  - Usually Euclidean

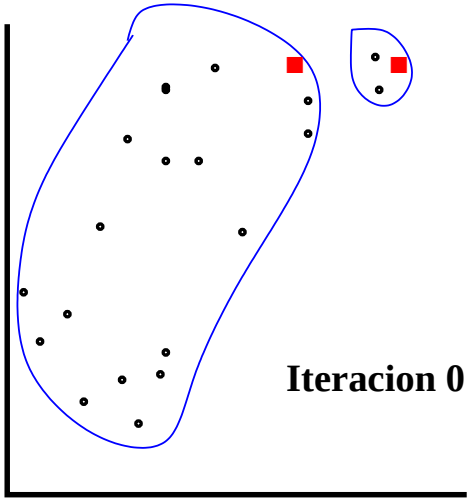
Classic algorithm: k-means

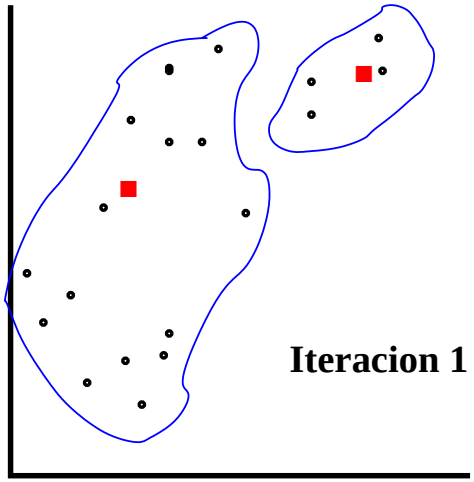
- Input: a number k of clusters, a data set $\{x_i\}_{i=1}^{N}$ and a distance function
- Output: a set of k centers $m_1, \ldots, m_k$

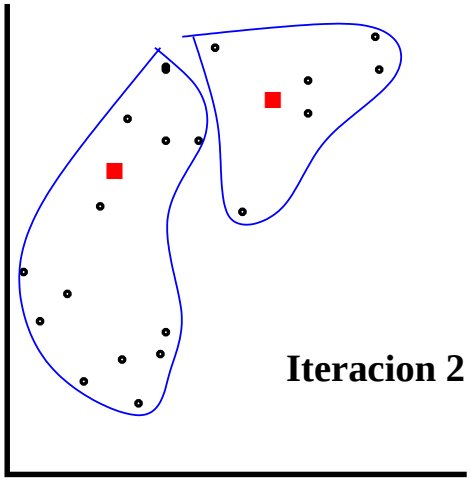
k-means(k,data,distance)

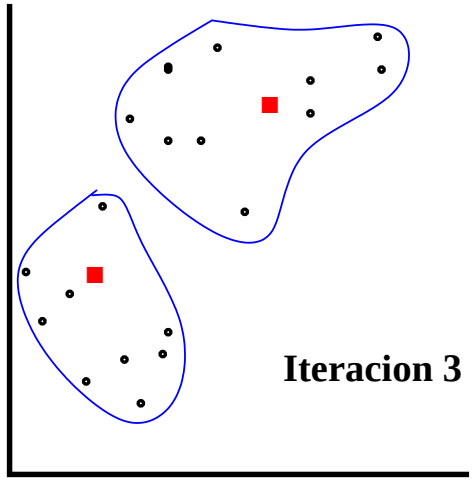
```
1. Initialize m_i (i=1,...,k) (randomly or with some heuristic
   criterion)
2. REPEAT (until m_i do not vary):
   2.1 FOR j=1,...,N, DO:
       Calculate the cluster corresponding to x_j, by selecting,
       among all m_i, the center m_h such that the
       distance(x_j,m_h) is minimum
   2.2 FOR i=1,...,k. DO:
       Update m_i assigning to it the arithmetic mean of the
       examples assigned to i-th cluster
3. Return m_1,...,m_n
```

Graphical intuition for k-means algorithm









Example of k-means algorithm

- Data about weght of the population: 51, 43, 62, 64, 45, 42, 46, 45, 45, 62, 47, 52, 64, 51, 65, 48, 49, 46, 64, 51, 52, 62, 49, 48, 62, 43, 40, 48, 64, 51, 63, 43, 65, 66, 65, 46, 39, 62, 64, 52, 63, 64, 48, 64, 48, 51, 48, 64, 42, 48, 41
- The algorithm, applied over $k = 2$ and with Euclidean distance, finds two centers $m_1 = 63{,}63$ and $m_2 = 46{,}81$ on three iterations
- 19 data belong to the first cluster and 32 to the second one

Other considerations on the k-means algorithm

- Local search:
  - It can be seen as a local search algorithm, trying to find the centers m_i which optimize $\sum_j \sum_i b_{ij} d(x_j, m_i)^2$, where b_{ij} is 1 if x_j has m_i as its closest center, or 0 otherwise
  - As any other local search algorithm, does not guarantee finding the global optimum
- Initialization: random or with some heuristic technique (for example, randomly split the data into k clusters, and start the algorithm execution using as input the centers of such clusters)
- In practice, the centers provided as input may make a great difference with respect to the quality of the obtained results

Another example of k-means algorithm

- The file **iris.arff** of WEKA contains 150 data vectors about the petal and sepal width and length of iris plants, which are classified into three different subtypes (setosa, versicolor and virginica)
  - Example of instance within **iris.arff** file: **5.1,3.5,1.4,0.2,Iris-setosa**
- We can apply k-means, with $k = 3$ and Euclidean distance, ignoring the last attribute (pretending it is unknown):
  - After 6 iterations it stabilizes
  - Of the three resulting clusters, the first of them is composed exactly by the 50 instances which were originally classified as "iris setosa"
  - The second cluster includes 47 versicolor and 3 virginicas
  - The third includes 14 versicolor and 36 virginicas
  - It was not able to correctly tell apart versicolor and virginica specimens

Bibliography

- Mitchell, T.M. *Machine Learning* (McGraw-Hill, 1997)
  - Chapters 3,8 and 10
- Russell, S. and Norvig, P. *Artificial Intelligence (A Modern Approach)* (3rd edition) (Prentice Hall, 2010)
  - Sections 18.1, 18.2, 18.3 and 18.8
- Witten, I.H. y Frank, E. *Data mining* (Third edition) (Morgan Kaufmann Publishers, 2011)
  - Chapters 3, 4, 5 and 6.
- Alpaydin, E. *Introduction to Machine Learning* (third edition) (The MIT Press, 2014)
  - Chapter 9, and Section 7.3