

---

# *Pattern Analysis and Machine Intelligence*

*Lecture Notes on Clustering (I)*  
*2013-2014*

Davide Eynard

davide.eynard@usi.ch

Department of Electronics and Information  
Politecnico di Milano

---

## Some Info

---

- Lectures given by:
  - Davide Eynard (Teaching Assistant)  
<http://davide.eynard.it>  
[davide.eynard@usi.ch](mailto:davide.eynard@usi.ch)
- Course Material on Clustering
  - These lecture notes
  - Papers and tutorials (check *Bibliography* at the end)
  - Hastie, Tibishirani, Friedman: "The Elements of Statistical Learning: Data Mining, Inference, and Prediction"
- Web Links
  - up-to-date links within these slides

## Course Schedule [*Tentative*]

---

Date	Topic
25/11/2013	Clustering I: Introduction, K-means
29/11/2013	Clustering II: K-M alternatives, Hierarchical, SOM
02/12/2013	Clustering III: Mixture of Gaussians, DBSCAN, J-P
13/12/2013	Clustering IV: Spectral Clustering, cluster evaluation

---

# Today's Outline

---

- clustering definition and application examples
- clustering requirements and limitations
- clustering algorithms classification
- distances and similarities
- our first clustering algorithm: K-means

---

## Clustering: a definition

---

“The process of organizing objects into *groups* whose members are *similar in some way*”

J.A. Hartigan, 1975

“An algorithm by which objects are grouped in *classes*, so that intra-class *similarity* is maximized and inter-class similarity is minimized”

J. Han and M. Kamber, 2000

“... grouping or segmenting a collection of objects into subsets or *clusters*, such that those within each cluster are more closely *related* to one another than objects assigned to different clusters”

T. Hastie, R. Tibshirani, J. Friedman, 2009

---

# Clustering: a definition

---

- Clustering is an *unsupervised learning* algorithm
  - “**Exploit regularities** in the inputs to **build a representation** that can be used for reasoning or prediction”
- Particular attention to
  - *groups/classes* (vs *outliers*)
  - *distance/similarity*
- What makes a good clustering?
  - No (independent) best criterion
  - **data reduction** (find representatives for homogeneous groups)
  - **natural data types** (describe unknown properties of natural clusters)
  - **useful data classes** (find useful and suitable groupings)
  - **outlier detection** (find unusual data objects)

---

## (Some) Applications of Clustering

---

- Market research
  - find groups of customers with similar behavior for targeted advertising
- Biology
  - classification of plants and animals given their features
- Insurance, telephone companies
  - group customers with similar behavior
  - identify frauds
- On the Web:
  - document classification
  - cluster Web log data to discover groups of similar access patterns
  - recommendation systems ("If you liked this, you might also like that")

---

## Example: Clustering (CDs/Movies/Books/...)

---

- Intuitively: users prefer some (music/movie/book/...) categories, but what are categories actually?
- Represent an item by the users who (like/rent/buy) it
- Similar items have similar sets of users, and vice-versa
- Think of a space with one dimension for each user (values in a dimension may be 0 or 1 only)
- An item point in the space is  $(x_1, x_2, \dots, x_k)$ , where  $x_i = 1$  iff the  $i^{th}$  user liked it
- Items are similar if they are close in this  $k$ -dimensional space
- Exploit a clustering algorithm to group similar items together



---

# Requirements

---

- Scalability
- Dealing with different types of attributes
- Discovering clusters with arbitrary shapes
- Minimal requirements for domain knowledge to determine input parameters
- Ability to deal with noise and outliers
- Insensitivity to the order of input records
- High dimensionality
- Interpretability and usability

# Question

---

What if we had a dataset like this?



---

# Problems

---

There are a number of problems with clustering. Among them:

- current clustering techniques do not address all the requirements adequately (and concurrently);
- dealing with large number of dimensions and large number of data items can be problematic because of time complexity;
- the effectiveness of the method depends on the definition of *distance* (for distance-based clustering);
- if an obvious distance measure does not exist we must define it (which is not always easy, especially in multi-dimensional spaces);
- the result of the clustering algorithm (that in many cases can be arbitrary itself) can be interpreted in different ways (see Boyd, Crawford: "Six Provocations for Big Data": pdf, video).

---

# Clustering Algorithms Classification

---

- Exclusive vs Overlapping
- Hierarchical vs Flat
- Top-down vs Bottom-up
- Deterministic vs Probabilistic
- Data: symbols or numbers

---

# Distance Measures

---

Two major classes of distance measure:

- Euclidean
  - A Euclidean space has some number of real-valued dimensions and "dense" points
  - There is a notion of *average* of two points
  - A Euclidean distance is based on the locations of points in such a space
- Non-Euclidean
  - A Non-Euclidean distance is based on properties of points, but not on their *location* in a space

# Distance Measures

---

Axioms of a Distance Measure:

- $d$  is a *distance measure* if it is a function from pairs of points to reals such that:
  1.  $d(x, y) \geq 0$
  2.  $d(x, y) = 0$  iff  $x = y$
  3.  $d(x, y) = d(y, x)$
  4.  $d(x, y) \leq d(x, z) + d(z, y)$  (triangle inequality)

---

## Distances vs Similarities

---

- Distances are normally used to measure the similarity or dissimilarity between two data objects...
- ... However they are two different things!
- e.g. dissimilarities can be judged by a set of users in a survey
  - they do not necessarily satisfy the triangle inequality
  - they can be 0 even if two objects are not the same
  - they can be asymmetric (in this case their average can be calculated)

---

## Similarity through distance

---

- Simplest case: one numeric attribute  $A$ 
  - $Distance(X, Y) = A(X) - A(Y)$
- Several numeric attributes
  - $Distance(X, Y) =$  Euclidean distance between  $X$  and  $Y$
- Nominal attributes
  - Distance is set to 1 if values are different, 0 if they are equal
- Are all attributes equally important?
  - Weighting the attributes might be necessary



## Distances for numeric attributes

---

- **Minkowski distance:**

$$d_{ij} = \sqrt[q]{\sum_{k=1}^n |x_{ik} - x_{jk}|^q}$$

- where  $i = (x_{i1}, x_{i2}, \dots, x_{in})$  and  $j = (x_{j1}, x_{j2}, \dots, x_{jn})$  are two p-dimensional data objects, and  $q$  is a positive integer

# Distances for numeric attributes

- **Minkowski distance:**

$$d_{ij} = \sqrt[q]{\sum_{k=1}^n |x_{ik} - x_{jk}|^q}$$

- where  $i = (x_{i1}, x_{i2}, \dots, x_{in})$  and  $j = (x_{j1}, x_{j2}, \dots, x_{jn})$  are two p-dimensional data objects, and  $q$  is a positive integer

- if  $q = 1$ ,  $d$  is **Manhattan distance:**

$$d_{ij} = \sum_{k=1}^n |x_{ik} - x_{jk}|$$

# Distances for numeric attributes

- **Minkowski distance:**

$$d_{ij} = \sqrt[q]{\sum_{k=1}^n |x_{ik} - x_{jk}|^q}$$

- where  $i = (x_{i1}, x_{i2}, \dots, x_{in})$  and  $j = (x_{j1}, x_{j2}, \dots, x_{jn})$  are two p-dimensional data objects, and  $q$  is a positive integer

- if  $q = 2$ ,  $d$  is **Euclidean distance:**

$$d_{ij} = \sqrt{2}{\sum_{k=1}^n |x_{ik} - x_{jk}|^2}$$

---

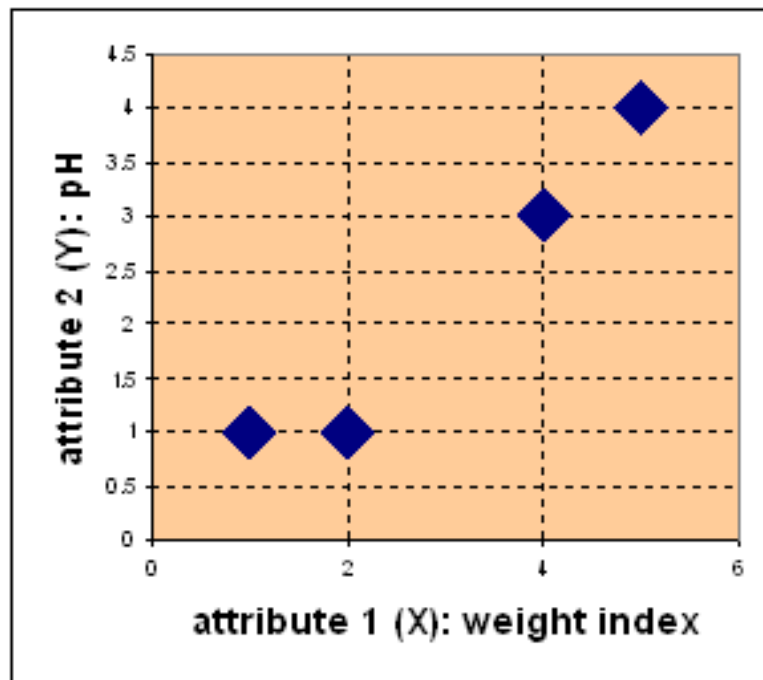
# K-Means Algorithm

---

- One of the simplest unsupervised learning algorithms
- Assumes Euclidean space (*works with numeric data only*)
- Number of clusters fixed a priori
- **How does it work?**
  1. Place  $K$  points into the space represented by the objects that are being clustered. These points represent initial group *centroids*.
  2. Assign each object to the group that has the closest centroid.
  3. When all objects have been assigned, recalculate the positions of the  $K$  centroids.
  4. Repeat Steps 2 and 3 until the centroids no longer move.

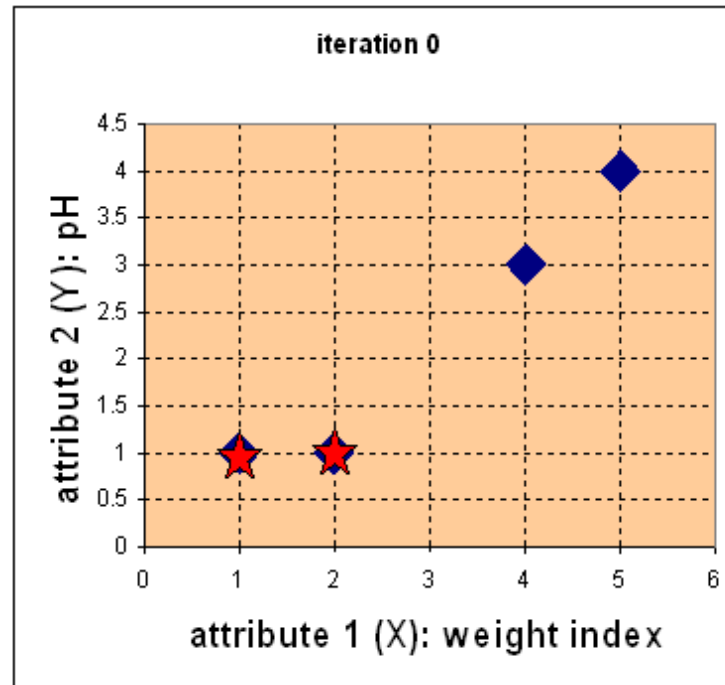
# K-Means: A numerical example

Object	Attribute 1 (X)	Attribute 2 (Y)
Medicine A	1	1
Medicine B	2	1
Medicine C	4	3
Medicine D	5	4



# K-Means: A numerical example

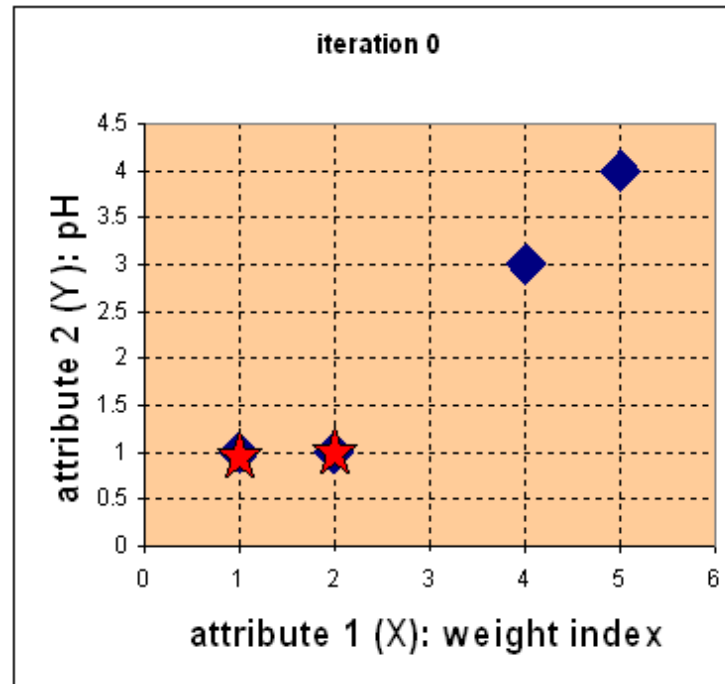
- Set initial value of centroids
  - $c_1 = (1, 1)$ ,  $c_2 = (2, 1)$



# K-Means: A numerical example

- Calculate Objects-Centroids distance

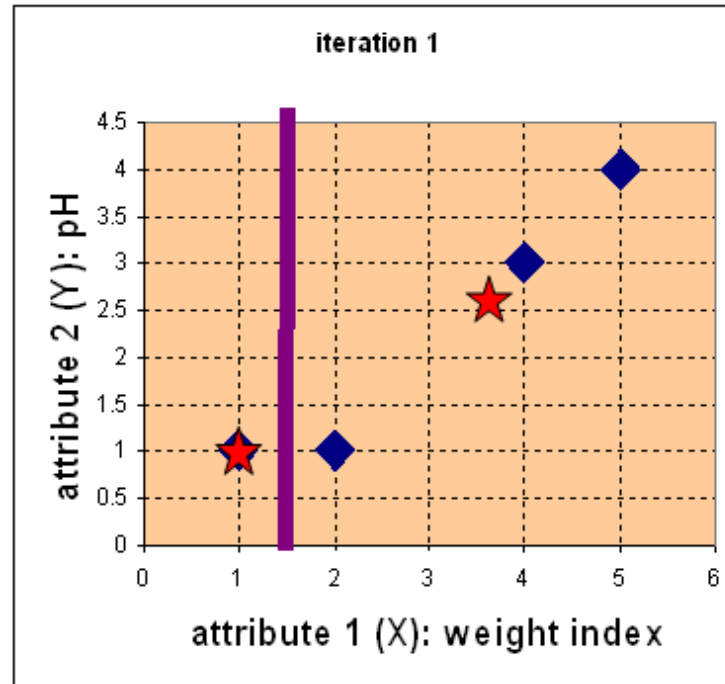
- $D^0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix}$   $c_1 = (1, 1)$   
 $c_2 = (2, 1)$



# K-Means: A numerical example

- Object Clustering

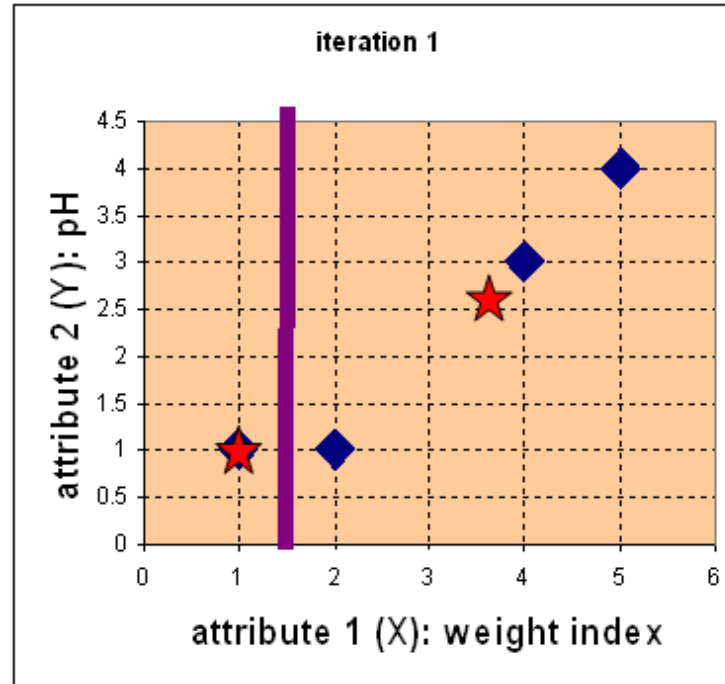
- $G^0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix}$  *group1*  
*group2*





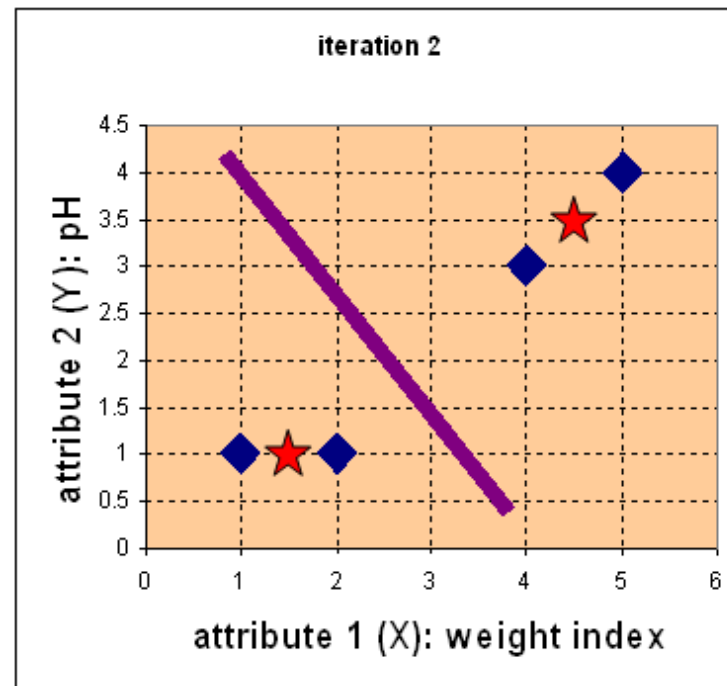
# K-Means: A numerical example

- Determine new centroids
  - $c_1 = (1, 1)$
  - $c_2 = \left(\frac{2+4+5}{3}, \frac{1+3+4}{3}\right) = \left(\frac{11}{3}, \frac{8}{3}\right)$



# K-Means: A numerical example

- $D^1 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix}$   $c_1 = (1, 1)$   
 $c_2 = (\frac{11}{3}, \frac{8}{3})$
- $G^1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \Rightarrow c_1 = (\frac{1+2}{2}, \frac{1+1}{2}) = (1.5, 1)$   
 $c_2 = (\frac{4+5}{2}, \frac{3+4}{2}) = (4.5, 3.5)$



---

## K-Means: still alive?

Time for some demos!

# K-Means: Summary

---

- Advantages:
  - Simple, understandable
  - Relatively efficient:  $O(tkn)$ , where  $n$  is #objects,  $k$  is #clusters, and  $t$  is #iterations ( $k, t \ll n$ )
  - Often terminates at a local optimum
- Disadvantages:
  - Works only when mean is defined (what about categorical data?)
  - Need to specify  $k$ , the number of clusters, in advance
  - Unable to handle noisy data (too sensible to outliers)
  - Not suitable to discover clusters with non-convex shapes
  - Results depend on the metric used to measure distances and on the value of  $k$
- Suggestions
  - Choose a way to initialize means (i.e. randomly choose  $k$  samples)
  - Start with *distant* means, run many times with different starting points
  - Use another algorithm ;-)

# K-Means application: Vector Quantization

---

- Used for image and signal compression
- Performs *lossy compression* according to the following steps:
  - break the original image into  $n \times m$  blocks (e.g. 2x2);
  - every fragment is described by a vector in  $\mathbb{R}^{n \cdot m}$ ; ( $\mathbb{R}^4$  for the example above)
  - K-Means is run in this space, then each of the blocks is approximated by its closest cluster centroid (called *codeword*);
  - NOTE: the higher K is, the better the quality (and the worse the compression!).  
Expected size for the compressed data:  $\log_2(K)/(4 \cdot 8)$ .

---

## Bibliography

---

- "Metodologie per Sistemi Intelligenti" course - Clustering Tutorial Slides by P.L. Lanzi
- "Data mining" course - Clustering, Part I Tutorial slides by J.D. Ullman
- Satnam Alag: "Collective Intelligence in Action" (Manning, 2009)
- Hastie, Tibishirani, Friedman: "The Elements of Statistical Learning: Data Mining, Inference, and Prediction"

- 
- The end