UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Pattern Recognition and Machine Learning

Unsupervised Learning

Matteo Tomei

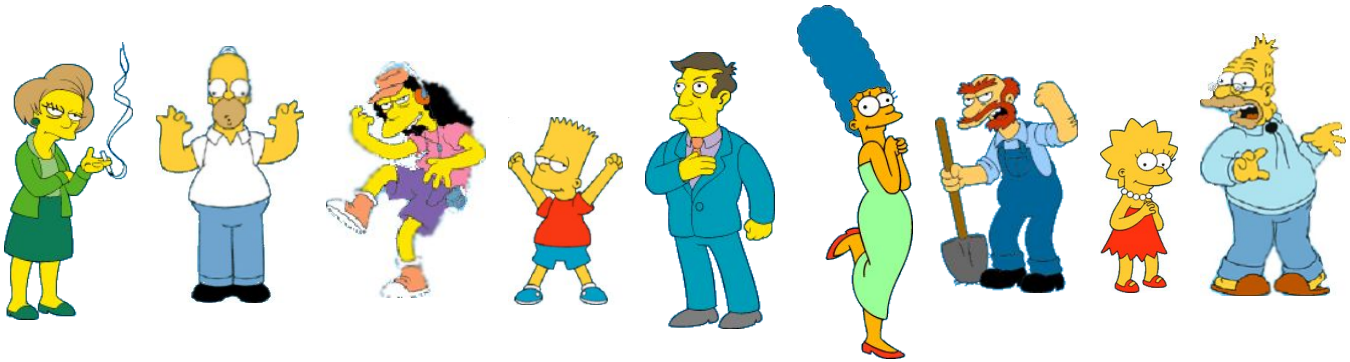
Dipartimento di Ingegneria «Enzo Ferrari»,
Università di Modena e Reggio Emilia

What is Clustering?

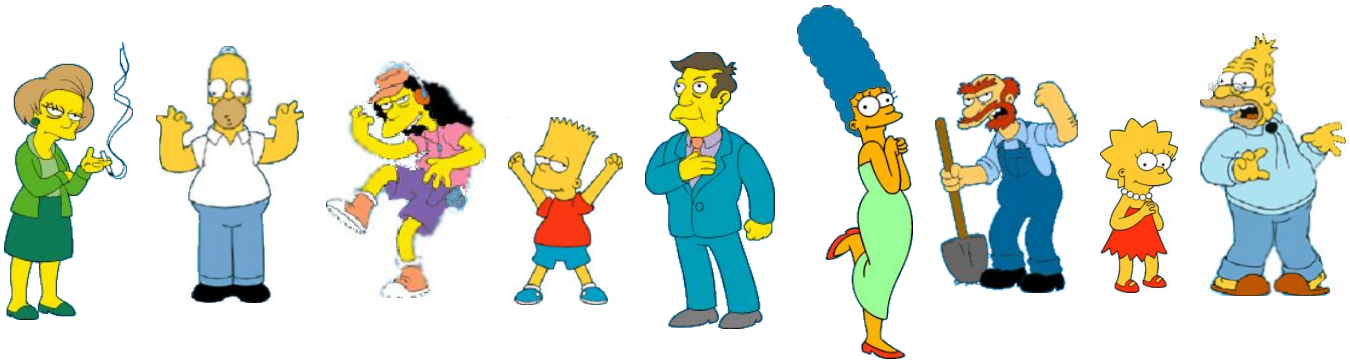
Also called *unsupervised learning*, sometimes called *classification* by statisticians and *sorting* by psychologists and *segmentation* by people in marketing

- Organizing data into classes such that there is
  - high intra-class similarity
  - low inter-class similarity
- Finding the class labels and the number of classes directly from the data (in contrast to classification).
- More informally, finding natural groupings among objects.

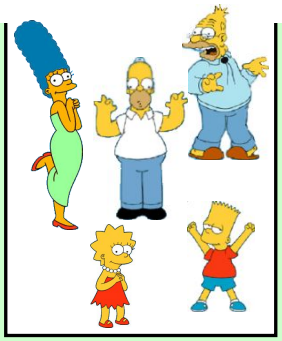
What is a natural grouping among these objects?



What is a natural grouping among these objects?

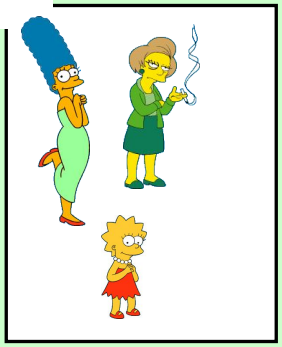


Clustering is subjective



Simpson's Family

School Employees



Females

Males

What is Similarity?

The quality or state of being similar; likeness; resemblance; as, a similarity of features.

Webster's Dictionary

Similarity is hard to define, but…
"*We know it when we see it*"

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

Intuitions behind desirable distance measure properties

$D(A,B) = D(B,A)$ *Symmetry*

Otherwise you could claim "Alex looks like Bob, but Bob looks nothing like Alex."

$D(A,A) = 0$ *Constancy of Self-Similarity*

Otherwise you could claim "Alex looks more like Bob, than Bob does."

$D(A,B) = 0$ IIf A=B *Positivity (Separation)*

Otherwise there are objects in your world that are different, but you cannot tell apart.

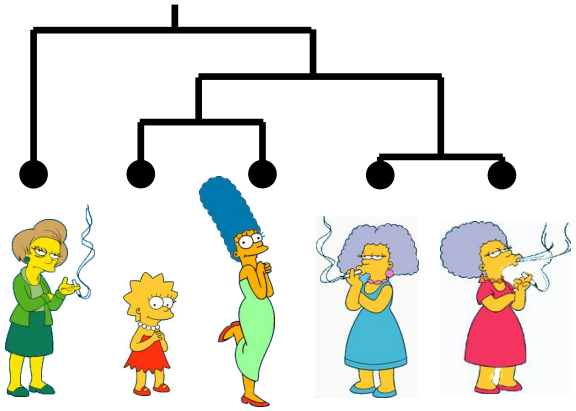
$D(A,B) \leq D(A,C) + D(B,C)$ *Triangular Inequality*

Otherwise you could claim "Alex is very like Bob, and Alex is very like Carl, but Bob is very unlike Carl."

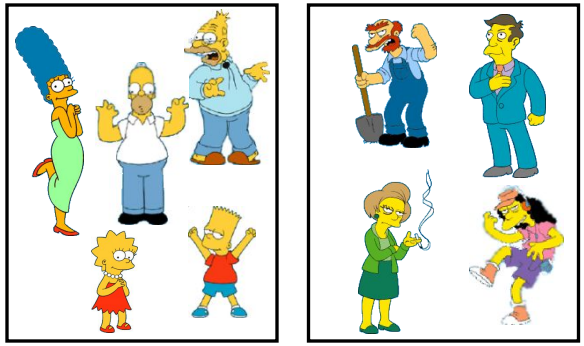
Two Types of Clustering

- **Partitional algorithms:** Construct various partitions and then evaluate them by some criterion (we will see an example called BIRCH)
- **Hierarchical algorithms:** Create a hierarchical decomposition of the set of objects using some criterion

Hierarchical



Partitional



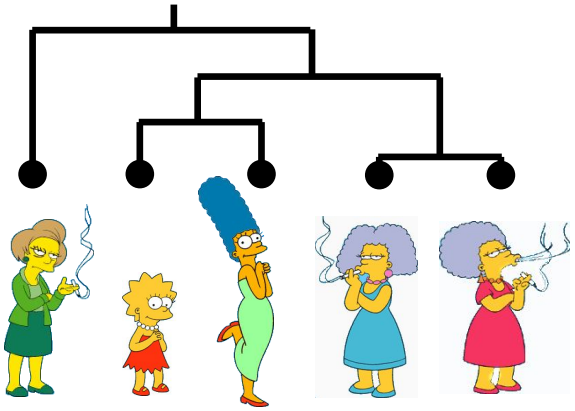
Desirable Properties of a Clustering Algorithm

- Scalability (in terms of both time and space)
- Ability to deal with different data types
- Minimal requirements for domain knowledge to determine input parameters
- Able to deal with noise and outliers
- Insensitive to order of input records
- Incorporation of user-specified constraints
- Interpretability and usability

(How-to) Hierarchical Clustering

The number of dendrograms with n leafs $= (2n-3)!/[(2^{(n-2)})(n-2)!]$

Number of Leafs	Number of Possible Dendrograms
2	1
3	3
4	15
5	105
...	…
10	34,459,425



Since we cannot test all possible trees we will have to heuristic search of all possible trees. We could do this..

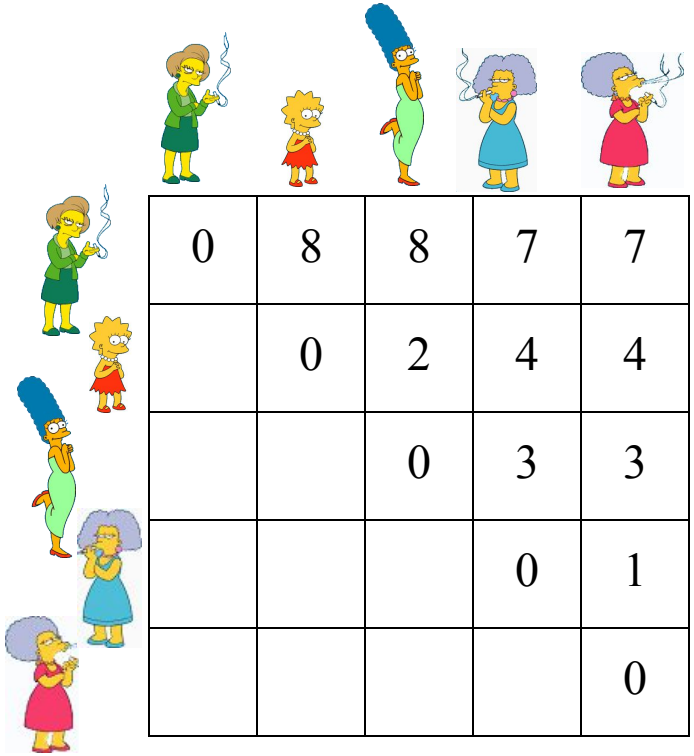
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

Top-Down (divisive): Starting with all the data in a single cluster, consider every possible way to divide the cluster into two. Choose the best division and recursively operate on both sides.

We begin with a distance matrix which contains the distances between every pair of objects in our database.

$D($ $) = 8$

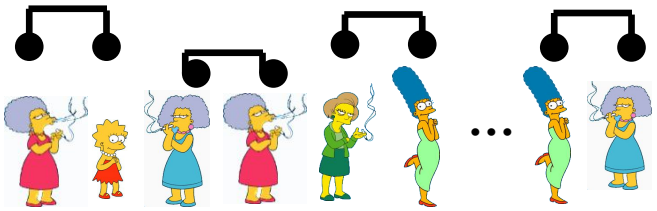
$D($ $) = 1$



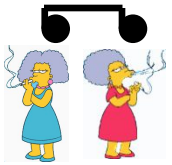
0	8	8	7	7
	0	2	4	4
		0	3	3
			0	1
				0

Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

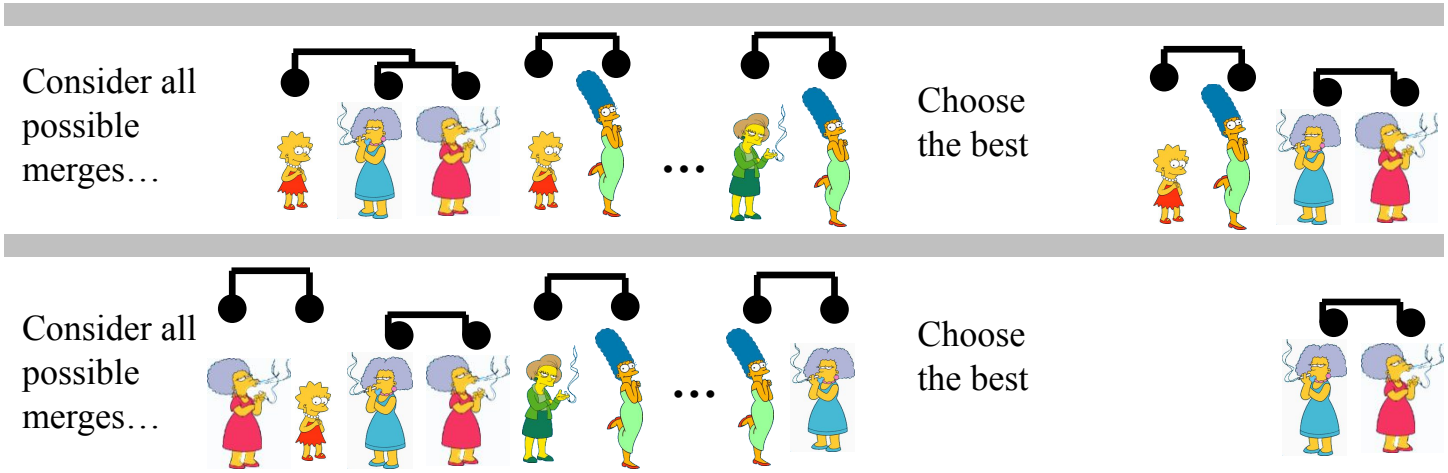
Consider all possible merges…



Choose the best

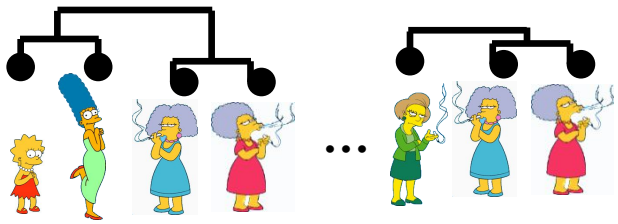


Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

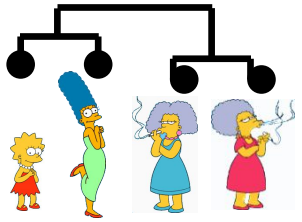


Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.

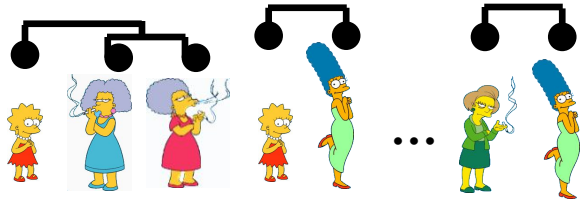
Consider all possible merges…



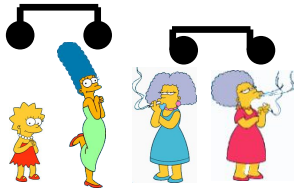
Choose the best



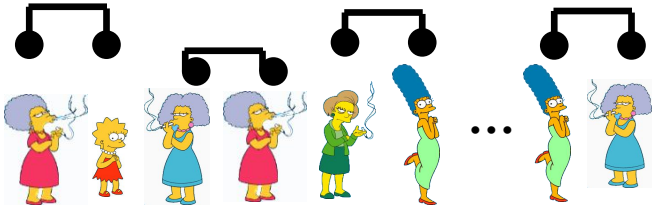
Consider all possible merges…



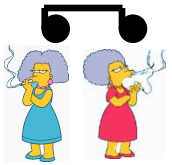
Choose the best



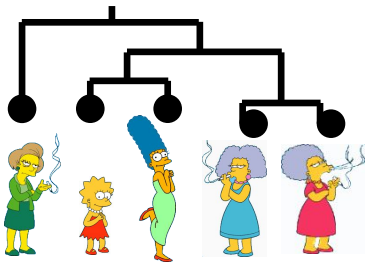
Consider all possible merges…



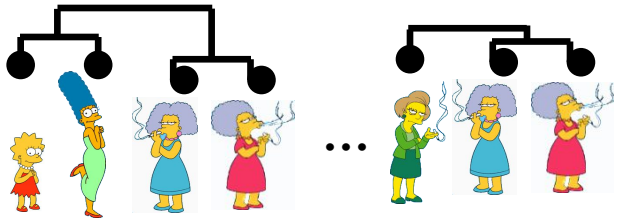
Choose the best



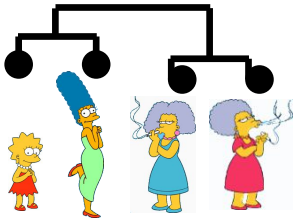
Bottom-Up (agglomerative): Starting with each item in its own cluster, find the best pair to merge into a new cluster. Repeat until all clusters are fused together.



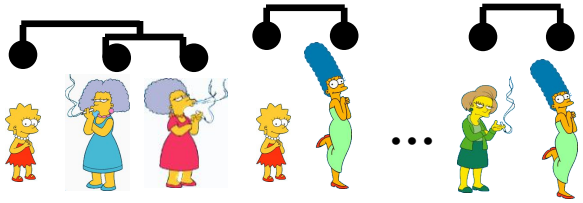
Consider all possible merges…



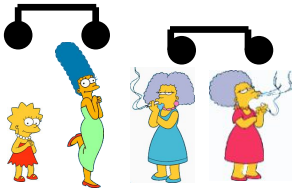
Choose the best



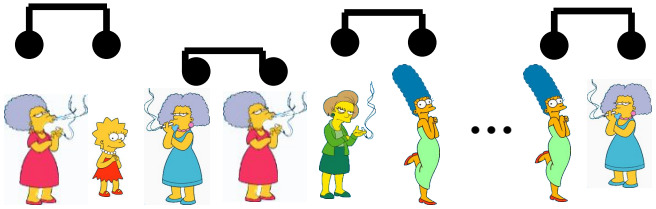
Consider all possible merges…



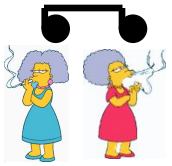
Choose the best



Consider all possible merges…

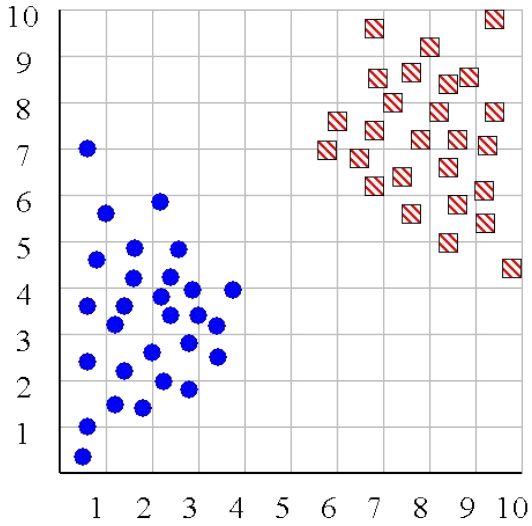


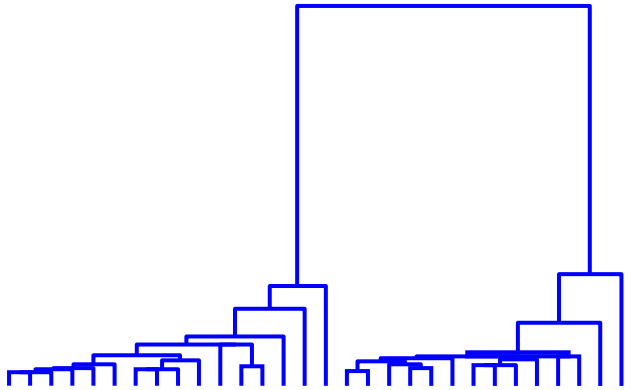
Choose the best



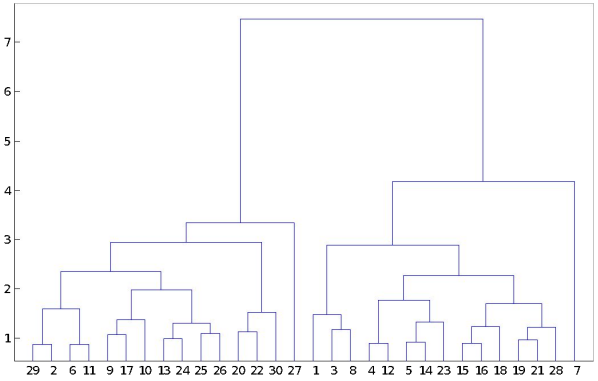
We know how to measure the distance between two objects, but defining the distance between an object and a cluster, or defining the distance between two clusters is non obvious.

- **Single linkage (nearest neighbor):** In this method the distance between two clusters is determined by the distance of the two closest objects (nearest neighbors) in the different clusters.
- **Complete linkage (furthest neighbor):** In this method, the distances between clusters are determined by the greatest distance between any two objects in the different clusters (i.e., by the "furthest neighbors").
- **Group average linkage:** In this method, the distance between two clusters is calculated as the average distance between all pairs of objects in the two different clusters.
- **Wards Linkage**: In this method, we try to minimize the variance of the merged clusters

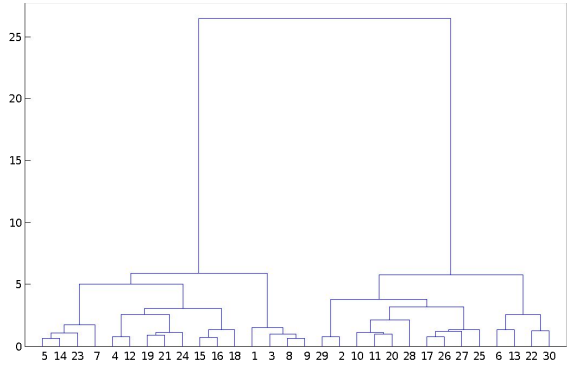




Single linkage



Average linkage



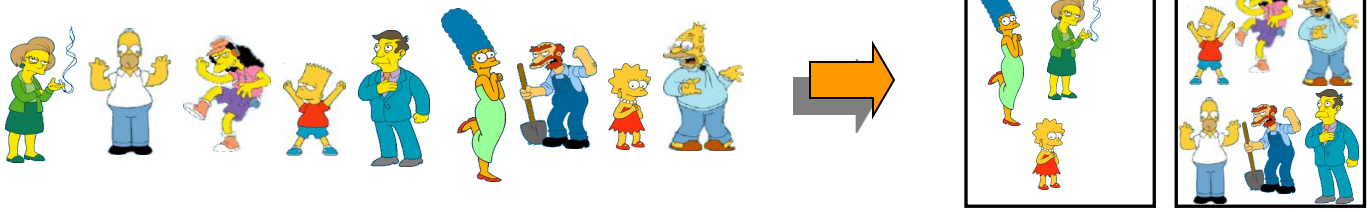
Wards linkage

Summary of Hierarchal Clustering Methods

- No need to specify the number of clusters in advance.
- Hierarchal nature maps nicely onto human intuition for some domains
- They do not scale well: time complexity of at least $O(n^2)$, where n is the number of total objects.
- Like any heuristic search algorithms, local optima are a problem.
- Interpretation of results is (very) subjective.

Partitional Clustering

- Nonhierarchical, each instance is placed in exactly one of K nonoverlapping clusters.
- Since only one set of clusters is output, the user normally has to input the desired number of clusters K.

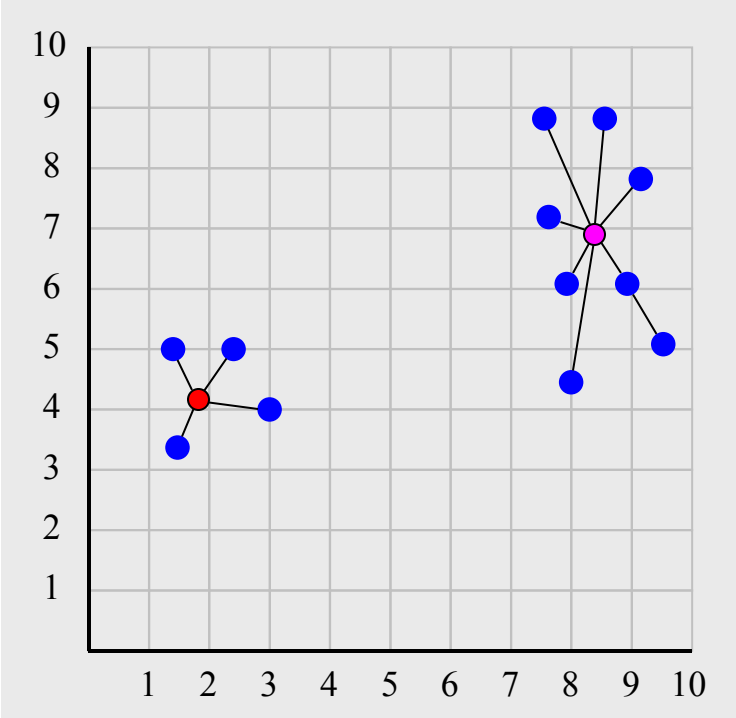


Squared Error

$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

Objective Function



Algorithm *k-means*

1. Decide on a value for *k*.

2. Initialize the *k* cluster centers (randomly, if necessary).

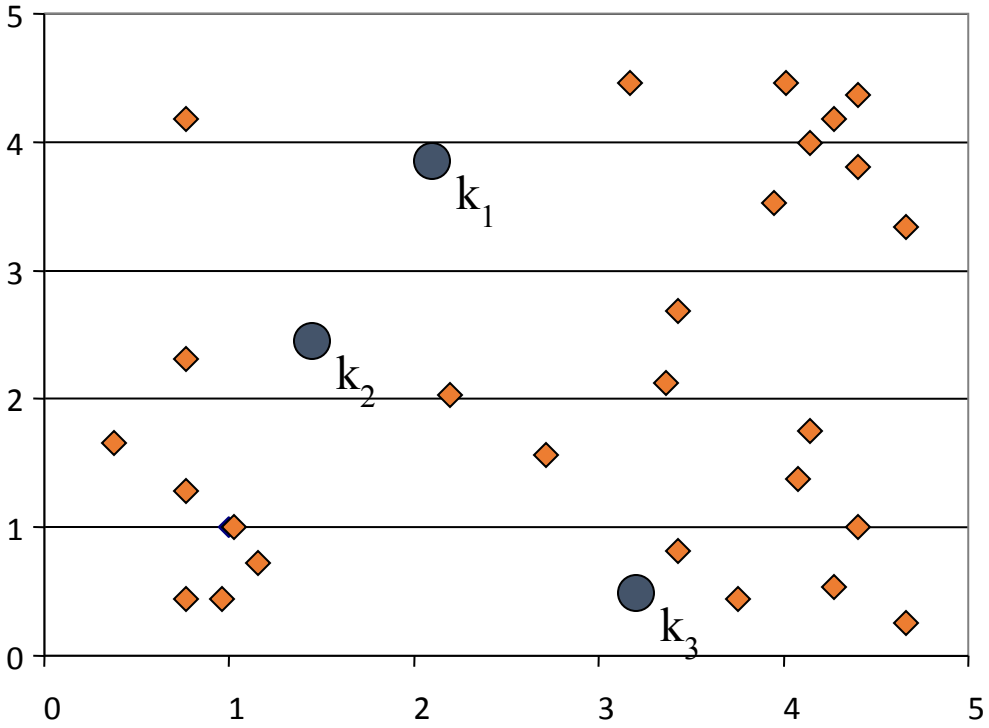
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster center.

4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.

5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise goto 3.

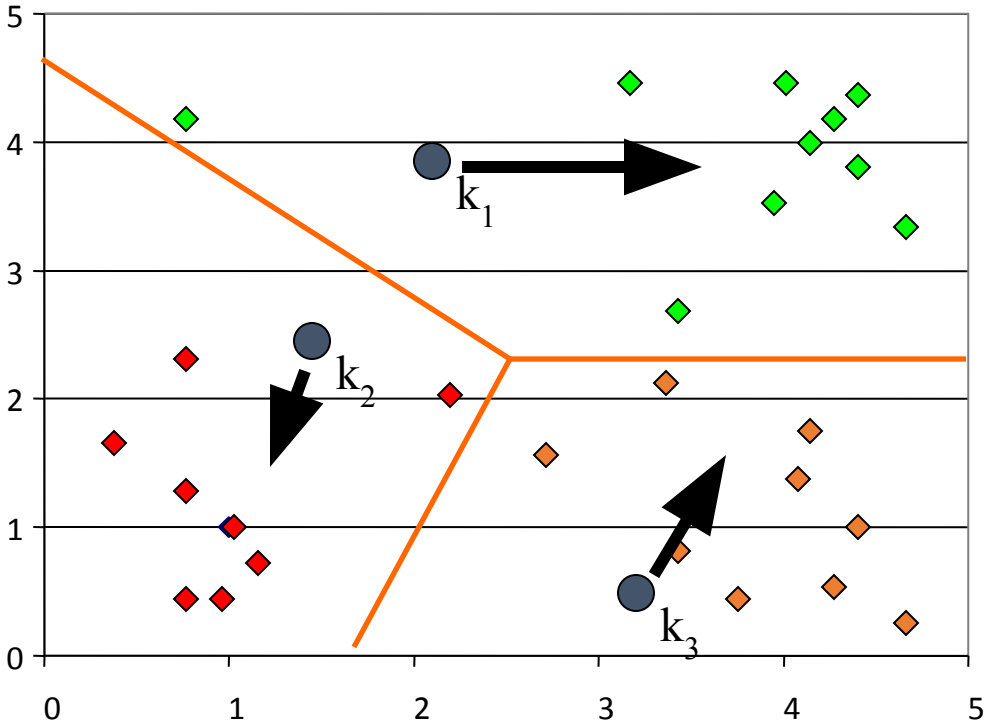
K-means Clustering: Step 1

Algorithm: k-means, Distance Metric: Euclidean Distance



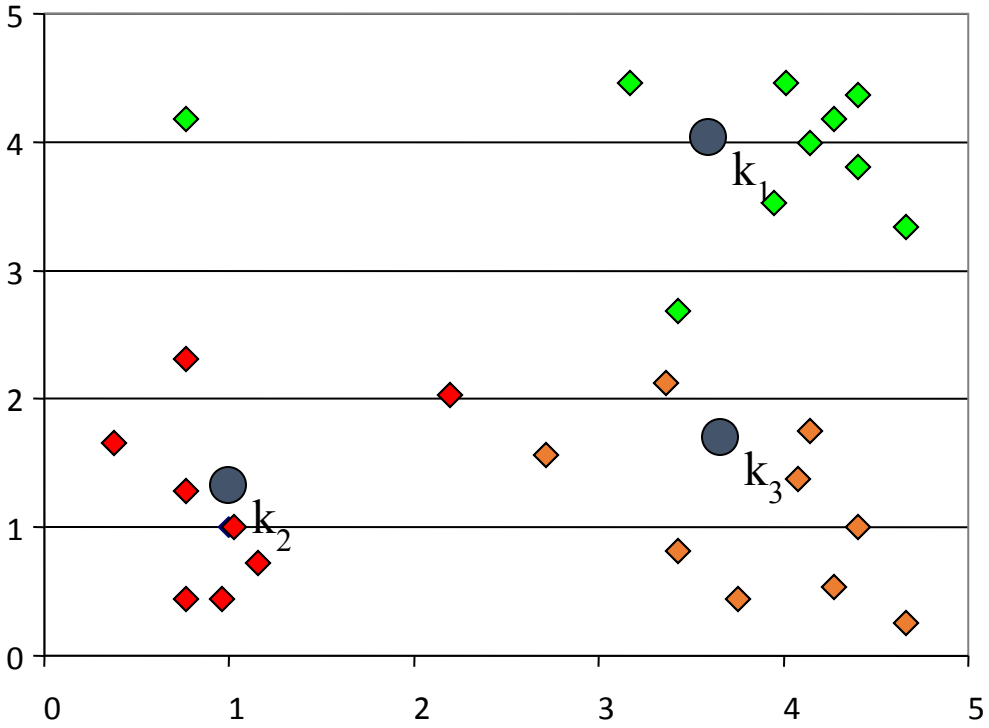
K-means Clustering: Step 2

Algorithm: k-means, Distance Metric: Euclidean Distance



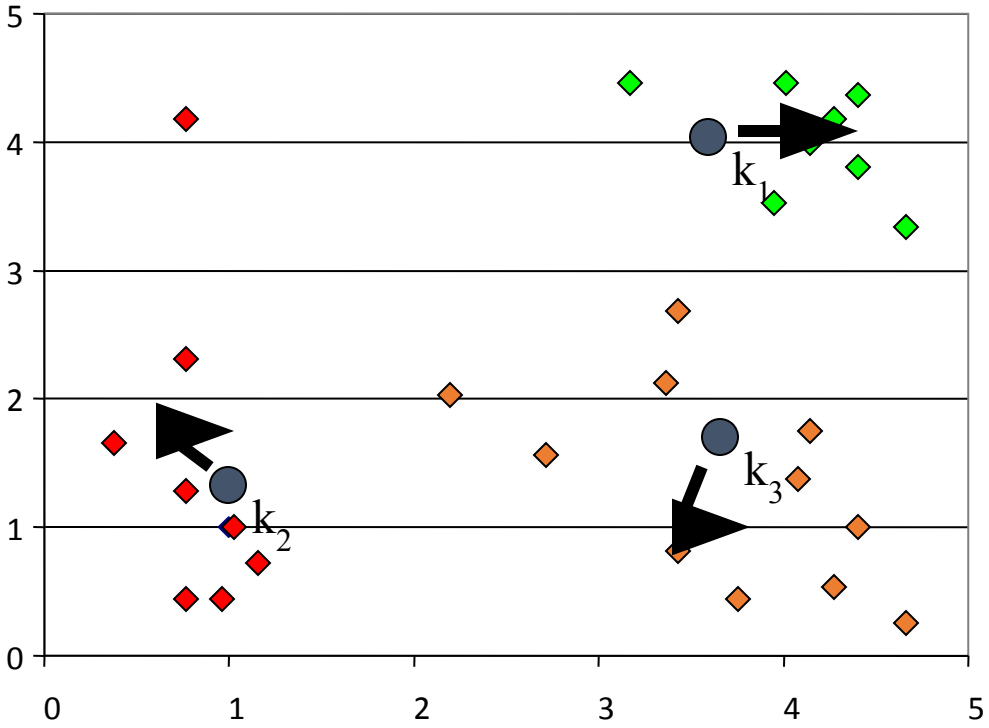
K-means Clustering: Step 3

Algorithm: k-means, Distance Metric: Euclidean Distance



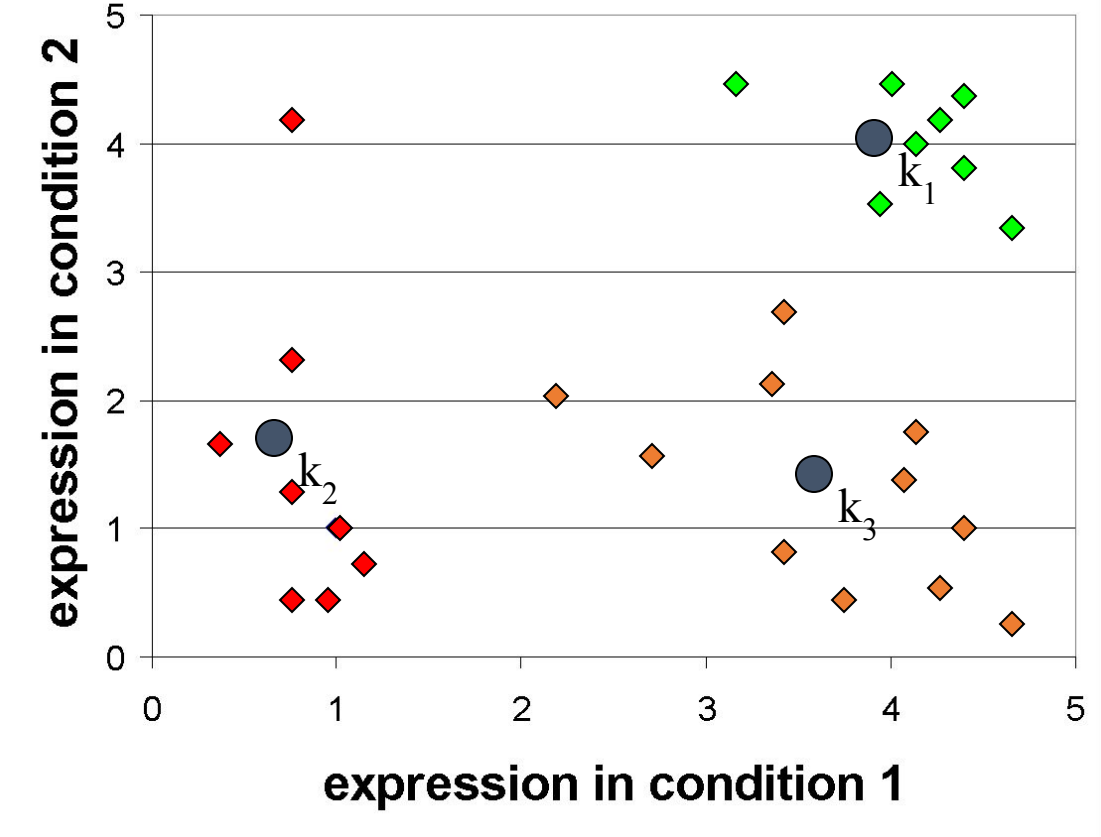
K-means Clustering: Step 4

Algorithm: k-means, Distance Metric: Euclidean Distance



K-means Clustering: Step 5

Algorithm: k-means, Distance Metric: Euclidean Distance



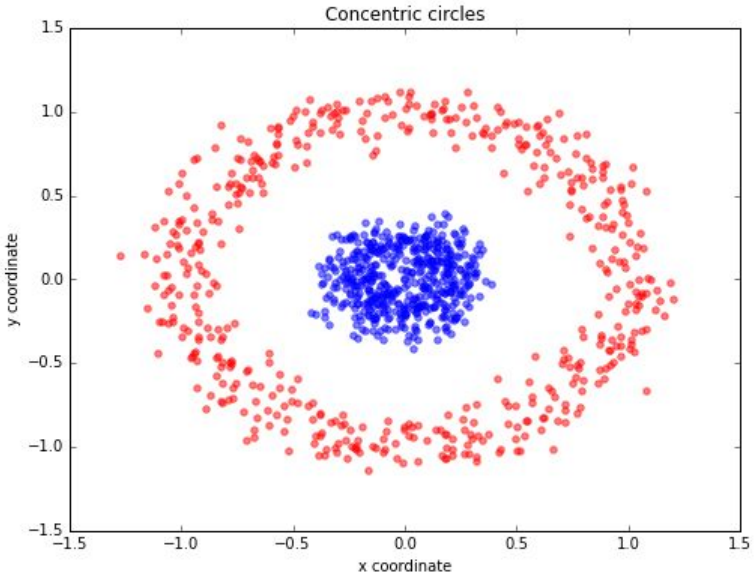
COMMENTS ON THE *K-MEANS* METHOD

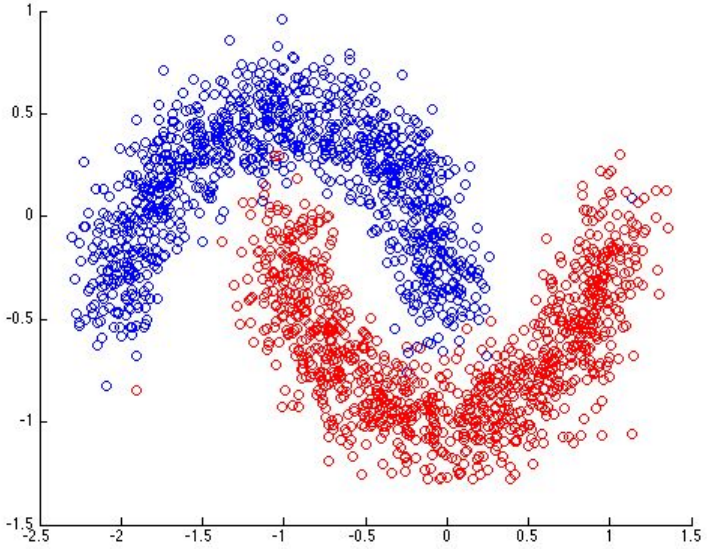
- Strength
  - *Relatively efficient*: *O*(*tkn*), where *n* is # objects, *k* is # clusters, and *t* is # iterations. Normally, *k*, *t* << *n*.
  - Often terminates at a *local optimum*. The *global optimum* may be found using techniques such as: *deterministic annealing* and *genetic algorithms*
- Weakness
  - Applicable only when *mean* is defined, then what about categorical data?
  - Need to specify *k,* the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*
  - Not suitable to discover clusters with *non-convex shapes*

THE *K-MEDOIDS* CLUSTERING METHOD

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets

WHAT IF THE DATA ARE NOT COMPACT?





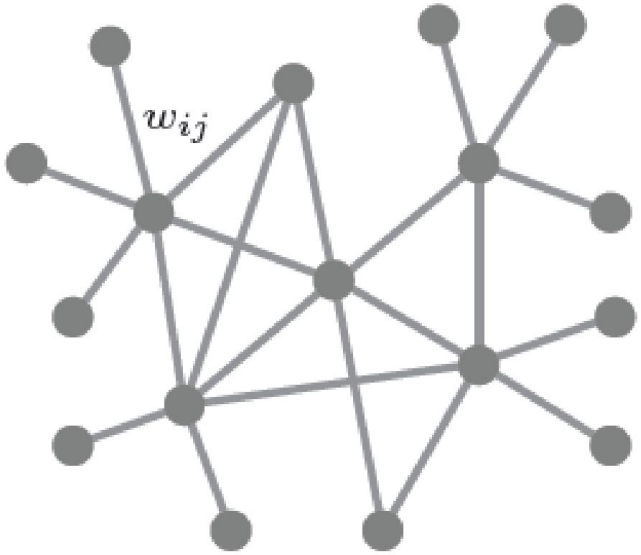
SPECTRAL CLUSTERING

- Algorithms that cluster points using eigenvectors of matrices derived from the data
- Obtain data representation in the low-dimensional space that can be easily clustered
- Variety of methods that use the eigenvectors differently
- Difficult to understand….

ELEMENTS OF GRAPH THEORY

- A ***graph*** $G = (V,E)$ consists of a *vertex set* V and an *edge set* E.
- If G is a ***directed graph***, each edge is an ordered pair of vertices
- A ***bipartite graph*** is one in which the vertices can be divided into two groups, so that all edges join vertices in different groups.

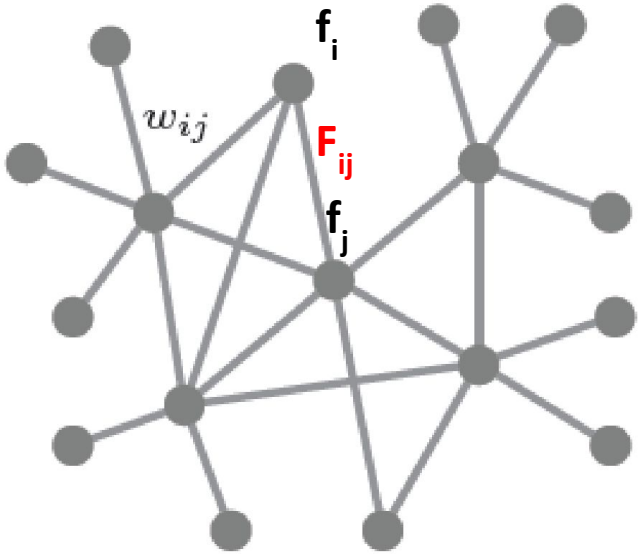
- **Vertex weights** $a_i > 0$ for $i \in V$
- **Edge weights** $w_{i,j} > 0$ for $i,j \in E$
- **Vertex fields L2(V) = {f:V->R}**
  vector of functions/coefficients **f=**$(f_1, f_2 \dots f_n)$



- **Hilbert space with inner product:**
- $< f, g >_{L2(V)} = \sum_{i\ in\ V} a_i f_i g_i$

GRAPH CALCULUS

- **Gradient**: L2(V)->L2(E)
- $(\nabla f)_{ij} = \sqrt{w_{ij}}(f_i - f_j)$



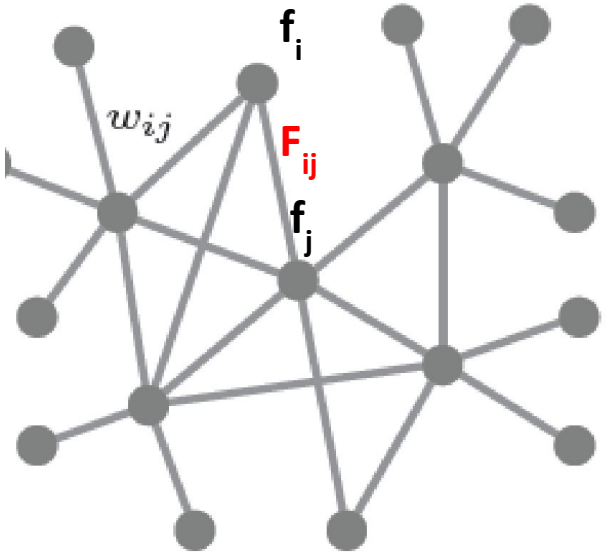
- Divergence L2(E)->L2(V)
- $div(F)_i = \frac{1}{a_i}\sum_{(i,j)\in E}\sqrt{w_{ij}}\,(F_{ij} - F_{ji})$

- Divergence and gradient are adjoint operators

$$\langle F, \nabla f\rangle_{L^2(\mathcal{E})} = \langle \nabla^* F, f\rangle_{L^2(\mathcal{V})} = \langle -\mathrm{div} F, f\rangle_{L^2(\mathcal{V})}$$

GRAPH CALCULUS

- **Gradient**: L2(V)->L2(E)
- $(\nabla f)_{ij} = \sqrt{w_{ij}}(f_i - f_j)$



- Divergence L2(E)->L2(V)
- $div(F)_i = \frac{1}{a_i}\sum_{(i,j)\in E}\sqrt{w_{ij}}\,(F_{ij} - F_{ji})$

- Divergence and gradient are adjoint operators

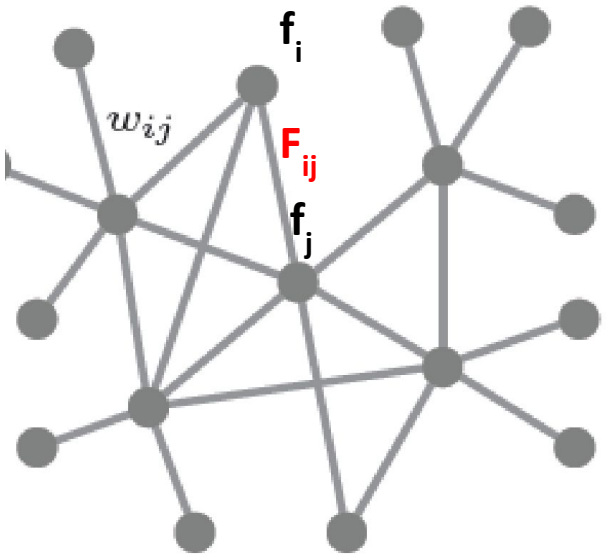
$$\langle F, \nabla f\rangle_{L^2(\mathcal{E})} = \langle \nabla^* F, f\rangle_{L^2(\mathcal{V})} = \langle -\mathrm{div} F, f\rangle_{L^2(\mathcal{V})}$$

- **Laplacian** operator L2(V)->L2(V)

$$(\Delta f)_i = \frac{1}{a_i}\sum_{j:(i,j)\in\mathcal{E}} w_{ij}(f_i - f_j)$$

LAPLACIANS

- The **laplacian** is the analogous of II order derivatives



- Represented as matrix algebra positive definite matrix L of size nxn

- n is the number of vertices

$$\Delta = \mathbf{A}^{-1}(\mathbf{D} - \mathbf{W})$$

W=(w_{ij}) A= ($\frac{1}{a_i}$) D=diag($\sum_{i \neq j} w_{ij}$)

- W adiacency matrix
  D degree matrix

- Unormalized Laplacian A=I
- Random Walk Laplacian A=D

LAPLACIANS

- The **laplacian L** admits n eigenvalues and eigenvectors

$$\Delta\phi_k = \lambda_k\phi_k, \qquad k = 1, 2, \ldots$$

- **Eigenvectors** are real and orthonormal (self adjointness)

- **Eigenvalues are real and non-negative** (positive semidefinite matrix)

ENERGY OF A GRAPH

- Dirichlet energy -> measure the smoothness of a function

- On a graph given a function f of vertices

$$E_{\mathrm{Dir}}(f) \quad = \quad \langle \nabla f, \nabla f \rangle_{L^2(\mathcal{E})} \quad = \quad \langle \Delta f, f \rangle_{L^2(\mathcal{V})}$$

In matrix vector notation

$$E_{\mathrm{Dir}}(\mathbf{f}) \quad = \quad \mathbf{f}^\top \mathbf{\Delta} \mathbf{f}$$

- Derivation make use of
- $(\nabla f)_{ij} = \sqrt{w_{ij}}(f_i - f_j)$

SMOOTHEST ORTHOBASIS OF A GRAPH

- Find a set of orthonormal vectors such that E_Dir is minimum

$$\min_{\phi_1} E_{\text{Dir}}(\phi_1) \quad \text{s.t.} \quad \|\phi_1\| = 1$$

$$\min_{\phi_k} E_{\text{Dir}}(\phi_k) \quad \text{s.t.} \quad \|\phi_k\| = 1, \quad k = 2, 3, \ldots n$$

$$\phi_k \perp \text{span}\{\phi_1, \ldots, \phi_{k-1}\}$$

What does it mean?

- It means find the set of vertices of the graph corresponding to a minimum in the smoothness function

- That is -> **set of vertices that in terms of edge weights are close together**
- Order of the vectors is proportional to the closeness

Solution

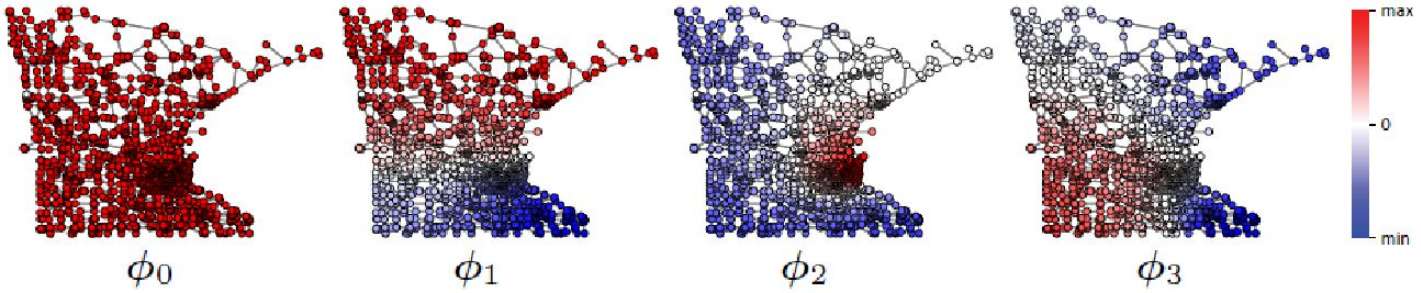
$$\min_{\mathbf{\Phi} \in \mathbb{R}^{n \times n}} \underbrace{\text{trace}(\mathbf{\Phi}^\top \mathbf{\Delta} \mathbf{\Phi})}_{\|\mathbf{\Phi}\|_{\mathcal{G}} \text{ Dirichlet norm}} \quad \text{s.t.} \quad \mathbf{\Phi}^\top \mathbf{\Phi} = \mathbf{I}$$

SMOOTHEST ORTHOBASIS OF A GRAPH

- Solutions are the first n **eigenvectors** of graph Lapalcian

$$\min_{\mathbf{\Phi}\in\mathbb{R}^{n\times n}} \underbrace{\operatorname{trace}(\mathbf{\Phi}^\top \mathbf{\Delta}\mathbf{\Phi})}_{\|\mathbf{\Phi}\|_{\mathcal{G}}\ \text{Dirichlet norm}} \quad \text{s.t.} \quad \mathbf{\Phi}^\top\mathbf{\Phi} = \mathbf{I}$$

- The energy value is the sum of eigenvalues



First Laplacian eigenvectors of a graph

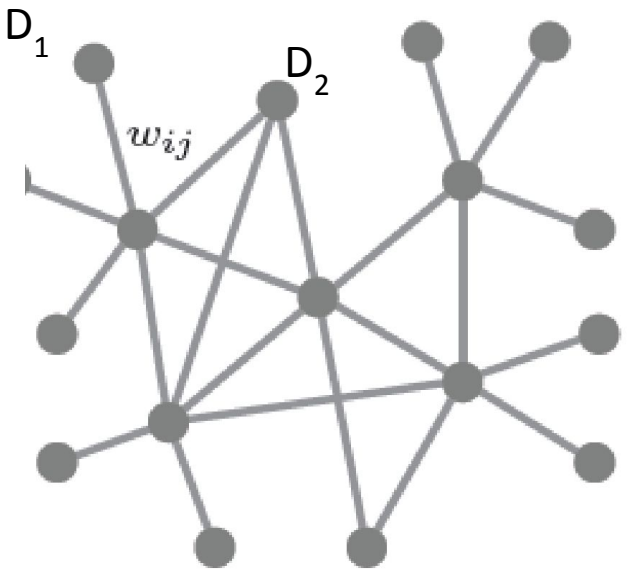
They are related to graph geometry (s.a. communities, hubs, etc).

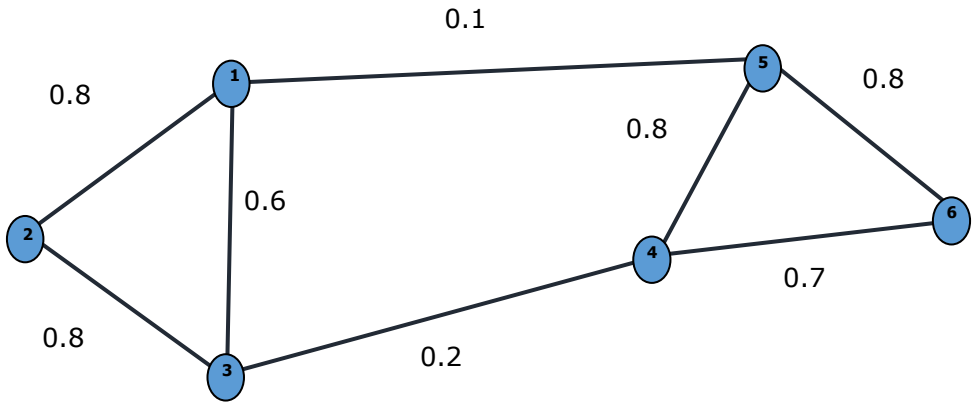
HOW TO CLUSTER USING GRAPHS

- Similarity Graph

- Given a Dataset D of N elements in R^d
- Points are graph vertices V={1….N }

- Edges are similarities between points $w_{ij} = f(V_i, V_j)$
- With f chosen similarity function

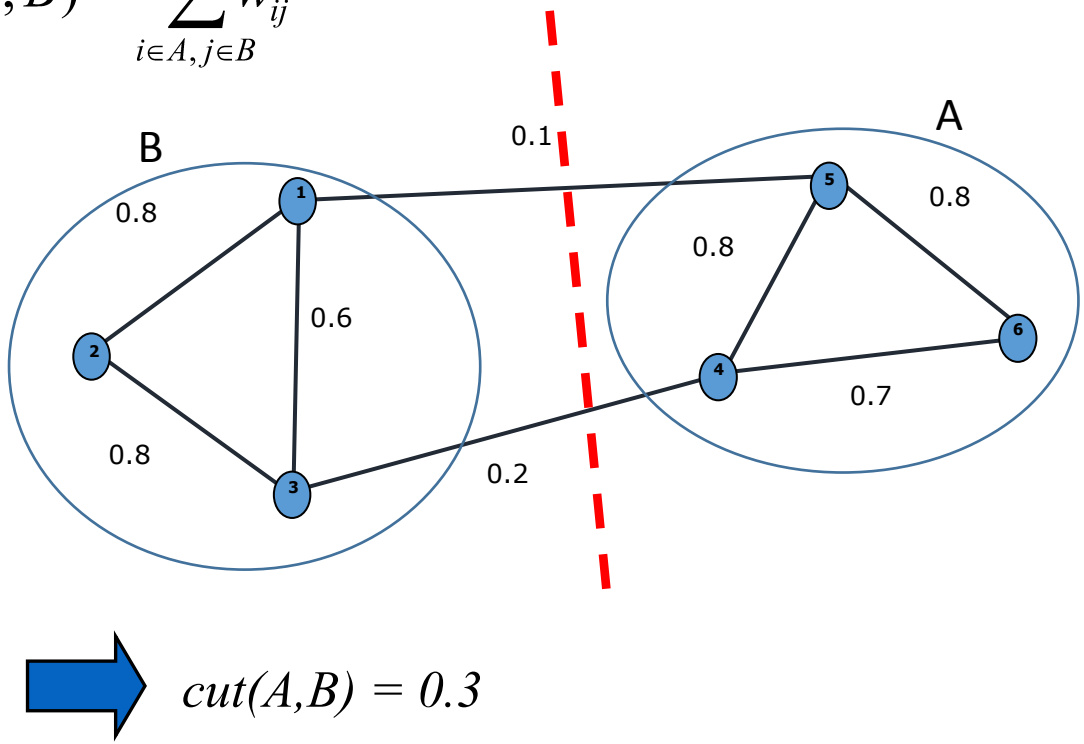




GRAPH PARTITIONING

- **BI-Partition of a graph**
- Obtain two set of vertices that are disjoint
  - Remove all edges that connect the sets (CUT)

$$cut(A,B) = \sum_{i \in A, j \in B} w_{ij}$$



$cut(A,B) = 0.3$

CLUSTERING OBJECTIVES

- Traditional definition of a “good” clustering:
  - Points assigned to same cluster should be highly similar.
  - Points assigned to different clusters should be highly dissimilar.

2 Objective Functions:

1. **Minimize the cut value -> Minimum cut**

$$\min cut(A,B)$$

2. **Find the minimum cut balancing the number of items per cluster -> Normalized cut (Shi Malik 1997)**

$$\min Ncut(A,B) = \frac{cut(A,B)}{vol(A)} + \frac{cut(A,B)}{vol(B)}$$

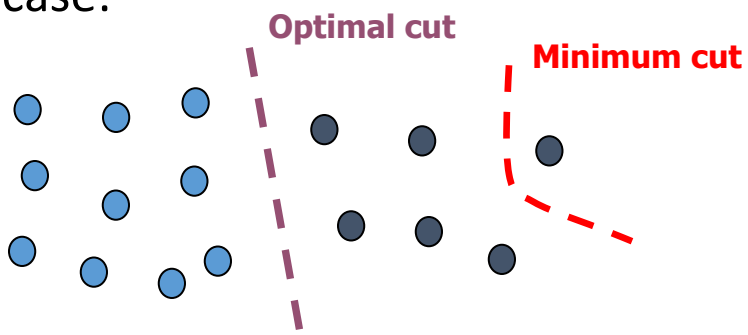
- Where vol(A) is the sum of edges inside set A

GRAPH CUT CRITERIA

- **Criterion: Minimum-cut**
  - Minimise weight of connections between groups

$$\min cut(A,B)$$

Degenerate case:



Problem:

- Only considers external cluster connections
- Does not consider internal cluster density

FIND AN OPTIMAL MIN-CUT (HALL'70, FIEDLER'73)

- Objective MIN CUT can be formulated in matrix notation
- Express a bi-partition (A,B) as a vector **p**

$$p = \begin{Bmatrix} +1 \; iff \; D_i \in A \\ -1 \; iff \; D_i \in B \end{Bmatrix}$$

$$\text{mincut}(A,B) = \text{argmin}_p \sum_V w_{ij}(p_i - p_j)^2$$

Using $f'Lf = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$

$$\text{mincut}(A,B) = \text{argmin}_p p'L\,p$$

Proof:

$$f'Lf = f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}$$

$$= \frac{1}{2}\left(\sum_{i=1}^{n} d_i f_i^2 - 2\sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2\right) = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$

FIND AN OPTIMAL MIN-CUT 2 (RAYLEIGH THEOREM REVISITED)

- solution $\text{mincut}(A, B) = \text{argmin}_p p' L\, p$ is NP HARD cannot be solved

p is discrete

- We relax p as a continuous vector
- $p \in \mathcal{R}$
- Then the mincut objective becomes $p' L\, p$ but calling p as f

- *Mincut(A,B) is the same as minimizing the Dirichlet Energy on the graph*

- Solution $mincut(A, B) = \underset{f}{argmin}\, f' L f = argmin_f E_{dir}(f)$
- Cut value is the energy value $mincut(A, B) = \min_f E_{dir}(f)$

- Solutions are :

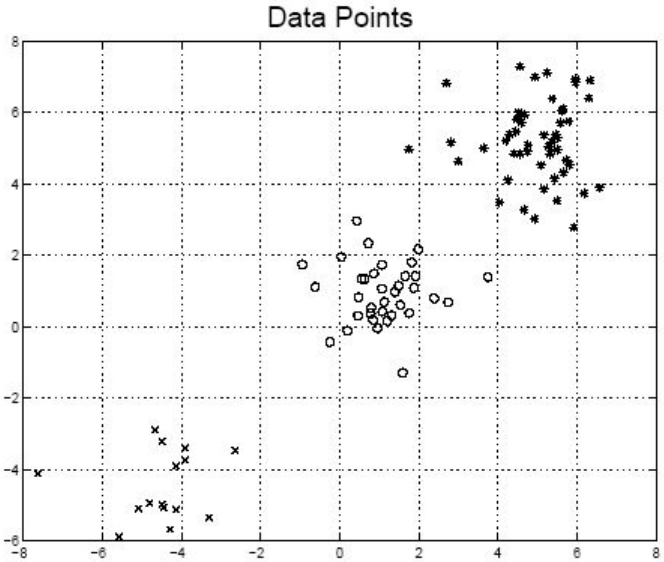
1. f is the **2 nd smallest eigenvector** Fiedler Vector
2. **Mincut** value is the **2 nd smallest eigenvalue**

SPECTRAL CLUSTERING PROCEDURE

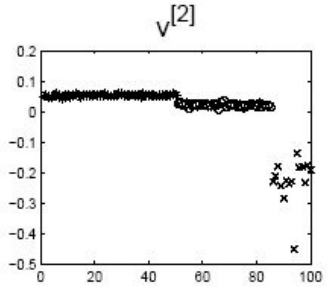
- Three basic stages:

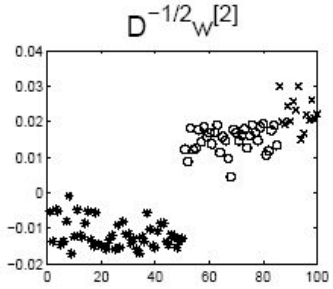
1. **Pre-processing**
   - Construct a matrix representation of the dataset.
2. **Decomposition**
   - Compute eigenvalues and eigenvectors of the matrix.
   - Map each point to a lower-dimensional representation based on one or more eigenvectors.
3. **Grouping**
   - Assign points to two or more clusters, based on the new representation.

3-CLUSTERS

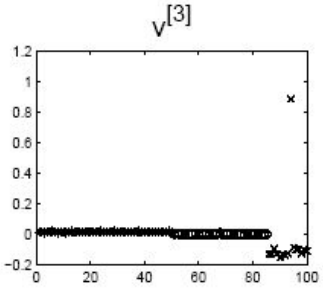


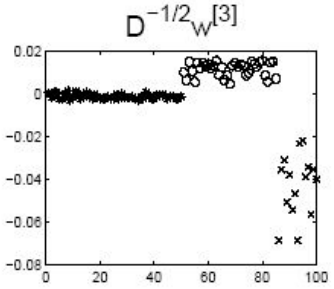
If we use the 2nd eigen vector





If we use the 3rd eigen vector





K-WAY SPECTRAL CLUSTERING

How do we partition a graph into k clusters?

- Two basic approaches:
  - **Recursive bi-partitioning** (Hagen et al.,'91)
    Recursively apply bi-partitioning algorithm in a hierarchical divisive manner.

    Disadvantages: Inefficient, unstable

  - **Cluster multiple eigenvectors** (Shi & Malik,'00)
    Build a reduced space from multiple eigenvectors.
    Commonly used in recent papers
    A preferable approach…but its like to do PCA and then k-means

CLUSTER MULTIPLE EIGENVECTORS

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

K-eigenvector Algorithm (Ng et al.,'01)

1. **Pre-processing**
   Construct the scaled adjacency matrix

2. **Decomposition**
   Find the eigenvalues and eigenvectors of L.

   Build embedded space from the eigenvectors corresponding to the k smallest eigenvalues

3. **Grouping**
   Apply K-Means to reduced n x k space to produce k clusters.

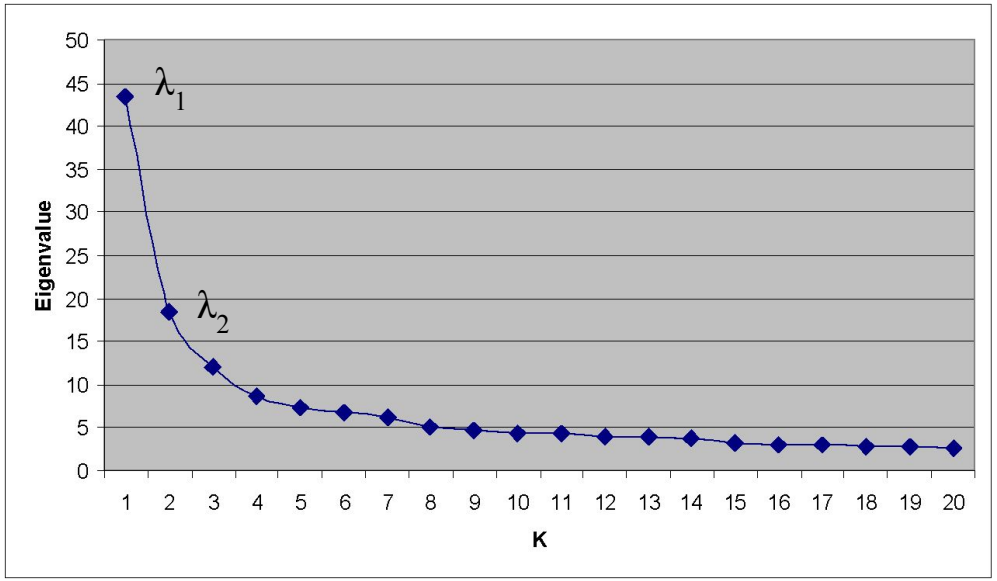
HOW TO SELECT K SPECTRAL GAP HEURISTIC

- Eigengap: the difference between two consecutive eigenvalues.

- Most stable clustering is generally given by the value k that maximises the expression

$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$

Largest eigenvalues of Cisi/Medline data



CONCLUSION

- Clustering as a graph partitioning problem
  - Quality of a partition can be determined using graph cut criteria.
  - Identifying an optimal partition is NP-hard.

- Spectral clustering techniques
  - Efficient approach to calculate near-optimal bi-partitions and k-way partitions.
  - Based on well-known cut criteria and strong theoretical background.

- ***Read: "A Tutorial on Spectral Clustering" V.Luxburg Aug 2006***