Collaborative filtering based on a random walk model on a graph

Marco Saerens, Francois Fouss, Alain Pirotte, Luh Yen, Pierre Dupont (UCL)

Jean-Michel Renders (Xerox Research Europe)

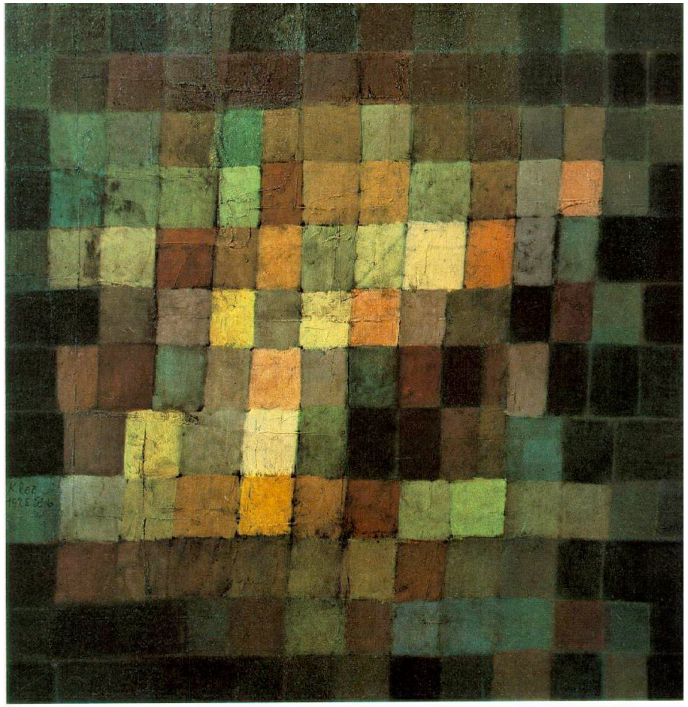
machine learning group @ UCL

UCL Université catholique de Louvain

Université partenaire de l'Académie universitaire 'Louvain'

Some recent methods: web link analysis

Exploiting links between documents



Web link analysis

- Suppose we performed a web search with a search engine
- **Objective**:
  - To improve the (content-based) ranking of the search engine
  - Based on the graph structure of the web hyperlinks

Web link analysis

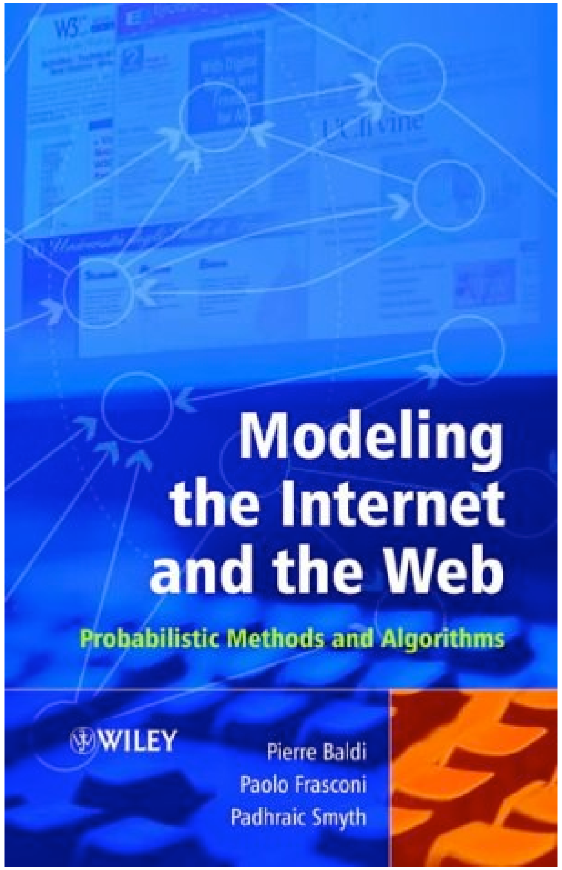
- The objective here is to exploit the links between documents (hyperlinks)
- Documents/hyperlinks can be viewed as a directed graph
- Two algorithms will be introduced:
  - PageRank
  - HITS
- Then, we will introduce a more general algorithm for collaborative recommendation

Web link analysis

- A set of techniques
  - Applied to: Hyperlink document repositories
  - Typically web pages
- Objective:
  - To exploit the link structure of the documents
  - In order to extract interesting information
  - Viewing the document repository as a graph where
    - Nodes are documents
    - Edges are directed links between documents

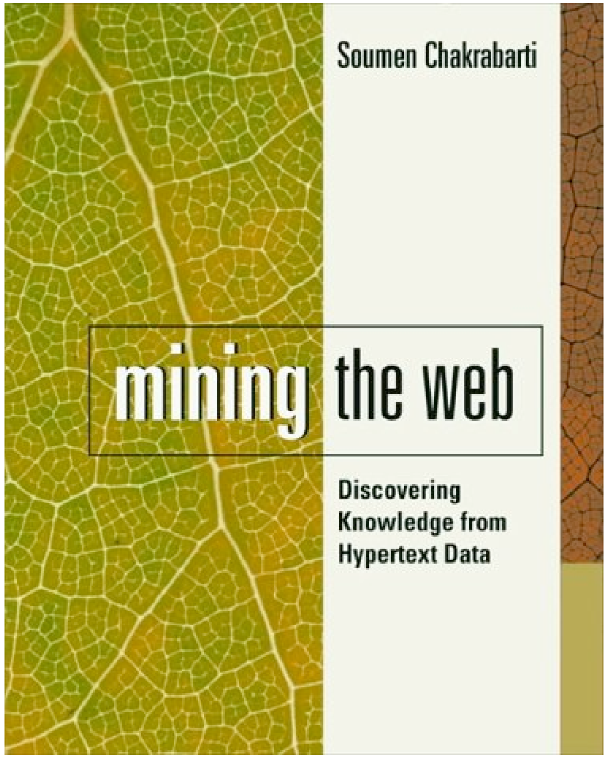
(1) The PageRank algorithm

- P. Baldi, P. Frasconi & P. Smyth (2003)
  - Modeling the Internet and the Web
  - J. Wiley & Sons



The PageRank algorithm

- S. Chakrabarti (2003)
  - Mining the Web
  - Morgan Kaufmann

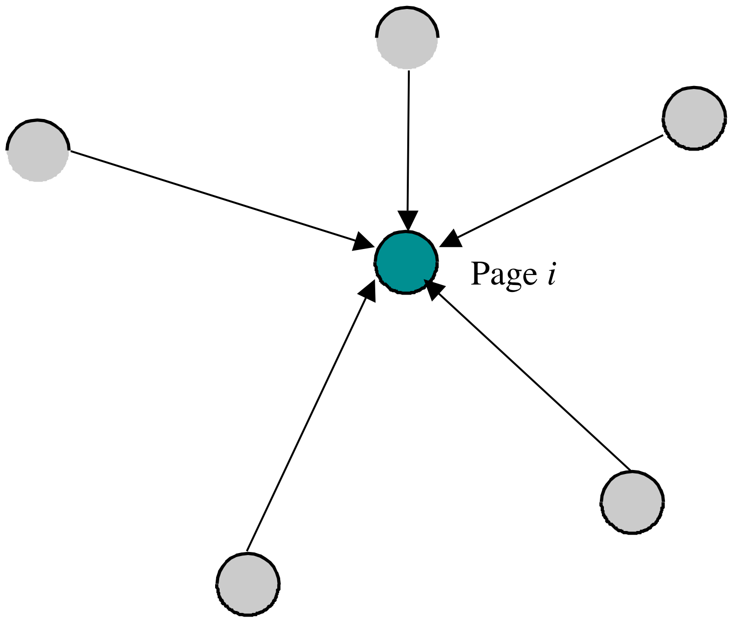


The PageRank algorithm

- Introduced by Page, Brin, Motwani & Winograd in 1998
- Partly implemented in Google

The PageRank algorithm

- To each web page we associate a score, x_i
  - The score of page i, x_i, is proportional to the weighted averaged score of the pages pointing to page i



The PageRank algorithm

- Let w_{ij} be the weight of the link connecting page i to page j
  - Usually, it is simply 0 or 1
  - Thus, $w_{ij} = 1$ if page i has a link to page j; $w_{ij} = 0$ otherwise

The PageRank algorithm

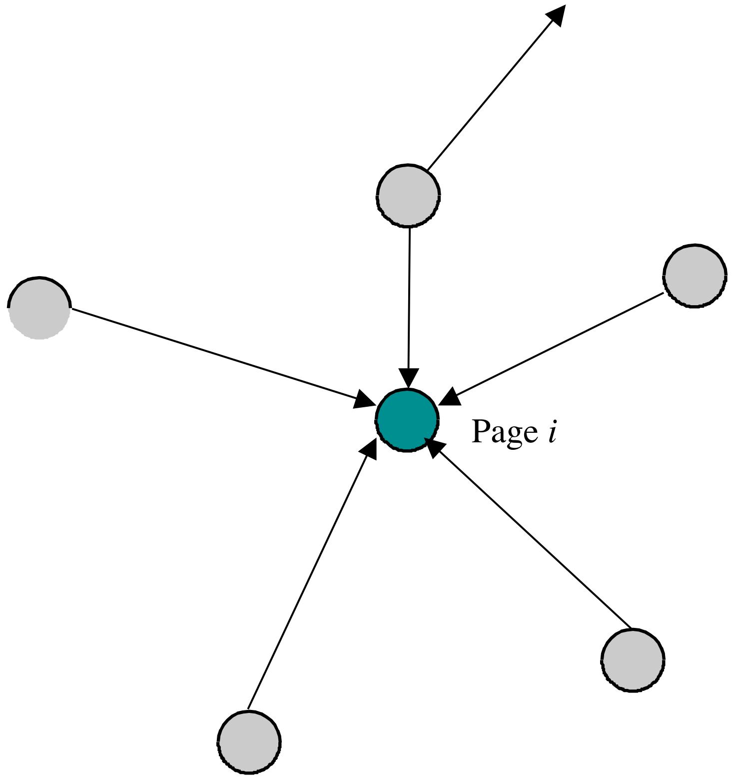
- Let $\mathbf{W}$ be the adjacency matrix made of the elements w_{ij}
  - Notice that this matrix is not symmetric
  - We suppose that the graph is strongly connected

The PageRank algorithm

- In other words

$$x_i \propto \sum_{j=1}^{n} \frac{w_{ji}\, x_j}{w_{j.}}$$

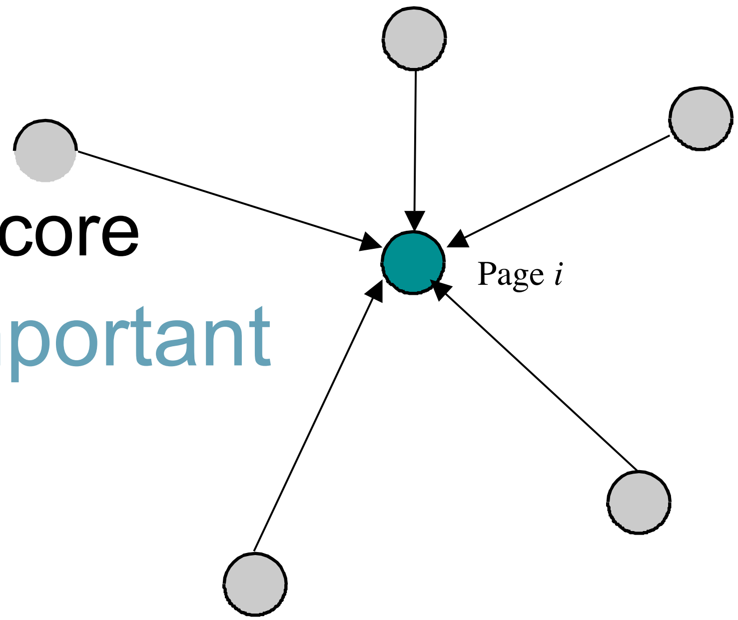
$$w_{j.} = \sum_{i=1}^{n} w_{ji}$$



– where $w_{j.}$ is the outdegree of page j

The PageRank algorithm

- In other words,
- A page with a high score is a page that is pointed by
  - Many pages
  - Having each a high score
- Thus a page is an important page if
  - It is pointed by many, important, pages



The PageRank algorithm

- These equations can be updated iterately until convergence
- In order to obtain the scores, x_i
  - We normalize the vector $\mathbf{x}$ at each iteration
- The pages are then ranked according to their score

The PageRank algorithm

- This definition has a nice interpretation in terms of random surfing
- If we define the probability of following the link between page i to page j as

$$\mathrm{P}(page(k+1)=i|page(k)=j)=\frac{w_{ji}}{w_{j.}}$$

$$w_{j.}=\sum_{i=1}^{n} w_{ji}$$

The PageRank algorithm

- We can write the updating equation as

$$x_i(k+1) = \mathrm{P}(page(k+1) = i)$$
$$= \sum_{j=1}^{n} \mathrm{P}(page(k+1) = i | page(k) = j)\, x_j(k)$$

- And thus we can define a random surfer following the links according to the transition probabilities

$$\mathrm{P}(page(k+1) = i | page(k) = j) = \frac{w_{ji}}{w_{j.}}$$

The PageRank algorithm

- This is the equation of a Markov model of random surf through the web
- This is exactly the same equation as before:

$$x_i \propto \sum_{j=1}^{n} \frac{w_{ji}\, x_j}{w_{j\cdot}}$$

$$w_{j\cdot} = \sum_{i=1}^{n} w_{ji}$$

The PageRank algorithm

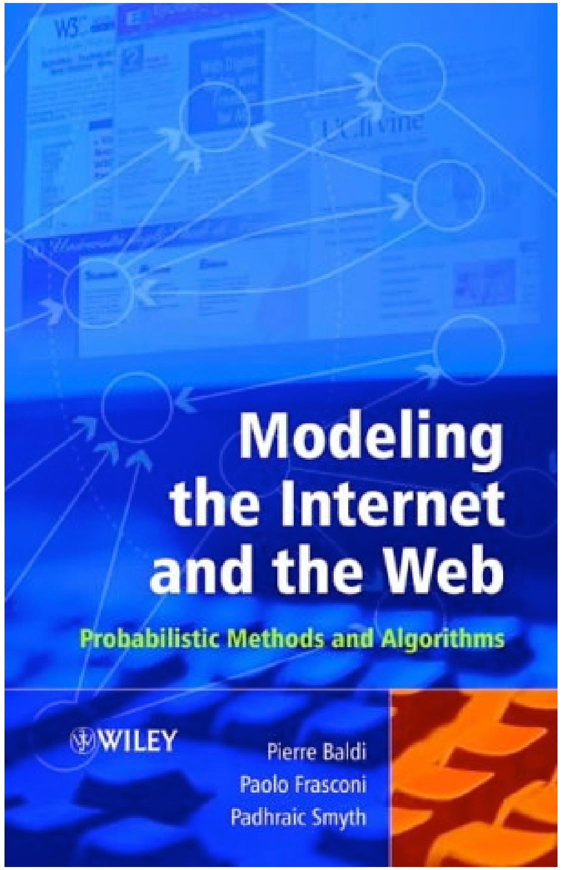
- x_i can then be viewed as the probability of being at page i
  - One can show that the solution to these equations is the stationary distribution of the random surf
  - Which is the probability of finding the surfer on page i on the long-term behaviour
- The most probable page is the best ranked

The PageRank algorithm

- One can show that the scores can also be obtained
  - By computing the principal left eigenvector of the matrix $\mathbf{P}$,
  - The probability transition matrix of the Markov process
  - Containing the transition probabilities

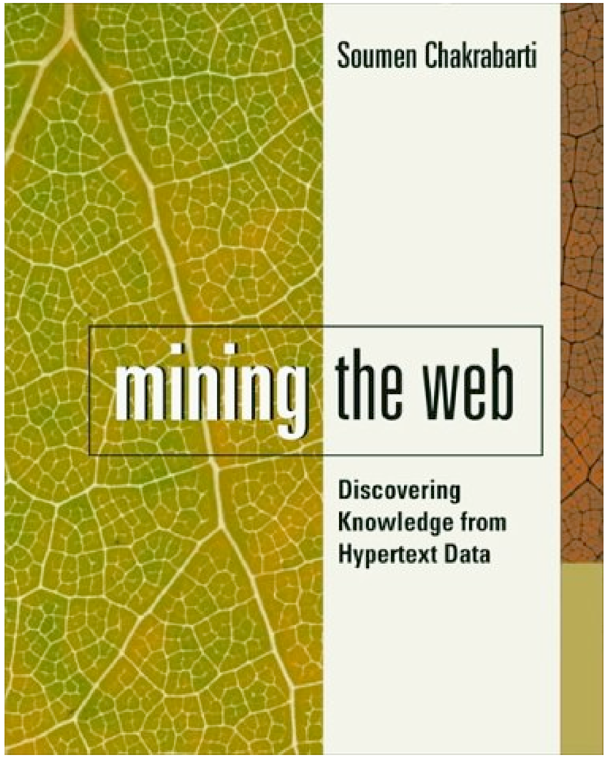
(2) The HITS algorithm

- P. Baldi, P. Frasconi & P. Smyth (2003)
  - Modeling the Internet and the Web
  - John Wiley & Sons



The HITS algorithm

- S. Chakrabarti (2003)
  - – Mining the web
  - – Morgan Kaufmann



The HITS algorithm

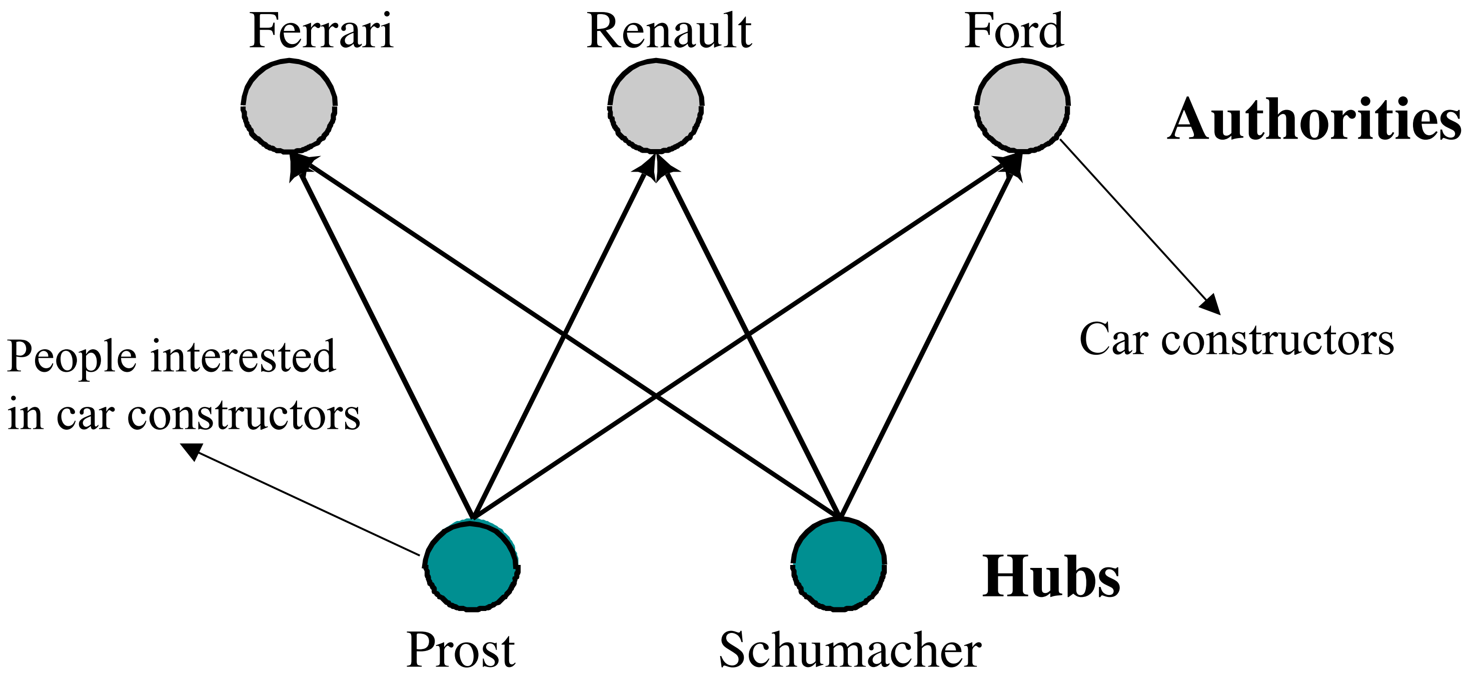
- Introduced by Kleinberg in 1998/1999

The HITS algorithm

- The model proposed by Kleinberg is based on two concepts
  - **Hub** pages
  - **Authorities** pages
- We thus assume that these are two categories of web pages
- These two concepts are strongly connected

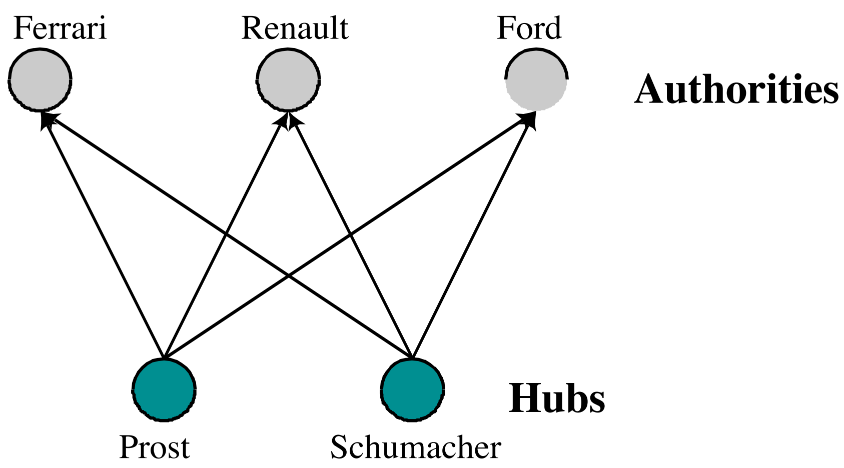
The HITS algorithm

- Example:
  - Suppose we introduced the query "Car constructors"



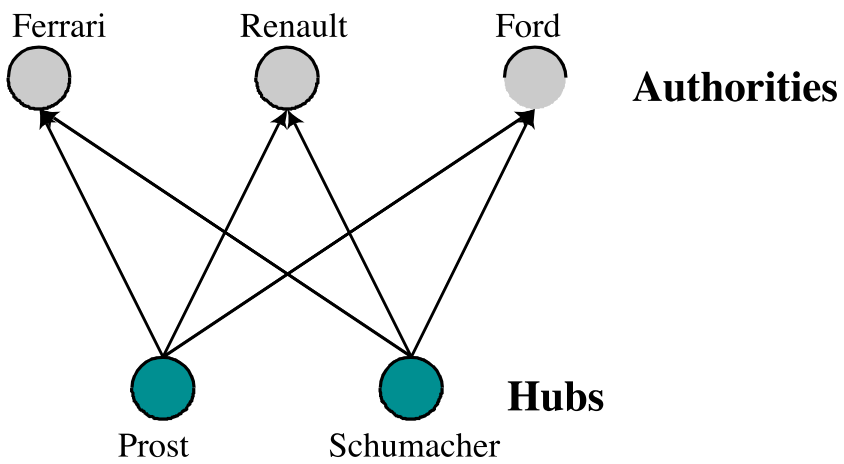
The HITS algorithm

- Hubs
  - Link heavily to authorities
  - A good hub points to many good authorities
  - Hubs have very few incoming links



The HITS algorithm

- Authorities
  - Do not link to other authorities
  - A good authority is pointed by many good hubs
  - The main authorities on a topic are often in competition with one another



The HITS algorithm

- The objective is to detect **good hubs** and **good authorities**
  - from the results of the search engine
- We therefore assign two numbers to each returned page i:
  - A **hub score,** x^h_i
  - An **authority score,** x^a_i

The HITS algorithm

- Let w_{ij} be the weight of the link connecting page i to page j
  – Usually, it is simply 0 or 1
  – Thus, $w_{ij} = 1$ if page i has a link to page j; $w_{ij} = 0$ otherwise
- Let $\mathbf{W}$ be the matrix made of elements w_{ij}
  – Notice that this matrix is not symmetric
  – We suppose that the graph is strongly connected

The HITS algorithm

- A possible procedure for computing hub/authorities scores (Kleinberg)
  - A page's **authority score** is proportional to the sum of the hub scores that link to it

$$x_j^a = \eta \sum_{i=1}^{n} w_{ij} x_i^h$$

  - A page's **hub score** is proportional to the sum of the authority scores that it links to

$$x_i^h = \mu \sum_{j=1}^{n} w_{ij} x_j^a$$

The HITS algorithm

- Kleinberg used this iterative procedure in order to estimate the scores (with a normalization)
  - He showed that this is equivalent to computing the eigenvectors of the following matrices

$$\mathbf{W}\mathbf{W}^{\mathrm{T}} \qquad\qquad \mathbf{W}^{\mathrm{T}}\mathbf{W}$$

  - To obtain respectively the vector of hubs scores and the vector of authorities scores

- We showed that this is exactly uncentered principal components analysis (PCA)

The HITS algorithm

- We further showed that this procedure is also related to both
  - Correspondence analysis
  - A random walk model through the graph

The multivariate analysis of undirected graphs

Context

- Main purpose: To exploit the graph structure of large repositories
  - Web environment
  - Digital documents repositories
  - Databases with metadata

- We will focus on databases and collaborative recommendation

Main goal

- To exploit and analyse
  - New similarity measures between the nodes of a graph
- To use these similarities for
  - Collaborative filtering
  - Clustering
  - Graph visualization
  - Etc…

Main point

- These similarity measures between two nodes not only depend on
  - The weights of the edges (like the « shortest path » distance)
- But also on
  - The number of paths connecting the two edges
- They take high connectivity into account

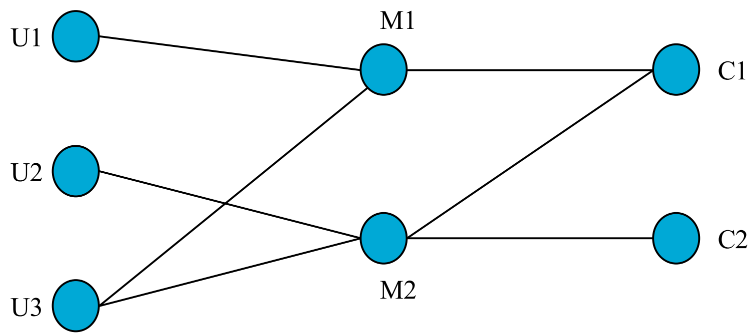
  ≠ shortest-path or geodesic (Dijkstra) distance

Graph: The adjacency matrix

- The elements a_{ij} of the adjacency matrix $\mathbf{A}$ of a weighted, undirected, graph are defined as

$$a_{ij} = \begin{cases} w_{ij} \text{ if node } i \text{ is connected to node } j \\ 0 \text{ otherwise} \end{cases}$$

  where $\mathbf{A}$ is symmetric
- The $w_{ij} \geq 0$ represent the strength of relationship between node i and node j



$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Graph: The Laplacian matrix

- The Laplacian matrix $\mathbf{L}$ of the graph is defined by

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

where $\mathbf{D} = \mathrm{diag}(a_{i.})$ with $d_{ii} = [\mathbf{D}]_{ii} = a_{i.} = \sum_{j=1}^{n} a_{ij}$
(the outdegree of each node)

- $\mathbf{L}$ is doubly centered
- If the graph is connected, the rank of $\mathbf{L}$ is $n - 1$, where n is the number of nodes
- $\mathbf{L}$ is symmetric
- $\mathbf{L}$ is positive semidefinite

A random walk model on the graph

- As for PageRank, every node is associated to a state of a Markov chain
- The random walk is defined by the single-step transition probabilities

$$\mathrm{P}(s(t+1)=j \mid s(t)=i)=p_{ij}=\frac{a_{ij}}{a_{i.}}$$

where

$$a_{i.}=\sum_{j=1}^{n} a_{ij}$$

- In other words, to any state or node i, we associate a probability of jumping to an adjacent node, $s(t+1)=j$
  - which is proportional to the weight w_{ij} of the edge connecting i and j

Two main quantities

- We then compute two main quantities from this Markov chain:
  - The average first passage time
  - The average commute time

Average first-passage time

- $m(k|i)$ = average number of steps a random walker, starting in state i, will take to enter state k for the first time

$$\begin{cases} m(k|i) = 1 + \sum\limits_{\substack{j=1 \\ j \neq k}}^{n} p_{ij}\, m(k|j), \text{ for } i \neq k \\ m(k|k) = 0 \end{cases}$$

- These equations can be used in order to iteratively compute the first-passage times.

Average commute time

- $n(i,j) = m(j|i) + m(i|j)$
  = average number of steps a random walker, starting in state $i \neq j$, will take before entering a given state j for the first time, and go back to i

- Note: while $n(i,j)$ is symmetric by definition, $m(i|j)$ is not.

Computation of the basic quantities by means of $\mathbf{L}^+$

- If we further define $\mathbf{e}_i$ as the ith column of $\mathbf{I}$

$$\mathbf{e}_i = [\underset{1}{0}, \ldots, \underset{i-1}{0}, \underset{i}{1}, \underset{i+1}{0}, \ldots, \underset{n}{0}]^{\mathrm{T}}$$

- we obtain the remarkable form

$$\boxed{n(i,j) = 2N_e(\mathbf{e}_i - \mathbf{e}_j)^{\mathrm{T}}\mathbf{L}^+(\mathbf{e}_i - \mathbf{e}_j)}$$

where each node i is represented by a unit basis vector, $\mathbf{e}_i$, in the node space

- $\mathbf{L}^+$ is the Moore-Penrose pseudoinverse of the Laplacian matrix of the graph

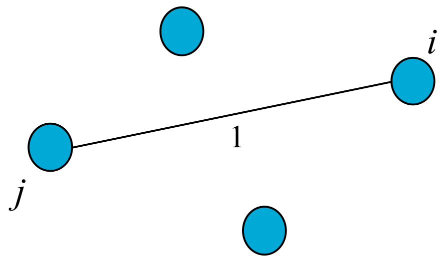
Computation of the basic quantities by means of $\mathbf{L}^+$

- Thus, $n(i,j)$ is a Mahalanobis distance
  = Commute Time Distance

- Indeed, one can show that $\mathbf{L}^+$ is
  - (1) Symmetric
  - (2) Positive semidefinite
  - (3) Doubly centered

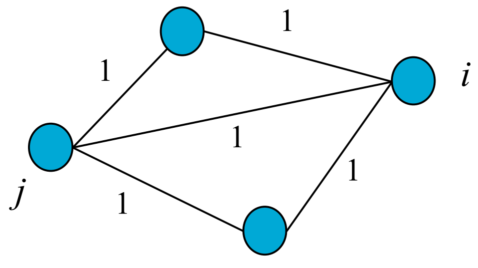
Commute time distance

- It takes high connectivity into account:



$n(i,j)$ *distance* $= Cst \cdot 1.0$

Shortest_path $= 1$



$n(i,j)$ *distance* $= Cst \cdot 0.5$

Shortest_path $= 1$

Embedding in an Euclidean space

- The node vectors can be mapped into an Euclidean space preserving the commute time distance
  - In this space, the node vectors are exactly separated by commute time distances
- The node vectors form a cloud of points, each point being a node
- So that any multivariate statistical analysis tool can be applied to analyse the graph

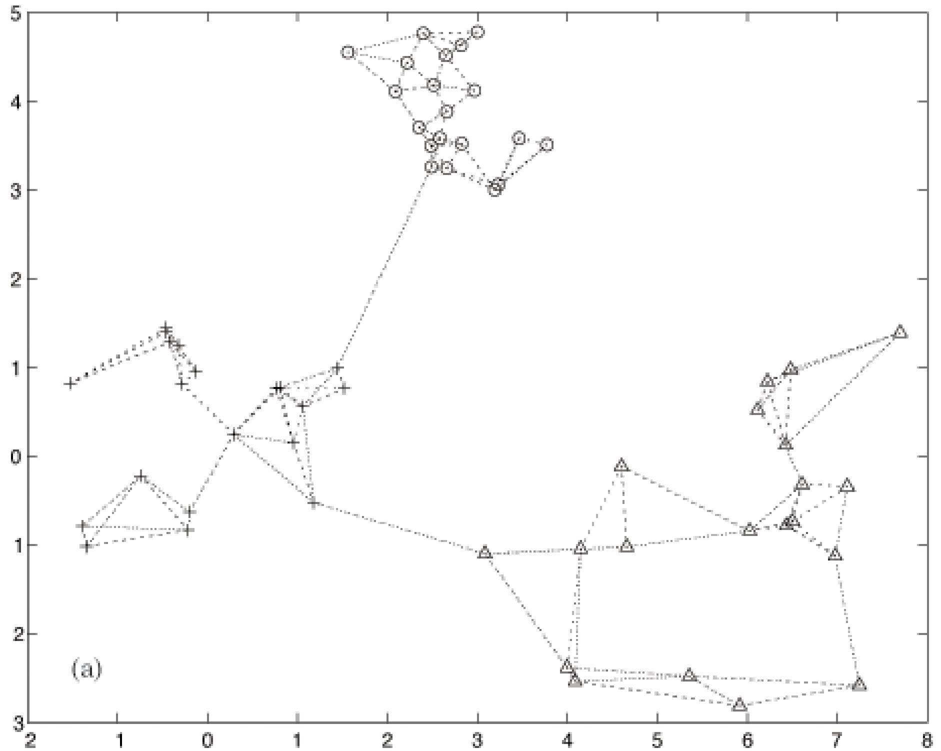
Embedding in an Euclidean space

- For instance:
  - Clustering of the nodes
  - Finding dense regions in the graph
  - Finding outlier nodes
  - Representing the graph in a low-dimensional space (principal components analysis)
  - Representing the graph in function of the similarity with some reference nodes (discriminant analysis)
  - Finding central nodes in the graph
  - Find the most similar node (nearest neighbour)
  - Etc…

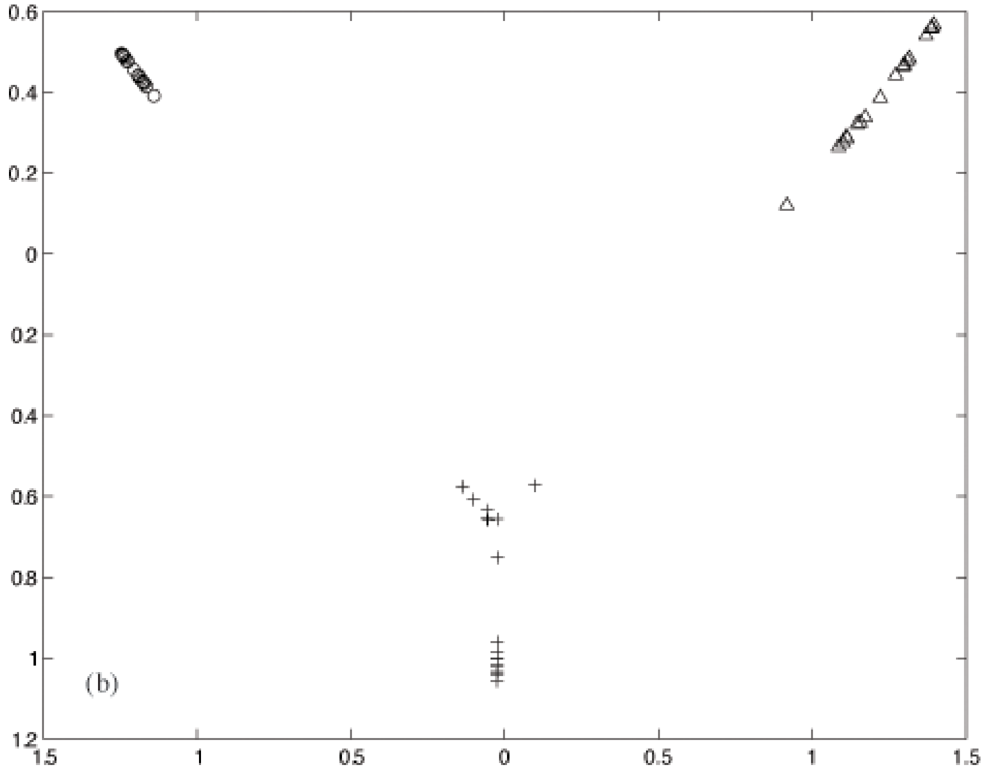
Maximum variance subspace projection of the nodes vectors

- This decomposition is similar to principal component analysis (PCA)
  - The projected node vectors has maximal variance
  - In terms of Euclidean commute time distance
  - Among all the possible candidate projections
- It allows us to visualize the graph

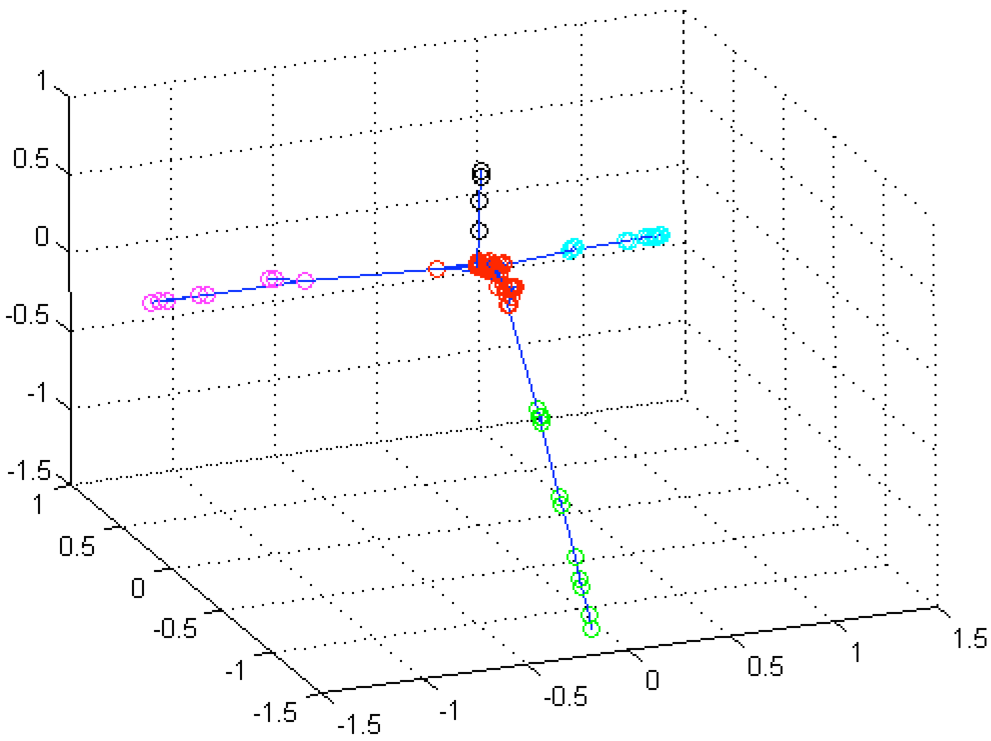
An example of PCA: Original graph



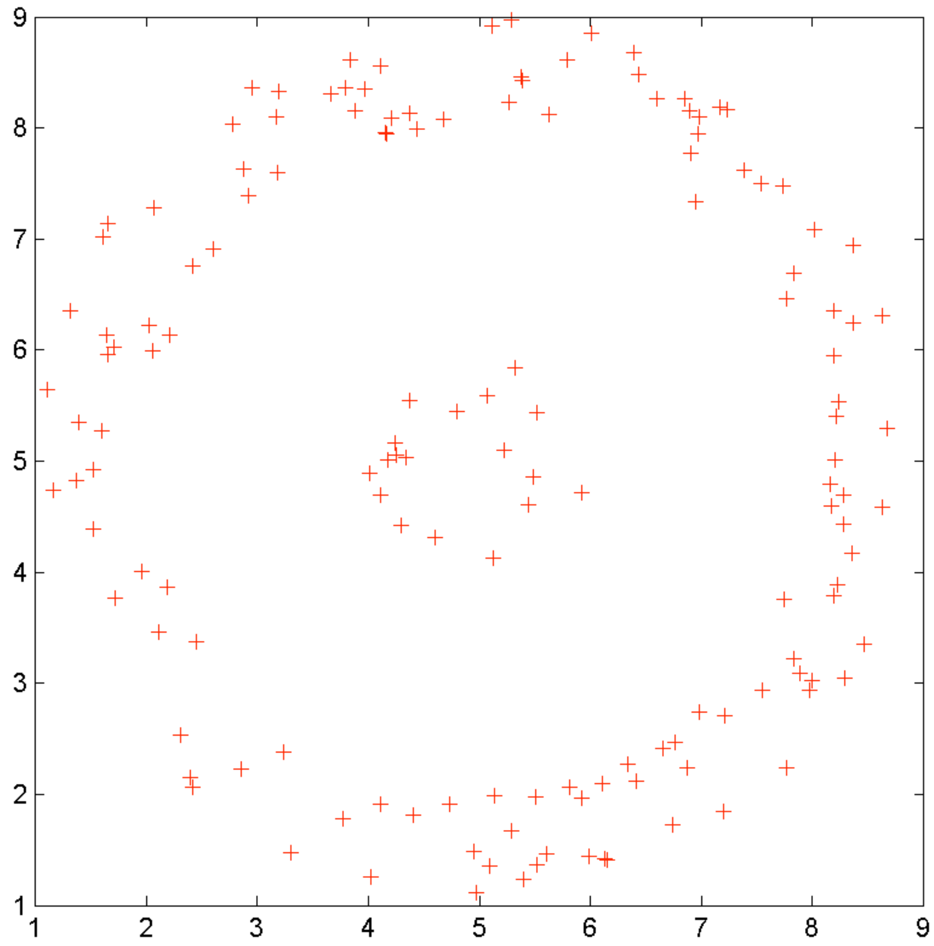
PCA: Projection of the nodes on the two first axis



PCA: Application to the visualization of a network of criminals

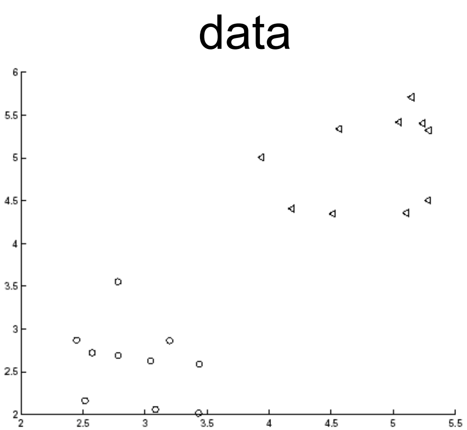


Another example : Application to clustering



Clustering with ECT distance

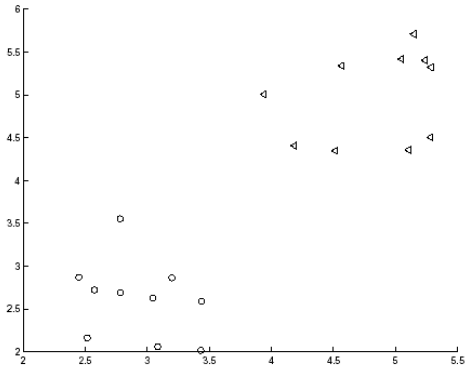
Graph construction :



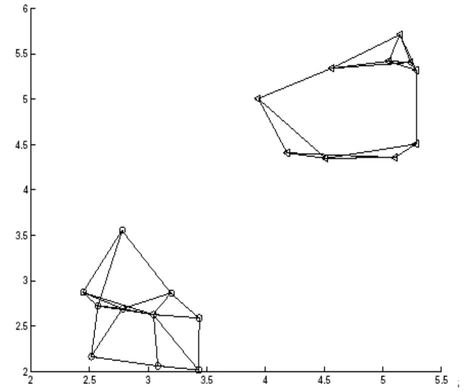
Clustering with ECT distance

Graph construction :

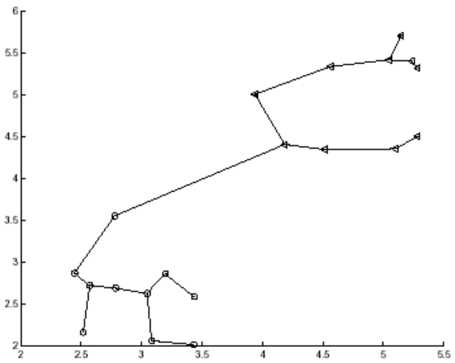
data



3 nearest neighbors



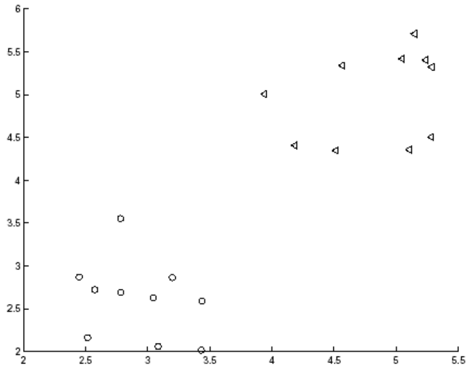
Minimum spanning tree



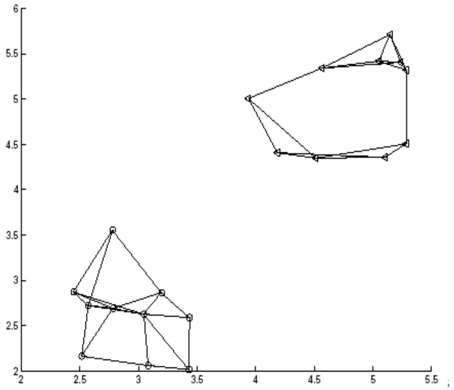
Clustering with ECT distance

Graph construction :

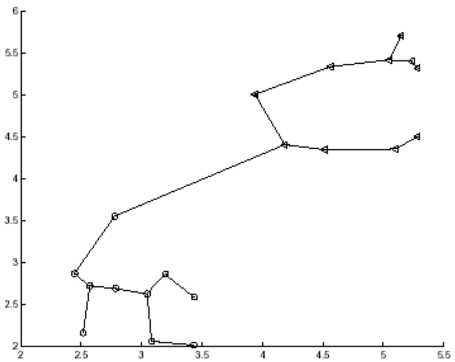
data



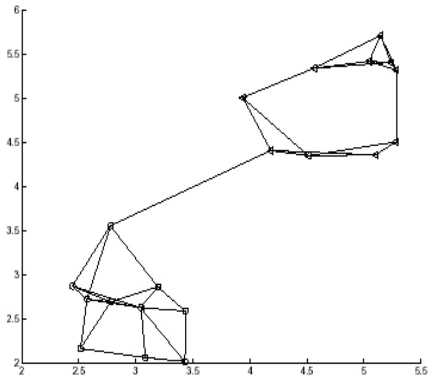
3 nearest neighbors



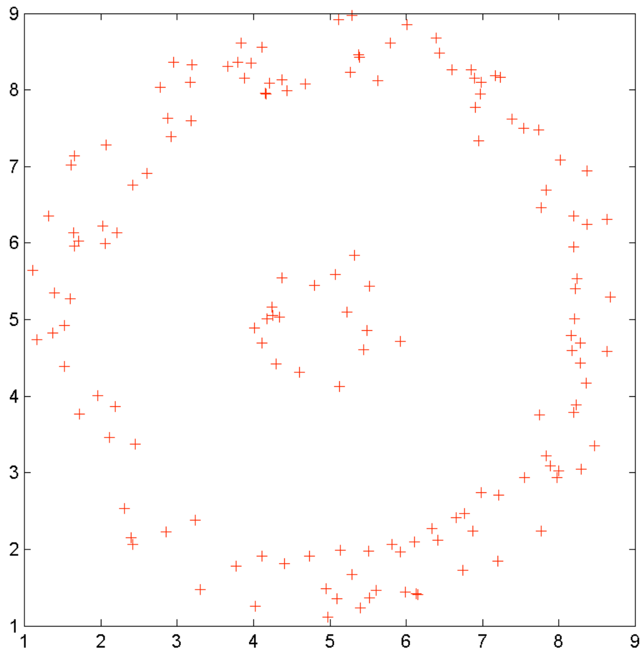
Minimum spanning tree



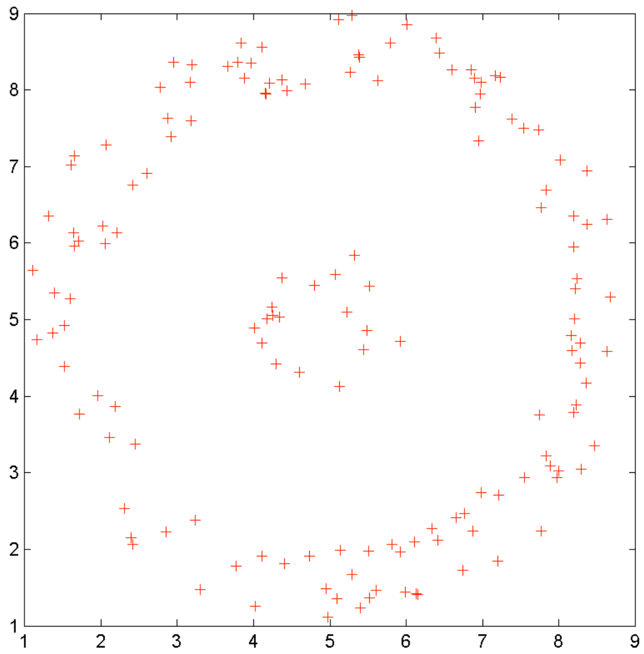
Final graph

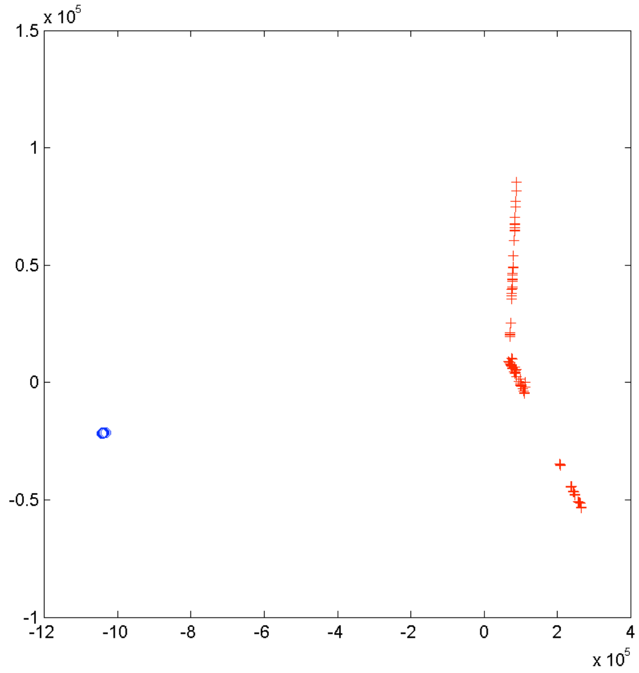


Clustering with ECT distance

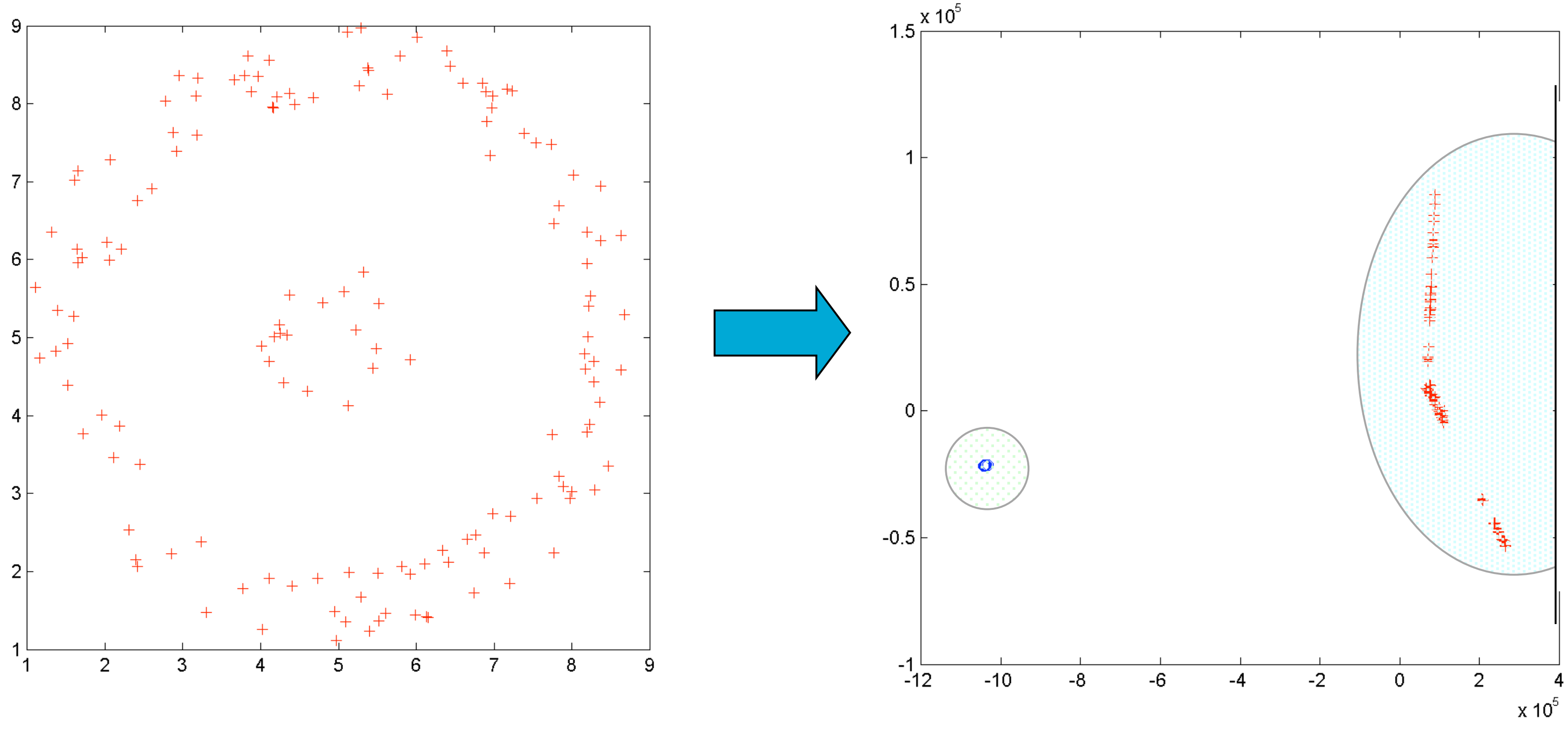


Clustering with ECT distance



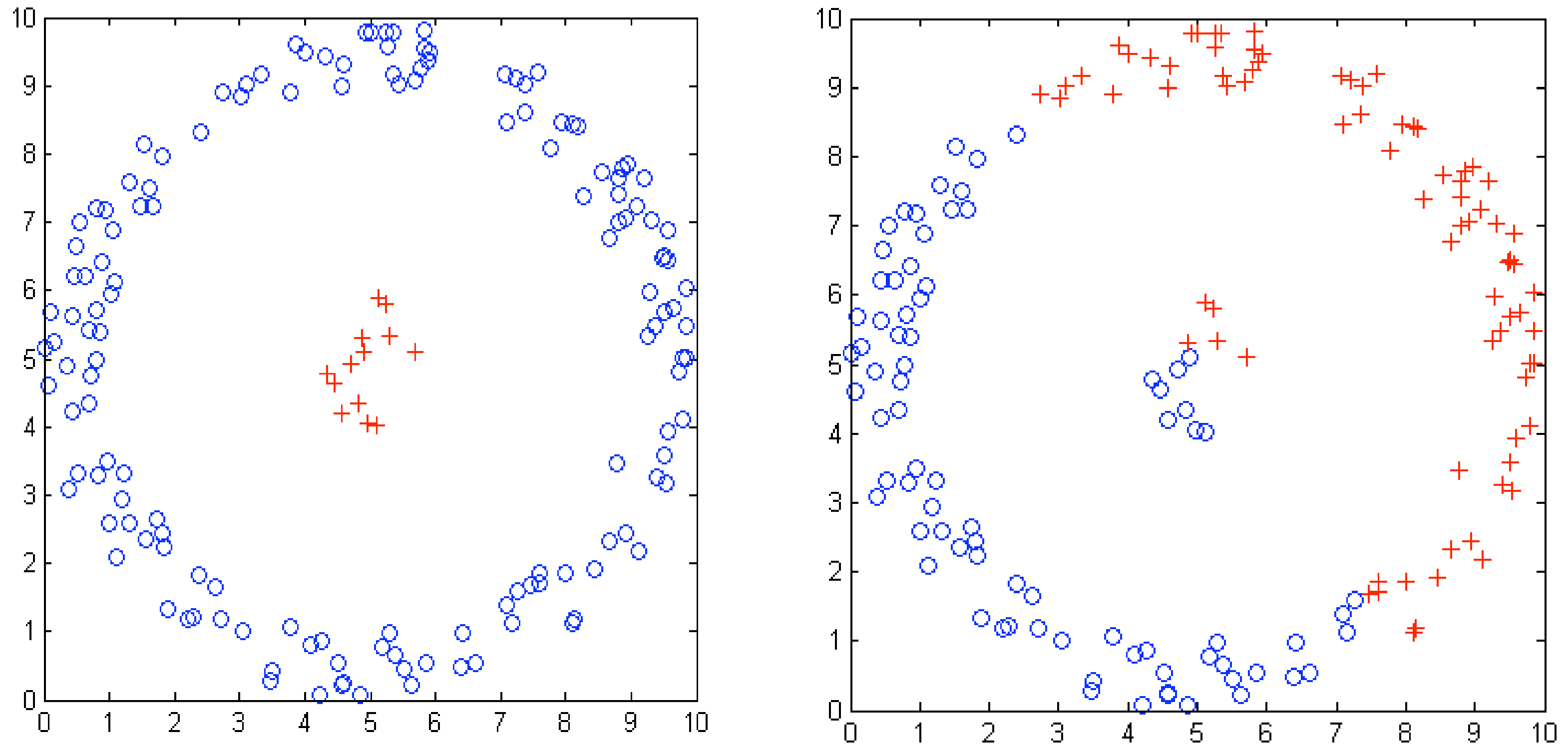


Clustering with ECT distance

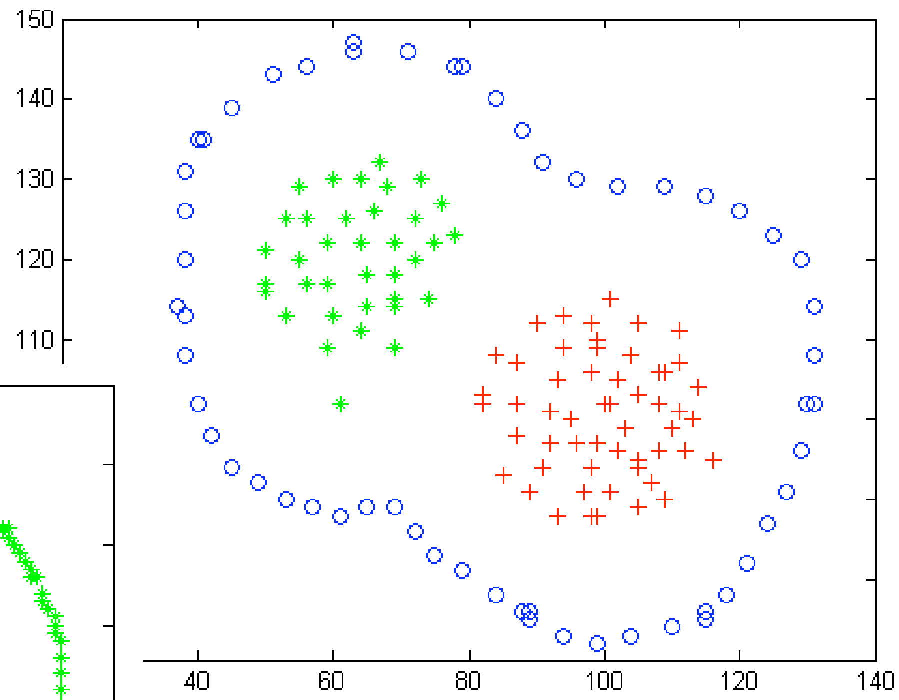


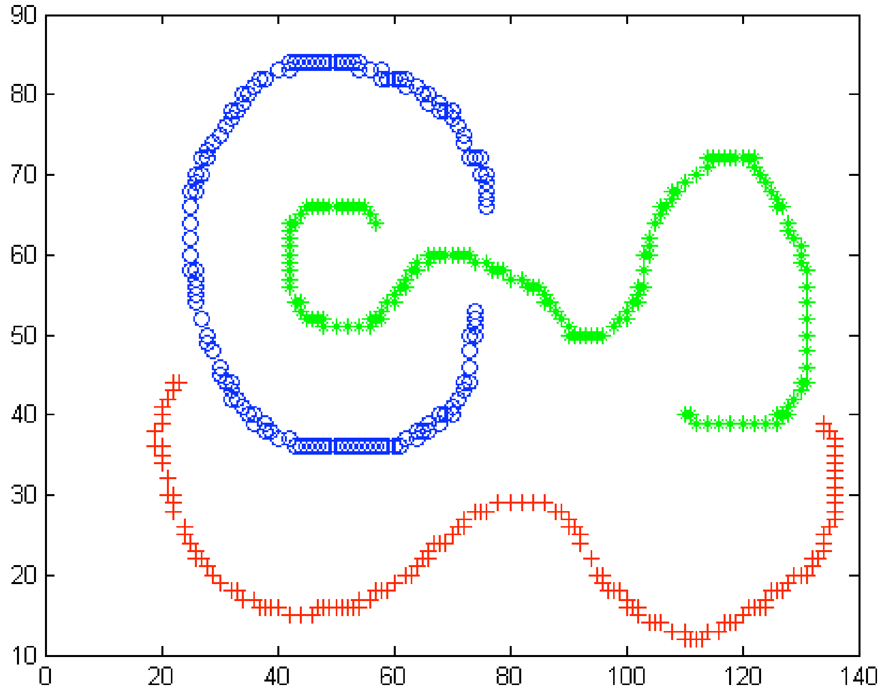
Clustering with ECT distance

- Clustering results using ECT distance k-means, in comparison with the classical k-means



Autres exemples





Links with other methods

- Very interesting links with
  - – Spectral clustering
  - – Graph visualization algorithms
  - – Electrical networks
    - The commute time distance is equivalent to the effective resistance
  - – Experimental design

Application to collaborative recommendation

Experimental results: Application to collaborative recommendation

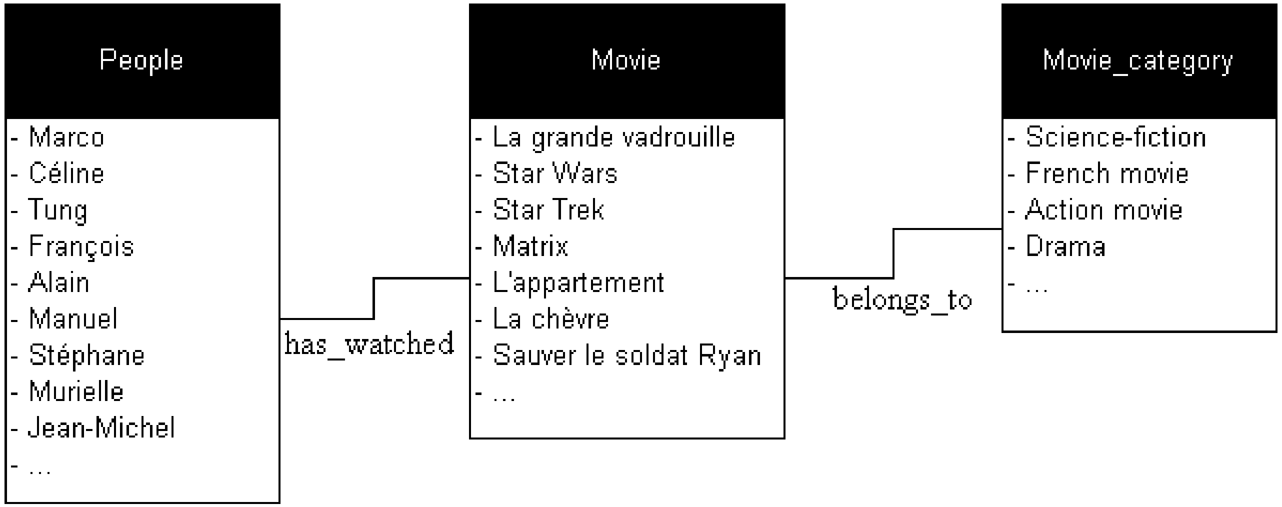
- Results on a real movie database: a sample of the MovieLens database
- 943 users
- 1682 movies
- 19 movie categories
- 100,000 ratings
- Divided into a training set and a test set

- **Experiment: suggest movies to people**

Experimental results: Application to collaborative recommendation

- Tables connected by relationships

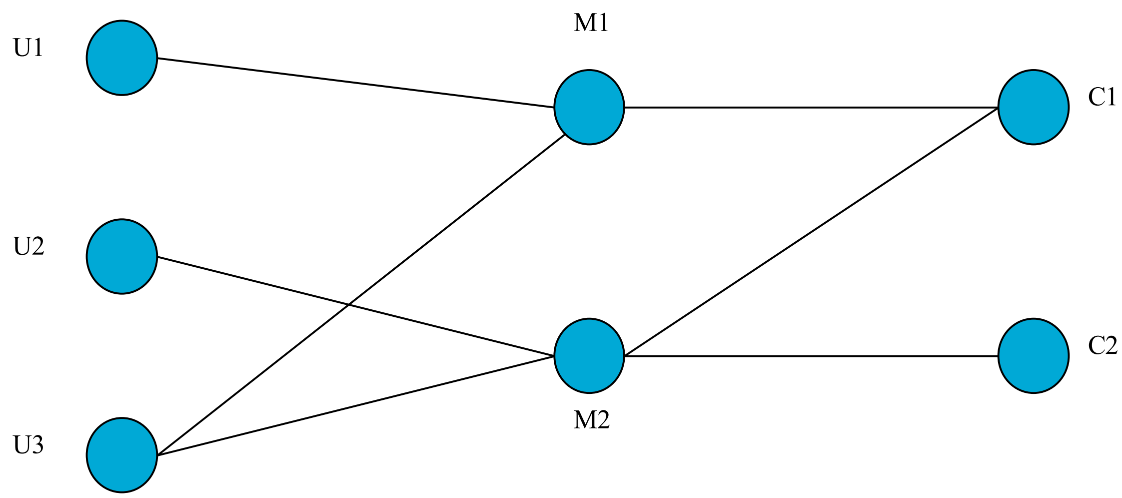
  Example: A movie database



→ Computing similarities beween people and movies allows to suggest movies to watch or not to watch (collaborative recommendation)

Experimental results: Application to collaborative recommendation

- Experiment: suggest unwatched movies to people



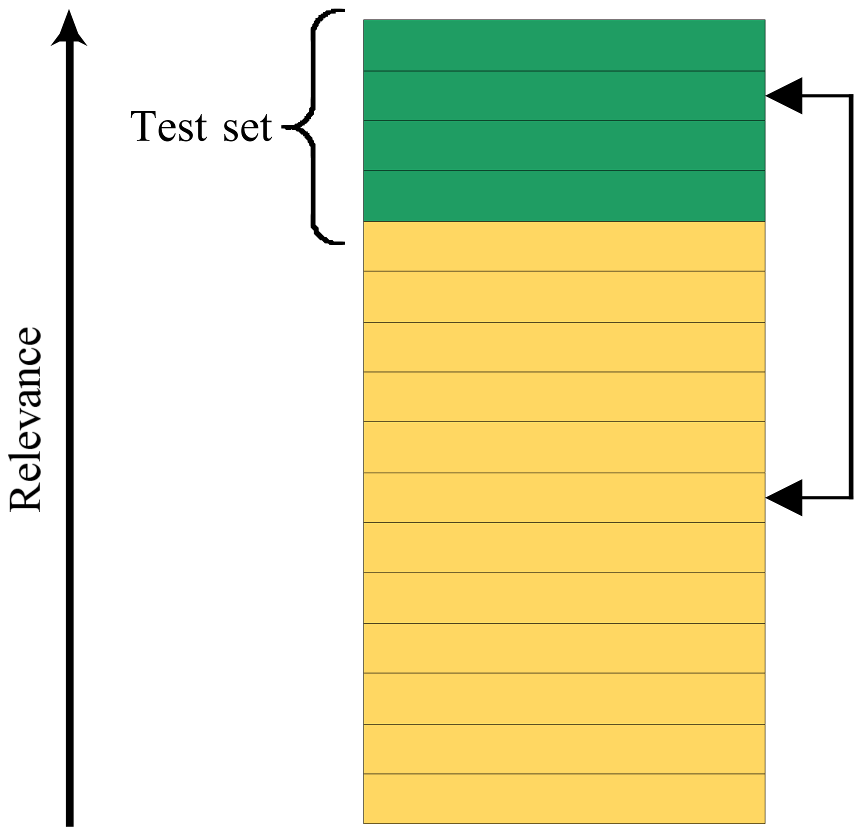
- The test set contains 10 movies for each user
  = 9430 movies

Scoring algorithms

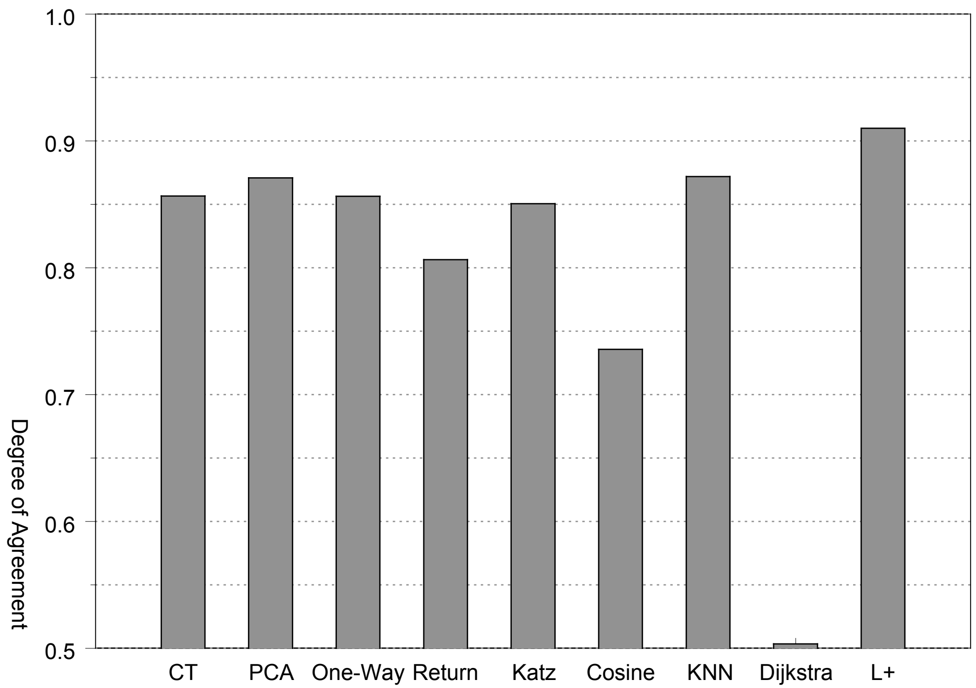
- Average commute time (CT)
- Average first-passage time (One-way)
- Average first-passage time (Return)
- $\mathbf{L}^{+}$ (pseudoinverse of the Laplacian matrix of the graph)
- Katz
- K-nearest neighbours (KNN) (Standard technique)
- Dijkstra (Standard technique)
- Cosine (Standard technique)

Performance evaluation: degree of agreement (a variant of Somers'D)

Ideal ranking



Results



CT	PCA CT	One-way	Return	Katz	KNN	Dijkstra	Cosine	$\mathbf{L}^+$
0.8566	0.8710	0.8564	0.8065	0.8506	0.8720	0.5034	0.7358	0.9101

Conclusion

- We introduced a general procedure for computing dissimilarities between any pair of elements
- The commute time is a distance metric in an Euclidean space
- This allows the application of data analysis methods (PCA, discriminant analysis, clustering) for graph analysis