A practical implementation of Spectral Clustering algorithm

Argimiro Arratia & Carlos Marijuán

LSI, Univ. Politécnica de Catalunya
argimiro@lsi.upc.edu
Universidad de Valladolid

JMDA 2010

Basic motivations

- Understand the spectral clustering algorithm
- and apply it to classifying pages in the internet ...

Basic motivations

- Understand the spectral clustering algorithm
- and apply it to classifying pages in the internet ...

What we achieved, what we learned

- The theory of Spectral clustering. Heavy Linear Algebra, graph theory.
- Set up and master a (very) professional open-source web crawler: WIRE.
- Master some high level MatLab functions,
- Work out some similarity measures
- Work out some ideas on determining the number of clusters

What we achieved, what we learned

- The theory of Spectral clustering. Heavy Linear Algebra, graph theory.
- Set up and master a (very) professional open-source web crawler: WIRE.
- Master some high level MatLab functions,
- Work out some similarity measures
- Work out some ideas on determining the number of clusters

What we achieved, what we learned

- The theory of Spectral clustering. Heavy Linear Algebra, graph theory.
- Set up and master a (very) professional open-source web crawler: WIRE.
- Master some high level MatLab functions,
- Work out some similarity measures
- Work out some ideas on determining the number of clusters

What we achieved, what we learned

- The theory of Spectral clustering. Heavy Linear Algebra, graph theory.
- Set up and master a (very) professional open-source web crawler: WIRE.
- Master some high level MatLab functions,
- Work out some similarity measures
- Work out some ideas on determining the number of clusters

What we achieved, what we learned

- The theory of Spectral clustering. Heavy Linear Algebra, graph theory.
- Set up and master a (very) professional open-source web crawler: WIRE.
- Master some high level MatLab functions,
- Work out some similarity measures
- Work out some ideas on determining the number of clusters

Clustering

We have data points $v_1, \ldots, v_n$ and a similarity relation $s_{ij} \geq 0$

The problem of clustering (I)

Goal: to divide the data into subsets such that points in the same subset are similar and points in different subsets are dissimilar

Representation

Similarity graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$ and $v_i v_j \in E \iff s_{ij} > 0$, and the edge $v_i v_j$ has weight s_{ij}.

Clustering as graph partition (II)

To find a partition of G such that edges between two different parts have low weight and edges within same part have high weight.

Clustering

We have data points $v_1, \ldots, v_n$ and a similarity relation $s_{ij} \geq 0$

The problem of clustering (I)

Goal: to divide the data into subsets such that points in the same subset are similar and points in different subsets are dissimilar

Representation

Similarity graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$ and $v_i v_j \in E \iff s_{ij} > 0$, and the edge $v_i v_j$ has weight s_{ij}.

Clustering as graph partition (II)

To find a partition of G such that edges between two different parts have low weight and edges within same part have high weight.

Clustering

We have data points $v_1, \ldots, v_n$ and a similarity relation $s_{ij} \geq 0$

The problem of clustering (I)

Goal: to divide the data into subsets such that points in the same subset are similar and points in different subsets are dissimilar

Representation

Similarity graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$ and $v_i v_j \in E \iff s_{ij} > 0$, and the edge $v_i v_j$ has weight s_{ij}.

Clustering as graph partition (II)

To find a partition of G such that edges between two different parts have low weight and edges within same part have high weight.

Clustering

We have data points $v_1, \ldots, v_n$ and a similarity relation $s_{ij} \geq 0$

The problem of clustering (I)

Goal: to divide the data into subsets such that points in the same subset are similar and points in different subsets are dissimilar

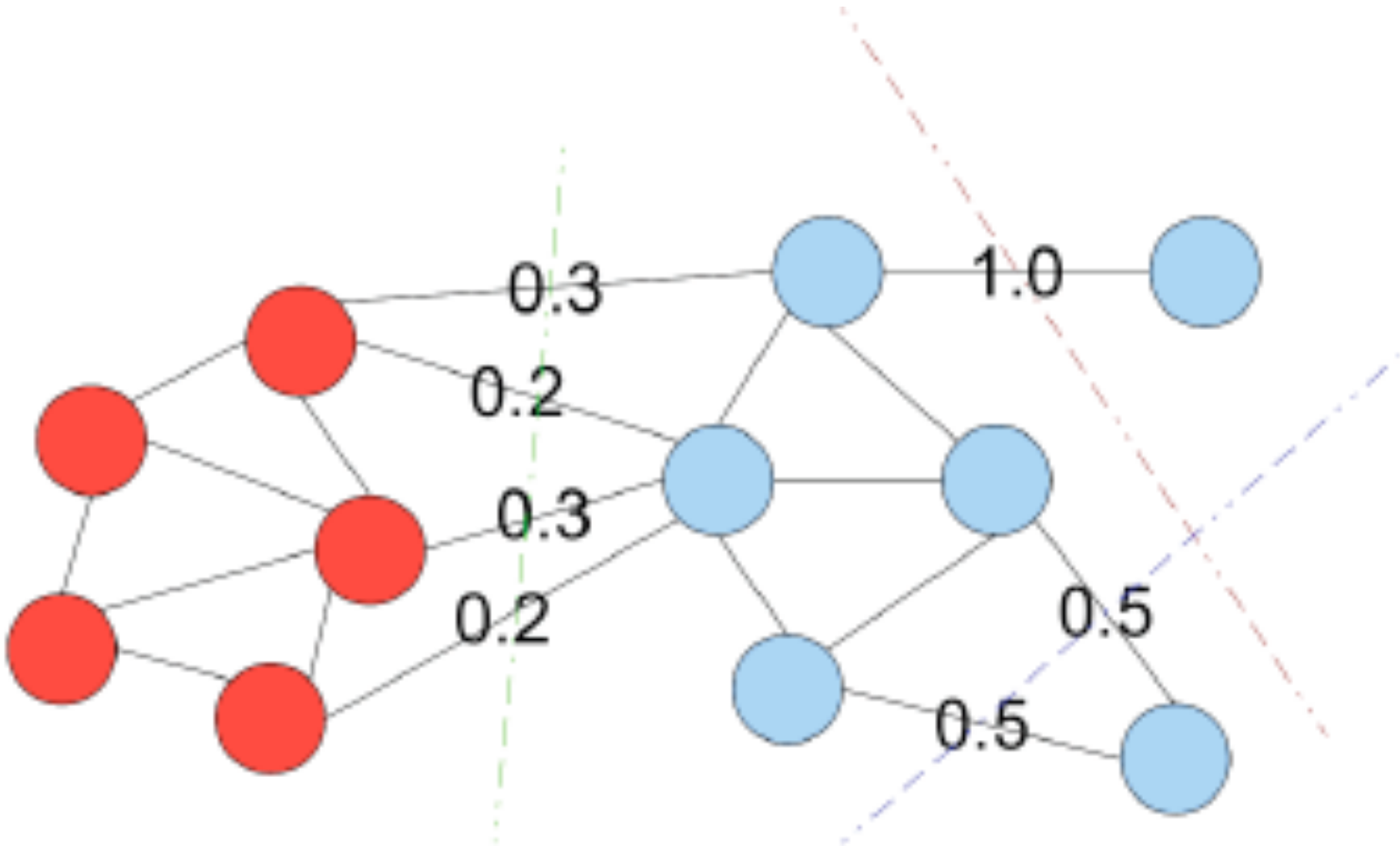
Representation

Similarity graph $G = (V, E)$, $V = \{v_1, \ldots, v_n\}$ and $v_i v_j \in E \iff s_{ij} > 0$, and the edge $v_i v_j$ has weight s_{ij}.

Clustering as graph partition (II)

To find a partition of G such that edges between two different parts have low weight and edges within same part have high weight.

A graph partition



One of the difficulties is to avoid trivial partitions

Spectral clustering in a nutshell

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation (or weights) $s_{ij} \geq 0$, let:

- $D_i = \sum_{j=1}^{n} s_{ij}$ the **volume** of v_i (aka degree, for $s_{ij} = 1$)
- $D = diag(D_1, \ldots, D_n)$ diagonal matrix of volumes
- $S = (s_{ij})_{1 \leq i,j \leq n}$ weighted (adjacency) matrix

Unnormalized Laplacian matrix

$$L = D - S$$

Two normalized Laplacian matrices

$$L_{sym} = D^{1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} S$$

Spectral clustering in a nutshell

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation (or weights) $s_{ij} \geq 0$, let:

- $D_i = \sum_{j=1}^{n} s_{ij}$ the **volume** of v_i (aka degree, for $s_{ij} = 1$)
- $D = diag(D_1, \ldots, D_n)$ diagonal matrix of volumes
- $S = (s_{ij})_{1 \leq i,j \leq n}$ weighted (adjacency) matrix

Unnormalized Laplacian matrix

$$L = D - S$$

Two normalized Laplacian matrices

$$L_{sym} = D^{1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} S$$

Spectral clustering in a nutshell

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation (or weights) $s_{ij} \geq 0$, let:

- $D_i = \sum_{j=1}^{n} s_{ij}$ the **volume** of v_i (aka degree, for $s_{ij} = 1$)
- $D = diag(D_1, \ldots, D_n)$ diagonal matrix of volumes
- $S = (s_{ij})_{1 \leq i,j \leq n}$ weighted (adjacency) matrix

Unnormalized Laplacian matrix

$$L = D - S$$

Two normalized Laplacian matrices

$$L_{sym} = D^{1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} S$$

Spectral clustering in a nutshell

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation (or weights) $s_{ij} \geq 0$, let:

- $D_i = \sum_{j=1}^{n} s_{ij}$ the **volume** of v_i (aka degree, for $s_{ij} = 1$)
- $D = diag(D_1, \ldots, D_n)$ diagonal matrix of volumes
- $S = (s_{ij})_{1 \leq i,j \leq n}$ weighted (adjacency) matrix

Unnormalized Laplacian matrix

$$L = D - S$$

Two normalized Laplacian matrices

$$L_{sym} = D^{1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} S$$

Spectral clustering in a nutshell

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation (or weights) $s_{ij} \geq 0$, let:

- $D_i = \sum_{j=1}^{n} s_{ij}$ the **volume** of v_i (aka degree, for $s_{ij} = 1$)
- $D = diag(D_1, \ldots, D_n)$ diagonal matrix of volumes
- $S = (s_{ij})_{1 \leq i,j \leq n}$ weighted (adjacency) matrix

Unnormalized Laplacian matrix

$$L = D - S$$

Two normalized Laplacian matrices

$$L_{sym} = D^{1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} S$$

Spectral clustering in a nutshell

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation (or weights) $s_{ij} \geq 0$, let:

- $D_i = \sum_{j=1}^{n} s_{ij}$ the **volume** of v_i (aka degree, for $s_{ij} = 1$)
- $D = diag(D_1, \ldots, D_n)$ diagonal matrix of volumes
- $S = (s_{ij})_{1 \leq i,j \leq n}$ weighted (adjacency) matrix

Unnormalized Laplacian matrix

$$L = D - S$$

Two normalized Laplacian matrices

$$L_{sym} = D^{1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} S$$

Spectral clustering in a nutshell

Given data set $V = \{v_1, \ldots, v_n\}$ and a similarity relation (or weights) $s_{ij} \geq 0$, let:

- $D_i = \sum_{j=1}^{n} s_{ij}$ the **volume** of v_i (aka degree, for $s_{ij} = 1$)
- $D = diag(D_1, \ldots, D_n)$ diagonal matrix of volumes
- $S = (s_{ij})_{1 \leq i,j \leq n}$ weighted (adjacency) matrix

Unnormalized Laplacian matrix

$$L = D - S$$

Two normalized Laplacian matrices

$$L_{sym} = D^{1/2} L D^{-1/2} = I - D^{1/2} S D^{-1/2}$$

$$L_{rw} = D^{-1} L = I - D^{-1} S$$

Basic spectral clustering algorithm

Input: $S = (s_{ij})_{1 \leq i,j \leq n}$ (similarity matrix),
k (number of clusters)

Output: Vector idx of k indices for the cluster sets

1. Compute the Laplacian matrix L associated to S
2. Compute the first k eigenvectors $u_1, \ldots, u_k$ of L
3. Let $U = (u_1 \ldots u_k)$, where u_i is a column in $\mathbb{R}^n$
4. For $i = 1, \ldots, n$, let y_i be the i-th row of U and use k-means to clusters the rows $\{y_i : i = 1, \ldots, n\}$ of U as points in $\mathbb{R}^k$
5. For $j = 1, \ldots, k$, let $idx(j)$ be the index set of all the rows of U labelled with j (i.e. $i \in idx(j)$ iff y_i is in the j-th cluster)

[Shi & Malik, 2000, use L_{rw}], [Ng, Jordan, Weiss, 2002, use L_{sym}]

Graph partition vs Spectral Clustering

Given similarity graph $G = (V, S)$, $S = (s_{ij})_{1 \leq i,j \leq n}$, to solve the partition problem is to solve the

mincut problem

To find a partition $A_1, \ldots, A_k$ of V which minimizes

$$RatioCut(A_1, \ldots, A_k) = \sum_{i=1}^{k} \frac{cut(A_i, \overline{A_i})}{|A_i|}$$

where $cut(A, \overline{A}) = \sum_{i \in A, j \notin A} s_{ij}$

Graph partition vs Spectral Clustering II

Given a partition $A_1, \ldots, A_k$ of V, let

$$h_{ij} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } \ i \in A_j \\ \\ 0 & \text{otherwise} \end{cases}$$

Define k indicator vectors $h_i = (h_{1i}, \ldots, h_{ni})^t$ and $H = (h_1, \ldots, h_k) \in \mathbb{R}^{n \times k}$.

We have

- $H^t H = I$
- For Laplacian $L = D - S$,

$$h_i^t L h_i = (H^t L H)_{ii} = 2\frac{cut(A_i, \overline{A_i})}{|A_i|}$$

Graph partition vs Spectral Clustering II

Given a partition $A_1, \ldots, A_k$ of V, let

$$h_{ij} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } \ i \in A_j \\ \\ 0 & \text{otherwise} \end{cases}$$

Define k indicator vectors $h_i = (h_{1i}, \ldots, h_{ni})^t$ and $H = (h_1, \ldots, h_k) \in \mathbb{R}^{n \times k}$.
We have

- $H^t H = I$
- For Laplacian $L = D - S$,

$$h_i^t L h_i = (H^t L H)_{ii} = 2\frac{cut(A_i, \overline{A_i})}{|A_i|}$$

Graph partition vs Spectral Clustering II

Given a partition $A_1, \ldots, A_k$ of V, let

$$h_{ij} = \begin{cases} 1/\sqrt{|A_i|} & \text{if } \ i \in A_j \\ \\ 0 & \text{otherwise} \end{cases}$$

Define k indicator vectors $h_i = (h_{1i}, \ldots, h_{ni})^t$ and $H = (h_1, \ldots, h_k) \in \mathbb{R}^{n \times k}$.
We have

- $H^t H = I$
- For Laplacian $L = D - S$,

$$h_i^t L h_i = (H^t L H)_{ii} = 2\frac{cut(A_i, \overline{A_i})}{|A_i|}$$

Graph partition vs Spectral Clustering III

Therefore

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2}\sum_{i=1}^{k} h_i^t L h_i = \frac{1}{2}\sum_{i=1}^{k}(H^t L H)_{ii} = \frac{1}{2}Tr(H^t L H)$$

Then minimizing $RatioCut(A_1, \ldots, A_k)$ is equivalent to

$$\min_{A_1,\ldots,A_k} Tr(H^t L H) \text{ subject to } H^t H = I,\ H = (h_1, \ldots, h_k)$$

Relax H to real values, and problem becomes

$$\min_{H \in \mathbb{R}^{n \times k}} Tr(H^t L H) \text{ subject to } H^t H = I$$

The solution is obtained by choosing H as matrix with columns the first k eigenvectors of L.

Graph partition vs Spectral Clustering III

Therefore

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2}\sum_{i=1}^{k} h_i^t L h_i = \frac{1}{2}\sum_{i=1}^{k}(H^t L H)_{ii} = \frac{1}{2}Tr(H^t L H)$$

Then minimizing $RatioCut(A_1, \ldots, A_k)$ is equivalent to

$$\min_{A_1,\ldots,A_k} Tr(H^t L H) \text{ subject to } H^t H = I,\ H = (h_1, \ldots, h_k)$$

Relax H to real values, and problem becomes

$$\min_{H \in \mathbb{R}^{n \times k}} Tr(H^t L H) \text{ subject to } H^t H = I$$

The solution is obtained by choosing H as matrix with columns the first k eigenvectors of L.

Graph partition vs Spectral Clustering III

Therefore

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2}\sum_{i=1}^{k} h_i^t L h_i = \frac{1}{2}\sum_{i=1}^{k}(H^t L H)_{ii} = \frac{1}{2}Tr(H^t L H)$$

Then minimizing $RatioCut(A_1, \ldots, A_k)$ is equivalent to

$$\min_{A_1,\ldots,A_k} Tr(H^t L H) \text{ subject to } H^t H = I,\ H = (h_1, \ldots, h_k)$$

Relax H to real values, and problem becomes

$$\min_{H \in \mathbb{R}^{n \times k}} Tr(H^t L H) \text{ subject to } H^t H = I$$

The solution is obtained by choosing H as matrix with columns the first k eigenvectors of L.

Graph partition vs Spectral Clustering III

Therefore

$$RatioCut(A_1, \ldots, A_k) = \frac{1}{2}\sum_{i=1}^{k} h_i^t L h_i = \frac{1}{2}\sum_{i=1}^{k}(H^t L H)_{ii} = \frac{1}{2}Tr(H^t L H)$$

Then minimizing $RatioCut(A_1, \ldots, A_k)$ is equivalent to

$$\min_{A_1,\ldots,A_k} Tr(H^t L H) \text{ subject to } H^t H = I,\ H = (h_1, \ldots, h_k)$$

Relax H to real values, and problem becomes

$$\min_{H \in \mathbb{R}^{n\times k}} Tr(H^t L H) \text{ subject to } H^t H = I$$

The solution is obtained by choosing H as matrix with columns the first k eigenvectors of L.

Practical consideration: Computing large eigensystem

We are concern with computing eigenvalues and eigenvectors of large Laplacians.

A simple but effective iterative method for computing the largest (in absolute value) eigenvalue is the Power Method:

Input: arbitrary n-dim vector x_0, matrix A
for $k = 1, \ldots, m$ do
$y_k = Ax_{k-1}$
$x_k = y_k / ||y_k||$
end for

If the true dominant eigenvector is u then the associated eigenvalue $\lambda = \frac{\langle u, Au \rangle}{\langle u, u \rangle}$.
Thus approximate the value of λ by the *Rayleigh quotient*:

$$\lambda \approx \frac{\langle x_m, Ax_m \rangle}{\langle x_m, x_m \rangle}.$$

Practical consideration: Computing large eigensystem

We are concern with computing eigenvalues and eigenvectors of large Laplacians.
A simple but effective iterative method for computing the largest (in absolute value) eigenvalue is the Power Method:

Input: arbitrary n-dim vector x_0, matrix A
for $k = 1, \ldots, m$ do
$y_k = Ax_{k-1}$
$x_k = y_k / ||y_k||$
end for

If the true dominant eigenvector is u then the associated eigenvalue $\lambda = \frac{\langle u, Au \rangle}{\langle u, u \rangle}$.
Thus approximate the value of λ by the *Rayleigh quotient*:

$$\lambda \approx \frac{\langle x_m, Ax_m \rangle}{\langle x_m, x_m \rangle}.$$

Practical consideration: Computing large eigensystem

We are concern with computing eigenvalues and eigenvectors of large Laplacians.
A simple but effective iterative method for computing the largest (in absolute value) eigenvalue is the Power Method:

Input: arbitrary n-dim vector x_0, matrix A
for $k = 1, \ldots, m$ do
$y_k = Ax_{k-1}$
$x_k = y_k / ||y_k||$
end for

If the true dominant eigenvector is u then the associated eigenvalue $\lambda = \frac{\langle u, Au \rangle}{\langle u, u \rangle}$.
Thus approximate the value of λ by the *Rayleigh quotient*:

$$\lambda \approx \frac{\langle x_m, Ax_m \rangle}{\langle x_m, x_m \rangle}.$$

Computing large eigensystem II

For computing the k first eigenvectors there are faster methods based on projections over Krylov subspaces

Within this family of spectral methods we chose to use the method of Arnoldi.

An advantage of using the method of Arnoldi for solving eigensystems is that it is implemented in MatLab under the function $eigs$

The function $eigs$ is based on the software package ARPACK written in Fortran

Computing large eigensystem II

For computing the k first eigenvectors there are faster methods based on projections over Krylov subspaces
Within this family of spectral methods we chose to use the method of Arnoldi.
An advantage of using the method of Arnoldi for solving eigensystems is that it is implemented in MatLab under the function $eigs$
The function $eigs$ is based on the software package ARPACK written in Fortran

Practical Considerations: Condition number

Every Laplacian matrix has $\lambda_1 = 0$
hence, computing Laplacians is an intrinsically badly conditioned problem

The **Condition Number** of a symmetric matrix A

$$\mu(A) = \frac{\max\{\lambda : \lambda \text{ eigenvalue of } A\}}{\min\{\lambda : \lambda \text{ eigenvalue of } A\}}$$

Solution: shift A adding $\rho(A)$ to diagonal
Fact: The shift does not changes the value of eigenvectors

Practical Considerations: Condition number

Every Laplacian matrix has $\lambda_1 = 0$
hence, computing Laplacians is an intrinsically badly conditioned problem
The **Condition Number** of a symmetric matrix A

$$\mu(A) = \frac{\max\{\lambda : \lambda \text{ eigenvalue of } A\}}{\min\{\lambda : \lambda \text{ eigenvalue of } A\}}$$

Solution: shift A adding $\rho(A)$ to diagonal
Fact: The shift does not changes the value of eigenvectors

Practical Considerations: Condition number

Every Laplacian matrix has $\lambda_1 = 0$
hence, computing Laplacians is an intrinsically badly conditioned problem
The **Condition Number** of a symmetric matrix A

$$\mu(A) = \frac{\max\{\lambda : \lambda \text{ eigenvalue of } A\}}{\min\{\lambda : \lambda \text{ eigenvalue of } A\}}$$

Solution: shift A adding $\rho(A)$ to diagonal

Fact: The shift does not changes the value of eigenvectors

Practical Considerations: Condition number

Every Laplacian matrix has $\lambda_1 = 0$
hence, computing Laplacians is an intrinsically badly conditioned problem
The **Condition Number** of a symmetric matrix A

$$\mu(A) = \frac{\max\{\lambda : \lambda \text{ eigenvalue of } A\}}{\min\{\lambda : \lambda \text{ eigenvalue of } A\}}$$

Solution: shift A adding $\rho(A)$ to diagonal
Fact: The shift does not changes the value of eigenvectors

Practical Consideration: crawling the Web

We chose WIRE, for extracting information from the web

- Powerful open source webcrawler by Carlos Castillo, 2004
- Oriented to information extraction and making of web statistics
- Allows planing in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place keeping history)
- Parametrize by a unique XML script (`wire.conf`)

Our small contribution: No webcrawler generates the adjacency matrix of the graph that is visiting, so modified the utility `wire-info-extract` to get adjacency matrix in CRS format.

Practical Consideration: crawling the Web

We chose WIRE, for extracting information from the web

- Powerful open source webcrawler by Carlos Castillo, 2004
- Oriented to information extraction and making of web statistics
- Allows planing in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place keeping history)
- Parametrize by a unique XML script (`wire.conf`)

Our small contribution: No webcrawler generates the adjacency matrix of the graph that is visiting, so modified the utility `wire-info-extract` to get adjacency matrix in CRS format.

Practical Consideration: crawling the Web

We chose WIRE, for extracting information from the web

- Powerful open source webcrawler by Carlos Castillo, 2004
- Oriented to information extraction and making of web statistics
- Allows planing in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place keeping history)
- Parametrize by a unique XML script (`wire.conf`)

Our small contribution: No webcrawler generates the adjacency matrix of the graph that is visiting, so modified the utility `wire-info-extract` to get adjacency matrix in CRS format.

Practical Consideration: crawling the Web

We chose WIRE, for extracting information from the web

- Powerful open source webcrawler by Carlos Castillo, 2004
- Oriented to information extraction and making of web statistics
- Allows planing in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place keeping history)
- Parametrize by a unique XML script (`wire.conf`)

Our small contribution: No webcrawler generates the adjacency matrix of the graph that is visiting, so modified the utility `wire-info-extract` to get adjacency matrix in CRS format.

Practical Consideration: crawling the Web

We chose WIRE, for extracting information from the web

- Powerful open source webcrawler by Carlos Castillo, 2004
- Oriented to information extraction and making of web statistics
- Allows planing in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place keeping history)
- Parametrize by a unique XML script (`wire.conf`)

Our small contribution: No webcrawler generates the adjacency matrix of the graph that is visiting, so modified the utility `wire-info-extract` to get adjacency matrix in CRS format.

Practical Consideration: crawling the Web

We chose WIRE, for extracting information from the web

- Powerful open source webcrawler by Carlos Castillo, 2004
- Oriented to information extraction and making of web statistics
- Allows planing in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place keeping history)
- Parametrize by a unique XML script (`wire.conf`)

Our small contribution: No webcrawler generates the adjacency matrix of the graph that is visiting, so modified the utility `wire-info-extract` to get adjacency matrix in CRS format.

Practical Consideration: crawling the Web

We chose WIRE, for extracting information from the web

- Powerful open source webcrawler by Carlos Castillo, 2004
- Oriented to information extraction and making of web statistics
- Allows planing in short as well as long term (e.g. the crawler can be stopped and later restart at last visited place keeping history)
- Parametrize by a unique XML script (`wire.conf`)

Our small contribution: No webcrawler generates the adjacency matrix of the graph that is visiting, so modified the utility `wire-info-extract` to get adjacency matrix in CRS format.

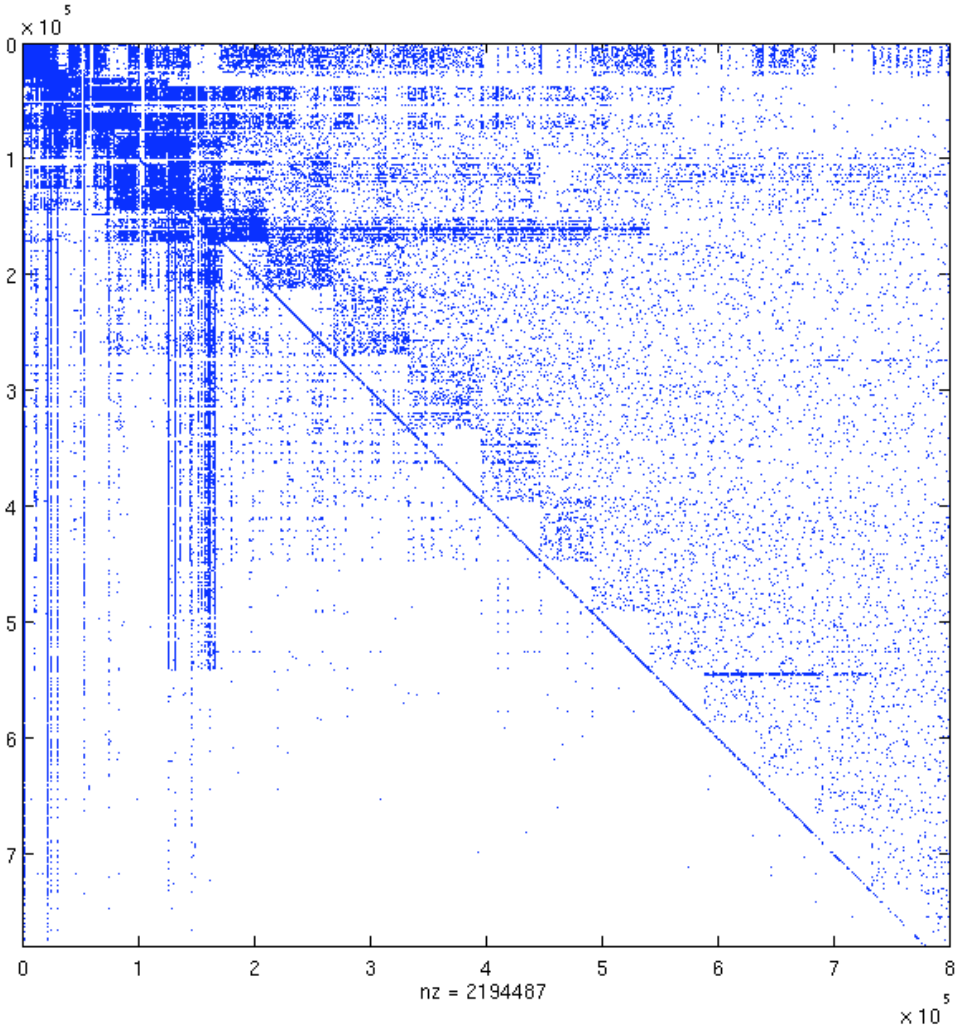
Modified normalized spectral clustering

Input: $S = (s_{ij})_{1 \leq i,j \leq n}$ (similarity matrix),
k (number of clusters)

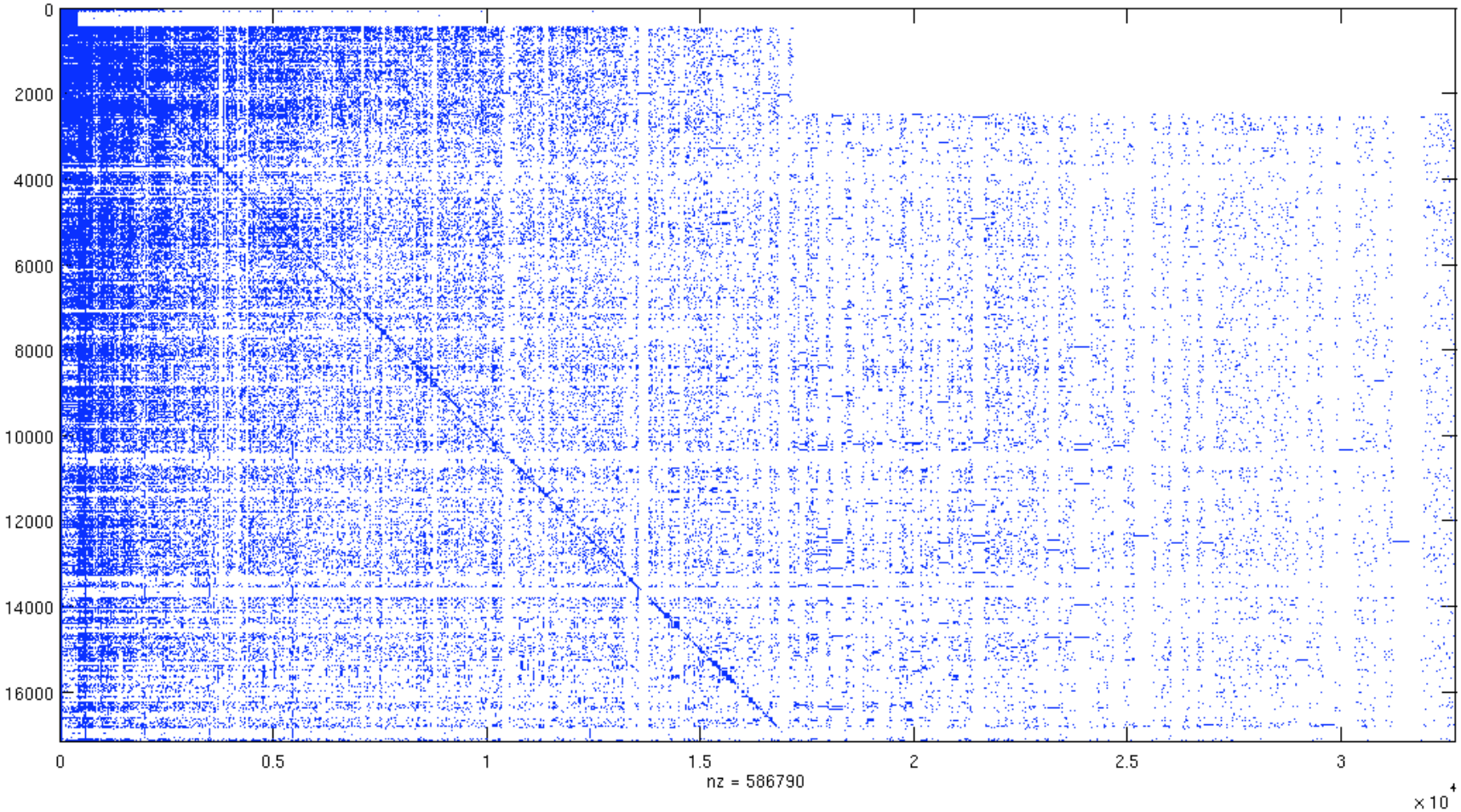
Output: Vector idx of k indices for the cluster sets

1. Compute the Laplacian matrix L_{rw} associated to S
2. Shift the spectrum of L_{rw} to enhance its condition num.
   by adding the spectral radio to the main diagonal:
   $major = eigs(L_{rw}, 1)$;
   $L_{rwt} = L_{rw} + (major * speye(size(L_{rw})))$;
3. Compute the first k eigenvectors $u_1, \ldots, u_k$ of L_{rw}
   Let $U = (u_1 \ldots u_k)$, where u_i is a column in $\mathbb{R}^n$
4. For $i = 1, \ldots, n$, let y_i be the i-th row of U and
   use k-means to cluster the rows $\{y_i : i = 1, \ldots, n\}$
   of U as points in $\mathbb{R}^k$. The procedure is repeated
   10 times to avoid local optimal points
   $idx = kmeans(U, k, `replicates`, 10)$
5. For $j = 1, \ldots, k$, let $idx(j)$ be the index set
   of all the rows of U labelled with j
   (i.e. $i \in idx(j)$ iff y_i is in the j-th cluster)

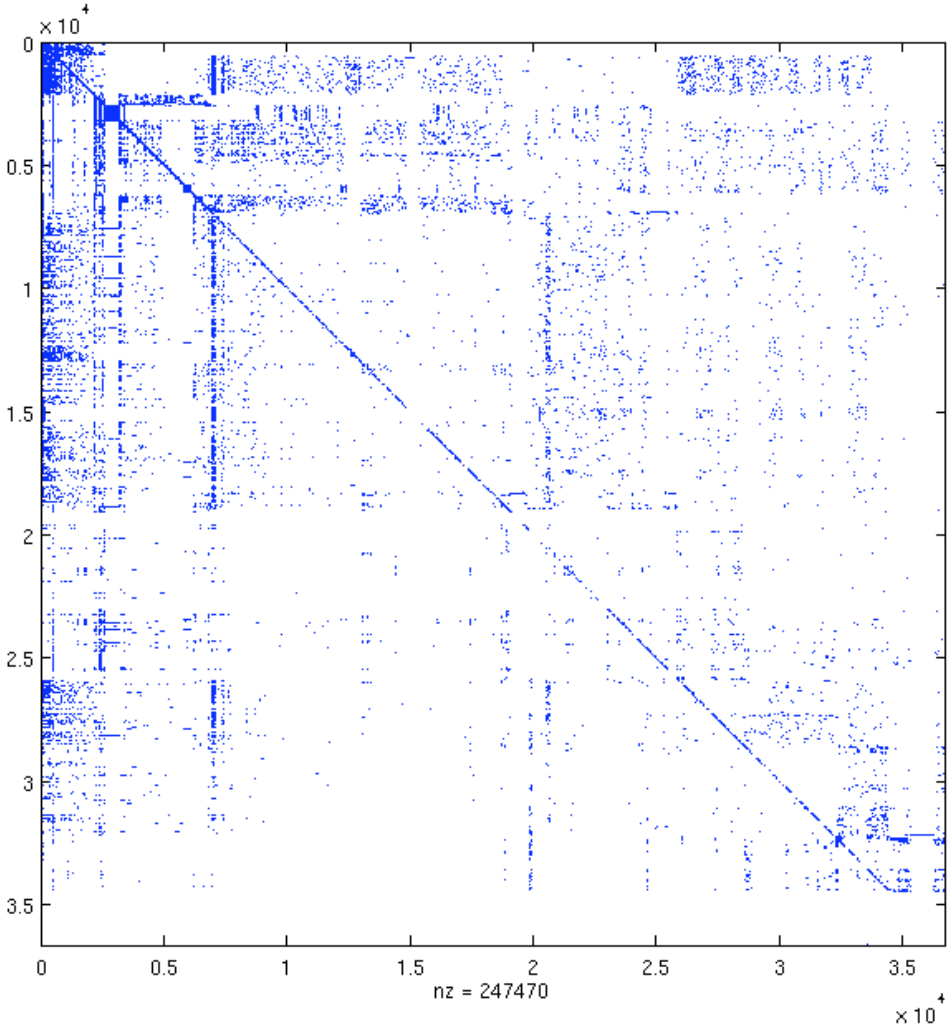
Experiments: adjacency matrices obtained by WIRE



Adjacency matrix of uva.es



Adjacency matrix of enciclopedia.us.es



Adjacency matrix of UAB.es

Experiments: A full running example

We applied a spectral clustering to the adjacency matrix M of UAB.es

1. Applied $jaccard$ to M to get (sparse) similarity matrix S based on Jaccard measure, where for sets A and B,

$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

2. Found appropriate number k of clusters by the eigengap (or eigenvalue jump) criteria. $(k = 6)$.
3. Run the spectral algorithm with S and k.

Experiments: A full running example

We applied a spectral clustering to the adjacency matrix M of UAB.es

1. Applied $jaccard$ to M to get (sparse) similarity matrix S based on Jaccard measure, where for sets A and B,

$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

2. Found appropriate number k of clusters by the eigengap (or eigenvalue jump) criteria. $(k = 6)$.
3. Run the spectral algorithm with S and k.

Experiments: A full running example

We applied a spectral clustering to the adjacency matrix M of UAB.es

1. Applied $jaccard$ to M to get (sparse) similarity matrix S based on Jaccard measure, where for sets A and B,

$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

2. Found appropriate number k of clusters by the eigengap (or eigenvalue jump) criteria. ($k = 6$).
3. Run the spectral algorithm with S and k.

Experiments: A full running example

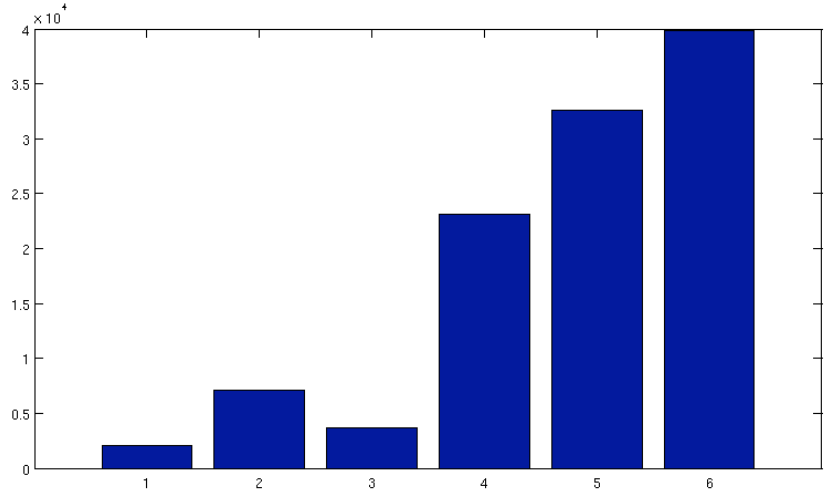
We applied a spectral clustering to the adjacency matrix M of UAB.es

1. Applied $jaccard$ to M to get (sparse) similarity matrix S based on Jaccard measure, where for sets A and B,

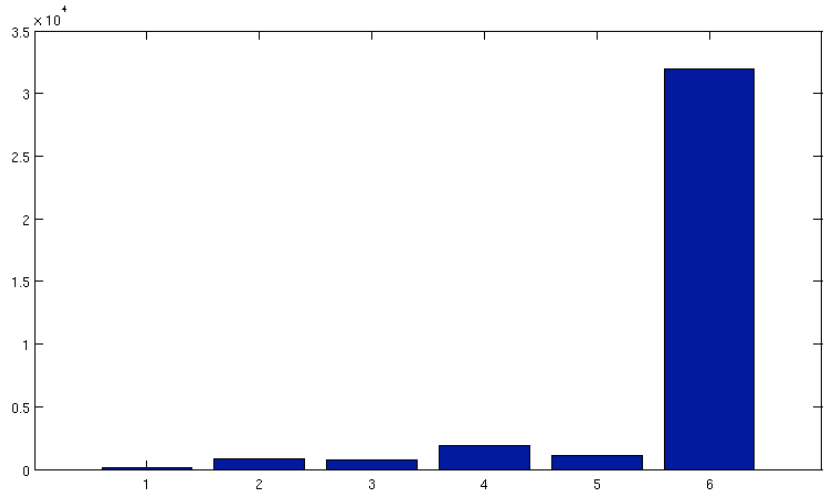
$$jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

2. Found appropriate number k of clusters by the eigengap (or eigenvalue jump) criteria. ($k = 6$).
3. Run the spectral algorithm with S and k.

Results



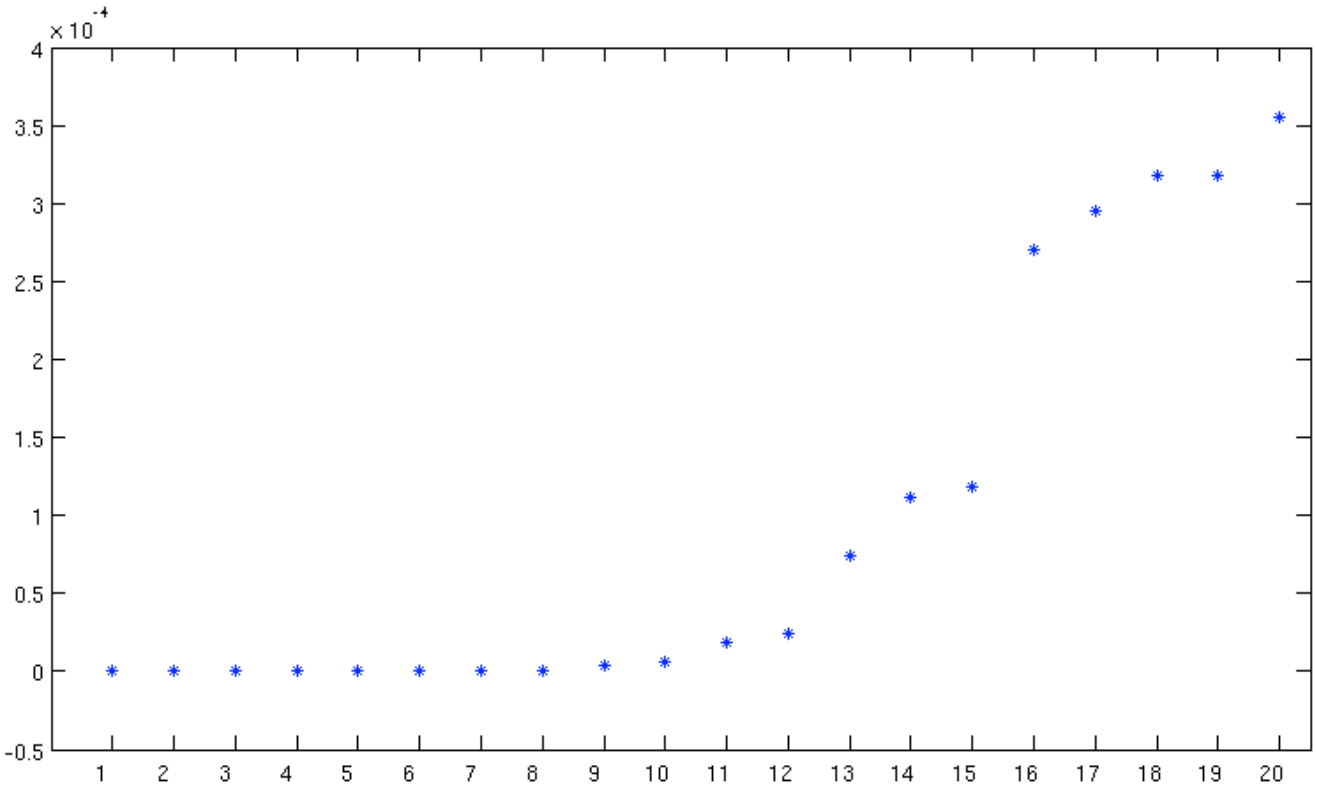
A diagram of volume of clusters



A diagram of number of elements per cluster

On choosing the number of clusters

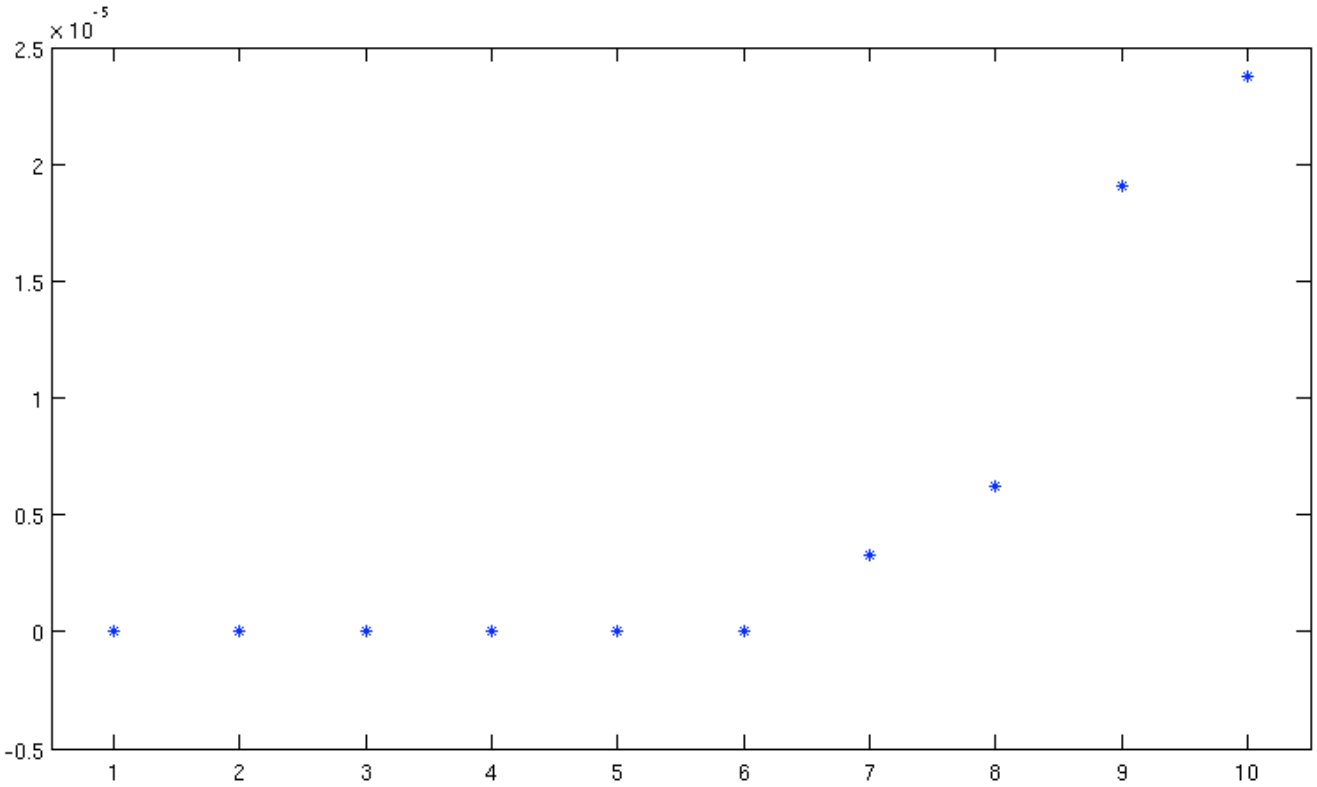
The visual criteria of eigengap can be misleading



20 eigenvalues, one see a jump from 10th to 11th

On choosing the number of clusters

However, increasing the scale



10 eigenvalues, one see a jump from 6th to 7th

Practical Considerations II

II. Why points as rows, centroids as columns

III. From matrices to graphs and back

Practical Considerations IV

IV. k-means performs better starting with eigenvectors