Spectral Clustering

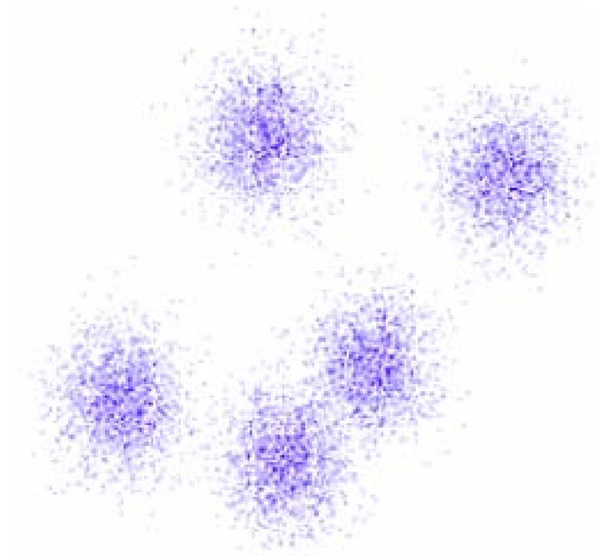
Aarti Singh

Machine Learning 10-701/15-781
Apr 7, 2010

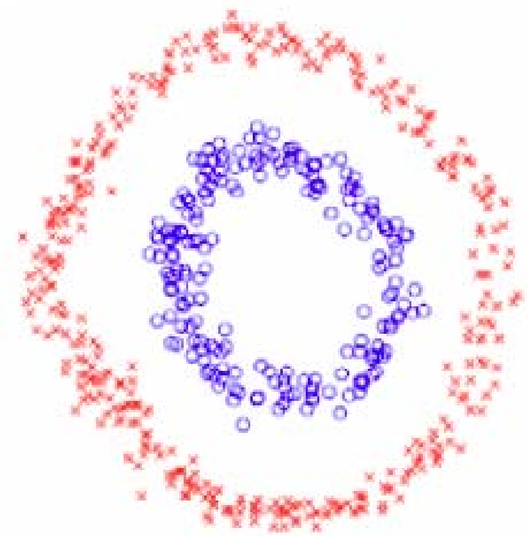
Slides Courtesy: Eric Xing, M. Hein & U.V. Luxburg

Data Clustering

- Two different criteria
  - Compactness, e.g., k-means, mixture models
  - Connectivity, e.g., spectral clustering



Compactness

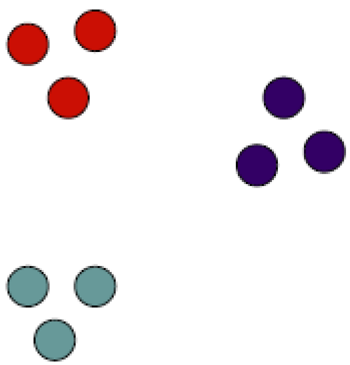


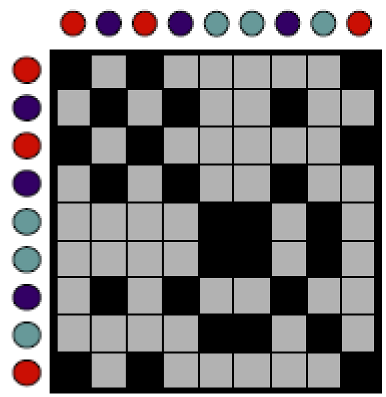
Connectivity

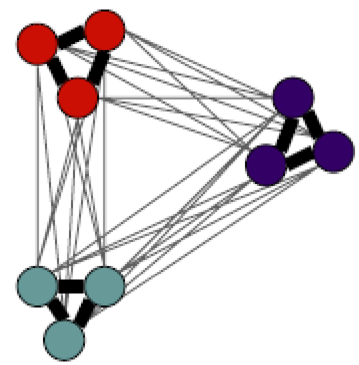
Graph Clustering

Goal: Given data points $X_1, \ldots, X_n$ and similarities $w(X_i,X_j)$, partition the data into groups so that points in a group are similar and points in different groups are dissimilar.

Similarity Graph: G(V,E) V – Vertices (Data points, pixels)
E – Edge if similarity > 0, Edge weights = similarities


Data


Similarities


Similarity graph

Partition the graph so that edges within a group have large weights and edges across groups have small weights.

Similarity graph construction

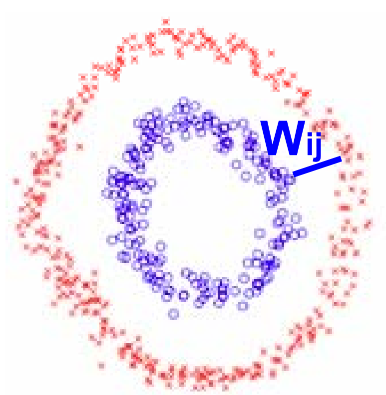
Similarity Graphs: Model local neighborhood relations between data points

G(V,E)   V – Vertices (Data points, pixels)

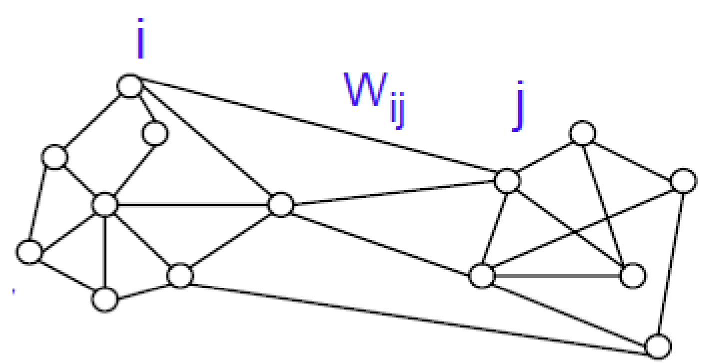
(1) E – Edge if similarity > 0,   Edge weights = similarities $w(x_i,x_j)$

E.g. Gaussian kernel similarity function

$$W_{ij} = e^{\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$ → Controls size of neighborhood



Data clustering



$G = \{V,E\}$

Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

G(V,E)   V – Vertices (Data points, pixels)

(2) E – Edge if ε-NN $||x_i - x_j|| \leq \varepsilon$,   Edge weights = 1   (ε-NN ~ equi-distant)

Similarity graph construction

Similarity Graphs: Model local neighborhood relations between data points

G(V,E)   V – Vertices (Data points, pixels)

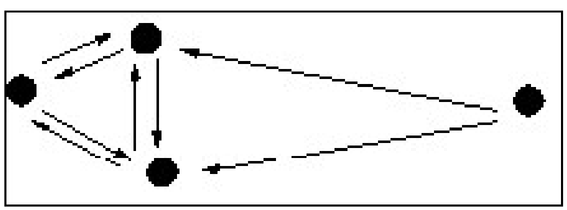
(2) E – Edge if ε-NN $||xi - xj|| \leq \varepsilon$,   Edge weights = 1   (ε-NN ~ equi-distant)

(3) E – Edge if k-NN,   Edge weights = similarities $w(x_i,x_j)$

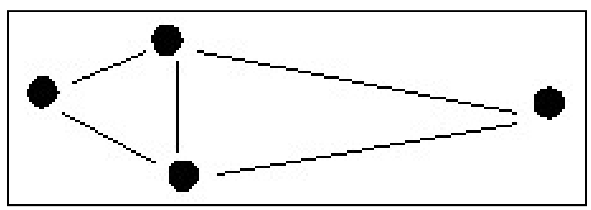
yields directed graph

connect A with B if $A \rightarrow B$ OR $A \leftarrow B$ (symmetric kNN graph)

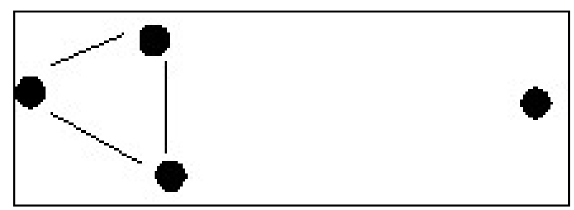
connect A with B if $A \rightarrow B$ AND $A \leftarrow B$ (mutual kNN graph)



Directed nearest neighbors



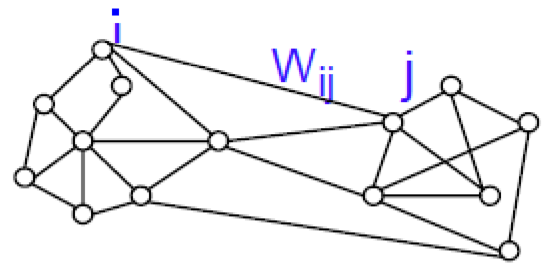
(symmetric) kNN graph



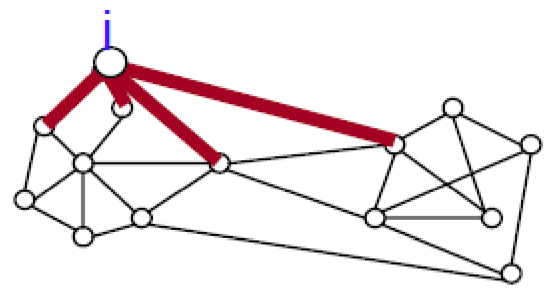
mutual kNN graph

Some Graph Notation

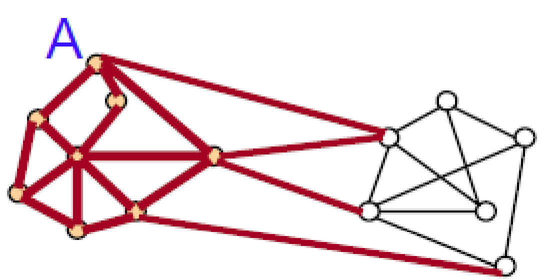
- $W = (w_{ij})$ adjacency matrix of the graph



- $d_i = \sum_j w_{ij}$ degree of a vertex
- $D = diag(d_1, \ldots, d_n)$ degree matrix



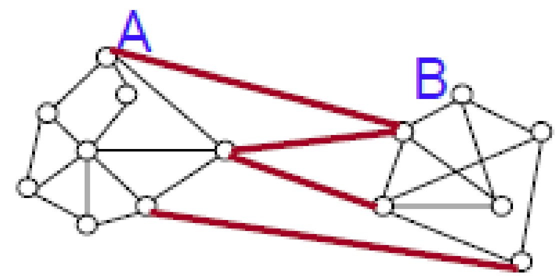
- $|A|$ = number of vertices in A
- $\text{vol}(A) = \sum_{i \in A} d_i$



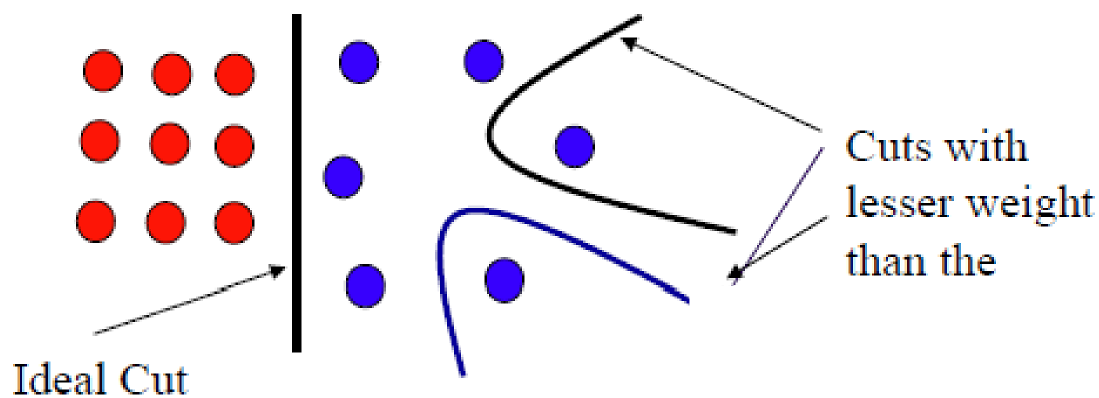
Partitioning a graph into two clusters

Min-cut: Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum.

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



- Easy to solve O(VE) algorithm
- Not satisfactory partition – often isolates vertices



Partitioning a graph into two clusters

Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum & size of A and B are very similar.

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

Balanced Min-cut: $\min_{A,B} \mathrm{cut}(A,B) \text{ s.t. } |A| = |B|$

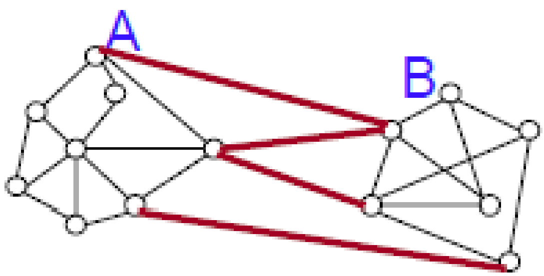
Ratio cut: $\mathrm{RatioCut}(A,B) := \mathrm{cut}(A,B)\left(\frac{1}{|A|} + \frac{1}{|B|}\right)$

Normalized cut: $\mathrm{Ncut}(A,B) := \mathrm{cut}(A,B)\left(\frac{1}{\mathrm{vol}(A)} + \frac{1}{\mathrm{vol}(B)}\right)$

But NP-hard to solve!!

Spectral clustering is a relaxation of these.

Graph cut



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

Choose $f = (f_1, ..., f_n)'$ with $f_i = \begin{cases} 1 & \text{if } X_i \in A \\ -1 & \text{if } X_i \in B \end{cases}$

$$\mathrm{cut}(A,B) = \sum_{i\in A, j\in B} w_{ij} = \frac{1}{4}\sum_{i,j} w_{ij}(f_i - f_j)^2 = f^T(D-W)f$$

RHS = $f^T(D-W)f = f^TDf - f^TWf = \sum_i d_i f_i^2 - \sum_{i,j} f_i f_j w_{ij}$

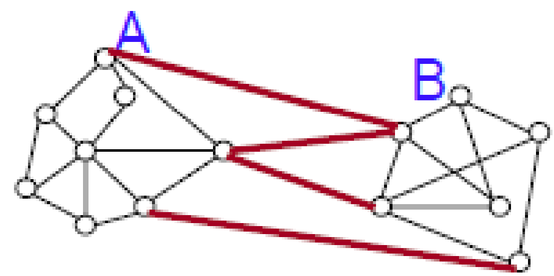
$$= \frac{1}{2}\left(\sum_i \left(\sum_j w_{ij}\right) f_i^2 - 2\sum_{ij} f_i f_j w_{ij} + \sum_j \left(\sum_i w_{ij}\right) f_j^2\right)$$

$$= \frac{1}{2}\sum_{ij} w_{ij}(f_i - f_j)^2$$ **= LHS**

Graph cut and Graph Laplacian

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

$= f^T(D\text{-}W)\, f \quad = f^T L\, f$



$L = D - W$ **Unnormalized Graph Laplacian**

Spectral properties of *L*:

- Smallest eigenvalue of L is 0, corresponding eigenvector is $\mathbb{1}$
- Thus eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

$$L\mathbf{1} = D\mathbf{1} - W\mathbf{1} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} - \begin{bmatrix} \sum_j w_{1j} \\ \sum_j w_{2j} \\ \vdots \\ \sum_j w_{nj} \end{bmatrix} = 0$$

Balanced min-cut

$$\min_{A,B} \text{cut}(A, B) \text{ s.t. } |A| = |B|$$

⇩

$$\min_{f \in \{-1,1\}^n} f^T L f \quad \text{s.t.} \quad f^T 1 = 0$$

(since $\sum f_i = \sum 1_{i \in A} - 1_{i \in B} = 0$)

Above formulation is still NP-Hard, so we relax *f* not to be binary:

$$\min_{f \in R^n} f^T L f \quad \text{s.t.} \quad f^T 1 = 0, \quad f^T f = n$$

$$\min_{f \in R^n} \frac{f^T L f}{f^T f} \quad \text{s.t.} \quad f^T 1 = 0$$

Relaxation of Balanced min-cut

$$\min_{f \in R^n} \frac{f^T L f}{f^T f} \quad \text{s.t.} \quad f^T 1 = 0$$

$$= \lambda_{min}(L)$$ - smallest eigenvalue of L (Rayleigh-Ritz theorem)

If f is eigenvector of L, then

$$\frac{f^T L f}{f^T f} = \frac{f^T \lambda f}{f^T f} = \lambda$$

Recall that smallest eigenvalue of $\boldsymbol{L}$ is $\boldsymbol{0}$ with corresponding eigenvector $\boldsymbol{1}$

But $\boldsymbol{f}$ can't be $\boldsymbol{1}$ according to constraint $\boldsymbol{f^T 1 = 0}$

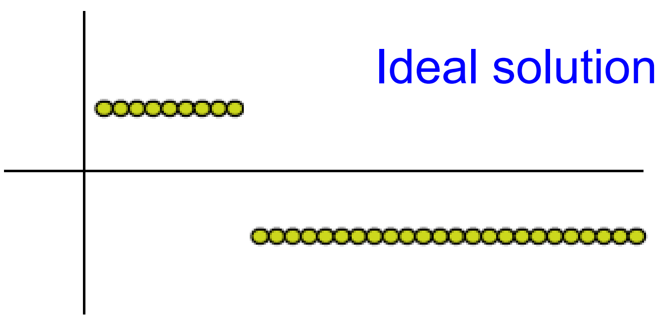
Therefore, solution *f* is the eigenvector of *L* corresponding to second smallest eigenvalue, aka second eigenvector.

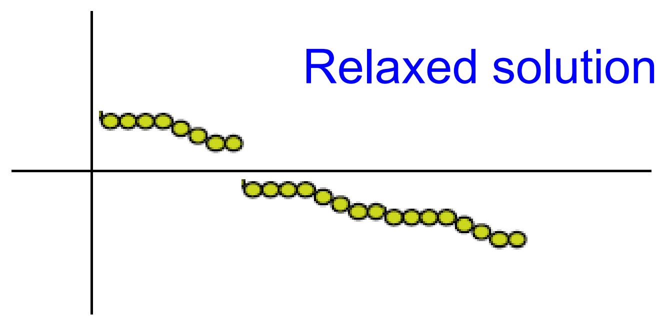
Approximation of Balanced min-cut

$$\min_{A,B} \text{cut}(A, B) \text{ s.t. } |A| = |B|$$

Let *f* be the second eigenvector of the unnormalized graph Laplacian *L.*

Recover binary partition as follows: $i \in A$ if $f_i \geq 0$; $i \in B$ if $f_i < 0$





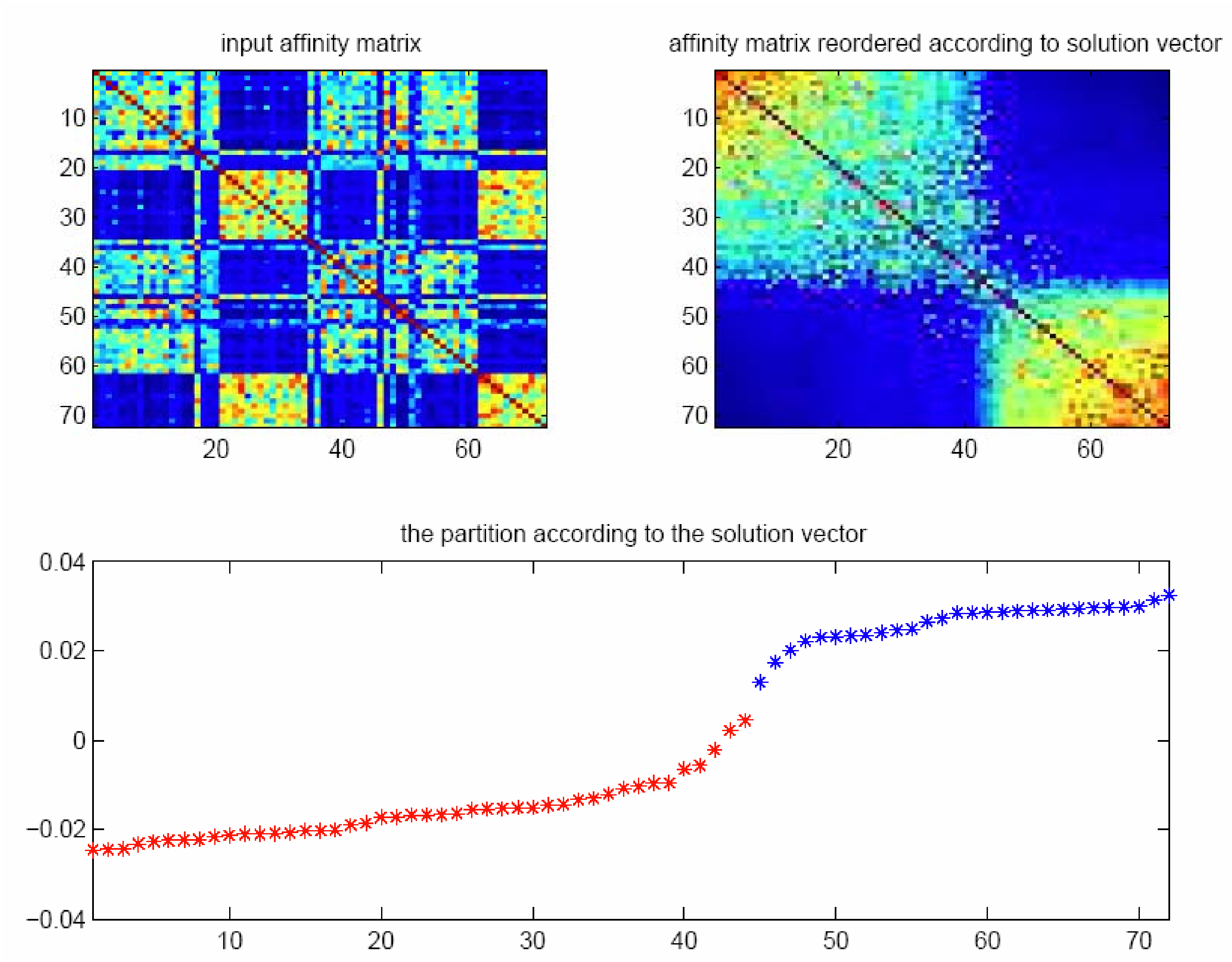
Similar relaxations work for other cut problems:

RatioCut - second eigenvector of unnormalized graph Laplacian $L = D - W$

Normalized cut – second eigenvector of normalized Laplacian $L' = I - D^{-1}W$

Example

Xing et al 2001



How to partition a graph into k clusters?

Spectral Clustering Algorithm

Input: Similarity matrix W, number k of clusters to construct

- Build similarity graph
- Compute the first k eigenvectors $v_1, \ldots, v_k$ of the matrix

$$\begin{cases} L & \text{for unnormalized spectral clustering} \\ L' & \text{for normalized spectral clustering} \end{cases}$$

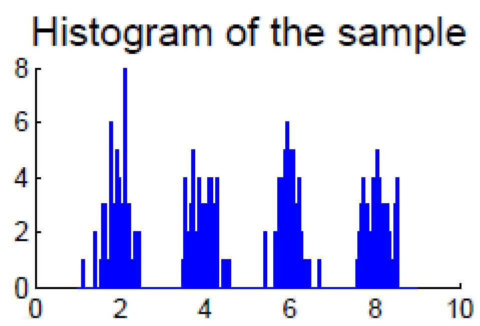
- Build the matrix $V \in \mathbb{R}^{n \times k}$ with the eigenvectors as columns
- Interpret the rows of V as new data points $Z_i \in \mathbb{R}^k$

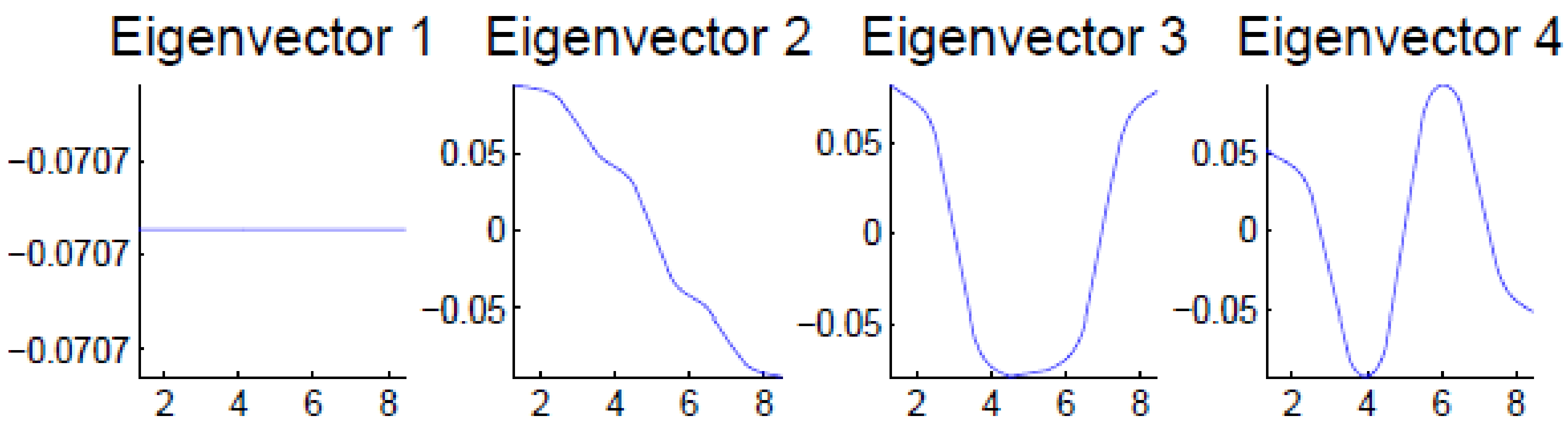
	v_1	v_2	v_3
Z_1	v_{11}	v_{12}	v_{13}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
Z_n	v_{n1}	v_{n2}	v_{n3}

Dimensionality Reduction
n x n $\rightarrow$ n x k

- Cluster the points Z_i with the k-means algorithm in $\mathbb{R}^k$.

Eigenvectors of Graph Laplacian





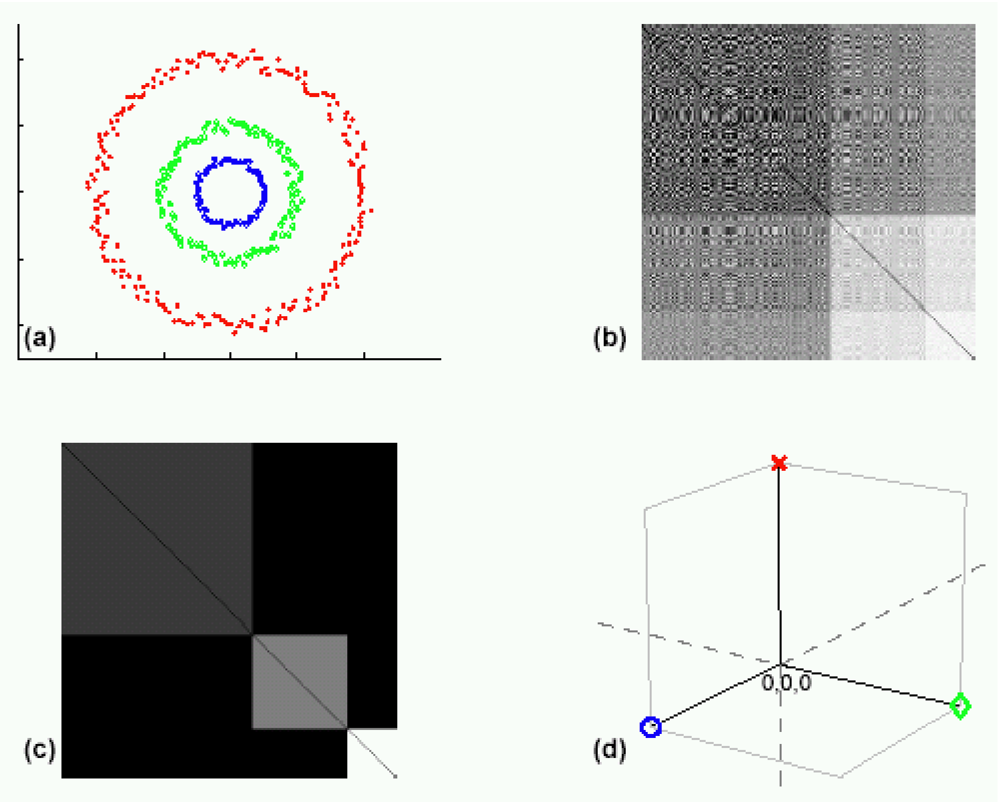
- 1st Eigenvector is the all ones vector ***1***
- 2nd Eigenvector thresholded at 0 separates first two clusters from last two
- k-means clustering of the 4 eigenvectors identifies all clusters

Why does it work?

Data are projected into a lower-dimensional space (the spectral/eigenvector domain) where they are easily separable, say using k-means.

Original data

Similarity between original data



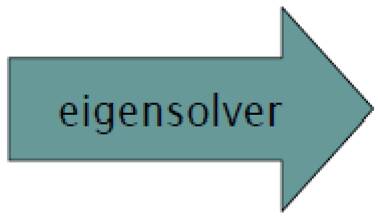
Similarity between projected data

Projected data

Why does it work?

- Block matrices have block eigenvectors:

1	1	0	0
1	1	0	0
0	0	1	1
0	0	1	1



$\lambda_1 = 2$

.71
.71
0
0

$\lambda_2 = 2$

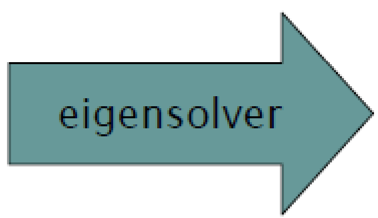
0
0
.71
.71

$\lambda_3 = 0$

$\lambda_4 = 0$

- Near-block matrices have near-block eigenvectors:

1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1



$\lambda_1 = 2.02$

.71
.69
.14
0

$\lambda_2 = 2.02$

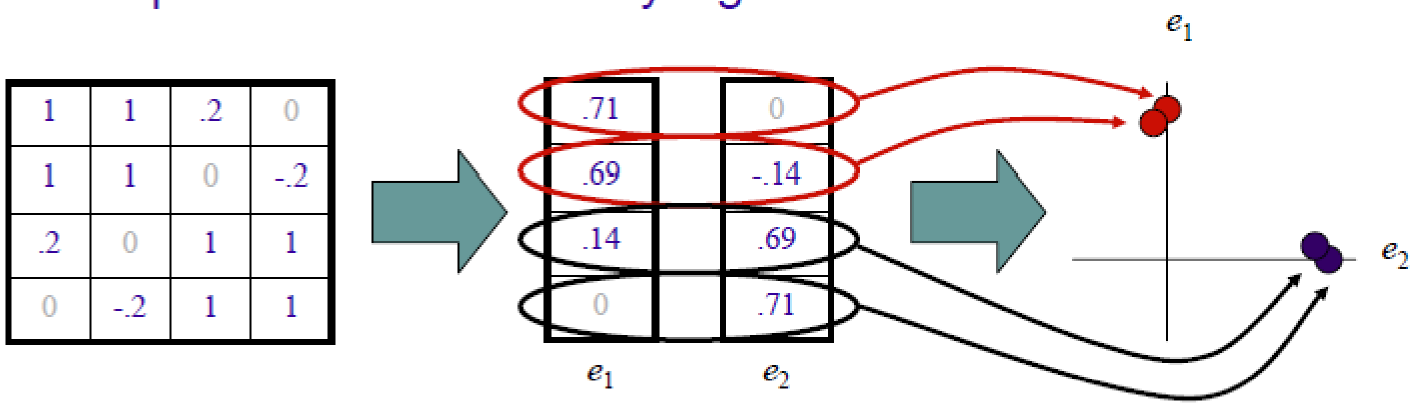
0
-.14
.69
.71

$\lambda_3 = -0.02$

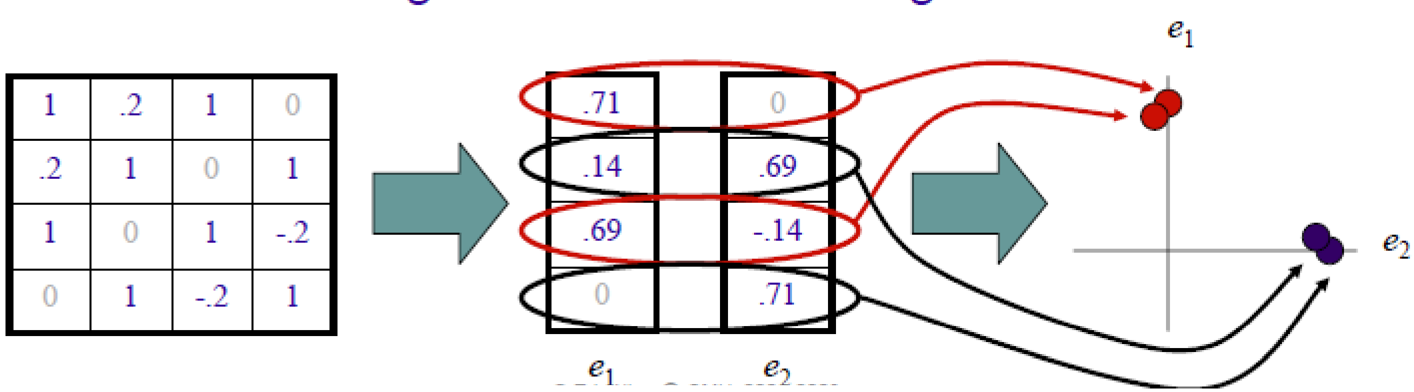
$\lambda_4 = -0.02$

Why does it work?

- Can put items into blocks by eigenvectors:

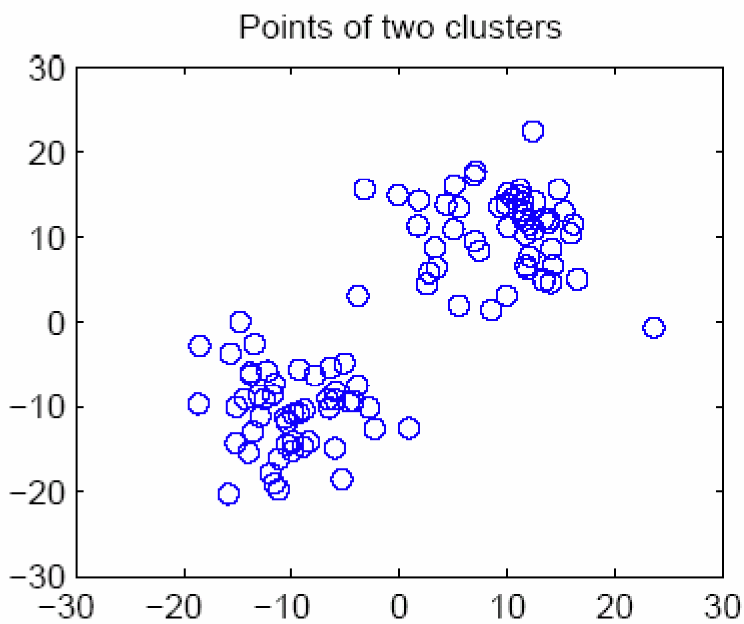


- Clusters clear regardless of row ordering:

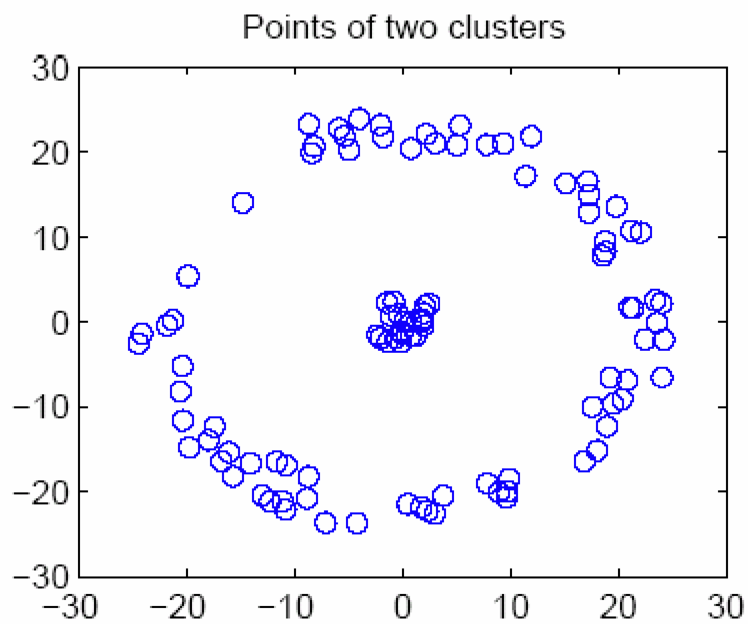


k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



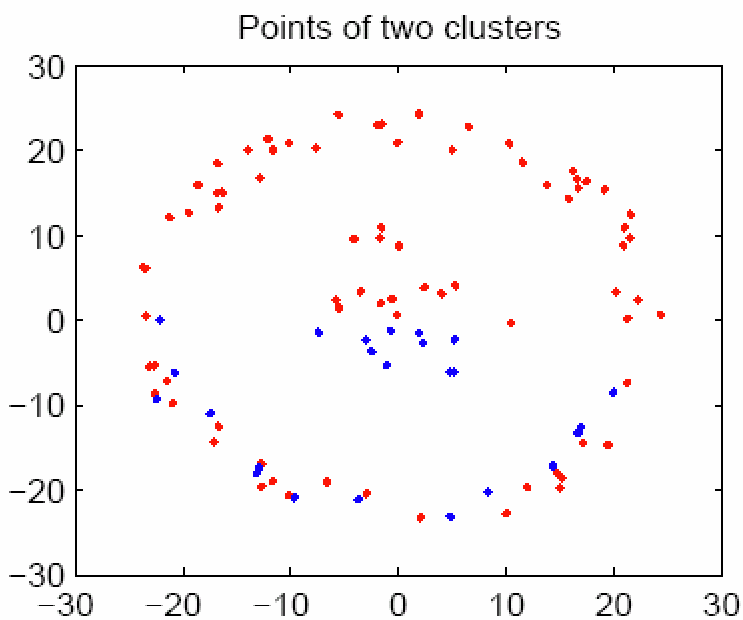
Both perform same



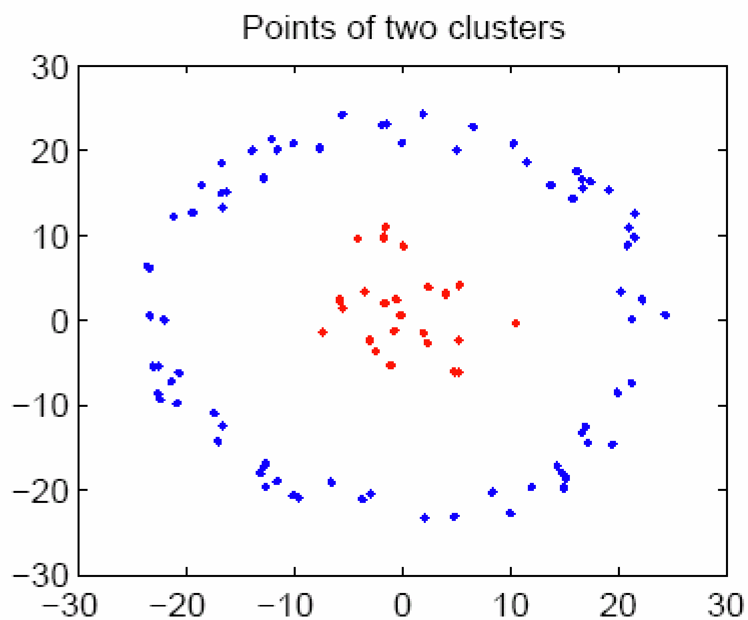
Spectral clustering is superior

k-means vs Spectral clustering

Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



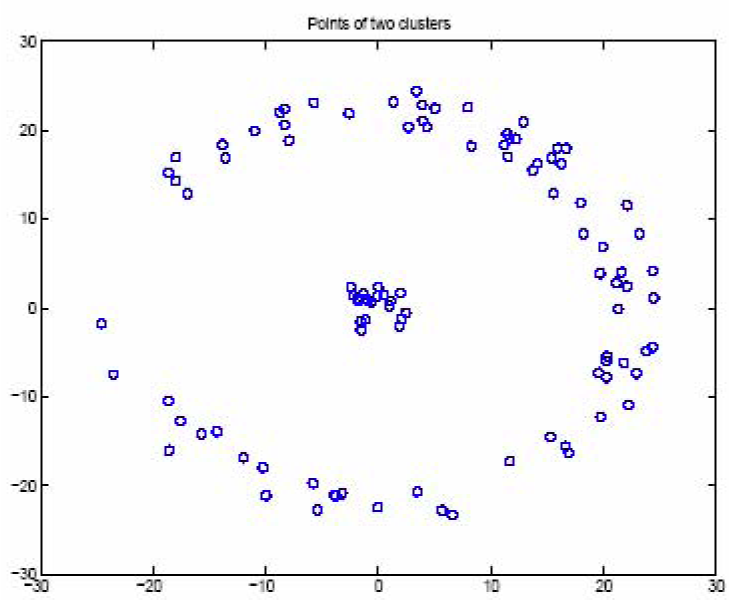
k-means output



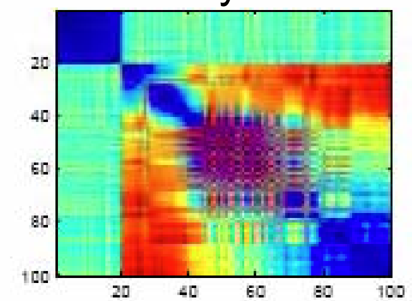
Spectral clustering output

k-means vs Spectral clustering

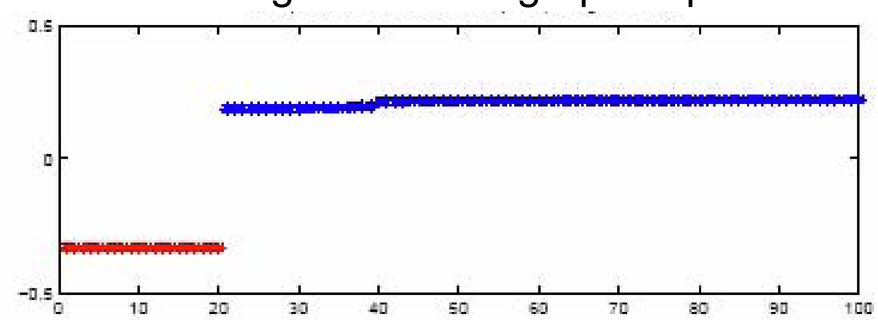
Applying k-means to laplacian eigenvectors allows us to find cluster with non-convex boundaries.



Similarity matrix

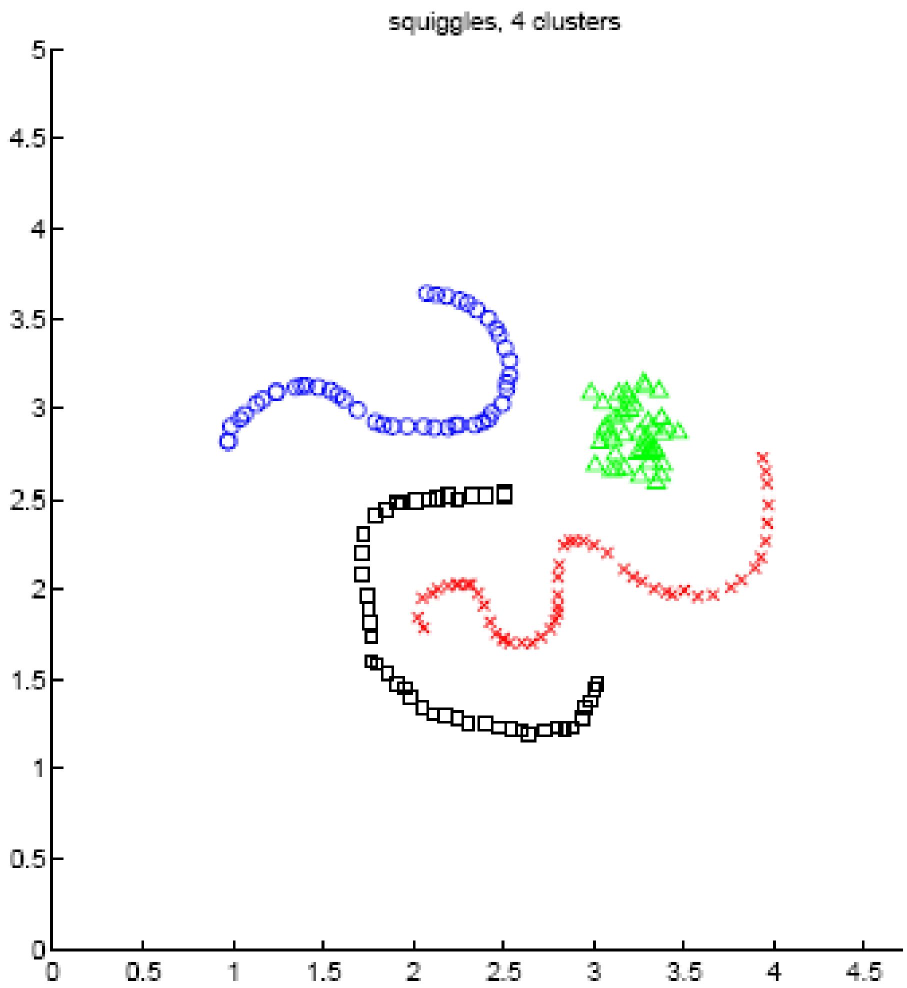


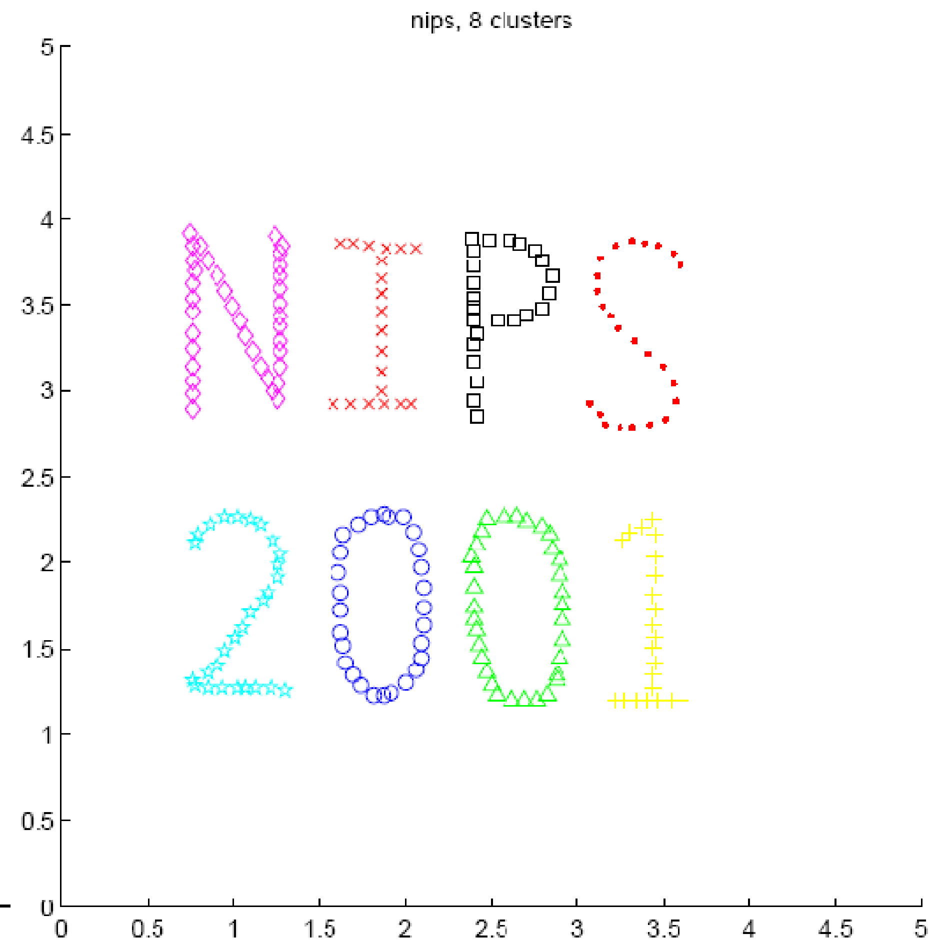
Second eigenvector of graph Laplacian



Examples

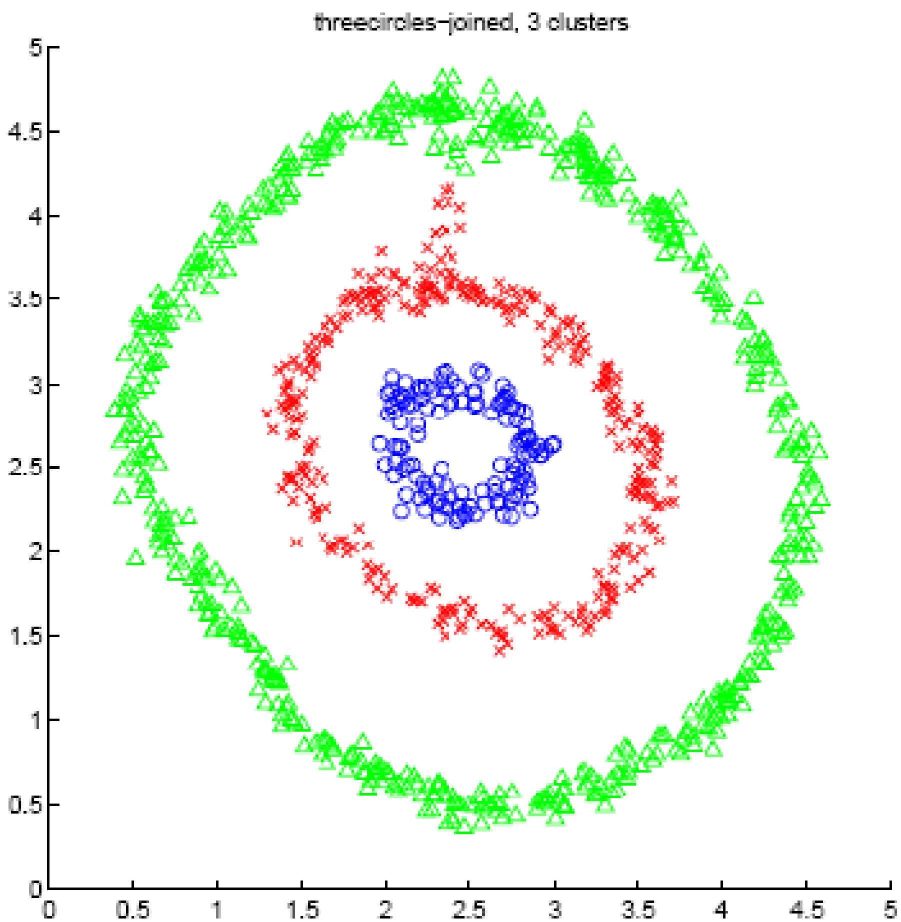
Ng et al 2001

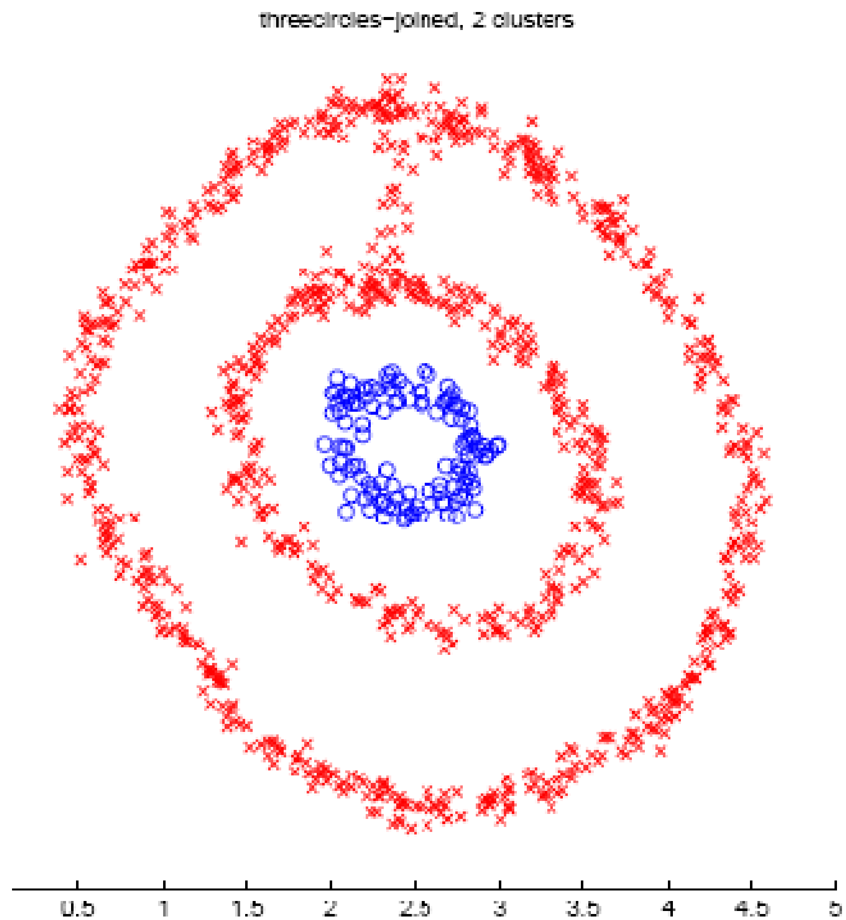




Examples (Choice of k)

Ng et al 2001



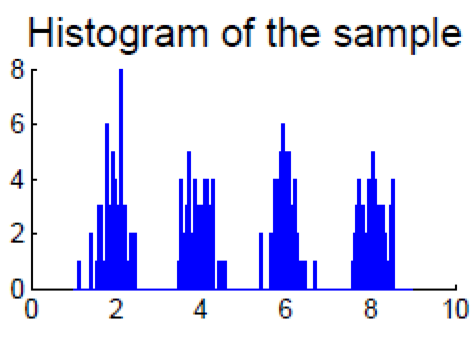


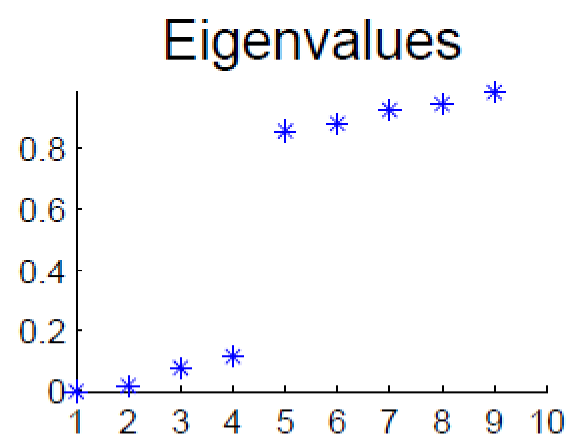
Some Issues

- Choice of number of clusters k

  Most stable clustering is usually given by the value of k that maximizes the eigengap (difference between consecutive eigenvalues)

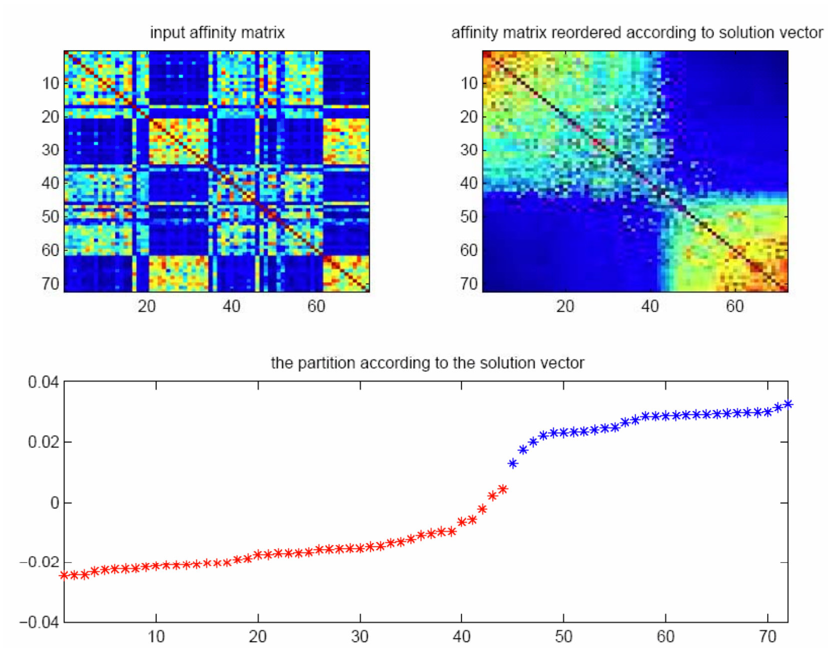
$$\Delta_k = \left| \lambda_k - \lambda_{k-1} \right|$$



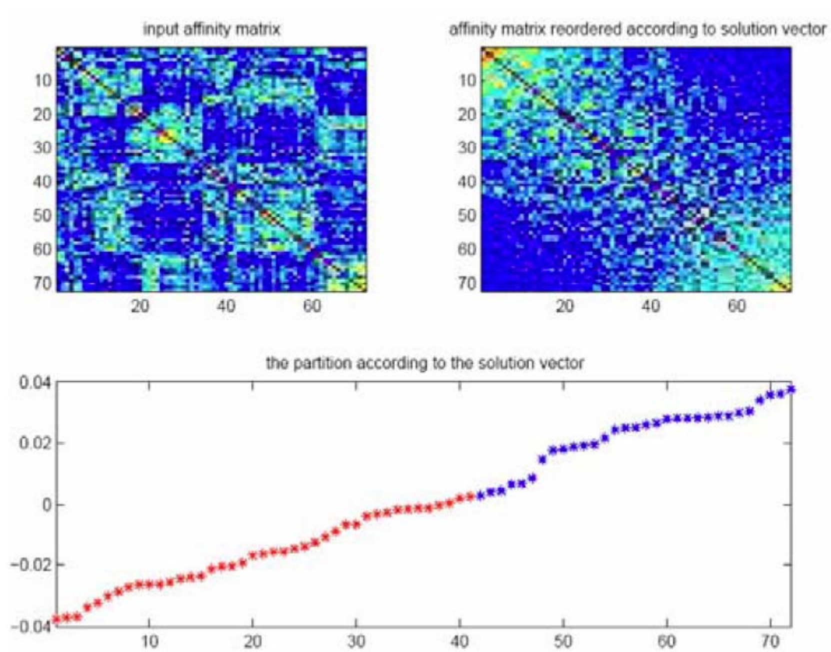


Some Issues

- Choice of number of clusters k
- Choice of similarity
  - choice of kernel
  - for Gaussian kernels, choice of σ



Good similarity measure



Poor similarity measure

Some Issues

- Choice of number of clusters k
- Choice of similarity
  - choice of kernel
  - for Gaussian kernels, choice of σ
- Choice of clustering method – k-way vs. recursive bipartite

Spectral clustering summary

- Algorithms that cluster points using eigenvectors of matrices derived from the data
- Useful in hard non-convex clustering problems
- Obtain data representation in the low-dimensional space that can be easily clustered
- Variety of methods that use eigenvectors of unnormalized or normalized Laplacian, different, how to derive clusters from eigenvectors, k-way vs repeated 2-way
- Empirically very successful

Comparison Chart

	Decision Trees	K-NN	Naïve Bayes	Logistic regression	SVM	Boosting	Neural Nwks	HMM	Bayes Net
Gen/Disc									
Loss functions									
Decision boundary									
Output									
Assumptions									
Structured version									
Algorithm									
Convergence									

Loss functions