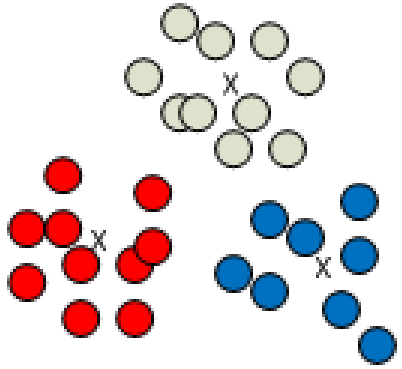
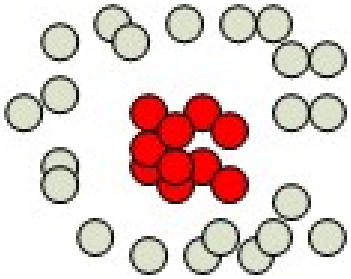
# Graph-based Clustering

# Traditional k-means – Assumptions

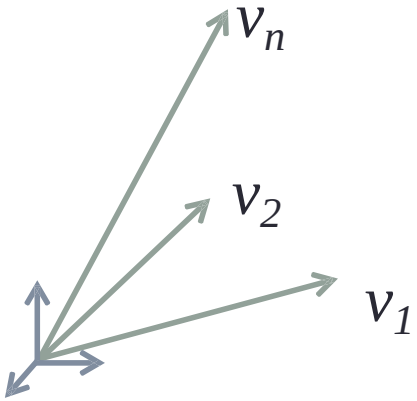
- Data are distributed around centroids

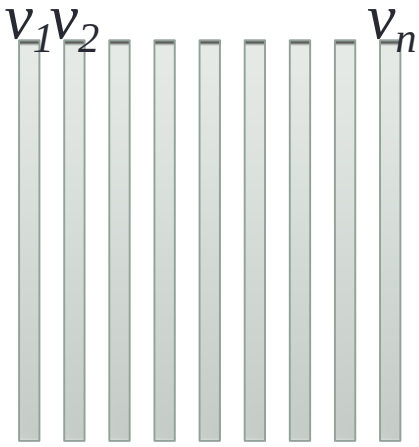


What about this?



- Data instances can be represented in a vector space



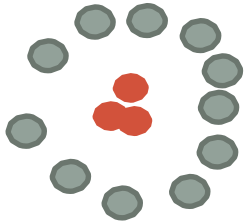




What about complex data structures?

# Similarity-based Clustering

**Idea**

- Calculate similarity between pairs of data instances
- Use these similarity measures to cluster data instances into groups



**Why**

- Capture non-linearity in the data
- Work on complex data structures (text, strings, trees...etc)

**Two Algorithms**

- Graph-based Clustering
- Kernel-based Clustering

# Clustering as Graph Partitioning

- Construct an undirected graph
  - G=(V, E)
  - V: data instances
  - E: weighted edges between (similar) instances
- **Goal**: Divide graph into sub-graphs
  - sum of weights on edges between each two sub-graphs is minimum
  - sum of weights on edges within sub-graphs is maximum

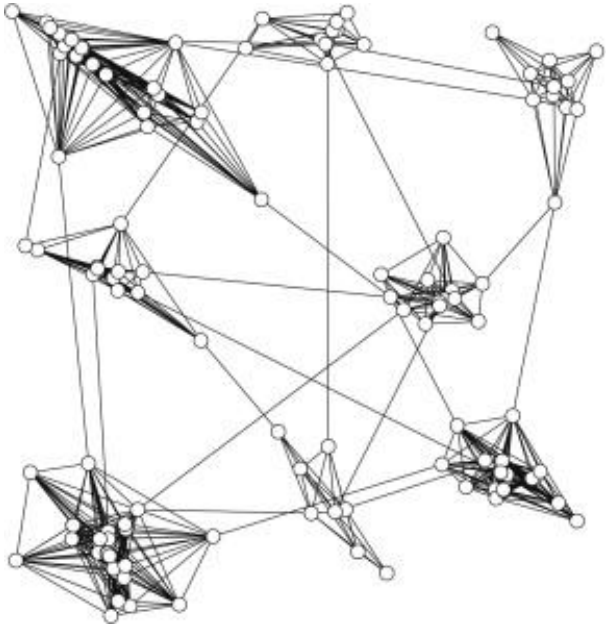


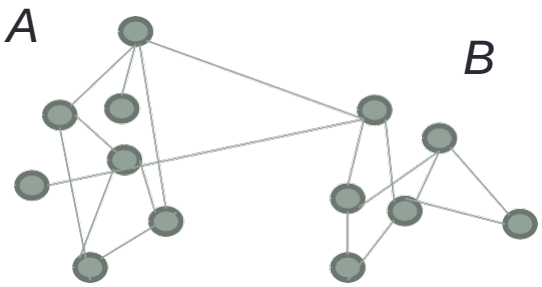
Figure from [Schaeffer 2007]

# Graph Cut

- Graph cut between two sub-graphs:

  = sum of weights of the edges connecting the two-sub-graphs

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

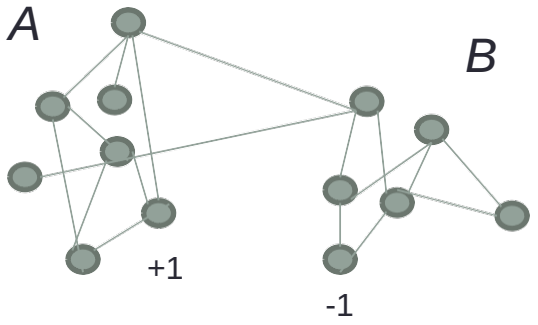



- **Goal**: Find two partitions such that the graph cut between them are minimum

# How to minimize cut(A,B)?

- Naïve algorithm: Enumerate all possible cuts, find the minimum ➔ NP-hard
- Solution: Define a cluster indicator vector *f*

$$f_i = \begin{cases} +1, & i \in A \\ -1, & i \in B \end{cases}$$



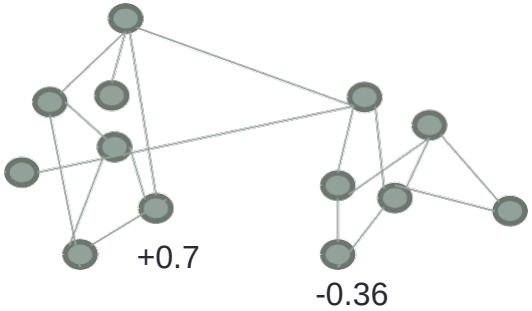

- It can be shown that:

$$\sum_{i,j} w_{ij} \left(f_i - f_j\right)^2 = 8\, cut(A,B)$$

# How to minimize cut(A,B)?

- Find a continuous vector *f* which minimizes

$$\sum_{i,j} w_{ij} \left(f_i - f_j\right)^2$$




- Find a threshold *T*, and divide data instances into two clusters as follows:

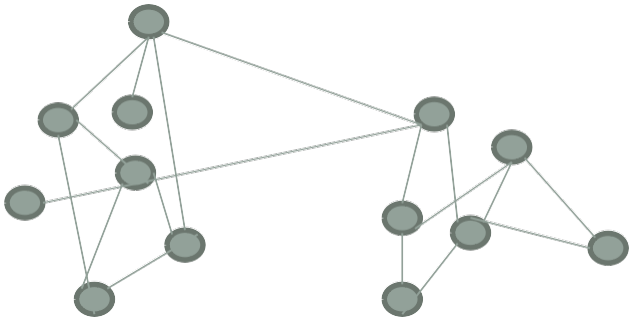
$$cluster(i) = \begin{cases} A, & f_i \geq T \\ B, & f_i \leq T \end{cases}$$

- Note: We have to add constraints on *f* to avoid the simple solution *f*=0 (e.g., maximize the length of *f*)

# Definitions

- W: Similarity matrix
- D: Degree matrix (diagonal)

$$D_{ii} = d_i = \sum_j w_{ij}$$



- L: Laplacian matrix

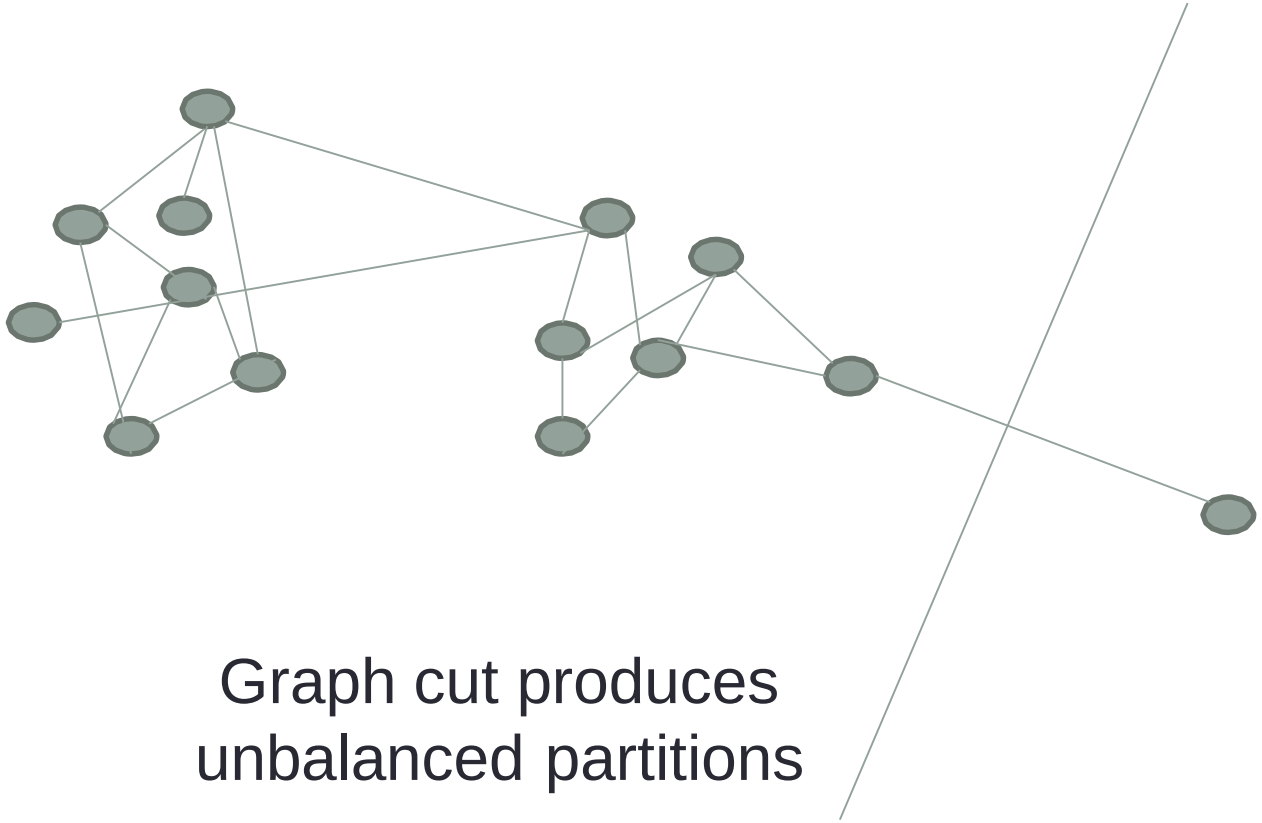
$$L = D - W$$

- We can show that for any *f*:

$$f^T L f = \frac{1}{2} \sum_{i,j} w_{ij} \left(f_i - f_j\right)^2$$

# Example

- Where is the minimum cut?



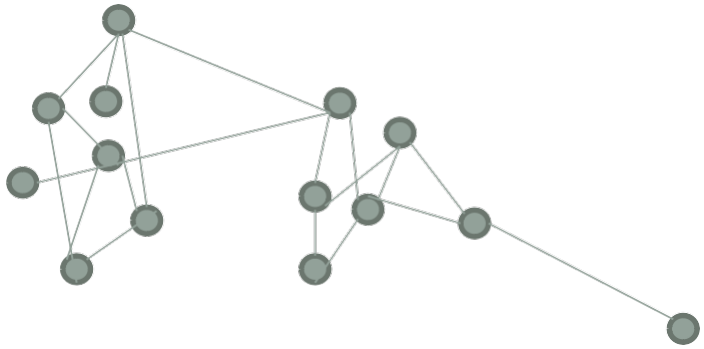
Graph cut produces
unbalanced partitions

# Ratio Cut

$$ratiocut(A,B) = \frac{cut(A,B)}{n_A} + \frac{cut(A,B)}{n_B}$$

- Define a cluster indicator vector *f*

$$f_i = \begin{cases} +\sqrt{\frac{n_B}{n n_A}}, & i \in A \\ -\sqrt{\frac{n_A}{n n_B}}, & i \in B \end{cases}$$



- We can show that:

$$\sum_{i,j} w_{ij} \left(f_i - f_j\right)^2 = 2 f^T L f = 2\, ratiocut(A,B)$$

# Ratio Cut (Cont.)

- Minimizing the ratio cut is equivalent to

$$\min \; f^T L f$$

s.t.

$$f^T f = 1, \boxed{f^T e = 0}$$

to enforce balanced partitions

- Solution: *f* is the 2nd smallest eigenvector of *L*

$$Lf = \lambda f$$

# Normalized Cuts

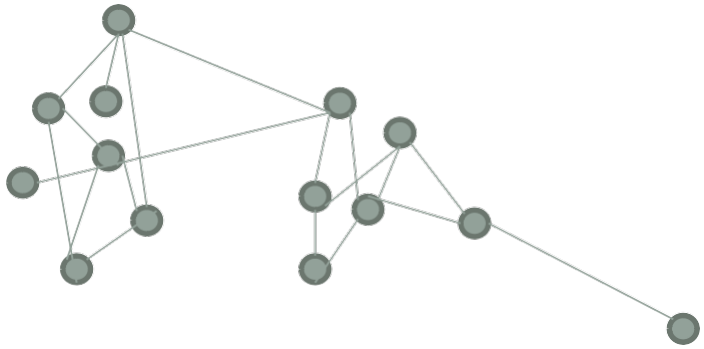
$$d_B = \sum_{i \in B} d_i \qquad d_A = \sum_{i \in A} d_i$$

$$d = \sum_i d_i$$

$$Ncut(A,B) = \frac{cut(A,B)}{d_A} + \frac{cut(A,B)}{d_B}$$

- Define a cluster indicator vector *f*

$$f_i = \begin{cases} +\sqrt{\frac{d_B}{d d_A}}, & i \in A \\ -\sqrt{\frac{d_A}{d d_B}}, & i \in B \end{cases}$$



- We can show that (see Proof 4):

$$\sum_{i,j} w_{ij} \left(f_i - f_j\right)^2 = 2\, Ncut(A,B)$$

# Normalized Cut (Cont.)

- Minimizing the normalized cut is equivalent to

$$\min \; f^T L f$$

s.t. $$f^T D f = 1, \; f^T D e = 0$$

- Solution: *f* is the 2nd smallest eigenvector of the generalized eigenproblem:

$$Lf = \lambda D f$$

- or that of the normalized Laplacian matrix

$$D^{-1/2} L D^{-1/2} f = \lambda f$$

# Example

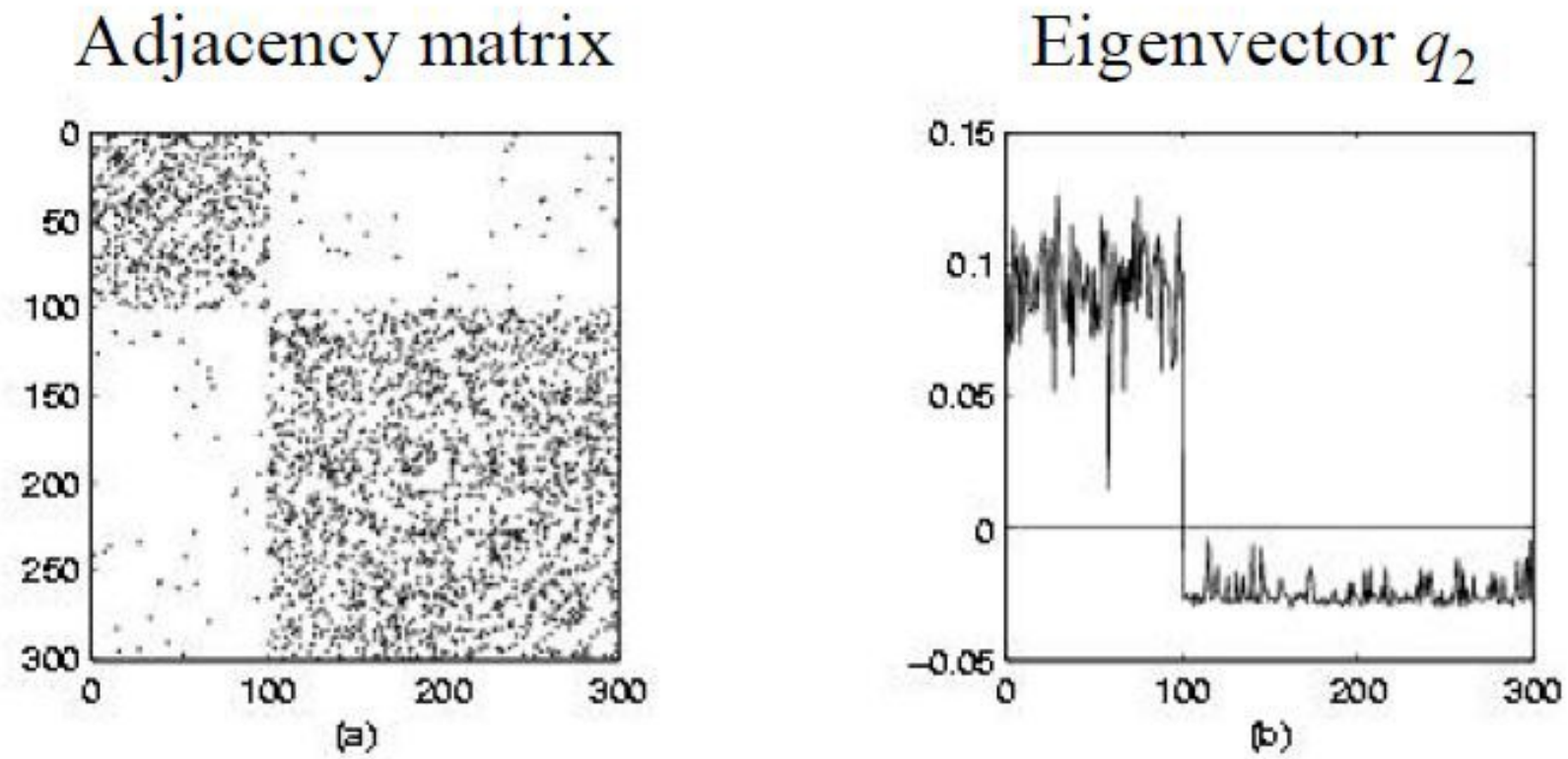



Figure from
[Ding et al 2008]

# SC Algorithm for *k* Clusters

- Calculate the normalized Laplacian matrix *LN*

$$D^{-1/2}LD^{-1/2}$$

- Find the 2nd smallest k eigenvectors of *LN*

$$V = [v_1, v_2, ...v_k]$$

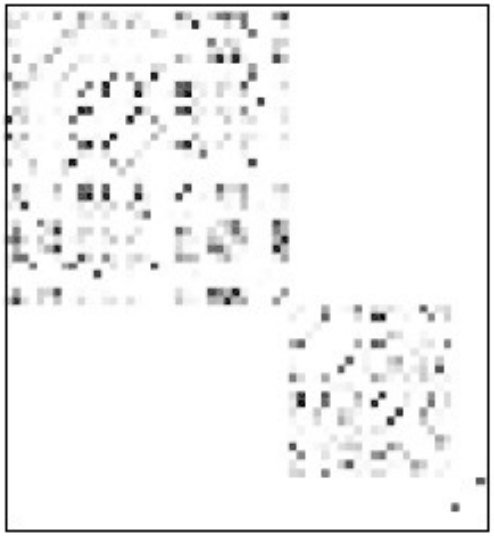
- Represent data instances in the space of *V*, and normalize data vectors

$$U_{ij} = \frac{V_{ij}}{\sqrt{\sum_l V_{il}^2}}$$

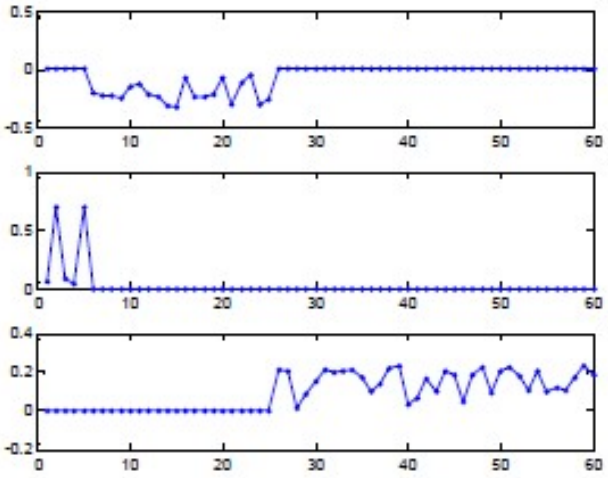
- Run k-means on the **rows** of *U*

# SC Algorithm for *k* Clusters

$W; A$



$V = [v_1, v_2, v_3]$



$U = [u_1, u_2, u_3]$

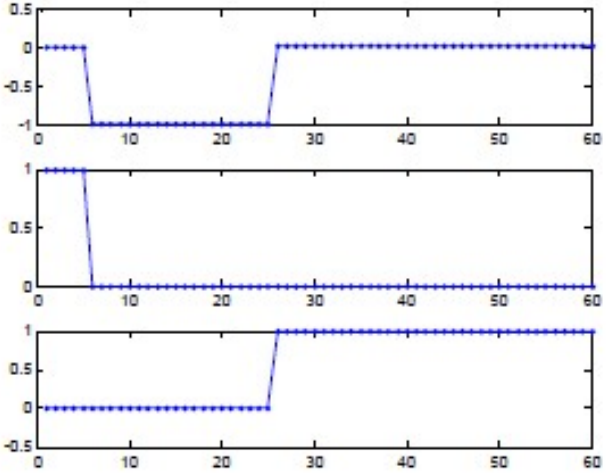


Figure from
[Azran 2007]

# How to construct graphs?

- Similarity function
  - Inner-products or cosine similarities

$$w_{ij} = x_i^T x_j$$

  - Gaussian similarity function

$$w_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right)$$

  - Application-specific similarity function
- Connectivity
  - Fully-connected graph
  - k-Nearest Neighbor Graph
  - ε-Nearest Neighbor Graph

# What SC can do?

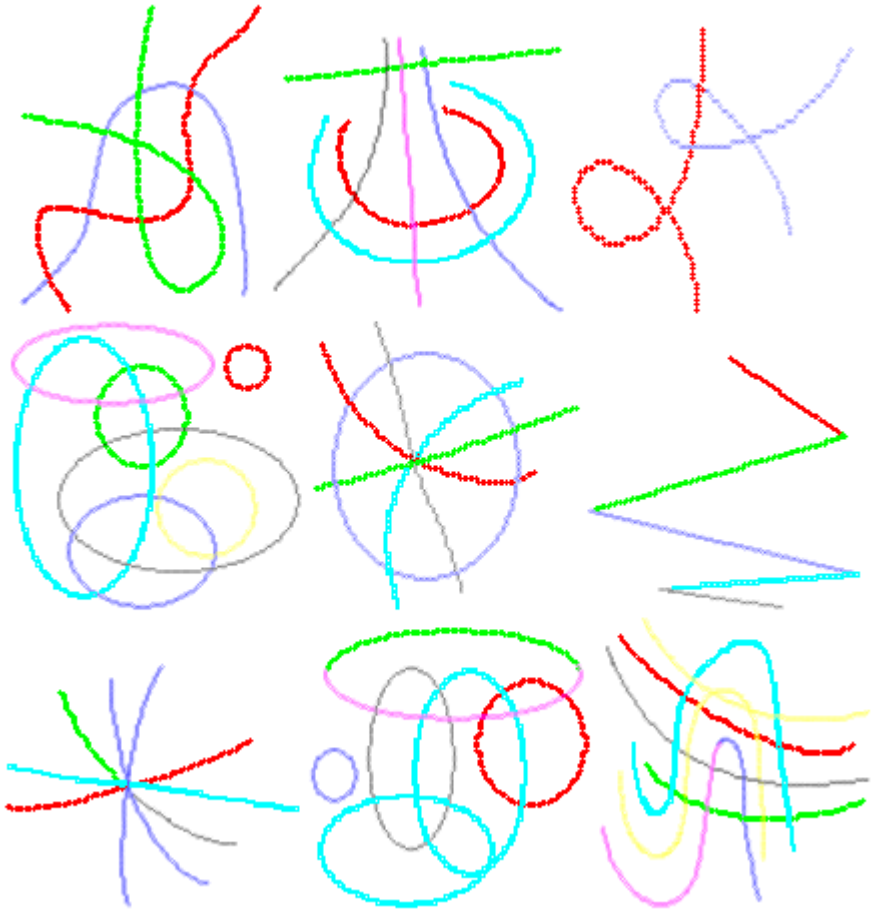


Figure from
[Bach & Jordan 2006]

# Image Segmentation

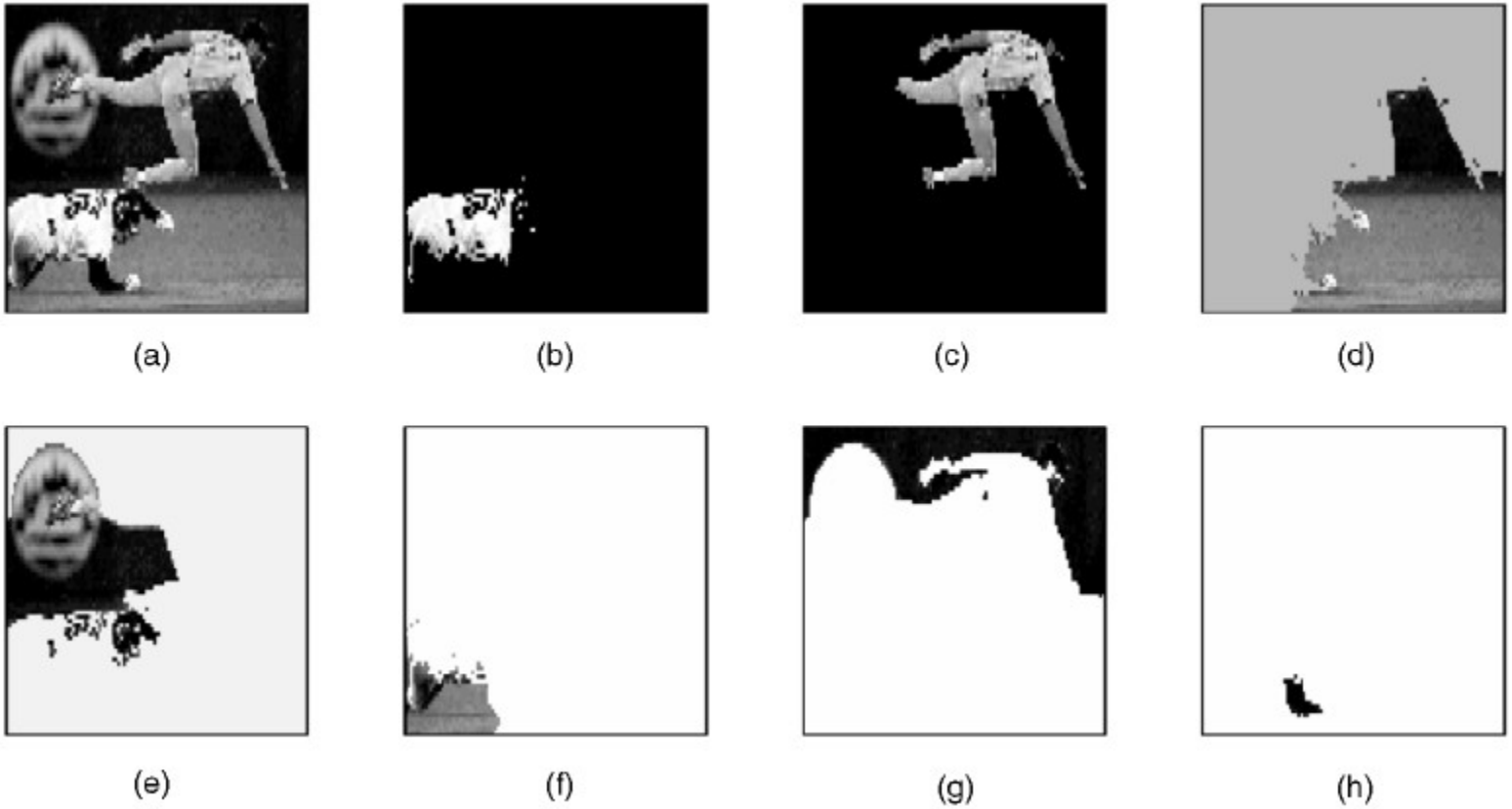
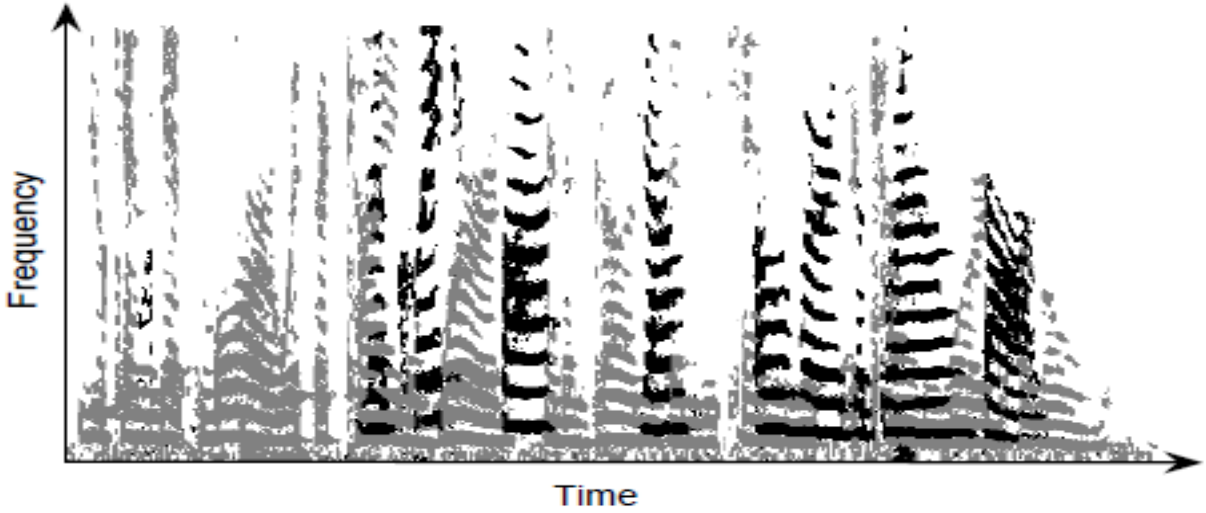



Figure from
[Shi & Malik 2000]

# Speech Separation

Truth




SC

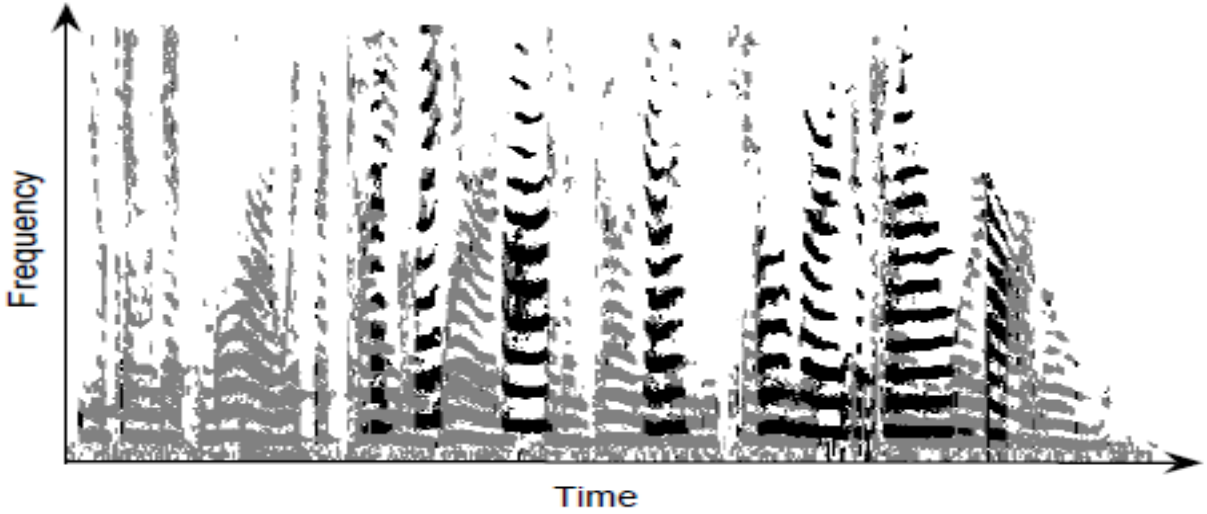



Figure from
[Bach & Jordan 2006]

# Finding Communities in Graphs

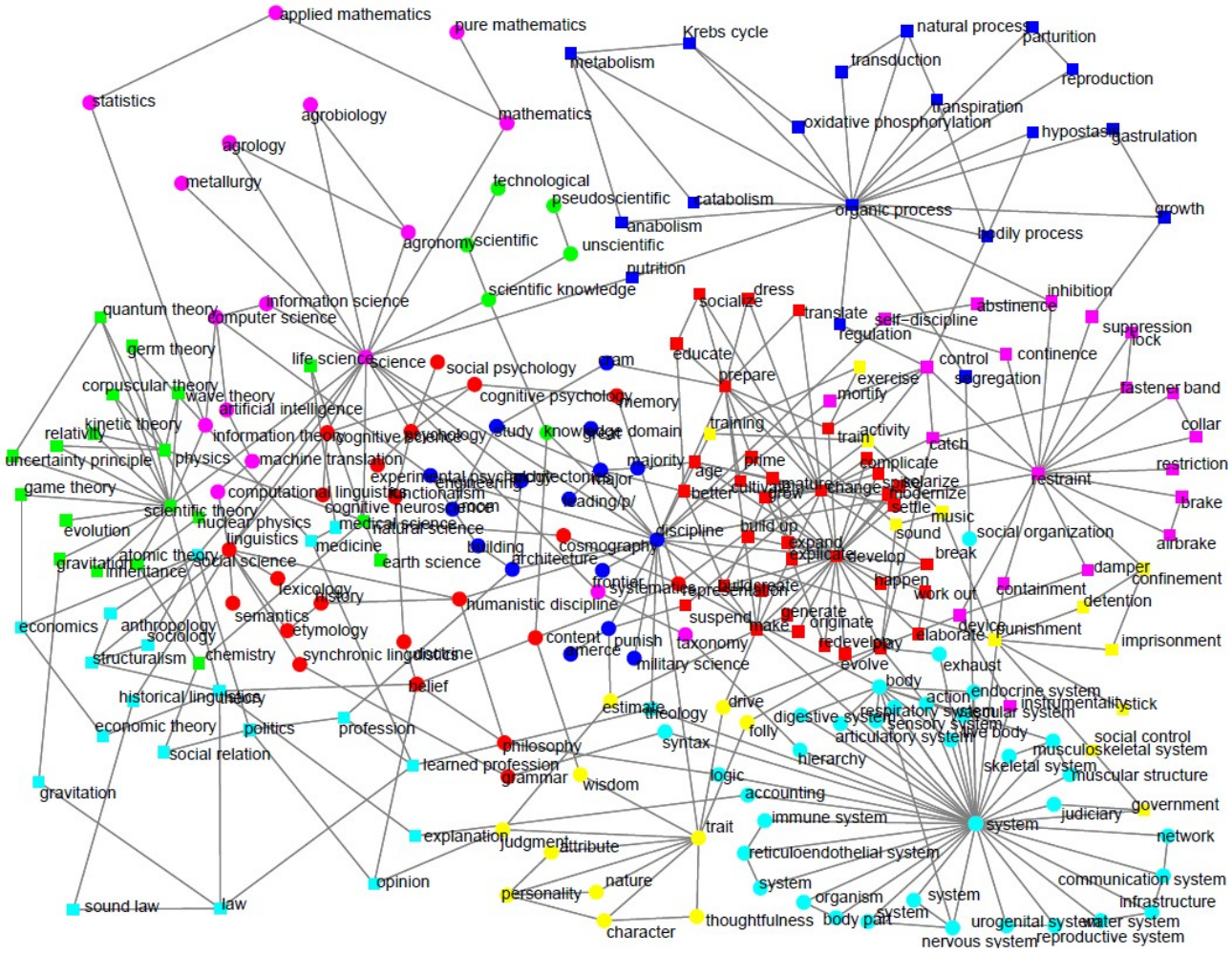



Figure from
[White & Smyth 2005]

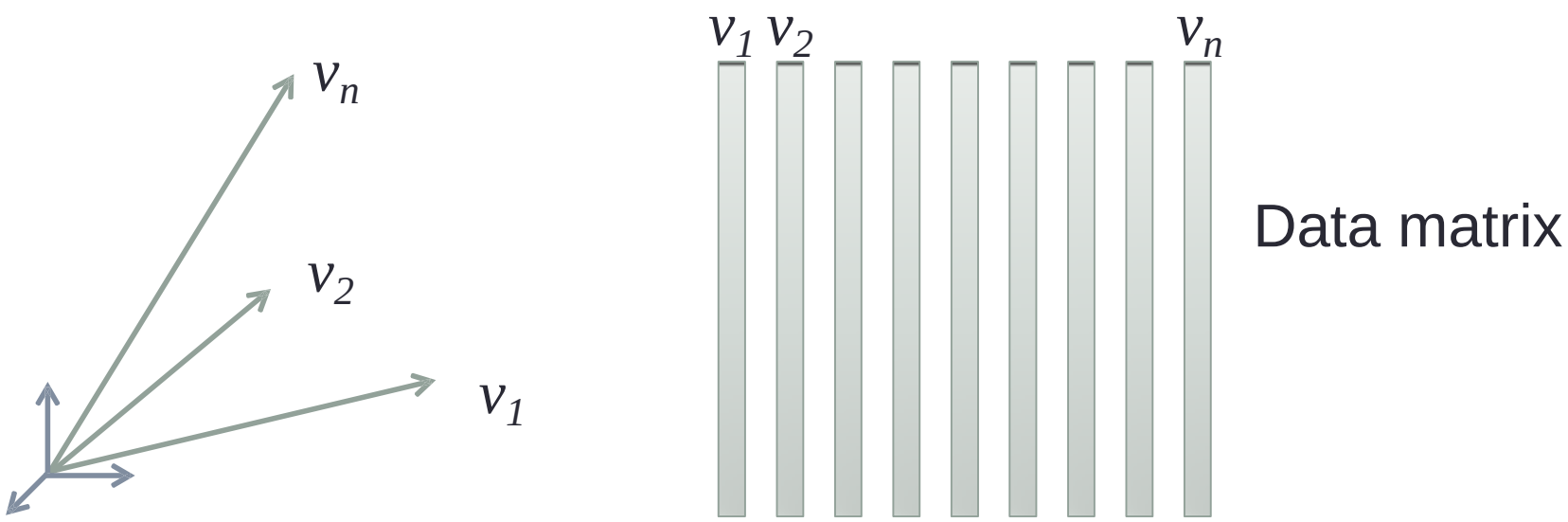
# Problems

- Calculating the eigenvectors of the similarity matrix between data instances is computationally expensive ($O(n^3)$) and memory inefficient ($O(n^2)$).
- Solutions:
  - Sparsify the similarity matrix
  - Approximate the eigenvectors of the Laplacian matrix
  - Landmark-based spectral clustering
  - Parallel and distributed implementations
  - ….

# Kernel-based Clustering

# K-means

- Works on data points (or vectors)




- What if the data cannot be represented as vectors?

# Back to k-means

- Initialize: Random partitions
- Repeat
  - Update Step: Calculate centroids

$$\mu_i = \frac{1}{n_i} \sum_{a \in \pi_i} v_a$$

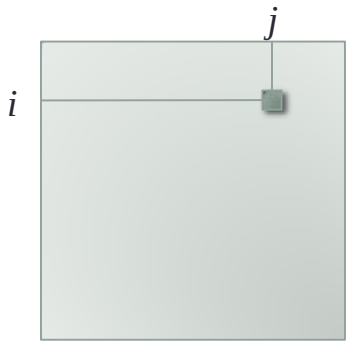
  - Assignment Step: Assign data points to clusters

$$c(j) = \arg\min_i \left\| v_j - \mu_i \right\|^2$$

# Kernel Matrices

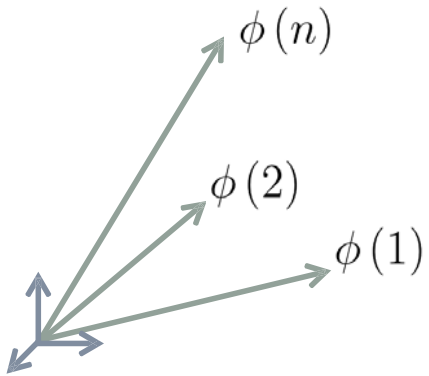
- Can we estimate how close two vectors are?

Kernel matrix




$$k(i,j) = \phi(i)^T \phi(j)$$

Implicit vector space




- Any algorithm that depends on inner-products between vectors can be “kernelized”.

# What about k-means?

- Initialize: Random partitions
- Repeat
  - Update Step: Calculate centroids

$$\mu_i = \frac{1}{n_i} \sum_{a \in \pi_i} v_a$$

  - Assignment Step: Assign data points to clusters

$$c(j,i) = \arg\min_i \|v_j - \mu_i\|^2$$

- Problem: Cannot calculate vectors for data instances and centroids using kernels, need to express everything in terms of dot product.

# From Distances to Inner-Products

$$\|v_a - v_b\|^2 = \|v_a\|^2 + \|v_a\|^2 - 2v_a^T v_b$$

$$\|v_a\|^2 = v_a^T v_a \qquad\qquad 2v_a^T v_b \rightarrow K_{ab}$$

$$v_a^T v_a \rightarrow K_{aa}$$

$$\|v_a - v_b\|^2 = K_{aa} + K_{bb} - 2K_{ab}$$

# Assignment Step

$$\|v_j - \mu_i\|^2 = v_j^T v_j + \mu_i^T \mu_i - 2 v_j^T \mu_i$$

$$v_j^T \mu_i = v_j^T \left( \frac{1}{n_i} \sum_{a \in \pi_i} v_a \right) = \frac{1}{n_i} \sum_{a \in \pi_i} v_j^T v_a = \frac{1}{n_i} \sum_{l \in \pi_i} K_{ja}$$

$$\mu_i^T \mu_i = \left( \frac{1}{n_i} \sum_{a \in \pi_i} v_a \right) \left( \frac{1}{n_i} \sum_{b \in \pi_i} v_b \right) = \frac{1}{n_i^2} \sum_{a \in \pi_i} \sum_{b \in \pi_i} v_a^T v_b = \frac{1}{n_i^2} \sum_{a \in \pi_i} \sum_{b \in \pi_i} K_{ab}$$

$$\|v_j - \mu_i\|^2 = K_{jj} + \frac{1}{n_i^2} \sum_{a \in \pi_i} \sum_{b \in \pi_i} K_{ab} - \frac{2}{n_i} \sum_{l \in \pi_i} K_{ja}$$

# Kernel k-means

- Initialize: Random partitions
- Repeat

$$c(j,i) = \arg\min_i \left( K_{jj} + \frac{1}{n_i^2} \sum_{a \in \pi_i} \sum_{b \in \pi_i} K_{ab} - \frac{2}{n_i} \sum_{l \in \pi_i} K_{ja} \right)$$

# Kernel Matrix and the Number of Clusters

Estimate $k$ using the Kernel Matrix:

- $NxN$ dim. sym. kernel matrix $K = k(x, x)$

- each element of the matrix is a dot-product distance in the kernel defined feature space

- matrix has block diagonal structure with groupings/clusters

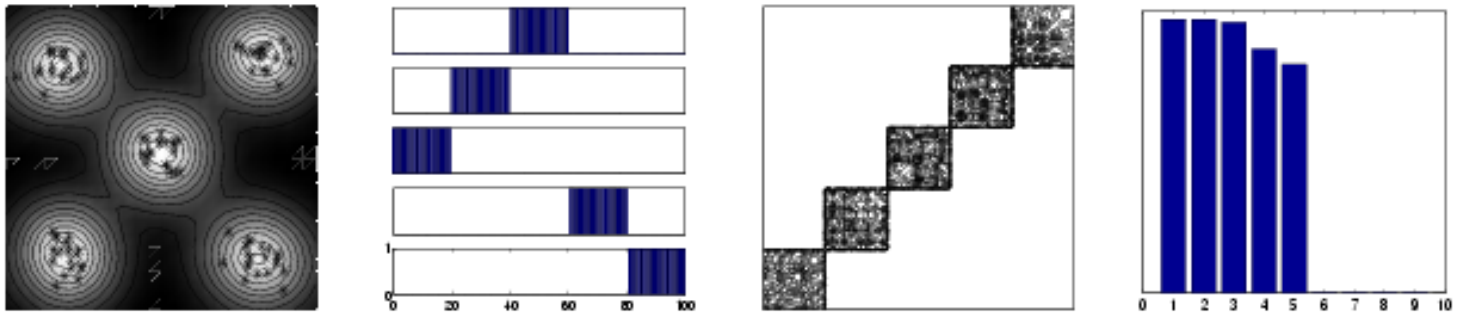



Fig. 1. **First Left** : The scatter plot of 100 points, composed of 20 datums drawn from five compact and well separated spherical Gaussian clusters. The iso-contours show the lines of constant value of $1-\mathcal{D}_{kj}$, light colours indicate high values whereas dark colours indicate low values. This was generated using an RBF kernel of width 0.1. **Second Left** : This plot shows the value of the binary indicator variables $\mathbf{Z}$ after convergence of the iterative routine to optimise the feature space sum-of-squares clustering criterion. Each row corresponds to a cluster centre and the individual data points, ordered in terms of cluster membership (purely for demonstrative purposes) run along the horizontal axis. The bars indicate a value of $z_{kj}$. It can be seen that there are no cluster assignment errors on this simple data set. **Third Left** : The contour plot of the $100 \times 100$ kernel matrix clearly showing the inherent block structure. The specific ordering has been used merely for purposes of demonstration and does not affect the results given by the proposed method. **Extreme Right** : The contribution to $\mathbf{1}_N^{\mathrm{T}}\mathbf{K}\mathbf{1}_N$ from the most significant terms $\lambda_i \left\{\mathbf{1}_N^{\mathrm{T}}\mathbf{u}_i\right\}^2$. It is most obvious that only five terms contribute to the overall value thus indicating that there are five dominant generators within the data sample.

Indication of the no. of clusters within data from eigenvalue decomposition of the kernel matrix.

If $K = U\Lambda U^t$, can write:

$$1_N^t K 1_N = 1_N^t \{\sum_{i=1}^N \lambda_i u_i u_i^t\} 1_N = \sum_{i=1}^N \lambda_i \{1_N^t u_i\}^2$$

$k$ dominant terms $\lambda_i \{1_N^t u_i\}^2$ in sum

$\rightarrow$ $k$ distinct clustered regions

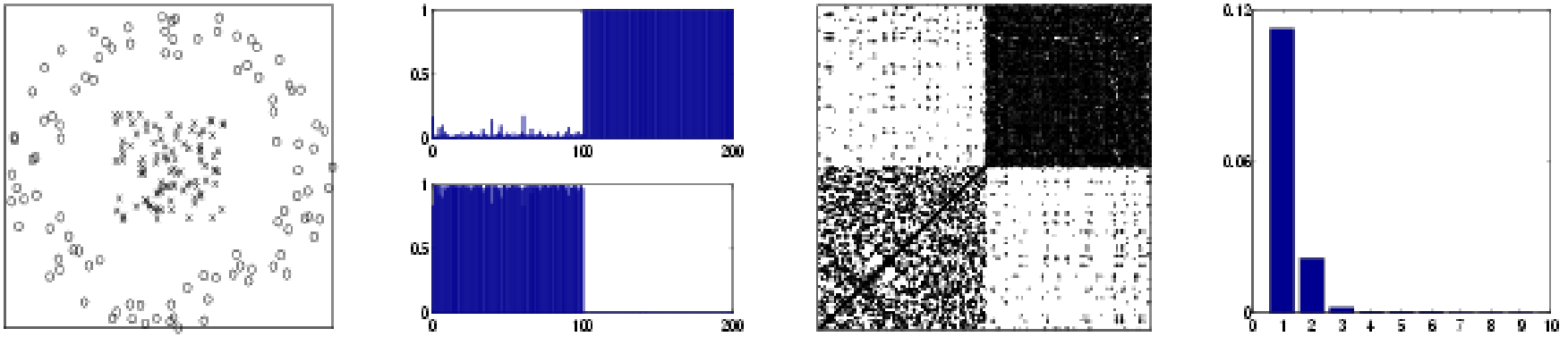



Fig. 2. **First Left** The scatter plot of the 'Ring Data', 100 samples from a uniform distribution centered at the origin and 100 samples uniformly drawn from an annular ring. **Second Left** The outcome of the clustering method showing that there are no partition errors. **Third Left** The contour plot of the associated kernel matrix (RBF width of 1.0), again note the block diagonal structure. **Extreme Right** The contribution to $\mathbf{1}_N^{\mathrm{T}}\mathbf{K}\mathbf{1}_N$ from the most significant terms $\lambda_i \left\{\mathbf{1}_N^{\mathrm{T}}\mathbf{u}_i\right\}^2$. It is most obvious that only two terms significantly contribute to the overall value thus indicating that there are two dominant generators within the data sample.

# Problems

- Calculating pairwise similarities between data instances is computationally expensive ($O(n^2)$) and memory inefficient ($O(n^2)$).
- Solutions:
  - Sparsify the similarity matrix
  - Restrict the centroids to be in the space of a few landmarks
  - Parallel and distributed implementations

# Reference


- Shi & Malik, Normalized Cuts and Image Segmentation, 2000
- Ng et al., On Spectral Clustering: Analysis and an algorithm , 2002
- Bach & Jordan, Learning Spectral Clustering, With Application To Speech Separation, 2006
- White & Smyth, Spectral Clustering Approach To Finding Communities in Graphs, 2005
- von Luxburg, A Tutorial on Spectral Clustering, 2007
- Ding et al. Tutorial on Spectral Clustering. ICDM 2004 (Link)
- Azran. Tutorial on Spectral Clustering. 2007 (Link)
- Ghodsi, Course Notes on Spectral Clustering, 2009 (Link)
- Schaeffer, Graph clustering, Computer Science Review, 1(1) pp. 27-64, 2007
- Dhillon et al. Kernel k-means, spectral clustering and normalized cuts. KDD, 2004.
- Rebecca Nugent Kernal K-Means Presentation. 4/21/2004