Graph (1)

- Consist of a set of nodes and a set of edges between those nodes
  - Most important model for applied mathematics

continuous	discrete
function	vector
derivative	difference
integral	sum
calculus	linear algebra

- Incidence matrix A (mxn)

$(m \text{ edges and } n \text{ nodes})$

edge i = row i, node $j \to k: -1$ in column j, $+1$ in column k

$N(\mathbf{A})$ contains all constant vectors: $\mathbf{x} = (c, c, \ldots, c)$

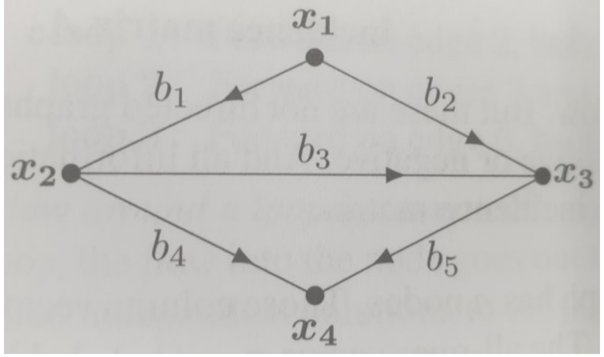
$$\dim N(\mathbf{A}) = 1,\ \dim C(\mathbf{A}) = \dim C(\mathbf{A}^T) = n-1,\ \dim N(\mathbf{A}^T) = m-(n-1)$$

row space contains all constant vectors $\mathbf{x}$ with $x_1 + x_2 + \cdots + x_n = 0(\mathbf{x} \perp \mathbf{1})$

$$\text{bases}\begin{cases} N(\mathbf{A}):\ \text{constant vector } \mathbf{1} \\ C(\mathbf{A}^T):(n-1) \text{ rows of } \mathbf{A} \text{ that produce a tree in the graph (a tree has no loop)} \\ C(\mathbf{A}):\ \text{any } (n-1) \text{ columns of } \mathbf{A} \\ N(\mathbf{A}^T):\ \text{flows around the } (m-n+1) \text{ small loops in the graph} \end{cases}$$

Graph (2)

- Example: m=5, n=4



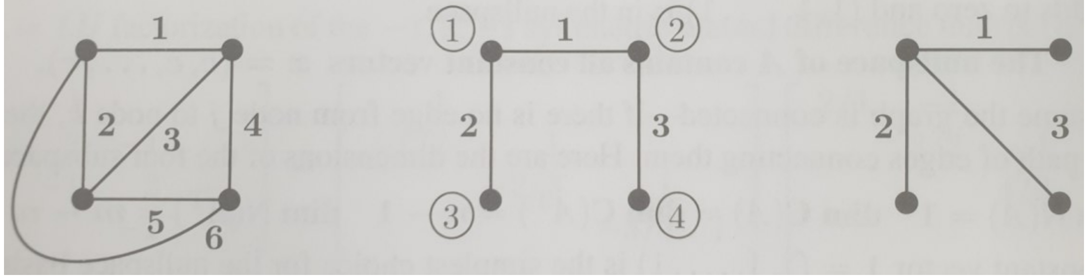
$$\begin{array}{rcccccc} & & & & & & \text{edges}\\ \mathbf{A}= & \lceil & -1 & 1 & 0 & 0 \rceil & 1\\ & | & -1 & 0 & 1 & 0 \;| & 2\\ & | & 0 & -1 & 1 & 0 \;| & 3\\ & | & 0 & -1 & 0 & 1 \;| & 4\\ & \lfloor & 0 & 0 & -1 & 1 \rfloor & 5\\ \text{nodes} & & 1 & 2 & 3 & 4 & \end{array}$$

- Graph Laplacian matrix $L=A^TA$
  - Symmetric, positive semidefinite

$$\mathbf{L}=\mathbf{A}^T\mathbf{A}=\begin{bmatrix}2&-1&-1&0\\-1&3&-1&-1\\-1&-1&3&-1\\0&-1&-1&2\end{bmatrix}=\underbrace{\begin{bmatrix}2&&&\\&3&&\\&&3&\\&&&2\end{bmatrix}}_{\substack{\text{degree matrix}\\ \text{count edges into node}}}-\underbrace{\begin{bmatrix}0&1&1&0\\1&0&1&1\\1&1&0&1\\0&1&1&0\end{bmatrix}}_{\substack{\text{adjacency matrix}\\ b_{jk}=1\text{: edge from } j \text{ to } k}}=\mathbf{D}-\mathbf{B}$$

Graph (2)

- Complete graph: every pair of nodes is connected by an edge, D=(n-1)I, B=all-ones minus I
- Tree: there are no loops in the connected graph



$$\underbrace{\mathbf{A}_1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix}}_{\text{complete graph: } m=\frac{1}{2}n(n-1)}, \quad \underbrace{\mathbf{A}_2 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}, \; \mathbf{A}_3 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}}_{\text{tree: } (n-1)}$$

$$\rightarrow \text{any graph: } (n-1) \le m \le \frac{1}{2}n(n-1)$$

Kirchhoff’s Current Law: A^{T}y=f

- KCL = balance of currents (forces, money)
  - Flow into each node equals flow out from that node
  - Key to solving A^{T}y=0 is to look at the small loops in the graph
  - (m-n+1) independent solutions
  - (number of nodes) – (number of edges) + (number of loops) = 1

$$\mathbf{A}_1=\begin{bmatrix}-1&1&0&0\\-1&0&1&0\\0&-1&1&0\\0&-1&0&1\\0&0&-1&1\\-1&0&0&1\end{bmatrix}\rightarrow\underbrace{\mathbf{A}_1\mathbf{x}=\mathbf{0}}_{\substack{\mathbf{x}=(1,1,1,1)\\\text{not interesting!}}},\ \mathbf{A}_1{}^T\mathbf{y}=\begin{bmatrix}-1&-1&0&0&0&-1\\1&0&-1&-1&0&0\\0&1&1&0&-1&0\\0&0&0&1&1&1\end{bmatrix}\begin{bmatrix}y_1\\y_2\\y_3\\y_4\\y_5\\y_6\end{bmatrix}=\mathbf{0}\rightarrow\mathbf{y}_1=\begin{bmatrix}-1\\1\\-1\\0\\0\\0\end{bmatrix},\ \mathbf{y}_2=\begin{bmatrix}0\\0\\1\\-1\\1\\0\end{bmatrix},\ \mathbf{y}_3=\begin{bmatrix}0\\-1\\0\\0\\-1\\1\end{bmatrix}$$

$$\text{outer loop}=\mathbf{y}_1+\mathbf{y}_2+\mathbf{y}_3$$

$$\mathbf{A}_2{}^T\mathbf{y}=0\rightarrow\mathbf{y}=0$$

A^TCA Framework in Applied Mathematics

- Graphs are perfect examples for three equations in engineering, science, economics
  - Describe a system in steady state equilibrium
  - Balance laws: conservation of charge, balance of force, zero net income in economics, conservation of mass and energy, continuity of every kind

$$\begin{cases} \text{voltages } \mathbf{x} = (x_1, x_2, x_3, x_4) \text{ at the four nodes} \\ \text{currents } \mathbf{y} = (y_1, y_2, y_3, y_4, y_5, y_6) \text{ atalong the six edges} \end{cases}$$

$$\begin{cases} \text{Voltage differences across edges} & \mathbf{e} = \mathbf{A}\mathbf{x} & e_1 = (\text{voltage at end node } 2) - (\text{voltage at end node } 1) \\ \text{Ohm's law on each edge} & \mathbf{y} = \mathbf{C}\mathbf{e} & \text{current } y_1 = c_1 \text{ times } e_1 = (\text{conductance})(\text{voltage}) \\ \text{Kirchhoff's Law with current sources} & \mathbf{f} = \mathbf{A}^T\mathbf{y} & \text{current sources } \mathbf{f} \text{ into nodes balance the internal currents } \mathbf{y} \end{cases}$$

$$\rightarrow \mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x} = \mathbf{f} \rightarrow \mathbf{K}\mathbf{x} = \mathbf{f}$$

$$\mathbf{K}: \text{ symmetric, positive semidefinite } \xrightarrow[\substack{x_4=0 \\ n-1=3 \text{ unknown voltages}}]{\text{boundary condition}} \underbrace{\text{reduced } \mathbf{K}}_{(3\times6)(6\times6)(6\times3)}: \text{ symmetric, invertible, positive definite}$$

$$\text{energy}: \ \mathbf{x}^T\left(\mathbf{A}^T\mathbf{C}\mathbf{A}\mathbf{x}\right) = (\mathbf{A}\mathbf{x})^T\,\mathbf{C}(\mathbf{A}\mathbf{x}) > 0 \text{ if } \mathbf{x} \neq \mathbf{0}$$

A^{T}CA Framework in Applied Mathematics

- Liner regression: least squares applied to Ax=b

$$\begin{cases} \mathbf{A}^T\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{b}: \text{ Normal equation for the vector } \hat{\mathbf{x}} \text{ that best fits the data } \mathbf{b} \\ \mathbf{A}^T\mathbf{C}\mathbf{A}\hat{\mathbf{x}} = \mathbf{A}^T\mathbf{C}\mathbf{b}: \text{ Least squares weighted by the inverse covariance matrix } \mathbf{C} = \mathbf{V}^{-1} \\ \min \left\| \mathbf{b} - \mathbf{A}\mathbf{x} \right\|_{\mathbf{C}}^2 : \text{ Minumum squared error } \left(\mathbf{b} - \mathbf{A}\mathbf{x}\right)^T \mathbf{C}\left(\mathbf{b} - \mathbf{A}\mathbf{x}\right) \end{cases}$$

- Graph Laplacian Matrix

$$\mathbf{K} = \mathbf{A}^T\mathbf{C}\mathbf{A}: \text{ weighted graph Laplacian, } \mathbf{G} = \mathbf{A}^T\mathbf{A}: \text{ standard Laplacian} \left(\mathbf{C} = \mathbf{I}\right)$$

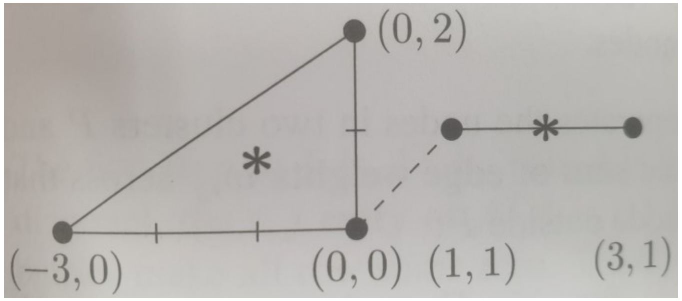
$$\mathbf{A}^T\mathbf{A} = \left(\text{diagonal}\right) + \left(\text{off-diagonal}\right) = \left(\text{degree matrix}\right) - \left(\text{adjacency matrix}\right) = \mathbf{D} - \mathbf{B}$$

$$\begin{cases} \text{every row and column of } \mathbf{G} \text{ and } \mathbf{K} \text{ adds to zero because } \mathbf{x} = \left(1, \ldots, 1\right) \text{ has } \mathbf{A}\mathbf{x} = \mathbf{0} \\ \mathbf{G} = \mathbf{A}^T\mathbf{A} \text{ is symmetric because edges go both ways (undirected graph)} \\ \text{The diagonal entry } \left(\mathbf{A}^T\mathbf{A}\right)_{ii} \text{ counts the edges meeting at node } i\text{: the degree} \\ \text{The off-diagonal entry is } \left(\mathbf{A}^T\mathbf{A}\right)_{ij} = -1 \text{ when an edge connects node } i \text{ and } j \\ \mathbf{G} \text{ and } \mathbf{K} \text{ are positive semidefinite but not positive definite (because } \mathbf{A}\mathbf{x} = \mathbf{0} \text{ in } \mathbf{1}\text{)} \end{cases}$$

Clustering

- How to understand a graph with many nodes?
  - Separate nodes into two or more clusters
  - Human Genome project: cluster genes that show highly correlated
- Break a graph in two pieces: clusters of nodes
  - Each cluster should contain roughly half of the nodes
  - The number of edges between clusters should be relatively small
- Examples
  - For load balancing in high computing, assign equal work to two processors
  - For social networks, identify two distinct groups
  - Segment an image
  - Reorder rows and columns of a matrix to make off-diagonal blocks sparse

Example with Two Clusters



$n = 5$ nodes, $k = 2$ clusters

centroid * : $\mathbf{c}_1 = (-1, 2/3),\ \mathbf{c}_2 = (2,1) \leftarrow$ minimize the sume of squared distances $\left\|\mathbf{c} - \mathbf{a}_j\right\|^2$

Approximate an $m \times n$ matrix of $\mathbf{A}$ by $\underset{\text{low rank}}{\mathbf{CR}} = \underbrace{(m \times k)}_{\substack{\text{only } k \text{ columns} \\ \text{centroids of clusters}}} \underbrace{(k \times n)}_{\substack{\text{single 1 and} \\ k-1 \text{ zeros}}}$

$R_{ij} = 1(\text{or } 0)$ if centroid i is closest (or not) to the point $\mathbf{x}_j$

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 3 & 0 & -3 \\ 0 & 1 & 1 & 2 & 0 \end{bmatrix} \approx \begin{bmatrix} -1 & 2 & 2 & -1 & -1 \\ 2/3 & 1 & 1 & 2/3 & 2/3 \end{bmatrix}$$

$$\mathbf{A} \approx \mathbf{CR} = \begin{bmatrix} -1 & 2 \\ 2/3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Four Methods for Clustering

- Spectral clustering (Fiedler vector)
  - using the graph Laplacian or the modularity matrix
- Minimum cut
- Weighted k-means

Four Methods for Clustering

I. $\mathbf{A}^T\mathbf{CAz} = \lambda\mathbf{Dz} \to \mathbf{z}:$ Fiedler vector $\begin{cases} \lambda_1 = 0 \to \mathbf{z}_1 = (1,\dots,1) \\ \lambda_2 \to \mathbf{z}_2 : \ +/- \text{ components indicate two clusters of nodes} \end{cases}$

II. $\mathbf{A}^T\mathbf{CA} \to \underset{\substack{\text{modularity}\\ \text{matrix}}}{\mathbf{M}} = \mathbf{B} - \frac{1}{2m}\mathbf{dd}^T$ where $\mathbf{d}:$ degrees of the n nodes (number of edges adjacent to the nodes)

choose eigenvector that comes with the largest eigenvalue of **M**

III. Find the minimum normalized cut that separates the nodes in two clusters P and Q

weight across cut: $links(P) = \sum w_{ij}$ for i in P and j not in P

size of cluster: $size(P) = \sum w_{ij}$ for i in P

normalized cut weight: $N_{cut}(P,Q) = \dfrac{links(P)}{size(P)} + \dfrac{links(Q)}{size(Q)} \xrightarrow{k-cut} N_{cut}(P_1,\dots,P_k) = \displaystyle\sum_{i=1}^{k} \frac{links(P_i)}{size(P_i)}$

minimize $N_{cut}(P,Q):$ good partition of the graph → application: segmentation of images

IV. nodes in the graph: $\mathbf{a}_1,\dots,\mathbf{a}_n$, clusters P and Q have centers $\mathbf{c}_P \left(= \dfrac{\sum \mathbf{a}_i}{|P|} \right)$ and $\mathbf{c}_Q$

minimize the total squred distance from nodes to those centroids: $E = \displaystyle\sum_{i \text{ in } P} \|\mathbf{a}_i - \mathbf{c}_P\|^2 + \sum_{i \text{ in } Q} \|\mathbf{a}_i - \mathbf{c}_Q\|^2$

Spectral Clustering (1)

$$\mathbf{A}^T\mathbf{CA} \xrightarrow[\text{the Laplacian}]{\mathbf{D}\text{ normalizes}} \mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}^T\mathbf{CAD}^{-1/2} = \mathbf{I} - \mathbf{N} \text{ where } n_{ij} = \frac{w_{ij}}{\sqrt{d_i d_j}} (\text{normalized weights})$$

triangular graph: $n = 3$ nodes, $m = 3$ edges, $c_1, c_2, c_3 = w_{12}, w_{13}, w_{23}$

$$\left.\begin{matrix} \mathbf{A}^T\mathbf{CA} = \mathbf{D} - \mathbf{W} \\ \begin{bmatrix} w_{12}+w_{13} & -w_{12} & -w_{13} \\ -w_{21} & w_{21}+w_{23} & -w_{23} \\ -w_{31} & -w_{32} & w_{31}+w_{32} \end{bmatrix} \end{matrix}\right\} \to \left\{\begin{matrix} \mathbf{L} = \mathbf{D}^{-1/2}\mathbf{A}^T\mathbf{CAD}^{-1/2} \\ \begin{bmatrix} 1 & -n_{12} & -n_{13} \\ -n_{21} & 1 & -n_{23} \\ -n_{31} & -n_{32} & 1 \end{bmatrix} \end{matrix}\right.$$

$\mathbf{L} = \mathbf{I} - \mathbf{N}$ is like a correlation matrix in statistics

1. $\mathbf{L}$ is symmetric positive semidefinite: orthogonal eigenvectors, all eigenvalues $\lambda \geq 0$

2. The eigenvectors for $\lambda = 0$ is $\mathbf{u} = \left(\sqrt{d_1}, \dots, \sqrt{d_n}\right)$. Then $\mathbf{Lu} = \mathbf{D}^{-1/2}\mathbf{A}^T\mathbf{CA1} = 0$.

3. The second eigenvector $\mathbf{v}$ of $\mathbf{L}$ minimizes the Rayleigh quotient on a subspace.

$$\begin{pmatrix} \lambda_2 = \text{smallest nonzero eigenvalue of } \mathbf{L} \to \min\limits_{\substack{\text{subject to} \\ \mathbf{x}^T\mathbf{u}=0}} \dfrac{\mathbf{x}^T\mathbf{Lx}}{\mathbf{x}^T\mathbf{x}} = \dfrac{\mathbf{v}^T\mathbf{Lv}}{\mathbf{v}^T\mathbf{v}} = \lambda_2 \text{ at } \mathbf{x} = \mathbf{v} \\ \\ \text{upper bound for } \lambda_2, \text{ for any } \mathbf{x} \text{ orthogonal to the first eigenvector } \mathbf{u} = \mathbf{D}^{-1/2}\mathbf{1} \end{pmatrix}$$

Spectral Clustering normalized vs. unnormalized

$$\mathbf{Lv} = \mathbf{D}^{-1/2}\mathbf{A}^T\mathbf{CAD}^{-1/2}\mathbf{v} = \lambda\mathbf{v} \xrightarrow[\text{normalized Fiedler vector}]{\mathbf{z}=\mathbf{D}^{-1/2}\mathbf{v}} \begin{cases} \mathbf{A}^T\mathbf{CAz} = \lambda\mathbf{Dz} \quad \text{with } \mathbf{1}^T\mathbf{Dz} = 0 \\ \text{generalized eigenvalue problem} \\ \text{eigenvector for } \lambda = 0 \text{ is } \mathbf{1} \\ \text{the next eigenvector } \mathbf{z} \text{ is } \mathbf{D}\text{-orthogonal to } \mathbf{1} \\ \mathbf{A}^T\mathbf{CAz} = \lambda_2\mathbf{Dz} \end{cases}$$

$$\min_{\substack{\text{subject to}\\ \mathbf{x}^T\mathbf{u}=0}} \frac{\mathbf{x}^T\mathbf{Lx}}{\mathbf{x}^T\mathbf{x}} \xrightarrow{\mathbf{x}=\mathbf{D}^{1/2}\mathbf{y}} \min_{\substack{\text{subject to}\\ \mathbf{1}^T\mathbf{Dy}=0}} \frac{\mathbf{y}^T\mathbf{A}^T\mathbf{CAy}}{\mathbf{y}^T\mathbf{Dy}} = \frac{\sum\sum w_{ij}\left(y_i - y_j\right)^2}{\sum d_i y_i^2} = \lambda_2 \text{ at } \mathbf{y} = \mathbf{z}$$

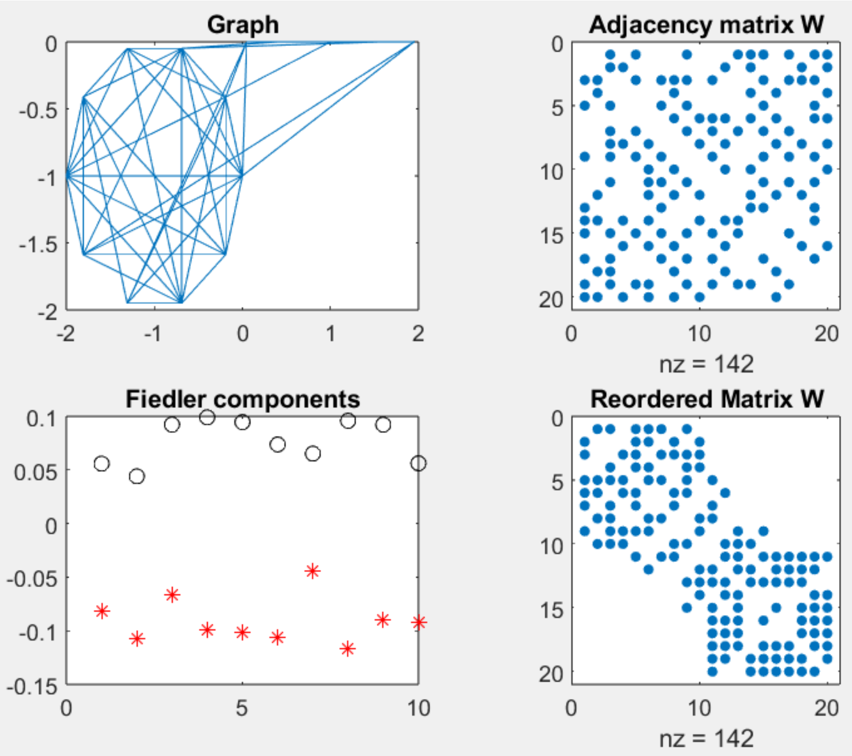
$\mathbf{Ay}$: incidence matrix $\mathbf{A}$ gives the differences $\left(y_i - y_j\right)$

- Example: 20-node graph has two 10-node clusters P and Q (to find from z)
  - Create edges within P and Q with probability 0.7. Edges between nodes in P and Q have smaller probability 0.1. All edges have wrights w_{ij}=1. (C=I)

Code: MATLAB

```
N=10; W=zeros(2*N,2*N); % Generate 2N nodes in two clusters
rand('state',100) % rand repeats to give the same graph
for i=1:2*N-1
for j=i+1:2*N
p=0.7-0.6*mod(j-i,2); % p=0.1 when j-i is odd, 0.7 else
W(i,j)=rand<p; % Insert edges with probability p
end % The weights are wi,j=1 (or 0)
end % So far W is strictly upper triangular
W=W+W'; D=diag(sum(W)); % Adjacency matrix W, degress in D
G=D-W; [V,E]=eig(G,D); % Eigenvalues of Gx=(lambda)Dx in E
[a,b]=sort(diag(E)); z=V(:,b(2));% Fiedler eigenvector z for (lambda)2
plot(sort(z),'.-'); % Show +- groups of Fiedler components


theta=[1:N]*2*pi/N; x=zeros(2*N,1); y=x; % Angles to plot graph
x(1:2:2*N-1)=cos(theta)-1; x(2:2:2*N)=cos(theta)+1;
y(1:2:2*N-1)=sin(theta)-1; x(2:2:2*N)=sin(theta)+1;
print theta,x,y
subplot(2,2,1), gplot(W,[x,y]), title('Graph')
subplot(2,2,2), spy(W), title('Adjacency matrix W')
subplot(2,2,3), plot(z(1:2:2*N-1),'ko'), hold on
plot(z(2:2:2*N),'r*'), hold off, title('Fiedler components')
[c,d]=sort(z); subplot(2,2,4), spy(W(d,d)), title('Reordered Matrix W')
```



Minimum Cut

(edge) weight across cut: $links(P)=\sum w_{ij}$ for i in P and j not in P

size of cluster: $size(P)=\sum w_{ij}$ for i in P

normalized cut weight: $Ncut(P,Q)=\dfrac{links(P)}{size(P)}+\dfrac{links(Q)}{size(Q)}$

normalized K-cut: $Ncut(P_1,\ldots,P_k)=\displaystyle\sum_{i=1}^{K}\frac{links(P_i)}{size(P_i)}$

$[\text{cuts connected to eigenvectors}]$

perfect indicator of a cut: vector $\mathbf{y}$ with all components equal to p or $-q$ (two values only) → node i goes $\begin{cases}\text{in } P \text{ if } y_i = p\\ \text{in } Q \text{ if } y_i = -q\end{cases}$

$$\left.\begin{array}{l}\mathbf{1}^T\mathbf{Dy} \text{ will multiply one group of } d_i \text{ by } p \text{ and the other group by } -q.\\ \text{The first } d_i \text{ add to } size(P)=\text{sum of } d_i \;\; (i \text{ in } P).\\ \text{The second group of } d_i \text{ add to } size(Q)\end{array}\right\}\rightarrow \mathbf{1}^T\mathbf{Dy}=0\rightarrow psize(P)=qsize(Q)$$

$$\frac{\mathbf{y}^T\mathbf{A}^T\mathbf{CAy}}{\mathbf{y}^T\mathbf{Dy}}=\frac{\sum\sum w_{ij}\left(y_i-y_j\right)^2}{\sum d_i{y_i}^2}=\frac{(p+q)^2\,links(P,Q)}{p^2size(P)+q^2size(Q)}=\frac{(p+q)links(P,Q)}{psize(P)}=\frac{links(P,Q)}{size(P)}+\frac{links(P,Q)}{size(Q)}=Ncut(P,Q)$$

Clustering by k-means

n points $\mathbf{a}_1,\ldots,\mathbf{a}_n$ in d-dimensional space $\rightarrow$ partition those points into k clusters

clusters $P_1,\ldots,P_k$ have centroids $\mathbf{c}_1,\ldots,\mathbf{c}_k$

$\mathbf{c}_j = \dfrac{\text{sum of } \mathbf{a}\text{'s}}{\text{number of } \mathbf{a}\text{'s}} \rightarrow$ minimize $\sum \|\mathbf{c}-\mathbf{a}\|^2$ for all $\mathbf{a}$'s in cluster P_j

clustering: to find the partition $P_1,\ldots,P_k$ with minimum total distance D to centroids:

$$\text{minimize } D = \sum_{j=1}^{k} D_j = \sum_{j=1}^{k} \|\mathbf{c}_j - \mathbf{a}_i\|^2 \text{ for } \mathbf{a}_i \text{ in cluster } P_j$$

step 1: find the centroids $\mathbf{c}_j$ of the (old) clustering $P_1,\ldots,P_k$.

step 2: find the (new) clustering that puts $\mathbf{a}$ in P_j if $\mathbf{c}_j$ is the closest centroid.

Clustering by k-means: Weights and Kernel Method

weighted distance: $d\left(\mathbf{x},\mathbf{a}_i\right) = w_i\left\|\mathbf{x}-\mathbf{a}_i\right\|^2$, $\mathbf{c}_j = \dfrac{\sum w_i\mathbf{a}_i}{\sum w_i}\left(\mathbf{a}_i \text{ in } P_j\right)$

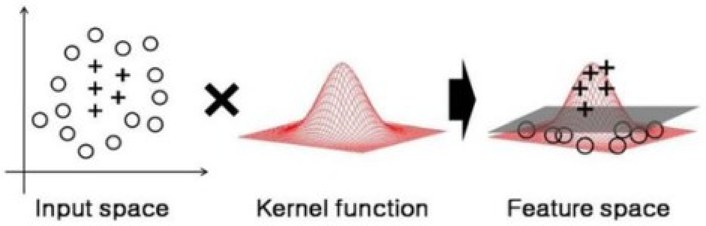
distances to centroids only require dot product $\mathbf{a}_i\cdot\mathbf{a}_j$: $\left(\text{each } i \text{ in } P_j\right)$ $\left\|\mathbf{c}_j-\mathbf{a}_i\right\|^2 = \mathbf{c}_j\cdot\mathbf{c}_j - 2\mathbf{c}_j\cdot\mathbf{a}_i + \mathbf{a}_i\cdot\mathbf{a}_i$

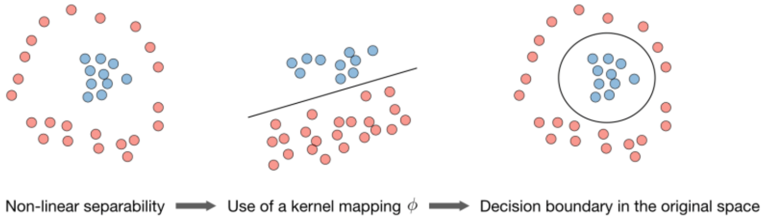
Kernel method: weighted kernel matrix $\mathbf{K}$ has entries $\mathbf{a}_i\cdot\mathbf{a}_l$

nodes are point $\mathbf{x}_i$ in input space $\rightarrow \mathbf{a}_i = \phi\left(\mathbf{x}_i\right)$ points in a high-dimensional feature space

$$\left(\text{sum over nodes in } P_j\right)\quad \sum\left\|\mathbf{c}_j-\mathbf{a}_i\right\|^2 = \frac{\sum w_i w_l \mathbf{K}_{il}}{\left(\sum w_i\right)^2} - 2\frac{\sum w_i\mathbf{K}_{il}}{\sum w_i} + \sum \mathbf{K}_{ii}$$

(vision)	polynomial	$\mathbf{K}_{il} = \left(\mathbf{x}_i\cdot\mathbf{x}_l + c\right)^d$
(statistics)	Gaussian	$\mathbf{K}_{il} = \exp\left(-\dfrac{\left\|\mathbf{x}_i-\mathbf{x}_l\right\|^2}{2\sigma^2}\right)$
(neural networks)	Sigmoid	$\mathbf{K}_{il} = \tanh\left(c\mathbf{x}_i\cdot\mathbf{x}_l + \theta\right)$




Non-linear separability
Use of a kernel mapping ϕ
Decision boundary in the original space

for large data sets, k-means and $\operatorname{eig}\left(\mathbf{A}^T\mathbf{C}\mathbf{A},\mathbf{D}\right)$ will be expensive $\rightarrow \begin{cases}\text{random sampling}\\ \text{multilevel clustering}\end{cases}$

Applications of Clustering

– Learning theory, training sets, neural networks, Hidden Markov Models
– Classification, regression, pattern recognition, Support Vector Machines
– Statistical learning, maximum likelihood, Bayesian statistics, spatial statistics, kriging, time series, ARMA models, stationary processes
– Social networks, organization theory
– Data mining, document indexing, image retrieval, kernel-based learning, Nystrom method, low rank approximation
– Bioinformatics, microarray data, systems biology
– Cheminformatics, drug design, decision trees
– Information theory, vector quantization, rate distortion theory, Bregman divergences
– Image segmentation, computer vision, texture, min cut
– Predictive control, feedback samples, robotics, adaptive control, Riccati equations