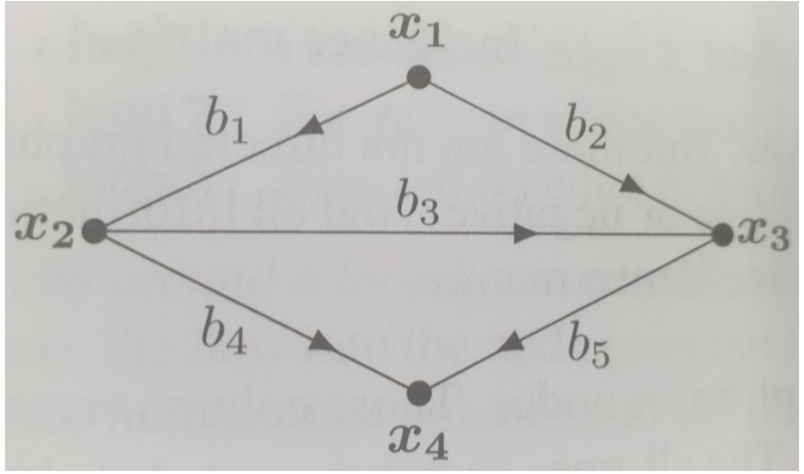
Graph (1)

- Consist of a set of nodes and a set of edges between those nodes
- Incidence matrix A (mxn)
  - m edges and n nodes
  - Dimensions of the four subspaces
- Graph Laplacian matrix $A^T A$
  - Symmetric, positive semidefinite
  - Degree matrix D, adjacency matrix B

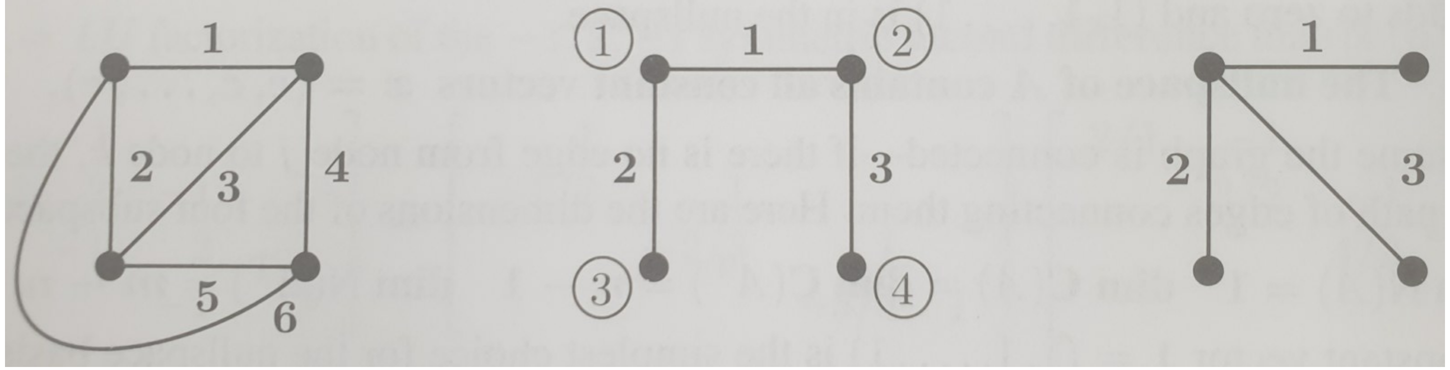
continuous	discrete
function	vector
derivative	difference
integral	sum
calculus	Linear algebra



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

Graph (2)

- Complete graph: every pair of nodes is connected by an edge
- Tree: there are no loops in the connected graph



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

Kirchhoff's Current Law

- 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

$$\mathbf{A}_1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \rightarrow \mathbf{A}_1\mathbf{x} = 0,\ \mathbf{A}_1{}^T\mathbf{y} = \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & -1 \\ 1 & 0 & -1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{bmatrix} = 0 \rightarrow \mathbf{y}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \end{bmatrix},\ \mathbf{y}_2 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 1 \\ 0 \end{bmatrix},\ \mathbf{y}_3 = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 0 \\ -1 \\ 1 \end{bmatrix}$$

$$\mathbf{A}_2{}^T\mathbf{y} = \begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = 0$$

A^TCA Framework in Applied Mathematics

- Graphs are perfect examples for three equations in engineering, science, economics
- Weighted graph Laplacian
- Describe a system in steady state equilibrium

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

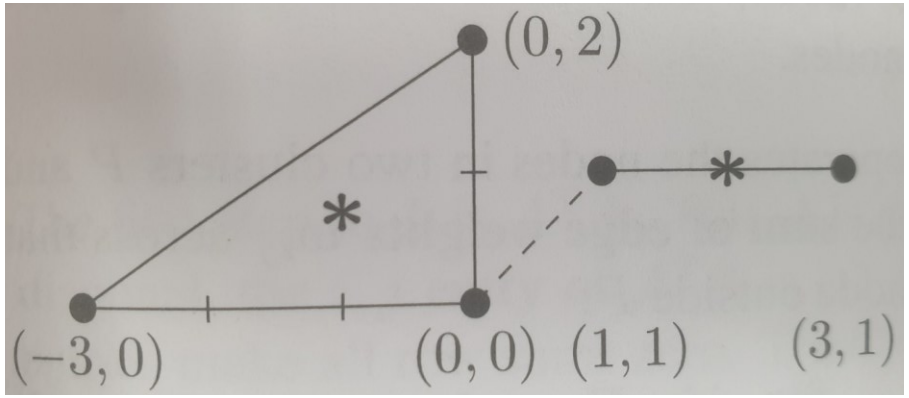
$$\begin{cases} \text{Voltage differences across edges} & \mathbf{e} = \mathbf{Ax} & e_1 = (\text{voltage at end node 2}) - (\text{voltage at end node 1}) \\ \text{Ohm's law on each edge} & \mathbf{y} = \mathbf{Ce} & \text{current } y_1 = (\text{conductance})(\text{voltage}) \\ \text{Kirchhoff's Law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

$$\rightarrow \mathbf{A}^T\mathbf{CAx} = \mathbf{f} \rightarrow \mathbf{Kx} = \mathbf{f}$$

$$\mathbf{K}: \text{ symmetric, positive semidefinite } \xrightarrow[x_4=0]{\text{boundary condition}} \text{reduced } \mathbf{K}: \text{ symmetric, positive definite}$$

Example with Two Clusters

- 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
  - 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



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

Approximate an $m \times n$ matrix of $\mathbf{A}$ by $\mathbf{CR} = (m \times k)(k \times n)$

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

Four Methods for Clustering

- Find the Fiedler vector z that solves $A^TCAz=\lambda Dz$. D normalizes the Laplacian. Positive and negative components of eigenvector of λ_2 indicate two clusters of nodes.
- Replace the graph Laplacian by the modularity matrix M=(adjacency matrix)-$dd^T/2m$. Choose the eigenvector that comes with the largest eigenvalue of M. vector d gives the degrees of the n nodes.
- Find the minimum normalized cut the separates the nodes in two clusters P and Q. The unnormalized measure of a cut is the sum of edge weights wij across that cut. Those edges connect a node in P to a node outside P.
- K-means

Spectral Clustering

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

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

$\mathbf{L}$ is symmetric positive semidefinite

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

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

$$\left(\lambda_2 = \text{smallest nonzero eigenvalue of } \mathbf{L} \rightarrow \min_{\substack{\text{subject to}\\ \mathbf{x}^T\mathbf{u}=0}} \frac{\mathbf{x}^T\mathbf{Lx}}{\mathbf{x}^T\mathbf{x}} = \frac{\mathbf{v}^T\mathbf{Lv}}{\mathbf{v}^T\mathbf{v}} = \lambda_2 \text{ at } \mathbf{x} = \mathbf{v}\right)$$

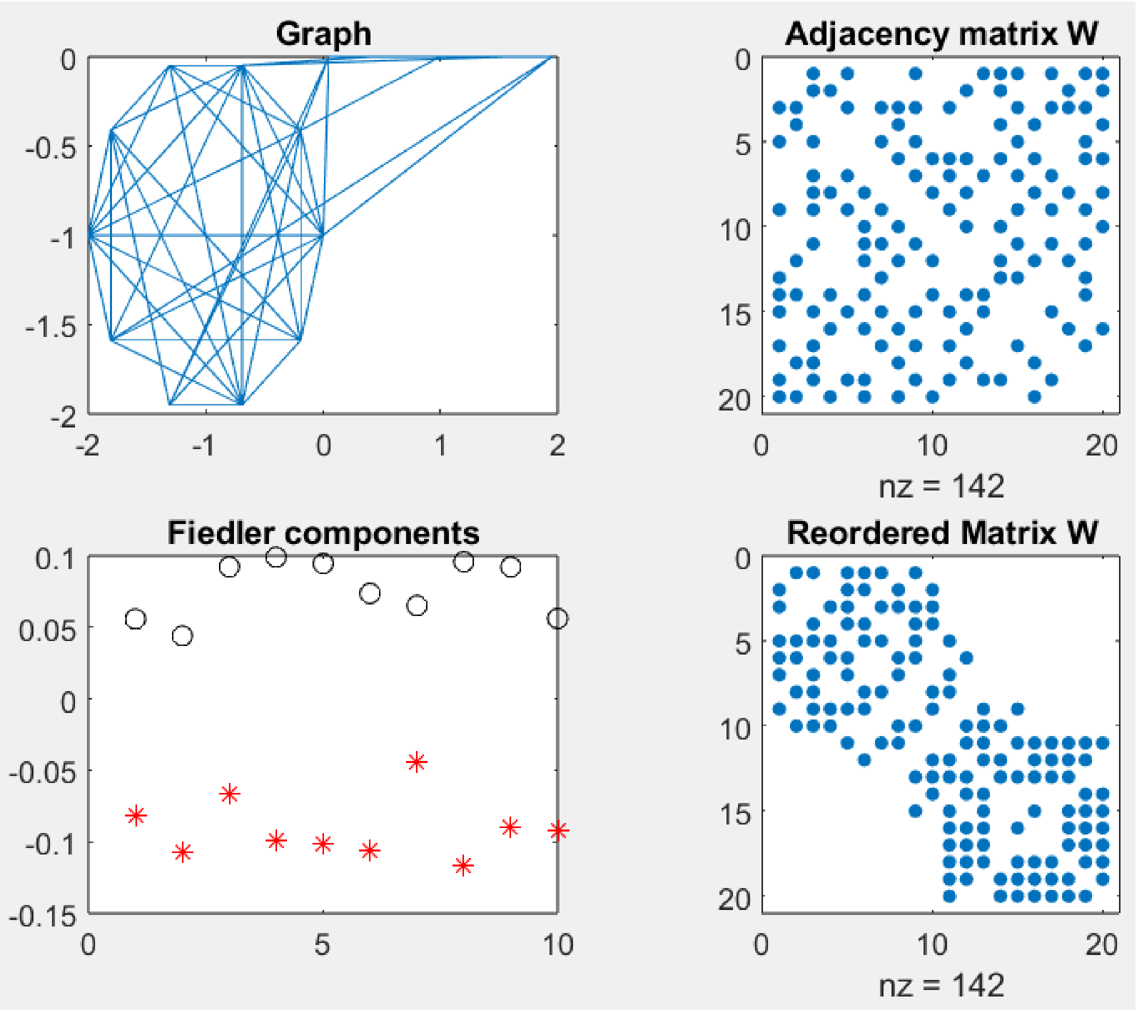
$$\mathbf{Lv} = \mathbf{D}^{-1/2}\mathbf{A}^T\mathbf{CAD}^{-1/2}\mathbf{v} = \lambda\mathbf{v} \xrightarrow[\text{normalized Fiedler vector}]{\mathbf{z}=\mathbf{D}^{-1/2}\mathbf{v}} \mathbf{A}^T\mathbf{CAz} = \lambda\mathbf{Dz} \text{ with } \mathbf{1}^T\mathbf{Dz} = 0$$

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

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: $Ncut(P_1,\ldots,P_k) = \sum_{i=1}^{K} \dfrac{links(P_i)}{size(P_i)}$

[cuts connected to eigenvectors]

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

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

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

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 $\mathbf{c}_1, \ldots, \mathbf{c}_k$

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

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

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.

Weights and Kernel Method

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

$$\|\mathbf{c}_j - \mathbf{a}_i\|^2 = \mathbf{c}_j \cdot \mathbf{c}_j - 2\mathbf{c}_j \cdot \mathbf{a}_i + \mathbf{a}_i \cdot \mathbf{a}_i$$

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

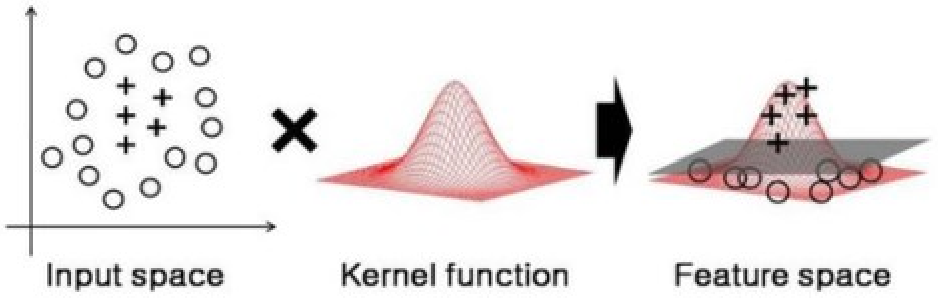
nodes are point $\mathbf{x}_i$ in input space $\rightarrow \mathbf{a}_i = \phi(\mathbf{x}_i)$ points in a high-dimensional feature space

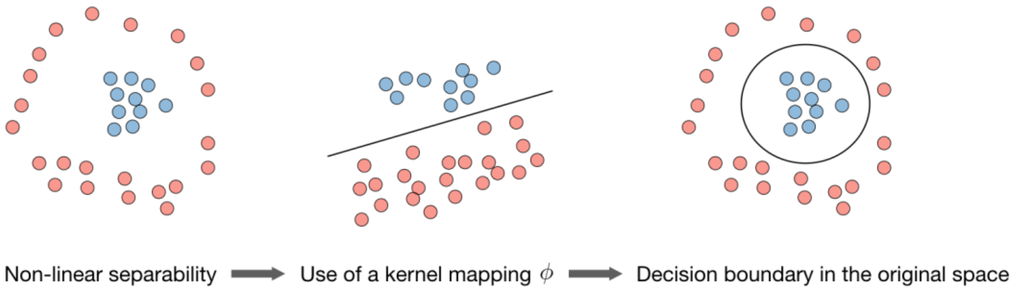
$$\sum \|\mathbf{c}_j - \mathbf{a}_i\|^2 = \frac{\sum w_i w_l \mathbf{K}_{il}}{\left(\sum w_i\right)^2} - 2\frac{\sum w_i \mathbf{K}_{il}}{\sum w_i} + \sum \mathbf{K}_{ii}$$

(vision) polynomial $\mathbf{K}_{il} = (\mathbf{x}_i \cdot \mathbf{x}_l + c)^d$

(statistics) Gaussian $\mathbf{K}_{il} = \exp\left(-\dfrac{\|\mathbf{x}_i - \mathbf{x}_l\|^2}{2\sigma^2}\right)$

(neural networks) Sigmoid $\mathbf{K}_{il} = \tanh(c\mathbf{x}_i \cdot \mathbf{x}_l + \theta)$





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
- Bioinformatics, microarray data, systems biology
- Cheminformatics, drug design, decision trees
- Information theory, vector quantization, rate distortion theory
- Image segmentation, computer vision, texture, min cut
- Predictive control, feedback samples, robotics, adaptive control