Unsupervised Clustering
Unsupervised Clustering

- Training samples are *not* labeled
- May not know
  - how many classes
  - *a priori* probability
  - state-conditional probability
- Automatic discovery of structures
  - Intuitively, objects in the same class stick together and form clusters
Unsupervised Clustering (cont)

Locating groups (clusters) having similar measurements

Given \( \mathcal{X} = \{x_1, x_2, ..., x_n\} \) (unlabelled)

partition into \( C \) clusters \( \mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_C \)
Similarity Measure

- Need a similarity measurement $s(x, x')$
  - (e.g., distance between $x$ and $x'$)

Minkowski Metric

$$d(x, y) = \left( \sum_{k=1}^{d} |x_k - y_k|^{q} \right)^{\frac{1}{q}}, q \geq 1$$

Mahalanobis Distance

$$d(x, y) = (x - y)^t \Sigma^{-1} (x - y)$$

Normalized inner product

$$d(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$

Commonalit $y$ (for binary features)

$$d(x, y) = \frac{x \cdot y}{d}$$
Similarity Measure (cont.)

- How to set the threshold?
  - Too large all samples assigned into a single class
  - Too small each sample in its own class

- How to properly weight each feature?
  - How do brightness of a fish and length of a fish relate?
  - How to scale (or normalize) different measures?
Axis Scaling (cont.)
Pathology of Similarity Measure

1. Scaling the samples changes the apparent clustering.

How do you compare the brightness of fish in one dimension with the length of the fish in another dimension?
Threshold
Criteria for Clustering

- A criterion function for clustering

\[ J_e = \sum_{i=1}^{c} \sum_{x \in N_i} |x - m_i|^2 \quad m_i = \frac{1}{n_i} \sum_{x \in N_i} x \]

\[ J = \sum_{i=1}^{c} \frac{1}{n_i} \sum_{x \in N_i} \sum_{x' \in N_i} |x - x'|^2 \]
Criteria for Clustering (cont.)

- **Within** and **Between** group variance
  - minimize and maximize the trace and determinant of the appropriate scatter matrix
  - **trace**: square of the scattering radius
  - **determinant**: square of the scattering volume

\[
S_W = \sum_{i=1}^{c} \sum_{x \in S_i} (x - m_i)(x - m_i)^t \\
m_i = \frac{1}{n_i} \sum_{x \in S_i} x
\]

\[
S_B = \sum_{i=1}^{c} n_i (m_i - m)(m_i - m)^t \\
m = \frac{1}{N} \sum_{x \in S} x
\]
Pathology: when class sizes differ

- Relaxed constraint
  - Allow the large class to grow into smaller class

- Stringent constraint
  - Disallow large class and split it

\[ J_e = \sum_{i=1}^{c} \sum_{x \in \mathbb{N}_i} |x - m_i|^2 \]
\[ m_i = \frac{1}{n_i} \sum_{x \in \mathbb{N}_i} x \]
**K-Means Algorithm**
*(fixed # of clusters)*

- Arbitrarily pick $N$ cluster centers, assign samples to nearest center
- Compute sample mean of each cluster
- Reassign samples to clusters with the nearest mean (for all samples)
- Repeat if there are changes, otherwise stop
**K-Means**

- In some sense, it is a simplified version of the case II of learning parametric form

\[ \hat{P}(w_i \mid x_k, \hat{\theta}) = \frac{p(x_k \mid w_i, \hat{\theta}_i)\hat{P}(w_i)}{\sum_{j=1}^{c} p(x_k \mid w_j, \hat{\theta}_j)\hat{P}(w_j)} \]

- In stead of multiple membership, give the sample to whichever class whose mean is closest to it

\[ \hat{P}(w_i \mid x_k, \hat{\theta}) = \begin{cases} 1 & \text{class with closest class mean} \\ 0 & \text{otherwise} \end{cases} \]
In Matlab, you can find k-mean (it is called c-mean) under fuzzy logic toolbox

It implements fuzzy k-mean

Where membership function is not 1 or 0, but can be fractional
Iterative K-means

- Iterative refinement of clusters to minimize

\[ J = \sum_{i=1}^{c} \sum_{x \in \mathcal{X}_i} |x - m_i|^2 \]

\[ m_i = \frac{1}{n_i} \sum_{x \in \mathcal{X}_i} x \]

- Each step: move a sample from one to another

\( \hat{x} \) from \( \mathcal{X}_i \) to \( \mathcal{X}_j \)

\[ m_i^* = m_i - \frac{\hat{x} - m_i}{n_i - 1} \]

\[ J_i^* = J_i - \frac{n_i}{n_i - 1} |\hat{x} - m_i|^2 \]

\[ m_j^* = m_j + \frac{\hat{x} - m_j}{n_j + 1} \]

\[ J_j^* = J_j + \frac{n_j}{n_j + 1} |\hat{x} - m_j|^2 \]

\[ \frac{n_i}{n_i - 1} |\hat{x} - m_i|^2 > \frac{n_j}{n_j + 1} |\hat{x} - m_j|^2 \]
1. Select an initial partition of the $n$ samples into clusters and compute $J, m_1, \ldots, m_c$

2. Select the next candidate sample $\hat{x}$

3. If the current cluster is larger than 1, then

$$\rho_j = \begin{cases} 
\frac{n_j}{n_j - 1} |\hat{x} - m_j|^2 & j = i \\
\frac{n_j}{n_j + 1} |\hat{x} - m_j|^2 & j \neq i 
\end{cases}$$

4. Transfer $\hat{x}$ to $\mathcal{S}_k$ if $\rho_k \leq \rho_j$ for all $j$

5. Update $J, m_i, m_k$

6. If $J$ has not changed in $n$ attempts, stop, otherwise go back to 2.
Hierarchical Clustering

- K-means assume a “flat” data description
- Hierarchical descriptions are more frequently
*Samples in the same cluster will remain so at a higher level*
Hierarchical clustering Options

- Bottom-up: agglomerate
- Top-down: divisive

1. Let \( \hat{c} = n \) and \( \mathcal{X} = \{x_1, \ldots, x_n\} \)
2. If \( \hat{c} \leq c \), stop
3. Find the nearest pair of distinct clusters, \( \mathcal{X}_i \) and \( \mathcal{X}_j \)
4. Merge \( \mathcal{X}_i \) and \( \mathcal{X}_j \), delete \( \mathcal{X}_j \), and decrement \( \hat{c} \) by 1
5. Go to 2.
Hierarchical clustering Procedure

- At a certain step, we have $c$ classes

$$|\mathbb{X}_i| = n_i, \quad m_i = \frac{1}{n_i} \sum_{x \in \mathbb{X}_i} x$$

Merge two clusters $i$ and $j$

$$\mathbb{X}_{ij} = \mathbb{X}_i \cup \mathbb{X}_j$$

$$|\mathbb{X}_{ij}| = n_i + n_j, \quad m_{ij} = \frac{1}{n_i + n_j} \sum_{x \in \mathbb{X}_{ij}} x = \frac{1}{n_i + n_j} (n_i m_i + n_j m_j)$$
Hierarchical clustering Procedure (cont.)

- Then certain criteria function will increase

\[
\Delta E_{ij} = \sum_{x \in \mathbb{N}_{ij}} |x - m_{ij}|^2 - \sum_{x \in \mathbb{N}_i} |x - m_i|^2 - \sum_{x \in \mathbb{N}_j} |x - m_j|^2
\]

\[
= n_i m_i^2 + n_j m_j^2 - (n_i + n_j) m_{ij}^2
\]

\[
= \frac{n_i n_j}{n_i + n_j} |m_i - m_j|^2
\]

\(I^*,j^*\) chosen in such a way

\[
< i^*, j^* > = \arg \min_{i,j} \frac{n_i n_j}{n_i + n_j} |m_i - m_j|^2 = \arg \min_{i,j} \|d(\mathbb{N}_i, \mathbb{N}_j)\|
\]

Until no such pair exist that cause an increase in the criteria function less than certain preset threshold
In fact, the criteria function can be chosen in many different ways, resulting in different clustering behaviors.

Nearest – neighbor (elongated classes)
\[ d_{\min} (\mathcal{N}_i, \mathcal{N}_j) = \min_{x \in \mathcal{N}_i, y \in \mathcal{N}_j} (x, y) \]

Furthest – neighbor (compact classes)
\[ d_{\max} (\mathcal{N}_i, \mathcal{N}_j) = \max_{x \in \mathcal{N}_i, y \in \mathcal{N}_j} (x, y) \]
- Starting from every sample in a cluster
- Merge them according to some criteria function
- Until two clusters exist
In Reality

- With $n$ objects, the distance matrix is $n \times n$
- For large databases, it is computationally expensive to compute and store the matrix
- Solution: storing only $k$ nearest clusters in the distance matrix: complexity is $k \times n$
Divisive Clustering

- Less often used
- Have to be careful that criterion function usually decreases monotonically (the samples become purer each time)
- Natural grouping is the one where a large drop in impurity occurs

![Impurity vs Iteration Graph]
ISODATA Algorithm

- Iterative self-organizing data analysis technique
- A tried-and-true clustering algorithm
- Dynamically update # of clusters
- Can go both top-down (split) and bottom-up (merge) directions
Notation:

\[ T \] threshold on number of samples in a cluster
\[ N_D \] approximate (desired) number of clusters
\[ \sigma_s \] maximum spread for splitting
\[ D_m \] maximum distance for merging
\[ N_{\text{max}} \] maximum number of clusters that can be merged
Algorithm

1. Cluster the existing data into $N_c$ clusters but eliminate any data and classes with fewer than $T$ members, decrease $N_c$. Exit when classification of samples has not changed.
2. If iteration odd and $N_c \leq \frac{N_D}{2}$ or $N_c \leq 2N_D$
   - Split clusters whose samples are sufficiently disjoint, increase $N_c$
   - If any clusters have been split, go to 1
3. Merge any pair of clusters whose samples are sufficiently close
4. Go to step 1
Step 1

Parameter $T$

Classify samples according to nearest mean

Change in classification?

- yes
  - Discard samples in clusters $< T$ samples, decrease $Nc$
  - Recompute sample mean of each cluster

- no
  - exit
Step 2

- Compute for each cluster
  \[
d_k = \frac{1}{N_k} \sum_{y^{(i)} \in \omega_k} |y^{(i)} - \mu_k|
  \]
  \[
  \sigma_k^2 = \max_j \frac{1}{N_k} \sum_{y^{(i)} \in \omega_k} |y^{(i)}_j - \mu_{k,j}|
  \]
  Compute globally
  \[
d = \frac{1}{N} \sum_{k=1}^{N_c} N_k d_k
  \]
\[ \sigma_s^2, N_D, T \]

\[ \forall k, \sigma_k^2 < \sigma_s^2 \]

\[ d_k > d \]

\[ \text{and } N_k > 2T + 1 \]

\[ \text{or } N_c < \frac{N_d}{2} \]

\[ \text{yes} \quad \text{no} \]

**Yes**

- Split cluster
- \( N_c = N_c + 1 \)

Cluster centers are displaced in opposite direction along the axis of largest variance
Step 3

- Compute for each pairs of clusters

\[ d_{ij} = | \mu_i - \mu_j | \]

Sort distance from smallest to largest less than \( Dm \)
\[ \forall d_{ij} < D_m \]

\# of merge < \( N_{\text{max}} \)

Neither cluster i nor j merged?

- yes
  - Merge i and j
  - \( N_c = N_c - 1 \)

- no

\[ \mu' = \frac{1}{N_i + N_j} \left( N_i \mu_i + N_j \mu_j \right) \]
Spectral Clustering

- Graph theoretical approach
- (Advanced) linear algebra is really important
- A field by itself
- Cover the basics here
Graph notations

- Undirected graph $G = (V, E)$
  - $V = \{v_1, ..., v_n\}$ are nodes (samples)
  - $E = \{e_{i,j} | 0 \leq i, j < n\}$ are edges, with weights (similarity) $w_{i,j}$
  - $W$: adjacency matrix with entries $w_{i,j}$
  - $D$: degree matrix (diagonal) with entries
  - $A$: subset of vertices, $\bar{A}$: complement $V \backslash A$
  - $1_A = (f_1, ..., f_n)^T$, $f_i = 1$ if $i$ in $A$ and 0 otherwise

\[ d_i = \sum_{j=1}^{n} w_{ij}. \]
More graph notations

- **Size of subset A in V**
  - Based on # of vertices
  - Based on its connections

- **Connected component A**
  - Any two vertices in A can be joined by a path where all intermediate points lie in A
  - No connection between A and \( \overline{A} \)

- **Partition with \( A_1, \ldots, A_k \)**, if

\[
A_i \cap A_j = \emptyset \\
A_1 \cup \ldots \cup A_k = V
\]

\( |A| := \) the number of vertices in A

\( \text{vol}(A) := \sum_{i \in A} d_i. \)
Similarity Definitions

- RBF (fully connected or thresholded):
  - E.g., Gaussian kernel gives $w_{i,j} = \exp\left(-\|x_i - x_j\|^2/(2\sigma^2)\right)$

- $\varepsilon$-neighborhood graph:
  - Connect all points whose pairwise distances less than $\varepsilon$

- K-nearest neighbor:
  - $v_i$ and $v_j$ are neighbors if either one is a kNN of the other

- Mutual k-nearest neighbor:
  - $v_i$ and $v_j$ are neighbors if both are a kNN of the other
Graph Laplacian: $L = D - W$

**Proposition 1 (Properties of $L$)** The matrix $L$ satisfies the following properties:

1. For every vector $f \in \mathbb{R}^n$ we have
   \[
   f'Lf = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.
   \]

2. $L$ is symmetric and positive semi-definite.

3. The smallest eigenvalue of $L$ is 0, the corresponding eigenvector is the constant one vector $\mathbf{1}$.

4. $L$ has $n$ non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$.

*Proof.*

Part (1): By the definition of $d_i$,
\[
   f'Lf = f'Df - f'Wf = \sum_{i=1}^{n} d_i f_i^2 - \sum_{i,j=1}^{n} f_i f_j w_{ij}
\]
\[
   = \frac{1}{2} \left( \sum_{i=1}^{n} d_i f_i^2 - 2 \sum_{i,j=1}^{n} f_i f_j w_{ij} + \sum_{j=1}^{n} d_j f_j^2 \right) = \frac{1}{2} \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.
\]

Part (2): The symmetry of $L$ follows directly from the symmetry of $W$ and $D$. The positive semi-definiteness is a direct consequence of Part (1), which shows that $f'Lf \geq 0$ for all $f \in \mathbb{R}^n$.

Part (3): Obvious.

Part (4) is a direct consequence of Parts (1) - (3).
One connected component

Proposition 2 (Number of connected components and the spectrum of $L$) Let $G$ be an undirected graph with non-negative weights. Then the multiplicity $k$ of the eigenvalue 0 of $L$ equals the number of connected components $A_1, \ldots, A_k$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $1_{A_1}, \ldots, 1_{A_k}$ of those components.

Proof. We start with the case $k = 1$, that is the graph is connected. Assume that $f$ is an eigenvector with eigenvalue 0. Then we know that

$$0 = f^t L f = \sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2.$$ 

As the weights $w_{ij}$ are non-negative, this sum can only vanish if all terms $w_{ij}(f_i - f_j)^2$ vanish. Thus, if two vertices $v_i$ and $v_j$ are connected (i.e., $w_{ij} > 0$), then $f_i$ needs to equal $f_j$. With this argument we can see that $f$ needs to be constant for all vertices which can be connected by a path in the graph. Moreover, as all vertices of a connected component in an undirected graph can be connected by a path, $f$ needs to be constant on the whole connected component. In a graph consisting of only one connected component we thus only have the constant one vector $1$ as eigenvector with eigenvalue 0, which obviously is the indicator vector of the connected component.
**Multiple connected components**

**Proposition 2 (Number of connected components and the spectrum of L)** Let \( G \) be an undirected graph with non-negative weights. Then the multiplicity \( k \) of the eigenvalue 0 of \( L \) equals the number of connected components \( A_1, \ldots, A_k \) in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors \( 1_{A_1}, \ldots, 1_{A_k} \) of those components.

Now consider the case of \( k \) connected components. Without loss of generality we assume that the vertices are ordered according to the connected components they belong to. In this case, the adjacency matrix \( W \) has a block diagonal form, and the same is true for the matrix \( L \):

\[
L = \begin{pmatrix}
L_1 & & \\
& L_2 & \\
& & \ddots \\
& & & L_k
\end{pmatrix}
\]

Note that each of the blocks \( L_i \) is a proper graph Laplacian on its own, namely the Laplacian corresponding to the subgraph of the \( i \)-th connected component. As it is the case for all block diagonal matrices, we know that the spectrum of \( L \) is given by the union of the spectra of \( L_i \), and the corresponding eigenvectors of \( L \) are the eigenvectors of \( L_i \), filled with 0 at the positions of the other blocks. As each \( L_i \) is a graph Laplacian of a connected graph, we know that every \( L_i \) has eigenvalue 0 with multiplicity 1, and the corresponding eigenvector is the constant one vector on the \( i \)-th connected component. Thus, the matrix \( L \) has as many eigenvalues 0 as there are connected components, and the corresponding eigenvectors are the indicator vectors of the connected components. \( \square \)
Normalized Laplacian

\[ L_{\text{sym}} := D^{-1/2}LD^{-1/2} = I - D^{-1/2}WD^{-1/2} \]
\[ L_{\text{rw}} := D^{-1}L = I - D^{-1}W. \]

**Proposition 3 (Properties of** \( L_{\text{sym}} \) **and** \( L_{\text{rw}} \) **)** The normalized Laplacians satisfy the following properties:

1. For every \( f \in \mathbb{R}^n \) we have
   \[
   f^t L_{\text{sym}} f = \frac{1}{2} \sum_{i,j=1}^n w_{ij} \left( \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right)^2.
   \]

2. \( \lambda \) is an eigenvalue of \( L_{\text{rw}} \) with eigenvector \( u \) if and only if \( \lambda \) is an eigenvalue of \( L_{\text{sym}} \) with eigenvector \( w = D^{1/2}u \).

3. \( \lambda \) is an eigenvalue of \( L_{\text{rw}} \) with eigenvector \( u \) if and only if \( \lambda \) and \( u \) solve the generalized eigenproblem \( Lu = \lambda Du \).

4. \( 0 \) is an eigenvalue of \( L_{\text{rw}} \) with the constant one vector \( 1 \) as eigenvector. \( 0 \) is an eigenvalue of \( L_{\text{sym}} \) with eigenvector \( D^{1/2}1 \).

5. \( L_{\text{sym}} \) and \( L_{\text{rw}} \) are positive semi-definite and have \( n \) non-negative real-valued eigenvalues \( 0 = \lambda_1 \leq \ldots \leq \lambda_n \).
Unnormalized Spectral Clustering

Unnormalized spectral clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct.
- Construct a similarity graph by one of the ways described in Section 2. Let $W$ be its weighted adjacency matrix.
- Compute the unnormalized Laplacian $L$.
- Compute the first $k$ eigenvectors $u_1, \ldots, u_k$ of $L$.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $U$.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the $k$-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{j \mid y_j \in C_i\}$. 
Normalized Spectral Clustering according to Shi and Malik (2000)

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number $k$ of clusters to construct.
- Construct a similarity graph by one of the ways described in Section 2. Let $W$ be its weighted adjacency matrix.
- Compute the unnormalized Laplacian $L$.
- Compute the first $k$ generalized eigenvectors $u_1, \ldots, u_k$ of the generalized eigenproblem $Lu = \lambda Du$.
- Let $U \in \mathbb{R}^{n \times k}$ be the matrix containing the vectors $u_1, \ldots, u_k$ as columns.
- For $i = 1, \ldots, n$, let $y_i \in \mathbb{R}^k$ be the vector corresponding to the $i$-th row of $U$.
- Cluster the points $(y_i)_{i=1,\ldots,n}$ in $\mathbb{R}^k$ with the $k$-means algorithm into clusters $C_1, \ldots, C_k$.

Output: Clusters $A_1, \ldots, A_k$ with $A_i = \{ j \mid y_j \in C_i \}$. 
Toy Example

- Random sample of 200 points drawn from 4 Gaussians
- Similarity based on $s(x_i, x_j) = \exp\left(-\frac{|x_i - x_j|^2}{(2\sigma^2)}\right)$ with $\sigma = 1$.
- Graph
  - Fully connected or
  - 10 nearest neighbors
Min-Cut Formulation

- If the edge weight represents degree of similarity, optimal bi-partitioning of a graph is to minimize the cut (so called min-cut problem).
- Intuition: Cut the connection between dis-similar samples, hence, edge weight (similarity) should be small.

$$\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

Feature space
Problem with Min-cut

- Tends to cut out small regions
  - Sum weight (green + red + blue) = constant
  - Min-cut minimizes sum of weights of blue edges, with no regard to green and red (half of the picture)
Remedy

- Distribute the total weight such that
  - Sum of weights of the blue edges are minimized
  - Max between group variance
  - Sum of weights of the red (green) edges are maximized
  - Min within group variance
**Normalized Cut**

- Penalize cutting out small, isolated clusters

\[
N_{\text{cut}}(A, B) = \frac{\text{cut}(A, B)}{\text{asso}(A, V)} + \frac{\text{cut}(A, B)}{\text{asso}(B, V)} = \frac{\text{blue}}{\text{total} - \text{red}} + \frac{\text{blue}}{\text{total} - \text{green}}
\]

\[
\text{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v) \quad \text{(blue)}
\]

\[
\text{asso}(A, V) = \sum_{u \in A, v \in V} w(u, v) \quad \text{(green + blue = total - red)}
\]

\[
\text{asso}(B, V) = \sum_{u \in A, v \in V} w(u, v) \quad \text{(red + blue = total - green)}
\]

\(V\) : full vertex set
Normalized Cut (cont.)

- Penalize cutting out small, isolated clusters
  - Small blue
  - Small red (or green)

\[
N_{\text{cut}}(A, B) = \frac{\text{cut}(A, B)}{\text{asso}(A, V)} + \frac{\text{cut}(A, B)}{\text{asso}(B, V)} = \frac{\text{blue}}{\text{total} - \text{red}} + \frac{\text{blue}}{\text{total} - \text{green}}
\]
Intuition

- Assoc reflects *intra*-class connection which should be maximized
- Ncut represents *inter*-class connection which should be minimized

\[ asso(A, V) = \sum_{u \in A, v \in V} w(u, v) = \sum_{u \in A, v \in A} w(u, v) + \sum_{u \in A, v \in B} w(u, v) = \sum_{u \in A, v \in A} w(u, v) + cut(A, B) \]

\[
\sum_{u \in A, v \in A} w(u, v) + \frac{cut(A, B)}{asso(A, V)} = 1
\]

\[
assoc(A, A) + \frac{cut(A, B)}{asso(A, V)} = 1 \quad assoc(B, B) + \frac{cut(A, B)}{asso(B, V)} = 1
\]

\[
assoc(A, A) + assoc(B, B) + \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)} = 2
\]

\[Nassoc(A, B) + Ncut(A, B) = 2\]
Solution

❖ How do you define similarity?

\[ d(i, j) = \sum_{k} w_k |v_k^i - v_k^j| \]

- Multiple measurements (i.e., a feature vector) can be used

❖ How do you find the minimal normalized cut?

- Solution turns out to be a generalized eigen value problem!
\[
x = \begin{bmatrix}
1 \\
-1 \\
\vdots \\
1 \\
\end{bmatrix}_{N \times 1}, \quad x_i = \begin{cases} 
1 & i \in A \\
-1 & i \in B \\
\end{cases}, \quad D = \begin{bmatrix} 
\begin{array}{c}
d_1 \\
0 \\
\vdots \\
0 \\
\end{array}
\end{bmatrix}_{N \times N}, \quad d_i = \sum_j w(i, j)
\]

\[
W = \begin{bmatrix} 
w(1,1) & w(1,2) & \cdots & w(1,N) \\
w(2,1) & w(2,2) & \cdots & w(2,N) \\
\vdots \\
w(N,1) & w(N,2) & \cdots & w(N,N) \\
\end{bmatrix}_{N \times N}
\]

\[
k = \frac{\sum_{x_i>0} d_i}{\sum_i d_i}
\]

\[
N = |V|
\]
\[ \text{Ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{asso}(A, V)} + \frac{\text{cut}(A, B)}{\text{asso}(B, V)} \]

\[ = \sum_{x_i > 0, x_j < 0} -w_{ij}x_ix_j + \sum_{x_i < 0, x_j > 0} -w_{ij}x_ix_j \]

\[ = \sum_{x_i > 0} d_i + \sum_{x_i < 0} d_i \]

\[ = \frac{(1 + x)^T(D - W)(1 + x)}{k1^T D1} + \frac{(1 - x)^T(D - W)(1 - x)}{(1 - k)1^T D1} \]

\[ \frac{1 + x}{2} = \begin{cases} 1 & x_i \in A \\ 0 & x_i \in B \end{cases} \]

\[ \frac{1 - x}{2} = \begin{cases} 0 & x_i \in A \\ 1 & x_i \in B \end{cases} \]

\[ d_i = \sum_{x_j < 0} w_{ij}x_j + \sum_{x_k > 0} w_{ik}x_k \Rightarrow \sum_{x_j < 0} w_{ij}x_j = d_i - \sum_{x_k > 0} w_{ik}x_k \]
\[
\min_x Ncut(x) = \min_y \frac{y^T(D - W)y}{y^T Dy}
\]

\[
y = (1 + x) - b(1 - x)
\]

\[
b = \frac{k}{1 - k}
\]

\[
(D - W)y = \lambda Dy
\]

\[
D^{\frac{1}{2}}(D - W)D^{\frac{1}{2}}z = \lambda z \quad z = D^{\frac{1}{2}}y
\]

- A symmetric semi-positive-definite matrix
  - Real, \( \geq 0 \) eigenvalues
  - Orthogonal eigen vectors

\[
eigenvector r : z_o = D^{\frac{1}{2}}1
\]

\[
eigenvalue : 0
\]
- Hence, the second smallest eigen vector contains the minimal cut solution (in floating point format)
- Even though computing all eigen vectors/values are expensive $O(n^3)$, computing a small number of those are not that expensive (Lanczos method)
- Recursive applications of the procedure
Results