Data Mining and Analysis: Fundamental Concepts and Algorithms
dataminingbook.info

Mohammed J. Zaki\textsuperscript{1} Wagner Meira Jr.\textsuperscript{2}

\textsuperscript{1}Department of Computer Science
Rensselaer Polytechnic Institute, Troy, NY, USA

\textsuperscript{2}Department of Computer Science
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

Chapter 16: Spectral & Graph Clustering
Given a dataset $D = \{x_i\}_{i=1}^n$ consisting of $n$ points in $\mathbb{R}^d$, let $A$ denote the $n \times n$ symmetric similarity matrix between the points, given as

$$ A = \begin{pmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n1} & a_{n2} & \cdots & a_{nn}
\end{pmatrix} $$

where $A(i, j) = a_{ij}$ denotes the similarity or affinity between points $x_i$ and $x_j$. We require the similarity to be symmetric and non-negative, that is, $a_{ij} = a_{ji}$ and $a_{ij} \geq 0$, respectively.

The matrix $A$ is the weighted adjacency matrix for the data graph. If all affinities are 0 or 1, then $A$ represents the regular adjacency relationship between the vertices.
Iris Similarity Graph: Mutual Nearest Neighbors

$|V| = n = 150, |E| = m = 1730$
For a vertex $x_i$, let $d_i$ denote the degree of the vertex, defined as

$$d_i = \sum_{j=1}^{n} a_{ij}$$

We define the *degree matrix* $\Delta$ of graph $G$ as the $n \times n$ diagonal matrix:

$$\Delta = \begin{pmatrix}
    d_1 & 0 & \cdots & 0 \\
    0 & d_2 & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & d_n
\end{pmatrix} = \begin{pmatrix}
    \sum_{j=1}^{n} a_{1j} & 0 & \cdots & 0 \\
    0 & \sum_{j=1}^{n} a_{2j} & \cdots & 0 \\
    \vdots & \vdots & \ddots & \vdots \\
    0 & 0 & \cdots & \sum_{j=1}^{n} a_{nj}
\end{pmatrix}$$

$\Delta$ can be compactly written as $\Delta(i, i) = d_i$ for all $1 \leq i \leq n$. 

Zaki & Meira Jr. (RPI and UFMG)  Data Mining and Analysis  Chapter 16: Spectral & Graph Clustering
The normalized adjacency matrix is obtained by dividing each row of the adjacency matrix by the degree of the corresponding node. Given the weighted adjacency matrix $A$ for a graph $G$, its normalized adjacency matrix is defined as

$$M = \Delta^{-1}A = \begin{pmatrix} \frac{a_{11}}{d_1} & \frac{a_{12}}{d_1} & \cdots & \frac{a_{1n}}{d_1} \\ \frac{a_{21}}{d_2} & \frac{a_{22}}{d_2} & \cdots & \frac{a_{2n}}{d_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{a_{n1}}{d_n} & \frac{a_{n2}}{d_n} & \cdots & \frac{a_{nn}}{d_n} \end{pmatrix}$$

Because $A$ is assumed to have non-negative elements, this implies that each element of $M$, namely $m_{ij}$ is also non-negative, as $m_{ij} = \frac{a_{ij}}{d_i} \geq 0$.

Each row in $M$ sums to 1, which implies that 1 is an eigenvalue of $M$. In fact, $\lambda_1 = 1$ is the largest eigenvalue of $M$, and the other eigenvalues satisfy the property that $|\lambda_i| \leq 1$. Because $M$ is not symmetric, its eigenvectors are not necessarily orthogonal.
Example Graph: Adjacency and Degree Matrices

Its adjacency and degree matrices are given as

\[
A = \begin{pmatrix}
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0
\end{pmatrix}
\]

\[
\Delta = \begin{pmatrix}
3 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 3 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 3 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 3 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 3
\end{pmatrix}
\]
The normalized adjacency matrix is as follows:

\[ M = \Delta^{-1}A = \begin{pmatrix}
0 & 0.33 & 0 & 0.33 & 0 & 0.33 & 0 \\
0.33 & 0 & 0.33 & 0.33 & 0 & 0 & 0 \\
0 & 0.33 & 0 & 0.33 & 0 & 0 & 0.33 \\
0.25 & 0.25 & 0.25 & 0 & 0.25 & 0 & 0 \\
0 & 0 & 0 & 0.33 & 0 & 0.33 & 0.33 \\
0.33 & 0 & 0 & 0 & 0.33 & 0 & 0.33 \\
0 & 0 & 0.33 & 0 & 0.33 & 0.33 & 0
\end{pmatrix} \]

The eigenvalues of \( M \) are: \( \lambda_1 = 1, \lambda_2 = 0.483, \lambda_3 = 0.206, \lambda_4 = -0.045, \lambda_5 = -0.405, \lambda_6 = -0.539, \lambda_7 = -0.7 \)
Graph Laplacian Matrix

The *Laplacian matrix* of a graph is defined as

\[
L = \Delta - A
\]

\[
= \begin{pmatrix}
\sum_{j=1}^{n} a_{1j} & 0 & \cdots & 0 \\
0 & \sum_{j=1}^{n} a_{2j} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & \sum_{j=1}^{n} a_{nj}
\end{pmatrix}
- \begin{pmatrix}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{n1} & a_{n2} & \cdots & a_{nn}
\end{pmatrix}
\]

\[
L = \begin{pmatrix}
\sum_{j \neq 1} a_{1j} & -a_{12} & \cdots & -a_{1n} \\
-a_{21} & \sum_{j \neq 2} a_{2j} & \cdots & -a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
-a_{n1} & -a_{n2} & \cdots & \sum_{j \neq n} a_{nj}
\end{pmatrix}
\]

\(L\) is a symmetric, positive semidefinite matrix. This means that \(L\) has \(n\) real, non-negative eigenvalues, which can be arranged in decreasing order as follows: \(\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0\). Because \(L\) is symmetric, its eigenvectors are orthonormal. The rank of \(L\) is at most \(n - 1\), and the smallest eigenvalue is \(\lambda_n = 0\).
\[ \Delta = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} \]

\[ \Delta = \Delta^{1/2} \Delta^{1/2} \]

\[ \Delta^{-1/2} \begin{pmatrix} \Delta_1 \\ \sqrt{d_1} \\ \sqrt{d_2} \\ \vdots \\ \sqrt{d_n} \end{pmatrix} = \begin{pmatrix} \Delta_1 \\ \sqrt{d_1} \\ \sqrt{d_2} \\ \vdots \\ \sqrt{d_n} \end{pmatrix} \times \begin{pmatrix} \Delta_1^{-1/2} \\ \sqrt{d_1}^{-1/2} \\ \sqrt{d_2}^{-1/2} \\ \vdots \\ \sqrt{d_n}^{-1/2} \end{pmatrix} \]

Asymmetric Laplacian

Symmetric Laplacian

Normalized
The graph Laplacian is given as

\[
L = \Delta - A = \begin{pmatrix}
3 & -1 & 0 & -1 & 0 & -1 & 0 \\
-1 & 3 & -1 & -1 & 0 & 0 & 0 \\
0 & -1 & 3 & -1 & 0 & 0 & -1 \\
-1 & -1 & -1 & 4 & -1 & 0 & 0 \\
0 & 0 & 0 & -1 & 3 & -1 & -1 \\
-1 & 0 & 0 & 0 & -1 & 3 & -1 \\
0 & 0 & -1 & 0 & -1 & -1 & 3
\end{pmatrix}
\]

The eigenvalues of \( L \) are as follows: \( \lambda_1 = 5.618, \lambda_2 = 4.618, \lambda_3 = 4.414, \lambda_4 = 3.382, \lambda_5 = 2.382, \lambda_6 = 1.586, \lambda_7 = 0 \)
Normalized Laplacian Matrices

The *normalized symmetric Laplacian matrix* of a graph is defined as

\[
L^s = \Delta^{-1/2} L \Delta^{-1/2} = \\
\begin{pmatrix}
\frac{\sum_{i \neq 1} a_{1j}}{\sqrt{d_1 d_1}} & \frac{a_{12}}{\sqrt{d_1 d_2}} & \ldots & \frac{a_{1n}}{\sqrt{d_1 d_n}} \\
-\frac{a_{21}}{\sqrt{d_2 d_1}} & \frac{\sum_{j \neq 2} a_{2j}}{\sqrt{d_2 d_2}} & \ldots & -\frac{a_{2n}}{\sqrt{d_2 d_n}} \\
\vdots & \vdots & \ddots & \vdots \\
-\frac{a_{n1}}{\sqrt{d_n d_1}} & -\frac{a_{n2}}{\sqrt{d_n d_2}} & \ldots & \frac{\sum_{j \neq n} a_{nj}}{\sqrt{d_n d_n}}
\end{pmatrix}
\]

\(L^s\) is a symmetric, positive semidefinite matrix, with rank at most \(n - 1\). The smallest eigenvalue \(\lambda_n = 0\).

The *normalized asymmetric Laplacian* matrix is defined as

\[
L^a = \Delta^{-1} L = \\
\begin{pmatrix}
\frac{\sum_{j \neq 1} a_{1j}}{d_1} & -\frac{a_{12}}{d_1} & \ldots & \frac{a_{1n}}{d_1} \\
-\frac{a_{21}}{d_2} & \frac{\sum_{j \neq 2} a_{2j}}{d_2} & \ldots & -\frac{a_{2n}}{d_2} \\
\vdots & \vdots & \ddots & \vdots \\
-\frac{a_{n1}}{d_n} & -\frac{a_{n2}}{d_n} & \ldots & \frac{\sum_{j \neq n} a_{nj}}{d_n}
\end{pmatrix}
\]

\(L^a\) is also a positive semi-definite matrix with \(n\) real eigenvalues \(\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n = 0\).
The normalized symmetric Laplacian is given as

\[
L^s = \begin{pmatrix}
1 & -0.33 & 0 & -0.29 & 0 & -0.33 & 0 \\
-0.33 & 1 & -0.33 & -0.29 & 0 & 0 & 0 \\
0 & -0.33 & 1 & -0.29 & 0 & 0 & -0.33 \\
-0.29 & -0.29 & -0.29 & 1 & -0.29 & 0 & 0 \\
0 & 0 & 0 & -0.29 & 1 & -0.33 & -0.33 \\
-0.33 & 0 & 0 & 0 & -0.33 & 1 & -0.33 \\
0 & 0 & -0.33 & 0 & -0.33 & -0.33 & 1
\end{pmatrix}
\]

The eigenvalues of \( L^s \) are as follows: \( \lambda_1 = 1.7, \lambda_2 = 1.539, \lambda_3 = 1.405, \lambda_4 = 1.045, \lambda_5 = 0.794, \lambda_6 = 0.517, \lambda_7 = 0 \).
The normalized asymmetric Laplacian matrix is given as

\[ \mathbf{L}^a = \mathbf{\Delta}^{-1} \mathbf{L} = \begin{pmatrix}
1 & -0.33 & 0 & -0.33 & 0 & -0.33 & 0 \\
-0.33 & 1 & -0.33 & -0.33 & 0 & 0 & 0 \\
0 & -0.33 & 1 & -0.33 & 0 & 0 & -0.33 \\
-0.25 & -0.25 & -0.25 & 1 & -0.25 & 0 & 0 \\
0 & 0 & 0 & -0.33 & 1 & -0.33 & -0.33 \\
-0.33 & 0 & 0 & 0 & -0.33 & 1 & -0.33 \\
0 & 0 & -0.33 & 0 & -0.33 & -0.33 & 1
\end{pmatrix} \]

The eigenvalues of \( \mathbf{L}^a \) are identical to those for \( \mathbf{L}^s \), namely

\[ \lambda_1 = 1.7, \lambda_2 = 1.539, \lambda_3 = 1.405, \lambda_4 = 0.794, \lambda_5 = 0.517, \lambda_6 = 0, \lambda_7 = 0. \]
$W(s, r) = \sum_{i \in S} w_{ij}$

weight of the cut jet
\( G = (V, E) \)

\( C = \{ C_1, C_2, \ldots, C_k \} \)

\# of clusters

K-way cut or partitioning

\[ C_i = \text{set of all vertices belonging to cluster } i \]

\[ V = \bigcup_{i=1}^{k} C_i \]

\[ C_i \cap C_j = \emptyset \]

\[ W(C_i, \overline{C_i}) \]
Ratio objective:

\[ \sum_{i=1}^{k} \frac{w(c_i, \overline{c_i})}{|c_i|} \]

\[ w(c_1, \overline{c_1}) + w(c_2, \overline{c_2}) + w(c_3, \overline{c_3}) \]
Normalized cor

$$\min \sum_{i=1}^{k} w(C_i, \overline{C_i}) \frac{Vol(C_i)}{Vol(C_i)}$$

$$Vol(C_i) = W(C_i, V) = \sum_{u \in C_i} \text{degree}(u)$$
A $k$-way cut in a graph is a partitioning or clustering of the vertex set, given as $\mathcal{C} = \{C_1, \ldots, C_k\}$. We require $\mathcal{C}$ to optimize some objective function that captures the intuition that nodes within a cluster should have high similarity, and nodes from different clusters should have low similarity.

Given a weighted graph $G$ defined by its similarity matrix $A$, let $S, T \subseteq V$ be any two subsets of the vertices. We denote by $W(S, T)$ the sum of the weights on all edges with one vertex in $S$ and the other in $T$, given as

$$W(S, T) = \sum_{v_i \in S} \sum_{v_j \in T} a_{ij}$$

Given $S \subseteq V$, we denote by $\overline{S}$ the complementary set of vertices, that is, $\overline{S} = V - S$. A (vertex) cut in a graph is defined as a partitioning of $V$ into $S \subseteq V$ and $\overline{S}$. The weight of the cut or cut weight is defined as the sum of all the weights on edges between vertices in $S$ and $\overline{S}$, given as $W(S, \overline{S})$.
Cuts and Matrix Operations

Given a clustering $C = \{C_1, \ldots, C_k\}$ comprising $k$ clusters. Let $c_i \in \{0, 1\}^n$ be the *cluster indicator vector* that records the cluster membership for cluster $C_i$, defined as

$$c_{ij} = \begin{cases} 1 & \text{if } v_j \in C_i \\ 0 & \text{if } v_j \notin C_i \end{cases}$$

The cluster size can be written as

$$|C_i| = c_i^T c_i = \|c_i\|^2$$

The *volume* of a cluster $C_i$ is defined as the sum of all the weights on edges with one end in cluster $C_i$:

$$vol(C_i) = W(C_i, V) = \sum_{v_r \in C_i} d_r = \sum_{v_r \in C_i} c_{ir} d_r c_{ir} = \sum_{r=1}^{n} \sum_{s=1}^{n} c_{ir} \Delta_{rs} c_{is} = c_i^T \Delta c_i$$

The sum of weights of all internal edges is:

$$W(C_i, C_i) = \sum_{v_r \in C_i} \sum_{v_s \in C_i} a_{rs} = \sum_{r=1}^{n} \sum_{s=1}^{n} c_{ir} a_{rs} c_{is} = c_i^T A c_i$$

We can get the sum of weights for all the external edges as follows:

$$W(C_i, \overline{C_i}) = \sum_{v_r \in C_i} \sum_{v_s \in V - C_i} a_{rs} = W(C_i, V) - W(C_i, C_i) = c_i^T (\Delta - A) c_i = c_i^T L c_i$$
\( \mathbf{c}_i \) ← cluster indicator vector

\( \| \mathbf{c}_i \|^2 = \mathbf{c}_i^T \mathbf{c}_i = |c_i| \)

\( \text{Vol}(C_i) = \mathbf{c}_i^T \Delta \mathbf{c}_i = \sum_{v \in C_i} d_i \)
Clustering Objective Functions: Ratio Cut

The clustering objective function can be formulated as an optimization problem over the $k$-way cut $C = \{C_1, \ldots, C_k\}$. The *ratio cut* objective is defined over a $k$-way cut as follows:

$$
\min_C J_{rc}(C) = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{|C_i|} = \sum_{i=1}^{k} \frac{c_i^T L c_i}{c_i^T c_i} = \sum_{i=1}^{k} \frac{c_i^T L c_i}{\|c_i\|^2}
$$

Ratio cut tries to minimize the sum of the similarities from a cluster $C_i$ to other points not in the cluster $\overline{C_i}$, taking into account the size of each cluster. Unfortunately, for binary cluster indicator vectors $c_i$, the ratio cut objective is NP-hard. An obvious relaxation is to allow $c_i$ to take on any real value. In this case, we can rewrite the objective as

$$
\min_C J_{rc}(C) = \sum_{i=1}^{k} \frac{c_i^T L c_i}{\|c_i\|^2} = \sum_{i=1}^{k} \left( \frac{c_i}{\|c_i\|} \right)^T L \left( \frac{c_i}{\|c_i\|} \right) = \sum_{i=1}^{k} u_i^T L u_i
$$

The optimal solution comprises the eigenvectors corresponding to the $k$ smallest eigenvalues of $L$, i.e., the eigenvectors $u_n, u_{n-1}, \ldots, u_{n-k+1}$ represent the relaxed cluster indicator vectors.
is similar to ratio cut, except that it divides the cut weight of each cluster by the volume of a cluster instead of its size. The objective function is given as

$$\min_C J_{nc}(C) = \sum_{i=1}^{k} \frac{W(C_i, \overline{C_i})}{vol(C_i)} = \sum_{i=1}^{k} \frac{c_i^T L c_i}{c_i^T \Delta c_i}$$

We can obtain an optimal solution by allowing $c_i$ to be an arbitrary real vector.

The optimal solution comprise the eigenvectors corresponding to the $k$ smallest eigenvalues of either the normalized symmetric or asymmetric Laplacian matrices, $L^s$ and $L^a$. 

$\mathbf{u}_i \in \mathbb{R}^n$
Spectral Clustering Algorithm

The spectral clustering algorithm takes a dataset \( D \) as input and computes the similarity matrix \( A \). For normalized cut we chose either \( L^s \) or \( L^a \), whereas for ratio cut we choose \( L \). Next, we compute the \( k \) smallest eigenvalues and corresponding eigenvectors of the chosen matrix.

The main problem is that the eigenvectors \( u_i \) are not binary, and thus it is not immediately clear how we can assign points to clusters.

One solution to this problem is to treat the \( n \times k \) matrix of eigenvectors as a new data matrix:

\[
U = \begin{pmatrix}
  u_1 & u_2 & \cdots & u_k
\end{pmatrix}
\]

\( \rightarrow \) normalize rows

\[
\begin{pmatrix}
  - & y_1^T & - & \\
  - & y_2^T & - & \\
  \vdots & & \ddots & \\
  - & y_n^T & - & \\
\end{pmatrix} = Y
\]

We then cluster the new points in \( Y \) into \( k \) clusters via the K-means algorithm or any other fast clustering method to obtain binary cluster indicator vectors \( c_i \).
Spectral Clustering Algorithm

**Spectral Clustering** \((D, k)\):

1. Compute the similarity matrix \(A \in \mathbb{R}^{n \times n}\)
2. **if** \(\text{ratio cut} \) **then** \(B \leftarrow L\)
3. **else if** \(\text{normalized cut} \) **then** \(B \leftarrow L^s \text{ or } L^a\)
4. Solve \(Bu_i = \lambda_i u_i\) for \(i = n, \ldots, n-k+1\), where \(\lambda_n \leq \lambda_{n-1} \leq \cdots \leq \lambda_{n-k+1}\)
5. \(U \leftarrow (u_n \ u_{n-1} \cdots u_{n-k+1})\)
6. \(Y \leftarrow \) normalize rows of \(U\)
7. \(C \leftarrow \{C_1, \ldots, C_k\} \) via K-means on \(Y\)
Spectral Clustering on Example Graph

$k = 2$, normalized cut (normalized asymmetric Laplacian)
Normalized Cut on Iris Graph

$k = 3$, normalized asymmetric Laplacian

<table>
<thead>
<tr>
<th></th>
<th>setosa</th>
<th>virginica</th>
<th>versicolor</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1$ (triangle)</td>
<td>50</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>$C_2$ (square)</td>
<td>0</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>$C_3$ (circle)</td>
<td>0</td>
<td>14</td>
<td>46</td>
</tr>
</tbody>
</table>
Maximization Objectives: Average Cut

The *average weight* objective is defined as

$$\max_{C} J_{aw}(C) = \sum_{i=1}^{k} \frac{W(C_i, C_i)}{|C_i|} = \sum_{i=1}^{k} \frac{c_i^T A c_i}{c_i^T c_i} = \sum_{i=1}^{k} u_i^T A u_i$$

where $u_i$ is an arbitrary real vector, which is a relaxation of the binary cluster indicator vectors $c_i$.

We can maximize the objective by selecting the $k$ largest eigenvalues of $A$, and the corresponding eigenvectors.

$$\max_{C} J_{aw}(C) = u_1^T A u_1 + \cdots + u_k^T A u_k$$

$$= \lambda_1 + \cdots + \lambda_k$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$. In general, while $A$ is symmetric, it may not be positive semidefinite. This means that $A$ can have negative eigenvalues, and to maximize the objective we must consider only the positive eigenvalues and the corresponding eigenvectors.
Maximization Objectives: Modularity

Given $A$, the weighted adjacency matrix, the modularity of a clustering is the difference between the observed and expected fraction of weights on edges within the clusters. The clustering objective is given as

$$
\max_{C} J_Q(C) = \sum_{i=1}^{k} \left( \frac{c_i^T A c_i}{\text{tr}(\Delta)} - \frac{(d_i^T c_i)^2}{\text{tr}(\Delta)^2} \right) = \sum_{i=1}^{k} c_i^T Q c_i
$$

where $Q$ is the modularity matrix:

$$
Q = \frac{1}{\text{tr}(\Delta)} \left( A - \frac{d \cdot d^T}{\text{tr}(\Delta)} \right)
$$

The optimal solution comprises the eigenvectors corresponding to the $k$ largest eigenvalues of $Q$. Since $Q$ is symmetric, but not positive semidefinite, we use only the positive eigenvalues.
Markov Chain Clustering

A Markov chain is a discrete-time stochastic process over a set of states, in our case the set of vertices $V$.

The Markov chain makes a transition from one node to another at discrete timesteps $t = 1, 2, \ldots$, with the probability of making a transition from node $i$ to node $j$ given as $m_{ij}$.

Let the random variable $X_t$ denote the state at time $t$. The Markov property means that the probability distribution of $X_t$ over the states at time $t$ depends only on the probability distribution of $X_{t-1}$, that is,

$$P(X_t = i | X_0, X_1, \ldots, X_{t-1}) = P(X_t = i | X_{t-1})$$

Further, we assume that the Markov chain is homogeneous, that is, the transition probability

$$P(X_t = j | X_{t-1} = i) = m_{ij}$$

is independent of the time step $t$. 
The normalized adjacency matrix $M = \Delta^{-1} A$ can be interpreted as the $n \times n$ transition matrix where the entry $m_{ij} = \frac{a_{ij}}{d_i}$ is the probability of transitioning or jumping from node $i$ to node $j$ in the graph $G$.

The matrix $M$ is thus the transition matrix for a Markov chain or a Markov random walk on graph $G$. That is, Given node $i$ the transition matrix $M$ specifies the probabilities of reaching any other node $j$ in one time step.

In general, the transition probability matrix for $t$ time steps is given as

$$M^{t-1} \cdot M = M^t$$
A random walk on $G$ thus corresponds to taking successive powers of the transition matrix $M$.

Let $\pi_0$ specify the initial state probability vector at time $t = 0$. The state probability vector after $t$ steps is

$$\pi^T_t = \pi^T_{t-1}M = \pi^T_{t-2}M^2 = \cdots = \pi^T_0M^t$$

Equivalently, taking transpose on both sides, we get

$$\pi_t = (M^t)^T\pi_0 = (M^T)^t\pi_0$$

The state probability vector thus converges to the dominant eigenvector of $M^T$. 
Markov Clustering Algorithm

Consider a variation of the random walk, where the probability of transitioning from node $i$ to $j$ is inflated by taking each element $m_{ij}$ to the power $r \geq 1$. Given a transition matrix $M$, define the inflation operator $\Upsilon$ as follows:

$$
\Upsilon(M, r) = \left\{ \frac{(m_{ij})^r}{\sum_{a=1}^{n} (m_{ia})^r} \right\}_{i,j=1}^{n}
$$

The net effect of the inflation operator is to increase the higher probability transitions and decrease the lower probability transitions.

The Markov clustering algorithm (MCL) is an iterative method that interleaves matrix expansion and inflation steps. Matrix expansion corresponds to taking successive powers of the transition matrix, leading to random walks of longer lengths. On the other hand, matrix inflation makes the higher probability transitions even more likely and reduces the lower probability transitions.

MCL takes as input the inflation parameter $r \geq 1$. Higher values lead to more, smaller clusters, whereas smaller values lead to fewer, but larger clusters.
Markov Clustering Algorithm: MCL

The final clusters are found by enumerating the weakly connected components in the directed graph induced by the converged transition matrix $M_t$, where the edges are defined as:

$$E = \{(i, j) \mid M_t(i, j) > 0\}$$

A directed edge $(i, j)$ exists only if node $i$ can transition to node $j$ within $t$ steps of the expansion and inflation process.

A node $j$ is called an *attractor* if $M_t(j, j) > 0$, and we say that node $i$ is attracted to attractor $j$ if $M_t(i, j) > 0$. The MCL process yields a set of attractor nodes, $V_a \subseteq V$, such that other nodes are attracted to at least one attractor in $V_a$.

To extract the clusters from $G_t$, MCL first finds the strongly connected components $S_1, S_2, \ldots, S_q$ over the set of attractors $V_a$. Next, for each strongly connected set of attractors $S_j$, MCL finds the weakly connected components consisting of all nodes $i \in V_t - V_a$ attracted to an attractor in $S_j$. If a node $i$ is attracted to multiple strongly connected components, it is added to each such cluster, resulting in possibly overlapping clusters.
Algorithm **MARKOV CLUSTERING**

**MARKOV CLUSTERING** ($A$, $r$, $\epsilon$):

1. $t \leftarrow 0$
2. Add self-edges to $A$ if they do not exist
3. $M_t \leftarrow \Delta^{-1}A$
4. repeat
5. \[ t \leftarrow t + 1 \]
6. \[ M_t \leftarrow M_{t-1} \cdot M_{t-1} \]
7. \[ M_t \leftarrow \Upsilon(M_t, r) \]
8. until \[ \|M_t - M_{t-1}\|_F \leq \epsilon \]
9. $G_t \leftarrow$ directed graph induced by $M_t$
10. $C \leftarrow \{\text{weakly connected components in } G_t\}$
MCL Attractors and Clusters

\( r = 2.5 \)
MCL on Iris Graph

(a) $r = 1.3$

(b) $r = 2$
Contingency Table: MCL Clusters versus Iris Types

\( r = 1.3 \)

<table>
<thead>
<tr>
<th></th>
<th>iris-setosa</th>
<th>iris-virginica</th>
<th>iris-versicolor</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_1 ) (triangle)</td>
<td>50</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( C_2 ) (square)</td>
<td>0</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>( C_3 ) (circle)</td>
<td>0</td>
<td>14</td>
<td>49</td>
</tr>
</tbody>
</table>