2
Foundations of Graphs

2.1 Introduction

Graphs, which describe pairwise relations between entities, are essential representations for real-world data from many different domains, including social science, linguistics, chemistry, biology, and physics. Graphs are widely utilized in social science to indicate the relations between individuals. In chemistry, chemical compounds are denoted as graphs with atoms as nodes and chemical bonds as edges (Bonchev, 1991). In linguistics, graphs are utilized to capture the syntax and compositional structures of sentences. For example, parsing trees are leveraged to represent the syntactic structure of a sentence according to some context-free grammar, while Abstract Meaning Representation (AMR) encodes the meaning of a sentence as a rooted and directed graph (Banarescu et al., 2013). Hence, research on graphs has attracted immense attention from multiple disciplines. In this chapter, we first introduce basic concepts of graphs and discuss the matrix representations of graphs, including adjacency matrix and Laplacian matrix (Chung and Graham, 1997) and their fundamental properties. Then we introduce attributed graphs where each node is associated with attributes and provide a new understanding of these graphs by regarding the attributes as functions or signals on the graph (Shuman et al., 2013). We present the concepts of graph Fourier analysis and graph signal processing, which lay essential foundations for deep learning on graphs. Next, we describe various complex graphs that are frequently utilized to capture complicated relations among entities in real-world applications. Finally, we discuss representative computational tasks on graphs that have been broadly served as downstream tasks for deep learning on graphs.

2.2 Graph Representations

In this section, we introduce the definition of graphs. We focus on simple unweighted graphs and will discuss more complex graphs in the following sections.

Definition 2.1 (Graph) A graph can be denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a set of $N = |\mathcal{V}|$ nodes and $\mathcal{E} = \{e_1, \ldots, e_M\}$ is a set of M edges.

Nodes are essential entities in a graph. In social graphs, users are viewed as nodes, while in chemical compound graphs, chemical atoms are treated as nodes. The size of a given graph $\mathcal{G}$ is defined by its number of nodes, i.e., $N = |\mathcal{V}|$. The set of edges $\mathcal{E}$ describes the connections between nodes. An edge $e \in \mathcal{E}$ connects two nodes v_e^1 and v_e^2; thus, the edge e can be also represented as (v_e^1, v_e^2). In directed graphs, the edge is directed from node v_e^1 to node v_e^2. While in undirected graphs, the order of the two nodes does not make a difference, i.e., $e = (v_e^1, v_e^2) = (v_e^2, v_e^1)$. Note that without specific mention, we limit our discussion to undirected graphs in this chapter. The nodes v_e^1 and v_e^2 are incident to the edge e. A node v_i is adjacent to another node v_j if and only if there exists an edge between them. In social graphs, different relations such as friendship can be viewed as edges between nodes, and chemical bonds are considered as edges in chemical compound graphs (we regard all chemical bonds as edges while ignoring their different types). A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ can be equivalently represented as an adjacency matrix, which describes the connectivity between the nodes.

Definition 2.2 (Adjacency Matrix) For a given graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the corresponding adjacency matrix is denoted as $\mathbf{A} \in \{0, 1\}^{N \times N}$. The i, j-th entry of the adjacency matrix $\mathbf{A}$, indicated as $\mathbf{A}_{i,j}$, represents the connectivity between two nodes v_i and v_j. More specifically, $\mathbf{A}_{i,j} = 1$ if v_i is adjacent to v_j, otherwise $\mathbf{A}_{i,j} = 0$.

In an undirected graph, a node v_i is adjacent to v_j, if and only if v_j is adjacent to v_i, thus $\mathbf{A}_{i,j} = \mathbf{A}_{j,i}$ holds for all v_i and v_j in the graph. Hence, for an undirected graph, its corresponding adjacency matrix is symmetric.

Example 2.3 An illustrative graph with 5 nodes and 6 edges is shown in Figure 2.1. In this graph, the set of nodes is represented as $\mathcal{V} = \{v_1, v_2, v_3, v_4, v_5\}$, and the set of edges is $\mathcal{E} = \{e_1, e_2, e_3, e_4, e_5, e_6\}$. Its adjacency matrix can be denoted as follows:

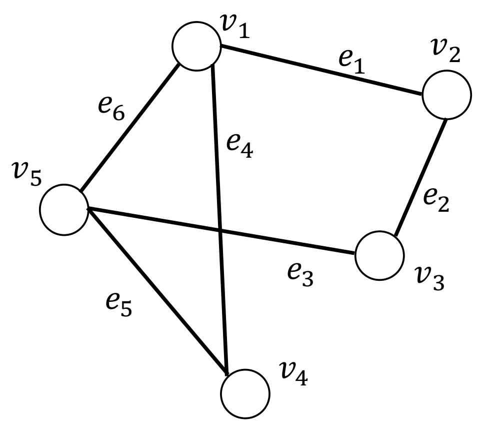


Figure 2.1 A graph with 5 nodes and 6 edges

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$

2.3 Properties and Measures

Graphs can have varied structures and properties. In this section, we discuss basic properties and measures for graphs.

2.3.1 Degree

The degree of a node v in a graph $\mathcal{G}$ indicates the number of times that a node is adjacent to other nodes. We have the following formal definition.

Definition 2.4 (Degree) In a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the degree of a node $v_i \in \mathcal{V}$ is the number of nodes that are adjacent to v_i.

$$d(v_i) = \sum_{v_j \in \mathcal{V}} \mathbb{1}_{\mathcal{E}}(\{v_i, v_j\}),$$

where $\mathbb{1}_{\mathcal{E}}(\cdot)$ is an indicator function:

$$\mathbb{1}_{\mathcal{E}}(\{v_i, v_j\}) = \begin{cases} 1 & \text{if } (v_i, v_j) \in \mathcal{E}, \\ 0 & \text{if } (v_i, v_j) \notin \mathcal{E}. \end{cases}$$

The degree of a node v_i in $\mathcal{G}$ can also be calculated from its adjacency matrix. More specifically, for a node v_i, its degree can be computed as:

$$d(v_i) = \sum_{j=1}^{N} \mathbf{A}_{i,j}. \tag{2.1}$$

Example 2.5 In the graph shown in Figure 2.1, the degree of node v_5 is 3, as it is adjacent to 3 other nodes (i.e., v_1, v_3 and v_4). Furthermore, the 5-th row of the adjacency matrix has 3 non-zero elements, which also indicates that the degree of v_5 is 3.

Definition 2.6 (Neighbors) For a node v_i in a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the set of its neighbors $\mathcal{N}(v_i)$ consists of all nodes that are adjacent to v_i.

Note that for a node v_i, the number of nodes in $\mathcal{N}(v_i)$ equals to its degree, i.e., $d(v_i) = |\mathcal{N}(v_i)|$.

Theorem 2.7 *For a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, its total degree, i.e., the summation of the degree of all nodes, is twice the number of edges in the graph.*

$$\sum_{v_i \in \mathcal{V}} d(v_i) = 2 \cdot |\mathcal{E}|.$$

Proof

$$\begin{aligned} \sum_{v_i \in \mathcal{V}} d(v_i) &= \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{V}} \mathbb{1}_{\mathcal{E}}(\{v_i, v_j\}) \\ &= \sum_{\{v_i, v_j\} \in \mathcal{E}} 2 \cdot \mathbb{1}_{\mathcal{E}}(\{v_i, v_j\}) \\ &= 2 \cdot \sum_{\{v_i, v_j\} \in \mathcal{E}} \mathbb{1}_{\mathcal{E}}(\{v_i, v_j\}) \\ &= 2 \cdot |\mathcal{E}| \end{aligned}$$

□

Corollary 2.8 *The number of non-zero elements in the adjacency matrix is also twice the number of the edges.*

Proof The proof follows Theorem 2.7 by using Eq. (2.1). □

Example 2.9 For the graph shown in Figure 2.1, the number of edges is 6. The total degree is 12 and the number of non-zero elements in its adjacent matrix is also 12.

2.3.2 Connectivity

Connectivity is an important property of graphs. Before discussing connectivity in graphs, we first introduce some basic concepts such as walk and path.

Definition 2.10 (Walk) A walk on a graph is an alternating sequence of nodes and edges, starting with a node and ending with a node where each edge is incident with the nodes immediately preceding and following it.

A walk starting at node u and ending at node v is called a u-v walk. The length of a walk is the number of edges in this walk. Note that u-v walks are not unique since there exist various u-v walks with different lengths.

Definition 2.11 (Trail) A trail is a walk whose edges are distinct.

Definition 2.12 (Path) A path is a walk whose nodes are distinct.

Example 2.13 In the graph shown in Figure 2.1, $(v_1, e_4, v_4, e_5, v_5, e_6, v_1, e_1, v_2)$ is a v_1-v_2 walk of length 4. It is a trail but not a path as it visits node v_1 twice. Meanwhile, $(v_1, e_1, v_2, e_2, v_3)$ is a v_1-v_3 walk. It is a trail as well as a path.

Theorem 2.14 *For a graph $\mathcal{G} = \{\mathcal{E}, \mathcal{V}\}$ with the adjacency matrix $\mathbf{A}$, we use $\mathbf{A}^n$ to denote the n-th power of the adjacency matrix. The i, j-th element of the matrix $\mathbf{A}^n$ equals to the number of v_i-v_j walks of length n.*

Proof We can prove this theorem by induction. For $n = 1$, according to the definition of the adjacency matrix, when $\mathbf{A}_{i,j} = 1$, there is an edge between nodes v_i and v_j, which is regarded as a v_i-v_j walk of length 1. When $\mathbf{A}_{i,j} = 0$, there is no edge between v_i and v_j, thus there is no v_i-v_j walk of length 1. Hence, the theorem holds for $n = 1$. Assume that the theorem holds when $n = k$. In other words, the i, h-th element of $\mathbf{A}^k$ equals to the number of v_i-v_h walks of length k. We then proceed to prove the case when $n = k + 1$. Specifically, the i, j-th element of $\mathbf{A}^{k+1}$ can be calculated by using $\mathbf{A}^k$ and $\mathbf{A}$ as

$$\mathbf{A}^{k+1}_{i,j} = \sum_{h=1}^{N} \mathbf{A}^k_{i,h} \cdot \mathbf{A}_{h,j}. \tag{2.2}$$

For each h in Eq. (2.2), the term $\mathbf{A}^k_{i,h} \cdot \mathbf{A}_{h,j}$ is non-zero only if both $\mathbf{A}^k_{i,h}$ and $\mathbf{A}_{h,j}$ are non-zero. We have already known that $\mathbf{A}^k_{i,h}$ denotes the number of v_i-v_h walks of length k while $\mathbf{A}_{h,j}$ indicates the number of the v_h-v_j walk of length 1.

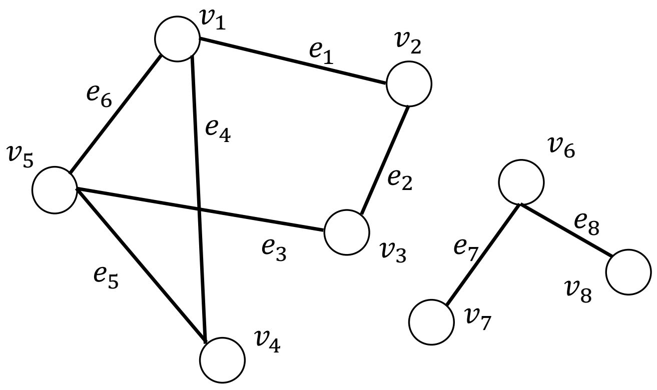


Figure 2.2 A graph with two connected components

Hence, the term $\mathbf{A}^k_{i,h} \cdot \mathbf{A}_{h,j}$ counts the number of v_i-v_j walks of length $k+1$ with v_h as the second last node in the walk. Thus, when summing over all possible nodes v_h, the i, j-th element of $\mathbf{A}^{k+1}$ equals to the number of v_i-v_j walks of length $k+1$, which completes the proof. □

Definition 2.15 (Subgraph) A subgraph $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}'\}$ of a given graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is a graph formed with a subset of nodes $\mathcal{V}' \subset \mathcal{V}$ and a subset of edges $\mathcal{E}' \subset \mathcal{E}$. Furthermore, the subset $\mathcal{V}'$ must contain all the nodes involved in the edges in the subset $\mathcal{E}'$.

Example 2.16 For the graph shown in Figure 2.1, the subset of nodes $\mathcal{V}' = \{v_1, v_2, v_3, v_5\}$ and the subset of edges $\mathcal{E}' = \{e_1, e_2, e_3, e_6\}$ form a subgraph of the original graph $\mathcal{G}$.

Definition 2.17 (Connected Component) Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, a subgraph $\mathcal{G}' = \{\mathcal{V}', \mathcal{E}'\}$ is said to be a connected component if there is at least one path between any pair of nodes in the graph and the nodes in $\mathcal{V}'$ are not adjacent to any vertices in $\mathcal{V}/\mathcal{V}'$.

Example 2.18 A graph with two connected components is shown in Figure 2.2, where the left and right connected components are not connected to each other.

Definition 2.19 (Connected Graph) A graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ is said to be connected if it has exactly one component.

Example 2.20 The graph shown in Figure 2.1 is a connected graph, while the graph in Figure 2.2 is not a connected graph.

Given a pair of nodes in a graph, there may exist multiple paths with different lengths between them. For example, there are 3 paths from node v_5 to node v_2 in the graph shown in Figure 2.1: $(v_5, e_6, v_1, e_1, v_2)$, $(v_5, e_5, v_4, e_4, v_1, e_1, v_2)$ and $(v_5, e_3, v_3, e_2, v_2)$. Among them, $(v_5, e_6, v_1, e_1, v_2)$ and $(v_5, e_3, v_3, e_2, v_2)$ with length 3 are the shortest paths from v_5 to v_2.

Definition 2.21 (Shortest Path) Given a pair of nodes $v_s, v_t \in \mathcal{V}$ in graph $\mathcal{G}$, we denote the set of paths from node v_s to node v_t as $\mathcal{P}_{st}$. The shortest path between node v_s and node v_t is defined as:

$$p_{\mathrm{st}}^{sp} = \arg\min_{p \in \mathcal{P}_{\mathrm{st}}} |p|,$$

where p denotes a path in $\mathcal{P}_{st}$ with $|p|$ its length and p_{st}^{sp} indicates the shortest path. Note that there could be more than one shortest path between any given pair of nodes.

The shortest path between a pair of nodes describes important information between them. A collective information of the shortest paths between any pairs of nodes in a graph indicates important characteristics of the graph. Specifically, the diameter of a graph is defined as the length of the longest shortest path in the graph.

Definition 2.22 (Diameter) Given a connected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, its diameter is defined as follows:

$$\mathrm{diameter}(\mathcal{G}) = \max_{v_s, v_t \in \mathcal{V}} \min_{p \in \mathcal{P}_{st}} |p|.$$

Example 2.23 For the connected graph shown in Figure 2.1, its diameter is 3. In detail, the longest shortest paths are between node v_2 and node v_4.

2.3.3 Centrality

In a graph, the centrality of a node measures the importance of the node in the graph. There are different ways to measure node importance. In this section, we introduce various definitions of centrality.

Degree Centrality

Intuitively, a node can be considered as important if there are many other nodes connected to it. Hence, we can measure the centrality of a given node based on its degree. In particular, for node v_i, its degree centrality can be defined as follows:

$$c_d(v_i) = d(v_i) = \sum_{j=1}^{N} \mathbf{A}_{i,j}.$$

Example 2.24 For the graph shown in Figure 2.1, the degree centrality for nodes v_1 and v_5 is 3, while the degree centrality for nodes v_2, v_3 and v_4 is 2.

Eigenvector Centrality

While the degree based centrality considers a node with many neighbors as important, it treats all the neighbors equally. However, the neighbors themselves can have different importance; thus they could affect the importance of the central node differently. The eigenvector centrality (Bonacich, 1972, 2007) defines the centrality score of a given node v_i by considering the centrality scores of its neighboring nodes as:

$$c_e(v_i) = \frac{1}{\lambda} \sum_{j=1}^{N} \mathbf{A}_{i,j} \cdot c_e(v_j),$$

which can be rewritten in a matrix form as:

$$\mathbf{c}_e = \frac{1}{\lambda} \mathbf{A} \cdot \mathbf{c}_e, \tag{2.3}$$

where $\mathbf{c}_e \in \mathbb{R}^N$ is a vector containing the centrality scores of all nodes in the graph. We can reform Eq. (2.3) as:

$$\lambda \cdot \mathbf{c}_e = \mathbf{A} \cdot \mathbf{c}_e.$$

Clearly, $\mathbf{c}_e$ is an eigenvector of the matrix $\mathbf{A}$ with its corresponding eigenvalue λ. However, given an adjacency matrix $\mathbf{A}$, there exist multiple pairs of eigenvectors and eigenvalues. Usually, we want the centrality scores to be positive. Hence, we wish to choose an eigenvector with all positive elements. According to PerronFrobenius theorem (Perron, 1907; Frobenius et al., 1912; Pillai et al., 2005), a real squared matrix with positive elements has a unique largest eigenvalue and its corresponding eigenvector has all positive elements. Thus, we can choose λ as the largest eigenvalue and its corresponding eigenvector as the centrality score vector.

Example 2.25 For the graph shown in Figure 2.1, its largest eigenvalue is 2.481 and its corresponding eigenvector is $[1, 0.675, 0.675, 0.806, 1]$. Hence, the eigenvector centrality scores for the nodes v_1, v_2, v_3, v_4, v_5 are $1, 0.675, 0.675,$ 0.806,and 1, respectively. Note that the degrees of nodes v_2, v_3 and v_4 are 2; however, the eigenvector centrality of node v_4 is higher than that of the other two nodes as it directly connects to nodes v_1 and v_5 whose eigenvector centrality is high.

Katz Centrality

The Katz centrality is a variant of the eigenvector centrality, which not only considers the centrality scores of the neighbors but also includes a small constant for the central node itself. Specifically, the Katz centrality for a node v_i can be defined as:

$$c_k(v_i) = \alpha \sum_{j=1}^{N} \mathbf{A}_{i,j} c_k(v_j) + \beta, \tag{2.4}$$

where β is a constant. The Katz centrality scores for all nodes can be expressed in the matrix form as:

$$\begin{aligned} \mathbf{c}_k &= \alpha \mathbf{A} \mathbf{c}_k + \boldsymbol{\beta} \\ (I - \alpha \cdot \mathbf{A})\mathbf{c}_k &= \boldsymbol{\beta} \end{aligned} \tag{2.5}$$

where $\mathbf{c}_k \in \mathbb{R}^N$ denotes the Katz centrality score vector for all nodes while $\boldsymbol{\beta}$ is the vector containing the constant term β for all nodes. Note that the Katz centrality is equivalent to the eigenvector centrality if we set $\alpha = \frac{1}{\lambda_{max}}$ and $\beta = 0$, with λ_{max} the largest eigenvalue of the adjacency matrix $\mathbf{A}$. The choice of α is important – a large α may make the matrix $I - \alpha \cdot \mathbf{A}$ ill-conditioned while a small α may make the centrality scores useless since it will assign very similar scores close to β to all nodes. In practice, $\alpha < \frac{1}{\lambda_{max}}$ is often selected, which ensures that the matrix $I - \alpha \cdot \mathbf{A}$ is invertible and $\mathbf{c}_k$ can be calculated as:

$$\mathbf{c}_k = (I - \alpha \cdot \mathbf{A})^{-1} \boldsymbol{\beta}.$$

Example 2.26 For the graph shown in Figure 2.1, if we set $\beta = 1$ and $\alpha = \frac{1}{5}$, the Katz centrality score for nodes v_1 and v_5 is 2.16, for nodes v_2 and v_3 is 1.79 and for node v_4 is 1.87.

Betweenness Centrality

The aforementioned centrality scores are based on connections to neighboring nodes. Another way to measure the importance of a node is to check whether it is at an important position in the graph. Specifically, if there are many paths passing through a node, it is at an important position in the graph. Formally, we define the betweenness centrality score for a node v_i as:

$$c_b(v_i) = \sum_{v_s \neq v_i \neq v_t} \frac{\sigma_{st}(v_i)}{\sigma_{st}}, \tag{2.6}$$

where σ_{st} denotes the total number of shortest paths from node v_s to node v_t while $\sigma_{st}(v_i)$ indicates the number of these paths passing through the node v_i. As suggested by Eq. (2.6), we need to compute the summation over all possible pairs of nodes for the betweenness centrality score. Therefore, the magnitude of the betweenness centrality score scales as the size of graph scales. Hence, to make the betweenness centrality score comparable across different graphs, we need to normalize it. One effective way is to divide the betweenness score by the largest possible betweenness centrality score given a graph. In Eq. (2.6), the maximum of the betweenness score can be reached when all the shortest paths between any pair of nodes passing through the node v_i. That is $\frac{\sigma_{st}(v_i)}{\sigma_{st}} = 1, \forall v_s \neq v_i \neq v_t$. There are, in total, $\frac{(N-1)(N-2)}{2}$ pairs of nodes in an undirected graph. Hence, the maximum betweenness centrality score is $\frac{(N-1)(N-2)}{2}$. We then define the normalized betweenness centrality score $c_{nb}(v_i)$ for the node v_i as:

$$c_{nb}(v_i) = \frac{2 \sum\limits_{v_s \neq v_i \neq v_t} \frac{\sigma_{st}(v_i)}{\sigma_{st}}}{(N-1)(N-2)},$$

Example 2.27 For the graph shown in Figure 2.1, the betweenness centrality score for nodes v_1 and v_5 is $\frac{3}{2}$, and their normalized betweenness score is $\frac{1}{4}$. The betweenness centrality score for nodes v_2 and v_3 is $\frac{1}{2}$, and their normalized betweenness score is $\frac{1}{12}$. The betweenness centrality score for node v_4 is 0 and its normalized score is also 0.

2.4 Spectral Graph Theory

Spectral graph theory studies the properties of a graph through analyzing the eigenvalues and eigenvectors of its Laplacian matrix. In this section, we introduce the Laplacian matrix of a graph and then discuss its key properties, eigenvalues, and eigenvectors.

2.4.1 Laplacian Matrix

In this subsection, we introduce the Laplacian matrix of a graph, which is another matrix representation for graphs in addition to the adjacency matrix.

Definition 2.28 (Laplacian Matrix) For a given graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathbf{A}$ as its adjacency matrix, its Laplacian matrix is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2.7}$$

where $\mathbf{D}$ is a diagonal degree matrix $\mathbf{D} = diag(d(v_1), \ldots, d(v_N))$.

Another definition of the Laplacian matrix is a normalized version of Eq. (2.7).

Definition 2.29 (Normalized Laplacian Matrix) For a given graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $\mathbf{A}$ as its adjacency matrix, its normalized Laplacian matrix is defined as:

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{D} - \mathbf{A})\mathbf{D}^{-\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}. \tag{2.8}$$

Next, we focus on the discussion of the unnormalized Laplacian matrix as defined in Definition 2.28. However, in some later chapters of this book, the normalized Laplacian matrix will also be utilized. Unless specific mention, we refer Laplacian matrix as the unnormalized one defined in Definition 2.28.

Note that the Laplacian matrix is symmetric as both the degree matrix $\mathbf{D}$ and the adjacency matrix $\mathbf{A}$ are symmetric. Let $\mathbf{f}$ denote a vector where its i-th element $\mathbf{f}[i]$ is associated with node v_i. Multiplying $\mathbf{L}$ with $\mathbf{f}$, we can get a new vector $\mathbf{h}$ as:

$$\begin{aligned}\mathbf{h} &= \mathbf{L}\mathbf{f}\\ &= (\mathbf{D} - \mathbf{A})\mathbf{f}\\ &= \mathbf{D}\mathbf{f} - \mathbf{A}\mathbf{f}.\end{aligned}$$

The i-th element of $\mathbf{h}$ can be represented as:

$$\begin{aligned}\mathbf{h}[i] &= d(v_i) \cdot \mathbf{f}[i] - \sum_{j=1}^{N} \mathbf{A}_{i,j} \cdot \mathbf{f}[i]\\ &= d(v_i) \cdot \mathbf{f}[i] - \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{A}_{i,j} \cdot \mathbf{f}[i]\\ &= \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}[i] - \mathbf{f}[j]).\end{aligned} \tag{2.9}$$

As informed by Eq. (2.9), $\mathbf{h}[i]$ is the summation of the differences between node v_i and its neighbors $\mathcal{N}(v_i)$. We next calculate $\mathbf{f}^T\mathbf{L}\mathbf{f}$ as below:

$$\begin{aligned}\mathbf{f}^T\mathbf{L}\mathbf{f} &= \sum_{v_i \in \mathcal{V}} \mathbf{f}[i] \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}[i] - \mathbf{f}[j])\\ &= \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}[i] \cdot \mathbf{f}[i] - \mathbf{f}[i] \cdot \mathbf{f}[j])\\ &= \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} (\frac{1}{2}\mathbf{f}[i] \cdot \mathbf{f}[i] - \mathbf{f}[i] \cdot \mathbf{f}[j] + \frac{1}{2}\mathbf{f}[j] \cdot \mathbf{f}[j])\\ &= \frac{1}{2} \sum_{v_i \in \mathcal{V}} \sum_{v_j \in \mathcal{N}(v_i)} (\mathbf{f}[i] - \mathbf{f}[j])^2.\end{aligned} \tag{2.10}$$

Thus, $\mathbf{f}^T\mathbf{L}\mathbf{f}$ is the sum of the squares of the differences between adjacent nodes.

In other words, it measures how different the values of adjacent nodes are. It is easy to verify that $\mathbf{f}^T\mathbf{L}\mathbf{f}$ is always non-negative for any possible choice of a real vector $\mathbf{f}$, which indicates that the Laplacian matrix is positive semi-definite.

2.4.2 The Eigenvalues and Eigenvectors of the Laplacian Matrix

In this subsection, we discuss main properties of eigenvalues and eigenvectors of the Laplacian matrix.

Theorem 2.30 *For a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, the eigenvalues of its Laplacian matrix $\mathbf{L}$ are non-negative.*

Proof Suppose that λ is an eigenvalue of the Laplacian matrix $\mathbf{L}$ and $\mathbf{u}$ is the corresponding normalized eigenvector. According to the definition of eigenvalues and eigenvectors, we have $\lambda\mathbf{u} = \mathbf{L}\mathbf{u}$. Note that $\mathbf{u}$ is a unit non-zero vector and we have $\mathbf{u}^T\mathbf{u} = 1$. Then,

$$\lambda = \lambda\mathbf{u}^T\mathbf{u} = \mathbf{u}^T\lambda\mathbf{u} = \mathbf{u}^T\mathbf{L}\mathbf{u} \geq 0$$

□

For a graph $\mathcal{G}$ with N nodes, there are, in total, N eigenvalues/eigenvectors (with multiplicity). According to Theorem 2.30, all the eigenvalues are non-negative. Furthermore, there always exists an eigenvalue that equals to 0. Let us consider the vector $\mathbf{u}_1 = \frac{1}{\sqrt{N}}(1, \ldots, 1)$. Using Eq. (2.9), we can easily verify that $\mathbf{L}\mathbf{u}_1 = \mathbf{0} = 0\mathbf{u}_1$, which indicates that $\mathbf{u}_1$ is an eigenvector corresponding to the eigenvalue 0. For convenience, we arrange these eigenvalues in non-decreasing order as $0 = \lambda_1 \leq \lambda_2 \leq, \ldots, \leq \lambda_N$. The corresponding normalized eigenvectors are denoted as $\mathbf{u}_1, \ldots, \mathbf{u}_N$.

Theorem 2.31 *Given a graph $\mathcal{G}$, the number of 0 eigenvalues of its Laplacian matrix $\mathbf{L}$ (the multiplicity of the 0 eigenvalue) equals to the number of connected components in the graph.*

Proof Suppose that there are K connected components in the graph $\mathcal{G}$. We can partition the set of nodes $\mathcal{V}$ into K disjoint subsets $\mathcal{V}_1, \ldots, \mathcal{V}_K$. We first show that there exist at least K orthogonal eigenvectors corresponding to the eigenvalue 0. Construct K vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$ such that $\mathbf{u}_i[j] = \frac{1}{\sqrt{|\mathcal{V}_i|}}$ if $v_j \in \mathcal{V}_i$ and 0 otherwise. We have that $\mathbf{L}\mathbf{u}_i = 0$ for $i = 1, \ldots, K$, which indicates that all the K vectors are the eigenvectors of $\mathbf{L}$ corresponding to eigenvalue 0. Furthermore, it is easy to validate that $\mathbf{u}_i^T\mathbf{u}_j = 0$ if $i \neq j$, which means that these K eigenvectors are orthogonal to each other. Hence, the multiplicity of the 0 eigenvalue is at least K. We next show that there are at most K orthogonal eigenvectors corresponding to the eigenvalue 0. Assume that there exists another eigenvector $\mathbf{u}^*$ corresponding to the eigenvalue 0, which is orthogonal to all the K aforementioned eigenvectors. As $\mathbf{u}^*$ is non-zero, there must exist an element in $\mathbf{u}^*$ that is non-zero. Let us assume that the element is $\mathbf{u}^*[d]$ associated with node $v_d \in \mathcal{V}_i$. Furthermore, according to Eq.(2.10), we have

$$\mathbf{u}^{*T}\mathbf{L}\mathbf{u}^* = \frac{1}{2}\sum_{v_i\in\mathcal{V}}\sum_{v_j\in\mathcal{N}(v_i)}(\mathbf{u}^*[i]-\mathbf{u}^*[j])^2.$$

To ensure $\mathbf{u}^{*T}\mathbf{L}\mathbf{u}^* = 0$, the values of nodes in the same component must be the same. It indicates that all nodes in $\mathcal{V}_i$ have the same value $\mathbf{u}^*[d]$ as node v_d. Hence, $\mathbf{u}_i^T\mathbf{u}^* > 0$. It means $\mathbf{u}^*$ is not orthogonal to $\mathbf{u}_i$, which leads to a contradiction. Therefore, there is no more eigenvector corresponding to the eigenvalue 0 beyond the K vectors we have constructed. □

2.5 Graph Signal Processing

In many real-world graphs, there are often features or attributes associated with nodes. This kind of graph-structured data can be treated as graph signals, which captures both the structure information (or connectivity between nodes) and data (or attributes at nodes). A graph signal consists of a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, and a mapping function f defined in the graph domain, which maps the nodes to real values. Mathematically, the mapping function can be represented as:

$$f : \mathcal{V} \to \mathbb{R}^{1\times d},$$

where d is the dimension of the value (vector) associated with each node. Without loss of generality, in this section, we set $d = 1$ and denote the mapped values for all nodes as $\mathbf{f} \in \mathbb{R}^N$ with $\mathbf{f}[i]$ corresponding to node v_i.

Example 2.32 A graph signal is shown in Figure 2.3, where the color of a node represents its associated value with smaller values tending toward blue and larger values tending toward red.

A graph is smooth if the values in connected nodes are similar. A smooth graph signal is with low frequency, as the values change slowly across the graph via the edges. The Laplacian matrix quadratic form in Eq. (2.10) can be utilized to measure the smoothness (or the frequency) of a graph signal $\mathbf{f}$ as it is the summation of the square of the difference between all pairs of connected nodes. Specifically, when a graph signal $\mathbf{f}$ is smooth, $\mathbf{f}^T\mathbf{L}\mathbf{f}$ is small. The value $\mathbf{f}^T\mathbf{L}\mathbf{f}$ is called as the smoothness (or the frequency) of the signal $\mathbf{f}$.

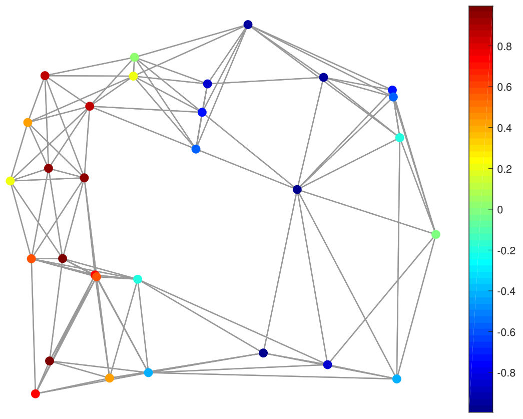


Figure 2.3 A one-dimensional graph signal

In the classical signal processing setting, a signal can be denoted in two domains, i.e., the time domain and the frequency domain. Similarly, the graph signal can also be represented in two domains, i.e., the spatial domain, which we just introduced, and the spectral domain (or frequency domain). The spectral domain of graph signals is based on the Graph Fourier Transform. It is built upon the spectral graph theory that we have introduced in the previous section.

2.5.1 Graph Fourier Transform

The classical Fourier Transform (Bracewell, n.d.)

$$\hat{f}(\xi) = < f(t), \exp(-2\pi i t \xi) >= \int_{-\infty}^{\infty} f(t) \exp(-2\pi i t \xi) dt$$

decomposes a signal $f(t)$ into a series of complex exponentials $\exp(-2\pi i t \xi)$ for any real number ξ, where ξ can be viewed as the frequency of the corresponding exponential. These exponentials are the eigenfunctions of the one-dimensional Laplace operator (or the second order differential operator) as we have

$$\begin{aligned}\nabla(\exp(-2\pi i t\xi)) &= \frac{\partial^2}{\partial t^2}\exp(-2\pi i t\xi)\\ &= \frac{\partial}{\partial t}(-2\pi i\xi)\exp(-2\pi i t\xi)\\ &= -(2\pi i\xi)^2\exp(-2\pi i t\xi).\end{aligned}$$

Analogously, the Graph Fourier Transform for a graph signal $\mathbf{f}$ on graph $\mathcal{G}$ can be represented as:

$$\hat{\mathbf{f}}[l] =< \mathbf{f}, \mathbf{u}_l >= \sum_{i=1}^{N} \mathbf{f}[i]\mathbf{u}_l[i], \tag{2.11}$$

where $\mathbf{u}_l$ is the l-th eigenvector of the Laplacian matrix $\mathbf{L}$ of the graph. The corresponding eigenvalue λ_l represents the frequency or the smoothness of the eigenvector $\mathbf{u}_l$. The vector $\hat{\mathbf{f}}$ with $\hat{\mathbf{f}}[l]$ as its l-th element is the Graph Fourier Transform of $\mathbf{f}$. The eigenvectors are the graph Fourier basis of the graph $\mathcal{G}$, and $\hat{\mathbf{f}}$ consists of the graph Fourier coefficients corresponding to these basis for a signal $\mathbf{f}$. The Graph Fourier Transform of $\mathbf{f}$ can be also denoted in the matrix form as:

$$\hat{\mathbf{f}} = \mathbf{U}^\top\mathbf{f} \tag{2.12}$$

where the l-th column of the matrix $\mathbf{U}$ is $\mathbf{u}_l$.

As suggested by the following equation:

$$\mathbf{u}_l^\top\mathbf{L}\mathbf{u}_l = \lambda_l \cdot \mathbf{u}_l^\top\mathbf{u}_l = \lambda_l,$$

the eigenvalue λ_l measures the smoothness of the corresponding eigenvector $\mathbf{u}_l$. Specifically, the eigenvectors associated with small eigenvalues vary slowly across the graph. In other words, the values of such eigenvector at connected nodes are similar. Thus, these eigenvectors are smooth and change with low frequency across the graph. However, the eigenvectors corresponding to large eigenvalues may have very different values on two nodes, even if connected. An extreme example is the first eigenvector $\mathbf{u}_1$ associated with the eigenvalue 0 – it is constant over all the nodes, which indicates that its value does not change across the graph. Hence, it is extremely smooth and has an extremely low frequency 0. These eigenvectors are the graph Fourier basis for the graph $\mathcal{G}$, and their corresponding eigenvalues indicate their frequencies. The Graph Fourier Transform, as shown in Eq. (2.12) can be regarded as a process to decompose an input signal $\mathbf{f}$ into graph Fourier basis with different frequencies. The obtained coefficients $\hat{\mathbf{f}}$ denote how much the corresponding graph Fourier basis contributes to the input signal.

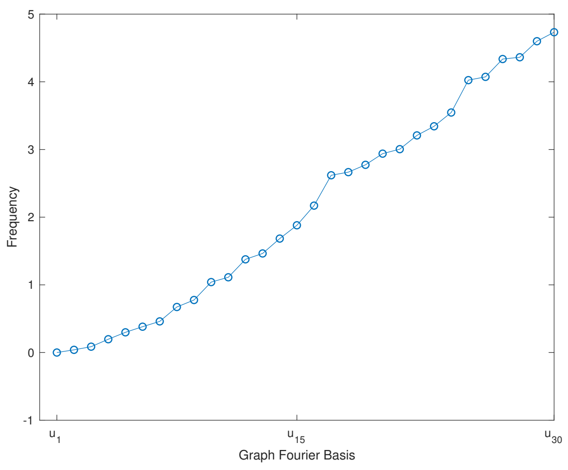


Figure 2.4 Frequencies of graph Fourier basis

Example 2.33 Figure 2.4 shows the frequencies of the graph Fourier basis of the graph shown in Figure 2.3. Note that the frequency of $\mathbf{u}_1$ is 0.

The graph Fourier coefficients $\hat{\mathbf{f}}$ are the representation of the signal $\mathbf{f}$ in the spectral domain. There is also the Inverse Graph Fourier Transform, which can transform the spectral representation $\hat{\mathbf{f}}$ to the spatial representation $\mathbf{f}$ as:

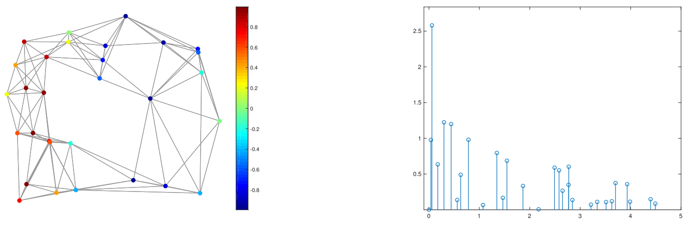
$$\mathbf{f}[i] = \sum_{l=1}^{N} \hat{f}[l]\mathbf{u}_l[i].$$

This process can also be represented in the matrix form as follows:

$$\mathbf{f} = \mathbf{U}\hat{\mathbf{f}}.$$

In summary, a graph signal can be denoted in two domains, i.e., the spatial domain and the spectral domain. The representations in the two domains can be transformed to each other via the Graph Fourier Transform and the Inverse Graph Fourier Transform, respectively.

Example 2.34 Figure 2.5 shows a graph signal in both the spatial and spectral domains. Specifically, Figure 2.5a shows the graph signal in the spatial domain and Figure 2.5b illustrates the same graph signal in the spectral domain. In Figure 2.5b, the x-axis is the graph Fourier basis and the y-axis indicates the corresponding graph Fourier coefficients.



(a) A graph signal in the spatial domain (b) A graph signal in the spectral domain

Figure 2.5 Representations of a graph signal in both spatial and spectral Domains

2.6 Complex Graphs

In the earlier sections, we introduced simple graphs and their fundamental properties. However, graphs in real-world applications are much more complicated. In this section, we briefly describe popular complex graphs with formal definitions.

2.6.1 Heterogeneous Graphs

The simple graphs we have discussed are homogeneous. They only have one type of nodes as well as a single type of edges. However, in many real-world applications, we want to model multiple types of relations between multiple types of nodes. As shown in Figure 2.6, in an academic network describing publications and citations, there are three types of nodes, including authors, papers, and venues. There are also various kinds of edges denoting different relations between the nodes. For example, there exist edges describing the citation relations between papers or edges denoting the authorship relations between authors and papers. Next, we formally define heterogeneous graphs.

Definition 2.35 (Heterogeneous Graphs) A heterogeneous graph $\mathcal{G}$ consists of a set of nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$ and a set of edges $\mathcal{E} = \{e_1, \ldots, e_M\}$ where each node and each edge are associated with a type. Let $\mathcal{T}_n$ denote the set of node types and $\mathcal{T}_e$ indicate the set of edge types. There are two mapping functions $\phi_n : \mathcal{V} \to \mathcal{T}_n$ and $\phi_e : \mathcal{V} \to \mathcal{T}_e$ that map each node and each edge to their types, respectively.

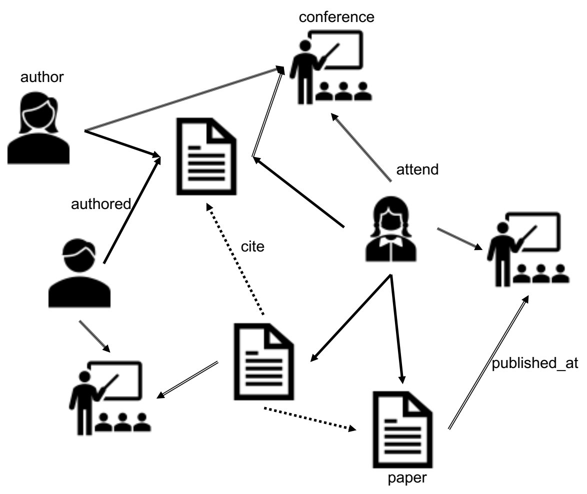


Figure 2.6 A heterogeneous academic graph

2.6.2 Bipartite Graphs

In a bipartite graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, its node set $\mathcal{V}$ can be divided into two disjoint subsets $\mathcal{V}_1$ and $\mathcal{V}_2$ where every edge in $\mathcal{E}$ connects a node in $\mathcal{V}_1$ to a node in $\mathcal{V}_2$. Bipartite graphs are widely used to capture interactions between different objects. For example, as shown in Figure 2.7, in many e-commerce platforms such as Amazon, the click history of users can be modeled as a bipartite graph where the users and items are the two disjoint node sets, and users' click behaviors form the edges between them. Next, we formally define bipartite graphs.

Definition 2.36 (Bipartite Graph) Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, it is bipartite if and only if $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$, $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ and $v_e^1 \in \mathcal{V}_1$ while $v_e^2 \in \mathcal{V}_2$ for all $e = (v_e^1, v_e^2) \in \mathcal{E}$.

2.6.3 Multi-dimensional Graphs

In many real-world graphs, multiple relations can simultaneously exist between a pair of nodes. One example of such graph can be found at the popular video-sharing site YouTube, where users are viewed as nodes. YouTube users can subscribe to each other, which is considered as one relation. Users can be connected via other relations such as "sharing" or "commenting" videos from other users. Another example is from e-commerce sites such as Amazon,

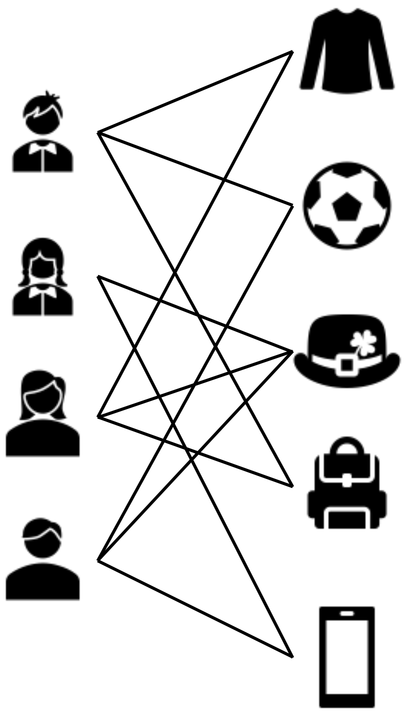


Figure 2.7 An e-commerce bipartite graph

where users can interact with items through various types of behaviors such as "click", "purchase" and "comment". These graphs with multiple relations can be naturally modeled as multi-dimensional graphs by considering each type of relations as one dimension.

Definition 2.37 (Multi-dimensional graph) A multi-dimensional graph consists of a set of N nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$ and D sets of edges $\{\mathcal{E}_1, \ldots, \mathcal{E}_D\}$. Each edge set $\mathcal{E}_d$ describes the d-th type of relations between the nodes in the corresponding d-th dimension. These D types of relations can also be expressed by D adjacency matrices $\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(D)}$. In the dimension d, its corresponding adjacency matrix $\mathbf{A}^{(d)} \in \mathbb{R}^{N \times N}$ describes the edges $\mathcal{E}_d$ between nodes in $\mathcal{V}$. Specifically, the i, j-th element of $\mathbf{A}_d$, denoted as $\mathbf{A}^{(d)}_{i,j}$, equals to 1 only when there is an edge between nodes v_i and v_j in the dimension d (or $(v_i, v_j) \in \mathcal{E}_d$), otherwise 0.

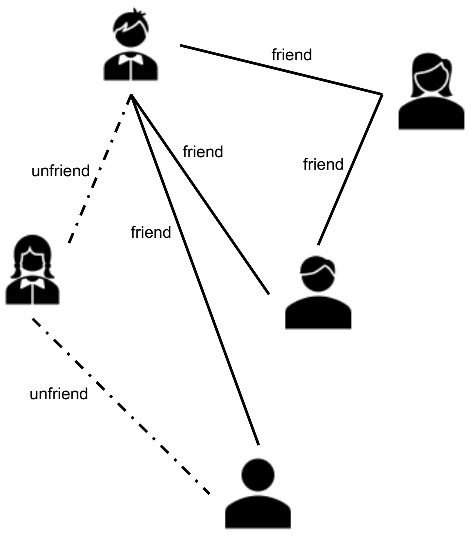


Figure 2.8 An illustrative signed graph

2.6.4 Signed Graphs

Signed graphs, which contain both positive and negative edges, have become increasingly ubiquitous with the growing popularity of the online social networks. Examples of signed graphs are from online social networks such as Facebook and Twitter, where users can block or unfollow other users. The behaviour of "block" can be viewed as negative edges between users. Meanwhile, the behaviour of "unfriend" can also be treated as negative edges. An illustrative example of a signed graph is shown in Figure 2.8, where users are nodes and "unfriend" and "friend" relations are the "negative" and "positive" edges, respectively. Next, we give the formal definition of signed graphs.

Definition 2.38 (Signed Graphs) Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}^+, \mathcal{E}^-\}$ be a signed graph, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of N nodes while $\mathcal{E}^+ \subset \mathcal{V} \times \mathcal{V}$ and $\mathcal{E}^- \subset \mathcal{V} \times \mathcal{V}$ denote the sets of positive and negative edges, respectively. Note that an edge can only be either positive or negative, i.e., $\mathcal{E}^+ \cap \mathcal{E}^- = \emptyset$. These positive and negative edges between nodes can also be described by a signed adjacency matrix $\mathbf{A}$, where $\mathbf{A}_{i,j} = 1$ only when there is a positive edge between node v_i and node v_j, $\mathbf{A}_{i,j} = -1$ denotes a negative edge, otherwise $\mathbf{A}_{i,j} = 0$.

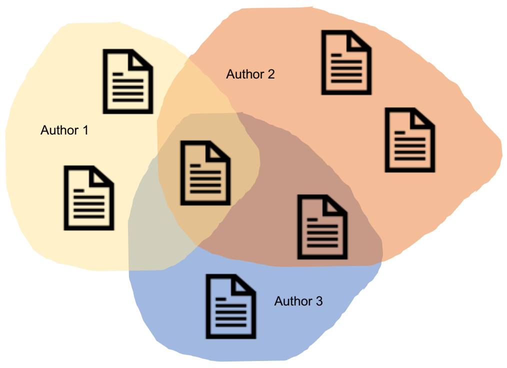


Figure 2.9 An illustrative hypergraph

2.6.5 Hypergraphs

The graphs we introduced so far only encode pairwise information via edges. However, in many real-world applications, relations are beyond pairwise associations. Figure 2.9 demonstrates a hypergraph describing the relations between papers. An specific author can publish more than two papers. Thus, the author can be viewed as a hyperedge connecting multiple papers (or nodes). Compared with edges in simple graphs, hyperedges can encode higher-order relations. The graphs with hyperedges are called as hypergraphs. Next, we give the formal definition of hypergraphs.

Definition 2.39 (Hypergraphs) Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ be a hypergraph, where $\mathcal{V}$ is a set of N nodes, $\mathcal{E}$ is a set of hyperedges and $\mathbf{W} \in \mathbb{R}^{|\mathcal{E}|\times|\mathcal{E}|}$ is a diagonal matrix with $\mathbf{W}[j, j]$ denoting the weight of the hyperedge e_j. The hypergraph $\mathcal{G}$ can be described by an incidence matrix $\mathbf{H} \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{E}|}$, where $\mathbf{H}_{i,j} = 1$ only when the node v_i is incident to the edge e_j. For a node v_i, its degree is defined as $d(v_i) = \sum_{j=1}^{|\mathcal{E}|} \mathbf{H}_{i,j}$, while the degree for a hyperedge is defined as $d(e_j) = \sum_{i=1}^{|\mathcal{V}|} \mathbf{H}_{i,j}$. Furthermore, we use $\mathbf{D}_e$ and $\mathbf{D}_v$ to denote the diagonal matrices of the edge and node degrees, respectively.

2.6.6 Dynamic Graphs

The graphs mentioned above are static, where the connections between nodes are fixed when observed. However, in many real-world applications, graphs are constantly evolving as new nodes are added to the graph, and new edges are

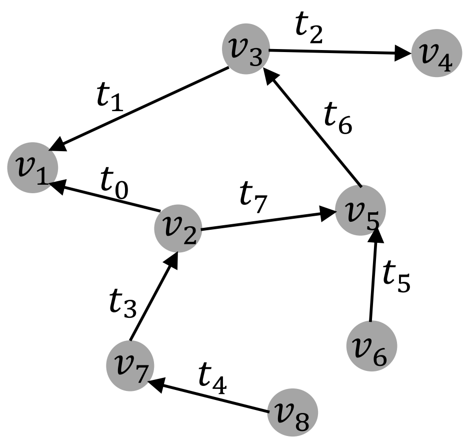


Figure 2.10 An illustrative example of dynamic graphs.

continuously emerging. For example, in online social networks such as Facebook, users can constantly establish friendships with others, and new users can also join Facebook at any time. These kinds of evolving graphs can be denoted as dynamic graphs where each node or edge is associated with a timestamp. An illustrative example of dynamic graphs is shown in Figure 2.10, where each edge is associated with a timestamp, and the timestamp of a node is when the very first edge involves the node. Next, we give a formal definition of dynamic graphs.

Definition 2.40 (Dynamic Graphs) A dynamic graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, consists of a set of nodes $\mathcal{V} = \{v_1, \ldots, v_N\}$ and a set of edges $\mathcal{E} = \{e_1, \ldots, e_M\}$ where each node and/or each edge is associated with a timestamp indicating the time it emerged. Specifically, we have two mapping functions ϕ_v and ϕ_e mapping each node and each edge to their emerging timestamps.

In reality, we may not be able to record the timestamp of each node and/or each edge. Instead, we only check from time to time to observe how the graph evolves. At each observation timestamp t, we can record the snapshot of the graph $\mathcal{G}_t$ as the observation. We refer to this kind of dynamic graphs as discrete dynamic graphs, which consist of multiple graph snapshots. We formally define the discrete dynamic graph as follows.

Definition 2.41 (Discrete Dynamic Graphs) A discrete dynamic graph consists of T graph snapshots, which are observed along with the evolution of a dynamic graph. Specifically, the T graph snapshots can be denoted as $\{\mathcal{G}_0, \ldots, \mathcal{G}_T\}$ where $\mathcal{G}_0$ is the graph observed at time 0.

2.7 Computational Tasks on Graphs

There are a variety of computational tasks proposed for graphs. These tasks can be mainly divided into two categories. One is node-focused tasks, where the entire data is usually represented as one graph with nodes as the data samples. The other is graph-focused tasks, where data often consists of a set of graphs, and each data sample is a graph. In this section, we briefly introduce representative tasks for each group.

2.7.1 Node-focused Tasks

Numerous node-focused tasks have been extensively studied, such as node classification, node ranking, link prediction, and community detection. Next, we discuss two typical tasks, including node classification and link prediction.

Node classification

In many real-world graphs, nodes are associated with useful information, often treated as labels of these nodes. For example, in social networks, such information can be demographic properties of users such as age, gender, and occupation, or users' interests and hobbies. These labels usually help characterize the nodes and can be leveraged for many important applications. For example, in social media such as Facebook, labels related to interests and hobbies can be utilized to recommend relevant items (i.e., news and events) to their users. However, in reality, it is often difficult to get a full set of labels for all nodes. For example, less than 1% of Facebook users provide their complete demographic properties. Hence, we are likely given a graph with only a part of the nodes associated with labels, and we aim to infer the labels for nodes without labels. It motivates the problem of node classification on graphs.

Definition 2.42 (Node classification) Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a graph with $\mathcal{V}$ the set of nodes and $\mathcal{E}$ the set of edges. Some of the nodes in $\mathcal{V}$ are associated with labels, and the set of these labeled nodes is represented as $\mathcal{V}_l \subset \mathcal{V}$. The remaining nodes do not have labels, and this set of unlabeled nodes is denoted as $\mathcal{V}_u$. Specifically, we have $\mathcal{V}_l \cup \mathcal{V}_u = \mathcal{V}$ and $\mathcal{V}_l \cap \mathcal{V}_u = \emptyset$. The goal of the node classification task is to learn a mapping ϕ by leveraging $\mathcal{G}$ and labels of $\mathcal{V}_l$, which can predict the labels of unlabeled nodes (or $v \in \mathcal{V}_u$).

The above definition is for simple graphs that can be easily extended to graphs with attributes and complex graphs we introduced in Section 2.6.

Example 2.43 (Node Classification in Flickr) Flickr is an image hosting platform that allows users to host their photos. It also serves as an online social community where users can follow each other. Hence, users in Flickr and their connections form a graph. Furthermore, users in Flickr can subscribe to interest groups such as "*Black and White*", "*The Fog and The Rain*", and "*Dog World*". These subscriptions indicate the interests of users and can be used as their labels. A user can subscribe to multiple groups. Hence, each user can be associated with multiple labels. A multi-label node classification problem on graphs can help predict the potential groups that users are interested in, but they have not yet subscribed to. One such dataset on Flickr can be found in (Tang and Liu, 2009).

Link Prediction

In many real-world applications, graphs are not complete but with missing edges. Some of the connections exist. However, they are not observed or recorded, which leads to missing edges in the observed graphs. Meanwhile, many graphs are naturally evolving. In social media such as Facebook, users can continuously become friends with other users. In academic collaboration graphs, a given author can constantly build new collaboration relations with other authors. Inferring or predicting these missing edges can benefit many applications such as friend recommendation (Adamic and Adar, 2003), knowledge graph completion (Nickel et al., 2015), and criminal intelligence analysis (Berlusconi et al., 2016). Next, we give the formal definition of the link prediction problem.

Definition 2.44 (Link Prediction) Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote a graph with $\mathcal{V}$ as its set of nodes and $\mathcal{E}$ as its set of edges. Let $\mathcal{M}$ denote all possible edges between nodes in $\mathcal{V}$. Then, we denote the complementary set of $\mathcal{E}$ with respect to $\mathcal{M}$ as $\mathcal{E}' = \mathcal{M}/\mathcal{E}$. The set $\mathcal{E}'$ contains the unobserved edges between the nodes. The goal of the link prediction task is to predict the edges that most likely exist. Specifically, a score can be assigned to each of the edges in $\mathcal{E}'$. It indicates how likely the edge exists or will emerge in the future.

Note that the definition is stated for simple graphs and can be easily extended to complex graphs we introduced in Section 2.6. For example, for signed graphs, in addition to the existence of an edge, we also want to predict its sign. For hypergraphs, we want to infer hyperedges which describe the relations between multiple nodes.

Example 2.45 (Predicting Emerging Collaborations in DBLP) DBLP is an online computer science bibliography website that hosts a comprehensive list of research papers in computer science. A co-authorship graph can be constructed from the papers in DBLP, where the authors are the nodes, and authors can be considered as connected if they have co-authored at least one paper as recorded in DBLP. Predicting what new collaborations between authors who never co-authored before is an interesting link prediction problem. A large DBLP collaboration dataset for link prediction research can be found in (Yang and Leskovec, 2015).

2.7.2 Graph-focused Tasks

There are numerous graph-focused tasks, such as graph classification, graph matching, and graph generation. Next, we discuss the most representative graph-focused task, i.e., graph classification.

Graph Classification

Node classification treats each node in a graph as a data sample and aims to assign labels to these unlabeled nodes. In some applications, each sample can be represented as a graph. For example, in chemoinformatics, chemical molecules can be denoted as graphs where atoms are the nodes, and chemical bonds between them are the edges. These chemical molecules may have different properties such as solubility and toxicity, which can be treated as their labels. In reality, we may want to predict these properties for newly discovered chemical molecules automatically. This goal can be achieved by the task of graph classification, which aims to predict the labels for unlabeled graphs. Graph classification cannot be carried out by traditional classification due to the complexity of graph structures. Thus, dedicated efforts are desired. Next, we provide a formal definition of graph classification.

Definition 2.46 (Graph Classification) Given a set of labeled graphs $\mathcal{D} = \{(\mathcal{G}_i, y_i)\}$ with y_i as the label of the graph $\mathcal{G}_i$, the goal of the graph classification task is to learn a mapping function ϕ with $\mathcal{D}$, which can predict the labels of unlabeled graphs.

In the definition above, we did not specify additional information potentially associated with the graphs. For example, in some scenarios, each node in a graph is associated with certain features that can be utilized for graph classification.

Example 2.47 (Classifying Proteins into Enzymes or Non-enzymes) Proteins can be represented as graphs, where amino acids are the nodes, and edges between two nodes are formed if they are less than $6\mathring{A}$ apart. Enzymes are a type of proteins which serve as biological catalysts to catalyze biochemical reactions. Given a protein, predicting whether it is an enzyme or not can be treated as a graph classification task where the label for each protein is either enzyme or non-enzyme.

2.8 Conclusion

In this chapter, we briefly introduced the concepts of graphs, the matrix representations of graphs, and the important measures and properties of graphs, including degree, connectivity, and centrality. We then discuss Graph Fourier Transform and graph signal processing, which lay the foundations for spectral based graph neural networks. We introduce a variety of complex graphs. Finally, we discuss representative computational tasks on graphs, including both node-focused and graph-focused tasks.

2.9 Further Reading

We briefly introduce many basic concepts in graphs. There are also other more advanced properties and concepts in graphs such as flow and cut. Furthermore, there are many problems defined on graphs, including graph coloring problem, route problem, network flow problem, and covering problem. These concepts and topics are covered by the books (Bondy *et al.*, n.d.; Newman, 2018). Meanwhile, more spectral properties and theories on graphs can be found in the book (Chung and Graham, 1997). Applications of graphs in different areas can be found in (Borgatti et al., 2009; Nastase et al., 2015; Trinajstic, 2018). The Stanford Large Network Dataset Collection (Leskovec and Krevl, 2014) and the Network Data Repository (Rossi and Ahmed, 2015), host large amounts of graph datasets from various areas. The python libraries *networkx* (Hagberg et al., 2008), *graph-tool* (Peixoto, 2014), and SNAP (Leskovec and Sosič, 2016) can be used to analyze and visualize graph data. The *Graph Signal Processing Toolbox* (Perraudin et al., 2014) can be employed to perform graph signal processing.