12

Graph Neural Networks in Data Mining

12.1 Introduction

Data mining aims to extract patterns and knowledge from large amounts of data (Han et al., 2011). Data from many real-world applications can be inherently represented as graphs. In the Web, relations among social media users such as friendships in Facebook and following relations in Twitter can be denoted as social graphs, and the historical interactions between e-commerce users and items can be modeled as a bipartite graph, with the users and items as the two sets of nodes and their interactions as edges. Roads or road sections in urban areas are often dependent due to spatial relations between them. These spatial relations can be represented by a traffic network where nodes are roads or road sections, and edges indicate the spatial relations. Therefore, graph neural networks have been naturally applied to facilitate various tasks of data mining. In the chapter, we illustrate how GNNs can be adopted for representative data mining tasks, including web data mining, urban data mining, and cybersecurity data mining.

12.2 Web Data Mining

Numerous Web-based applications, such as social media and e-commerce, have produced a massive volume of data. Web data mining is the application of data mining techniques to discover patterns from such data. This section demonstrates how GNNs advance two representative tasks of Web data mining, i.e., social network analysis and recommender systems.

12.2.1 Social Network Analysis

Social networks, which characterize relationships and/or interactions between users, are ubiquitous in the Web, especially social media. Social networks can be naturally modeled as graphs where users in the networks are the nodes, and the relationships and/or interactions are the edges. Graph neural networks have been adopted to facilitate various tasks on social networks such as social influence prediction (Qiu et al., 2018a), political perspective detection (Li and Goldwasser, 2019), and social representation learning (Wang et al., 2019a). Next, we detail some of these tasks.

Social Influence Prediction

In social networks, a person's emotions, opinions, behaviors, and decisions are affected by others. This phenomenon, which is usually referred to as social influence, is widely observed in various physical and/or online social networks. Investigating social influence is important for optimizing advertisement strategies and performing personalized recommendations. In (Qiu et al., 2018a), graph neural networks are adopted to predict local social influence for users in social networks. More specifically, given the local neighborhood of a user and the actions of users in the neighborhood, the goal is to predict whether the user will take the actions in the future or not. For example, in the Twitter platform, the prediction task can be whether a user would retweet posts (the action) on a certain topic given the action status (whether retweet or not) of other closed users (local neighborhood).

The relations between the users in the social network can be modeled as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ denotes the set of users in the social network and $\mathcal{E}$ denotes the relations between the users. For a node $v_i \in \mathcal{V}$, its local neighborhood is defined as its r-ego network $\mathcal{G}^r_{v_i}$ (Qiu et al., 2018a), which is a subgraph of $\mathcal{G}$ containing all nodes that are within r-hop away from the node v_i. Formally, the node set $\mathcal{V}^r_{v_i}$ and the edge set $\mathcal{E}^r_{v_i}$ for the r-ego network $\mathcal{G}^r_{v_i}$ can be defined as:

$$\mathcal{V}^r_{v_i} = \left\{ v_j \in \mathcal{V} \mid \mathrm{dis}(v_i, v_j) \leq r \right\},$$
$$\mathcal{E}^r_{v_i} = \left\{ (v_j, v_k) \in \mathcal{E} \mid v_j, v_k \in \mathcal{V}^r_{v_i} \right\},$$

where $\mathrm{dis}(v_i, v_j)$ denotes the length of the shortest path between nodes v_i and v_j. Furthermore, for each node $v_j \in \mathcal{V}^r_{v_i}/\{v_i\}$, there is an associated binary action state $s_j \in \{0, 1\}$. For example, in the case of Twitter, the action state $s_j = 1$ if the user v_j retweeted posts on a certain topic, otherwise $s_j = 0$. The action statuses for all nodes in $v_j \in \mathcal{V}^r_{v_i}/\{v_i\}$ can be summarized as $\mathcal{S}^r_{v_i} = \left\{ s_j \mid v_j \in \mathcal{V}^r_{v_i}/\{v_i\} \right\}$.

The goal of social influence prediction is to predict the action status of node v_i given $\mathcal{G}^r_{v_i}$ and $\mathcal{S}^r_{v_i}$, which is modeled as a binary classification problem.

To predict the action status s_i for node v_i, graph neural network models are applied to the ego-network to learn the node representation for node v_i, which is then utilized to perform the classification. Specifically, the GCN-Filter and the GAT-Filter (see Section 5.3.2 for details on GCN-Filter and GAT-Filter) are adopted as graph filters to build the graph neural networks in (Qiu et al., 2018a). The following features shown in Eq. (12.1) are utilized as the initial input for the graph neural network models.

$$\mathbf{F}^{(0)}_j = [\mathbf{x}_j, \mathbf{e}_j, s_j, ind_j], v_j \in \mathcal{V}^r_{v_i}. \tag{12.1}$$

In Eq. (12.1), $\mathbf{x}_j$ denotes the pre-trained embedding for node v_j learned by methods such as DeepWalk or LINE (see Section 4.2.1 for details on DeepWalk and LINE) over graph $\mathcal{G}$. The instance normalization trick, which normalizes the embeddings for nodes in $\mathcal{V}^r_{v_i}$, is adopted by (Qiu et al., 2018a) to improve the performance of the model. The vector $\mathbf{e}_j$ contains other node features such as structural features, content features, and demographic features if available. For node v_j, s_j is initialized to be 0 as its action status is unknown. The last element $ind_j \in \{0, 1\}$ is a binary variable indicating whether a node v_j is the ego user, i.e., $ind_j = 1$ only if $v_j = v_i$, otherwise 0.

Social Representation Learning

With the rapid development of social media such as Facebook, more and more services have been provided to users in social networks. For example, users can express their preferences for various movies, sports, and books on Facebook. The availability of these different types of social network services leads to different categories of user behaviors. Users may have similar preferences in one category of behaviors while they have quite different preferences in other behaviors. For example, two users may like the same kind of movies, but they like very different types of sports. To capture users' preference similarities in different behaviors, multiple vectors are utilized to represent each user where each vector corresponds to a specific category of behaviors (Wang et al., 2019a). In detail, for each user, these representations for different behaviors are conditioned on a general representation for this user. To learn these user representations, a graph neural network model is adapted to capture various preference similarities between users in different behaviours (Wang et al., 2019a). Next, we first formally describe the problem setting and then introduce the graph neural network model developed to learn conditional representations.

A social network can be modeled as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ represents the set of nodes (social users) and $\mathcal{E}$ denotes the edges

(social relations) connecting them. These relations between users can also be represented by the adjacency matrix of the graph $\mathbf{A}$. Furthermore, the users also have interactions with items such as movies, books and sports, which are organized to different categories. Specifically, the set of items for a category c (e.g. books) is denoted as $\mathcal{I}_c$ and the interactions between the users and these items are described by an interaction matrix $\mathbf{R}^c$, where $\mathbf{R}^c_{i,j} = 1$ only when the user v_i has interacted with the j-th item in the category c, otherwise 0. The goal of conditional representation learning is to learn a set of representations for each user v_j, where each conditional representation for a specific category c can capture the social structure information in $\mathbf{A}$ and also the preference in the category c described in $\mathbf{R}^c$. The representation learning framework is designed based on the MPNN framework as introduced in Section 5.3.2. Its message function $M()$ and the update function $U()$ (for the l-th layer) are described as follows. The message function $M()$ in the MPNN framework generates a message to pass to the center node v_i from its neighbor v_j. To capture various similarities between nodes v_i and v_j in different categories, the representations of these nodes are mapped to different categories as:

$$\mathbf{F}^{(l-1)}_{j|c} = \mathbf{F}^{(l-1)}_{j} \odot \mathbf{b}^{(l-1)}_{c},$$

where $\mathbf{b}^{(l-1)}_c$ is a learnable binary mask shared by all nodes to map the input representation $\mathbf{F}^{(l-1)}_j$ to the conditional representation $\mathbf{F}^{(l-1)}_{j|c}$ for the category c. Then, the message from node v_j to node v_i is generated as:

$$\mathbf{F}^{(l-1)}_{v_j \to v_i} = M(\mathbf{F}^{(l-1)}_i, \mathbf{F}^{(l-1)}_j) = \sum_{c=1}^{C} \alpha^{(l-1)}_{i,j|c} \cdot \mathbf{F}^{(l-1)}_{j|c},$$

where C denotes the number of categories and $\alpha^{(l-1)}_{i,j|c}$ is the attention score learned as:

$$e^{(l-1)}_{i,j|c} = \mathbf{h}^{(l-1)^\top} \text{ReLU}\left(\left[\mathbf{F}^{(l-1)}_{i|c}, \mathbf{F}^{(l-1)}_{j|c}\right] \mathbf{\Theta}^{(l-1)}_a\right),$$
$$\alpha^{(l-1)}_{i,j|c} = \frac{\exp\left\{e^{(l-1)}_{i,j|c}\right\}}{\sum_{c=1}^{C} \exp\left\{e^{(l-1)}_{i,j|c}\right\}},$$

where $\mathbf{h}^{(l-1)}$ and $\mathbf{\Theta}^{(l-1)}_a$ are parameters to be learned. The attention mechanism is utilized to ensure that more similar behaviours between users will contribute more when generating the message. After generating the messages, the repre-

sentation for node v_i is updated with the update function as:

$$\mathbf{m}_i^{(l-1)} = \sum_{v_j \in \mathcal{N}(v_i)} \mathbf{F}_{v_j \to v_i}^{(l-1)}, \tag{12.2}$$

$$\mathbf{F}^{(l)} = U(\mathbf{F}_i^{(l-1)}, \mathbf{m}_i^{(l-1)}) = \alpha\left(\left[\mathbf{F}_i^{(l-1)}, \mathbf{m}_i^{(l-1)}\right] \mathbf{\Theta}_u^{(l-1)}\right), \tag{12.3}$$

where $\mathbf{\Theta}_u^{(l-1)}$ is the parameters for the update function and $\alpha()$ denotes some activation functions. Note that, after stacking L layers of the above MPNN filtering operations, the final representations $\mathbf{F}_i^L$ can be obtained, which are then mapped to the conditional representation $\mathbf{F}_{i|c}^L$. The final conditional representation $\mathbf{F}_{i|c}^L$ is utilized to recover the interaction information $\mathbf{R}^c$, which serves as the training objective of the framework. Hence, the learned conditional representations capture both the social structure information and the user-item interaction information for a specific category.

12.2.2 Recommender Systems

Recommender systems have been widely applied to many online services such as e-commerce, video/music streaming services, and social media to alleviate the problem of information overload. Collaborative filtering (CF) (Goldberg et al., 1992; Resnick and Varian, 1997; Goldberg et al., 2001), which utilizes users' historical behavior data to predict their preferences, is one of the most important techniques for developing recommender systems. A key assumption of the collaborative filtering technique is that users with similar historical behaviors have similar preferences. Collaborative filtering approaches usually encode such information into vector representations of users and items, which can reconstruct the historical interactions (Koren et al., 2009; Wang et al., 2019h). When learning these representations, the historical interactions are usually not explicitly utilized but only served as the ground truth for the reconstruction. These historical interactions between users and items can be modeled as a bipartite graph $\mathcal{G} = \{\mathcal{U} \cup \mathcal{V}, \mathcal{E}\}$. Specifically, the set of users can be denoted as $\mathcal{U} = \{u_1, \ldots, u_{N_u}\}$, the set of items can be indicated as $\mathcal{V} = \{v_1, \ldots, v_{N_v}\}$, and the interactions between them can be represented as $\mathcal{E} = \{e_1, \ldots, e_{N_e}\}$, where $e_i = (u_{(i)}, v_{(i)})$ with $u_{(i)} \in \mathcal{U}$ and $v_{(i)} \in \mathcal{V}$. These interactions can also be described by an interaction matrix $\mathbf{M} \in \mathbb{R}^{N_u \times N_v}$, where the i, j-th element of $\mathbf{M}$ indicating the interaction status between the user u_i and item v_j. Specifically, $\mathbf{M}_{i,j}$ can be the rating value user u_i gave to item v_j. It can also be a binary value with $\mathbf{M}_{i,j} = 1$ indicating user u_i interacted with item v_j. With the bipartite graph, the historical interactions can be explicitly utilized to model the representations for users and items by adopting graph neural network models (Berg et al., 2017; Ying et al., 2018b; Wang et al., 2019h).

Furthermore, side information about users and items such as social networks for users and knowledge graphs for items can also be modeled in the form of graphs. It is also incorporated for learning the representations with graph neural network models (Wang et al., 2019b,c,g; Fan et al., 2019). Next, we introduce representative collaborative filtering methods based on graph neural network models.

Collaborative Filtering

Typically, a collaborative filtering approach can be viewed as an encoder-decoder model, where the encoder is to encode each user/item into vector representations and the decoder is to utilize these representations to reconstruct the historical interactions. Hence, the decoder is usually modeled as a regression task (when reconstructing rating) or a binary classification task (when reconstructing the existence of the interactions). Thus, we mainly introduce the encoder part designed based on graph neural network models. The spatial graph filtering operations are adopted to update the representations for users and items. Specifically, for a given user, its representation is updated utilizing the information from its neighbors, i.e., the items he/she has interacted with. Similarly, for a given item, its representation is updated utilizing the information from its neighbors, i.e., the users which have interacted with it. Next, we describe the graph filtering process from the perspective of a given user u_i since the graph filtering process for items is similar. The graph filtering process (for the l-th layer) can be generally described using the MPNN framework as introduced in Section 5.3.2 as follows:

$$\begin{aligned}\mathbf{m}_i^{(l-1)} &= \text{AGGREGATE}\left(\left\{M(\mathbf{u}_i^{(l-1)}, \mathbf{v}_j^{(l-1)}, \mathbf{e}_{(i,j)}) \mid v_j \in \mathcal{N}(u_i)\right\}\right), \\ \mathbf{u}_i^{(l)} &= U(\mathbf{u}_i^{(l-1)}, \mathbf{m}_i^{(l-1)}),\end{aligned} \tag{12.4}$$

where $\mathbf{u}_i^{(l-1)}$, $\mathbf{v}_j^{(l-1)}$ denote the input representations of user u_i and item v_j for the l-th layer, $\mathbf{e}_{(i,j)}$ is the edge information (for example, rating information if it is available), $\mathcal{N}(u_i)$ indicates the neighbors of user u_i, i.e., the items he/she has interacted with and AGGREGATE(), $M()$, $U()$ are the aggregation function, the message function and the update function to be designed, respectively. In (Berg et al., 2017), different aggregation functions are proposed and one example is summation. The message function is designed to incorporate discrete ratings information associated with the interaction as below:

$$M\left(\mathbf{u}_i^{(l-1)}, \mathbf{v}_j^{(l-1)}\right) = \frac{1}{\sqrt{|\mathcal{N}(u_i)||\mathcal{N}(v_j)|}} \mathbf{v}_j^{(l-1)} \mathbf{\Theta}_{r(u_i,v_j)}^{(l-1)},$$

where $r(u_i, v_j)$ denotes the discrete rating (e.g, 1-5) the user u_i gave to the item v_j and $\mathbf{\Theta}_{r(u_i,v_j)}^{(l-1)}$ is shared by all the interactions with this rating. The update

function is implemented as:

$$U(\mathbf{u}_i^{(l-1)}, \mathbf{m}_i^{(l-1)}) = \text{ReLU}(\mathbf{m}_i^{(l-1)}\mathbf{\Theta}_{up}^{(l-1)}),$$

where $\mathbf{\Theta}_{up}^{(l-1)}$ is the parameter to be learned.

In (Wang et al., 2019h), summation is adopted as the AGGREGATE() function and the message function and the update function are implemented as:

$$M\left(\mathbf{u}_i^{(l-1)}, \mathbf{v}_j^{(l-1)}\right) = \frac{1}{\sqrt{|\mathcal{N}(u_i)||\mathcal{N}(v_j)|}}\left(\mathbf{v}_j^{(l-1)}\mathbf{\Theta}_1^{(l-1)} + (\mathbf{u}_i^{(l-1)}\mathbf{\Theta}_2^{(l-1)} \odot \mathbf{v}_j^{(l-1)})\right),$$
$$U(\mathbf{u}_i^{(l-1)}, \mathbf{m}_i^{(l-1)}) = \text{LeakyReLU}\left(\mathbf{u}_i^{(l-1)}\mathbf{\Theta}_3^{(l-1)} + \mathbf{m}_i^{(l-1)}\right).$$

where $\mathbf{\Theta}_1^{(l-1)}$, $\mathbf{\Theta}_2^{(l-1)}$ and $\mathbf{\Theta}_3^{(l-1)}$ are the parameters to be learned.

Collaborative Filtering with Side Information For Items

Knowledge graphs, which describe the relations between items, are utilized as another resource of information in addition to the historical interactions. Graph neural network models have been adopted to incorporate the information encoded in knowledge graphs while learning representations for items (Wang et al., 2019c,b,g). Specifically, a knowledge graph with the set of items $\mathcal{V}$ as entities can be denoted as $\mathcal{G}_k = \{\mathcal{V}, \mathcal{E}_k, \mathcal{R}\}$, where $\mathcal{R}$ denotes the set of relations in the knowledge graph and each relational edge $e \in \mathcal{E}_k$ can be denoted as $e = (v_i, r, v_j)$ with $r \in \mathcal{R}$. For an item v_i, its connected items in the knowledge graph provide another resource to aggregate information. To aggregate the information while differentiating the importance of various relations, attention mechanism is adopted. Specifically, in (Wang et al., 2019g), the attention score α_{irj} for a relation (v_i, r, v_j) are calculated following the idea of knowledge graph embedding method TransR (Lin et al., 2015) as:

$$\pi(v_i, r, v_j) = \left(\mathbf{v}_j^{(0)}\mathbf{\Theta}_r^{(l-1)}\right)^\top \tanh\left(\mathbf{v}_i^{(0)}\mathbf{\Theta}_r^{(l-1)} + \mathbf{e}_r\right),$$
$$\alpha_{irj} = \frac{\exp(\pi(v_i, r, v_j))}{\sum_{(r,v_j)\in\mathcal{N}^k(v_i)} \exp(\pi(v_i, r, v_j))},$$

where $\mathbf{v}_i^{(0)}, \mathbf{e}_r$, and $\mathbf{\Theta}_r^{(l-1)}$ are the entity embedding, relation embedding, and the transformation matrix learned from TransR (Lin et al., 2015), and $\mathcal{N}^k(v_i)$ denotes the neighbors of v_i in the knowledge graph $\mathcal{G}_k$. The graph filtering process to update the representation for an item v_i (for the l-th layer) is as:

$$\mathbf{m}_i^{(l-1)} = \sum_{(r,v_j)\in\mathcal{N}^k(v_i)} \alpha_{irj}\mathbf{v}_j^{(l-1)},$$
$$\mathbf{v}_i^{(l)} = U(\mathbf{v}_i^{(l-1)}, \mathbf{m}_i^{(l-1)}) = \text{LeakyReLU}([\mathbf{v}_i^{(l-1)}, \mathbf{m}_i^{(l-1)}]\mathbf{\Theta}_{up}^{(l-1)}), \tag{12.5}$$

where $U()$ is the update function and $\mathbf{\Theta}_{up}^{(l-1)}$ is the parameter to be learned. The embeddings $\mathbf{v}_i^{(0)}$ learned from TransR are served as the input for the first layer. Note that the entity embeddings, relation embeddings and the transformation matrix learned from TransR are fixed during the propagation described by Eq. (12.5). Hence, the attention score α_{irj} is shared in different graph filter layers. Furthermore, the interactions between users and items are incorporated into the knowledge graph as a special relation *interaction* (Wang et al., 2019g). Specifically, each $e_i = (u_{(i)}, v_{(i)}) \in \mathcal{E}$ is transformed to a relational edge $(u_{(i)}, r, v_{(i)})$ with $r = interaction$. Hence, both the user representations and the item representations can be updated utilizing Eq. (12.5).

On the other hand, in (Wang et al., 2019c,b), the attention score is designed to be personalized for each user. In particular, when considering the impact from one entity v_j on another entity v_i, the user we want to recommend items to should also be considered. For example, when recommending movies to users, some users may prefer to movies from certain directors while others might prefer to movies acted by certain actors. Hence, when learning item embeddings specifically for performing recommending items to a user u_k, the attention score for aggregation can be modeled as:

$$\pi(v_i, r, v_j|u_k) = \mathbf{u}_k^T \mathbf{e}_r,$$

where $\mathbf{u}_k$ and $\mathbf{e}_r$ are the user embedding and the relation embedding, respectively. Specifically, this process can be also regarded as inducing a knowledge graph for each user. Note that, in (Wang et al., 2019c,b), only the knowledge graph is explicitly utilized for learning representations, while the historical interactions are only served as the ground-truth for the reconstruction. Hence, the user representation $\mathbf{u}_k$ is just randomly initialized as that in matrix factorization (Koren et al., 2009).

Collaborative Filtering with Side Information For Users

Social networks, which encode the relations/interactions between the users in $\mathcal{U}$ can serve as another resource of information besides the user-item interaction bipartite graph. The social network can be modeled as a graph $\mathcal{G}_s = \{\mathcal{U}, \mathcal{E}_s\}$, where $\mathcal{U}$ is the set of nodes (the users) and $\mathcal{E}_s$ is the set of edges describing the social relations between the users. In (Fan et al., 2019), graph neural network models have been adopted to learn representations for users and items utilizing both information. Specifically, the items' representations are updated by aggregating information from neighboring nodes (i.e., the users that have interacted with the item) in the interaction bipartite graph $\mathcal{G}$ as similar to the GNN models with pure collaborative filtering introduced in the previous sections. For users, the information from the two resources (i.e., the user-item

interaction bipartite graph $\mathcal{G}$ and the social network $\mathcal{G}_s$) are combined together to generate the user representations as:

$$\mathbf{u}_i^{(l)} = \left[\mathbf{u}_{i,\mathcal{I}}^{(l)}, \mathbf{u}_{i,\mathcal{S}}^{(l)}\right] \mathbf{\Theta}_c^{(l-1)},$$

where $\mathbf{u}_{i,\mathcal{I}}^{(l)}$ denotes the representation for user u_i learned by aggregating information from the neighboring items in the interaction bipartite graph, and $\mathbf{u}_{i,\mathcal{S}}^{(l)}$ indicates its representation learned by aggregating information from neighboring users in the social network. $\mathbf{\Theta}_c^{(l-1)}$ is the parameters to be learned. Specifically, $\mathbf{u}_{i,\mathcal{S}}^{(l)}$ is updated with the parameter $\mathbf{\Theta}_{\mathcal{I}}^{(l-1)}$ as:

$$\mathbf{u}_{i,\mathcal{S}}^{(l)} = \sigma\left(\sum_{u_j \in \mathcal{N}^s(u_i)} \mathbf{u}_j^{(l-1)} \mathbf{\Theta}_{\mathcal{I}}^{(l-1)}\right),$$

where $\mathcal{N}^s(u_i)$ is the set of neighboring users of user u_i in the social network. Meanwhile, $\mathbf{u}_{i,\mathcal{I}}^{(l)}$ is generated with the parameter $\mathbf{\Theta}_{\mathcal{S}}^{(l-1)}$ as:

$$\mathbf{u}_{i,\mathcal{I}}^{(l)} = \sigma\left(\sum_{v_j \in \mathcal{N}(u_i)} \left[\mathbf{v}_j^{(l-1)}, \mathbf{e}_{r(i,j)}\right] \mathbf{\Theta}_{\mathcal{S}}^{(l-1)}\right), \tag{12.6}$$

where $\mathcal{N}(u_i)$ is the set of items that user u_i has interacted with and $\mathbf{e}_{r(i,j)}$ is the rating information. In (Fan et al., 2019), the ratings are discrete scores and the rating information $\mathbf{e}_{r(i,j)}$ is modeled as embeddings to be learned.

12.3 Urban Data Mining

The development of sensing technologies and computing infrastructures has enabled us to collect large volumes of data in urban areas such as air quality, traffic, and human mobility. Mining such urban data provides us unprecedented opportunities to tackle various challenging problems introduced by urbanization, such as traffic congestion and air pollution. Next, we demonstrate how GNNs can advance urban data mining tasks.

12.3.1 Traffic Prediction

Analyzing and forecasting the dynamic traffic conditions are of great significance to the planning and construction of new roads and transportation management of smart cities in the new era. In traffic study, the traffic flow data can be usually treated as time series. It consists of traffic flow information such as traffic speed, volume, and density at multiple time steps. Meanwhile,

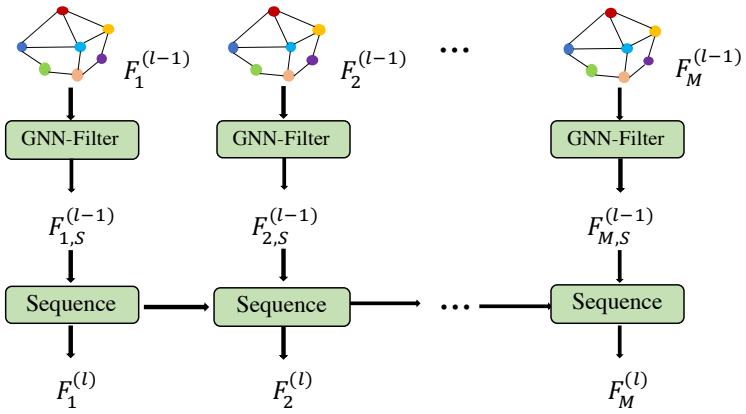

Figure 12.1 A learning layer in a typical framework for traffic prediction

roads or road sections are not independent of each other since there exist spatial relations between them. These spatial relations between roads are typically represented by a traffic network, where roads or road sections are the nodes, and spatial relations are encoded by the edges between them. To achieve better performance for traffic forecasting, it is desired to capture both the spatial and temporal information. Graph neural network models are adopted for spatial relations, while temporal information is captured by sequential modeling methods such as convolutional neural networks, recurrent neural networks, and transformers (Yu et al., 2017; Guo et al., 2019; Wang et al., 2020a). Next, we describe how graph neural networks can capture spatial relations and be incorporated with sequential modeling methods for both the spatial and temporal information.

The traffic network can be denoted as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the set of nodes (roads or road sections) with N as the number of nodes in the traffic network, and $\mathcal{E}$ is the set of edges describing the spatial relations between nodes. The connections between the nodes can also be described by an adjacency matrix $\mathbf{A}$. The traffic status information such as traffic speed in the traffic network at a specific time t can be represented at a vector $\mathbf{x}_t \in \mathbb{R}^{N \times d}$, where the i-th row of $\mathbf{x}_t$ is corresponding to node v_i in the traffic network. The task of traffic forecasting is to predict the traffic status for the next H time steps given the observations from the previous M time steps. Specifically, it can be expressed as:

$$(\hat{\mathbf{X}}_{M+1}, \ldots, \hat{\mathbf{X}}_{M+H}) = f(\mathbf{X}_1, \ldots, \mathbf{X}_M), \tag{12.7}$$

where $f()$ is the model to be learned, and $\hat{\mathbf{X}}_t$ denotes the predicted traffic status at time step t. A typical framework to tackle this problem is first to learn refined node representations at each time step by capturing both spatial and temporal information and then utilize node representations to perform predictions for the future time steps. These representations are refined layer by layer where each learning layer updates the representations by capturing both spatial and temporal relations. The l-th learning layer is shown in Figure 12.1, which consists of two components: 1) the spatial graph filtering operation to capture the spatial relations; and 2) the sequence model to capture the temporal relations. The spatial graph filtering operation is applied to node representations in each time step as follows:

$$\mathbf{F}_{t,\mathcal{S}}^{(l)} = \text{GNN-Filter}(\mathbf{F}_t^{(l-1)}, \mathbf{A}), t = 1, \ldots, M, \tag{12.8}$$

where $\mathbf{F}_t^{(l-1)}$ is the node representations at time step t after the $(l-1)$-th learning layer while $\mathbf{F}_{t,\mathcal{S}}^{(l)}$ denotes the node representations after the l-th spatial graph filtering layer, which will serve as the input for the l-th sequence model. Note that the GNN-Filter is shared by all time steps. Different graph filtering operations can be adopted. For example, in (Yu et al., 2017), the GCN-Filter (see details on GCN-Filter in Section 5.3.2) is adopted while in (Guo et al., 2019; Wang et al., 2020a), attention mechanism is utilized to enhance the graph filtering operation. The output of the spatial graph filtering operation is a sequence, i.e.,$(\mathbf{F}_{1,\mathcal{S}}^{(l)}, \ldots, \mathbf{F}_{M,\mathcal{S}}^{(l)})$, which will be fed into a sequence model to capture the temporal relations as:

$$\mathbf{F}_1^{(l)}, \ldots, \mathbf{F}_M^{(l)} = \text{Sequence}(\mathbf{F}_{1,\mathcal{S}}^{(l)}, \ldots, \mathbf{F}_{M,\mathcal{S}}^{(l)}), \tag{12.9}$$

where the output $\mathbf{F}_1^{(l)}, \ldots, \mathbf{F}_M^{(l)}$ will in turn serve as the input to the next spatial graph filtering layer. Various sequence modeling methods can be adopted as the Sequence() function. For example, in (Yu et al., 2017), 1-D convolutional neural networks are adopted to deal with the temporal information. GRU model and the transformer are adopted in (Wang et al., 2020a) to capture the temporal relations. The final representations $\mathbf{F}_1^{(L)}, \ldots, \mathbf{F}_M^{(L)}$ are obtained by applying L learning layers as described above and then utilized to predict the traffic status in the future. Note that $\mathbf{F}_1^{(0)}, \ldots, \mathbf{F}_M^{(0)}$ can be initialized with node information such as traffic status $\mathbf{X}_1, \ldots, \mathbf{X}_M$.

12.3.2 Air Quality Forecasting

Air pollution has raised public concerns due to its adverse effects on the natural environment and human health. Hence, it is important to forecast the air quality, which can provide public guidance for people affected by this issue. The

air quality forecasting problem can be formulated in a spatial-temporal form as the air quality in nearby locations is related, and the air quality in one location is temporally evolving. The spatial relations between different locations can be denoted as a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the locations are the nodes, and the edges describe the geographic relations between nodes. The air quality status includes different indices such as $PM_{2.5}$, PM_{10}, NO_2, SO_2, O_3, and CO. The air quality status measured at time t for all locations in $\mathcal{V}$ is denoted as $\mathbf{X}_t$, where the i-th row of $\mathbf{X}_t$ is corresponding to the air quality status of location $v_i \in \mathcal{V}$. In the task of air quality forecasting, we aim to predict air quality status for all locations in a future time slot given the historical status. Generally, we denote the air quality status we plan to predict at time t as $\mathbf{Y}_t$, where the i-th row is corresponding to the i-th location v_i. Then, the air quality forecasting problem can be formulated as:

$$(\mathbf{Y}_{M+1}, \ldots, \mathbf{Y}_{M+H}) = f(\mathbf{X}_1, \ldots, \mathbf{X}_M), \tag{12.10}$$

where $(\mathbf{X}_1, \ldots, \mathbf{X}_M)$ is the observed air quality status from the previous M steps and $(\mathbf{Y}_{M+1}, \ldots, \mathbf{Y}_{M+H})$ is the air quality status we aim to forecast in the next H steps. The framework introduced in Section 12.3.1 can be used to forecast air quality. In (Qi et al., 2019) where the goal is to predict only $PM_{2.5}$, GCN-Filter is utilized to capture the spatial relations between different locations while an LSTM model is adopted to capture the temporal relations.

12.4 Cybersecurity Data Mining

With the growing use of the Internet, new vulnerabilities and attack schemes are discovering every day for computer and communication systems. These changes and dynamics have posed tremendous challenges to traditional security approaches. Data mining can discover actionable patterns from data, and thus it has been employed to tackle these cybersecurity challenges. Given that cybersecurity data can be denoted as graphs, GNNs have facilitated various aspects of cybersecurity data such as spammer detection and fake news detection.

12.4.1 Malicious Account Detection

Cyber attackers aim to attack large-scale online services such as email systems, online social networks, e-commerce, and e-finance platforms by creating malicious accounts and propagating spamming messages. These attacks are harmful to these online services and could even cause a huge financial loss in certain

circumstances. Hence, it is important to detect these malicious accounts effectively. Graph neural network models have been utilized to facilitate the task of malicious account detection. In (Liu et al., 2018b), two patterns on the malicious accounts are observed. First, malicious accounts from the same attacker tend to signup or login to the same device or a common set of devices due to the attackers' limited resources. Second, malicious accounts from the same group tend to behave in batches, i.e., they signup or login in a burst of time. A graph between accounts and devices is built based on these two observations, and the malicious account detection task is treated as a binary semi-supervised classification task on this graph, where the goal is to tell whether an account is malicious or not. Next, we first describe the graph construction process and then discuss how graph neural network models can be leveraged for malicious account detection.

Graph Construction

There are two types of objects involving in this task: accounts and devices. The concept of "device" can be very general, including IP addresses or phone numbers. We denote the set of types of devices as $\mathcal{D}$. Assume that there are N nodes, including accounts and devices. An edge is observed between an account and a device if an account has activities (e.g., signups and logins) in this specific device. This constructed graph can be denoted as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ and $\mathcal{E}$ are the node and edge sets, respectively. The relations between these nodes in $\mathcal{V}$ can also be described by an adjacency matrix $\mathbf{A}$. $|\mathcal{D}|$ subgraphs $\{\mathcal{G}^{(d)} = \{\mathcal{V}, \mathcal{E}^{(d)}\}\}$ are extracted from the graph $\mathcal{G}$, where $\mathcal{G}^{(d)}$ is the subgraph consisting of all nodes but only edges involving type $d \in \mathcal{D}$ devices. We use $\mathbf{A}^{(d)}$ to denote the adjacency matrix of the subgraph $\mathcal{G}^{(d)}$. A feature vector $\mathbf{x}_i \in \mathbb{R}^{p+|\mathcal{D}|}$ is associated with each node $v_i \in \mathcal{V}$. Specifically, the first p elements in $\mathbf{x}_i$ denote the frequency of activities in p consecutive periods. For example, in (Liu et al., 2018b), $p = 168$ and each time period is 1 hour. The last $|\mathcal{D}|$ elements in $\mathbf{x_i}$ indicate the type of the device. If the node is a device, it is a one-hot indicator of its type, and it is all 0 if it is an account.

Malicious Account Detection with Graph Neural Networks

Graph neural networks are utilized to refine the node features, which are then leveraged to perform malicious account detection (or the binary classification). The formulation of the semi-supervised classification task is the same as that in Section 5.5.1. Hence, we mainly introduce the process of learning node features. More specifically, we introduce a graph filter dedicated to this task as

follows:

$$\mathbf{F}^{(l)} = \sigma\left(\mathbf{X}\mathbf{\Theta}^{(l-1)} + \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}}\mathbf{A}^{(d)}\mathbf{F}^{(l-1)}\mathbf{\Theta}_{(d)}^{(l-1)}\right),$$

where $\mathbf{X}$ denotes the input features of all nodes, $\mathbf{F}^{(l)}$ is the hidden representations after l-th graph filtering layer with $\mathbf{F}^{(0)} = \mathbf{0}$, and $\{\mathbf{\Theta}^{(l-1)}, \mathbf{\Theta}_{(d)}^{(l-1)}\}$ are parameters to be learned. Note that the graph filtering operation differs from those introduced in Section 5.3.2, as it utilizes the input features $\mathbf{X}$ in each graph filtering layer. The goal is to ensure that the account activity patterns encoded by $\mathbf{X}$ can be better preserved. After L graph filtering layers, these features are used to perform the binary classification. Since the malicious accounts (from the same attacker) tend to connect with the same set of device nodes, the graph filtering process will enforce them to have similar features. Meanwhile, the activity patterns of the accounts are captured by the input feature $\mathbf{X}$. Thus, aforementioned two patterns can be captured by the graph neural network models, which benefits the task of malicious account detection.

12.4.2 Fake News Detection

Online social media has become one of the critical sources for people to obtain news due to its easy access and instant dissemination. While being extremely convenient and efficient, these platforms also significantly increase the risk of propagating fake news. Fake news could lead to many negative consequences or even severe societal issues and substantial financial loss. Hence, it is immensely vital to detect fake news and prevent it from propagating through social media. Substantial empirical evidence has shown that fake news has different propagation patterns from real news in online social media (Vosoughi et al., 2018), which can be utilized to facilitate the task of fake news detection. In (Monti et al., 2019), each story is modeled as a graph, which characterizes its diffusion process and social relations in the social network platform such as Twitter. Then, the task of fake news detection is treated as a binary graph classification task, and graph neural network models are adopted to improve its performance. Next, we describe the process to form the graphs for the stories and then briefly introduce the graph neural network model designed for this task.

Graph Construction

We take the news diffusion process in Twitter as an example to illustrate the process of graph construction for each story. Given a story u with its corresponding tweets $\mathcal{T}_u = \{t_u^{(1)}, \ldots, t_u^{(N_u)}\}$ which mention u, the story u is described

by a graph $\mathcal{G}_u$. The graph $\mathcal{G}_u$ consists of all the tweets in $\mathcal{T}_u$ as nodes, while the edges either describe the news diffusion process or the social relations between the authors of these tweets. We next describe the two types of edges in this graph $\mathcal{G}_u$. We use $a(t_u^{(i)})$ to denote the author of a given tweet $t_u^{(i)}$. The first type of edges between the tweets is defined based on their authors, i.e, an edge exists between two tweets $t_u^{(i)}$ and $t_u^{(j)}$ if the $a(t_u^{(i)})$ follows $a(t_u^{(j)})$ or $a(t_u^{(j)})$ follows $a(t_u^{(i)})$. The second type of edges is based on the diffusion process of this news u through the social network, i.e., an edge exists between two tweets $t_u^{(i)}$ and $t_u^{(j)}$ if the news u spreads from one to the other. The news diffusion path is estimated via (Vosoughi et al., 2018), which jointly considers the timestamps of the tweets and the social connections between their authors. For convenience, we assume that the superscript of a tweet $t_u^{(i)}$ indicates its timestamp information, i.e., all tweets with superscripts smaller than i are created before $t_u^{(i)}$ while the ones with superscripts larger than i are created after $t_u^{(i)}$. Then, for a given tweet $t_u^{(i)}$, we estimate its spreading path as:

- If $a(t_u^{(i)})$ follows at least one author of the previous tweets $\{a(t_u^{(1)}), \ldots, a(t_u^{(i-1)})\}$, we estimate that the news spreads to $t_u^{(i)}$ from the very last tweet whose author is followed by $a(t_u^{(i)})$.
- If $a(t_u^{(i)})$ does not follow any authors of the previous tweets $\{a(t_u^{(1)}), \ldots, a(t_u^{(i-1)})\}$, then we estimate that the news spreads to $t_u^{(i)}$ from the tweet in $\{t_u^{(1)}, \ldots, t_u^{(i-1)}\}$ whose author has the largest number of followers.

Fake News Detection as Graph Classification

We can build a graph for each story u as described above, and then we treat the task of fake news detection as a binary graph classification task. The graph neural network framework for graph classification is introduced in Section 5.5.2, which can be directly applied to this task. Specifically, in (Monti et al., 2019), two graph filtering layers are stacked together to refine the node features, which are followed by a graph mean-pooling layer to generate a graph representation. The generated graph representation is then utilized to perform the binary classification.

12.5 Conclusion

This chapter describes how graph neural network models can be applied to advance various sub-fields of data mining, including Web data mining, urban data mining, and cybersecurity data mining. In Web data mining, we introduce representative methods using GNNs for social network analysis and recommender systems. In urban data mining, we discuss GNNs based models for

traffic prediction and air quality prediction. In cybersecurity data mining, we provide representative algorithms built on GNNs to advance malicious account detection and fake news detection.

12.6 Further Reading

There are more existing methods than the representative ones we have detailed for the data mining tasks introduced in this chapter. For example, social information is encoded by graph neural networks to predict the political perspective (Li and Goldwasser, 2019), and graph neural networks are employed for fraud detection (Wang et al., 2019d; Liu et al., 2020), and anti-money laundering (Weber et al., 2019). In addition, graph neural networks have been utilized to help more data mining tasks, such as community detection (Chen et al., 2017; Shchur and Günnemann, 2019) and anomaly detection (Wang et al., 2020b; Chaudhary et al., 2019).