Chapter 7

Traditional Graph Generation Approaches

The previous parts of this book introduced a wide variety of methods for learning representations of graphs. In this final part of the book, we will discuss a distinct but closely related task: the problem of *graph generation*.

The goal of graph generation is to build models that can generate realistic graph structures. In some ways, we can view this graph generation problem as the mirror image of the graph embedding problem, which we focused on in the previous parts of this book. Instead of assuming we are given a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ as *input* to our model, in graph generation we want the *output* of our model to be a graph $\mathcal{G}$.

Of course, simply generating an arbitrary graph is not necessarily that challenging. For instance, it is trivial to generate a fully connected graph or a graph with no edges. The key challenge in graph generation, however, is generating graphs that have certain desirable properties. As we will see in the following chapters, the way in which we define "desirable properties"—and how we perform graph generation—varies drastically between different approaches.

In this chapter, we begin with a discussion of traditional approaches to graph generation. These tradiational approaches pre-date most research on graph representation learning—and even machine learning research in general. The methods we will discuss in this chapter thus provide the backdrop to motivate the deep learning-based approaches that we will introduce in Chapter 8 and Chapter 9.

7.1 Traditional Approaches

Traditional approaches to graph generation generally involve specifying some kind of *generative process*, which define how the edges or relations in a graph form. In most cases we can frame this generative process as a way of specifying

the probability or likelihood $P(\mathbf{A}[u, v] = 1)$ of an edge existing between two nodes u and v.

The challenge is crafting some sort of generative process that is both tractable and also able to generate graphs with non-trivial properties or characteristics. Tractability is essential because we want to be able to sample or analyze the graphs that are generated. However, we also want these graphs to have some properties that make them good models for the kinds of graphs we see in the real world.

The three models we review in this subsection represent a small but representative subset of the traditional graph generation approaches that exist in the literature. For a more thorough survey and discussion, we recommend Newman [2018] as a useful resource.

7.1.1 Erdos-Renyi Model

Perhaps the simplest and most well-known generative model of graphs is the *Erdos-Renyi (ER)* model. In this model we define the likelihood of an edge occurring between any pair of nodes as

$$P(\mathbf{A}[u, v] = 1) = r, \forall u, v \in \mathcal{V}, u \neq v, \tag{7.1}$$

where $r \in [0, 1]$ is parameter controlling the density of the graph. In other words, the ER model simply assumes that the probability of an edge occurring between any pairs of nodes is equal.

The ER model is attractive due to its simplicity. To generate a random ER graph, we simply choose (or sample) how many nodes we want, set the density parameter r, and thee use Equation 7.1 to generate the adjacency matrix. Since the edge probabilities are all independent, the time complexity to generate a graph is $O(|\mathcal{V}|^2)$, i.e., linear in the size of the adjacency matrix.

The downside of the ER model, however, is that it does not generate very realistic graphs. In particular, the only property that we can control in the ER model is the density of the graph—since the parameter r is equal (in expectation) to the average degree in the graph. Other graph properties—such as the degree distribution, existence of community structures, node clustering coefficients, and the occurrence of other structural motifs—are not captured by the ER model. Moreover, it is well known that graphs generated by the ER model fail to reflect the distribution of these more complex graph properties, which are known to be important in the structure of real world graphs.

7.1.2 Stochastic Block Models

Most traditional graph generation approaches seek to improve the ER model by better capturing additional properties of real-world graphs, which the ER model ignores. One prominent example is the class of *stochastic block models (SBMs)*, which seek to generate graphs with community structure.

In a basic SBM model, we specify a number γ of different blocks: $\mathcal{C}_1, ..., \mathcal{C}_\gamma$. Every node $u \in \mathcal{V}$ then has a probability p_i of belonging to block i, i.e. $p_i =$

$P(u \in \mathcal{C}_i), \forall u \in \mathcal{V}, i = 1, ..., \gamma$ where $\sum_{i=1}^{\gamma} p_i = 1$. Edge probabilities are then specified by a block-to-block probability matrix $\mathbf{P} \in [0, 1]^{\gamma \times \gamma}$, where $\mathbf{C}[i, j]$ gives the probability of an edge between a node in block $\mathcal{C}_i$ and $\mathcal{C}_j$. The generative process for the basic SBM model is thus:

1. For every node $u \in \mathcal{V}$, we assign u to a block $\mathcal{C}_i$ by sampling from the categorical distribution defined by $(p_i, ..., p_\gamma)$.

2. For every pair of nodes $u \in C_i$ and $v \in C_j$ we sample an edge according to
$$P(\mathbf{A}[u, v] = 1) = \mathbf{C}[i, j]. \tag{7.2}$$

The key innovation in the SBM is that we can control the edge probabilities within and between different blocks or communities, and this allows us to generate graphs that exhibit community structure. For example, a common SBM practice is to set a constant value α on the diagonal of the $\mathbf{C}$ matrix—i.e., $\mathbf{C}[i, i] = \alpha, i = 1, ..., \gamma$—and a separate constant $\beta < \alpha$ on the off-diagonal entries—i.e., $\mathbf{C}[i, j] = \beta, i, j = 1, ..\gamma, i \neq j$. In this setting, nodes have a probability α of having an edge with another node that assigned to the *same community* and a smaller probability $\beta < \alpha$ of having another node that is assigned to a *different community*.

The SBM model described above represents just the most basic variation of the general SBM framework. There are many variations of the SBM framework, including approaches for bipartite graphs and graphs with node features [Newman, 2018]. The key insight that is shared across all these approaches, however, is the identify of crafting a generative graph model that can capture the notion of communities in a graph.

7.1.3 Preferential Attachment

The SBM framework described in the previous section can generate graphs with community structures. However, like the simple ER model, the SBM approach is limited in that it fails to capture the structural characteristics of individual nodes that are present in most real-world graphs.

For instance, in an SBM model, all nodes within a community have the same degree distribution. This means that the structure of individual communities is relatively homogeneous in that all the nodes have similar structural properties (e.g., similar degrees and clustering coefficients). Unfortunately, however, this homogeneity is quite unrealistic in the real world. In real-world graphs we often see much more heterogeneous and varied degree distributions, for example, with many low-degree nodes and a small number of high-degree "hub" nodes.

The third generative model we will introduce—termed the preferential attachment (PA) model–attempts to capture this characteristic property of real-world degree distributions Albert and Barabási [2002]. The PA model is built around the assumption that many real-world graphs exhibit *power law* degree distributions, meaning that the probability of a node u having degree d_u is

roughly given by the following equation:

$$P(d_u = k) \propto k^{-\alpha}, \tag{7.3}$$

where $\alpha > 1$ is a parameter. Power law distributions—and other related distributions—have the property that they are *heavy tailed*. Formally, being heavy tailed means that a probability distribution goes to zero for extreme values slower than an exponential distribution. This means that heavy-tailed distributions assign non-trivial probability mass to events that are essentially "impossible" under a standard exponential distribution. In the case of degree distributions, this heavy tailed nature essentially means that there is a non-zero chance of encountering a small number of very high-degree nodes. Intuitively, power law degree distributions capture the fact that real world graphs have a large number of nodes with small degrees but also have a small number of nodes with extremely large degrees.[1]

The PA model generates graphs that exhibit power-law degree distributions using a simple generative process:

1. First, we initialize a fully connected graph with m_0 nodes.

2. Next, we iteratively add $n - m_0$ nodes to this graph. For each new node u that we add at iteration t, we connect it to $m < m_0$ existing nodes in the graph, and we choose its m neighbors by sampling without replacement according to the following probability distribution:

$$P(\mathbf{A}[u, v]) = \frac{d_v^{(t)}}{\sum_{v' \in \mathcal{V}^{(t)}} d_{v'}^{(t)}}, \tag{7.4}$$

   where $d_v^{(t)}$ denotes the degree of node v at iteration t and $\mathcal{V}^{(t)}$ denotes the set of nodes that have been added to the graph up to iteration t.

The key idea is that the PA model connects new nodes to existing nodes with a probability that is proportional to the existing nodes' degrees. This means that high degree nodes will tend to accumulate more and more neighbors in a "rich get richer" phenomenon as the graph grows. One can show that the PA model described above generates connected graphs that have power law degree distributions with $\alpha = 3$ [Albert and Barabási, 2002].

An important aspect of the PA model—which distinguishes it from the ER and SBM models—is that the generation process is *autoregressive*. Instead of specifying the edge probabilities for the entire graph in one step, the PA model relies on an iterative approach, where the edge probabilities at step t depend on the edges that were added at step $t - 1$. We will see that this notion of autoregressive generation will reoccur within the context of deep learning approaches to graph generation in Chapter 9.

[1] There is a great deal of controversy regarding the prevalence of actual power law distributions in real-world data. There is compelling evidence that many supposedly power-law distributions are in fact better modeled by distributions like the log-normal Clauset et al. [2009] contains a useful discussion and empirical analysis of this issue.

7.2 Traditional Applications

The three previous subsections outlined three traditional graph generation approaches: the Erdos-Renyi (ER) model, the stochastic block model (SBM), and the preferential attachment (PA) model. The insight in these models is that we specify a generation process or probability model, which allows us to capture some useful property of real-world graphs while still being tractable and easy to analyze. Historically, these traditional generation models have been used in two key applications:

Generating synthetic data for benchmarking and analysis tasks

The first useful application of these generative models is that they can be used to generate synthetic graphs for benchmarking and analysis tasks. For example, suppose you've developed a community detection algorithm. It would be reasonable to expect that your community detection approach should be able to infer the underlying communities in a graph generated by an SBM model. Similarly, if you have designed a network analysis engine that is suppose to scale to very large graphs, it would be good practice to test your framework on synthetic graphs generated by the PA model, in order to ensure your analysis engine can handle heavy-tailed degree distributions.

Creating null models

The second key application task for traditional graph generation approaches is the creation of null models. Suppose you are researching a social network dataset. After analyzing this network and computing various statistics—such as degree distributions and clustering coefficients—you might want to ask the following question: How surprising are the characteristics of this network? Generative graph models provide a precise way for us to answer this question. In particular, we can investigate the extent to which different graph characteristics are probable (or unexpected) under different generative models. For example, the presence of heavy-tailed degree distributions in a social network might seem surprising at first glance, but this property is actually expected if we assume that the data is generated according to a preferential attachment process. In general, traditional generative models of graphs give us the ability to interrogate what sorts of graph characteristics can be easily explained by simple generative processes. In a statistical sense, they provide us with *null models* that we can use as reference points for our understanding of real-world graphs.