An Overview of Graph-based Data and Related Machine Learning Methods

Barry Xue
University of California, San Diego
zex001@ucsd.edu

Abstract

Graphs capture the relationship of data within a large complex system. As the world accumulates more data, we start seeing complex systems with graph structures appear across many fields. Some notable examples include giant online social networks, 3D manifolds in computer vision, and molecular models. In this report, we will summarize the terminology and mathematical foundations related to the study of graph structures (degree distribution, graph Laplacian matrix), identify and present graph-related machine learning tasks (node classification, link prediction), and explain some notable concepts that help to solve those tasks (node2vec, GCN). We will also present examples that utilized these tools introduced, and report their performance. This report will establish a basic understanding of graph theory, and summarize applicable models using these facts. The examples reported are good potential baselines for the quarter II project.

1. Introduction

Today's world is driven by algorithms that feed on data of various forms, including images, text, audio, and much more. Large among of data forms complex systems such as video feeds, social networks, molecular structures, etc. Each system has its unique structure and features. As a result, to extract utility from the data, we have considered the underlying structure corresponding to each data type. In this report, we are going to introduce several machine learning tasks on a graph; this includes node classification, link prediction, and graph clustering. In addition to identifying the three graph ML tasks, we will introduce the theoretical fundamentals and state-of-the-art techniques for completing these tasks. Graph structures can help capture the structure of data in different complex systems with generalizable features and statistics. This type of characterization allows people to gain a better understanding of each network, compare systems despite their corresponding data types, and enable different machine learning tasks. Figure 1.1 has a list of popular networks and their graph statistics. This is an example of understanding the networks and comparison between networks through their graph statistics.

Network	Nodes	Links	Directed / Undirected	N	L	‹K›
Internet	Routers	Internet connections	Undirected	192,244	609,066	6.34
WWW	Webpages	Links	Directed	325,729	1,497,134	4.60
Power Grid	Power plants, transformers	Cables	Undirected	4,941	6,594	2.67
Mobile-Phone Calls	Subscribers	Calls	Directed	36,595	91,826	2.51
Email	Email addresses	Emails	Directed	57,194	103,731	1.81
Science Collaboration	Scientists	Co-authorships	Undirected	23,133	93,437	8.08
Actor Network	Actors	Co-acting	Undirected	702,388	29,397,908	83.71
Citation Network	Papers	Citations	Directed	449,673	4,689,479	10.43
E. Coli Metabolism	Metabolites	Chemical reactions	Directed	1,039	5,802	5.58
Protein Interactions	Proteins	Binding interactions	Undirected	2,018	2,930	2.90

Figure 1.1: A list of networks with different data types with their corresponding graph statistics (number of nodes, number of links, average degree). [1]

The Network Science by Albert-László Barabási provides a detailed description of different measurements of graphs, such as degree distribution, connectedness measurements, and clustering coefficient. These metrics are useful for the evaluation of different graphs as early analysis, and the insight we learned can then help us diagnose issues related to specific tasks when visualization becomes unreliable [1]. Other mathematic definitions are often written into matrix formation to enable linear operation, this includes the degree matrix and the Laplacian matrix. These matrices are important as they repeatedly appear in graph-based operations. One concept is random walks on graphs, from which more advanced feature encoding techniques like node2vec and graph convolution are developed. In essence, the random walk procedure correlates the edges of connected nodes with a local probability distribution; by taking steps to traverse the graph according to the local distribution, we generate a larger distribution with the state of the graph at each time step. [2]. With the random walk, we can sample nodes and form subgraphs, and collect the structural context of nodes, all while maintaining control of the computational complexity thanks to its iterative nature. In addition, the linear algebra tools introduced previously enable the walking procedure to be applied on large-scale graph networks, which is ideal when we want to train a model on massive datasets.

Node2vec is an algorithm inspired by the ideas of sampling local structure from a graph with the random walk and creating embedding from structural context. Intuitively, node2vec is very similar to word2vec: they both generate embedding for the target based on their context. For word2vec, the target is one word and the context is the other words in the same sentence, whereas in node2vec, the target is a node and the context is that node's neighboring nodes. There are two differences between node2vec-generated embedding and a feature generated from the node's neighboring connection. First, node2vec transforms the raw connection to a low-dimensional representation; it can be viewed as a type of dimensionality reduction. Second, the node2vec algorithm considers the 2nd-order neighborhood (the immediate connections and their connections) of a node with user-defined bias terms. The original node2vec paper [3] highlights the purpose of the bias terms in their version of the random walk; there are two terms, the in-out parameter and the return parameter, both are defined by a probability provided as hyperparameters. The in-out parameter defines the probability for the walk to traverse out of the 1st-order neighbors to the

2nd-order neighbors. The return parameter defines the probability for the walk to return to the original node. With these two parameters tuned, this becomes a biased random walk that lies between complete Breath First Samping, which only considers the 1st-order neighborhood, and Depth First Sampling, which sample across the entire graph. In summary, node2vec is a scalable, effective method to generate node feature encoding for all types of tasks.

Graph Convolutional Neural Network (GCN) [4] is an extension of node2vec's approach to generating node's structure embedding and using it for classification and prediction tasks. The idea remains similar, utilizing the 2nd-order connections to make an educated description of the node's role in the local neighborhood of the graph. However, GCNs don't need to generate the node embedding beforehand, which is a separate learning task by itself. Instead, GCNs generate a new embedding and train for a specific task within one neural network. GCNs do this by convolving the graph's adjacency matrix with the original feature representation of the nodes in an iterative manner. Each iteration of convolution is one convolutional layer; the result of the connected convolutional layers will be fed directly to generate the result. For example, if we are using a GCN network for node classification, we will input the node connect feature from the adjacency matrix and train that network with the nodes' target labels. This makes GCNs more intuitive to use compared to other models because it doesn't require complex feature engineering and several pipelines to complete one task. In addition, GCNs are highly customizable. We can change the propagation rule of each of the convolution layers to change the network's behavior. More specifically, different propagation rules can be created by changing the normalization factor. And GCNs can adapt to different tasks because we can fit different types of output layers for training.

To present the aforementioned concepts, we will be using a couple of datasets from SNAP [4] and CORA [5]. SNAP stands for Stanford Large Network Dataset Collection. There are large network datasets available on SNAP from domains spanning social networks, website connections, and more. CORA is a collection of scientific publication and their citations. There are 2708 publications included with 5429 citations. The CORA dataset also includes the top 1433 unique words for each publication along with their category label. We will be relying on CORA to evaluate the models' performance.

Reference

[1] Barabási, A.-L. Network science by Albert-László Barabási. *BarabásiLab*. http://networksciencebook.com/chapter/2#networks-graphs. Accessed 30 October 2022

[2] Spielman, D. A. *Random Walks on Graphs*. *Spectral Graph Theory*. lecture. http://cs.yale.edu/homes/spielman/561/lect10-18.pdf. Accessed 30 October 2022

[3] Grover, A., & Leskovec, J. (2016, July 3). Node2vec: Scalable Feature Learning for Networks. *arXiv*. arXiv:1607.00653v1 [cs.SI]. https://arxiv.org/pdf/1607.00653.pdf. Accessed 30 October 2022

[4] Spielman, D. A. *Random Walks on Graphs*. *Spectral Graph Theory*. lecture. http://cs.yale.edu/homes/spielman/561/lect10-18.pdf. Accessed 30 October 2022

[5] Jure Leskovec, & Andrej Krevl. (2014). SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[6] Motl, J., & Schulte, O. (2015). The CTU Prague Relational Learning Repository. doi:10.48550/ARXIV.1511.03086. https://arxiv.org/abs/1511.03086.