Graph Embedding for Dimensionality Reduction

By Anuththari Gamage, ECE '18

Introduction

A graph is a powerful tool for modeling data and their interreationships. Using this representation, data sets can be embedded into Euclidean space, providing a more intuitive geometric interpretation of abstract data points. Additionally, such a graph embedding paves the way for dimensionality reduction, which is embedding high-dimensional data in a lower dimension so as to retain only the most critical features of the dataset. Dimensionality reduction is at the heart of all machine learning tasks, as it allows us to perform tasks such as data clustering and visualization more efficiently. This note explores some commonly used techniques for graph embedding and their applications.

Graphs

A graph is a versatile tool for modeling phenomena that we observe in a variety of contexts. The vertex and edge structure of a graph is particularly useful for modeling interactions between data points, which can then be used in combination with graph theoretic and machine learning methods to analyze the underlying data and make meaningful inferences. One commonly recognized example of a graph representation of data is a social network [1]. The vertices, or nodes, in this network each represent one person, and the edges between nodes represent a relationship between them, such as being connected through a social media platform. While this graph may seem simplistic, important inferences about the underlying data can be made with an accurate or nearly-accurate graph of this nature. For example, we can categorize or cluster people according to their political views using a graph representing each person and their interrelationships with other people, provided that we know the political affiliations of some of the people in this network.

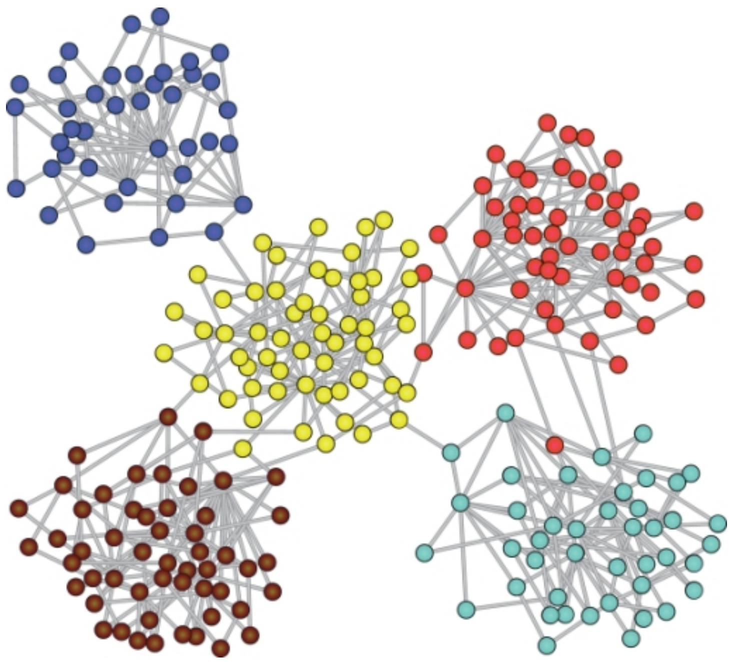

Figure 1: A graph with clustered identified (Image credits: openi.nlm.nih.gov)

Embedding Graphs

A graph embedding is simply a vector representation of the nodes of a graph in an arbitrary vector space. The dimension of the vector space is generally chosen according to the dataset at hand and our purpose in embedding it. An extremely low dimensional representation, such as vectors in $\mathbf{R}^2$ or $\mathbf{R}^3$, is optimal for visualizing data, since we can easily generate 2D and 3D plots. However, this loses much of the latent information of the dataset. A high-dimensional representation, $\mathbf{R}^{300}$ for example, would preserve this information leading to more accurate inferences from the embeddings, but this would present problems in scalability, since large vectors are costly in both time and memory required to process them.

Graph Embedding Techniques

Random Walks

In this approach, a graph embedding is generated by constructing a matrix based on the nodes and edges of a given graph. This matrix could be the adjacency matrix, transition probability matrix, the graph Laplacian or some other similar construction. Then, this matrix is factorized to obtain a vector for each node, usually by exploiting some property of the matrix that characterizes the graph, such as its eigendecomposition. Some factorization-based methods include Locally Linear Embedding (LLE) [2], Multidimensional Scaling (MDS) [3], Laplacian Eigenmaps [4], and spectral clustering [5]. This approach to graph embedding accounts for the entire set of nodes and their interconnections when generating the embedding, and therefore, they are well-suited for preserving the global cluster structure of the graph.

Random Walks

Graph embedding using random walks provides more flexibility in choosing how much information we preserve in the embedding. The parameters used in a random walk can be changed easily to constrain the range over which it collects information on the graph. These methods generally involve performing multiple random walks on the graph starting at each node. The nodes encountered in each random walk are recorded and the neighborhood of each node is characterized based on these co-occurrence counts. This information is then used to optimize randomly generated vector representations for each node such that the graph structure is preserved. Some graph emebedding algorithms based on random walks include node2vec [6], DeepWalk [7], and Vec [8].

Deep Learning

Given the recent developments in neural networks, graph embedding using deep learning techniques has also been explored. Unlike the factorization methods, which computes the embeddings using linear functions on the graph, deep learning methods allow us to compute non-linear functions on the graphs, which can lead to improved performance. However, deep learning based methods may be more difficult to set up than factorization or random walks graph embedding methods. Some examples for deep learning graph embedding methods include using an autoencoder to generate a low-dimensional representation of the data (SDNE) [9], using graph convolutional networks (GCN) [10], and using deep neural networks for generating graph representations (DNGR) [11].

Conclusion

Graph embedding is a powerful tool for analyzing large data sets for machine learning tasks. A good graph embedding retains the inherent geometric structure of the data and reduces the dimensionality of the data, leading to efficient data clustering and visualization. There are multiple approaches to graph embedding which are used for modeling a large variety of real-world networks, and there is much future research to be conducted for optimizing these methods for improved accuracy and speed. For our Senior Design project, we explored a number of different graph embedding and clustering algorithms, and we developed our own algorithm for faster clustering use random walks [12]. We also developed an intuitive graphical user interface for using these algorithms and testing them on various data sets to aid in future research and to introduce this topic to students who are interested in learning more about them.

References

[1] Rongjing Xiang, Jennifer Neville, and Monica Rogati, “Modeling relationship strength in online social networks,” in Proceedings of the 19th International Conference on World Wide Web, 2010.

[2] Sam T Roweis and Lawrence K Saul, “Nonlinear Dimension-ality Reduction by Locally Linear Embedding,” Science, vol.290, 2000.

[3] Josh B Tenenbaum, V de Silva, and J C Langford,“A Global Geometric Framework for Nonlinear Dimensionality Reduction,” Science, 2000.

[4] Mikhail Belkin and Partha Niyogi, “Laplacian Eigenmaps for Dimensionality Reduction and Data Representation,” Neural Computation, vol. 15, no. 6, pp. 13731396, 2003.

[5] Ulrike Von Luxburg, “A tutorial on spectral clustering,” Statistics and Computing, vol. 17, no. 4, pp. 395416, 2007.

[6] Aditya Grover and Jure Leskovec, “Node2Vec: Scalable Fea-ture Learning for Networks,” Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016.

[7] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena, “DeepWalk:online learning of social representations,” Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014.

[8] Weicong Ding, Christy Lin, and Prakash Ishwar,“Node Embed-ding via Word Embedding for Network Community Discovery,”pp. 110, 2016.

[9] Daixin Wang, Peng Cui, and Wenwu Zhu, “Structural DeepNetwork Embedding,” in Proceedings of the 22nd ACMSIGKDD International Conference on Knowledge Discoveryand Data Mining - KDD, 2016.

[10] T. N. Kipf and M. Welling, “Semi-Supervised Classificationwith Graph Convolutional Networks,” ArXiv e-prints, Sept.2016, https://arxiv.org/abs/1609.02907.

[11] Shaosheng Cao, Wei Lu, and Qiongkai Xu,“Deep neural networks for learning graph representations,” 2016, AAAI Conference on Artificial Intelligence.

[12] Brian Rappaport, Anuththari Gamage, and Shuchin Aeron, “Faster Clustering via Non-Backtracking Random Walks,” August 2017, http://arxiv.org/abs/1708.07967.