Notes on Graph Representation Learning

Rom Parnichkun

August 16, 2022

1 Representing Graphs

Formally, a graph is defined as follows.

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}). \tag{1}$$

In which, $\mathcal{V}$ and $\mathcal{E}$ are the nodes and edges of the graphs. Edges going from $u \in \mathcal{V}$ to $v \in \mathcal{V}$ are represented as $(u, v) \in \mathcal{E}$.

Graphs can be represented using an *adjacency matrix* $\mathbf{A} \in \mathbb{R}^{|\mathcal{V}|x|\mathcal{V}|}$, in which $A[u, v]$ represent $(u, v) \in \mathcal{E}$. Given output unit of every node $\mathbf{u} \in \mathbb{R}^{|V|}$, $\mathbf{s} = \mathbf{Au}$ is the combined weighed signal from every adjacent node. Moreover, $\mathbf{A}^i[u, v]$ represents the number of i-length paths that connects u and v.

Nodes within a graph may have *features* or *attributes* associated with them. They can be represented with matrix $\mathbf{X} \in \mathbb{R}^{|V| \times m}$, for nodes with m features.

1.1 Graph Laplacians

Graph *Laplacians* have many useful mathematical properties.

1.1.1 Unnormalized Laplacian

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{2}$$

where $\mathbf{D}$ is matrix with node degrees (see Equation 3) on its diagonals. The Laplacian summarizes many important properties of the graph.

2 Types of Graphs

- **Simple graphs** are graphs with at most one edge between each pair of nodes, and all edges are undirected.
- **Heterogeneous graphs**: Nodes have types, and edges satisfy constraints according to those types.
- **Multiplex graphs**: Graph is decomposed in a set of k *layers*. Every node is assumed to belong to every layer, and each layer corresponds to a unique relation.

3 Machine Learning on Graphs

The following lists the types of tasks that may be done on graphs.

- **Node classification**: Classify a node based on its features and relations to other nodes. Ideas such as *homophily*, which is the tendency for nodes to share attributes with their neighbors; *structural equivalence*, which is the idea that nodes with similar local neighborhood structures will have similar labels; and *heterophily*, which presumes that nodes will be preferentially connected to nodes with different labels, have been used to assist node classification.
- **Relation prediction**: Also known as link prediction, graph completion, relational inference, is the task of inferring edges between nodes in a graph.
- **Clustering and community detection**: Finding subgroups that may be useful.
- **Graph classification, regression, and clustering**: Graphs are treated as data points, which is to be classified, regressed, or clustered.

4 Graph Statistics and Features

The following lists node-level statistics and features.

- **Node degree**: The number of edges incident to a node.

$$d[u] = \sum_{v \in V} \mathbf{A}[u, v], \tag{3}$$

$$\mathbf{d} = \mathbf{A}\mathbf{1}_{|V|}, \tag{4}$$

  in which, $\mathbf{1}_{|V|} = (1, ..., 1) \in \mathbb{R}^{|V|}$.

- **Node centrality**: *Eigenvector centrality* is a measure of node importance. The eigenvector $\mathbf{e}$ corresponding to the largest eigenvalue λ contains the importance of each node on its indices.

$$\lambda \mathbf{e} = \mathbf{A}\mathbf{e} \tag{5}$$

$$\mathbf{e}[u] = \frac{1}{\lambda} \sum_{v \in V} \mathbf{A}[u, v]\mathbf{e}[v]. \tag{6}$$

- **Clustering coefficient**: Measures the degree of clustering for a node. It is the ratio between the number of adjacent node pairs that are also adjacent to each other by the total combination of node pairs for u.

$$c[u] = \frac{|(v_1, v_2) \in \mathcal{E} : v_1, v_2 \in \mathcal{N}(u)|}{\binom{d[u]}{2}}. \tag{7}$$

  An alternative way of viewing the clustering coefficient is that it counts the number of closed triangles that is connected to a particular node.

The following lists graph-level features and kernels.

- **Bag of nodes**: An aggregated view of node-level representations. Such as histograms of degrees, centralities, and clustering coefficients of the nodes in the graph.
- **Weisfeiler-Lehman (WL)**: Is a kernel and an algorithm that iteratively aggregates the neighborhood to find statistics beyond the immediate neighbor of a node. The algorithm is as follows:

1. Assign an initial label to each node. In many cases, this is the degree of each node; $l_0(v) = d[v]$.
2. Next, a new label is iteratively assigned with an aggregation function.

$$l_i(v) = \text{AGGR}\big(\{l_{i-1}(u)\ \forall u \in \mathcal{N}(v)\}\big) \tag{8}$$

3. After K iterations we have $l_K(v)$, which is a K-hop summary of every node. Which can be used to compare nodes at a higher level.

- **Graphlets**: Graphlets are subgraph structures that may commonly exhibit within larger graphs. The number of these different structures can be treated as graph-level features.

The following lists statistics on node-to-node relationships within graphs.

- **Similarity matrix**: $\mathbf{S} \in \mathbb{R}^{|V|\times|V|}$ is a matrix containing the number of shared neighbors of each node pairs.

$$\mathbf{S}[u, v] = |\mathcal{N}(u) \cap \mathcal{N}(v)|. \tag{9}$$

- **Sorenson index**: Similarity matrix with normalization.

$$\mathbf{S}_{\text{Sorenson}}[u, v] = \frac{2\mathbf{S}[u, v]}{d[u] + d[v]}. \tag{10}$$

- **Salton index**: Similarity matrix with normalization.

$$\mathbf{S}_{\text{Salton}}[u, v] = \frac{2\mathbf{S}[u, v]}{\sqrt{d[u]d[v]}}. \tag{11}$$

- **Jaccard index**: Similarity matrix with normalization.

$$\mathbf{S}_{\text{Jaccard}}[u, v] = \frac{\mathbf{S}[u, v]}{|\mathcal{N}(u) \cup \mathcal{N}(v)|}. \tag{12}$$

- **Resource allocation (RA) index**: Counts the inverse degrees of common neighbors,

$$\mathbf{S}_{\text{RA}}[v_1, v_2] = \sum_{u\in\mathcal{N}(v_1)\cap\mathcal{N}(v_2)} \frac{1}{d[u]} \tag{13}$$

- **Adamic-Adar (AA) index**: Similar to RA index but using the inverse log.

$$\mathbf{S}_{\text{AA}}[v_1, v_2] = \sum_{u\in\mathcal{N}(v_1)\cap\mathcal{N}(v_2)} \frac{1}{\log(d[u])} \tag{14}$$

- **Katz index**: Counts the number of paths of all lengths between a pair of nodes.

$$\mathbf{S}_{\text{Katz}}[u, v] = \sum_{i=1}^{\infty} \beta^i \mathbf{A}^i[u, v], \tag{15}$$

where $\beta \in \mathbb{R}^+$ is a user-defined parameter controlling how much weight is given to short versus long paths.

- **LHN similarity**: A normalized version of the Katz index, which reduces a high-degree bias in the Katz index.

$$\mathbf{S}_{\text{LNH}}[u, v] = \mathbf{I}[u, v] + \frac{2m}{d[u]d[v]} \sum_{i=0}^{\infty} \beta^i \lambda_1^{1-i} \mathbf{A}^i[u, v], \tag{16}$$

in which λ_1 is the largest eigenvalue of $\mathbf{A}$.

- **Random walk similarity**: Similarity between two nodes is proportional to how likely we are to reach each node from random walk starting from the other node.

$$\mathbf{P} = \mathbf{A}\mathbf{D}^{-1}. \tag{17}$$

Here $\mathbf{P}[\mathbf{u}, \mathbf{v}]$ is a stochastic adjacency matrix, with adjacency probability scaled proportional to the node's inverse degree.

$$\mathbf{q}_u = c\mathbf{P}\mathbf{q}_u + (1 - c)\mathbf{e}_u. \tag{18}$$

This implicit equation models the probability of reaching each node with a random walk policy. The c term determines the probability that the random walk restarts at u and $\mathbf{e}_u$ is a one-hot indicator vector for node u. The solution to this recurrence is as follows.

$$\mathbf{q}_u = (1 - c)(\mathbf{I} - c\mathbf{P})^{-1}\mathbf{e}_u. \tag{19}$$

Finally, random walk similarity formulated as follows.

$$\mathbf{S}_{\text{RW}}[u, v] = \mathbf{q}_u[v] + \mathbf{q}_v[u]. \tag{20}$$