Hamilton, Ying et al.: Representation Learning on Graphs. Methods and Applications

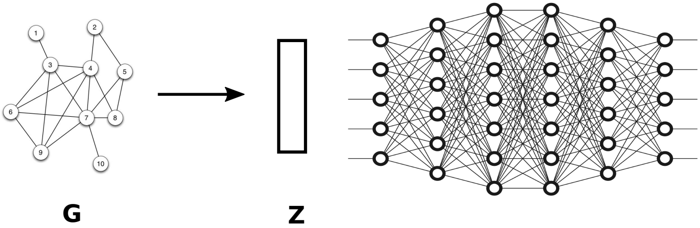
November 12, 2018

Graph Representation Learning

- graph input $\rightarrow$ feature vector
- Framework different approaches
- Embedding: Nodes / Subgraphs

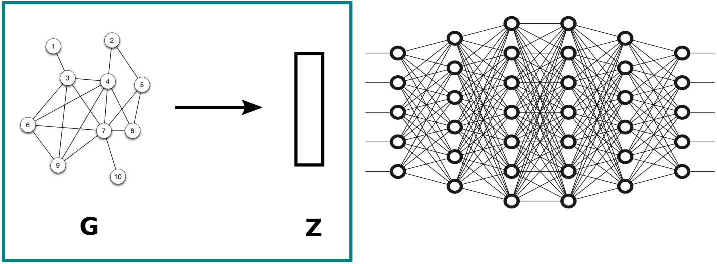
Graph Representation Learning

- graph input → feature vector
- Framework different approaches
- Embedding: Nodes / Subgraphs



Graph Representation Learning

- graph input → feature vector
- Framework different approaches
- Embedding: Nodes / Subgraphs



Problem Statement

$$
\begin{aligned}
G = (V, E) & \qquad \text{graph} \\
A & \qquad \text{adjacency} \\
X \in \mathbb{R}^{m \times |V|} & \qquad \text{features} \\
v_i \to z_i \in \mathbb{R}^d & \qquad \text{latent rep.} \\
& \\
& \qquad \text{where} \\
& v_i \in V, \quad d \ll |V|
\end{aligned} \tag{1}
$$

SOLUTION

Encoder - Decoder Model

$$\begin{aligned} ENC : V \rightarrow \mathbb{R}^d \\ DEC : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}^+ or \quad \mathbb{R}^d \rightarrow \mathbb{R}^+ \end{aligned} \tag{2}$$

Pairwise similarity function

$$s_G : V \times V \rightarrow \mathbb{R}^+ \tag{3}$$

Reconstruction Objective

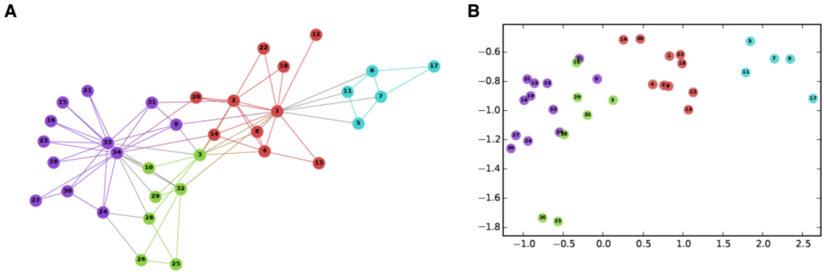
$$DEC(ENC(v_i), ENC(v_j)) = DEC(z_i, z_j) \approx s_G(v_i, v_j) \tag{4}$$

Loss

$$\mathcal{L} = \sum_{v_i, v_j \in V} \ell(DEC(z_i, z_j), s_G(v_i, v_j)) \tag{5}$$

Mapping

Optimize encoder by minimizing loss



Shallow Embedding

= Embedding *lookup*

$$ENC(v_i) = Zv_i \tag{6}$$

$$\begin{aligned} Z &\quad \text{embedding vectors} \\ v_i &\quad \text{one-hot indicator} \end{aligned}$$

- Factorization Based

$$DEC(z_i, z_j) = \quad ||z_i - z_j||_2^2 \quad \text{or} \quad z_i^T z_j \tag{7}$$

- Random Walk

$$DEC(z_i, z_j) = p_T(v_i|v_j) \quad \text{where} \quad T\text{...length of walk} \tag{8}$$

Problem: No parameters shared, no node attributes, no representation for new nodes

Neighborhood Autoencoder

→ v_i's neighborhood relation with entire graph
→ using Autoencoders
→ unary decoder

$$s_i \in \mathbb{R}^{|V|} \quad \forall \quad v_i \quad \text{neighborhood vector} \tag{9}$$

Objective

$$DEC(ENC(s_i)) = DEC(z_i) \approx s_i \tag{10}$$

Loss

$$\mathcal{L} = \sum ||DEC(z_i) - s_i||_2^2 \tag{11}$$

Neighborhood Aggregation and Convolutional Encoders

→ v_i's *local* neighborhood
→ aggregation of node attributes
→ iterative / recursive algorithm + dense NN layer

Algorithm 1: Neighborhood-aggregation encoder algorithm. Adapted from [29].

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\{\mathbf{W}^k, \forall k \in [1, K]\}$; non-linearity σ; differentiable aggregator functions $\{\text{AGGREGATE}_k, \forall k \in [1, K]\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$

Output: Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1 ... K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{COMBINE}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \text{NORMALIZE}(\mathbf{h}_v^k), \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

Adavantages: Shared Parameters, structure, attributes, new nodes

Graph Neural Networks

→ Learn representation of subgraphs
→ Mostly supervised
→ Idea: Message passing between nodes

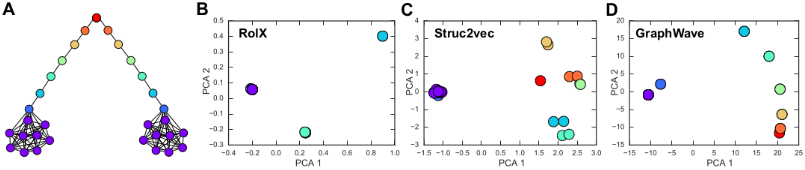
Aggregation

$$h_i^k = \sum_{v_j \in N(v_i)} h(h_j, x_i, x_j) \tag{12}$$

k iterations

h arbitrary function $\in C^1$ that is a contraction map

Structural Roles



Neighbor Aggregation

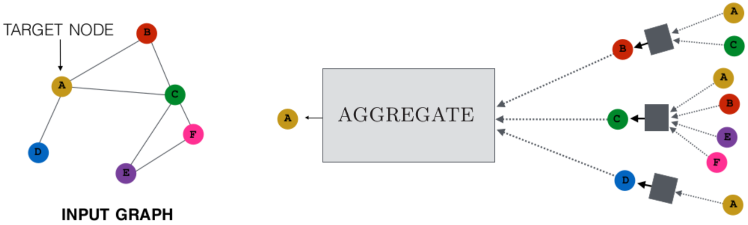


Figure 7: Overview of encoding in the neighborhood aggregation methods. To generate an embedding for node A, the model aggregates messages from A's local graph neighbors (*i.e.*, B, C, and D), and in turn, the messages coming from these neighbors are based on information aggregated from their respective neighborhoods, and so on. A "depth-2" version of this idea is shown (*i.e.*, information is aggregated from a two-hop neighborhood around node A), but in principle these methods can be of an arbitrary depth. At the final "depth" or "layer" the initial messages are based on the input node attributes.