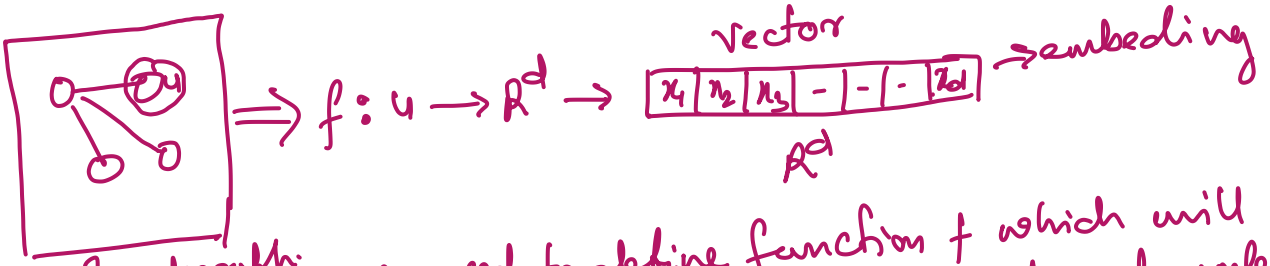
Node Embedding

Till now we have seen hand-crafted method to generate feature set for Nodes, links, Graph where main idea was that based on structural property two feature set should be closer if structural property is same. But we have seen constructing hand-crafted feature set is computationaly expensive and time consuming.
To overcome this we want mechanism which will automatically extract the features which is relevent for the graph.
Hand Crafted feature extraction is called feature engineering.
When we extract features automaticaly then that is called Graph representation learning.



Input graph. we need to define function f which will read input node u and transform in d-dimensional real number vector space. This vector can be used for various downstream task like

1. Node classification
2. Link prediction
3. Graph classification
4. Anomalus node detection
5. clustering etc.

Similarity in embeding also indicate similarity of node in network too.
Process of embeding is also called encoding network information.

Node Embeding frame work:
2 component:
1. Encoder
2. Node similarity function
3. Decoder

1. Encoder: Process of generating $\mathbb{R}^d$ vector space embeding.
2. In original network we need to define node similarity function which measures the similarity in original network.

③ Decoder: Maps the embedding to similarity score.

④ Optimize the parameter of encoder so that-

similarity$(u,v) \approx z_v^T z_u$ → similarity in embedding space.

↓ similarity in original network

$ENC(u) \rightarrow z_u$

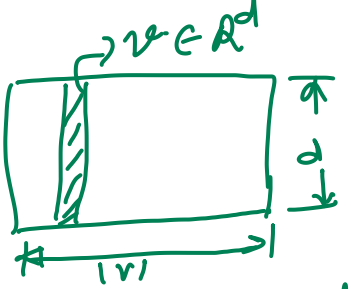
We will study two kind of node embedding:

① Random walk

② Node2Vec

when size of embedding vector is d and suppose we have $|V|$ number of nodes then all will be stacked in single matrix Z

$Z \in \mathbb{R}^{d \times |V|}$



Here each column represent one nodes embedding vector.

Since embedding is lookup in Matrix Z so this is called shallow embedding. When size of graph is large (billions of node) then size of Z becomes issue. Each element of Z is parameter that needs to learn.

Here objective function is to maximize $z_v^T z_v$ as per similarity function for actual network. So decoder is designed based on similarity function.

Similarity function can be defined based on many criterion e.g

Nodes are linked

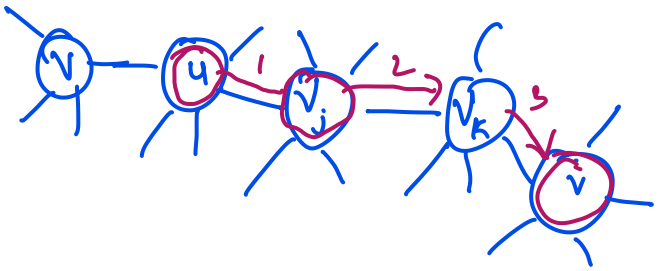
Share neighbours

have similar structural role etc.

Here node embedding is task independent function and we do not use node lable, node feature etc. Our aim is to create node embedding which capture some aspect of network structure by encoder. This embedding can be used for any task.

(Deepwalk) Random Walk Approaches for Node Embeddings

① Random Walk strategy: We need to fix how randomly we are going to walk over graph. One of simple way can be pick randomly some node and choose randomly some next node and keep on walking till predefined fixed length not finished.



Let 3-hop random walk from u to v with red path.

② We need to randomly initialize Z_u vector of length d with real value which will be optimized based on the idea that similar node embedding will have closer value.

③ Probability of reaching node v from u using random walk strategy R will be multiplication of each step as all hop is independent - because its random walk.

$$P(v/Z_u) = P(v_j/Z_u) \times P(v_k/Z_{v_j}) \times P(v/Z_{v_k}) \rightarrow \text{similarity func.}$$

if we take log likelyhood then we can generalize as

$$= \sum \log P(N_R(u)/Z_u)$$

④ In random walk if probability of some node accurance is high means they are closer and considers as similar.

⑤ Using dot product - we can find similarity score

$Z_u^T Z_v$ $\quad \cos\theta \propto P_R(v/Z_u)$

we can use cos θ as encoding function.

⑥ Our objective is to learn d-dimensional embedding vector that preserves the similarity.
Here if nods are nearby then there embedding should be similar.

⑦ Let $G(V,E)$ is graph where we need to define

$$f: u \rightarrow \mathbb{R}^d, \quad f(u) = Z_u$$

⑧ Since nearby nodes should have similar embedding then we need to maximize log-likelihood function

$$\max_f \sum_{u \in V} \log P(N_R(u)/Z_u)$$ where $N_R(u)$ is neighbour nodes as per strategy R.

Here $N_R(u)$ will be multiset- because of strategy R.

(9) Now our problem converted into optimization problem. which can be written as minimization problem.

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|z_u)) \rightarrow \text{for given fixed length.}$$

For all node in $G(V,E) \Rightarrow u \in V$

(10) similarity probability can be defined using softmax function.

$z_u^T z_v \Rightarrow$ Gives similarity score.

Softmax function $\Rightarrow \dfrac{\exp(z^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \Rightarrow$ This must be same as $P_R(v|z_u)$

$$\Rightarrow P_R(v|z_u) = \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$$

$P_R(v|z_u)$: similarity function of actual network

$\dfrac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)}$: node embedding similarity function

(11) Now optimization function be written as

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \Rightarrow \text{Minimization problem.}$$

(12) Solving this naively will be very expensive because of nesting of summation over all nodes. Complexity $O(|V|^2)$ ($u \in V$, $n \in V$ (nested summation two times))

(13) Normalization in softmax $\sum_{n \in V} \exp(z_u^T z_n)$ is culprit. We need to approximate this. This can be done using negative sampling. We do not take all nodes but k-nodes for approximation and instead of exponential function we use sigmoid function (σ) for probability.

$$\log \frac{\exp(z_u^T z_v)}{\sum_{n \in V} \exp(z_u^T z_n)} \approx \log \left[\frac{\sigma(z_u^T z_v)}{\sum_{n \in K} \sigma(z_u^T z_n)} \right]$$

$$= \log \sigma(z_u^T z_v) - \sum_{i=1}^{k} \log(\sigma(z_u^T z_{n_i}))$$

Here n_i is random distribution over nodes.

By using K-sample improves the efficiency of calculation.

- Higher K gives more robust estimate
- Higher K corresponds to higher bias on negative events.
- Sample K negative nodes each with probability proportional to its degree.
- K = 5-20 suggested
- Usualy negative sampling for any node done bcz most efficient way is to take nodes which ~~on the~~ not on the walk.

(14) Once optimation function defined then using gradiant descent we can optimize L wrt z_4

$$\frac{\partial(L)}{\partial(z_4)}$$

$$z_4 \leftarrow z_4 - \eta \frac{\partial L}{\partial z_4}$$ when η is learning rate.

(15) By stochastic gradient-descent method we can train z_4 by optizing L for given training examples.

We choose Random walk because this is expressive due to giving higher probability to closer nodes. other reasion is we do not need to consider all pair node bcz only those node which is participating in random walk.

NODE2VEC

Now question came, can we tune the random walk more so that embedding is more expressive? How to make random walk (DeepWalk) more richer? currently using uniformly distributed random walk which may be too constraint- for downstream task.