Notes on:
Node2Vec: Scalable Feature Learning for Networks

September 2, 2024

1 Abstract

node2vec is an algorithmic framework for learning continuous feature representations for nodes in networks. It learns a mapping of nodes to a low-dimensional space of features that maximises the likelihood of preserving network neighborhoods of nodes.

A biased random walk procedure is presented, exploring diverse neighborhoods more efficient. This is essential for learning richer representation of the neighborhood.

2 Introduction

In a typical node classification task, we are interested in predicting the most probable labels of nodes in a network. Similarly, in link prediction, we wish to predict wheter a pair of nodes in a network should have an edge connecting them.

It is essential to allow for a flexible algorithm that can learn node representations obeying both principles: ability to learn representations that embbed nodes from same network closely together, as well as to learn representations where nodes that similar roles have similar embeddings

Feature representations of indidividual nodes can be extended to apirs of nodes (i.e. edges). In order to generate feature representations of edges, a learned feature representation of the individual nodes using simple binary operators is presented. This compositionality lends node2vec to predict tasks involving nodes as well as edges.

3 Feature learning framework (Method)

The feature learning in networks is formulated as the maximum likelihood optimisation problem. Let $G = (V, E)$ be a given network. Let $f : V \mapsto \mathbb{R}^d$ be the

mapping function from nodes to feature representations.

Where d is a parameter specifying the number of dimensions of the feature representation. f is a matriz of size $|V| \times d$ parameters. For every source node $u \in V$, $N_S(u) \subset V$ is defined as a network neighborhood of node u generated through a neighborhood sampling strategy S.

This paper proceed by expanding the skip-gram archiecture to networks. To optimise the following objective function, which maximises the log-probability of observing a network neighborhood $N_S(u)$ for a node u conditioned on its feature representation, given by f:

$$\max_f \sum_{u \in V} \log \Pr(N_S(u) \mid f(u)) \tag{1}$$

The objective function aims to find a node embedding function f that maximises the likelihood (log-probability) of correctly predicting the neighborhoods of all nodes in the graph.

In order to make the optimisation tractable, there are two standard assumptions:

- Conditional Independence. Factorize the likelihood by assuming that the likelihood of observing a neighborhood node is independent of observing any other neighborhood node given the feature representation.

  $$\Pr(N_S(u) \mid f(u)) = \prod_{n_i \in N_S(u)} \Pr(n_i \mid f(u)) \tag{2}$$

  In other terms, The assumption is that the likelihood of observing any particular node in the neighborhood of a given source node u is independent of observing any other node in that neighborhood given the feature representation f_u of the source node.

- Symmetry in feature space. A source node and a neighborhood node have a symmetric effect over each other in feature space. This is modelled as a softmax likelihood function:

  $$\Pr(n_i \mid f(u) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{u \in V} \exp(f(v) \cdot f(u))} \tag{3}$$

  Here, the numerator represents the similarity (using dot product) between the feature representations (embeddings) of the source node u and a neighborhood node n_i. The denominator is a normalisation factor over all nodes in the graph V.

With those assumptions, the objective Eq. 1 simplifies to:

$$\max_f \sum_{u \in V} \left[-\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \right] \tag{4}$$

The per-node partition function, $Z_u = \sum_{v \in V} \exp(f(u) \cdot f(v))$, is expensive to compute for large networks. To solve this issue, a randomised procedure that samples many different neighborhoods $N_S(u)$.

4 Classic search strategies

In this case the sampling neighborhoods problem is viewed as form of local search. Importantly, to be able to fairly compare different sampling strategies S, the neighborhood N_S is constrained to k nodes and then sample multiple sets for a single node u. Generally there are two extreme sampling strategies for generating neighborhood set(s) N_S of k nodes:

- **Breadth-first Sampling (BFS)** The neighborhood N_S is restricted to nodes which are immediate neighbors of the source.
- **Depth-first Sampling (DFS)** The neighborhood consists of nodes sequentially sampled at increasing distances from the source.

Prediction tasks on nodes in networks often shuttle between two kinds of similarities: homophily and structural equivalence. Under the homophily hypothesis, nodes that are highly interconnected and belong to similar network clusters or communities should be embedded closely together. In contrast, under the structural equivalence hypothesis, nodes that have similar structural roles in the network should be embedded closely together.

In real-world, these equivalence notions are not exclusive, networks commonly exhibit homophily while others reflect structural equivalence. By restricting search to nerby nodes only, BFS achieves structural equivalence characterization (a microscopic view) of the neighborhood of every node The opposite is true for DFS, which can explore larger parts of the network inferring succesfully coimmunities based on homophily (macroview of the neighborhood).

5 node2vec

node2vec was segined with a flexible neighborhood sampling strategy which allows to interpolate between BFS and DFS.

5.1 Random Walks

Given a source node u, a random walk of fixed length l is simulated. Let c_i denote the ith node in the walk, starting with $c_0 = u$. Nodes c_i are generated by the following distribution:

$$P(c_i = z \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

This means $P(c_i = x \mid c_{i-1} = v)$ conditional probability, representing the probability of moving to node x as the next step in the random walk, given the current node is v. π_{vx} is the unnormalised transition probability between nodes v and x. Z is the normalisation constant. It ensures that the total probability of moving to any possible next node x sums to 1.

5.2 Search bias α

The simplest way to bias a random walk would be to sample the next node based on the static edge weights w_{vx} i.e., $\pi_{vx} = w_{vx}$ ($w_{vx} = 1$ for undirected graphs).

In this work a 2^{nd} order random walk with two parameter p and q which guide the walk: Consider a random walk that just traversed edge (t, v) and is in node v. The walk now need to decide on the next step so it evaluates the transition probabilities π_{vx} on edges (v, x) leading from v. The unnormalised transition probability to $\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$, where

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{6}$$

The two parameters p (return parameter) and q (in-out parameter) are introduced to influence how the randome walk explores.

Interpretation of $\alpha_{pq}(t, x)$:

- $d_{tx} = 0$: x is the node the walk came from, and $\alpha_{t,x} = 1/p$. the walk is more likely to backtrack if p is low
- $d_{tx} = 1$: x is a neighbor of v that is neither too close nor too far, so $\alpha_{t,x} = 1$. This is the default unbiased choice.
- $d_{tx} = 2$: x is further from the starting point t, and $\alpha_{t,x} = 1/q$. The walk is more likely to explore outward if q is low.

6 Learning edge features

Node2vec method learns rich feature representation for nodes. However, often we are interested in prediction tasks involving a pair of nodes (an edge). Since node2vec random walks are naturally based on the connectivity structure it is expanded to pairs of nodes using a bootstrapping approach over the feature representation of the individual nodes.

Given two nodes u and v we defnie a binary operator o over the corresponding features $f(u)$ and $f(v)$ in order to generate a representation $g(u, v)$ such that $g : V \times V \mapsto \mathbb{R}^{d'}$ where d' is the representation size for the pair (u, v).