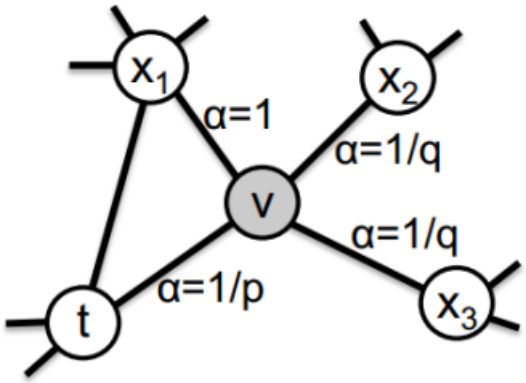
An Example of Node2Vec[1] for the Graph Machine Learning Course[2]

Zahra Taheri[3]

Apr 2023

Node2Vec Algorithm:



In this algorithm, p is the "Return parameter" to determine the probability of returning back to the previous node, and q is "In-out parameter" to determine the probability of moving outwards (DFS) vs. inwards (BFS)).

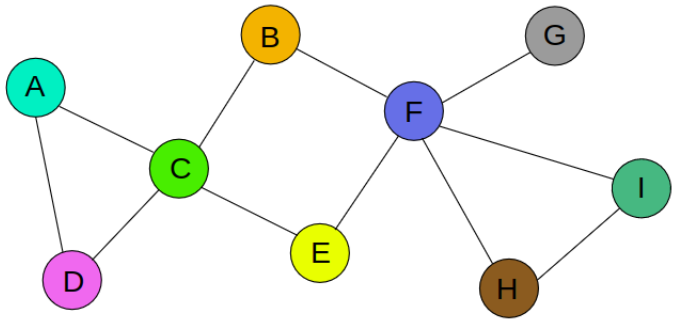
1) Compute random walk probabilities

2) Simulate r random walks of length l starting from each node u

3) Optimize the node2vec objective using Stochastic Gradient Descent (with the same negative sampling objective function that is discussed in DeepWalk[4])

[1] "node2vec: Scalable Feature Learning for Networks" `https://arxiv.org/abs/1607.00653`
[2] `https://github.com/zahta/graph_ml`
[3] `https://github.com/zahta`
[4] "Online Learning of Social Representation" `https://arxiv.org/pdf/1403.6652.pdf`

Example (Random Walks Sampling):



In this example, suppose that p = 200 and q = 1. Therefore,

$\frac{1}{p} = \frac{1}{200} = 0.005$

$\frac{1}{q} = \frac{1}{1} = 1$

(1) Suppose we have started at node D and we want to construct a random walk of length 2 (the length of the random walk is the number of nodes (with repetition) in the random walk except the starting node) based on the node2vec algorithm.

In the first step, we choose to go from D to C (with random uniform distribution). Then, at C we have to decide where to go next. Now, we should compute random walk probabilities. To this end, we have to normalize the transition probabilities as follows:

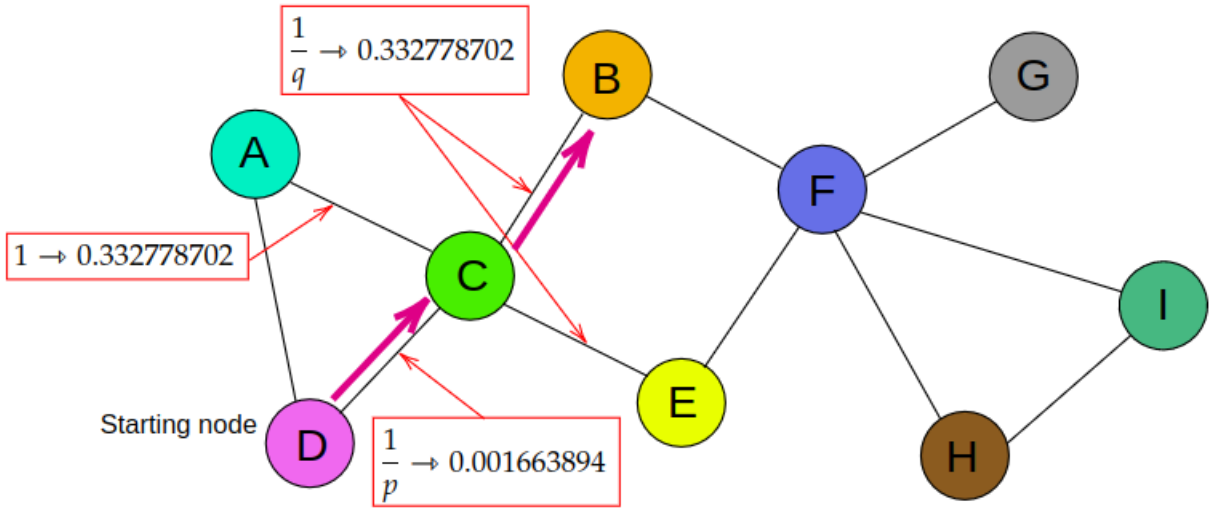
$n = \frac{1}{p} + \frac{1}{q} + \frac{1}{q} + 1 = 3.005$

$\frac{1}{p} \times \frac{1}{n} = 0.001663894$ (return probability - the probability of going from C to D)

$\frac{1}{q} \times \frac{1}{n} = 0.332778702$ (DFS probability - the probability of going from C to B, and the probability of going from C to E))

$1 \times \frac{1}{n} = 0.332778702$ (BFS probability - the probability of going from C to A)

Therefore, $0.001663894 + 0.332778702 + 0.332778702 + 0.332778702 \approx 1$. These probabilities are shown in the graph figure (with red color). Based on these probabilities, we choose to go to the next node (for example, suppose that we choose to go to B. Therefore, the sampled random walk of length 2 starting from the node D is {D,C,B}.

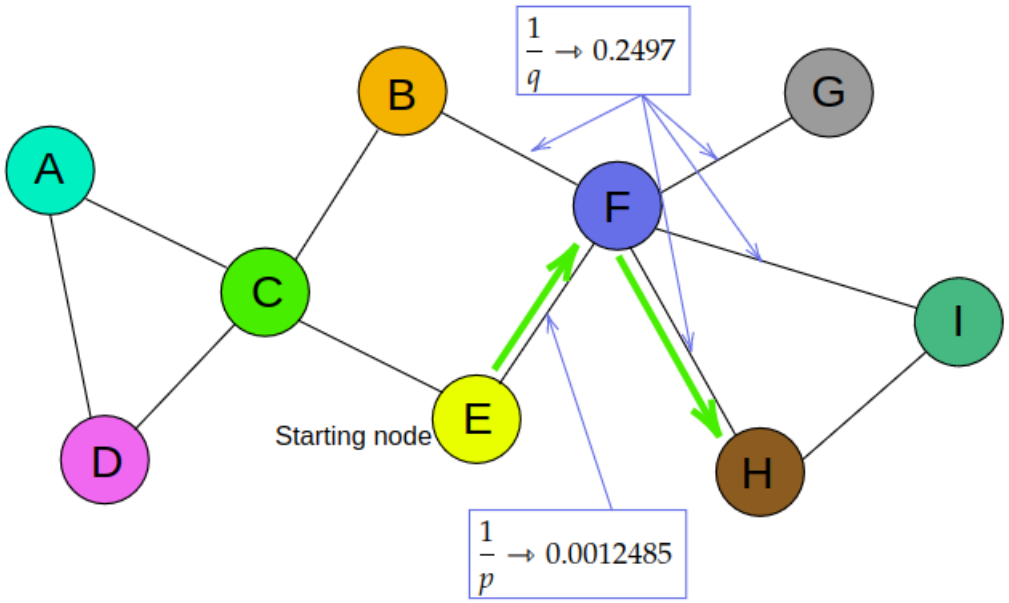


(2) Suppose we have started at node E and we want to construct a random walk of length 2 based on the node2vec algorithm.

In the first step, we choose to go from E to F (with random uniform distribution). Then, at F we have to decide where to go next. Now, we should compute random walk probabilities. To this end, we have to normalize the transition probabilities as follows:

$$
\begin{aligned}
n &= \frac{1}{p} + \frac{1}{q} + \frac{1}{q} + \frac{1}{q} + \frac{1}{q} = 4.005 \\
\frac{1}{p} \times \frac{1}{n} &= 0.0012485 \text{ (return probability)} \\
\frac{1}{q} \times \frac{1}{n} &= 0.2497 \text{ (DFS probability)}
\end{aligned}
$$

Therefore, $0.0012485 + 0.2497 + 0.2497 + 0.2497 + 0.2497 \approx 1$. These probabilities are shown in the graph figure (with blue color). Based on these probabilities, we choose to go to the next node (for example, suppose that we choose to go to H. Therefore, the sampled random walk of length 2 starting from the node E is {E, F, H}.



1. Run **short fixed-length** random walks starting from each node on the graph
2. For each node u collect $N_R(u)$, the multiset of nodes visited on random walks starting from u.
3. Optimize embeddings using Stochastic Gradient Descent:

$$\mathcal{L} = \sum_{u \in V} \sum_{v \in N_R(u)} -\log(P(v|\mathbf{z}_u))$$

We can efficiently approximate this using negative sampling!

An Implementation with PyG:

DeepWalk and node2vec Implementation

Other References to Implement the Node2Vec:

(1) Video: https://www.youtube.com/watch?v=5YOcpI3dB7I

(2) Node2Vec doc: https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.nn.models.Node2Vec.html#torch_geometric.nn.models.Node2Vec

(3) Node2Vec source code: https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/nn/models/node2vec.html