

Social Network Analysis
Prof. Tanmoy Chakraborty
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Chapter - 09

Lecture - 08



Let us discuss another type of algorithms for graph learning and these algorithms are based on random walk techniques right. We know what is random walk, we discussed in the in chapter I think link analysis link prediction chapters.

(Refer Slide Time: 00:39)

I am going to my office

Random Walk Based Methods

- Use random walks to learn the low-dimensional latent embedding of each node
- Captures the local neighborhood and structure of a node by performing enough random walks with a seed node as the starting node
- Output of multiple random walks is combined together and used in an optimization function
- The method is fast
 - Multiple random walks on different seed nodes could be computed in parallel
 - If a change occurs in large real-world networks; only the effected nodes needs to be recomputed
- Popular algorithms from the category:
 - DeepWalk
 - node2vec



So, what is the idea here? The idea here is that we use random walk to capture the context of a node ok. So, what happens, if we have a sentence right in language? Say I am going to my office right and if I ask you to you know collect the set of context words for the word going, you can say that, I am, to my office these are the context words.

In fact, the word going let us say the word going appears in multiple sentences. Therefore, you basically look at two words before going two words after going and you cumulate all the you know words which are which appear around the word going right, but in case of graph right context is very difficult to capture right.

As I mentioned earlier graph does not have any unique structure, if we change the ids of nodes the entire thing will change right. So, what we do here? We use random walk to capture

the context of nodes right. So, it captures the local neighborhood and structure of a node by performing enough random walks. We repeat random walks multiple times in fact, from a given node and capture the context right.


And then why these methods are useful? Because these methods are fast because you can run random walks from multiple nodes simultaneously right. If you if a change occurs in a large network that only affects those nodes around which the changes happened right.

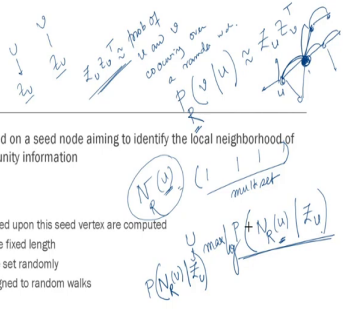
(Refer Slide Time: 02:45)


DeepWalk

- Uses uniform random walks based on a seed node aiming to identify the local neighborhood of the node and capture local community information
- Generating random walk:
 - a seed vertex is sampled
 - a set of multiple random walks based upon this seed vertex are computed
 - random walks are uniform and have fixed length
 - length of the random walk could be set randomly
 - a teleport probability could be assigned to random walks
- Updating the parameters:
 - a skip-gram language model is used to maximize the co-occurrence probability among the nodes
 - the skip-gram probability minimization function:

$$\min_{U, V} -\log \left(\prod_{(u, v) \in \mathcal{E}} P(n_{i+1}, \dots, n_{i+k} | U(n_i)) \right)$$







So, let us first understand what we are doing. Again let me recap right. So, we are coming up for node u and node v , we are coming up with embeddings Z_u, Z_v such that if node u and v are closer in the graph Z_u and Z_v should be also closer right. By the way this part of the lecture is highly motivated by Jure Leskovec course on Deep Learning for Graphs.

So, essentially the dot product $Z_u \cdot Z_v^T$ ok this should be this should mimic the probability of u and v right cooccurring over random walk. Meaning that if you start a random walk from u what is the probability of encountering v , this should be this should be captured using this one right.

So, in other words the probability of probability that given a node u you start your random walk R you encounter v should be approximated by $Z_u \cdot Z_v^T$ ok. So, what we do here? We define a neighborhood right of node u using a random walk R . Let us say you have

a network like this, you start an unbiased random walk from u , you move here move here then move here and let us say the random walk size is 3 right.

So, you encounter this node, this node, this node in the first random walk. You repeat the random walk again, you may encounter again this node then this node and this node you again repeat the same process you encounter say this node then this node right and this node and so on and so forth. So, the length of the random walk is a hyper parameter and the number of random walks that you want to perform is also hyper parameter.

So, for every node you repeat multiple such random walks to collect its neighbors the context and you prepare a set. The set contains the nodes which are the neighbors of which are the neighbors of context of the given node u right and this set is a multi set remember this. Why this is a multi set? Because here an entity can repeat say for example, this node can appear multiple in multiple random walks. So, this entry this node will appear multiple times in the in this set, this is multi set.

In general, when I when you talk about a set the entities should be unique, but in multi set the entities can be repeated ok. So, for every node u I get a neighbor set and what is the target? What is the target that giving this node u let us say the embedding of this node is Z_u right, what is the probability that given this node you will see other nodes all their neighbors?

Probability given the embedding of a node which is unknown what is the probability of its context nodes? This we want to estimate, this we want to maximize, why? Because these are neighbor nodes, the closeness proximity should be higher for the neighbor nodes right given the target node right. So, we want to maximize this. Instead of maximizing this we essentially maximize the log ok.

(Refer Slide Time: 07:44)



$$\begin{aligned}
 & \max_u \sum_v \log \left(\frac{N(v|z_u)}{Z_u} \right) \\
 & \max_u \sum_{v \in N(u)} \log \left(\frac{v|z_u}{Z_u} \right) \\
 & \log \left(\frac{v|z_u}{Z_u} \right) = \frac{\exp(z_u z_v^T)}{\sum_{m \in V} \exp(z_u z_m^T)} \\
 \kappa = 5 \rightarrow \min & = \sum_{u \in V} \sum_{v \in N(u)} - \log \left(\frac{\exp(z_u z_v^T)}{\sum_{m \in V} \exp(z_u z_m^T)} \right) \\
 & \text{Negative S.} \\
 & - \log \left(\frac{\exp(z_u z_v^T)}{\sum_{m \in V} \exp(z_u z_m^T)} \right) = \log \left(\sum_{m \in V} \exp(z_u z_m^T) \right) - \log \left(\exp(z_u z_v^T) \right)
 \end{aligned}$$



So, let me rewrite. We maximize the log of which is log likelihood; maximize the log likelihood given the source node u . You have context nodes right using some random walk right and if we unfold it remember this is a set right $N(u)$ is a multi set right. We can write in this way.

Same as for every u we look at their neighbors right we look at its neighbors log of v given right this is something that you maximize ok. Now how do we calculate this one log of v given Z_u ? So, v is a node, v will also have an embedding Z_v right. So, what they proposed in the deep walk paper? What they proposed is that we can use sort max kind of again operation; exponential of dot product $Z_u Z_v^T$ divided by summation of all nodes exponential of $Z_u Z_m^T$ ok.

So, what is our final objective function? This is max ok, generally we minimize. So, maximize this one is same as minimize the negative of this. So, min right sum of sum over all u present in the graph sum over all of its neighbors minus log of this one exponential of right you must have already understood the problem here ok.

So, the what is the problem here? For every node you have to calculate this one, the same as the one that we discussed in line in grad app right. For every node we need to calculate this sum over all the nodes is quadratic, it takes a lot of time. So, what we do? We take we basically look at negative sampling we apply negative sampling here and then if we apply

negative sampling then this term would be approximated by $\log \frac{x}{y}$ when we unfold it this is $\log x - \log y$.

So, this is $\log x$, \log exponential will cancel out right, but we really want to make it a probability kind of notion. Therefore, we replace exponential by a sigmoid right. There is a paper which basically suggested this one. So, sigmoid minus this part will be there; \log of I basically sample k such nodes from the distribution $P(v|o_k)$. Now we run this over all such k \log of sigmoid $Z_i^T o_k$ where I sample k such negative samples o_k .

So, this is something that we want to and along with this we have what else? No I think this is the final objective function right and we want to minimize it, we have minus here. So, this is deep walk ok. Couple of things to note k number of negative samples; what is the impact? If higher k value your method would be more robust right at what happens is that k approximates all the nodes right.

Higher k also you know corresponds to higher bias towards negative sample. Therefore, we generally assign k to 5 to 20, k 's value would be between 5 to 20. So, this is deep walk ok.

(Refer Slide Time: 13:48)

BFS vs DFS on Graphs

- Breadth First Search (BFS): All the k^{th} proximity neighbors of a node are visited before proceeding to $(k + 1)^{\text{th}}$ proximity neighbors
 - Solid Arrow shows the BFS execution on the example graph
- Depth First Search (DFS): All the neighbors of the currently visiting node in a level are explored before proceeding to the next node from that level
 - Dashed Arrow shows the DFS execution on the example graph



So, the problem in deep walk is that it uses an unbiased random walk right. Sometimes the problem in unbiased random walk is that you tend to capture only local context of a node, you may want to capture the global context as well. For example, say you have this structure

ok, this is u. If you only do random walk it may happen that you will not encounter this node at all or this node at all right.

You would only restrict yourself within this part, but if you think about it this node and this node although they are far their neighborhood structure is very same right. Had it been a you know depth wise random walk you could have also consider this node in your context. So, after deep walk there was another method proposed by Leskovec and team which was called as a node2vec ok.

So, in node2vec the idea is that you bias your random walk to either confine itself within the local part or move to far apart from the node ok. And how do you do that? You do it using BFS and DFS; Breadth First Search and Depth First Search. If you are not aware of what is BFS, DFS, you can go back and look at some of the data structure courses.

So, if you do BFS breadth random walk from this node you should be able to capture these nodes. If you use DFS you should be able to capture these nodes ok. So, we will use BFS and DFS in the random walk process and in the objective function as well ok. So, we captured the local and global context using BFS and DFS respectively and to do so, we define two parameters.

(Refer Slide Time: 16:20)

node2vec

- Expands upon DeepWalk
 - random walks conducted in node2vec is biased
 - bias the second-order random walk by defining two parameters - p and q
- Random walker sourced at node t decides the next node x that it would visit from currently visiting node v on the basis of a transition probability from v to x given by

$$P_{vx} = a_{pv}(t,x) \omega_{v,x}$$

Where $\omega_{v,x}$ is the weight of the edge between v and x, and

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

One is called p right. So, p is the return parameter; p is the return parameter right and you have q which is called in out parameter, I will explain ok. So, what is the idea? Let us say you

have a graph like this ok and this is so, let us say this is u , this is v , w , x right and let us say you start you have already started your random walk from u and you are currently at v .

So, with the parameter based on the parameter p ok, based on the parameter p , you jump from v to u ; so, return parameter, u is the source node. Based on the parameter let us say you have another based on the parameter q , you decide to move further through this path or through this path right and right.

So, what it is saying is now this parameter p and q right from this we basically calculate the probability. So, what is we are saying is that, when we start when we have started our random walk from u and currently at v with probability 1 by p we come here to the seed node with probability 1 by q , we move there. With probability 1 by q we move there and with probability 1 we come here.

Why? Because think about it, v and w both are u 's first of neighbors. So, when we do BFS you should be able to explore both v and w right. Now, if you fix one parameter 1 then then the other would vary accordingly. This is 1 , this is 1 by p ok. So, what does it mean? It means that if you decrease this p value, this probability will increase right; meaning that you will encourage BFS. If you decrease q right this will increase; meaning you encourage DFS right.

So, q controls the in out probability in out movement and p controls the return kind of movement ok. So, this is the so, this is the idea right. Now, so, this is called node2vec . So, in the optimization function also you basically incorporate you make your; so, this is your transition right, this is your transition matrix right and as you see this α is determined by this one ok and w is the weight right.

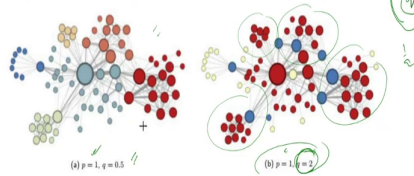
So, using this transition matrix you basically run your random walk process and using the random walk process you will get N R u the neighbors right and then you repeat the same process that we discussed in case of deep walk right. So, only difference is that you have a control on p and q whether you want to do BFS or you want to do DFS right.

(Refer Slide Time: 20:32)

Les Misérables Network: Impact of p and q in node2vec



- Return Parameter (p): determines how far the random node is to explore from the source node. A high value of p indicate that random walker is more likely to walk 'away' from the source node
- In-out Parameter (q): guides the random walker between wither the inward or the outward nodes.
 - If $q < 1$, then the walker is more biased to move away from t (more like DFS)
 - if $q > 1$, the walker is more likely to move inwards or towards t (more like BFS)



Now, look at this figure. So, this is a real world network and you see here that if p equals to 1 and q goes to 0.5 and p equals to 1, q 0.2 q 2 right. So, what happens is that in this setting you are essentially capturing right; you are essentially capturing different clusters because you are allowing you are allowed to move further because of this high because of this because of this low high q value which is low 1 by q right compared to this one right.

So, low 1 by q basically prefers low 1 by q right prefers what? So, if q is high 1 by q would be below and low 1 by q would not prefer DFS right ok. And you see here in this case things like hubs right nodes with high degree those nodes get prominent in this particular figure.

So, this is about node2vec and deep walk. These are very famous graph learning algorithms based on random walk; unsupervised methods, task independent methods right, non-neural network based approaches. So, in the next lecture we will start deep learning based approaches for network embedding. We will discuss different GNN techniques; Graph Neural network techniques, GCN, graph attention network, graph stage and so on ok.

Thank you.