

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

**Chapter - 09**  
**Lecture - 05**

Let us look at the you know taxonomy of the graph representation learning methods right.

(Refer Slide Time: 00:28)



And this is the taxonomy that you know I have come up with and I think this has covered pretty much everything. If you look at the overall you GRL techniques, you can broadly classify into 4 categories, 6 categories. The first one is normal Dimensional Reduction techniques, we have methods which use LDA or PCA kind of approaches to reduce the dimension of the adjacency matrix.

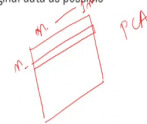
Then we have Matrix Factorization based approach where we use say GraRep write HOPE you know some variation of deep work right graph Laplacian and so on. We have large graph based approaches where we use line right GPNN and so on, there are approaches which use random walk based methods like DeepWalk node2vec there are approaches, which use neural networks like Graph Convolution Network GraphSAGE right Sign Graph Convolution Network and so on.

Then we have other methods which take care of the Sign graph Hypergraph right more of an attention approach and so on Heterogeneous graph and so on. I would not be able to cover each and every algorithm here, but I will try to cover important algorithms that are generally used. For example, I will cover I will briefly cover GraRep, HOPE, LINE, DeepWalk, node2vec and GCN and GraphSAGE ok.

(Refer Slide Time: 02:07)

## GRL Method Categories: Dimensionality Reduction based

- Reduce the dimension of a high-dimensional graph data into low dimension
- Preserve as many properties of the original data as possible
- Extremely General Methods
- Popular methods in this category
  - Principle Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA)



So, I will you know let us try to understand each category one by one. So, the first one is a normal dimensional reduction technique. Where so we have this adjacency matrix right this is  $n$  cross  $n$  and if say  $n$  is of size 1 million then for every node you can think of a vector of size 1 million right. So, now this 1 million size vector is huge right. So, what you can do? You can simply use say PCA kind of techniques principle component analysis kind of techniques or LDA kind of techniques to reduce the dimensionality.

If you are not aware of what is PCA what is LDA please go back and check there are many beautiful materials available online you can look at it and see right.

(Refer Slide Time: 03:04)

## GRL Method Categories: Matrix Factorization based

$A = X \cdot Y^T$   
n x n  
n x p



- Older paradigm of learning graph features
- Adjacency matrix of a graph is a good representative of its connectivity
- Large dimensions of the adjacency matrix restricts its direct use in representation learning
- Factorize adjacency matrix of the graph keeping structure that needs to be highlighted preserved after the factorization
- Provides some key insights on network embedding
- Slower compared to random walk based or neural network based methods



So, the second category is more of a matrix factorization based approach, where what we do we have this adjacency matrix  $A$ , we try to factorize it into low dimension right say this is  $n$  cross  $p$ . So, let us say this is  $n$  cross  $n$ . So, we map it to  $n$  cross  $p$  and  $p$  cross  $n$ , where  $p$  is much lesser than  $n$  right.

And this kind of matrix factorization is used heavily in the recommendation system that we will discuss in the next chapter ok.

(Refer Slide Time: 03:38)

## GRL Method Categories: Random Walk based



- Do not enforce traversing all the nodes in a graph
- Only a small neighborhood of a node in the graph is traversed with the help of random walks
- Performs extremely well on large networks



And then we have Random Walk based approaches, DeepWalk and node2vec there here the idea is that, so here the idea is how do we capture a context of a node right. So, we start a random walk from a particular node and we do k HOPE as say we run the random walk k times.

And during that random walk process we basically sample a set of nodes through which this random walk has basically the random walker has moved right and that defines the context of a particular node right and then we try to predict this context from the given node. So, we will use you know a technique which has had which mostly has been used in the language literature it is called skip gram node2vec skip gram kind of methods.

And then we will see how skip gram kind of method can be useful here for getting the embedding of nodes right.

(Refer Slide Time: 04:39)

## GRL Method Categories: *CNN* Neural Network based



- To design graph representation learning algorithms based on neural networks
- Recently gained popularity due to the rapid rise of computing power
- GPU computing methodologies enable neural network based methods to run efficiently
- Neural network methods specialize in
  - abstracting a lot of details of the problem description, and
  - implicitly representing complex mathematical functions



Then we have Neural Network approaches, where we will see how simple CNN kind of approach convolution neural network approach can be used. In case of graph where we define right we basically define a neighborhood structure and from neighborhood structure.

We aggregate you know representations into a node that then we run some convolution operation and so on and then pass it to the next layer and so on we will discuss this one.

(Refer Slide Time: 05:07)

## GRL Method Categories: Large Graph based LINE



- Large graphs have vast real-life existence
- Several space and time complexity restrictions
- Need to develop more efficient, yet accurate, representation methods

+



And then we have algorithms for large graphs, where we use we discussed line and other methods specifically designed for massive networks right.

(Refer Slide Time: 05:19)

## Matrix Factorisation based GRL Method: Node Proximity Matrix Factorization



- Each node is representation using a  $d$ -dimensional embedding ( $d \ll |V|$ )  $X_{101 \times d}$
- Resultant Matrix  $X \in \mathbb{R}^{|V| \times d}$
- context matrix,  $X^c$ , for the graph is defined on the basis of a property
- Example: If one need encoding neighborhood information, context matrix of a source node is a combined polynomial matrix that contains all the representations of its corresponding neighbors
- Problem statement: Given a higher dimensional matrix  $W$  the aim is to produce a low-dimensional representation  $X \in \mathbb{R}^{|V| \times d}$ , given the context matrix  $X^c$
- Find closeness of matrix  $W$  with representation  $X$  using  $L_2$  norm:

$$\min \|W - X(X^c)^T\|$$

$A_{m \times n}$   $X_{n \times d}$   $B_{m \times d}$



So, let us look at right let us look at the Matrix Factorization based approaches.

Because the first one the dimension deduction approach LDA PCA these are normal and let us look at matrix factorization approaches and so each node is represented is representation using a  $d$ -dimensional vector where  $d$  is much lesser than the number of nodes. So,

essentially we would try to come up with a matrix  $X$  whose dimension is whose dimension would be  $m$  cross  $d$  right.

And so let us take an example. So, if one needs to you know needs encoding neighborhood information context matrix of a source node is a combination combined polynomial matrix that contains all the presentation representation of it is corresponding neighbors ok. Say you have let us say you have an high dimensional matrix  $W$  right. So, the aim is to produce a low dimensional representation  $X$  right given the context matrix  $X$  c.

Now, what is  $X$  c?  $X$  c I mean  $X$  c can capture context of a node, a context can be say a node feature or some sort of aggregated information of the neighbors and so on. So, say let us say  $W$  is  $A$  ok and this is say  $n$  cross  $n$  as I mentioned earlier. So, we will try to come up with a matrix  $X$ , which would be  $n$  cross  $p$  and  $x$  c would be  $p$  cross  $n$   $X$  would also be  $n$  cross  $p$  we take the transpose and this would be  $p$  cross  $n$  right.

(Refer Slide Time: 07:16)

### Matrix Factorisation based GRL Method: Node Proximity Matrix Factorization

□ Singular Value Decomposition (SVD): an approach to obtain  $X$  from  $W$ :

$$W = \sum_{i=1}^m \sigma_i u_i (u_i^T)^T \approx \sum_{i=1}^d \sigma_i u_i (u_i^T)^T$$

$\sigma_i, i = 1, \dots, N$  are the singular values of  $W$  in descending order

$u_i$  and  $u_i^T$ : singular vectors of  $\sigma_i$

□ Optimal embedding:

$$X = [\sqrt{\sigma_1} \cdot u_1^T, \dots, \sqrt{\sigma_d} \cdot u_d^T]$$

$$X^T = [\sqrt{\sigma_1} \cdot v_1^T, \dots, \sqrt{\sigma_d} \cdot v_d^T]$$

*A = USV<sup>T</sup>*  
*with rank m*  
*diagonal matrix*  
*UU<sup>T</sup> = I*  
*VV<sup>T</sup> = I*



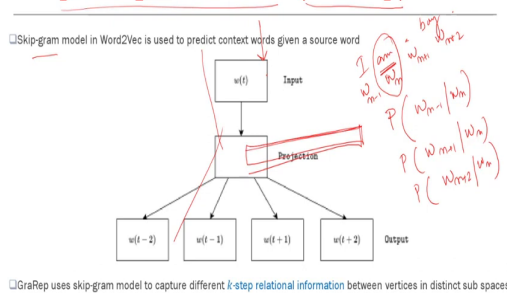
So this Node Proximity Matrix Factorization, what we can do? We can use say a singular value decomposition SVD kind of techniques. And what SVD does? If you are aware of you know SVD singular value decomposition, what it does? It basically decomposes you know it can be a non-regular real non-rectangular real or complex matrix. It basically decomposes say  $A$  real or complex matrix is decomposes  $A$  into 3 matrices  $U S V$  transpose right.  $U$  and  $V$ , so  $U$  is called the Left singular matrix and  $V$  would be Right singular matrix and  $S$  would be the diagonal matrix and  $U$  and  $V$  both of them would be orthogonal.

So, that  $UU^T$  transpose would be identity matrix and  $VV^T$  transpose also be identity matrix right. So, at the end of the day what we will get? We will get we will decompose  $W$  into this way where you see sigma, sigma is basically the diagonal element of  $S$  right and you have  $S$  you have  $U$  and  $U^T$  right left and  $I$  mean  $u_i$  and  $u_i^c$  left and right singular vector and then you have embedding's right.

So, you essentially do you run singular value decomposition you decompose  $W$  into left singular right singular, you take left singular say you left singular vectors and then we get embedding for every vector every node right it is a straightforward approach.

(Refer Slide Time: 09:22)

### Matrix Factorisation based GRL Method: Graph Representation (GraRep)



Now, let us look at let us look at another matrix factorization based approach which is GraRep ok.


So, this is graph representation learning technique and here the idea is that it basically uses the you know the what to wake and the skip-gram approach that is there in natural language. So, in natural language what happens is that say again say I am a boy and you want to capture the representation of  $m$  right. So, and we define a context now let us say the context is 2 words before the word  $m$  and 2 words after the word  $m$  right say this is  $w$  and this is  $w$ . So, this is  $w_{n-2}$  this is  $w_{n-1}$   $n+1$   $w_{n+1}$   $w_{n+2}$  and so on.

So, we will basically capture the probability of  $w_{n-1}$  given  $w_n$ , probability of  $w_{n+1}$  given  $w_n$  probability of  $w_{n+2}$  given  $w_n$  and so on and so forth. So, in a very

simple idea, so you have a normal say a multilayer perception kind of network, your input is the representation of the central node and the output is the representation of the neighborhood node right. And basically the idea is that you try to approximate the, you know embedding of the output embedding of the neighbors, based on the embedding of the central node central word right.


And we basically you know pass it through a projection layer and so on, the way we do multilayer perception kind of technique and we get here an, embedding right and this is low dimensional embedding because you project it into low dimension and this would be the embedding of the central node the central word right this is the core idea.

(Refer Slide Time: 11:40)



## Word2Vec and Skip-gram CBAW

- Word2Vec: a word-representation technique used to represent words as vectors of a given size
- skip-gram:
  - an algorithm used by Word2Vec to construct the vectors
  - to predict the 'context' given an input word
  - to find words in the context so that the probability of the surrounding context is maximized
- The log likelihood:
 
$$\text{maximize } J = \log P(\omega(c-m), \dots, \omega(c-1), \omega(c+1), \dots, \omega(c+m) | \omega(c))$$
- Using Markov Assumption:
 
$$\text{minimize } J = -\log \prod_{j=-m}^m P(\omega(c-m+j) | \omega(c))$$
- In case of graphs, to replace the sequence of words by a sequence of nodes obtained by a random walk




So, and this is called Skip-gram. In skip gram you predict the embedding of the context right, given the central node central word; whereas, there is another approach called C Bau right.

In C Bau method you basically predict the embedding of the central node given the context nodes ok. So, this would be probability of  $w_{n+1}$  given sorry probability of  $w_n$  given  $w_{n+1}$  probability of  $w_n$  given  $w_{n-1}$  and so on and so forth right. So, you see here in case of skip-gram right given the probability of the central word, you predict the probability of the context words. So, there are let us say there are  $m$  context words before  $w$  before  $c$   $m$  context was after  $c$ .

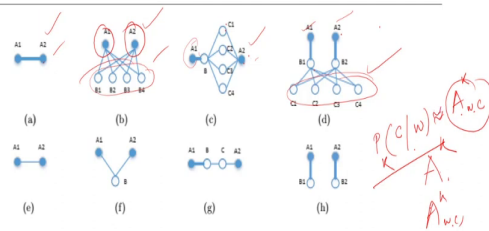


So, you measure the probability log probability this is basically log likelihood right of the context word m context word m context words before c and m context words after c right. So, you can easily write this is log of this one you can easily write it into this bulky multiplication right operation and then we take we move log inside we take the we convert multiplication to summation and then we solve the problem right and this is Skip-gram.

(Refer Slide Time: 13:34)



### Matrix Factorisation based GRL Method: Graph Representation (GraRep)



- Example graphs highlight the importance of capturing  $k$ -step relational information
- An increment in  $k$  value captures an increasing global structural information of the graph relative to a particular node
- Probability of transition from current vector  $w$  to context vector  $c$  in exactly  $k$  steps:  $p_k(c|w) = A_{w,c}^k$



Now, if you look at GraRep right, GraRep exactly do the does the same thing in different manner. So, the idea in graph is that you try to capture different levels of proximity right, say this one 2 nodes are directly linked right therefore they are similar right. If you think of this one 2 nodes are not directly linked, but all their neighbors are same they are also similar. If you look at this A1 and A2 they are not directly linked.

But if you look at the A's 2 hope neighbors and A 1's 2 hope neighbors A 2's 1 hope neighbors they are similar. If you think of this one right again they are they are not directly connected A1 and A2, but they are similar in 2 hopes. In 2 hopes all the neighbors are connected right and so on and so forth right. So, how do we capture? So, if you see so all these things are example graphs highlighting the importance of capturing  $k$  step relational information.

So, an increment in  $k$  now  $k$  is the  $k$ -step relation right this is  $k$  equals to 1  $k$  equals to 2. So, an increment in  $k$  value captures an increasing global structural information of the graph relative to the particular node right. So, probability of transition from current vector  $w$  to

context vector  $c$  right, so these are contexts. So, let us say, let us say this is your current vector and this is your context vector right.

So, probability of a context vector  $c$  given the current vector with respect to  $k$  hope is essentially I mean you basically capture the  $k$  hope distance using the adjacency matrix, this is essentially  $A$  of adjacency matrix  $w$  comma  $c$  entry and you multiply this  $k$  times right. So, you multiply  $k$  you multiply  $A$   $k$  times right and then you check this  $w$  comma  $c$  entry right and that would be your probability. So, you basically capture this is in the probability right not exactly probability, but it basically approximates the probability ok.


(Refer Slide Time: 16:11)

## Graph Representation (GraRep): Positive and Negative Sampling

- Current vector  $\omega$  and context vector  $c$  are connected via a path of maximal path length of  $k$
- Current vector  $\omega$  and negatively sampled context vector  $c'$  are not connected via a path of maximal path length of  $k$
- Negatively sampled context vector at step  $k$  is a context node which is not at a distance of  $k$  from the current vertex
- For a particular  $k$ , the loss function, motivated by the skip-gram model and negative sampling is:

$$L_k(\omega) = \left( \sum_{c \in V} p_k(c|\omega) \cdot \log(\sigma(\omega \cdot c)) \right) + \lambda \sum_{c' \in V} p_k(V) \left[ \log \sigma(-\omega \cdot c') \right] + \dots (*)$$

$p_k(V)$ : distribution over the vertices in the graph  
 $\lambda$ : hyper parameter for the number of negative samples





So, what is the idea here? So, here the idea is that for a particular node  $w$  for which you want to capture the, you want to measure the embedding right and let us say you have a context vector context node  $c$  and the corresponding context vector  $c$ . So, what you do? You maximize the similarity between that word and the context word and you minimize the similarity between that word and the non-context words right.

So, this is the optimization function and  $k$ -hope, so let us say  $k$  equals to 2. So, for a given word what you do here you take a context node  $c$  and this is the probability and this is essentially this one right ok and then you multiply this. Because this is a number of shortest paths between  $c$  and  $w$  right and what is this is? The now this is sigmoid ok non-linearity right ok. So, sigmoid of  $w$  vectors  $w$  vector and  $c$  vector  $w$  vector is the representation of the word  $w$  and this is the representation of the context vector  $c$ .

So, you pass this through a non-linearity right and then through a log, so higher this one that would basically give higher weightage to this one right. So, if  $c$  and  $w$  are similar they have lots of shortest paths of size 2 right, we basically give more weightage to this one right. At the same time we have negative sampling right. So, what we do here  $p$  of  $k$   $v$  is essentially a distribution of vectors distribution of vertices right.

So, we take we sample a vortex  $c$  dash right which presume which we assume to be the non-context vertex and then we pass the same thing using a logic, a sigmoid activation, but this time minus right. So, it means that we give more weightage here and less weightage here, if we maximize this it means we maximize this component and we minimize this component ok. So, this is the idea of GraRep.

(Refer Slide Time: 19:32)

## Graph Representation (GraRep): Positive and Negative Sampling



□ Rewrite equation (\*) as:

$$L_k(w, c) = p_k(c|\omega) \cdot \log(\sigma(\bar{w} \cdot \bar{c})) + \lambda \cdot p_k(c) \cdot \log \sigma(-\bar{w} \cdot \bar{c}) \dots \dots (*)$$

□ Assuming a normal distribution for the probability of selecting  $w'$  as the seed vertex, (\*\*) takes the form:

$$L_k(w, c) = A_{w,c}^k \cdot \log(\sigma(\bar{w} \cdot \bar{c})) + \frac{\lambda}{N} \cdot \sum_{w'} A_{w',c}^k \cdot \log \sigma(-\bar{w} \cdot \bar{c}) \dots \dots (***)$$

□ Setting  $a = (\bar{w} \cdot \bar{c})$  and letting  $\frac{\partial L_k}{\partial a} = 0$ , we get

$$\bar{w} \cdot \bar{c} = \log \left( \frac{A_{w,c}^k}{\sum_{w'} A_{w',c}^k} \right) - \log \left( \frac{\lambda}{N} \right)$$



(Refer Slide Time: 19:47)

## Graph Representation (GraRep): Algorithm



□ We need to factorize matrix  $X$  into two matrices  $W$  and  $C$  such that

$$Y_{n_1, n_2}^k = W_{n_1}^k \cdot C_{n_2}^k = \log \left( \frac{A_{n_1, n_2}^k}{\sum_n A_{n, n_2}^k} \right) - \log \left( \frac{\lambda}{N} \right)$$

\*Step 1: Compute  $k$ -step transition probability matrix  $A^k$  for  $k = 1, \dots, K$ , the maximal path length of the graph

\*Step 2: Compute  $k$ -step log probability matrix  $X^k$  and subtract by the normalized constant hyperparameter  $\lambda$ . Replace negative entries by 0

\*Step 3: Apply SVD to get the final representation vector



So and then we do a lot of approximations I am not going into details of this and then we take the derivative because this is the objective function, we take the derivative with respect to  $A$ , then we update the values using normal gradient descent kind of approaches right. So, we stop here in the next lecture we will discuss the other approaches.

Thank you.