

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

**Chapter - 09**

**Lecture - 11**

Alright. Let us now discuss the last concept of this chapter which is graph attention network, right.

(Refer Slide Time: 00:30)

**Variations of GCN: Graph Attention Network**

- In many cases, it is better to give more weight to, say, more "influential" nodes
- Importance is often called as attention
- Not possible with vanilla GCN
- In a plain GCN, we give equal weight to all the neighbors of any given node
- The revised formulation:

$$h_v^k = \sum_{u \in N(v)} \alpha_{uv}^k W^{k-1} \frac{h_u^{k-1}}{|N(v)|} + h_v^{k-1}$$

Additional Challenge: Learning the attention  $\alpha$

NPTEL

GA, it is also called GAT or whatever GA. GAN is not an right analogy, because we have generative adversarial network which is also abbreviated as GAN. So, GAT is the right term ok. So, what is attention? I will not explain what is attention, but I think it has already been explained by my TA right.

But basically in a nutshell the idea is that attention is a mechanism by which you can pinpoint that look I want to give more weightage to this entity rather than this entity for you know deciding certain things, right. For example, let us say language right, and let us say I saw the movie right yesterday and it was awesome ok and let us say the task is movie review analysis, sentiment analysis of movie reviews.

If you think of this one, this review right you want to give more weightage to the term awesome right and less weightage to terms like I saw, these are generic words right. The term

awesome plays an important role in the deciding whether this is the positive review or the negative review right.

Let us say another example, let us say you know say I mean you take any example, let us say machine translation right. You translate one sentence from one language to another language right. Sometimes you want to you want to give more emphasis to certain phrases or certain words right, which may you know may bring additional importance to the task.

Here also in case of graphs right, a graph stage or GCN normal GCN paradigm, when you aggregate right. So, so far how we have how we have been aggregating? So, if you remember the equation  $h_v^k$  is sigmoid let us say GCN  $w_k$  summation of  $h_u$  right,  $u$  is the neighbor of  $v$   $k$  minus 1  $N_v$  plus  $B_k h_v^k$  minus 1 ok.

If you think of this aggregation operation right, this is essentially mean and the weight of every component is  $1/N_v$  right; should be mod because it is a set ok. And this  $1/N_v$  is the weight for every neighbor. So, what you are saying is that for this neighbor right, when I basically take the messages from neighbors all these messages are equally important, right.

So, but sometimes what happens is that some messages are more important than other messages, right. For example, this sentence right, this word is more important the other words for some tasks for example. So, you also want to give weight corresponding to the edges through which these nodes are the these messages are coming in ok.

You may say that ok, let us use the weight of the edge. If the weight of the edge is available let us use it for aggregating messages right rather than learning the weight right. Let us say this the weight this is the weighted graph and let us say the weight is  $w_1, w_2, w_3, 3$  weights. So, this would be for this one  $w_1$  by  $w_1 w_2 w_3$  times this one plus  $w_2$  by  $w_1 w_2 w_3$  time this one and so on and so forth.

But for an unweighted graph how do you get this one, right? A second point is that remember this weight was created, weight was assigned based on certain objective right. Let us say the I mean the objective was let us say the weight was derived based on the number of interactions right. But let us say the that the task is you know fraud versus genuine node classification.


So, these weights may not be the desired weights for this task. So, you may want to learn the weights depending upon the tasks right you want to give more attention to say this neighbor

rather than these neighbors or these neighbors, right. So, therefore, we will introduce another component which is alpha, right. Now, this alpha is the weight, it is also called attention right which we will learn ok.

If you are not aware of what is attention, you should look at the seminal paper attention you know the seminal paper by Vaswani right by Google Brain on Attention Network right. Attention network it was introduced mostly for languages, but then people basically started using it for other task.

(Refer Slide Time: 06:25)

## Variations of GCN: Graph Attention Network



- In many cases, it is better to give more weight to, say, more "influential" nodes
- Importance is often called as *attention*
- not possible with vanilla GCN
- In a plain GCN, we give equal weight to all the neighbors of any given node
- The revised formulation:

$$h_v^k = \sum_{u \in N(v)} \alpha_{uv}^k W^{k-1} \frac{h_u^{k-1}}{|N(v)|} + b_v^k$$


- Additional Challenge: Learning the attention  $\alpha$

Attention is all you need

GAT  
GAT

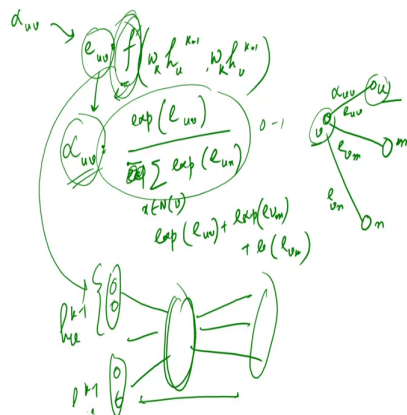
I saw the man  
it was (wasn't)

$h_v^k = \sum_{u \in N(v)} \alpha_{uv}^k W^{k-1} \frac{h_u^{k-1}}{|N(v)|} + b_v^k$



So, the name is attention. Attention is all you need. This is the famous paper by Vaswani, Google Brain 2017, if I not mistaken. So, let us look at graph you know attention network ok. So, what is the idea here?

(Refer Slide Time: 07:02)



The idea here is that we essentially you know learn the attention weight right, alpha u v ok alpha u v. Let us say this is v, the node for which you want to get the embedding and this is one of its neighbors u right. And this edge weight is alpha u v that you want to learn, this is attention, right.

So, this we will learn through though the embeddings of v and embeddings of u right. Let us assume that let us denote another variable e u v right. Now, e u v is basically you know it is which is basically a function f of h of u k minus 1 and h of v k minus 1, right.

And of course, weights say w k, this function is something that we will discuss ok. You can take a set of functions ok, but this function will. So, what this basically does? It takes the embedding of this node and embedding of this node at the previous time stamp, k minus 1 pass it through this w k, which we also want to learn right and generate some number which is e u v, right.

Now, from e u v then you derive your alpha. So, now, remember for v you have e u v here, you also have say this is the this is say m e v m here, this is n e v n here. The alpha u v would be you just you know take this thing, you pass it through some sort max, it would be exponential of e u v by you know sum of exponential of e u x right, where x is the neighbor of v right. In this case this would be exp of e u v plus exp of e v n plus exp of e v m ok.

Now, sigmoid you when you pass it to sort max, it always guarantees that it ranges between 0 to 1 ok and this is your alpha, alright. So, you may wonder what types of functions we can use here, you can use many things. For example, you can use dot products right or in fact, you can use the neural network here right.

What you say is that ok, I have right neurons ok, this would be h of u k minus 1, this would be h of v k minus 1, pass it through a neuron and neural network and let it produce something, right. So, this is also parameterized. So, this the weight of the neuron will also be learned through back propagation, right.

And I mean you can use whole bunch of things right. I mean you can play with whole bunch of things here and accordingly you generate number. You pass it through sort max and that will produce something and this alpha right, this alpha will later be used right here as you see here ok in this part right.

(Refer Slide Time: 11:22)

## Variations of GCN: Graph Attention Network



- In many cases, it is better to give more weight to, say, more "influential" nodes
  - Importance is often called as *attention*
  - not possible with vanilla GCN
  - In a plain GCN, we give *equal weight* to all the neighbors of any given node
- The revised formulation:

$$h_v^k = \sum_{u \in N(v)} \left( \alpha_{uv}^{k-1} W^{k-1} \frac{h_u^{k-1}}{|N(v)|} + h_v^{k-1} \right)$$

$$\alpha_{uv}^k = \sigma \left( w_k \sum_{u \in N(v)} \alpha_{uv}^{k-1} w^{k-1} \right)$$

- Additional Challenge: [Learning the attention  \$\alpha\$](#)



So, then your hidden state at k for node v would be sigmoid of w k right, this one alpha right ok, k minus 1 w k minus 1 whatever.

(Refer Slide Time: 11:56)

## Variations of GCN: Graph Attention Network

- In many cases, it is better to give more weight to, say, more "influential" nodes
- Importance is often called as attention
- not possible with vanilla GCN
- In a plain GCN, we give equal weight to all the neighbors of any given node

□ The revised formulation:

$$h_v^k = \sum_{u \in N(v)} \alpha_{uv}^{k-1} W^{k-1} h_u^{k-1} + h_v^{k-1}$$

□ Additional Challenge: Learning the attention  $\alpha$

Transformer  
Graph Transformer  
$$h_v^k = \alpha \left( W^k \sum_{u \in N(v)} \frac{h_u^{k-1}}{|N(u)|} \right) + h_v^{k-1}$$
  
NeurIPS, ICML, ICLR, SIG KDD  
WWW, AAAI



I mean you just you can you know at this part, you can just say that ok, this is  $h$  of  $u$   $k$  minus 1, this is normal aggregation right. And now this is the weighted part, earlier it was 1 by  $n$   $v$  just right and then you have another part here which parameter is to be  $B$  right ok. So, now this is graph attention network, right.

In fact, if you look at the literature these days there is this famous architecture called transformer which is also introduced, I mean in this paper attention is what you need right. The transformers transformer model was proposed for mostly for languages and later then and after that people also used this for graph, for you know for vision related issue problems images and computer vision problems.

In fact, you can also you may also have heard of something called graph transformer network right and this field is ever evolving. So, I mean whenever you look at recent papers published in say NeurIPS, ICML, ICLR right, KDD, these kind of venues you always see you know multiple papers on graph neural network in general ok. So, in short what I can say is that, this area is the cutting edge area and that is evolving so fast that even a single lecture will not be able to cover of everything.

So, I suggest that instead of just focusing on a few papers on right existing blogs and so on you just keep track of the papers published in graph learning venues, deep learning, ML kind of venues. Particularly NeurIPS, ICML, ICLR right, SIG KDD, WWW ok, triple AI these venues, you will see a lot of you know fundamental approaches of for graph neural network

and of also a lots of applications for you know different applications using graph neural network in general ok.

So, with this I stop here and this brings us to the end of this chapter. So, and I think this has been one of the longest chapter so far, right. So, you have learned we have learned plenty of things right from traditional approaches for graph learning, matrix attrition based approaches.

Then we moved into random work-based approaches, then we understood the problems of random based approaches, particularly in the inductive versus transactive setup. And then we looked at a few examples of graph neural networks, right. So, we stop here and the next chapter which is going to be the last chapter, there we will discuss different applications of I mean graph learning in general whatever we have discussed so far particularly we will see the applications of graph neural network, ok.

Thank you.