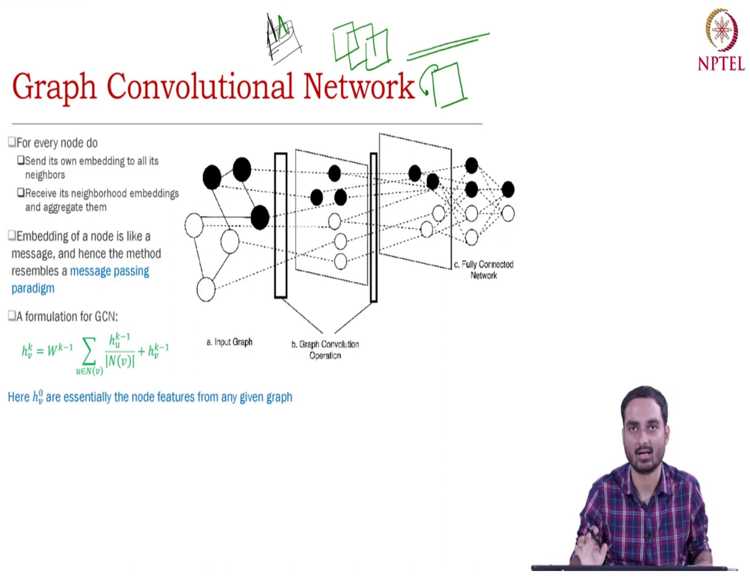
Social Network Analysis
Prof. Tanmoy Chakraborty
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Chapter - 09
Lecture - 09

Hi everyone, let us now look at the last part of this chapter which is a Graph Neural Network.

(Refer Slide Time: 00:30)



Graph Convolutional Network

- For every node do
  - Send its own embedding to all its neighbors
  - Receive its neighborhood embeddings and aggregate them
- Embedding of a node is like a message, and hence the method resembles a message passing paradigm
- A formulation for GCN:

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

Here h_v^0 are essentially the node features from any given graph

So, far you know the methods that we have discussed right for example, node to wake, deep work kind of methods, where we look at random work for embedding, we also have looked at matrix actuation based approaches. So, all these approaches are kind of statistical approaches in the sense like it basically tries to minimize or maximize some object objective function without considering you know explicit neural network structure as such right.

So, now what we will discuss today is graph neural network where we will see how we can incorporate the you know the vast amount of neural network architecture that have been proposed so far for different other domains like computer vision, speech, right those domains and how we can incorporate those for graph processing graph embedding.

So, neural network deep learning we all know you know the impact of deep learning in today's world you know I would say post 2010 2011 right era we have seen multiple such cases where deep learning based approaches just outperformed statistical approaches in most of the domains. And graph domain is also you know one such domain where you know deep learning completely overshoot most of the traditional methods.

I am not going into the details of deep learning the key of this course, Shivani has already talked about deep learning the foundation of deep learning in the previous lectures. I think she has already talked about you know normal neural network, feed forward network, RNN, CNN, attention, this kind of methods. Hope, you have a fair bit of ideas about this I am assuming that you guys are aware of all these techniques at least briefly.

So, I will try to explain graph neural network again in a very abstract manner I will not go into the details of every bit of this because if I go into the details of every bit of this part this would this would take another course separately because the way this particular domain is evolving right after 2017 2018 you know years. So, after that people have seen plenty of methods where you know this kind of neural networks have been incorporated for graph processing ok.

So, what is the fundamental ideas behind, I mean what is the fundamental idea behind graph neural network? You basically have an image right and let us assume that you have an image right and for image domain how we generally you know extract information extract features we essentially use some sort of convolution operation.

We have fixed grid and that grid moves over the image and you basically get a condensed representation of every grid right from every grid and then you basically squeeze the representation and you keep on squeezing it and finally, you get you know some sort of application right, you feed the that representation to the application.

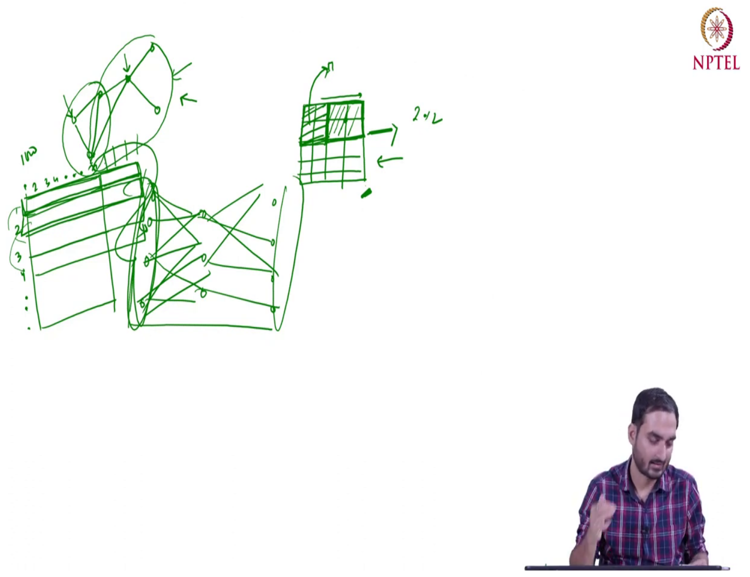
So, now one may wonder that let us use GCN let us use convolution operation on graphs right, but the problem in graphs is that unlike image right which basically uses a RGB kind of you know matrices for representation or speech which is basically sequence sequential in nature right, where you can quite easily do convolution operation because you have a fixed structure and if you manipulate if you modify the grid for example.

If you modify the sequence you get a separate representation altogether right. So, your image will change altogether, speech signal speech sequence will change altogether, but in our case when you have an adjacency matrix and if you swap rows and columns we discussed earlier if you swap rows and columns we basically rename nodes right. So, the network structure will remain same, but the adjacency matrix will become now different right.

Now, on a different metrics if you do the same operation you may I mean you expect that you get different output, but the graph structure remains same, right. So, you should not get different output. So, that is the major concern here that is why traditionally the way we you know use computational operation does not work in case of graphs right.

So, the rest of the part that we are going to discuss today, that part is highly motivated by Euros Cubics course on machine learning for graphs as well as the beautiful the book right on deep learning for 'Deep Learning on Graphs' by Jiliang Tang ok. So, if you are interested you can also look at this book and the lecture series you get fair bit of ideas about graph learning using neural network. So, in case of graph right how do we; how do we think of a composition operation ok.

(Refer Slide Time: 06:18)



So, let us draw a picture here. So, let us assume that right. So, what a convolution operation does? If you think of it carefully you have an image right, let us think of it in case of image and you have this matrix right. Now, you have a kernel which keeps on moving right and from each particular region right you get a representation right. What you basically do, you do some convolution operation and you get a condensed representation of this part here right.

So, let us say you have an 8 cross 8 matrix here from 8 cross 8 matrix you possibly get a say a 2 cross 2 matrix right. Now, the good part about image is that as I mentioned is the reference point here is very clear. For example when you think of this region right you know that the central point is this one this is the central point and with respect to the center point you can think of a 2 by 2 cross 2 region right, but think of a graph right.

In case of graph how do you; how do you decide the central point, what do you mean by a central point right and another point and another interesting feature in case of image is that you know that this kernel right the dimension is always same right and it focuses on a 2 cross 2 region right and that region is very you know that region is fixed right.

But, in case of graph so, let us say you think of these as a central point you can think of these as a central point, you can think of this as a region one of say egonet right, as a region. You can also think of now if you assume this as a central node, you have 1 2 3 4 neighbors, what about this one if you consider this as a central point, then you have these as a region right where you have two nodes right you have two nodes in this case as neighbors in this region, but here you have four nodes right as neighbors in the region.

Now, the region is not same what I mean to say in case of graphs the region is not same. So, how can we you know tactfully use convolution operation here in case of graph we will discuss this, but another point to ponder is that we also want that the topological structure as well as the feature of nodes would also be preserved would also be considered while doing neural network while running neural network right.

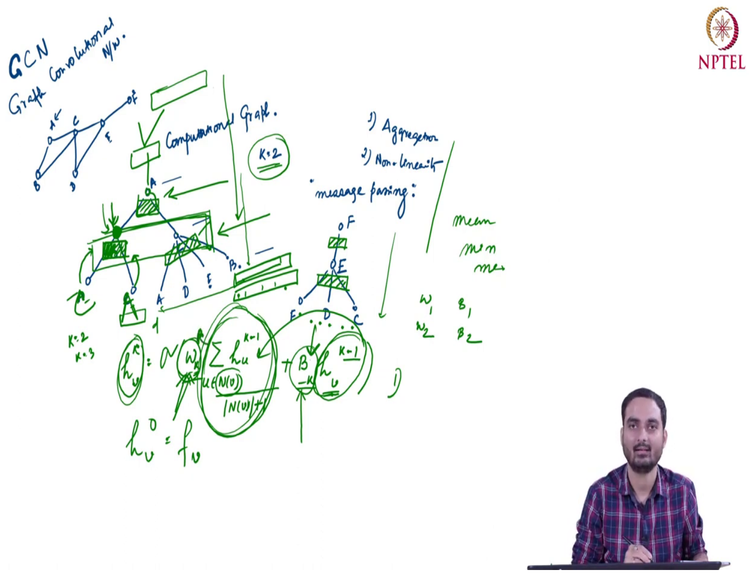
So, I mean one can think of you know one can think of a normal feed say forward network where you have say a two layer network right like this a two layer network and you can say that ok, let us think of the adjacency matrix here and say 1 2 3 4 n, 1 2 3 4 n you have rows indicating you know neighbors, you have you can also add additional rows indicating features right for each node and then you have a concatenated version of feature vector right.

This spot indicates neighbors and this part indicates features right and you can say that what I can do, I can just feed this right in the neural network and let the neural network you know produce some representation similarly you feed this to the neural network and let it produce something and so on and so forth right. But this is a terrible idea, why this is a terrible idea?

The reason is that the reason is that first of all let us say you have a graph of 100 nodes right. So, you have a neural network of 100 dimension right, now let us say you want to use the same neural network for another graph right of size 50, now how can you use this one? You cannot use it, because you have now 1000 or 100 nodes whatever right, how can you fill the remaining part here, this is one problem and the second problem already mentioned this is this should be; this should be order variant, but here this is order invariant right.

If you change the order what would happen is that if you swap nodes and columns tactfully you may I mean you will end up getting the same graph, but the neural network will would not be able to understand whether this is the same graph or different graph ok. So, how do we tackle this problem?

(Refer Slide Time: 11:55)



So, in case of graph right, GCN graph convolutional network so, in a normal convolution operation the CNN kind of you know tool when we use what are the two basic operations that we do. So, one is the aggregation. So, we basically fix a particular region put a kernel aggregate information from that region. And then non-linearity right we that aggregated information is passed through a non-linearity and it produces something. So, here also we will do the same thing in case of graphs right, but in a different manner.

So, what we do here, the neural network that we create right would be would basically driven by the topological structure of the graph ok. Let us say; let us say you have a graph like this say A B C D E F for every node we will first create something called computational graph ok. Now, remember this computational graph would be different for different nodes ok.

Let us say; let us say we fix that the depth of the computational graph would be 2 right. So, we will not move beyond second half right, now for node A; for node A so, node A will basically receive information from B and C right if you look at; if you look at right it is immediate neighbors. So, this is A, B and C if you now if you look at B, B is one of the neighbors right would be A and C, C is one of neighbors would be A, D, E and B ok.

So, this is the 2 hop or 2 label computational graph for A right. Now, remember what we do here, there is something called message passing setting right message passing. So, in a normal message passing setting what we do, we basically pass messages from the neighbors to the original node. So, say you have v and v's neighbors. So, first you basically pass messages from v's neighbors to v, now v will then further pass that message to its neighbors and so on and so forth right.

Similarly, here we use the same idea. So, message passing setting message passing paradigm would be used for graph convolutional network. Now, so, this is the computational graph for A, but the computational graph for let us say F right would be different, for F it would be E then F D C this is more skewed right, similarly the computational graph for C B D other nodes would be different.

Now, what we do once we create these computational graph right we will basically add a convolutional block right here at every junction here, here, here, in this case here and here. So, this is these are convolutional blocks right, what it does each of these computational blocks what it does, it basically takes inputs from the corresponding nodes and aggregate it right pass it to some non-linearity and then generate the output and the output will be passed to the next layer right.

So, this is the idea, but remember; this computational graph would be different from different nodes right therefore, I said that you know the neural network that we design. So, this is a the this is a neural network of depth 2 neural network of depth 2. Now, K equals to 2 this depth is an hyper parameter that we needed to set and now let us look at individual region let us look at let us look at each of these blocks right what it does right.

So, it basically you know does the following. So, let us say this one right. So, it basically aggregates information messages passing from A and C coming from A and C it first aggregates right and then it basically passes it through pass it basically passes this aggregated information through some weights again some neural network you can think of it as a one I mean depth 1 neural network.

And it also aggregates its own information. So, message. So, every node has some message at particular time I mean at particular iteration right. Let us say at; let us say at K equals to 2 this node has a message, this node has a message, this node also has a message right, this node also has a message.

So, the updated message at K equals to 3 in this node would be a function of these two nodes; these two nodes messages and the message that is there in this node ok. So, let me write what I exactly meant to say. Now, this message is nothing but representation right. So, at every step your representation is getting updated right.

So, representation denoted by each of a particular node v at a time K right would look like this. So, you have some aggregated function, aggregation function let us assume that the aggregation function is simple mean right. So, it would be summation of the representation right all v's neighbors v's neighbors are A and C.

So, u which is a neighbor of phi right u K minus 1 right and how many such; how many such this one how many such neighbors are there right. So, right this would be the mean right. So, this is the main aggregation right of v's neighbors right and you basically you have a parameter W K with this.

And you have another information coming from this node which is h v K minus 1 right this would also be parameterized B K. Now [FL] what is this? This is the message of this particular node at K minus 1th time and you are planning to get the message for v at Kth time ok. So, this is the aggregation ok, these are the two parameters that you learn these are basically neural network parameters and then you pass it through a non-linearity sigmoid kind of non-linearity right.

So, at every block at every such box it basically takes its neighbors aggregates and then it is parameterized similarly you have your own message parameterized, aggregate them, pass it through nonlinearity and get the output ok. So, this happens at every stage ok. So, this W and

B right these are the parameters that we need to learn think about it. So, think about it. Now this W and B right it also depends on the depth K time K if there are two depths.

So, you will have W 1 B 1 for this depth and W 2 B 2 for this depth right and these are trainable parameters. Remember, these parameters W and B these parameters are shared across nodes these parameters are shared across nodes. W K the value of W K B K the same value would be here and here because the dimension is same remember ultimately you are aggregating. So, the dimension would be the dimension of the original message D after aggregation is not concatenation aggregation.

So, the dimension will remain same ok. So, this is the idea. Now, a few points to mention here, this is the hyper parameter that we set generally it is set to 2 it can be 3 also, but not very I mean very high because if you keep if you make it very high then you exhaust with all the nodes right with the entire graph which we do not want.

Secondly, this aggregation operation right should be this should be order invariant, in the sense like if I make this as C and this as A the aggregated output will remain same right. It should not be dependent on the order, sum is an order invariant aggregation function mean is also an order invariant right mean, max, these are all order invariant operations right. So, you can use all sorts of aggregation operation for I mean for here for this part right.

You may also wonder why do we need these two separate parameters why are you not just adding this part here and dividing it by N v plus 1 right. This can also be done, but remember the impact of my neighbors should always be different from impact of my own message right. I want to keep these two things separate I always sometimes I want that my own message would be of more importance than my neighbors message right, therefore, I would want to give more weightage here and less weightage here right.

So, now, how do you initialize this neural network? So, I mean how do we initialize these messages, what do you mean by message here? Message is basically I mean if you have a feature right node feature you can initialize. So, this h is basically hidden state right, what would be h v at 0? At the initial stage it would just be the feature of node v right.

So, you are basically passing the features from one hop to another hop right and this is the idea and if you do not have features then you can just use some sort of node level information like degree or clustering coefficient these information can also be used as a proxy of features right. Or you just set everything as one right and you basically pass it and let the neural network decide because ultimately it also depends on the topology right.

So, and, how do we train this model? There are two ways to train this model, first way is a simple unsupervised way for example, node to wake, deep walk type ways we can train the model. The second approach is more of a supervised approach let us say the task is node classification right.

So, after this layer you can think of a classification layer right which will say whether it is say fraud or genuine or whatever red and blue right and you have say a loss function right, you basically take the derivative of this loss function and back propagate it right and you update the representations ok.

So, this is the idea and this is the this is vanilla GCN, on top of this you can also add additional constraints like the relations and so on and so forth and depending on that methods like RGCN all these methods have been proposed you know in many literature and you can also look at those things right. For example, RGCN and you can also consider edge types the relations right particularly for knowledge graphs right.

And you can have embeddings I mean you can think of this neighbor as not a raw neighbor, but neighbors are now dependent on the relations right and you can have this learnable parameters which would be a function of the relation R right and so on and so forth. So, in the next lecture we will discuss two other methods which is graph sage and another would be graph of attention network.

Thank you.