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

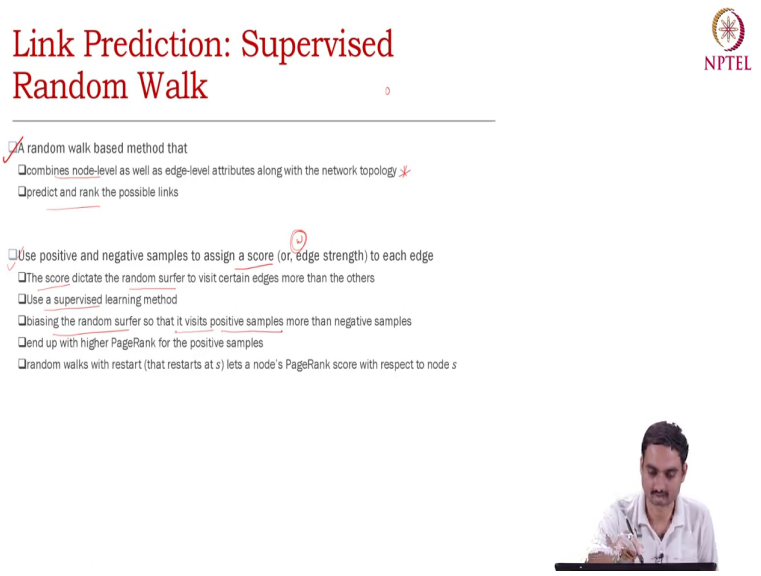
Chapter - 06
Lecture - 05

Ok. So we are almost at the end of this chapter on link prediction and today we will discuss a beautiful algorithm called supervised random walk and supervised random walk algorithm although it was proposed in 2011, but this is you know by far one of the method that people use extensively for link prediction, and the beauty about this method the unique point about this method is that it actually you know captures the topological structure of a network as well as the meta information of nodes and edges ok.

So, so far the methods that we have you know discussed we have observed that mostly we look at the topological structure right, we look at you know whether they are connected in 2 hops, 3 hops whether they have common neighbors and how these neighbors are distributed we look at you know we basically try to understand, try to unfold the underlying hierarchical structure of a network and so on.

But, this was to the best of my knowledge this was the first algorithm which combined the idea of you know random walk based methods for link prediction and how you know one can use the meta information node level information, edge level information right into the random walk process for link prediction ok.

(Refer Slide Time: 01:51)



So, and this method was proposed by Jure Leskovec and his team in 2011 2012 right during that time.

So, what is random walk? We all know what is random walk basically we a random walker starts from a particular node and the walker essentially you know chooses one of the outgoing nodes uniformly at random and then you know follows the particular follows that edge and then basically moves to the next node right. So, you know by default random walk is random because you know we are not making it bias.

But in this particular algorithm it they suggested that you know how we can make the algorithm how can we make the random work process biased ok, and what do you mean by bias. So, I actually you know when this random walk process will go on we would essentially we would insist the random walker to move through those paths right which we want them to move right. So, we already fix a preference and we allow the random worker to basically follow most of the times follow those paths ok.

So, a random walk based method which essentially combines node level as well as edge level attributes along with the network topology. This is the unique selling point of supervised random walk and then it basically predicts and ranks you know all possible links which are which are going to form ok. So, the use of positive and negative samples to assign a score right or edge strength to each edge. So, what it would do, we will discuss it would you know over time it would assign a strength of an edge or strength of a node pair ok.

Now, the strength of a of an edge I mean one may say that ok the strength of an edge can easily be; easily be obtained from the weight right the weight of an edge right, but most of the networks are random weighted right. So, we will see how we can use node attributes and edge attributes to measure the strength of a node pair. The node pair can have an edge right in between them or a node pair may not have edge.

But we can; predict the strength of these two; these two nodes ok and remember the score that we are going to compute this edge strength this is a parametric this is a parametric function ok. This is parameterized by w I mean a set of parameters which we will discuss therefore, and these parameters will be learned over time ok. So, if this is not cleared this moment just hold on we will discuss elaborately in the subsequent slides.

So, the score this edge strength score this basically dictates the random walker to visit certain edges more often than the others ok therefore, I mentioned that this is basically a biased random walk we bias the random walk process.

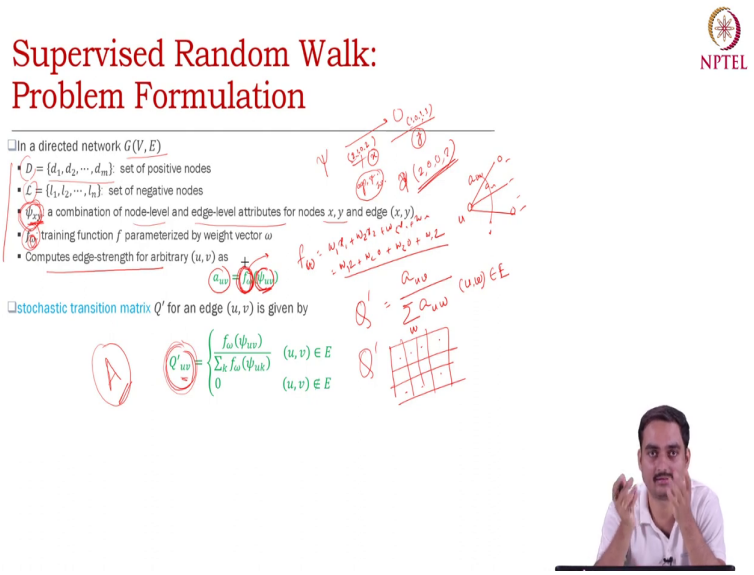
So, that the worker essentially moves through those edges which have high edge strength ok and then we use some sort of supervised learning paradigm to learn these weights and we will see that how eventually we would be able to and when we learn this weight right naturally we can measure the strength of a node pair now if we measure the strength of a node pair we can easily identify those node pairs which have high strength and essentially you can say that look possibly these two node pairs, these types of node pairs with high strength they would form edges in the near future ok.

So, it is biased random walk. So, that it visits positive samples more often than the negative samples what do you mean by positive samples.

So, positive samples are those nodes which are already connected to a given node say for example, you are standing on a particular node and you know that this node u is connected to w v x right, but it is not connected to a b c for example. So, for the random walk the this you know. So, u w x v w x these nodes are basically positive nodes and other nodes are negative nodes ok. So, we bias the random walk to move to basically walk to the to move to the positive nodes more often than the negative nodes.

And, then we essentially over this over the iterations we essentially follow the PageRank kind of process and we will see that we get a stationary distribution and that stationary distribution will basically help you identify nodes which are highly likely to be connected ok.

(Refer Slide Time: 07:33)



So, let us look at the in a formal definition of supervised random walk. So, you are given a graph G V comma E and you are also given a set of nodes set of positive nodes.

Now, the set of positive nodes are those which are connected ok you can also say that the set of positive nodes are those which you prefer to visit ok. So, D is a set of positive nodes divided by d 1 d 2 dot dot d m and L is the set of negative nodes which are not connected of course, this set would be much much higher than the positive node, but that is ok right. And we basically learn you know this kind of function this is basically function called psi.

Now, psi x y psi sub x y is a combination of node level and edge level attributes for a node pair x and y ok. Say for example, x and y they are 2 users ok. So, x has say attributes like age, gender, job, location y also has the same attributes. So, since let us say x and y are not connected right. So, we can come up with the function psi which takes into account the attributes of x and y and produces a value a real value and this real value is this real value indicates the strength of the pair of nodes ok.

And, we will see that the output of this ok will then be passed through a function f which is parameterized by w or omega whatever right. So, this w will be learned over time ok. So,

then based on that we compute the a strength. So, we basically have three things here the first one is this function psi which combines the node features, I mean the features of two nodes right and produces something produces a number for example, or a vector right and then that is passed through this function f ok.

Let us say let us say the attributes the attributes of x is 2, 1, 0, 2 and attributes of y is 1, 0, 1, 1 right. Now, the psi function can simply be the dot product let us say right dot plot will basically produce a scalar number. So, this dot product value will be passed through this function f. Now, f is a f is a parametric function right say let us say let us say it is not a dot product it is just a you know element wise multiplication. So, 2 times 1, 2 one times 0, 0 0 times 1 0 and 2 times 1, 2 ok.

So, the output of this psi function is essentially a vector 2 0 2 right and say you have this attribute you have this function f which has four parameters w 1, w 2, w 3, w 4 and essentially let us say the simple way to combine this thing is f of f w is w 1 x 1 plus w 2 x 2 plus w 3 x 3 plus w 4 x 4. So, in our case this would be w 1 2 plus w 2 0 plus w 3 0 plus w 4 2.

So, it will produce a scalar number right and this is your a u v right. So, a u v. So, a u v is essentially you know kind of a edge strength you would say edge strength or whatever a pair I mean strength of a pair of nodes ok then. So, then what you do what we do? We compute a transition matrix now remember whenever we do random walk process, we need some transition matrix the transition matrix can be the adjacency metric itself or it can be computed in different ways.

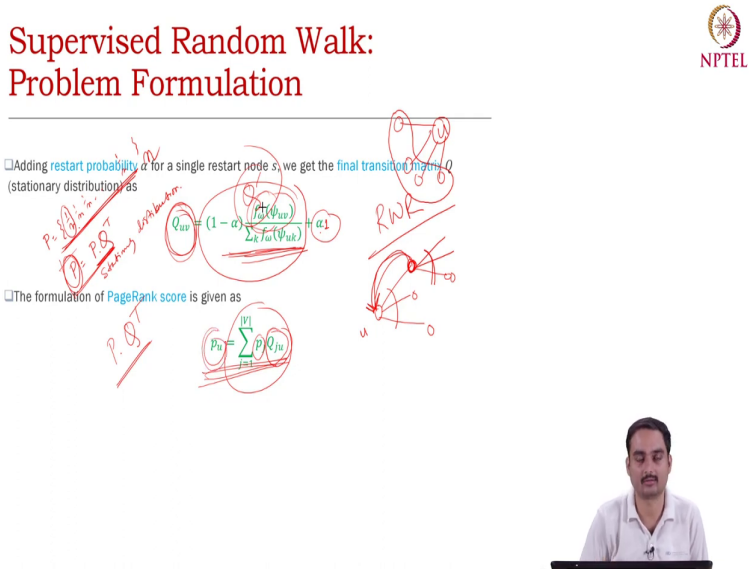
So, in our case it basically is computed in a different manner ok. So, this Q dash is what. Q dash is a u v divided by sum of a u w for all w such as u and w are connected belongs to this edge set E right. So, what does it mean? It means that say this is u ok these are different w s you got a u v a u w and so on and so forth. Now, you basically normalize it. So, that it becomes a stochastic matrix right the sum should be 1.

So, that when we do this PageRank process right you get a transition matrix and it converges ok. So, this Q matrix Q dash right. Q dash is a matrix right n cross n matrix and this is the stochastic transition matrix remember Q and A, A is a adjacent the matrix they are different right, because here Q is exhaustive in the sense that you are measuring this Q dash u v right for all pairs of nodes whereas, A I mean what I mean to say is that if you look at Q dash you

will see cells most of the cells are non zero whereas, in case of A, A is a adjacency matrix and you know its parts.

Therefore most of the cells will be 0 ok.

(Refer Slide Time: 14:22)



Now, what you do. So, from the stochastic matrix, from the stochastic transition matrix you get the actual transition matrix right, and remember what we do here we use something called random walk with restart, and we have already discussed what is random walk with restart in the chapter on link analysis right. So, the idea behind random walk with restart is that you start from a node u right.

And you follow one of its outgoing edges right uniformly at random move there right. Now, from this node you either move you either chose you either choose one of the outgoing edges or you jump to the original node again. So, with certain probability you choose one of the outgoing edges and with the remaining one 1 minus that probability you then jump to the previous node seed node right. So, this is random walk with restart.

So, you see in random walk with restart there are two components. The first component is the one that we already derived which is Q dash, it is a stochastic matrix and using stochastic matrix you do the random walk process. Remember in the normal random walk with restart we use adjacency matrix for this you know all this deciding whether to jump or not or which

edge to follow, but here the random work process is happening on this Q dash adjacency matrix or Q dash whatever transition matrix right.

So, you basically have all pairs of nodes and all pairs of nodes are virtually connected with some edge strengths and you decide based on the edge strength value ok. So, in other words we are computing the entire thing the entire process on a modified graph right. This modified graph is basically generated based on this Q dash ok where nodes are connected nodes are virtually connected and the connected the connection strength is derived by the a strength that we discussed ok.

So, this part is the random jump part sorry, but this part is the part I mean based on which you basically choose one of the outgoing edges and this is alpha. So, alpha 1 alpha is damping factor right in case of PageRank we discussed ok, and this is alpha times one it means that you are only allowed to move to the source node jump to the source node ok. So, this is now the adjacency matrix or whatever transition matrix final transition matrix ok.

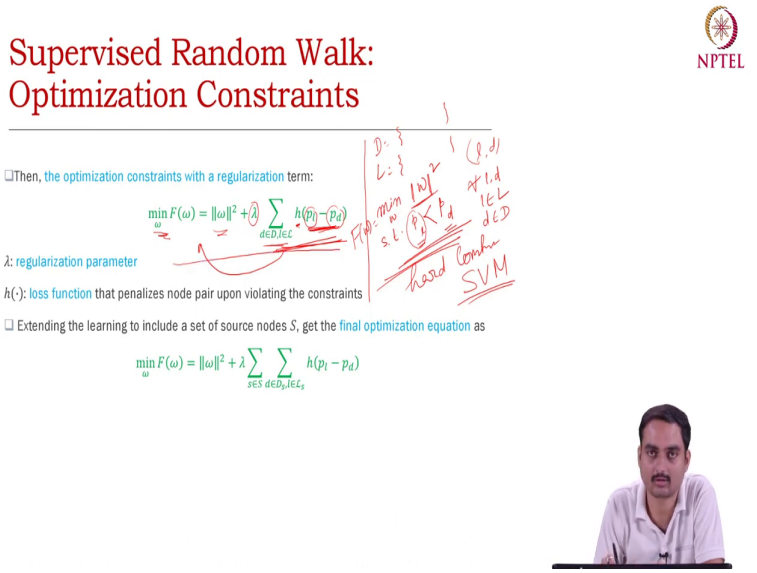
Now, if you only consider this part then this is a pure random walk random walk on the adjacency on the modified transition matrix, but since you add this part also it is a random walk with restart ok. And what is the expectation? So, when we start the random walk. So, say P is the distribution. So, you start with all uniform distribution right and then over times you multiply P with Q whatever Q transpose or Q does not matter, you multiply P with Q, this is the Q ok.

And, you keep on multiplying this thing when you see the stationary distribution meaning that after multiplying right you would see a certain stage, you would basically reach a certain stage where even after multiplication you get the same P right and that is the that is called stationary distribution. So, at the end of the day you would have a P right which would give you the stationary distribution and you would have probability associated with every node right.

So, the size of this vector is n right. So, if you unfold it what you are doing you are essentially doing this thing you are multiplying i mean say this is u ok and these are the nodes which are pointing to you. So, essentially you are multiplying you are taking the summation of all its neighbors; all its neighbors prestige right in our case prestige is basically P right and what are the neighbors how do we know which are the neighbors you will get it from Q ok.

So, for every node you are calculating this one. If you write it in a matrix form you are basically multiplying P times P and Q ok. So, now. So, I mean once we reach the stationary distribution you are done ok. So, we. So, far we have not explicitly understood the relation between w P Q all these things right. So, let me now derive this thing in a different manner I hope you understood the entire concept.

(Refer Slide Time: 20:29)



Now, I am defining it a different manner ok. So, what is the idea? The idea is that I do this I repeat this random walk with restart right and the expectation is that, the expectation is that the PageRank value of the positive nodes should be higher than the PageRank values of the negative nodes ok. D set of positive nodes and L set of negative nodes ok.

So, what is our optimization function here? The optimization function is we minimize w with respect to w right, and w is a parameter such that p l less than p d for all; for all l d where l belongs to L and d belongs to D.

So, let me explain. So, we want that our random walk process would produce a rank list or a PageRank value such that all the negative nodes would have lower PageRank values compared to the positive nodes ok and w is the parameter. So, the entire thing is remember. So, how do you calculate p? So, we calculate p based on Q, I am actually you know moving backwards right. So, I get. So, I will obtain p, p is dependent on Q, Q is dependent on f and f is a function f is a function of w. So, ultimately p is also dependent on w right.

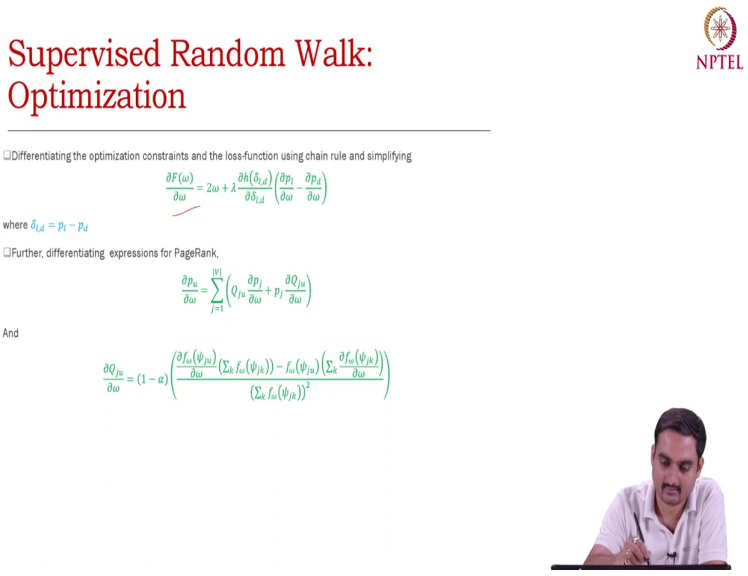
So, we minimize it why you minimize it because of the because we want that you know we prefer shortest you know smaller value of w right, smaller parameter values just for regularization ok, because if we make it high then you may end up overfitting or whatever. I mean you may explicitly extensively give more weight as to some features which are not that important. So, this is the this is the objective function and let us denote this by F w right.

But this is actually hard this is hard constraint right. If you know SVM support vector machine right, you will understand what do you mean by hard. It is hard because it is very difficult to guarantee that for every pair of l d, l and d this will satisfy ok. So, the way we use, the way we calculate you know soft margin in case of SVM right we do the same thing here. So, we define a loss function. So, this h is a loss function ok.

And I mean what does it do. It basically says that whenever now for every pair for every l d pair whenever I see that p l is greater than p d right. I will penalize you see here if p l is greater than p d this would be positive and this will this part will positively contribute, but we minimize it. So, that is bad ok. If pl is less than pd then it is ok because it is negative and we want to minimize that is ok.

So, this is a loss function and we will discuss what kind of loss function they have used in their paper, and lambda is a regularizer ok which basically gives a weight between this part and this part ok.

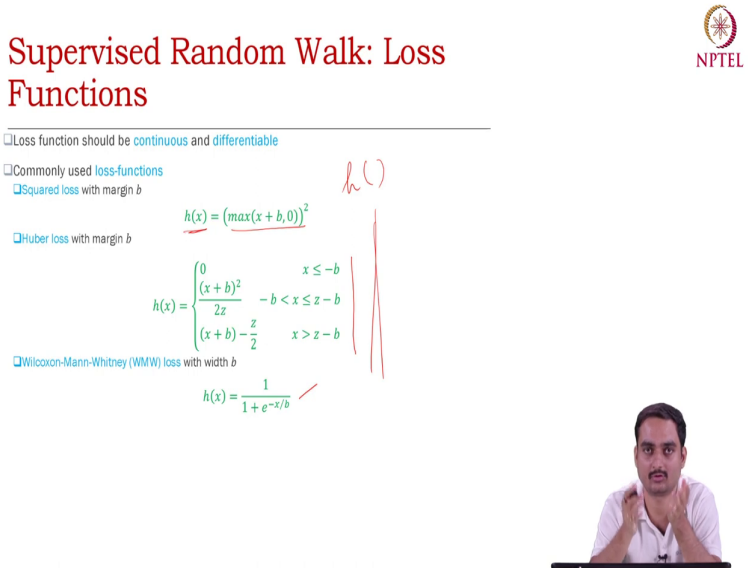
(Refer Slide Time: 25:03)



So, now, how to solve it? It is a standard optimization function although it is not convex, but you can still use gradient descent to get the values right. Take gradient descent I am not going the details of the this mathematical part.

But you take gradient descents and you essentially would be able to update the values over different times this is not a convex function, you what you can do you can start from different seed nodes and you try to approximate it ok.

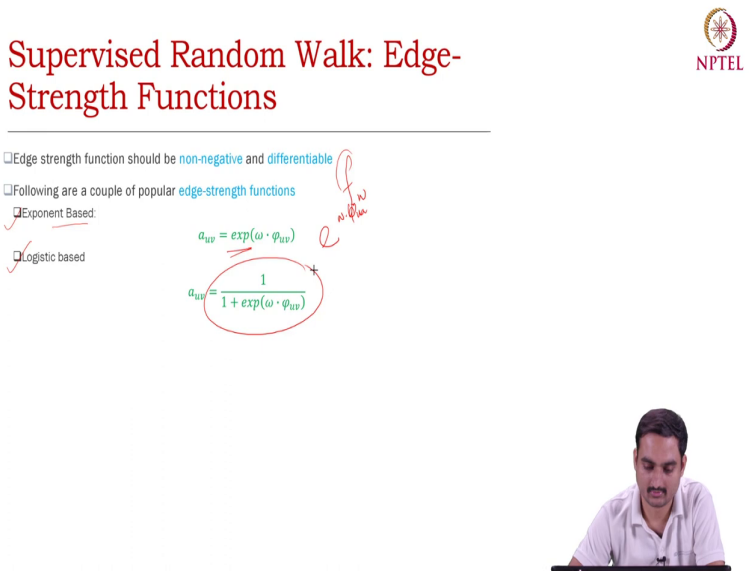
(Refer Slide Time: 25:37)



So, now, let us look at some of the components right the I mean the first component was this h x the loss function. So, the loss function in their paper they used a simple squared loss function which is this one ok or you can do even complicated loss functions like this or this one right.

So, this h was unknown, now this kind of h functions that I mean you can use in the optimization right.

(Refer Slide Time: 26:09)



So, the other one which was missing is this f w right. How does this f function look like? So, in their paper they used two types of functions, one is the this exponent based function right where, it is simply e to the power w times right phi u v ok, or you can do you can take some sort of logistic function 1 by 1 plus e to the power this one ok.

So, that is all. Now, let me go back again and recap what we have discussed. So, first we have this adjacency matrix we separate out the positive samples and negative samples. Positive samples are those, which are those nodes which are connected; negative samples are those which are not connected, then we compute this psi function right. Psi can be any function it can; combine right, then we have this f the f is a parameterized function right, f can be f can be this or this from there we get a u v.

So, a u v is not normalized. So, from a we will get A dash or Q dash whatever. So, Q dash is the stochastic transition matrix from Q dash we will get Q. Q is the final transition matrix which combines two factors the stochastic part whether you jump through one of the one of the through one of the outgoing edges, and the second component was the jump random jump. Combine them you get the actual transition matrix Q and then you repeat the PageRank right that is the process.

So, you see how you can utilize the node level attributes, edge level attributes as well as the topological structure in a nice manner right. The problem in supervised random walk is that it is time taking because you have to do this random walk with restart kind of process for every

nodes and you have to take all pairs. Although you can do some sort of you can take negative sampling and so on and so forth to squeeze the sample space, but still you need to do a lot of you basically need to do random walk restart for every node from every node right.

So, that is time taking, therefore it is not scalable the crude version of it, the you know the paper which was published in 2011-12 that paper was not scalable for a massive network, but then people actually took this thing forward try to make it scalable and so on and so forth ok. So, so this brings us to the end of this chapter. So, we have discussed you know one of the important applications of social network analysis called link prediction.

We have discussed heuristic based approaches. We have discussed you know maximum likelihood kind of approaches (Refer Time: 29:39) based approaches, we also discussed we have discussed the supervised approach to deal with link prediction problem. We also discussed matrix which we generally use for evaluating you know link prediction methods ok.

So, we stop here in the next chapter we will discuss you know another important application which is cascade right cascade growth prediction information diffusion and so on.

Thank you.