Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Computer Science & Engineering, Electrical Engineering, Mathematics

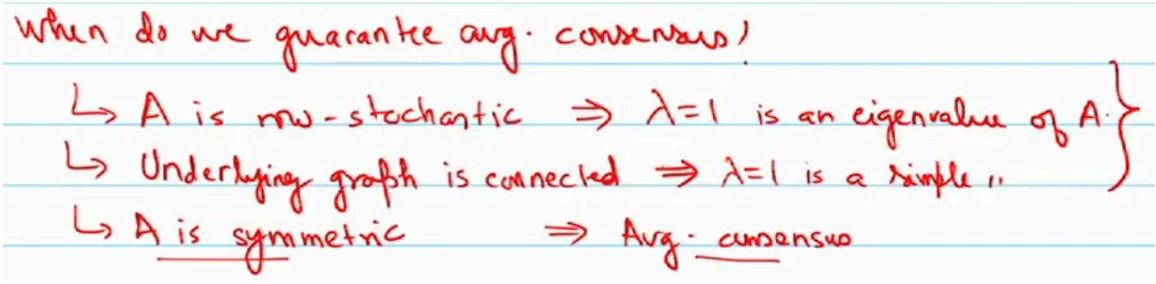
Indian Institute of Technology Bombay

Week-9

Lecture 30: Consensus Algorithms

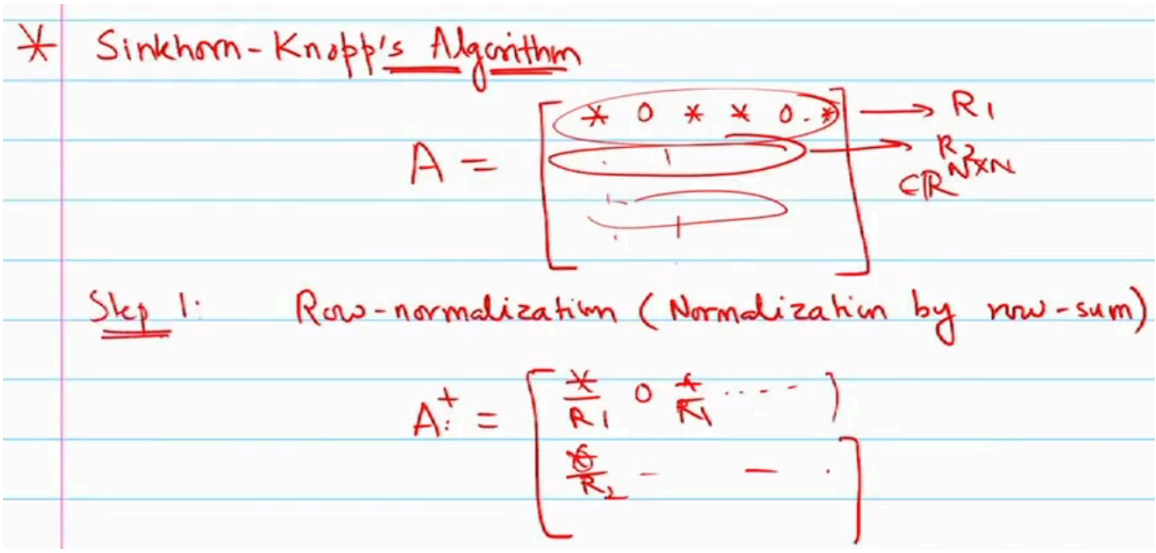
So, in the last lecture we looked at average consensus and consensus algorithms right. And one of the things that we saw was the role of the topology or the structure of the underlying network. that plays an important role right. So, when we ran the discretized consensus algorithm for one particular choice of A matrix, it had a consensus, but it did not converge to the average consensus value right. And for the other choice of A, it converged to the average consensus value. So, and the conditions that we derived was, so when do we guarantee average consensus? So, what are the sufficient conditions? When do we guarantee average consensus? So, A is row stochastic ok, which implies that lambda is equal to 1 is an eigenvalue ok.

Then we want underlying graph to be connected is connected which implies what is the simpler? Lambda is equal to 1 is a simple eigenvalue right ok. And A is symmetric which implies average consensus. So, with just these two you guarantee consensus, but if you have a to b symmetric that means a is also doubly stochastic both row stochastic as well as column stochastic then you can guarantee average consensus ok. So, how do we construct such a? So, one particular approach is using something called sinc on Does anyone know what Sinkhorn Knopp's algorithm is? So, let us say you are given a matrix A, a non-negative matrix A.

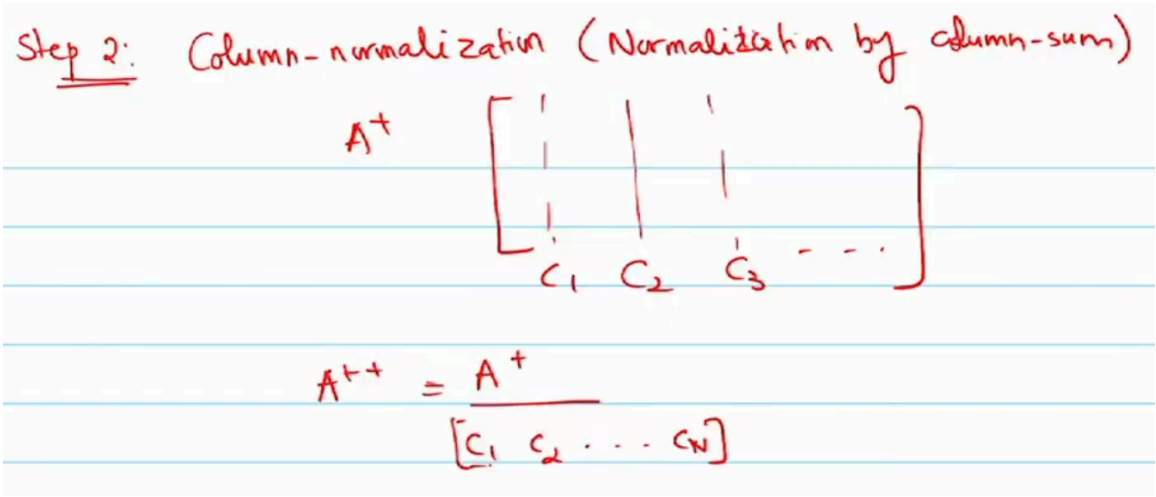


So, what do we mean by that? There will be bunch some 0s and some non some positive values ok. So, it is a non negative matrix, it is a square matrix ok. Now, what Sinkhorn Knopp's algorithm does is, so in the first iteration you are going to do row normalization or rather that is normalization by row sum. So that means, you are going to get these row sums, let us say this is this gives you r1, this gives you r2, r3 and so on. And then you

construct a new way, let us call it a plus, which is going to be this divided by r1, 0, r1 and so on.

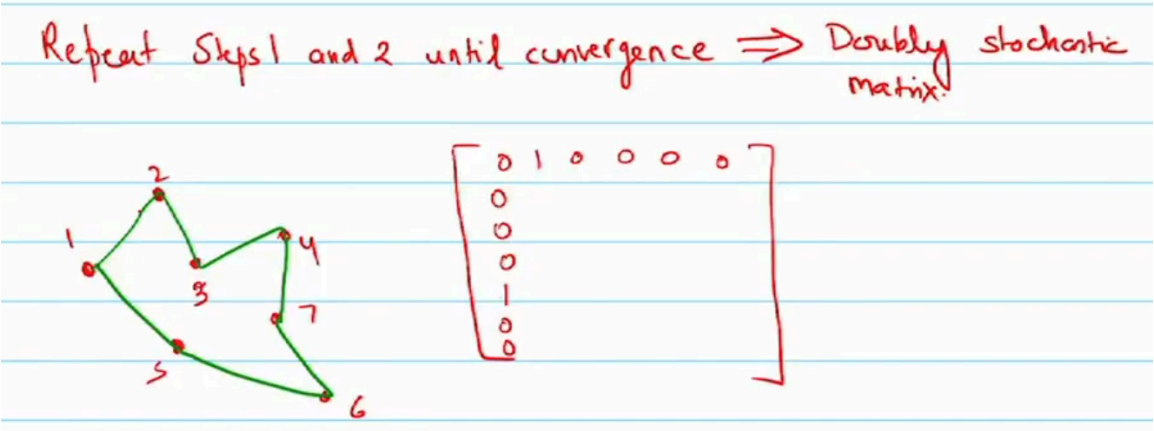


Likewise, this entry divided by So, you are going to do a simple row sum normalization of every row. So, that it becomes row stochastic. So, that is step 1. Step 2 is column normalization or column sum normalization by column sum. So this A plus matrix that you have now obtained, so you are going to be obtaining column sums of this particular matrix and then you are going let us say these column sums are C1, C2, C3 and so on.



So you are going to be constructing a new matrix A plus plus which is just going to be A plus divided by this column sum C1, C2. So, the respective columns are going to be divided by these column sums. So, now it becomes column stochastic, but then it need not be row stochastic anymore right because now you have divided this by column sums. So, it need not be row stochastic, but it turns out that if you keep repeating this process. So, rows make it row stochastic followed by column stochastic, row stochastic, column stochastic.

If you keep repeating this multiple times for a non negative matrix, it will eventually converge to a doubly stochastic matrix. So, this is what Sinkhorn Knopp's algorithm is. So, we repeat the steps 1 and 2 until convergence and this basically gives you doubly stochastic matrix. So, this is one particular way to obtain W stochastic matrix and this is often used let us say when we talk about. So, in the context of machine learning I mean this is aside from the course let us say I am trying to, so everyone knows what travelling salesman problem is.



So, travelling salesman problem is let us say you have these nodes that need to be visited by a salesman or a postman let us say. and in such a manner so that the total distance. So, every node is visited just once and the total length of travel is minimized. So, and then wherever the salesman starts basically he or she needs to I mean end their trajectory at the same point. So, for instance let us say this is the this turns out to be the optimal sort of route which minimizes the total travel distance.

So, in general it is an NP-hard problem right and if you look at, but then if I try to look at this solution to this particular thing, what does this turn out? Let us say this is city 1, 2, 3, 4, 5, 6. So, from 1 I need to go to 2, let us call it 7. So, from 1 I need to go to and from 5 I need to go to 1 right. So, you can see that this matrix is a special class of doubly stochastic matrix with exactly one entry being positive and every other entry being 0 right. So, if you know what softmax is in context of neural network, if I have a doubly stochastic matrix as a softmax kind of matrix and I just keep doing repeated sink on knobs algorithm on top of it, then you would hope that eventually this converges to my 1 0 kind of matrix that I want and this would be one way to generate solution to a graph structured travelling salesman kind of problem.

So, this Sinkhorn Knopp's algorithm is often used when you want to when your decision space is a stochastic or row stochastic matrix or a doubly stochastic matrix. So, Sinkhorn Knopp's algorithm is often used on these logits or the softmax scores. So, that is just an aside thing, but this is where you can see the role of doubly stochastic matrix coming. But what is the shortcoming of this like let us say if I want to use Sinkhorn Knopp's for

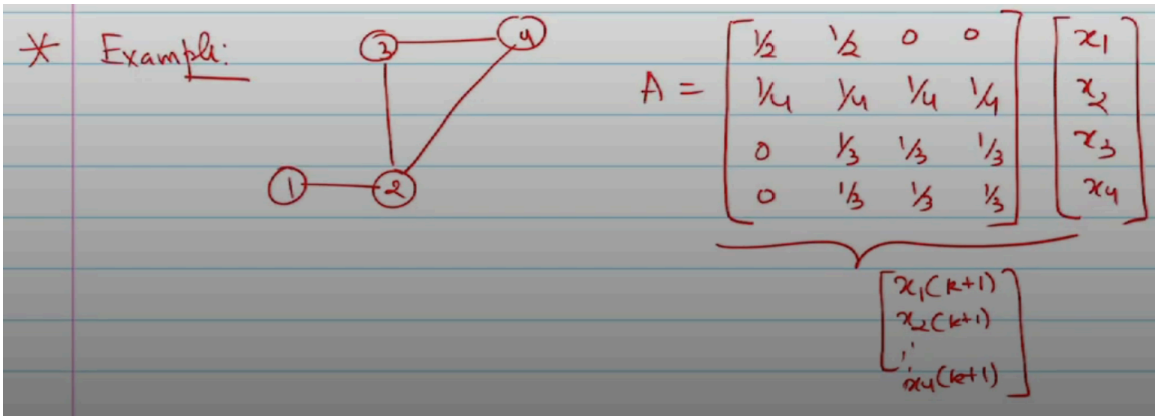
generating a doubly stochastic matrix. What is the shortcoming of this approach? Small what? That is fine I mean how does it matter because everything is going to be row and column not like everything is going to be row stochastic and doubly stochastic.

So, even. So, usually in practice I mean you mean the this particular thing is half. So, this is usually like 6, 7 iterations are enough. So, it is not really about that. So, not so much on the entries, but number of non-zero entries and zero entries is what it would not depend on, but that aside what is a bigger sort of difficulty in working let's say you are trying I mean there are multiple agents in the network and they are trying to I mean your goal is to be able to arrive at let's say as a centralized I mean you there is no centralized entity so you do not know anything about the topology you know what how your neighbors are right you know know you know about your connectivity and maybe to some extent you can talk to your neighbors and figure out their connectivity but you do not know the connectivity of the entire graph right So, for doing this row normalization and column normalization, I would actually need to know that. In order to compute the row sum, I need to know how my neighbors are also connected to other agents and so on and likewise column sums and so on.

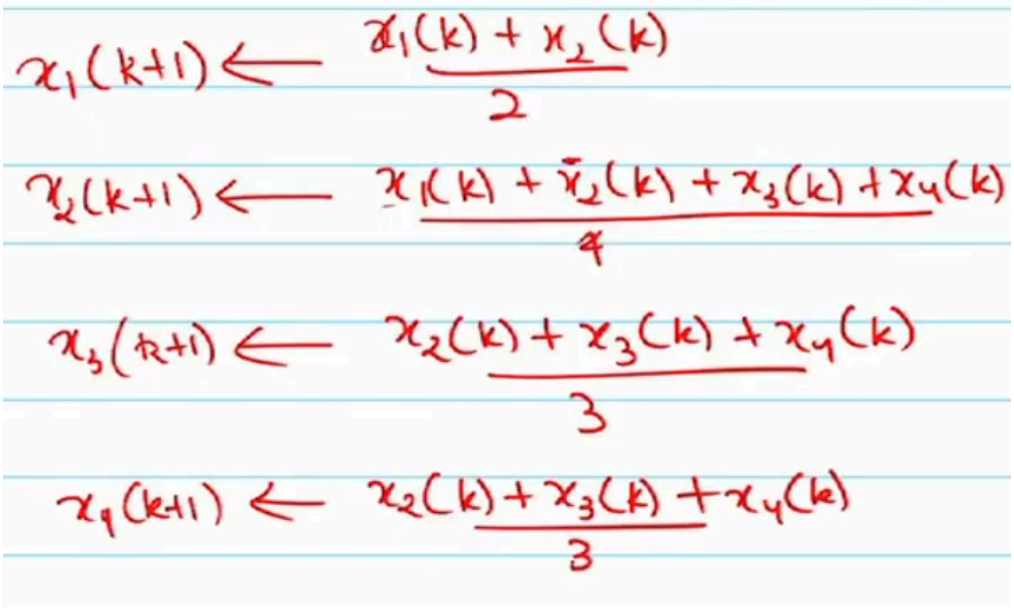
So, I would need to know the entire topology and that means I assume that there is a centralized entity which has access to all the information and that is something that anyway we want to get away with. So, this is not a very viable approach. for constructing an A matrix that is a row stochastic symmetric and also guarantee that guarantees that well I mean underlying graph is connected with the property of the graph. So, we are going to assume that the graph is connected, but this is not a very good way to generate W stochastic matrix. So, in the example.

So, you can assume that if there is an edge then you place 1 otherwise 0 means you can start with that. Yeah, graph you are going to be assuming that the graph otherwise there is I mean you cannot reach consensus if the graph is not connected right. So, if you remember that example of the temperature census. So, I think it was something like this and underlying topology was something like this right. So, what kind of a did we choose in the first case? So, it was half and half 0 0.

Now 2 is connected to every just had one fourth each 3 is connected to 2 and 4. So, 0 one third one third one third and likewise 0. and what is it really trying to do. So, suppose I multiply this with x 1, x 2, x 3 and x 4 and this basically gives me new x right, x i let us say x 1 k plus 1, x 2 k plus 1 to x 4 k plus 1, this basically gives me new x i's. So, according to this it basically tells you that x i k plus 1 is going to be an average of your own estimate and your neighbors estimate right.

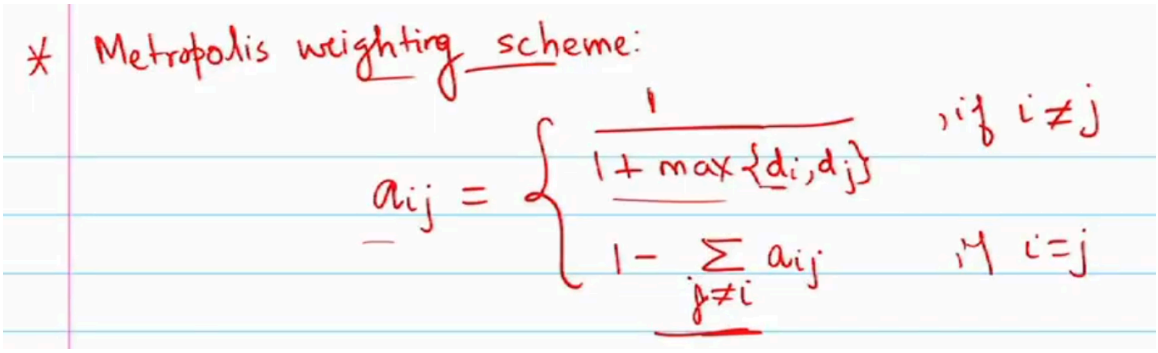


Likewise x2k plus 1 this is going to be an average of x1k. So, your own estimate and then your neighbors estimates and you weigh all of them like in this case we have an equal weighting for all the information that we have in the Similarly, x 3 k plus 1 is going to be x 2 k, okay. Now if I look at this particular matrix, this is definitely row stochastic right, but this is not column stochastic and that's why we saw that when we ran this consensus algorithm which is xk plus 1 is a times xk, it converged to the common consensus value, but it did not converge to the average consensus value that we wanted it to converge to right. And for that we had a different choice of a and let me tell you I mean let me also tell you how to construct such an A. So, we are going to be using something called metropolis weighting scheme and it goes something like this.

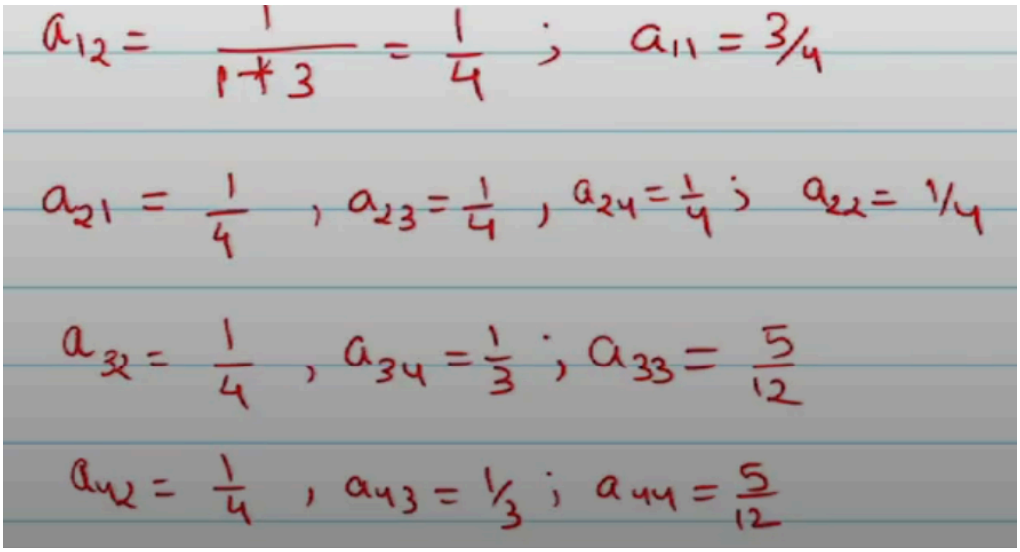


So, aij is going to be 1 over 1 plus max of the degree of the ith node and the degree of the jth node. Now, this is a local piece of information. You can talk to your neighbor, you can ask his or her degree and that way you will be able to ascertain whichever one is the maximum quantity. So, you do not need to know the entire topology. So, in constructing aij, I just need to know the degree of my neighbors.

and myself right and that is it. And I would know of my degree because I know who are my neighbors. So, I know of my degree the number of agents that I am connected with. My neighbor would know the number of agents that he or she is connected with and this way you can estimate this information or you can exactly obtain this particular information max of d i comma d j without having to know the entire topology right. so in this case agent i need not know about the entire topology of the network likewise agent j also does or any agent for that matter it doesn't need to know the entire topology of the network is this clear so what would be the a matrix in this case so let's first look at a 1 2 now what is the degree of the first of node 1 1, 2 is 3 and 3 and 4 are 2.

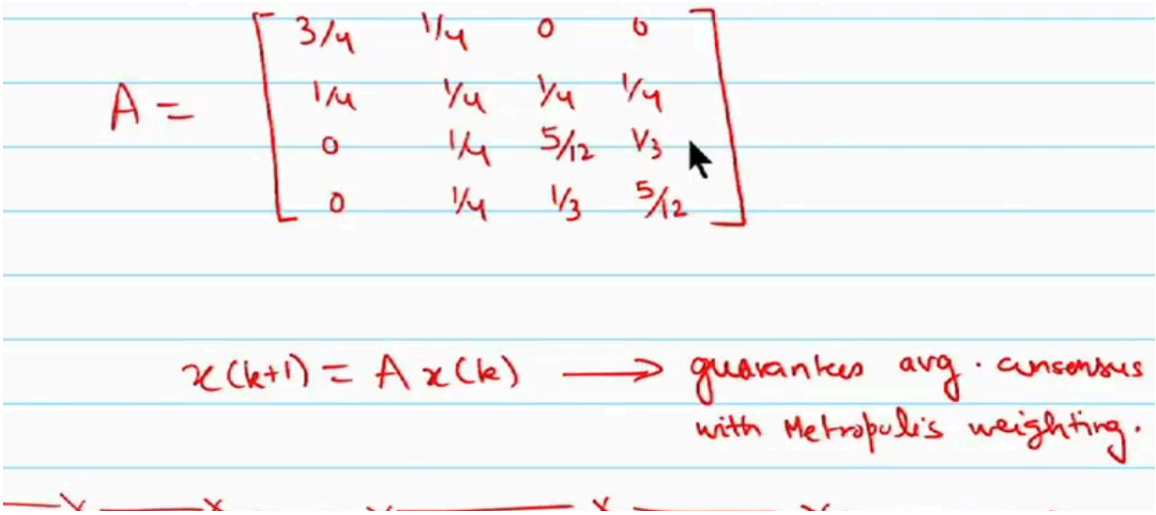


So, a 1 2 is 1 over, so max of 1 comma 3 is 3. So, 1 over 1 plus 3 which is 1 over 1 fourth and which basically gives you a 1 1 is 1 minus of this and it becomes 3 fourth. What about a 2 1? Now, a21 again is going to be one-fourth because it is again going to be max of 1,3. In fact, 2 has the highest degree which is 3. So, for that matter a23 is also going to be one-fourth and a24 is also going to be one-fourth which implies a22 is going to be 1 minus all of this which is also going to be one-fourth.



Now, we talk about a 3 1 is not going to be there because 3 is not connected to 1. So, that is going to be 0. What about a 2 3 or a 3 2? This is going to be 1 4th. What about a 3 4? Both are degree 3, both are degree 2 each.

So, it is going to be 1 3rd. So, a 3 4 is going to be 1 3rd and therefore a 3 3 is going to be which is 5 12 right. And likewise a 4 2 is going to be 1 4th, a 4 3 is going to be 1 3rd and a 4 4 is going to be 5 12th. And now if I try to construct this a, this a turns out to be 3 4th one fourth 0 0. So, this is 0 0 and you have a 3 2. And in fact, this is precisely the matrix that we had used in the second algorithm.



And you can also verify this is doubly stochastic, symmetric doubly stochastic and therefore, you can guarantee average consensus. Is this clear? So, this is the metropolis. In fact, this is often used in Markov chain Monte Carlo, this kind of metropolis scheme. this is often this metropolis weighting scheme is often used and that is why the name that is where the name comes from. But the idea is this is how you are going to be locally constructing your matrix without having to know.

So, in just one round of communication you would be able to know the degrees of every node like of your neighboring nodes and you would be able to construct this. Any questions on this? So, this algorithm that you are going to be running is xk plus 1 is a times xk and this would guarantee average consensus assuming the graph is connected with metropolis waiting ok. I mean technically we you can do that you can apply the same I mean in that case you have to be careful about the in degree and the out degree rather. So, since you are going to be exchanging information with your neighbors or you are going to be communicating with the neighbors. So, you would have to I think in that case you would have to just focus on out degree and I think it should be fine.

The only thing is you need in that case you need the underlying graph to be strongly connected and not just connected for you to be able to guarantee average consensus. So, this is the discrete time consensus algorithm. So, in the rest of the lecture we are going to sort of look since we had studied stability of dynamical system. So, we are going to look at the standard consensus algorithm in continuous time and then we are going to look at the fixed time variant of it. Because in consensus you, so average consensus is important

because I mean you just like let us say you are gossiping among yourself and if you converse to some belief that you cannot track for instance, I mean then there is no point right.

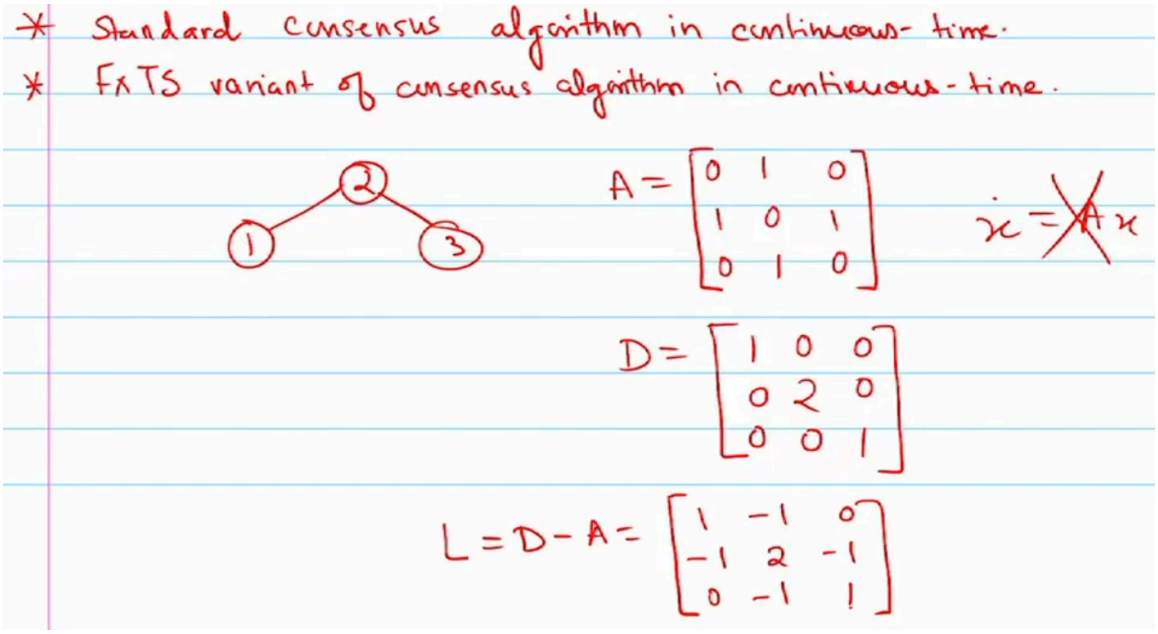
Everyone just says that I am 0, like let us say I mean you are trying to estimate the number of M&M's inside an M&M packet. and no one knows the exact number, but maybe you have an idea, your neighbors may have an idea and then you are trying to converge to a useful number and that would be an average of everyone's beliefs. If you just want to run a consensus, then you can just say that there are zero M&M candies inside this packet and if everyone does so, you are at a consensus value, but it's of no use. Yeah, so I mean in fact we will come to that. So, when you mix this average consensus scheme with another scheme which let us say you are trying to optimize a function and you are trying to track the optimizer of it, then I mean it is not just the average consensus part that is important and in fact you would see that this average would also move in some sense.

So, the quantity that remains that should ideally remain invariant is the summation of x i's. and that that you want to be I mean that you want this. So, you want that to kind of remain invariant and that is what we are going to look at. So, we are going to look at standard consensus algorithm first. So, in continuous time consensus and then we are going to look at f .

In fact, again in continuous time. In fact, you would see that in this context, in the context of continuous time algorithms, I mean you do not need this special structured A. So, this would, this makes sense when you are running discrete time algorithms or continuous time algorithms, this special type of A is not even needed. So, let us take an example. Let us say this, consider a very simple graph like this, ok. What is the adjacency matrix for this undirected unweighted graph? So, the a 1 1 is 0, there are no self loops.

So, 1 to 2 is 1 right, again you have 1s here and then you have. So, this is not a doubly stochastic matrix right. In fact, there is I mean unlike the previous case where we wanted a to b, I mean you can see that there are self edges right. 3 4th, 1 4th, 5 12th and 5 12th then no self edges here. Certainly we are not going to run the dynamics x dot is some a x.

So, this is this is not some this is not the conscious algorithm that we are going to be running. So, instead ok. So, let us let us also look at these quantities as well. So, what is the degree matrix and this? So, 1 2 1 ok. And let us also look at the Laplacian which is d minus a and it is going to be 1, minus 1, 0, minus 1, 2, minus 1 and then you have 0, minus 1, 1 this is the Laplacian.



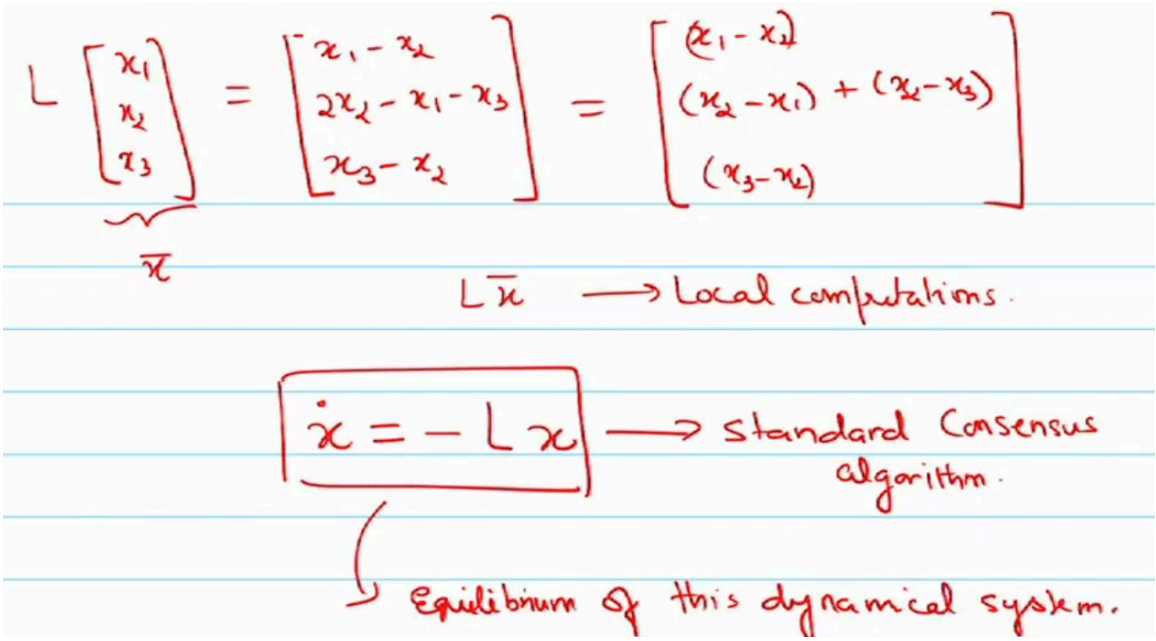
And what is one property of Laplacian? symmetric is 1 and then Laplacian times a vector of 1 is, so 0 is a, like if the graph is connected then 0 is a simple eigenvalue of the Laplacian. So, if I multiply this with vector of 1, this is going to give me 0s. So, let us also do one thing, let us multiply this Laplacian with this quantity x1, x2, x3 and let us see the output. So, the first entry is x1 minus x2, second entry is 2 x2 minus x1 minus x3, the third entry is x3 minus x2 ok. Which is another and another way to write this is x1 minus x2, this is x2 minus x1 plus x2 minus x3 and this is x3 minus x2.

So that means agent, so if I multiply any vector x, so this is my vector x. So I multiply this vector x with this Laplacian. What is it really doing? Basically agent i is basically subtracting agent j's belief from its own belief. And in case they have multiple neighbors, they are going to be getting the relative belief and just adding those up. So this computation, if I do, if I say l times x, if I define this to be, this is one, let's say this is my computation.

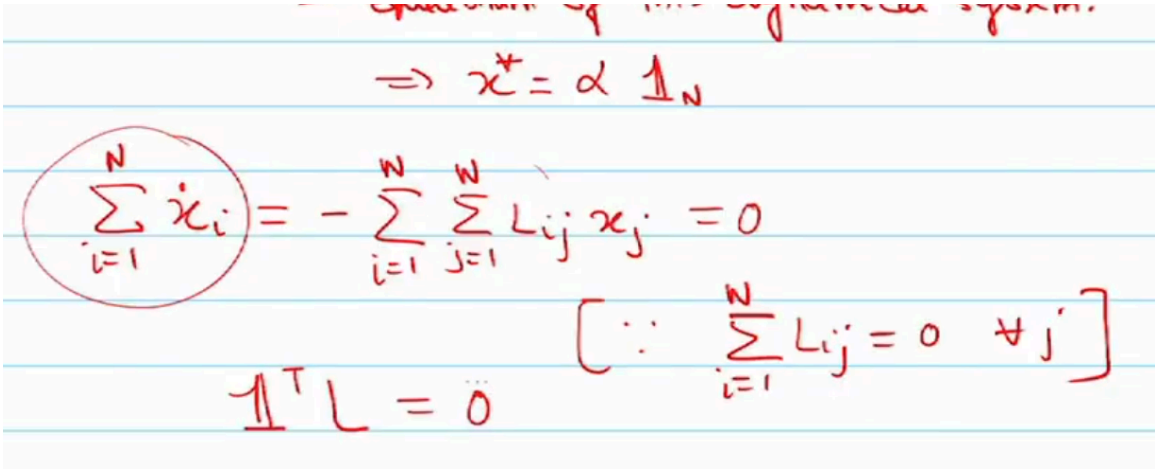
This computation is simply local, because I can subtract my neighbor's beliefs from my own belief and this is what I am doing and just summing those relative quantities. So, this computation is completely local. So, in this case, for instance, agent 1 does not need to know about what happens with agent 3. So, these computations are local. ok and when is this quantity 0 let's say when would this result in 0 when x1 is equal to x2 is equal to x3 all are like essentially the vector of ones right if the graph is connected it should I mean it should be collinear with the vector of ones.

So, that means x1 is equal to x2 is equal to x3 is equal to. So, that pretty much gives you an idea how to run a consensus algorithm and the continuous simple the standard

continuous algorithm or standard algorithm for consensus in continuous time is x dot is simply negative of L of x. So, first of all let us look at the equilibrium of this particular dynamical system. Yeah, so that is what we are looking at. So, what is the equilibrium of this dynamical system? So, when x dot is equal to 0.



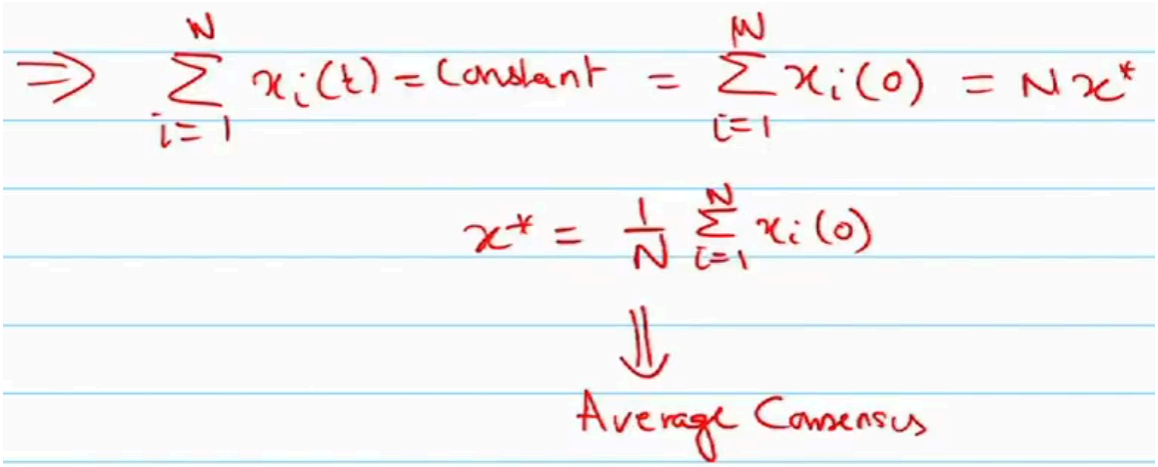
So, that means l of x is equal to 0. So, this implies x is basically some alpha times vector of 1. So, this is the equilibrium. So, you know that if the trajectories are converged to the equilibrium of this dynamical system, then the agents are going to be in consensus. We haven't said anything about average consensus yet, but we know that the agents are going to be in consensus. Okay, is this clear? Okay, so let's see what happens if I try to, let's say the agents, they run this locally, right? So when I say x dot is negative of L of x, essentially from the perspective of ith agent, it's just doing local computations.



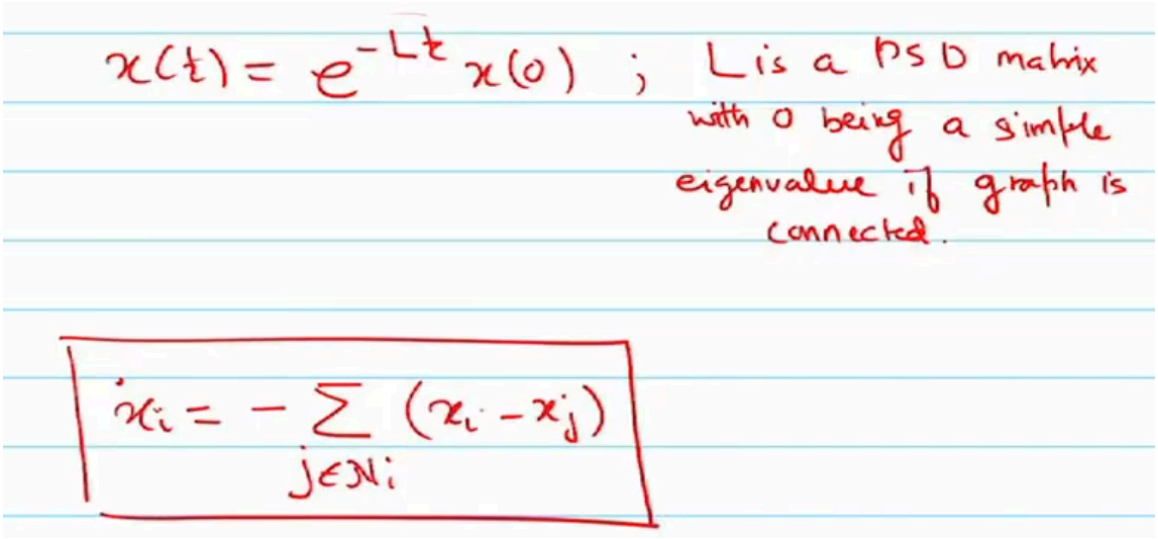
Okay, so what is this quantity? okay which is going to be here xj right lij xj that is what you are doing another adding the yeah if I do one transpose l what is this quantity which is? Yeah, which is? It is 0, right? Okay. So, what is this quantity? This is nothing but 1

transpose x dot. So, 1 transpose L that is 0. Why? Because if I can just sum it over Lij, summation i equal to 1 Lij and that is going to be 0.

Okay. So, this is going to be 0. since summation i equal 1 through n, l ij is going to be 0 for every j. One is both the left and the right, vector of 1 is both the left and the right eigenvector for this Laplacian. So, 1 transpose l is a 0, is going to give you 0 vector. So, that means this particular quantity is equal to 0. So, therefore, summation i equal 1 through n x i t is going to be constant because the time derivative of this is 0 which means this is equal to summation i equal 1 through n x i 0.



So, the summation is going to be constant. always if you run the standard consensus algorithm, the summation of x is always going to be constant. And now if everyone converges to the same value, let us say everyone converges to the same value, which is going to be n times x star because everyone converges to the same value. So, then the consensus value turns out to be x star is 1 over n summation i equal 1 to n x i 0, which implies average consensus. ok ok so why did we use x dot as negative l x and not positive l x it would have been the same thing right yeah so why because l is supposed to be positive semi definite matrix right so if i look at the solution to this x t x t is matrix exponentially to the negative l t times x naught right and l is l we know is positive semi definite matrix with 0 being a simple eigenvalue simple and eigenvalue if graph is connected ok. So, this basically tells you that you want this dynamics to be stable and because l all the eigenvalues of l are greater than equal to 0.



So, you want this dynamics to be stable and that is why we work the standard consensus algorithm is x dot is negative of l of x l times x. Even though we write this in a vectorized form from the perspective of individual agent i. So, agent i is simply running. This is what agent i is running.

So, these computations are local computations. even though it may look like the x dot, I am writing this in a nice vector form, but then agent i is basically running this particular dynamics. And if I try to combine everyone else's dynamics, so you get this x dot is negative of L x. A respective of the matrix A.

A respective of the matrix A. So, as long as the graph is connected. So, this would happen for any matrix. Thank you very much.