Distributed Optimization and Machine Learning

Prof. Mayank Baranwal

Computer Science & Engineering, Electrical Engineering, Mathematics

Indian Institute of Technology Bombay

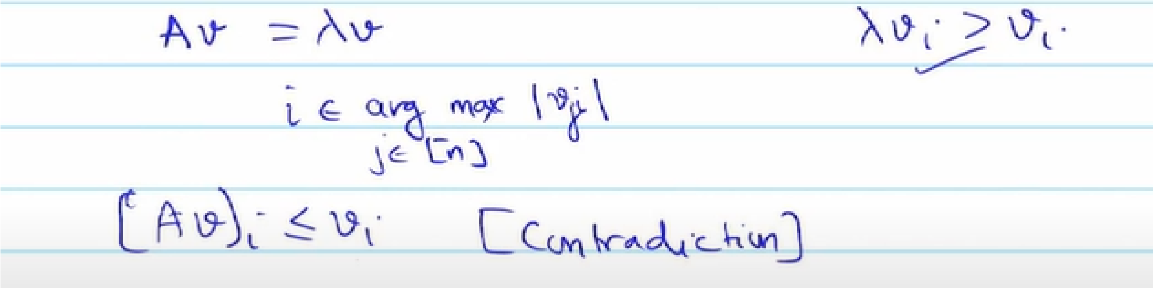
Week-8

Lecture 29: Consensus and Average Consensus-2

So, for any row stochastic matrix. max eigenvalue is 1 and if the graph is and if the underlying graph is strongly connected, strongly connected then 1 is also a simple eigenvalue. So in order to look at this, so let us look at, so fact 1 was every, so we were using this particular, it was here. A real symmetric matrix has real eigenvalues and it is also diagonalizable. Let us look at fact 2. So something that we have already looked at which is a row stochastic matrix has an eigenvalue which is 1 right, so 1 is an eigenvalue.

So, everyone with me on this? So, A times vector of 1 is simply 1. So, 1 is one of the eigenvalues of a row stochastic matrix. Fact number 3, no so which is basically related to the point that we just above mentioned just above. So, the no other eigenvalue is more than 1 for the row stochastic matrix.

So, how do we show this? So, let us say there exists a lambda greater than 1. So, we are going to show this by contradiction, there exists lambda greater than 1 which is also the eigenvalue of this of a row stochastic matrix right. matrix A with let us say eigenvalue v eigenvector v with v being the eigenvector. So, A times v it is going it is first of all it is going to be lambda times v and let us say i being the So, choose a max largest entry of your eigenvector ok. So, if I do A times v I am going to get.



$Av = \lambda v$ $\qquad$ $\lambda v_i > v_i.$

$i \in \arg\max_{j \in [n]} |v_j|$

$[Av)_i \leq v_i$ $\qquad$ [Contradiction]

So, basically the effect of this A times v would be lambda times v. So, that means for that particular ith entry I am going to be getting what? So, lambda times vi for that ith entry ok which is going to be greater than vi because lambda is more than 1, but at the same time what do the entries of A look like? It is essentially convex combinations of different v's right like it is essentially like if I do A times v it is essentially doing convex combinations of different v's if every row of A and that means every element is going to be smaller than the, so this A times v the ith element of this is going to be less than equal to vi. right. So, from one end we are getting it strictly greater than v i, from other end we are saying it is less than equal to v i. So, which is a contradiction and therefore, you cannot have an eigenvalue more than.

So, this is a contradiction and therefore, you cannot have an eigenvalue which is more than 1 for a row stochastic matrix ok. So, that is the fact that is fact number 3. And the fact number 4 is if the underlying graph is strongly connected, then 1 is also a simple eigenvalue. and the proof is of this is going to be very similar to how we looked at the proof for the Laplacians for the number of connected components right. So, if the graph is connected then 1 is the simple eigenvalue that means algebraic multiplicity of 1 is also is same as geometric multiplicity of this 1 eigenvalue which is going to be 1 ok.

So, with all the sort of key ingredients So, I am going to list out this theorem I mean you can I will maybe share post a proof on teams, but we will probably not have proof in this class. But then this would be this using this particular result we are then going to conclude that for rows stochastic matrix which are also symmetric in that case you are going to be arriving at average consensus and not just the consensus. be a non-negative. So, let A be a non-negative matrix with dominant eigenvalue lambda and the right and left eigenvectors are v and w such that v transpose w is 1. If lambda is simple and strictly larger in magnitude than all other eigenvalues, then we have limit k going to infinity ok.

So, let us revisit this statement. So, what this says is that let A be an A be a square matrix which is a non-negative matrix. So, that means every entry is greater than equal to 0 and the dominant eigenvalue of this matrix is lambda. for that eigenvalue lambda the right and the left eigenvectors are v and w ok. So, what do you mean by right eigenvector? So, a A times v is lambda v.

So, that is a right eigenvector w transpose A is equal to lambda w transpose that is basically your left eigenvector ok. So, the right eigen right and the left eigenvectors are v and w for corresponding to this lambda and we also normalize these eigenvectors. So, that v transpose w is equal to 1. So, if lambda is simple ok and strictly larger in magnitude than all other eigenvalues, then limit of this is essentially vw transpose ok.

$$\lim_{k\to\infty} \frac{A^k}{\lambda^k} = v w^T$$

So, let us look at the consequence of this in the context of consensus problems.

So, application of our theorem in the context of the average consensus. So, first of all we are going to be working with matrices which are row stochastic right. So, A is row stochastic. So, what is the dominant eigenvalue? 1 right. So, is rows A is row stochastic.

So, lambda is equal to 1 and what is the right and the left eigenvectors? So, v is going to be let us say vector of all ones. The next thing that we introduce is lambda being simple right. So, if the underlying graph is connected So, then lambda is simple. So, if and that also makes sense if the graph is connected that means, then we would be able to exchange in like any node will be able to exchange information with any other node and that is when we can talk about nodes arriving at consensus right. So, if the underlying graph is connected graph is connected.

this implies lambda equal to 1 is simple ok. So, on top of this if A is symmetric which means it is also column stochastic ok, then what is a left eigenvector? It will be a vector of all ones divided by n, divided by n because we want to make sure that v transpose w is equal to 1. right. So, ok. Is this clear? So, now if A is row stochastic and symmetric which makes A to be column stochastic, we assume that the underlying graph is connected.

So, therefore, this lambda equal to 1 is simple and because A is row stochastic lambda is equal to 1 is also dominant. So, all the conditions of this particular theorem are being satisfied and therefore, limit k goes to infinity A to the k I mean lambda is equal to 1 right. So, this implies as a consequence of this particular theorem limit k going to infinity A to the k essentially is equal to v w transpose which is 1 1 transpose divided by n ok. And if I look at the consensus algorithm which looks something like this x k plus 1 is a to the k plus 1 x naught. So, limit k goes to infinity x k plus 1 is essentially limit k goes to infinity A to the k which is 1 1 transpose over n.

$$x(k+1) = A^{k+1} x(0)$$

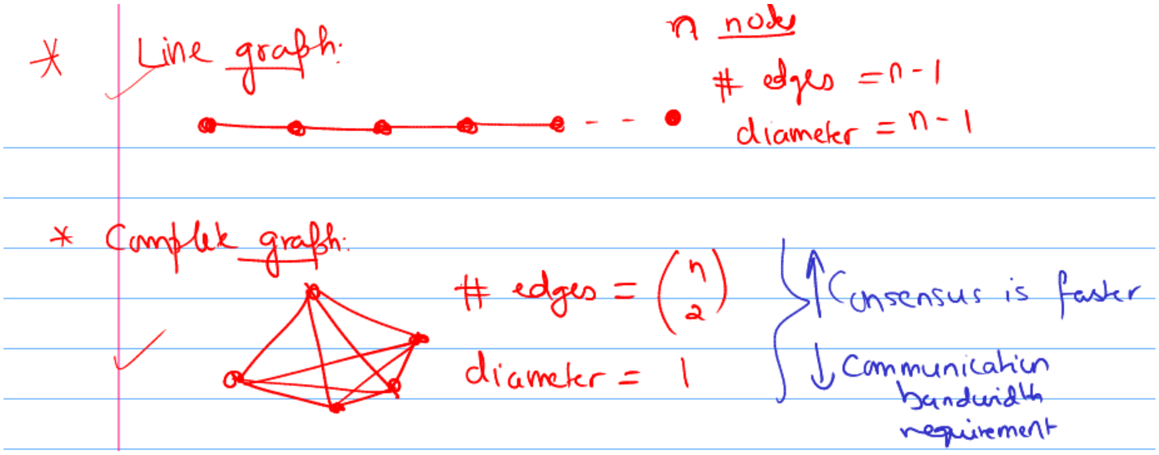
$$\lim_{k\to\infty} x(k+1) = \frac{\mathbb{1}\,\mathbb{1}^T x(0)}{n}$$

$$= \frac{1}{n}\begin{bmatrix} \sum_{i=1}^{n} x_i(0) \\ \vdots \\ \vdots \end{bmatrix} \implies \text{Average Consensus}$$

So, this is equal to 1 1 transpose x naught over n right and this means that every agent arrives at the. So, 1 over n summation i equal 1 through n x i 0 right ok. So, 1 transpose x A x naught is basically the summation of all the and then it is the same vector that is getting repeated and then you divide it by n. So, that means you arrive at the average consensus. So, a sufficient condition for average consensus is, the row stochastic, A is symmetric, underlying graph is connected.

So, if all these three are satisfied, then you can guarantee that the nodes they reach average. Is this clear? And that is why now if I If you look at if you consider the previous example that we looked at in the second case when A 2 or the in the second algorithm when A was doubly stochastic or I mean both row stochastic as well as symmetric then we saw that the agents or the sensors values they reached like all of them they converge to this number 24 right. So, this is just a sufficient condition. it may happen that with just with a non-symmetric A which is just host stochastic in certain cases you may get average consensus, but this is as I said this is not necessary this is just sufficient condition ok. So, if I consider two different types of graphs, so something like let us say a line graph or you have a complete graph right from which let us say there are 5 nodes.

So, in a complete graph every node is connected to every other node and so on ok. So, how many nodes like let us say if you have n nodes here, how many edges are here? edges is n minus 1 and diameter, diameter is also n minus 1 right. So, you would need what about number of edges here and yeah n choose 2 and the diameter just one right because in one step you would be able to reach any other node. So, the diameter is just one. So, which is preferred a line graph or a complete graph? In general let us say you want to run a decentralized distributed algorithm which.

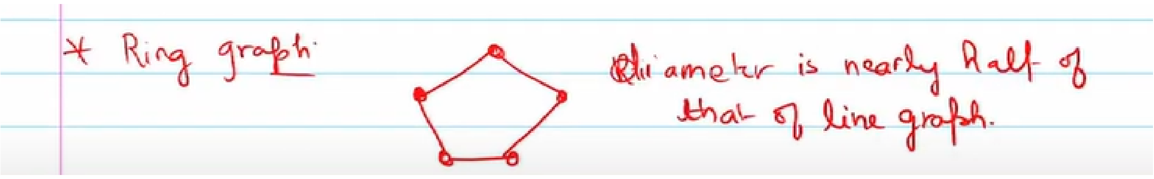


So, first of all which one do you think would it is easier to arrive at consensus or it is faster to arrive at consensus? Faster right. So, consensus is faster here. right. But then what is the shortcoming of? Yeah, so communication I mean you have a high communication bandwidth requirement for every node right. So, that is also not preferred.

So, consensus I mean it is a plus, but then it is not so good in when it comes to communication bandwidth requirement right? Okay.

Okay. Yeah. Which one? From. Yeah. So, again the topology plays a key role I mean obviously, you can in one case you can arrive at the let us say I mean it right now we are just talking about consensus, but when we are talking about solving a distributed optimization problem. In one case it is because the consensus happens faster it is also possible to arrive at the optimal solution for the common goal faster, but then it comes at the cost of requiring high communication bandwidth right. So, every node is going to be communicating with n minus 1 nodes in this in this case, whereas here every node is communicating with just 2 nodes at best. So, this I mean so in terms of communication line graph essentially has a very low communication bandwidth requirement right.

So it is basically somewhere midway is what I mean what an ideal topology would look like and if you and in that case that is something called static and we will come to this later maybe towards the last set of lectures is something called static exponential graphs which in some sense basically achieve the sweet spot between the communication bandwidth requirement as well as the like basically also trying to minimize the diameter. So, what static exponential graph is that every node is let us say the node 1 it is going to be connected to its second node, then 2 to the which is the fourth node, then the eighth node and so on. So, every node sort of in a modular fashion is going to be connected to its immediate, then 2 to the 1 which is second node, then 2 to the 2 which is fourth node, 2 to the 3 which is eighth node. And that's how you in fact it's a directed graph and you can show in fact there is very recent paper it showed that this is kind of is better than any known topologies whether it complete graph line graph is also something called ring graph which is essentially an extension of line graph, but it just by closing the ring it reduces the diameter by half right. because now the information earlier it was n minus 1, now it will be n minus 1 over 2 or n by 2 depending on.



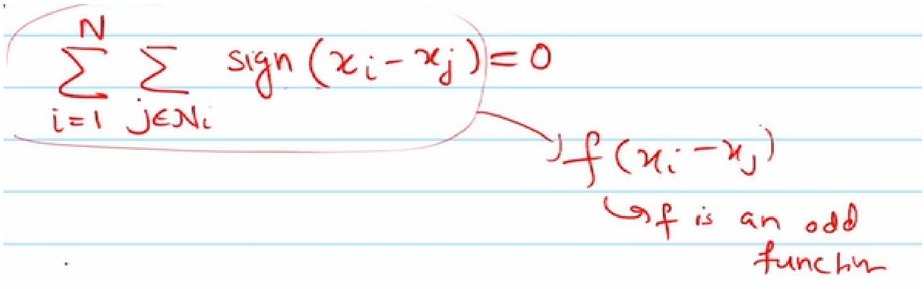
* Ring graph:

Diameter is nearly half of that of line graph.

So, this is so, the diameter becomes half whereas, the communication bandwidth requirement is pretty much the same except for the two end nodes which were earlier exchanging information with just one neighbor. Now, they are going to be everyone is going to be exchanging information with just two neighbors, but the diameter kind of becomes half of the original one right. nearly half of ok. So, the role of the topology is going to play a key I mean the topology is going to play a very key role and as I said in the previous lecture that related to topology or the graph diameter is your Fiedler value and you would see that the algorithms that we are going to be deriving the rate basically the rate of convergence that is going to be dependent on the Fiedler value of the underlying network. So, smaller diameter means larger Fiedler value and smaller diameter means you can arrive at consensus faster.

So, they are going to be proportional to the Fiedler value ok. So, before we arrive like before we start discussing the algorithm, I just wanted to briefly discuss two key results that we would eventually use in the subsequent lectures some important results on. So, let us say if I have something like this, so then there are n n agents in the network and we assume that it is an undirected unweighted graph. So, aijs are going to be 1 if there is an edge and 0 else.

The sign is a signum function ok. So sign of xi minus xj, so that is going to be, I mean you can also imagine these to be vector-valued functions. So xi can, so every agent need not just have a scalar quantity, every agent can have a vector-valued quantity, right. So then you basically evaluate this component-wise. So the idea is if you have an odd function like this, like a signum function here, then So, the summation of something like this is going to be. So, it could have been f of x i minus x j where f is an odd function.



$$\sum_{i=1}^{N}\sum_{j\in N_i} \text{sign}(x_i - x_j) = 0$$

$$\to f(x_i - x_j)$$

$\hookrightarrow$ f is an odd function

So, as long as you have that odd function this is going to be 0. So, quick proof of this. Component wise. So let us say T is this is what we are trying to evaluate ok. So by definition, this is nothing but j from 1 through n aij by the way this is true for undirected ok.

So, is this clear? So, instead of writing j over the neighborhood set I basically iterate j from 1 through n, but I use aij ok. And since i and j are dummy indices I can write this as j 1 through a j i and these are undirected unweighted graph. So, a i j is same as a j i both are equal to 1 and sin of x j minus x i is minus of sin of x i minus x j. So, this becomes a 1 through which is minus T right. So, this implies that T is equal to 0 or essentially what we wanted ok.

So, we use sign because eventually, the optimization algorithm that we are going to be using it is going to be a signed gradient flow. So, that is why I am using sign here, but as I said this would hold true for any odd function. The only property of signum that we use here was I mean it is essentially an odd function. So, any odd function and this would have still worked.

$$T = \sum_{i=1}^{N} \sum_{\underline{j \in N_i}} \text{sign}\,\underline{(x_i - x_j)}$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}\, \text{sign}(x_i - x_j)$$

$$= \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ji}\, \underline{\text{sign}\,(x_j - x_i)}$$

$$= -\sum_{i,j=1}^{N} a_{ij}\, \text{sign}(x_i - x_j) = -T$$

$$\Rightarrow T = 0 \quad [\text{Hence Proved}]$$

It can also be a vector-valued odd function. In that case, we would assume it basically applies component wise ok. So, that is what that is one result. Another result that we wanted to arrive at was 1 through n. So, the another result that we want to derive and again this is the result that we would be eventually using later. So, that is why I am deriving it right now is this particular term which is nothing but saying that summation i 1 through n summation j in the neighborhood set of i e i transpose w x i j this is equal to this particular term ok.

$$\sum_{i,j=1}^{N} \underline{a_{ij}\; e_i^T\, w(x_{ij})} = \frac{1}{2} \sum_{i,j=1}^{N} a_{ij}\, \underline{e_{ij}}^T\, w(x_{ij})$$

where, w is an odd function,

i.e. $w(-x) = -w(x)$

$e_{ij} := e_i - e_j$

$x_{ij} := x_i - x_j$

$$\sum_{i=1}^{N} \sum_{j \in N_i} e_i^T\, w(x_{ij}).$$

So, e i x i j is essentially the difference between x i and x j and I am just denoting this by x i j. So, in the previous example, in the previous result, this is x i j, x i minus x j. And let us say node i also has another vector e i, another piece of information e i. So, this is nothing but this particular term if w is an odd function. So, let us quickly Take a look at the proof So, since w is an odd function xij is xi minus xj.

So, this becomes xj, if I write it in terms of xji, then it basically aij ei transpose w xji and again since i and j are dummy indices, I can write this as ij 1 through a j i e j transpose w x i j. And a j i is same as a i j because it is an undirected unweighted graph. So, therefore, this is a i j e j transpose w x i j.

$$\underbrace{\sum_{i,j=1}^{N} a_{ij}\, e_i^T w(x_{ij})}_{} = -\sum_{i,j=1}^{N} a_{ij}\, e_i^T w(x_{ji})$$

$$= -\sum_{i,j=1}^{N} a_{ji}\, e_j^T w(x_{ij})$$

$$= -\sum_{i,j=1}^{N} a_{ij}\, e_j^T w(x_{ij})$$

$$2\sum_{i,j=1}^{N} a_{ij}\, e_i^T w(x_{ij}) = \sum_{i,j=1}^{N} a_{ij}(e_i - e_j)^T w(x_{ij})$$

$$= \sum_{i,j=1}^{N} a_{ij}\, e_{ij}^T w(x_{ij})$$

$$\sum_{i,j=1}^{N} a_{ij}\, e_i^T w(x_{ij}) = \frac{1}{2}\sum_{i,j=1}^{N} a_{ij}\, e_{ij}^T w(x_{ij})$$

ok and this pretty much. So, this is 2 times of. So, what do we want to arrive? Oh right, I just add the same quantity to it right. So, if I add this quantity over here. So, essentially the same quantity. So, 2 times of this e i transpose w x i j essentially I add the same quantity over here and this gives me ok, which is same as 1 through n a i j. e ij transpose w x ij and this basically gives us the result that we wanted to arrive at So, basically a key consequence of this result is, so this one is in terms of my own information right, but then I can sort of write it in terms of the relative information that I am going to be receiving from my neighbors.

So, essentially I am writing this e i and using e ij which is the relative information that I have with respect to my neighbors. So, essentially as long as w is an odd function, this holds true for undirected unweighted graphs. So, these two results, so I guess that is all I wanted to cover in today's lecture and these two results we are eventually, so in the next set of lectures we are going to be basically both describing the algorithm for consensus first of all and also then for distributed optimization, but in both these scenarios we would be making use of these results. and the same idea of fixed time gradient flow that we looked at. So, if you think of fixed time gradient flows what we do is you have gradient divided by the norm of the gradient right which kind of almost looks like you are trying to use some kind of signum information and that is where this sign thing comes into picture because you are going to be using the signum or the signed gradient flow there ok. Thank you very much.