Artificial Intelligence for Economics

Prof. Dripto Bakshi

Humanities and Social Sciences

Indian Institute of Technology Kharagpur

Week – 01

Lecture - 02

Lecture 02 : The Stable Matching Algorithm

Welcome to lecture two of artificial intelligence for economics. In lecture one we looked at two examples of network data of interpreting network data. In today's lecture we will look at something completely different. We look at what's called the stable matching algorithm. Now what is that? It's basically a matching problem as the name suggests but what exactly do we mean by that? Let's say we have two heterogeneous populations x and y. It could be a set of boys and a set of girls.

Every element in x or every member of set X has a preference ordering over the elements of set Y. Similarly, every element of set Y has a preference ordering over the elements of set X. If that's the case, who should be matched with whom? That's the kind of question which we want to answer in this lecture. Where do we see situations like this? Well, in the marriage or dating market, let's say there are a bunch of men and a bunch of women, every man has a preference ordering over the set of women and every woman has a preference ordering over the set of men.

Labor market, let's say there are X CEOs and X firms or Y firms and the CEOs have a preference ordering over the set of firms and every firm has a preference ordering over the set of individuals or CEOs. Credits and banks, right? So there are firms and banks. The firms have a preference ordering over which bank they want to borrow from. And the banks also have a preference ordering over the set of firms. Similarly, we have buyers and sellers, so on and so forth.

So these are typical examples of where we encounter the need for matching. There could be many to one problems too. For example, campus placements. There are students and then there are firms who visit the campus. The students definitely have a preference ordering over the set of firms who have come to recruit.

The firms also have a preference ordering over the set of students based on their GPA or other credentials. Great. So, let's think of a particular context, let's think of marriage or

the dating market which is the first example I cited. So, we are going to make a simplifying assumption, we make an assumption of monogamy that is one to one matching, one boy for one girl, one girl for one boy. We also assume equal sized population that is the set of boys and the set of girls are they have same cardinality.

If that's the situation the question is how to optimally match, but when I am using the word optimal what exactly do I mean by that? What do I mean by optimal? Let's look at an example. a set of two boys and two girls, the two boys are named Rahul and Aman and the two girls are let's say Anjali and Tina. Now every boy remember has a preference ordering over the set of girls, so Rahul prefers Tina the most and then Anjali, Aman also prefers Tina the most and then Anjali Tina prefers Rahul and then Aman, Anjali prefers Rahul and then Aman and this can be represented by this easy graph or network whatever you might call it. Now the question is if this is the preference orders of the different individuals, how to optimally match? What are the possible matchings? Well these are my two possible matchings. Rahul matched with Anjali, Aman matched with Tina or Rahul matched with Tina and Aman matched with Anjali, correct? Fine.

Let's look at matching one, the first matching. Rahul-Anjali and Aman-Tina, let's look at this matching. Is there any problem with this matching? Let's understand, let's look at the preference relationships. preference orderings. Rahul and Anjali and Aman and Tina, these are my two pairs.

Who does Rahul prefer the most? Rahul prefers Tina the most. Who does Tina prefer the most? Tina prefers Rahul the most. So in matching one, Rahul is paired with Anjali but Rahul likes Tina more. Tina is paired with Aman and Tina likes Rahul more than Aman. So in this case, both Rahul and Tina prefer each other over their current mates or current partners, which means they will break out of their current partnerships.

Rahul is paired with Anjali. Rahul has all the incentive to break out from that partnership if Tina says yes. Because why? Because Rahul prefers Tina more than Anjali. Tina on the other hand has an incentive to break out of her partnership with Aman because Tina prefers Rahul more than Aman. So Tina would be more than eager to break out if Rahul says yes.

Now both Rahul and Tina are eager to break out. So both of them will say yes and they will break out of their current partnerships. So matching one is unstable. We say that given a matching M, two individuals X and Y form a rogue couple if they prefer each other over their mates. So, in this case, Rahul and Tina in matching 1, Rahul and Tina form a rogue couple.

Matching 2 by the way has no row couples. Great. Now that we know what a row couple

means, here is the definition. So what is a stable matching? A perfect matching is where all individuals are paired. A stable matching is a perfect matching such that there are no row couples.

So matching 1 is not a stable matching. Matching two on the other hand is a stable matching. Great, let's move on. Let's look at one more example. Let's take a movie example.

Let's say we have these four movie stars Akshay, Salman, Amitabh and John and let's say there are two movies which are being made. and in each movie there will be two stars. Now who will be paired with whom? Now what is their preference relation preference ordering? Well Akshay prefers Amitabh the most then Salman then John as you can see from here. Akshay prefers Amitabh the most this is one I am sorry. So Akshay prefers Amitabh the most, then Salman and then John.

Similarly, Salman prefers Akshay, then Amitabh, then John, Amitabh prefers Salman, then Akshay, then John. Well, John's preferences are inconsequential because he is everybody's last choice. No offences against John, but then this is just hypothetical. Okay let's move on. The theorem is there does not exist a stable match.

If this is the preference ordering and if we want to pair, if we want to form two pairs there we can't do it. We can't form a stable match. Okay let's see. Assume that there exists a stable match n. So we are going to prove by contradiction here.

Assume that there exists a stable match. So without loss of generality, John will be matched with somebody in that stable matching. Let us say John has been matched with Akshay without loss of generality. WLOG is without loss of generality. Now if John is matched with Akshay, then by default Salman is matched with Amitabh.

great, but then look at the look at the preference relations preference orderings. So, Salman is matched with Amitabh, Akshay is matched with John let us go back to our preference ordering. Salman is matched with Amitabh and Akshay is matched with John, but Akshay prefers Amitabh Salman prefers, sorry Akshay is matched with John. So, Akshay prefers John the least. So, Akshay will definitely want to move out.

Salman on the other hand prefers Akshay the most. So, Akshay and Salman will form a rogue couple. Akshay has been matched with John, let us say without loss of generality, then Akshay prefers John the least, so Akshay wants to break out and Salman prefers Akshay the most, so Salman and Akshay will form a rogue couple. Salman also would want to break out because he prefers Akshay more than Amitabh. Great, so this is not, there cannot be a stable match, so M is not stable.

And we can prove this for, if I match John with, this is an exercise for all of you, if you match John with Salman and then see if you can find a rogue couple, you will be able to find one. No matter whom you match John with, there will be a rogue couple. Great, so this was my preference orderings, I cannot find a stable match. this just gives you an inkling towards a more general result which I am going to talk about now. This is the more important theorem.

This theorem states that a stable match will necessarily exist if the preference list can be represented by a bipartite graph. What is a bipartite graph? That is if we have two mutually exclusive sets and any member of set 1 has a preference ordering over the individuals of set 2 and any individual of set 2 has a preference ordering over the individuals of set 1 in such a setting a stable match necessarily exists okay if in this case which we just talked about it was not a bipartite scenario right. We cannot find two distinct, two mutually exclusive sets such that every individual in that set has a preference ordering over others, it is not the case here, this is not a bipartite graph ok. So, now this is theorem 2. Now we have seen that if it is not bipartite, we have found an example where a stable match does not exist.

But does it tell us for sure that if there is a bipartite graph representing the preference orderings, then we will necessarily have a stable match? The answer is yes and we are going to prove that. So in the next part of the lecture what we will do is the following. We will try to prove that in such a scenario a stable match necessarily exists and we will also propose an algorithm to find that stable match. But we will go the other way around. We will first propose the algorithm and then claim that the algorithm works.

We'll first propose an algorithm to find the stable match and then prove the existence of the stable match by proving that the algorithm necessarily works under all scenarios. Okay? Great. So first, I'll try to propose an algorithm for finding a stable matching. But while proposing the algorithm, I will take help of an example. So let's say we have five girls and five boys.

The girls are named A, B, C, D, E and the boys are named 1, 2, 3, 4, 5. These are my preference orderings. Every boy has a preference ordering over the girls. So, this is boy 1's preference ordering let's say. So, boy 1 prefers girl C the most and then B and then E and then A and then D.

Similarly, every boy has a preference ordering. Similarly, every girl also has a preference ordering over the set of boys. Girl A for example, prefers boy 3 the most and boy 4 the least. Okay, fine let's move on. So, if this is the situation can we find a stable matching that is a matching where there will be no rogue couples.

So, first which is what usually we try to do, we will try to propose a greedy algorithm. A greedy algorithm is the most intuitively obvious algorithm. What is a greedy algorithm? A greedy algorithm is where we optimize stepwise and we don't go back. So we'll propose a greedy algorithm and we'll try to guess or we'll try to see if this greedy algorithm works or not.

Great. Let's begin. Start with boy 1. So this is the algorithm. We start with boy 1 and allocate the best possible girl. That is, what is best possible girl? the highest in his preference list. Allocate him that girl. Next, look at boy two and match him with the best available girl.

Again, what do I mean by best? According to his preference list. Remember, these are the preference lists. So what's gonna happen? So who does boy one prefer? By the way, and we are going to carry on like this. So, let us see what outcome the gradial algorithm gives us.

So, who does boy 1 prefer the most? C. So, I am going to match 1 with C. That is what I do. 1 likes C the most, 1 with C. Then, I will come to boy 2.

Who does boy 2 like the most? A. Is A available? That is, is A already matched? No, A is available. So, I am going to match 2 with A. So, C and A have been taken. Now, I come to 3, boy 3. Who does boy 3, whom does boy 3 like the most? D, girl D.

Is girl D available? Yes, only C and A have been already matched, girl D is available, so I am going to match 3 with D, great, so C, A, D have been taken. Now I come to boy 4, who does boy 4 like the most? A, but A has already been matched with 2, so I cannot do that. Next C, well C has already been matched with 1, so I cannot do that. then comes D, well D has already been matched with 3, I cannot do that either. So, 4 will be matched with the best available girl which is B, right.

And coming to 5, 5 will be matched with the only girl who is left which is E, ok. So, that is it, that is the match which we get by using the greedy algorithm. But now the question is, is this matching which we have got, is this a stable match? How do we inspect that? We try to look at these couples and see if any of these couples form or is a rogue couple. Okay, so let's inspect and it turns out that boy four and girl C form a rogue couple.

Let's understand why. Let's look at boy four and girl C. Who is four matched with? Four is matched with B and one is matched with C. Fine. Now, who does four prefer the most? 4 prefers, 4 is now hitched with B right, but 4 prefers C more than B, 4 prefers C more than B. C right now is matched with 1, but C prefers 4 the most. see look at C's

preference ordering C prefers 4 the most and 4 prefers C more than his own partner which is B so C will definitely want to break out because C prefers 4 the most and 4 will also break out because 4 prefers C more than the current partner which 4 has which is B So which means boy four and girl C will form a rogue couple.

So we see that the greedy algorithm gives us a matching which is not stable. So the greedy algorithm has failed. So what do we do? Naturally we'll have to propose an alternative algorithm. So here we are proposing our stable matching algorithm.

Please understand this algorithm carefully. You can pause the video and read the slide or listen to this once more. So this is how the algorithm goes. Every day a boy will go and stand in front of the balcony of the girl he likes the most. Every day the boy, every boy, any boy will go and stand in front of the balcony of the girl he likes the most. ok the girl the each day a girl can either tell a boy standing in front of the balcony there could be more than one boy standing in front of a girl's balcony the girl can tell the boy come back next day or reject ok once the girl says reject to a boy the boy crosses that girl off from his list of possibilities and never goes back to that balcony ever.

This continues until every girl has exactly one boy standing in front of the balcony. When there is exactly one boy standing in front of each girl's balcony then the algorithm terminates. This is the stable matching algorithm and let's see if this works. and initially all girls are in the boys list he will keep the boy will keep crossing a girl out once the girl says reject okay fine let's look at the iterations now let's apply this algorithm on the example which we have got. So this is our preference lists remember so let's see this is day 1 what's gonna happen Every boy will go and stand in front of the balcony of the girl he likes the most.

So, one, whom does one like the most? Boy one. Boy one likes C. So, boy one goes and stands in front of C's balcony. Whom does two like the most? A. So, two goes and stands in front of A's balcony. Whom does three like the most? Where does D go? D goes and stand in front of sorry 3 goes and stand in stands in front of D's balcony. What about boy 4? Boy 4 likes A the most and boy 5 also likes A the most.

So, both of them again go and stand in front of A's balcony. So, this is how it is operating. Great. 2 knows that both 2, 4 and 5 prefer her the most. Girl A knows that boys 2, 4 and 5 prefer her the most.

Now look at girl A's preference ordering. Whom does girl A prefer the most amongst these boys 2, 4 and 5? Well clearly girl A prefers 5 the most. Okay, so girl A knows that 5 is available to her, then why should she bother about boys 2 and 4? So what will she say? She will say reject to 2 and 4. Okay, she will say reject to 2 and 4. So 2 and 4 will

now, what will 2 and 4 do? Boys 2 and 4, what will they do? well 2 and 4 will cross A out of their list so like this A is out of their list now mark A red fine now day 2 comes in day 2 again every boy goes and stands in front of the balcony of the girl he likes the most in his list in his list so again one goes in front of C's balcony 2 will now go in front of B's balcony, 3 will now go in front of C's balcony, 4 will now go in front of C's balcony, 5 in front of A.

So this is what happens now. Now C is having 2 boys standing in front of her balcony, 1 and 4. Now whom does she like more? Well C likes 4 the most and she knows that right now she is the most preferred girl in boy 4's list. So why should C bother about boy 1? Why should girl C bother about boy 1? So girl C rejects boy 1. So boy one now crosses out C from his list.

So these are my cross outs now. Right? Fine. Now again, now day three comes. Everybody goes, every boy goes and stands in front of the balcony of the girl he likes the most in his list. Okay? Remember, if you look at boy one's list, C is not there anymore. So where will he go? He will go and stand in front of B's balcony. 2 again will go and stand in front of B's balcony, right? So 1 and 2 both go and stand in front of B's balcony, right? You can see that.

3 will go in front of D's balcony, 4 will go and stand in front of C's balcony because A is not there in 4's list anymore. five again will go and stand in front of A's balcony. So this is how the balconies look like now. Now B has two men standing in front of her balcony, one and two.

Look at B's ordering, B's preference ordering. Between one and two, who does B prefer? Well, B clearly prefers two more than one, right? And B knows that right now she is she tops in the preference list of two. If that's the case, why should she bother about boy one? So she rejects boy one. So B will reject boy one and boy one in turn will cross out B from his list. This is the updated list now. Great, what happens next day? Where will boy one go? in front of the most preferred girl of his list, these have been crossed out.

So, boy 1 goes and stands in front of E's balcony, 2 goes and stands in front of B's balcony, 3 in front of D's balcony, 4 in front of C's balcony, 5 in front of A's balcony. This is what we have. Now, we have the terminating condition. We have one boy standing in front of or every balcony has exactly one boy standing in front and remember this was my condition for termination of the stable matching algorithm.

So, this is where the algorithm terminates fine. Now, is this a stable match? Is this match which we have got now? Is this stable? Remember we had got a similar match using the greedy algorithm which turned out to be unstable because we could find a rogue couple.

but this match which we have got let's see if this is stable and the answer is if you take a look at the preference of the boys and the preference of the girls and if you take a look at this matching it indeed turns out to be stable you will not be able to find any rogue couple in this matching. I will urge all of you to take a little pause and work it out yourself, try to find a rogue couple and you will see that you cannot. Now the question is fine, we have proposed an algorithm, now let us look at some desirable properties of the stable matching algorithm and this in turn will kind of make it clear that this algorithm works generally under any situation for any preference orderings if the preference orderings can be if it is a bipartite structure ok so the first result the stable matching algorithm necessarily terminates it will terminate at some point what is the proof for that it is very simple In fact, it terminates in less than equal to n square plus 1 days.

That's the upper bound. Why? What is the logic? What is happening in this algorithm? Every day, at least one boy is being crossed out. Or in other words, one boy crosses out a girl from his list. When every day, one girl is crossed out from some boy's list.

So every day there is a cross out. Now there are n boys and n girls. And so every boy has a list which has a cardinality n of size n. So how many total cross outs are possible at max? Well at max n square cross outs are possible. Right. Which means on the n square plus 1th day the algorithm will necessarily terminate. So, this is the proof that the stable matching algorithm will necessarily terminate if we have a bunch of boys and girls and the preference orderings and if we apply the algorithm it will necessarily terminate at some point after a finite amount of time and the upper bound is n square plus 1 upper bound of time.

What is the next one? Everybody gets married or paired. So, this is the definition of a perfect matching remember. In perfect matching everybody gets paired. In our example or we have made a simplifying assumption at the start of the lecture that the set of the cardinality of the two sets which are being matched are equal.

So, let us prove this. Let us say boy B has not been is not married at the end. Okay any boy I am naming him B which means B has been rejected by every girl right which means every girl is already married but if every girl is married every boy is married because this cardinality of the two sets are equal so if every girl is married every girl is married to one boy at least which means every boy is married. which means B is also married. So, which leads to a contradiction that B is not married ok. So, if the cardinality of the two sets are equal this is not possible. And finally, the last one which is that it this algorithm necessarily produces a stable match.

We have seen it does right, we have seen it does in the example which we worked out. But what is the intuitive explanation that it always does? It's the following. It's a pretty

intuitively easy proof to think about. Let us say it does not. Let us say we have applied stable matching and we actually end up getting a rogue couple.

Let's say the rogue couple is called Johnny and Amber. Now if it is a rogue couple it means it is not a couple in the matching right now. Now how come Johnny and Amber are not couple right now in the match which we have got? Either case one is either Amber rejected Johnny, right? If Amber rejected Johnny it means Amber must have had a boy more preferred than Johnny standing in front of her balcony, only then Amber would have rejected Johnny. And the final choice which Amber got was definitely somebody who was preferred to Johnny, which means Johnny and Amber cannot be a rogue couple, so it's a contradiction. What is case 2? Case 2 is Johnny never went and serenaded Amber, that is Johnny never went and stood in front of Amber's balcony. What does this mean? Now when is a boy going to a balcony of a girl? When is a boy not going in front of balcony of a girl? When he is not being rejected by a girl whom he prefers more than the girl under whose balcony he has not been to.

So which means if Johnny has not been under the balcony of Amber it simply means that Johnny has not been rejected by some girl whom he prefers more than Amber. If he has not been rejected by a girl whom he prefers more than Amber which means Johnny's current partner whom he has been matched with by the algorithm he prefers that person more than Amber which means Johnny does not prefer Amber over his current partner which means Johnny and Amber again does not form a rogue couple, right. So we see again we prove it by contradiction that formation of a rogue couple or occurrence of a rogue couple is impossible once we apply stable matching algorithm, okay. Great, so we have learnt the stable matching algorithm in this lecture. So we have learnt two different kinds of things, we have had two different exposures in the first two lectures.

In the first lecture we talked about network data and we tried to interpret two different situations, one in history, one in finance. In this lecture we have looked at something else, we have looked at stable matching algorithm. In the next two or three lectures I will deal with something completely different. I will talk about modeling of uncertainty. In artificial intelligence, training an agent to behave optimally in uncertain situations is a key thing.

So is true in finance. So in the next two lectures, I will delve a little more in finance and talk about the idea of hedging and risk management. See you in the next lecture.