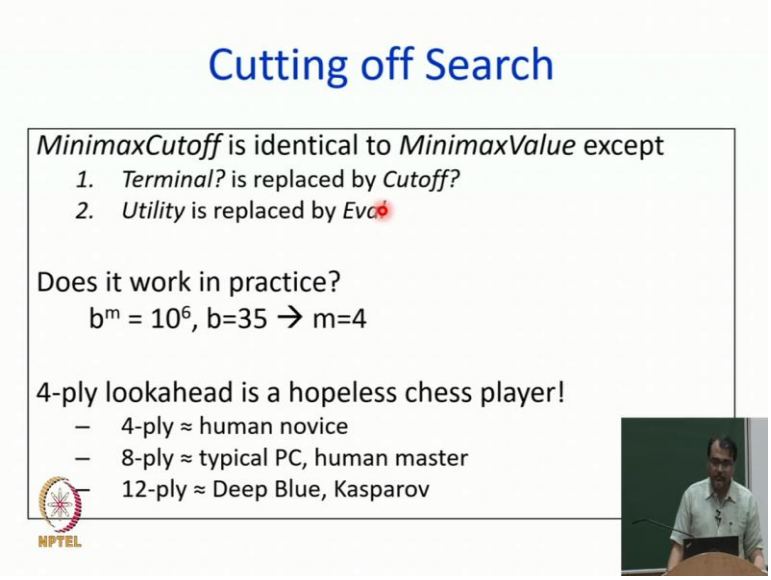
Artificial Intelligence
Prof. Mausam
Department of Computer Science and Engineering
Indian Institute of Technology Delhi

Lecture-33
Adversarial Search: Cutting of Search-Part-5

So what we need is something very obvious which is what we have been sort of alluding to also we cannot search in the leaf we have to stop in the middle b depth 4 b depth 8 whatever it is but we have to stop in the middle and that algorithm is called mini max with cut off.

(Refer Slide Time: 00:39)



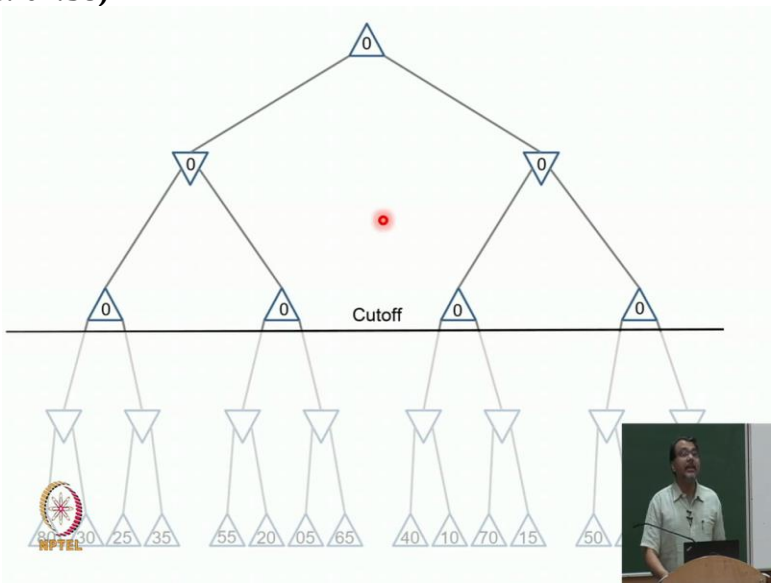
So depending upon how much you can see even if you play chess or any game what do you do? You think about if I move my pawn then the opponent will move their bishops and I will have to move my this and they will do this and at some point it is too much for you. So you stop. And then this is how it looks like when I moved upon. Well what if I moved my king. Then how does it look like.

So with every move you think a little bit about how the future is going to look like you do not go to the very end of the game you do not analyze all possibilities for you and for the opponent for you for the opponent till the very end of the game and then make a decision you say when I do this I might be able to capture that. So that looks like a good move to me. That is it, you do some

sort of shallow reasoning, and how much you are able to look in the future determines how good a player you are.

For example if you do only 4 plies then you are a human novice if you do 8 ply like a human master and this was the typical PC 20 years ago and then if you do 12 ply or 14 ply then you get to Kasparov and deep blue level. And not today all the chess players are like much better.

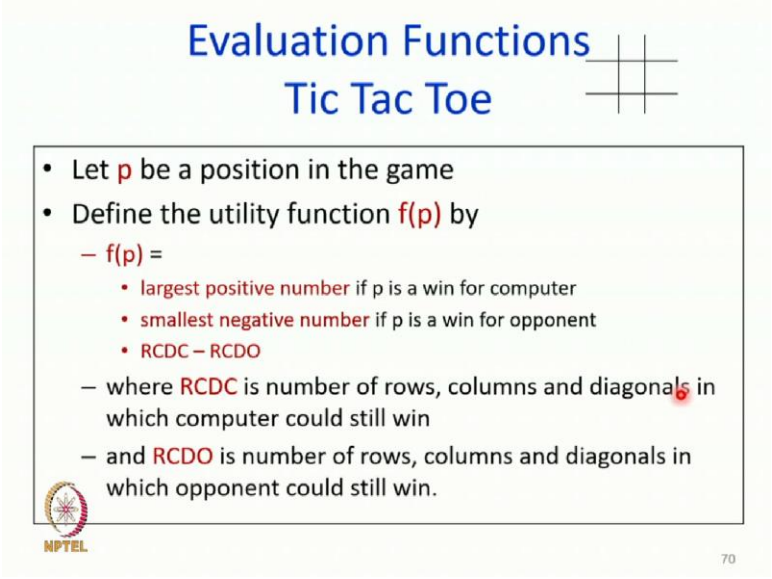
(Refer Slide Time: 01:58)



So our point is well we are not going to be able to the very end of the depth we are just going to cut it off in the middle. And you will have to do it because we just cannot get to the end too much. But of course now the question is what path has raised? The question is suppose I only search till this point what do I back up? Eventually I know nothing about anything. I have not reached the end of the game if I have not reached the end of the game I am stuck. What do I do?

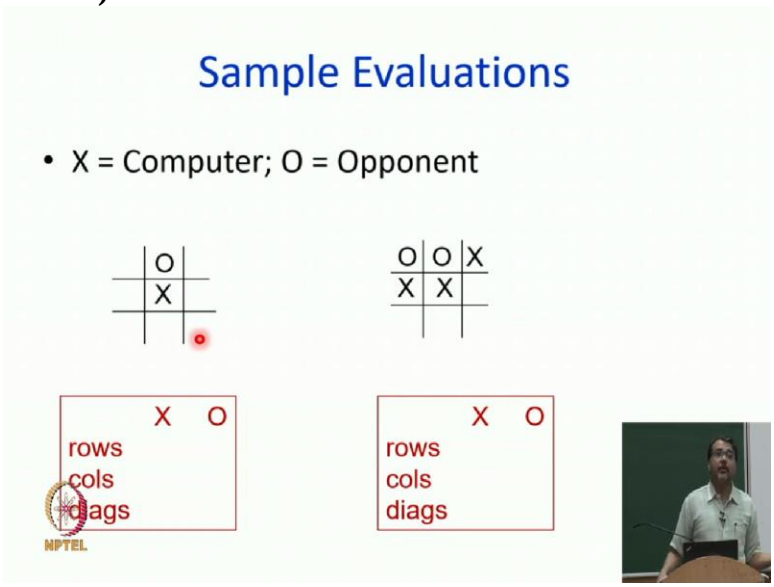
And the intuition is going to be or sorter the answer is going to be what (())(02:43)has already given. We will allow our search to stop in the middle and we will think this is leaves but we do not know who wins or who loses. So we will somehow evaluate whose position is strong me or my opponent in the leaf or the this artificial leaf somewhere in the middle of the node we will define an evaluation function that evaluates this board position and says at this point I think that you the computer is this much better than the opponent. And this is going to be the key to making any gameplay agent.

(Refer Slide Time: 03:37)



So we have called an evaluation functions. For example if I want to Tic Tac Toe and of course Tic Tac Toe is a very simple game boring game. You can even search till the very end. But suppose I stopped in the middle and I want to predict who is going to win or whose position is stronger. What may be a simple way to achieve this well let us say I defined this idea that I will count the number of rows columns are diagnosed in which the computer can win still. And numbers of those columns are diagonals in which the opponent could win still and then take a difference and say that is my evaluation function.

(Refer Slide Time: 04:23)



So for example if I give you the first board position then for computer cross how many ways in which it can win? It can win in 1 diagonal and 2 diagonal. It can win in this horizontal position

and another horizontal position. It can win in this vertical position in another vertical position. So how many ways can it win? 6 but for not how many ways can it win? No diagonal. It can win in 1 horizontal and 2 horizontal it can win in 1 vertical and 2 vertical.

So in this case which looks stronger, the cross look stronger and look stronger by let us said 2. Now, this is a evaluation function, I can multiply everything with 20 and I can multiply everything add everything with 10 whatever it does not matter it is some notion of how much stronger the computer is versus the opponent or vice versa. So this way I have what have I done? This actually important?

What have I done? I brought out this idea that the number of those columns and diagonals in which I can win minus to opponent can win is a good property to have in the board. Where did this idea come from? This idea came from my pocket. I looked at the game I thought about the game I analyze the game and I said this is a good property to have this property came completely from my pocket.

And now I told the computer that this is a good property use this as an evaluation function this property will determine the evaluation function of that point. And I will use that to back it up. Is it a well formed? Setup now does that make sense? Just to repeat this is actually important and the most important part of today's lectures. So I am doing minimax search with alpha beta building and I do a cutoff I stopped somewhere in the middle I cannot search further I do not have the time.

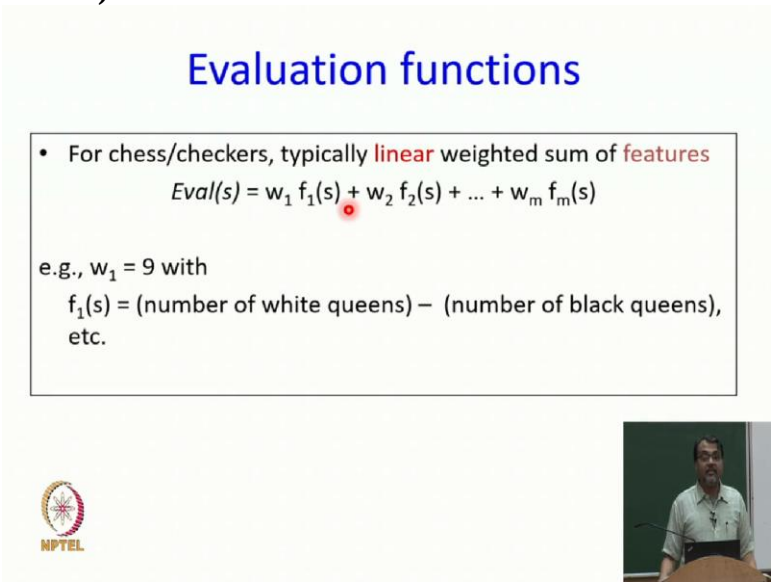
At that point I have only a partial board position I have not reached till the end I will evaluate this partial board position. How would I evaluate it I will come up with some property of the domain which says which whether this board position is in my favor or in the opponent's favor and how much and that will be the value of the leaf and then I will back it up. So at the top I will be taking that action which maximizes the utility function of the point of cutoff.

And the utility function will be defined by me. But of course it is very hard for me to define the utility function for every state. For a game like Tic Tac Toe it might work out but let us say I am

giving you a game like chess. How would you do it? So there the traditional way and there is a modern way also which you know which we will come to towards the very end of the course. But the traditional way is that I know do not come up with one property of the board I come up with many properties of the board.

How many queens do I have? How many pawns are near? You know, queening how many of my pieces are getting a tab? How many pieces do I have? How many pieces does my opponent have? How many bishops do I have? How many bishops do does my opponent have? Etc. I define all of these what we call features.

(Refer Slide Time: 08:03)



Where do these features come from? They come from my pocket. This is very important so I define features. For example, one feature would be number of white queens minus number of black queens. If I am white this is a feature now I will define many such features let us say I define 50 features 100 features like this. Now my goal is to define the partial I mean a board configuration and for each board configuration.

Each feature will have some value like number of white queen minus number of black queens, maybe 0 number of you know white bishops minus number of black bishops maybe 1 etc. Numbers of you know white pawns minus number of black pawns maybe minus 2 feature can be negative the opponent has more pawns. Now in this situation I need to somehow consolidate all

these feature values and create one function which is what we are going to call the utility function of the stopping point, intermediate board position.

How do we do that? And as a simple starting point, we will take a weighted sum of all the feature values. So my evaluation function would be w 1 f 1 + w 2 f 2 up to w m f m if m are the number of features and these w i are the weights of each feature, it is the importance of each feature. For example number of white queens minus number of black queens is a very important feature.

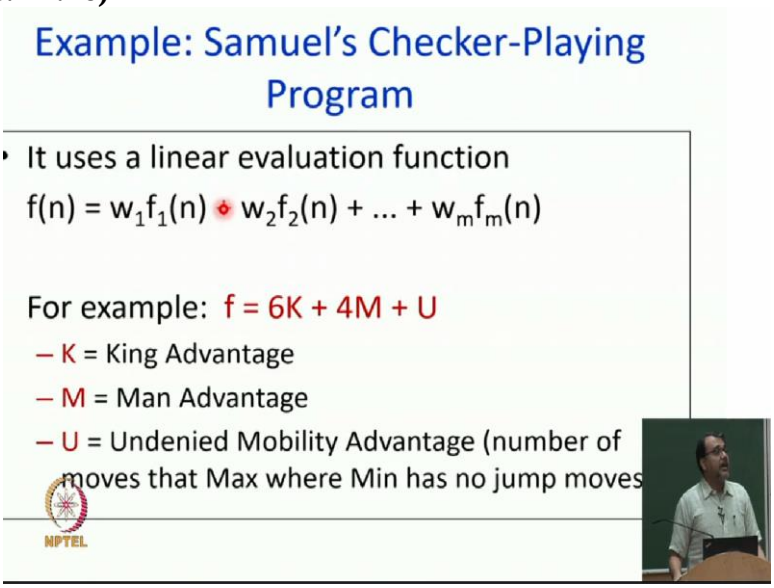
Number of white pawns minus number of black pawns important but not as important as number of queen difference. If I have 2 queens now and my opponent has 0 I am like super strong. Whereas if I have 2 points less I am weaker but not by as much weaker as if I had been if I had 2 queen difference. So w 1 would be much higher w for the pawn would be much lower comparatively.

So, now we have done this division of labour, the human the domain design that the game developer has defined all these features and has defined a representation the linear sum of basis for how so this is also causes sum of basis function representation. Each of these features sometimes also called a basis function does not matter these are different terms of the same thing.

So I have not defined all these features now I need to somehow compute the weights. So now we have whenever I am in the setting where I have to learn something some data that is where we get into machine learning. So we did sign up to study machine learning in this lecture. But this is a bonus we are going to study machine learning because until we study some machine learning, we will not be able to complete the story of gameplay.

And there is a full field which does this called machine learning. It is now a very famous, very popular, very influential field is a full course on machine learning that you should do after you do the AI course. But for now, without going into all the details, I am going to give you the flavor of how these things work.

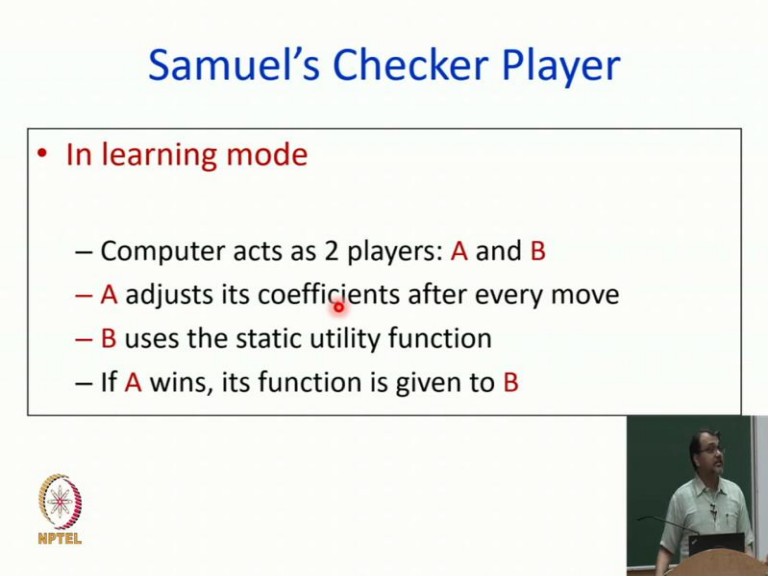
(Refer Slide Time: 11:46)



So for example, the very first time some of these ideas were used were in Samuels checker playing program. Arthur Samuel was a researcher IBM in the 50s. He created this checker planes program and he used machine learning. In fact, he is credited to coining the term machine learning. We earlier told you who coined the term AI or artificial intelligence, Arthur Samuel who played the created the checkers program, coined the term machine learning he may not have known that will become that important to them.

So, for example, we are looking for an evaluation function which is like something like 6K + 4M + U were you know, this is checklist formulas not everybody may or may not know it would even know a checklist, but 6 times K advantage + 4 times Man advantage U times undenied mobility advantage and so this is what we are looking for. We have defined KMU us features. We are not looking for 6 4 and 1 as the weights of that. So, how do we do that? And we are going to do that by experience. We are going to do that by playing. So what is going to happen is let us say we have some initial value of the weights?

(Refer Slide Time: 13:05)



If I have initial value of the weights I can play the game. Now I am done I can just play the game using those values of the weights. But I cannot play because I need an opponent. So let us see the opponent also has the same values of weights. So let us say now I have some values of the weights, I and I have 2 game players for those particular ways. Let us call them A and B. Now B is going to be fixed A is going to be learning. This is important stuff.

If you have any questions, you know, feel free to stop me and make sure that you understand this. So what is happening is that I have some weight values for each feature. If I have a weight values, then I can compute the evaluation function and every node if I can complete the evaluation function, I can do mini max alpha beta pruning and cut off and I can search and I can decide which action to take which move to take and I can play the game. So basically, I have completed my game player.

And let us say I make 2 copies of this game player A and B, let us say B is going to be state going to stay fixed. B will not change. A, however is going to learn against B. And what does learning mean? Updates rates very good and for most of modern machine learning, learning basically means update waves. You may have millions of waves in the most modern world. In this small case, we may have 3 weeks or 10 weeks. But learning for most modern machine learning is learning rates.

So what is going to do is A is going to adjust its coefficients after every move and how is it going to adjust its coefficients. That is very important role. And again, I am going to just give you the sort of post basic code. So intuitively, suppose f 1 n is positive. When should I increase w 1? I should increase w 1 if, initially I thought this board is good, this position is good, but it is actually even better.

That is the intuition. I have initial idea of how good this board is. And then I did some play and I get a better idea of how good this board is, and support this new idea of how good this board is says that this board is better than what you originally thought it to be. I am going to increase the weight of all the features which are positive and I am going to reduce all the weights of features which are negative that is it.

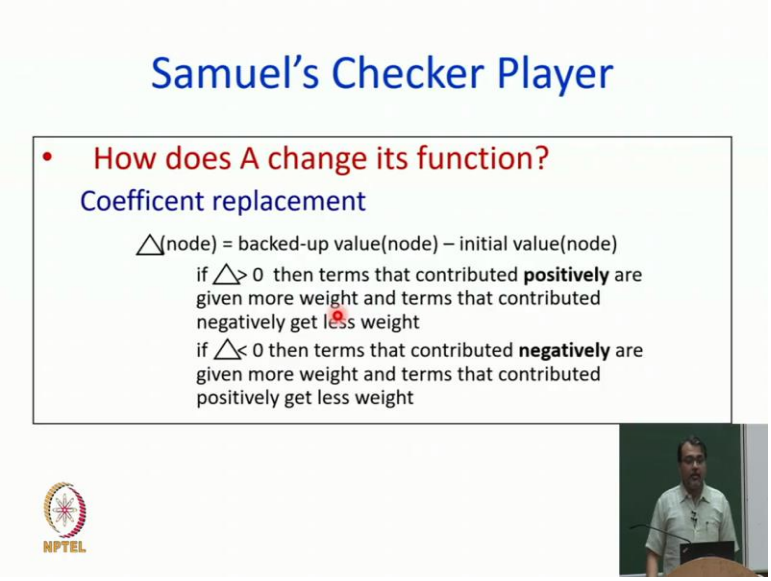
So, us sort of going to do the w i = w i plus or minus some learning rate some difference, some delta that is what you have to do this is the goal of all machine learning, then the only question is how do you compute how much you increase it or decrease it by and then people do gradient descent and is that in the error but essentially all w i = w i plus something. And that is all update equations and all of machine learning.

So, coming back now, the issue is, how do I know what was my old original board positions value and how do I estimate a new value. Make sense? Now you know how to estimate the old value? What is the old value? Look, I already have some w i. For every board position I can, evaluate the w i f i summation. So I have some f n for that board position. So this is my original board positions value original value of the board position.

Now, what am I going to do next is I am going to play the game up to some cut off, backtrack and compute the best value based on this look ahead search. So this is important. So what do I do? Initially I came to this node i say there is no little as good as 20 based on my current weight values, then I started playing this game up to let us said depth 5, whatever, I did this, I backtracked not at the depth 5, I am still using the same evaluation function.

It is not like I am using something very different. I use the same evaluation function. But now I am back tracking these values. And finally, my original load which had value 20, let us say has value 30. So now what do I need to do fundamentally, I need to move my weights such that this 20 moves in the direction of 30.

(Refer Slide Time: 18:43)



So the initial value of the node is the original f value and the backed up value of the node is the value that I used with minimax and cut off and this evaluation function at the cut off point. If this delta is greater than 0, then all the terms that were contributing positively to me, all my features which are positive there weights have to increase and all the features that were negative, their weights have to decrease.

On the other hand after backing up, I get no, this is not 30 it is actually 10 early I thought it is as good as 20 but not as good as 10 that means I need to reduce the value of my node, that means all the features that are contributing positively there which need to go down and all the features that are contributing negatively they need to go up how much they go down, how much they go up, depending upon depends upon how much delta is and what is the learning rate, how quickly do I want to move my weights.

Usually learning rates are not kept very high; they are kept low so that I sort of average it out over time and over different notes. This is the main idea of machine learning in the context of gameplay and in the previous generation of AI and machine learning systems, all the features

came from our pocket. But in the most modern version features are not given by humans. The machine learns the features.

And this they do by deep neural networks. And we will talk about that towards the end of the course, questions. Yes, so, the question is, is it possible that at this intermediate point, when I did some learning I have gone in the wrong direction? Yes, it is possible because it is not like this. This is sacrosanct because even at the end of cut off, I am still using this evaluation function to evaluate and I am only playing against this fixed B.

Now, this particular opponent may have weaknesses I may learn to exploit those weaknesses. And start to think that this board position is really good because in this board position, I can exploit my opponent's weakness. But that is not going to give me the best game play. So what how do we fix that problem, by the way very good. What is your name? Divya says if I get a better A, at some point I make A the B and restart.

This is what it says here. If A wins, if I am actually making progress, if I am improving my value function, if I am improving my evaluation function, at some point, you say look B your good for making me learn. But now you are a crappy opponent. I have really learned so well, that I do not want to learn against you. And so I mean, in practice in human world, we will say look, you opponent, you go improve your game.

I will play this better opponent so that I improve my game. In the context of game playing, I become the opponent somebody else is being developed, which is better than B. Slightly different. But it is the same idea. I would not want to play Roger Federer in London I can play my mom though. I am a terrible today. So that is the point, so you want to always play it to a point where you can learn something more. And after that you are the opponent is not useful very much.

That is exactly what you do. You start with A and B, once you have a better A, you make that B and you create A, which is better than the new B, which is the old A and you repeat this process and you keep doing this. And the more you do this, the better A evaluation function gets, and the

better it gets, you know, you get a better game. Yes and you go to the next level, and you are the same way values, then levels want to make a difference.

So the question is that if I go to the depth 5 and I have an evaluation function, the depth and then I back it up what happens to levels 1 to 4 when you are also getting some insight about 1 to 4, because even for 1 to 4, you had some evaluation function, and you have some backed up value in that case you are backed up value is less than 5 depth, it is 4 depth or 3 depth or 2 depth, but you can still use it to improve.

Maybe you will trust it less because the look I had was less. But see this is essentially internal consistency property. It is saying that whatever w i is you get it should be such that whether you back it up, or you evaluate the top, the difference is very small. It is almost the same thing. Its internal consistency, but then you play against it and you improve it further. So this is how the world works.

There are many tricks that you can use. For example, you play the game till the end do not back up anything but then start backing up in the reverse direction because at the end, you are getting some real information about who won the game who did not win the game. So you can back it up. You can also do some kind of exploration step in this exploration step sometimes your opponent will take a random action.

That way you know you will not be playing against a very fixed deterministic opponent, you will be playing against slightly stochastic opponent that way you will become more robust to different kinds of opponents etc. So, there are lots of tricks of the trade that one can develop this and you know, as you go into the topic of reinforcement learning, we will discuss some of those ideas as well, any other questions? We should stop? Thank you