Foundations of Cyber Physical Systems

Prof. Soumyajit Dey

Department of Computer Science and Engineering

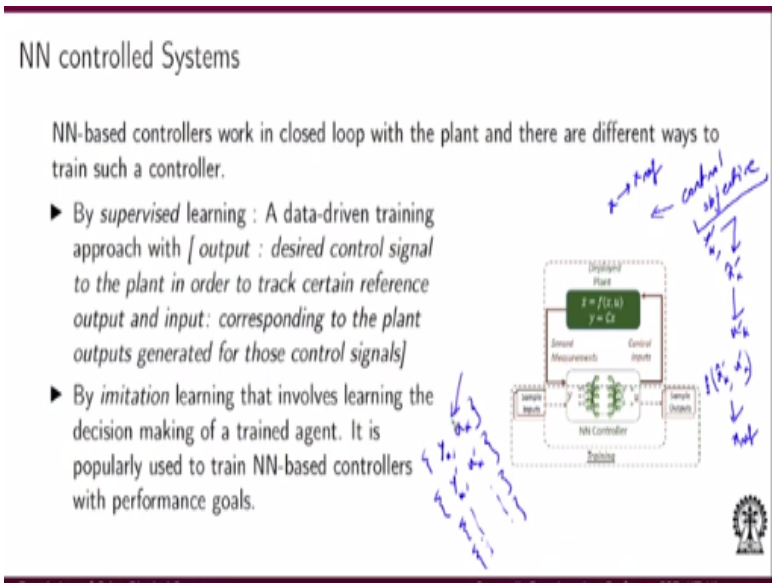
Indian Institute of Technology-Kharagpur

Lecture - 49

Neural Network (NN) Based Controllers in CPS (Contd.)

Welcome back to this lecture series on Foundations of Cyber Physical Systems. So in the previous lecture, we introduced the idea of neural networks and training of neural networks and also we said that well neural networks can be a nice way to kind of approximate the behavior of plants right. So in this section, we will focus on how neural networks can be used to design controllers and what can be the possible control architectures to use for such designs. So let us proceed.

(Refer Slide Time: 00:53)



So uh let us let us discuss some of these ideas here that how an NN control systems can work. So the idea is that well we not only can have neural networks approximating plants, but we can also use neural networks to design complex controllers. One may question that well why do you want to design a neural network based controller? The reasons can be many.

For example, the plant dynamics is quite complex. It may be known, but it may be very complex. And for that the standard analytical methods for doing control design may not work, right. And also there can be other issues like the plant dynamics is not known. So for that you have created an arbitrary neural network based approximation, but even then that that that approximation that you have created, it is again very complex.

So there are several reasons for which in general people will like to have neural network based controllers. So now the question is well, what can be the methods for designing such neural network based controllers. There are different possible methods. One method is supervised learning based method.

So what can be done is, we can we can first create a label data set and you can create this label data set and design a suitable neural network based controller using the data set. Now the question is how will you create the data set? So suppose you have a deployed plant here okay, and there is some control objective and right now I know that the plant's output measurement is some y(k).

And using some expert knowledge, I try to figure out that well, let us let us say the control objective is that the value the state must be moving on to some x reference okay that is the objective. So right now if my state is, my measurement is y(k) and I estimate from that the state is x hat. So given x k hat, I use some expert knowledge to figure out that well what should be my control input, so that given the the function with s k hat and this control input, it will be approaching x_{ref} okay.

So suppose I gain this expert knowledge. Then I am creating a label data pair here right y(k) and u(k). So again, I can take a measurement, some y(k') and I take the estimate and again for that estimated state, I figure out what is the control input which will steer the system more towards x_{ref} right. So in that way, I figure out another data element, right.

So the idea is that the the input, which is corresponding to the plant, the output generated for a control signal that means the measurement okay, and the output is the desired control signal to the plant in order to track certain reference outputs. So using such

expert knowledge, which may be a domain expert or maybe some test input that are being generated, you are trying to create this kind of a dataset.

Note that this dataset creation would be maybe specific to a control objective. And if you want to make it quite general, then you should ideally train and train the train the train the controller with data gathered for multiple variety of control objectives. So that the so that then your training has more enriched diverse set of data and you really learn that well how to give control inputs for a diversity for a variety of control objectives.

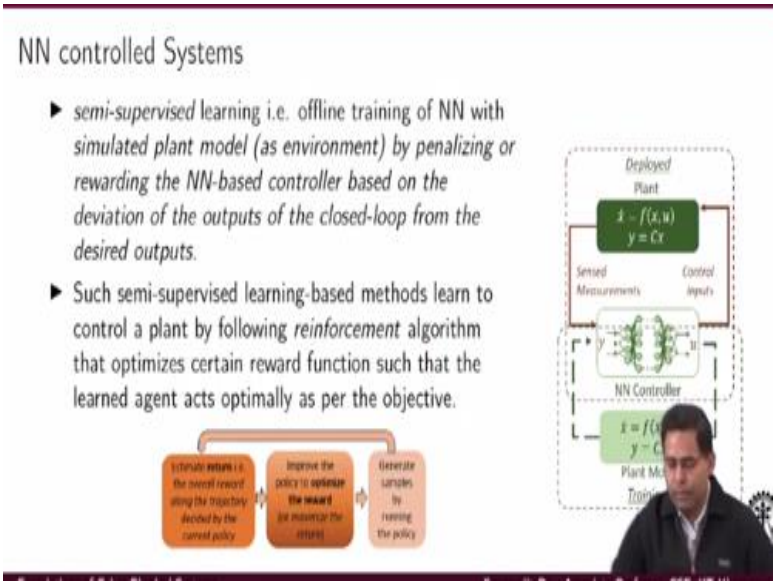
So once this data set is created, then you use some supervised learning technique using which you train a neural network so that given the sense measurement, the neural network outputs a predicted value of u(k) okay. So that is a supervised learning based method. That means you need I mean, this kind of a data driven training approach here. Now there can be other techniques of learning also which can be used.

For example, there is something called imitation learning, which means that well, there exists a trained agent. That means, let us say there is a controller. Let us say let us say I have purchased a blackbox controller for a system, okay. So that is, that is already a trained agent.

And there exist well known imitation learning algorithms where I will initially bootstrap my neural network based controller, and it will start observing what kind of activities this trained agent is doing and over the training phase, this NN based controller will start imitating this trained agent, okay.

So uh so basically given a performance goal, which the trained agent knows how to fulfill my my NN based controller will over over in the loop slowly working, it will slowly learn that well how to mimic the moves of the trainer agent. So that that can be one kind of one class of algorithms from a machine learning which maybe which may be used in this for this purpose.

(Refer Slide Time: 06:12)



Now the other can be this semi-supervised learning. That is, you have some offline training of neural network, which you already do using a simulated plant. Now why a simulated plant because these are mostly safety critical systems, right. So you do not want to have a completely untrained untrustworthy this neural network trying to give control inputs to a real life plant, right.

So rather than that, let us create a simulated plant model. It can be an approximate model, right. And you you you have your neural network-based controller, which will kind of get training in the simulated environment and how will the train training happen? Well, this NN based controller will try different possible control actions and it will observe the effect of this possible control actions.

That means it will try to see how are these control actions performing given if I give some u(k) what is the measurement y(k) that I get? Is that good enough? Is that satisfactory for given the control objective? So and based on that, if the if the move is good enough, then the algorithm will kind of reward the move and if the move is a bad one, then the algorithm will kind of penalize or kind of kind of yeah, kind of penalize the move.

So uh if this is being done, then what happens is that there are this similar algorithms, we specifically what we know know as reinforcement learning based algorithms, where your controller is trying different kinds of moves on the plant and it is figuring out

which of the moves work okay. And based on based on this, it is kind of learning in the simulated environment a model for which the reward gets maximized, okay.

So such such semi-supervised learning methods specifically the methods based on reinforcement learning are extremely popular nowadays and they are getting widely used in cyber physical system design. So using such a reward penalty scheme, and using suitable reinforcement learning based algorithms, which maximize the reward, it may be possible to create such controllers in the simulated environment.

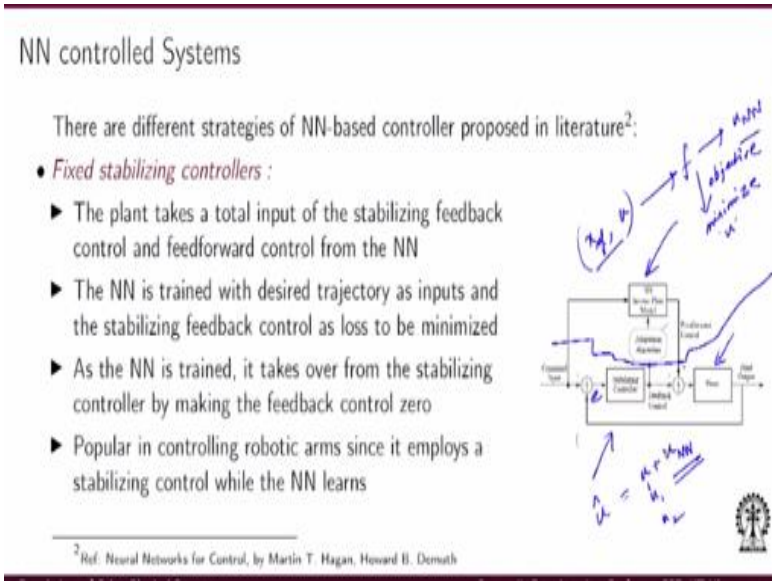
And then you can deploy it in the actual environment and when you deploy it in the actual environment, you can further optimize it. So we have this sample picture here. You have this training happening in this environment and based on that, actions are being tried out by the algorithm. And based on the observations of how the actions are playing out on this model, the actions are suitably rewarded or penalized and using the RL algorithm the neural network is trained in that way.

Essentially, what you get is what we call as a reinforcement learning policy. The controller is now in form of a policy. And this policy is basically represented by a neural network, okay. Now you deploy this neural network here and fine tune it. That means, you keep on refining this policy to policies where the rewards are maximized through this kind of a loop.

So what you do is you deploy the policy and you observe generated samples by running the policy and then you estimate the return that is the overall reward along the trajectory that is decided by the current policy and based on that you further improve the policy. So we are not going into the details of this. There are well known RL toolboxes that exist. They also exist inside the fabric of MATLAB.

We will show through our some example tutorials that how such RL based controllers can be created and how such RL based controllers can be deployed in practice. So well we understand these are the different ways in which you can create a neural network based controller, but the question is what will be the deployment architecture for such neural network based controllers. That we we must figure out right.

(Refer Slide Time: 10:21)



So let us see that what can be different possible strategies of employing the NN based controller. So let us pick some of the example strategies. And then we start with something called fixed stabilizing controller. So let us see what this is. So here the plant is in loop with a stabilizing controller. So this is something which you already understand right, whatever is on the this side of that line.

So this is my standard control loop. But now we say that well, this is there, but let us create another controller here. And the idea is that well, the plant takes a total control input and that total control input is my normal feedback control input plus a control input coming from that network here, okay. And this neural network, which is kind of outputting this u NN it is taking as input this x ref the command as well as this feedback control signal u.

So with this input is generating this outputs, and it has been given an objective which is the loss function again, and the objective is to minimize the value of u. So essentially, this NN based control, suppose this is getting trained with more and more data as the system is running. So as the system is running, it is trying a move, and based on that it has an objective.

That objective is that well, you should reduce u. You should give such a u NN, so this is the overall u. And objective is that you should give such a u NN, so let us say right now the u 1 is there for u. So with u 1 you give a u NN such that the plant output comes

here, and it creates an error value. Due to the error value, whatever the u(2) the stabilizing controller will generate must be smaller than u(1).

So that in the next iteration, u NN will increase. And if this will continue, and the value of u getting out of that feedback control here should get should get refined, and slowly approach zero. So that is how I have set up a neural network with an objective here. The objective has been clearly defined that try to make the feedback control here zero.

And that means you should figure out some value here so that when this value and this value is added up to the plant, and then effectively, this feed stabilizing controller gets such an error value here, that it does not generate a significant u. So that is, that is how the, so you can understand. That is the objective I am giving it to this.

And if this model is effectively able to figure out what that objective, then it will train itself in such a way that it happens. And when that happens, what will be the case. The case will be that well, this will almost approach zero, and this adaptation algorithm will receive the training algorithm for this network, which is aware of the objective.

If it is successful in the objective, this becomes zero, and then it will kind of cut out this part of the loop right. Because essentially, there is nothing here. So you you do not have anything. So this entire part is not there. So then what remains is the NN followed by the plant and it is producing an output, which is kind of identical with the input. So that is what I want. I want this entire thing to be like an identity map.

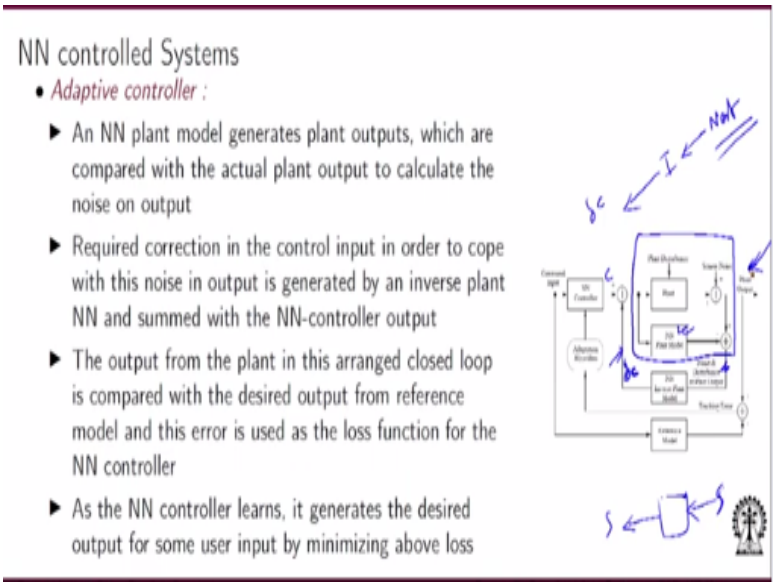
And that can happen in real world if this NN is like an inverse of this plant model, right. So that is why we call this an NN which is like an inverse plant model. So that is what is the objective of this NN and you fix some NN here and you give it that objective that in the long run you should try to generate enough I mean, you should gather data.

And you should do some training in such a way that you are able to generate control input so that this feedback control output becomes becomes zero. So that now the NN will take over and it will it will do the complete control just in a feed forward loop without any feedback.

So these are simple idea, but it may not be realizable in all situations because you can understand, you essentially want to create inverse of the plant model and that is and once you are able to create that, then you do not need the feedback loop. So that is that is your target here. But it is popular in systems like robotic arms, because you are starting to use it along with the stabilizing control which is already there.

So you have a fixed stabilizing controller and in the loop you are bringing in this NN which is trying to create an inverse of the plant model and in the long run if it is able to do that, then the stabilizing controller goes out of the loop.

(Refer Slide Time: 15:31)



So now let us check this next architecture. This is an adaptive control design. So it is not a fixed stabilizing controller. So what you have new here is you not only have the controller which is a neural network, but there are actually three neural networks in this architecture. You have the original plant and you have a neural network-based plant model.

So we are assuming that this has been created with suitable data generator generation okay. And that is kind of a good approximation of the plant's dynamics, okay. So the NN plant model is generating plant outputs and in this deployment architecture, what is happening is these plant outputs will be getting compared with the actual plant output. Now how does that help?

So if you assume that the neural network based plant model is well equipped to mimic the dynamics of the actual plant, then it is missing only one thing. That is the disturbance that is getting getting added to the plant in terms of process noise and sensor or measurement noise. So in the output if you compare the real plant output and the NN based plant model's output the difference essentially will be these noises only right?

So now the question is when we have created our original neural network based controller, it is it is trying to generate some control input which will steer the plant model towards the control objective right, towards some command some some reference, okay. But it does not know how to deal with this noise.

So that means essentially the control tasks refinement can happen here in a way that well if I can modify the assuming that the original controller is perfect enough to do this, this neural network based controller knows exactly how to drive the plant to a to an objective given there is no noise, we are assuming that is known to it okay?

So then for this residual noise, we need to cancel it, out we need to handle it by doing suitable modifications to the control input that the NN controller is generating, right. So we want to add or subtract something to the NN controller's output and this will be some amount of control which is just good enough to modify the plant's response and compensate for whatever extra response is there due to the noise, right.

So we filter out the noise using this part of the scheme. And then we want to generate the compensatory control. The compensatory control if you want to generate like we will use the previous idea of creating an inverse plant model. So assuming that we are able to create an inverse plant model, so if we give the inverse plant model the extra disturbance input it will output what the extra control effort which is required.

So it will output the extra control effort which which is required to compensate for this disturbance here, right. So whatever is the disturbance which we can estimate here, let us say noise estimate, we give it to this inverse model I, we get some some delta c which is the amount of control that is required to to kind of compensate this, right.

So then this delta c comes and it gets subtracted from the NN's original control values so that now with this modified control when the real plant will work, it will take care of the disturbance and it will try to make in this loop's design it will try to make this difference zero, right. That means I mean I mean the objective would be that to cancel out this noise so that I mean whatever the noise is coming that is kind of suitably suitably compensated here.

So just to recall again, the NN controller generates control just for the plant, but the noise is there. The noise is kind of estimated through this kind of plant NN based controller structure assuming these are good approximation. So this noise that is there in the observational output right, I want to generate a control so that what happens this I mean, whatever deviation is happening in the plant output that will be that will be handled, right.

So for for handling the deviation what we do, we apply this inverse model. So we give this extra deviation that we have here right, we give this extra deviation that we have here in the output due to the noise and we generate the extra deviation we should have in the control so that once it is added to the control and fed back to the loop, the in the plant output we do not get to see that effect.

I mean that means the plant output should be reaching the original objective even in presence of this this noise that is there, right. Now is that good enough, not really, right. Because eventually what do I want? Eventually I want that well, the NN must work nicely even in the presence of this this noise and everything else, right. So this command input needs to be compared with the plant output.

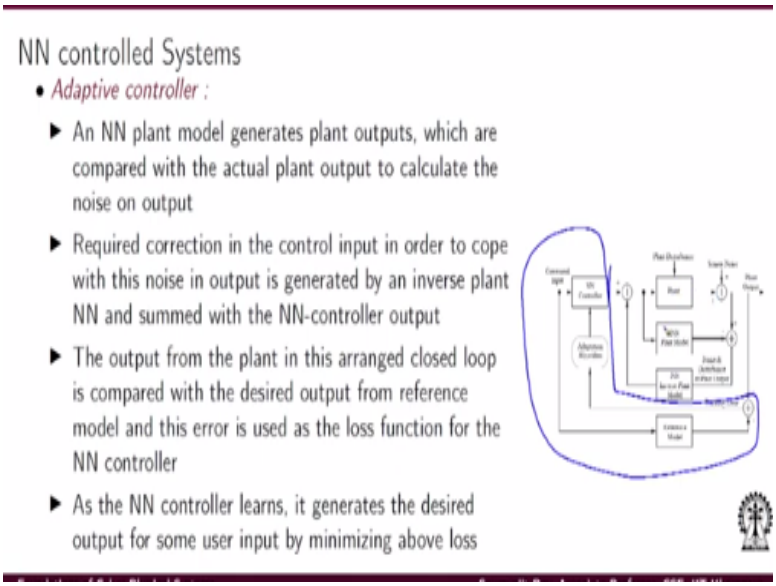
So this command input is being compared with the plant output and this tracking error is being created. And this tracking error is now being given as an as an control objective.

That means, is given as an objective through the training algorithm for this NN based controller so that in the loop when the tracking error is getting generated as a function of the command as a function of the command input and the neural network output the

NN based controller will try to update its weight in such a way that this tracking error is minimized.

That means, this NN based controller will update itself in such a way that it does the control in a nice way. So basically the NN's the NN's update loop is given by this.

(Refer Slide Time: 21:47)

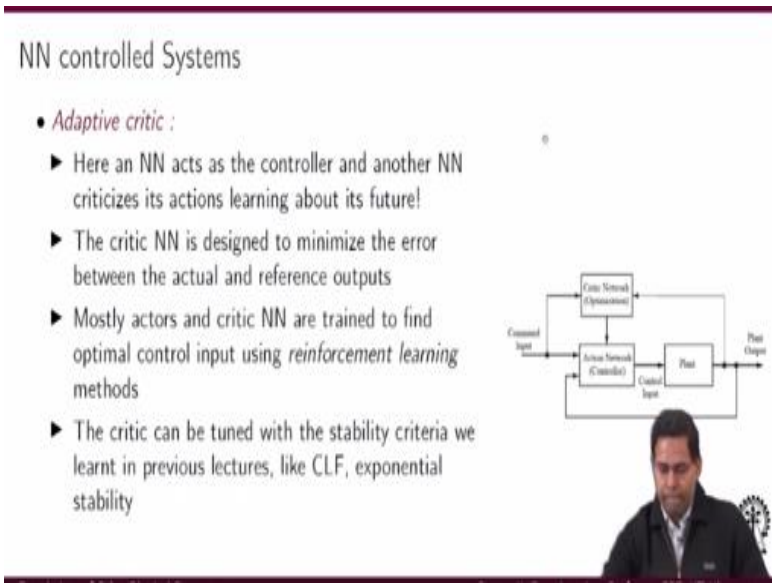


It is this neural network's update loop because this will get be getting tuned so that the tracking error is getting minimized, okay. But like we said that this update loop does not take into account the the effect of noise. So for that, because it is assuming that the ideal plant is there. So for for taking that into account the plant is put in the loop with its with its inverse model with its ideal model.

And then the noise that is coming into the output due to the physical disturbances they are getting generated and then to get the extra control control output, so that the plant output again goes to whatever we wherever we want it to be in spite of the noise, that compensatory control will come here and we will get the desired output. So this is a complex architecture, but all these blocks have got their own individual motivations here, okay.

And eventually what should happen is as the neural network is learning, it will generate the desired output for some for some user input by minimizing this, this this loss here.

(Refer Slide Time: 23:05)



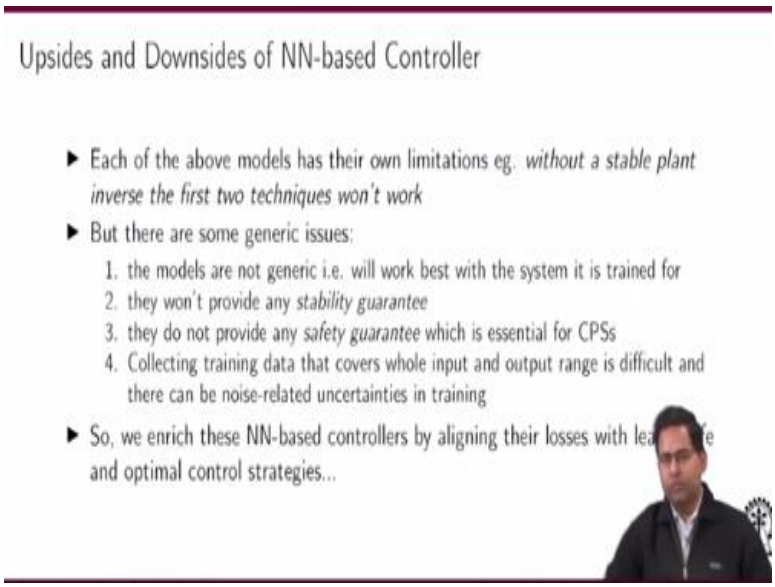
Now the last architecture we will talk about here is called the adaptive critic architecture. So here what happens is one neural network will act as a controller and another neural network will criticize these actions learning about its future. That means the critic NN is designed to minimize this error between whatever is the actual and whatever is the reference output.

So mostly these neural networks that is presenting the actor and the critic, they are they are trained to figure out the optimal control input using a paradigm of learning methods called reinforcement learning methods. And also note that the critic here can be quite flexible.

That means, for the critic, it will have a, it will be given the objective in terms of a loss function and that need not be only to minimize the error between the command output command input and the plant output, but it can be more varied like it can be the stability criteria, it can be also the the the control Lyapunov function-based performance criteria of of exponential stability and it also can be some some safety criteria.

So the critic may evaluate those criterias and accordingly it may minimize the loss function, but then the loss function needs to be suitably designed so that it nicely captures all those criteria. So that is one way of using an actor critic paradigm for doing the control design.

(Refer Slide Time: 24:31)



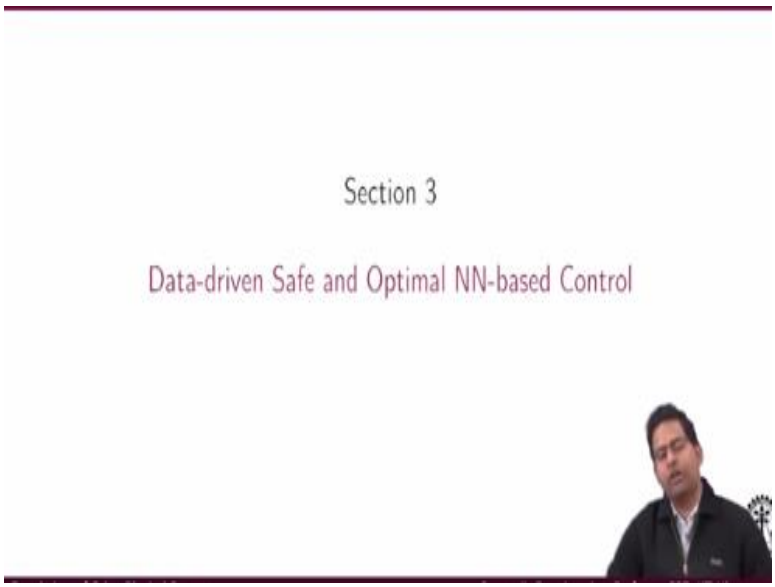
So overall if we try to summarize that well, in each of these situations that we have discussed, there are some some downsides also, right. For example, in the first two methods, we need a stable plant inverse, otherwise the methods do not work right, and that inverse may not be computable in many cases. And also in general, the models are not generic and they will mostly work for systems where they are trained and also they do not have any guarantee.

Like whenever you are doing a analytical control design, you get mathematical guarantees. But here in the NN based control designs you do not have safety or stability guarantee, right. So the point is that well, can we also get some guarantee while using a neural network. So that is a very important topic and we will try to see that how such neural network based safe controllers can be designed.

And the other issue can be that when you are collecting such training data, I mean you need to have a very diverse data set. That means the data set should be covering most of the regions in the state space. Because if it is not so then the data data set will be biased. And if the neural network is is faced with input situations for for a completely different region in the state space, then the outputs can be due to a very much over fitted neural network.

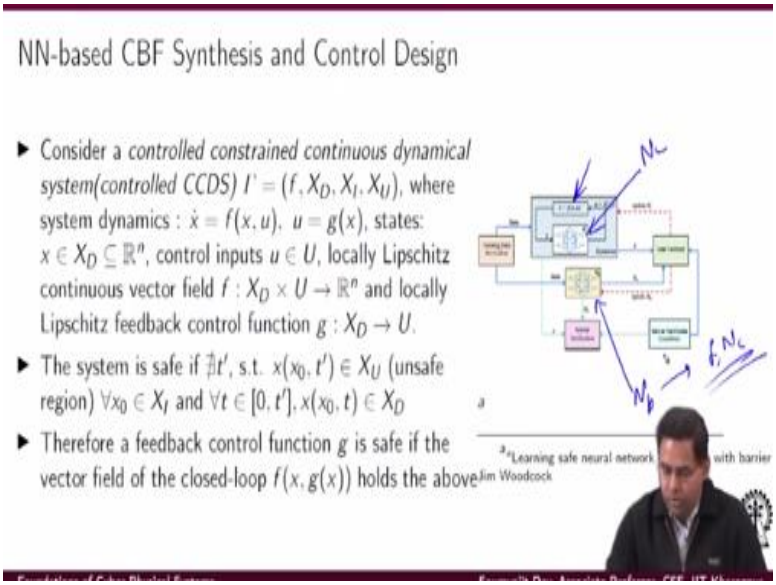
The outputs can be completely different than what we really want them to be.

(Refer Slide Time: 25:57)



So okay, we come to this last section of this topic here, which is that well, how to use say use use methods where of NN based control deployment where we are also tribal to kind of bring in other constants like safety and performance.

(Refer Slide Time: 26:14)



So let us consider a constrained the continuous dynamical system or CCDS like controlled constrained continuous dynamical system. So the plant function is f. X_D denotes the domain of operations of the system. X_I denotes the initial states and X_U denotes a set of unsafe states for the system okay. So you have the state of control inputs given by this capital capital U and you have a locally Lipschitz vector field so that which is defined over this domain cross the set of control inputs.

And also you have a feedback control law g which is defined from the state domain to the control input domain. Now we want the system to be safe. That means, the system should never reach this set X_U of set of unsafe states okay. So we want that to happen. So how to do that? If you recall that earlier we we handled safety using these concepts of barrier functions.

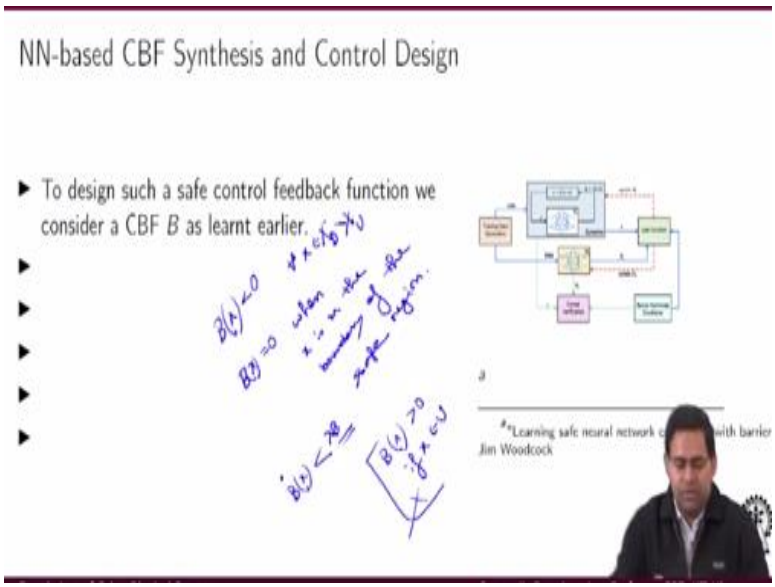
But there also we had a problem right, like creating barrier functions may be difficult. For a sufficiently complex system suppose I am given a simple plant dynamic C model and I am given a corresponding controller. It may be easy to figure out barrier barrier polynomial. And it may be easy to show that well this plant controller combination is stable because because of this, because this polynomial satisfies the required barrier properties, right.

But it may not be easy in other cases. So the question is well, using barrier functions if you is nice, but you need to get one barrier function. Even getting a barrier function for a complex dynamics is difficult. And it may be more difficult if the if the dynamics is nonlinear, very complex, etc., etc.

So in this work learning safe control, safe neural network controllers, the people associated they had a nice proposition, which was that well, let us not only learn a controller, but let us also learn a new barrier function for that controller and let both these things happen together. So this is their target architecture. You have the plant. You learn a controller for the plant and you also learn a barrier for the plant.

So this is N_b the barrier, this is the N_c which is the controller, okay. And then you do a verification that well this barrier is really making the plant controller combination safe. So this barrier is safe for f, N_c. So let us let us see how that can be done.

(Refer Slide Time: 28:42)

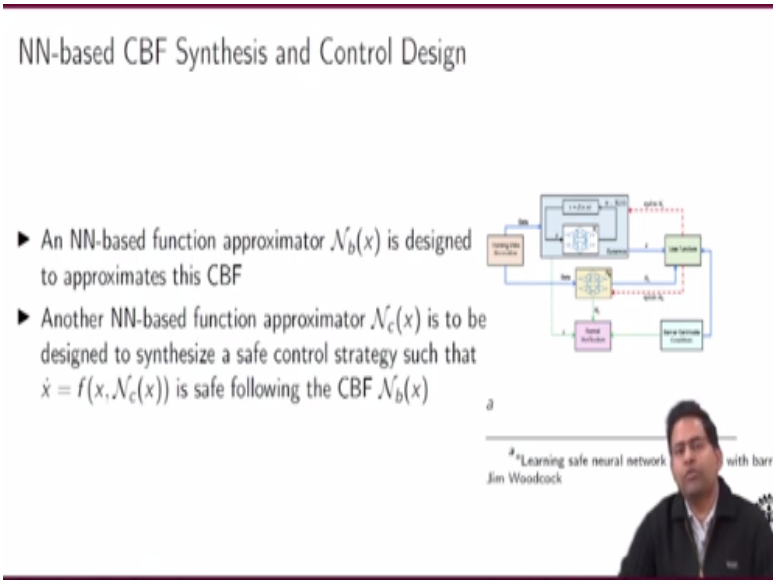


So first, let us recall that how the barrier functions were supposed to be. So for any point x this was supposed to be negative for all x in the domain apart from the unsafe region and the barrier will be zero when x is on the boundary of the safe region and the barrier's derivative will be less than some positive value, okay.

I mean like we discussed in case of control barrier function, it should be lambda B right, so that the barrier's value may increase up to zero, but the moment it approaches zero, the rule will force the barrier value to decrease, okay. And the barrier's value can be greater than zero if x is a value in unsafe. And this is where I do not want the system to go, okay. So that was our control barrier function definition.

So just make a point here, they maybe, you will find treatments of barrier, control barrier function specifically where people use the opposite science, but here in our entire discussion in this course, we try to keep it exactly same as the real barrier function and also same as the Lyapunov function so that things are really uniform, and we particularly like our definitions, and they work nicely. So let us see.

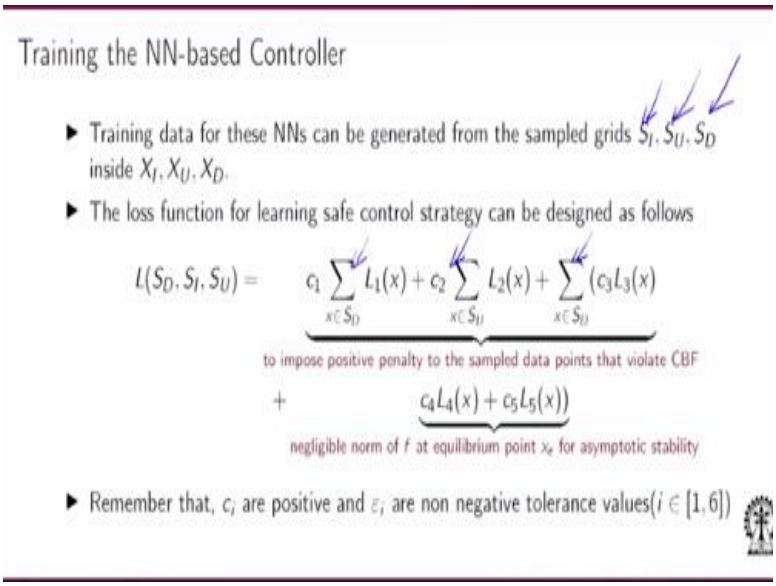
(Refer Slide Time: 30:33)



So we can create this kind of function approximator for the barrier function, and we can create a function approximator N_c. Both are neural networks for the control strategy and what we want to say is that we want to prove that well, now this is my closed loop right, x dot equal to f of x comma the U getting generated by N_c is safe, and its safety is established by this barrier function N_b, right.

So that is the idea here that well, designing controllers for complex systems may be difficult. It is even difficult to get a barrier function for a complex system. So let both of them be neural networks.

(Refer Slide Time: 31:12)



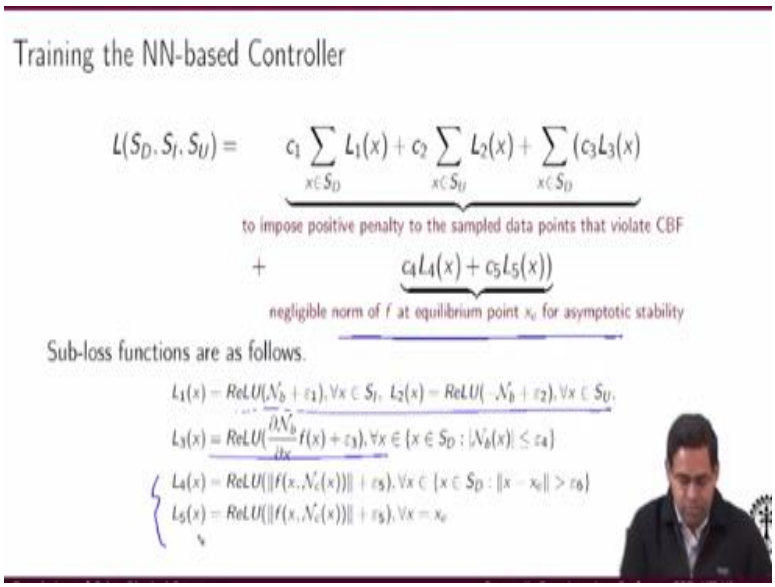
Question is how do I train this neural network? So the idea is that well, you have to generate training data for these NN's by taking suitable values from these different

positions of the state space. You have the initial region, you have the unsafe region, you have the entire domain, right. So you have to you have to sample data from them. And then you create a complex loss function with several components and each of these components addresses some requirement.

So these first three components of the loss function would be with respect to the different conditions in the barrier function. If you recall, my barrier function requires that well, if it is inside the domain, I mean I would love to be at positions where it is negative. If it is unsafe, then it is positive, but I do not want to be in the unsafe region. And as the moment I go near the boundary of the barrier function, then it should be approaching zero.

So that was the idea right. And not only that, suppose I also have some stability requirement, I have some performance requirement. So those are the control Lyapunov function like conditions, those can also be captured using some other components of the loss function. So we are not going into too much details of this. You can figure out the details by reading the associated paper (32:25). But we do not need you to do that for the course. This is only for exciting you and trying to tell you that how these things can be done and how this works. We are not dealing with the why part too much here.

(Refer Slide Time: 32:37)



So for this problem, this is how the loss function is. So there are five components. And as you can see that I was as I was saying that we are we are considering a value-based activation, and for the different parts of this like we are defining different epsilon based

slack variables, and we are saying that well these components of L_1, L_2 and L_3 are corresponding to imposing positive penalty on the sample data points, which violate the barrier function.

I am not explaining how. If you are interested, you can figure that out from these notations and with increased understanding of value. If we want, we can actually discuss that in our live sessions or in our or in our online forums that why and how these loss functions are working. We are not explaining it for the general audience, it is only if you are interested, okay.

So there are three components here and these three components will create a positive penalty in case the sample data violates the CBF. Similarly, the other two components which are these, they will create penalty in case the stability conditions and those conditions are violated. So they are being modeled like this here, okay. So fine, if we now go ahead this, so that is how this works.

But the question is, well, by using this loss function, I can make training and I can create the barrier function and the controller, right.

(Refer Slide Time: 34:09)

But is that good enough? Is it safe? How do how do I justify? That is that is something. But that is a bigger question. Before that, we also talk about some other approaches which require which are there for NN based control design. So this is another approach, which we can see in this paper here.

So in this approach, what they do is they say that well, let us assume that I am having a control barrier function with safe controller known to me, okay. So that is known to me, but it is not practical because it is not an optimizing controller. The only thing it optimizes is that well, it makes all move safe, right.

So I can create an NN based controller, because NN based controllers can be good, because I can give them multiple objectives like we saw using the previous loss function, I can give them some performance objective, performance objective 1, performance objective 2, several some power consumption objective, things like that, okay. And then I do the, so those are my objective.

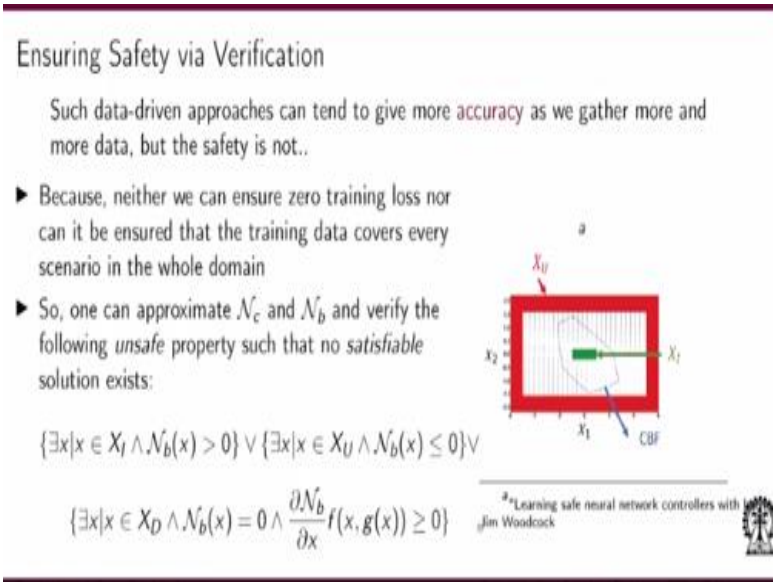
But then I wanted to learn safe moves also. So what I can do is I can make this NN based controller learn initially the safe moves using some imitation learning technique by observing the moves which are safe from the CBF based controller, okay. So that can be also one technique, that you observe the CBF based controller and you learn learn the moves. There can be another technique.

Let let let this NN based controller learn with respect to the performance objectives I gave, and then I figure out which are the moves which it is making which are not good ones and let us try to filter them out, but we will talk about that. So this imitation based learning techniques also have some limitations like they may not face some sample inputs in real life, because, that expert knowledge is not available in its database.

Like we have not given some training inputs from that part of its domain. But such issues can be eliminated by regularly updating the training database and retraining the NN based controller, but this is this is also an interesting approach like let it have its own performance-based objectives, but that is not all. Let it use the learning method be such that it learns imitating the safe controller.

Then it will not make moves. I mean, it will discourage moves where in the in the greed of satisfying its objective, it makes unsafe moves. So that that can be one kind of objective.

(Refer Slide Time: 36:40)



But so there can be several such methods using which you can do NN based controller training, NN based barrier function creation. But finally, we will like to, once I have those controllers, we will like to know that well they are really safe, right. So how do I do that? Note that whatever NN based controller and NN based barrier functions we train, eventually I can write them out as these polynomials because I have them as a neural network.

So I can write them out as nice polynomials, right. So then, what we can do is, we can use these polynomial expressions and use some solver to write and see that well, does this this polynomial based representations of these neural networks, do they satisfy, do do they do they satisfy value assignment, which is unsafe for the system. Now what are the possible such things? So I am writing a logical statement here.

So it may happen that if x is in the initial state, and the barrier function's value is greater than zero, right? So that is not, that is that is an unsafe assignment. So this should not happen. Then it may happen that while x is in the unsafe state right, and then N_b is less than equal to 0. That should not happen because if I go to the unsafe state, N_b is supposed to be greater than zero.

So each of these conjunctive clauses that you have, I mean, overall this is a disjunction and inside the disjunction, you have multiple conjunction of clauses here. So it should not happen that any of these clauses is true, right. For whatever is the value of x you feed, any of these clauses need not be true, right.

Because if that is true, then there is a situation in your design, such that your neural networks do not satisfy any of the guarantees that we talked about. Because you can see that each of these clauses here capture some of the guarantees.

This is about the barrier function being positive in initial state, this is about the barrier function being negative in the in the unsafe state, this is about the you you are approaching the safety boundary here when the barrier is zero, but then the gradient is positive. This should also not happen, right.

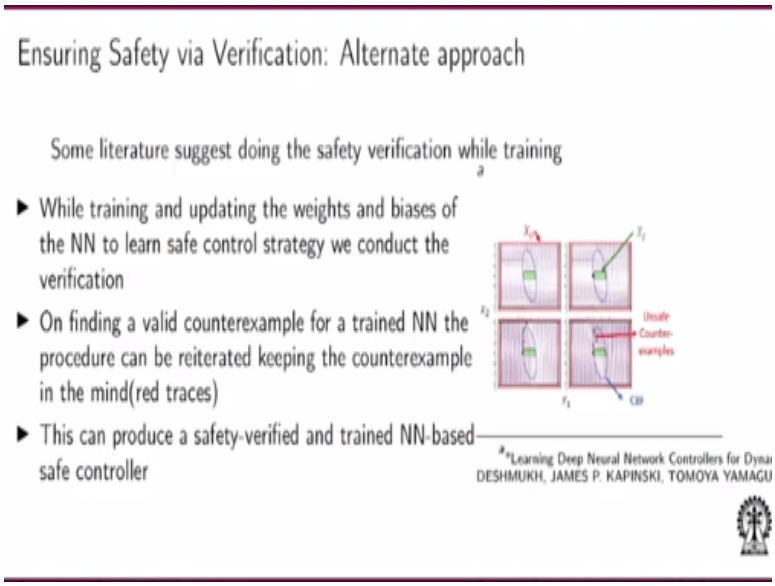
So these are the different barrier conditions and we are writing a set of logical conjuncts right, logical conjuncts internally which are again disjuncted. We are writing this logical formula here, which captures this unsafe property. And we can check using a solver that well if this logical formula is satisfiable.

Because if that is satisfiable that means it is possible to be somewhere in this domain of x_1 and x_2 here so that I do not have my barrier function properties holding for that value assignment. But if my solver whatever, I mean you can use a SAT solver like ISAC in case the system is nonlinear here. So if my solver is telling me that well, this formula is unsatisfiable, then that is a good news.

Because that means there is no possible value of x_1, x_2 in the state space where this formula will have some part true which makes the formula true, okay. So that that is how a formal verification based technique can be applied here using a solver technique here. So that through which we are able to show that well, whatever NN based controllers we have trained, they are safe with respect and the and the barrier function is also fine.

The barrier function satisfies the properties and since the barrier function is defined over the closed loop activation of the NN and the plant dynamics, so everything is fine.

(Refer Slide Time: 40:13)



So there is also other techniques like doing the safety verification in the loop during the training. So while training and updating the weights and biases of the NN you can see that well I mean, suppose you have trained a NN and at some intermediate point you stop the training and then you do such a formal verification run. That means you write such kind of a nice property of safety.

And you check using a formal verification using that well, does this current neural network satisfy the safety property or not. Suppose the the solver gives you a counter example. That means it is telling you that well, if you are in this position in the trajectory, then whatever the neural network is telling you to do, whatever control control control move it is suggesting, that is taking you to an unsafe region.

So that is a bad that is a counter example, right. What you can do is now you can give this last move or or those, I mean there are several techniques to do that. Now you can refine, if you are using a RL based method, you can actually refine this policy that you have of the neural network based controller by removing these kind of actions.

So so there are techniques using which I mean, you can I mean, you can prune the neural network and you can eliminate such actions, whichever are showing up as counter examples when you do the verification, fine. So there are works addressing this. So

basically what we are saying is you you get a neural network, then you formally verify, you see what are the counter examples, use those counter examples inside your training algorithm and make them disappear.

That means, the training algorithm is told that well, I trained you and then I figured out that you are making the system go to this this unsafe regions, then then the training algorithm will will be running again. And there are techniques of reinforcement learning based policy refinements and stuff using which you can retrain the network and make those actions disappear.

And this can and then, but that does not mean that it becomes safe. Now you will get a modified neural network. If you are running this neural network again, you may see that well again there are counter examples. So this will keep on continuing unless you arrive at a situation where there are no such examples. So that is how it works for these systems. So I think we are done here. With this we will end this lecture. Thank you for your attention.