NPTEL

NPTEL ONLINE CERTIFICATION COURSE

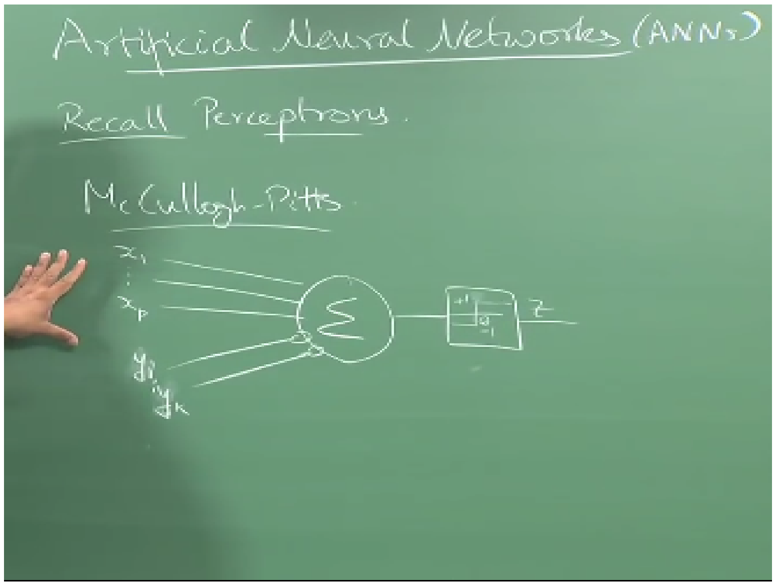
Introduction to Machine Learning

Lecture 32

Prof. Balaraman Ravindran
Computer Science and Engineering
Indian institute of technology

Artificial Neural Networks I – Early Models

(Refer Slide Time: 00:14)



Okay so we have been having discussions about neural networks off and on right the last time we had a discussion about ANN$_S$ was went and we did perceptrons right so basically so the whole class of solution methods which are lumped under ANN$_S$ are artificial neural networks were primarily inspired by trying to emulate the brain architecture right yeah, so then after a while the field split into two right one class of researchers who are looking at neural networks as just computing elements right trying to interpret it in terms of linear algebra right and partial analysis and other mathematical tools and then trying to understand what computing these artificial elements were doing.

And others who are still trying to make the neural models biologically relevant right, so now the communities have become fairly divergent right, so there are this set of people neuroscience people who are trying to build computational models of the brain right and then there are machine learning people who just use neural architectures without worrying anything about biological relevance right, so yeah we will take the latter approach right there will not be looking too much about the biological relevance of the neural networks I will just try to understand it in terms of computing.

So do not two people have ever been seen a neuron synapse did not write all those pictures so you do not have to really tax my drawing skills right so I will not draw anything on the board about the neurons right, so the whole idea behind all these biological neurons is that it is pulling in inputs from a variety of other neurons right and some computation is goes on within the neuron and there is a result of this it might actually fire right in which case it will cause it another input to be activated for yet another neuron right or a set of other neurons depending on who they are connected to.

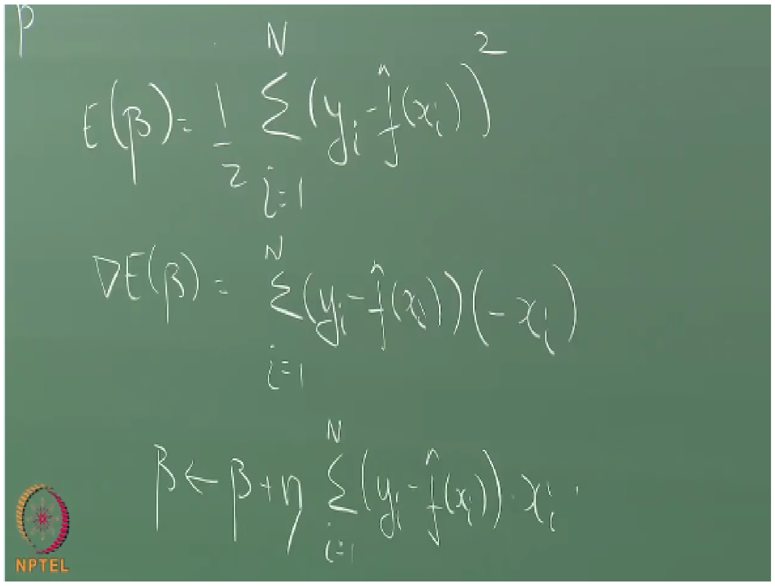
Then these neurons are all connected in a very complex networks and even though it is a very simple computing element each neuron can be thought of as a very simple computing element the whole the fact that they are connected together in a large network allows them to do all kinds of cool things right, I mean if you really want to know what a neural network can accomplish they stop and think what you can do right, so one of the earliest all of the earliest models of a neuron was the Mcculloch Pitts model right.

So it is a very simple model so it essentially it has a Σ unit right so it has a set of inputs right and then it has not set off inhibitory signals okay right, so it has a set of inputs right and a set of inhibitory signals if we what we are talking about artificial neural networks and I started explaining the Mcculloch Pitts model so you have a simple unit which has P inputs right and then it also has k inhibitory inputs right, so what does this neuron do is essentially adds up these k inputs right.

And if the if they exceed a certain threshold right some threshold θ then it will output a 1 if it is below a threshold θ it will output - 1 or a 0 depending on how you are encoding it right and the inputs are all considered either to be 1 shot – 1 shot 0 is depending on how we encode it right what are these inhibitory in inputs for if any one of these inhibitory inputs was one there was no output no I 0 okay yeah if any one of these inhibitory outputs is inputs is one then there will be the output will be - 1 okay or 0 depending on how we were encoding it again right.

So that is the basic Mcculloch Pitts model but then just to give you the historic perception right and then what happened people modified this way to propose the perceptron right.

(Refer Slide Time: 06:21)



p
$E(\beta) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{f}(x_i))^2$
$\nabla E(\beta) = \sum_{i=1}^{N}(y_i - \hat{f}(x_i))(-x_i)$
$\beta \leftarrow \beta + \eta \sum_{i=1}^{N}(y_i - \hat{f}(x_i)) \cdot x_i$
NPTEL

So if you think about what the perceptron does it is essentially saying that right so instead of doing x_1 to x_p and adding it up I am going to multiply it by β_1 to β_P so X_1 β_1 +X_2 β_2 +X_3 β_3 and so on so forth now if this entire thing is greater than a threshold okay I will output a +1 right the entire thing is lesser than a threshold I will output -1 so what is the threshold meet or not right so another way of doing it is to actually add a add another input label that β_0 I mean way is that as β_0 and put a 1 and then and right okay.

So you could think of it that way right so usually it is written without the - sign right that essentially means β_0 will be a negative quantity okay so this is essentially what our perceptron this you can see it is very closely related to the original Mcculloch Pitts model except that there are no inhibitory inputs and they the actual inputs are weighted okay and we all know how to estimate the parameters of the perceptron right how did we get there we use the gradient descent rule right.

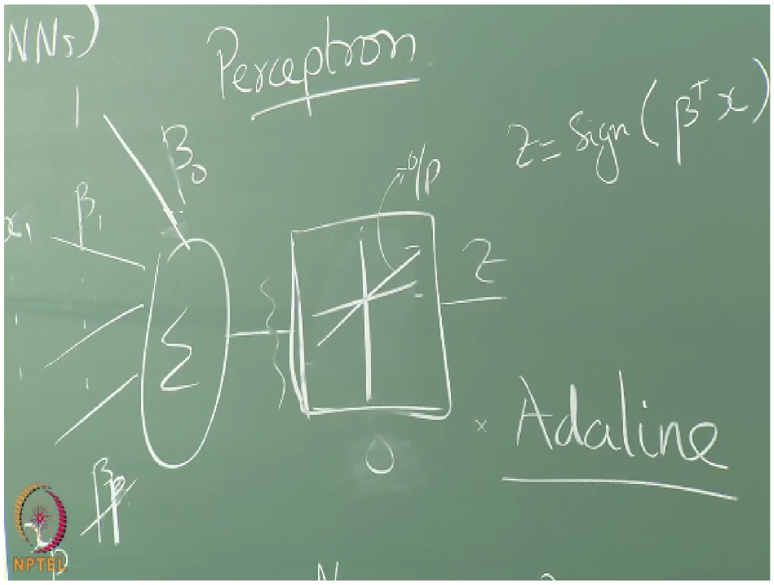
We started with that error function and then we took the derivative and quite look like the perceptron ruled as it does not quite look like the perceptron what it do the perceptron update so wherever there was a missed classification what did you do with this we just added those data points to added those data points times Yi to so we did essentially read X_i Y_i to the input right we did not do anything like this right no right, so what we essentially done here is minimize the squared error all right.

So this is another way of training perceptron is it something wrong here what is wrong here yeah so with the with the perceptron training so we did something very different from whatever I have done so far so it is perceptron training algorithm we had a very different objective function that we were optimizing right, so what were we trying to do people remember that you have a quiz like two days away should have revised all of this by now yeah minimize the mean you might see what minimize the distance to the hyper plane of the misclassified points right.

And the way you can minimize it is we get a distance of zero total distance of zero is when all the points are correctly classified right, so what we had actual objective function we wrote down was we are trying to minimize the distance to the hyper plane of the misclassified points we remember we wrote it only over the misclassified points right so in this case what we are doing you are writing a plane with plain old squared error function right but this is actually the squared error there right.

If you think of it otherwise this is anon linearity right this is a non-linearity the threshold is a non-linearity right I can just take the derivative of z with respect to x it does not make sense I cannot take the derivative of z with respect to x because that is a nonlinear function of x right.

(Refer Slide Time: 12:09)

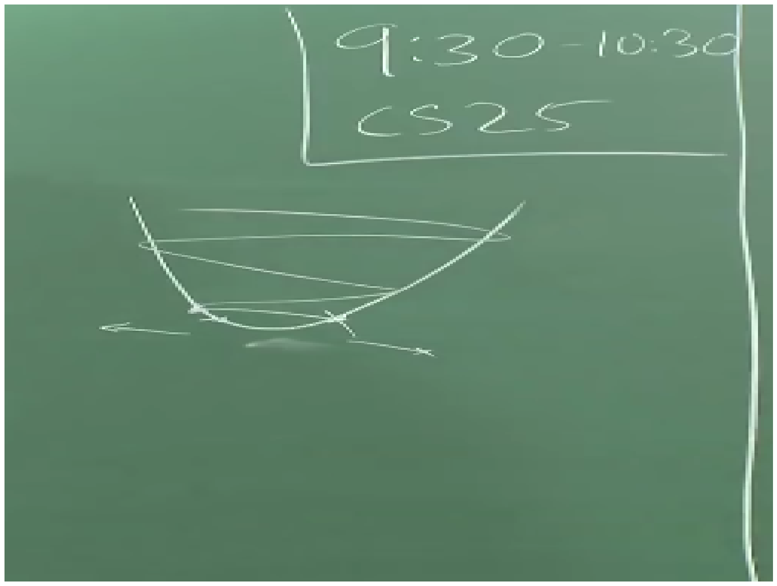


So z is what right and sign is a nonlinear functions non differentiable so I cannot really take the derivative. So in fact what I have written down here is for a slightly modified model of the neuron where the output is taken at this point okay the non-linearity is not there, so one way of thinking about this is that is the output like it is like a straight line right, so whatever comes as the input it will be produced as the output right, so fool yourself for a minute that this line has a slope of 1 okay.

So whatever comes as the input will go as the output right in that case I can take the derivative of the output with respect to the input does it make sense right, so this is not the perceptron by the way they are sometimes called the but what is either line stand for any guesses adaptive closed linear adaptive linear so it is called adaptive linear units so they are Adaline right so on the way you train Adeline is just you straight forward gradient descent right and you end up with this okay.

So people are familiar with gradient descent right I do not have to explain gradient descent to people everyone familiar with gradient descent right okay, so why am I using η here step size yeah why do I need a step size okay so why would what would produce oscillations okay great yeah.

(Refer Slide Time: 14:42)



So let us look at it in a 1 dk may suppose I have a function that I want to optimize right let us assume that I cannot measure the gradient properly and I am actually making estimates of the gradient right so I am somewhere here right and I find the gradient what direction is the gradient here that way hey that is the way I have to change x to go up right and what they have to do I have to move in the opposite direction.

So if I take a large step in the opposite direction what will happen is I will actually end up this same look I will end up somewhere here now again I will I the x the gradient is in that direction here again I take a large step I would end up here like I could keep going back and forth then I might not actually converge okay that is it make sense in fact it is even worse I could go back and forth and I might even diverge if I am taking very large steps right so that is why I have to take small steps when you are following the gradient right.

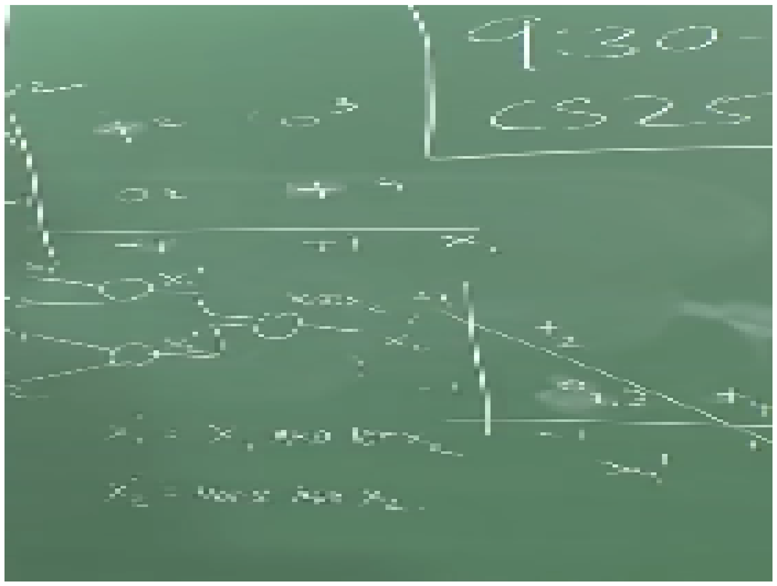
And this these kinds of methods are called stochastic gradient why no it is essentially the so if you think if you just think about it a little bit right where do these excise come from for you not from some underlying distribution you do not know right so what you are essentially doing is there is an error function which is the error function of β on the entire input space right what I am really trying to minimize is the error on the entire input space right but I cannot compute that gradient because I do not have the data distribution I do not have the the entire data set available to me right.

I only have a sample and the sample was chosen in a stochastic fashion, so if I am trying to minimize the error with respect to the entire data so what I am actually finding out here is some sample radiantly it whatever sample data said was given to me I am finding out the gradient with respect to that sample data right that is one part right the second thing is typically I end up doing this one data point at a time instead of doing it n data points right if you I spoke about this in perceptron, so when I am doing it one data point at a time that is essentially not even the correct computation of the gradient.

So even given the data that is available to me I am only doing it one data point at a time that means I am actually doing some incomplete computations of the gradients right and if I take really tiny steps in the direction I am hoping that on an average I will move in the right direction so if I had computed for all the data points and taken a step then well given the data that is the best direction I can move in what if I computed one point at a time then I have to really take tiny steps.

So that I am sure that on an average I am going in the right direction okay, so that is that is idea here, so remember this is useful for the rest of the class so what is the problem with the perceptron people remember let goodness in something as simple as Excel because it was not linearly separable right.

(Refer Slide Time: 18:48)



So it is not able to solve this is there some way you can think of solving this let us call this x_1 and x_2 should be -1 how do I move it to a different space change the basis function right I cannot rotating the axis would not be sufficient for me to still be not like I still not be linearly separable right I think I said that right so what do we do now too many x's on the board yeah so I can define a new feature but what do what should I define will it one x times okay x_1 x -x_2 and x_2 into -x_1 okay.

So now what will happen if I do the projection of this data okay, so we are at this point go -1 -1 -1 is it, so where would 1 go -1 -1 where would 2 go 2 is here, so there will be -1 x -1 will be +1 okay then this will be +1 x +1 will be +1 is it okay does not sound very promising what about 4 is also + 1 what about 3 which is +1 4 so where will 4 go is my question 4 when there 4 is what +1 x +1 okay -1 x -1 +1 +1 that is 4 goes there right now this is -1 that will be -1 is +1 yeah there you go what about that k it is -1- 1 - what are they the same see you okay you negate x_2 you first negate x_1 and then pick the product okay.

Is essentially the same no see the idea is that I want my $x`_1 = x_1$ and 0$x_2$ they are not the same thank okay now tell me so we know that we can actually implement and using a perceptron right you can implement or not using a perceptron okay, so now do this so where will $x`_1$ be for 1 to next one is -1 0x_2 so there will be -1 right $x`_2$ will be -1 okay what about 2 -1 that is not the same thing okay does that make sense, so in this transformation where $x`_1$ x is x1and not x2 right and $x`_2$ is not x_1 and x_2 right where true is +1 false is -1 and if you do that then my projection will be 1 and 3 will get projected to -1 -1 and 2 and 4 will get projected to the same coordinates as they were earlier in this modified space okay.

Now is this linearly separable right, so now I can separate this right, so I can listen sleep construct a perceptron that does that so what have we done here is that xr is hard to solve using a single perceptron but I can hookup several of them and I can solve the problem right, so is that computable using a single perceptron yes or no that is computable using a semi perceptron that is computable using a single perceptron huh first one is not yeah so I am just giving people a chance to change their minds you need to for each of them one for the not do you need to do a not can I feed in the x_2 to do that can you do or not okay so if I can do x_1 and x_2 is it sufficient in invert the weight on x_2 to get x_1 and not x_2 you think an answer anyway so the point is essentially what I have done this I can actually build a perceptron that can compute x'_1 I can build another perceptron that can compute x'_2 so take these fill it to another perceptron which will actually compute right.

So essentially we have somebody sink another so essentially we have found they have shown that I can actually put this perceptron in layers and I can solve what was originally thought to be a problem that cannot be solved by perceptron right if you remember I was telling you in the beginning neural networks are very powerful because they are all hooked up in a very complex network right.

Individually they need on saw a very simple computing elements individually the neuron was a simple computing element and therefore you could not find the answer to $x^{\wedge}$ but then by hooking them up in appropriate Cascades you can find the solution to $x^{\wedge}$ right does it make sense, so sorry for the initial confusion on the feature transformation but now it should be clear right yes okay great.