Neural Network and Applications

Prof. S. Sengupta
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture - 21
Solution of Non – Linearly Separable Problems Using MLP

Lecture is on the Solutions of Non linearly Separable Problems Using Multilayer perceptrons instead of writhing big term, like multilayer perceptron I just use the abbreviation MLP. So, this is what we are going to discuss today. But before I go into the actually topic of today which, it is not going to cover too much of the time. Before that, I should continue the unfinished discussion, which we had in the last class related to the debate regarding the batch mode versus the sequential mode of processing.

And then, we will also spend a little time to discuss about the stopping criteria for the back propagation algorithm. And then, we will go over to the actual topic of today, that is solution of non-linearly separable problems in fact, that is one of the reasons, why we rejected the single layer perceptrons and went in for the multilayered prerceptron. Because, single layer prerceptrons were unable to solve the non linearly separable problems.

So, even we saw that the simplest one that is to say the exclusive or problem could not be solved using single layer perceptron. So, at least we should spend some time to explore, that if the exclusive or type of problems or to generalized it further into any non linearly separable problems, whether that can be solved using multilayered perceptrons or not. Now, the debate that we had initiated in the last class, that is batch mode versus sequential mode I think general feeling came from the class, that the sequential mode is preferred.

I can see that there are some reasons, which lead to such kind of thinking's in the mind of the students. So, where present here and may be that some of you who are distance viewers also got a very similar kind of a feeling, that as if to say that the sequential mode is good. The thing is that, there are certain points in favor of the sequential mode of learning no doubt.

But, there are also some favorable points and favorable batch mode of learning, let us not ((Refer Time: 03:46)) that ((Refer Time: 03:48)), first off all before we go into the debate, there after two other aspects that should be consider. First is that in the case of the sequential mode of learning, what we said is that, we will be doing the weight adjustments on a pattern by pattern basis.

So, we will feed one pattern based on the error, that we receive we adjust the synaptic weights for the entire network. And then, we feed the next pattern as well as the desire output and then we see the error, then we readjust that is how it goes on a pattern by pattern basis. Now, when we complete all the patterns like this, we will be completing one epoch of training.

And then, we have to repeat that epoch in a very similar way, again starting with some pattern. Now, the question is that the order in which we have placed the patterns during the first epoch let say, the question is that is it mandatory that the same ordering of pattern should remain in the subsequent epoch also. Apparently we tent to feel it this way, that the same ordering should be maintain, but really speaking there is no hard and first rule.

Because, after all what you are seeing in a sequential learning, which depends on the instantaneous value of the error energy. It is only the instantaneous value of energy that we are taking and the weight adjustments that are taking place is happening, because of some agglomerated effects of all the past patterns that we have fade. So, really speaking it does not matter, that in order to create that agglomerated effects of weight adjustments.

If you are looking at it this way, that the beginning of the epoch in sequential learning. And the end of the epoch in sequential learning, ultimately there is some weight change that has taken place starting with the beginning to the end, some weight change as taken place. Now, the question is that whether that weight change is certainly, because of some agglomerated effect of all the patterns that we have fade one after the other.

So, in the next epoch if you disturb that order, it should not matter in the end when we complete the epoch we should see a very similar thing. Only thing is that, if you randomize the order of presentation of the patterns, which is normally done actually, that the order in which the patterns are presented is normally randomized. So, that in the epoch the patterns will be fade in a different order in the third epoch it will fade in another different order.

So, in effect what we are doing is that although I may be considering agglomerated effect of weight adjustments, we still have very similarity. But, one thing is there that in that process, we are going to make the whole process stochastic in nature. Please understand that, that it is very much stochastic. Because, from one epoch to the other the trajectory, that we are going to follow in the weight space, is going to be different.

May be that the trajectory part is something, that we have got in first epoch, of course as we continue with more and more epochs, it will go further towards the minimum solution. If it is I am in already near the global minimum or if there is no local minimum existing, then the solution would be of course, for the global minimum or otherwise it will be a local minimum.

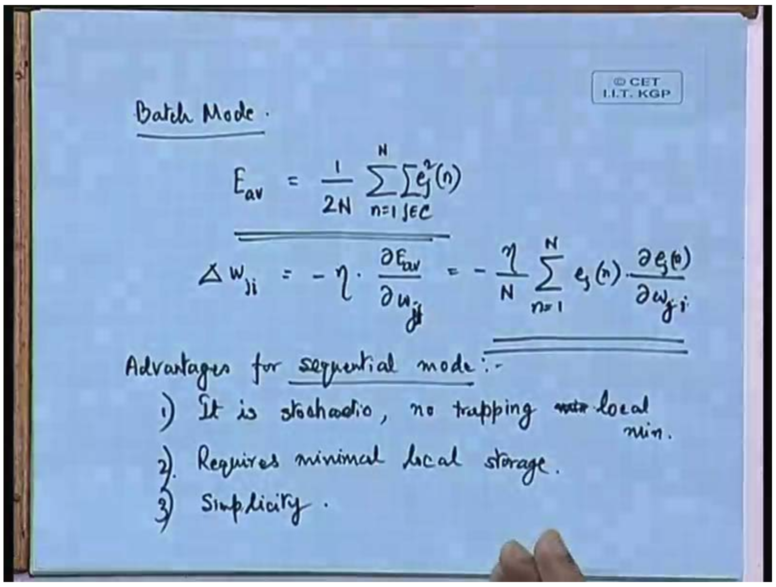
But, it will follow some trajectory and that trajectory may jitter from epoch to epoch if you randomize the thing. But, let us not all the time think, that such a kind of jittery trajectory is bad for us, such kind of a jittery trajectory could be there. In fact, we may tend to feel that as if to say that this kind of jittery trajectory, rather than a smoother trajectory which. In fact, we would be getting a much smoother trajectory, in the case of batch learning mode.

But, a jittery trajectory sometimes helps you in the sense that because it is stochastic it has got a tendency to for the network to sometimes drive out of the local minimum. You know, if there are some small pockets of local minimum, which requires some energy to drive it out of the local minimum it may be that. Because, there are some errors suddenly coming in the error terms, may be at times higher. That is, why such error can drive us out of the local minimum.

Now, such a kind of driving out thing is certainly not welcome when it is a global minimum. But, such a kind of driving out, which in term talks a bit about the instability kind of effect if I am correct in saying, so that has got some effect in driving it out of the local minimum. And if the global minimum is in the vicinity, then the solution could approach a global minimum.

So, sequential mode that way has got an advantage, if you make it stochastic by making the order of presentation of patterns in an epoch in a randomized way. So, that is one of the points that certainly it goes in favor of the sequential mode of learning, there is another point to consider, that in the case of batch processing. Let us look at the batch processing equation once again.

(Refer Slide Time: 10:32)



Batch Mode.

$$E_{av} = \frac{1}{2N} \sum_{n=1}^{N} \sum_{j \in C} e_j^2(n)$$

$$\Delta w_{ji} = -\eta \cdot \frac{\partial E_{av}}{\partial w_{ij}} = -\frac{\eta}{N} \sum_{n=1}^{N} e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}$$

Advantages for sequential mode :-

1) It is stochastic, no trapping ~~into~~ local min.
2). Requires minimal local storage.
3) Simplicity.

What we are doing in the case of batch processing remember is that, we are taking a cost function that is nothing but, the E average. We are not making it as E as a function of N where N is the number of iterations or the pattern number. So, in this case it will be the E average certainly becomes 1 by 2 N, how 1 by 2 N comes that should be very clear to you summation from n is equal to 1 to N e j square n.

Certainly, e j square n is the error squared term for the jth neuron in the output layer in fact, it should be double summation. So, here j should belong to the set c, where set c is the set of all the output neurons, so it is summation of e j square and in fact, we have been taking all the time as half of e j square. Because, the term half is convenient mathematically, since it cancels out the when we take the derivative then, that two gets canceled out with this half.

So, now because it is half term that comes for each of these, that again constant x out of the summation sign. And we are feeding n number of patterns within one epoch and this capital N naturally we have to divide the summation of all the error terms by 1 by N in order to get an average. So, in effect it becomes 1 by 2 N of this term, so this is the E average and consequently based on this E average, we are going to compute the delta w j i the change of weight as minus eta dau E average with respect to dau w i j.

And this in effect becomes equal to minus eta by N summation of n is equal to 1 to capital N e j of n into dau e j of n dau w j i here, this is a mistake this should be j i. Because, we are considering the neuron j and the connection that exist from neuron j to the other neurons, so definitely it is j i that we have to consider. Now, the look at the expression, that we have we got for the delta j i in the case of the batch mode of learning.

What is interesting to note here is that, we have a term which involves a summation of n is equal to 1 to n. And naturally in order to do such a kind of a summation we need to know this information, that is e j of n and dau e j dau w j i, this we should know for each and every n from n is equal to 1 to capital N we must everything.

That means, to say that we must have a storage, we must have a local storage to store these individual terms. Because, we are feeding one pattern after the other and when pattern is taken out and a new pattern is fade in, then we are normally losing that information's. So, we should not be losing that we should be storing it in some local memory.

So, that is one part that we have to remember, that we have to store individually all this things in some local memory. So, that at the end, because this delta w j i what we are doing is at the end of the epoch, that we are doing, because in that is how we have to operate in the batch mode. Because, batch mode will have the weight adjustment only at the end of the epoch, not with every patterns in the case of sequential mode.

So, you need local storage, which you do not need in the case of sequential processing. Because, in sequential processing you are only bothered about the pattern, which is now under consideration. And the memory effect in indeed the two implementation has got a difference in this respect, actually both need memory in terms of the weight adjustments. But, in the case of the sequential mode, that memory is embedded with the w j i term, because w j i term is accordingly getting updated with every pattern.

So, what you are seeing as w j i is indeed the effect of the previous patterns, that have been already consider an updated. Whereas, in the case of batch mode that delta w j i that you are going to see is only at the end; that means, to say that externally we have to support a memory in order to store this term. And then, at the end of epoch we do the adjustment based on this. So, naturally some of the two advantages that we can site for the batch mode of learning is advantages for no first let us talk about the sequential mode.

The advantages for the sequential mode, that we can talk of first is that it is stochastic, if you assume that the pattern feeding order is randomized, then it is indeed stochastic. So, no trapping in local minimum or less possibilities of trapping in local minimum, if I am correct in saying, so. The second advantage that we are going to have for the sequential mode is that, requires minimum local storage in fact, nothing.

So, these two are advantage and of course, this simplicity of the algorithm, the simplicity in fact, should attract us. So, sequential mode is definitely having has got this advantage, which you must have felt intuitively when you said that, you prefer sequential mode of a batch mode. But, now coming to some other factors, firstly is that look at the cost function of the batch mode.

Indeed this is the cost function with which are bother with, because we are bothered about the training for not one pattern, we are bothered about the training over a pattern set. So, naturally the E average is indeed a better cost function for us, now you can argue like this that in case of sequential processing, we are using instantaneous value of the error energy as a cost function.

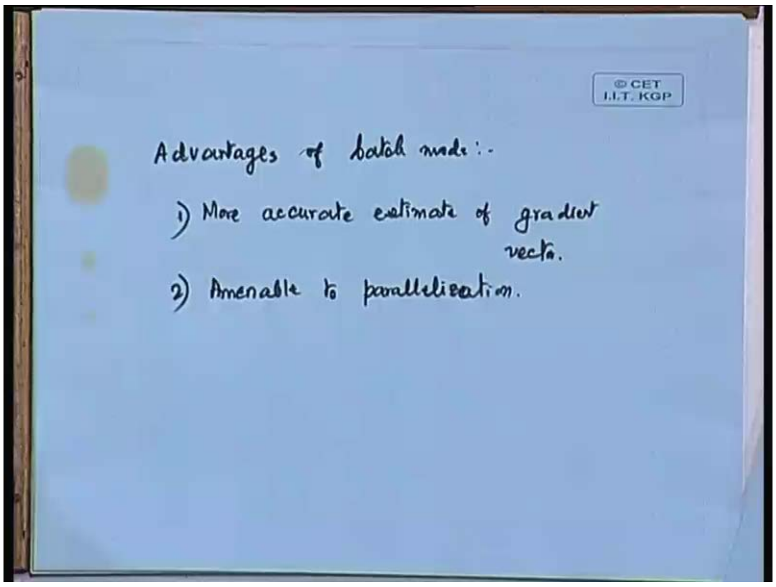
But, ultimately it is some agglomerated effect, ultimately it takes into account the effect of all the patterns. Now, it is true that way in fact, just to compare the effects I should say that in one case, you are doing the delta w j i adjustment in every with every pattern. Whereas, in this case the delta w j i your are going to do at the end, now are these two delta w j i's going to be the same, ideally they should, but they are not the same.

Now, the question is that which of this delta w j i's is better in terms of the a computational consideration this is better. Because, the cost function which we are taking in this case is the actual cost function that we should have taken. Whereas, in the case of sequential mode by simplifying with the saying, that we are taking the instantaneous error energy, we are making some approximation in the cost function.

So, this is more accurate representation of the cost function and that is why, this is a more closer solution to the steepest descent. In other words the trajectory that the batch mode is going to follow is a much smother trajectory, as compare to the sequential mode. In fact, it is expected even intuitively also, because batch mode already does by having E average and delta w j i like this. We are already in cooperating some amount of low pass filtering followed, we are already doing some filtering on it by taking the process of averaging.

And so naturally the filtering looks much better, the filtering effect is much better as compare to the instantaneous error energy, which is going to have lot of noisy components. Which will result in some jitters in the weight space, when it follows the trajectory of course, the disadvantage could be is that, in the case of batch mode I am going to the minimal is guaranteed. But, if it is a local minimum, it will be at the local minimum, because batch mode is not stochastic, so there are some kind of advantage that we can talk for the batch mode also.

(Refer Slide Time: 20:55)



© CET
I.I.T. KGP

Advantages of batch mode:-

1) More accurate estimate of gradient vector.

2) Amenable to parallelization.

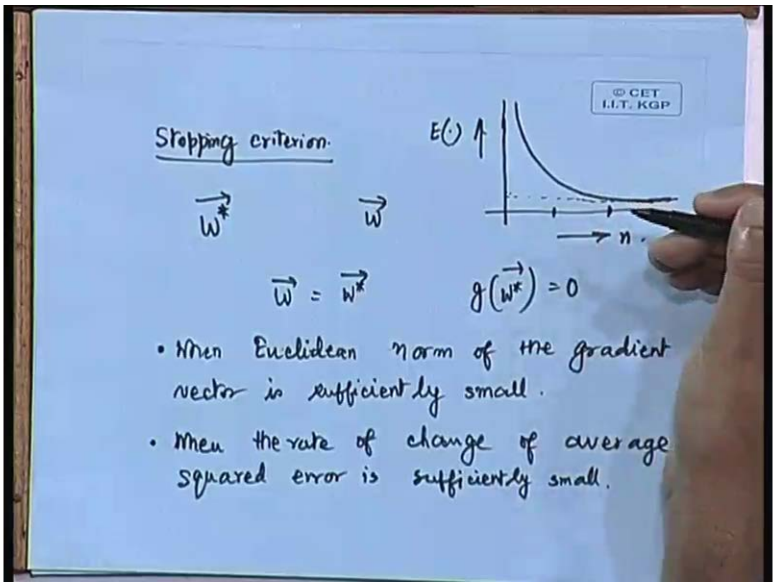
So, the advantages that we can talk in favor of the batch mode are, firstly that it is a more accurate estimate of gradient vector. And there is another aspect, that we have not really discuss upon, now remember I been I think we have to go back to our lecture number 1. When, we talked about one of the great advantage of the neural network being the parallelization. That you have, so many neurons which can do parallel processing.

Now, if I look at the problem like this, if I have a multilayered perceptron and I have a choice that I can give it a sequential training or I can give it a batch training. From the parallelization point of view, if I had used a batch mode I have the advantage, that I could replicate all these MLP’s. So, that I fade individual patterns to each one of these multilayer perceptrons, the patterns that I have in my pattern set.

If cost is not a concern for me, then I will be able to parallelize the computation of all the patterns. Because, the computation of all the patterns are independent in the case of batch mode, whereas in the case of sequential mode it is dependent from pattern to pattern. So, that is why the batch mode is amenable to parallelization. So, this is amenable to parallelization, certainly this is not an advantage that sequential mode is having.

So, that is more or less the kind of advantage is that each one of these techniques, that is sequential and batch mode as got. Computationally of course, batch mode is closer to the accurate estimate of gradient factor. But, in terms of simplicity, in terms of a requirement of less storage sequential mode is certainly better. Now, we go over to the next discussion that we have in our mind and that is related to the stopping criteria.

(Refer Slide Time: 23:50)



© CET I.I.T. KGP
$E(\cdot)$
n

Stopping criterion.

$\vec{W}^*$ $\vec{W}$

$\vec{W} = \vec{W}^*$ $g(\vec{W}^*) = 0$

- When Euclidean norm of the gradient vector is sufficiently small.
- When the rate of change of average squared error is sufficiently small.

Now, we have been discussing, so long about the back propagational algorithm. Now, one question that might have come to your mind is that, when are we going to stop it, how long are we going to continue this the training as such. Now, what is the kind of feeling that I can get from this audience, let see that if I give you multilayered perceptron to train using the back propagation algorithm, when are you going to stop any opinion that you can give.

Student: Zero cost function.

Zero cost function good, any other consideration.

Student: No changes in the two computational cost function.

No change in the.

Student: ((Refer Time: 24:47))

No change in cost function, perhaps your friend has said that the cost function becoming zero and you are saying that no change. When, no change does not mean that it is no change at zero, no change could be at a high cost also or that no change could be a just an accidental no change. May be, that if I continue the pass again, there will be some change or if I just give some stochastic push to the system, there will be some change.

Any other approaches that you would like to adopt. Now, you are thinking in terms of the cost function, you could think in terms of the gradient also, gradient of the cost function. Because, after all when you are taking the cost function, the best thing will be to take the gradient of the cost function, that is what we are taking is not it.

So, in fact we are in the w space the weight factor w, this is in the m dimensional space and after all what we are doing is that, we are making a search in the m dimensional space for a minimum of the gradient. So, that is why we are going to find out a solution of w star vector, such that at the point w equal to w star, we must have g of w star to be equal to 0.

If we have the gradient to be equal to 0, then we are at the point of minima of course, gradient becoming 0 could be a maximum or minimum. But, because of in this case we are going in the negative direction of the gradient, so naturally the path that we follow is always towards the minimum. Now, it is not that every time we can reach the g w to be equal to 0, ideally we should reach, but we may not reach the exact value of gradient equal to 0.

So, one of the stopping criteria that we can see from a more practical point of view, that instead of telling 0, we can say that the gradient vector is sufficiently small. So, we can say that one of the stopping criteria's could be is that, when Euclidean norm of the gradient vector is sufficiently small. Now, coming to the second point which the students present here have already raised, that to say that the cost function reaches a minimum or when the cost function as such becomes 0 somebody said that 0.

I should say, that what we should look for is that the rate of change in the average square error, average square error is after all ours cost functions. So, when the rate of change is sufficiently small, when the rate of change of average square error is sufficiently small. In fact, we can say that if this is the number of iterations, the pattern numbers that we are fading.

And if this is or to say even question is that, from here to here it is 0 to n, where n is the number of patterns is in an epoch. And then, again n patterns or if you are considering a epoch to epoch learning, whatever way you do, the thing is that if this is the if the y axis is the error term, which is the cost function in this case. Now, what happens is that the learning normally follows a curve like this. And when the learning follows curve like this, we tent to stop over here somewhere over here.

Because, we find that beyond this point there is no further change in the error energy. So, if you are thinking in terms of 0 error, may be that it is not converging to a 0 error. Absolute 0 error you are not able to get, your are getting some residual error, which actually remains in the system. So, may be that if this is the residual error that is going to remain, we can say that the point where the rate of change of the average square error is sufficiently small almost equal to 0 or it is not changing at all from one iteration to the other from epoch to the other.

That is the point where we should stop the training, then we can stop the training and use that same network for in a tested environment. Because, ultimately that is what we are going with the neural network. Now, some people say that these two are very good from a theoretical point of view, in the first case you need to compute the gradient. And in the second case you need to compute the average square error which of course, you have to...

So, the second approach is definitely simpler as compare to the first one. But, one thing which is very much required is that, no matter what this cost function is or what that gradient vector Euclidean norm is. We are going to use the multilayer perceptron only with a good generalization capability. We can say, that subjectively we should see that when the back propagation network when the multilayer perceptron as got a good generalization capability, that is the point when we should stop the training.

So, some alternative approaches which people have worked out is that, instead of getting it trained and going over form iteration to iteration and reducing the cost function like this. Sometimes, it may be a better approach that you train it with patterns, few epochs or some number of epochs you do, you sufficiently train it. And when you train it, in between you stop the network and test the network for it is generalizations capabilities.

So, what you should do is that, you should have the training patterns at your disposal, you should also have the test patterns at your disposal. So, what you do is that, you feed the training patterns and then once it is trained for some iterations, you fade some patterns from the tests. Again these test matters are the once for which you are knowing the desired output.

And you see, that how good it can generalized, means whether by fading the test pattern the output that you are getting is close to the desire or not. If it is close to the desire, then the network already has got a generalization capability, in the sense that it is able to compute correct output for the patterns, that it never encountered before. If we find that it is not at generalized, then what we have to do is just to retrain it once more, means just to continue with the training.

And again stopping it and then test for a generalization, now sometimes it is seen that by doing like this. For many small scale problem it is possible to find a weaker convergence, even though we are not measuring the convergence strictly in terms of a characteristic curve like this, the error versus iterations. Instead of a curve like this, we test the generalization capability that is also one of the approach is that you take.

So, these are the some of the basic discussions, that I wanted to continue in connection with the back propagation algorithm as such. Before I go over to the non linearly separable problems that is the topic for today is there any question, that you would like to ask in this regard. Anything pertaining to the back propagation algorithm it is batch mode versus sequential mode stopping criteria.

So, then I can go over to the topic of the day that is solution of non linearly separable problem at least that something, which we should explore once more, that whether multilayer perceptron can do it. Now, let me tell you one thing, that when I said multilayer perceptron, this time I use the term multilayer perceptron I did not use the term back propagation network. Now, again I have seen that sometimes people make a misconception as if to say, that all multilayer perceptrons are back propagation network, only it is not correct.

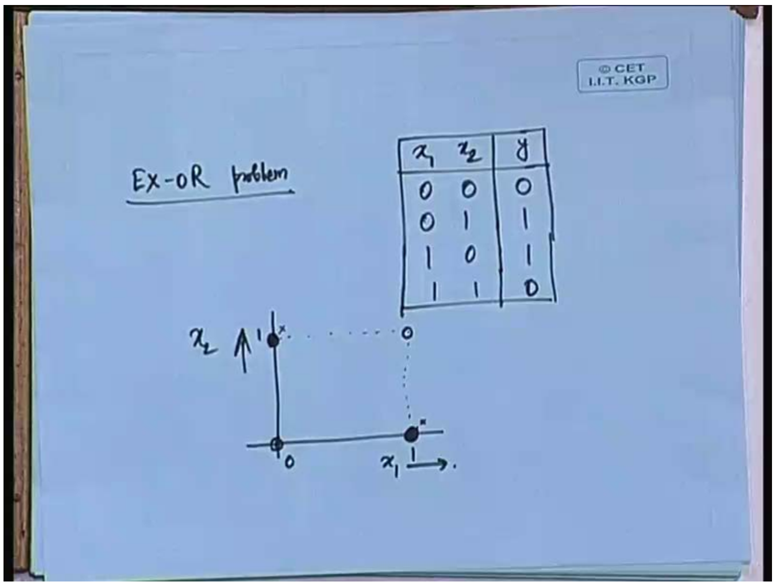
Back propagation is a algorithm, where by the weight adjustment is propagated from the outer layer to the hidden and the inner layers, gradually that is a kind of algorithm. Which essentially requires one fundamental conditions for a back propagation algorithm to work is that, the five function the activation function that we are using, that must be differentiable continuous and differentiable.

So, now in the last class we discussed that some of the typical activation functions, like the sigmoidal non linearity. In terms of logistic function we already derived, that what it is derivatives and what the corresponding delta term the local gradients are, these things we have discussed. But, we should also keep in mind that it is possible for us to construct a multilayer perceptron, even with neurons which are having activation function like signum or McCulloch and Pitts, which are in fact, having discontinuous phi functions.

But, multilayer perceptron definition is that, any perceptron network arrangement that you are having, where there is one input layer there is one output layer. And there can be at least one hidden layer, but we are not putting any restrictions on the activation functions. Whereas, in the case of back propagation algorithm, there is a restriction on the activation functions that it must be differentiable.

So, we can just see that whether any multilayered perceptron with of course, suitable choice of weights, can solve a non linearly separable problem or not. And the simplest of that which we are going to take is the exclusive or problem.

(Refer Slide Time: 37:35)



EX-OR problem

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

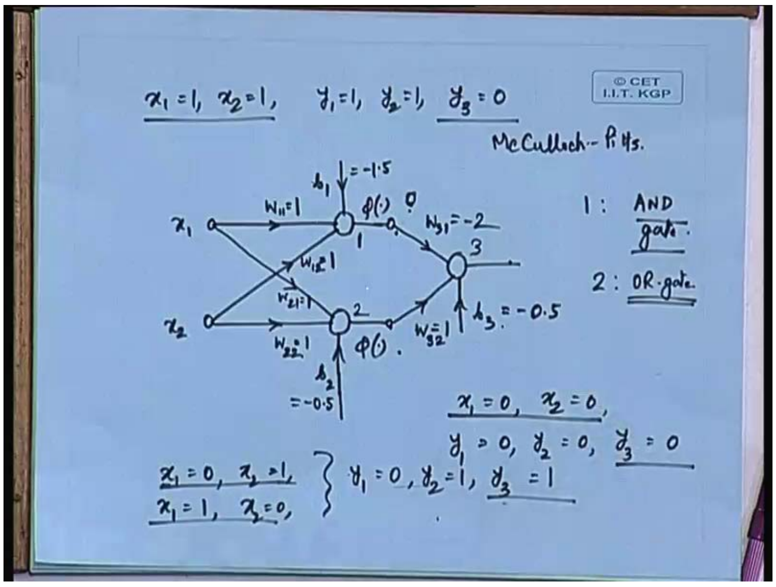
So, we reconsider the EX-OR problem and we had shown in the during the discussion of single layer perceptron, that single layer perceptron is unable to solve exclusive or. Now, what that exclusive or problem is that, we are having two inputs x 1 and x 2 and we are having a truth table like this by y being the output. When it is 0 and 0 the output is 0 0 1 or 1 0 the output is 1 and 1 and 1 that is to say when both of them are 1 then the output is 0.

So; that means, to say that only if just one of this is 1, then the output is 1 that is the exclusive or problem. Which basically means that if we try to see in the two dimensional space. Where x 1 and this axis and x 2 in this axis, this means the 0, this being 1, this being 1 for x 2 we are having 4 corners of a square to which I have insist the patterns are 0 0 0 1 1 0 and 1 1. We can feed the patterns, we can keep the patterns in the 4 corners of the square, which we have formed like this.

Now, what we are doing is that two out of this; that means, to say that when it is 0 1 and when it is 1 0, for those patterns we are going to have the output as 1. So, the once that I have just now marked up here and this problem and this two are open circles indicating that, they are 0s. Now, the basic problem of linear separability which we had already define is that, we are not able to imagine a line that separates the 0 patterns from the 1 patterns.

In that sense it is non linearly separable, but let us see that if we could solve the problem using the multilayer perceptron or not.

(Refer Slide Time: 40:02)



Now, we just take an example of a multilayer perceptron, where let us say that we have got 2 inputs x 1 and x 2. And we are going to use only 3 neurons in this, 2 neurons in the hidden layer and 1 neuron in the output layer. Now, what we are going to do is that, we are going to connect the x 1 and x 2 to this neurons to this hidden layer, so this two are forming the hidden layer. Whereas, this one is the output and again this as well as this should be connected to this neuron.

Now, both this neurons will be having some bias, now this neurons will go through some activation functions. So, let us say that these are the phi functions of this neurons and then, the activation outputs that we will be getting from this neurons, they will be connected to the output neuron. Output neuron also will be having some bias and then, this is the output that we are doing and we are getting.

And here, there will be synaptic weights between this x connection of x 1 to this neuron, let us say this is neuron 1, this is neuron 2 and this is neuron 3. So, this will be in effect the w 1 1, this will be w 1 2, this will be w 2 1 and this will be w 2 2, this will be bias b 2, this will be the bias b 1 and this will be the bias b 3. And here it will be because it is connected from 1 to 3 it should be w 3 1 and here it should be w 3 2.

And for the sake of simplicity, because I do not want into the computation of 1 by 1 upon exponential term, etcetera for which I will needing a calculator. I will simply apply McCulloch Pitts model, that looks much easy for me. So, that all that I am going to do is to compute the activations and then I will be decided that whether it is 1 or 0. Now, you can argue that McCulloch Pitts model, certainly does not make the network trainable with back propagational algorithm.

Because, it is not differentiable, but to everything there is an approximation, that you can always do in the sense that you could take a logistic function only. And all that you do is that you put a sufficiently the a, that to say the slope that you are having for the logistic function, you make that sufficiently high. If you are making it sufficiently high, still as long as you follow the function as 1 by 1 plus exponential to the power minus a v, that function becomes differentiable and we can more or less approximate that.

So, that it is not correct to say that I cannot train it using back propagation network, I can train McCulloch and Pitts model, with a closest approximation. That I take a to be quite large for all practicable purposes. And let us say that somebody has trained this network like that and obtain a set of weights as follows, say that I take w 1 1 to be equal to 1, w 1 2 to be equal to 1, w 2 1 to be equal to 1 and w 2 2 also to be equal to 1.

So, all these weights are equal to 1 and I take b 2 to be equal to minus 0.5 and b 1 to be equal to minus 1.5 the whole idea of doing it is, if you look at this two neurons, let us have a look at the neuron 1 and 2 now. What is the total induced local field at the neuron 1, let us say yes x 1 plus x 2 minus 1.5. So, the discussion boundary is that x 1 plus x 2 minus 1.5 should be greater than or equal to 0.

So, when x 1 is, so we go through the truth table ((Refer Time: 45:47)), what happens for 0 0 is the output active or not this McCulloch and Pitts mode. So, is the output of this neuron 0 or 1 0 0 0 is 0, what about 0 1 0, because the total activation is 1 minus 1 minus 1.5 which is equal to minus 0.5 less than 0. So, the output remains 0, so any one of these means that it is still 0.

Only when both of them are 1, when only when x 1 is equal to 1 and x 2 is equal to 1, this is having a an activation value, net activation value equal to 0.5. So, then it goes 1, so it is acting as what logical gate AND gate. So, the neuron 1 is acting as a normal AND gate, what is the logic function realized by neuron 2.

Student: ((Refer Time: 45:49))

OR gate neuron 2 is indeed an OR gate, you just see when x 1 is equal to 1 when both are 0s, that time the net activation is equal to minus 0.5 which is less than 0. So, the output is 0, but when any one of them is 1 or both of them are 1, in this cases it will be when any one of them is 1, then the net activations is one this one is 0 and this one is minus 0.5. So, it is 0.5 greater than 0, when both of them are 1 1 plus 1 2 minus 0.5 net activation 1.5 still greater than 0. So, number 2 neuron is acting as an OR gate.

Now, what we are doing is that we are combining the outputs from this AND gate and OR gate, into another neuron which is there in the output layer. And for that matter we have judiciously chosen a weight like this, we have chosen w 3 2 to be equal to 1. The one which is connected to OR gate, I put w 3 2 to be equal to 1 and w 3 1 in a very interesting way we make it minus 2. The purpose of doing that is that, we are going to make an excitatory connection with w 3 2.

And we are making an inhibitory connection, because a weight here is minus 2, we are making an inhibitory connection with this neuron 1 to neuron 3. And if we make things like that, you can now see that when the output of this neurons, when the neuron 1 output is 0. And I took b 3 to equal to be equal to minus 0.5, so that when both these neuron 1 and neuron 2 outputs are equal to 0, in that case output is going to be 0.

When this is 0, but this is 1 the output of OR gate is 1, but the output of AND gate is 0, in that case what we are getting, we are getting 1. Because, this is 1, this is 0 and this is minus 0.5, so this is 0.5 which is greater than 0, so this will be 1 and when this is also 1 and this is also 1, in that case what do we get 0. Because, the net activation is 1 minus 2 minus 0.5 which is less than 0.

So, the output will be 0 and when is the condition that we are getting both of the outputs as 1.

Student: ((Refer Time: 48:54))

X 1, x 2 both equal to 1, only in that case both neuron 1 and neuron 2 will be having there outputs equal to 1. And only then we are going to and in that case we are going to have no matter what we yes when both of them are 1; that means, to say both of them are 1 over here. Then, we are getting 1 here 1 here and that will lead to output being 0, when both of them are 0 x 1 and x 2 0, in that case this is inactive, this is having output 0, this is also having output 0.

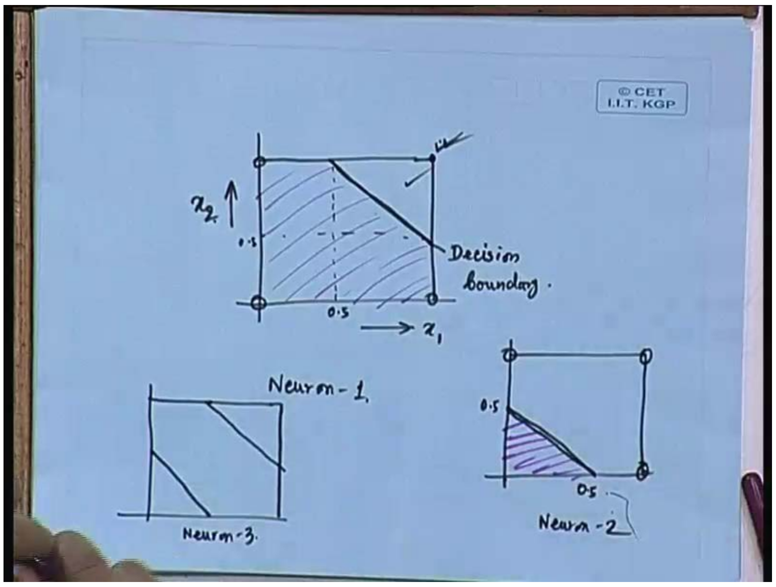
And this is minus 0.5. So, we are again having the output as 0, 0 0 is 0, 1 1 is also 0. And when it is 0 1 or 1 0, in that case this output is 0 and the neuron 2 output is 1 and that will make the output 3 to be equal to 1. So, we got what we wanted is it solving exclusive or problem, those who are not convinced please feel free to ask me, it is very simple to see that I can just repeat it once more. When x 1 is equal to 0, x 2 is equal to 0 please verify that neuron 1 or to say I can say y 1, y 1 is equal to what.

Student: 0.

Y 1 is equal to 0, y 2 is equal to 0 and y 3 is equal to 0, when x 1 is equal to 0, x 2 is equal to 1, what happens y 1 is equal to 0, y 2 is equal to 1, y 3 is equal to 1. When, x 1 is equal to 1, x 2 is equal to 0, then same thing, same thing happens y 1 is equal to 0, y 2 is equal to 1, y three is equal to 1 you can see that. And when x 1 is equal to 1, x 2 is equal to 1, in that case y 1 equal to 1, y 2 equal to 1, but y 3 is equal to 0.

So, in effect if we are looking at only the table patterning x 1, x 2, y 3, x 1, x 2, y 3, x 1, x 2, y 3, x 1, x 2, y 3 we are getting exclusive or solution. So, what a single layer perceptron could not do a multilayered perceptron can do this.

(Refer Slide Time: 52:07)



In fact, if you are looking at the two dimensional space in fact, the problem should again be post this way, that we are getting x 1 and x 2. Now, if you look at the output of the neuron 1((Refer Time: 52:21)), neuron 1 in this case is acting as an AND gate, as you rightly found out. So, AND gate means only when this one both of them are once, then only it is active.

So, in fact it is line of separability should be something like this in fact, it will be like that here 0.5 ((Refer Time: 52:48)). Because, you see the way they this weights have been chosen over here 1 and 1, so as long as this is greater than 1.5 it is going to give you an output equal to 1. So, here also 0.5 and here also 0.5 and will be having a line like this, which will separate the patterns of this.

So, this will be the case where the output will be equal to 0 and this will be the zone, where the output will be equal to 1. So, this will be the situation for neuron 1, this will be the discussion boundary and for the case of neuron 2 ((Refer Time: 53:36)), which is follow the logic of an OR gate, what is going to be such kind of a discussion boundary, in this case should be this side or should it be this side, that is very correct.

In fact, again because we have chosen 0.5 it I easy to see here, that both of these are 0.5. So, the slope of the line will be 45 degree and in this case the 0 patterns are situated over here and the 1 patterns are situated over here, because neuron 2 is an OR gate. So, for this 1 this 1 as well as this 1 the output is going to be equal to 1 only for this it is 0.

Whereas, for the case of AND gate this one, this one and this one, the output is 0 and only for this the output is equal to 1, so that for neuron 2.

And in fact, neuron 3 coming to neuron 3, it becomes a combination of neuron 1 and neuron 2. In such a manner that neuron 1 is having inhibitory contribution as if to say it is reversing, but the effect indeed and neuron 2 is having an exited connection. And finally, in the case of neuron 3, the decision boundary that we are going to get is something like this there will be in fact, two decision boundaries.

There will be one decision boundary contributed from here and there will be another decision boundary contributed from here. And in this case what happens is that for the patterns lying inside this decision boundary, the output is equal to 0, for the patterns lying beyond this decision boundary the output is 0. Essentially, contributed by this decision boundary, but everything reversed inverted.

Because, in effect this is an inverter connection ((Refer Time: 55:48)) this is a not gate if you look at it from 1 to 3, because it is having a negative weight here. So, the for this patterns the output is 0, for this pattern the output is 0, so that means, to say that for this 2 corners the outputs are 0. Whereas, for this one and this one, the output is equal to 1, so this is the boundary decision boundary, that we are following from neuron 3, which indeed solves the exclusive or problem.

So, that is the greatest advantage in fact, it is easy to train a network like this ((Refer Time: 56:33)) those who are conversant with the programming at a camp very easily try it out. What you do is that, you can instead of taking McCulloch and Pitts model, you can take linear logistic function you can take. And then, you can start with some arbitrary weight and arbitrary bias and train this network, because they exclusive or truth table you know.

And train it sufficiently and you will see that after training it will be able to classify the exclusive or patterns in the desired way. So, this can be experimented very easily using your own computers, so that all for today.

Thank you very much.