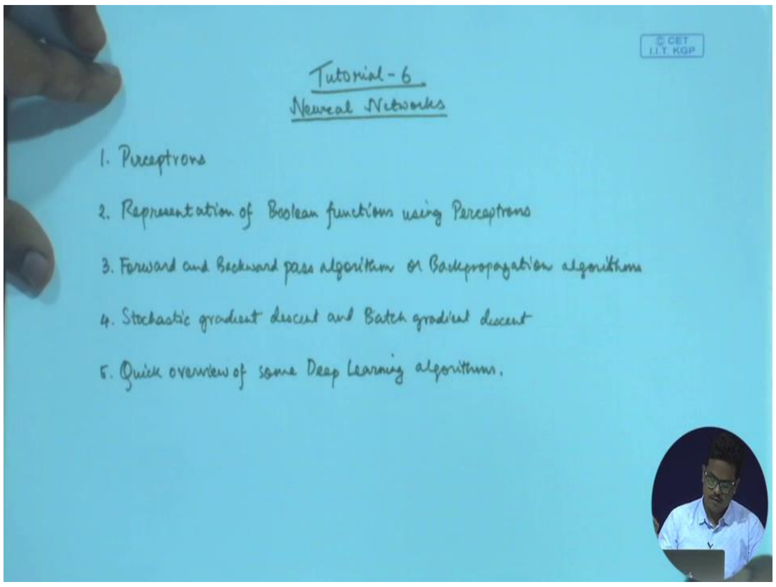
Introduction to Machine Learning
Prof. Mr. Anirban Santara
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 35
Tutorial VI

Hello friends, welcome to the tutorial session of the sixth week of this course. Today's topic is Artificial Neural Networks. We are going to summarize the content covered in the theory session and learn how to solve problems in exams. Let me jot down the sub topics first.

(Refer Slide Time: 00:33)



Tutorial - 6
Neural Networks

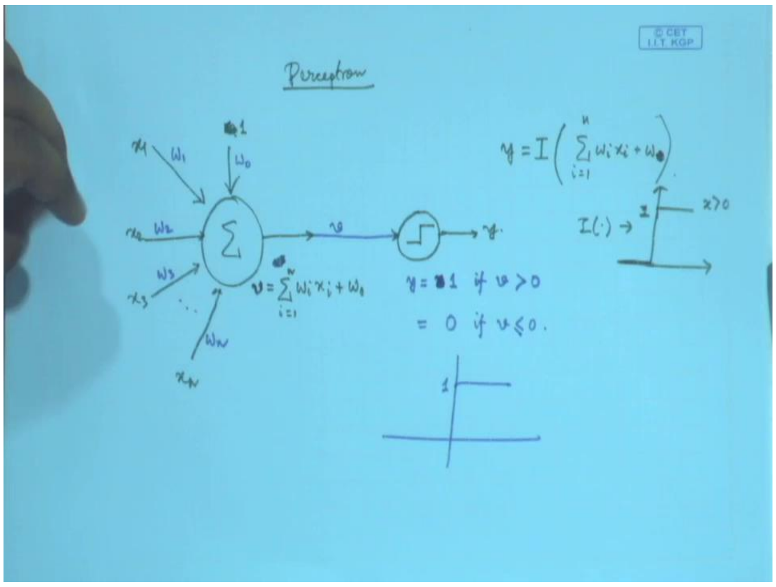
1. Perceptrons
2. Representation of Boolean functions using Perceptrons
3. Forward and Backward pass algorithm or Backpropagation algorithms
4. Stochastic gradient descent and Batch gradient descent
5. Quick overview of some Deep Learning algorithms.

Number 1 is perceptrons, which forms the basics of artificial neural network architectures. Number 2 is representation of Boolean functions using perceptrons which gives a quick look at how perceptron can actually come into use.

Next, we will look at the forward and backward pass algorithm which is also known as back propagation algorithm which is used for training artificial neural networks, an efficient algorithm for training artificial neural networks. Fourth, we will look at different variants of art of the back propagation algorithm likes stochastic gradient descent and batch gradient descent. Fifth, we will give a quick overview of some deep learning algorithms.

So, as people have already been told that artificial neural networks form the basics of the foundation of deep learning algorithm, so deep learning is nowadays the state of the heart in almost all machinery applications and it is a highly powerful artificial intelligence tool and personally I am obsessed with deep learning and I am doing my PhD in deep learning and hence this class is special for me. So, without further adieu, let us jump right in.

(Refer Slide Time: 02:49)



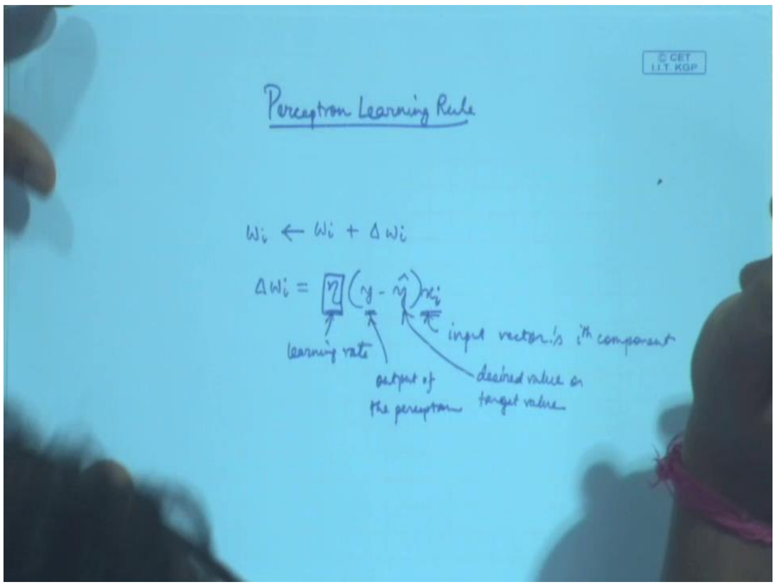
The first topic for today is perceptron. So, the idea of perceptron is a unit which acts as a small unit of computation right. What does a perceptron consists of? A perceptron is a very simple computational unit which does a simple linear combination of the inputs, say the inputs are x 1, x 2, x 3 till x n and there perceptron is first going to multiply each of these inputs with a weight w, like this and then there is a bias term as well which is simply we represent it as like this way in the same way. So, you multiply 1 with this term w 0. So, this is a bias term and after this has been done what you end up with is called v which is equal to submission over i equal to 1 though n w i x i plus w 0.

So, after this linear combination of inputs has been done, the perceptron goes ahead and passes this through a non-linearity which is nothing, but this step function. So, we define this step function as y equal to x, say the input is x or the input is v. So, let me write it here, and this v is equal to this, let me keep the color consistent, this v. So, y is equal to 1, if v is greater than 0; y is equal to 0, if v is less than or equal to 0. So, this is the function is the non-linear function that we use here, which is simply threshold right. So, it is 1 for x greater than 0 otherwise is equal to 0. So, after whatever comes out is the weight is our final output of the perceptron, let us call it y.

So, the overall equation of the perceptron, overall function which it modules can be written like this. So, it is y equal to indicate a function of submission i equal to 1 through n w i x i plus w 0 and this indicator function indicates whether it is positive or non-positive. So, this is actually 0, i is this function. It is 1 for greater than 0. So, y is equal to 1, otherwise it is equal to 0. So, this is the function that the perceptron models and the perceptron learning rule is a rule which is an update rule for these weights all right.

So, how does this perceptron actually put into to work? So, you have a set of training examples training examples come in the form of trainee in an input vector which is nothing, but this x 1 through x n and output label right this y. So, you will be given a ton of these training examples and you will be updating the weights and the bias term of the perceptrons, so that the input-output function which is present in the data is module accurately by this perceptron right. So, this perceptron learning rule is given by this.

(Refer Slide Time: 07:04)



© CET I.I.T. KGP

Perceptron Learning Rule

$$w_i \leftarrow w_i + \Delta w_i$$

$$\Delta w_i = \boxed{\eta}\,(y - \hat{y})\,x_i$$

learning rate

output of the perceptron

desired value or target value

input vector's i^{th} component

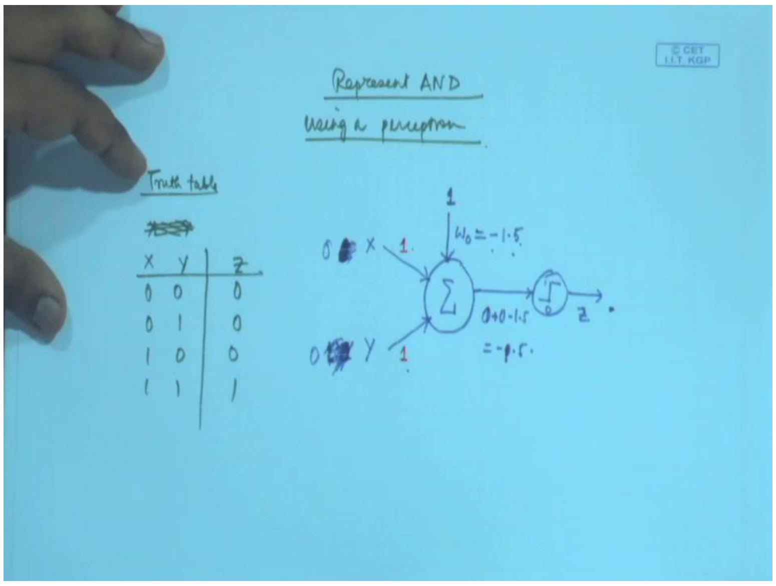
This is perceptron learning rule. So, the perceptron learning rule says that w i plus delta of w i and like after every single examples as been shown right. So, I will come to this later to talk more about this later right and this delta w i is equal to eta y minus y cap x i. So, x i is the input vector ith component. So, this quantity eta called is the learning rate and this basically comes from the variant descent algorithm. So, the gradient descent algorithm was this way that at every single update at the time of each update you are going to figure out the direction of the gradient of the loss function in terms of the parameters and then you are going to take a small step backwards in the opposite direction of the gradient.

So, you find the direction of the gradient, this is the direction in which the function increases the error increases. You are going to take a step backward in the opposite direction of the increase of error. So, you are sure that the next time the parameter values that we land up at the error value will be lesser at the particular value of parameters.

So, you go in the opposite direction of gradient and that particular algorithm in that gradient descent algorithm the learning rate term this eta over here. This term actually says it indicates how much how bigger step, we are going to take in the direction opposite to the gradient, the higher the value of eta the more we go in the larger is the step the further we go in the direction opposite to the gradient. This is the way and y is the output of the perceptron and y hat or y cap is the desired value or the target value which came from your training set right.

So, pretty straight forward and you have been already thought in the theory class. So, what kind of question can you expect from the exam, you will be given this set of examples in the training set and you will be asked to find out the update rule for a particular weight may be right. So, just plug in these values into this formula and here good to go. Let us move on to the next topic, which is representation of Boolean functions, this is pretty cool.

(Refer Slide Time: 10:21)



Represent AND

using a perceptron

Truth table

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

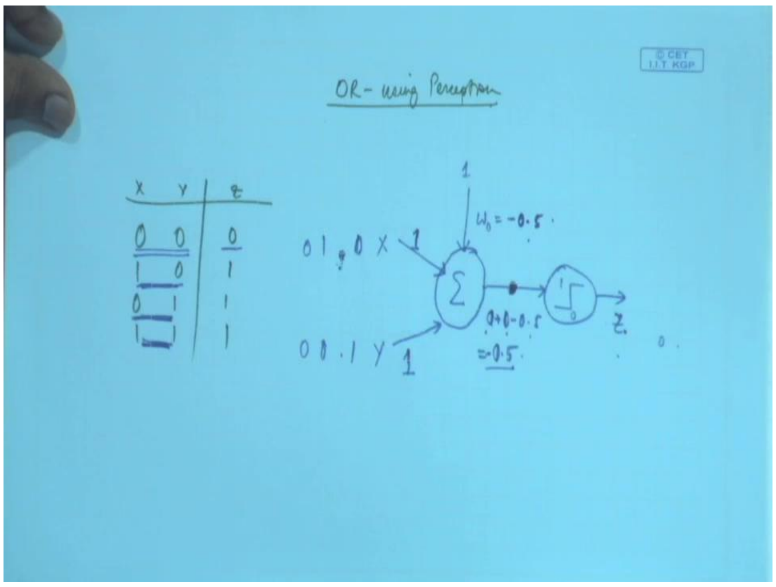
Let us go ahead and represent the function and the Boolean function and using a neural network, using a perceptron see this is pretty straight forward. So, the truth table of and is like this. Let the inputs x and y and the output be z right. So, for different values of x and y we are going to the value of z.

So, when x and y are 0 0 for and function this is going to be 0 0 and 1 is also 0 1 and 0 is also 0 1 and 1 are 1. So, it is going to be 1 only when both the inputs are 1 right. So, let us make a neural network or just a perceptron which works, which models this function. So, this is how it looks right, this is at 0, this is 1 and this is my output z and this is nothing, but the sum there is this bias term 1 and then multiply by w 0 and this is what we get here, let us keep our lives simpler. So, you are going to have x and y over here and let us start with this is just. So, simple as a basic function that you can just put weights of 1 right and just notice that what are we going to have, the output whatever. So, the outputs z is going to be 1 only when both the inputs are 1.

So, what if we put a value of w 0 equal to minus 1.5 which, like when both the inputs x and y are both of them are 1. So, this part 1 plus 1 gives you two over here, you subtract 1.5 from that and you end up with 0.5, which is greater than 0 and you have an output of 1 right and if you have 1 of them, 1 of the inputs 1 and the other 1 0 or both the inputs equal to 0 then this bias term when added to it. So, like see what happens in this case the output over here becomes 1 plus 0 minus 1.5. So, which is minus 0.5 which is a negative number and the output will be 0 in this case right.

And the same is the case when this is 0 and the other 1 is 1 right the same scenario and when both of them are 0 when both of them are 0 then again you are going to have 0 plus 0 minus 1.5, which is minus 1.5 which is again lesser than 0 and hence the output is going to be 0. So, this particular neural network with the weights equal to 1 and 1 and bias term equal to minus 1.5 acts like an and get which outputs the value of 1 only when both the inputs are equal to 1 and if either of them happen to become 0 then the output is 0 right. So, you got the trick right, if you did not get the trick then you think over it.

(Refer Slide Time: 13:53)



OR- using Perceptron

X	Y	Z
0	0	0
1	0	1
0	1	1
1	1	1

Next, go ahead and try to represent the function or using perceptron. So, the OR function works this way. So, x y z, only when the when both the inputs are 0 its going to be 0 otherwise if any 1 of them is 1 then the output is going to be one. So, in this case let us have the same trick. This is 0, this is 1, this is going to be your z and let us have 1 as weights as before right and let us play with the bias. So, in this case you have to make sure that the output, the number over here becomes lesser then 0 only when both the inputs are not 1, correct.

So, let us put the value like 0 minus 0.5. So, when both the inputs are 1 then this quantity becomes 1 plus 1 minus 0.5 which is equal to 1.5 right which is greater than 0 and if I said something wrong. So, the trick is you have to make sure that the value over here is always greater than 0, if the input both of the inputs are not 0s right. So, this is the condition of the OR gate. So, let us put a value of minus 0.5 over here, let us see what happen. So, both the inputs equal to 1 the case when both the inputs are equal to 1. So, this becomes this gives value of 1.5 over here which becomes 1 over here.

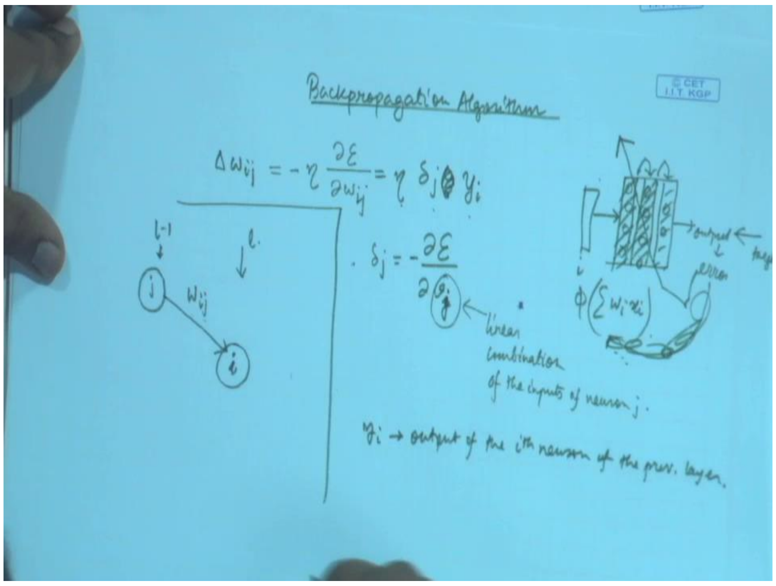
So, as this is the positive value, the output is going to be 1. However, just like change the change of flip 1 of the bits let us push in 0 and 1. In this case, it is going to be again just 0.5 again still its greater than 0. So, this case is also covered right and similarly this case will also be covered. So, this was already shown this is already this was just shown now and this also works in the same way. So, you put 1 0 over here 0 here the output is still greater than 0 and if we have z both 0s both the inputs are 0 in this case the number over here is going to be 0 plus 0 minus 0.5, which is minus 0.5. So, I choose a value of bias in such a way that this value over here becomes negative only if both the inputs are 0 and in this case the output is going to be 0.

So, again like this condition is also satisfied. This is the neural network; this is perceptron which models the all function. You will be asked questions as this way in the class in the exam in which you will be like given perceptron with the certain values the certain distribution of weights. So, w 1, w 2, w 3 will be given. So, there values will also be given and the particular value of the bias will be given and you would have to evaluate what particular Boolean function its modeling or you would be asked to design perceptron the way I did over here which models are particular given Boolean function.

So, the truth table will be given and you would be asked to construct this link, the perceptron. So, both ways given a perceptron construct the truth table or given the truth table construct the perceptron. So, this kind of questions will come from this part.

Let us go ahead to multi layer neural networks. So, multi layer neural networks and has been thought in the class that single neural network with sufficient non-linear hidden units is capable of modeling Boolean functions of any degree of complexity. So, that is why neural networks are universal approximators. So, this quantity, this thing has been already been thought in class and I just want to remind you.

(Refer Slide Time: 18:37)


l-1
l
j
w_{ij}
i
output
error
$\phi(\sum w_i x_i)$
© CET I.I.T. KGP

Backpropagation Algorithm

$$\Delta w_{ij} = -\eta \frac{\partial \mathcal{E}}{\partial w_{ij}} = \eta \, \delta_j \, y_i$$

$$\delta_j = -\frac{\partial \mathcal{E}}{\partial \vartheta_j}$$

ϑ_j ← linear combination of the inputs of neuron j.

y_i → output of the ith neuron of the prev. layer.

Let us go the back propagation algorithm which is our next topic back propagation algorithm is invited in 1989 around a time by a person called Geoffrey Hinton and Rumelhart. So, these people where the founding fathers of deep learning in the way it exists in nowadays.

So, artificial neural networks when they have a lot of number, lot of hidden units and their deep, they have several hidden layers. So, in that case the number of parameters becomes so large that and if the task that you are trying to solve during the neural network is also complex then it becomes very tough to optimize the laws functions this optimization problem. So, doing this gradient descent becomes tough right and back propagation algorithm is a very elegant learning rule that leverages the chain rule of derivatives and it was like the revolution, after introduction of this back propagation algorithm the training of artificial neural networks became very more bit very more easier than before and newer and newer tougher and tougher problems could be solved using these neural networks.

Let us have a look at how the back propagation algorithm says, I would not go into the derivation as you can already always like look up the book, the text book and see the derivation how can you do it yourselves because this is the simple chain rule of derivatives, what I am going just give you is the update rules. So, bang in back propagation what you have to do is or in just like gradient descent you have to find an update for each weight w right. So, let us have this convention that you have a neuron in a particular layer.

So, this layer l you have this neuron i right and it receives in the previous layer l minus 1 there was another node it was the jth node say and the weight that takes the activation of the jth neuron to ith neuron of the next layer is written as is represented as w i j. So, the relation is right to left, w i j means that it connects the edge which goes from the jth neuron of 1 layer to the ith neuron of the next layer. So, j to i right and in this case, after how does the training of artificial neural networks happen?

So, first you have this neural networks say in to which you have several layers of nodes finally, you have an output and you what you do is like you present the say the ith training example over here. So, how the forward propagation happens? You simply do like this like, w i x i summation over that and you pass it through the activation function and this you do at every single layer right. So, a first the activations of this layer are calculated the first hidden layer then these activations go as input to the next hidden layer and the activations there are calculated using same rule.

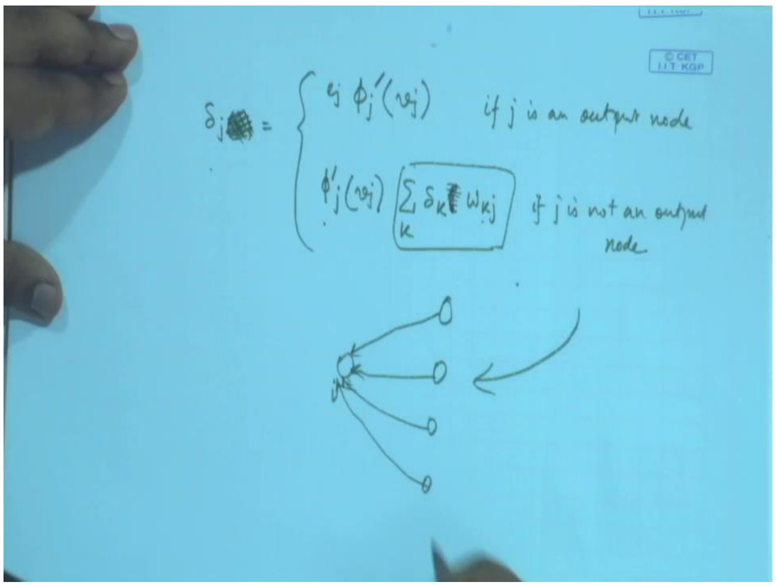
And then you go on to the next layer. So, this is called a forward propagation algorithm and at every singles layer you do a linear combination of the inputs followed by a non-linear transformation, which is given by the activation function of the different nodes of that particular layer right and finally, you have the output and you compare this output with a target value the value that was intended right the correct value and then whatever error was incurred you use that error and no, I should work this way. So, this error goes and it some it you know updates the network parameters cool.

So, these have been already covered in class I would not go into the details, but let me summarize the learning rules. So, w delta w i j is going to be updated as minus eta times gradient of error with respect to w i j correct and this quantity can be further written as this times y i where delta j is equal to negative of gradient of e with respect to a quantity called v j, where v j is equal to just the linear combination.

I am following the just for your information. So, in case you are confused with my notation my notation is quite standard and it is the same notation that was introduced by the standard text book of neural networks which is neural networks are comprehensive foundation by Simon Haykin, I will put a link of this text book in the description of the video. So, that you can go and look up right.

So, and this is a pretty is you know common textbook and may be you are already familiar with this notation if you are into in neural networks. So, the quantity v j is a linear combination of the inputs of neuron j and just calculates yes. So, this quantity again like this update rule it boils down to eta times delta j times y i and y i is the output of the ith neuron of the previous layer.

(Refer Slide Time: 25:37)



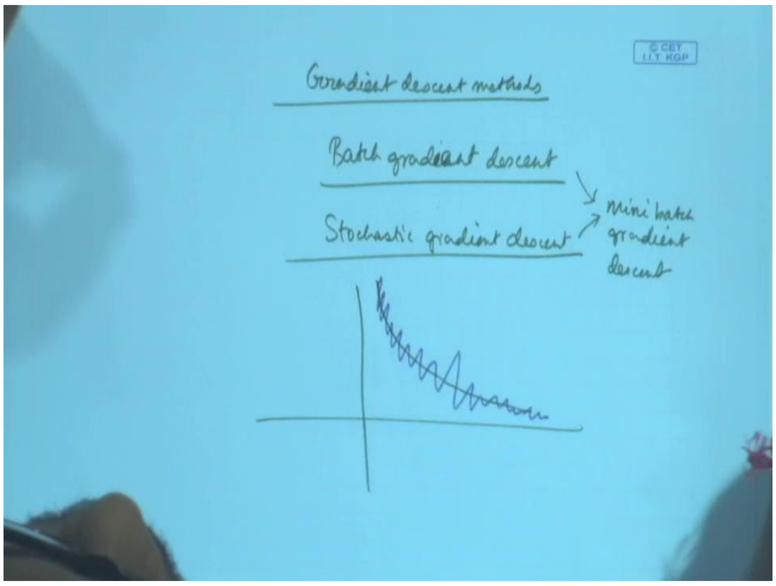
$$\delta_j = \begin{cases} e_j \, \phi_j'(v_j) & \text{if } j \text{ is an output node} \\ \phi_j'(v_j) \boxed{\sum_k \delta_k w_{kj}} & \text{if } j \text{ is not an output node} \end{cases}$$

So, this is the update rule right and this update rule can be written as delta j n equal to it takes the values of. So, let me forget the in the (Refer Time: 25:45) number or the example number just simply write v j if j is an output node. So, e j is the error. So, if the jth node the particular neuron that we are considering right now is an output neuron then you are going to calculate what particular error that particular node had incurred. So, e j is that error term and if it were not an output node then it would be phi dash j of v j times summation over k delta k into w k j. So, this is nothing.

So, what you basically doing is as you are doing like you are propagating the error you going from the output side to the input side right. So, for calculating this value, the delta value we call it the error signal. So, this delta value of the ith node, ith neuron of a particular layer what you are going to do is you are going to calculate the delta value of this node in terms of delta values of all the node of the next layer. So, each of these delta values of the next layer have to be linearly combined with the weights that were connecting this node to the previous last to the corresponding nodes of the next layer. So, this is what is happening over here.

So, the delta values of the next layer are been linearly combine with the same weights as before and to get this linear combination and this is multiplied by the derivative of the you know of the activation function with respect to the previous value of v j right. So, this is how in this update rules happen if j is not an output node. So, this is the learning rule and in the exam you can expect questions in which you will be given a very simple scenario in which you have to calculate the delta values and update rules for different nodes right cool

(Refer Slide Time: 28:29)



So, in the next topic that we are going to just mention is different gradient descent methods. So, this back propagation algorithm is an adaptation of the gradient descent algorithm right. So, this gradient descent algorithm can be executed in two major variants alright. So, the first variant is called batch gradient descent in which you update the weights and the parameters of your model after all the examples have been forward propagated all the examples have been shown to the model.

So, if our propagate all the way all the examples of the training set you calculate the errors you accumulate the errors and then you back propagate and update the weights. So, if you are updating after showing the entire training set to the neural network the that is known as batch gradient descent; however, batch gradient descent is the ideal thing that is going to be according to the theory of gradient descent all right; however, however batch gradient descent is has several drawbacks for example, batch gradient descent is extremely slow right.

So, before every update you have to show all your training examples to the network all right and this many training examples the number of these training examples could run into millions. So, in this case batch gradient descent. So, each update can happen after all the million examples of our propagate. So, that a lot of time right.

So, batch gradient decent is slow next is it could get stuck in local minima it has been observed that batch gradient descent with a small learning rate has the k should be happens to get stuck in local minima and its always not it is like you always un recommended you should not use batch gradient descent ever it is kind of like impractical and (Refer Time: 30:29) it does not work well right though it is a theoretically sound; however, there is another variant which is called online gradient descent or stochastic gradient descent.

So, this was just this was proposed just to you know approximate the batch gradient descent algorithm. So, batch gradient descent is the ideal thing, but it is not fusible computationary most cases and it is bad. So, we stochastic gradient descent attempts to approximate the effects of batch gradient descent by you know updating after every single example as been shown. So, in batch gradient descent you have to update after all the examples have been shown to the network in stochastic gradient descent you update after every single example that after you know forward propagating every single example through the network.

So, in that case the updates are happening much more frequently. So, previous in the previous case for a data set of 1 million size 1 update would happen after all the hundred or 1 million examples have been forward propagated in case of stochastic gradient descent in the same time roughly in the same time 1 million updates would have happened right. So, that is the big deal right.

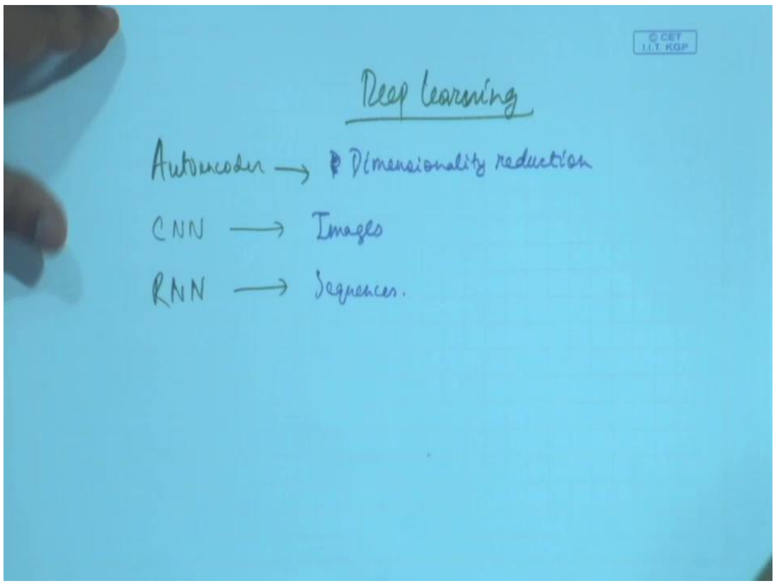
So, that is the big gain and it actually works much better and if you can properly you know control the learning rate and things like that and the other parameters of the optimization algorithm that it can then it can actually like a over you know jump out of local minima and actually need to a better solution than batch gradient descent cool. So, this is all about stochastic gradient descent wait well there is a trade of between the two. So, stochastic gradient descent is like you know batch gradient descent would lead of be the optimization in this mode way; however, stochastic gradient descent is going to both follow the same process same you trajectory in mean, but it is going to be a bit rough.

But in mean it is going to approximate the batch gradient descent only it is theoretical. So, this rough projector is actually help in the optimization by jumping out of local bad minima right. So, I will put in the another link in the description of this video which leads to which gives you a set of really cool and you know practical gradient descent algorithm alternatives right. So, all other alternatives to batch gradients descent and stochastic gradient descent which have been developed over the years and are currently the state of the art in deep learning literature right and those things are not going to be included in your exams. So, in your exam you only have batch gradient descent and stochastic gradient descent that is it.

So, there is a trade of between the two. So, batch gradient descent updates after the entire training set stochastic gradient updates is that the other pool. So, it updates after every single examples there is something called mini batch gradient descent which is a trade of between the two. So, the mini batch gradient descent you update you first split up split the entire training set million examples you have in your training set it split that into you know subsets of 100 or 1000 examples.

So, these are mini batches of thousand examples and after each of these mini batches have been forward propagated right the 1 update is done to the weights. So, for our training set of size 1 million batch gradient descent would do 1 update after million examples have been shown stochastic gradient descent algorithm would do 1 million updates after all the million examples have been shown and the mini batch gradient descent with batch size of 1000 would do 1000 that is 1 million by 1000 updates right after the entire million examples have been shown. So, you will get questions from in the exam about these algorithms.

(Refer Slide Time: 34:42)


Deep learning

Autoencoder → Dimensionality reduction

CNN → Images

RNN → Sequences.

Let me give you a quick look at different deep learning architectures. So, you have been, you would not get questions in the exam from deep learning architecture. I am just going to tell what different things deep learning architectures do. So, I would not write anything I would just tell you. So, the first architecture that ma'am talked about in the class was about auto encoder. So, auto encoders are a neural network class of neural networks which have been developed for unsupervised learning, neural networks are you know have been made for the architecture of the neural network is to do supervised learning.

So, you have inputs and you have labels you try to map between the two model of function which maps the inputs to the labels right in auto encoder what happens you do not have labels. So, you present the same input at the output side and you ask the auto encoder to reconstruct the input. So, this is the input and this is your output. So, you ask the auto encoder in between to you know compress the input and decompress it into the same shape and make sure that the input is reconstructed back at the output with minimum distortion.

So, you are like taking the input through a bottle neck and pushing it back to its original size with minimum deformation all right. So, in the bottle neck in between you compress the entire information of the input into very small size right and that acts as happens to work as a good representation a compact and efficient representation of the input in many cases, but in not all cases right.

So, auto encoders are use for dimensionality reductions. So, auto encoder predominately it is used for dimensionality reduction that is first squeezing the information of the input into a small shape the second thing that I am talked about our conventional neural networks. So, conventional neural networks have other kind of neural network architectural that have been developed for dealing with data which appears like a grid in the form of a grid right like images. So, images like a grid of pixels.

So, conventional neural networks are made for this kind of data for images, you can say custom made for images and they happen to work really well and they have different kind of layers like conventional pooling and so I am going to put a link in the description to a very nice tutorial of conventional neural networks and currently they are the state of the heart in analyzing images right and another kind of. So, this is for images and the last architecture that I talked about our recurrent neural networks.

So, recurrent neural networks are neural networks with memory. So, if you have already realized or you can look up in the web the fact that neural network some multi layer perceptrons, they are memory less. So, they just a modular function from the input to the output and the say input number 1 it produces output number 1 and when the same neural network produces the output for input number two it has already forgotten everything about input number 1.

So, the neural network the multiple perceptron does not have memory. So, neural network the recurrent neural networks have been built in a way that the neural network can actually remember whatever it has seen before. So, the output for input number two would actually be dependent on the input number 1 as well it is going to be a function of the input number 1 and input number 2. So, recurrent neural networks have been custom made for modeling sequences like speech, whatever I am talking there is a recurrent neural network and recurrent neural network can be built which listens to their sequence of sound signals which is coming out from my mouth and convert that into a sequence of characters which could be the transcription, whatever I am talking about alright, the speech to text.

So, you people may be have are already familiar with CD in iphones and Google now in android phones. So, there you have this Google and you start speaking. So, whatever it hears, it records it like it writes down whatever you have spoken and then looks up the web or does some action. So, this kind of speech face interfaces they are for the current point of time and the best speech recognitions systems. You use recurrent neural networks. So, you people may be interested. So, recurrent neural networks are capable of doing incredible things like generating lyrics of rap songs then like humanizing music. So, it listens to a song and generates the music may be or like things like that right and it can even like read up and essay and answer question from that. So, such cool you know actions it can perform.

So, this is recurrent neural networks is first sequences and it can like do pretty good national language processing as well. So, you can read up a length of text review on Amazon, for example, and say whether the person is happy or not with the product. So, pretty good app pretty nice application of neural networks in different areas and I would recommend you to go ahead and perusal neural networks and learn deep learning and use it in your own field for your solving your problems right and I personally feel this, if I like feel great, when I am studying neural networks and I am personally obsessed with this particular topic and hope you people will also enjoy.

So, that is all for today see you in the next video and yes the deadline for the assignment would not be postponed, so no requests in the forum. So, deadline is not going to be postponed at any cost in any condition. So, remember that and solve your assignment fast.

See you next time, bye-bye.