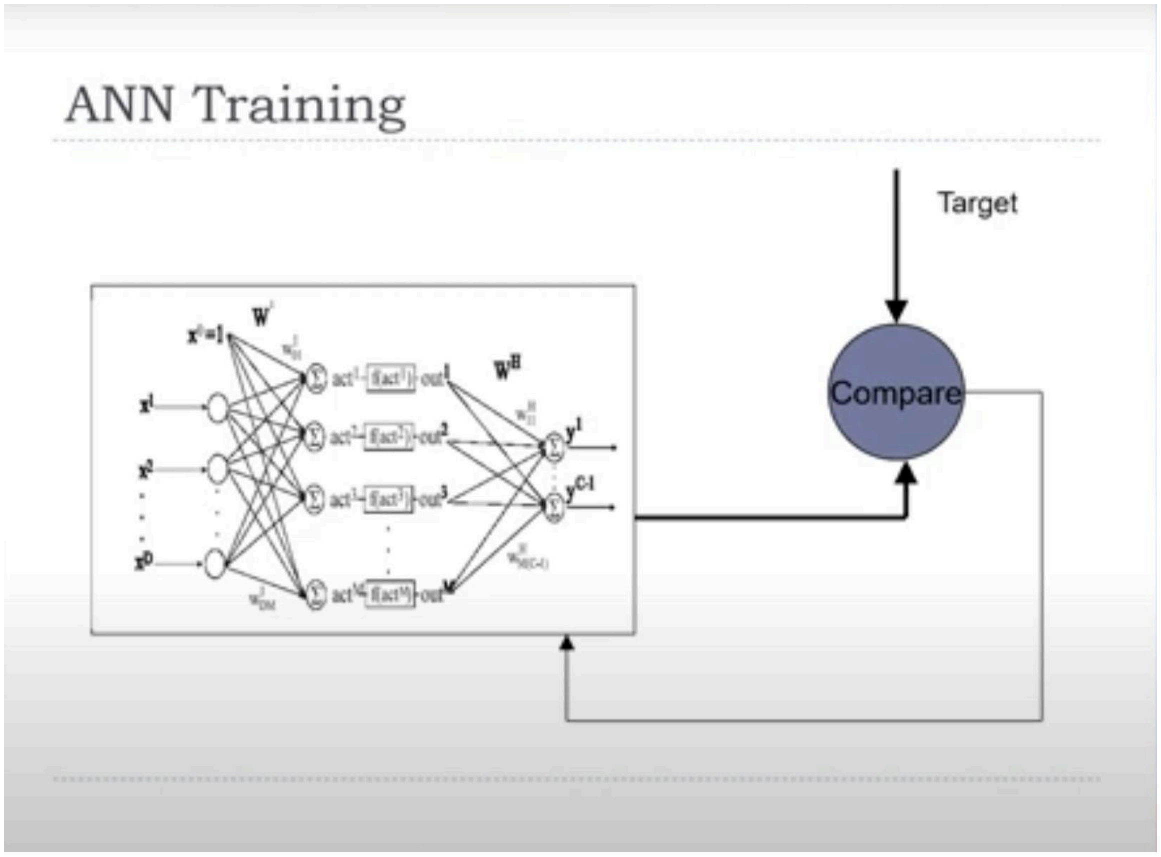
Course Name:Business Intelligence and Analytics
Professor Name:Prof. Saji.K.Mathew
Department Name:Department of Management Studies
Institute Name:Indian Institute of Technology Madras
Week:11
Lecture:41

ANN TRAINING | BI&A | Prof. Saji K Mathew



ANN Training

Now, today we see a bloom or a you know big buzz of AI, but you can see that AI or a neuron was sort of discovered in the 40s and in the 50s there was improvements, but only today we are actually seeing a big boom. And why is it so? The representation of a neuron was done much earlier or neuron structure was discovered much earlier. What delayed? Gradient descent algorithm or these kind of algorithms were actually in the 80s that, they evolved. So, the training algorithm, sorry in simple words the structure of neuron was actually discovered, but the training algorithms to become effective, it took much longer time. You can see about 40 years when the gradient descent kind of algorithm or its variance actually came and of course, it is in the current era that we see large volumes of data with computing power.

It is good to have algorithms, but you also need to have computing power to use large training data to train this algorithms. So, you see that has become possible today. Now, ANN training can be explained using this slide or this graph, picture. So, the idea here is to illustrate how training happens.

You can see that there is a black box here, which is the neural network. The ANN is a software which is designed and as a structure it exists. Now, our aim is to train this network for a purpose and assume, here in the neural networks it is shown that it has d tuples or the size of the data is d. And each data may be actually consisting of multiple attributes. It is ok.

So, let us not really use this, in order to avoid confusion, let me actually use an input x1 here ok. I am inputting an x1 and suppose it is the stock price which we are going to discuss soon. Stock price of an asset say 10 years ago. I have 10 years data, say of IBM or Infosys and it is daily data. So I picked a date, which is 10 years ago.

I have a collection of data, but I am putting x1 as a starting point. And I am inputting x1 to the neural network which is a black box, just do not worry about what is inside . I want this neural network to learn to predict, my purpose is I want to design a neural network and train it so that it predicts future stock prices. So, when I give x1 as the input, what should be my target? I have collected historical data of stock prices, 10 years data, starting from a date that is 10 years back.

I am starting with a historic date. My objective is to train this network so that it learns to predict. So, I am going by tuple by tuple or data by data. And my objective is that the neural network should constantly adjust its weights.

With each time I input the data, it should adjust its weight so that what, an error, there is some error here, right. What is that error? This is the output, neural network will produce some output. And there is some target. What is the target? When I am giving x1 as a target, sorry as the input, what should be the target? The next day's data.

Let me call it x2. Because my data is x1 to xn, 10 years data. So, this is historical data starting at some point of time 1. When I give x1 as the input, what the network should do is, give me the output of x2, x2 is the predicted output.

So, therefore, I am telling the neural network, look when x1 is the input, you should output x2, not o, ok. There is a difference between o and x2 which is e, which is e, the error. Error is o-x2, right. There is an error and you can see that the error is something that is fed back. We call it, another term for this is back propagation or BP.

BP algorithms are, they are called. Gradient descent is a type of back propagation algorithm. Back propagation meaning, the error is propagated back into the network to adjust weights. And we just saw what is the method by which the weights are adjusted. It actually look at the squared error and calculates the differential weight vector to readjust its weights.

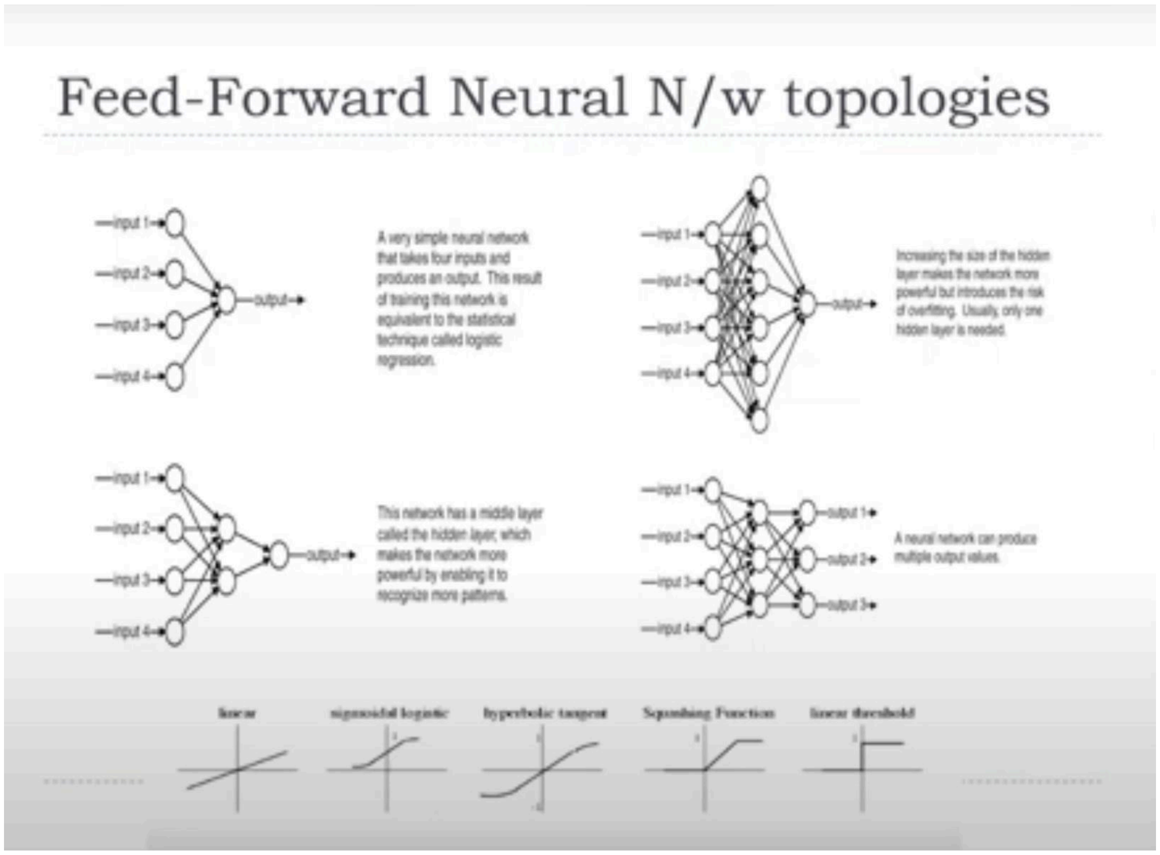
So, the quantum of error determines the adjustment that is required. The error is 0, no adjustment required. And therefore, the purpose of training is to constantly reduce this error. And when the network receive all the n inputs, it has been adjusting its weight to minimize the error each time.

So, one way of adjusting the weights is to adjust the weights n times. And then continue this iteration again. In multiple iterations, n times you did it, do it again do it again. So, that the weights actually get finally adjusted till the error actually the total error comes down or gets minimized. And that is the process of ANN training.

And I have explained it for a forecasting problem. This is a forecasting problem because the purpose is to forecast the next data in time. Next data point in time, that is forecasting.Ok. Now, we are slowly moving from understanding how neural networks function to understanding how neural networks could be applied, how neural networks could be applied to solve problems.

So, we have treated neural network as a black box so far. But it is important to understand neural networks or the network aspect of the neurons a bit more before we apply it. So, the purpose of this diagram is to explain how neurons connect together as layered networks. Layered networks. And this is a category of layered networks known as feed forward, in the sense there is no feedback from the output to the input.

It is only in one direction and this is the type of neural networks that is used for forecasting problems. So we are looking at this closely. This could also be used for classification.



Feed-Forward Neural N/w topologies

A very simple neural network that takes four inputs and produces an output. This result of training this network is equivalent to the statistical technique called logistic regression.

Increasing the size of the hidden layer makes the network more powerful but introduces the risk of overfitting. Usually, only one hidden layer is needed.

This network has a middle layer called the hidden layer, which makes the network more powerful by enabling it to recognize more patterns.

A neural network can produce multiple output values.

So, first of all I said neural networks is layered. What do you mean by layer? Look at the first neural network. It is a layered network meaning it has more than one layer. How many layers are there in this network? There are two layers. This is input layer, this is output layer. So, for any neural network, there should be two layers. There should be two layers, input layer and output layer.

And it becomes a network because they are connected. The neurons from the previous layer are connected to the neurons in the next layer, you can see that. Therefore, it becomes a network of neurons with multiple layers. Now question number one, how many neurons in input layer? This is a design question. How do you design neural networks? How many neurons in input layer? Here four.

What is the basis for determining how many layers should be there? The number of neurons in the input layer is equal to the number of inputs or the number of variables. Number of x variables or number of input variables, number of input variables that solves one problem. Second problem, how many neurons in the output? How many neurons should be there in the output? Depending on how many output variable is there. If it is y equals f x and y being only one variable, there will be one output, output variable. But we see in one network, there are three outputs.

Are there cases when there are more than one variable or one output, which is that case? We discussed that already, in one of the problems where there is more than one output or more than one class, right classification. You have multiple classes, could be two or could be three. So if there are three classes class labels, there are three outputs right. So therefore, it depends on the problem. Therefore, depending on the problem you decide how many input variable, how many output, sorry, how many input neurons, how many output neurons.

Now coming to the second diagram you see, well this is solved, this is solved, but do you see something different here? There is input neuron, there is output neuron, there is a layer in between, there is a layer in between, such a layer would be called a hidden layer. A layer between input and output is a hidden layer, correct. And now we ask this question, why do you need hidden layer? Ok, why do you need hidden layer? Hidden layer is useful in data analysis to capture non-linearity. If you have non-linear data, then in order to capture non-linearity, hidden layers are useful. It is like going back to our first discussion or original discussion on the choice of models.

We discussed a tradeoff between flexibility versus prediction error, flexibility versus prediction error. So, what was our observation? Flexible models actually fit well the data. And particularly when the data is non-linear you need to have some flexibility, so that the bias or the error is less. But when you increase the flexibility, there is also a tradeoff with prediction error. The model over fits and therefore, it is not desirable to have a highly flexible model because it leads to more variance.

And therefore, there has to be some tradeoff between bias and variance, in the choice of models. So, non-linearity is a problem and you need hidden layer, but the number of hidden layers is by, is like choosing the degree of a polynomial. You can go for a very highly non-linear polynomial like the second order, third order, fourth order etc. It becomes more non-linear, but then the problem is of over fitting and therefore, that is not recommended.

So therefore, the question of how many hidden layers? How many hidden layers is a question of determining or solving the bias versus variance problem or too much of flexibility increases prediction error, too less of flexibility fails to capture non-linearity. Therefore, there has to be, you know reasonable non-linearity in the model. So, generally you say 2, 3 hidden layers is what pragmatically one could actually use and then determine how the model performs. It is also empirical, you can try different number of hidden layers and see empirically how the model performs and choose a right number of hidden layers. Essentially hidden layers capture non-linearity.

So, how many layers? Not too many, so that the model does not over fit. The other question is, you see that there are 2 neurons in the hidden layer here. We are also curious to know how many neurons should be there in a hidden layer. For input layer, go by number of variables input variables, output layer go by number of output variables, number of hidden layers, how much of non-linearity you want to capture, not you know like no hidden layer versus 2 or 3, that is also solved. How many neurons, there is heuristics or thumb rules available.

There is a research paper that I recommend you to read from the journal of neural computing which is used, is a paper published way to describing the design of neural networks for the purpose of stock price forecasting. So, I am referring to that paper. So, there is a heuristics recommended in the paper, that is number of neurons in hidden layer is equal to n 1× n 2. What is n 1? Number of neurons in the input layer, n 2 is the number of neurons in the output layer, square root of the product of this, that is a heuristic, not that just like your elbow rule, it is a heuristics. It is something that you can use to assess not exactly, but you can actually use it as a recommendation.

So it is a heuristics rule, practically useful. So, sometimes when you multiply n 1 and n 2, for example, here it is following 4×1=4, square root of 4 is 2. So, this is fine, the design is based on this thumb rule, but do you see that thumb rule here? That is not followed here, but meaning you can actually try, I will try 2, I may try 3 also, 3 neurons in the hidden layer and see if the model is performing better or worse. So, you can try different possibilities, this seems to be very much of, you know the thumb rule, but this is fine, this is fine. So, these are broad guidelines available in the design of neural networks.

What we are doing here is that we are designing a network, in terms of choosing number of input neurons, number of output neurons, number of hidden layers, number of neurons in the hidden layer and another choice is, the choice of transfer functions. We saw that at the output of each layer, there is a transfer function, output of each neuron, there is a transfer function. When we represented a neuron we saw a neuron does a computation and output is determined also by the, is also determined by the transfer function. So, the choices are there are linear transfer functions, sigmoid or hyperbolic, squashing, linear, threshold and so on, like a step function. So, the recommendation based on the paper that we are discussing now, is the output transfer function .

When we are using it for time series sort of an application, the output transfer function should be linear. For other layers, you can choose non-linear transfer functions like the sigmoid and these are choices that you can make or you can try different transfer functions and also test the performance of models. So, this has to be kept in mind when you develop feed forward neural networks for applications that we are discussing in the class. Ok, one more thing before we move on to understand the design, the application of neural networks for time series, there is a way that we can code neural networks or, you know in codify them. Codify means I will write this neural network as a 4 1 network, when you say 4 1 network, this is a 4 1 network.

How do I code this? This is a 4 6 1 network. Number of digits represent the number of layers, number of digits represent the number of layers, very easy coding scheme. Number of digits equal number of layers and each digit represent the number of neurons in that layer. This is input layer, 4 input neurons, 6 hidden neurons, 1 output neuron and 1 2 3, 3 layers ok, that is how you codify a neural network. So, in papers or in text book you will see neural networks will be referred to as a 4 3 1 network or a 4 2 1 network etc, you should immediately be able to decode what does that mean.