

Mathematical Modelling and Simulation of Chemical Engineering Process
Professor Doctor Sourav Mondal
Department of Chemical Engineering
Indian Institute of Technology, Kharagpur
Lecture – 60
Supervised Training

Hey everyone. So, in this this is a follow up class on the neural networks, where we left off discussing about how do we do this training or learning process. So essentially, learning or training is can be done in two types; one is supervised learning, one is unsupervised learning. So, in this class we will focus on the supervised learning; unsupervised learning is something beyond the scope of this course again.

(Refer Slide Time: 00:54)

The slide features a dark blue header with the text "CONCEPTS COVERED" in yellow. Below the header, there are two bullet points with handwritten annotations:

- ❖ Training of neural network model (supervised learning)
- ❖ Optimisation of the weights (unknown).

Handwritten in blue ink next to the first bullet point is the equation $y = f(x)$. In the bottom right corner, there is a small video inset showing a man in a light blue shirt. At the bottom of the slide, there are logos for IIT Kharagpur and NPTEL.

So, this what does this supervised learning actually mean? So, essentially supervised learning is, in a similar way, how do we learn from our teacher? So, teacher, parents, they try to make us learn about different situations of different problems, or our response to different inputs or stimulus; that is how we learn. How I mean, that is how we start to learn to behave, learn to talk, learn to analyze, learn to think, is that the different inputs that we received by our senses is

actually processed. And then we can draw a conclusion and then we can relate that what this actually mean.

And then finally, for some unknown things, we can also try to relate this; based on our previous learning, we try to respond to that. So, in the same way, these ones the this idea of training or supervised supervision of the neural network is essentially estimation of these weights, or the unknowns in our parameter. So, now the model is fully equipped and as you feed some unknowns, it can immediately generate the output for any unknown value also; because, it is now a model equation. This is what the equation tells us if you have an equation y is equal to fx ; and every all the parameters in the effects are known to you.

Then, if you feed into x , it will generate the y as simple as that. Now, if the weights I mean the and the accuracy of this output or the value of the y is is better, when you have better choice or the values of those weights. So, these weights or the unknowns in the neural network model are essentially play a big role in the accuracy of the predictions. So, that is how the, this human thing also works. So, if when you are trying to learn something, the better you learn, the better you learn, the better would be your knowledge; and the better would be your application and your execution. So, the way this is done is there is a method known as gradient descent.

(Refer Slide Time: 03:24)

Gradient descent. weights.

$$f = \psi\left(\underbrace{\sum w_i x_i}_u\right)$$

$$w_i(t+1) = w_i(t) + \underbrace{\Delta w_i(t)}_{\text{correction.}}$$

$$\Delta w_i(t) = \eta \left(-\frac{\partial E}{\partial w_i}\right)$$

learning rate

$$-\eta \cdot 2 \cdot (t_p - f_p) \cdot x_i \cdot \frac{\partial f_p}{\partial u_p}$$

Error $E = \sum_p (t_p - f_p)^2$

train data value (known) NN out put

E vs iteration graph showing a star at the minimum.

NPTEL logo and name at the bottom.

So, let us say the function the activation function, or the function which is the result of the inputs or the output essentially. You can think of f_j is nothing but the output essentially. So, ψ is the activation function and you know that activation function is dependent on the summation of all the cumulative scaled inputs, multiplied with the weights. So, this is the weights; let us simplify it. Let us not write f_j , let us read only f ; so, this is w_i, x_i . So, if there are multiple inputs that is why I wrote f_j ; but this is a simplified case y_i, w_i, x_i . So, this is the cumulative thing and written down as something like new.

So, the error is defined as error E is generally defined as t_p minus f_p or let us says this; let us say f and this is summed over all the different outputs that you are having maybe, so, in a single case. So, this is like the NN output. So, in one case, for single output, there is only one output. And, but if you are changing different input variables, then different outputs will be coming. So, whatever this is, that is a reason why I wrote this I mean, these errors can be summed error also something like that, by changing different data points; or known data you can have different errors also whatever.

And this is the test value or a known value, test value or which is known; and generally square is taken for this error. Why? is because it is squares are always positive; either square error or square relative error, this is what is known as, so the squared relative error. Now the, if you try to do a iteration of the weights, so this W_i, t is the iteration.

So, existing weight is iterated with some correction; let us call that correction as W_i . This is how any just how you try to find out the roots of a nonlinear equation, you try to do the sort of iterations. In the same way, here the unknowns are being calculated based on the idea of the iteration, iterative calculation. And this is the correction to the existing value of the W , this is the correction term.

So, now what is this correction term? This W_i, t is equal to the differentiation of the error with respect to that particular weight, multiplied with a factor η . So, this η if you recall this η is a sort of learning rate we call or the acceleration parameter. If you remember, I think in that multicomponent distillation, where we had the Newton-Raphson method. There was also this parameter α , where we said it is the acceleration. And if this learning rate is too small, then you will need more number of steps or iterations to attain the minimum error condition. So, the errors will be changing with iteration, is not it?

Now, this ideally, we have to find the case when the error is minimum, and take that value of the weights. So, if the learning rate is too small, then the contribution of the correction to the new weight in each iteration will be less. But, one advantage is that if the learning rate is too much, then it may not be able to; so the how does the curve looks like. So, the error that you have with the iteration will be like this, then again it will try to increase. So, we have to locate this minimum position. So, if the learning rate is too high, then it may even that; it may happen that the error overshoots the minimum position. So, that is one downside of a high learning rate.

And a low running rate means it will take a lot of steps, to reach a lot of iterative steps to reach that minimum situation where the error is minimum. So, when error is minimum, that is where we can say yes, that is what we consider the best-case scenario. And we say that the weights are optimized; because for that whatever the weights that we have attained, gives us the minimum error. Sorry, this is not test value, this is train trained data. Yesterday, something after the optimization, we use or check; but this is whatever it is a known quantity, with which we are trying to optimize our weight. Now, the question is that how this dE , dW_i this thing is calculated?

So, you can see that this is equal to in this case, if the error is like this, this becomes 2 minus; this entire thing let we write as, minus 2 minus 2 eta. Then, you have t_p minus f_p x_i , and df_p by d up. Because, ultimately f_p is dependent on u , that is what we are writing df_p by d up. But, in that you are already having this, I mean this is a derivative with respect to W_i ; so, x_i will be coming from there. So, this is essentially the derivative of du by i , i by dW_i ; I hope all of you got that. So, first we are differentiating this, how do we are we getting this? So, essentially let me write it further down.

How are we getting this? So, dE this is equal to by the chain rule if you write this is dE by df_p ; then df_p by d up, then d up by dw . And this last part results into this x_i ; and this part results into the first part eta. Then, you have to 2 into this quantity, the squared term. So, now, the question comes how do you estimate or how do we estimate this this quantity, df_p by d up? And of course, this depends on the kind of activation function you have chosen.

(Refer Slide Time: 12:00)

Least Mean Sq. method (LMS).

So, $\frac{\partial f_p}{\partial w_i} \sim 1. \rightarrow (\psi \text{ is linear})$

$$\frac{\partial E}{\partial w_i} = -2(t_p - f_p) \frac{\partial f_p}{\partial w_i} \cdot x_i$$
$$= -2 x_i (t_p - f_p).$$

$w_i(t+1) = w_i(t) + 2\eta x_i (t_p - f_p) :$
random value @ $t=0$

So, the least mean square method tells you that; sorry, least mean square method also known as LMS method, tells you that this $\frac{df_p}{dw_i}$ by $\frac{d}{dw_i}$ is approximately equal to 1. So, which the consequences that that this activation function is linear; this is the consequence for this case only $\frac{df_p}{dw_i}$ by $\frac{d}{dw_i}$ will be 1.

So, this value of $\frac{dE}{dw_i}$ will be equal to minus 2 t_p minus f_p , this is 1; so, $\frac{df_p}{dw_i}$ by $\frac{d}{dw_i}$ to into x_i that we have and this becomes 1. So, it is $2x_i t_p$ minus f_p ; so, the new weight will be found out by the existing weight and that t is equal to zero iteration. This is a random value, so this is a random value. That is what initialization tells you at the zeroth iteration or initially. And then you have this as, sorry, plus $2x_i t_p$ minus f_p ; this is the least mean square method. Next, we have generalized delta method.

(Refer Slide Time: 13:57)

Generalized method


$$\frac{\partial E}{\partial w_i} = -2(tp - fp) \left(\frac{\partial fp}{\partial up} \right) x_i$$

ψ is sigmoid.

$$fp(1 - fp)$$

$$w_i(t+1) = w_i(t) + 2\eta (tp - fp) fp(1 - fp) x_i$$

become small when E is minimum.
NOT zero



Gradient descent: weights.

$$f = \psi(\underbrace{\sum w_i x_i}_u)$$

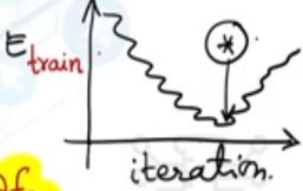

Error $E = \sum_p (tp - fp)^2$

train data value (known). NN out put

$$w_i(t+1) = w_i(t) + \underbrace{\Delta w_i(t)}_{\text{correction}}$$

$$\Delta w_i(t) = \eta \left(-\frac{\partial E}{\partial w_i} \right)$$

learning rate

$$-\eta 2(tp - fp) x_i \frac{\partial fp}{\partial up}$$



This is another method. In this method, once again let me write down this correction factor as $\psi = 2 - f_p$. This quantity is written down as $f_p(1 - f_p)$; and how it is possible that ψ is sigmoidal function. So, in this case, the derivative of the activation function is sigmoidal in nature; then you get the differentiation as $f_p(1 - f_p)$ like this. So, this, so depending on what sort of activation function you have, the different methods are actually related of this error optimization; sorry, of this weights optimization. So, this is the generalized delta method that we see in this case.

There are some other methods also; but these are the generally the popular ones. The most popular one is the least mean square, because a linear function is the most easiest the simplest version. If something does not be described by a linear relationship; then only we go for a sigmoidal or nonlinear versions, exponential ramp and all those.

And not ramp, it is exponential sigmoid Gaussian functions, and accordingly the methods for optimization of the weights also gets modified. So, using this method, we can essentially train our model. So, the as I said, the learning rate is very essential because learning rates controls the contribution of the error correction term to the mean this optimization of the weights.

So, with iteration this correction needs to be added; and based on a situation when the error is finally minimum. So, as you can also see if error is minimum, this quantity that we see here also, $f_p(1 - f_p)$ will actually become small when error approaches when error is minimum. But, not zero, unless the error is zero, this term will not be zero; but, it will be minimum. So, the contribution of the correction at the minimum error also will be small; but please note that it is not a zero.

Since, it is not zero; there is a possibility that these corrections, slowly this correction factor will further deteriorate the solution. If you are continuously adding something to the this optimization of the weight, so the weights will get again modified; and it will again move beyond the minimum error condition.

So, at the minimum, it is not possible that that error is zero; but, it will be small. So, at the minimum location, the contribution of the correction to the weights will be minimum. But, there will be some still some nonzero contributions, it will not be zero. And in that situation, it may

happen that the final weights are slowly deviating from the minimum error condition; and again it starts to change, so the optimized condition is gone.

That is where this learning rate η plays a role. If the learning rate is too small; this region would be very nicely understood or captured; but it will require more number of iterations to reach that point. So, I think this is all that I have to talk about the neural network. Things are best understood as you try to solve a problem on your own. So, you can pick up any dataset around you. And try to prepare a model all by your yourself this neural network model, where you have this sort of, you choose any function, you choose some linear weights. So, if you are having a linear activation function, essentially, the model equation is very similar to the same as the first order model with some bias.

If you are using linear weights, and let us say two or three data points, so not data points; the two or three inputs that you are having or variables that you are having in your system, then you need to frame your own neural network model, and you try to train it. Have some known quantities of your known value with that you try to train; and then you have some known data set for which the answer is known to you.

That is considered a test set over which you apply this, this regression equation with the known regression coefficients or the known weights. We do not call regression coefficient here, we call them as weights. So, with the weights optimized, you apply this model for that known data set, and try to see how is the error.

If you see that the, this test error, so there is something called training error; and there is something called test error. So, whatever we said here, this is actually the train error. Because, during the training, you are optimizing the weights; so whatever error you are getting is a train error. So, train error would be always very small. So, if you apply the same optimized model into the trained set only, the error will obviously be low; because it is low, that is why these are optimized. So, there is no point in checking your neural network on the train data set. You should try to check on the test data set.

(Refer Slide Time: 20:20)

$$E_{\text{test}} = \sum_{x_{\text{test}}} \left| \frac{y_{\text{test}} - y_{\text{NN}}(x_{\text{test}})}{y_{\text{test}}} \right| \leq \text{tol.}$$

- * Validation of any NN model should be always be done using the TEST dataset.
- * Larger is the size of the TEST dataset, the more reliable the NN predictions are.
- * NN with less number of nodes (satisfying the desired accuracy) should be preferred over large no. of nodes.

So, always try to check the error with test data set, E_{test} . And E_{test} is something like this x sorry, the known response as I said y_{test} minus y_{NN} of this x_{test} values, divided by y_{test} . So, these results you see that how it is performing; this gives a performance or of a neural network; and you can really realize this data set, a test data. I mean, you can also have a summation over all the x_{test} values, and see what is this error that you are getting. If this error is smaller than a tolerance value, you accept that model. And you consider that everything is properly optimized and you are getting reasonable outputs.

We already these are checked with test data sets, which for which the values are known; but they are not fed into the system, they act as unknown to your model. So, that is how you, and the checking or the validation of the model should always be done over the test data set. So, the validation of any neural network model should always be done, be done using the test data set. So, test data set is also a part or a subset of the total data set, but it does not contain any values of a train data set. Because on the trained data set, things are already used to optimize your weights, or to find out the unknown coefficients, or the unknown parameters in your model.

So, always the test data sets should be used for validation of the neural network. So, more larger is the size, larger is the size of the test data set; the more reliable the results are, more reliable the neural network predictions are. So, this is something that should be understood. Then, also another thing I must also summarize here for the neural networks, that neural networks with less number of nodes; of course, satisfying the accuracy level.

It is not that we only want low (number), less number of nodes satisfying the desired accuracy should be preferred over large number of nodes. You should at least try to find a situation where the accuracy can be achieved with less number of nodes. So, this is something these are some of the key points that one should always remember in working or deciding how many number of nodes should be chosen for this problem. And also, the number of nodes should satisfy the criteria.

(Refer Slide Time: 24:30)

* The number of train data-set $> \frac{\#Weights}{1 - accuracy}$

weights \sim # nodes for feedforward network & 1 hidden layer.

NPTEL

Number of nodes, let us say should be greater than equal to a number of weights, divided by 1 minus accuracy, number of weights in your problem. This is not number of nodes, but the number of train data set; at least this much should be used. So, number of nodes or number of weights are the same thing essentially unless. So, you can say the number of weights, the number of nodes if you are having just one hidden layer, feed forward network. So, the number of weights is equivalent to number of nodes for feed forward network, feed forward network and one hidden layer.

So, whatever the number of weights is what that matters, because they are only the essentially the the number of unknowns in your problem. So, the number of train data set should satisfy these criteria. And you should pay a close attention to the number of weights, and 1 minus the accuracy for this thing, whatever the desired level of accuracy we want to set. So, less than that it is not a proper training, if you if you do not have this number of data sets to comply this criteria; then it is inappropriate training. Or, in that case you need to reduce your number of weights or essentially the number of nodes.

You cannot have too many number of nodes, if you cannot satisfy this criteria. So, these are some of the essential points that you should always remember while trying to frame, or prepare, or develop a neural network model. So, with this I think we will close this class on the neural network. I hope I could discuss quite some useful; I mean this neural network or the black box model, and this is a very upcoming topic. And slowly there are areas in chemical engineering where you see these neural network models to be utilized. And they are quite useful essentially, when you have systems where too much variabilities and too much of system parameters which cannot be effectively or physically model.

That is why neural network comes in very handy. And these models can also be very useful for control systems also in process chemical engineering. So, I hope this is all that I have to say in this course on process modelling and simulation. Just a quick summary of this course that I would like to highlight is that here we have seen different types of processes starting from mass transfer, heat transfer, different types of physical models, multi-stage distillations. We have also seen in a heat exchanger networks, we have seen a lot of these first principle models; we have also studied different solution techniques, particularly handling PDEs numerical techniques.

We have seen software demonstration using console for solution of PDEs or distributed parameter systems. And then we also seen a software demonstration using Aspen Plus which is a steady state processor simulator, for chemical engineers to solve this industrial problem related to multiple unit operations connected together. So, I hope all of this is very useful to your career. I believe you have learned and gained a good fair amount of knowledge on this. Every of the topics that you we have discussed or touched upon is of course, to some extent on the introductory level. That is because of the limitation of the course time and the scope of this course; it is not a dedicated full course for only a particular topic.

But, I hope this overview of or the introductory knowledge about the different types of model, model formulation techniques, model assumptions will essentially be helpful in your career in different aspects that you are involved in, starting from your project, starting from your higher education problems or problems specific to particular area. You will find some relation or usefulness of this course in the times to come. Thank you and wish you good luck with all your career.