Mine Automation and Data Analytics

Prof. Radhakanta Koner

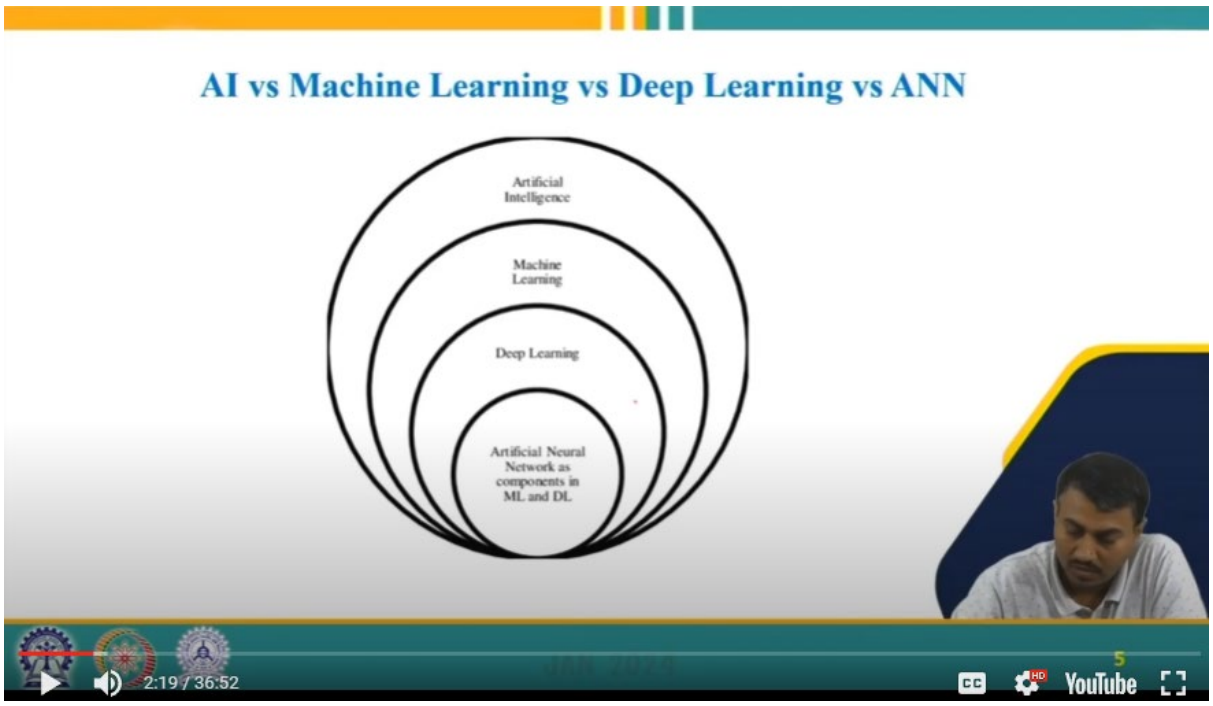
Department of Mining Engineering

IIT (ISM) Dhanbad

Week - 11

Lecture - 51

Welcome back to my course on artificial neural networks. It is one of the very important supervised machine learning approaches that is basically used for classification as well as regression. So in this lecture, we will cover the following: This ANN is basically part of the deep neural network. So we will discuss deep learning, then the artificial neural network and its different components, the mathematical representation of the ANN networks, the different activation functions and their types, and the back propagation methods with their mathematical approach. Then there are a few more important concepts related to the artificial neural network system.



Deep learning is basically a subset of machine learning. This is basically the structure. This is the whole gamut of the problem of artificial intelligence. So out of this deep learning is a subset, and within that artificial neural network is a component of machine learning and deep learning. So ANN is basically a small subset of the overall artificial intelligence system. So deep learning is basically used; this algorithm is basically used for learning the hierarchical patterns in the data, the complex patterns in the data, and these are basically achieved by the neurons and their activation functions, and by that, we are basically getting the abstract features of the data. And

this has a very wide application in image processing, speech recognition, natural language processing, and NLP. This is a very huge amount of use for neural network systems and also for different games like Go as well.

So this is basically used for its capability to capture or map the complex pattern that is within the data. So because of this advantage of the facilities, we basically use the deep learning network. So this is something we have already discussed. Under this, today we will focus on the neural network. So this neural network is basically inspired by the human brain network. In our human nervous system, the central nervous system, we have learned in biology how the signal is being transferred from neuron to neuron and how the synaptic nerve basically maintains the connections and how the information is exchanged. So here we are basically developing or creating artificial neurons that have the capability to process some information, and these neurons are interconnected, and an artificial network is created between each neuron. So here, the neuron is the smallest part, and it is also the part where the computation is basically performed.

So there are several levels of layers of neurons, starting from the input layer to the output layer, and in between we have the hidden layers. So basically, the structure of the artificial neural network is a multi-layer structure, at least 3. And in most of the cases, it is found at 4, 5, 6, and depending on the complexity of the data. So it is a very useful method for classification as well as for regression applications. And the mathematical framework is also very useful for learning, which can be done in a mathematical way for a small data set. So this is the structure; this is the input layer, and we have n inputs starting from 1, and here we are showing 1 output. Typically, for regression purposes, we have only 1 output, but if we are using it for classification, maybe 2, 3, or maybe multiple numbers of output will be there based on the number of classes we are predicting. So the structure remains the same.

So these input layers take the input value and transfer it by some weight, summing over those weights from the first layer to the first hidden layer, then the second hidden layer, and finally converge to the output. Now I have used the term supervised learning. Supervised learning means, I want to mean, that the input data that we are using with each input data has a corresponding level. So that is why it is called a supervised learning method. And these levels are based on the data set. We basically subdivide these data sets into several components, particularly training and testing.

So with the training data set and the level output, we basically match, and this match is to capture the complexity within the data structure or data that is to be captured by the different kinds of activation functions to be used in the neuron. So the neuron basically receives the input signal here and then, using the activation functions, processes and produces the output signal for the next layer. So this way, layer by layer, information is transferred, and the network is learning. So here is the output. So now we have to check whether these outputs match the level data set. So again, we have to come back and update the weights of the learning that has already been developed. So that is also this is feed forward, that is, back propagation, the learning to reduce the loss like that.

So neurons are very much interconnected. As you can see, these neurons are connected with each other in the next layer, or the majority of the neuron layer, based on the structure we basically select. And this connection is basically defined by weight. How much weightage am I giving to transfer the data to the next layer? So each connection has an associated weight that basically modulates the strength of the signal passing through it. So weight basically decides

how much weight is to be given to this input in the next layer. So during training, we basically adjust this weight so that this network can perform at its optimum level.

So this weight is regularly updated with the training set of data, and finally, the network performs very well. So we have several components of the NN; there are three types of layers. One is the input layer, which basically receives the raw input data in the neuron. Second is the hidden layer, which is the intermediate between the input layer and the output layer. And here the main performance, the computational performance, is done on the input data. And it basically extracts the features that are present in the input data. And also, the network learns from the input data here. The output layer is the third layer, or the final layer here. So this output layer basically produces the final output of the network.

And this number of neurons basically depends on the task of the classification. If the number of classes is 2, 3, or 4, the number of outputs would be that or for the regression single output. So this is the structure, and this is basically a picture of a regression in a network. Only one output we are predicting out of N inputs and two hidden layers. So the function of NN is feed-forward propagation. So during the forward pass, the input data is basically propagated through the network layer by layer. And each neuron computes a weighted sum of its input, wij plus x plus b; this is the expression we use. So summation wij xi plus bj. This is j, which is equal to 1 to m. So this is basically the typical expression we use. And here, applying an activation function to the sum passes the result to the next layer.

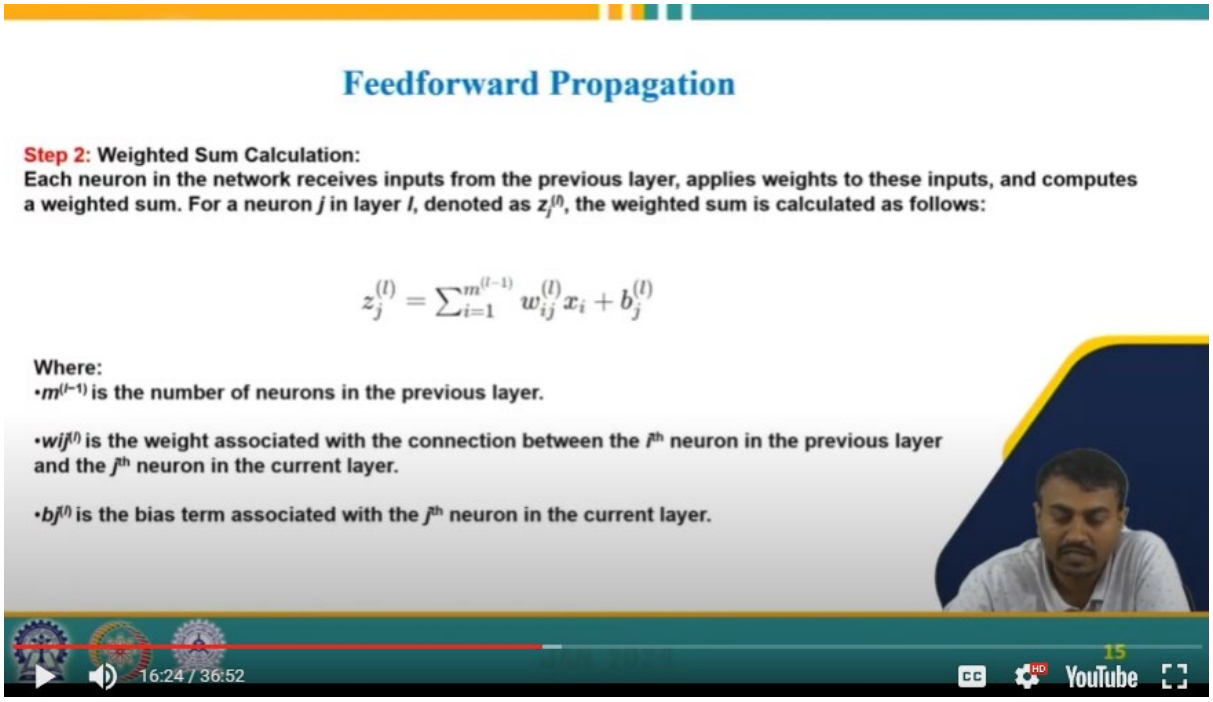
Now comes the activation function. Why do you use the activation function? An activation function is required to introduce nonlinearity into the network. And it allows us to learn the complex patterns that are present in the dataset. So there are four types of activation functions we will discuss today. The first one is the sigmoid, the second one is the tanh, the third one is the ReLU, which is a rectified linear unit, and the final one is the softmax. Training. The process of training NN involves adjusting the weight of the connection to minimize the predefined loss function. So this loss function is basically calculated based on the total error. So we have to reduce the error. So here is the loss function, and this loss function is to be reduced. So here is the importance of the training dataset. So typically, we use an optimization algorithm such as gradient descent. It is one of the most popular methods. And its variant is stochastic gradient descent, or ADAM.

So the next part is the back propagation. So back propagation is the core algorithm that is used to update weights in the network during the training. So it involves computing the gradient of the loss function with respect to the weight and adjusting the weights in the opposite direction of the gradient to minimize the loss. Then comes the learning from the data. So NN learns from the level dataset through supervised learning. And they are presented with input-output pairs. The weights of the network are adjusted iteratively to minimize the differences between the predicted output and the ground-truth data level. So let us start with the feed-forward propagation.

So feed-forward propagation is the process that ensures that data flows in a in a forward direction through the layers of the neural networks from the input layer to the output layers without any feedback loops or recursive connections. And it involves computing the output of each neuron in the network based on the weighted sum of its input and applying an activation function to the sum. So let us break down step by step what we do so that mathematically we can understand. So step 1 is input data representation. So data is X, and basically this vector of X is represented by X1, X2, up to XN transpose, representing a single data point of our input.

So Xi represents the ith feature of the input data, and N is the number of features. So now this is step number 2, the weighted sum calculation. I have already shown you that Wij, Xi, and Bj summations are equal to 1 to N. So each neuron in the network receives input from the previous layer, applies weight to this input, and computes a weighted sum for a neuron j layer of L denoted Zj. The weighted sum is calculated as Zj of the L layer is equal to the summation of i, which is equal to 1 to m of the layer L-1, Wij of L, and Xi plus Bj of L. So here, mL-1 is basically the number of neurons in the previous layer.

$$z_j^{(l)} = \sum_{i=1}^{m^{(l-1)}} w_{ij}^{(l)} x_i + b_j^{(l)}$$



And WijL is the weight associated with the connection between the jth neuron of the previous layer, the ith neuron of the previous layer, and the jth neuron of the current layer. And Bj is the bias term associated with the jth neuron in the current layer. Step number 3 is the activation function application. So once the weighted sum is computed, an activation function f is applied element-wise to introduce nonlinearity into the network. And this is done by ajl, which is equal to Z. Xi is basically the output of the earlier layer inputs and outputs. Now these outputs are taken as inputs for the activation function. function of ZjL. So in the output calculation, the output of each neuron becomes the input to the neuron in the next layer. So this process repeats until the output layer is reached. But the final output of the network is generated. So for a network L layer, L, the output of the network Y cap is computed as the output of the neuron in the output layer. Y cap is equal to AJL.

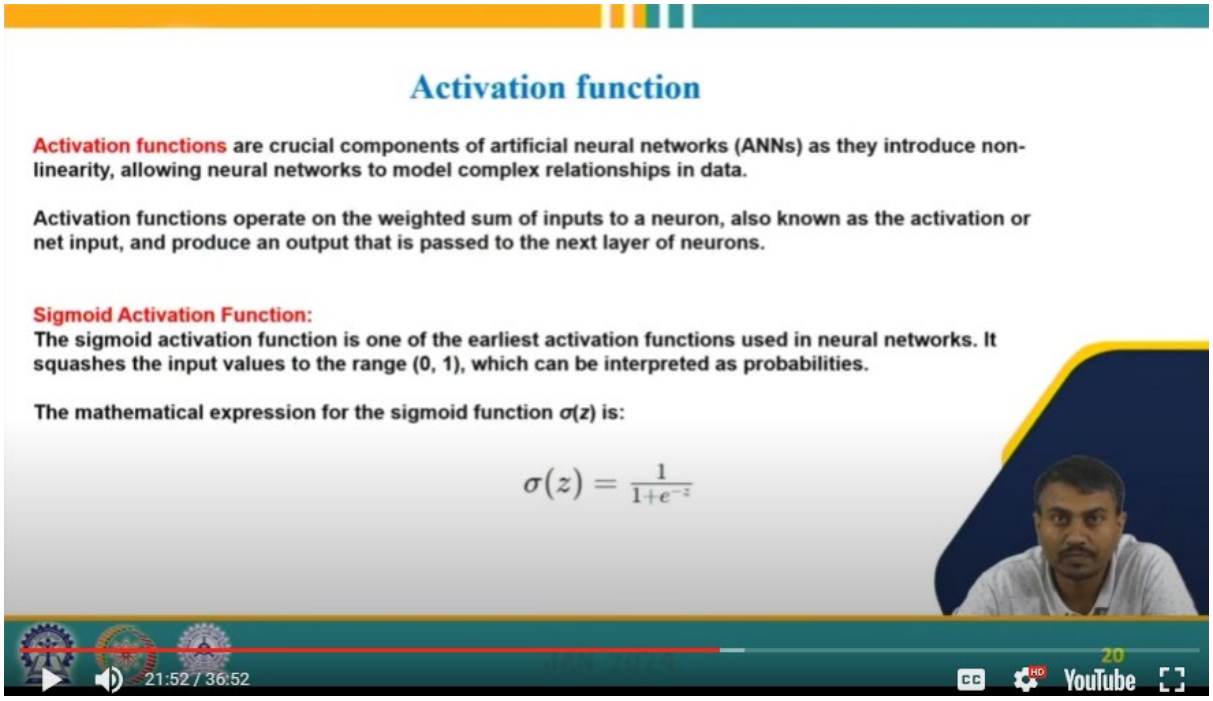
So ajl represents the activation of the jth neuron in the output layer. So let us take an example: a simple network with one hidden network layer and one output neuron. And we assume the linear activation function for the output neuron and the sigmoid activation function for the hidden layer neuron. So input is X1, X2 hidden layer, one layer with sigmoid activation

function, and output is one neuron with linear activation function. So Z1 of 1 is equal to W11 of 1 X1 plus W21 of 1 X2, which is equal to B1 of 1. And A1 is equal to the sigma of Z1 of 1.

So sigma is the sigmoidal activation function. And Wij represents the weight, and Bj represents the biases. Z1 of 2 is equal to W11 A1 plus B1 of 2. So this is basically the output, which is input in the second layer. So Z1 is 2, so Y cap is equal to A1 of 2. Feedforward propagation. So this process of feedforward propagation is repeated for each data point in the data set during training. And the network's predictions are compared to the ground truth levels to compute the loss. The weights and biases of the network are then adjusted using an optimization algorithm like gradient descent during the training process to minimize this loss and improve the network's performance. So it is the weight W1, W2. This is the input here. X is the input. Corresponding weights: this is the bias. And Z is used as the activation function for H1. Now H1 is the input on the output layer, summing over, and the activation function Y of Z2 is the output.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

So this is the summation function used in the functional aspect of the layers. So all weights are summed over, and individual biases are used for each neuron in the layer. And this is the activation function, and finally, with that, we are predicting the final output. So the activation function here is a very crucial component of the ANN, and they introduce nonlinearity, allowing the neural network to model complex relationships in the data. An activation function operates on the weighted sum of input to a neuron, also known as the activation or net input, and produces an output that is passed to the next layer of neurons.



The sigmoid activation function, which is the typical form of sigma Z, is equal to 1 by 1 plus e to the power minus Z. And this basically gives the data between 0 and 1. That is interpreted as a probability. And this is one of the earliest activation functions used in the neural network

system. And Z is the net input to the neuron and output of the sigmoid function, which ranges from 0 to 1. So it smoothes in the S separate curve and allows a smooth gradient during the back propagation. But it suffers from the vanishing gradient problem, where the gradient becomes very small for extreme values of Z. So it indicates poor or slow learning of the neural network. Hyperbolic tangent, or tanH activation function.

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

This is similar to the sigmoidal function; the only difference is e to the power minus Z divided by e to the power minus Z plus e to the power minus Z. And it returns the value between minus 1 and 1. So it also suffers from a vanishing gradient, but the data is basically around 0. So that is why sometimes the hyperbolic tangent activation function is preferred. Another activation function is ReLU, or Rectified Linear Unit. It basically returns 0 for the negative value, and for the positive, it is unchanged. So it is very computationally efficient and helps alleviate the vanishing gradient problem by avoiding saturation for positive inputs. However, it suffers from the dying ReLU problem, where neurons may become inactive.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$$

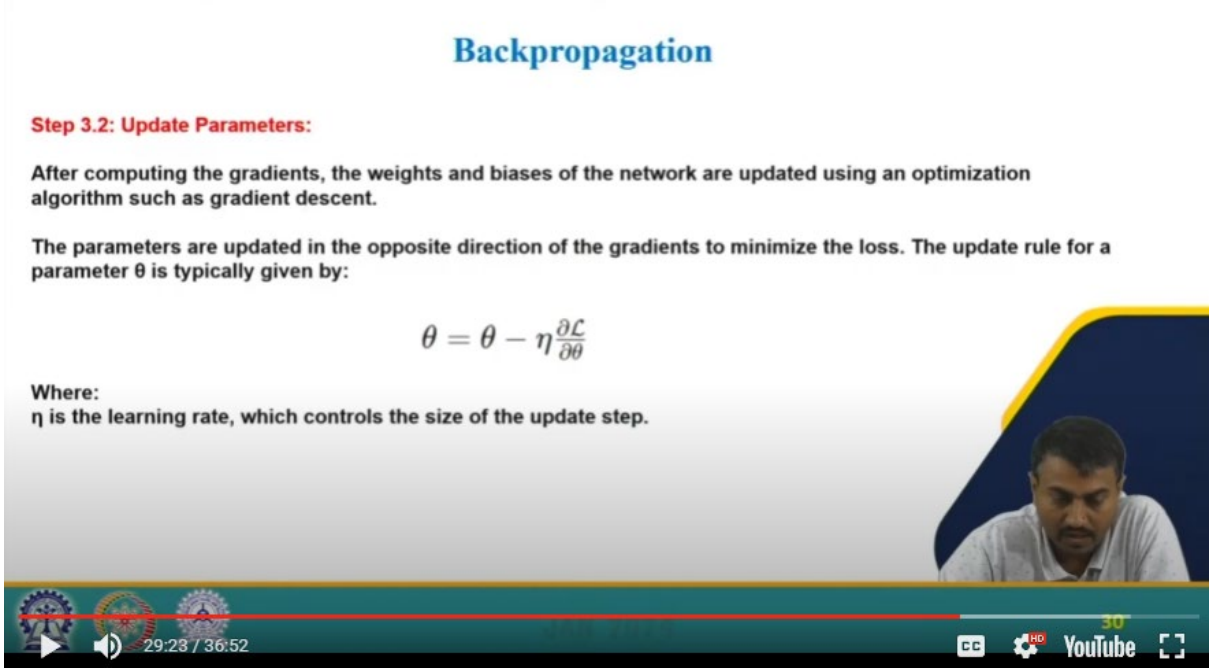
Output is 0 for negative inputs during training and never recovers. Soft mass activation function. The soft mass activation function is commonly used in the output layer of the neural network for multiclass classification. So it takes a vector of raw score logits as input and outputs a probability distribution over multiple classes. So it basically takes e to the power Zi divided by summing over j, which is equal to 1 to n e to the power Zj. So Zi is the raw score of the output, and n is the total number of classes. So the soft mass function ensures the output probability sums to 1 probabilities, making it suitable for multiclass classification problems.

So in summary, activation functions play a very important role in the ANN by introducing non-linearity that allows them to be modeled in complex relationship data. So understanding these properties and characteristics of different activation functions is essential in designing effective neural network architecture and achieving better performance in various machine learning tasks. Now how do we reduce errors? That is the back propagation. Back propagation is the fundamental algorithm used to train ANN, and it enables the network to learn from its mistakes by adjusting the weight and biases of the neuron based on the gradient of the loss function with respect to these parameters.

That is weights, bias, and the parameters. So let us take it step by step. Step number 1, during the forward pass, input data is passed from one layer to the other layer and by summing and adding some bias to the next layer. So let's revisit that. Compute the weighted sum. So Zj of the L is equal to the summation of i, which is equal to 1 to m Wij ai ai plus bj. Applying this activation function, aj, is equal to the function of Zj, and passing the activation forward to the next layer. So here, once the output of the network is obtained, it is compared with the ground truth level to compute the loss using a predefined loss function such as mean squared error, MSE, or categorical cross entropy.

So the back propagation involves computing the gradient of the loss function with respect to the weight and biases of the network and then using this gradient to update the parameter to minimize the loss. So step number 1 is to compute the gradient. So for the for the output layer gradient, compute the gradient of the loss function with respect to the activation of the neuron in the output layer. So dll, divided by the del aj of L, propagates the gradient backward through the network to compute the gradient of the loss function with respect to the activation of neurons in the hidden layer. So dL of ai L is summing over of j dL aj dAj dz dz divided by dA. So this is basically the same representation of the same. We are basically component-wise subdividing the problem. So we basically use the chain rule.

So compute the gradient of the weights and biases. So using the chain rule to compute the gradient of the loss function with respect to weight and bias. So dL of wij is equal to dLw dz dz of dw. dL dl b is equal to dL L and dL z multiplied by dL z and dL b. Then we have to update the parameters. After computing the gradient, the weights and biases of the network are updated using an optimization algorithm such as gradient descent. So theta is the parameter to be updated, and we use the update rule. So theta is equal to theta minus dl-L-dl theta. So nu is the learning rate and controls the size of the update step.



So steps 1 to 3 are repeated terribly for multiple epochs and pass through the entire training data set. And during each epoch, the network learns from the training data and adjusts its parameters to minimize the loss. So in summary, backpropagation is a crucial algorithm for training artificial neural networks, and it involves computing the gradient of the loss function with respect to the weight and biases of the network using the chain rule and updating this parameter using an optimization algorithm using gradient descent. And it has been done iteratively to learn from the new data set as well. Now we want to discuss the bias and variance concepts in the model. So bias refers to the approximation of the real error that is introduced in the real-world problem, which may be complex in a simplified model. So a high-bias model makes strong assumptions about the form of the underlying relationship in the data and may oversimplify the problem. Variance is the amount by which a model prediction would change if it were trained on a different data set. So a high-variance model is sensitive to fluctuation in

the training data and may fit the noise rather than underlying patterns. So a balanced approach should be there between the bias and variance so that the model will perform better.

So what kind of problem might arise that is overfitting or underfitting? So overfitting occurs when the model is too complex and captures the noise in the training data while performing poorly on new and/or unseen data. So these result in low bias and high variance. Underfitting occurs when the model is too simple and unable to capture the underlying patterns in the data, which leads to high bias and low variance. So model complexity—the complexity of a model—is a key factor in the bias variance tradeoff. Increasing complexity often reduces bias but increases variance, as is true for the other cases in the reverse case. Regularization: regularization techniques such as L1 and L2 regularization can be used to control model complexity and find an optimal tradeoff between bias and variance. So cross-validation is a model that we use to get or basically overcome these kinds of problems.

So it identifies the point at which the bias and variance are balanced and models the generalized well for new data. The ensemble approach is also used, just like random forest and gradient descent. We basically combine multiple models, so that can also be helpful to mitigate the weaknesses in these kinds of problems. So understanding this bias-variance tradeoff is crucial for model selection and hyperparameter tuning. So the two commonly used cross-validation methods that are used for handling this kind of situation are basically splitting the training data set or the data set into different parts. We use leave-one-out cross validation and k-fold cross validation, two kinds of methods. So let us go through that. So leave one out. Cross-validation is basically used in the case when each data point in the data set trains the model on all data points except that one and then evaluates the model performance on the omitted data points. So repeat this process for all data points in the data. The advantage of LOOCV is that it provides a nearly unbiased evaluation of the model's performance because each data point is used for both training and testing. It can be useful when the data set is small. It is computationally expensive for the large data set.

For k-fold cross-validation, it divides the data set into k equally sized folds and trains the model k times each time, using k-1 folds for training and the remaining folds for testing. And rotate the testing fold in each iteration until all folds have been used for testing. So this method provides a good balance between computational efficiency and reliable performance estimation. And it is widely used and accepted in practice. The choice of k can affect the model evaluation. So k is equal to 5 or k is equal to 10 is used. So let's summarize.

So ANN is found to be a good model to be used for image classification, image and speech recognition, natural language processing and recommendation systems, autonomous vehicles, financial forecasting, healthcare diagnostics, and robotics. The highlight is that ANN is a powerful computational model capable of learning complex patterns and representations from the data. And with the advent of deep learning, ANNs have achieved remarkable success in a wide range of tasks, making them indispensable tools in modern machine learning and artificial intelligence. So these are the references.

So let me summarize in a few sentences what we have covered. So we discussed the deep learning part of the AI part. We then discussed the ANN in detail, its components and functioning, the different types of activation functions and back propagation methods with a mathematical approach, and finally the important concept of ANN and their applications. Thank you.