# Lecture\_6a

Hello and welcome to the first lecture in this week. We are going to be discussing applying neural networks. In this lecture video, we are going to look at what is a neural network. We are going to look at biological neural networks and artificial neural networks with a focus on the analogy between them. We are also going to discuss how a neural network works with a focus on the perceptron. We are going to be discussing how a neural network works, with the focus on the multilayer perceptron.

What is a neural network? Neural networks are widely used for implementing machine learning and

conventionally we can define a neural network as a model of reasoning based on the human brain.

As we all know, the human brain consists of densely interconnected set of nerve cells, which can also be called basic information-processing units, that are generally referred to as neurones. The human brain incorporates nearly 10 billion neurones and 60 trillion connections, also called synapsis between the neurones, and by using multiple neurons concurrently, the human brain is able to perform its functions much faster than the fastest computers available today. Each neuron in the human brain has a very simple structure, but a phalanx of such elements constitutes a tremendous processing power. By definition, the neuron primarily consists of a cell body, also called sama, a number of fibres called dendrites and a single long fibre called the axon as illustrated here. We can also consider the human brain as a highly complex, nonlinear and parallel information-processing system.

Information is stored and processed in the neural network simultaneously throughout the whole network rather than at specific locations, as we would have in the human brain. In other words, both data and its processing are global, not local. Learning is a fundamental and essential characteristic of biological neural networks and the ease with which they can learn led to attempts to emulate a biological neural network in a computer. Again, a typical neural network consists of a group of nodes or process, also called neurones, artificial neurones which are analogous to biological neurones in the brain, thus the name artificial neural network (ANN). Then the neurones are connected by weighted links which pass signals from one neurone to another in a way a makes a typical neurone network a universal function approximator. It means that for the right combination of nodes and connections, a typical neural network can be set up to model any input an output relationship. In a neural network, each neuron receives a number of input signals through its connections. However, it should be noted that it never produces more than a single output signal. The output signal is transmitted through the neurone's outgoing connection, which corresponds to the biological axon in the human brain when, and the outgoing connection, in turn, splits into a number of branches that transmit the same signal, the same output now and the signal is not divided amongst these branches in any particular way, and as stated here, we have the soma in the biological neurone network being analogous to the neuron in the neuron artificial network, and we have dendrites in the biological neural network being analogous to the input in the artificial neural network.

We have the axon in the biological neural network being analogous to the out in the artificial neural network, and we have the synapse in the biological neural network being analogous to the weight in the artificial neural network. Also, it should be noted that the outgoing branches terminate at the

incoming connections of other neurones in the network, as illustrated here. Activation function. The neurone in a typical neural network computes the weighted sum of the input signals and compares this with the threshold value such as theta. Take for instance, if the net input to the neurone is less than the threshold, theta in this case, then the output becomes a negative one if we are using the functions stated below. However, if the net input is greater than or equal to threshold, which is theta in this case, the neurone becomes activated and its output attains a value of positive one

if we are using this mathematical function below. Continuing from the previous slide, the function employed by the neuron to determine its output based on a given input is called its transfer or activation function.

The transfer or activation function described in the previous slide and shown again below is called a sign function. In neuron networks, there are several approaches to how neurons calculate the weighted sums of their inputs, add biases and then decide whether they should be activated or not

and this gives rise to a number of transfer or activation functions from neural networks. Some of the very popular ones are shown here. The step function, the sign function, sigmoid function and the linear function. We also have the mathematical descriptions clearly stated there for each one of these activation functions, the very popular ones. The step and sign activation functions are also called hard limit functions, illustrated again and they are used in decision making neurones for classification and pattern recognition tasks. The sigmoid function works by transforming the input to the neurone, which can have any value between plus and minus infinity into a reasonable value in the range between zero and one as illustrated there. Neurones within this function, that is the sigmoid activation function are used in the back propagation networks to be discussed very shortly. The linear activation function works by providing an output equal to the neurone's weighted input.

Neurons with the linear function are often used for linear approximation and it's illustrated again there. The perceptron. Now the question is this: Can a single neurone learn a task in the context of neural networks? In 1958, Frank Rosenblatt introduced a training algorithm that provided the very first procedure for training a simple artificial neural network, also called the perceptron. And by implication, the perceptron is the simplest form of the neural network. As shown here, it consists of a single neurone with adjustable synaptic weights and a hard limiter. The mode of operation of Rosenblatt's perceptron is based on McCulloch and Pitts neurone model and this model consists of a linear combiner, followed by a hard limiter as shown here. So we have the inputs, x1 x2. They are weighted. They are combined, and we have a hard limiter before the output is decided. Again, under risk of repetition, the weighted sum of the inputs is applied to the hard limiter, which produces an output equal to positive one if its input is positive and negative one if it is negative. The primary goal of the perceptron is to classify inputs such as x1, x2 until the number of observations that we have into one of two classes, such as a1 and a2, as illustrated here.

As you can tell, because this is classification, it's definitely supervised learning because of the categorical data. In the case of an elementary perceptron, the N – dimensional space is divided by hyperplane into two decision regions, as illustrated here. The hyper is defined by the linearly separable function stated there. Now the question is this: how does the perceptron learn its classification tasks? It does this by making small adjustments in the weights to reduce the difference between the actual and desired outputs of the perceptron. Initial weights are randomly assigned, usually in the range of -0.5 to +0.5 and they are then updated to obtain the output consistent with the training examples or the observations. If at attrition p, the actual output is Y(p) and the desired output is Yd(p), then the error is given by the difference between the desired output and the actual output. Iteration p here refers to pth training example presented to the perceptron. If the error is positive, then the output of the perceptron is to be increased. However, if the error is negative, then the output needs to be decreased. This is the learning rule for the perceptron as put forward by Frank Rosenblatt. A is the learning rate, which is a positive constant less than unity. W are the weights. Ep is the error as it is before an p is as defined before. Again, at the risk of repetition,

the perceptron learning rule was first proposed by Rosenblatt in 1960, and by employing this rule, the perceptron training algorithm for classification tasks can be derived. The training algorithm for the perceptron. The first step is initialisation. The initial weights and threshold are set to random numbers in the range -0.5 to +0.5. As before, if the error is positive, the output of the perceptron is increased and if the error is negative, the output of the perceptron is decreased. The second step is activation.

The perceptron is activated by applying the inputs and desired output and the actual output at the first iteration is calculated using the step function as the activation function. Where n is the number of perceptron inputs or number of observations and step is a step activation function, as I've mentioned already. The third step is weight training. At this stage, the weight of the perceptron are updated using this mathematical relation. Where Δwi(p) is the weight correction at the p iteration. The weight correction is computed using this mathematical relation. Δ is still the learning rate, e(p) is still the error as before. And the forth step is the iteration, which is just an increment in p by one and the procedure is repeated while going back to the second step until convergence. The perceptron is also capable of learning logical operations, particularly the and and or operations as illustrated here. However, it can not learn exclusive-or. Multilayered neural networks,

Conventionally, neural networks have nested mathematical functions. Take for instance, if we have y = fnn(x), fnn(x) is just a depiction of several functions of x which are nested together into fnn. In other words, we have f1, f2 and f3 all nested to have fnn, and f2 and f3 are vector functions, which can take the following from: where l in this mathematical equation is the layer index that can span from one to any number of layers and the function gl is the activation function, as previously discussed. To have a very good understanding of how multilayered neural networks work, a common architecture called multilayered perceptron or the vanilla neural network is going to be examined. Generally, a multilayered perceptron is a feed-forward neural network. A feed-forward neural networks are a variant of neural networks. A multilayer perceptor is characterised with one or more hidden layers. In other words, it consists of an input layer, at least one middle or hidden layer and an output layer as illustrated here. And the input signals in a multilayer perceptron are propagated in a forward direction on a layer-by-layer basis. In a multilayer perceptron, the hidden layer hides its desired output. As a result, neurons in the hidden layer cannot be observed through the input-output behaviour of the network. There is no obvious way to know what the desired output of the hidden layer should be. Commercial neural networks tend to incorporate three and sometimes four layers,

including one or two hidden layers. In contrast to commercial neural networks, experimental neural networks may have five or even six layers, including three or four hidden layers and they can utilise millions of neurons. Learning in a multiplayer neural network occurs in a very similar way as for a perceptron as we've discussed. A training set of input patterns is firstly presented to the network.

The network and computes its output patterns, and if there is an error, that is if there is a difference between the actual output and the desired output patterns, the weights of the neural network are adjusted to reduce this error. And this is generally referred to as back-propagation, hence the name, back-propagation neural networks. In a back-propagation neural network, the learning algorithm works primarily in two phases. In the first phase, the training input pattern is presented to the network input layer.

The network propagates the input pattern from layer to layer until the output pattern is generated by the output layer. If this pattern, that is the output pattern is different from the desired output, an error is calculated and then propagated backwards through the network from the output layer to the input layer as illustrated here. Weights are modified as illustrated here as the error is propagated. To have a clearer understanding of how the multilayer neural network works, we need to see how the training algorithm works. This is the back-propagation neural network. The first step, as we've had in the perceptron is initialisation. All the weights and threshold values of the network are set to random numbers uniformly distributed inside a small range, typically -2.4/Fi + 2.4/Fi.

Where Fi is the total number of inputs of neuron i in the network and the initialisation of the weights is done on a neuron-by-neuron basis.

The second step is activation. At this step, the back-propagation neural network is activated by applying inputs and desired outputs. The actual outputs of the neurons in the hidden layer is calculated using this mathematical relation which is based on a sigmoid activation function as discussed before. N as the number of inputs of neuron j in the hidden layer, and sigmoid, like I said, is the sigmoid activation function. The actual outputs of he neurons in the output layer are then calculated, also using sigmoid as the activation function, using this mathematical relation, where m is the number of inputs of neuron k in the output layer. The third step is weight trained. At this stage, the weights in the back-propagation network are updated by propagating backward the error associated with the output neurons.

To do this, the error gradient for the neurons in the output layer are calculated using this mathematical relation and the error there is the desired output minus the actual output, very similar to the perceptron. The weight corrections are calculated using this mathematical relation where alpha is the learning rate and the weights are updated and the neurons using this mathematical relation. After we've done for the output layer, we also need to calculate the error gradient for the neurons in the hidden layer as the errors are propagated backwards and this is the mathematical relation we use. Weight corrections are calculated, remember alpha is the learning rate.

Just as we've carried out at the output layer, we also need to calculate the error gradient for the neurons in the hidden layer as the errors are being propagated backwards in the neural network

and this is the mathematical relation that is used. Weight corrections are also calculated. Remember, alpha here is the learning rate and the weights at the hidden neuron are updated according to this mathematical relation.

The forth step is iteration and at this stage the iteration p is increased by one

and the algorithm goes back to step two and repeats the entire process until the selected error criterion is satisfied. In this video we've discussed what is a neural network. We've also discussed biological neural network and artificial or a network, and we've tried to establish the relationship between these two types of neural networks. We've also discussed how a neural network works which focus on the perceptron. We've also discussed how a neural network works with the focus on multilayer perceptron.