Artificial Intelligence for Economics

Prof. Adway Mitra

Artificial Intelligence

Indian Institute of Technology Kharagpur

Week – 03

Lecture - 14

Lecture 14: Neural Networks

Hello everyone, welcome to this course on Artificial Intelligence for Economics. I am Adway Mitra, an Assistant Professor in Indian Institute of Technology, Kharagpur and today we are going to start our lecture 14, the topic of which is going to be Neural Networks. So, you remember that in the past few lectures we have been dealing with the general idea of how to learn from data that is if we have observations from the past, how can we extract important information from it or important knowledge from it based on which we can make some kind of prediction about the future. So, we have we started off with unsupervised learning which are clustering and we studied some algorithms for it. After that we came to supervised learning we using in which we have learnt decision trees as well as linear classifiers. So, linear classifiers if you remember like we are trying to fit a linear function to the like which I mean we are trying to find a linear function which will map the feature space to the label space and we will do the classification.

So, that every feature vector can be labeled as either $+1$ or -1. So, that is binary classification and if we can do binary classification we can do multiclass classification as well. However, the assumption which we have made here is that this mapping from the feature space to the label space is linear. That is this as under this assumption we can work well only if the data is linearly separable, but in general that is not the case and we may need non-linear functions from the label space to the from the feature space to the label space.

Now, this kind of non-linear functions can be represented by the important concept called neural networks and that is what we are going to learn today. So, today we are going to start with non-linear classifications. We will discuss what is the general structure of a neural network and what are its components. We will see how neural network can work as a function approximator and then we will see how the parameters of a neural network can be estimated by the famous algorithm called back propagation. So, let us start with linear classifier once again.

So, you remember that linear classifier was defined in this way $y = sign(w \cdot x)$. So, what is x? x is the feature vector and what is y? y is the predicted label which is either $+1$ or -1 in case of binary classification. Now, what is w? w is the parameter of the model which basically means the coefficients of the of these features. It also indicates the or the w also indicates the line the linear structure which we using which we aim to differentiate between the different classes of data. That is w; w represents a linear structure such as a line in 2D or a plane in 3D or a hyperplane in higher dimensions.

such that the all the examples whose label is $+1$ should lie on one side of the linear structure and the other labels with whose or other examples whose label is -1 they lie on the other side. Now, this simple linear classifier we can represent it by drawing a graph. So, like we can write it like this that all the features $x_1, x_2, \ldots, x_d$ let or let us say there are 4 features initially. So, like we consider like we represent all of them as a nodes and then like we these are the input nodes and then we have the output node. Now, what happens in the output node? The output node will you will is since the output depends on all the all of the inputs that is all the input features, we connect the output node to all the input nodes by using these kinds of edges.

And we write these as directional edges that is as you can see we have put an arrow indicating the direction of the edge, because the information will flow from the input to the output that is given the input then only we can predict the output. Now, each of these edges they will be weighted edges that is we with each edge we will associate a weight and that is going to be these coefficients and additionally we have the bias term. So, for that we will have this extra node it is basically a dummy node whose feature value is 1 and it also has its edge whose weight is w_0. So, that is like the bias. And now what happens at the output node is that it receives all of these inputs including the bias.

Now it carries out the multiplication followed by addition. What multiplication? Each of the inputs is going to be the multiplied by the weight of the edge along with it which it came that is x_1 will be multiplied by w_1, x_2 will be multiplied by w_2, x_4 will be multiplied by w_4 and so on and so forth. and all of these products will then be added up by a like this sigma function and after that we will apply the sin function on this and that will be our output that is that is the value which will be computed at the output node and that is what we will get. So, this is like clearly the what the output is the same as like what we have the I mean the formula for the linear classifier, we have just represented this whole operation with the help of nodes and edges. Now, so each input dimension is like one input node or by input dimension I mean feature or attribute and all of them are connected to the attributes I mean sorry to all of them are connected to the output node and then each connection edge from the input to the output it carry the weight.

So, this is like the basic the very basic or fundamental structure of a neural network. And so, as you can see it is right now it is just an interesting representation of the linear classifier. Now, the question is can the using the structure if we expand the structure a little bit we can do things or we can represent functions which are more complex than just the linear classifier. Say like consider the situation of this multilinear classifier. So, what is multilinear classifier? It is a like it is a like we can say a combination of linear classifiers which must be taken together to make the prediction.

So, here if you look at this data set. So, you will see that there are red points on this side as well as on this side and in between there are some blue points which are like we label as -1 the red points are labeled as $+1$. So, now I want to separate the red points from the blue points. Now like you can understand that there cannot exist any linear structure which achieves this that is the data is not linearly separable. However, instead of one linear structure if we can use two linear structures say a line like this C_1 another line like this C_2 then we can however like use these two classify.

That is if C_1 predicts $+1$ or if C_2 predicts $+1$ then we can say that the final output is $+1$. So, C_1 predicts $+1$ for the points which are on this side and C_2 will predict $+1$ for points which are on this side. And if like if both of them are predicting -1 in that case the like the point is -1. So, like here the classification takes place through some like it requires some logical operations namely the logical AND or the logical OR. So, this again can be represented with the help of a neural network like this.

The only thing is that in this case we need some intermediate operations. So, like we have a classifier which is which corresponds to C_1, a linear classifier corresponding to C_1, another classifier which corresponds to C_2. So, here we what we are doing is that first we are calculating the result of with respect to each of these classifiers separately C_1, the result with respect to C_1 whether the output is $+1$ or -1 and also the result with respect to C_2 whether the output is $+1$ or -1. And then once we have obtained Then we can apply the AND OR function to get the final output whether it is the final output is going to be $+1$ or -1 right. So, like now you can understand that to do this calculation we need some intermediate step from the input we do not directly go to the output as we are doing in this time, but instead we have some intermediate steps.

We first from the input we first calculate what is the result of C_1 and we also separately calculate what is the result of C_2. Then these two intermediate steps intermediate results we combine to get the final output. So, this is how we expand the idea of this graphical representation or the basic neural network to like represent this multilinear classifier. So, now these intermediate steps we can call these as the hidden nodes of the neural network. So, why hidden because like these are not things which are either the input or the output

these are things which these are calculations which we carry out in like as intermediate steps.

So, that is what is called as the hidden step. Now, suppose from the multilinear classifier we want to go to a non-linear classifier that is suppose the data is like this that is they can there is exist no linear structure which separates the data from the different classes, but there can exist non-linear classifier which separates the data. Like for example, in this situation like where the data is organized in the like form of concentric circles. We may not be able to find any kind of linear structure which separates the two classes of data, but we may find as some circle in between these two circles such that all the points are with inside that circle and all the green points are outside that circle. So, we can like write this as the classifier.

So, like we can we know that this is the equation of a circle. So, for the point the center of the circle is x naught somewhere here and the radius of the circle is or the squared radius of the circle is r square. So, now we know that if there is any point which is inside the circle that is whose distance from the center of the circle is less than the radius, then this thing will then this expression will be less than r or this whole expression will be negative. So, like if we apply the sine function on it we will get -1, but if there are for those points which are outside the circles then we know that this term will be more than r. So, if we subtract it from r we will get something some positive quantity to which we apply the sign we will get $+1$.

So, this will be a successful circular classifier. However, we note that the circular classifier is non-linear. In fact, it is quadratic because it has this x transpose x terms. Now, so obviously, the linear classifier will not work in this case we need a non-linear classifier. So, how do we achieve or how are we able to implement this kind of a non-linear classifier? Now, when we have a neural network like the basic neural network as we have seen already.

So, it like we saw that it generally implements the this the linear function itself that is it calculates the product of the all the input at the input nodes I mean the feature vectors along with their weights and sums them up. So, that is a linear operation and on top of it, it applies the sine function. Now, instead of that if we could somehow apply any non-linear function f on it in that case we would get an output y which is non-linear in x_1 x_2 etcetera. Now, what that non-linear function can be that is a different question, but like we can like the general I think is that we can impose some or we can incorporate some non-linearity in the output of the neural network by like adding us by replacing the sign function which we got earlier by like suitable non-linear function called f. So, now the question is why are we thinking of all these things I mean what is the speciality of this

kind of a structure.

So, this particular structure this is an imitation of what is the like biological neurons which are present in our inside our human body. So, like you would have studied in biology class in school sometime. like a like what a biological neuron looks like it has things like the exons the dendrites and so so on. So, it receives the inputs from the using these dendrites it carries out some some processing of those input signals I mean that the which are the neural signals. And, then it transmits it to for for further processing to other parts of the body using the exon terminals which are like basically our the output nodes.

So, similarly so, the structure of this basic neural network which we which we make also call as a So, this is also some close imitation of this it receives the input and then the inputs is processed and then an output is sent out. So, this neural this very fundamental structure of the neural network it is used to carry out some very specific pieces of operations. And the like so, it the like the aim of this neural network is to calculate a very complicated mathematical function which can be highly non-linear. Now, such a complicated mathematical function we cannot compute directly in one step, but we can distribute it into many small small steps. And, it can be done in a in a hierarchical way where at the first step we may carry out some of the initial preprocessing using various parts of the input.

Now, based on these preprocessings we get some intermediate results which are stored in the first hidden layer. Then the results of the first hidden layer can be further grouped together and further processed to get the outputs of the second hidden layer. Now from that we can go to the third hidden layer and so on until the final results are ready. The final output that may be either a single real number or it can be like a vector also or it can be something even more complex like matrix. in certain situations we are right now we may not need that situation, but we will discuss about that a little later.

So, this is the general structure of the neural network. So, this is the input layer the from the input layer we like the we pass the information to the first hidden layer. And in between this multiplication by the edge weights takes place. So, like the it is we can consider that all the input nodes which basically stands for the different features vector I mean the features in the vector they are connected to all the nodes in this hidden layers using weights which are collected together in this matrix called w_1. Then some of these weights of course, could be high some of them could be low some of them could even be 0 which means that effectively those two nodes are not connected.

So, like here like we can assume that the first stage of computations take place in and their results are stored in the first layer of hidden nodes. Next, these are again connected

to the second layer of hidden nodes. Again there are these edges connecting all the nodes in this layer to all the nodes in this layer using edges whose weights are stored in this matrix called w_2. So, we can in general we can consider that there are h_1 nodes in this hidden layer and h_2 nodes in the second hidden layer. So, there like if we are considering that all nodes are this are layer are connected to all nodes of this layer then total $h_1 \times h_2$ edges will be there.

So, they are edge weights will be stored in this matrix called w_2 which will be of size $h_1 \times h_2$. Similarly, so like here the results of the first layer of intermediate calculations like are further grouped together and processed and the results are the second layer of or the second intermediate results which are stored in the second hidden layer. and and so on and so forth and till and this process goes on till we reach the output layer. So, the calculations can take place like so, like the results of the first hidden layer we can write it in this notation $h_1 = f_1(w_1 \cdot x + b_1)$, b_1 means the bias which we can also denote by w_0. So, w_1 is the edge weights x_1 is the or x is the input.

So, basically a dot product takes place the bias is added and some non-linear activation function f_1 is is applied on it. So, the result is the this d_1 dimensional vector which is stored in this first hidden after that again in like this result h_1 like we apply the w to the edge wedge w_2 on it. So, another dot product on a matrix operation takes place. So, h_1 is a vector and w_2 is a matrix. So, this like the matrix multiplication takes place and we get a new vector.

we add the new bias term and apply a new application activation function f_2 to it to get the new vector h_2 whose size is d_2 and this whole process takes on goes on and on for l hidden layers and finally, we get the output which we can call as y. So, like this is this operation in which the calculations are done from I mean input is provided and then the calculations are done in layers from the right side to the left side. So, from the left side to the right side until we reach the output this is known as the feed forward operation of a neural network. So, as you can see initially the these red these nodes turn red indicating that some values are supplied to it in the input then after that the first layer of calculations is done and we get the values of the first hidden layer that is the first set of intermediate results. After that we calculate the second set of intermediate results which are the nodes of the hidden layer h_2 then the third then the fourth and finally, the last or the lth hidden layer and from that we finally, calculate the output.

The like at the output layer we can again have some function activation function g the calculation remains the same that is you have the edge weights v and the last layer of intermediate results is called as h_l. So, we do this another round of matrix multiplications

we add another biased term we apply the this non-linear function g and we get the final outputs. So, our so like from input to output we are repeatedly doing this matrix multiplication operation and we are applying a non-linear activation function to it. So, that ultimate the final result y is a very complex representation of the input x that is it has undergone a large number of transformations which are which can be non-linear depending on what these functions are and hopefully and and if they are suitably chosen these activation functions if they are suitably chosen this y can be used to represent any arbitrary non-linear function of the input vector x. So, now so this is the general structure of neural network and its functionalities.

So, now when we are building a neural network there are many components in it. First of all how many hidden layers will be there like this L how many like what will be the value of L or then how what will be these numbers d_1, d_2 etcetera. So, means how many units will be there in each of the hidden layers. And third is what about these functions f_1 f_2 up to f_l or g. So, what will be the types of activation functions in the hidden layers.

So, these are decisions which the network designer has to take. So, there is no gold I mean golden rule for deciding these things. So, typically the neural network I mean the machine learning scientist who is developing the neural network they have to choose suitable values for these all of these things. However, there is another set of parameters that is these $w_1, w_2,$ etcetera.

So, like the I mean the weights themselves. Now, these the numbers like $d, d_1, d_2,$ etcetera those are chosen by the machine learning engineer who develops the neural networks, but the edge weights themselves the numerical values of the edge weights like $w_1, w_2,$ etcetera they are not chosen by the neural by the engineer instead they have to be calculated. So, they are like calculating these parameters is essentially the process of training the neural network just like we estimate the coefficients w of a linear classifier through algorithms like perceptron or support vector machine which we discussed in the last lecture with where we are essentially trying to minimize a loss function. Here also theidea is the same we have to choose these parameters in such a way we have to estimating made them from using data by considering a loss function. So, the what is the loss function the loss function it basically compares the neural networks output which is $f(x)$.

with the true value of y. So, what is y? y is the true output corresponding to the input x, what is f? f is the function which is represented by the neural network. So, the neural so, given any input x the neural network it calculates a function I mean it it calculates an output which is known which we can call as $f(x)$. So, now that $f(x)$ is to be compared with the true output y which we should expect to get from the input x and then we

compare them using this loss function. So, what function this L will be depends on the exact problem which we are trying to solve again choice of the loss function is again the task of the machine learning engineer, but we have to choose the values of the weight parameters to minimize this loss function. So, now how is that done? So, like we use this like we cannot like solve this problem analytically that is we cannot just calculate the derivative of the loss function with respect to w and then equate it to 0 it will the it will not be tractable.

So, we have to use approximate or numerical methods such as gradient descent. So, we have to calculate the gradient of this loss function with respect to each and every weight in the or each and every parameter in the model. So, like here a simple example is worked out if the error like if the loss function is the simple squared error as it happens in case of linear regression and if there is only one set of these weights like this. Then we can simply calculate the gradients of each I mean the like each weight age weight by calculating a simple derivative of the loss function with respect to that and then we can update the that weights using the usual form of gradient descent. That is we have some initial value of the weight we calculate the gradient using the at that particular value I mean we we have the formula for the gradient we just plug in the current value of of w_j to it and then we multiply it with this learning rate alpha and we get an updated value of this parameter w_j and this we process we keep on until we reach some sort of convergence for all of the edge weights.

So, that gives us the a local minima of the edge weights with respect to the this loss function. Now, if when we so this is of course, a straight forward calculation assuming a very simple loss function and a very simple structure of neural network where there is just one layer of I mean one hidden layer, but in if we if it is a deep neural network with many hidden layers like this with l hidden layers and so are in modern days a typical neural network has hundreds of hidden layers. So, so and each of in each hidden layer there are many parameters. So, to like a neural current neural network can easily have millions of parameters. So, each of them have to be updated using this process of gradient descends descent.

So, like the in that case we have to apply what is known as the back propagation algorithm in which the weights are updated turn wise from the output layer to the input layer. So, like first the this back propagation is nothing, but calculating or applying the chain rule of differentiation. So, the loss function is of course, calculated at the output layer. So, the first step is to update it should be easiest to update the parameters of this output layer which which are the v. So, we can easily calculate the derivative of the loss function with respect to v because the output neural networks output is like it can be easily represented as a function of this v.

So, we can carry out the derivative and like update this v. Now, we after we have updated v the next task will be to update w_l. So, for that we have to calculate the derivative of the loss function l with respect to the weights in this layer that is with for each of the parameters in the like w_l we have to calculate the derivative of the loss function. So, this will of course, be difficult because the loss function is not directly a function of w_l, it is the loss function is of function of w_l via this intermediate step v, but fortunately we have already calculated the derivatives of the loss function with respect to v. So, we just need the loss function I mean the derivative of v with respect to w l using the chain rule of differentiation. So, like we are effectively doing something like $\frac{\partial L}{\partial v} \times \frac{\partial v}{\partial w_L}$ something like that.

So, we were using the chain rule of differentiation and reusing the values that have already been calculated. So, using the derivatives of L with respect to these V we can calculate the derivative of L with respect to w_L and then those again will be reused to calculate the derivative of w_{L-1} then w_{L-2} all the way back to the like the first few layers w_1. So, this is the famous back propagation algorithm. So, which is used for training the neural network that is estimating the parameters of the neural network. And needless to say deeper the neural network is more of the training data we will need to estimate all these parameters.

So, in conclusion a linear classifier can be expressed as a neural network with nodes and edges each input feature is expressed as an input and a node and output the output is expressed as an output node. The output can be either a single number or it can be a vector also it can be even be more complex like a matrix or a tensor. Addition of hidden layers allows us to express more complex decision boundaries or more complex functions for the neural network. Non-linear activation functions they enable us to express arbitrarily complex functions. And in the feed forward mode computations are carried out in steps from the input to the output.

So, at each hidden layer we have basically some intermediate calculation results which are then passed on to the next layer where some more intermediate calculations are done and so on and so forth until we get the final output. and the like the machine learn the neural network structure has various design choices like number of hidden layers number of nodes in each hidden layers type of activation function and things like that which are chosen by the machine learning engineer but the edge weights that is the parameters of the numerical net of the of this neural network they have to be calculated numerically with the help of algorithms such as gradient descent with the of minimizing some particular loss function. The loss function again has to be chosen by the machine learning

engineer, but those derivatives of the loss function with respect to each of the parameter in the neural network they can be calculated using the back propagation algorithm. So, like so this is the basic story of neural network. So, like in the domain of economics neural networks like they find a lot of applications.

So, we have already seen a number of classification tasks in the domain of economics. So, some like we have when we discussed the linear classifiers. So, in many of those tasks the linear the choice of linearity as a function is often not enough and we need a non-linear function. So, in such situations it is good idea to use the neural networks to get more accurate results in the same task.

So, with that we come to the end of this lecture. In the next lecture which will be lecture number 15, we will discuss some more applications of modern neural networks and how they can be used for the task of time series forecasting which is particularly important in economics. So, till then all of you please take care stay well and we will see you soon bye.