Why do you need non-linear activation functions?

(DESCRIPTION)
Text, One hidden layer Neural Network. Why do you need non-linear activation functions? Website, deep learning, dot, A.I.

(SPEECH)
Why does a neural network need a non-linear activation function?

Turns out that your neural network to compute interesting functions, you do need to pick a non-linear activation function, let's see one.

(DESCRIPTION)
New slide, Activation function.

(SPEECH)
So, here's the four prop equations for the neural network.

Why don't we just get rid of this?

Get rid of the function g?

And set a1 equals z1.

Or alternatively, you can say that g of z is equal to z, all right?

Sometimes this is called the linear activation function.

Maybe a better name for it would be the identity activation function because it just outputs whatever was input.

For the purpose of this, what if a(2) was just equal z(2)?

It turns out if you do this, then this model is just computing y or y-hat as a linear function of your input features, x, to take the first two equations.

If you have that a(1) = Z(1) = W(1)x + b, and then a(2) = z (2) = W(2)a(1) + b.

Then if you take this definition of a1 and plug it in there, you find that a2 = w2(w1x + b1), move that up a bit.

Right? So this is a1 + b2, and so this simplifies to: (W2w1)x + (w2b1 + b2).

So this is just, let's call this w' b'.

SO this is just equal to w' x + b'.

If you were to use linear activation functions or we can also call them identity activation functions, then the neural network is just outputting a linear function of the input.

And we'll talk about deep networks later, neural networks with many, many layers, many hidden layers. And it turns out that if you use a linear activation function or alternatively, if you don't have an activation function, then no matter how many layers your neural network has, all it's doing is just computing a linear activation function.

So you might as well not have any hidden layers.

Some of the cases that are briefly mentioned, it turns out that if you have a linear activation function here and a sigmoid function here, then this model is no more expressive than standard logistic regression without any hidden layer.

So I won't bother to prove that, but you could try to do so if you want.

But the take home is that a linear hidden layer is more or less useless because the composition of two linear functions is itself a linear function.

So unless you throw a non-linear [INAUDIBLE] in there, then you're not computing more interesting functions even as you go deeper in the network.

There is just one place where you might use a linear activation function.

$g(x) = z$.

And that's if you are doing machine learning on the regression problem.

So if y is a real number.

So for example, if you're trying to predict housing prices.

So y is not 0, 1, but is a real number, anywhere from - I don't know - $0 is the price of house up to however expensive, right, houses get, I guess.

Maybe houses can be potentially millions of dollars, so however much houses cost in your data set.

But if y takes on these real values, then it might be okay to have a linear activation function here so that your output y hat is also a real number going anywhere from minus infinity to plus infinity.

But then the hidden units should not use the activation functions.

They could use ReLU or tanh or Leaky ReLU or maybe something else.

So the one place you might use a linear activation function is usually in the output layer.

But other than that, using a linear activation function in the hidden layer except for some very special circumstances relating to compression that we're going to talk about using the linear activation function is extremely rare.

And, of course, if we're actually predicting housing prices, as you saw in the week one video, because housing prices are all non-negative, Perhaps even then you can use a value activation function so that your output y-hats are all greater than or equal to 0.

So I hope that gives you a sense of why having a non-linear activation function is a critical part of neural networks.

Next we're going to start to talk about gradient descent and to do that to set up for our discussion for gradient descent, in the next video I want to show you how to estimate-how to compute-the slope or the derivatives of individual activation functions.

So let's go on to the next video.