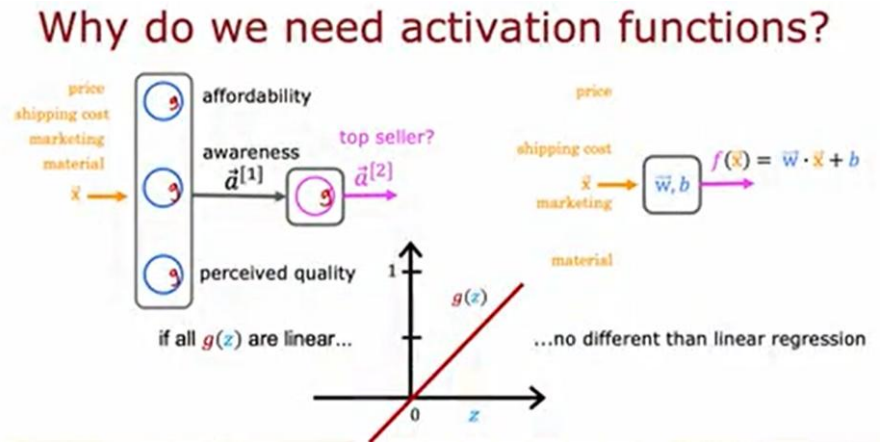
# Why do we need activation functions?

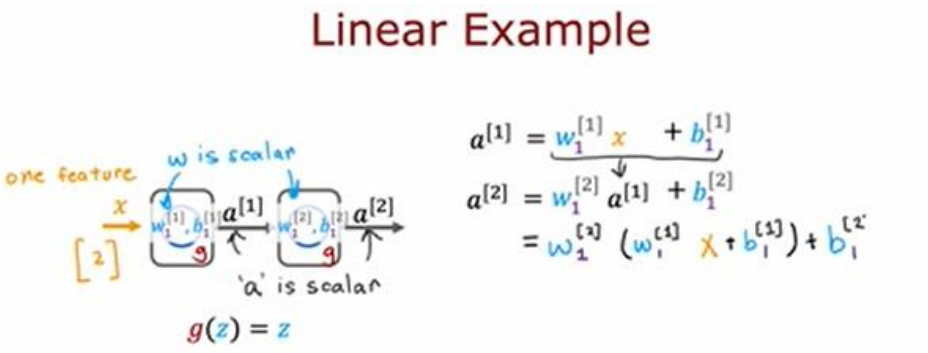
Let's take a look at why neural networks need activation functions and why they just don't work if we were to use the linear activation function in every neuron in the neural network. Recall this demand prediction example. What would happen if we were to use a linear activation function for all of the nodes in this neural network? It turns out that this big neural network will become no different than just linear regression.

So this would defeat the entire purpose of using a neural network because it would then just not be able to fit anything more complex than the linear regression model that we learned about in the first course. Let's illustrate this with a simpler example.

Let's look at the example of a neural network where the input x is just a number and we have one hidden unit with parameters w1 and b1 that outputs a1, which is here, just a number, and then the second layer is the output layer and it has also just one output unit with parameters w2 and b2 and then output a2, which is also just a number, just a scalar, which is the output of the neural network f of x.



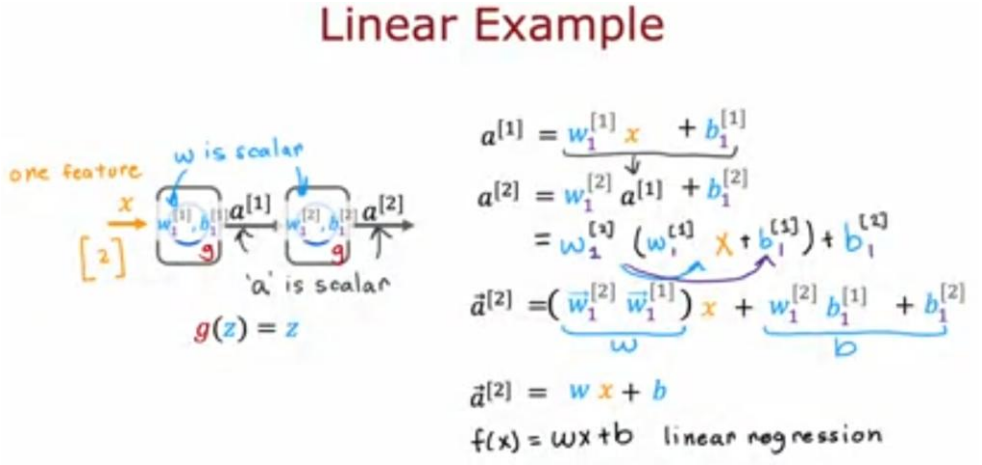

Let's see what this neural network would do if we were to use the linear activation function g of z equals z everywhere. So to compute a1 as a function of x, the neural network will use a1 equals g of w1 times x plus b1. But g of z is equal to z. So this is just w1 times x plus b1. Then a2 is equal to w2 times a1 plus b2, because g of z equals z. Let me take this expression for a1 and substitute it in there. So that becomes w2 times w1 x plus b1 plus b2.




If we simplify, this becomes w2, w1 times x plus w2, b1 plus b2. It turns out that if I were to set w equals w2 times w1 and set b equals this quantity over here, then what we've just shown is that a2 is equal to wx plus b. So a2 is just a linear function of the input x. Rather than using a neural

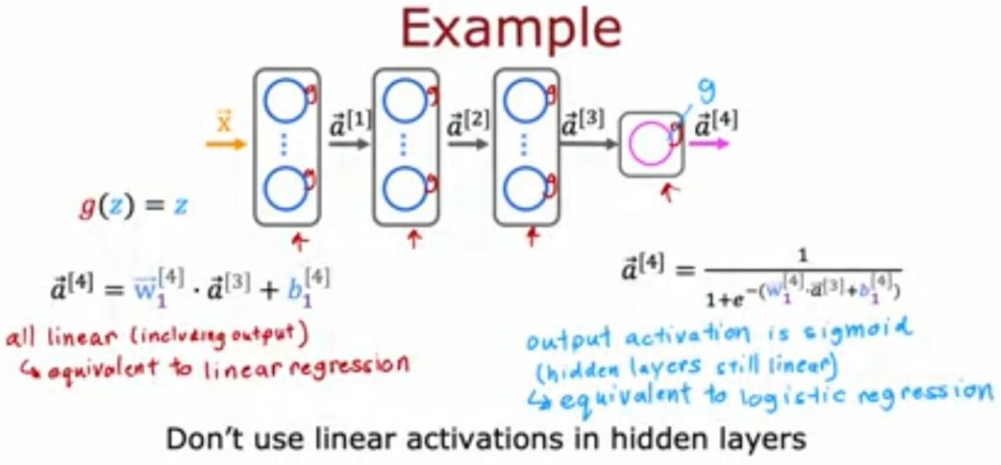
network with one hidden layer and one output layer, we might as well have just used a linear regression model. If you're familiar with linear algebra, this result comes from the fact that a linear function of a linear function is itself a linear function.

This is why having multiple layers in a neural network doesn't let the neural network compute any more complex features or learn anything more complex than just a linear function. So in the general case, if you had a neural network with multiple layers like this and say you were to use a linear activation function for all of the hidden layers and also use a linear activation function for the output layer, then it turns out this model will compute an output that is completely equivalent to linear regression.




The output a4 can be expressed as a linear function of the input features x plus b. Or alternatively, if we were to still use a linear activation function for all the hidden layers, for these three hidden layers here, but we were to use a logistic activation function for the output layer, then it turns out you can show that this model becomes equivalent to logistic regression, and a4, in this case, can be expressed as 1 over 1 plus e to the negative wx plus b for some values of w and b.

So this big neural network doesn't do anything that you can't also do with logistic regression. That's why a common rule of thumb is don't use the linear activation function in the hidden layers of the neural network. In fact, I typically recommend using the ReLU activation function should do just fine. So that's why a neural network needs activation functions other than just the linear activation function everywhere. So far, you've learned to build neural networks for binary classification problems where y is either zero or one.



As well as for regression problems where y can take negative or positive values, or maybe just positive and non-negative values. In the next video, I'd like to share with you a generalization of what you've seen so far for classification. In particular, when y doesn't just take on two values, but may take on three or four or ten or even more categorical values. Let's take a look at how you can build a neural network for that type of classification problem.