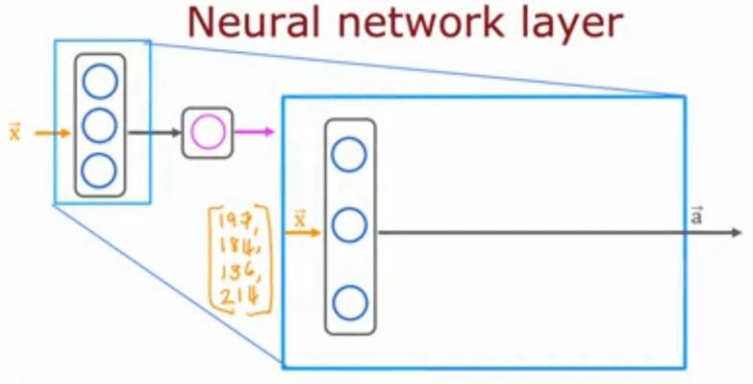
Neural network layer

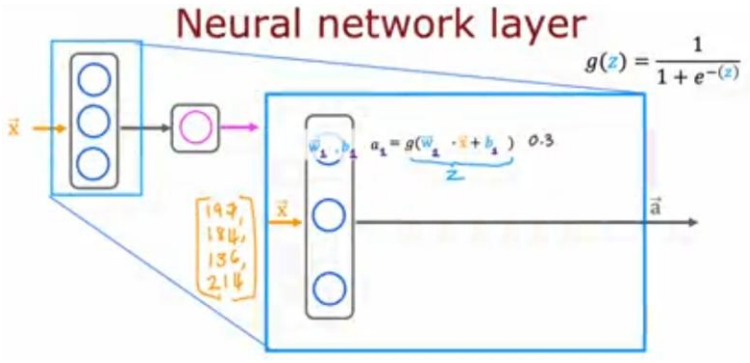
The fundamental building block of most modern neural networks is a layer of neurons. In this video, you'll learn how to construct a layer of neurons and once you have that down, you'd be able to take those building blocks and put them together to form a large neural network. Let's take a look at how a layer of neurons works. Here's the example we had from the demand prediction example where we had four input features that were set to this layer of three neurons in the hidden layer that then sends its output to this output layer with just one neuron.

Let's zoom in to the hidden layer to look at its computations. This hidden layer inputs four numbers and these four numbers are inputs to each of three neurons. Each of these three neurons is just implementing a little logistic regression unit or a little bit logistic regression function.



Take this first neuron. It has two parameters, w and b. In fact, to denote that, this is the first hidden unit, I'm going to subscript this as w_1, b_1. What it does is I'll output some activation value a, which is g of w_1 in a product with x plus b_1, where this is the familiar z value that you have learned about in logistic regression in the previous course, and g of z is the familiar logistic function, 1 over 1 plus e to the negative z.

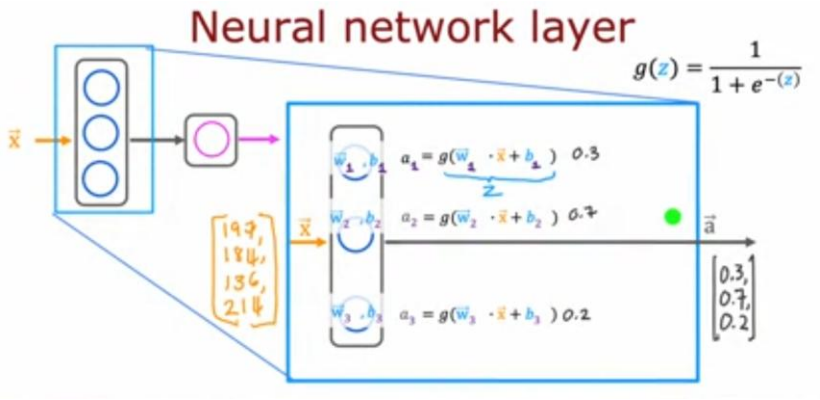
Maybe this ends up being a number 0.3 and that's the activation value a of the first neuron. To denote that this is the first neuron, I'm also going to add a subscript a_1 over here, and so a_1 may be a number like 0.3. There's a 0.3 chance of this being highly affordable based on the input features.



Now let's look at the second neuron. The second neuron has parameters w_2 and b_2, and these w, b or w_2, b_2 are the parameters of the second logistic unit. It computes a_2 equals the logistic function g applied to w_2 dot product x plus b_2 and this may be some other number, say 0.7.

Because in this example, there's a 0.7 chance that we think the potential buyers will be aware of this t-shirt. Similarly, the third neuron has a third set of parameters w_3, b_3. Similarly, it computes an

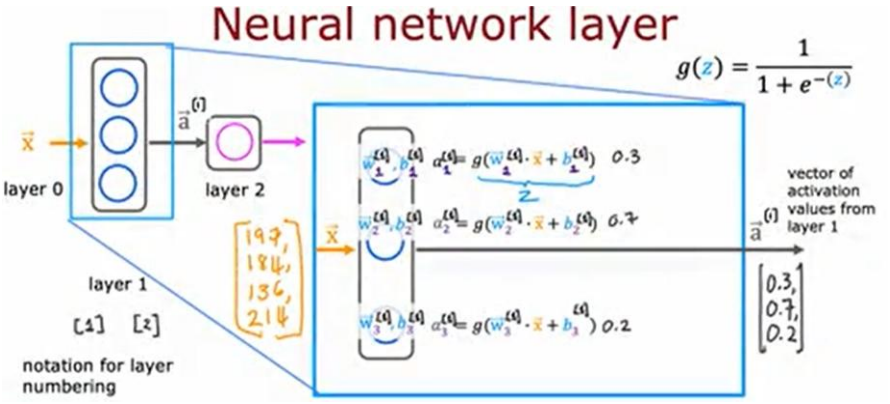
activation value a_3 equals g of w_3 dot product x plus b_3 and that may be say, 0.2. In this example, these three neurons output 0.3, 0.7, and 0.2, and this vector of three numbers becomes the vector of activation values a, that is then passed to the final output layer of this neural network.



Now, when you build neural networks with multiple layers, it'll be useful to give the layers different numbers. By convention, this layer is called layer 1 of the neural network and this layer is called layer 2 of the neural network. The input layer is also sometimes called layer 0 and today, there are neural networks that can have dozens or even hundreds of layers.

But in order to introduce notation to help us distinguish between the different layers, I'm going to use superscript square bracket 1 to index into different layers. In particular, a superscript in square brackets 1, I'm going to use, that's a notation to denote the output of layer 1 of this hidden layer of this neural network, and similarly, w_1, b_1 here are the parameters of the first unit in layer 1 of the neural network, so I'm also going to add a superscript in square brackets 1 here, and w_2, b_2 are the parameters of the second hidden unit or the second hidden neuron in layer 1.

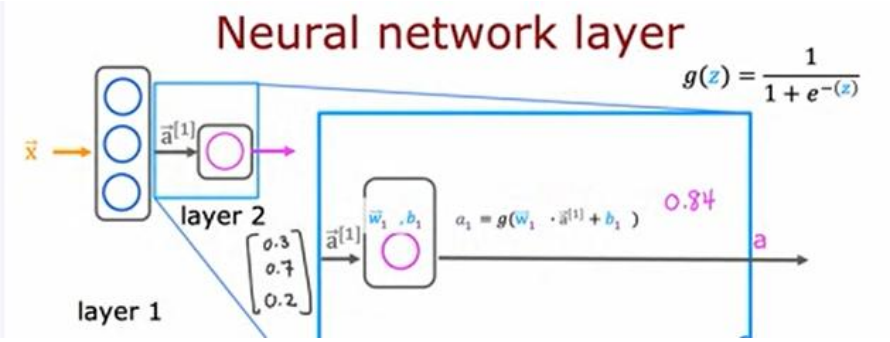
Its parameters are also denoted here w^[1] like so. Similarly, I can add superscripts square brackets like so to denote that these are the activation values of the hidden units of layer 1 of this neural network.



I know maybe this notation is getting a little bit cluttered. But the thing to remember is whenever you see this superscript square bracket 1, that just refers to a quantity that is associated with layer 1 of the neural network. If you see superscript square bracket 2, that refers to a quantity associated with layer 2 of the neural network and similarly for other layers as well, including layer 3, layer 4 and so on for neural networks with more layers.

That's the computation of layer 1 of this neural network. Its output is this activation vector, a^[1] and I'm going to copy this over here because this output a_1 becomes the input to layer 2. Now let's

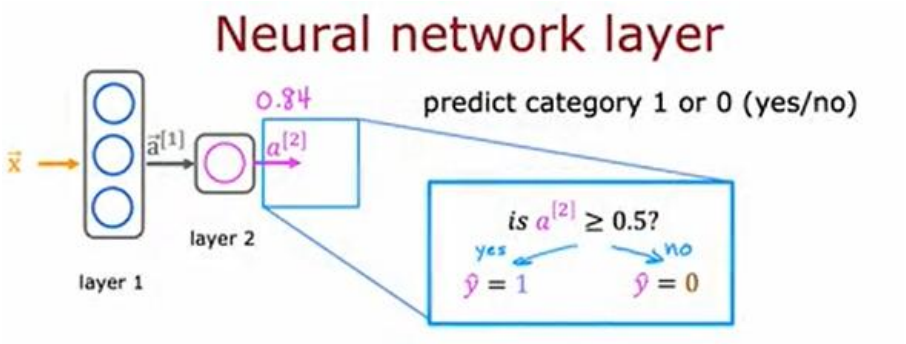
zoom into the computation of layer 2 of this neural network, which is also the output layer. The input to layer 2 is the output of layer 1, so a_1 is this vector 0.3, 0.7, 0.2 that we just computed on the previous part of this slide.



Because the output layer has just a single neuron, all it does is it computes a_1 that is the output of this first and only neuron, as g, the sigmoid function applied to w _1 in a product with a^[1], so this is the input into this layer, and then plus b_1. Here, this is the quantity z that you familiar with and g as before is the sigmoid function that you apply to this.

If this results in a number, say 0.84, then that becomes the output layer of the neural network. In this example, because the output layer has just a single neuron, this output is just a scalar, is a single number rather than a vector of numbers. Sticking with our notational convention from before, we're going to use a superscript in square brackets 2, to denote the quantities associated with layer 2 of this neural network, so a^[2] is the output of this layer, and so I'm going to also copy this here as the final output of the neural network.

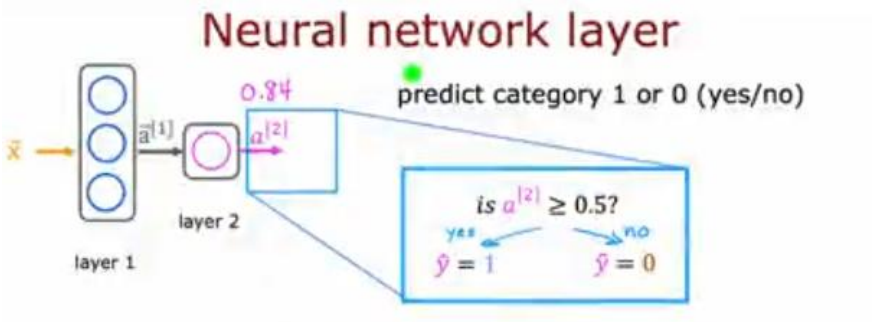
To make the notation consistent, you can also add these superscripts square bracket 2s to denote that these are the parameters and activation values associated with layer 2 of the neural network. Once the neural network has computed a_2, there's one final optional step that you can choose to implement or not, which is if you want a binary prediction, 1 or 0, is this a top seller?



Yes or no? As you can take the number a superscript square brackets 2 subscript 1, and this is the number 0.84 that we computed, and threshold this at 0.5. If it's greater than 0.5, you can predict y hat equals 1 and if it is less than 0.5, then predict your y hat equals 0.

We saw this thresholding as well when you learned about logistic regression in the first course of the specialization. If you wish, this then gives you the final prediction y hat as either one or zero, if you don't want just the probability of it being a top seller. So that's how a neural network works.

Every layer inputs a vector of numbers and applies a bunch of logistic regression units to it, and then computes another vector of numbers that then gets passed from layer to layer until you get to the final output layers computation, which is the prediction of the neural network.



Then you can either threshold at 0.5 or not to come up with the final prediction. With that, let's go on to use this foundation we've built now to look at some even more complex, even larger neural network models. I hope that by seeing more examples, this concept of layers and how to put them together to build a neural network will become even clearer.