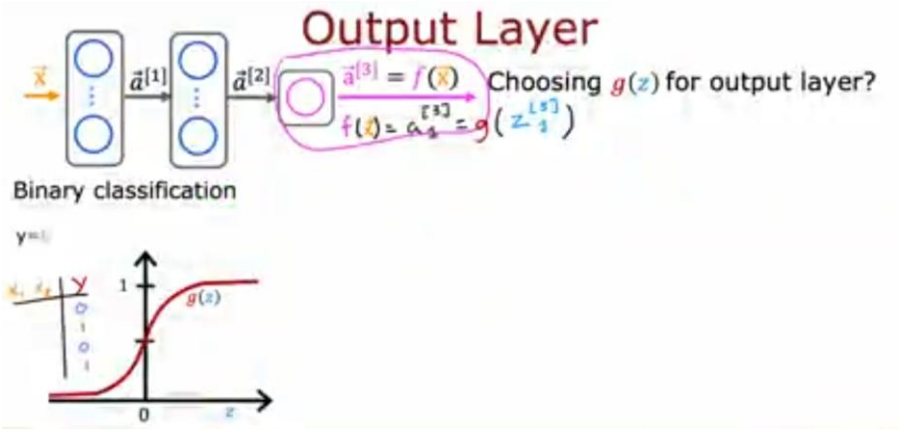
Selection of activation functions

Let's take a look at how you can choose the activation function for different neurons in your neural network. We'll start with some guidance for how to choose it for the output layer. It turns out that depending on what the target label or the ground truth label y is, there will be one fairly natural choice for the activation function for the output layer, and we'll then go and look at the choice of the activation function also for the hidden layers of your neural network.

Let's take a look. You can choose different activation functions for different neurons in your neural network, and when considering the activation function for the output layer, it turns out that there'll often be one fairly natural choice, depending on what is the target or the ground truth label y.

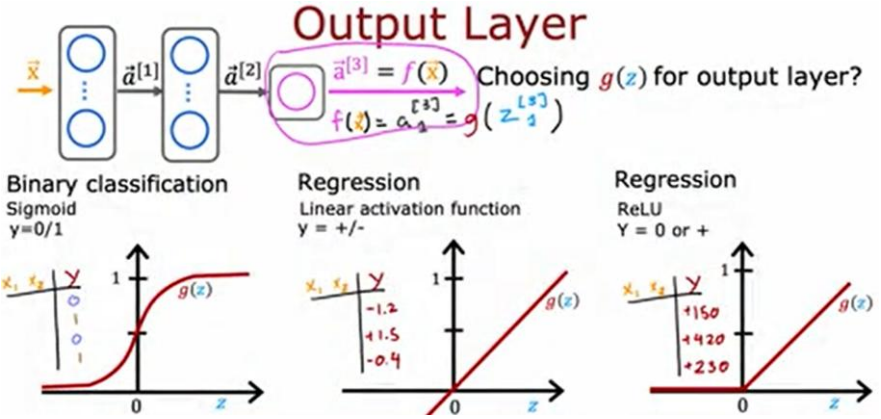
Specifically, if you are working on a classification problem where y is either zero or one, so a binary classification problem, then the sigmoid activation function will almost always be the most natural choice, because then the neural network learns to predict the probability that y is equal to one, just like we had for logistic regression. My recommendation is, if you're working on a binary classification problem, use sigmoid at the output layer.



Alternatively, if you're solving a regression problem, then you might choose a different activation function. For example, if you are trying to predict how tomorrow's stock price will change compared to today's stock price. Well, it can go up or down, and so in this case y would be a number that can be either positive or negative, and in that case I would recommend you use the linear activation function. Why is that?

Well, that's because then the outputs of your neural network, f of x, which is equal to a^3 in the example above, would be g applied to z^3 and with the linear activation function, g of z can take on either positive or negative values. So y can be positive or negative, use a linear activation function.

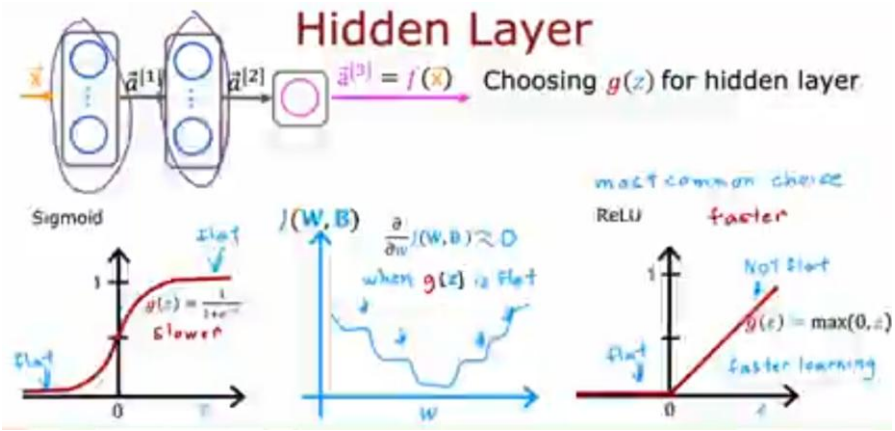
Finally, if y can only take on non-negative values, such as if you're predicting the price of a house, that can never be negative, then the most natural choice will be the ReLU activation function because as you see here, this activation function only takes on non-negative values, either zero or positive values.



In choosing the activation function to use for your output layer, usually depending on what is the label y you're trying to predict, there'll be one fairly natural choice. In fact, the guidance on this slide is how I pretty much always choose my activation function as well for the output layer of a neural network. How about the hidden layers of a neural network?

It turns out that the ReLU activation function is by far the most common choice in how neural networks are trained by many practitioners today. Even though we had initially described neural networks using the sigmoid activation function, and in fact, in the early history of the development of neural networks, people use sigmoid activation functions in many places, the field has evolved to use ReLU much more often and sigmoids hardly ever.

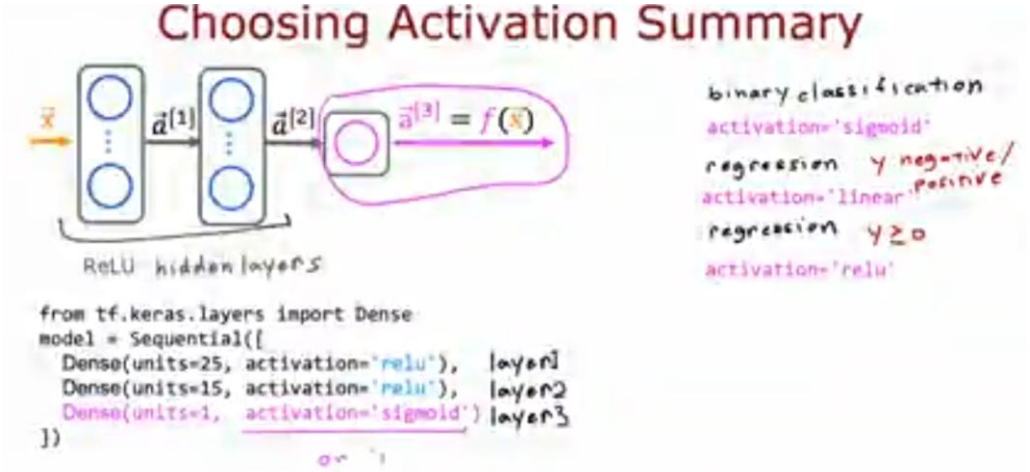
Well, the one exception that you do use a sigmoid activation function in the output layer if you have a binary classification problem. So why is that? Well, there are a few reasons.



First, if you compare the ReLU and the sigmoid activation functions, the ReLU is a bit faster to compute because it just requires computing max of 0, z, whereas the sigmoid requires taking an exponentiation and then a inverse and so on, and so it's a little bit less efficient. But the second reason which turns out to be even more important is that the ReLU function goes flat only in one part of the graph; here on the left is completely flat, whereas the sigmoid activation function, it goes flat in two places.

It goes flat to the left of the graph and it goes flat to the right of the graph. If you're using gradient descent to train a neural network, then when you have a function that is fat in a lot of places, gradient descents would be really slow. I know that gradient descent optimizes the cost function J of W, B rather than optimizes the activation function, but the activation function is a piece of what goes into computing, and that results in more places in the cost function J of W, B that are flats as well and with a small gradient and it slows down learning.

I know that that was just an intuitive explanation, but researchers have found that using the ReLU activation function can cause your neural network to learn a bit faster as well, which is why for most practitioners if you're trying to decide what activation functions to use with hidden layer, the ReLU activation function has become now by far the most common choice. In fact that I'm building a neural network, this is how I choose activation functions for the hidden layers as well.



To summarize, here's what I recommend in terms of how you choose the activation functions for your neural network. For the output layer, use a sigmoid, if you have a binary classification problem; linear, if y is a number that can take on positive or negative values, or use ReLU if y can take on only positive values or zero positive values or non-negative values.

Then for the hidden layers I would recommend just using ReLU as a default activation function, and in TensorFlow, this is how you would implement it. Rather than saying activation equals sigmoid as we had previously, for the hidden layers, that's the first hidden layer, the second hidden layer as TensorFlow to use the ReLU activation function, and then for the output layer in this example, I've asked it to use the sigmoid activation function, but if you wanted to use the linear activation function, is that, that's the syntax for it, or if you wanted to use the ReLU activation function that shows the syntax for it.

With this richer set of activation functions, you'll be well-positioned to build much more powerful neural networks than just once using only the sigmoid activation function. By the way, if you look at the research literature, you sometimes hear of authors using even other activation functions, such as the tan h activation function or the LeakyReLU activation function or the swish activation function. Every few years, researchers sometimes come up with another interesting activation function, and sometimes they do work a little bit better.

For example, I've used the LeakyReLU activation function a few times in my work, and sometimes it works a little bit better than the ReLU activation function you've learned about in this video. But I think for the most part, and for the vast majority of applications what you learned about in this video would be good enough. Of course, if you want to learn more about other activation functions, feel free to look on the Internet, and there are just a small handful of cases where these other activation functions could be even more powerful as well. With that, I hope you also enjoy practicing these ideas, these activation functions in the optional labs and in the practice labs. But this raises yet another question. Why do we even need activation functions at all? Why don't we just use the linear activation function or use no activation function anywhere? It turns out this does not work at all.