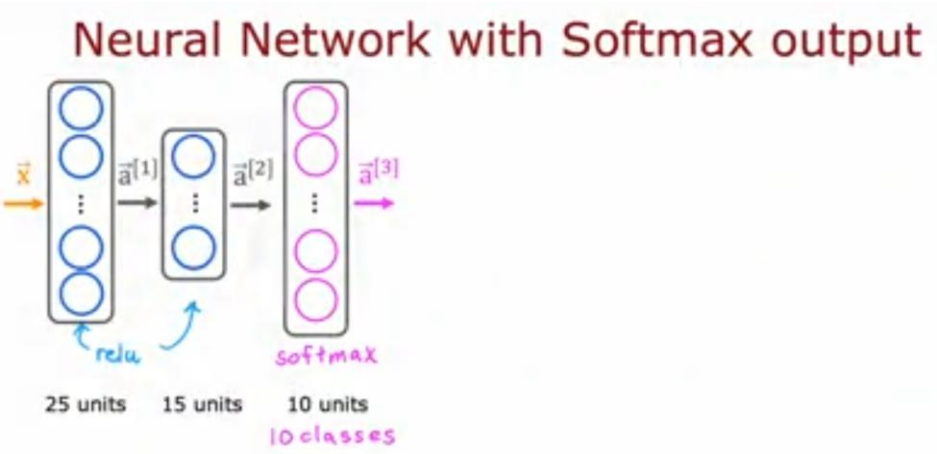
Neural network with Softmax output

In order to build a Neural Network that can carry out multi class classification, we're going to take the Softmax regression model and put it into essentially the output layer of a Neural Network. Let's take a look at how to do that. Previously, when we were doing handwritten digit recognition with just two classes. We use a new Neural Network with this architecture.

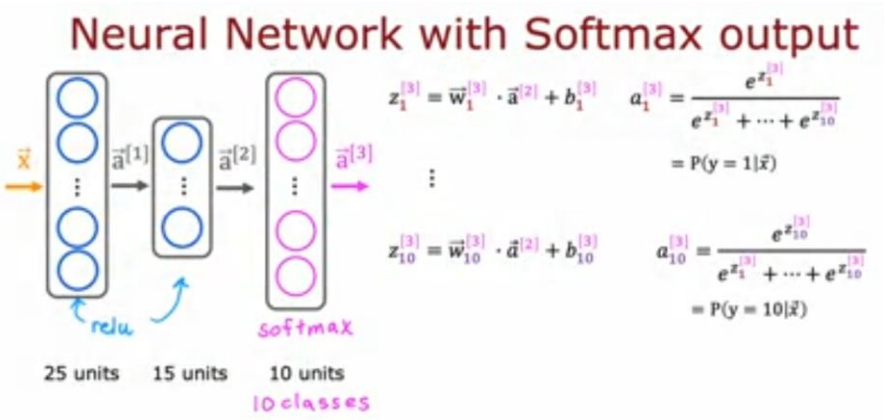
If you now want to do handwritten digit classification with 10 classes, all the digits from zero to nine, then we're going to change this Neural Network to have 10 output units like so. And this new output layer will be a Softmax output layer. So sometimes we'll say this Neural Network has a Softmax output or that this upper layer is a Softmax layer.



And the way forward propagation works in this Neural Network is given an input X A1 gets computed exactly the same as before. And then A2, the activations for the second hidden layer also get computed exactly the same as before. And we now have to compute the activations for this output layer, that is a3. This is how it works.

If you have 10 output classes, we will compute Z1, Z2 through Z10 using these expressions. So this is actually very similar to what we had previously for the formula you're using to compute Z. Z1 is W1.product with a2, the activations from the previous layer plus b1 and so on for Z1 through Z10.

Then a1 is equal to e to the Z1 divided by e to the Z1 plus up to e to the Z10. And that's our estimate of the chance that y is equal to 1. And similarly for a2 and similarly all the way up to a10. And so this gives you your estimates of the chance of y being good to one, two and so on up through the 10th possible label for y.

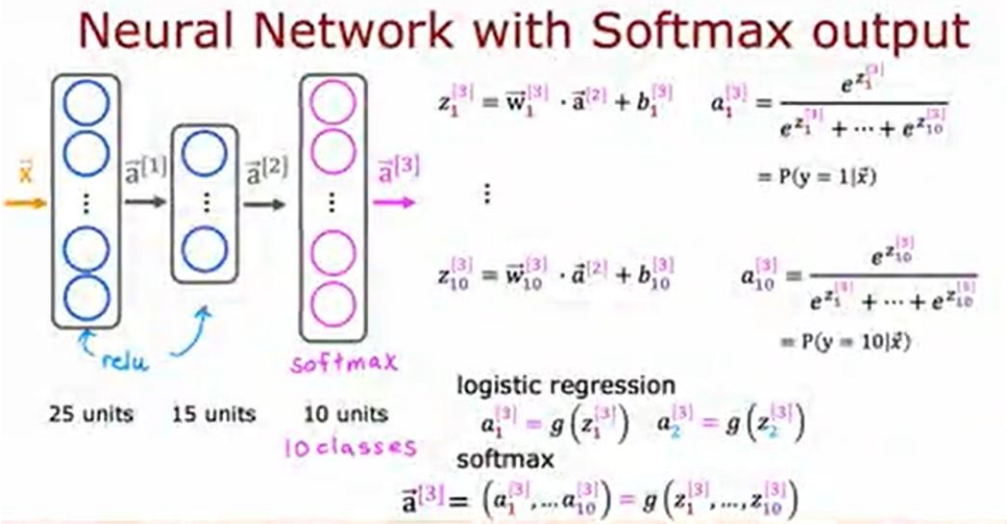


And just for completeness, if you want to indicate that these are the quantities associated with layer three, technically, I should add these super strip three's there. It does make the notation a little bit more cluttered. But this makes explicit that this is, for example, the Z(3), 1 value and this is the parameters associated with the first unit of layer three of this Neural Network.

And with this, your Softmax open layer now gives you estimates of the chance of there being any of these 10 possible output labels. I do want to mention that the Softmax layer will sometimes also called the Softmax activation function, it is a little bit unusual in one respect compared to the other activation functions we've seen so far, like sigma, radial and linear, which is that when we're looking at sigmoid or value or linear activation functions, a1 was a function of Z1 and a2 was a function of Z2 and only Z2.

In other words, to obtain the activation values, we could apply the activation function g be it sigmoid or rarely or something else element wise to Z1 and Z2 and so on to get a1 and a2 and a3 and a4. But with the Softmax activation function, notice that a1 is a function of Z1 and Z2 and Z3 all the way up to Z10. So each of these activation values, it depends on all of the values of Z.

And this is a property that's a bit unique to the Softmax output or the Softmax activation function or stated differently if you want to compute a1 through a10,hat is a function of Z1 all the way up to Z 10 simultaneously. And this is unlike the other activation functions we've seen so far.



Finally, let's look at how you would implement this in tensorflow. If you want to implement the neural network that I've shown here on this slide, this is the code to do so. Similar as before there are three steps to specifying and training the model. The first step is to tell tensorflow to sequentially string together three layers.

First layer, is this 25 units with rail you activation function. Second layer, 15 units of rally activation function. And then the third layer, because there are now 10 output units, you want to output a1 through a10, so they're now 10 output units.

And we'll tell tensorflow to use the Softmax activation function. And the cost function that you saw in the last video, tensorflow calls that the SparseCategoricalCrossentropy function. So I know this name is a bit of a mouthful, whereas for logistic regression we had the BinaryCrossentropy function, here we're using the SparseCategoricalCrossentropy function. And what sparse categorical refers to is that you're still classified there into categories. So it's categorical. This takes on values from 1 to 10. And sparse refers to that y can only take on one of these 10 values.

So each image is either 0 or 1 or 2 or so on up to 9. You're not going to see a picture that is simultaneously the number two and the number seven so sparse refers to that each digit is only one of these categories. But so that's why the loss function that you saw in the last video, it's called intensive though the sparse categorical cross entropy loss function. And then the code for training the model is just the same as before.

And if you use this code, you can train a neural network on a multi class classification problem. Just one important note, if you use this code exactly as I've written here, it will work, but don't actually use this code because it turns out that in tensorflow, there's a better version of the code that makes tensorflow work better. So even though the code shown in this slide works.

Don't use this code the way I've written it here, because in a later video this week you see a different version that's actually the recommended version of implementing this, that will work better. But we'll take a look at that in a later video. So now, you know how to train a neural network with a softmax output layer with one caveal.

There's a different version of the code that will make tensorflow able to compute these probabilities much more accurately. Let's take a look at that in the next video, we should also show you the actual code that I recommend you use if you're training a Softmax neural networ