Computing a Neural Network's Output

(SPEECH)
In

(DESCRIPTION)
Text, One hidden layer Neural Network. Computing a Neural Network's Output. Website, deep learning, dot, A.I.

(SPEECH)
the last video, you saw what a single hidden layer neural network looks like.

In this video, let's go through the details of exactly how this neural network computes these outputs.

What you see is that is like logistic regression, the repeater a lot of times.

Let's take a

(DESCRIPTION)
New slide, Neural Network Representation.

(SPEECH)
look. So, this is what a two-layer neural network looks.

Let's go more deeply into exactly what this neural network computes.

Now, we've said before that logistic regression, the circle in logistic regression, really represents two steps of computation rows.

You compute z as follows, and a second, you compute the activation as a sigmoid function of z.

So, a neural network just does this a lot more times.

Let's start by focusing on just one of the nodes in the hidden layer.

Let's look at the first node in the hidden layer.

So, I've grayed out the other nodes for now.

So, similar to logistic regression on the left, this nodes in the hidden layer does two steps of computation.

The first step and think of as the left half of this node, it computes z equals w transpose x plus b, and the notation we'll use is, these are all quantities associated with the first hidden layer.

So, that's why we have a bunch of square brackets there.

This is the first node in the hidden layer.

So, that's why we have the subscript one over there.

So first, it does that, and then the second step, is it computes a_[1]_1 equals sigmoid of z_[1]_1, like so.

So, for both z and a, the notational convention is that a, l, i, the l here in superscript square brackets, refers to the layer number, and the i subscript here, refers to the nodes in that layer.

So, the node we'll be looking at is layer one, that is a hidden layer node one.

So, that's why the superscripts and subscripts were both one, one.

So, that little circle, that first node in the neural network, represents carrying out these two steps of computation.

Now, let's look at the second node in the neural network, or the second node in the hidden layer of the neural network.

Similar to the logistic regression unit on the left, this little circle represents two steps of computation.

The first step is it computes z.

This is still layer one, but now as a second node equals w transpose x, plus b_[1]_2, and then a_[1] two equals sigmoid of z_[1]_2.

Again, feel free to pause the video if you want, but you can double-check that the superscript and subscript notation is consistent with what we have written here above in purple.

So, we've talked through the first two hidden units in a neural network, having units three and four also represents some computations.

So now, let me take this pair of equations, and this pair of equations, and let's copy them to the next slide.

So, here's our neural network, and here's the first, and here's the second equations that we've worked out previously for the first and the second hidden units.

If you then go through and write out the corresponding equations for the third and fourth hidden units, you get the following.

So, let me show this notation is clear, this is the vector w_[1]_1, this is a vector transpose times x.

So, that's what the superscript T there represents.

It's a vector transpose.

Now, as you might have guessed, if you're actually implementing a neural network, doing this with a for loop, seems really inefficient.

So, what we're going to do, is take these four equations and vectorize.

So, we're going to start by showing how to compute z as a vector, it turns out you could do it as follows.

Let me take these w's and stack them into a matrix, then you have w_[1]_1 transpose, so that's a row vector, or this column vector transpose gives you a row vector, then w_[1]_2, transpose, w_[1]_3 transpose, w_[1]_4 transpose.

So, by stacking those four w vectors together, you end up with a matrix.

So, another way to think of this is that we have four logistic regression units there, and each of the logistic regression units, has a corresponding parameter vector, w. By stacking those four vectors together, you end up with this four by three matrix.

So, if you then take this matrix and multiply it by your input features x1, x2, x3, you end up with by how matrix multiplication works.

You end up with w_[1]_1 transpose x, w_2_[1] transpose x, w_3_[1] transpose x, w_4_[1] transpose x.

Then, let's not figure the b's.

So, we now add to this a vector b_[1]_1 one, b_[1]_2, b_[1]_3, b_[1]_4.

So, that's basically this, then this is b_[1]_1, b_[1]_2, b_[1]_3, b_[1]_4.

So, you see that each of the four rows of this outcome correspond exactly to each of these four rows, each of these four quantities that we had above.

So, in other words, we've just shown that this thing is therefore equal to z_[1]_1, z_[1]_2, z_[1]_3, z_[1]_4, as defined here.

Maybe not surprisingly, we're going to call this whole thing, the vector z_[1], which is taken by stacking up these individuals of z's into a column vector.

When we're vectorizing, one of the rules of thumb that might help you navigate this, is that while we have different nodes in the layer, we'll stack them vertically.

So, that's why we have z_[1]_1 through z_[1]_4, those corresponded to four different nodes in the hidden layer, and so we stacked these four numbers vertically to form the vector z[1].

To use one more piece of notation, this four by three matrix here which we obtained by stacking the lowercase w_[1]_1, w_[1]_2, and so on, we're going to call this matrix W capital [1].

Similarly, this vector, we're going to call b superscript [1] square bracket.

So, this is a four by one vector.

So now, we've computed z using this vector matrix notation, the last thing we need to do is also compute these values of a.

So, prior won't surprise you to see that we're going to define a_[1], as just stacking together, those activation values, a [1], 1 through a [1], 4.

So, just take these four values and stack them together in a vector called a[1].

This is going to be a sigmoid of z[1], where this now has been implementation of the sigmoid function that takes in the four elements of z, and applies the sigmoid function element-wise to it.

So, just a recap, we figured out that z_[1] is equal to w_[1] times the vector x plus the vector b_[1], and a_[1] is sigmoid times z_[1].

Let's just copy this to the next slide.

(DESCRIPTION)
New slide, Neural Network Representation learning.

(SPEECH)
What we see is that for the first layer of the neural network given an input x, we have that z_[1] is equal to w_[1] times x plus b_[1], and a_[1] is sigmoid of z_[1].

The dimensions of this are four by one equals, this was a four by three matrix times a three by one vector plus a four by one vector b, and this is four by one same dimension as end.

Remember, that we said x is equal to a_[0].

Just say y hat is also equal to a two.

If you want, you can actually take this x and replace it with a_[0], since a_[0] is if you want as an alias for the vector of input features, x.

Now, through a similar derivation, you can figure out that the representation for the next layer can also be written similarly where what the output layer does is, it has associated with it, so the parameters w_[2] and b_[2].

So, w_[2] in this case is going to be a one by four matrix, and b_[2] is just a real number as one by on.

So, z_[2] is going to be a real number we'll write as a one by one matrix.

Is going to be a one by four thing times a was four by one, plus b_[2] as one by one, so this gives you just a real number.

If you think of this last upper unit as just being analogous to logistic regression which have parameters w and b, w really plays an analogous role to w_[2] transpose, or w_[2] is really W transpose and b is equal to b_[2].

I said we want to cover up the left of this network and ignore all that for now, then this last upper unit is a lot like logistic regression, except that instead of writing the parameters as w and b, we're writing them as w_[2] and b_[2], with dimensions one by four and one by one.

So, just a recap.

For logistic regression, to implement the output or to implement prediction, you compute z equals w transpose x plus b, and a or y hat equals a, equals sigmoid of z.

When you have a neural network with one hidden layer, what you need to implement, is to computer this output is just these four equations.

You can think of this as a vectorized implementation of computing the output of first these for logistic regression units in the hidden layer, that's what this does, and then this logistic regression in the output layer which is what this does.

I hope this description made sense, but the takeaway is to compute the output of this neural network, all you need is those four lines of code.

So now, you've seen how given a single input feature, vector a, you can with four lines of code, compute the output of this neural network.

Similar to what we did for logistic regression, we'll also want to vectorize across multiple training examples.

We'll see that by stacking up training examples in different columns in the matrix, with just slight modification to this, you also, similar to what you saw in this regression, be able to compute the output of this neural network, not just a one example at a time, prolong your, say your entire training set at a time.

So, let's see the details of that in the next video.