Vectorizing across multiple examples

(DESCRIPTION)
One hidden layer Neural Network. Vectorizing across multiple examples. Website, deep learning, dot, A.I.

(SPEECH)
In the last video, you saw how to compute the prediction on a neural network, given a single training example.

In this video, you see how to vectorize across multiple training examples.

And the outcome will be quite similar to what you saw for logistic regression.

Whereby stacking up different training examples in different columns of the matrix, you'd be able to take the equations you had from the previous video.

And with very little modification, change them to make the neural network compute the outputs on all the examples on pretty much all at the same time.

So let's see the details on how to do

(DESCRIPTION)
New slide, Vectorizing across multiple examples.

(SPEECH)
that.

These were the four equations we have from the previous video of how you compute z1, a1, z2 and a2.

And they tell you how, given an input feature back to x, you can use them to generate a2 =y hat for a single training example.

Now if you have m training examples, you need to repeat this process for say, the first training example.

x superscript (1) to compute y hat 1 does a prediction on your first training example.

Then x(2) use that to generate prediction y hat (2).

And so on down to x(m) to generate a prediction y hat (m).

And so in all these activation function notation as well, I'm going to write this as a[2](1).

And this is a[2](2), and a(2)(m), so this notation a[2](i).

The round bracket i refers to training example i, and the square bracket 2 refers to layer 2, okay.

So that's how the square bracket and the round bracket indices work.

And so to suggest that if you have an unvectorized implementation and want to compute the predictions of all your training examples, you need to do for i = 1 to m.

Then basically implement these four equations, right?

You need to make a z[1](i) = W(1) x(i) + b[1], a[1](i) = sigma of z[1](1).

z[2](i) = w[2]a[1](i) + b[2] andZ2i equals w2a1i plus b2 and a[2](i) = sigma point of z[2](i).

So it's basically these four equations on top by adding the superscript round bracket i to all the variables that depend on the training example.

So adding this superscript round bracket i to x is z and a, if you want to compute all the outputs on your m training examples examples.

What we like to do is vectorize this whole computation, so as to get rid of this for.

And by the way, in case it seems like I'm getting a lot of nitty gritty linear algebra, it turns out that being able to implement this correctly is important in the deep learning era.

And we actually chose notation very carefully for this course and make this vectorization steps as easy as possible.

So I hope that going through this nitty gritty will actually help you to more quickly get correct implementations of these algorithms working.

All right so let me just copy this whole block of code to the next slide and then we'll see how to vectorize this.

So here's what we have from the previous slide with the for loop going over our m training examples.

So recall that we defined the matrix x to be equal to our training examples stacked up in these columns like so.

So take the training examples and stack them in columns.

So this becomes a n, or maybe nx by m diminish the matrix.

I'm just going to give away the punch line and tell you what you need to implement in order to have a vectorized implementation of this for loop.

It turns out what you need to do is compute Z[1] = W[1] X + b[1], A[1]= sig point of z[1].

Then Z[2] = w[2] A[1] + b[2] and then A[2] = sig point of Z[2].

So if you want the analogy is that we went from lower case vector xs to just capital case X matrix by stacking up the lower case xs in different columns.

If you do the same thing for the zs, so for example, if you take z[1](i), z[1](2), and so on, and these are all column vectors, up to z[1](m), right.

So that's this first quantity that all m of them, and stack them in columns.

Then just gives you the matrix z[1].

And similarly you look at say this quantity and take a[1](1), a[1](2) and so on and a[1](m), and stacked them up in columns.

Then this, just as we went from lower case x to capital case X, and lower case z to capital case Z.

This goes from the lower case a, which are vectors to this capital A[1], that's over there and similarly, for z[2] and a[2].

Right they're also obtained by taking these vectors and stacking them horizontally.

And taking these vectors and stacking them horizontally, in order to get Z[2], and E[2].

One of the property of this notation that might help you to think about it is that this matrixes say Z and A, horizontally we're going to index across training examples.

So that's why the horizontal index corresponds to different training example, when you sweep from left to right you're scanning through the training cells.

And vertically this vertical index corresponds to different nodes in the neural network.

So for example, this node, this value at the top most, top left most corner of the mean corresponds to the activation of the first heading unit on the first training example.

One value down corresponds to the activation in the second hidden unit on the first training example, then the third heading unit on the first training sample and so on.

So as you scan down this is your indexing to the hidden units number.

Whereas if you move horizontally, then you're going from the first hidden unit.

And the first training example to now the first hidden unit and the second training sample, the third training example.

And so on until this node here corresponds to the activation of the first hidden unit on the final train example and the nth training example.

Okay so the horizontally the matrix A goes over different training examples.

And vertically the different indices in the matrix A corresponds to different hidden units.

And a similar intuition holds true for the matrix Z as well as for X where horizontally corresponds to different training examples.

And vertically it corresponds to different input features which are really different than those of the input layer of the neural network.

So of these equations, you now know how to implement in your network with vectorization, that is vectorization across multiple examples.

In the next video I want to show you a bit more justification about why this is a correct implementation of this type of vectorization.

It turns out the justification would be similar to what you had seen [INAUDIBLE].

Let's go on to the next video.