Softmax Regression

(DESCRIPTION)
Text, Multi-class classification. Softmax regression. Website, deep learning, dot, A.I.

(SPEECH)
So far, the classification examples we've talked about have used binary classification, where you had two possible labels, 0 or 1.

Is it a cat, is it not a cat?

What if we have multiple possible classes?

There's a generalization of logistic regression called Softmax regression.

The less you make predictions where you're trying to recognize one of C or one of multiple classes, rather than just recognize two classes.

Let's take a look.

(DESCRIPTION)
New slide, Recognizing cats, dogs, and baby chicks. Eight photos of cute animals are presented. In order, a chick, kitten, puppy, koala, chick, puppy, koala, and kitten.

(SPEECH)
Let's say that instead of just recognizing cats you want to recognize cats, dogs, and baby chicks.

So I'm going to call cats class 1, dogs class 2, baby chicks class 3.

And if none of the above, then there's an other or a none of the above class, which I'm going to call class 0.

So here's an example of the images and the classes they belong to.

That's a picture of a baby chick, so the class is 3.

Cats is class 1, dog is class 2, I guess that's a koala, so that's none of the above, so that is class 0, class 3 and so on.

So the notation we're going to use is, I'm going to use capital C to denote the number of classes you're trying to categorize your inputs into.

And in this case, you have four possible classes, including the other or the none of the above class.

So when you have four classes, the numbers indexing your classes would be 0 through capital C minus one.

So in other words, that would be zero, one, two or three.

In this case, we're going to build a new XY, where the upper layer has four, or in this case the variable capital alphabet C upward units.

So N, the number of units upper layer which is layer L is going to equal to 4 or in general this is going to equal to C.

And what we want is for the number of units in the upper layer to tell us what is the probability of each of these four classes.

So the first node here is supposed to output, or we want it to output the probability that is the other class, given the input x, this will output probability there's a cat.

Give an x, this will output probability as a dog.

Give an x, that will output the probability.

I'm just going to abbreviate baby chick to baby C, given the input x.

So here, the output labels y hat is going to be a four by one dimensional vector, because it now has to output four numbers, giving you these four probabilities.

And because probabilities should sum to one, the four numbers in the output y hat, they should sum to one.

(DESCRIPTION)
New slide, Softmax layer.

(SPEECH)
The standard model for getting your network to do this uses what's called a Softmax layer, and the output layer in order to generate these outputs.

Then write down the map, then you can come back and get some intuition about what the Softmax there is doing.

So in the final layer of the neural network, you are going to compute as usual the linear part of the layers.

So z, capital L, that's the z variable for the final layer.

So remember this is layer capital L.

So as usual you compute that as wL times the activation of the previous layer plus the biases for that final layer.

Now having computer z, you now need to apply what's called the Softmax activation function.

So that activation function is a bit unusual for the Softmax layer, but this is what it does.

First, we're going to computes a temporary variable, which we're going to call t, which is e to the z L.

So this is a part element-wise.

So zL here, in our example, zL is going to be four by one.

This is a four dimensional vector.

So t Itself e to the zL, that's an element wise exponentiation.

T will also be a 4.1 dimensional vector.

Then the output aL, is going to be basically the vector t will normalized to sum to 1.

So aL is going to be e to the zL divided by sum from J equal 1 through 4, because we have four classes of t substitute i.

So in other words we're saying that aL is also a four by one vector, and the i element of this four dimensional vector.

Let's write that, aL substitute i that's going to be equal to ti over sum of ti, okay?

In case this math isn't clear, we'll do an example in a minute that will make this clearer.

So in case this math isn't clear, let's go through a specific example that will make this clearer.

Let's say that your computer zL, and zL is a four dimensional vector, let's say is 5, 2, -1, 3.

What we're going to do is use this element-wise exponentiation to compute this vector t.

So t is going to be e to the 5, e to the 2, e to the -1, e to the 3.

And if you plug that in the calculator, these are the values you get.

E to the 5 is 1484, e squared is about 7.4, e to the -1 is 0.4, and e cubed is 20.1.

And so, the way we go from the vector t to the vector aL is just to normalize these entries to sum to one.

So if you sum up the elements of t, if you just add up those 4 numbers you get 176.3.

So finally, aL is just going to be this vector t, as a vector, divided by 176.3.

So for example, this first node here, this will output e to the 5 divided by 176.3.

And that turns out to be 0.842.

So saying that, for this image, if this is the value of z you get, the chance of it being called zero is 84.2%.

And then the next nodes outputs e squared over 176.3, that turns out to be 0.042, so this is 4.2% chance.

The next one is e to -1 over that, which is 0.042.

And the final one is e cubed over that, which is 0.114.

So it is 11.4% chance that this is class number three, which is the baby C class, right?

So there's a chance of it being class zero, class one, class two, class three.

So the output of the neural network aL, this is also y hat.

This is a 4 by 1 vector where the elements of this 4 by 1 vector are going to be these four numbers.

Then we just compute it.

So this algorithm takes the vector zL and is four probabilities that sum to 1.

And if we summarize what we just did to math from zL to aL, this whole computation confusing exponentiation to get this temporary variable t and then normalizing, we can summarize this into a Softmax activation function and say aL equals the activation function g applied to the vector zL.

The unusual thing about this particular activation function is that, this activation function g, it takes a input a 4 by 1 vector and it outputs a 4 by 1 vector.

So previously, our activation functions used to take in a single row value input.

So for example, the sigmoid and the value activation functions input the real number and output a real number.

The unusual thing about the Softmax activation function is, because it needs to normalized across the different possible outputs, and needs to take a vector and puts in outputs of vector.

So

(DESCRIPTION)
New slide, Softmax examples.

(SPEECH)
one of the things that a Softmax cross layer can represent, I'm going to show you some examples where you have inputs x1, x2.

And these feed directly to a Softmax layer that has three or four, or more output nodes that then output y hat.

So I'm going to show you a new network with no hidden layer, and all it does is compute z1 equals w1 times the input x plus b.

And then the output a1, or y hat is just the Softmax activation function applied to z1.

So in this neural network with no hidden layers, it should give you a sense of the types of things a Softmax function can represent.

So here's one example with just raw inputs x1 and x2.

A Softmax layer with C equals 3 upper classes can represent this type of decision boundaries.

Notice this kind of several linear decision boundaries, but this allows it to separate out the data into three classes.

And in this diagram, what we did was we actually took the training set that's kind of shown in this figure and train the Softmax cross fire with the upper labels on the data.

And then the color on this plot shows fresh holding the upward of the Softmax cross fire, and coloring in the input base on which one of the three outputs have the highest probability.

So we can maybe we kind of see that this is like a generalization of logistic regression with sort of linear decision boundaries, but with more than two classes [INAUDIBLE] class 0, 1, the class could be 0, 1, or 2.

Here's another example of the decision boundary that a Softmax cross fire represents when three normal datasets with three classes.

And here's another one, rIght, so this is a, but one intuition is that the decision boundary between any two classes will be more linear.

That's why you see for example that decision boundary between the yellow and the various classes, that's the linear boundary where the purple and red linear in boundary between the purple and yellow and other linear decision boundary.

But able to use these different linear functions in order to separate the space into three classes.

Let's look at some examples with more classes.

So it's an example with C equals 4, so that the green class and Softmax can continue to represent these types of linear decision boundaries between multiple classes.

So here's one more example with C equals 5 classes, and here's one last example with C equals 6.

So this shows the type of things the Softmax crossfire can do when there is no hidden layer of class, even much deeper neural network with x and then some hidden units, and then more hidden units, and so on.

Then you can learn even more complex non-linear decision boundaries to separate out multiple different classes.

So I hope this gives you a sense of what a Softmax layer or the Softmax activation function in the neural network can do.

In the next video, let's take a look at how you can train a neural network that uses a Softmax layer.