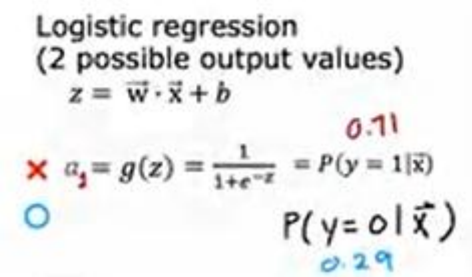
Softmax

The softmax regression algorithm is a generalization of logistic regression, which is a binary classification algorithm to the multiclass classification contexts. Let's take a look at how works. Recall that logistic regression applies when y can take on two possible output values, either zero or one, and the way it computes this output is, you would first calculate z equals w.product of x plus b, and then you would compute what I'm going to call here a equals g of z which is a sigmoid function applied to z.

We interpreted this as logistic regressions estimates of the probability of y being equal to 1 given those input features x. Now, quick quiz question; if the probability of y equals 1 is 0.71, then what is the probability that y is equal to zero? Well, the chance of y being the one, and the chances of y being the zero, they've got to add up to one, right? So there's a 71 percent chance of it being one, there has to be a 29 percent or 0.29 chance of it being equal to zero.



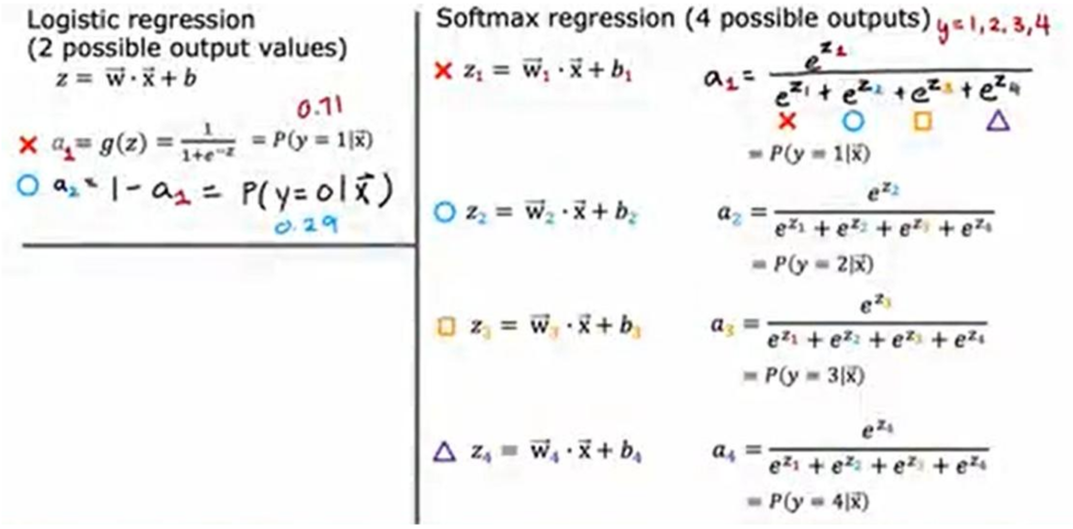
To embellish logistic regression a little bit in order to set us up for the generalization to softmax regression, I'm going to think of logistic regression as actually computing two numbers:

First a_1 which is this quantity that we had previously of the chance of y being equal to 1 given x, and second, I'm going to think of logistic regression as also computing a_2, which is 1 minus this which is just the chance of y being equal to zero given the input features x, and so a_1 and a_2, of course, have to add up to 1.

Let's now generalize this to softmax regression, and I'm going to do this with a concrete example of when y can take on four possible outputs, so y can take on the values 1, 2, 3 or 4. Here's what softmax regression will do, it will compute z_1 as w_1.product with x plus b_1, and then z_2 equals w_2.product of x plus b_2, and so on for z_3 and z_4. Here, w_1, w_2, w_3, w_4 as well as b_1, b_2, b_3, b_4, these are the parameters of softmax regression.

Next, here's the formula for softmax regression, we'll compute a_1 equals e^z_1 divided by e^ z_1, plus e^ z_2, plus e^z_3 plus, e^ z_4, and a_1 will be interpreted as the algorithms estimate of the chance of y being equal to 1 given the input features x.

Then the formula for softmax regression, we'll compute a_2 equals e^ z_2 divided by the same denominator, e^z_1 plus e^z_2, plus e^z_3, plus e^z4, and we'll interpret a_2 as the algorithms estimate of the chance that y is equal to 2 given the input features x. Similarly for a_3, where here the numerator is now e^z_3 divided by the same denominator, that's the estimated chance of y being a_3, and similarly a_4 takes on this expression.

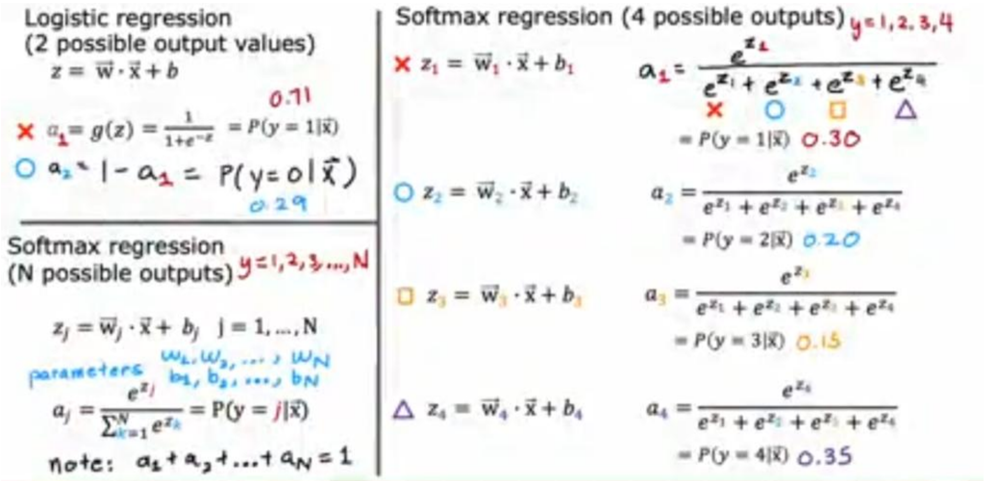


Whereas on the left, we wrote down the specification for the logistic regression model, these equations on the right are our specification for the softmax regression model. It has parameters w_1 through w_4, and b_1 through b_4, and if you can learn appropriate choices to all these parameters, then this gives you a way of predicting what's the chance of y being 1, 2, 3 or 4, given a set of input features x.

Quick quiz, let's see, run softmax regression on a new input x, and you find that a_1 is 0.30, a_2 is 0.20, a_3 is 0.15. What do you think a_4 will be? Why don't you take a look at this quiz and see if you can figure out the right answer? You might have realized that because the chance of y take on the values of 1, 2, 3 or 4, they have to add up to one, a_4 the chance of y being with a four has to be 0.35, which is 1 minus 0.3 minus 0.2 minus 0.15.

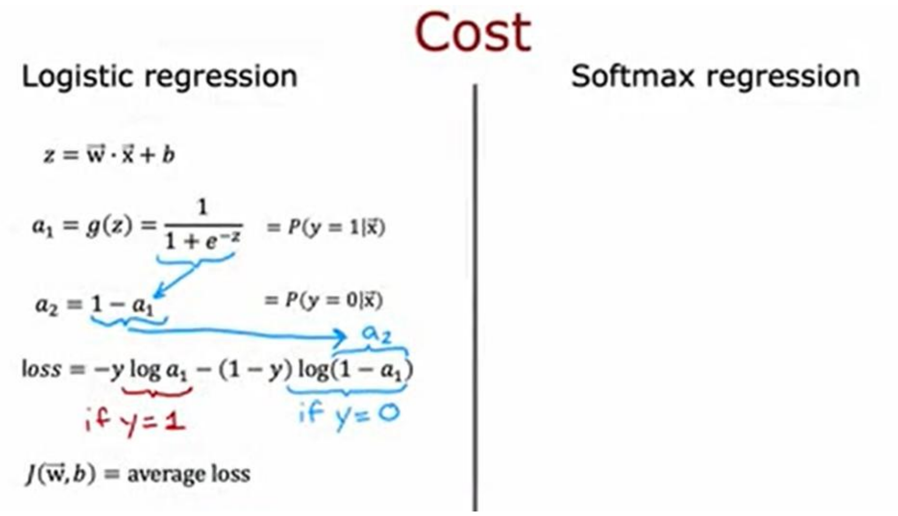
Here I wrote down the formulas for softmax regression in the case of four possible outputs, and let's now write down the formula for the general case for softmax regression.

In the general case, y can take on n possible values, so y can be 1, 2, 3, and so on up to n. In that case, softmax regression will compute to z_ j equals w_ j.product with x plus b_j, where now the parameters of softmax regression are w_1, w_2 through to w_n, as well as b_1, b_2 through b_n. Then finally, we'll compute a j equals e to the z j divided by sum from k equals 1 to n of e to the z sub k.



While here I'm using another variable k to index the summation because here j refers to a specific fixed number like j equals 1. A, j is interpreted as the model's estimate that y is equal to j given the input features x. Notice that by construction that this formula, if you add up a1, a2 all the way through a n, these numbers always will end up adding up to 1. We specified how you would compute the softmax regression model.

I won't prove it in this video, but it turns out that if you apply softmax regression with n equals 2, so there are only two possible output classes then softmax regression ends up computing basically the same thing as logistic regression. The parameters end up being a little bit different, but it ends up reducing to logistic regression model.

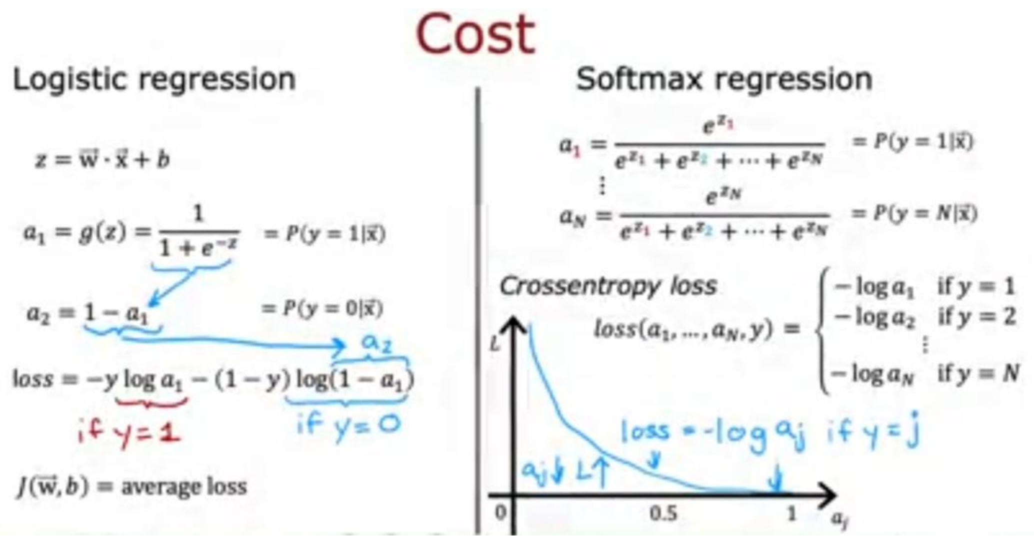


But that's why the softmax regression model is the generalization of logistic regression. Having defined how softmax regression computes it's outputs, let's now take a look at how to specify the cost function for softmax regression. Recall for logistic regression, this is what we had. We said z is equal to this. Then I wrote earlier that a1 is g of z, was interpreted as a probability of y is 1.

We also wrote a2 is the probability that y is equal to 0. Previously, we had written the loss of logistic regression as negative y log a1 minus 1 minus y log 1 minus a1. But 1 minus a1 is also equal to a2, because a2 is one minus a1 according to this expression over here.

I can rewrite or simplify the loss for logistic regression little bit to be negative y log a1 minus 1 minus y log of a2. In other words, the loss if y is equal to 1 is negative log a1. If y is equal to 0, then the loss is negative log a2, and then same as before the cost function for all the parameters in the model is the average loss, average over the entire training set. That was a cost function for this regression.

Let's write down the cost function that is conventionally use the softmax regression. Recall that these are the equations we use for softmax regression. The loss we're going to use for softmax regression is just this. The loss for if the algorithm puts a1 through an. The ground truth label is why is if y equals 1, the loss is negative log a1. Says negative log of the probability that it thought y was equal to 1, or if y is equal to 2, then I'm going to define as negative log a2. Y is equal to 2



The loss of the algorithm on this example is negative log of the probability it's thought y was equal to 2. On all the way down to if y is equal to n, then the loss is negative log of a n. To illustrate what this is doing, if y is equal to j, then the loss is negative log of a j.

That's what this function looks like. Negative log of a j is a curve that looks like this. If a j was very close to 1, then you beyond this part of the curve and the loss will be very small. But if it thought, say, a j had only a 50% chance then the loss gets a little bit bigger.

The smaller a j is, the bigger the loss. This incentivizes the algorithm to make a j as large as possible, as close to 1 as possible. Because whatever the actual value y was, you want the algorithm to say hopefully that the chance of y being that value was pretty large. Notice that in this loss function, y in each training example can take on only one value.

You end up computing this negative log of a j only for one value of a j, which is whatever was the actual value of y equals j in that particular training example. For example, if y was equal to 2, you end up computing negative log of a2, but not any of the other negative log of a1 or the other terms here. That's the form of the model as well as the cost function for softmax regression.

If you were to train this model, you can start to build multiclass classification algorithms. What we'd like to do next is take this softmax regression model, and fit it into a new network so that you really do something even better, which is to train a new network for multi-class classification.