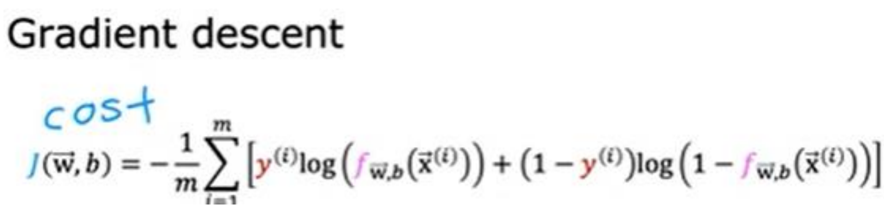
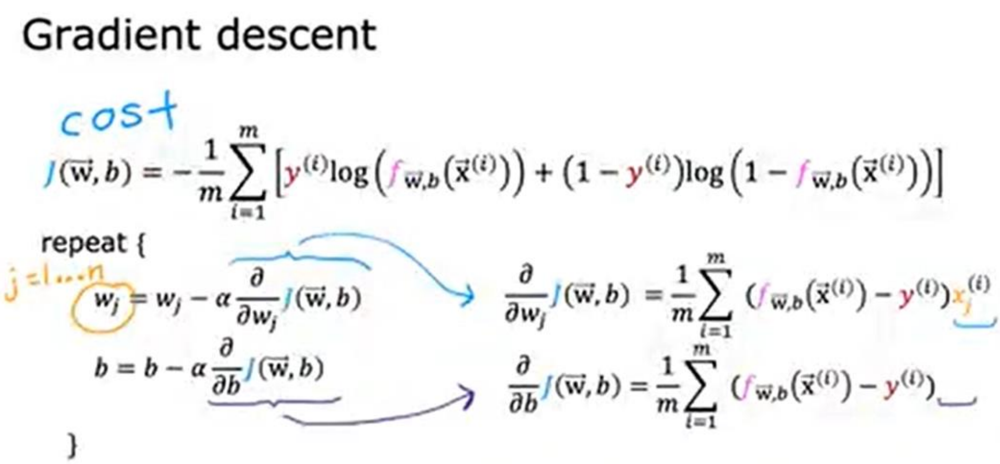
Mise en œuvre de la descente de gradient

To fit the parameters of a logistic regression model, we're going to try to find the values of the parameters w and b that minimize the cost function J of w and b, and we'll again apply gradient descent to do this. Let's take a look at how. In this video we'll focus on how to find a good choice of the parameters w and b. After you've done so, if you give the model a new input, x, say a new patients at the hospital with a certain tumor size and age, then these are diagnosis. The model can then make a prediction, or it can try to estimate the probability that the label y is one.

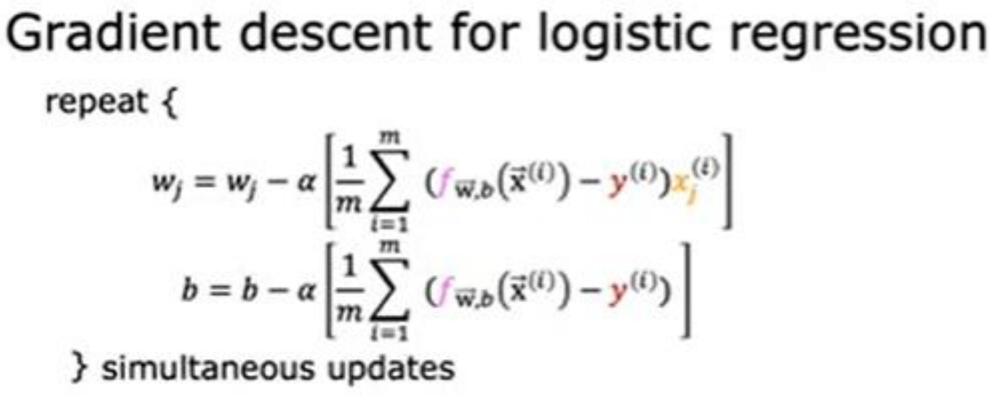


The average you can use to minimize the cost function is gradient descent. Here again is the cost function. If you want to minimize the cost j as a function of w and b, well, here's the usual gradient descent algorithm, where you repeatedly update each parameter as the 0 value minus Alpha, the learning rate times this derivative term. Let's take a look at the derivative of j with respect to w_j. This term up on top here, where as usual, j goes from one through n, where n is the number of features.

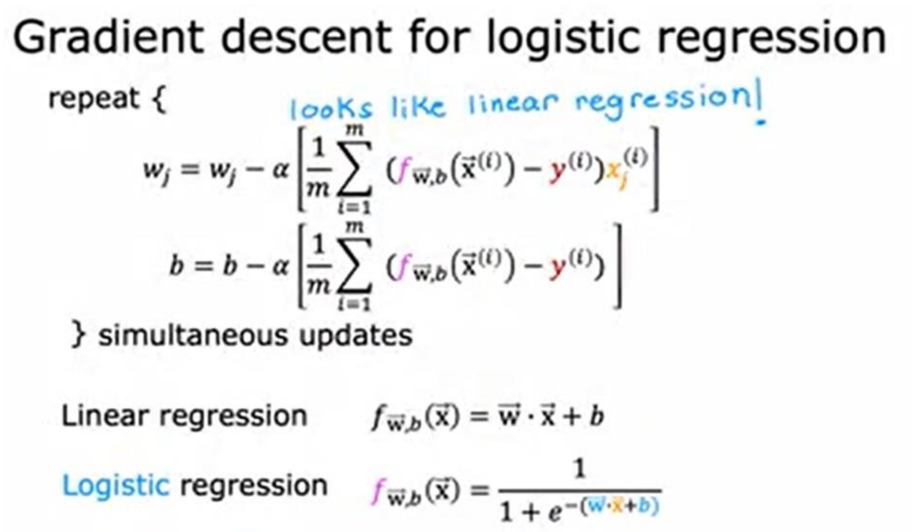
If someone were to apply the rules of calculus, you can show that the derivative with respect to w_j of the cost function capital J is equal to this expression over here, is 1 over m times the sum from 1 through m of this error term. That is f minus the label y times x_j. Here are just x I j is the j feature of training example i. Now let's also look at the derivative of j with respect to the parameter b. It turns out to be this expression over here. It's quite similar to the expression above, except that it is not multiplied by this x superscript i subscript j at the end.



Just as a reminder, similar to what you saw for linear regression, the way to carry out these updates is to use simultaneous updates, meaning that you first compute the right-hand side for all of these updates and then simultaneously overwrite all the values on the left at the same time. Let me take these derivative expressions here and plug them into these terms here.



Now, one funny thing you might be wondering is, that's weird. These two equations look exactly like the average we had come up with previously for linear regression so you might be wondering, is linear regression actually secretly the same as logistic regression? Well, even though these equations look the same, the reason that this is not linear regression is because the definition for the function f of x has changed. In linear regression, f of x is, this is wx plus b. But in logistic regression, f of x is defined to be the sigmoid function applied to wx plus b.



Although the algorithm written looked the same for both linear regression and logistic regression, actually they're two very different algorithms because the definition for f of x is not the same. When we talked about gradient descent for linear regression previously, you saw how you can monitor a gradient descent to make sure it converges. You can just apply the same method for logistic regression to make sure it also converges.

I've written out these updates as if you're updating the parameters w_j one parameter at a time. Similar to the discussion on vectorized implementations of linear regression, you can also use vectorization to make gradient descent run faster for logistic regression. I won't dive into the details of the vectorized implementation in this video. But you can also learn more about it and see the code in the optional labs. Now you know how to implement gradient descent for logistic regression. You might also remember feature scaling when we were using linear regression.

Where you saw how feature scaling, that is scaling all the features to take on similar ranges of values, say between negative 1 and plus 1, how they can help gradient descent to converge faster. Feature scaling applied the same way to scale the different features to take on similar ranges of values can also speed up gradient descent for logistic regression.