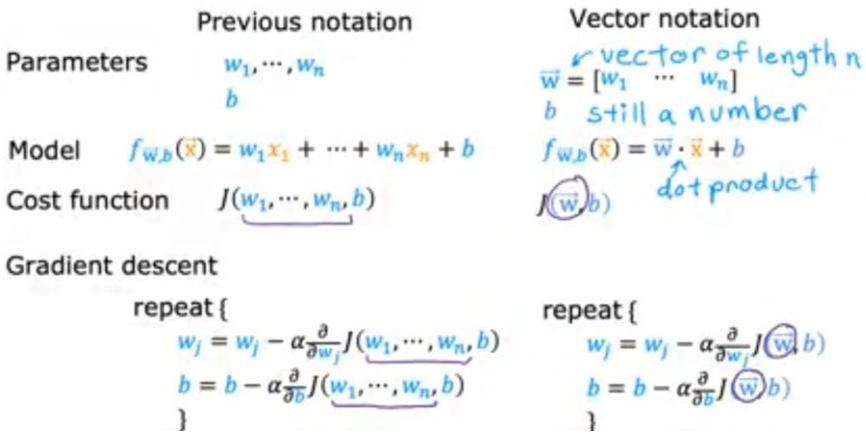
Descente de gradient pour la régression linéaire multiple

This would be cool. Let's quickly review what multiple linear regression look like. Using our previous notation, let's see how you can write it more succinctly using vector notation. We have parameters w_1 to w_n as well as b. But instead of thinking of w_1 to w_n as separate numbers, that is separate parameters, let's start to collect all of the w's into a vector w so that now w is a vector of length n. We're just going to think of the parameters of this model as a vector w, as well as b, where b is still a number same as before.

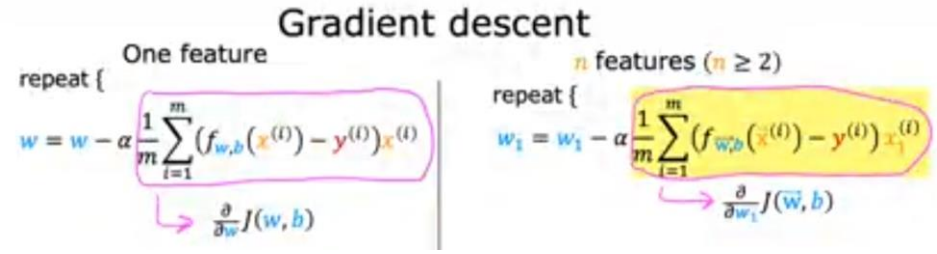
Whereas before we had to find multiple linear regression like this, now using vector notation, we can write the model as f_w, b of x equals the vector w dot product with the vector x plus b. Remember that this dot here means.product. Our cost function can be defined as J of w_1 through w_n, b. But instead of just thinking of J as a function of these and different parameters w_j as well as b, we're going to write J as a function of parameter vector w and the number b.

This w_1 through w_n is replaced by this vector W and J now takes this input of vector w and a number b and returns a number. Here's what gradient descent looks like. We're going to repeatedly update each parameter w_j to be w_j minus Alpha times the derivative of the cost J, where J has parameters w_1 through w_n and b. Once again, we just write this as J of vector w and number b.



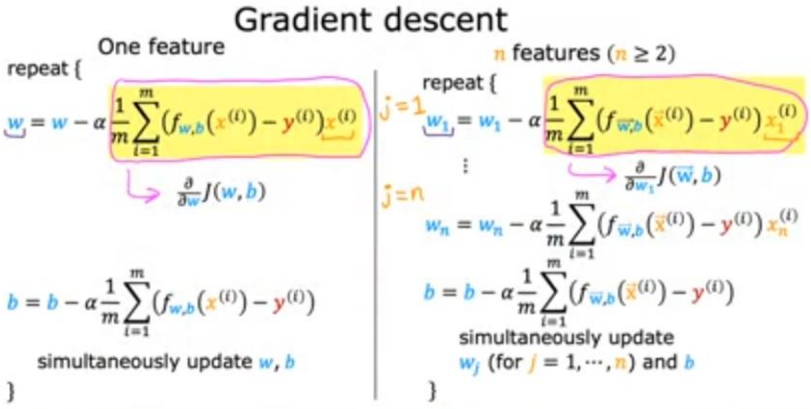
We'll see that gradient descent becomes just a little bit different with multiple features compared to just one feature. Here's what we had when we had gradient descent with one feature. We had an update rule for w and a separate update rule for b. Hopefully, these look familiar to you. This term here is the derivative of the cost function J with respect to the parameter w.

Similarly, we have an update rule for parameter b, with univariate regression, we had only one feature. We call that feature xi without any subscript. Now, here's a new notation for where we have n features, where n is two or more. We get this update rule for gradient descent. Update w_1 to be w_1 minus Alpha times this expression here and this formula is actually the derivative of the cost J with respect to w_1.



The formula for the derivative of J with respect to w_1 on the right looks very similar to the case of one feature on the left. The error term still takes a prediction f of x minus the target y. One difference is that w and x are now vectors and just as w on the left has now become w_1 here on the right, xi here on the left is now instead xi _1 here on the right and this is just for J equals 1. For multiple linear regression, we have J ranging from 1 through n and so we'll update the parameters w_1, w_2, all the way up to w_n, and then as before, we'll update b.

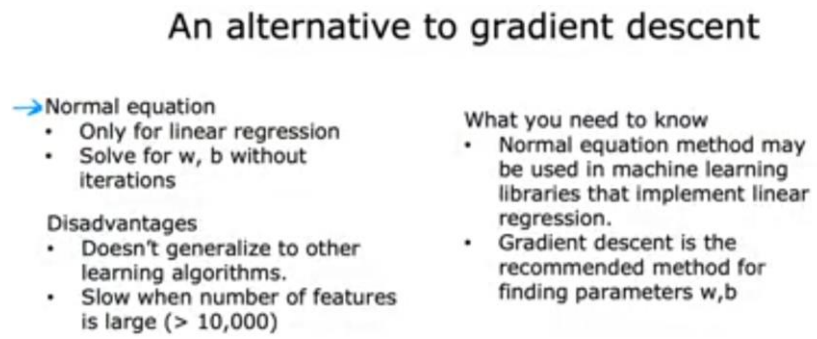
If you implement this, you get gradient descent for multiple regression. That's it for gradient descent for multiple regression.



Before moving on from this video, I want to make a quick aside or a quick side note on an alternative way for finding w and b for linear regression. This method is called the normal equation. Whereas it turns out gradient descent is a great method for minimizing the cost function J to find w and b, there is one other algorithm that works only for linear regression and pretty much none of the other algorithms you see in this specialization for solving for w and b and this other method does not need an iterative gradient descent algorithm.

Called the normal equation method, it turns out to be possible to use an advanced linear algebra library to just solve for w and b all in one goal without iterations. Some disadvantages of the normal equation method are; first unlike gradient descent, this is not generalized to other learning algorithms, such as the logistic regression algorithm that you'll learn about next week or the neural networks or other algorithms you see later in this specialization.

The normal equation method is also quite slow if the number of features and this large. Almost no machine learning practitioners should implement the normal equation method themselves but if you're using a mature machine learning library and call linear regression, there is a chance that on the backend, it'll be using this to solve for w and b.



If you're ever in the job interview and hear the term normal equation, that's what this refers to. Don't worry about the details of how the normal equation works. Just be aware that some machine learning libraries may use this complicated method in the back-end to solve for w and b. But for most learning algorithms, including how you implement linear regression yourself, gradient descents offer a better way to get the job done.