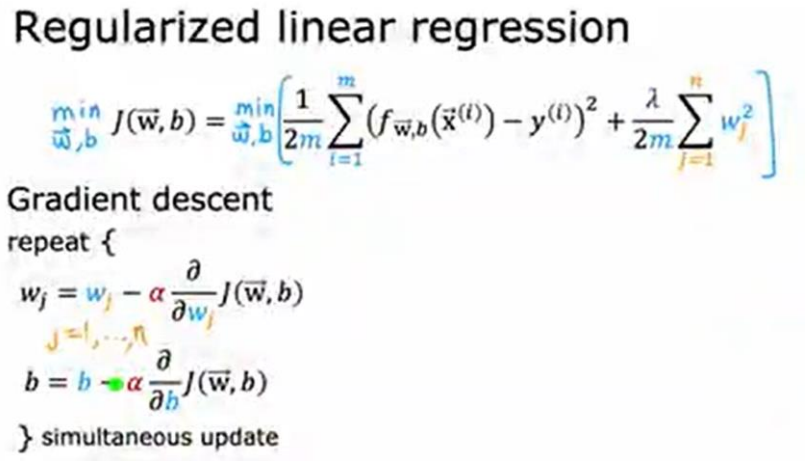
Régression linéaire régularisée

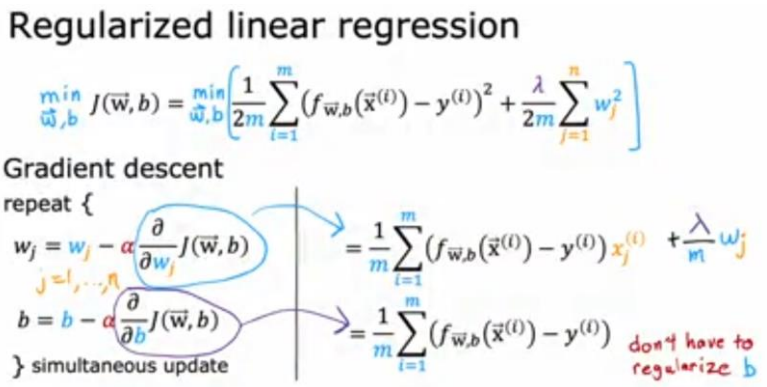
In this video, we'll figure out how to get gradient descent to work with regularized linear regression. Let's jump in. Here Is a cost function we've come up with in the last video for regularized linear regression. The first part is the usual squared error cost function, and now you have this additional regularization term, where Lambda is the regularization parameter, and you'd like to find parameters w and b that minimize the regularized cost function.

Previously we were using gradient descent for the original cost function, just the first term before we added that second regularization term, and previously, we had the following gradient descent algorithm, which is that we repeatedly update the parameters w, j, and b for j equals 1 through n according to this formula and b is also updated similarly. Again, Alpha is a very small positive number called the learning rate.

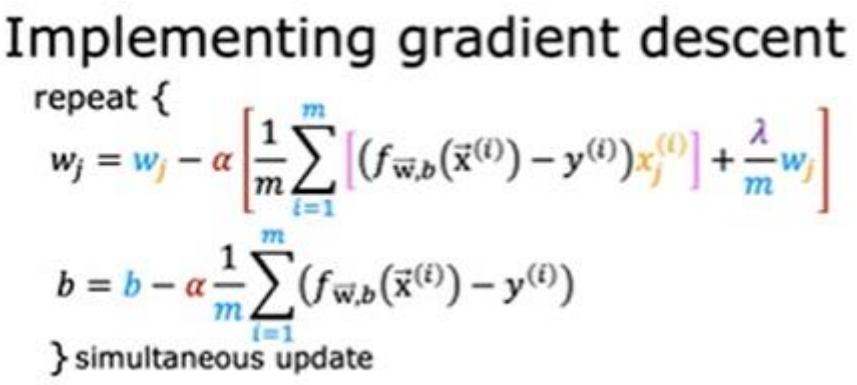


In fact, the updates for a regularized linear regression look exactly the same, except that now the cost, J, is defined a bit differently. Previously the derivative of J with respect to w_j was given by this expression over here, and the derivative respect to b was given by this expression over here. Now that we've added this additional regularization term, the only thing that changes is that the expression for the derivative with respect to w_j ends up with one additional term, this plus Lambda over m times w_j. And in particular for the new definition of the cost function j, these two expressions over here, these are the new derivatives of J with respect to w_j and the derivative of J with respect to b.

Recall that we don't regularize b, so we're not trying to shrink B. That's why the updated B remains the same as before, whereas the updated w changes because the regularization term causes us to try to shrink w_j. Let's take these definitions for the derivatives and put them back into the expression on the left to write out the gradient descent algorithm for regularized linear regression.



To implement gradient descent for regularized linear regression, this is what you would have your code do. Here is the update for w_j, for j equals 1 through n, and here's the update for b. As usual, please remember to carry out simultaneous updates for all of these parameters.



Now, in order for you to get this algorithm to work, this is all you need to know. But what I like to do in the remainder of this video is to go over some optional material to convey a slightly deeper intuition about what this formula is actually doing, as well as chat briefly about how these derivatives are derived. The rest of this video is completely optional. It's completely okay if you skip the rest of this video, but if you have a strong interests in math, then stick with me.

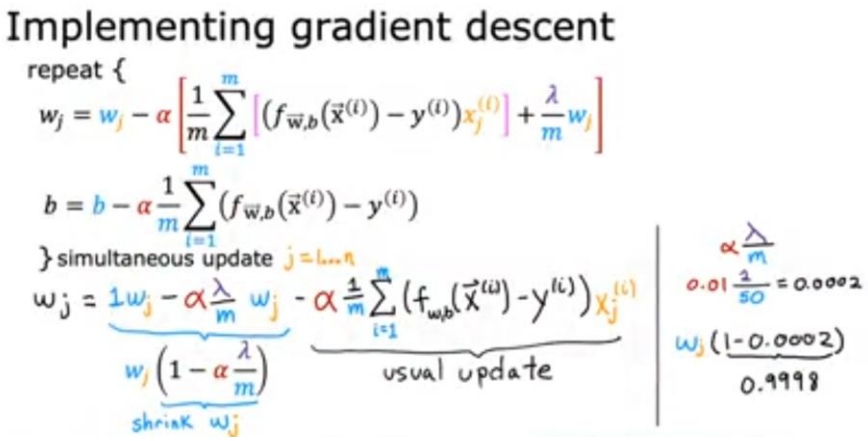
It is always nice to hang out with you here, and through these equations, perhaps you can build a deeper intuition about what the math and what the derivatives are doing as well. Let's take a look. Let's look at the update rule for w_j and rewrite it in another way. We're updating w_j as 1 times w_j minus Alpha times Lambda over m times w_j. I've moved the term from the end to the front here. Then minus Alpha times 1 over m, and then the rest of that term over there.

We just rearranged the terms a little bit. If we simplify, then we're saying that w_j is updated as w_j times 1 minus Alpha times Lambda over m, minus Alpha times this other term over here. You might recognize the second term as the usual gradient descent update for unregularized linear regression. This is the update for linear regression before we had regularization, and this is the term we saw in Week 2 of this course.

The only change we add regularization is that instead of w_j being set to be equal to w_j minus Alpha times this term is now w times this number minus the usual update. This is what we had in Week 1 of this course. What is this first term over here? Well, Alpha is a very small positive number, say 0.01. Lambda is usually a small number, say 1 or maybe 10.

Let's say lambda is 1 for this example and m is the training set size, say 50. When you multiply Alpha Lambda over m, say 0.01 times 1 divided by 50, this term ends up being a small positive number , say 0.0002, and thus, 1 minus Alpha Lambda over m is going to be a number just slightly less than 1, in this case, 0.9998. The effect of this term is that on every single iteration of gradient descent, you're taking w_j and multiplying it by 0.9998, that is by some numbers slightly less than one and for carrying out the usual update.

What regularization is doing on every single iteration is you're multiplying w by a number slightly less than 1, and that has effect of shrinking the value of w_j just a little bit. This gives us another view on why regularization has the effect of shrinking the parameters w_j a little bit on every iteration, and so that's how regularization works.



If you're curious about how these derivative terms were computed, I've just one last optional slide that goes through just a little bit of a calculation of the derivative term. Again, this slide and the rest of this video are completely optional, meaning you won't need any of this to do the practice labs and the quizzes. Let's step through quickly to derivative calculation. The derivative of J with respect to w_j looks like this.

Recall that f of x for linear regression is defined as w dot x plus b or w dot product x plus b. It turns out that by the rules of calculus, the derivatives look like this, is 1 over 2m times the sum i equals 1 through m of w dot x plus b minus y times 2x_j plus the derivative of the regularization term, which is Lambda over 2m times 2 w_j. Notice that the second term does not have the summation term from j equals 1 through n anymore. The 2's cancel out here and here, and also here and here. It simplifies to this expression over here.

Finally, remember that wx plus b is f of x, and so you can rewrite it as this expression down here. This is why this expression is used to compute the gradient in regularized linear regression. You now know how to implement regularized linear regression. Using this, you really reduce overfitting when you have a lot of features and a relatively small training set. This should let you get linear regression to work much better on many problems.