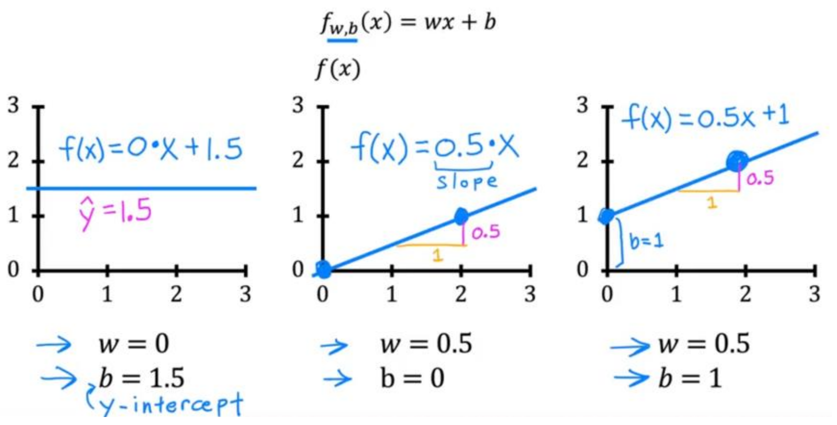
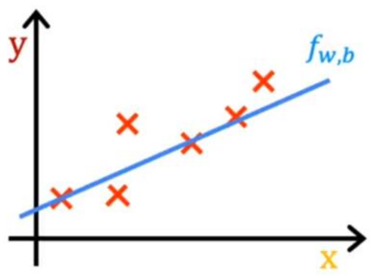
Formule de la fonction de coût

In order to implement linear regression the first key step is first to define something called a cost function.

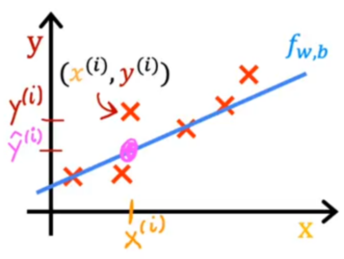
To introduce a little bit more terminology the w and b are called the parameters of the model. In machine learning parameters of the model are the variables you can adjust during training in order to improve the model. Sometimes you also hear the parameters w and b referred to as coefficients or as weights. Now let's take a look at what these parameters w and b do. Depending on the values you've chosen for w and b you get a different function f of x, which generates a different line on the graph.



When w is equal to 0 and b is equal to 1.5, then f looks like this horizontal line. In this case, the function f of x is 0 times x plus 1.5 so f is always a constant value. It always predicts 1.5 for the estimated value of y. Y hat is always equal to b and here b is also called the y intercept because that's where it crosses the vertical axis or the y axis on this graph. As a second example, if w is 0.5 and b is equal 0, then f of x is 0.5 times x. When x is 0, the prediction is also 0, and when x is 2, then the prediction is 0.5 times 2, which is 1. You get a line that looks like this and notice that the slope is 0.5 divided by 1. The value of w gives you the slope of the line, which is 0.5. Finally, if w equals 0.5 and b equals 1, then f of x is 0.5 times x plus 1 and when x is 0, then f of x equals b, which is 1 so the line intersects the vertical axis at b, the y intercept. Also when x is 2, then f of x is 2, so the line looks like this. Again, this slope is 0.5 divided by 1 so the value of w gives you the slope which is 0.5.



Recall that you have a training set like the one shown here. With linear regression, what you want to do is to choose values for the parameters w and b so that the straight line you get from the function f somehow fits the data well. Like maybe this line shown here. When I see that the line fits the data visually, you can think of this to mean that the line defined by f is roughly passing through or somewhere close to the training examples as compared to other possible lines that are not as close to these points.



$$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$$

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

Find w, b:
$\hat{y}^{(i)}$ is close to $y^{(i)}$ for all $(x^{(i)}, y^{(i)})$.

Just to remind you of some notation, a training example like this point here is defined by x superscript i, y superscript i where y is the target. For a given input x^i, the function f also makes a predictive value for y and a value that it predicts to y is y hat i shown here. For our choice of a model f of x^i is w times x^i plus b. Stated differently, the prediction y hat i is f of wb of x^i where for the model we're using f of x^i is equal to wx^i plus b. Now the question is how do you find values for w and b so that the prediction y hat i is close to the true target y^i for many or maybe all training examples x^i, y^i.

To answer that question, let's first take a look at how to measure how well a line fits the training data. To do that, we're going to construct a cost function. The cost function takes the prediction y hat and compares it to the target y by taking y hat minus y. This difference is called the error, we're measuring how far off to prediction is from the target. Next, let's computes the square of this error. Also, we're going to want to compute this term for different training examples i in the training set. When measuring the error, for example i, we'll compute this squared error term.

Finally, we want to measure the error across the entire training set. In particular, let's sum up the squared errors like this. We'll sum from i equals 1,2, 3 all the way up to m and remember that m is the number of training examples, which is 47 for this dataset. Notice that if we have more training examples m is larger and your cost function will calculate a bigger number. This is summing over more examples. To build a cost function that doesn't automatically get bigger as the training set size gets larger by convention, we will compute the average squared error instead of the total squared error and we do that by dividing by m like this. We're nearly there. Just one last thing.

By convention, the cost function that machine learning people use actually divides by 2 times m. The extra division by 2 is just meant to make some of our later calculations look neater, but the cost function still works whether you include this division by 2 or not.

Cost function: Squared error cost function

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \Big(\underbrace{\hat{y}^{(i)} - y^{(i)}}_{\text{error}} \Big)^2$$

m = number of training examples

$$\underset{\substack{\uparrow \\ \text{intuition}}}{J}(w,b) = \frac{1}{2m} \sum_{i=1}^{m} \big(f_{w,b}(x^{(i)}) - y^{(i)} \big)^2$$

This expression right here is the cost function and we're going to write J of wb to refer to the cost function. This is also called the squared error cost function, and it's called this because you're taking the square of these error terms.This expression right here is the cost function and we're going to write J of wb to refer to the cost function. This is also called the squared error cost function, and it's called this because you're taking the square of these error terms.

Just as a reminder, the prediction y hat is equal to the outputs of the model f at x. We can rewrite the cost function J of wb as 1 over 2m times the sum from i equals 1 to m of f of x^i minus y^i the quantity squared. Eventually we're going to want to find values of w and b that make the cost function small. But before going there, let's first gain more intuition about what J of wb is really computing.

Intuition de la fonction de coût

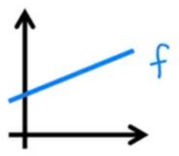
To recap, here's what we've seen about the cost function so far. You want to fit a straight line to the training data, so you have this model, fw, b of x is w times x, plus b. Here, the model's parameters are w, and b. Now, depending on the values chosen for these parameters, you get different straight lines like this. You want to find values for w, and b, so that the straight line fits the training data well. To measure how well a choice of w, and b fits the training data, you have a cost function J. What the cost function J does is, it measures the difference between the model's predictions, and the actual true values for y. What you see later, is that linear regression would try to find values for w, and b, then make a J of w be as small as possible. In math, we write it like this.

model:

$$\underline{f_{w,b}(x) = wx + b}$$

parameters:

$$\underline{w, b}$$



cost function:

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

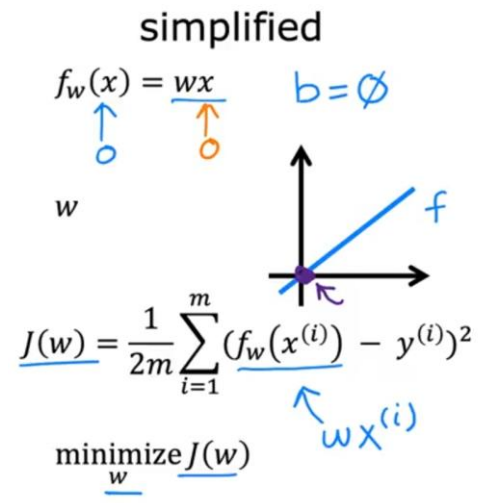
goal:

$$\underset{w,b}{\text{minimize}}\, J(w,b)$$

Now, in order for us to better visualize the cost function J, this work of a simplified version of the linear regression model. We're going to use the model fw of x, is w times x. You can think of this as taking the original model on the left, and getting rid of the parameter b, or setting the parameter b equal to 0.

It just goes away from the equation, so f is now just w times x. You now have just one parameter w, and your cost function J, looks similar to what it was before. Taking the difference, and squaring it, except now, f is equal to w times xi, and J is now a function of just w. The goal becomes a little bit different as well, because you have just one parameter, w, not w and b. With this simplified model, the goal is to find the value for w, that minimizes J of w.

To see this visually, what this means is that if b is set to 0, then f defines a line that looks like this. You see that the line passes through the origin here, because when x is 0, f of x is 0 too. Now, using this simplified model, let's see how the cost function changes as you choose different values for the parameter w.


simplified
$f_w(x) = wx$
$b=\emptyset$
0
0
w
f
$J(w) = \frac{1}{2m}\sum_{i=1}^{m}(f_w(x^{(i)}) - y^{(i)})^2$
$wx^{(i)}$
$\underset{w}{\text{minimize}}\, J(w)$

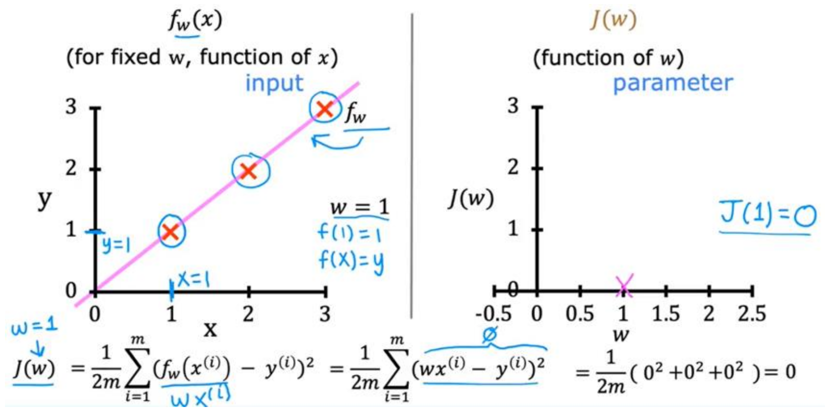
First, notice that for f subscript w, when the parameter w is fixed, that is, is always a constant value, then fw is only a function of x, which means that the estimated value of y depends on the value of the input x. In contrast, looking to the right, the cost function J, is a function of w, where w controls the slope of the line defined by f w. The cost defined by J, depends on a parameter, in this case, the parameter w. Let's go ahead, and plot these functions, fw of x, and J of w side-by-side so you can see how they are related.

We'll start with the model, that is the function fw of x on the left. Here are the input feature x is on the horizontal axis, and the output value y is on the vertical axis. Here's the plots of three points representing the training set at positions 1, 1, 2, 2, and 3,3. Let's pick a value for w. Say w is 1. For this choice of w, the function fw, they'll say this straight line with a slope of 1.

Now, what you can do next is calculate the cost J when w equals 1. You may recall that the cost function is defined as follows, is the squared error cost function. If you substitute fw(X^i) with w times X^i, the cost function looks like this. Where this expression is now w times X^i minus Y^i. For this value of w, it turns out that the error term inside the cost function, this w times X^i minus Y^i is equal to 0 for each of the three data points.

Because for this data-set, when x is 1, then y is 1. When w is also 1, then f(x) equals 1, so f(x) equals y for this first training example, and the difference is 0. Plugging this into the cost function J, you get 0 squared. Similarly, when x is 2, then y is 2, and f(x) is also 2. Again, f(x) equals y, for the second training example. In the cost function, the squared error for the second example is also 0 squared. Finally, when x is 3, then y is 3 and f(3) is also 3.

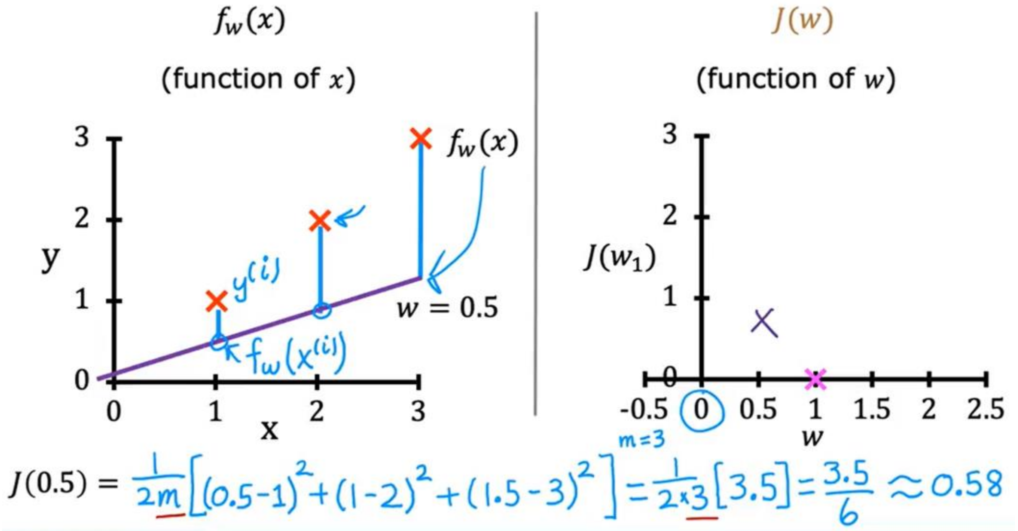
In a cost function the third squared error term is also 0 squared. For all three examples in this training set, f(X^i) equals Y^i for each training example i, so f(X^i) minus Y^i is 0. For this particular data-set, when w is 1, then the cost J is equal to 0. Now, what you can do on the right is plot the cost function J. Notice that because the cost function is a function of the parameter w, the horizontal axis is now labeled w and not x, and the vertical axis is now J and not y. You have J(1) equals to 0.



In other words, when w equals 1, J(w) is 0, so let me go ahead and plot that. Now, let's look at how F and J change for different values of w. W can take on a range of values, so w can take on negative values, w can be 0, and it can take on positive values too.

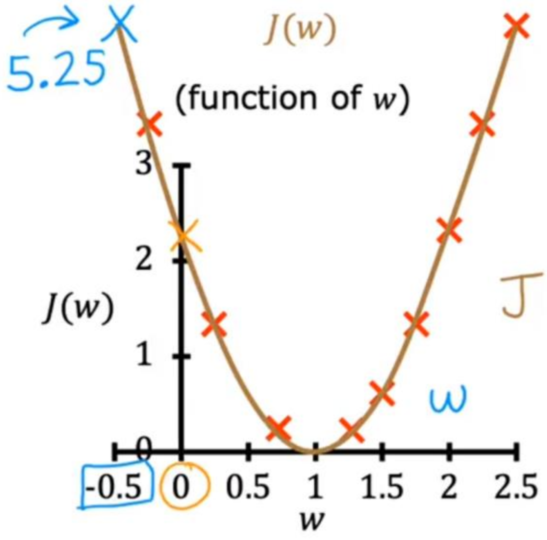
What if w is equal to 0.5 instead of 1, what would these graphs look like then? Let's go ahead and plot that. Let's set w to be equal to 0.5, and in this case, the function f(x) now looks like this, is a line with a slope equal to 0.5. Let's also compute the cost J, when w is 0.5. Recall that the cost function is measuring the squared error or difference between the estimator value, that is y hat I, which is F(X^i), and the true value, that is Y^i for each example i. Visually you can see that the error or difference is equal to the height of this vertical line here when x is equal to 1. Because this lower line is the gap between the actual value of y and the value that the function f predicted, which is a bit further down here. For this first example, when x is 1, f(x) is 0.5.

The squared error on the first example is 0.5 minus 1 squared. Remember the cost function, we'll sum over all the training examples in the training set. Let's go on to the second training example. When x is 2, the model is predicting f(x) is 1 and the actual value of y is 2. The error for the second example is equal to the height of this little line segment here, and the squared error is the square of the length of this line segment, so you get 1 minus 2 squared. Let's do the third example. Repeating this process, the error here, also shown by this line segment, is 1.5 minus 3 squared. Next, we sum up all of these terms, which turns out to be equal to 3.5.



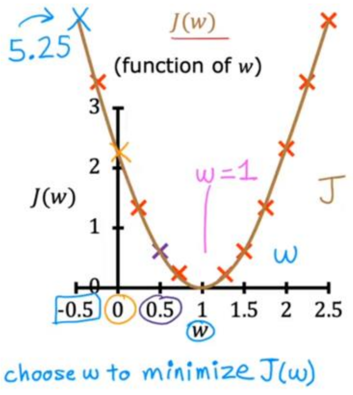
Then we multiply this term by 1 over 2m, where m is the number of training examples. Since there are three training examples m equals 3, so this is equal to 1 over 2 times 3, where this m here is 3. If we work out the math, this turns out to be 3.5 divided by 6. The cost J is about 0.58. Let's go ahead and plot that over there on the right.

You can keep doing this for other values of w. Since w can be any number, it can also be a negative value. If w is negative 0.5, then the line f is a downward-sloping line like this. It turns out that when w is negative 0.5 then you end up with an even higher cost, around 5.25, which is this point up here. You can continue computing the cost function for different values of w and so on and plot these. It turns out that by computing a range of values, you can slowly trace out what the cost function J looks like and that's what J is.



To recap, each value of parameter w corresponds to different straight line fit, f of x, on the graph to the left. For the given training set, that choice for a value of w corresponds to a single point on the graph on the right because for each value of w, you can calculate the cost J of w. For example, when w equals 1, this corresponds to this straight line fit through the data and it also corresponds to this point on the graph of J, where w equals 1 and the cost J of 1 equals 0. Whereas when w equals 0.5, this gives you this line which has a smaller slope.

This line in combination with the training set corresponds to this point on the cost function graph at w equals 0.5. For each value of w you wind up with a different line and its corresponding costs, J of w, and you can use these points to trace out this plot on the right. Given this, how can you choose the value of w that results in the function f, fitting the data well? Well, as you can imagine, choosing a value of w that causes J of w to be as small as possible seems like a good bet.



J is the cost function that measures how big the squared errors are, so choosing w that minimizes these squared errors, makes them as small as possible, will give us a good model. In this example, if you were to choose the value of w that results in the smallest possible value of J of w you'd end up picking w equals 1.

As you can see, that's actually a pretty good choice. This results in the line that fits the training data very well. That's how in linear regression you use the cost function to find the value of w that minimizes J. In the more general case where we had parameters w and b rather than just w, you find the values of w and b that minimize J.

goal of linear regression:

$$\underset{w}{\text{minimize}}\, J(w)$$

general case:

$$\underset{w,b}{\text{minimize}}\, J(w,b)$$

To summarize, you saw plots of both f and J and worked through how the two are related. As you vary w or vary w and b you end up with different straight lines and when that straight line passes across the data, the cause J is small. The goal of linear regression is to find the parameters w or w and b that results in the smallest possible value for the cost function J. Now in this video, we worked through our example with a simplified problem using only w. In the next video, let's visualize what the cost function looks like for the full version of linear regression using both w and b. You see some cool 3D plots.