**LINEAR REGRESSION NOTES**

**NOTES FROM SPRING 2023:**

* Did about 15 minutes on day 2 of class of this material, from 1st notebook
* Next day, did all the way through linear regression 1 notebook and did cost function intuition notebook. Ended with finishing contour plots and introducting (5 mins) idea of grad descent.
* Next day: Review. Begin cost function playground notebook. Go through it.
  + Then go into gradient descent (lin reg 2 notebook).

Start in linear-regression-1 notebook. Go through notebook. At end of notebook, go here:

**COST FUNCTION INTUITION**

**[This is done mostly on the board, with backup from the cost-func-intuition notebook]**

Let’s get some intuition about how this cost function works.

Chart

Description automatically generated

Let’s simplify this:

Schematic

Description automatically generated with medium confidence

Restricted to always go through origin.

So now we have two functions:

Graphical user interface

Description automatically generated with low confidence

Let’s imagine some training data:

Graphical user interface, application

Description automatically generated

Let’s pick a value for w, say w=1. So slope is 1.

(plot the graph)

A picture containing text, watch, clock, gauge

Description automatically generated

Calculate this for each data point. They should all come out to zero!

Let’s plot this separately. W on indep axis, J(w) dep axis.

Chart

Description automatically generated

What if w=0.5?

Chart

Description automatically generated

What if w=0? (horizontal line)

Chart

Description automatically generated

Chart, line chart

Description automatically generated

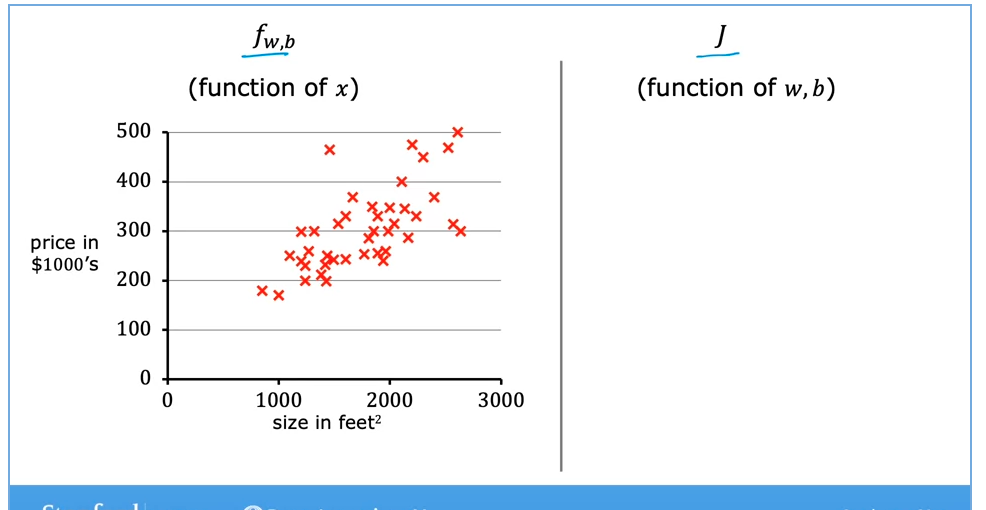
Key ideas:

* Each choice of w produces a LINE on the left graph, but a POINT on the right graph.
* How do we choose a w so that the J(w) function is minimized?
* For this example, it’s w=1
* Text

  Description automatically generated

**COST FUNCTION INTUITION IN GENERAL CASE**

But, now what if we let b be anything, not just zero?



Let’s make up some numbers for w and b. say w=0.06 and b=50.

Chart, scatter chart

Description automatically generated

So this model isn’t great, it consistently underestimates. Our old graph of J(w) had w on the x-axis and J on the y-axis. So we had our graph that always looked like a parabola. But now this new graph of J(w,b) will be in 3 dimensions, with w and b as independent variables and J has the dependent variable.

[show 3d plot from notebook]. NOTE THAT THIS IS JUST AN EXAMPLE, not the real thing.

So this now a surface. And any point on the surface represents a particular choice of w and b. Just like in the previous example where any particular point on the line represented a particular choice of w.

**CONTOUR PLOTS**

Draw a mountain. Imagine we draw lines of equal height above the ground around the mountain. Fly above the mountain, looking down.

Imagine slicing the mountain horizontally. So each slice gets a collection of points all at the same height above the ground.

Do an elongated mountain in 1 dimension, so we have ellipses.

[**skip to cost func intuition notebook.** Run first 2 plots with Z=Z1, Z2, Z3, Z4. There is a regular plot and a contour plot. Show them to students.]

**[now run 2nd set of plots in notebook, for our real data]**

Chart

Description automatically generated

So now return to the plots. The left plot has lines of f for different values of w, b. These correspond to points on the right graph of J. So where is the minimum on the right graph? It’s at the center of the circles/ovals.

(Remember J is not the y axis anymore, it’s represented abstractly as a “height” coming out of the page/screen).

So now, equally bad lines on the left graph (that generate the exact same amount of squared error) all correspond to the same value of J on the right graph. So they’ll all be on the same circle/oval.

**OUR CONTOUR PLOTS**

Put on board the housing data on left, and right draw the contour plots from the notebook above.

**Chart, scatter chart

Description automatically generated**

So imagine w=-0.15, b = 800. Then we get this LINE on the left, and this POINT on the right.

This line sucks! Many of the predictions are quite far from the line!

So the point on the right graph is far from the center.

Another example. W = 0, b = 360.

Chart, scatter chart

Description automatically generated

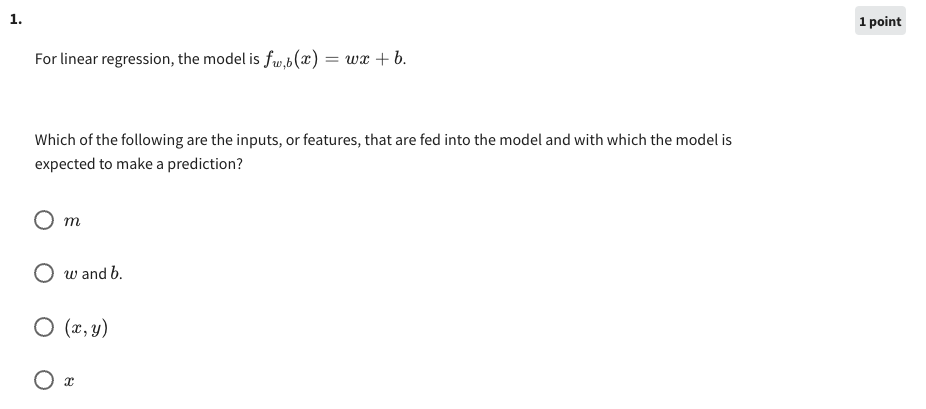
One more example

Chart, scatter chart

Description automatically generated

**Go to cost-function-playground notebook.**

Ask students:



Answer = x

2. For linear regression, if you find parameters *w* and *b* so that *J*(*w*,*b*) is very close to zero, what can you conclude?

**1 point**

This is never possible -- there must be a bug in the code.

ANSWER: The selected values of the parameters �*w* and �*b* cause the algorithm to fit the training set really well.

The selected values of the parameters �*w* and �*b* cause the algorithm to fit the training set really poorly.

**LOGISTIC REGRESSION INTUITION**

* Let’s figure out why this update equation works.
* Write this on board:
* Diagram, schematic

  Description automatically generated
* Now we know that in our example, we are fitting f(x) = wx +b, and our parameters are w & b. Therefore the cost function J(w, b) is a 3-dimensional function with two inputs (w, b) and one output (the error).
* Let’s look at how this would look in the single dimensional case, if we were to fix b at some number (which doesn’t matter). I’m doing this mostly so I can draw on the board in two dimensions, because then:
  + If our model is f(x) = wx + b. [but b is fixed], then J(w) is just 2 dimensional plot.
  + A picture containing graphical user interface

    Description automatically generated
* Now we want to minimize J(w).
* We know J is squared error, so when we graph it, it’s always a parabola.
* A picture containing icon

  Description automatically generated
* Now we know how to minimize functions of one variable – we learned this in calculus. We can take the derivative of J(w) wrt to w, set it equal to zero and solve it for w.
  + And it turns out we actually can do this for linear regression, though it’s a little messy, mostly because of the summation in the equation for the cost function. And we can’t always do it for other types of regression or other machine learning situations. What do we do in those situations where we can’t solve it exactly? Turns out as long as the function J is differentiable, we can find a very close approximation to the solution (that the limit, converges to the correct solution under most conditions). This is what gradient descent does.
* Remember gradient descent starts by picking initial guesses for all its variables, in this case w.
* A picture containing shape

  Description automatically generated
* Icon

  Description automatically generated
* Notice that this “d” is a regular derivative d.
* We remember from calculus that this gives us a tangent line.
* Shape

  Description automatically generated
* So at our dot, the derivative is positive (upward slope).
* So we have w = w – (alpha)(positive number).
* This will move w to the left (because w gets smaller).
* Reverse it for if the dot is on the other side.
* Diagram

  Description automatically generated with low confidence
* Illustrate the learning rate how the learning rate works. (jump back to notebook)