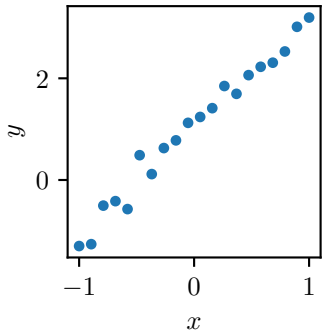
SEMT20003, Part 2: Linear regression

Laurence Aitchison

1 Regression as curve fitting

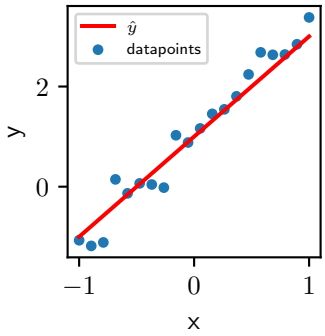
Ultimately, regression is just curve-fitting. And the simplest form of regression is fitting a straight line to a dataset involving inputs, x_i and targets, y_i.



You can draw a line by hand, or guess a good function in simple cases such as this. For instance, looking at the plot, we can see that the intercept (where the line crosses $x = 0$) is around 1 and the slope is around 2 (it goes up roughly 2 on the y-axis if we move right 1 on the x-axis). Thus, we could guess:

$$\hat{y}(x) = 2x + 1 \tag{1}$$

where $\hat{y}(x_i)$ is our prediction of the value of y_i.



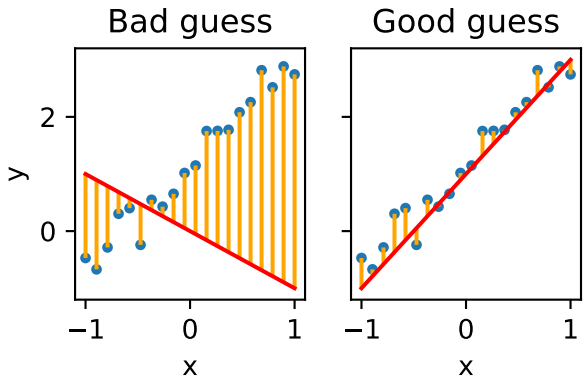
But there are many situations where this simple approach isn't going to work:

- Too many datapoints to visualise easily.
- Too complicated function to guess.
- The input, x_i isn't just one number, but a vector of numbers, $\mathbf{x}_i$ representing multiple "features" (for instance, if we're trying to predict sales of a product, our prediction might depend on size, shape, color, weight of a product).

In these more complicated contexts, we can't just draw something by eye. We're going to need to fit the curves using maths! And it turns out that this curve-fitting is the starting point for AI.

2 Quantifying good and bad predictions using distances/losses

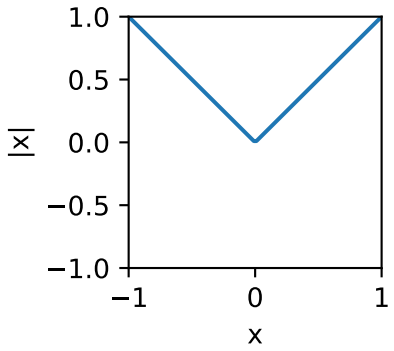
To get a mathematical procedure for fitting good predictions, we need to know what makes a good prediction and what makes a bad prediction. Intuitively, a good prediction is close to the y_i's in the data, and a bad prediction is far from the y_i's in the data.



But to choose better predictors, we need to formally nail down this notion of distance between the prediction and the data. Specifically, the goal is to measure the distance between the prediction, $\hat{y}(x_i)$, and the data, y_i. There are a huge number of sensible things we could choose, corresponding to different mathematical formalisations of "distance". Any given notion of distance will give rise to a "loss", $\mathcal{L}$. Generally speaking, the loss is a sum of distances for each datapoint, because we want the prediction for every datapoint to be close to the data. Perhaps the most obvious is to use the sum of distances between y_i and $\hat{y}_i$,

$$\mathcal{L} = \sum_i |\hat{y}(x_i) - y_i| \,, \tag{2}$$

where,



This has lots of nice properties, and is a good option in many cases. However, the "kink" at $x = 0$ can cause numerical problems, and can make analytic maths harder. To make the maths easier, the standard approach is to use the sum of

squared distances,

$$\mathcal{L} = \sum_i \left(\hat{y}(x_i) - y_i\right)^2, \tag{3}$$

To check that this loss does something sensible, we computed it for the Good Guess/Bad Guess Figure above. The obviously badly-fitting line (left) has a loss of 34.1 and the obviously better-fitting line (right) has a loss of 4.8. So the better-fitting line indeed has a lower loss.

3 Different choices of functions

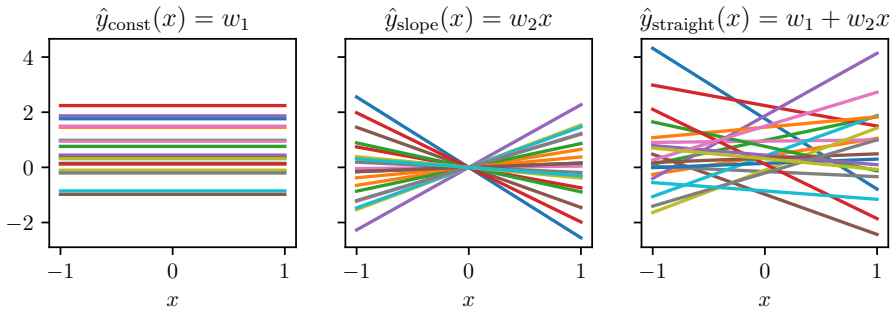
We now have a "loss" the quantifies how good any given prediction-function, $\hat{y}(x_i)$, is. In the previous section, we used the loss to show that one particular choice of prediction-function was better than another choice of prediction-function. So we might want to try loads of different choices of prediction-function, and find the one with the lowest loss. But that doesn't scale:

- Where are these possible prediction-functions going to come from?
- What if we have *loads* of possible prediction-functions?

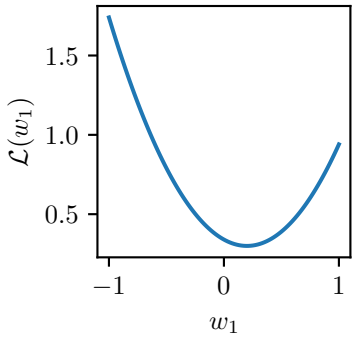
To solve these problems, we're going to write our family of possible prediction functions using parameters (NNs will turn out to be the same idea, but where we have *loads* more parameters: usually somewhere between a thousand and a trillion). For instance we could consider three predictors: the constant predictor, the "slope" predictor, and the straight-line predictor,

$$\hat{y}_{\text{const}}(x_i) = w_1 \qquad \hat{y}_{\text{slope}}(x_i) = w_2 x_i \qquad \hat{y}_{\text{straight}}(x_i) = w_1 + w_2 x_i. \tag{4}$$

We can choose different values for the parameters, w_1 and w_2. Possible functions in that family are,



There are infinitely many functions in these families, because there are infinitely many possible choices of w_1 and w_2. That means we can't try all possible values of w_1 and w_2. What can we do instead? It turns out we can find the lowest-loss function analytically, using calculus. Remember that the minimum of a function is often where the function has slope 0.



3.1 Finding the best constant predictor

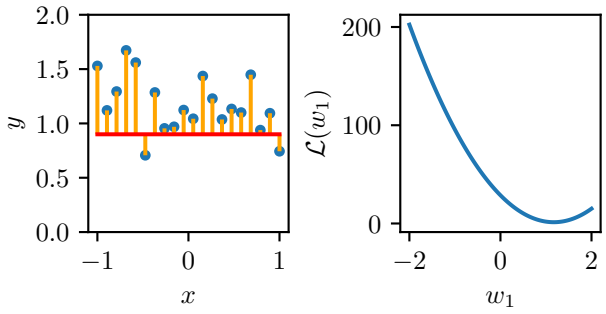
The constant predictor always predicts that y is w_1,

$$\hat{y}_{\text{const}}(x_i) = w_1 \tag{5}$$

That means the loss (sum of squared distances between the prediction and actual value) becomes,

$$\mathcal{L}_{\text{const}}(w_1) = \sum_i \left(\hat{y}_{\text{const}}(x_i) - y_i\right)^2 = \sum_i \left(w_1 - y_i\right)^2 \tag{6}$$

Our goal is to to find the prediction-function with the lowest loss. We can do this graphically,



and we can see that the optimal value of w_1 is about 1.1, which is about in the middle of the data.

But can we do a better job? Look at the smooth curve of loss against w_1: the minimum of the curve is the location where the gradient is zero. So, we can find that minimum by treating $\mathcal{L}_{\text{const}}(w_1)$, as a function of the parameters. We find the gradient of $\mathcal{L}_{\text{const}}(w_1)$, then solve for the value of w_1 at which the gradient

is zero. We start by differentating the loss,

$$\frac{\partial \mathcal{L}_{\text{const}}(w_1)}{\partial w_1} = \frac{\partial}{\partial w_1}\left[\sum_{i=1}^{N}(y_i - w_1)^2\right]. \tag{7}$$

(Note I'm playing fast-and-loose with d and ∂. This is done all the time by deep learners, so it is important to be able to work with it. I'm going to be more careful when we come to talking about backprop in general, where this distinction matters a bit more). Swap the gradient and the sum,

$$\frac{\partial \mathcal{L}_{\text{const}}(w_1)}{\partial w_1} = \sum_{i=1}^{N}\frac{\partial}{\partial w_1}(y_i - w_1)^2. \tag{8}$$

Now, we take $u_i = y_i - w_1$,

$$\frac{\partial \mathcal{L}_{\text{const}}(w_1)}{\partial w_1} = \sum_{i=1}^{N}\frac{\partial u_i^2}{\partial w_1} \tag{9}$$

Apply the chain rule,

$$\frac{\partial \mathcal{L}_{\text{const}}(w_1)}{\partial w_1} = \sum_{i=1}^{N}\frac{\partial u_i^2}{\partial u_i}\frac{\partial u_i}{\partial w_1} \tag{10}$$

The individual derivatives are,

$$\frac{\partial u_i^2}{\partial u_i} = 2u_i \tag{11}$$

$$\frac{\partial u_i}{\partial w_1} = \frac{\partial}{\partial w_1}[y_i - w_1] = -1 \tag{12}$$

(note that the data, y_i are fixed and independent of w_1, so $\frac{\partial y_i}{\partial w_1} = 0$). Thus,

$$\frac{\partial \mathcal{L}_{\text{const}}(w_1)}{\partial w_1} = 2\sum_{i=1}^{N}(w_1 - y_i). \tag{13}$$

Now, we solve for the value of w_1 at which the gradient is zero,

$$0 = \frac{\partial \mathcal{L}_{\text{const}}(w_1)}{\partial w_1} = 2\sum_{i=1}^{N}(w_1 - y_i). \tag{14}$$

Divide by 2 and expand the bracket,

$$0 = \sum_{i=1}^{N} w_1 - \sum_{i=1}^{N} y_i \tag{15}$$

The first term, $\sum_{i=1}^{N} w_1$, is just $w_1 + w_1 + \cdots + w_1$ N times, so it equals Nw_1,

$$0 = Nw_1 - \sum_{i=1}^{N} y_i \tag{16}$$

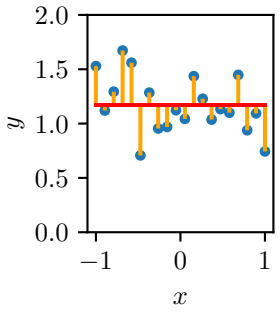
Add $\sum_{i=1}^{N} y_i$ to both sides,

$$Nw_1 = \sum_{i=1}^{N} y_i \tag{17}$$

Finally, solve for w_1, by dividing both sides by N,

$$w_1 = \frac{1}{N} \sum_{i=1}^{N} y_i. \tag{18}$$

Is what we've ended up with sensible? Yes! Remember that this was a constant predictor, $\hat{y}_{\text{const}}(x_i) = w_1$. What might be a sensible choice of constant predictor, $\hat{y}_{\text{const}}(x_i) = w_1$? Well, we might expect w_1 to be in the middle of the datapoints, y_i. Averages formalise this notion of the "middle" of the datapoints, so we might expect w_1 to be some kind of average of the data, y_i, maybe the mean or median. What have we come out with? The mean!



And this optimal / best fitting predictor looks pretty good!

3.2 Finding the best "slope" predictor

Obviously, that was alot of work to just end up back at the mean. But we confirmed that minimizing the sum-squared error loss does something sensible. And we're going to be able to extend the sum-squared error loss idea to more complex cases. The next step is to consider a predictor that only has one parameter, w_2, but depends on the data,

$$\hat{y}_{\text{slope}}(x_i) = w_2 x_i \tag{19}$$

Again, the loss can be written as a function of the parameter, w_2.

$$\mathcal{L}_{\text{slope}}(w_2) = \sum_i \left(\hat{y}_{\text{slope}}(x_i) - y_i\right)^2 = \sum_i \left(w_2 x_i - y_i\right)^2. \tag{20}$$

Our goal is to to find the prediction-function with the lowest loss. To do that, we need to find the place where the derivative wrt w_2 is zero.

$$\frac{\partial \mathcal{L}_{\text{slope}}(w_2)}{\partial w_2} = \frac{\partial}{\partial w_2}\left[\sum_{i=1}^{N}\left(w_2 x_i - y_i\right)^2\right] \tag{21}$$

we apply the chain rule, with $u_i = w_2 x_i - y_i$,

$$\mathcal{L}_{\text{slope}}(w_2) = \sum_i u_i^2 \tag{22}$$

$$\frac{\partial \mathcal{L}_{\text{slope}}(w_2)}{\partial w_2} = \sum_{i=1}^{N}\frac{\partial u_i^2}{\partial w_2} = \sum_{i=1}^{N}\frac{\partial u_i^2}{\partial u_i}\frac{\partial u_i}{\partial w_2} \tag{23}$$

the individual derivatives are,

$$\frac{\partial u_i^2}{\partial u_i} = 2u_i \tag{24}$$

$$\frac{\partial u_i}{\partial w_2} = \frac{\partial}{\partial w_2}\left[w_2 x_i - y_i\right] = x_i \tag{25}$$

Putting everything back together,

$$\frac{\partial \mathcal{L}_{\text{slope}}(w_2)}{\partial w_2} = \sum_{i=1}^{N}\frac{\partial u_i^2}{\partial u_i}\frac{\partial u_i}{\partial w_2} = 2\sum_{i=1}^{N}(w_2 x_i - y_i)x_i \tag{26}$$

Now, we solve for the value of w_2 at which the gradient is zero,

$$0 = \frac{\partial \mathcal{L}_{\text{slope}}(w_2)}{\partial w_2} = 2\sum_{i=1}^{N}(w_2 x_i - y_i)x_i \tag{27}$$

Expanding the bracket,

$$0 = 2\sum_{i=1}^{N}\left(-x_i y_i + w_2 x_i^2\right) \tag{28}$$

Divide both sides by 2,

$$0 = \sum_{i=1}^{N}\left(w_2 x_i^2 - y_i x_i\right). \tag{29}$$

Now we sum over each term separately,

$$0 = \sum_{i=1}^{N} w_2 x_i^2 - \sum_{i=1}^{N} y_i x_i. \tag{30}$$

Notice that w_2 doesn't depend on i, so it can be brought outside the sum,

$$0 = w_2 \sum_{i=1}^{N} x_i^2 - \sum_{i=1}^{N} y_i x_i. \tag{31}$$

Now rearrange (add $\sum_{i=1}^{N} y_i x_i$ to both sides, and divide by $\sum_{i=1}^{N} x_i^2$,)

$$w_2 = \frac{\sum_{i=1}^{N} y_i x_i}{\sum_{i=1}^{N} x_i^2}. \tag{32}$$

Does this make sense? Yes! Remember that w_2 is the slope. The slope is bigger if for positive x_i, we have big positive y_i, and for negative x_i we have big negative y_i. And the numerator in this expression is big in exactly the same situation.

3.3 Straight line fitting

Above, we discussed the basic idea, but we only ever tried to fit a single parameter. What happens when we fit a full straight line,

$$\hat{y}(x_i) = w_1 + w_2 x_i \tag{33}$$

with an intercept, w_1 and slope, w_2 parameter? Conceptually, it is the same idea we've seen previously: the loss is a function of w_1 and w_2,

$$\mathcal{L}(w_1, w_2) = \sum_i \left(\hat{y}(x_i) - y_i\right)^2 = \sum_i \left(w_1 + w_2 x_i - y_i\right)^2 \tag{34}$$

We find the value of w_1, w_2 such that both derivatives are simultaneously zero,

$$0 = \frac{\partial}{\partial w_1}\mathcal{L}(w_1, w_2) \qquad\qquad 0 = \frac{\partial}{\partial w_2}\mathcal{L}(w_1, w_2) \tag{35}$$

This is harder but still possible, so we're not going to give the proof here.

Instead, see the next section for a solution in an even more general setting.

4 Multivariable regression

So far, we've always considered single-variable inputs, x, so that we can fit a curve on a 2D plot. But in the real-world, we might want to predict an outcome from many features. For instance, we might want to predict a product's sales

from many different features (length, x_1, height, x_2, weight, x_3 etc.) How can we do predictions with multiple input features? To start with, we need to talk about how we're going to represent the data. We say the data has D features (e.g. for length, height, weight, we would have $D = 3$ features) and N datapoints. In that case, we could represent all the input features as an big $N \times D$ matrix, $\mathbf{X}$. To give an example for $D = 3$.

$$\mathbf{X} = \begin{pmatrix} X_{11} & X_{12} & X_{13} \\ X_{21} & X_{22} & X_{23} \\ \vdots & \vdots & \vdots \\ X_{N1} & X_{N2} & X_{N3} \end{pmatrix} \qquad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \qquad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} \tag{36}$$

For both $\mathbf{X}$ and $\mathbf{y}$, each row corresponds to a datapoint. Thus, for each datapoint, we have an output y_i and a row-vector of input features,

We could predict all outputs simultaneously, using,

$$\hat{\mathbf{y}}(\mathbf{X}) = \mathbf{X}\mathbf{w}. \tag{37}$$

Alternatively, we could take the feature vector for a single datapoint (note this is a *row* vector),

$$\mathbf{x}_i = \begin{pmatrix} X_{i1} & X_{i2} & X_{i3} \end{pmatrix} \tag{38}$$

and make a prediction for a single datapoint,

$$\hat{y}(\mathbf{x}_i) = \mathbf{x}_i\mathbf{w} = \sum_j X_{ij} w_j. \tag{39}$$

Note that this expression wouldn't work if $\mathbf{x}_i$ and $\mathbf{w}$ were both column vectors; it only works because $\mathbf{x}_i$ is a row-vector, and $\mathbf{w}$ is a column-vector.

The loss function is, as usual,

$$\mathcal{L}(\mathbf{w}) = \sum_i \left(\hat{y}(\mathbf{x}_i) - y_i\right)^2. \tag{40}$$

The optimal weights are,

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{41}$$

I could give the derivation for this. But its quite tedious, and isn't really super-relevant for later work in neural nets (as neural nets don't have analytic solutions like this). Instead, I just want to make sure we think carefully about this expression and how the vectors/matrices in it work. Specifically, lets look a bit closer at the sizes,

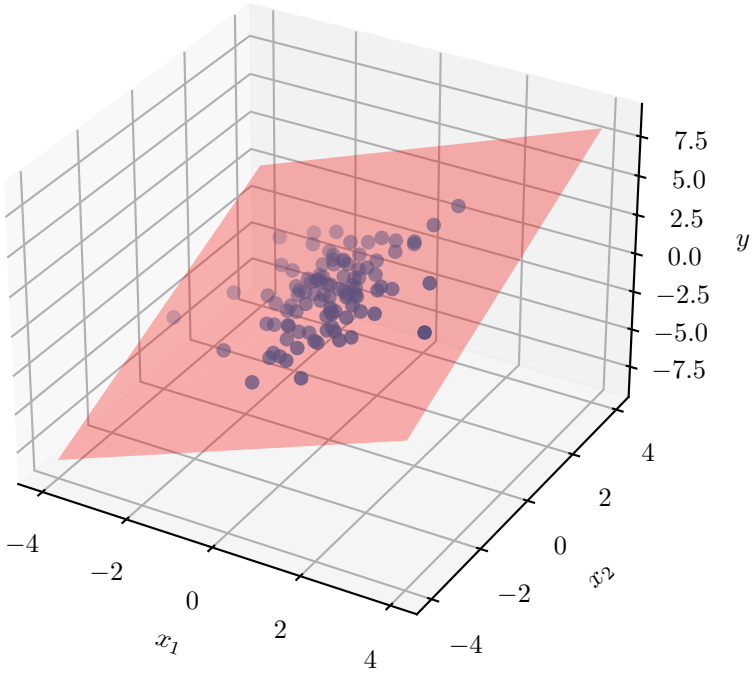
$$\underbrace{\mathbf{w}^*}_{D\times 1} = \big(\underbrace{\mathbf{X}^T}_{D\times N}\underbrace{\mathbf{X}}_{N\times D}\big)^{-1}\underbrace{\mathbf{X}^T}_{D\times N}\underbrace{\mathbf{y}}_{N\times 1} \tag{42}$$

Or, if we look at $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ and $\mathbf{X}^T\mathbf{y}$ separately,

$$\underbrace{\mathbf{w}^*}_{D\times 1} = \underbrace{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}}_{D\times D}\underbrace{\mathbf{X}^T\mathbf{y}}_{D\times 1}. \tag{43}$$

When doing this by hand, you'd usually compute $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ and $\mathbf{X}^T\mathbf{y}$ first, as the number of dimensions, D, is usually going to be smaller than the number of datapoints (see Exercises for details).

The result will look something like,

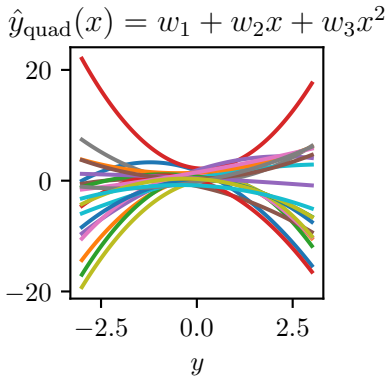


4.1 Regression with nonlinear features

What if we want to fit a model of the form,

$$\hat{y}_{\text{quad}}(x) = w_1 + w_2 x + w_3 x^2. \tag{44}$$

If we plot a bunch of these functions, with random w_1, w_2 and w_3, we get,



This is different from anything we've seen before because it is nonlinear: $\hat{y}_{\text{quad}}(x)$ is a nonlinear function of x. As such, it doesn't seem like anything we've done so far would help: we've only seen how to work with linear functions with a single or with multiple inputs. However, it turns out that we can adapt the previous formula (for multivaraible regression) for this case. The basic idea is to form multiple inputs for multivariable linear regression by applying a bunch of different functions (here, $f_1(\cdot)$, $f_2(\cdot)$ and $f_3(\cdot)$), to the actual input, x_i,

$$\mathbf{x}(x) = \begin{pmatrix} f_1(x) & f_2(x) & f_3(x) \end{pmatrix} \tag{45}$$

So the big matrix $\mathbf{X}$ becomes,

$$\mathbf{X} = \begin{pmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) \\ \vdots & \vdots & \vdots \\ f_1(x_N) & f_2(x_N) & f_3(x_N) \end{pmatrix} \tag{46}$$

Then, our prediction is a linear combination of the functions, $f_1(x)$, $f_2(x)$ and $f_3(x)$,

$$\hat{y}(x) = \mathbf{x}(x)\mathbf{w} = w_1 f_1(x) + w_2 f_2(x) + w_3 f_3(x). \tag{47}$$

To get back to our quadratic model (Eq. 44), we choose,

$$f_1(x) = 1 \tag{48}$$
$$f_2(x) = x \tag{49}$$
$$f_3(x) = x^2 \tag{50}$$

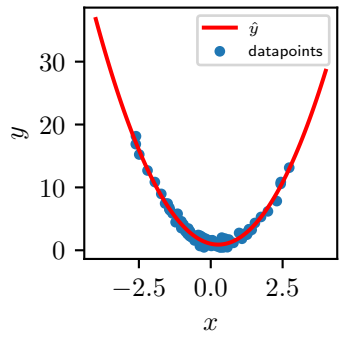
These choices for the functions give a feature vector (Eq. 45) of,

$$\mathbf{x}(x) = \begin{pmatrix} 1 & x & x^2 \end{pmatrix} \tag{51}$$

That gives a big matrix of the form,

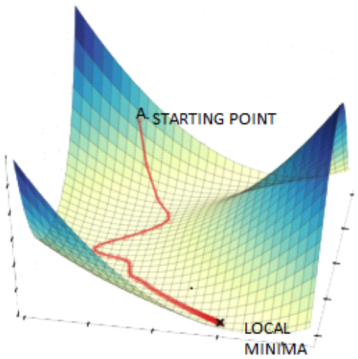
$$\mathbf{X} = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{pmatrix} \tag{52}$$

To get the optimal weights, we can plug this $\mathbf{X}$ and the usual outputs, $\mathbf{y}$ into Eq. (41). The resulting fitted line looks something like,



5 Gradient descent

In the last part, we looked at analytic solutions. However, analytic solutions are available in very few settings. Instead, the approach taken in NNs is "gradient descent": computing the gradient of the loss wrt the parameters, then changing the parameters in the direction of minus the gradient, to reduce the loss. Inuitively, we're following the gradient downhill, until we find the minimum.



Mathematically, the updates to the weight are,

$$\Delta w_i = -\eta \frac{\partial \mathcal{L}}{\partial w_i} \tag{53}$$

where η is the learning rate, which is typically small (e.g. we might use $\eta = 0.1$). Note that the minus-sign is there, because the gradients point uphill, while we want to go downhill.

If you repeat this process, you should eventually end up at a place where the gradients are zero, at which point gradient descent will stop moving (remember that in the previous part, we were solving for the location where the gradients were zero).

6 Gradient descent for linear regression

So what do the actual gradient descent updates look like for linear regression? Remember that in linear regression, we are fitting a straight line of the form,

$$\hat{y}(x_i) = w_1 + w_2 x_i \tag{54}$$

with an intercept, w_1 and slope, w_2 parameter. Conceptually, it is the same idea we've seen previously: the loss is a function of w_1 and w_2,

$$\mathcal{L}(w_1, w_2) = \sum_i \left(\hat{y}(x_i) - y_i\right)^2 \tag{55}$$

Now, we differentiate the loss wrt w_1,

$$\frac{\partial}{\partial w_1}\mathcal{L}(w_1, w_2) = \sum_i \left(\hat{y}(\mathbf{x}_i) - y_i\right)^2. \tag{56}$$

Now we use the chain rule, with $u_i = \hat{y}(x_i) - y_i = w_1 + w_2 x_i - y_i$ (the final equality comes from substituting Eq. 54). With this choice of u_i,

$$\mathcal{L}(\mathbf{w}) = \sum_i u_i^2. \tag{57}$$

So,

$$\frac{\partial}{\partial w_1}\mathcal{L}(w_1, w_2) = \sum_i \frac{\partial u_i^2}{\partial w_1} = \sum_i \frac{\partial u_i^2}{\partial u_i}\frac{\partial u_i}{\partial w_1} \tag{58}$$

The individual derivatives are,

$$\frac{\partial u_i^2}{\partial u} = 2u_i = 2\left(\hat{y}(x_i) - y_i\right) \tag{59}$$

$$\frac{\partial u_i}{\partial w_1} = \frac{\partial}{\partial w_1}\left(w_1 + w_2 x_i - y_i\right) = 1 \tag{60}$$

Putting everything together,

$$\frac{\partial}{\partial w_1}\mathcal{L}(w_1, w_2) = \sum_i \frac{\partial u_i^2}{\partial u_i}\frac{\partial u_i}{\partial w_1} = 2\sum_i \left(\hat{y}(x_i) - y_i\right) \tag{61}$$

and finally our update is,

$$\Delta w_1 = -\eta \frac{\partial}{\partial w_1} \mathcal{L}(w_1, w_2) \tag{62}$$

$$= 2\eta \sum_i (y_i - \hat{y}(x_i)) \tag{63}$$

These updates make sense: if the real data, y_i, is generally higher than our prediction, $\hat{y}(x_i)$, then we should increase the bias, which increases all our predictions.

Now, we apply the same idea to w_2. We use the same choice of u_i, so we can start at,

$$\frac{\partial}{\partial w_2} \mathcal{L}(w_1, w_2) = \sum_i \frac{\partial u_i^2}{\partial w_2} = \sum_i \frac{\partial u_i^2}{\partial u_i} \frac{\partial u_i}{\partial w_2} \tag{64}$$

The individual derivatives are,

$$\frac{\partial u_i^2}{\partial u_i} = 2u_i = 2\left(\hat{y}(x_i) - y_i\right) \tag{65}$$

$$\frac{\partial u_i}{\partial w_2} = \frac{\partial}{\partial w_2}\left(w_1 + w_2 x_i - y_i\right) = x_i \tag{66}$$

Putting everything together,

$$\frac{\partial}{\partial w_2} \mathcal{L}(w_1, w_2) = \sum_i \frac{\partial u_i^2}{\partial u_i} \frac{\partial u_i}{\partial w_2} = 2 \sum_i x_i \left(\hat{y}(x_i) - y_i\right) \tag{67}$$

and finally our update is,

$$\Delta w_2 = -\eta \frac{\partial}{\partial w_2} \mathcal{L}(w_1, w_2) \tag{68}$$

$$= 2\eta \sum_i x_i \left(y_i - \hat{y}(x_i)\right) \tag{69}$$

To interpret these updates, consider a setting where again, the data, y_i, is bigger than our prediction, $\hat{y}(x_i)$. If x_i is positive, then we can reduce the error by *increasing* w_2 (as that will increase the size of $w_2 x_i$ in the predcition Eq. 54); and that's exactly what these updates do. In contrast, if x_i is negative, then we can reduce the error by *decreasing* w_2, as that will make $w_2 x_i$ more positive.

7 Gradient descent for multivariate linear regression

Critically, gradient descent is much easier to generalise to more complex cases than analytic solutions, like those in the previous part. As such, we can now

derive the updates for multivariate linear regression (note that while there was an analytic solution in this case, we didn't derive it because its painful and not particularly instructive). From last time, the prediction in multivariate linear regression was a weighted sum of features,

$$\hat{y}(\mathbf{x}_i) = \mathbf{x}_i \mathbf{w} = \sum_\lambda X_{i\lambda} w_\lambda \tag{70}$$

We use the usual squared-error loss,

$$\mathcal{L}(\mathbf{w}) = \sum_i \left(\hat{y}(\mathbf{x}_i) - y_i\right)^2 = \sum_i \left(\sum_\lambda X_{i\lambda} w_\lambda - y_i\right)^2 \tag{71}$$

Note that I have used i to index datapoints, and λ to index features. Now, we differentiate wrt an individual weight,

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_\nu} = \sum_i \frac{\partial}{\partial w_\nu} \left(\sum_\lambda X_{i\lambda} w_\lambda - y_i\right)^2. \tag{72}$$

As before, we apply the chain rule, with $u_i = \hat{y}(x_i) - y_i = \sum_\lambda X_{i\lambda} w_\lambda - y_i$. With this choice of u_i,

$$\mathcal{L}(\mathbf{w}) = \sum_i u_i^2. \tag{73}$$

So,

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_\nu} = \sum_i \frac{\partial u_i^2}{\partial w_\nu} = \sum_i \frac{\partial u_i^2}{\partial u_i} \frac{\partial u_i}{\partial w_\nu} \tag{74}$$

The individual derivatives are,

$$\frac{\partial u_i^2}{\partial u_i} = 2u_i = 2\left(\hat{y}(x_i) - y_i\right) \tag{75}$$

$$\frac{\partial u_i}{\partial w_\nu} = \frac{\partial}{\partial w_\nu} \left(\sum_\lambda X_{i\lambda} w_\lambda - y_i\right) \tag{76}$$

But this time, we need to think a bit more about the $\frac{\partial u_i}{\partial w_\nu}$. In particular, the gradient of $X_{i\lambda}$ and y_i wrt w_ν is zero, as $X_{i\lambda}$ and y_i are data, and data is fixed!

$$\frac{\partial u_i}{\partial w_\nu} = \sum_\lambda X_{i\lambda} \frac{\partial w_\lambda}{\partial w_\nu} \tag{77}$$

The gradient of w_λ wrt w_ν is 1 when $\lambda = \nu$, and zero otherwise. That means its the Kronecker delta (see Part 1, mathematical prerequisites!)

$$\frac{\partial u_i}{\partial w_\nu} = \sum_\lambda X_{i\lambda} \delta_{\lambda\nu} \tag{78}$$

As discussed in the prerequistes, the Kronecker delta picks out one component of the sum, where $\lambda = \nu$,

$$\frac{\partial u_i}{\partial w_\nu} = X_{i\nu}. \tag{79}$$

Putting everything back together,

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_\nu} = \sum_i \frac{\partial u_i^2}{\partial u_i} \frac{\partial u_i}{\partial w_\nu} = 2 \sum_i \left(\hat{y}(x_i) - y_i\right) X_{i\nu}. \tag{80}$$

So our gradient descent updates are,

$$\Delta w_\nu = -\eta \frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_\nu} = 2\eta \sum_i \left(y_i - \hat{y}(x_i)\right) X_{i\nu}. \tag{81}$$

These updates again make sense. If $X_{i\nu}$ is positive, and the error, $y_i - \hat{y}(x_i)$, then increasing w_ν will reduce the error.

8 Exercises

All these exercises are designed to be possible by hand (with a calculator). We'd advise doing them that way, as that's how the exam will be set!

The following three exercises are based on this dataset.

x	y
-2	-6.2
-1	-2.6
0	0.5
1	2.7
2	5.7

Exercise 1. *Which of the following predictions has the lowest loss (specifically, the sum-squared distance),*

$$\hat{y}_1(x) = x \tag{82}$$
$$\hat{y}_2(x) = 2x \tag{83}$$
$$\hat{y}_3(x) = 3x \tag{84}$$

Exercise 2. *Fit a straight line of the form,*

$$\hat{y}(x) = w_1 + w_2 x \tag{85}$$

using the formulae in Sec. 4. Specifically, use two features:

$$f_1(x) = 1 \qquad\qquad f_2(x) = x \tag{86}$$

Exercise 3. *Fit a nonlinear prediction of the form,*

$$\hat{y}(x) = w_1 x + w_2 x^2. \tag{87}$$

Again, use the formulae in Sec. 4. What f_1 and f_2 do you need to use?

Exercise 4. *The absolute value is defined as,*

$$|x| = \begin{cases} x & \text{if } x > 0 \\ -x & \text{otherwise.} \end{cases} \tag{88}$$

Sketch a plot of $y = |x|$, and compute,

$$\frac{\partial |x|}{\partial x} \tag{89}$$

Compute the gradient separately for $0 < x$ and $x < 0$. Don't worry about $x = 0$, where the gradient is undefined. And consider using the sign function,

$$\text{sign}(x) = \begin{cases} 1 & \text{if } 0 < x \\ 0 & \text{if } 0 = x \\ -1 & \text{if } x < 0 \end{cases} \tag{90}$$

Exercise 5. *We have N datapoints, x_i. We seek to find a location, b, which minimizes the sum of distances (not squared distances) between each datapoint, x_i and b. Specifically, our loss function is,*

$$\mathcal{L}(b) = \sum_{i=1}^{N} |x_i - b|, \tag{91}$$

Part 1: Compute the gradient of this following loss function, wrt b. Write your answer in terms of the sign *function (see answer to the prevous question for details).*

Part 2: Rewrite the gradient in words in terms of the number of datapoints above and below b.

Part 3: Give an interpretation of the optimal value of b (i.e. the value of b for which the gradient is zero). This is going to be easiest when there is an odd number of datapoints.

9 Answers

Answer 1. *Remember, the goal was to find which of the following prediction functions,*

$$\hat{y}_1(x) = x \tag{92}$$

$$\hat{y}_2(x) = 2x \tag{93}$$

$$\hat{y}_3(x) = 3x \tag{94}$$

had the lowest loss. To do that, we need to evaluate the loss for $\hat{y}_1$, $\hat{y}_2$ and $\hat{y}_3$. The predictions for $\hat{y}_1$ are,

$$\hat{y}_1(x_1) = x_1 = -2 \tag{95}$$
$$\hat{y}_1(x_2) = x_2 = -1 \tag{96}$$
$$\hat{y}_1(x_3) = x_3 = 0 \tag{97}$$
$$\hat{y}_1(x_4) = x_4 = 1 \tag{98}$$
$$\hat{y}_1(x_5) = x_5 = 2 \tag{99}$$

Thus, the loss for $\hat{y}_1$ is

$$\mathcal{L}_1 = \sum_{i=1}^{5}(y_i - \hat{y}_1(x_i))^2 \tag{100}$$
$$= ((-6.2) - (-2))^2 + ((-2.6) - (-1))^2 + (0.5 - 0)^2 + (2.7 - 1)^2 + (5.7 - 2)^2$$
$$= (-4.2)^2 + (-1.6)^2 + (0.5)^2 + (1.7)^2 + (3.7)^2 \tag{101}$$
$$= 37.03 \tag{102}$$

The predictions for $\hat{y}_2$ are,

$$\hat{y}_2(x_1) = 2x_1 = 2 \times (-2) = -4 \tag{103}$$
$$\hat{y}_2(x_2) = 2x_2 = 2 \times (-1) = -2 \tag{104}$$
$$\hat{y}_2(x_3) = 2x_3 = 2 \times 0 = 0 \tag{105}$$
$$\hat{y}_2(x_4) = 2x_4 = 2 \times 1 = 2 \tag{106}$$
$$\hat{y}_2(x_5) = 2x_5 = 2 \times 2 = 4 \tag{107}$$

Thus, the loss for $\hat{y}_2$ is

$$\mathcal{L}_2 = \sum_{i=1}^{5}(y_i - \hat{y}_2(x_i))^2 \tag{108}$$
$$= ((-6.2) - (-4))^2 + ((-2.6) - (-2))^2 + (0.5 - 0)^2 + (2.7 - 2)^2 + (5.7 - 4)^2$$
$$= (-2.2)^2 + (-0.6)^2 + (0.5)^2 + (0.7)^2 + (1.7)^2 \tag{109}$$
$$= 8.83 \tag{110}$$

The predictions for $\hat{y}_3$ are,

$$\hat{y}_3(x_1) = 3x_1 = 3 \times (-2) = -6 \tag{111}$$
$$\hat{y}_3(x_2) = 3x_2 = 3 \times (-1) = -3 \tag{112}$$
$$\hat{y}_3(x_3) = 3x_3 = 3 \times 0 = 0 \tag{113}$$
$$\hat{y}_3(x_4) = 3x_4 = 3 \times 1 = 3 \tag{114}$$
$$\hat{y}_3(x_5) = 3x_5 = 3 \times 2 = 6 \tag{115}$$

Thus, the loss for $\hat{y}_3$ is

$$\mathcal{L}_3 = \sum_{i=1}^{5}(y_i - \hat{y}_3(x_i))^2 \tag{116}$$

$$= ((-6.2) - (-6))^2 + ((-2.6) - (-3))^2 + (0.5 - 0)^2 + (2.7 - 3)^2 + (5.7 - 6)^2$$

$$= (-0.2)^2 + (0.4)^2 + (0.5)^2 + (-0.3)^2 + (-0.3)^2 \tag{117}$$

$$= 0.63 \tag{118}$$

Overall, the three losses are:

$$\mathcal{L}_1 = 37.03 \tag{119}$$

$$\mathcal{L}_2 = 8.83 \tag{120}$$

$$\mathcal{L}_3 = 0.63 \tag{121}$$

So the best (lowest loss) prediction is $\hat{y}_3$. This makes sense if we e.g. sketch a plot of the data itself: the slope is about 3.

Answer 2. *Remember that when we're doing regression with multiple features, our predictor is Eq. (47),*

$$\hat{y}(x) = \mathbf{x}(x)\mathbf{w} = w_1 f_1(x) + w_2 f_2(x). \tag{122}$$

Substituting the $f_1(x)$ and $f_2(x)$ given in the question, we get back Eq. (85), as we'd hope,

$$\hat{y}(x) = \mathbf{x}(x)\mathbf{w} = w_1 + w_2 x. \tag{123}$$

Now, we can compute the optimal weights using,

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{124}$$

To do that, we first need to construct the big feature matrix $\mathbf{X}$ and the big result vector, $\mathbf{y}$. The result vector is just a vector of the y's

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} -6.2 \\ -2.6 \\ 0.5 \\ 2.7 \\ 5.7 \end{pmatrix} \tag{125}$$

We get the feature matrix, $\mathbf{X}$, by applying $f_1(\cdot)$ and $f_2(\cdot)$ to the x's,

$$\mathbf{X} = \begin{pmatrix} f_1(x_1) & f_2(x_1) \\ f_1(x_2) & f_2(x_2) \\ f_1(x_3) & f_2(x_3) \\ f_1(x_4) & f_2(x_4) \\ f_1(x_5) & f_2(x_5) \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} \tag{126}$$

Now, in principle all we have to do is to substitute this form for $\mathbf{X}$ *and* $\mathbf{y}$ *into the form for the optimal weights,*

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{127}$$

However, it turns out that we can simplify the computation a bit if we think carefully about the order of operations. And it turns out this thinking is helpful if we're doing the computation by hand (e.g. in an exam) or at a large scale on a computer. Specifically, remember that

$$\underbrace{\mathbf{w}^*}_{D\times 1} = \underbrace{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}}_{D\times D}\underbrace{\mathbf{X}^T\mathbf{y}}_{D\times 1}. \tag{128}$$

and that the number of features ($D = 2$ *in our case) is typically smaller than the number of datapoints (*$N = 5$ *in our case). Thus, it is easiest to first compute* $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ *as that's just a* 2×2 *matrix, and to compute* $\mathbf{X}^T\mathbf{y}$ *as that's just a length 2 vector. Thus, we start by computing the two-by-two matrix,* $\mathbf{X}^T\mathbf{X}$*,*

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 5 & 0 \\ 0 & 10 \end{pmatrix} \tag{129}$$

For instance, the top-left element is given by,

$$5 = 1\times 1 + 1\times 1 + 1\times 1 + 1\times 1 + 1\times 1 = 1 + 1 + 1 + 1 + 1 \tag{130}$$

For instance, and top-right element is given by,

$$0 = 1\times(-2) + 1\times(-1) + 1\times 0 + 1\times 1 + 1\times 2 = (-2) + (-1) + 0 + 1 + 2 \tag{131}$$

We can then compute $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ *by using the usual formula for the* 2×2 *matrix inverse,*

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{132}$$

Thus,

$$\left(\mathbf{X}\mathbf{X}^T\right)^{-1} = \begin{pmatrix} 5 & 0 \\ 0 & 10 \end{pmatrix}^{-1} \tag{133}$$

$$= \frac{1}{5\times 10 - 0\times 0}\begin{pmatrix} 10 & 0 \\ 0 & 5 \end{pmatrix} \tag{134}$$

$$= \frac{1}{50}\begin{pmatrix} 10 & 0 \\ 0 & 5 \end{pmatrix} \tag{135}$$

$$= \begin{pmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{10} \end{pmatrix} \tag{136}$$

Next, we compute $\mathbf{X}^T\mathbf{y}$*, which is a matrix-vector product,*

$$\mathbf{X}^T\mathbf{y} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} -6.2 \\ -2.6 \\ 0.5 \\ 2.7 \\ 5.7 \end{pmatrix} \tag{137}$$

$$= \begin{pmatrix} 1\times(-6.2) + 1\times(-2.6) + 1\times0.5 + 1\times2.7 + 1\times5.7 \\ (-2)\times(-6.2) + (-1)\times(-2.6) + 0\times0.5 + 1\times2.7 + 2\times5.7 \end{pmatrix} \tag{138}$$

$$= \begin{pmatrix} (-6.2) + (-2.6) + 0.5 + 2.7 + 5.7 \\ 12.4 + 2.6 + 0 + 2.7 + 11.4 \end{pmatrix} \tag{139}$$

$$= \begin{pmatrix} 0.1 \\ 29.1 \end{pmatrix} \tag{140}$$

Now, we can substitute our values for $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ *and* $\mathbf{X}^T\mathbf{y}$ *into the form for the optimal weights,*

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{141}$$

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{5} & 0 \\ 0 & \frac{1}{10} \end{pmatrix} \begin{pmatrix} 0.1 \\ 29.1 \end{pmatrix} = \begin{pmatrix} 0.02 \\ 2.91 \end{pmatrix} \tag{142}$$

So $w_1 = 0.02$ *and* $w_2 = 2.91$*, which is exactly the same answer using the alternative strategy in the previous question!*

Answer 3. *Remember that when we're doing regression with multiple features, our predictor is (Eq. 47),*

$$\hat{y}(x) = \mathbf{x}(x)\mathbf{w} = w_1 f_1(x) + w_2 f_2(x). \tag{143}$$

Looking back at Eq. (87)*,*

$$\hat{y}(x) = w_1 x + w_2 x^2, \tag{144}$$

we need to choose,

$$f_1(x) = x \qquad\qquad f_2(x) = x^2. \tag{145}$$

Now, we can compute the optimal weights using,

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{146}$$

To do that, we first need to construct the big feature matrix $\mathbf{X}$ *and the big result vector,* $\mathbf{y}$*. The result vector is just a vector of the* y*'s*

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix} = \begin{pmatrix} -6.2 \\ -2.6 \\ 0.5 \\ 2.7 \\ 5.7 \end{pmatrix} \tag{147}$$

We get the feature matrix, $\mathbf{X}$, by applying $f_1(\cdot)$ and $f_2(\cdot)$ to the x_i's,

$$\mathbf{X} = \begin{pmatrix} f_1(x_1) & f_2(x_1) \\ f_1(x_2) & f_2(x_2) \\ f_1(x_3) & f_2(x_3) \\ f_1(x_4) & f_2(x_4) \\ f_1(x_5) & f_2(x_5) \end{pmatrix} = \begin{pmatrix} x_1 & x_1^2 \\ x_2 & x_2^2 \\ x_3 & x_3^2 \\ x_4 & x_4^2 \\ x_5 & x_5^2 \end{pmatrix} = \begin{pmatrix} -2 & (-2)^2 \\ -1 & (-1)^2 \\ 0 & 0^2 \\ 1 & 1^2 \\ 2 & 2^2 \end{pmatrix} = \begin{pmatrix} -2 & 4 \\ -1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 2 & 4 \end{pmatrix} \tag{148}$$

We start by computing the two-by-two matrix, $\mathbf{X}^T\mathbf{X}$,

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} -2 & 4 \\ -1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 10 & 0 \\ 0 & 34 \end{pmatrix} \tag{149}$$

We can then compute $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ by using the usual formula for the 2×2 matrix inverse,

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \tag{150}$$

Thus,

$$\left(\mathbf{X}\mathbf{X}^T\right)^{-1} = \begin{pmatrix} 10 & 0 \\ 0 & 34 \end{pmatrix}^{-1} \tag{151}$$

$$= \frac{1}{10 \times 34 - 0 \times 0} \begin{pmatrix} 34 & 0 \\ 0 & 10 \end{pmatrix} \tag{152}$$

$$= \frac{1}{340} \begin{pmatrix} 34 & 0 \\ 0 & 10 \end{pmatrix} \tag{153}$$

$$= \begin{pmatrix} \frac{1}{10} & 0 \\ 0 & \frac{1}{34} \end{pmatrix} \tag{154}$$

Next, we compute $\mathbf{X}^T\mathbf{y}$, which is a matrix-vector product,

$$\mathbf{X}^T\mathbf{y} = \begin{pmatrix} -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} -6.2 \\ -2.6 \\ 0.5 \\ 2.7 \\ 5.7 \end{pmatrix} \tag{155}$$

$$= \begin{pmatrix} (-2)\times(-6.2) + (-1)\times(-2.6) + 0\times0.5 + 1\times2.7 + 2\times5.7 \\ 4\times(-6.2) + 1\times(-2.6) + 0\times0.5 + 1\times2.7 + 4\times5.7 \end{pmatrix} \tag{156}$$

$$= \begin{pmatrix} 29.1 \\ -1.9 \end{pmatrix} \tag{157}$$

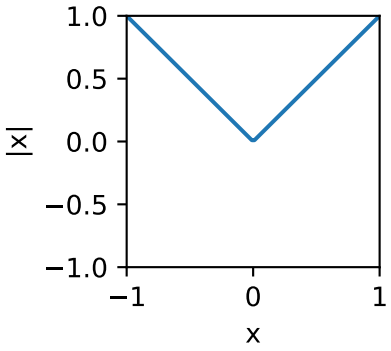
Now, we can substitute our values for $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$ *and* $\mathbf{X}^T\mathbf{y}$ *into the form for the optimal weights,*

$$\mathbf{w}^* = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{158}$$

$$\begin{pmatrix} w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} \frac{1}{10} & 0 \\ 0 & \frac{1}{34} \end{pmatrix}\begin{pmatrix} 29.1 \\ -1.9 \end{pmatrix} = \begin{pmatrix} 2.91 \\ -0.0559 \end{pmatrix} \tag{159}$$

So $w_1 = 2.91$*. Remember that this is now the linear weight (i.e. the term that multiplies* x*). Surprisingly, it has the same value as we saw previously (this doesn't happen in general, and only occurs in this case because the off-diagonals of* $\mathbf{X}^T\mathbf{X}$ *are zero). Additionally, the quadratic weight,* $w_2 = 0.0559$*, is small. That makes sense because the underlying data looks more like a straight line than a quadratic.*

Answer 4. *The sketch plot looks like:*



We can differentiate each "case" separately.

$$\frac{\partial|x|}{\partial x} = \begin{cases} \frac{\partial x}{\partial x} & \text{if } 0 < x \\ \frac{\partial -x}{\partial x} & \text{if } x < 0 \end{cases} \tag{160}$$

$$= \begin{cases} 1 & \text{if } 0 < x \\ -1 & \text{if } x < 0. \end{cases} \tag{161}$$

$$= \text{sign}(x) \tag{162}$$

(If we play a little fast-and-loose with the zero.)

FYI, there is no well-defined gradient at the "kink" (i.e. $x = 0$*), but that isn't really relevant in practice, so we aren't going to worry about it here.*

Answer 5. *Part 1 (gradient of* $\mathcal{L}(b)$ *in terms of* sign*),*

$$\frac{\partial\mathcal{L}(b)}{\partial b} = \sum_{i=1}^{N}\frac{\partial|x_i - b|}{\partial b} \tag{163}$$

use the chain rule with $u_i = x_i - b$,

$$\frac{\partial \mathcal{L}(b)}{\partial b} = \sum_{i=1}^{N} \frac{\partial |u_i|}{\partial u_i} \frac{\partial u_i}{\partial b} \tag{164}$$

$$\frac{\partial \mathcal{L}(b)}{\partial b} = \sum_{i=1}^{N} \text{sign}(x_i - b)(-1) \tag{165}$$

$$\frac{\partial \mathcal{L}(b)}{\partial b} = -\sum_{i=1}^{N} \text{sign}(x_i - b). \tag{166}$$

Part 2 (interpretation in terms of the number of datapoints above and below a threshold):

$$\text{sign}(x_i - b) = \begin{cases} 1 & \text{if } b < x_i \\ -1 & \text{if } x_i < b \end{cases} \tag{167}$$

Thus, the gradient of the loss in effect counts the number of datapoints above and below b,

$$\frac{\partial \mathcal{L}(b)}{\partial b} = (\textit{Number of datapoints below } b) - (\textit{Number of datapoints above } b). \tag{168}$$

Part 3: the optimal value of b *is at,*

$$0 = \frac{\partial \mathcal{L}(b)}{\partial b} = (\textit{Number of datapoints below } b) - (\textit{Number of datapoints above } b). \tag{169}$$

i.e. this is a value of b *for which,*

$$(\textit{Number of datapoints below } b) = (\textit{Number of datapoints above } b) \tag{170}$$

And this value for b *is the median.*

FYI: There are some subtleties here about how exactly how we define the median. If there an odd number of datapoints, then the median really is the "middle" datapoint. If there an even number of datapoints, then there isn't a single "middle" datapoint: there's two. Most computer implementations of the median would return halfway between the two middle datapoints. But we have $0 = \frac{\partial \mathcal{L}(b)}{\partial b}$ *for any* b *between those two datapoints.*