Least Squares Regression

September 12, 2023

Consider a set of points (x_i, y_i) for $i = 1, 2, 3, \ldots, N$. The problem of least squares regression is one that will help us try to discern the relationship between $\{x_i, i = 1, 2, 3, \ldots, N\}$ and $\{y_i, i = 1, 2, 3, \ldots, N\}$. At this point, let's suppose that x_i and y_i are just numbers, but we'll need to generalize to the case of vectors very soon.

Suppose I have some function $f(x, a)$. I want to pick a function so that the qualitative behavior of f changes significantly with different values of a[1]. The least squares problem is to find the value of a that minimizes

$$S = \sum_{i=1}^{N} (f(x_i, a) - y_i)^2. \tag{1}$$

Let's suppose that $f(x_i, a) = a$. Then

$$S = \sum_{i=1}^{N} (a - y_i)^2. \tag{2}$$

To minimize this with respect to a, we take

$$0 = \frac{\partial S}{\partial a} = \sum_{i=1}^{N} (a - y_i). \tag{3}$$

Rearranging terms,

$$Na = \sum_{i=1}^{N} y_i. \tag{4}$$

or rather

$$a = \bar{y} \tag{5}$$

This is one of the special properties of the arithmetic mean; it minimizes least squared distance.

1 Linear Least Squares

Let's do another example where, now, we'll treat $x_i \in \mathbb{R}^n$ and $y_i \in \mathbb{R}$ as vectors and we will use the fit function

$$f(x) = \beta \cdot x \tag{6}$$

where β is a vector. You might wonder why there is no affine term, but, as we will see, it is straightforward to extend to that case from this one.

Now, with

$$X^{\mathrm{T}} = \begin{pmatrix} x_1 & x_2 & x_3 & \ldots & x_N \end{pmatrix} \tag{7}$$

and y is a vector. So we have N data points who each have associated with their "value" a single number but multiple inputs.

[1] a may be a vector, for example.

As an aside, this perscription lends itself to a few different types of "machine learning tasks". For example, if we consider the number drawing data, there are 28×28 pixels, which each have an associated number (brightness), but we are interested in **classifying** the digits as being part of a finite set, $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. So, with this example in mind, the y_i variable would be the actual digit represented by a series of pixels, and the pixes associated with the i'th image are "flattened" into X_{ji} so that, explicitly, X_{ji} would be the number representing the color of the j'th pixel of the i'th image.

Now, we repeat our above calculation slightly more generally. As a note, S, is typically called a **loss function**.

$$\begin{aligned} S &= \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{n} \beta_j (x_i)_j \right)^2 = \sum_{i=1}^{N} (\beta \cdot x_i - y_i)^2 \\ &= \sum_{i=1}^{N} \left[(\beta \cdot x_i)^2 - 2\beta \cdot x_i y_i + y_i^2 \right] \\ &= \beta X X^{\mathrm{T}} \beta^{\mathrm{T}} - 2\beta X y + y \cdot y = \|\beta X - y\|^2 \end{aligned} \tag{8}$$

All we have to do at this point is to take derivatives. Explicitly,

$$\begin{aligned} \frac{\partial S}{\partial \beta_{j_0}} &= 2 \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{n} \beta_j (x_i)_j \right) \times (x_i)_{j_0} \\ &= \sum_{i=1}^{N} (y - \beta X)_i X^{\mathrm{T}}_{ij_0}. \end{aligned} \tag{9}$$

More consicely,

$$\nabla_\beta S = (y - \beta X) X^{\mathrm{T}} = y X^{\mathrm{T}} - \beta X X^{\mathrm{T}} \tag{10}$$

By $\nabla_\beta S$, i just mean the vector

$$\nabla_\beta S = \begin{pmatrix} \frac{\partial S}{\partial \beta_1} \\ \frac{\partial S}{\partial \beta_2} \\ \vdots \\ \frac{\partial S}{\partial \beta_n} \end{pmatrix}. \tag{11}$$

Anyway, if $\nabla_\beta S = 0$, that means

$$y X^{\mathrm{T}} = \beta X X^{\mathrm{T}} \tag{12}$$

or

$$\beta = y X^{\mathrm{T}} (X X^{\mathrm{T}})^{-1} \tag{13}$$

where the -1 just indicates matrix inversion.

So evidently, if we just format our data in the right way, we can just **solve** the linear least squares problem on paper. Typically, when doing this kind of fit, an analytical solution like this is not available, so we would have to resort to some minimum–finding technique (which we will see some of later).

1.1 About that affine term

So in the previous discussion, I basically only considered functions of the form

$$f(x_1, x_2) = \beta_1 x_1 + \beta_2 x_2. \tag{14}$$

You may be wondering why I've done this; after all, it would be natural to consider instead functions of the form

$$g(x_1, x_2) = \beta_1 x_1 + \beta_2 x_2 + \beta_0. \tag{15}$$

There is a simple (and standard) way of fixing this problem without having to do any extra formal stuff. Suppose our data has (x_1, x_2) as input, and y as output. Just define a new input for every data point that

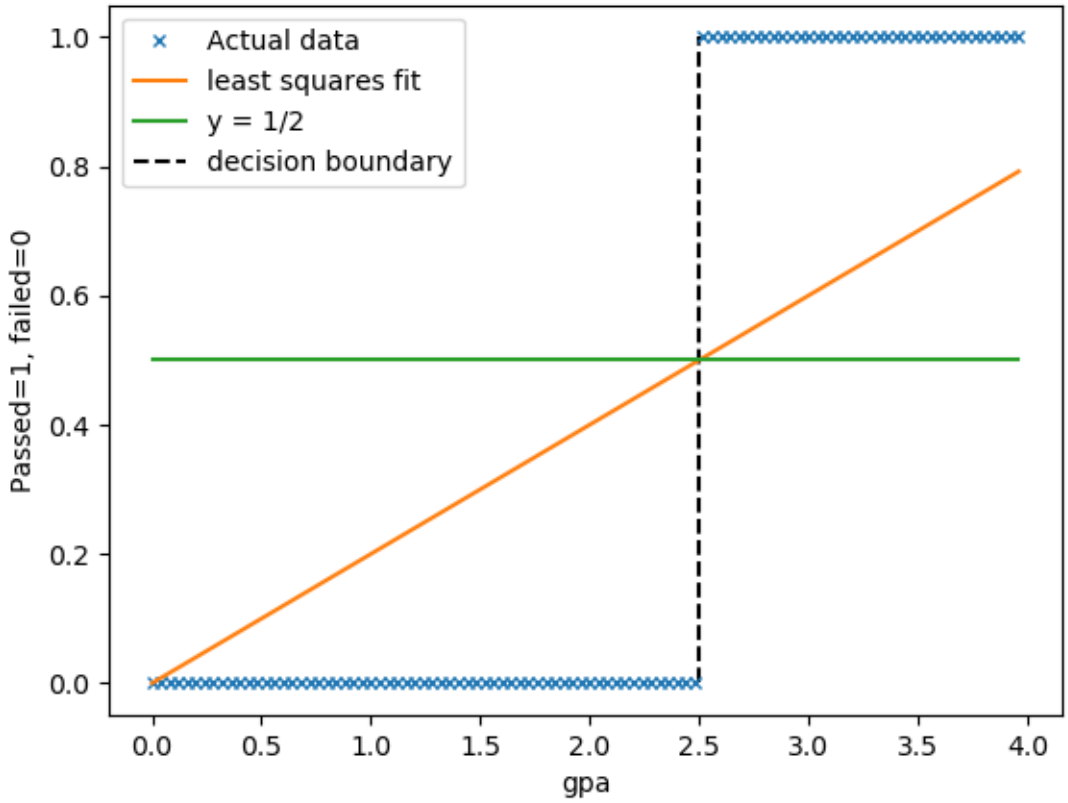

Figure 1:

takes on the same value everywhere, so that our **new** data is like $(x_1, x_2, x_3 = 1)$ as input and y as output. Then the function

$$f(x_1, x_2, x_3) = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 \tag{16}$$

evaluated at

$$f(x_1, x_2, 1) = \beta_1 x_1 + \beta_2 x_2 + \beta_3 \tag{17}$$

is precisely as good as $g(x)$ above. It is common practice to add "new" data which is just some function of old data in order to improve the fit of some model.

1.2 Data types and indicator variables

We have another issue! For the sake of simplicity, I have invented some data which I have plotted in Fig 1. I polled 100 fake students and took data of the form

GPA	passed?
0.0	no
⋮	
2.4	no
⋮	
2.8	yes

There's a problem though; we need many more bits to store the words "yes" and "no" than are contained in the words themselves, and the techniques we're using are relying on our ability to do arithmetic with the data, so we need to turn the words into numbers somehow. The most obvious way to do this is just to say that failure should get assigned 0 instead of 'no' and success should be assigned 1 instead of 'yes'. After performing this operation, our data would look like what's plotted in Fig 1.

Now we arrive at the actual issue: there are only two possible outcomes, namely, passing and failing, but our least squares fit can (and does) take on values in the range $[0, 1]$, so we need to find some way to turn a number between zero and one into a 0 or a 1. In machine learning, this procedure is what defines **classification**. In this case, the resolution is simple enough. We predict that a student would fail if our least squares fit is closer to 0 than it is to 1 and similarly we say that students should be predicted to pass if our least square value is greater than $\frac{1}{2}$. When you are asked to categorize data into a finite number of buckets, it is always necessary to do something like this.

In this case, we perfectly describe the phenomenon of passing or failing, because the relationship between success and gpa is very simple, but this principle remains useful even when relationships are not so clear cut. It is possible to cut the domain of some data in such a manner that the data is partitioned into regions where points in a given region are all classified the same. This boundary is called **The decision boundary** and I've plotted it along with the ficticious data and the least squares fit. In classification problems, it is often this decision boundary which is the most desired for visualization.

Our task will be to impliment linear least squares and use some of the techniques above on a particularly modified version of our number data.