Chapter 2

Least Squares Regression

We have so far predicted salaries without taking into account any information about the person whose salary is being predicted. Of course, there are lots of things which can influence someone's salary: how much experience they have, their college GPA, what city they live in, and so forth. We call these things **features**. In this chapter, we'll see how to apply the framework of empirical risk minimization to the problem of predicting based on one or more features.

2.1 Simple Linear Regression

As mentioned above, there are many features that could be useful in predicting someone's salary, but we will begin by considering just one: years of experience. That is, we wish to make an accurate prediction of a data scientist's salary by using only their experience as an input. The feature x that we'll base our prediction on is called the **predictor variable** and the quantity y that we are trying to predict is called the **response variable**. In our example, experience is the predictor variable and salary is the response variable.

We believe that experience is related to pay. In particular, we have a feeling that the more experience someone has, the higher their salary is likely to be. This relationship is not *exact*, of course – there is no law that says that someone with x years of experience should be paid *exactly* some amount. But for the purposes of prediction, it is believable that there is some function $H(x)$ which takes in someone's experience level and outputs a prediction of their salary which is reasonably close to their actual salary.

We call such a function a **prediction rule**. You can think of it as a formula for making predictions. Here is one example of a prediction rule:

$$H_1(\text{years of experience}) = \$50{,}000 + \$2,000 \times (\text{years of experience})$$

This rule says that, to predict your pay, we should start with \$50,000 and add \$2,000 for every year of experience you have. In other words, pay increases **linearly** with experience. Another prediction rule is:

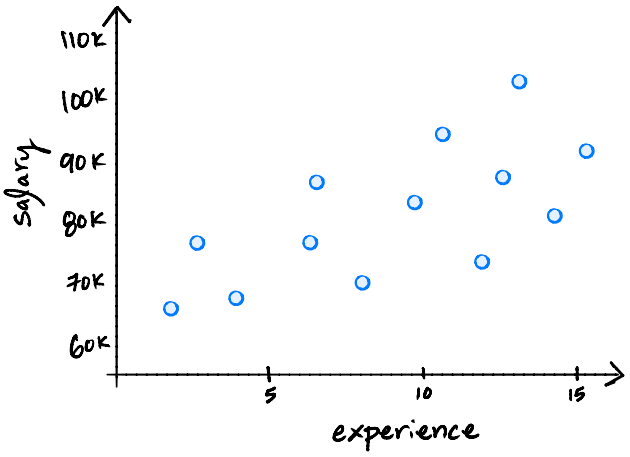
$$H_2(\text{years of experience}) = \$60{,}000 \times 1.05^{(\text{years of experience})}$$

In this rule, pay increases **exponentially** with experience. Yet another prediction rule is:

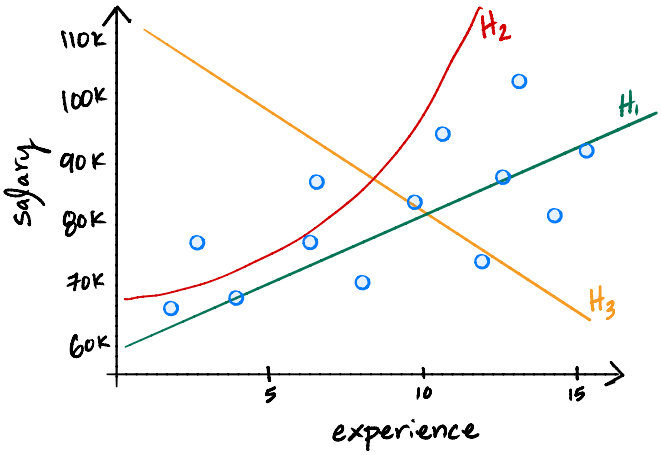
$$H_3(\text{years of experience}) = \$100{,}000 - \$5{,}000 \times (\text{years of experience})$$

This one says that your pay *decreases* with experience. This is (hopefully) not the case, and it goes against our intuition. This is probably a *bad* prediction rule in the sense that it does not make accurate predictions, but it is a prediction rule nonetheless.

As before, we will assess whether a prediction rule is good or bad by seeing if the predictions it makes match the data. The figure below shows what we might expect to see if we ask several data scientists how much they make and how much experience they have.



In general, we observe a *positive* trend: the more experience someone has, the higher their pay tends to be. We can plot the prediction rules H_1, H_2, and H_3 on top of this data to get a sense for the accuracy of their predictions.



Out of these, H_1 appears to "fit" the data the best. We'll now make this more precise.

2.1.1 Loss Functions

Suppose we have surveyed n data scientists, and for each, recorded their experience, x_i, and their salary, y_i. We want to assess the accuracy of a prediction rule, $H(x)$, which takes as input years of experience and outputs the predicted salary.

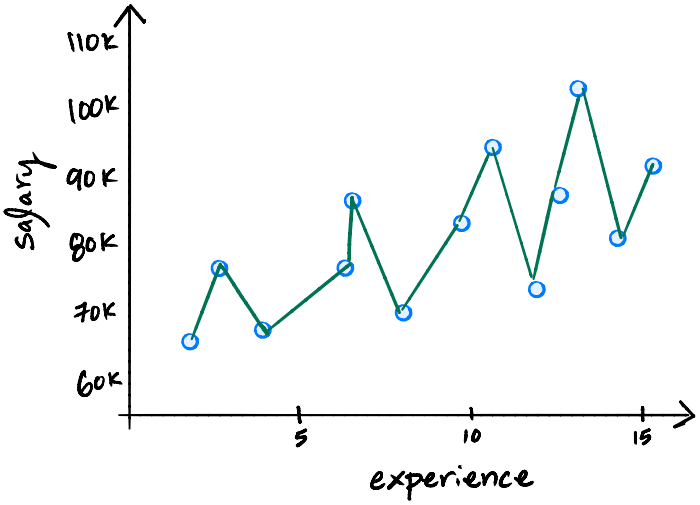
Consider an arbitrary person: person i. Their salary is y_i, and their experience is x_i. Our prediction for their salary is $H(x_i)$. The **absolute loss** of our prediction is

$$|y_i - H(x_i)|$$

In the last chapter, we saw that the *empirical risk*, or average loss incurred when using H to predict the salary for everyone in the data set, is one way to quantify how good or bad H is. In this case, the **mean absolute error** is:

$$R_{\text{abs}}(H) = \frac{1}{n}\sum_{i=1}^{n} |y_i - H(x_i)|$$

Our goal now is to find a prediction rule H which results in the smallest mean absolute error. Here we run into our first problem: it turns out that we can find a function which has *zero* absolute error, but it isn't as useful as we'd like! Here it is:



This prediction rule makes exactly the right prediction for every person in the data set. But in order to go through every data point, this function has made itself quite wiggly. We don't believe that pay is truly related to experience in this way. To put it differently, we feel that the data has some noise, and the above function describes the noise rather than the underlying pattern that we are interested in. When this is the case, we say that the line has **overfit** the data.

One antidote to overfitting is to mandate that the hypothesis not be so wiggly; in other words, restrict the hypothesis space so that it doesn't include such complicated functions. This is an instance of Occam's Razor: we should choose the simplest possible description of the data which actually works. In this example, a straight line is a good, simple description of the data, and so we'll restrict the prediction rule to be of the form $H(x) = w_1 x + w_0$, i.e., straight lines. This setting is called **simple linear regression**.

Our new goal is to find a *linear* prediction rule with the smallest mean absolute error. That is, we want to solve:

$$H^* = \underset{\text{linear } H}{\arg\min} \frac{1}{n}\sum_{i=1}^{n} |y_i - H(x_i)|$$

But there is still a problem with this: it turns out that R_{abs} is relatively hard to minimize.[1]

The cause of this difficulty is our use of the absolute loss. Instead, we'll try another approach to quantifying the error of our prediction rule, the **square loss**:

$$(y_i - H(x_i))^2.$$

The average loss on the data set becomes

$$R_{\text{abs}}(H) = \frac{1}{n}\sum_{i=1}^{n}(y_i - H(x_i))^2.$$

This is also known as the **mean squared error**. Minimizing this is often called **least squares regression**.

2.1.2 Minimizing the Mean Squared Error

Let's now minimize the mean squared error, $R_{\text{sq}}(h)$. Since R_{sq} is a function of H but H is determined by our choice of w_0 and w_1, we will also write $R_{\text{sq}}(h)$ as $R_{\text{sq}}(w_0, w_1)$ to indicate more clearly that the loss is determined by the values of w_0 and w_1. When we minimize the loss function, we are really trying to find the optimal values of w_0, w_1, those that define the best-fitting line for the data. We have:

$$\begin{aligned}
R_{\text{sq}}(H) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - H(x_i))^2 \\
R_{\text{sq}}(w_0, w_1) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - (w_0 + w_1 x_i))^2 \\
R_{\text{sq}}(w_0, w_1) &= \frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2
\end{aligned}$$

This is differentiable. To find the best w_0 and w_1, we'll use calculus. Taking the partial derivatives with respect to w_0 and w_1 and setting them to zero gives

$$\begin{aligned}
\frac{\partial R_{\text{sq}}(w_0, w_1)}{\partial w_0} &= 0 = \sum_{i=1}^{n} -2(y_i - w_0 - w_1 x_i), \\
\frac{\partial R_{\text{sq}}(w_0, w_1)}{\partial w_1} &= 0 = \sum_{i=1}^{n} -2x_i(y_i - w_0 - w_1 x_i).
\end{aligned}$$

We must now solve this system of equations to find the optimal values of w_0 and w_1.

[1] Although it can be done with linear programming.

We will start by setting $\partial R_{\text{sq}}/\partial w_0 = 0$ and solving for w_0.[2]

$$\frac{\partial R_{\text{sq}}}{\partial w_0} = 0 \implies \frac{1}{n}\sum_{i=1}^{n} 2\left(w_1 x_i + w_0 - y_i\right) = 0$$

We can take the 2 outside of the sum:

$$\implies \frac{2}{n}\sum_{i=1}^{n} \left(w_1 x_i + w_0 - y_i\right) = 0$$

If we multiply both sides by $n/2$ we are left with a constant factor of 1 on the left hand side, whereas the right hand side remains zero. In effect, we can get rid of the $2/n$:

$$\implies \sum_{i=1}^{n} \left(w_1 x_i + w_0 - y_i\right) = 0$$

Our goal is to isolate the w_0 on one side of the equation, but we are momentarily prevented from this by the fact that w_0 is inside of the sum. We can remove it from the summation, however; the first step is to break it into three independent sums:

$$\begin{aligned}
&\implies \sum_{i=1}^{n} w_1 x_i + \sum_{i=1}^{n} w_0 - \sum_{i=1}^{n} y_i = 0 \\
&\implies \sum_{i=1}^{n} w_0 = \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} w_1 x_i \\
&\implies n w_0 = \sum_{i=1}^{n} y_i - w_1 \sum_{i=1}^{n} x_i \\
&\implies w_0 = \frac{1}{n}\left(\sum_{i=1}^{n} y_i - w_1 \sum_{i=1}^{n} x_i\right)
\end{aligned}$$

Define $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$. If x_i and y_i are experience and salary, then $\bar{x}$ is the mean experience and $\bar{y}$ is the mean salary of people in our data set. Using this notation

$$\implies w_0 = \bar{y} - w_1 \bar{x}$$

And so we have successfully isolated w_0. We now solve $\partial R_{\text{sq}}/\partial w_1 = 0$ for w_1:

$$\frac{\partial R_{\text{sq}}}{\partial w_1} = 0 \implies \frac{1}{n}\sum_{i=1}^{n} 2\left(\left(w_1 x_i + w_0\right) - y_i\right) x_i = 0$$

[2]Note that we could have also decided to solve this equation for w_1, as long as we then solve the other equation for w_0.

We'll get rid of the 2 and the $1/n$ straight away:

$$\implies \sum_{i=1}^{n} \left((w_1 x_i + w_0) - y_i \right) x_i = 0$$

We now substitute our solution for w_0:

$$\implies \sum_{i=1}^{n} \left((w_1 x_i + \bar{y} - w_1 \bar{x}) - y_i \right) x_i = 0$$

We'll group the x_i with the $\bar{x}$ and the y_i with the $\bar{y}$:

$$\implies \sum_{i=1}^{n} \left((w_1 (x_i - \bar{x}) - (y_i - \bar{y}) \right) x_i = 0$$

Splitting the summand:

$$\begin{aligned}
&\implies \sum_{i=1}^{n} w_1 (x_i - \bar{x}) x_i - \sum_{i=1}^{n} (y_i - \bar{y}) x_i = 0 \\
&\implies w_1 \sum_{i=1}^{n} (x_i - \bar{x}) x_i = \sum_{i=1}^{n} (y_i - \bar{y}) x_i \\
&\implies w_1 = \frac{\sum_{i=1}^{n} (y_i - \bar{y}) x_i}{\sum_{i=1}^{n} (x_i - \bar{x}) x_i}
\end{aligned}$$

We could stop here; this is a totally valid formula for w_1. But we'll continue to get a formula with a little more symmetry. The key is that $\sum_{i=1}^{n} (y_i - \bar{y}) = 0$ and $\sum_{i=1}^{n} (x_i - \bar{x})$, as can be verified. This enables us to write:

$$\implies w_1 = \frac{\sum_{i=1}^{n} (y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

We'll show that the numerators of these last two formulas for w_1 are the same (and you can

show that the denominators are, too). We have:

$$\begin{aligned}\sum_{i=1}^{n}(y_i-\bar{y})(x_i-\bar{x}) &= \sum_{i=1}^{n}(y_i-\bar{y})x_i - \sum_{i=1}^{n}(y_i-\bar{y})\bar{x} \\ &= \sum_{i=1}^{n}(y_i-\bar{y})x_i - \bar{x}\sum_{i=1}^{n}(y_i-\bar{y}) \\ &= \sum_{i=1}^{n}(y_i-\bar{y})x_i - 0 \\ &= \sum_{i=1}^{n}(y_i-\bar{y})x_i\end{aligned}$$

We have arrived at a formula for w_1, the slope of our linear prediction rule, which can be computed from the data. Notice that the formula for the intercept, w_0, involves w_1. So to find the linear prediction rule, we should first calculate w_1, then plug it into the formula to find w_0.

In summary, the formulas we have found are:

$$w_1 = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{n}(x_i-\bar{x})^2}$$

$$w_0 = \bar{y} - w_1\bar{x}$$

where $\bar{x}$ and $\bar{y}$ are the mean of the x_i and y_i, respectively. These are called the **least squares solutions** for the slope and intercept parameters.

It should be emphasized that the fact that we have *formulas* which allow us to calculate w_0 and w_1 directly is a consequence of using the mean squared error. If we had used another loss function, for example, the mean absolute error, then we may not be so lucky to have formulas. Instead, we would have to minimize the loss using an algorithmic approach in which we iterate towards the correct answer.[3]

2.1.3 Fitting non-linear trends

You may be surprised to learn that the formulas we have just derived for fitting a straight line to the data can sometimes be cleverly used to to fit certain nonlinear curves to data. By applying a suitable transformation, it can be possible to turn a nonlinear relationship between

[3]There are even some losses which are so hard to optimize that we can't even hope to find an algorithm which does so exactly; we have to approximate their minimizers.

x and y into a linear relationship between two quantities that can be directly computed from x and y.

For instance, we can fit a curve of the form

$$y = w_0 + w_1 \ln x$$

to data with the tools we already have. While y does not vary linearly with x, it does vary linearly with $\ln x$. So if we let $z = \ln x$ be our feature variable, we can use the formulas we have developed to find the slope and intercept of the linear relationship.

$$w_1 = \frac{\sum_{i=1}^{n}(\ln x_i - \frac{1}{n}\sum_{i=1}^{n}\ln x_i)(y_i - \bar{y})}{\sum_{i=1}^{n}(\ln x_i - \frac{1}{n}\sum_{i=1}^{n}\ln x_i)^2}$$

$$w_0 = \bar{y} - w_1 \cdot \frac{1}{n}\sum_{i=1}^{n}\ln x_i$$

We can use w_1 and w_0 as the parameters that define the best-fitting curve of the form $y = w_0 + w_1 \ln x$.

Similarly, we can use the least squares solutions we derived above to fit data with a curve of the form

$$y = c_0 x^{c_1}.$$

Here, it is not so obvious that this relationship can be written as a linear relationship in new variables. The trick is taking a logarithm of both sides. We can use a logarithm with any base. We'll pick base ten. The relationship becomes

$$\begin{aligned}\log y &= \log(c_0 x^{c_1}) \\ \log y &= \log c_0 + \log x^{c_1} \\ \log y &= \log c_0 + c_1 \log x.\end{aligned}$$

If we let $z = \log x$ be our predictor variable, and $v = \log y$ be our response variable, we see that there is a linear relationship $v = w_0 + w_1 z$, where the intercept is $w_0 = \log c_0$ and the slope is $w_1 = c_1$.

Therefore, we can use the following formulas for w_0 and w_1, which we can then use to find c_0 and c_1.

$$w_1 = \frac{\sum_{i=1}^{n}(\log x_i - \frac{1}{n}\sum_{i=1}^{n}\log x_i)(\log y_i - \frac{1}{n}\sum_{i=1}^{n}\log y_i)}{\sum_{i=1}^{n}(\log x_i - \frac{1}{n}\sum_{i=1}^{n}\log x_i)^2}$$

$$w_0 = \frac{1}{n}\sum_{i=1}^{n}\log y_i - w_1 \cdot \frac{1}{n}\sum_{i=1}^{n}\log x_i$$

Notice that $w_1 = c_1$ and $w_0 = \log c_0$, so we can find c_0 by computing $c_0 = 10^{w_0}$. This gives us a way to find the parameters c_0, c_1 for our original prediction rule $y = c_0 x^{c_1}$.

In general, we can use the formulas we've derived to fit any prediction rule that can be written in the form $g(y) = w_1 \cdot f(x) + w_0$, where $f(x)$ is some transformation of x, such as x^2, e^x, $\log x$, etc. and $g(y)$ is some transformation of y. Here is the procedure:

1. Create a new data set $(z_1, v_1), \ldots, (z_n, v_n)$, where $z_i = f(x_i)$ and $v_i = g(y_i)$.
2. Fit $v = w_1 z + w_0$ using familiar least squares solutions:

$$w_1 = \frac{\sum_{i=1}^{n}(z_i - \bar{z})(v_i - \bar{v})}{\sum_{i=1}^{n}(z_i - \bar{z})^2} \qquad w_0 = \bar{v} - w_1 \cdot \bar{z}$$

   where $\bar{z}$ is the mean of the z_i's and $\bar{v}$ is the mean of the v_i's.
3. If necessary, use w_0 and w_1 to find the parameters of the original prediction rule.

2.2 Multiple Linear Regression

Next, we will look at linear regression through the lens of linear algebra, which will eventually allow us to extend what we know to new settings. In particular, we'll be able to include more than one predictor variable in making our predictions.

We have defined the least squares line as the prediction rule $H(x) = w_0 + w_1 x$ for which the mean square error is minimized. That is, it is the line that minimizes the risk:

$$R_{\text{sq}}(H) = \frac{1}{n}\sum_{i=1}^{n}(y_i - H(x_i))^2$$

or, equivalently:

$$R_{\text{sq}}(w_0, w_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w_0 - w_1 x_i)^2.$$

Notice that the risk is $\frac{1}{n}$ times a sum of squares, and remember from linear algebra that we also measure the length of a vector using a sum of squares. For example, the length of the vector

$$\vec{x} = \begin{bmatrix} 2 \\ 5 \\ 4 \end{bmatrix}$$

is computed as

$$\sqrt{2^2 + 5^2 + 4^2} = \sqrt{45} = \sqrt{9 \cdot 5} = 3\sqrt{5}.$$

If we use $||\vec{v}||$ to denote the length of a vector $\vec{v}$, this means we can write $R_{\text{sq}}(H) = \frac{1}{n}||\vec{e}||^2$, where $\vec{e}$ is the vector whose i^{th} component is $e_i = y_i - H(x_i)$, which is the error in the i^{th} prediction. We'll call $\vec{e}$ the **error vector**. Since each component of $\vec{e}$ comes from a difference between y_i and $H(x_i)$, we can think of the vector $\vec{e}$ as a difference of two vectors $\vec{y}$ and $\vec{h}$, where the i^{th} component of $\vec{y}$ is y_i and the i^{th} component of $\vec{h}$ is $H(x_i)$. We call $\vec{y}$ the **observation vector** and $\vec{h}$ the **hypothesis vector** or **prediction vector**. We are writing the error vector as the difference between the observation vector and the prediction vector. Since $\vec{e} = \vec{y} - \vec{h}$, the risk can be written as $R_{\text{sq}}(H) = \frac{1}{n}\|\vec{e}\|^2 = \frac{1}{n}\|\vec{y} - \vec{h}\|^2$.

Now, since our function $H(x)$ is a linear function of the form $H(x) = w_0 + w_1 x$, this means we can write the prediction vector $\vec{h}$ as follows:

$$\vec{h} = \begin{bmatrix} H(x_1) \\ H(x_2) \\ \vdots \\ H(x_n) \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_1 \\ w_0 + w_1 x_2 \\ \vdots \\ w_0 + w_1 x_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}.$$

Letting

$$X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix} \text{ and } \vec{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix},$$

we have $\vec{h} = X\vec{w}$. The matrix X is called the **design matrix**, and the vector $\vec{w}$ is called the **parameter vector**.

Notice that we are able to write $\vec{h}$ as the product of a matrix and a vector because our choice of hypothesis $H(x)$ was linear in the parameters w_0, w_1. We'll soon be able to adjust the form of $H(x)$ to fit any model that is linear in the parameters, including arbitrary polynomials.

For now, we continue with our linear fit $H(x) = w_0 + w_1 x$. We have established that

$$R_{\text{sq}}(H) = R_{\text{sq}}(w_0, w_1) = \frac{1}{n}\|\vec{y} - \vec{h}\|^2 = \frac{1}{n}\|\vec{y} - X\vec{w}\|^2.$$

Note that we can think of the risk as a function of the parameter vector $\vec{w}$, since changing this parameter vector produces different values for the risk. The values of X and y are

completely determined by the data set, so they are fixed, and $\vec{w}$ is the only thing that can change. Our goal is to find the choice of $\vec{w}$ for which $R_{\text{sq}}(\vec{w}) = \frac{1}{n}||\vec{y} - X\vec{w}||^2$ is minimized. Equivalently, our goal is to find the vector $\vec{w}$ for which the vector $\vec{y} - X\vec{w}$ has the smallest length, since minimizing a constant times the square of the length is equivalent to minimizing the length itself. We'll use calculus to find the value of $\vec{w}$ where the minimum is achieved.

2.2.1 Minimizing the Mean Squared Error

At this point, we have an expression for the mean squared error in matrix notation:

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n}\|\vec{y} - X\vec{w}\|^2.$$

Given the data, X and $\vec{y}$ are fixed, known quantities. Our goal is to find the "best" $\vec{w}$, in the sense that $\vec{w}$ results in the smallest mean squared error. We'll use vector calculus to do so.

Our main strategy for minimizing risk thus far has been to take the derivative of the risk function, set it to zero, and solve for the minimizer. In this case, we want to differentiate $R_{\text{sq}}(\vec{w})$ with respect to a *vector*, $\vec{w}$. The derivative of a scalar-valued function with respect to a vector input is called the **gradient**. The **gradient of $R_{\text{sq}}(\vec{w})$ with respect to $\vec{w}$**, written $\nabla_{\vec{w}} R_{\text{sq}}$ or $\frac{dR_{\text{sq}}}{d\vec{w}}$, is defined to be the vector of partial derivatives:

$$\frac{dR_{\text{sq}}}{d\vec{w}} = \begin{bmatrix} \frac{dR_{\text{sq}}}{dw_0} \\ \frac{dR_{\text{sq}}}{dw_1} \\ \vdots \\ \frac{dR_{\text{sq}}}{dw_d}, \end{bmatrix}$$

where $w_0, w_1, \ldots, w_d$ are the entries of the vector $\vec{w}$. In our case, since we are using a linear prediction rule, $\vec{w}$ has only two components, w_0 and w_1, but we define the gradient for the more general case where $\vec{w}$ has an arbitrary number of components. This definition of the gradient with respect to a vector says that when we think of $R_{\text{sq}}(\vec{w})$ as a function of a vector, we are really thinking of it as a function of multiple variables, which are the components of the vector. The gradient of $R_{\text{sq}}(\vec{w})$ with respect to $\vec{w}$ is the same as the gradient of $R_{\text{sq}}(w_0, w_1, \ldots, w_d)$, a multivariable function.

Our goal is to find the gradient of the mean squared error. We start by expanding the mean squared error in order to get rid of the squared norm, and we write the mean squared error in terms of dot products instead. We need to recall a few key facts from linear algebra, which

we will make use of shortly. Here, A and B are matrices, and $\vec{u}, \vec{v}, \vec{w}, \vec{z}$ are vectors:

$$\begin{aligned}
&(A+B)^T = A^T + B^T, \\
&(AB)^T = B^T A^T, \\
&\vec{u} \cdot \vec{v} = \vec{v} \cdot \vec{u} = \vec{u}^T \vec{v} = \vec{v}^T \vec{u}, \\
&\|\vec{u}\|^2 = \vec{u} \cdot \vec{u}, \\
&(\vec{u} + \vec{v}) \cdot (\vec{w} + \vec{z}) = \vec{u} \cdot \vec{w} + \vec{u} \cdot \vec{z} + \vec{v} \cdot \vec{w} + \vec{v} \cdot \vec{z}.
\end{aligned}$$

Therefore:

$$\begin{aligned}
R_{\text{sq}}(\vec{w}) &= \frac{1}{n} \|\vec{y} - X\vec{w}\|^2 \\
&= \frac{1}{n} (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w}) \\
&= \frac{1}{n} \left(\vec{y}^T - \vec{w}^T X^T\right) (\vec{y} - X\vec{w}) \\
&= \frac{1}{n} \left[\vec{y}^T \vec{y} - \vec{y}^T X\vec{w} - \vec{w}^T X^T \vec{y} + \vec{w}^T X^T X \vec{w}\right].
\end{aligned}$$

Observe that $y^T X\vec{w}$ and $\vec{w}^T X^T \vec{y}$ are both the dot product of $\vec{w}$ with $X^T \vec{y}$ and are therefore equal. So,

$$\begin{aligned}
R_{\text{sq}}(\vec{w}) &= \frac{1}{n} \left[\vec{y}^T \vec{y} - \vec{2} X^T \vec{y} \cdot \vec{w} + \vec{w}^T X^T X \vec{w}\right] \\
&= \frac{1}{n} \left[\vec{y} \cdot \vec{y} - \vec{2} X^T \vec{y} \cdot \vec{w} + \vec{w}^T X^T X \vec{w}\right]
\end{aligned}$$

Now we take the gradient. We have:

$$\begin{aligned}
\frac{dR_{\text{sq}}}{d\vec{w}} &= \frac{d}{d\vec{w}} \left(\frac{1}{n} \left[\vec{y} \cdot \vec{y} - \vec{2} X^T \vec{y} \cdot \vec{w} + \vec{w}^T X^T X \vec{w}\right]\right) \\
&= \frac{1}{n} \left[\frac{d}{d\vec{w}} (\vec{y} \cdot \vec{y}) - \frac{d}{d\vec{w}} \left(\vec{2} X^T \vec{y} \cdot \vec{w}\right) + \frac{d}{d\vec{w}} \left(\vec{w}^T X^T X \vec{w}\right)\right].
\end{aligned}$$

For the first term, the gradient of $\vec{y} \cdot \vec{y}$ with respect to $\vec{w}$ is zero, since $\vec{y}$ does not change as $\vec{w}$ changes, so we treat it as a constant. For the second term, we will use a property shown in the homework, which said that for any constant vector $\vec{v}$, $\frac{d}{d\vec{w}}(\vec{v} \cdot \vec{w}) = \vec{v}$. Letting $\vec{v} = -2X^T \vec{y}$ and applying this property shows the gradient of the $-2X^T \vec{y} \cdot \vec{w}$ with respect to $\vec{w}$ is $-2X^T \vec{y}$. For the last term, we also rely on a homework problem where you show that $\frac{d}{d\vec{w}} \left(\vec{w}^T X^T X \vec{w}\right) = 2X^T X \vec{w}$. Therefore,

$$\frac{dR_{\text{sq}}}{d\vec{w}} = -2X^T \vec{y} + 2X^T X \vec{w}.$$

This is the gradient of the mean squared error. Our next step is to set this equal to zero and solve for $\vec{w}$:

$$\begin{aligned}\frac{dR_{\text{sq}}}{d\vec{w}} = 0 &\implies -2X^T\vec{y} + 2X^TX\vec{w} = 0 \\ &\implies X^TX\vec{w} = X^T\vec{y}.\end{aligned}$$

The last line is an important one. The equation $X^TX\vec{w} = X^T\vec{y}$ defines a system of equations in matrix form known as the **normal equations**. This system can be solved using Gaussian elimination, or, if the matrix X^TX is invertible, by multiplying both sides by its inverse to obtain $\vec{w} = (X^TX)^{-1}X^T\vec{y}$.

The solution $\vec{w}$ of the normal equations, $X^TX\vec{w} = X^T\vec{y}$, is the solution to the least squares problem. Before, we had formulas for the least squares solutions for w_0 and w_1; the normal equations give equivalent solutions. In fact, you can derive our old formulas from the normal equations with a little bit of algebra. The advantage of the normal equations will be that they generalize more easily. We'll be able to use the normal equations to fit arbitrary polynomials to data, as well as to make predictions based on multiple features.

Let's try out an example. We will use the normal equations to find the linear function that best approximates the data $(2, 1)$, $(5, 2)$, $(7, 3)$, $(8, 3)$, where the first element of each pair is the predictor variable x, and the second element is the response variable, y.

We have

$$\vec{y} = \begin{bmatrix}1\\2\\3\\3\end{bmatrix} \text{ and } X = \begin{bmatrix}1 & 2\\1 & 5\\1 & 7\\1 & 8\end{bmatrix}.$$

From this, we can calculate

$$X^TX = \begin{bmatrix}4 & 22\\22 & 142\end{bmatrix} \text{ and } X^Ty = \begin{bmatrix}9\\57\end{bmatrix},$$

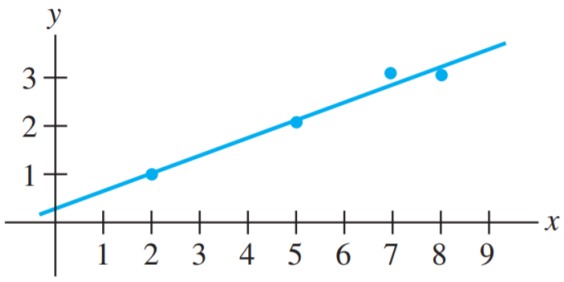
so the normal equations are

$$\begin{bmatrix}4 & 22\\22 & 142\end{bmatrix}\begin{bmatrix}w_0\\w_1\end{bmatrix} = \begin{bmatrix}9\\57\end{bmatrix}.$$

Now, we just need to solve this linear system. The left-hand side matrix can be inverted and left multiplication by its inverse gives

$$\vec{w} = \begin{bmatrix}\frac{2}{7}\\\frac{5}{14}\end{bmatrix}.$$

The regression line is therefore $y = \frac{2}{7} + \frac{5}{14}x$, as shown in the picture below.



You can verify that the formulas for the least squares solutions derived above in section 2.1.2 give the same slope and intercept as we found using the normal equations.

2.2.2 Nonlinear Prediction Rules

The really nice thing about this linear algebra approach to least squares is that we can use the same strategy to fit many different curves to a set of data points. Before, we only considered the case where the prediction rule $H(x)$ was a linear function of x. We were not able to, say, fit an arbitary quadratic of the form $H(x) = w_0 + w_1 x + w_2 x^2$. With the linear algebra method, we can now do this easily.

For example, consider the set of five points $(-2, 0), (-1, 0), (0, 1), (1, 0), (2, 0)$, where the first element of each pair is the predictor variable, x, and the second element is the response variable, y. The system of equations corresponding to a parabola that best fits these points has design matrix

$$X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix},$$

parameter vector

$$\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix},$$

and observation vector

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

We can calculate that

$$X^T X = \begin{bmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{bmatrix} \text{ and } X^T y = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

so that the least squares solution satisfies

$$\begin{bmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{bmatrix} \vec{w} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

Solving this system of equations gives $\vec{w} = \begin{bmatrix} \frac{17}{35} \\ 0 \\ \frac{-1}{7} \end{bmatrix}$ so that the best-fitting quadratic is $y = \frac{17}{35} - \frac{1}{7}x^2$. The plot below shows the data points and the best fitting quadratic.

The least squares method we have developed using linear algebra can be used to find the best-fitting curve of any function form that is **linear in the parameters**, such as

$$H(x) = w_0 + w_1 e^x$$

or

$$H(x) = w_0 + w_1 x + w_2 \sin x.$$

This includes all polynomials because any polynomial

$$H(x) = w_0 + w_1 x + w_2 x^2 + \cdots + w_d x^d$$

is linear when thought of as a function of the coefficients $w_0, w_1, w_2, \ldots, w_d$.

2.2.3 Using Multiple Features

The linear algebra approach is also useful for fitting a function of multiple variables to data, under the same stipulation that the function should be linear in the parameters. This is called **multiple regression** because we are using multiple predictor variables to predict the value of the response variable. This is highly useful because the more predictor variables we take into consideration when predicting the value of the response variable, the more likely we are to make an accurate prediction. For example, Galton might have been able to make more accurate predictions had he used the mother's height and father's height as separate predictor variables, rather than lumping them together into a single midparent height.

For example, suppose we want to predict the price y of a laptop computer based on its weight $x^{(1)}$ and amount of memory $x^{(2)}$. Here we are using superscript notation with parentheses to distinguish between the different predictor variables. In particular, the parentheses are used to emphasize that these are indices, not exponents. We could just use different letters for each predictor variable (and it may help to think of the superscripts as such) but we'll use this notation because we want to be able to generalize to situations where we have an arbitrary number of predictor variables, maybe even more than letters of the alphabet.

To make a prediction, we could collect data for n laptops, recording each laptop's weight in ounces, amount of random access memory in gigabytes, and price in dollars. We use subscripts to distinguish between the different laptops in our data set. The i^{th} laptop in our data set has weight $x_i^{(1)}$, memory $x_i^{(2)}$, and price y_i. Suppose that a plot of the data points $(x_i^{(1)}, x_i^{(2)}, y_i)$ shows that it makes sense to use a prediction rule of the form

$$H(x^{(1)}, x^{(2)}) = w_0 + w_1 x^{(1)} + w_2 x^{(2)},$$

which is the equation of a plane.

Since our function $H(x)$ is linear in the parameters, we can write the prediction vector $\vec{h}$ as follows:

$$\vec{h} = \begin{bmatrix} H(x_1^{(1)}, x_1^{(2)}) \\ H(x_2^{(1)}, x_2^{(2)}) \\ \vdots \\ H(x_n^{(1)}, x_n^{(2)}) \end{bmatrix} = \begin{bmatrix} w_0 + w_1 x_1^{(1)} + w_2 x_1^{(2)} \\ w_0 + w_1 x_2^{(1)} + w_2 x_2^{(2)} \\ \vdots \\ w_0 + w_1 x_n^{(1)} + w_2 x_n^{(2)} \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} \\ 1 & x_2^{(1)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}.$$

This corresponds to a design matrix of

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} \\ 1 & x_2^{(1)} & x_2^{(2)} \\ \vdots & \vdots & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} \end{bmatrix}$$

and a parameter vector of

$$\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}.$$

Letting our observation vector be

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

we have everything we need to solve the normal equations $X^T X \vec{w} = X^T \vec{y}$ and find the parameters of the prediction rule

$$H(x^{(1)}, x^{(2)}) = w_0 + w_1 x^{(1)} + w_2 x^{(2)}$$

with minimum mean squared error.

If you collected actual data on laptop computers and used the normal equations to find the values of the parameters of the prediction rule, what signs would you expect w_0, w_1, and w_2 to have? The more memory the computer has, the more expensive it probably is, and so w_2 would be positive. As for w_1, you could make the case either way: people are willing to spend extra for ultralight laptops, but at the same time, smaller, cheaper computers tend to weigh less than expensive and large gaming laptops with bigger screens. So it isn't clear whether w_1 would be positive or negative. The constant term w_0, known as the **bias**, is often hard to interpret because it represents a nonsensical situation, in this case, the price of a computer that weighs nothing and has no memory. It's again not clear whether w_0 would be positive or negative.

What do the magnitudes of the parameters w_0, w_1, w_2 represent? It is tempting to say that the parameters indicate the contribution of each feature variable towards the overall price of a laptop. For example, if $w_2 > w_1$, you might think that memory is therefore more important than weight in determining a laptop's price. However, there's a problem with this line of reasoning, which is that it doesn't take into account the *scale* of the data. For example, if we had measured a computer's weight in pounds instead of ounces, we'd expect the associated parameter w_2 to be one sixteenth as large. Even if $w_2 > w_1$, reducing w_2 to be a fraction of its original value could make it smaller than w_1, but it wouldn't make sense to say that weight was now less important than memory. In order to directly compare the magnitudes of the parameters, we need to make sure the data is all measured on the same scale. The best way to accomplish this is to standardize each feature variable by subtracting the mean value of that variable and dividing by the standard deviation. This ensures that all feature variables use the same scale, since they are all measured in standard units. Constructing the design matrix from the standardized feature variables and solving the resulting normal equations gives parameters that can be directly compared. These parameters are known as the **standardized regression coefficients.**

In general, suppose we want to use d predictor variables $x^{(1)}, x^{(2)}, \ldots, x^{(d)}$ in our prediction of a response variable y, using a rule of the form

$$H(x^{(1)}, x^{(2)}, \ldots, x^{(d)}) = w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \cdots + w_d x^{(d)}.$$

We collect data from n individuals, and for each individual i, create a **feature vector**, $\vec{x_i}$, which is a vector in $\mathbb{R}^d$ containing each of the d features corresponding to this one individual:

$$\vec{x_i} = \begin{bmatrix} x_i^{(1)} \\ x_i^{(2)} \\ \cdots \\ x_i^{(d)} \end{bmatrix}.$$

Define the **augmented feature vector** $Aug(\vec{x_i})$, to be the vector in $\mathbb{R}^{d+1}$ whose first com-

ponent is 1 and whose other d components are the components of $\vec{x_i}$:

$$Aug(\vec{x_i}) = \begin{bmatrix} 1 \\ x_i^{(1)} \\ x_i^{(2)} \\ \dots \\ x_i^{(d)} \end{bmatrix}.$$

We want to find a prediction rule of the form

$$H(x^{(1)}, x^{(2)}, \dots, x^{(d)}) = w_0 + w_1 x^{(1)} + w_2 x^{(2)} + \cdots + w_d x^{(d)},$$

so we use a design matrix of

$$X = \begin{bmatrix} 1 & x_1^{(1)} & x_1^{(2)} & \dots & x_1^{(d)} \\ 1 & x_2^{(1)} & x_2^{(2)} & \dots & x_2^{(d)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n^{(1)} & x_n^{(2)} & \dots & x_n^{(d)} \end{bmatrix} = \begin{bmatrix} Aug(\vec{x_1})^T \\ Aug(\vec{x_2})^T \\ \vdots \\ Aug(\vec{x_n})^T \end{bmatrix}$$

and a parameter vector of

$$\vec{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_d \end{bmatrix},$$

so we can see that

$$Xw = \begin{bmatrix} Aug(\vec{x_1}) \cdot \vec{w} \\ Aug(\vec{x_2}) \cdot \vec{w} \\ \vdots \\ Aug(\vec{x_n}) \cdot \vec{w} \end{bmatrix} = \begin{bmatrix} H(\vec{x_1}) \\ H(\vec{x_2}) \\ \vdots \\ H(\vec{x_n}) \end{bmatrix}.$$

We let our observation vector be

$$\vec{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

and as before, our goal is to minimize

$$R_{\text{sq}}(\vec{w}) = \frac{1}{n} \|\vec{y} - X\vec{w}\|^2.$$

We already know that the choice of $\vec{w}$ that minimizes $R_{\text{sq}}(\vec{w})$ is the same vector that solves the normal equations $X^T X\vec{w} = X^T\vec{y}$, so we can find the optimal parameters for our prediction rule by solving the normal equations.

Now we can make predictions based on lots of predictor variables instead of just one. Of course, the more predictor variables we have, the better predictions we are able to make, right? Not necessarily. There is such a thing as having *too many* variables. Suppose we are trying to predict a child's height. To do so, we gather as many predictor variables as possible. Some predictors, like the height of each parent, are clearly related to the outcome. But other variables, like the day of the week on which the child was born, are probably unrelated. However, if we have too many of these unrelated variables, by sheer *chance* a pattern can emerge. This pattern isn't meaningful - it's just due to noise - but least squares regression does not know the difference and will happily *overfit* the noise.

More predictor variables also introduce other challenges. For one, it is very hard to visualize multidimensional data, especially data in more than three dimensions, since we don't have a good way of plotting such data. Then it can be hard to guess the appropriate form of the prediction rule. Even if we can come up with an appropriate function form, actually solving the normal equations can be computationally difficult if the design matrix is very large.