Everything you need to know about Simple Linear Regression

I believe a lot of you have used *Add Trendline* in Excel. I bet some of you may even choose to *Display R-squared value on chart* after adding your trendline. Maybe you heard someone talking about this as *regression* or sometimes *linear regression*. But what does a trendline mean? What about regression? How does computer find that trendline? And what does R-squared mean anyway? If you are curious about these questions, here is everything you need to know about simple linear regression, plus some bonus inspirations on machine learning and statistics.

1 What is regression?

Regression is one type of supervised machine learning.

Well that does not seem to help a lot. Let's start with a real-life example instead. When we build a bookshelf, we have to choose plywood. By common sense, we know if our bookshelf is wide, we need thicker or stronger plywood, otherwise the bookshelf would bow or sag because of the weights of books. In contrast, if our bookshelf is narrow, perhaps thinner or weaker plywood would be fine. In this case, using thicker or stronger plywood would add unnecessary cost and thus is not an optimal choice.

Here we have two groups of data: "plywood" and "width of bookshelf". Our task is to answer this question: given width of bookshelf, which plywood should we use? If we call "width of bookshelf" as x and "plywood" as y, then our task is simply: given x, what is y?

How do we answer this question? Admittedly there are many ways, but probably the go-to method for most people is simply looking around existing bookshelves. We just look around, find existing bookshelves, gather their width and which plywood did they use, and then determine the plywood for our new bookshelves. The underlying rationale is that we assume our new bookshelf is similar in nature to these existing ones. If it has similar width as some of the existing ones, maybe we can just go ahead and use a similar type or thickness of plywood as those ones.

In other word, to answer the question "given x, what is y" , we just gather a bunch of existing x and y. We then try to "learn" something from these existing x and y. Based on what we learned, we then make an educated guess of y for our new x. What we described here is a brief picture of supervised learning, in which the existing x and y are called *training* data, and new x is called *testing* data. That is part of story where the word *machinelearning* comes from, in which *machine (computer)* is doing the *learning* instead of us.

In the world of supervised learning, if y is regarded to a label or a type, such as "pine plywood", "cedar plywood", "redwood plywood", and so on, then this problem is called "classification". In contrast, if y is regarded to a continuous number, such as the thickness of a wood piece like "0.32 inch", "0.44 inch", "0.58 inch", and so on, then this problem is called "regression".

We admit this is not a great regression example since dimensions in traditional woodworking are usually limited to multiples of 1/8 inch, but assume we have a high-tech workshop using CNC machines to do woodworking and we can achieve any random dimension like 0.572 inch, then this is truly a regression problem.

2 Math expression of a linear regression

A regression problem, like we said above, is to answer "given x, what is y" by looking at some existing pairs of x and y. In this setting, x is usually called *attribute* or *feature*, while y is usually called *response* or *outcome*.

Assuming we have n existing pairs of x and y, which we can written as

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & ... & x_n \end{bmatrix} \tag{1}$$

$$\mathbf{Y} = \begin{bmatrix} y_1 & y_2 & ... & y_n \end{bmatrix} \tag{2}$$

The task of regression is to find the relation from x to y. In machine learning, sometimes people call this as a *function approximation*, emphasizing we are trying to approximate a function whose input is x and output is y. This function can be any type and you can see some of these functions in the *Trendline Options* in Excel. Today we only talk about one of them, which is linear regression. We will talk about other types of functions in later posts.

As the name linear regression implies, we assume this function is linear. That is saying, we think y is approximately linear to x.

$$\mathbf{Y} \approx w\mathbf{X} + b \tag{3}$$

In this equation, w is usually called *coefficient* in statistics and *weight* is machine learning, while b is usually called *intercept* in in statistics and *bias* is machine learning. The *predicted* or *modeled* outcome $wx + b$ is often denoted as $\hat{y}$.

$$\hat{\mathbf{Y}} = w\mathbf{X} + b \tag{4}$$

In the simplest case, when $\mathbf{X}$ only has one attribute, like out example, "the width of bookshelf", then w is one single scalar. This is often called *simple* *linear regression (SLR)* in statistics.

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_n \end{bmatrix} = w \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} + b \tag{5}$$

For most of the cases, X often have more than one attribute. In our bookshelf example, maybe not only the width but also the height affect the choice of plywood. For a lot of real-world problems, we have dozens or even hundreds of attributes and maybe we do not even know which attributes actually affect the outcome. These problems are often called *multiple linear regression (MLR)* which we will talk about in later post. Today we only focus on simple linear regression.

Let put some numbers into our bookshelf example. We look around our home, our friends' home, our office, and so on, and we find ten bookshelves. We then take a tape measure and write down the width and plywood thickness of each bookshelf.

Maybe our data looks like the following (disclaimer: this data set is made up solely for the purpose of this post. It does not contain any real information regarding bookshelves of plywood. Please do not use this as a reference when building your bookshelf):

$$\begin{aligned} \mathbf{X} &= \text{width of ten bookshelves (in units of inches)} \\ &= \begin{bmatrix} 16 & 20 & 22 & 24 & 28 & 30 & 36 & 38 & 40 & 48 \end{bmatrix} \end{aligned} \tag{6}$$

$$\begin{aligned} &\mathbf{Y} = \text{plywood thickness of ten bookshelves (in units of inches)} \\ &\quad = \begin{bmatrix} 0.245 & 0.235 & 0.414 & 0.244 & 0.487 & 0.522 & 0.487 & 0.692 & 0.744 & 0.721 \end{bmatrix} \end{aligned} \tag{7}$$

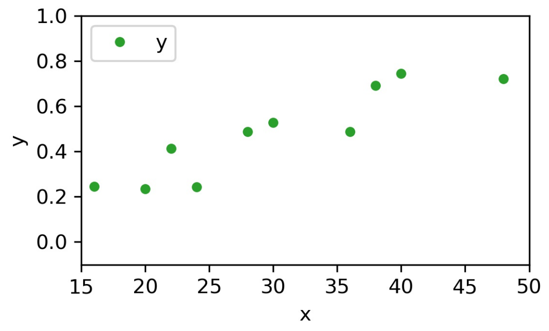

Figure 1: Visualization of our bookshelf data set

Now we have a math representation of simple linear regression. The true outcome is $\mathbf{Y}$ and the predicted outcome is $\hat{\mathbf{Y}} = w\mathbf{X} + b$. Our task is simply "finding proper values of w and b so that $\hat{\mathbf{Y}}$ is as close to $\mathbf{Y}$ as possible".

3 What do we mean by *as close as possible*?

Again, our task is simply "finding proper values of w and b so that $\hat{\mathbf{Y}}$ is as close to $\mathbf{Y}$ as possible". To achieve this task, the first thing we need to do is to define what do we mean by "close to". In other words, we need some quantitative measure to represent how far away is $\hat{\mathbf{Y}}$ from $\mathbf{Y}$.

To define this quantitative measure, first we ask this question, what is the worst predicted $\hat{\mathbf{Y}}$ a *reasonable* person can make?

Admittedly, we can make a lot of arbitrarily bad predictions of $\hat{\mathbf{Y}}$ which is absurdly not close to $\mathbf{Y}$ at all. However, as *reasonable* people, we can probably all agree that there is a lower benchmark for $\hat{\mathbf{Y}}$ which is $\bar{y}$, the mean value of $\mathbf{Y}$.

In our bookshelf example, the mean value of y is 0.479 inch. So in this case, we always use 0.479 inch of plywood for our new bookshelf, regardless of the width. This choice might be wrong, but the point is that it is not absurdly wrong. Usually, we can use this "wrong" answer as a starting point for further improvement.

In math expression, guess everything as the mean of Y is basically let b equals $\bar{y}$ and let w equals 0.

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \dots \\ \hat{y}_n \end{bmatrix} = 0 \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} + \bar{y} = \begin{bmatrix} \bar{y} \\ \bar{y} \\ \dots \\ \bar{y} \end{bmatrix} \tag{8}$$

In this case, our true outcome is $\mathbf{Y} = [y_1, y_2, ..., y_n]$ and our predicted outcome is $\hat{\mathbf{Y}} = [\bar{y}, \bar{y}, ..., \bar{y}]$. These two are visualized in Figure 2. In our example,

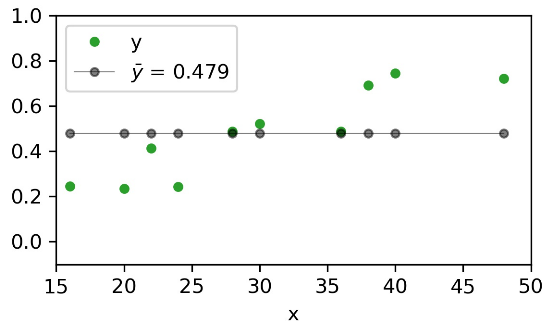

Figure 2: y and $\bar{y}$ of our bookshelf data set

Now our question is how close is our predicted value to the true value? The "distance" between two are usually represented as *sum of squares*. In this case, the distance between $\mathbf{Y}$ and $\hat{\mathbf{Y}}$ can be calculated as

$$SS_T = \sum_{i=1}^{n} (y_i - \bar{y})^2 \tag{9}$$

In our example, SS_T is

$$\begin{aligned} SS_T &= \sum_{i=1}^{n} (y_i - \bar{y})^2 \\ &= (y_1 - \bar{y})^2 + (y_2 - \bar{y})^2 + ... + (y_n - \bar{y})^2 \\ &= (0.245 - 0.479)^2 + (0.235 - 0.479)^2 + ... + (0.721 - 0.479)^2 \\ &= 0.350 \end{aligned} \tag{10}$$

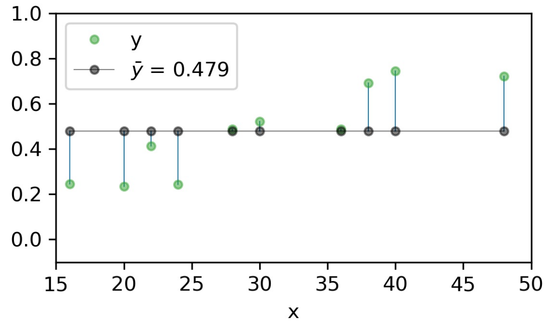

Figure 3: SS_T of our bookshelf data set

The term SS_T stands for *total sum of squares*, which is visualized in Figure 3. In this figure, the vertical lines connecting corresponding y and $\bar{y}$ represents $|y - \bar{y}|$, and thus SS_T is the sum of the squares of the length of these lines.

In some way, we can think of SS_T as a lower bound benchmark of our model. A reasonable model should at least works better than this.

Now let's come up a better linear regression model. For now, let's say we just make a somewhat reasonable guess. Let's guess b as 0.4 and w as 0.02. So now our model becomes $\hat{y} = 0.02x + 0.4$.

Here comes our second question: how much better is this model $\hat{y} = 0.02x + 0.4$ compared to the lower bound $\hat{y} = 0.479$? To compare these two, we can plot them on the same graph.

Similarly we use sum of square to represent the distance between them.

$$SS_R = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 \tag{11}$$

The term SS_R stands for *regression sum of squares*, also called *explained sum of squares*. In some sense, we can think of SS_R as how much better our model is

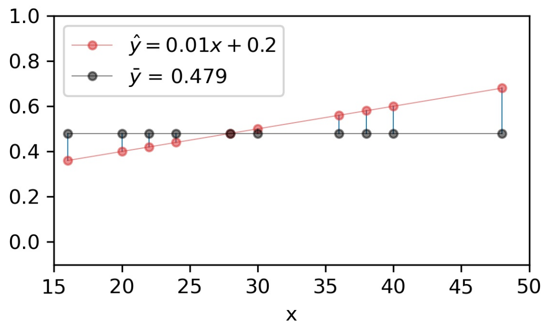

Figure 4: SS_R of our bookshelf data set

compared to the lower bound benchmark. In other words, compared to simply saying everything equals average, now our model at least *explains* something, which gives us some insights of what true value y looks like.

In our example,

$$\begin{aligned}SS_T &= \sum_{i=1}^{n}(y_i - \bar{y})^2 \\ &= (\hat{y}_1 - \bar{y})^2 + (\hat{y}_2 - \bar{y})^2 + ... + (\hat{y}_n - \bar{y})^2 \\ &= ((0.01 \times 16 + 0.2) - 0.479)^2 + ((0.01 \times 20 + 0.2) - 0.479)^2 + ... + ((0.01 \times 48 + 0.2) - 0.479)^2 \\ &= (0.36 - 0.479)^2 + (0.40 - 0.479)^2 + ... + (0.68 - 0.479)^2 \\ &= 0.098\end{aligned} \tag{12}$$

What can I say about SS_T? In some sense, SS_T tells us that our model $\hat{y} = 0.02x + 0.4$ is 0.098 better than our baseline $\hat{y} = 0.479$. So we do get a little better. Nevertheless, no matter how much better our model is, we probably will not achieve perfect. Our third question is then how far away is our model from absolutely perfect?

Our model predicts the outcome as $[\hat{y}_1, \hat{y}_2, ..., \hat{y}_n]$ while the prefect answer is the true outcome which is $[y_1, y_2, ..., y_n]$.

The distance from our model to absolute perfectness is then

$$SS_E = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{13}$$

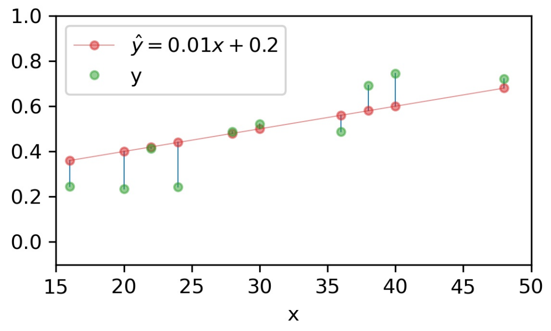

Figure 5: SS_E of our bookshelf data set

In our example,

$$\begin{aligned} SS_E &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= (y_1 - \hat{y}_1)^2 + (y_2 - \hat{y}_2)^2 + ... + (y_n - \hat{y}_n)^2 \\ &= (0.245 - 0.36)^2 + (0.235 - 0.4)^2 + ... + (0.721 - 0.68)^2 \\ &= 0.120 \end{aligned} \tag{14}$$

The term SS_E stands for *error sum of squares*, also called *residual sum of squares*. The smaller SS_E is, the less "error", the closer our model is to perfectness. In our example, we can say our model is still 0.120 away from absolute perfectness.

In summary, SS_T is the distance from the worst possible model to perfectness, SS_R is the distance from the worst possible model to our model, and SS_E is the distance from our model to perfectness.

For our task to make $\hat{y}$ as close to y as possible, we now define it as making our model as close to absolute perfectness as possible, which is minimize SS_E.

4 How do we minimize SS_E?

As we explained above, SS_E indicates the distance between our model to perfectness, and thus we would like to minimize it.

$$SS_E = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{15}$$

where $\hat{y}_i$ is the predicted outcome from our model

$$\hat{y}_i = wx_i + b \tag{16}$$

Therefore, SS_E is written as

$$SS_E = \sum_{i=1}^{n} [y_i - (wx_i + b)]^2 \tag{17}$$

The problem is then given x_i and y_i, find w and b that minimize SS_E. This is an example of optimization problem, which can be solved by many method. For linear regression, there is one particular effective method: *gradient descend*.

The basic idea of gradient descend it to find the derivative relation from SS_E to w and b, and gradually changing w and b to the desired direction. In other word, if increase of w or b would decrease SS_E, we will increase w or b by a little bit and see how it goes. In contrast, if increase of w or b would increase SS_E,we will decrease w or b by a little bit and see how it goes.

$$\begin{aligned}
\frac{\partial SS_E}{\partial w} &= \frac{\partial}{\partial w} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
&= \frac{\partial}{\partial w} \sum_{i=1}^{n} [y_i - (wx_i + b)]^2 \\
&= \sum_{i=1}^{n} \left\{ 2[y_i - (wx_i + b)] \frac{\partial}{\partial w}(-wx_i) \right\} \\
&= \sum_{i=1}^{n} \{ -2[y_i - (wx_i + b)]x_i \} \\
&= -2 \sum_{i=1}^{n} [(y_i - \hat{y}_i)x_i]
\end{aligned} \tag{18}$$

$$\begin{aligned}
\frac{\partial SS_E}{\partial b} &= \frac{\partial}{\partial b} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\
&= \frac{\partial}{\partial b} \sum_{i=1}^{n} [y_i - (wx_i + b)]^2 \\
&= \sum_{i=1}^{n} \left\{ 2[y_i - (wx_i + b)] \frac{\partial}{\partial b}(-b) \right\} \\
&= \sum_{i=1}^{n} \{ -2[y_i - (wx_i + b)] \} \\
&= -2 \sum_{i=1}^{n} (y_i - \hat{y}_i)
\end{aligned} \tag{19}$$

This can also be written as

$$\begin{aligned}
\frac{\partial SS_E}{\partial w_j} &= -2 \sum_{i=1}^{n} [(y_i - \hat{y}_i)x_{i,j}] \\
&= -2 [(y_1 - \hat{y}_1)x_{1,j} + (y_2 - \hat{y}_2)x_{2,j} + ... + (y_n - \hat{y}_n)x_{n,j}]
\end{aligned} \tag{20}$$

In matrix expression, this is

$$\frac{\partial SS_E}{\partial w_j} = -2(Y - \hat{Y})X \tag{21}$$

We can verify this expression as

$$\begin{bmatrix} \frac{\partial SS_E}{\partial w_0} \\ \frac{\partial SS_E}{\partial w_1} \\ \frac{\partial SS_E}{\partial w_2} \\ ... \\ \frac{\partial SS_E}{\partial w_k} \end{bmatrix} = -2 \begin{bmatrix} y_1 - \hat{y}_1 & y_2 - \hat{y}_2 & ... & y_n - \hat{y}_n \end{bmatrix} \begin{bmatrix} 1 & x_{1,1} & x_{1,2} & ... & x_{1,k} \\ 1 & x_{2,1} & x_{2,2} & ... & x_{2,k} \\ ... & ... & ... & ... & ... \\ 1 & x_{n,1} & x_{n,2} & ... & x_{n,k} \end{bmatrix}$$

$$= -2 \begin{bmatrix} (y_1 - \hat{y}_1) \times 1 + (y_2 - \hat{y}_2) \times 1 + ... + (y_n - \hat{y}_n) \times 1 \\ (y_1 - \hat{y}_1)x_{1,1} + (y_2 - \hat{y}_2)x_{2,1} + ... + (y_n - \hat{y}_n)x_{n,1} \\ (y_1 - \hat{y}_1)x_{1,2} + (y_2 - \hat{y}_2)x_{2,2} + ... + (y_n - \hat{y}_n)x_{n,2} \\ ... \\ (y_1 - \hat{y}_1)x_{1,k} + (y_2 - \hat{y}_2)x_{2,k} + ... + (y_n - \hat{y}_n)x_{n,k} \end{bmatrix} \tag{22}$$

One we get $\frac{\partial SS_E}{\partial w_j}$, we then update w_j accordingly. If $\frac{\partial SS_E}{\partial w_j} > 0$, this indicates an increase of w_j would increase SS_E. Therefore we should decrease w_j as the following:

$$w_j = w_j - \alpha \frac{\partial SS_E}{\partial w_j} \tag{23}$$

where α is an arbitrary coefficient to represent how much distance we would like w_j to move along the desired direction.

In contrast, if $\frac{\partial SS_E}{\partial w_j} < 0$, this indicates an increase of w_j would decrease SS_E. Therefore we should increase w_j as the following:

$$w_j = w_j - \alpha \frac{\partial SS_E}{\partial w_j} \tag{24}$$

Notice in this equation, sine $\frac{\partial SS_E}{\partial w_j} < 0$, w_j minus $\frac{\partial SS_E}{\partial w_j}$ is in fact increasing w_j.

This is one step of gradient descend. To find the optimal w, we iterate over and over again, moving w_j along the desired direction a little bit in each iteration. Eventually, all w_j is converged to its optimal value. To check if the iteration of gradient descend is converged, we can monitor the value of SS_E. If SS_E does not decrease from iteration to iteration or if the decrease is too small, we can conclude the iterations are converged and there is no need to keep updating w.