Supervised machine learning: Simple linear regression algorithm

Alon B. Muhame[1]

[1] *Women's Institute of Technology and Innovation (WITI)*

December 19, 2022

Abstract

Simple Linear Regression is a statistical method that allows us to summarize and study relationships between two continuous (quantitative) variables. It is probably one of the most popular and well-inferential algorithms in statistics and machine learning. One variable is considered as an explanatory variable and the other one as a dependent variable. See Figure 1 for summary description.

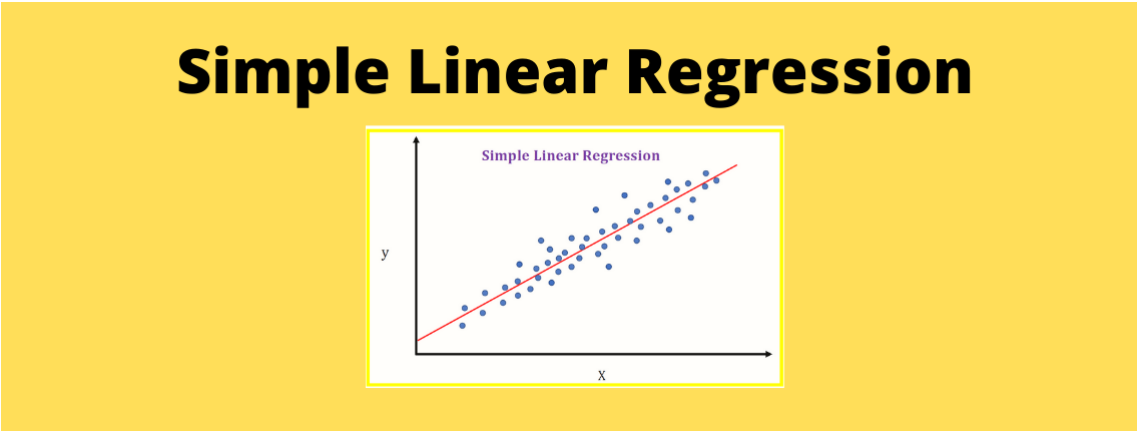

Figure 1: A description of simple linear regression algorithm

1 Introduction

1.1 Predicting a response using a single feature- simple linear regression

It is a method for predicting a dependent variable ($\boldsymbol{Y}$) based on values of independent variable ($\boldsymbol{X}$). It is assumed that two variable are linearly related. Hence, we try to find a linear function that predicts the response of value ($\boldsymbol{y}$) as accurately as possible as a function of feature value or independent variable ($\boldsymbol{x}$):

$$\boldsymbol{y} = \boldsymbol{b_0} + \boldsymbol{b_1}\boldsymbol{x_1}, \tag{1}$$

where the expression shows the linear relationship between the dependent variable ($\boldsymbol{y}$) and independent variable ($\boldsymbol{x}$).

1.2 How to find the best fit line

In this regression model, we are trying to minimise the errors in prediction by finding the 'line of best fit' - the regression line from the errors would be minimal. Basically, we are trying to minimise the length between the observed value ($\boldsymbol{y_i}$) and the predicted value ($\boldsymbol{y_p}$).
That is: $\mathbf{min}\{\ \mathbf{sum}(\boldsymbol{y_i}$ - $\boldsymbol{y_p})\ \}$

1.3 A motivating example

In the regression task, we will predict the percentage marks that student is expected to score based upon the number of hours they studied.

$$\boldsymbol{score} = \boldsymbol{b_0} + \boldsymbol{b_1} * \boldsymbol{hours}, \tag{2}$$

where the expression shows the linear relationship between the dependent variable ($\boldsymbol{y}$) and independent variable ($\boldsymbol{hours}$).

2 Conceptual definition

Linear regression is a prediction method that is more than 200 years old. Simple linear regression is a great first machine learning algorithm to implement as it requires you to estimate properties from your training dataset, but is simple enough for beginners to understand. In this tutorial, you will discover how to implement the simple linear regression algorithm from scratch in Python.

While not exciting, linear regression finds widespread use both as a standalone learning algorithm and as a building block in more advanced learning algorithms. The output layer of a deep neural network trained for regression with MSE loss, simple AR time series models, and the "local regression" part

of LOWESS smoothing are all examples of linear regression being used as an ingredient in a more sophisticated model.

Linear regression is also the "simple harmonic oscillator" of machine learning; that is to say, a pedagogical example that allows us to present deep theoretical ideas about machine learning in a context that is not too mathematically taxing.

There is also the small matter of it being the most widely used supervised learning algorithm in the world; although how much weight that carries I suppose depends on where you are on the "applied" to "theoretical" spectrum.

$$\arg\min_f L(X, Y, f(X))$$

In summary, a machine learning minimizes a loss function L by finding the best function f that to predict target Y from features X. A good machine learning model has a low loss on the test data.

2.1 Statistical motivation

In this section we will use statistical considerations to motivate the definition of a particular mathematical optimization problem under linear regression. Once we have posed this problem, we will afterwards ignore the statistics altogether and focus on numerical methods for solving the optimization problem using a linear regression - in particular the least squares method.

Let's start by deriving the so-called normal equation from a statistical model. Let's say that $\boldsymbol{X}$ is a random vector of length $\boldsymbol{m}$ and $\boldsymbol{Y}$ is a scalar random variable. $\boldsymbol{X}$ and $\boldsymbol{Y}$ are not independent, but have a joint probability distribution $\boldsymbol{F(x, y; \Theta, \sigma)}$ parameterized by a non-random parameter vector Θ, a non-negative scalar σ, and a random error term $\epsilon \sim \boldsymbol{N}(0, \sigma^2)$. The model is:

$$\boldsymbol{Y} = \boldsymbol{X}^T\Theta + \epsilon, \tag{3}$$

Now suppose we sample $\boldsymbol{n}$ independent observations from $\boldsymbol{F}$. We place these into a real-valued $n \times m$ matrix $\boldsymbol{X}$ and a real-valued vector $\boldsymbol{y}$. Just to be absolutely clear, $\boldsymbol{X}$ and $\boldsymbol{y}$ are not random variables; they are the given data used to fit the model. We can then ask, what is the likelihood of obtaining the data $(\boldsymbol{X}, \boldsymbol{y})$ given a parameter vector Θ? By rearranging our equation (3) as
$\boldsymbol{Y} - \boldsymbol{X}.\Theta = \epsilon \sim \boldsymbol{N}(0, \sigma^2)$,
and using the probability density function (p.d.f) of the normal distribution, we can see that:

$$L(\Theta; \mathbf{X}, \mathbf{y}) = P(\mathbf{X}, \mathbf{y}; \Theta) \tag{4}$$
$$= \prod_{i=1}^{n} P(\mathbf{X}_i, \mathbf{y}_i; \Theta) \tag{5}$$
$$= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\Big(\frac{-(\mathbf{y}_i - \mathbf{X}_i^T\Theta)^2}{2\sigma^2}\Big) \tag{6}$$
$$\tag{7}$$

That looks pretty awful, but there are a couple easy things we can do to make it a look a lot simpler. First, that constant term out front doesn't matter at all: let's just call it C or something. We can also take that $e^{-2\sigma^2}$ outside the product as $e^{-2N\sigma^2}$, which we'll also stuff into the constant C because we're only interested in Θ right now. Finally, we can take a log to get rid of the exponential and turn the product into a sum. All together, we get the log-likelihood expression:

$$\ell(\Theta; \mathbf{X}, \mathbf{y}) = \log L(\Theta; \mathbf{X}, \mathbf{y}) \tag{8}$$
$$= C - \sum_{i=1}^{N} -(\mathbf{y}_i - \mathbf{X}_i^T\Theta)^2 \tag{9}$$
$$= C - \|\mathbf{y} - \mathbf{X}\Theta\|^2 \tag{10}$$
$$\tag{11}$$

Now, because log is a monotonically increasing function, maximizing ℓ is the same as maximizing $\boldsymbol{L}$. Furthermore, the constant $\boldsymbol{C}$ has no effect whatsoever on the location of the maximum. We can also remove the minus sign and consider the problem as a minimization problem instead. Therefore our maximum likelihood estimate of Θ for a given data set (X, y) is simply:

$$\hat{\Theta} \triangleq \underset{\Theta}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\Theta\|^2 \tag{12}$$

Why do we use the squared error as a loss function? In short, using the MSE corresponds to computing a maximum likelihood solution to our problem. For a more detailed explanation [1]. If statistics isn't really your thing, I have some good news for you: we're quits with it. Everything in this final equation is now a real-valued vector or matrix: there's not a random variable or probability distribution in sight. It's all over except for the linear algebra.

What we did above was essentially a short sketch of the relevant parts of the Gauss-Markov theorem. In particular, we've shown that the OLS solution to $\mathbf{y} - \mathbf{X}\Theta$ is the Maximum Likelihood Estimate for the parameters of the particular statistical model we started with. This isn't true in general, but it is *exactly* true for linear regression with a normally distributed error term. The full Gauss-Markov theorem proves a bunch of other nice properties: for example, it turns out this estimator is unbiased and we can even say that it's optimal in the sense that it is the best possible linear unbiased estimator. But we won't need these further theoretical results to implement a working model.

[1] `https://datascience.stackexchange.com/questions/10188/why-do-cost-functions-use-the-square-error`

If there's one thing you should remember, it's this: the fact that the p.d.f. of the Gaussian distribution has the quadratic term $(x-\mu)^2$ in the exponent is the reason why squared error is the "right" loss function for linear regression. If the error of our linear model had a different distribution, we'd have to make a different choice.

Our key takeaway is that if it's true that our response variable is related to our predictor variables by a linear equation plus a certain amount of random Gaussian noise, then we can recover good, unbiased estimates of that linear equations coefficients from nothing more than a finite number of data points sampled from the underlying distribution, and the way to actually calculate those estimates is to solve the OLS problem for the data set.

2.2 Ordinary Least Squares (OLS)

Note: for this next section, we're going to be doing some light vector calculus. I suggest you reference the *matrix cookbook* [2] if any of the notation or concepts aren't familiar.

Let's call the right-hand side (the part we're trying to minimize) $\boldsymbol{J}$. Then we have:

$$J(\Theta) = \|\mathbf{y} - \mathbf{X}\Theta\|^2 \tag{13}$$

And the problem is to minimize $\boldsymbol{J}$ with respect to Θ. As optimization problems go, this one is pretty well behaved: it's continuous, quadratic, convex, everywhere continuously differentiable, and unconstrained. That's a fancy way of saying that it's shaped like a big, round bowl. A bowl has a unique lowest point and it can always be found simply by letting a marble roll down hill until it comes to rest exactly at the lowest point.

Because of these nice properties (and a very useful set of theorems called the *KKT* [3] conditions we know that these properties guarentee that $\boldsymbol{J}$ has a unique global minimum and that we can find the minimum - the bottom of the bowl - by finding the one place where the gradient is zero in all directions.

Now, it may not be obvious at first how to take the gradient of the squared norm of a vector, but recall that it is the inner product of that vector with its dual:

$$\nabla_\Theta J = \nabla_\Theta \, (\mathbf{y} - \mathbf{X}\Theta)^T(\mathbf{y} - \mathbf{X}\Theta) \tag{14}$$

Expanding it out with FOIL:

$$\nabla_\Theta J = \nabla_\Theta \, (\mathbf{y}^T\mathbf{y} - (\mathbf{X}\Theta)^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\Theta + \Theta^T(\mathbf{X}^T\mathbf{X})\Theta) \tag{15}$$

It's obvious that $\mathbf{y}^T\mathbf{y}$ is constant with respect to Θ so the first term simply vanishes. It's less obvious but also true that the next two terms are equal to each other - just remember that a $\boldsymbol{J}$ is a scalar, so those terms are each scalar, and the transpose of a scalar is itself. The final term is a quadratic form, and the general rule [4] is $x^TAx = A^Tx + Ax$ but because the product of a matrix with itself is always symmetric $X^TX = (X^TX)^T$ we can use the simpler form $\nabla x^TAx = 2Ax$

$$\nabla_\Theta J = -2\mathbf{X}^T\mathbf{y} + 2\mathbf{X}^T\mathbf{X}\Theta \tag{16}$$

[2] https://www.math.uwaterloo.ca/ hwolkowi/matrixcookbook.pdf

[3] `https://en.wikipedia.org/wiki/Karush%E2%80%93Kuhn%E2%80%93Tucker_conditions`

[4] https://www.cs.ubc.ca/ schmidtm/Courses/340-F16/linearQuadraticGradients.pdf

Setting this equal to zero at the minimum, which we will call $\hat{\Theta}$, and dividing both sides by two, we get:

$$\mathbf{X}^T\mathbf{X}\hat{\Theta} = \mathbf{X}^T\mathbf{y} \qquad \textbf{(Normal Equation)}$$

This is the so-called **normal equation**. The importance of this step is that we've reduced the original optimization problem to a system of linear equations which may be solved purely by the methods of linear algebra. To see this, note that we know $\boldsymbol{X}$ and $\boldsymbol{y}$, so the right hand side is a known vector, the left-hand side is a matrix times an unknown vector, so this is just the familiar equation for solving for a particular solution to a system of equations $Ax = b$.

Because $\mathbf{X}^T\mathbf{X}$ is square and non-singular and therefore invertible, we could just left-multiply both sides by its inverse to get an explicit closed form for $\hat{\Theta}$:

$$\hat{\Theta} = (\mathbf{X}^T\mathbf{X})^{-1}X^T\mathbf{y} \tag{17}$$

However, it turns out there is a faster and more numerically stable way of solving for $\hat{\Theta}$ which relies on the QR Decomposition of the matrix $\mathbf{X}$.

3 Implementing Linear Regression:

Recall that our problem will always be to solve the normal equation:

$$X^TX\hat{\Theta} = X^Ty \tag{18}$$

or as written in equation (17) above. One can get a another version for proof of the least squares methods on [5]

3.1 Preprocess the data

The following steps are key as we have implemented in class:

1. Import the Libraries
2. Import the DataSet
3. Check for nay missing data points
4. Split the DataSet
5. Additionally: One may need to carryout feature scaling for some cases under multiple linear regression but not simple linear regression

[5] `https://en.wikipedia.org/wiki/Least_squares`

Conclusion note:
That was linear regression from scratch. There's a lot more that could be said about linear regression even as a black box predictive model: polynomial and interaction terms, L1 and L2 regularization, and linear constraints on coefficients come to mind.

There's also a whole universe of techniques for doing statistical modeling and inference with linear regression: testing residuals for homoscedasticity, normality, autocorrelation, variance inflation factors, orthogonal polynomial regression, Cook's distance, leverage, Studentized residuals, ANOVA, AIC, BIC, Omnibus F-tests on nested models, etc., etc. Just to be clear, these aren't variations or generalization of linear regression (although there are tons of those too) these are just standard techniques for analyzing and understanding linear regression models of the exact same form we calculated above. The topic is very mature and a huge amount of auxiliary mathematical machinery has been built up over the centuries (Gauss was studying OLS around 1800, and the problem is older than that.)

However, if we go too deeply into linear regression, we won't get a chance to explore the rest of machine learning.