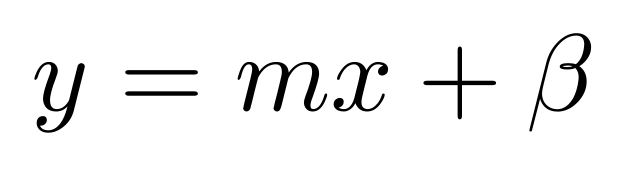
A beginner Machine Learning Engineer's biggest fear: Mathematics

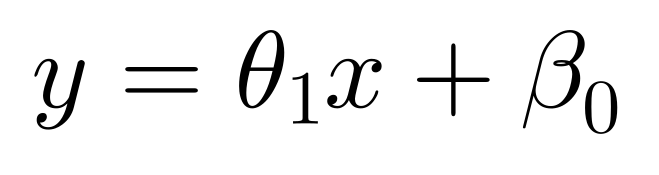
…and for good reason, because it is after all a daunting thing to learn in of itself, given its immensely vast disciplines. But fear not, I believe firmly that one need not learn every ins, outs, and backwards of this seemingly dreadful thing we call mathematics, not at all, not for machine learning at least, because machine learning not just in my personal experience, but I'm sure for others as well, involves three main key concepts in order to be able to build so called Artificial Intelligence/Machine learning systems: Matrix and vector operations from the sub-discipline of mathematics called Linear Algebra, Partial differentiation from calculus, and basic Standardization and Normalization of values from the discipline of Statistics.

Learning merely these three most basic foundational concepts I personally believe will take you farther in your journey in learning the field than doing so otherwise, because knowing these will give you the ease of mind to understand how most machine learning systems work under the hood, being able to debug and fix whatever error should arise when need be, and most importantly being able to build them from scratch as opposed to using frameworks and libraries that already have these systems implemented and optimized which don't get me wrong is not at all bad, but at best these would be at first glance mere black box objects where one would have no idea how these systems work except the input it takes and output it gives.

To start off I think any machine learning engineer would tell you that in their humble beginnings the simplest model/system they would have built was a Linear Regression model. Now what is a Linear Regression model? Well if you remember the most basic math concept in high school or elementary, it was simply the slope intercept formula defined by the following.



And in which we will now be defining as…

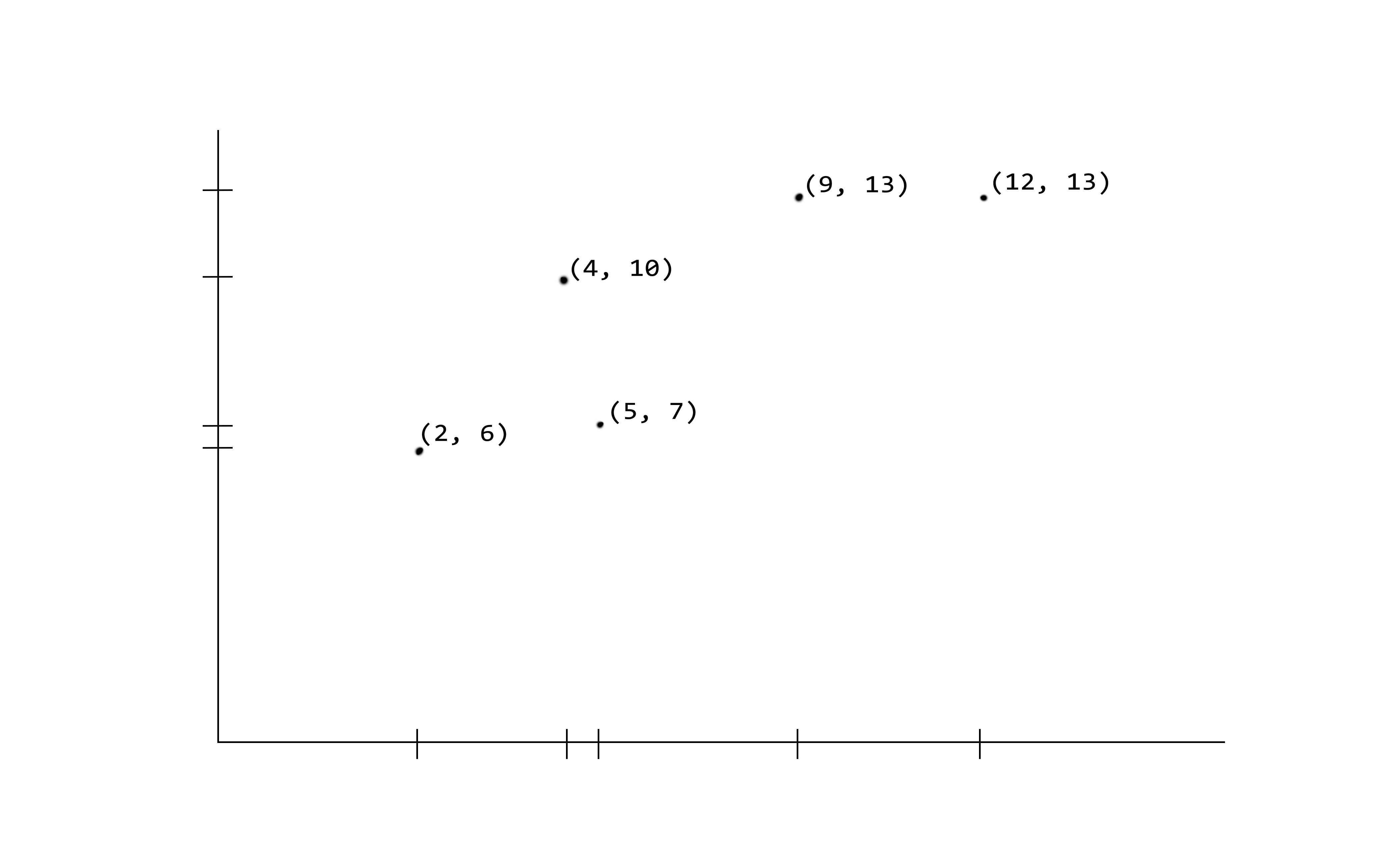


Where (theta subscript 1) is denoted as our first coefficient or what we previously knew as our slope, and (beta subscript 0) is denoted as our bias coefficient or what we previously knew as our y-intercept.

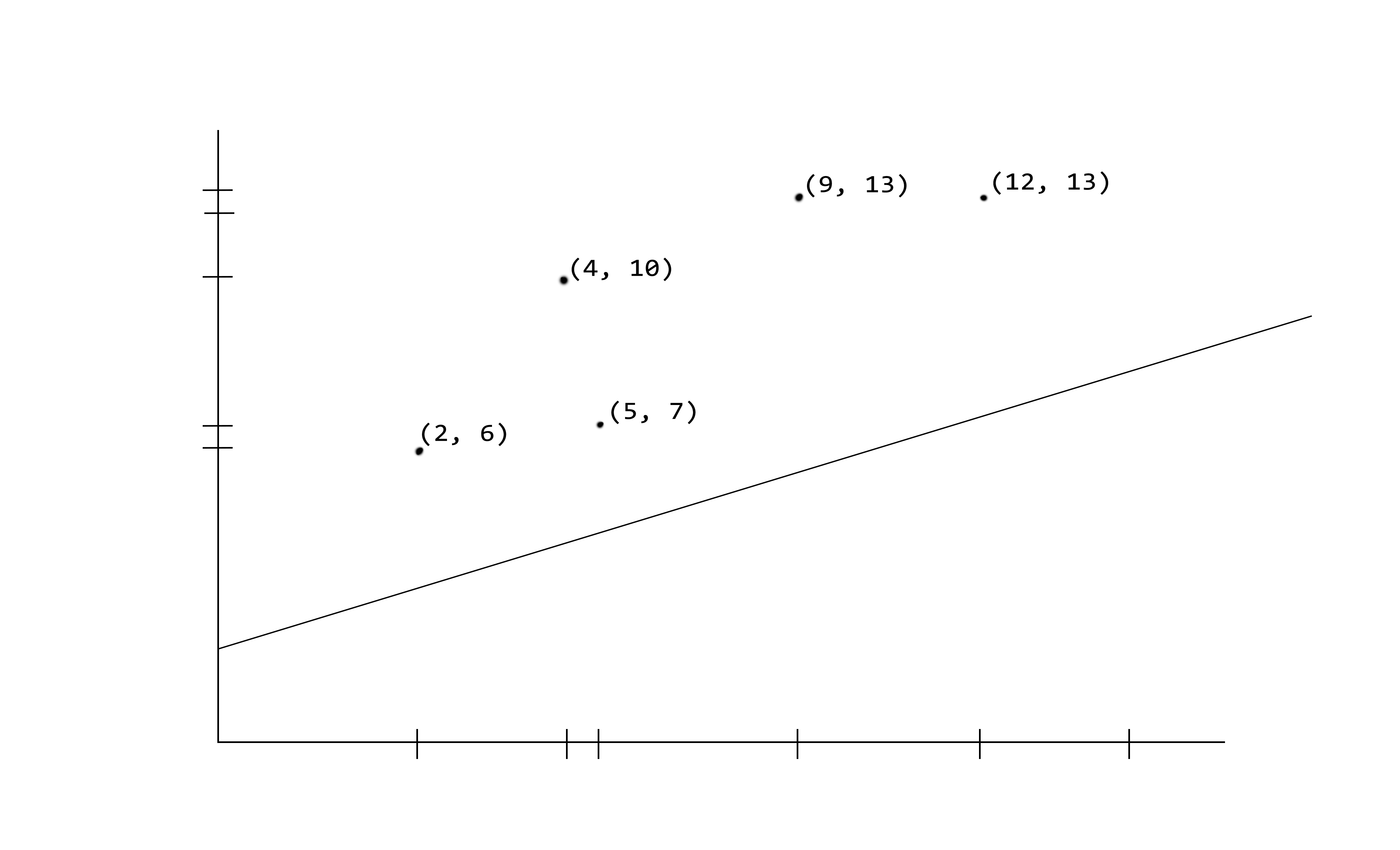
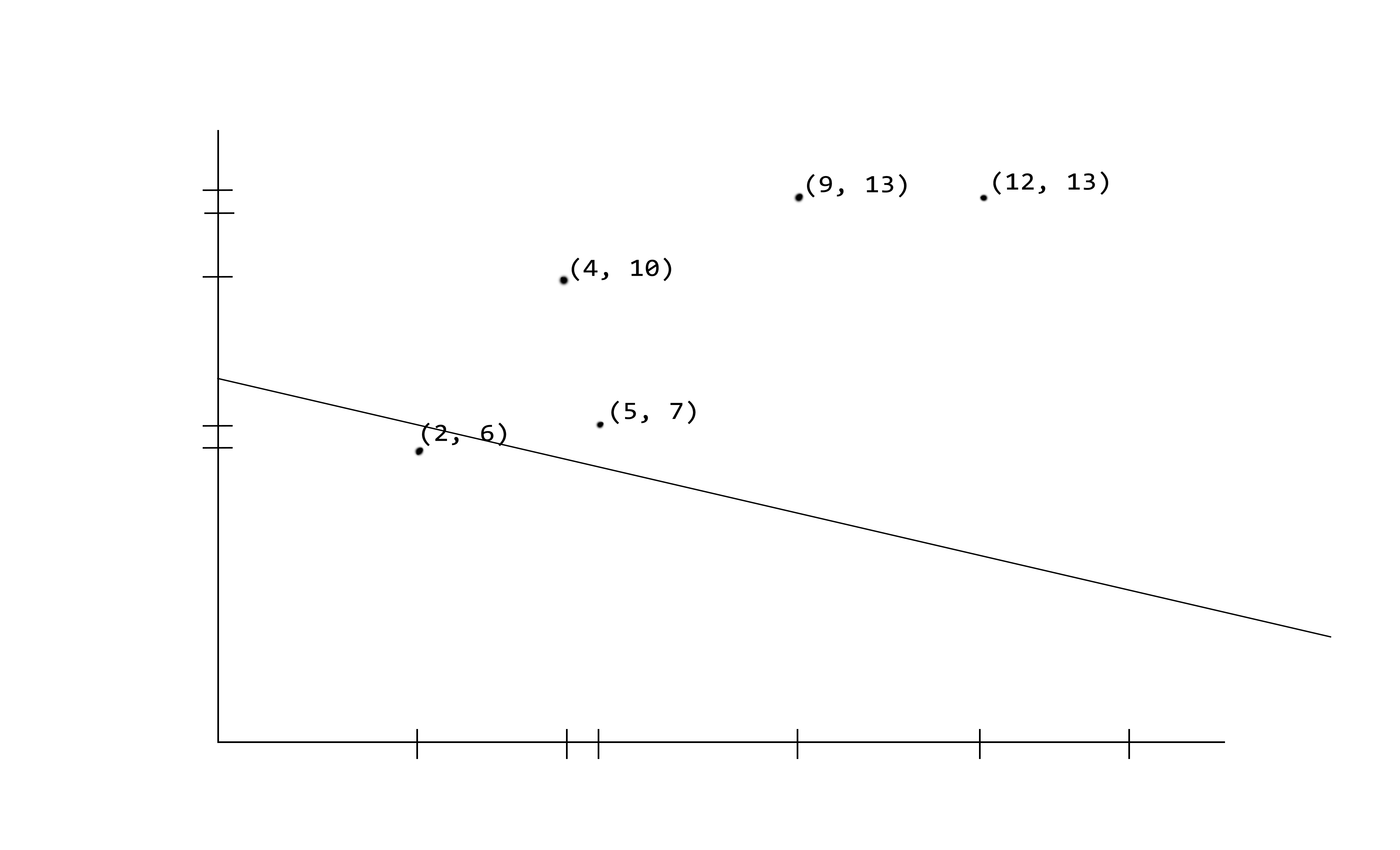
And in any machine learning/artificial intelligence system/model there exists always the data it was trained on. But how would such a simple formula we defined above have something to do with the most intelligent systems we have today? The answer is simple we feed the so called data for training a machine learning algorithm then it so called learns the mapping between the input we fed and the output variables so it can classify or predict a novel output   given an unseen input .

Now take these set of coordinates for example, the set of coordinates in the right hand-side represent our training data that we feed to our machine learning algorithm, and the set of coordinates albeit the same as the former has a newly added data point or coordinate, this newly added data point is what we call our testing data point, or coordinates having an input that our algorithm "has not seen" or has not been trained on and where it corresponds a novel output . When only our training data points are plotted we have the following graph:



With that said the so called mapping we have mentioned that we need to "learn" is none other than the slope intercept formula or more specifically its coefficients and which we learned in high school and have defined earlier (albeit a slightly more generalized version which we will explore later in the next part of this article), which would also be called, in the context of machine learning, our hypothesis function. But how exactly can we learn this mapping? Well if we recall our slope intercept formula when we plug in random input 's and calculate its corresponding output we are eventually able to trace these coordinates and draw a line.

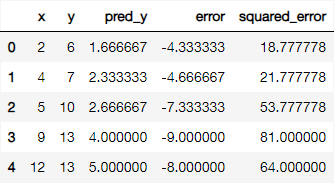


Of course because the independent variable is already used to take in our input values, how exactly do we manipulate this line such that we are able to change how it is oriented, whether in a horizontal, vertical, diagonal manner, or positioned in the upper part or lower part of the Cartesian plane? The answer is yet again rather simple…to simply change the coefficients in our slope-intercept function and which if we recall the times when we were plotting the graph of this function, changing or changes the angle of our line, and changing or moves the line either downwards or upwards since it is after all our y-intercept, which allows our line to move along the y-axis. In the left-hand graph we see, we could describe the lines slope intercept equation as .

So far we've learned that we need to "learn" the mapping between our inputs and outputs but why do we exactly need to learn this mapping? Well…the whole point of machine learning and more so deep learning is that given only mere data points or in this case our set of and coordinates, we use our slope intercept function and try to as much as possible predict a set of 's, using only our input 's, to be as close/approximating to or altogether the same as, our original values. Now you may think that simply just programmatically predicting a value the same as our original value given is the way to go, e.g. if given 2 predict 6, if given 5 predict 7, if given 4 predict 10, and so on. This would be an undeniably naïve and worthless approach at best, since we would in theory have to create so many conditional statements to take into account all possible input 's and their corresponding values, and even more so if another data point or a set of data points are added because e.g. given 2.2 predict 6.134 and so on. And so this is the whole reason for machine learning/deep learning, that instead of explicitly giving a computer/machine a set of instructions or an algorithm to perform a task, we use the power of mathematics particularly linear algebra, calculus, and statistics in order for the machine to be able to learn this "algorithm" in order to be perform and be better at a certain task.

Going back, in reality our data would not merely range in 5 to 10 coordinates/data points but rather in the hundreds, thousands, tens of thousands, and even millions. And so sometimes I think we would not have the luxury of using a tool in plotting every single data point just to see if we can directly figure out if we can position the line by testing out trial and error different coefficients such that the predicted values of our hypothesis function or in this case our slope intercept formula using certain coefficients would result in values approximating if not altogether the same as our original values. And so we would completely start at random coefficients initially as a start in our training process. Take our previous slope intercept equation with the coefficients and , or , because the gauge if indeed our machine learning model is learning is seeing if the predicted y values of this function approximates or is altogether equal to the original y value.

By having an error value close/approximating or altogether zero, because this essentially indicates that the values we obtain or predict using our slope-intercept function is close/approximating or altogether the same as, our original values



We measure how large the error is by getting simply the difference of our predicted Y value to our real Y value,

One way we minimize this error is to move our line such that with each and every single data point we have our hypothesis function or slope intercept function going back to linear regression is able to get close to these data points as much as possible. And how do we move the line in the slope intercept formula? By simply changing the coefficients, which in the simplest form of the slope intercept formula this would be our y-intercept and our slope . And what might the minimization of the error and changing of the coefficients entail, you may ask? Well this is the heart and soul of all machine learning algorithms so to speak. To "learn" or rather optimize these coefficients "m" and "b" and "move" the line over and over or for a certain number of iterations, such that the error we calculate for all training data points is minimal.

So far we've already established that we define use a hypothesis function to fit its graph hopefully to all our training data points. The second was measuring how far our training data points are to the plotted graph of the hypothesis function. And the third and final point was that we ought to "learn" or rather optimize the coefficients "m" and "b" such that we are able to minimize this error by moving the graph of our hypothesis function closer and mimicking the pattern of our training data points

established that we optimize our coefficients

Something to do with using the line as an objective function that we optimize

When we recall in highschool given a points coefficients m and b we can essentially plot the

Obviously when we set the coefficients to different values we would get a differently positioned line every time, and when we give it an input x it always corresponds to an output y

What we want to do is position this line such that given the inputs x which are our points

After one variable linear regression we now move on to multivariate linear regression

Which uses matrix operations

And as we venture further into the likewise vast world of machine learning and all of its different domains and sub disciplines, we ought to do our very best to muster up the courage within ourselves to face that which we fear the most, not only the Mathematics in the field of AI or Machine Learning but in life as well.