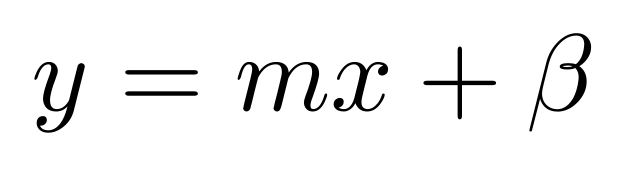
A beginner Machine Learning Engineer's biggest fear: Mathematics (Part I)

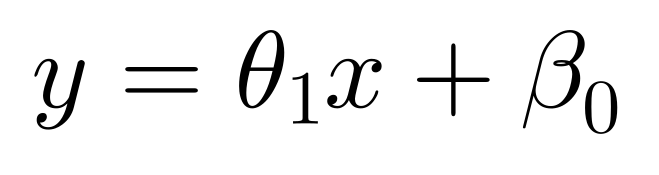
…and for good reason, because it is after all a daunting thing to learn in of itself, given its immensely vast disciplines. But fear not, I believe firmly that one need not learn every ins, outs, and backwards of this seemingly dreadful thing we call mathematics, not at all, not for machine learning 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…

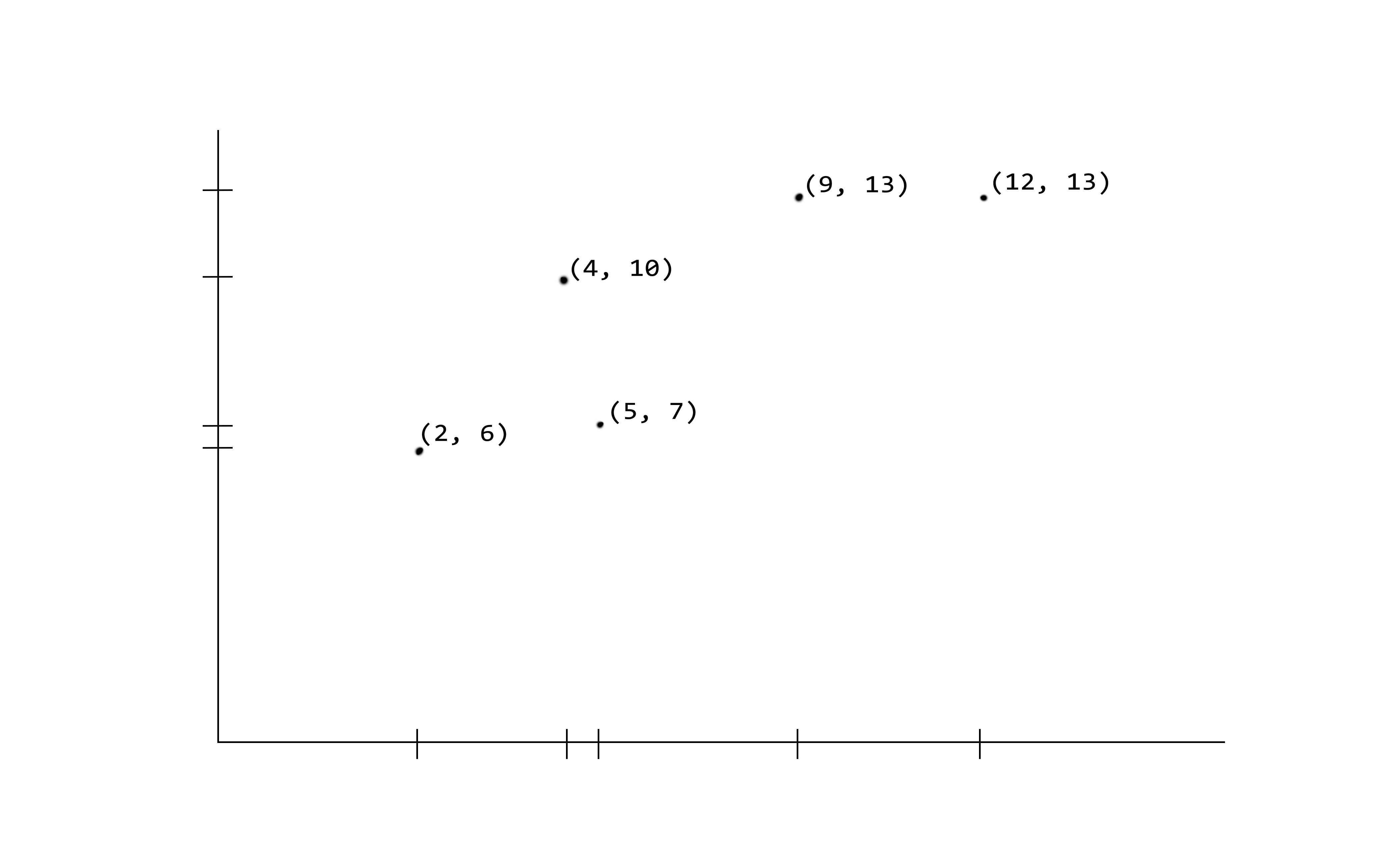


Where (theta subscript 1) is denoted as our first coefficient or what we previously knew as our slope, and (beta subscript 0) is denoted as our bias coefficient or what we previously knew as our y-intercept.

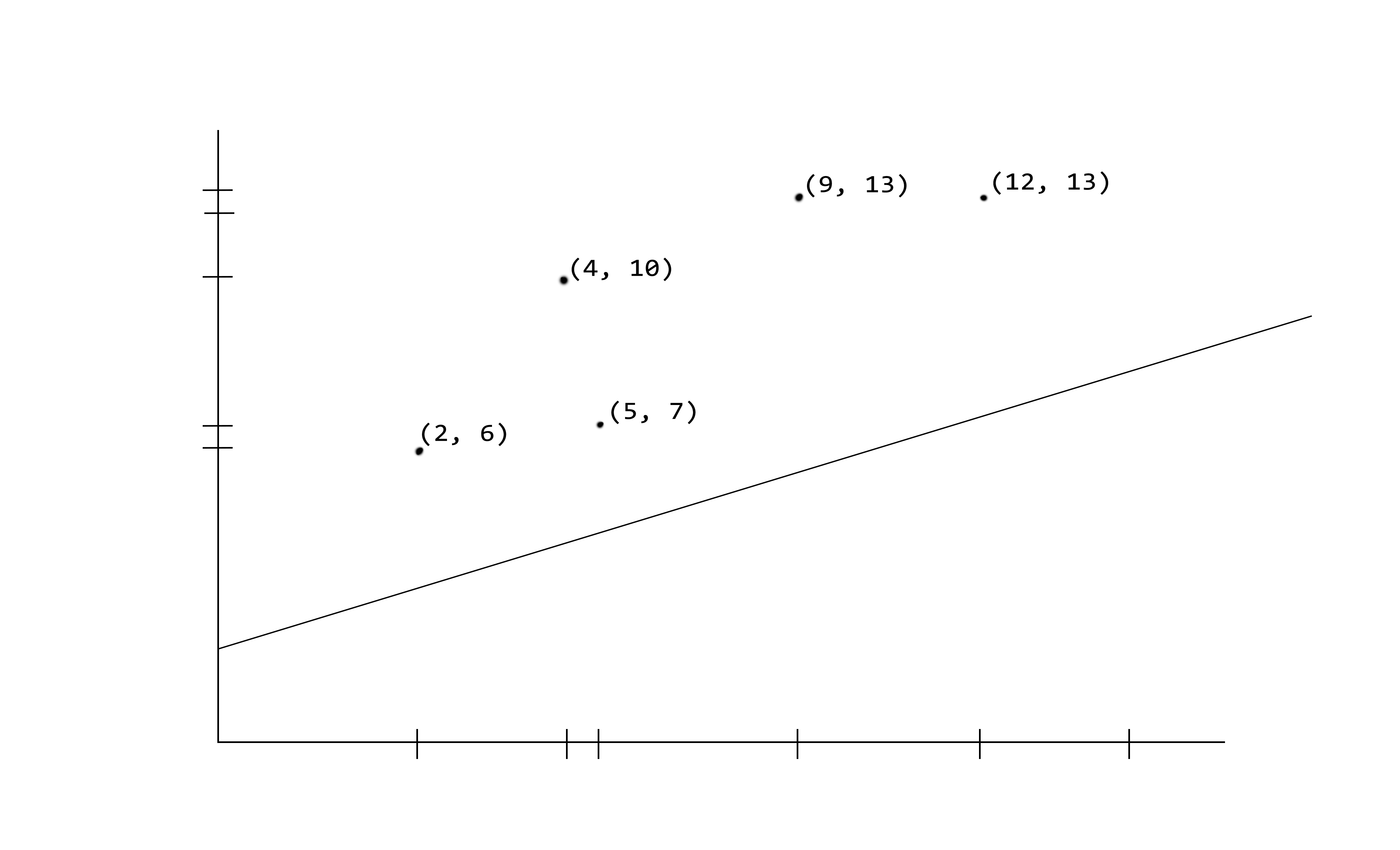
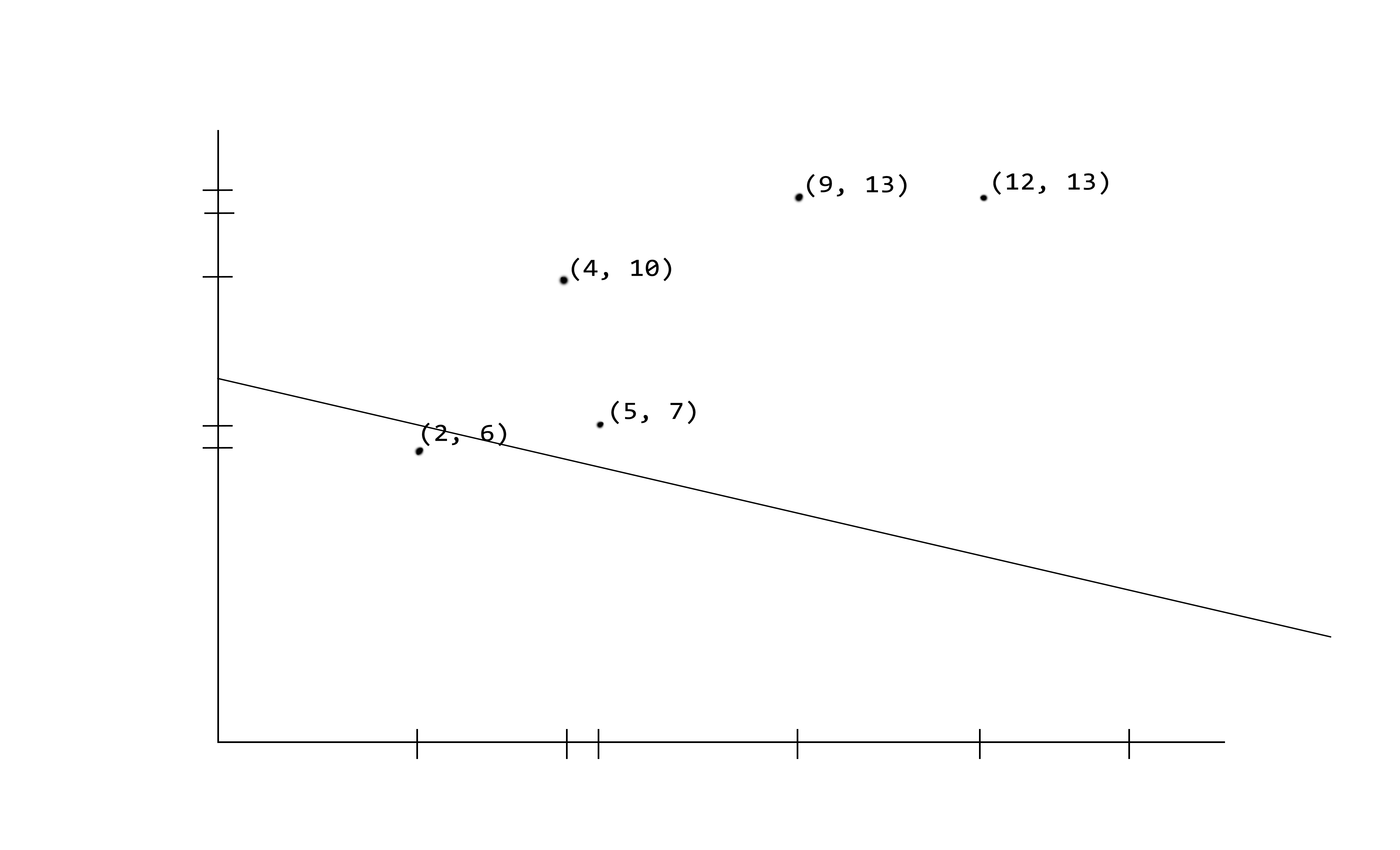
And in any machine learning/artificial intelligence system/model there exists always the data it was trained on. But how would such a simple formula we defined above have something to do with the most intelligent systems we have today? The answer is simple we feed the so called data for training a machine 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:



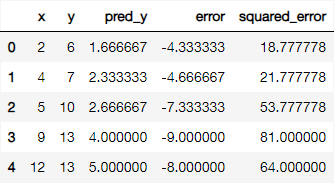
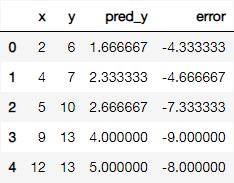
With that said the so called mapping we have mentioned that we need to "learn" is none other than the slope intercept formula or more specifically its coefficients and which we learned in high school and have defined earlier (albeit a slightly more generalized version which we will explore later in the 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.



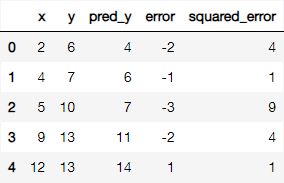
Of course because the independent variable is already used to take in our input values, how exactly do we manipulate this line such that we are able to change how it is oriented, whether in a horizontal, vertical, diagonal manner, or positioned in the upper part or lower part of the Cartesian plane? 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.

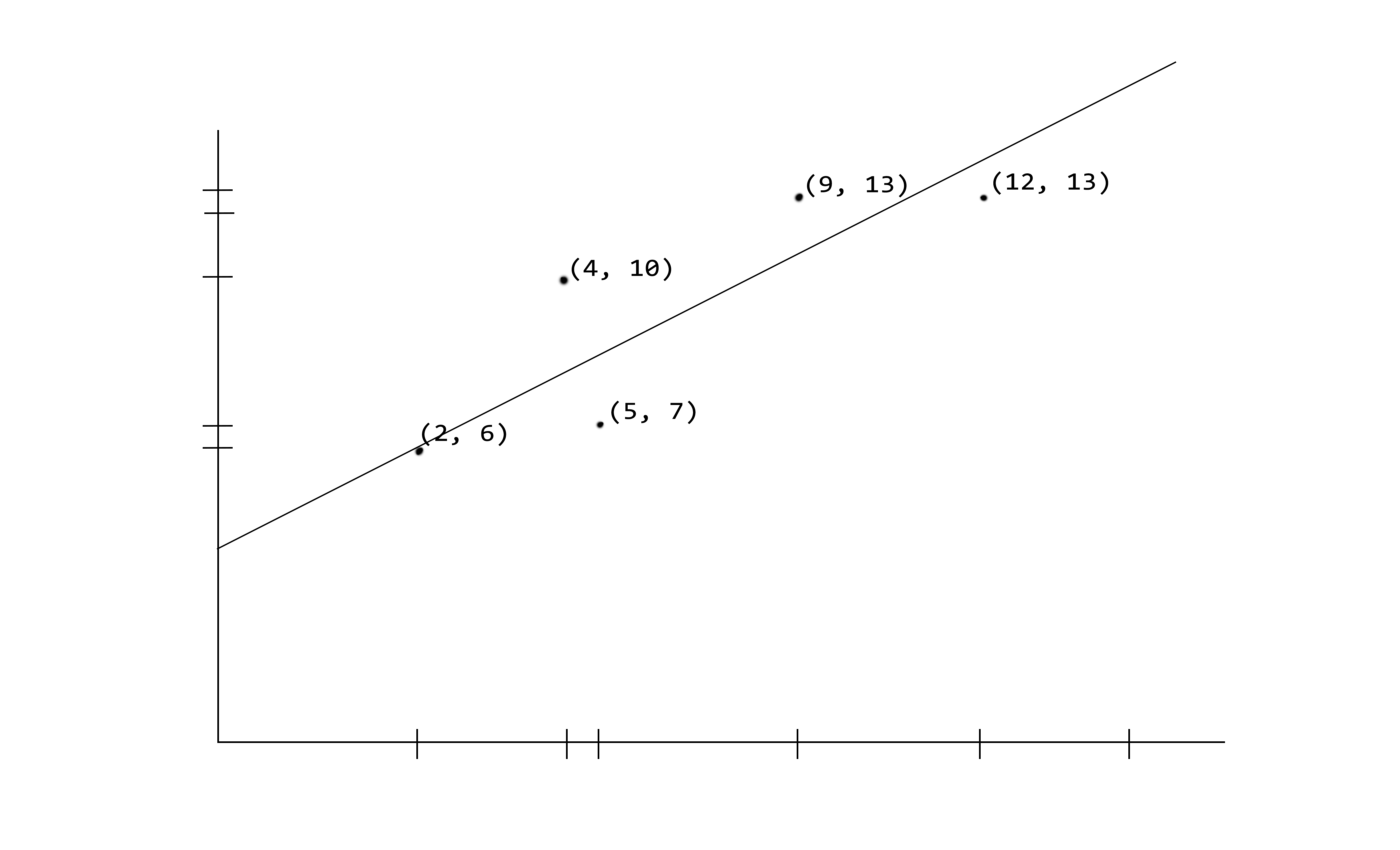
Going back, in reality our data would not merely range in 5 to 10 coordinates/data points but rather in the hundreds, thousands, tens of thousands, and even millions. And so sometimes I think we would not have the luxury of using a tool in plotting every single data point just to see if we can directly 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 values of this function approximates or is altogether equal to the original value using the coefficients we have. When we do input our training 's to this function with these coefficients, we would get the following predicted values



And the way to measure if our model is has learned is by computing the difference between each data points original value and the predicted value. As we can see for each coordinate or data point, we have the following errors, but how do we know whether such errors are good or bad? Well we can do a calculation yet again using our slope intercept equation but for different coefficients let's say and denoting our and coefficients. Now because our error may contain negative values we can transform them into positive values by squaring them, which as we know multiplying two negative values by each other results always in a positive value, which make the errors for our data points much easier to quantify and measure; this would now be called our squared error.



If we observe closely the two tables that used different coefficients, we can see that our squared error for the latter is lower, which is a remarkable improvement from the equation that used coefficients and , thus making our slope intercept equation . And why did we achieve this? Because we chose or optimized our coefficients in such a way that our predicted values were now closer to the original values. And if we plotted the line of the slope intercept function with these coefficients we would see that indeed our line using coefficients and is closer to our data points. Indicating that our predicted values even when plotted in a Cartesian plane approximate the original values in our dataset.



However as I said sometimes depending on our dataset we may not have the luxury of visualizing our data points to see if our line has indeed "fit" or approximated our data points, luckily another and even a much better way to measure the error not only for a single data point but for ALL our data points is to use another very important function, which pretty much everyone starting out in the field machine learning will learn, called the loss/cost function, which measures the total error accrued across all data points. Now this may vary depending on our dataset and the algorithm we will use but in our simplest case which is a Linear Regression model, its cost function called the mean squared error is defined as:

Where the squared errors for each training data point we have calculated earlier we now just simply take the summation and then average by the total number of all training data points (multiplied by in other definitions of the mean square error cost function such as this). And when we calculate both the mean squared error using coefficients and and coefficients and

We can see that indeed the latter which results in a MSE of , is better than the former which resulted in , and by having an MSE value close/approximating to zero as iterations go by like we did in these 2 iterations 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, meaning that our model overall has now "learned" to predict values that are close to the original values in our dataset given values.

And these are the basic foundational concepts needed in order to eventually develop more complex machine learning systems/models, because the goal is to essentially minimize the error across a defined number of iterations. And what might the minimization of the error and changing of the coefficients entail, you may ask yet again? Well this is the heart and soul of all machine learning algorithms so to speak. To "learn", change, or rather optimize these coefficients "" and "" and "move" the line over and over or for a certain number of iterations through an algorithm such that we are able to minimize each error for each iteration ultimately indicating that our predicted values approximates the original values in our dataset.

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 "" and "" 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. In the next part of this article we will dive further and discuss the algorithm needed in order to automatically "learn" our coefficients m and b for our simple Linear Regression model, involving the core concepts you will more than likely use for the rest of your career in the field machine learning: matrix operations, partial differentiation, and basic data normalization and standardization.

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.