

Linear Regression

Our today's topic is Linear regression and we will try to understand it with stochastic gradient descent

In this tutorial we will try to cover following topics

- The form of the Simple Linear Regression model.
- The difference between gradient descent and stochastic gradient descent
- How to use stochastic gradient descent to learn a simple linear regression model.

What is linear regression:

As such, linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm.

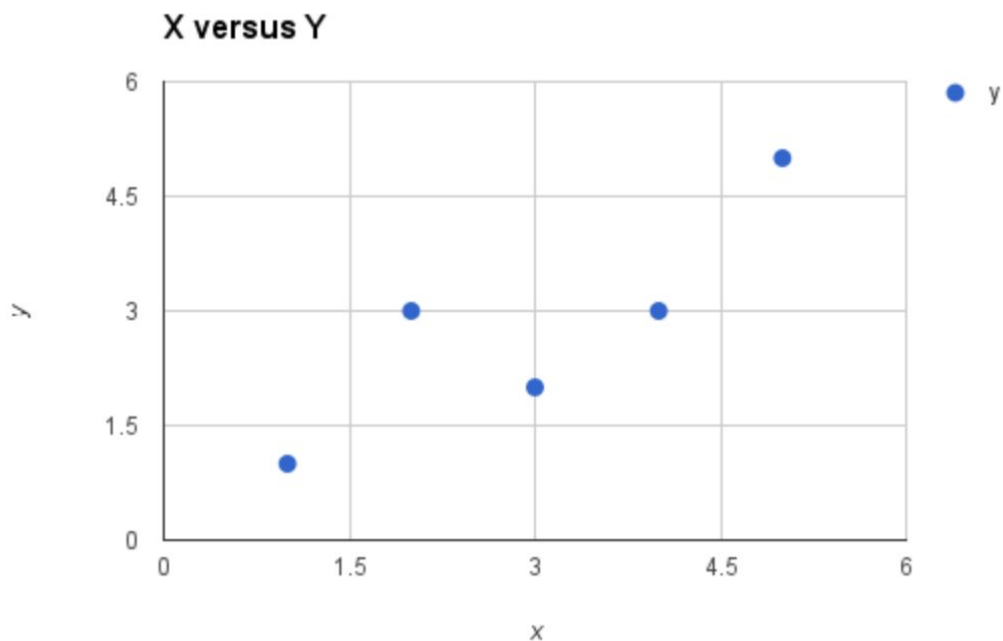
In machine learning most import thing to discuss anything is data set. Now first we will be having a look to our dataset.

Tutorial Data Set:

The data set we are using is completely made up.

Here is the raw data. The attribute x is the input variable and y is the output variable that we are trying to predict. If we got more data, we would only have x values and we would be interested in predicting y values.

x	y
1	1
2	3
4	3
3	2
5	5



We can see the relationship between x and y looks kind-of linear. As in, we could probably draw a line somewhere diagonally from the bottom left of the plot to the top right to generally describe the relationship between the data. This is a good indication that using linear regression might be appropriate for this little dataset.

Simple Linear Regression:

When we have a single in-put attribute (x) and we want to use linear regression, this is called simple linear regression.

With simple linear regression we want to model our data as follows:

$$y = B0 + B1 * x$$

This is a line where y is the output variable we want to predict, x is the input variable we know and B0 and B1 are coefficients we need to estimate.

B0 is called the intercept because it determines where the line intercepts the y axis. In machine learning we can call this the bias, because it is added to offset all predictions

that we make. The B1 term is called the slope because it defines the slope of the line or how x translates into a y value before we add our bias.

The model is called Simple Linear Regression because there is only one input variable (x). If there were more input variables (e.g. x1, x2, etc.) then this would be called multiple regression.

Stochastic Gradient Descent:

Gradient Descent is the process of minimizing a function by following the gradients of the cost function (At the end of tutorial).

This involves knowing the form of the cost as well as the derivative so that from a given point you know the gradient and can move in that direction, e.g. downhill towards the minimum value.

In Machine learning we can use a similar technique called stochastic gradient descent to minimize the error of a model on our training data.

The way this works is that each training instance is shown to the model one at a time. The model makes a prediction for a training instance, the error is calculated and the model is updated in order to reduce the error for the next prediction.

This procedure can be used to find the set of coefficients in a model that result in the smallest error for the model on the training data. Each iteration the coefficients, called weights (w) in machine learning language are updated using the equation:

$$w = w - \alpha * \text{delta}$$

Where w is the coefficient or weight being optimized, alpha is a learning rate that you must configure (e.g. 0.1) and gradient is the error for the model on the training data attributed to the weight.

Simple Linear Regression with Stochastic Gradient Descent:

The coefficients used in simple linear regression can be found using stochastic gradient descent.

Linear regression is a linear system and the coefficients can be calculated analytically using linear algebra. Stochastic gradient descent is not used to calculate the coefficients for linear regression in practice (in most cases).

Linear regression does provide a useful exercise for learning stochastic gradient descent which is an important algorithm used for minimizing cost functions by machine learning algorithms.

As stated above, our linear regression model is defined as follows:

$$y = B0 + B1 * x$$

Gradient Descent Iteration #1:

Let's start with values of 0.0 for both coefficients.

$$\begin{aligned} B0 &= 0.0 \\ B1 &= 0.0 \\ y &= 0.0 + 0.0 * x \end{aligned}$$

We can calculate the error for a prediction as follows:

$$\text{error} = p(i) - y(i)$$

Where $p(i)$ is the prediction for the i 'th instance in our dataset and $y(i)$ is the i 'th output variable for the instance in the dataset.

We can now calculate the predicted value for y using our starting point coefficients for the first training instance:

$$\begin{aligned} x &= 1, \quad y = 1 \\ p(i) &= 0.0 + 0.0 * 1 \\ p(i) &= 0 \end{aligned}$$

Using the predicted output, we can calculate our error:

$$\begin{aligned} \text{error} &= 0 - 1 \\ \text{error} &= -1 \end{aligned}$$

We can now use this error in our equation for gradient descent to update the weights. We will start with updating the intercept first, because it is easier. We can say that B0 is accountable for all of the error. This is to say that updating the weight will use just the error as the gradient. We can calculate the update for the B0 coefficient as follows:

$$B0(t+1) = B0(t) - \alpha * error$$

Where B0(t+1) is the updated version of the coefficient we will use on the next training instance, B0(t) is the current value for B0 alpha is our learning rate and error is the error we calculate for the training instance. Let's use a small learning rate of 0.01 and plug the values into the equation to work out what the new and slightly optimized value of B0 will be:

$$B0(t+1) = 0.0 - 0.01 * -1.0$$

$$B0(t+1) = 0.01$$

Now, let's look at updating the value for B1. We use the same equation with one small change. The error is filtered by the input that caused it. We can update B1 using the equation:

$$B1(t+1) = B1(t) - \alpha * error * x$$

Where B1(t+1) is the update coefficient, B1(t) is the current version of the coefficient, alpha is the same learning rate described above, error is the same error calculated above and x is the input value.

We can plug in our numbers into the equation and calculate the updated value for B1:

$$B1(t+1) = 0.0 - 0.01 * -1 * 1$$

$$B1(t+1) = 0.01$$

We have just finished the first iteration of gradient descent and we have updated our weights to be B0=0.01 and B1=0.01. This process must be repeated for the remaining 4 instances from our dataset.

One pass through the training dataset is called an epoch.

Gradient Descent Iteration #20:

Let's jump ahead.

You can repeat this process another 19 times. This is 4 complete epochs of the training data being exposed to the model and updating the coefficients.

Here is a list of all of the values for the coefficients over the 20 iterations that you should see:

B0	B1
0.01	0.01
0.0397	0.0694
0.066527	0.176708
0.08056049	0.21880847
0.1188144616	0.410078328
0.1235255337	0.4147894001
0.1439944904	0.4557273134
0.1543254529	0.4970511637
0.1578706635	0.5076867953
0.1809076171	0.6228715633
0.1828698253	0.6248337715
0.1985444516	0.6561830242
0.2003116861	0.6632519622
0.1984110104	0.657549935
0.2135494035	0.7332419008
0.2140814905	0.7337739877
0.2272651958	0.7601413984
0.2245868879	0.7494281668
0.219858174	0.7352420252
0.230897491	0.7904386102

I think that 20 iterations or 4 epochs is a nice round number and a good place to stop.

You could keep going if you wanted.

Your values should match closely, but may have minor differences due to different spreadsheet programs and different precisions. You can plug each pair of coefficients back into the simple linear regression equation. This is useful because we can calculate a prediction for each training instance and in turn calculate the error.

Below is a plot of the error for each set of coefficients as the learning process unfolded. This is a useful graph as it shows us that error was decreasing with each iteration and starting to bounce around a bit towards the end.



You can see that our final coefficients have the values $B_0=0.230897491$ and $B_1=0.7904386102$

Let's plug them into our simple linear Regression model and make a prediction for each point in our training dataset.

x	y	prediction
1	1	0.9551001992
2	3	1.690342224
4	3	3.160826275
3	2	2.42558425
5	5	3.8960683

We can plot our dataset again with these predictions overlaid (x vs y and x vs prediction). Drawing a line through the 5 predictions gives us an idea of how well the model fits the training data.

Cost Function:

We can measure the accuracy of our hypothesis function by using a cost function. This takes an average difference (actually a fancier version of an average) of all the results of the hypothesis with inputs from x's and the actual output y's.

Making Predictions with Linear Regression:

Given the representation is a linear equation, making predictions is as simple as solving the equation for a specific set of inputs.

Let's make this concrete with an example. Imagine we are predicting weight (y) from height (x). Our linear regression model representation for this problem would be:

$$\begin{aligned} y &= B_0 + B_1 * x_1 \\ &\text{or} \\ \text{weight} &= B_0 + B_1 * \text{height} \end{aligned}$$

Where B0 is the bias coefficient and B1 is the coefficient for the height column. We use a learning technique to find a good set of coefficient values. Once found, we can plug in different height values to predict the weight.

For example, let's use B0 = 0.1 and B1 = 0.05. Let's plug them in and calculate the weight (in kilograms) for a person with the height of 182 centimeters.

$$\begin{aligned} \text{weight} &= 0.1 + 0.05 * 182 \\ \text{weight} &= 91.1 \end{aligned}$$

You can see that the above equation could be plotted as a line in two-dimensions. The B0 is our starting point regardless of what height we have. We can run through a bunch of heights from 100 to 250 centimeters and plug them to the equation and get weight values, creating our line.

Preparing Data For Linear Regression:

Linear regression is been studied at great length, and there is a lot of literature on how your data must be structured to make best use of the model.

As such, there is a lot of sophistication when talking about these requirements and expectations which can be intimidating. In practice, you can use these rules more as rules of thumb when using Ordinary Least Squares Regression, the most common implementation of linear regression.

Try different preparations of your data using these heuristics and see what works best for your problem.

Linear Assumption:

Linear regression assumes that the relationship between your input and output is linear. It does not support anything else. This may be obvious, but it is good to remember when you have a lot of attributes. You may need to transform data to make the relationship linear (e.g. log transform for an exponential relationship).

Remove Noise:

Linear regression assumes that your input and output variables are not noisy. Consider using data cleaning operations that let you better expose and clarify the signal in your data. This is most important for the output variable and you want to remove outliers in the output variable (y) if possible.

Remove Collinearity:

Linear regression will over-fit your data when you have highly correlated input variables. Consider calculating pairwise correlations for your input data and removing the most correlated.

Gaussian Distributions:

Linear regression will make more reliable predictions if your input and output variables have a Gaussian distribution. You may get some benefit using transforms (e.g. log or BoxCox) on your variables to make their distribution more Gaussian looking.

Rescale Inputs:

Linear regression will often make more reliable predictions if you rescale input variables using standardization or normalization.