# Linear Regression Tutorial Using Gradient Descent for Machine Learning

by Jason Brownlee on March 30, 2016 in Understand Machine Learning Algorithms



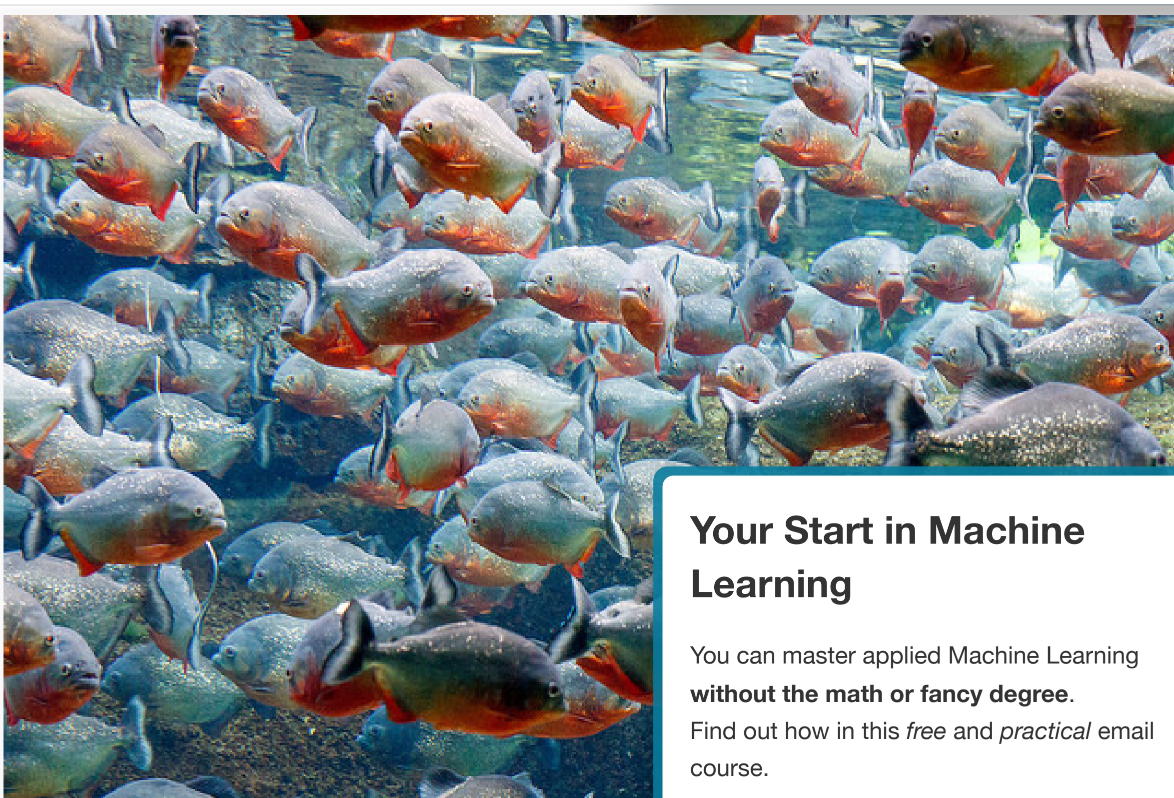
Stochastic Gradient Descent is an important and widely used algorithm in machine learning.

In this post you will discover how to use Stochastic Gradient Descent to learn the coefficients for a simple linear regression model by minimizing the error on a training dataset.

After reading this post you will know:

- 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.

Let's get started.



Linear Regression Tutorial Using Gradient D
Photo by Stig Nygaard, some



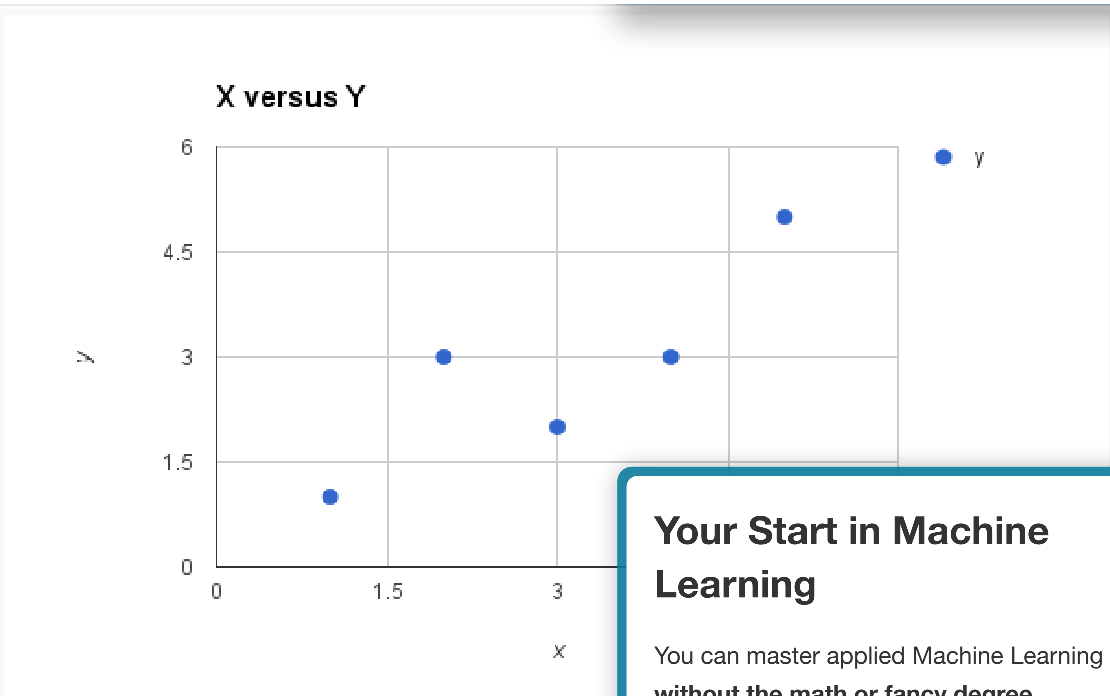
## Tutorial Data Set

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
```

Below is a simple scatter plot of x versus y.




Plot of the Dataset for Simple L



We can see the relationship between x and y looks kind-of linea re diagonally from the bottom left of the plot to the top right to gen . This is a good indication that using linear regression might be a



## 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 foll

This involves knowing the form of the cost as well as the derivat ient and can move in that direction, e.g. downhill towards the minim

In Machine learning we can use a similar technique called stoch model on our training data.

The way this works is that each training instance is shown to th prediction for a training instance, the error is calculated and the the next prediction.

This procedure can be used to find the set of coefficients in a m l on the training data. Each iteration the coefficients, called weights ing the equation:



w = w – alpha * 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.

$$B0 = 0.0$$

$$B1 = 0.0$$

$$y = 0.0 + 0.0 * x$$

We can calculate the error for a prediction as follows:

$$error = p(i) – y(i)$$

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 he predicted value for y using our starting

$$x=1, y=1$$

$$p(i) = 0.0 + 0.0$$

$$p(i) = 0$$

Using the predicted output, we can calculate our error:

$$error = 0 –$$

$$error = -1$$



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 complet … he model and updating the coefficients.

Here is a list of all of the values for the coefficients over the 20 i…

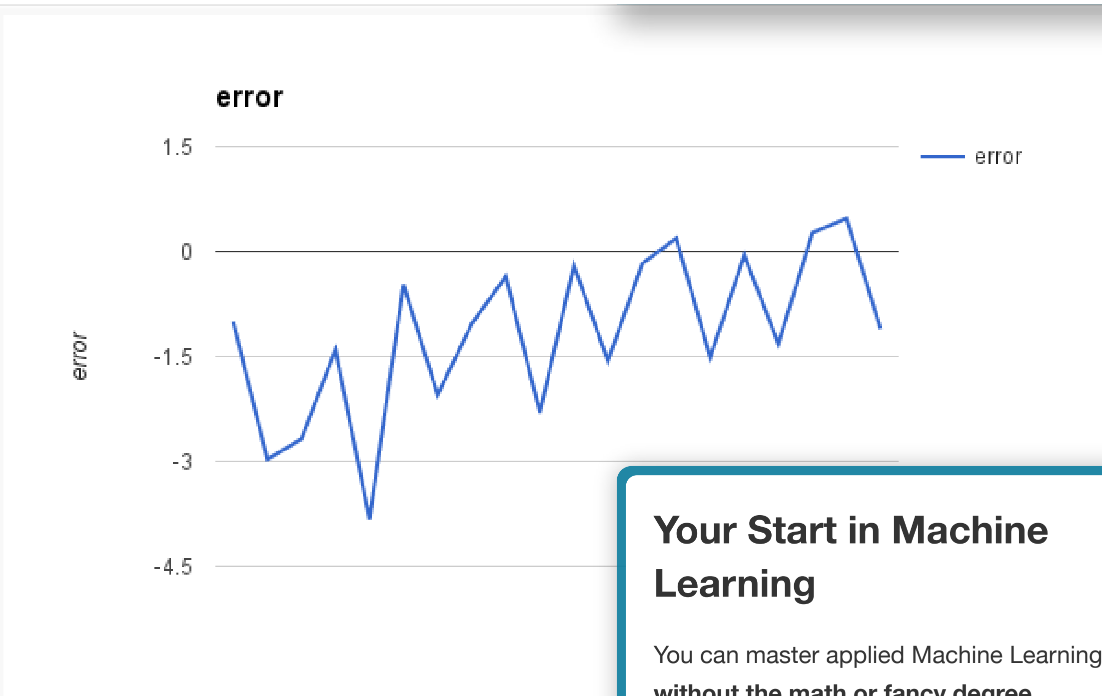
```
B0	B1
0.01	0.01
0.0397	0.0694
0.066527	0.176708
0.08056049	0.21880847
0.1188144616	0.410078328
0.1235255337	0.4147894001
0.1439944904	0.4557273134
0.1543254529	0.4970511637
0.1578706635	0.5076867953
0.1809076171	0.6228715633
0.1828698253	0.6248337715
0.1985444516	0.6561830242
0.2003116861	0.6632519622
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.




Linear Regression Gradient Descen

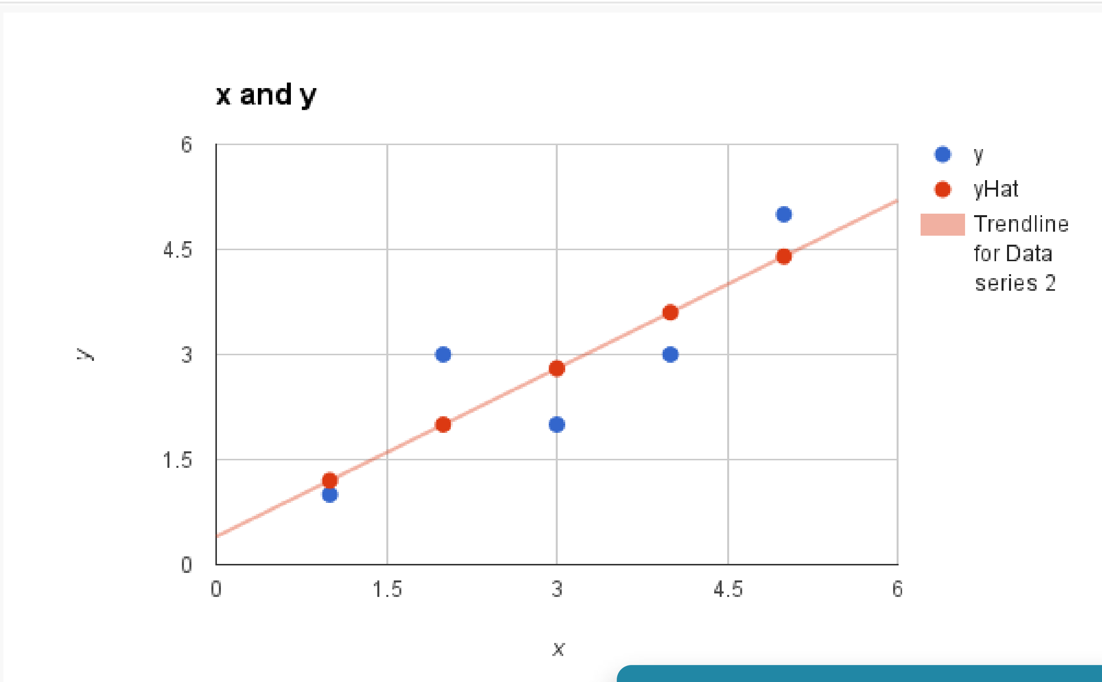


You can see that our final coefficients have the values B0=0.230

Let's plug them into our simple linear Regression model and ma ... set.

```
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.




Simple Linear Regression Model

## Summary

In this post you discovered the simple linear regression model and how to train it using stochastic gradient descent.

You work through the application of the update rule for gradient descent. You also learned how to make predictions with a learned linear regression model.

Do you have any questions about this post or about simple linear regression with stochastic gradient descent? Leave a comment and ask your question and I will do my best to answer it.



### About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.
View all posts by Jason Brownlee →

< Simple Linear Regression Tutorial for Machine Learning

Logistic Regression for Machine Learning >

## 45 Responses to *Linear Regression Tutorial Using Gradient Descent for Machine Learning*

**Tadele** November 10, 2016 at 7:50 pm #

REPLY

God Bless you and your family . Your duties was every bright so keep it up.

My loard jesus bless your mind and your duties. I don't have more words.

**Jason Brownlee** November 11, 2016 at 10:00 am #

REPLY

Thanks Tadele.

**effa** November 15, 2016 at 2:53 pm #

u explain it very well. thank you so much.

**Jason Brownlee** November 16, 2016 at 9:24 am #

Thanks for your kind words effa. I'm glad you found



**sphurti** November 28, 2016 at 4:46 am #

I am getting somewhat confused between epoch and iteration.Is epoch or iteration depends on number of observations in training dataset?

**sphurti** November 28, 2016 at 5:06 am #

REPLY

I am having dataset as (year,cost)= [(2005,40.15), (2006,49.8), (2007,60), (2008,75), (2009,83), (2010,90), (2011,111), (2012,128). (2013,128), (2014,138), (2015,160),(2016,175) and I want to apply linear regression with stochastic gradient descent.what epoch or iteration should i set?

**Jason Brownlee** November 28, 2016 at 8:46 am #

REPLY

One epoch is one run through the entire training dataset.

An iteration may be an epoch or it may be an update for one training observation (one row in the training data), depending on the context (training iteration vs update iteration).

**Alex** December 5, 2016 at 11:13 pm #

REPLY

Hi Jason, i am investgating stochastic gradient descent for logistic regression with more than 1 response variable and am struggling.

I have tried this using the same formula but with a different calculation for the error term [error=Y-(1/1+exp(-BX))]

I have plugged this into the equations you have provided but the coefficients to not seem to be converging. Is there anything that i am missing? A

**Jason Brownlee** December 6, 2016 at 8:25 am #

REPLY

See this tutorial Alex:
http://machinelearningmastery.com/implement-logistic-regre



**Serb** December 25, 2016 at 7:33 am #

Hi Jason,

where is the parameter m (number of training examples) in updat

B0(t+1) = B0(t) – alpha / m * error
B1(t+1) = B1(t) – alpha / m * error * x

**Jason Brownlee** December 26, 2016 at 7:43 am #

I'm not sure what you mean Serb. There is no "m" in the above equations.

**Serb** December 26, 2016 at 8:28 am #

REPLY

Yes, i see that there is no m, but it should be there. Since the cost function is defined as follows:

J(B0, B1) = 1/(2*m) * (p(i) – y(i))^2

in order to determine the parameters B0 and B1 it is necessary to minimize this function using a gradient descent and find partial derivatives of the cost function with respect to B0 and B1. At the end you get equations for B0 and B1 where there is "m".

**Daniel Deychakiwsky** January 27, 2017 at 4:02 am #

REPLY

Jason,

You mention these weight updating equations:

B0(t+1) = B0(t) – alpha / m * error
B1(t+1) = B1(t) – alpha / m * error * x

B0 representing the slope of our to-be regression line and B1 the intercept.

In other tutorials I see people (https://www.youtube.com/watch?v=JsX0D92q1EI&t=16s) multiplying x into the slope weight update calculation and not the intercept like so:

B0(t+1) = B0(t) – alpha / m * error * x
B1(t+1) = B1(t) – alpha / m * error

Can you explain if this is incorrect or what I've mistaken?

**Jason Brownlee** January 27, 2017 at 12:20 pm #

REPLY

Hi Daniel,

The update equations used in this post are based on those p
Modern Approach", section 18.6.1 Univariate linear regressio

I cannot speak for the equations in the youtube video.

**Charles M.** November 14, 2017 at 9:28 pm #

Hi Daniel,

I believe you might be mixing up stochastic and batch gradie

In batch gradient descend you calculate the total error for all
examples for a 'mean error'.

On the other hand, in stochastic gradient descend, as in this article, you tackle one example at a time, so no need to calculate a mean by diving with the number of examples.

I hope this helps.

**Jason Brownlee** November 15, 2017 at 9:51 am #

REPLY

Also this post might help clear thing up:
https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

**Kevin Mathew** June 2, 2018 at 6:53 pm #

REPLY

Actually the m isn't necessary. It all depends upon the learning rate chosen the whole thing (alpha / m) may be regarded as a single constant. The point is, the learning rate is all the matters, the m is just another constant.

**Akash** February 11, 2017 at 9:46 pm #

REPLY

Can you make a similar post on logistic regression where we could get to actually see some interations of the gradient descent?

Ty.

**Jason Brownlee** February 12, 2017 at 5:34 am # REPLY

Hi Akash,

Here is a tutorial for Logistic Regression with SGD:
http://machinelearningmastery.com/implement-logistic-regression-stochastic-gradient-descent-scratch-python/

**pavan** February 24, 2017 at 11:08 pm #

while i am trying to calculate the second example, i am
in the picture… please help me

**Jason Brownlee** February 25, 2017 at 5:57 am #

Remember to use the updated values from the last

I provide complete spreadsheet examples in my book:
http://machinelearningmastery.com/master-machine-learning



**pavan** February 24, 2017 at 11:09 pm # REPLY

i mean in the second iteration, i am getting the values as 0.03 and 0.06 instead of 0.0397 0.0694. Please help me ASAP

**Jenny Ischakov** April 23, 2017 at 10:47 pm # REPLY

Hi, what is the convergence point? How we understand that is the minimum point of the function? You stopped calculation with B0=0.230897491 and B1=0.7904386102. And then calculated predicted values. Can you please explain why it stopped on this B0 B1 values? It should be error=0? How we see it? Thank you!

**Jason Brownlee** April 24, 2017 at 5:35 am # REPLY

Great question.

You can evaluate the coefficients after each update to get an idea of the model error.

You can then use the model error to determine when to stop updating the model, such as when the error levels out and stops decreasing.

**Belal C** April 26, 2017 at 12:51 am #

REPLY

thanks for the post/tutorial Jason! In relation to Jenny's question on when does the model converge – in the plot you showed, error seems generally to be getting closer to zero per iteration (I guess we could say it is being minimized). I just wanted to confirm 2 points:

1 – the error you plotted is the model error (computed by evaluating the coefficients and comparing to the correct values) right?

2 – we often see graphs plotting error vs iteration with the error decreasing over time (http://i42.tinypic.com/dvmt6o.png); is error in your graph just plotted on a different scale? or why do most training graphs have error decreasing from a positive number to zero?

Would really appreciate some clarification, and thanks again for the tutorial!

Belal

**Jason Brownlee** April 26, 2017 at 6:23 am #

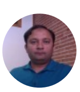
The error is calculated on the data and how

predictions.



**Vasu Sharma** April 27, 2017 at 3:48 pm #

Thanks a lot for such a nice post, I have doubt in calculation of y coordinate using B0 and B1 value. Acc. to me we are using y=B0+B1*x (to calculate y predicted), but considering B0=0.230897491 and B1=0.7904386102, answer of first instance(x=1,y=1) should be y(predicted)=1.0213361012 as (y=(0.230897491)+(0.7904386102)*1), but in your post it is 0.9551001992., so am I doing something wrong or intercepted it wrongly?
Guide me if I am missing somewhere?



**Azhaar** June 15, 2017 at 3:35 pm #

REPLY

I am stuck at same question.

**Athif** June 22, 2017 at 12:41 am #

REPLY

The values of B0=0.230897491 and B1=0.7904386102 are actually for the 21th iteration therefore its wrong.If you look at the graph of the values you would notice the 20th iteration values of B0 and B1 are 0.219858174 0.7352420252 respectively.Substitute the values gives the correct predictions(.95…). Small human error I guess 🙂

**Yasir** June 6, 2017 at 11:02 pm #

REPLY

If anyone wants to learn more about Simple linear regression, visit below link
http://yasirchoudhary.blogspot.in/2017/06/linear-regression.html?m=1

**Jason Brownlee** June 7, 2017 at 7:14 am # REPLY

Thanks for sharing.

**Athif** June 18, 2017 at 5:41 am # REPLY

Thanks a lot!
Finally understood this .

**Jason Brownlee** June 18, 2017 at 6:33 am #

I'm glad to hear it!

**Sonia arya** June 18, 2017 at 5:51 pm #

How can I find the value of theta 0 and theta 1 with the [illegible] able to fit the data perfectly..?

**Jason Brownlee** June 19, 2017 at 8:42 am # REPLY

Rarely do models fit the data perfectly unless the data was contrived.

Using a linalg approach will give a more robust estimate if the data can fit into memory.

A model is a tradeoff.

**Khushi** January 27, 2018 at 11:02 pm # REPLY

Is SGD the same as backpropagation? When classifying images into two categories (e.g. Cats and Dogs) is the model computing linear regression. If not, what would this be classified as?

**Jason Brownlee** January 28, 2018 at 8:24 am # REPLY

No, gradient descent is a search algorithm, backpropagation is a way of estimating error in a neural net.

**Eric** January 29, 2018 at 12:53 pm #

REPLY

Hi Dr. Brownlee: Definitely a great tutorial! I was able to reproduce the same results. Can this algorithm be modified for multiple parameters? I am trying to understand linear regression for more than one parameter and the tutorials I have found use excel or some other tool the black boxes the actual algorithm.

**Jason Brownlee** January 30, 2018 at 9:45 am #

REPLY

Yes, linear regression can have multiple inputs.



**Ged** March 19, 2018 at 10:27 pm #

Did I miss the derivative here?

**Jason Brownlee** March 20, 2018 at 6:19 am #

I do not cover the derivation.

**Bhagirath** March 29, 2018 at 7:41 am #

I have read the post, but I am not very clear about the difference between Gradient Descent and Stochastic Gradient Descent in this particular example. You have shown that in Stochastic Gradient Descent, we take one example at a time and update the coefficients. But what happens in case of Gradient Descent?

**Jason Brownlee** March 29, 2018 at 3:15 pm #

REPLY

Perhaps this post will make it clearer:
https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/

**INJETI MARUTHI PRASAD** August 23, 2018 at 9:44 pm #

REPLY

hi sir, im maruthi please let tell me step by step procedure.As a fresher i understand 50% only.

**Jason** April 25, 2018 at 9:48 am #

REPLY

Why do you choose not to square the sum of distances in the loss function? In my class we did everything similar to what you outlined except that part in order to make the function differentiable.

## Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT



**Welcome to Machine Learning Mastery!**

Hi, I'm Jason Brownlee, PhD
I write tutorials to help developers (*like you*) get results with machine learning.

Read More

**Understand How Algorithms Work**
No theory, just clear descriptions.

POPULAR


**How to Develop a Neural Machine Translation System from Scr**
JANUARY 10, 2018


**Difference Between Classification and Regression in Machine L**
DECEMBER 11, 2017


**How to Develop an N-gram Multichannel Convolutional Neural Network for Sentiment Analysis**
JANUARY 12, 2018


**So, You are Working on a Machine Learning Problem…**
APRIL 4, 2018


**Encoder-Decoder Models for Text Summarization in Keras**
DECEMBER 8, 2017


**How to Make Predictions with Keras**
APRIL 9, 2018


**11 Classical Time Series Forecasting Methods in Python (Cheat Sheet)**
AUGUST 6, 2018

### You might also like…

- How to Install Python for Machine Learning
- Your First Machine Learning Project in Python
- Your First Neural Network in Python
- Your First Classifier in Weka
- Your First Time Series Forecasting Project