Pattern Recognition

ECSE 4410/6410 CAPA

Fall 2022

Linear Regression +

Course Instructor - Thirimachos Bourlai

Please Check the 2022 Syllabus

OVERVIEW

Linear Regression +

- The representation and learning algorithms used to create a linear regression model
- How to best prepare your data when modeling using linear regression
- How to calculate a simple linear regression step-by-step
- How to make predictions on new data using your model
- A shortcut that greatly simplifies the calculation
- How stochastic GD can be used to search for the coefficients of a regression model
- How repeated iterations of GD can create an accurate regression model

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Simple Linear Regression

When we have a **single input** → we can use statistics to estimate the coefficients

Ordinary Linear Regression

- When we have **more than one input** -- we can use OLS to **estimate the values of the coefficients**
- The **OLS procedure** seeks to minimize the sum of the squared residuals
  - Given a regression line that crosses the data we
    - **Calculate** the distance from each data point to the regression line → square it → sum all squared errors → estimate if is minimum for that line and if not, we change the coefficients

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Gradient Descent

Used when we have either **a single input or more input**

Regularized Linear Regression

Regularization methods: used to **minimize** the sum of the squared error of the model on the training data and **reduce** the model complexity (size of the sum of all coefficients in the model).

Regularization procedures for LR:

- Lasso Regression: where Ordinary Least Squares is modified to also **minimize the absolute sum** of the coefficients (called **L1 regularization**).
- Ridge Regression: where Ordinary Least Squares is modified to also **minimize the squared absolute sum** of the coefficients (called **L2 regularization**).

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Lasso vs. Ridge Regularization

The key difference between these two is the penalty term.

Ridge regression adds "*squared magnitude*" of coefficient as penalty term to the loss function. Here the *highlighted* part represents L2 regularization element.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Cost function

Here, if *lambda* is zero then you can imagine we get back OLS. However, if *lambda* is very large then it will add too much weight and it will lead to under-fitting. Having said that it's important how *lambda* is chosen. This technique works very well to avoid over-fitting issue.

Lasso vs. Ridge Regularization

Lasso Regression (Least Absolute Shrinkage and Selection Operator) adds "*absolute value of magnitude*" of coefficient as penalty term to the loss function.

$$\sum_{i=1}^{n}(Y_i - \sum_{j=1}^{p} X_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Cost function

Again, if *lambda* is zero then we will get back OLS whereas very large value will make coefficients zero hence it will under-fit.

The **key difference** between these techniques is that Lasso shrinks the less important feature's coefficient to zero thus, removing some feature altogether. So, this works well for **feature selection** in case we have a huge number of features.

https://towardsdatascience.com/l1-and-l2-regularization-methods-ce25e7fc831c

Linear Regression: Data Preparation

Tasks

T#1: Linear Assumption

- LR assumes that the I/O linear relationship
- Does not support anything else - remember this 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).

T#2: Remove Noise

- LR assumes that your I/O variables are not noisy.
- **Need to be using data cleaning operations** in your input data → to impact the output variable (y)
- **Remove outliers in the output variable** (y) if possible

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Linear Regression: Data Preparation

Tasks

T#3: Remove Collinearity

- LR will overfit your data when you have highly correlated input variables
- **Calculate pairwise correlations** for your input data and **removing the most correlated**

T#4: Gaussian Distributions

- LR will make more reliable predictions if your I/O variables have a Gaussian distribution
- There may be some benefit by **applying certain transformations** (e.g. log or BoxCox) **on our variables** to make their distribution more **Gaussian looking**.

Linear Regression: Data Preparation

Tasks

T#3: Remove Collinearity…

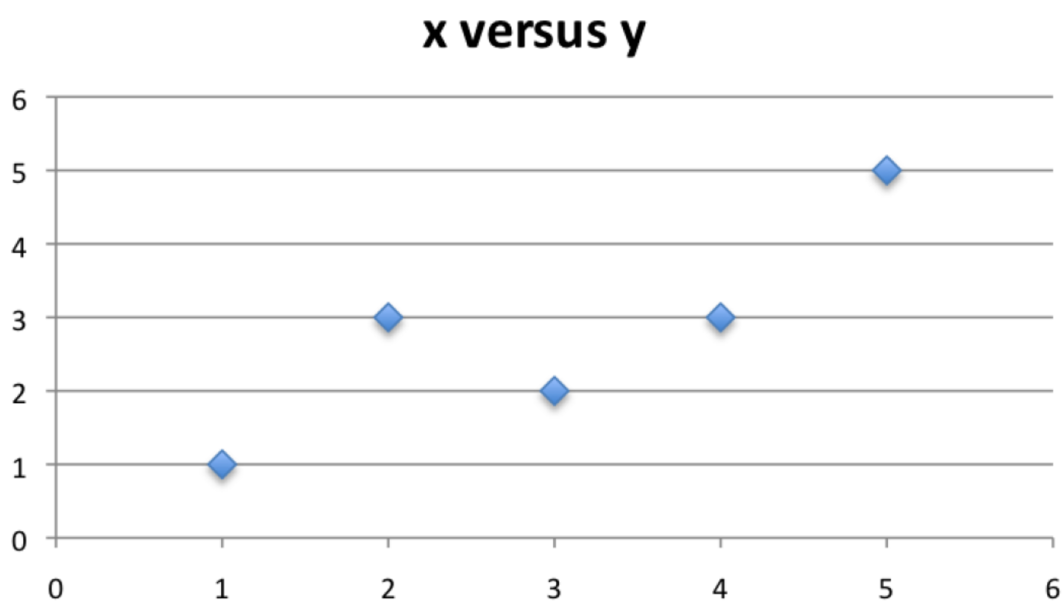
T#4: Gaussian Distributions…

T5: Rescale Inputs

- Rescale input variables using standardization or normalization

  so that LR can make more reliable predictions

Simple Linear Regression



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

- Simple Linear Regression Dataset
- Kind of linear relationship, we can use LR - **y = B0 + B1 • x**

How easy it is to calculate B0 and B1!

- B0 = mean(y) - B1 • mean(x)
- $$B1 = \frac{\sum_{i=1}^{n}(x_i - mean(x)) \times (y_i - mean(y))}{\sum_{i=1}^{n}(x_i - mean(x))^2}$$ SLOPE

Simple 3-STEP Solution Using just Excel

x	mean(x)	x-mean(x)	[x-mean(x)]^2	x-mean(x) - y-mean(y)	B1	B0
1	3	-2	4	3.6	0.8	0.4
2		-1	1	-0.2	SUM(H7:H11*H13:H17)/SUM(I7:I11)	G13-K7*G7
4		1	1	0.2		
3		0	0	0		
5		2	4	4.4		
y	**mean(y)**	**y-mean(y)**				
1	2.8	-1.8				
3		0.2				
3		0.2				
2		-0.8				
5		2.2				

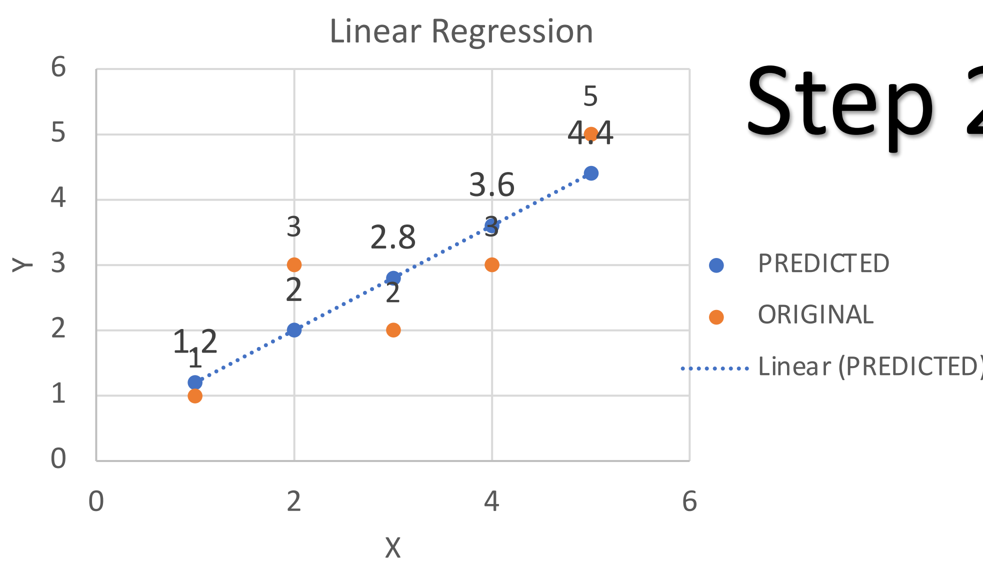
Step 1

Making Predictions

$$y = B0 + B1 \times x$$

$$y = 0.4 + 0.8 \times x$$

PREDICTIONS	
X	**Y**
1	1.2
2	2
4	3.6
3	2.8
5	4.4



Step 2

STEP 3

Pred Y	y	Pred-y	SQR Error	RMSE = sqrt(Sum(Sqr Error)/5)
1.2	1	0.2	0.04	0.692820323
2	3	-1	1	
3.6	3	0.6	0.36	
2.8	2	0.8	0.64	
4.4	5	-0.6	0.36	

Thus, we can say that:
"each prediction is on average wrong by about 0.692 units"

Simple 3-STEP Solution Using just Excel

Speed up process on Step 1

B1 SPEED CALCULATION - corr(x,y)*stdev(y)/stdev(x)
0.8
PEARSON - Correlation x,y
0.852802865
STDEV X
1.58113883
STDEV Y
1.483239697

- ❑ Standard deviation is a measure of how much on average the data is spread out from the mean
- ❑ Correlation (also known as Pearson's correlation coefficient) is a measure of how related two variables are in the range of -1 to 1
  - 1 = the two variables are perfectly positively correlated, they both move in the same direction
  - -1 = they are perfectly negatively correlated, when one moves the other moves in the other direction.

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Linear Regression – Using GD [1 Epoch]

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

$$B0 = 0.0$$

$$B1 = 0.0$$

$$y = 0.0 + 0.0 \times x$$

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 the predicted value for y using our starting point coefficients for the first training instance: $x = 1, y = 1$.

$$p(i) = 0.0 + 0.0 \times 1$$

$$p(i) = 0$$

$$error = (0 - 1)$$

$$error = -1$$

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

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

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

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

$$B1(t+1) = 0.0 - 0.01 \times -1 \times 1$$

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

Linear Regression – Using GD [1 Epoch]

$$p(i) = 0.0 + 0.0 \times 1$$
$$p(i) = 0$$

$$error = (0 - 1)$$
$$error = -1$$

$$B0(t+1) = B0(t) - alpha \times error$$
$$B0(t+1) = 0.0 - 0.01 \times -1.0$$
$$B0(t+1) = 0.01$$

$$B1(t+1) = B1(t) - alpha \times error \times x$$
$$B1(t+1) = 0.0 - 0.01 \times -1 \times 1$$
$$B1(t+1) = 0.01$$

We have just finished the first iteration of gradient descent and we have updated our weights to be:

- B0 = 0.01
- 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**.

TRAINING DATA	
x	y
1	1
2	3
4	3
3	2
5	5

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Linear Regression – Using GD Complete Solution

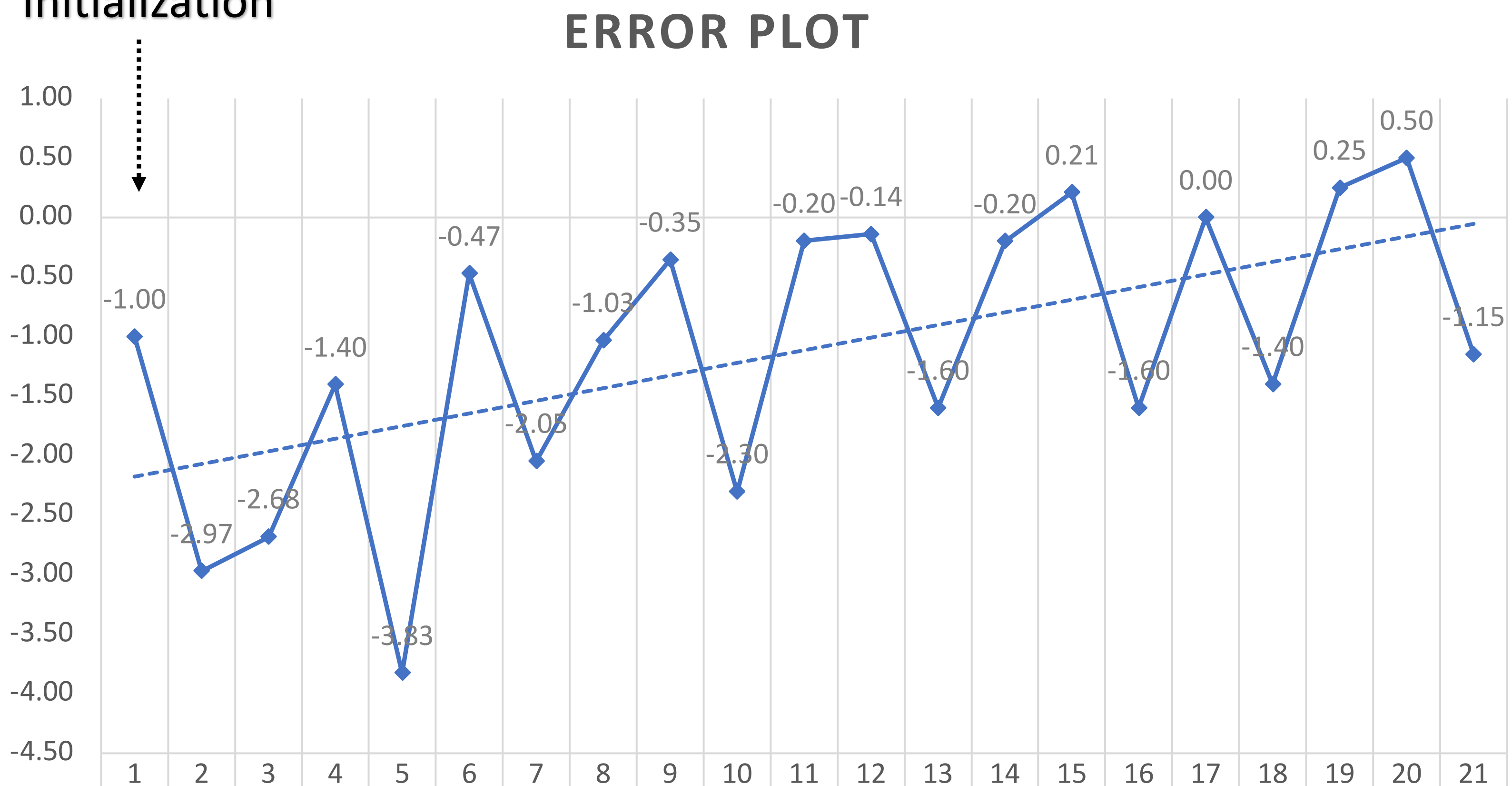
TRAINING DATA	
x	y
1	1
2	3
4	3
3	2
5	5

			Learning Rate (a)	0.01			
-			Initialization				
#		B0(t + 1) = B0(t) - alpha *error	B0(i)	B1(t + 1) = B1(t) - alpha*error*x	B1(i)	p(i) = B0(i) + B1(i)*x(i)	Error(i)=p(i)-y(i)
0	-	-	0	-	0	0	-1
1	**EPOCH 1**	B0(1) = B0(0) - alpha *error	**0.01**	B1(1) = B1(0) - alpha *error *x(0)	**0.01**	0.03	-2.97
2		B0(2) = B0(1) - alpha *error	**0.0397**	B1(2) = B1(1) - alpha *error *x(1)	**0.0694**	0.3173	-2.6827
3		B0(3) = B0(2) - alpha *error	**0.066527**	B1(3) = B1(2) - alpha *error *x(2)	**0.176708**	0.596651	-1.403349
4		B0(4) = B0(3) - alpha *error	**0.08056049**	B1(4) = B1(3) - alpha *error *x(3)	**0.21880847**	1.17460284	-3.82539716
5		B0(5) = B0(4) - alpha *error	**0.118814462**	B1(5) = B1(4) - alpha *error *x(4)	**0.410078328**	0.52889279	-0.47110721
6	EPOCH 2	B0(6) = B0(5) - alpha *error	0.123525534	B1(1) = B1(0) - alpha *error *x(0)	0.4147894	0.953104334	-2.046895666
7		B0(7) = B0(6) - alpha *error	0.14399449	B1(2) = B1(1) - alpha *error *x(1)	0.455727313	1.966903744	-1.033096256
8		B0(8) = B0(7) - alpha *error	0.154325453	B1(3) = B1(2) - alpha *error *x(2)	0.497051164	1.645478944	-0.354521056
9		B0(9) = B0(8) - alpha *error	0.157870663	B1(4) = B1(3) - alpha *error *x(3)	0.507686795	2.69630464	-2.30369536
10		B0(10)= B0(9) - alpha *error	0.180907617	B1(5) = B1(4) - alpha *error *x(4)	0.622871563	0.80377918	-0.19622082
11	**EPOCH 3**		**0.182869825**		**0.624833772**		
12			**0.198544452**		**0.656183024**		
13			**0.200311686**		**0.663251962**		
14			**0.19841101**		**0.657549935**		
15			**0.213549404**		**0.733241901**		
16	EPOCH 4		0.21408149		0.733773988		
17			0.227265196		0.760141398		
18			0.224586888		0.749428167		
19			0.219858174		0.735242025		
20			0.230897491		0.79043861		

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Error Plot – what do you notice?

Initialization



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
Series1	-1.0	-2.9	-2.6	-1.4	-3.8	-0.4	-2.0	-1.0	-0.3	-2.3	-0.2	-0.1	-1.6	-0.2	0.21	-1.6	0.00	-1.4	0.25	0.50	-1.1

Source: ML-Mastery – Master Machine Learning Algorithms
2020 by J. Brownlee

After 4 Epochs that we stopped

Out final coefficients have the values:

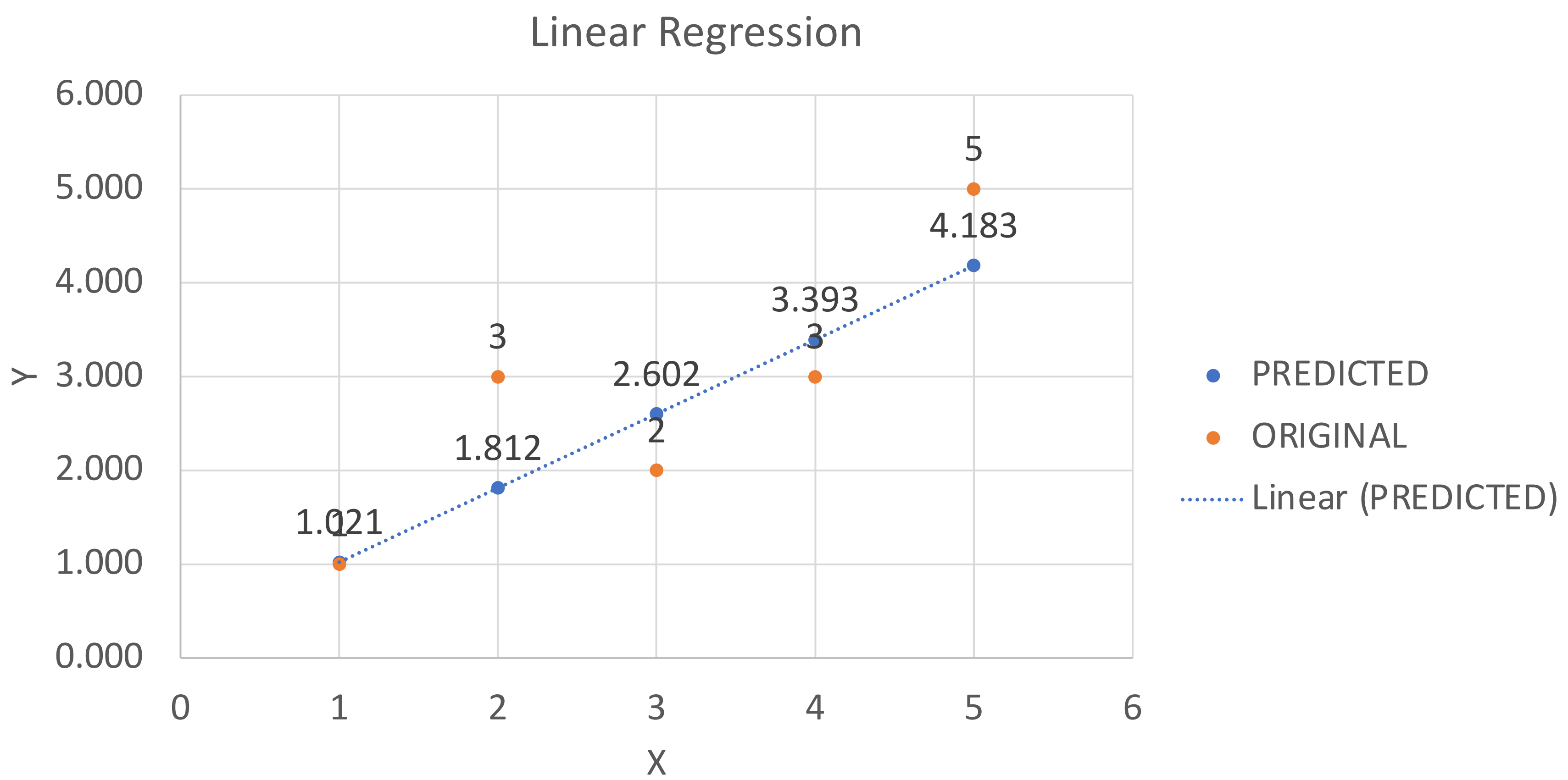
- B0 = 0.230897491
- B1 = 0.79043861

If we plug them into **our simple linear Regression model** and make a prediction for each point in our training dataset.

PREDICTIONS	
X	Pred Y
1	1.021
2	1.812
4	3.393
3	2.602
5	4.183

Pred Y	y	Pred-y	SQR Error	RMSE = sqrt(Sum(Sqr Error)/5)
1.021	1	0.021	0.000	0.721
1.812	3	-1.188	1.412	
3.393	3	0.393	0.154	
2.602	2	0.602	0.363	
4.183	5	-0.817	0.667	

Predictions – Outcome using GD



Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

Homework

1. Generate a **1000** points **synthetic dataset (x, y)** with data fluctuating in a specific angle, e.g., 45 degrees. Use the same linear equation as in the set of slides. Solve the problem using the **analytical** and the **GD** method. Demonstrate and discuss your results.

2. Repeat (1) using 3 features, x1, x2, and x3, i.e. (x1,x2,x3, y) and $y = b0 + b1 \cdot \mathbf{x1} + b2 \cdot \mathbf{x2} + b3 \cdot \mathbf{x3}$. Solve the problem using the **analytical** and the **GD** method. Demonstrate and discuss your results.

Source: ML-Mastery – ML Algorithms 2020 by J. Brownlee

Linear Regression – Key Learning Points

- The representation and learning algorithms used to create a linear regression model
- How to best prepare your data when modeling using linear regression
- How to calculate a simple linear regression step-by-step
- How to make predictions on new data using your model
- A shortcut that greatly simplifies the calculation
- How stochastic GD can be used to search for the coefficients of a regression model
- How repeated iterations of GD can create an accurate regression model

Source: ML-Mastery – Master Machine Learning Algorithms 2020 by J. Brownlee

What did we cover?

We discussed the LR algorithm for ML. We covered:

- The common names used when describing linear regression models
- The representation used by the model
- Learning algorithms used to estimate the coefficients in the model
- Rules of thumb to consider when preparing data for use with linear regression
- How to implement simple linear regression step-by-step in a spreadsheet
- How to estimate the coefficients for a simple linear regression model from your training data
- How to make predictions using your learned model

What did we cover?

You also, discovered the simple linear regression model and how to train it using stochastic gradient descent.

You learned:

- How to work through the application of the update rule for gradient descent
- How to make predictions using a learned linear regression model.
- You now know how to implement linear regression using stochastic gradient descent

NEXT

In the next chapter you will discover the logistic regression algorithm for binary classification

Questions?

THANK YOU!