Chapter 2

Linear Regression

Linear regression is perhaps one of the most well known and well understood algorithms in statistics and machine learning. In this chapter you will discover the linear regression algorithm, how it works and how you can best use it in on your machine learning projects. In this chapter you will learn:

Why linear regression belongs to both statistics and machine learning.

The many names by which linear regression is known.

The representation and learning algorithms used to create a linear regression model.

How to best prepare your data when modeling using linear regression.

Let's get started.

10.1 Isn't Linear Regression from Statistics?

Before we dive into the details of linear regression, you may be asking yourself why we are looking at this algorithm. Isn't it a technique from statistics?

Machine learning, more specifically the field of predictive modeling is primarily concerned with minimizing the error of a model or making the most accurate predictions possible, at the expense of explainability. In applied machine learning we will borrow, reuse and steal algorithms from many different fields, including statistics and use them towards these ends.

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. Next, let's review some of the common names used to refer to a linear regression model.

10.2 Many Names of Linear Regression

When you start looking into linear regression, things can get very confusing. The reason is because linear regression has been around for so long (more than 200 years). It has been studied from every possible angle and often each angle has a new and different name.

Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (*x*) and the single output variable (*y*). More specifically, that *y* can be calculated from a linear combination of the input variables (*x*). When there is a single input variable (*x*), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression. Different techniques can be used to prepare or train the linear regression equation from data, the most common of which is called Ordinary Least Squares. It is common to therefore refer to a model prepared this way as Ordinary Least Squares Linear Regression or just Least Squares Regression. Now that we know some names used to describe linear regression, let's take a closer look at the representation used.

10.3 Linear Regression Model Representation

Linear regression is an attractive model because the representation is so simple. The representation is a linear equation that combines a specific set of input values (*x*) the solution to which is the predicted output for that set of input values (*y*). As such, both the input values (*x*) and the output value are numeric.

The linear equation assigns one scale factor to each input value or column, called a coefficient that is commonly represented by the Greek letter Beta (β). One additional coefficient is also added, giving the line an additional degree of freedom (e.g. moving up and down on a two-dimensional plot) and is often called the intercept or the bias coefficient. For example, in a simple regression problem (a single *x* and a single *y*), the form of the model would be:

$$y = B0 + B1 \times x \tag{10.1}$$

In higher dimensions when we have more than one input (*x*), the line is called a plane or a hyper-plane. The representation therefore is the form of the equation and the specific values used for the coefficients (e.g. *B*0 and *B*1 in the above example). It is common to talk about the complexity of a regression model like linear regression. This refers to the number of coefficients used in the model.

When a coefficient becomes zero, it effectively removes the influence of the input variable on the model and therefore from the prediction made from the model ($0 \times x = 0$). This becomes relevant if you look at regularization methods that change the learning algorithm to reduce the complexity of regression models by putting pressure on the absolute size of the coefficients, driving some to zero. Now that we understand the representation used for a linear regression model, let's review some ways that we can learn this representation from data.

10.4 Linear Regression Learning the Model

Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available. In this section we will take a brief look at four techniques to prepare a linear regression model. This is not enough information to implement them from scratch, but enough to get a flavor of the computation and trade-offs involved.

There are many more techniques because the model is so well studied. Take note of Ordinary Least Squares because it is the most common method used in general. Also take note of Gradient Descent as it is the most common technique taught from a machine learning perspective.

10.4.1 Simple Linear Regression

With simple linear regression when we have a single input, we can use statistics to estimate the coefficients. This requires that you calculate statistical properties from the data such as means, standard deviations, correlations and covariance. All of the data must be available to traverse and calculate statistics. This is fun as an exercise in a spreadsheet, but not really useful in practice.

10.4.2 Ordinary Least Squares

When we have more than one input we can use Ordinary Least Squares to estimate the values of the coefficients. The Ordinary Least Squares procedure seeks to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line, square it, and sum all of the squared errors together. This is the quantity that ordinary least squares seeks to minimize.

This approach treats the data as a matrix and uses linear algebra operations to estimate the optimal values for the coefficients. It means that all of the data must be available and you must have enough memory to fit the data and perform matrix operations. It is unusual to implement the Ordinary Least Squares procedure yourself unless as an exercise in linear algebra. It is more likely that you will call a procedure in a linear algebra library. This procedure is very fast to calculate.

10.5 Gradient Descent

When there are one or more inputs you can use a process of optimizing the values of the coefficients by iteratively minimizing the error of the model on your training data. This operation is called Gradient Descent and works by starting with zero values for each coefficient. The sum of the squared errors are calculated for each pair of input and output values. A learning rate is used as a scale factor and the coefficients are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error is achieved or no further improvement is possible.

When using this method, you must select a learning rate (*alpha*) parameter that determines the size of the improvement step to take on each iteration of the procedure. Gradient descent is often taught using a linear regression model because it is relatively straightforward to understand. In practice, it is useful when you have a very large dataset either in the number of rows or the number of columns that may not fit into memory.

10.5.1 Regularized Linear Regression

There are extensions of the training of the linear model called regularization methods. These seek to both minimize the sum of the squared error of the model on the training data (using Ordinary Least Squares) but also to reduce the complexity of the model (like the number or absolute size of the sum of all coefficients in the model). Two popular examples of regularization procedures for linear regression are:

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).

These methods are effective to use when there is collinearity in your input values and ordinary least squares would overfit the training data. Now that you know some techniques to learn the coefficients in a linear regression model, let's look at how we can use a model to make predictions on new data.

10.6 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:

$$\begin{aligned} y &= B0 + B1 \times X1 \\ weight &= B0 + B1 \times height \end{aligned} \tag{10.2}$$

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 $B1 = 0.5$. Let's plug them in and calculate the weight (in kilograms) for a person with the height of 182 centimeters.

$$\begin{aligned} weight &= 0.1 + 0.05 \times 182 \\ weight &= 91.1 \end{aligned} \tag{10.3}$$

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.

Now that we know how to make predictions given a learned linear regression model, let's look at some rules of thumb for preparing our data to make the most of this type of model.

10.7 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 uses 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.

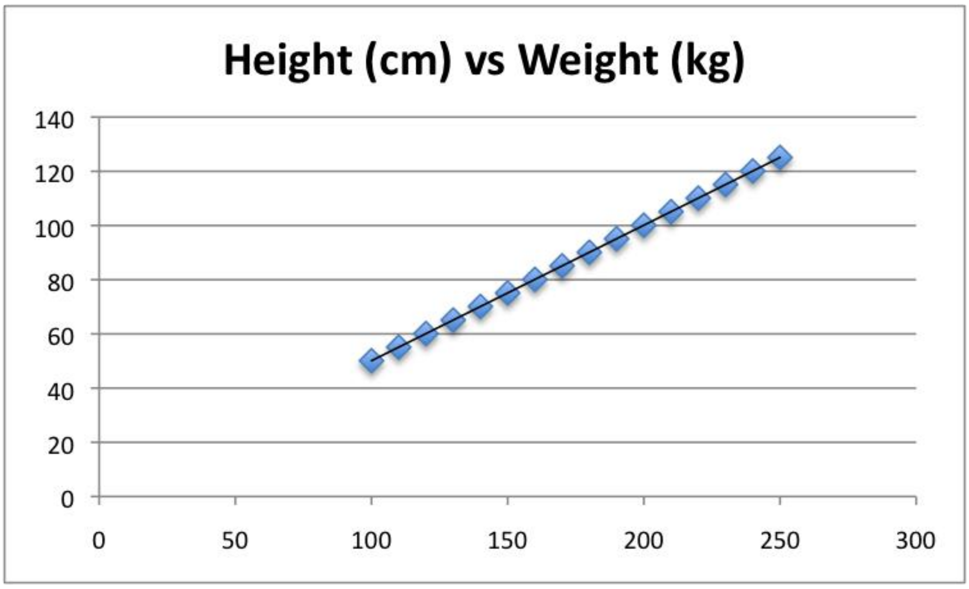

Figure 10.1: Sample Height vs Weight Linear Regression.

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 you 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.

10.8 Summary

In this chapter you discovered the linear regression algorithm for machine learning. You covered a lot of ground including:

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.

You now know about the linear regression algorithm for making real-valued predictions. In the next chapter you will discover how to implement the simple linear regression algorithm from scratch.

Simple Linear Regression Tutorial

Linear regression is a very simple method but has proven to be very useful for a large number of situations. In this chapter you will discover exactly how linear regression works step-by-step. After reading this chapter you will know:

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.

Let's get started.

11.1 Tutorial Data Set

The data set we are using is completely made up. Below is the raw data.

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

Listing 11.1: Tutorial Data Set.

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. Below is a simple scatter plot of *x* versus *y*.

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.

11.2 Simple Linear Regression

When we have a single input attribute (*x*) and we want to use linear regression, this is called simple linear regression. If we had multiple input attributes (e.g. *X*1, *X*2, *X*3, etc.) This would

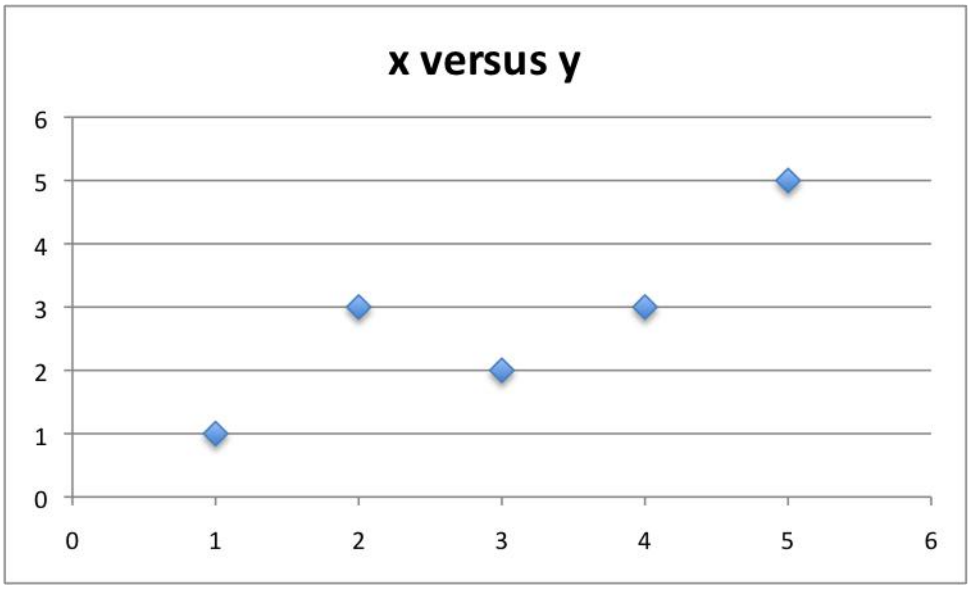

Figure 11.1: Simple Linear Regression Dataset.

be called multiple linear regression. The procedure for linear regression is different and simpler than that for multiple linear regression, so it is a good place to start. In this section we are going to create a simple linear regression model from our training data, then make predictions for our training data to get an idea of how well the model learned the relationship in the data. With simple linear regression we want to model our data as follows:

$$y = B0 + B1 \times x \tag{11.1}$$

This is a line where y is the output variable we want to predict, *x* is the input variable we know and *B*0 and *B*1 are coefficients that we need to estimate that move the line around. Technically, *B*0 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 *B*1 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 goal is to find the best estimates for the coefficients to minimize the errors in predicting *y* from *x*. Simple regression is great, because rather than having to search for values by trial and error or calculate them analytically using more advanced linear algebra, we can estimate them directly from our data. We can start off by estimating the value for *B*1 as:

$$B1 = \frac{\sum_{i=1}^{n}(x_i - mean(x)) \times (y_i - mean(y))}{\sum_{i=1}^{n}(x_i - mean(x))^2} \tag{11.2}$$

Where *mean*() is the average value for the variable in our dataset. The x_i and y_i refer to the fact that we need to repeat these calculations across all values in our dataset and *i* refers to the *i*'th value of *x* or *y*. We can calculate *B*0 using *B*1 and some statistics from our dataset, as follows:

$$B0 = mean(y) - B1 \times mean(x) \tag{11.3}$$

Not that bad right? We can calculate these right in our spreadsheet.

11.2.1 Estimating The Slope (B1)

Let’s start with the top part of the equation, the numerator. First we need to calculate the mean value of *x* and *y*. The mean is calculated as:

$$\frac{1}{n} \times \sum_{i=1}^{n} x_i \tag{11.4}$$

Where *n* is the number of values (5 in this case). You can use the AVERAGE() function in your spreadsheet. Let’s calculate the mean value of our *x* and *y* variables:

$$\begin{aligned} mean(x) &= 3 \\ mean(y) &= 2.8 \end{aligned} \tag{11.5}$$

Now we need to calculate the error of each variable from the mean. Let’s do this with *x* first:

```
x  mean(x)   x - mean(x)
1  3         -2
2            -1
4            1
3            0
5            2
```

Listing 11.2: Residual of each x value from the mean.

Now let’s do that for the *y* variable.

```
y  mean(y)   y - mean(y)
1  2.8       -1.8
3            0.2
3            0.2
2            -0.8
5            2.2
```

Listing 11.3: Residual of each y value from the mean.

We now have the parts for calculating the numerator. All we need to do is multiple the error for each *x* with the error for each *y* and calculate the sum of these multiplications.

```
x - mean(x)   y - mean(y)   Multiplication
-2            -1.8          3.6
-1            0.2           -0.2
1             0.2           0.2
0             -0.8          0
2             2.2           4.4
```

Listing 11.4: Multiplication of the x and y residuals from their means.

Summing the final column we have calculated our numerator as 8. Now we need to calculate the bottom part of the equation for calculating *B*1, or the denominator. This is calculated as the sum of the squared differences of each *x* value from the mean. We have already calculated the difference of each *x* value from the mean, all we need to do is square each value and calculate the sum.

```
x - mean(x)   squared
-2            4
-1            1
```

```
1          1
0          0
2          4
```

Listing 11.5: Squared residual of each x value from the mean.

Calculating the sum of these squared values gives us up denominator of 10. Now we can calculate the value of our slope.

$$\begin{aligned} B1 &= \frac{8}{10} \\ B1 &= 0.8 \end{aligned} \tag{11.6}$$

11.2.2 Estimating The Intercept (B0)

This is much easier as we already know the values of all of the terms involved.

$$\begin{aligned} &B0 = mean(y) - B1 \times mean(x) \\ &B0 = 2.8 - 0.8 \times 3 \\ &B0 = 0.4 \end{aligned} \tag{11.7}$$

11.3 Making Predictions

We now have the coefficients for our simple linear regression equation.

$$\begin{aligned} y &= B0 + B1 \times x \\ y &= 0.4 + 0.8 \times x \end{aligned} \tag{11.8}$$

Let's try out the model by making predictions for our training data.

```
x   Predicted  Y
1   1.2
2   2
4   3.6
3   2.8
5   4.4
```

Listing 11.6: Predicted y value for each x input value.

We can plot these predictions as a line with our data. This gives us a visual idea of how well the line models our data.

11.4 Estimating Error

We can calculate an error score for our predictions called the Root Mean Squared Error or RMSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (p_i - y_i)^2}{n}} \tag{11.9}$$

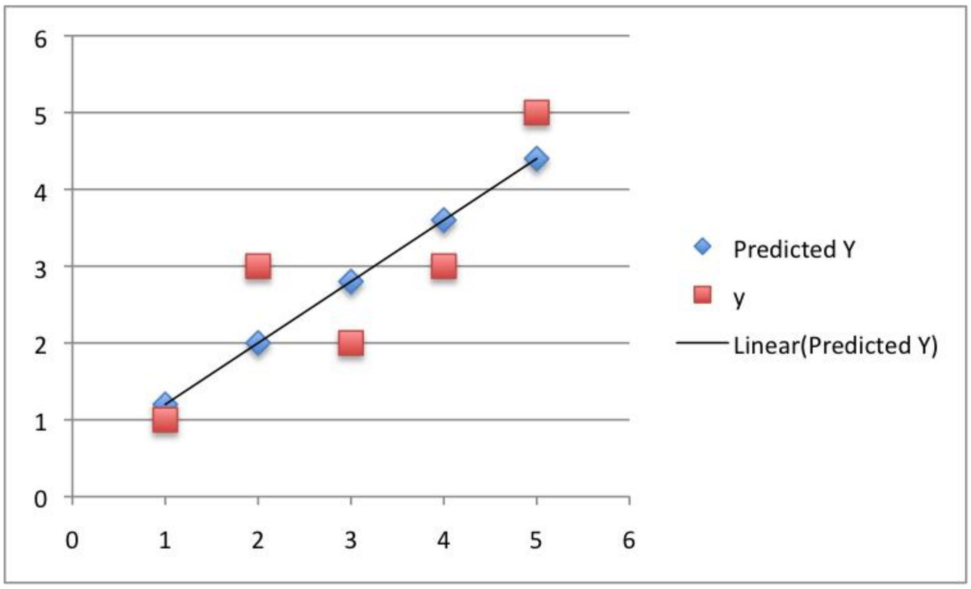

Figure 11.2: Simple Linear Regression Predictions.

Where you can use SQRT() function in your spreadsheet to calculate the square root, *p* is the predicted value and *y* is the actual value, *i* is the index for a specific instance, because we must calculate the error across all predicted values. First we must calculate the difference between each model prediction and the actual *y* values.

```
Predicted   y   Predicted - y
1.2         1   0.2
2           3   -1
3.6         3   0.6
2.8         2   0.8
4.4         5   -0.6
```

Listing 11.7: Error for predicted values.

We can easily calculate the square of each of these error values ($error \times error$ or $error^2$).

```
Predicted - y    squared error
0.2              0.04
-1               1
0.6              0.36
0.8              0.64
-0.6             0.36
```

Listing 11.8: Squared error for predicted values.

The sum of these errors is 2.4 units, dividing by 5 and taking the square root gives us:

$$RMSE = 0.692820323 \tag{11.10}$$

Or, each prediction is on average wrong by about 0.692 units.

11.5 Shortcut

Before we wrap up I want to show you a quick shortcut for calculating the coefficients. Simple linear regression is the simplest form of regression and the most studied. There is a shortcut that you can use to quickly estimate the values for $B0$ and $B1$. Really it is a shortcut for calculating $B1$. The calculation of $B1$ can be re-written as:

$$B1 = corr(x, y) \times \frac{stdev(y)}{stdev(x)} \tag{11.11}$$

Where $corr(x)$ is the correlation between x and y an $stdev()$ is the calculation of the standard deviation for a variable. Correlation (also known as Pearson's correlation coefficient) is a measure of how related two variables are in the range of -1 to 1. A value of 1 indicates that the two variables are perfectly positively correlated, they both move in the same direction and a value of -1 indicates that they are perfectly negatively correlated, when one moves the other moves in the other direction.

Standard deviation is a measure of how much on average the data is spread out from the mean. You can use the function PEARSON() in your spreadsheet to calculate the correlation of x and y as 0.852 (highly correlated) and the function STDEV() to calculate the standard deviation of x as 1.5811 and y as 1.4832. Plugging these values in we have:

$$\begin{aligned} B1 &= 0.852802865 \times \frac{1.483239697}{1.58113883} \\ B1 &= 0.8 \end{aligned} \tag{11.12}$$

11.6 Summary

In this chapter you discovered how to implement simple linear regression step-by-step in a spreadsheet. You learned:

- How to estimate the coefficients for a simple linear regression model from your training data.
- How to make predictions using your learned model.

You now know how to implement the simple linear regression algorithm from scratch. In the next section, you will discover how you can implement linear regression from scratch using stochastic gradient descent.

Linear Regression Tutorial Using Gradient Descent

Stochastic Gradient Descent is an important and widely used algorithm in machine learning. In this chapter 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 chapter you will know:

- How stochastic gradient descent can be used to search for the coefficients of a regression model.
- How repeated iterations of gradient descent can create an accurate regression model.

Let's get started.

12.1 Tutorial Data Set

The dataset is the same as that used in the previous chapter on Simple Linear Regression. It is listed again for completeness.

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

Listing 12.1: Tutorial Data Set.

12.2 Stochastic Gradient Descent

Gradient Descent is the process of minimizing a function by following the gradients of the cost function. 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 technique that evaluates and update the coefficients every iteration called stochastic gradient descent to minimize the error of a model on our training data.

The way this optimization algorithm 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:

$$w = w - alpha \times delta \tag{12.1}$$

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.

12.3 Simple Linear Regression with Stochastic Gradient Descent

The coefficients used in simple linear regression can be found using stochastic gradient descent. Stochastic gradient descent is not used to calculate the coefficients for linear regression in practice unless the dataset prevents traditional Ordinary Least Squares being used (e.g. a very large dataset). Nevertheless, linear regression does provide a useful exercise for practicing stochastic gradient descent which is an important algorithm used for minimizing cost functions by machine learning algorithms. As stated in the previous chapter, our linear regression model is defined as follows:

$$y = B0 + B1 \times x \tag{12.2}$$

12.3.1 Gradient Descent Iteration #1

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

$$\begin{aligned} B0 &= 0.0 \\ B1 &= 0.0 \\ y &= 0.0 + 0.0 \times x \end{aligned} \tag{12.3}$$

We can calculate the error for a prediction as follows:

$$error = p(i) - y(i) \tag{12.4}$$

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 point coefficients for the first training instance: $x = 1, y = 1$.

$$\begin{aligned} p(i) &= 0.0 + 0.0 \times 1 \\ p(i) &= 0 \end{aligned} \tag{12.5}$$

Using the predicted output, we can calculate our error:

$$\begin{aligned} error &= (0 - 1) \\ error &= -1 \end{aligned} \tag{12.6}$$

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 *B*0 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 *B*0 coefficient as follows:

$$B0(t+1) = B0(t) - alpha \times error \tag{12.7}$$

Where *B*0(*t* + 1) is the updated version of the coefficient we will use on the next training instance, *B*0(*t*) is the current value for *B*0, *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 *B*0 will be:

$$\begin{aligned} B0(t+1) &= 0.0 - 0.01 \times -1.0 \\ B0(t+1) &= 0.01 \end{aligned} \tag{12.8}$$

Now, let's look at updating the value for *B*1. We use the same equation with one small change. The error is filtered by the input that caused it. We can update *B*1 using the equation:

$$B1(t+1) = B1(t) - alpha \times error \times x \tag{12.9}$$

Where *B*1(*t* + 1) is the update coefficient, *B*1(*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 *B*1:

$$\begin{aligned} B1(t+1) &= 0.0 - 0.01 \times -1 \times 1 \\ B1(t+1) &= 0.01 \end{aligned} \tag{12.10}$$

We have just finished the first iteration of gradient descent and we have updated our weights to be *B*0 = 0.01 and *B*1 = 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.

12.3.2 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
0.118814462	0.410078328
0.123525534	0.4147894
0.14399449	0.455727313
0.154325453	0.497051164

```
0.157870663     0.507686795
0.180907617     0.622871563
0.182869825     0.624833772
0.198544452     0.656183024
0.200311686     0.663251962
0.19841101      0.657549935
0.213549404     0.733241901
0.21408149      0.733773988
0.227265196     0.760141398
0.224586888     0.749428167
0.219858174     0.735242025
0.230897491     0.79043861
```

Listing 12.2: Simple linear regression coefficients after 20 iterations.

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.

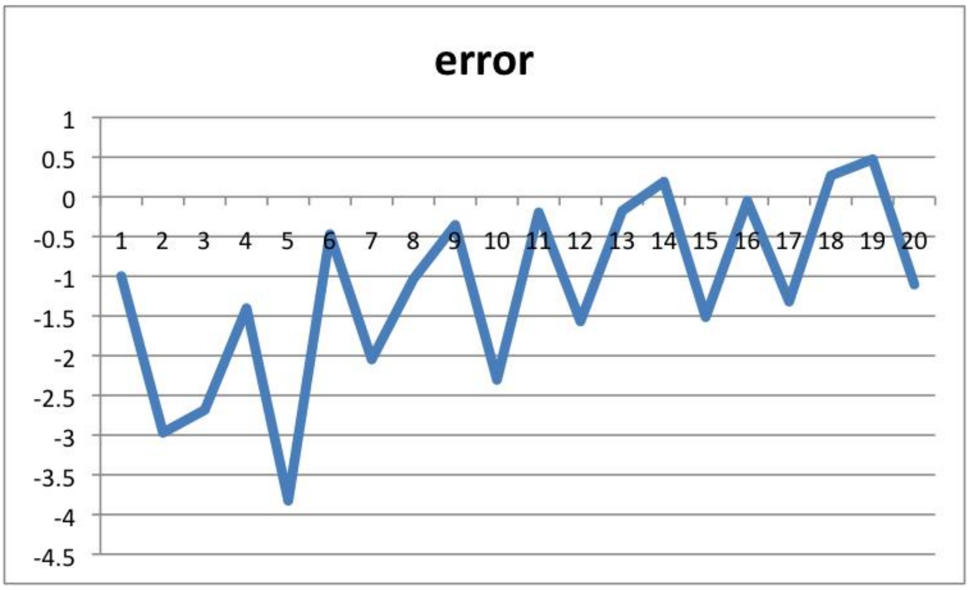

Figure 12.1: Simple Linear Regression Performance Versus Iteration.

You can see that our final coefficients have the values $B0$ = 0.230897491 and $B1$ = 0.79043861. Let's plug them into our simple linear Regression model and make a prediction for each point in our training dataset.

```
x   Prediction
1   1.021336101
2   1.811774711
4   3.392651932
3   2.602213322
```

```
5     4.183090542
```

Listing 12.3: Simple linear regression predictions for the training dataset.

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.

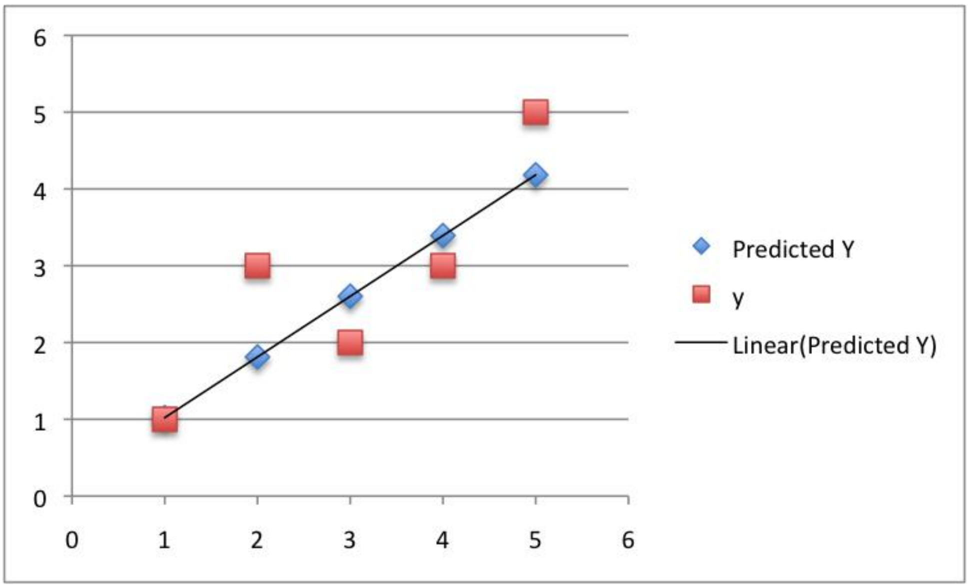

Figure 12.2: Simple Linear Regression Predictions.

We can calculate the RMSE for these predictions as we did in the previous chapter. The result comes out to be $RMSE = 0.720626401$.

12.4 Summary

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