Chapter 11

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.

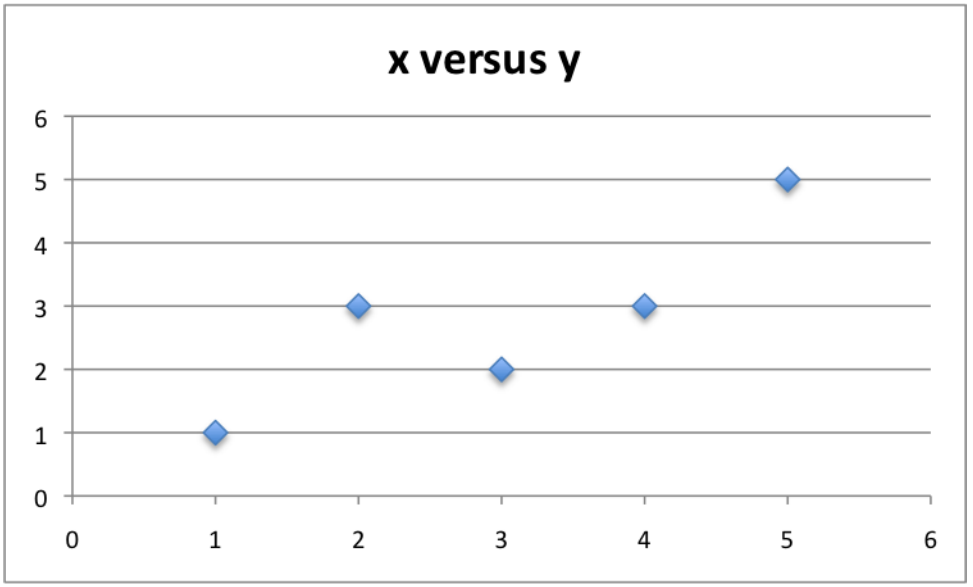

Figure 11.1: Simple Linear Regression Dataset.

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. $X1$, $X2$, $X3$, etc.) This would 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 $B0$ and $B1$ are coefficients that we need to estimate that move the line around. Technically, $B0$ 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 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 $B1$ 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 $B0$ using $B1$ 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 $B1$, 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 a 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.

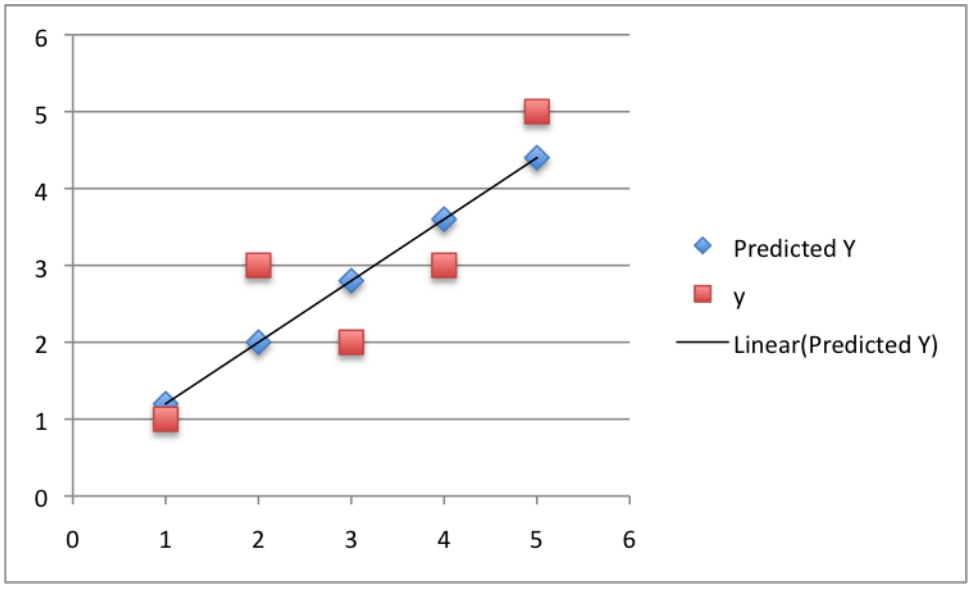

Figure 11.2: Simple Linear Regression Predictions.

11.4 Estimating Error

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

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

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 $corr(x, y)$ 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.

Chapter 12

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 *delta* 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 the 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 $B0$ 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:

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

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:

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

Now, let's look at updating the value for $B1$. 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:

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

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 $B1$:

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

We have just finished the first iteration of gradient descent and we have updated our weights to be $B0 = 0.01$ 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.

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.

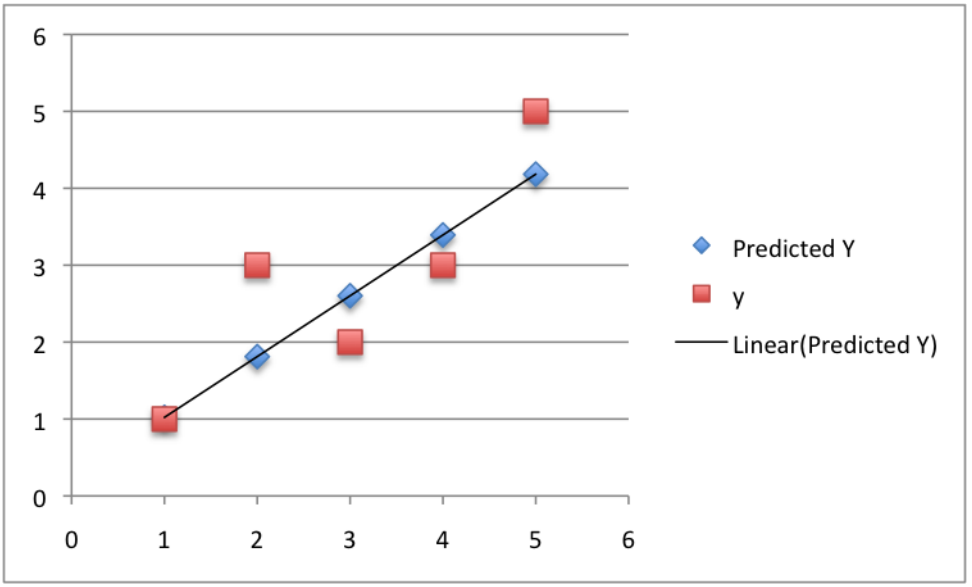

Figure 12.2: Simple Linear Regression Predictions.

We can calculate the RMSE for these predictions as we did in the previous chapter. The result comes out to be $RMSE = 0.720626401$. This is very close to the RMSE achieved in the previous section, but not the same. This is because RMSE is an optimization procedure and must discover a good solution which may not be the same set of coefficients calculated analytically. Gradient descent should only be used when analytical methods cannot be used, such as having very large amounts of data.

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.

Chapter 14

Logistic Regression Tutorial

Logistic regression is one of the most popular machine learning algorithms for binary classification. This is because it is a simple algorithm that performs very well on a wide range of problems. In this chapter you are going to discover the logistic regression algorithm for binary classification, step-by-step. After reading this chapter you will know:

- How to calculate the logistic function.
- How to learn the coefficients for a logistic regression model using stochastic gradient descent.
- How to make predictions using a logistic regression model.

Let's get started.

14.1 Tutorial Dataset

In this tutorial we will use a contrived dataset. This dataset has two input variables ($X1$ and $X2$) and one output variable (Y). The input variables are real-valued random numbers drawn from a Gaussian distribution. The output variable has two values, making the problem a binary classification problem. The raw data is listed below.

```
X1            X2            Y
2.7810836     2.550537003   0
1.465489372   2.362125076   0
3.396561688   4.400293529   0
1.38807019    1.850220317   0
3.06407232    3.005305973   0
7.627531214   2.759262235   1
5.332441248   2.088626775   1
6.922596716   1.77106367    1
8.675418651   -0.242068655  1
7.673756466   3.508563011   1
```

Listing 14.1: Tutorial Data Set.

Below is a plot of the dataset. You can see that it is completely contrived and that we can easily draw a line to separate the classes. This is exactly what we are going to do with the logistic regression model.

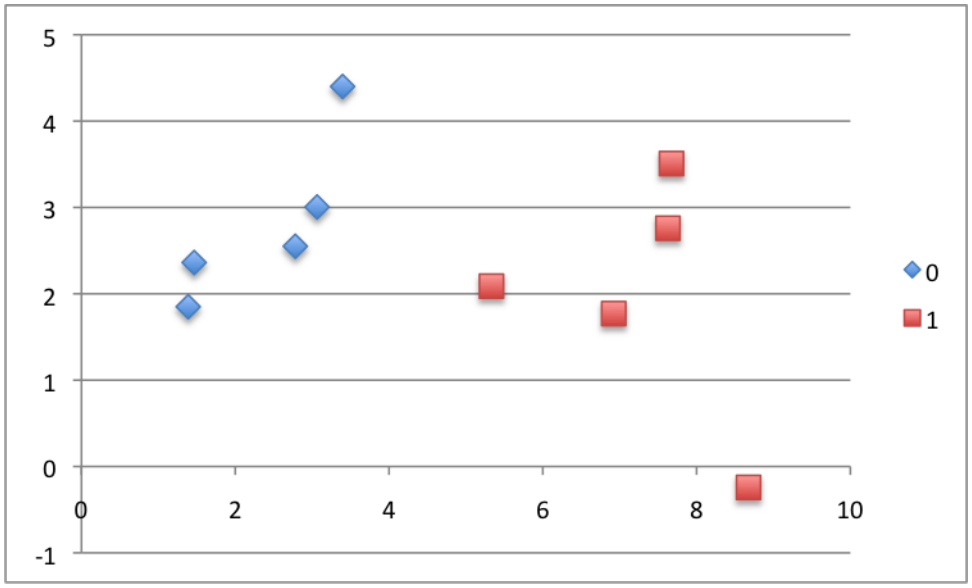

Figure 14.1: Logistic Regression Dataset.

14.2 Logistic Regression Model

The logistic regression model takes real-valued inputs and makes a prediction as to the probability of the input belonging to the default class (class 0). If the probability is greater than 0.5 we can take the output as a prediction for the default class (class 0), otherwise the prediction is for the other class (class 1). For this dataset, the logistic regression has three coefficients just like linear regression, for example:

$$output = B0 + B1 \times X1 + B2 \times X2 \tag{14.1}$$

The job of the learning algorithm will be to discover the best values for the coefficients ($B0$, $B1$ and $B2$) based on the training data. Unlike linear regression, the output is transformed into a probability using the logistic function:

$$p(class = 0) = \frac{1}{1 + e^{-output}} \tag{14.2}$$

In your spreadsheet this would be written as:

$$p(class = 0) = \frac{1}{1 + EXP(-output)} \tag{14.3}$$

14.3 Logistic Regression by Stochastic Gradient Descent

We can estimate the values of the coefficients using stochastic gradient descent. We can apply stochastic gradient descent to the problem of finding the coefficients for the logistic regression model.

14.3.1 Calculate Prediction

Let's start off by assigning 0.0 to each coefficient and calculating the probability of the first training instance that belongs to class 0.

$$\begin{aligned} B0 &= 0.0 \\ B1 &= 0.0 \\ B2 &= 0.0 \end{aligned} \tag{14.4}$$

The first training instance is: $X1 = 2.7810836$, $X2 = 2.550537003$, $Y = 0$. Using the above equation we can plug in all of these numbers and calculate a prediction:

$$\begin{aligned} prediction &= \frac{1}{1 + e^{-(B0 + B1 \times X1 + B2 \times X2)}} \\ prediction &= \frac{1}{1 + e^{-(0.0 + 0.0 \times 2.7810836 + 0.0 \times 2.550537003)}} \\ prediction &= 0.5 \end{aligned} \tag{14.5}$$

14.3.2 Calculate New Coefficients

We can calculate the new coefficient values using a simple update equation.

$$b = b + alpha \times (y - prediction) \times prediction \times (1 - prediction) \times x \tag{14.6}$$

Where b is the coefficient we are updating and prediction is the output of making a prediction using the model. *Alpha* is a parameter that you must specify at the beginning of the training run. This is the learning rate and controls how much the coefficients (and therefore the model) changes or learns each time it is updated. Larger learning rates are used in online learning (when we update the model for each training instance). Good values might be in the range 0.1 to 0.3. Let's use a value of 0.3.

You will notice that the last term in the equation is x, this is the input value for the coefficient. You will notice that the $B0$ does not have an input. This coefficient is often called the bias or the intercept and we can assume it always has an input value of 1.0. This assumption can help when implementing the algorithm using vectors or arrays. Let's update the coefficients using the prediction (0.5) and coefficient values (0.0) from the previous section.

$$\begin{aligned} B0 &= B0 + 0.3 \times (0 - 0.5) \times 0.5 \times (1 - 0.5) \times 1.0 \\ B1 &= B1 + 0.3 \times (0 - 0.5) \times 0.5 \times (1 - 0.5) \times 2.7810836 \\ B2 &= B2 + 0.3 \times (0 - 0.5) \times 0.5 \times (1 - 0.5) \times 2.550537003 \end{aligned} \tag{14.7}$$

or

$$\begin{aligned} B0 &= -0.0375 \\ B1 &= -0.104290635 \\ B2 &= -0.095645138 \end{aligned} \tag{14.8}$$

14.3.3 Repeat the Process

We can repeat this process and update the model for each training instance in the dataset. A single iteration through the training dataset is called an epoch. It is common to repeat the stochastic gradient descent procedure for a fixed number of epochs. At the end of epoch you can calculate error values for the model. Because this is a classification problem, it would be nice to get an idea of how accurate the model is at each iteration. The graph below show a plot of accuracy of the model over 10 epochs.

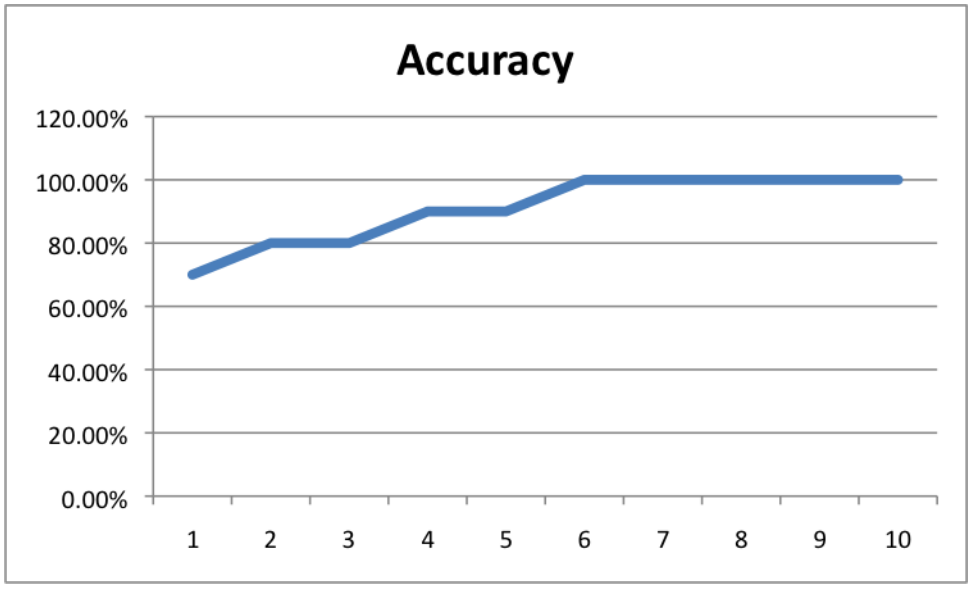

Figure 14.2: Logistic Regression with Gradient Descent Accuracy versus Iteration.

You can see that the model very quickly achieves 100% accuracy on the training dataset. The coefficients calculated after 10 epochs of stochastic gradient descent are:

$$\begin{aligned} B0 &= -0.406605464 \\ B1 &= 0.852573316 \\ B2 &= -1.104746259 \end{aligned} \tag{14.9}$$

14.3.4 Make Predictions

Now that we have trained the model, we can use it to make predictions. We can make predictions on the training dataset, but this could just as easily be new data. Using the coefficients above learned after 10 epochs, we can calculate output values for each training instance:

X1	X2	Prediction
2.7810836	2.550537003	0.298756986
1.465489372	2.362125076	0.145951056
3.396561688	4.400293529	0.085333265
1.38807019	1.850220317	0.219737314
3.06407232	3.005305973	0.247059
7.627531214	2.759262235	0.954702135
5.332441248	2.088626775	0.862034191
6.922596716	1.77106367	0.971772905

```
8.675418651    -0.242068655   0.999295452
7.673756466    3.508563011    0.905489323
```

Listing 14.2: Raw Logistic Regression Predictions.

These are the probabilities of each instance belonging to $Y = 0$. We can convert these into crisp class values using:

$$prediction = \text{IF } (output < 0.5) \text{ Then 0 Else 1} \tag{14.10}$$

With this simple procedure we can convert all of the outputs to class values:

```
Prediction     Crisp
0.298756986    0
0.145951056    0
0.085333265    0
0.219737314    0
0.247059       0
0.954702135    1
0.862034191    1
0.971772905    1
0.999295452    1
0.905489323    1
```

Listing 14.3: Crisp Logistic Regression Predictions.

Finally, we can calculate the accuracy for the model on the training dataset:

$$\begin{aligned} accuracy &= \frac{CorrectPredictions}{TotalPredictions} \times 100 \\ accuracy &= \frac{10}{10} \times 100 \\ accuracy &= 100\% \end{aligned} \tag{14.11}$$

14.4 Summary

In this chapter you discovered how you can implement logistic regression from scratch, step-by-step. You learned:

- How to calculate the logistic function.
- How to learn the coefficients for a logistic regression model using stochastic gradient descent.
- How to make predictions using a logistic regression model.

You now know how to implement logistic regression from scratch using stochastic gradient descent. In the next chapter you will discover the linear discriminant analysis algorithm for classification.