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.

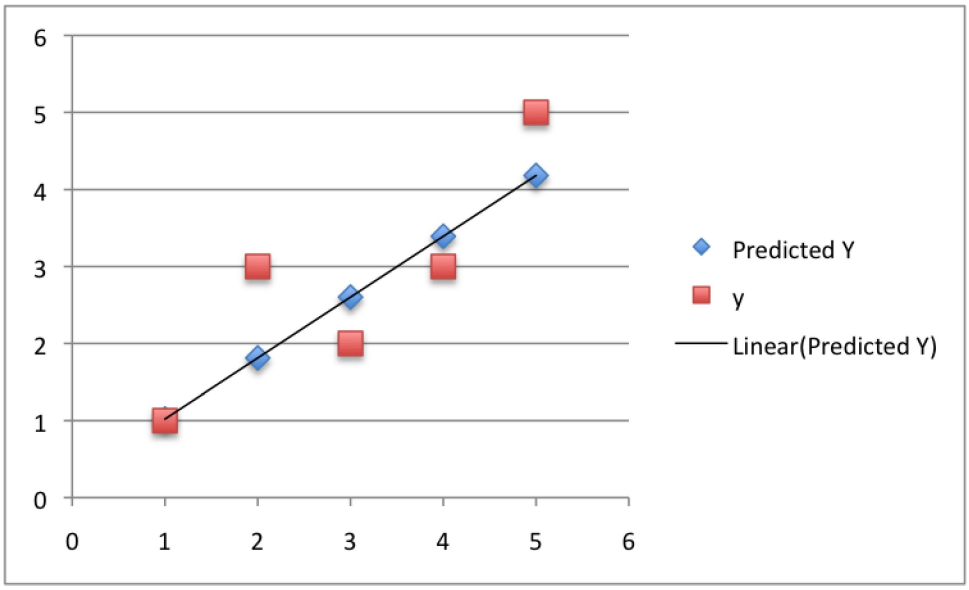

Figure 12.2: Simple Linear Regression Predictions.

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.