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.

After reading this 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.

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). In 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
2.7810836	2.55053700
1.465489372	2.36212507
3.396561688	4.40029352

```
 1  X1 X2 Y
 2  2.7810836 2.550537003 0
 3  1.465489372 2.362125076 0
 4  3.396561688 4.400293529 0
 5  1.38807019 1.850220317 0
 6  3.06407232 3.005305973 0
 7  7.627531214 2.759262235 1
 8  5.332441248 2.088626775 1
 9  6.922596716 1.77106367 1
10 8.675418651 -0.2420686549 1
11 7.673756466 3.508563011 1
```

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.
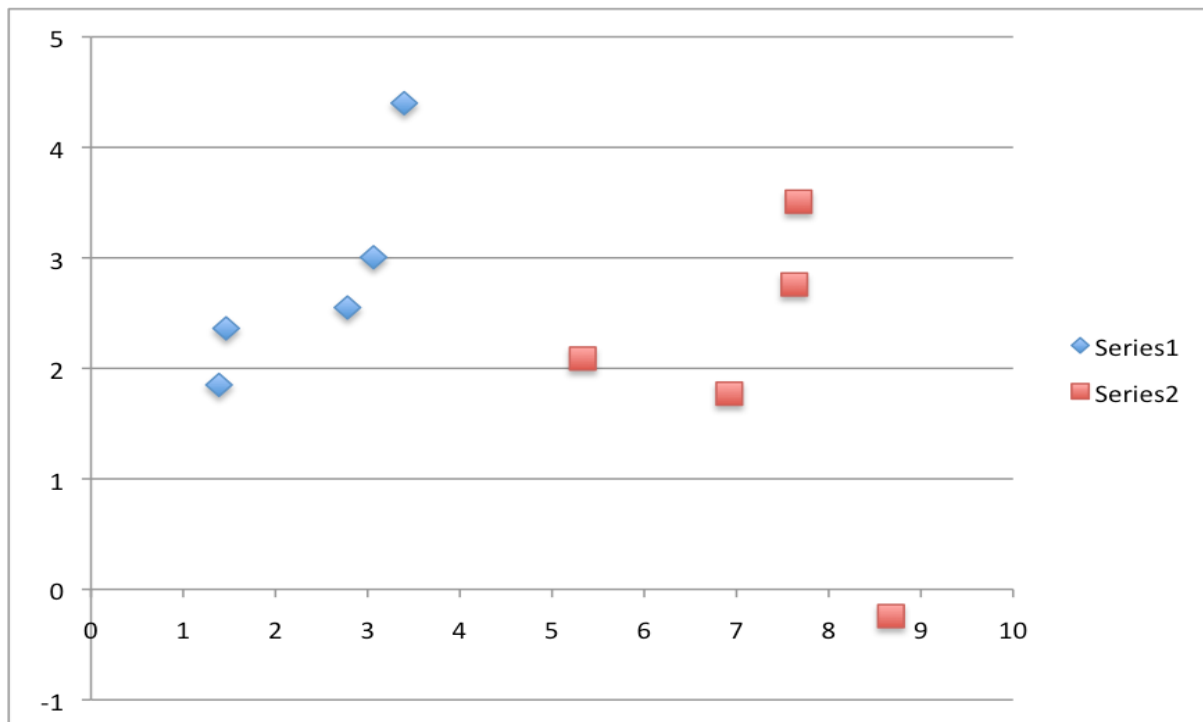
Logistic Regression Tutorial Dataset

Logistic Function

Before we dive into logistic regression, let's take a look at the logistic function, the heart of the logistic regression technique.

The logistic function is defined as:

$$transformed = 1 / (1 + e^{-x})$$

Where e is the numerical constant Euler's number and x is a input we plug into the function.

Let's plug in a series of numbers from -5 to +5 and see how the logistic function transforms them:

X	Transformed
-5	0.006692850924
-4	0.01798620996
-3	0.04742587318

	X	Transformed
1		
2	-5	0.006692850924
3	-4	0.01798620996
4	-3	0.04742587318
5	-2	0.119202922
6	-1	0.2689414214
7	0	0.5

8   1 0.7310585786
9   2 0.880797078
10 3 0.9525741268
11 4 0.98201379
12 5 0.9933071491

You can see that all of the inputs have been transformed into the range [0, 1] and that the smallest negative numbers resulted in values close to zero and the larger positive numbers resulted in values close to one. You can also see that 0 transformed to 0.5 or the midpoint of the new range.

From this we can see that as long as our mean value is zero, we can plug in positive and negative values into the function and always get out a consistent transform into the new range.
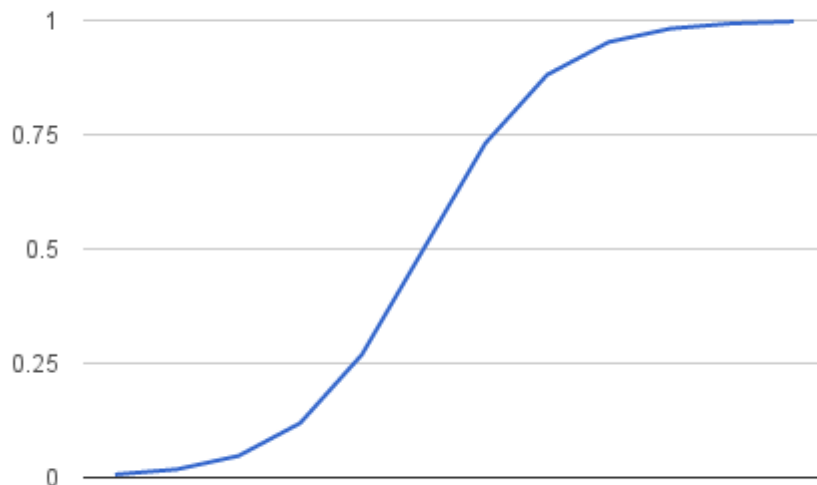
Logistic Function

Get your FREE Algorithms Mind Map
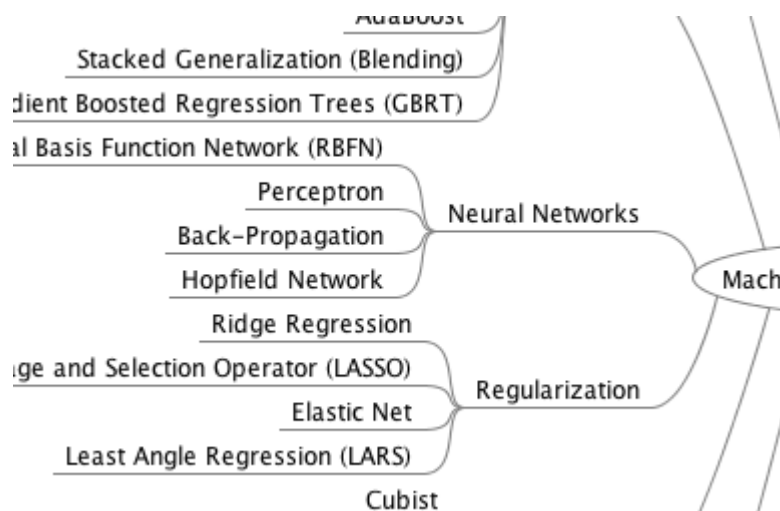
Logistic Regression Model

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 > 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:

$$output = b0 + b1*x1 + b2*x2$$

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:

$$p(class=0) = 1 / (1 + e^{\wedge}(-output))$$

In your spreadsheet this would be written as:

$$p(class=0) = 1 / (1 + EXP(-output))$$

Logistic Regression by Stochastic Gradient Descent

We can estimate the values of the coefficients using stochastic gradient descent.

This is a simple procedure that can be used by many algorithms in machine learning. It works by using the model to calculate a prediction for each instance in the training set and calculating the error for each prediction.

We can apply stochastic gradient descent to the problem of finding the coefficients for the logistic regression model as follows:

Given each training instance:

1. Calculate a prediction using the current values of the coefficients.
2. Calculate new coefficient values based on the error in the prediction.

The process is repeated until the model is accurate enough (e.g. error drops to some desirable level) or for a fixed number iterations. You continue to update the model for training instances and correcting errors until the model is accurate enough orc cannot be made any more accurate. It is often a good idea to randomize the order of the training instances shown to the model to mix up the corrections made.

By updating the model for each training pattern we call this online learning. It is also possible to collect up all of the changes to the model over all training instances and make one large update. This variation is called batch learning and might make a nice extension to this tutorial if you're feeling adventurous.

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.

$$B0 = 0.0$$

$$B1 = 0.0$$

$$B2 = 0.0$$

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:

$$prediction = 1 / (1 + e^{\wedge}(-(b0 + b1*x1 + b2*x2)))$$

$$prediction = 1 / (1 + e^{\wedge}(-(0.0 + 0.0*2.7810836 + 0.0*2.550537003)))$$

$$prediction = 0.5$$

Calculate New Coefficients

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

$$b = b + alpha * (y - prediction) * prediction * (1 - prediction) * x$$

Where b is the coefficient we are updating and prediction is the output of making a prediction using the model.

Alpha is 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.

$$b0 = b0 + 0.3 * (0 - 0.5) * 0.5 * (1 - 0.5) * 1.0$$

$$b1 = b1 + 0.3 * (0 - 0.5) * 0.5 * (1 - 0.5) * 2.7810836$$

$$b2 = b2 + 0.3 * (0 - 0.5) * 0.5 * (1 - 0.5) * 2.550537003$$

or

$$b0 = -0.0375$$

$$b1 = -0.104290635$$

$$b2 = -0.09564513761$$

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.
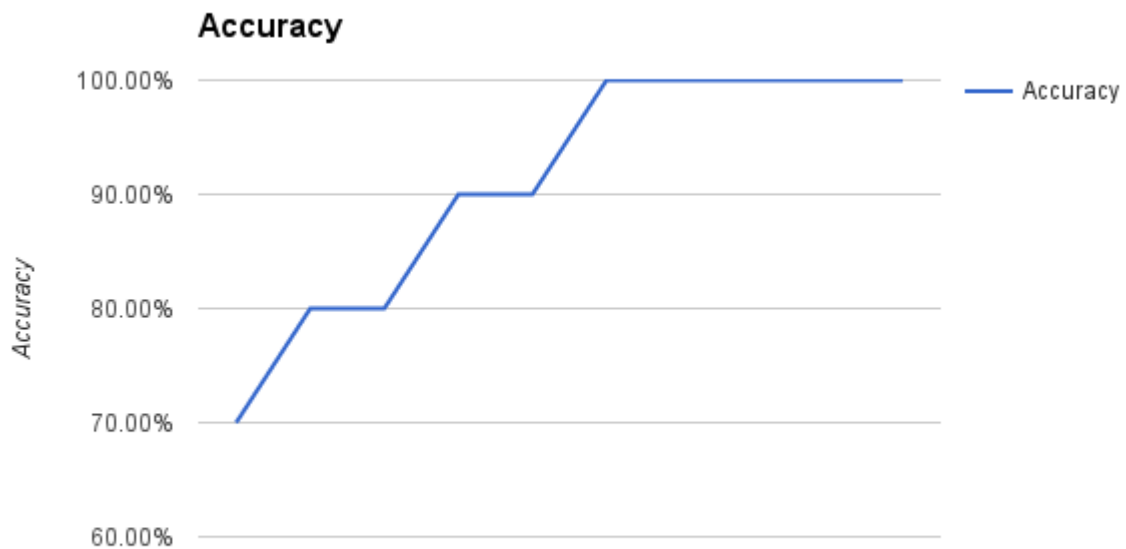
Logistic Regression with Gradient Descent Accuracy versus Iteration

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:

$$b0 = -0.4066054641$$

$$b1 = 0.8525733164$$

$$b2 = -1.104746259$$

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:

```
0.2987569857
0.145951056
0.08533326531
0.2197373144
```

1   0.2987569857
2   0.145951056

3 0.08533326531
4 0.2197373144
5 0.2470590002
6 0.9547021348
7 0.8620341908
8 0.9717729051
9 0.9992954521
10 0.905489323

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

$$prediction = IF\ (output < 0.5)\ Then\ 0\ Else\ 1$$

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

```
0
0
0
0
```
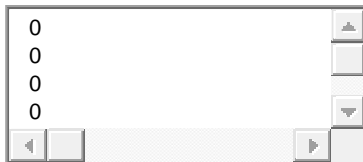
1 0
2 0
3 0
4 0
5 0
6 1
7 1
8 1
9 1
10 1

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

$$accuracy = (correct\ predictions\ /\ num\ predictions\ made) * 100$$

$$accuracy = (10\ /10) * 100$$

$$accuracy = 100\%$$

Summary

In this post 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.