WEEk-07: LOGISTIC REGRESSION USING STOCHASTIC GRADIENT DESCENT

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, you are going to discover the logistic regression algorithm for binary classification, step-by-step.

After reading this post 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.

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	Y
2.781084	2.550537	0
1.465489	2.362125	0
3.396562	4.400294	0
1.38807	1.85022	0
3.064072	3.005306	0
7.627531	2.759262	1
5.332441	2.088627	1
6.922597	1.771064	1
8.675419	-0.24207	1
7.673756	3.508563	1

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)

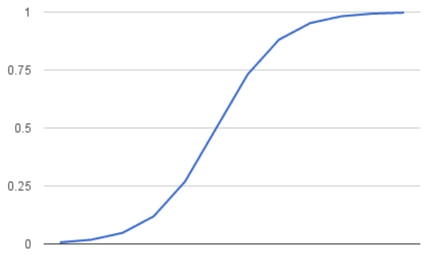
Where e is the numerical constant Euler’s number and x is a input we plug into the function.

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

X	Transformed
-5	0.006693
-4	0.017986
-3	0.047426
-2	0.119203
-1	0.268941
0	0.5
1	0.731059
2	0.880797
3	0.952574
4	0.982014
5	0.993307

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

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^(-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:

- Calculate a prediction using the current values of the coefficients.
- 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^(-(b0 + b1*x1 + b2*x2)))

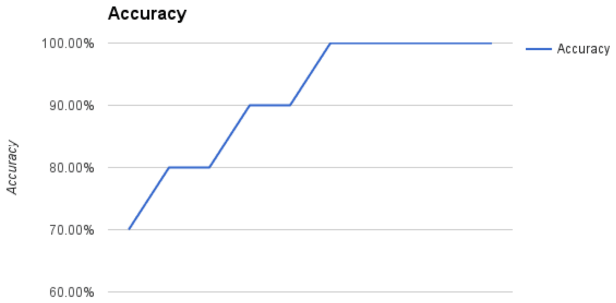
prediction = 1 / (1 + e^(-(0.0 + 0.0*2.7810836 + 0.0*2.550537003)))

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
0.2470590002
0.9547021348
0.8620341908
0.9717729051
0.9992954521
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
0
1
1
1
1
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%

Questions

1. Create the following data set for two independent variable (X1,X2) and one dependent variable (Y) in CSV. Apply the Logistic Regression to perform the following.
a. Calculate the coefficients (B0, B1 and B2).
b. Apply the sigmoid function to get the prediction and calculate error.
c. From the predicted values calculate the accuracy.
d. List the model parameters along with error for every instance of the training data.
e. Plot the graph of B1 v/s error and B2 v/s error.
f. Use scikit learn model to repeat the above steps and compare the results.

X1	**X2**	**Y**
2.781084	2.550537	0
1.465489	2.362125	0
3.396562	4.400294	0
1.38807	1.85022	0
3.064072	3.005306	0
7.627531	2.759262	1
5.332441	2.088627	1
6.922597	1.771064	1
8.675419	-0.24207	1
7.673756	3.508563	1

2. Use above data set for one independent variable (X=X1) and one dependent variable (Y) in CSV. Applying Logistic Regression, explore the relationship between independent and dependent variables.
a. Calculate the coefficients (B0, and B1).
b. Apply the sigmoid function to get the prediction and calculate error.
c. From the predicted values calculate the accuracy.
d. List the model parameters along with error for every instance of the training data.

e. Plot the graph of B1 v/s error.

f. Visualize the following binary cross entropy function for logistic regression using the above training data Plot y=1 and y=0 cases separately, and then plot the combined graph by considering X in X-axis and cost in Y-axis.

$$L = -\frac{1}{m}\sum_{i=1}^{m}(y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i))$$

g. Use scikit learn model to repeat the above steps and compare the results.

3. Use the above data set for two independent variable (X1,X2) and one dependent variable (Y) in CSV. Apply the Logistic Regression with SGD to perform the following.

a. Calculate the coefficients (B0, B1 and B2) and arrive at different values of B0, B1, B2, and error for 50 iterations of 5 epochs.

b. Apply the sigmoid function to get the prediction and calculate error.

c. From the predicted values calculate the accuracy.

d. Plot the graph of epoch (X-axis) v/s Accuracy (Y-axis).

f. Use scikit learn model to repeat the above steps and compare the results.

Additional Questions

Apply scikit learn model for Logistic regression using SGD of the given Salary_Data.csv dataset, and arrive at different values of B0, B1 and error for varying iterations. Plot the graph of epoch(X-axis) versus error(Y-axis).