

[Click to Take the FREE Probability Crash-Course](#)

Search...



A Gentle Introduction to Logistic Regression With Maximum Likelihood Estimation

by **Jason Brownlee** on October 28, 2019 in **Probability**



Tweet



Tweet



Share



Share

Last Updated on October 28, 2019

Logistic regression is a model for binary classification predictive modeling.

The parameters of a logistic regression model can be estimated by the probabilistic framework called [maximum likelihood estimation](#). Under this framework, a probability distribution for the target variable (class label) must be assumed and then a likelihood function defined that calculates the probability of observing the outcome given the input data and the model. This function can then be optimized to find the set of parameters that results in the largest sum likelihood over the training dataset.

The maximum likelihood approach to fitting a logistic regression model both aids in better understanding the form of the logistic regression model and provides a template that can be used for fitting classification models more generally. This is particularly true as the negative of the log-likelihood function used in the procedure can be shown to be equivalent to cross-entropy loss function.

In this post, you will discover logistic regression with maximum likelihood estimation.

After reading this post, you will know:

- Logistic regression is a linear model for binary classification predictive modeling.
- The linear part of the model predicts the log-odds of an example belonging to class 1, which is converted to a probability via the logistic function.

- The parameters of the model can be estimated by maximizing a likelihood function that predicts the mean of a Bernoulli distribution for each example.

Kick-start your project with my new book [Probability for Machine Learning](#), including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.



A Gentle Introduction to Logistic Regression With Maximum Likelihood Estimation

Photo by [Samuel John](#), some rights reserved.

Overview

This tutorial is divided into four parts; they are:

1. Logistic Regression
2. Logistic Regression and Log-Odds
3. Maximum Likelihood Estimation
4. Logistic Regression as Maximum Likelihood

Logistic Regression

Logistic regression is a classical linear method for binary classification.

Classification predictive modeling problems are those that require the prediction of a class label (e.g. 'red', 'green', 'blue') for a given set of input variables. Binary classification refers to those classification problems that have two class labels, e.g. true/false or 0/1.

Logistic regression has a lot in common with linear regression, although linear regression is a technique for predicting a numerical value, not for classification problems. Both techniques model the target variable with a line (or hyperplane, depending on the number of dimensions of input). Linear regression fits the line to the data, which can be used to predict a new quantity, whereas logistic regression fits a line to best separate the two classes.

The input data is denoted as X with n examples and the output is denoted y with one output for each input. The prediction of the model for a given input is denoted as \hat{y} .

- $\hat{y} = \text{model}(X)$

The model is defined in terms of parameters called coefficients (β), where there is one coefficient per input and an additional coefficient that provides the intercept or bias.

For example, a problem with inputs X with m variables x_1, x_2, \dots, x_m will have coefficients $\beta_1, \beta_2, \dots, \beta_m$, and β_0 . A given input is predicted as the weighted sum of the inputs for the example and the coefficients.

- $\hat{y} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \dots + \beta_m * x_m$

The model can also be described using linear algebra, with a vector for the coefficients (β) and a matrix for the input data (X) and a vector for the output (y).

- $y = X * \beta$

So far, this is identical to linear regression and is insufficient as the output will be a real value instead of a class label.

Instead, the model squashes the output of this weighted sum using a nonlinear function to ensure the outputs are a value between 0 and 1.

The [logistic function](#) (also called the sigmoid) is used, which is defined as:

- $f(x) = 1 / (1 + \exp(-x))$

Where x is the input value to the function. In the case of logistic regression, x is replaced with the weighted sum.

For example:

- $\hat{y} = 1 / (1 + \exp(-(X * \text{Beta})))$

The output is interpreted as a probability from a Binomial probability distribution function for the class labeled 1, if the two classes in the problem are labeled 0 and 1.



Notice that the output, being a number between 0 and 1, can be interpreted as a probability of belonging to the class labeled 1.

— Page 726, [Artificial Intelligence: A Modern Approach](#), 3rd edition, 2009.

The examples in the training dataset are drawn from a broader population and as such, this sample is known to be incomplete. Additionally, there is expected to be measurement error or statistical noise in the observations.

The parameters of the model (*beta*) must be estimated from the sample of observations drawn from the domain.

There are many ways to estimate the parameters. There are two frameworks that are the most common; they are:

- Least Squares Optimization ([iteratively reweighted least squares](#)).
- [Maximum Likelihood Estimation](#).

Both are optimization procedures that involve searching for different model parameters.

Maximum Likelihood Estimation is a frequentist probabilistic framework that seeks a set of parameters for the model that maximizes a likelihood function. We will take a closer look at this second approach in the subsequent sections.

Want to Learn Probability for Machine Learning

Take my free 7-day email crash course now (with sample code).

Click to sign-up and also get a free PDF Ebook version of the course.

Download Your FREE Mini-Course

Logistic Regression and Log-Odds

Before we dive into how the parameters of the model are estimated from data, we need to understand what logistic regression is calculating exactly.

This might be the most confusing part of logistic regression, so we will go over it slowly.

The linear part of the model (the weighted sum of the inputs) calculates the log-odds of a successful event, specifically, the log-odds that a sample belongs to class 1.

- $\text{log-odds} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \dots + \beta_m * x_m$

In effect, the model estimates the log-odds for class 1 for the input variables at each level (all observed values).

What are odds and log-odds?

Odds may be familiar from the field of gambling. Odds are often stated as wins to losses (wins : losses), e.g. a one to ten chance or ratio of winning is stated as 1 : 10.

Given the probability of success (p) predicted by the logistic regression model, we can convert it to [odds of success](#) as the probability of success divided by the probability of not success:

- $\text{odds of success} = p / (1 - p)$

The logarithm of the odds is calculated, specifically log base-e or the natural logarithm. This quantity is referred to as the log-odds and may be referred to as the logit (logistic unit), a unit of measure.

- $\text{log-odds} = \log(p / (1 - p))$

Recall that this is what the linear part of the logistic regression is calculating:

- $\text{log-odds} = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \dots + \beta_m * x_m$

The log-odds of success can be converted back into an odds of success by calculating the exponential of the log-odds.

- $\text{odds} = \exp(\text{log-odds})$

Or

- $\text{odds} = \exp(\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \dots + \beta_m * x_m)$

The odds of success can be converted back into a probability of success as follows:

- $p = \text{odds} / (\text{odds} + 1)$

And this is close to the form of our logistic regression model, except we want to convert log-odds to odds as part of the calculation.

We can do this and simplify the calculation as follows:

- $p = 1 / (1 + \exp(-\log\text{-odds}))$

This shows how we go from log-odds to odds, to a probability of class 1 with the logistic regression model, and that this final functional form matches the logistic function, ensuring that the probability is between 0 and 1.

We can make these calculations of converting between probability, odds and log-odds concrete with some small examples in Python.

First, let's define the probability of success at 80%, or 0.8, and convert it to odds then back to a probability again.

The complete example is listed below.

```
1 # example of converting between probability and odds
2 from math import log
3 from math import exp
4 # define our probability of success
5 prob = 0.8
6 print('Probability %.1f' % prob)
7 # convert probability to odds
8 odds = prob / (1 - prob)
9 print('Odds %.1f' % odds)
10 # convert back to probability
11 prob = odds / (odds + 1)
12 print('Probability %.1f' % prob)
```

Running the example shows that 0.8 is converted to the odds of success 4, and back to the correct probability again.

```
1 Probability 0.8
2 Odds 4.0
3 Probability 0.8
```

Let's extend this example and convert the odds to log-odds and then convert the log-odds back into the original probability. This final conversion is effectively the form of the logistic regression model, or the logistic function.

The complete example is listed below.

```
1 # example of converting between probability and log-odds
2 from math import log
3 from math import exp
4 # define our probability of success
5 prob = 0.8
6 print('Probability %.1f' % prob)
7 # convert probability to odds
8 odds = prob / (1 - prob)
9 print('Odds %.1f' % odds)
10 # convert odds to log-odds
11 logodds = log(odds)
12 print('Log-Odds %.1f' % logodds)
13 # convert log-odds to a probability
14 prob = 1 / (1 + exp(-logodds))
15 print('Probability %.1f' % prob)
```

Running the example, we can see that our odds are converted into the log odds of about 1.4 and then correctly converted back into the 0.8 probability of success.

```
1 Probability 0.8
2 Odds 4.0
3 Log-Odds 1.4
4 Probability 0.8
```

Now that we have a handle on the probability calculated by logistic regression, let's look at maximum likelihood estimation.

Maximum Likelihood Estimation

[Maximum Likelihood Estimation](#), or MLE for short, is a probabilistic framework for estimating the parameters of a model.

In Maximum Likelihood Estimation, we wish to maximize the conditional probability of observing the data (X) given a specific probability distribution and its parameters (θ), stated formally as:

- $P(X ; \theta)$

Where X is, in fact, the joint probability distribution of all observations from the problem domain from 1 to n .

- $P(x_1, x_2, x_3, \dots, x_n ; \theta)$

This resulting conditional probability is referred to as the likelihood of observing the data given the model parameters and written using the notation $L(\theta)$ to denote the [likelihood](#)

function. For example:

- $L(X ; \theta)$

The joint probability distribution can be restated as the multiplication of the conditional probability for observing each example given the distribution parameters. Multiplying many small probabilities together can be unstable; as such, it is common to restate this problem as the sum of the log conditional probability.

- $\sum_{i=1}^n \log(P(x_i ; \theta))$

Given the frequent use of log in the likelihood function, it is referred to as a log-likelihood function. It is common in optimization problems to prefer to minimize the cost function rather than to maximize it. Therefore, the negative of the log-likelihood function is used, referred to generally as a Negative Log-Likelihood (NLL) function.

- minimize $-\sum_{i=1}^n \log(P(x_i ; \theta))$

The Maximum Likelihood Estimation framework can be used as a basis for estimating the parameters of many different machine learning models for regression and classification predictive modeling. This includes the logistic regression model.

Logistic Regression as Maximum Likelihood

We can frame the problem of fitting a machine learning model as the problem of probability density estimation.

Specifically, the choice of model and model parameters is referred to as a modeling hypothesis h , and the problem involves finding h that best explains the data X . We can, therefore, find the modeling hypothesis that maximizes the likelihood function.

- maximize $\sum_{i=1}^n \log(P(x_i ; h))$

Supervised learning can be framed as a conditional probability problem of predicting the probability of the output given the input:

- $P(y | X)$

As such, we can define conditional maximum likelihood estimation for supervised machine learning as follows:

- maximize $\sum_{i=1}^n \log(P(y_i|x_i ; h))$

Now we can replace h with our logistic regression model.

In order to use maximum likelihood, we need to assume a probability distribution. In the case of logistic regression, a Binomial probability distribution is assumed for the data sample, where each example is one outcome of a Bernoulli trial. The Bernoulli distribution has a single parameter: the probability of a successful outcome (p).

- $P(y=1) = p$
- $P(y=0) = 1 - p$

“The probability distribution that is most often used when there are two classes is the binomial distribution.⁵ This distribution has a single parameter, p , that is the probability of an event or a specific class.

— Page 283, [Applied Predictive Modeling](#), 2013.

The expected value (mean) of the Bernoulli distribution can be calculated as follows:

- $\text{mean} = P(y=1) * 1 + P(y=0) * 0$

Or, given p :

- $\text{mean} = p * 1 + (1 - p) * 0$

This calculation may seem redundant, but it provides the basis for the likelihood function for a specific input, where the probability is given by the model (\hat{y}) and the actual label is given from the dataset.

- $\text{likelihood} = \hat{y} * y + (1 - \hat{y}) * (1 - y)$

This function will always return a large probability when the model is close to the matching class value, and a small value when it is far away, for both $y=0$ and $y=1$ cases.

We can demonstrate this with a small worked example for both outcomes and small and large probabilities predicted for each.

The complete example is listed below.

```
1 # test of Bernoulli likelihood function
2
3 # likelihood function for Bernoulli distribution
4 def likelihood(y, yhat):
5     return yhat * y + (1 - yhat) * (1 - y)
6
7 # test for y=1
8 y, yhat = 1, 0.9
9 print('y=%.1f, yhat=%.1f, likelihood: %.3f' % (y, yhat, likelihood(y, yhat)))
```


```

10 y, yhat = 1, 0.1
11 print('y=%.1f, yhat=%.1f, likelihood: %.3f' % (y, yhat, likelihood(y, yhat)))
12 # test for y=0
13 y, yhat = 0, 0.1
14 print('y=%.1f, yhat=%.1f, likelihood: %.3f' % (y, yhat, likelihood(y, yhat)))
15 y, yhat = 0, 0.9
16 print('y=%.1f, yhat=%.1f, likelihood: %.3f' % (y, yhat, likelihood(y, yhat)))

```

Running the example prints the class labels (*y*) and predicted probabilities (*yhat*) for cases with close and far probabilities for each case.

We can see that the likelihood function is consistent in returning a probability for how well the model achieves the desired outcome.



```

1 y=1.0, yhat=0.9, likelihood: 0.900
2 y=1.0, yhat=0.1, likelihood: 0.100
3 y=0.0, yhat=0.1, likelihood: 0.900
4 y=0.0, yhat=0.9, likelihood: 0.100

```

We can update the likelihood function using the log to transform it into a log-likelihood function:

- $\text{log-likelihood} = \log(\text{yhat}) * y + \log(1 - \text{yhat}) * (1 - y)$

Finally, we can sum the likelihood function across all examples in the dataset to maximize the likelihood:

- $\text{maximize sum } i \text{ to } n \log(\text{yhat}_i) * y_i + \log(1 - \text{yhat}_i) * (1 - y_i)$

It is common practice to minimize a cost function for optimization problems; therefore, we can invert the function so that we minimize the negative log-likelihood:

- $\text{minimize sum } i \text{ to } n -(\log(\text{yhat}_i) * y_i + \log(1 - \text{yhat}_i) * (1 - y_i))$

Calculating the negative of the log-likelihood function for the Bernoulli distribution is equivalent to calculating the [cross-entropy](#) function for the Bernoulli distribution, where $p()$ represents the probability of class 0 or class 1, and $q()$ represents the estimation of the probability distribution, in this case by our logistic regression model.

- $\text{cross entropy} = -(\log(q(\text{class0})) * p(\text{class0}) + \log(q(\text{class1})) * p(\text{class1}))$

Unlike linear regression, there is not an analytical solution to solving this optimization problem. As such, an iterative optimization algorithm must be used.

“ Unlike linear regression, we can no longer write down the MLE in closed form. Instead, we need to use an optimization algorithm to compute it. For this, we need

to derive the gradient and Hessian.

— Page 246, [Machine Learning: A Probabilistic Perspective](#), 2012.

The function does provide some information to aid in the optimization (specifically a Hessian matrix can be calculated), meaning that efficient search procedures that exploit this information can be used, such as the [BFGS algorithm](#) (and variants).

Further Reading

This section provides more resources on the topic if you are looking to go deeper.

Posts

- [A Gentle Introduction to Maximum Likelihood Estimation for Machine Learning](#)
- [How To Implement Logistic Regression From Scratch in Python](#)
- [Logistic Regression Tutorial for Machine Learning](#)
- [Logistic Regression for Machine Learning](#)

Books

- Section 4.4.1 Fitting Logistic Regression Models, [The Elements of Statistical Learning](#), 2016.
- Section 4.3.2 Logistic regression, [Pattern Recognition and Machine Learning](#), 2006.
- Chapter 8 Logistic regression, [Machine Learning: A Probabilistic Perspective](#), 2012.
- Chapter 4 Algorithms: the basic methods, [Data Mining: Practical Machine Learning Tools and Techniques](#), 4th edition, 2016.
- Section 18.6.4 Linear classification with logistic regression, [Artificial Intelligence: A Modern Approach](#), 3rd edition, 2009.
- Section 12.2 Logistic Regression, [Applied Predictive Modeling](#), 2013.
- Section 4.3 Logistic Regression, [An Introduction to Statistical Learning with Applications in R](#), 2017.

Articles

- [Maximum likelihood estimation](#), Wikipedia.
- [Likelihood function](#), Wikipedia.
- [Logistic regression](#), Wikipedia.
- [Logistic function](#), Wikipedia.
- [Odds](#), Wikipedia.

Summary

In this post, you discovered logistic regression with maximum likelihood estimation.

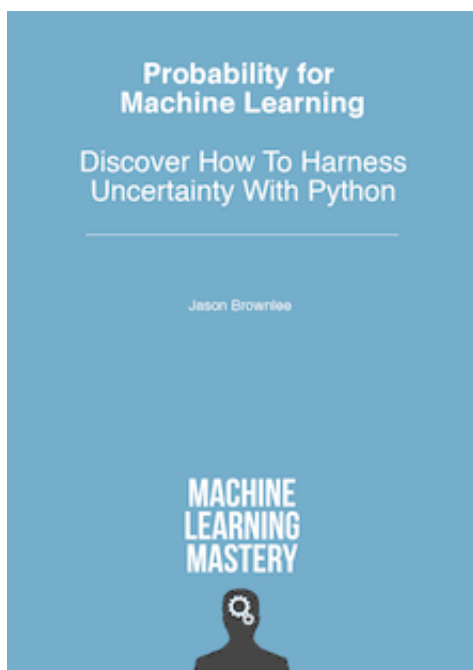
Specifically, you learned:

- Logistic regression is a linear model for binary classification predictive modeling.
- The linear part of the model predicts the log-odds of an example belonging to class 1, which is converted to a probability via the logistic function.
- The parameters of the model can be estimated by maximizing a likelihood function that predicts the mean of a Bernoulli distribution for each example.

Do you have any questions?

Ask your questions in the comments below and I will do my best to answer.

Get a Handle on Probability for Machine Learning!



Develop Your Understanding of Probability

...with just a few lines of python code

Discover how in my new Ebook:
[Probability for Machine Learning](#)

It provides **self-study tutorials** and **end-to-end projects** on:
Bayes Theorem, Bayesian Optimization, Distributions, Maximum Likelihood, Cross-Entropy, Calibrating Models
and much more...

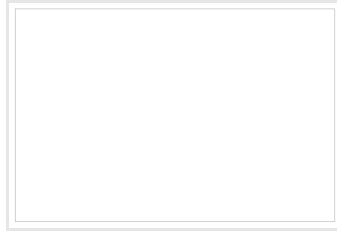
Finally Harness Uncertainty in Your Projects

Skip the Academics. Just Results.

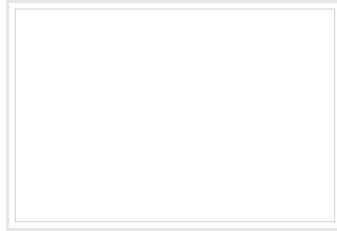
SEE WHAT'S INSIDE



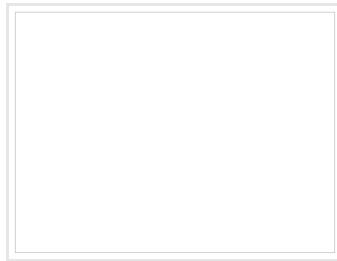
More On This Topic



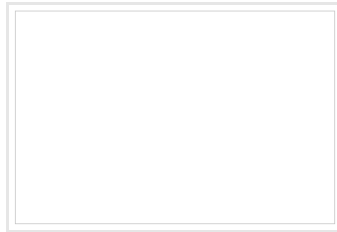
A Gentle Introduction to Linear Regression With...



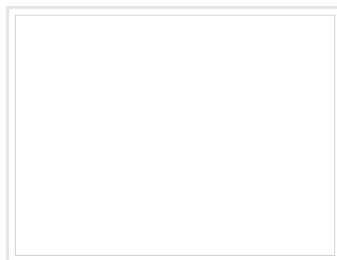
A Gentle Introduction to Maximum Likelihood...



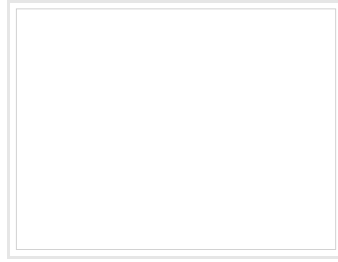
Multinomial Logistic Regression With Python



Logistic Regression for Machine Learning



A Gentle Introduction to Cross-Entropy for Machine Learning



A Gentle Introduction to Expectation-Maximization...

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.

[View all posts by Jason Brownlee →](#)

< A Gentle Introduction to Linear Regression With Maximum Likelihood Estimation

Probabilistic Model Selection with AIC, BIC, and MDL >

38 Responses to *A Gentle Introduction to Logistic Regression With Maximum Likelihood Estimation*

Norman November 1, 2019 at 4:18 pm #

REPLY ↩

Hi, Jason

One suggestion. Could you consider to use the notation " $\hat{y} * (1 - \hat{y})$ " to define the likelihood? This one has the same result as the original one for Bernoulli distribution. But this one is easier for calculating log-likelihood by math. These links below for your reference.

<https://web.stanford.edu/class/cs109/reader/11%20Parameter%20Estimation.pdf>

<https://stats.stackexchange.com/questions/275380/maximum-likelihood-estimation-for-bernoulli-distribution>

Jason Brownlee November 2, 2019 at 6:39 am #

REPLY ↩

Thanks for the suggestion Norman.

REPLY REPLY REPLY

The second part of your question is about the likelihood function for a Bernoulli trial.

E.g. we are reporting a probability of matching the positive outcome. It's redundant, but it provides the basis for the log likelihood that follows – in that section I want to see that it's not doing anything exciting.

Adam December 13, 2019 at 2:50 pm #

REPLY ↩

Does $\hat{y}_i = P(y_i|x_i; h)$?

Jason Brownlee December 14, 2019 at 6:05 am #

REPLY ↩

Yes.

Robert August 17, 2020 at 10:51 pm #

REPLY ↩

Hmm... isn't $P(y_i|x_i; h) = \hat{y}_i y_i$?

Or what am I missing in the last example at the very end when the (log-)likelihood is calculated going from the expression

maximize $\sum_i \log(P(y_i|x_i; h))$, introduced earlier to the expression

maximize $\sum_i \log(\hat{y}_i * y_i + \log(1 - \hat{y}_i) * (1 - y_i))$

Omar December 21, 2019 at 9:00 pm #

REPLY ↩

Hello Jason, I have been asking a lot in your LSTM posts, thanks for replying 😊

I have some conceptual questions to ask. Mostly referring to log-odds with natural logarithm is written as $\ln(\text{prob_event} / (1 - \text{prob_event})) = b_0 + b_1 * X_1 + \dots + b_n * X_n$

1. When dealing with real data, how do I know the probability of the event?
2. Suppose that I have no idea about the probability of the event. I assumed we can calculate the log-odds by fitting multiple linear regression (please correct me if I am wrong) since the right hand side of the equation above is a multiple linear regression. What should I do to calculate the beta (b_0, b_1, \dots, b_n) ? I understand how to get the values using python, but have no idea to calculate them manually. I also know that we can fit the logistic regression using Maximum Likelihood Estimation but I don't know how to do it manually.
3. Also suppose that I have a dataset with 100 rows, divided into 20 windows with each

window containing 5 rows to do classification with labels corresponding to the window. Logistic Regression from sklearn can classify them. Do you have any intuition to guide me how can sklearn's logistic regression do this? What I understand is that after we have the beta, we can easily plug the data into X, but I don't know what actually happens if the value we want to plug to X is not a single row (in this case, 5 rows).

Thank you very much for your assistance.



Jason Brownlee December 22, 2019 at 6:11 am #

REPLY ↩

Good questions!

We estimate the probability of an event from historical observations (Frequency), or we use domain expertise to define our belief about the likelihood of an event (Bayesian).

In a model, we can assume a likelihood distribution over events, and guess at the probability of new events. This is what we do in logistic regression.

Coefficients are optimized to minimize log loss on a training dataset. No need to worry about the coefficients for a single observation.

A prediction is made by multiplying input by the coefficients.



Omar December 22, 2019 at 6:32 pm #

REPLY ↩

Ah I see, thanks!

Please let me rephrase the third question.

Assume we have $Y = b_0 + b_1X_1$ (a logistic regression model with only one predictor). Let's say that my data is only 20 samples with 20 target variable, with each sample contain 5 rows (so that the total rows is 100).

My question is, what is the math behind fitting/predicting samples with multiple rows inside?

For example, if $b_0 = 2$, $b_1 = 0.5$, and $X_1 = 4$, I can calculate $Y = 2 + 0.5 * 4$. But if X_1 is a list $[2,3,4]$, I don't know the math to predict it. Plugging more than one row as a sample in sklearn seems fine (no error or warning shown)



Jason Brownlee December 23, 2019 at 6:47 am #

REPLY ↩

Sorry, I don't follow.

Logistic regression assumes you have one target variable with 2 classes and some number of input variables. I don't know what "20 samples with 20 target variable, with each sample contain 5 rows" means.



Omar December 23, 2019 at 10:06 pm #

Sorry, my bad. I'll show more detailed explanation. The context is about predictive maintenance.

Please refer to this image: <https://imgur.com/VHikbUn>

Given 6 observations, I want to make "windows of observations" which each window contains 3 observations. In this case, I have 2 samples, the blue and orange one.

The blue window will be assigned class 0 (not fault), and orange with class 1 (fault).

This means I will have 2 samples (blue and orange).

My goal is whenever I pass 3 observation, my model will see them as "one window" and will predict whether it is a fault or not.

Using fit method in sklearn Logistic Regression, this means X has two samples (blue and orange), and y also two samples (0 and 1).

I successfully fitted them. My question may be answered from two perspective:

1. From python or scikit-learn perspective, how is it not throwing any error?
2. From statistics perspective, how does one estimate beta parameters and also do prediction with X similar with my case?

This is my best effort to explain the case. Very much appreciated your help Jason.



Jason Brownlee December 24, 2019 at 6:42 am #

Thanks for elaborating.

We would call the 3 numbers a sample, not a window.

This is a standard prediction problem, you pass samples of 3 number and predict the class. You can do this manually or via sklearn.

The parameters will be estimated from training data.

Omar December 24, 2019 at 12:28 am #

REPLY ↩

Dear Jason, please ignore my last question, looks like I made an error on the code that somehow makes it able to be processed and misunderstood the complete context of the problem.

Jason Brownlee December 24, 2019 at 6:42 am #

REPLY ↩

No problem.

Omar December 25, 2019 at 2:22 am #

REPLY ↩

Dear Jason, I think now I have a bit of insights about my case above. Please refer again to this image: <https://imgur.com/VHikbUn>

I was able to construct the data to be 3-dimensional array (can be plugged into LSTM models), with shape (2,3,1) for (samples, timesteps, feature).

To make sure I can pass the sample (with three numbers inside like blue or orange) to sklearn classifier, I reshaped the data with this code:

```
nsamples, nx, ny = sample_array.shape
new_sample_array = sample_array.reshape((nsamples,nx*ny)) #could also be done using
.reshape((nsamples, -1))
```

In my understanding, this will cause my new_sample_array having shape of (2,3). It seems that the three rows inside my sample turned into three columns. I assumed that the columns mean first sample with first time steps, first sample with second time steps, and so on. (Please also refer to this image for the reshaping reference: <https://imgur.com/E3G4rLb>)

Therefore, I'd like to ask two questions:

1. Is my reshaping method correct?
2. If I pass the reshaped data into Logistic Regression (let's say the classifier is clf) and do `clf.coef_`, I got an array with three values. Does this corresponds to `b_1`, `b_2`, and `b_3`?

Thank you very much for your help

Jason Brownlee December 25, 2019 at 10:37 am #

REPLY ↩

An LSTM would not be appropriate as it is tabular data, not a sequence.

More on what models to use when here:

<https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/>

Omar December 25, 2019 at 1:22 pm #

REPLY ↩

Sorry I left an important note. The data was actually a time series data from a sensor with each row represents the sensor's value in time t (collected every several minutes).

But the point I want to ask here is not about LSTM, it is actually about the two questions about reshaping and passing the reshaped data into sklearn's classifier.

I hope this clarifies the circumstances.

Thank you Jason.

Jason Brownlee December 26, 2019 at 7:33 am #

REPLY ↩

For more on how to prepare data for LSTMs, see this
<https://machinelearningmastery.com/faq/single-faq/what-is-the-difference-between-samples-timesteps-and-features-for-lstm-input>

Grzegorz Kępisty April 21, 2020 at 4:06 pm #

REPLY ↩

Good morning Jason!

Thank you for the article. Two questions on the topic:

- 1) In this blog entry you describe that the model can be optimized by maximizing the likelihood function for given input data. On the other hand there ("How To Implement Logistic Regression From Scratch in Python") you show that we can optimize a model by minimizing error of predictions. When and how do we choose between those 2 aim functions?
- 2) What would be the difference between those models optimized in two different ways (maximum likelihood or minimizing the error)? If error is minimal in the second case why shall we use likelihood?

Thank you in advance for your response and have a pleasant day!

Jason Brownlee April 22, 2020 at 5:50 am #

REPLY ↩

You're welcome.

They both solve the problem the same general way, just with different optimization algorithms.

Use the solver if the data fits in ram, use SGD if it doesn't.

Sabine Katzdobler July 20, 2020 at 10:12 pm #

REPLY ↩

Good day,

currently, I started to rethink my usage of the (ordinal) logit distribution and wondered if maximum-likelihood-distribution might be better suited.

Thus, I am asking: Would you recommend maximum likelihood estimation (with Bernoulli-distribution), when the dependent variable is ordinal involving 5 levels?

Jason Brownlee July 21, 2020 at 6:03 am #

REPLY ↩

The distribution does not matter for the framework.

I believe for a binomial distribution, you will arrive at a cross-entropy loss.

Tristan October 11, 2020 at 9:58 pm #

REPLY ↩

Hi Jason,

Firstly, thanks a lot for the insightful post. I have one question which I am trying to find an answer to and that no searches have provided insight on. You stated "Recall that this is what the linear part of the logistic regression is calculating:

$$\text{log-odds} = \text{beta0} + \text{beta1} * x_1 + \text{beta2} * x_2 + \dots + \text{betam} * x_m$$

Where is the step that shows we can state that the log odds is equal to linear equation? I have gone through 5 derivations and they all do the same thing as you have done. I am curious to understand how that statement is derived.

Kind regards,
Tristan

Jason Brownlee October 12, 2020 at 6:43 am #

REPLY ↩

It is a sum of weighted terms which by definition is a linear equation.

Tarun Gupta January 19, 2021 at 8:50 pm #

REPLY ↩

Same question !! How does these two become equal. "sum of weighted terms which by definition is a linear equation" true but how they became equal ?

Jason Brownlee January 20, 2021 at 5:42 am #

REPLY ↩

We are defining a term, it is the log odds defined as a weighted sum.

Chew Jing Wei October 29, 2020 at 2:38 pm #

REPLY ↩

Hello! Thank you for the post, your explanations are very clear.

However, I think there might be a mistake in this equation: $\text{likelihood} = \hat{y} * y + (1 - \hat{y}) * (1 - y)$

Based on the lecture notes here:

<http://web.stanford.edu/class/archive/cs/cs109/cs109.1178/lectureHandouts/220-logistic-regression.pdf>, the likelihood based on a single observation should be $\hat{y} ** y * (1 - \hat{y}) ** (1 - y)$ instead.

Jason Brownlee October 30, 2020 at 6:48 am #

REPLY ↩

I believe it is correct, I recommend the references at the end of the tutorial.

Harrison Bohl November 5, 2020 at 12:52 pm #

REPLY ↩

Please use latex to write your maths equations, it's really hard to understand what is happening and also it looks bad. Thanks

Jason Brownlee November 5, 2020 at 1:05 pm #

REPLY ↩

Great suggestion, thanks.

Jon March 14, 2021 at 9:36 am #

REPLY ↩

How do you use the above (Logistic Regression Likelihood Function) to calculate AIC or BIC

Jason Brownlee March 15, 2021 at 5:51 am #

REPLY ↩

Perhaps this will help:

<https://machinelearningmastery.com/probabilistic-model-selection-measures/>

Paul September 30, 2021 at 8:19 pm #

REPLY ↩

Is the this method an alternative to Gradient Descent? A bit confused by the difference between the two.
Thank you!

Adrian Tam October 1, 2021 at 12:43 pm #

REPLY ↩

If you mean logistic regression and gradient descent, the answer is no. Logistic regression is to take input and predict output, but not in a linear model. Gradient descent is an algorithm to do optimization. In this case, we optimize for the likelihood score by comparing the logistic regression prediction and the real output data.

John December 20, 2021 at 1:21 am #

REPLY ↩

How do I calculate the intercept and coefficients of logistic regression?



James Carmichael February 15, 2022 at 12:41 pm <#>

REPLY

Hi John...The following should help add clarity on this process.

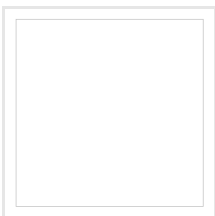
<https://www.theanalysisfactor.com/interpreting-the-intercept-in-a-regression-model/>

Leave a Reply

Name (required)

Email (will not be published) (required)

SUBMIT COMMENT



Welcome!

I'm *Jason Brownlee* PhD

and I **help developers** get results with **machine learning**.

[Read more](#)

Never miss a tutorial:



Picked for you:

[How to Use ROC Curves and Precision-Recall Curves for Classification in Python](#)

[How and When to Use a Calibrated Classification Model with scikit-learn](#)

[How to Implement Bayesian Optimization from Scratch in Python](#)

[How to Calculate the KL Divergence for Machine Learning](#)

[A Gentle Introduction to Cross-Entropy for Machine Learning](#)

Loving the Tutorials?

The [Probability for Machine Learning](#) EBook is where you'll find the ***Really Good*** stuff.

>> SEE WHAT'S INSIDE

© 2021 Machine Learning Mastery. All Rights Reserved.

[LinkedIn](#) | [Twitter](#) | [Facebook](#) | [Newsletter](#) | [RSS](#)

[Privacy](#) | [Disclaimer](#) | [Terms](#) | [Contact](#) | [Sitemap](#) | [Search](#)