# **Logistic Regression**

Binary Logistic Regression is a special type of regression where a binary response variable is related to a set of explanatory variables, which can be discrete and/or continuous. The important point here to note is that in linear regression, the expected values of the response variable are modeled based on combination of values taken by the predictors. In logistic regression *Probability* or *Odds* of the response taking a particular value is modeled based upon a combination of values taken by the predictors.

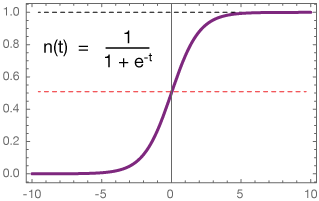
In Apache Ignite ML module this is implemented via LogisticRegressionModel that solves a binary classification problem. It is a linear method as described in equation (1), with the loss function in the formulation given by the logistic loss:

L(\wv;\x,y) := \log(1+\exp( -y \wv^T \x)). (Tex formula)

L(w;x,y):=log(1+exp(−ywTx)).

For binary classification problems, the algorithm outputs a binary logistic regression model. Given a new data point, denoted by x, the model makes predictions by applying the logistic function:

|  |
| --- |
| \mathrm{f}(z) = \frac{1}{1 + e^{-z}} (Tex-command) |

or picture 

where z=wTxz=wTx. or

|  |
| --- |
| z = \wv^T \x (in Tex) |

By default, if f(wTx)>0.5 or \mathrm{f}(\wv^T x) > 0.5 (Tex formula), the outcome is positive, or negative otherwise, though unlike linear SVMs, the raw output of the logistic regression model f(z) has a probabilistic interpretation (i.e., the probability that is positive).

## **Model**

A Model in the case of Binary Logistic Regression is represented by the class LogisticRegressionModel. This enables a prediction to be made for a given vector of features, in the following way:

LogisticRegressionModel mdl = …

**double** prediction = mdl.withRawLabels(true).withThreshold(0.5).apply(**observation**)

Presently Ignite supports several parameters for LogisticRegressionModel :

* isKeepingRawLabels - controls the output label format: 0 and 1 for false value and raw distances from the separating hyperplane otherwise (default value: false)
* threshold - a threshold to assign label ‘1’ to the observation if the raw value is more than this threshold (default value: 0.5)

LogisticRegressionModel mdl = …

**double** prediction = mdl.withRawLabels(true).withThreshold(0.5).apply(**observation**)

## **Trainer**

Trainer of the logistic regression model is based on a stochastic gradient descent algorithm and internally uses MultilayerPerceptron with the simplest configuration and activation function.

Presently, Ignite supports the following parameters for LogisticRegressionSGDTrainer

* **updatesStgy** - update strategy
* **maxIterations - max amount of iterations before convergence**
* **batchSize - The size of learning batch**
* **locIterations - the amount of local iterations of SGD algorithm**
* **seed - seed value for internal random purposes to reproduce training results**

// Set up the trainer  
LogisticRegressionSGDTrainer<?> trainer = new LogisticRegressionSGDTrainer<>(UPDATES\_STRATEGY, MAX\_ITERATIONS, BATCH\_SIZE, LOC\_ITERATIONS, SEED);  
  
 // Build the model  
 LogisticRegressionModel mdl = trainer.fit(  
 ignite,  
 dataCache,  
 featureExtractor,  
 labelExtractor

);

# **Example**

To see how LogisticRegressionModel can be used in practice, try this [example](https://github.com/apache/ignite/blob/master/examples/src/main/java/org/apache/ignite/examples/ml/regression/logistic/binary/LogisticRegressionSGDTrainerExample.java) that is available on GitHub and delivered with every Apache Ignite distribution.

The training dataset is the subset of the Iris dataset (classes with labels 1 and 2, which are presented linear separable two-classes dataset) which could be loaded from the [UCI Machine Learning Repository](https://archive.ics.uci.edu/ml/datasets/iris).