Logistic Regression

NAME

Logistic Regression - A classification algorithm supported by Frovedis to predict the binary output with logistic loss.

SYNOPSIS

```
#include <frovedis/ml/glm/logistic_regression_with_sgd.hpp>
```

```
logistic_regression_model<T>
logistic_regression_with_sgd::train (crs_matrix<T>& data,
      dvector<T>& label,
      size_t numIteration = 1000,
      T alpha = 0.01,
      T miniBatchFraction = 1.0,
      T regParam = 0.01,
      RegType regtyp = ZERO,
      bool isIntercept = false,
      T convergenceTol = 0.001,
      MatType mType = HYBRID)
```

```
logistic_regression_model<T>
logistic_regression_with_sgd::train (crs_matrix<T>& data,
      dvector<T>& label,
      logistic_regression_model<T>& initModel,
      size_t numIteration = 1000,
      T alpha = 0.01,
      T miniBatchFraction = 1.0,
      T regParam = 0.01,
      RegType regtyp = ZERO,
      bool isIntercept = false,
      T convergenceTol = 0.001,
      MatType mType = HYBRID)
```

```
#include <frovedis/ml/glm/logistic_regression_with_lbfgs.hpp>
```

```
logistic_regression_model<T>
logistic_regression_with_lbfgs::train (crs_matrix<T>& data,
      dvector<T>& label,
      size_t numIteration = 1000,
      T alpha = 0.01,
      size_t hist_size = 10,
      T regParam = 0.01,
      RegType regtyp = ZERO,
```

bool isIntercept = false,
T convergenceTol = 0.001,
MatType mType = HYBRID)

`logistic_regression_model<T>`
logistic_regression_with_lbfgs::train (`crs_matrix<T>`& data,
`dvector<T>`& label,
`logistic_regression_model<T>`& initModel,
size_t numIteration = 1000,
T alpha = 0.01,
size_t hist_size = 10,
T regParam = 0.01,
RegType regtyp = ZERO,
bool isIntercept = false,
T convergenceTol = 0.001,
MatType mType = HYBRID)

DESCRIPTION

Classification aims to divide items into categories. The most common classification type is binary classification, where there are two categories, usually named positive and negative. Frovedis supports binary classification algorithm only.

Logistic regression is widely used to predict a binary response. It is a linear method with the loss function given by the **logistic loss**:

```
L(w;x,y) := log(1 + exp(-ywTx))
```

Where the vectors x are the training data examples and y are their corresponding labels (can be either -1 for negative response or 1 for positive response) which we want to predict. w is the linear model (also called as weight) which uses a single weighted sum of features to make a prediction. Logistic Regression supports ZERO, L1 and L2 regularization to address the overfit problem.

The gradient of the logistic loss is: -y(1 - 1 / (1 + exp(-ywTx))).x
The gradient of the L1 regularizer is: sign(w)
And The gradient of the L2 regularizer is: w

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:

```
f(z) := 1 / 1 + exp(-z)
```

Where z = wTx. By default, if f(wTx) > 0.5, the response is positive (1), else the response is negative (-1).

Frovedis provides implementation of logistic regression with two different optimizers: (1) stochastic gradient descent with minibatch and (2) LBFGS optimizer.

The simplest method to solve optimization problems of the form **min f(w)** is gradient descent. Such first-order optimization methods well-suited for large-scale and distributed computation. Whereas, L-BFGS is an optimization algorithm in the family of quasi-Newton methods to solve the optimization problems of the similar form.

Like the original BFGS, L-BFGS (Limited Memory BFGS) uses an estimation to the inverse Hessian matrix to steer its search through feature space, but where BFGS stores a dense nxn approximation to the inverse Hessian (n being the number of features in the problem), L-BFGS stores only a few vectors that represent the approximation implicitly. L-BFGS often achieves rapider convergence compared with other first-order optimization.

Detailed Description

logistic_regression_with_sgd::train()

Parameters
data: A `crs_matrix<T>` containing the sparse feature matrix
label: A `dvector<T>` containing the output labels
numIteration: A size_t parameter containing the maximum number of iteration count (Default: 1000)
alpha: A parameter of T type containing the learning rate (Default: 0.01)
minibatchFraction: A parameter of T type containing the minibatch fraction (Default: 1.0)
regParam: A parameter of T type containing the regularization parameter (also called lambda) (Default: 0.01)
regtyp: A parameter of the type frovedis::RegType, which can be either ZERO, L1 or L2 (Default: ZERO)
isIntercept: A boolean parameter to specify whether to include intercept term (bias term) or not (Default: false)
convergenceTol: A parameter of T type containing the threshold value to determine the convergence (Default: 0.001)
mType: frovedis::MatType parameter specifying the matrix type to be used for internal calculation (Default: HYBRID for SX architecture, CRS for other architectures)

Purpose
It trains a logistic regression model with stochastic gradient descent with minibatch optimizer and with provided regularizer (if not ZERO). It starts with an initial guess of zeros for the model vector and keeps updating the model to minimize the cost function until convergence is achieved or maximum iteration count is reached. After the training, it returns the trained output model.

Return Value
After the successful training, it returns a trained model of the type `logistic_regression_model<T>`.

logistic_regression_with_sgd::train()

Parameters
data: A `crs_matrix<T>` containing the sparse feature matrix
label: A `dvector<T>` containing the output labels
initModel: A `logistic_regression_model<T>` containing the user provided initial model values
numIteration: A size_t parameter containing the maximum number of iteration count (Default: 1000)
alpha: A parameter of T type containing the learning rate (Default: 0.01)
minibatchFraction: A parameter of T type containing the minibatch fraction (Default: 1.0)
regParam: A parameter of T type containing the regularization parameter (also called lambda) (Default: 0.01)
regtyp: A parameter of the type frovedis::RegType, which can be either ZERO, L1 or L2 (Default: ZERO)
isIntercept: A boolean parameter to specify whether to include intercept term (bias term) or not (Default: false)
convergenceTol: A parameter of T type containing the threshold value to determine the convergence (Default: 0.001)
mType: frovedis::MatType parameter specifying the matrix type to be used for internal calculation (Default: HYBRID for SX architecture, CRS for other architectures)

Purpose
It trains a logistic regression model with stochastic gradient descent with minibatch optimizer and with provided regularizer (if not ZERO). Instead of an initial guess of zeors, it starts with user provided initial model values and keeps updating the model to minimize the cost function until convergence is achieved or maximum iteration count is reached. After the training, it returns the trained output model.

Return Value
After the successful training, it returns a trained model of the type `logistic_regression_model<T>`.

logistic_regression_with_lbfgs::train()

Parameters
data: A `crs_matrix<T>` containing the sparse feature matrix
label: A `dvector<T>` containing the output labels
numIteration: A size_t parameter containing the maximum number of iteration count (Default: 1000)
alpha: A parameter of T type containing the learning rate (Default: 0.01)
hist_size: A parameter of size_t type containing the number of gradient history to be stored (Default: 10)
regParam: A parameter of T type containing the regularization parameter (also called lambda) (Default: 0.01)
regtyp: A parameter of the type frovedis::RegType, which can be either ZERO, L1 or L2 (Default: ZERO)
isIntercept: A boolean parameter to specify whether to include intercept term (bias term) or not (Default: false)
convergenceTol: A parameter of T type containing the threshold value to determine the convergence (Default: 0.001)
mType: frovedis::MatType parameter specifying the matrix type to be used for internal calculation (Default: HYBRID for SX architecture, CRS for other architectures)

Purpose
It trains a logistic regression model with LBFGS optimizer and with provided
regularizer (if not ZERO). It starts with an initial guess of zeros for the model vector and keeps updating the model to minimize the cost function until convergence is achieved or maximum iteration count is reached. After the training, it returns the trained output model.

Return Value
After the successful training, it returns a trained model of the type `logistic_regression_model<T>`.

logistic_regression_with_lbfgs::train()

Parameters
data: A `crs_matrix<T>` containing the sparse feature matrix
label: A `dvector<T>` containing the output labels
initModel: A `logistic_regression_model<T>` containing the user provided initial model values
numIteration: A size_t parameter containing the maximum number of iteration count (Default: 1000)
alpha: A parameter of T type containing the learning rate (Default: 0.01)
hist_size: A parameter of size_t type containing the number of gradient history to be stored (Default: 10)
regParam: A parameter of T type containing the regularization parameter (also called lambda) (Default: 0.01)
regtyp: A parameter of the type frovedis::RegType, which can be either ZERO, L1 or L2 (Default: ZERO)
isIntercept: A boolean parameter to specify whether to include intercept term (bias term) or not (Default: false)
convergenceTol: A parameter of T type containing the threshold value to determine the convergence (Default: 0.001)
mType: frovedis::MatType parameter specifying the matrix type to be used for internal calculation (Default: HYBRID for SX architecture, CRS for other architectures)

Purpose
It trains a logistic regression model with LBFGS optimizer and with provided regularizer (if not ZERO). Instead of an initial guess of zeors, it starts with user provided initial model values and keeps updating the model to minimize the cost function until convergence is achieved or maximum iteration count is reached. After the training, it returns the trained output model.

Return Value
After the successful training, it returns a trained model of the type `logistic_regression_model<T>`.

SEE ALSO

logistic_regression_model, linear_svm