Linear Regression

NAME

Linear Regression - A regression algorithm supported by Frovedis to predict the continuous output without any regularization.

SYNOPSIS

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

```
linear_regression_model<T>
linear_regression_with_sgd::train (crs_matrix<T>& data,
      dvector<T>& label,
      size_t numIteration = 1000,
      T alpha = 0.01,
      T miniBatchFraction = 1.0,
      bool isIntercept = false,
      T convergenceTol = 0.001,
      MatType mType = HYBRID)
```

```
linear_regression_model<T>
linear_regression_with_sgd::train (crs_matrix<T>& data,
      dvector<T>& label,
      linear_regression_model<T>& initModel,
      size_t numIteration = 1000,
      T alpha = 0.01,
      T miniBatchFraction = 1.0,
      bool isIntercept = false,
      T convergenceTol = 0.001,
      MatType mType = HYBRID)
```

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

```
linear_regression_model<T>
linear_regression_with_lbfgs::train (crs_matrix<T>& data,
      dvector<T>& label,
      size_t numIteration = 1000,
      T alpha = 0.01,
      size_t hist_size = 10,
      bool isIntercept = false,
      T convergenceTol = 0.001,
      MatType mType = HYBRID)
```

```
linear_regression_model<T>
linear_regression_with_lbfgs::train (crs_matrix<T>& data,
```

`dvector<T>`& label,
`linear_regression_model<T>`& initModel,
size_t numIteration = 1000,
T alpha = 0.01,
size_t hist_size = 10,
bool isIntercept = false,
T convergenceTol = 0.001,
MatType mType = HYBRID)

DESCRIPTION

Linear least squares is the most common formulation for regression problems. It is a linear method with the loss function given by the **squared loss**:

```
L(w;x,y) := 1/2(wTx-y)^2
```

Where the vectors x are the training data examples and y are their corresponding labels which we want to predict. w is the linear model (also known as weight) which uses a single weighted sum of features to make a prediction. The method is called linear since it can be expressed as a function of wTx and y. Linear regression does not use any regularizer.

The gradient of the squared loss is: (wTx-y).x

Frovedis provides implementation of linear 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

linear_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)
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 linear regression model with stochastic gradient descent with minibatch optimizer, but without any regularizer. 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 `linear_regression_model<T>`.

linear_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 `linear_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)
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 linear regression model with stochastic gradient descent with minibatch optimizer, but without any regularizer. 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 `linear_regression_model<T>`.

linear_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)
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 linear regression model with LBFGS optimizer, but without any regularizer. 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 `linear_regression_model<T>`.

linear_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 `linear_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)
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 linear regression model with LBFGS optimizer, but without any regularizer. 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 `linear_regression_model<T>`.

SEE ALSO

linear_regression_model, lasso_regression, ridge_regression