frovedis::logistic_regression_model<T>

NAME

`logistic_regression_model<T>` - A data structure used in modeling the outputs of the frovedis logistic regression algorithm.

SYNOPSIS

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

Constructors

logistic_regression_model ()
logistic_regression_model (size_t num_ftr, T intercpt=0.0, T threshold=0.5)
logistic_regression_model (const `logistic_regression_model<T>`& model)
logistic_regression_model (`logistic_regression_model<T>`&& model)

Overloaded Operators

`logistic_regression_model<T>`& operator= (const `logistic_regression_model<T>`& model)
`logistic_regression_model<T>`& operator= (`logistic_regression_model<T>`&& model)
`logistic_regression_model<T>` operator+ (const `logistic_regression_model<T>`& model) const
`logistic_regression_model<T>` operator- (const `logistic_regression_model<T>`& model) const
void operator+= (const `logistic_regression_model<T>`& model)
void operator-= (const `logistic_regression_model<T>`& model)

Public Member Functions

`std::vector<T>` predict (DATA_MATRIX& mat)
`std::vector<T>` predict_probability (DATA_MATRIX& mat)
void set_threshold (T threshold)
size_t get_num_features () const
void save (const std::string& path) const
void savebinary (const std::string& path) const
void load (const std::string& path) const
void loadbinary (const std::string& path) const
void debug_print() const
`node_local<logistic_regression_model<T>>` broadcast ()

DESCRIPTION

`logistic_regression_model<T>` models the output of the frovedis logistic regression algorithm, the trainer interface of which aims to optimize an initial model and outputs the same after optimization. This model has the below structure:

```
template <class T>
struct logistic_regression_model {
  std::vector<T> weight; // the weight vector associated with each input training features
  T intercept;           // the bias intercept term
  T threshold;           // the threshold value used in prediction
  SERIALIZE (weight, intercept, threshold)
};
```

This is a template based data structure, where "T" is supposed to be "float" (single-precision) or "double" (double-precision). Note this is a serialized data structure. The detailed description can be found in subsequent sections.

Constructor Documentation

logistic_regression_model ()

Default constructor. It creates an empty logistic regression model with default "intercept" value as 0.0 and default "threshold" value as 0.5.

logistic_regression_model (size_t num_ftr, T intercept=0.0, T threshold=0.5)

Parameterized constructor. It accepts the number-of-features input from the user and allocates the memory for the model of the same size. If no initial value of the "intercept" is provided, it considers the default value as 0.0. If no "threshold" value is provided, it considers the default value as 0.5.

logistic_regression_model (const `logistic_regression_model<T>`& model)

Copy constructor. It accepts an lvalue object of the same type and deep-copies the same in the newly constructed object.

logistic_regression_model (`logistic_regression_model<T>`&& model)

Move constructor. It accepts an rvalue object of the same type and instead of copying, it moves the contents in the newly constructed object.

Overloaded Operator Documentation

`logistic_regression_model<T>`& operator= (const `logistic_regression_model<T>`& model)

It deep-copies the contents of the input lvalue model into the left-hand side model of the assignment operator "=".

`logistic_regression_model<T>`& operator= (`logistic_regression_model<T>`&& model)

Instead of copying, it moves the contents of the input rvalue model into the left-hand side model of the assignment operator "=".

`logistic_regression_model<T>` operator+ (const `logistic_regression_model<T>`& model) const

This operator is used to add two logistic regression models and outputs the resultant model. If m1 and m2 are two models, expression like "m1 + m2" can easily be evaluated on them.

`logistic_regression_model<T>` operator- (const `logistic_regression_model<T>`& model) const

This operator is used to subtract two logistic regression models and outputs the resultant model. If m1 and m2 are two models, expression like "m1 - m2" can easily be evaluated on them.

void operator+= (const `logistic_regression_model<T>`& model)

This operator is used to add two logistic regression models. But instead of returning a new model, it updates the target model with the resultant model. If m1 and m2 are two models, then "m1 += m2" will add m2 with m1 and update m1 itself.

void operator-= (const `logistic_regression_model<T>`& model)

This operator is used to subtract two logistic regression models. But instead of returning a new model, it updates the target model with the resultant model. If m1 and m2 are two models, then "m1 -= m2" will subtract m2 from m1 and update m1 itself.

Pubic Member Function Documentation

`std::vector<T>` predict (DATA_MATRIX& mat)

This function is used on a trained model (after training is done) to predict the unknown output labels based on the given input matrix. It uses prediction logic according to logistic regression algorithm.

This function expects any input data matrix which provides an overloaded multiply "*" operator with a vector type object. E.g., if "v" is an object of `std::vector<T>` type, then "mat * v" should be supported and it should return the resultant vector of the type `std::vector<T>`. DATA_MATRIX can be `frovedis::crs_matrix_local<T>`, `frovedis::ell_matrix_local<T>` etc.

On succesful prediction, this function returns the predicted values in the form of `std::vector<T>`. Currently, it supports only binary prediction in the form of 1 (yes) and -1 (no). It will throw an exception, if any error occurs.

`std::vector<T>` predict_probability (DATA_MATRIX& mat)

This function is also used on trained model (after training is done) to predict the unknown output labels based on the given input matrix. But instead of returning yes/no predictions, it returns the raw probabilities in the form of `std::vector<T>` corresponsing to each new feature vector in the given matrix. Like predict(), it can also accept any data matrix, if support of "*" operator with a vector is provided for that matrix.

void set_threshold (T threshold)

It sets threshold value of the target model with the provided value. It will throw an exception, if negative value is provided.

size_t get_num_features () const

It returns the number-of-features in the target model.

void save (const std::string& path) const

It saves the target model in the specified path in simple text format. It will throw an exception, if any error occurs during the save operation.

void savebinary (const std::string& path) const

It saves the target model in the specified path in (little-endian) binary data format. It will throw an exception, if any error occurs during the save operation.

void load (const std::string& path) const

It loads the target logistic regression model from the data in specified text file. It will throw an exception, if any error occurs during the load operation.

void loadbinary (const std::string& path) const

It loads the target logistic regression model from the data in specified (little-endian) binary file. It will throw an exception, if any error occurs during the load operation.

void debug_print() const

It prints the contents of the model on the user terminal. It is mainly useful for debugging purpose.

`node_local<logistic_regression_model<T>>` **broadcast ()**

It broadcasts the target model to all the participating MPI processes (worker nodes) in the system. This is an efficient (as it does not involve the serialization overhead of the model weight vector) implementation than simple "frovedis:broadcast(model)" call.

Public Data Member Documentation

weight

An object of `std::vector<T>` type. It is used to store the weight/theta components associated with each training features.

intercept

A "T" type object (mainly "float" or "double"). It is used to store the bias intercept term of the model.

threshold

A “T” type object (mainly “float” or “double”). It is used to hold the threshold value used in prediction.

SEE ALSO

linear_regression_model, svm_model