DBM Documentation

Release 1.1

Simon Lee

Mar 02, 2017

CONTENTS:

This is the document of the Python APIs of Delta Boosting Machine. Classes and functions are listed and described.

- genindex

CHAPTER

ONE

CLASSES

class `dbm_py.interface.`**`Matrix`**(*height=None*, *width=None*, *val=None*, *file_name=None*, *sep=None*, *mat=None*)

`__init__`(*height=None*, *width=None*, *val=None*, *file_name=None*, *sep=None*, *mat=None*)

This is the class of Matrix used in DBM. To feed the training and prediction data to DBM, they should be converted to Matrix first of all. The Matrix interface provides four ways of initialization, i.e. initialization with random values in [-1, 1], initialization with a user-provided value, initialization from a file and initialization with a Float_Matrix object. One may also initializing a matrix with any values and then use the method from_np2darray to transfer the values from a numpy array of the same shape to it.

Initialization with random values in [-1, 1] :param height: height of the matrix :param width: width of the matrix

Initialization with a user-provided value :param val: a particular value for initialization

Initialization from a file :param file_name: file name of the file where data comes from :param sep: seperator used in the file

Initialization with a Float_Matrix object :param mat: a Float_Matrix object

Note: 1. When initializing from a file, the format should be correct. One may first of all save a matrix to a file and look at the file and see how it looks like. 2. Avoid directly using Float_Matrix. 3. Converting tools np2darray_to_float_matrix and float_matrix_to_np2darray are provided.

`assign`(*i*, *j*, *val*)

Assign a value to a particular element.

Parameters

- **`i`** – height of the element
- **`j`** – width of the element
- **`val`** – value to be assigned

`clear`()

Set all elements to 0.

`from_np2darray`(*source*)

Assign the data stored in a two-dimensional numpy array to this matrix.

Parameters **`source`** – a two-dimensional numpy array of the same

shape as this matrix

`get`(*i*, *j*)

Access to a particular element in the matrix.

Parameters

- **i** – height of the element
- **j** – width of the element

Returns the element

Note: i and j should be in the correct ranges

save (*file_name*, *sep='\t'*)
Save the data stored in it to a file.

Parameters

- **file_name** – a string
- **sep** – a character

shape ()
Return a list containing the shape of the matrix.

Returns [matrix height, matrix width]

show ()
Print to screen the data stored in the matrix.

to_np2darray ()
Assign the data stored in this matrix to a two-dimensional numpy array and return it.

Returns a two-dimensional numpy array of the same shape as

this matrix

class dbm_py.interface.**Data_set** (*data_x*, *data_y*, *portion_for_validating*)

__init__ (*data_x*, *data_y*, *portion_for_validating*)
This is the class of Data_set that provides an easy to tool for splitting all data into training and validating parts.

Parameters

- **data_x** – a Matrix object
- **data_y** – a Matrix object
- **portion_for_validating** – percentage of the whole data

used for validating

get_train_x ()
Return the part of predictors for training.

Returns a Matrix object

get_train_y ()
Return the part of responses for training.

Returns a Matrix object

get_validate_x ()
Return the part of predictors for validating.

Returns a Matrix object

get_validate_y ()
Return the part of responses for validating.

Returns a Matrix object

class `dbm_py.interface.`**`Params`**(*params=None*)

`__init__`(*params=None*)
This is class of Params storing parameters used in DBM.

Parameters **`params`** – a Params object

`print_all`()
Print all parameters and their values to the screen.

`set_params`(*string*, *sep=' '*)
Set values of parameters.

Usage: [sep] represents the character used as the separator

'parameter_name[sep]parameter_value' 'parameter_name[sep]parameter_value[sep]

parameter_name[sep]parameter_value'

Parameters

- **`string`** – a string storing the parameters to be set
- **`sep`** – separator used in the string

class `dbm_py.interface.`**`DBM`**(*params*)

`__init__`(*params*)
This is the class of DBM.

Parameters **`params`** – a Params object

`calibrate_plot`(*observation*, *prediction*, *resolution*, *file_name=''*)
This is exactly the same as the one in GBM in R.

Parameters

- **`observation`** – a Matrix object
- **`prediction`** – a Matrix object
- **`resolution`** – a scalar
- **`file_name`** – save the result if provided

Returns a Matrix object

`interact`(*data*, *predictor_ind*, *total_no_predictor*)
This is exactly the same as the one in GBM in R.

Parameters

- **`data`** – a Matrix object
- **`predictor_ind`** – a Matrix object
- **`total_no_predictor`** – a scalar

Returns a scalar

`load`(*file_name*)
Load from a file.

Parameters **`file_name`** – a string

pdp (*data_x*, *feature_index*)
Calculate the data used in partial dependence plots.

Parameters

- **data_x** – a Matrix object used for calculating
- **feature_index** – the index of the predictor of

interest (the No. of the column) :return: a Matrix object storing the data used in partial dependence plots

predict (*data_x*)
Predict if it has been trained or it has been loaded from a trained model.

Parameters **data_x** – a Matrix object

Returns

save (*file_name*)
Save the DBM after trained.

Parameters **file_name** – a string

save_performance (*file_name*)
Save the training and validating losses.

Parameters **file_name** – a string

ss (*data_x*)
Calculate statistical signifiance of every predictor.

Parameters **data_x** – a Matrix object used for calculating

Returns a Matrix object storing P-values for every predictor

train (*data_set*)
Train the DBM.

Parameters **data_set** – a Data_set object

class dbm_py.interface.**AUTO_DBM** (*params*)

__init__ (*params*)
This is the class of DBM.

Parameters **params** – a Params object

calibrate_plot (*observation*, *prediction*, *resolution*, *file_name*)
This is exactly the same as the one in GBM in R.

Parameters

- **observation** – a Matrix object
- **prediction** – a Matrix object
- **resolution** – a scalar
- **file_name** – save the result if provided

Returns a Matrix object

interact (*data*, *predictor_ind*, *total_no_predictor*)
This is exactly the same as the one in GBM in R.

Parameters

- **data** – a Matrix object
- **predictor_ind** – a Matrix object
- **total_no_predictor** – a scalar

Returns a scalar

load (*file_name*)

Load from a file.

Parameters **file_name** – a string

pdp (*data_x*, *feature_index*)

Calculate the data used in partial dependence plots.

Parameters

- **data_x** – a Matrix object used for calculating
- **feature_index** – the index of the predictor of interest

(the No. of the column) :return: a Matrix object storing the data used in partial dependence plots

predict (*data_x*)

Predict if it has been trained or it has been loaded from a trained model.

Parameters **data_x** – a Matrix object

Returns

save (*file_name*)

Save the DBM after trained.

Parameters **file_name** – a string

save_performance (*file_name*)

Save the training and validating losses.

Parameters **file_name** – a string

ss (*data_x*)

Calculate statistical signifiance of every predictor.

Parameters **data_x** – a Matrix object used for calculating

Returns a Matrix object storing P-values for every predictor

train (*data_set*)

Train the DBM.

Parameters **data_set** – a Data_set object

CHAPTER

TWO

FUNCTIONS

dbm_py.interface.**np2darray_to_float_matrix**(*source*)

Convert a two-dimensional numpy array to a Matrix.

Parameters **source** – a two-dimensional numpy array

Returns a Matrix object of the same shape as the numpy array

dbm_py.interface.**float_matrix_to_np2darray**(*source*)

Convert a Matrix to a two-dimensional numpy array.

Parameters **source** – a Matrix object

Returns a two-dimensional numpy array of the same shape as the

Matrix

dbm_py.interface.**string_to_params**(*string*, *sep*=' ')

Directly transfer a string to a Params object.

Parameters

- **string** – a string
- **sep** – a character

Returns a Params object

INDEX