DBM Documentation

Release 1.1

Simon Lee

Mar 07, 2017

CONTENTS:

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

CHAPTER

ONE

CLASSES

Matrix

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

Data Set

class `dbm_py.interface.`**`Data_set`**(*data_x*, *data_y*, *portion_for_validating*, *random_seed=-1*)

`__init__`(*data_x*, *data_y*, *portion_for_validating*, *random_seed=-1*)
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
- **`random_seed`** – optional random seed (random if negative or fixed if non-negative)

`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

Parameters

Parameter Name	Type	Meaning
dbm_no_bunches_of_learners	int	number of boostraped BLs
dbm_no_candidate_feature	int	number of features for each BL (< total number of features)
dbm_portion_train_sample	double	percentage for training each BL
dbm_no_cores	int	number of BL in each bunch (number of cores used)
dbm_loss_function	char	(n)ormal, (b)ernoulli, (p)oisson or (t)weedie
dbm_display_training_progress	bool	whether to display training progress or not
dbm_record_every_tree	bool	whether to record trees in a file or not
dbm_freq_showing_loss_on_test	int	show loss on test after how many bunches of BLs
dbm_shrinkage	double	shrinkage for each BL
dbm_nonoverlapping_training	int	whether to BLs in a bunch use nonoverlapping samples or not
dbm_remove_rows_containing_nans	int	whether to remove rows containing NaNs in training every BL
dbm_min_no_samples_per_bl	int	minimal number of samples for trainin every BL
dbm_portion_for_trees	double	percentage of BLs using trees
dbm_random_seed	int	random seed (random < 0 and fixed >= 0)
dbm_portion_for_lr	double	percentage of BLs using linear regression
dbm_portion_for_s	double	percentage of BLs using splines
dbm_portion_for_k	double	percentage of BLs using k-means
dbm_portion_for_nn	double	should be 0
dbm_portion_for_d	double	percentage of BLs using dominating principal component stairs
dbm_accumulated_portion	double	unused
_shrinkage_for_selected_b		
dbm_portion_shrinkage_for_unselected_bl	double	unused
tweedie_p	double	p of tweedie should in (1, 2)
splines_no_knot	int	number of knots of splines
splines_portion_of_pairs	double	percentage of pairs of perdictors considered
splines_regularization	double	ridge regression penalty
splines_hinge_coefficient	double	coefficient in splines
kmeans_no_centroids	int	number of centroids
kmeans_max_iteration	int	max number of iterations of training
kmeans_tolerance	double	max tolerated error
kmeans_fraction_of_pairs	double	percentage of pairs of predictors considered
nn_no_hidden_neurons	int	number of hidden neurons

Continued on next page

Table 1.1 – continued from previous page

nn_step_size	double	stochastic gradient descent step size
nn_validate_portion	double	percentage of samples used for validating
nn_batch_size	int	number of samples in a batch
nn_max_iteration	int	maximal number of iterations of training
nn_no_rise_of_loss_on_validate	int	maximal number rises of loss on validation set
cart_min_samples_in_a_node	int	minimal numbers in a node of a tree
cart_max_depth	int	maximal numbers of levels of a tree
cart_prune	int	whether to prune after training
lr_regularization	double	ridge regression penalty
dpcs_no_ticks	int	number of stairs in the direction of dominating principal component
dpcs_range_shrinkage_of_ticks	double	shrinkage of the range in the direction of dominating principal component
dbm_do_perf	bool	whether to record performance on both training sets
pdp_no_x_ticks	int	number of ticks in x-axis
pdp_no_resamplings	int	number of resamplings for bootstrapping
pdp_resampling_portion	double	percentage of samples in each bootstrap
pdp_ci_bandwidth	double	width of the confidence interval
pdp_save_files	int	whether to save the result

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_va

Parameters

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

Delta Boosting Machines

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)

Returns 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

Delta Boosting Machines with Automatic BL Selection

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)

Returns 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