LightTwinSVM Documentation

Release 0.5.0

Mir, A.

Mar 09, 2019

Contents:

1 Modules of LightTwinSVM's package **1**
- 1.1 twinsvm . . . 1
- 1.2 eval_classifier . . . 3
- 1.3 dataproc . . . 4
- 1.4 misc . . . 5

2 Indices and tables **7**

Python Module Index **9**

CHAPTER 1

Modules of LightTwinSVM's package

This page contains the list of the project's modules

twinsvm	Classes and functios are defined for training and testing TwinSVM classifier.
eval_classifier	In this module, methods are defined for evluating TwinSVM perfomance such as cross validation train/test split, grid search and generating the detailed result.
dataproc	In this module, functions for reading and processing datasets are defined.
misc	In this module, several miscellaneous functions are defined for using in other modules.

1.1 twinsvm

Classes and functios are defined for training and testing TwinSVM classifier.

TwinSVM classifier generates two non-parallel hyperplanes. For more info, refer to the original papar. Khemchandani, R., & Chandra, S. (2007). Twin support vector machines for pattern classification. IEEE Transactions on pattern analysis and machine intelligence, 29(5), 905-910.

Motivated by the following paper, the multi-class TSVM is developed. Tomar, D., & Agarwal, S. (2015). A comparison on multi-class classification methods based on least squares twin support vector machine. Knowledge-Based Systems, 81, 131-147.

Functions

rbf_kernel(x, y, u)	It transforms samples into higher dimension Input: x,y: Samples u: Gamma parameter Output: Samples with higher dimension

Classes

HyperPlane()	
MCTSVM([kernel, C, gamma])	Multi Class Twin Support Vector Machine One-vs-All Scheme
OVO_TSVM([kernel, C1, C2, gamma])	Multi Class Twin Support Vector Machine One-vs-One Scheme This classifier is scikit-learn compatible, which means scikit-learn features such as cross_val_score and GridSearchCV can be used for OVO_TSVM
TSVM([kernel, rect_kernel, C1, C2, gamma])	

class twinsvm.**TSVM**(*kernel='linear'*, *rect_kernel=1*, *C1=1*, *C2=1*, *gamma=1*)

Bases: sklearn.base.BaseEstimator

get_params_names()

It returns the names of hyper-parameters of this classifier.

fit(*X_train*, *y_train*)

It trains TwinSVM classfier on given data Input:

X_train: Training samples y_train: Samples' category

output:

w1, w2: Coordinates of two non-parallel hyperplanes b1, b2: Biases

predict(*X_test*)

Predictes class of test samples Input:

X_test: Test samples

twinsvm.**rbf_kernel**(*x*, *y*, *u*)

It transforms samples into higher dimension Input:

x,y: Samples u: Gamma parameter

Output: Samples with higher dimension

class twinsvm.**MCTSVM**(*kernel='linear'*, *C=1*, *gamma=1*)

Bases: sklearn.base.BaseEstimator

Multi Class Twin Support Vector Machine One-vs-All Scheme

get_params_names()

It returns the names of hyper-parameters of this classifier.

fit(*X_train*, *y_train*)

Input: X_train: Training samples y_train: Lables of training samples

predict(*X_test*)

Predictes class of test samples

Input: X_test: Test samples

class twinsvm.**OVO_TSVM**(*kernel='linear'*, *C1=1*, *C2=1*, *gamma=1*)

Bases: sklearn.base.BaseEstimator, sklearn.base.ClassifierMixin

Multi Class Twin Support Vector Machine One-vs-One Scheme This classifier is scikit-learn compatible, which means scikit-learn features such as cross_val_score and GridSearchCV can be used for OVO_TSVM

get_params_names()

It returns the names of hyper-parameters of this classifier.

fit(*X*, *y*)

Given training set, it creates a SVM model

Parameters: X_train: Training samples, (n_samples, n_features) y_train: Target values, (n_samples,)

predict(*X*)

Predicits lables of test samples

Parameters: X_test: test samples, (n_samples, n_features)

Returns: y_pred: array, (n_samples,)

1.2 eval_classifier

In this module, methods are defined for evluating TwinSVM perfomance such as cross validation train/test split, grid search and generating the detailed result.

Functions

eval_metrics(y_true, y_pred)	Input:
grid_search(search_space, func_validator)	It applies grid search which finds C and gamma paramters for obtaining best classification accuracy.
initializer(user_input_obj)	It gets user input and passes function and classes arguments to run the program Input: user_input_obj: User input (UserInput class)
save_result(file_name, validator_obj, ...)	It saves detailed result in spreadsheet file(Excel).
search_space(kernel_type, class_type, ... [, ...])	It generates combination of search elements for grid search Input: kernel_type: kernel function which is either linear or RBF c_l_bound, c_u_bound: Range of C penalty parameter for grid search(e.g 2^-5 to 2^+5) rbf_lbound, rbf_ubound: Range of gamma parameter Output: return search elements for grid search (List)

Classes

Validator(X_train, y_train, validator_type, ...)	It applies a test method such as cross validation on a classifier like TSVM

eval_classifier.**eval_metrics**(*y_true*, *y_pred*)

Input:

y_true: True label of samples y_pred: Prediction of classifier for test samples

output: Elements of confusion matrix and Evalaution metrics such as accuracy, precision, recall and F1 score

class eval_classifier.**Validator**(*X_train*, *y_train*, *validator_type*, *obj_tsvm*)
Bases: object

It applies a test method such as cross validation on a classifier like TSVM

cv_validator(*dict_param*)
It applies cross validation on instance of Binary TSVM classifier Input:

dict_param: A dictionary of hyper-parameters (dict)

output: Evaluation metrics such as accuracy, precision, recall and F1 score for each class.

split_tt_validator(*dict_param*)
It trains TwinSVM classifier on random training set and tests the classifier on test set. output:

Evaluation metrics such as accuracy, precision, recall and F1 score for each class.

cv_validator_mc(*dict_param*)
It applies cross validation on instance of multiclass TSVM classifier

choose_validator()
It returns choosen validator method.

eval_classifier.**search_space**(*kernel_type*, *class_type*, *c_l_bound*, *c_u_bound*, *rbf_lbound*, *rbf_ubound*, *step=1*)
It generates combination of search elements for grid search Input:

kernel_type: kernel function which is either linear or RBF c_l_bound, c_u_bound: Range of C penalty parameter for grid search(e.g 2^-5 to 2^+5) rbf_lbound, rbf_ubound: Range of gamma parameter

Output: return search elements for grid search (List)

eval_classifier.**grid_search**(*search_space*, *func_validator*)
It applies grid search which finds C and gamma paramters for obtaining best classification accuracy.

Input: search_space: search_elements (List) func_validator: Validator function

output: returns classification result (List)

eval_classifier.**save_result**(*file_name*, *validator_obj*, *gs_result*, *output_path*)
It saves detailed result in spreadsheet file(Excel).

Input: file_name: Name of spreadsheet file col_names: Column names for spreadsheet file gs_result = result produced by grid search output_path: Path to store the spreadsheet file.

output: returns path of spreadsheet file

eval_classifier.**initializer**(*user_input_obj*)
It gets user input and passes function and classes arguments to run the program Input:

user_input_obj: User input (UserInput class)

1.3 dataproc

In this module, functions for reading and processing datasets are defined.

Functions

conv_str_fl(data)	It converts string data to float for computation.
read_data(filename[, header])	It converts CSV file to NumPy arrays for further operations like training
read_libsvm(filename)	It reads LIBSVM data files for doing classification with TwinSVM Input: file_name: Path to the dataset file

dataproc.**conv_str_fl**(*data*)
It converts string data to float for computation.

dataproc.**read_data**(*filename*, *header=True*)
It converts CSV file to NumPy arrays for further operations like training

Input: file_name: Path to the dataset file ignore_header: Ignoring first row of dataset because of header names

output: data_samples: Training samples in NumPy array data_labels: labels of samples in NumPy array file_name: Name of dataset

dataproc.**read_libsvm**(*filename*)
It reads LIBSVM data files for doing classification with TwinSVM Input:

file_name: Path to the dataset file

output: data_samples: Training samples in NumPy array data_labels: labels of samples in NumPy array file_name: Name of dataset

1.4 misc

In this module, several miscellaneous functions are defined for using in other modules. Such as date time formatting and customized progress bar

Functions

progress_bar_gs(iteration, total, e_time, ...)	A customized progress bar for grid search Input: iteration: currrent iteration total: total iteration e_time: Elapsed time accuracy: Current accuracy and its std (Tuple) best_acc: Best accuracy and its std (Tuple) prefix: prefix string suffix: suffix string decimals: number of decimals in percent length: character length of bar fill: bar fill character
time_fmt(t_delta)	It convets datetime objects to formatted string

misc.**time_fmt**(*t_delta*)
It convets datetime objects to formatted string

misc.**progress_bar_gs**(*iteration*, *total*, *e_time*, *accuracy*, *best_acc*, *prefix=''*, *suffix=''*, *decimals=1*, *length=25*, *fill='#'*)
A customized progress bar for grid search Input:

iteration: currrent iteration total: total iteration e_time: Elapsed time accuracy: Current accuracy and its std (Tuple) best_acc: Best accuracy and its std (Tuple) prefix: prefix string suffix: suffix string decimals: number of decimals in percent length: character length of bar fill: bar fill character

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

d

dataproc, 4

e

eval_classifier, 3

m

misc, 5

t

twinsvm, 1

Index

C

choose_validator() (eval_classifier.Validator method), 4
conv_str_fl() (in module dataproc), 5
cv_validator() (eval_classifier.Validator method), 4
cv_validator_mc() (eval_classifier.Validator method), 4

D

dataproc (module), 4

E

eval_classifier (module), 3
eval_metrics() (in module eval_classifier), 3

F

fit() (twinsvm.MCTSVM method), 2
fit() (twinsvm.OVO_TSVM method), 3
fit() (twinsvm.TSVM method), 2

G

get_params_names() (twinsvm.MCTSVM method), 2
get_params_names() (twinsvm.OVO_TSVM method), 3
get_params_names() (twinsvm.TSVM method), 2
grid_search() (in module eval_classifier), 4

I

initializer() (in module eval_classifier), 4

M

MCTSVM (class in twinsvm), 2
misc (module), 5

O

OVO_TSVM (class in twinsvm), 2

P

predict() (twinsvm.MCTSVM method), 2
predict() (twinsvm.OVO_TSVM method), 3
predict() (twinsvm.TSVM method), 2
progress_bar_gs() (in module misc), 5

R

rbf_kernel() (in module twinsvm), 2
read_data() (in module dataproc), 5
read_libsvm() (in module dataproc), 5

S

save_result() (in module eval_classifier), 4
search_space() (in module eval_classifier), 4
split_tt_validator() (eval_classifier.Validator method), 4

T

time_fmt() (in module misc), 5
TSVM (class in twinsvm), 2
twinsvm (module), 1

V

Validator (class in eval_classifier), 3