
LightTwinSVM Documentation

Release 0.6.0

Mir, A.

Dec 29, 2019

Contents:

1	API documentation	3
1.1	twinsvm	3
1.2	eval_classifier	8
1.3	dataproc	12
1.4	misc	13
2	Classification results	15
3	Indices and tables	17
	Python Module Index	19
	Index	21

You can read API documentation, if you want to use the LightTwinSVM's estimators and tools directly in your project. Furthermore, if you have a question or problem regarding this program, please see the [support section](#) of the project in its GitHub repo.

This page contains the list of the project's modules

<i>twinsvm</i>	Classes and functions are defined for training and testing TwinSVM classifier.
<i>eval_classifier</i>	In this module, classes and methods are defined for evaluating the performance of the TwinSVM model.
<i>dataproc</i>	In this module, functions for reading and pre-processing datasets are defined.
<i>misc</i>	In this module, several miscellaneous functions are defined for using in other module, such as date time formatting and customized progress bar.

1.1 twinsvm

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

TwinSVM classifier generates two non-parallel hyperplanes. For more info, refer to the original paper. 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

<i>rbf_kernel</i> (x, y, u)	It transforms samples into higher dimension using Gaussian (RBF) kernel.
-----------------------------	--

Classes

<i>HyperPlane()</i>	Its object represents a hyperplane
<i>MCTSVM</i> ([kernel, C, gamma])	Multi-class Twin Support Vector Machine (One-vs-All Scheme)
<i>OVO_TSM</i> ([kernel, C1, C2, gamma])	Multi Class Twin Support Vector Machine (One-vs-One Scheme)
<i>TSM</i> ([kernel, rect_kernel, C1, C2, gamma])	Twin Support Vector Machine for binary classification.

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

Bases: sklearn.base.BaseEstimator

Twin Support Vector Machine for binary classification.

Parameters **kernel** : str, optional (default='linear')

Type of the kernel function which is either 'linear' or 'RBF'.

rect_kernel : float, optional (default=1.0)

Percentage of training samples for Rectangular kernel.

C1 : float, optional (default=1.0)

Penalty parameter of first optimization problem.

C2 : float, optional (default=1.0)

Penalty parameter of second optimization problem.

gamma : float, optional (default=1.0)

Parameter of the RBF kernel function.

Attributes

mat_C_t	(array-like, shape = [n_samples, n_samples]) A matrix that contains kernel values.
cls_name	(str) Name of the classifier.
w1	(array-like, shape=[n_features]) Weight vector of class +1's hyperplane.
b1	(float) Bias of class +1's hyperplane.
w2	(array-like, shape=[n_features]) Weight vector of class -1's hyperplane.
b2	(float) Bias of class -1's hyperplane.

Methods

<i>fit</i> (X_train, y_train)	It fits the binary TwinSVM model according to the given training data.
<i>get_params</i> ([deep])	Get parameters for this estimator.
<i>get_params_names</i> ()	For retrieving the names of hyper-parameters of this classifier.
<i>predict</i> (X_test)	Performs classification on samples in X using the TwinSVM model.
<i>set_params</i> (**params)	Set the parameters of this estimator.

get_params_names ()

For retrieving the names of hyper-parameters of this classifier.

Returns parameters : list of str, {[‘C1’, ‘C2’], [‘C1’, ‘C2’, ‘gamma’]}

Returns the names of the hyperparameters which are same as the class’ attributes.

fit (*X_train*, *y_train*)

It fits the binary TwinSVM model according to the given training data.

Parameters X_train : array-like, shape (n_samples, n_features)

Training feature vectors, where n_samples is the number of samples and n_features is the number of features.

y_train : array-like, shape(n_samples,)

Target values or class labels.

predict (*X_test*)

Performs classification on samples in X using the TwinSVM model.

Parameters X_test : array-like, shape (n_samples, n_features)

Feature vectors of test data.

Returns output : array, shape (n_samples,)

Predicted class lables of test data.

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

It transforms samples into higher dimension using Gaussian (RBF) kernel.

Parameters x, y : array-like, shape (n_features,)

A feature vector or sample.

u : float

Parameter of the RBF kernel function.

Returns float

Value of kernel matrix for feature vector x and y.

class `twinsvm.HyperPlane`

Bases: `object`

Its object represents a hyperplane

Attributes

w	(array-like, shape (n_features,)) Weight vector. If the RBF kernel is used, the shape will be (n_samples,)
b	(float) Bias.

class `twinsvm.MCTSVM` (*kernel*=‘linear’, *C*=1, *gamma*=1)

Bases: `sklearn.base.BaseEstimator`

Multi-class Twin Support Vector Machine (One-vs-All Scheme)

Parameters kernel : str, optional (default=‘linear’)

Type of the kernel function which is either ‘linear’ or ‘RBF’.

C : float, optional (default=1.0)

Penalty parameter.

gamma : float, optional (default=1.0)

Parameter of the RBF kernel function.

Attributes

classifiers	(dict) Stores an instance of <i>HyperPlane</i> class for each binary classifier.
mat_D_t	(list of array-like objects) Stores kernel matrix for each binary classifier.
cls_name	(str) Name of the classifier.

Methods

<i>fit</i> (X_train, y_train)	It fits the OVA-TwinSVM model according to the given training data.
<i>get_params</i> ([deep])	Get parameters for this estimator.
<i>get_params_names</i> ()	For retrieving the names of hyper-parameters of this classifier.
<i>predict</i> (X_test)	Performs classification on samples in X using the OVA-TwinSVM model.
<i>set_params</i> (**params)	Set the parameters of this estimator.

get_params_names ()

For retrieving the names of hyper-parameters of this classifier.

Returns parameters : list of str, [['C'], ['C', 'gamma']]

Returns the names of the hyperparameters which are same as the class' attributes.

fit (X_train, y_train)

It fits the OVA-TwinSVM model according to the given training data.

Parameters X_train : array-like, shape (n_samples, n_features)

Training feature vectors, where n_samples is the number of samples and n_features is the number of features.

y_train : array-like, shape(n_samples,)

Target values or class labels.

predict (X_test)

Performs classification on samples in X using the OVA-TwinSVM model.

Parameters X_test : array-like, shape (n_samples, n_features)

Feature vectors of test data.

Returns output : array, shape (n_samples,)

Predicted class labels of test data.

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)

The *OVO_TSMV* classifier is scikit-learn compatible, which means scikit-learn tools such as `cross_val_score` and `GridSearchCV` can be used for an instance of *OVO_TSMV*

Parameters **kernel** : str, optional (default='linear')

Type of the kernel function which is either 'linear' or 'RBF'.

C1 : float, optional (default=1.0)

Penalty parameter of first optimization problem for each binary *TSMV* classifier.

C2 : float, optional (default=1.0)

Penalty parameter of second optimization problem for each binary *TSMV* classifier.

gamma : float, optional (default=1.0)

Parameter of the RBF kernel function.

Attributes

cls_name	(str) Name of the classifier.
bin_TSMV_models_	(list) Stores instances of each binary <i>TSMV</i> classifier.

Methods

<i>fit</i> (X, y)	It fits the OVO-TwinSVM model according to the given training data.
<i>get_params</i> ([deep])	Get parameters for this estimator.
<i>get_params_names</i> ()	For retrieving the names of hyper-parameters of this classifier.
<i>predict</i> (X)	Performs classification on samples in X using the OVO-TwinSVM model.
<i>score</i> (X, y[, sample_weight])	Return the mean accuracy on the given test data and labels.
<i>set_params</i> (**params)	Set the parameters of this estimator.

get_params_names ()

For retrieving the names of hyper-parameters of this classifier.

Returns **parameters** : list of str, [['C1', 'C2'], ['C1', 'C2', 'gamma']]

Returns the names of the hyperparameters which are same as the class' attributes.

fit (X, y)

It fits the OVO-TwinSVM model according to the given training data.

Parameters **X** : array-like, shape (n_samples, n_features)

Training feature vectors, where n_samples is the number of samples and n_features is the number of features.

y : array-like, shape(n_samples,)

Target values or class labels.

Returns **self** : object

predict (*X*)

Performs classification on samples in *X* using the OVO-TwinSVM model.

Parameters *X* : array-like, shape (n_samples, n_features)

Feature vectors of test data.

Returns *y_pred* : array, shape (n_samples,)

Predicted class labels of test data.

1.2 eval_classifier

In this module, classes and methods are defined for evaluating the performance of the TwinSVM model. Also, a method for saving detailed classification result.

Functions

<code>eval_metrics(y_true, y_pred)</code>	It computes common evaluation metrics such as Accuracy, Recall, Precision, F1-measure, and elements of the confusion matrix.
<code>grid_search(search_space, func_validator)</code>	It does grid search to find the optimal values of hyperparameters for the TwinSVM model, which results in the best classification accuracy.
<code>initializer(user_input_obj)</code>	It passes a user's input to the functions and classes for solving a classification task.
<code>save_result(file_name, validator_obj, ...)</code>	It saves the detailed classification results in a spreadsheet file (Excel).
<code>search_space(kernel_type, class_type, ..., ...)</code>	It generates all combination of search elements based on the given range of hyperparameters.

Classes

<code>Validator(X_train, y_train, validator_type, ...)</code>	It evaluates the TwinSVM model based on the specified evaluation method.
---	--

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

It computes common evaluation metrics such as Accuracy, Recall, Precision, F1-measure, and elements of the confusion matrix.

Parameters *y_true* : array-like

Target values of samples.

y_pred : array-like

Predicted class labels.

Returns *tp* : int

True positive.

tn : int

True negative.

fp : int

False positive.

fn : int

False negative.

accuracy : float

Overall accuracy of the model.

recall_p : float

Recall of positive class.

precision_p : float

Precision of positive class.

f1_p : float

F1-measure of positive class.

recall_n : float

Recall of negative class.

precision_n : float

Precision of negative class.

f1_n : float

F1-measure of negative class.

class `eval_classifier.Validator` (*X_train*, *y_train*, *validator_type*, *obj_tsvm*)

Bases: `object`

It evaluates the TwinSVM model based on the specified evaluation method.

Parameters **X_train** : array-like, shape (n_samples, n_features)

Training feature vectors, where n_samples is the number of samples and n_features is the number of features.

y_train : array-like, shape (n_samples,)

Target values or class labels.

validator_type : tuple

A two-element tuple which contains type of evaluation method and its parameter. Example: ('CV', 5) -> 5-fold cross-validation, ('t_tsplit', 30) -> 30% of samples for test set.

obj_tsvm : object

A TwinSVM model. It can be an instance of *TSVM*, *MCTSVM*, or *OVO_TSVM*.

Methods

choose_validator()

It selects the appropriate evaluation method based on the input paramters.

Continued on next page

Table 9 – continued from previous page

<code>cv_validator(dict_param)</code>	It evaluates the TwinSVM model using the cross-validation method.
<code>cv_validator_mc(dict_param)</code>	It evaluates the multi-class TwinSVM model using the cross-validation.
<code>split_tt_validator(dict_param)</code>	It evaluates the TwinSVM model using the train/test split method.

cv_validator (*dict_param*)

It evaluates the TwinSVM model using the cross-validation method.

Parameters **dict_param** : dict

Values of hyper-parameters for the TwinSVM model.

Returns float

Mean accuracy of the model.

float

Standard deviation of accuracy.

dict

Evaluation metrics such as Recall, Percision and F1-measure for both classes as well as elements of the confusion matrix.

split_tt_validator (*dict_param*)

It evaluates the TwinSVM model using the train/test split method.

Parameters **dict_param** : dict

Values of hyper-parameters for the TwinSVM model.

Returns float

Accuracy of the model.

float

Zero standard deviation.

dict

Evaluation metrics such as Recall, Percision and F1-measure for both classes as well as elements of the confusion matrix.

cv_validator_mc (*dict_param*)

It evaluates the multi-class TwinSVM model using the cross-validation.

Parameters **dict_param** : dict

Values of hyper-parameters for the multicls TwinSVM model.

Returns float

Accuracy of the model.

float

Zero standard deviation.

dict

Evaluation metrics such as Recall, Percision and F1-measure.

choose_validator()

It selects the appropriate evaluation method based on the input parameters.

Returns object

An evaluation method for assessing the model's performance.

`eval_classifier.search_space(kernel_type, class_type, c_l_bound, c_u_bound, rbf_lbound, rbf_ubound, step=1)`

It generates all combination of search elements based on the given range of hyperparameters.

Parameters `kernel_type` : str, {'linear', 'RBF'}

Type of the kernel function which is either 'linear' or 'RBF'.

`class_type` : str, {'binary', 'ovo', 'ova'}

Type of classification.

`c_l_bound` : int

Lower bound for C penalty parameter.

`c_u_bound` : int

Upper bound for C penalty parameter.

`rbf_lbound` : int

Lower bound for gamma parameter which is the hyperparameter of the RBF kernel function.

`rbf_ubound` : int

Upper bound for gamma parameter.

`step` : int, optional (default=1)

Step size to increase power of 2.

Returns list

Search elements.

Examples

```
>>> from ltsvm import eval_classifier
>>> eval_classifier.search_space('RBF', 'binary', -1, 1, -1, 1)
[{'C1': 0.5, 'C2': 0.5, 'gamma': 0.5}...
{'C1': 1.0, 'C2': 1.0, 'gamma': 0.5}... {'C1': 2.0, 'C2': 2.0, 'gamma': 2.0}]
```

`eval_classifier.grid_search(search_space, func_validator)`

It does grid search to find the optimal values of hyperparameters for the TwinSVM model, which results in the best classification accuracy.

Parameters `search_space` : list

All combination of search elements.

`func_validator` : object

An evaluation method for assessing the TwinSVM model's performance.

Returns list

Classification results of the TwinSVM classifier using different set of hyperparameters.

`eval_classifier.save_result` (*file_name*, *validator_obj*, *gs_result*, *output_path*)

It saves the detailed classification results in a spreadsheet file (Excel).

Parameters `file_name` : str

Name of the spreadsheet file.

validator_obj : object

The evaluation method that was used for the assesment of the TwinSVM classifier.

gs_result : list

Classification results of the TwinSVM classifier using different set of hyperparameters.

output_path : str

Path at which the spreadsheet file will be saved.

Returns str

Path to the saved spreadsheet (Excel) file.

`eval_classifier.initializer` (*user_input_obj*)

It passes a user's input to the functions and classes for solving a classification task. The steps that this function performs can be summarized as follows:

1. Specifies a TwinSVM classifier based on the user's input.
2. Chooses an evaluation method for assessment of the classifier.
3. Computes all the combination of search elements.
4. Computes the evaluation metrics for all the search element using grid search.
5. Saves the detailed classification results in a spreadsheet file (Excel).

Parameters `user_input_obj` : object

An instance of `UserInput` class which holds the user input.

1.3 dataproc

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

Functions

<code>conv_str_fl</code> (<i>data</i>)	It converts string data to float for computation.
<code>read_data</code> (<i>filename</i> [, <i>header</i>])	It converts a CSV dataset to NumPy arrays for further operations like training the TwinSVM classifier.
<code>read_libsvm</code> (<i>filename</i>)	It reads LIBSVM data files for doing classification using the TwinSVM model.

`dataproc.conv_str_fl` (*data*)

It converts string data to float for computation.

Parameters `data` : array-like, shape (n_samples, n_features)

Training samples, where n_samples is the number of samples and n_features is the number of features.

Returns array-like

A numerical dataset which is suitable for further computation.

`dataproc.read_data(filename, header=True)`

It converts a CSV dataset to NumPy arrays for further operations like training the TwinSVM classifier.

Parameters `filename` : str

Path to the dataset file.

header : boolean, optional (default=True)

Ignores first row of dataset which contains header names.

Returns `data_train` : array-like, shape (n_samples, n_features)

Training samples in NumPy array.

data_labels : array-like, shape(n_samples,)

Class labels of training samples.

file_name : str

Dataset's filename.

`dataproc.read_libsvm(filename)`

It reads [LIBSVM](#) data files for doing classification using the TwinSVM model.

Parameters `filename` : str

Path to the LIBSVM data file.

Returns array-like

Training samples.

array-like

Class labels of training samples.

str

Dataset's filename

1.4 misc

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

Functions

<code>progress_bar_gs(iteration, total, e_time, ...)</code>	It shows a customizable progress bar for grid search.
<code>time_fmt(t_delta)</code>	It converts datetime objects to formatted string.

`misc.time_fmt(t_delta)`

It converts datetime objects to formatted string.

Parameters `t_delta` : object

The difference between two dates or time.

Returns str

A readable formatted-datetime string.

`misc.progress_bar_gs(iteration, total, e_time, accuracy, best_acc, prefix="", suffix="", decimals=1, length=25, fill='#')`

It shows a customizable progress bar for grid search.

Parameters **iteration** : int

Current iteration.

total : int

Maximum number of iterations.

e_time : str

Elapsed time.

accuracy : tuple

The accuracy and its std at current iteration (acc, std).

best_acc : tuple

The best accuracy and its std that were obtained at current iteration (best_acc, std).

prefix : str, optional (default='')

Prefix string.

suffix : str, optional (default='')

Suffix string.

decimals : int, optional (default=1)

Number of decimal places for percentage of completion.

length : int, optional (default=25)

Character length of the progress bar.

fill : str, optional (default='#')

Bar fill character.

Classification results

The LightTwinSVM program saves classification results in an Excel file. Here, the description of each column of the Excel file for both binary and multi-class classification problems is given in below tables. It helps you analyze the classification results.

Table 1: The description of the Excel file for a binary classification problem.

Column name	Description
accuracy	The overall classification accuracy
acc_std	The standard deviation of the overall classification accuracy
recall_p	The recall for the positive class
r_p_std	The standard deviation of the recall for the positive class
precision_p	The precision for the positive class
p_p_std	The standard deviation of the precision for the positive class
f1_p	The F1-measure for the positive class
f1_p_std	The standard deviation of the F1-measure for the positive class
recall_n	The recall for the negative class
r_n_std	The standard deviation of the recall for the negative class
precision_n	The precision for the negative class
p_n_std	The standard deviation of the precision for the negative class
f1_n	The F1-measure for the negative class
f1_n_std	The standard deviation of the F1-measure for the negative class
tp	True positive
tn	True negative
fp	False positive
fn	False negative
C1	The value of the first penalty parameter for TwinSVM
C2	The value of the second penalty parameter for TwinSVM
gamma	The value of RBF kernel's parameter

Table 2: The description of the Excel file for a multi-class classification problem.

Column name	Description
accuracy	The overall classification accuracy
acc_std	The standard deviation of the overall classification accuracy
micro_recall	The micro-averaged recall for all classes
m_rec_std	The standard deviation of the micro-averaged recall for all classes
micro_precision	The micro-averaged precision for all classes
m_prec_std	The standard deviation of the micro-averaged precision for all classes
micro_f1	The micro-averaged F1-measure for all classes
m_f1_std	The standard deviation of the micro-averaged F1-measure for all classes
C1	The value of the first penalty parameter for TwinSVM
C2	The value of the second penalty parameter for TwinSVM
gamma	The value of RBF kernel's parameter

CHAPTER 3

Indices and tables

- `genindex`
- `modindex`
- `search`

d

`dataproc`, [12](#)

e

`eval_classifier`, [8](#)

m

`misc`, [13](#)

t

`twinsvm`, [3](#)

C

choose_validator() (*eval_classifier.Validator method*), 10
conv_str_fl() (*in module dataproc*), 12
cv_validator() (*eval_classifier.Validator method*), 10
cv_validator_mc() (*eval_classifier.Validator method*), 10

D

dataproc (*module*), 12

E

eval_classifier (*module*), 8
eval_metrics() (*in module eval_classifier*), 8

F

fit() (*twinsvm.MCTSVM method*), 6
fit() (*twinsvm.OVO_TSVM method*), 7
fit() (*twinsvm.TSVM method*), 5

G

get_params_names() (*twinsvm.MCTSVM method*), 6
get_params_names() (*twinsvm.OVO_TSVM method*), 7
get_params_names() (*twinsvm.TSVM method*), 5
grid_search() (*in module eval_classifier*), 11

H

HyperPlane (*class in twinsvm*), 5

I

initializer() (*in module eval_classifier*), 12

M

MCTSVM (*class in twinsvm*), 5
misc (*module*), 13

O

OVO_TSVM (*class in twinsvm*), 6

P

predict() (*twinsvm.MCTSVM method*), 6
predict() (*twinsvm.OVO_TSVM method*), 7
predict() (*twinsvm.TSVM method*), 5
progress_bar_gs() (*in module misc*), 14

R

rbf_kernel() (*in module twinsvm*), 5
read_data() (*in module dataproc*), 13
read_libsvm() (*in module dataproc*), 13

S

save_result() (*in module eval_classifier*), 11
search_space() (*in module eval_classifier*), 11
split_tt_validator() (*eval_classifier.Validator method*), 10

T

time_fmt() (*in module misc*), 13
TSVM (*class in twinsvm*), 4
twinsvm (*module*), 3

V

Validator (*class in eval_classifier*), 9