bace documentation

Release 0.1.0

Krzysztof Joachimiak

Jul 04, 2019

Contents

A deck of Naive Bayes algorithms with sklearn-like API.

CHAPTER 1

Algorithms

- Complement Naive Bayes
- Negation Naive Bayes
- Universal-set Naive Bayes
- Selective Naive Bayes

CHAPTER 2

Installation

You can install this module directly from GitHub repo with command:

or as a PyPI package

CHAPTER 3

Usage

bace mimics Scikit-Learn API, so usage is very simple.

```
from bace import ComplementNB
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer

# Train set
newsgroups_train = fetch_20newsgroups(subset='train', shuffle=True)
X_train = vectorizer.fit_transform(newsgroups_train.data)
y_train = newsgroups_train.target

# Test set
newsgroups_test = fetch_20newsgroups(subset='test', shuffle=True)
X_test = vectorizer.fit_transform(newsgroups_test.data)
y_test = newsgroups_test.target

# Score
cnb = ComplementNB()
cnb.fit(X_train, y_train).accuracy_score(X_test, y_test)
```

Contents:

3.1 bace API

ComplementNB([alpha, weight_normalized])	Complement Naive Bayes classifier
NegationNB([alpha])	Negation Naive Bayes classifier
UniversalSetNB([alpha])	Universal-set Naive Bayes classifier
SelectiveNB([alpha])	Selective Naive Bayes classifier

class bace.**ComplementNB**(*alpha=1.0*, *weight_normalized=False*)

Bases: bace.base.BaseNB

Complement Naive Bayes classifier

References

Rennie J. D. M., Shih L., Teevan J., Karger D. R. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers

https://people.csail.mit.edu/jrennie/papers/icml03-nb.pdf

Parameters

- **alpha** (*float*) – Smoothing parameter
- **weight_normalized** (*bool, default False*) – Enable Weight-normalized Complement Naive Bayes method.

alpha_sum_
Sum of alpha params

Type int

classes_
Classes list

Type array, shape (n_classes,)

class_count_
number of training samples observed in each class.

Type array, shape (n_classes,)

Examples

```
>>> from sklearn.datasets import fetch_20newsgroups
>>> from sklearn.feature_extraction.text import CountVectorizer
>>> from bace import ComplementNB
Prepare data
>>> vectorizer = CountVectorizer()
>>> categories = ['alt.atheism', 'talk.religion.misc','comp.graphics', 'sci.space
↪']
Train set
>>> newsgroups_train = fetch_20newsgroups(subset='train', categories=categories,␣
↪shuffle=True)
>>> train_vectors = vectorizer.fit_transform(newsgroups_train.data)
Test set
>>> newsgroups_test = fetch_20newsgroups(subset='test', categories=categories,␣
↪shuffle=True)
>>> test_vectors = vectorizer.transform(newsgroups_test.data)
>>> clf = ComplementNB()
>>> clf.fit(newsgroups_train, train_vectors).accuracy_score(newsgroups_test, test_
↪vectors)
```

accuracy_score (*X*, *y*)
Return acuracy score

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.

Returns **accuracy_score** – Accuracy on the given test set

Return type float

class_log_proba_
Log probability of class occurrence

complement_class_count_
Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class c

complement_class_log_proba_
Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class c

fit (*X*, *y*)
Fit model to given training set

Parameters

- **X** (*array-like, shape (n_samples, n_features)*) – Training 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.

Returns **self** – Returns self.

Return type Naive Bayes estimator object

get_params (*deep=True*)
Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit (*X*, *y*, *classes=None*)
Incremental fit on a batch of samples.

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.
- **classes** (*array-like, shape = [n_classes], optional (default=None)*) – List of all the classes that can possibly appear in the y vector. Must be provided at the first call to partial_fit, can be omitted in subsequent calls.

Returns **self** – Returns self.

Return type object

predict (*X*)
Perform classification on an array of test vectors X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) – Unseen samples vector

Returns **C** – Predicted target values for X

Return type array, shape = [n_samples]

predict_log_proba(*X*)
Return log-probability estimates for the test vector X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) –

Returns **C** – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

predict_proba(*X*)
Return probability estimates for the test vector X. :param X: :type X: array-like, shape = [n_samples, n_features]

Returns **C** – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

set_params(***params*)
Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns

Return type self

class bace.**NegationNB**(*alpha=1.0*)
Bases: bace.base.BaseNB

Negation Naive Bayes classifier

Parameters **alpha** (*float*) – Smoothing parameter

References

Komiya K., Sato N., Fujimoto K., Kotani Y. (2011). Negation Naive Bayes for Categorization of Product Pages on the Web

http://www.aclweb.org/anthology/R11-1083.pdf

accuracy_score(*X*, *y*)
Return acuracy score

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.

Returns **accuracy_score** – Accuracy on the given test set

Return type float

class_log_proba_
Log probability of class occurrence

complement_class_count_
Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class c

complement_class_log_proba_
Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class c

fit(*X*, *y*)
Fit model to given training set

Parameters

- **X** (*array-like, shape (n_samples, n_features)*) – Training 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.

Returns **self** – Returns self.

Return type Naive Bayes estimator object

get_params(*deep=True*)
Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit(*X*, *y*, *classes=None*)
Incremental fit on a batch of samples.

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.
- **classes** (*array-like, shape = [n_classes], optional (default=None)*) – List of all the classes that can possibly appear in the y vector. Must be provided at the first call to partial_fit, can be omitted in subsequent calls.

Returns **self** – Returns self.

Return type object

predict(*X*)
Perform classification on an array of test vectors X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) – Unseen samples vector

Returns **C** – Predicted target values for X

Return type array, shape = [n_samples]

predict_log_proba(*X*)
Return log-probability estimates for the test vector X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) –

Returns **C** – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

predict_proba(*X*)
Return probability estimates for the test vector X. :param X: :type X: array-like, shape = [n_samples, n_features]

Returns **C** – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

set_params(***params*)
Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns

Return type self

class bace.**UniversalSetNB**(*alpha=1.0*)
Bases: bace.base.BaseNB

Universal-set Naive Bayes classifier

Parameters **alpha** (*float*) – Smoothing parameter

References

Komiya K., Ito Y., Kotani Y. (2013). New Naive Bayes Methods using Data from All Classes

https://github.com/krzjoa/bace/blob/master/papers/snb.pdf

accuracy_score(*X*, *y*)
Return acuracy score

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.

Returns **accuracy_score** – Accuracy on the given test set

Return type float

class_log_proba_
Log probability of class occurrence

complement_class_count_
Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class c

complement_class_log_proba_
Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class c

fit (*X*, *y*)
Fit model to given training set

Parameters

- **X** (*array-like, shape (n_samples, n_features)*) – Training 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.

Returns **self** – Returns self.

Return type Naive Bayes estimator object

get_params (*deep=True*)
Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit (*X*, *y*, *classes=None*)
Incremental fit on a batch of samples.

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.
- **classes** (*array-like, shape = [n_classes], optional (default=None)*) – List of all the classes that can possibly appear in the y vector. Must be provided at the first call to partial_fit, can be omitted in subsequent calls.

Returns **self** – Returns self.

Return type object

predict (*X*)
Perform classification on an array of test vectors X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) – Unseen samples vector

Returns **C** – Predicted target values for X

Return type array, shape = [n_samples]

predict_log_proba (*X*)
Return log-probability estimates for the test vector X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) –

Returns **C** – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

predict_proba(*X*)
Return probability estimates for the test vector X. :param X: :type X: array-like, shape = [n_samples, n_features]

Returns **C** – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

set_params(***params*)
Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns

Return type self

class bace.**SelectiveNB**(*alpha=1.0*)
Bases: bace.base.BaseNB

Selective Naive Bayes classifier

Parameters **alpha** (*float*) – Smoothing parameter

References

Komiya K., Ito Y., Kotani Y. (2013). New Naive Bayes Methods using Data from All Classes

https://github.com/krzjoa/bace/blob/master/papers/snb.pdf

accuracy_score(*X*, *y*)
Return acuracy score

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.

Returns **accuracy_score** – Accuracy on the given test set

Return type float

class_log_proba_
Log probability of class occurrence

complement_class_count_
Complement class count, i.e. number of occurrences of all the samples with all the classes except the given class c

complement_class_log_proba_
Complement class probability, i.e. logprob of occurrence of a sample, which does not belong to the given class c

fit(*X*, *y*)
Fit model to given training set

Parameters

- **X** (*array-like, shape (n_samples, n_features)*) – Training 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.

Returns **self** – Returns self.

Return type Naive Bayes estimator object

get_params(*deep=True*)
Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

partial_fit(*X*, *y*, *classes=None*)
Incremental fit on a batch of samples.

Parameters

- **X** (*{array-like, sparse matrix}, shape = [n_samples, n_features]*) – Training 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.
- **classes** (*array-like, shape = [n_classes], optional (default=None)*) – List of all the classes that can possibly appear in the y vector. Must be provided at the first call to partial_fit, can be omitted in subsequent calls.

Returns **self** – Returns self.

Return type object

predict(*X*)
Perform classification on an array of test vectors X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) – Unseen samples vector

Returns **C** – Predicted target values for X

Return type array, shape = [n_samples]

predict_log_proba(*X*)
Return log-probability estimates for the test vector X.

Parameters **X** (*array-like, shape = [n_samples, n_features]*) –

Returns **C** – Returns the log-probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

predict_proba(*X*)
Return probability estimates for the test vector X. :param X: :type X: array-like, shape = [n_samples, n_features]

Returns **C** – Returns the probability of the samples for each class in the model. The columns correspond to the classes in sorted order, as they appear in the attribute *classes_*.

Return type array-like, shape = [n_samples, n_classes]

set_params(***params*)
Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns

Return type self

class bace.**Benchmark**(*classifiers*, *verbose=False*)
Bases: `sklearn.base.BaseEstimator`

scikit-learn like classifiers benchmark

Parameters

- **classifiers** (*list of sklearn.base.BaseEstimator*) – List of sklearn classifiers
- **verbose** (*bool*) – Print training details

compare(*X*, *y*, *metrics={'Accuracy': <function accuracy_score>}*)
Compare predictions of multiple classifiers

Parameters

- **X** (*numpy.ndarray*) – Features
- **y** (*numpy.ndarray*) – Targets
- **metrics** (*dict of callable*) – List of metric functions

fit(*X*, *y*)
Fit several classifiers

Parameters

- **X** (*numpy.ndarray*) –
- **y** (*numpy.ndarray*) – Labels

get_params(*deep=True*)
Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

predict(*X*)

set_params(***params*)
Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form `<component>__<parameter>` so that it's possible to update each component of a nested object.

Returns

Return type self

class bace.**BenchmarkNaiveBayes**

Bases: bace.benchmark.Benchmark

CLASSIFIERS = [MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True), ComplementN

compare(*X*, *y*, *metrics={'Accuracy': <function accuracy_score>}*)

Compare predictions of multiple classifiers

Parameters

- **X** (*numpy.ndarray*) – Features
- **y** (*numpy.ndarray*) – Targets
- **metrics** (*dict of callable*) – List of metric functions

fit(*X*, *y*)

Fit several classifiers

Parameters

- **X** (*numpy.ndarray*) –
- **y** (*numpy.ndarray*) – Labels

get_params(*deep=True*)

Get parameters for this estimator.

Parameters **deep** (*boolean, optional*) – If True, will return the parameters for this estimator and contained subobjects that are estimators.

Returns **params** – Parameter names mapped to their values.

Return type mapping of string to any

predict(*X*)

set_params(***params*)

Set the parameters of this estimator.

The method works on simple estimators as well as on nested objects (such as pipelines). The latter have parameters of the form <component>__<parameter> so that it's possible to update each component of a nested object.

Returns

Return type self

- genindex

modindex search

Python Module Index

b

Index

S

U