
sparse_som Documentation

Release 0.5

J. Melka

Nov 26, 2020

Contents

1	Module contents	1
1.1	enums	1
1.2	classes	1
2	Submodules	5
2.1	sparse_som.classifier module	5
3	Indices and tables	7
	Python Module Index	9
	Index	11

CHAPTER 1

Module contents

1.1 enums

```
class sparse_som.cooling

LINEAR = <cooling.LINEAR: 0>
EXPONENTIAL = <cooling.EXPONENTIAL: 1>

class sparse_som.topology

CIRC = <topology.CIRC: 4>
HEXA = <topology.HEXA: 6>
RECT = <topology.RECT: 8>
```

1.2 classes

Self-Organizing Maps wrappers for python, intended for sparse input data.

class sparse_som.BSom

Uses the batch algorithm and can take advantage from multi-core processors to learn efficiently.

Parameters

- **h** (*int*) – the network height
- **w** (*int*) – the network width
- **d** (*int*) – the dimension of input vectors
- **topol** (*topology.RECT* or *topology.HEXA*) – the network topology
- **verbose** (*int (0..2)*) – verbosity parameter

train (*data*, *epochs*=10, *r0*=0, *rN*=0.5, *std*=0.3, *cool*=*cooling.LINEAR*)

Train the network with data.

Parameters

- **data** (`scipy.sparse.spmatrix`) – sparse input matrix (ideally `csr_matrix` of `numpy.single`)
- **epochs** (`int`) – number of epochs
- **r0** (`float`) – radius at the first iteration
- **rN** (`float`) – radius at the last iteration
- **cool** (`cooling.LINEAR` or `cooling.EXPONENTIAL`) – cooling strategy

bmus (*data*)

Return the best match units for data.

Parameters **data** (`scipy.sparse.spmatrix`) – sparse input matrix (ideally `csr_matrix` of `numpy.single`)

Returns an array of the bmu coordinates (y,x)

Return type 2D `numpy.ndarray`

codebook

Returns a view of the internal codebook.

Return type 3D `numpy.ndarray`

dim

Returns the dimension of the input vectors.

Return type `int`

ncols

Returns the number of columns in the network.

Return type `int`

nrows

Returns the number of rows in the network.

Return type `int`

class `sparse_som.Som`

Uses the SD-SOM algorithm (online learning).

Parameters

- **h** (`int`) – the network height
- **w** (`int`) – the network width
- **d** (`int`) – the dimension of input vectors
- **topol** (`topology.RECT` or `topology.HEXA`) – the network topology
- **verbose** (`int (0..2)`) – verbosity parameter

train (*data*, *tmax*, *r0*=0, *a0*=0.5, *rN*=0.5, *aN*=0., *std*=0.3, *rcool*=*cooling.LINEAR*, *acool*=*cooling.LINEAR*)

Train the network with data.

Parameters

- **data** (`scipy.sparse.spmatrix`) – sparse input matrix (ideally `csr_matrix` of `numpy.single`)
- **tmax** (`int`) – number of iterations
- **r0** (`float`) – radius at the first iteration
- **a0** (`float`) – learning-rate at the first iteration
- **rN** (`float`) – radius at the last iteration
- **aN** (`float`) – learning-rate at the last iteration
- **rcool** (`cooling.LINEAR` or `cooling.EXPONENTIAL`) – radius cooling strategy
- **acool** (`cooling.LINEAR` or `cooling.EXPONENTIAL`) – alpha cooling strategy

bmus (`data`)

Return the best match units for data.

Parameters `data` (`scipy.sparse.spmatrix`) – sparse input matrix (ideally `csr_matrix` of `numpy.single`)

Returns an array of the bmu coordinates (y,x)

Return type 2D `numpy.ndarray`

codebook

Returns a view of the internal codebook.

Return type 3D `numpy.ndarray`

dim

Returns the dimension of the input vectors.

Return type `int`

ncols

Returns the number of columns in the network.

Return type `int`

nrows

Returns the number of rows in the network.

Return type `int`

CHAPTER 2

Submodules

2.1 sparse_som.classifier module

```
class sparse_som.SomClassifier(cls=<type 'sparse_som.som.BSom'>, *args, **kwargs)
```

```
__init__(cls=<type 'sparse_som.som.BSom'>, *args, **kwargs)
```

Parameters

- **cls** (*Som* or *BSom*) – SOM constructor
- ***args** – positional parameters for the constructor
- ****kwargs** – named parameters for the constructor

```
fit(data, labels, **kwargs)
```

Training the SOM on the the data and calibrate itself.

After the training, *self.quant_error* and *self.topog_error* are respectively set.

Parameters

- **data** (`scipy.sparse.csr_matrix`) – sparse input matrix (ideal dtype is `numpy.float32`)
- **labels** (`iterable`) – the labels associated with data
- ****kwargs** – optional parameters for `train()`

```
bmus_with_errors(data)
```

Compute common error metrics (Quantization err. and Topographic err.) for this data.

Parameters **data** (`scipy.sparse.csr_matrix`) – sparse input matrix (ideal dtype is `numpy.float32`)

Returns the BMUs, the QE and the TE

Return type `tuple`

`predict` (*data, unknown=None*)

Classify data according to previous calibration.

Parameters

- **data** (`scipy.sparse.csr_matrix`) – sparse input matrix (ideal dtype is `numpy.float32`)
- **unknown** – the label to attribute if no label is known

Returns the labels guessed for data

Return type `numpy.array`

`fit_predict` (*data, labels, unknown=None*)

Fit and classify data efficiently.

Parameters

- **data** (`scipy.sparse.csr_matrix`) – sparse input matrix (ideal dtype is `numpy.float32`)
- **labels** (`iterable`) – the labels associated with data
- **unknown** – the label to attribute if no label is known

Returns the labels guessed for data

Return type `numpy.array`

`get_precision()`

Returns the ratio part of the dominant label for each unit.

Return type 2D `numpy.ndarray`

`histogram` (*bmus=None*)

Return a 2D histogram of bmus.

Parameters **bmus** (`numpy.ndarray`) – the best-match units indexes for underlying data.

Returns the computed 2D histogram of bmus.

Return type `numpy.ndarray`

CHAPTER 3

Indices and tables

- genindex
- search

Python Module Index

S

[sparse_som](#), 1

Symbols

`__init__()` (*sparse_som.SomClassifier method*), 5

B

`bmus()` (*sparse_som.BSom method*), 2

`bmus()` (*sparse_som.Som method*), 3

`bmus_with_errors()` (*sparse_som.SomClassifier method*), 5

`BSom` (*class in sparse_som*), 1

C

`CIRC` (*sparse_som.topology attribute*), 1

`codebook` (*sparse_som.BSom attribute*), 2

`codebook` (*sparse_som.Som attribute*), 3

`cooling` (*class in sparse_som*), 1

D

`dim` (*sparse_som.BSom attribute*), 2

`dim` (*sparse_som.Som attribute*), 3

E

`EXPONENTIAL` (*sparse_som.cooling attribute*), 1

F

`fit()` (*sparse_som.SomClassifier method*), 5

`fit_predict()` (*sparse_som.SomClassifier method*), 6

G

`get_precision()` (*sparse_som.SomClassifier method*), 6

H

`HEXA` (*sparse_som.topology attribute*), 1

`histogram()` (*sparse_som.SomClassifier method*), 6

L

`LINEAR` (*sparse_som.cooling attribute*), 1

N

`ncols` (*sparse_som.BSom attribute*), 2

`ncols` (*sparse_som.Som attribute*), 3

`nrows` (*sparse_som.BSom attribute*), 2

`nrows` (*sparse_som.Som attribute*), 3

P

`predict()` (*sparse_som.SomClassifier method*), 5

R

`RECT` (*sparse_som.topology attribute*), 1

S

`Som` (*class in sparse_som*), 2

`SomClassifier` (*class in sparse_som*), 5

`sparse_som` (*module*), 1

T

`topology` (*class in sparse_som*), 1

`train()` (*sparse_som.BSom method*), 1

`train()` (*sparse_som.Som method*), 2