
SimpSOM Documentation

Release 1.2.0

Federico Comitani

Jun 17, 2017

CONTENTS:

1	Introduction	1
1.1	Dependencies	1
1.2	Example of Usage	1
2	SimpSOM	3
2.1	SimpSOM package	3
3	Indices and tables	9
	Python Module Index	11

INTRODUCTION

SimpSOM is a lightweight implementation of Kohonen Self-Organising Maps (SOM) for Python 2.7, useful for unsupervised learning, clustering and dimensionality reduction.

The package is now available on PyPI, to retrieve it just type `pip install SimpSOM` or download it from [here](#) and install with `python setup.py install`.

It allows to build and train SOM on your dataset, save/load the trained network weights, and display or print graphs of the network with selected features. The function `run_colorsExample()` will run a toy model, where a number of colors will be mapped from the 3D RGB space to the 2D network map and clustered according to their similarity in the origin space.

1.1 Dependencies

- Numpy 1.11.0 (older versions may work);
- Matplotlib 1.5.1 (older versions may work);
- Sklearn 0.15 (older versions may work), optional, needed only for clustering with algorithms other than Quality Threshold.

1.2 Example of Usage

Here is a quick example on how to use the library with a `raw_data` dataset:

```
#Import the library
import SimpSOM as sps

#Build a network 20x20 with a weights format taken from the raw_data.
net = sps.somNet(20, 20, raw_data)

#Train the network for 10000 epochs and with initial learning rate of 0.1.
net.train(10000, 0.01)

#Save the weights to file
net.save('filename_weights')

#Print a map of the network nodes and colour them according to the first feature_
↪ (column number 0) of the dataset
#and then according to the distance between each node and its neighbours.
net.nodes_graph(colnum=0)
net.diff_graph()
```

```
#Project the datapoints on the new 2D network map.
net.project(raw_data, labels=labels)

#Cluster the datapoints according to the Mean Shift algorithm from sklearn.
net.cluster(raw_data, type='MeanShift')
```

2.1 SimpSOM package

2.1.1 Submodules

2.1.2 SimpSOM.densityPeak module

A Rodriguez, A Laio, Clustering by fast search and find of density peaks SCIENCE, 1492, vol 322 (2014)

F. Comitani @2017

class SimpSOM.densityPeak.**collection** (*coorArray*, *typeFunc*='gaussian', *percent*=0.02)

Class for a collection of point objects.

cluster_assign ()

Assign a cluster to each point according according to its nearest neighbour with higher density.

core_assign ()

Assign points as belonging to the core or the halo of a cluster.

decision_graph (*show*=False, *printout*=True)

Calculate the decision graph, delta vs rho for the points belonging to the collection and find the cluster centers.

Parameters

- **show** (*bool*, *optional*) – Choose to display the plot.
- **printout** (*bool*, *optional*) – Choose to save the plot to a file.

get_clusterList ()

Returns the indeces of the clustered points as a list.

Returns a list of lists containing the points indices belonging to each cluster

Return type clusters (list, int)

refd = None

Make sure rhos are set before setting deltas

set_deltas ()

Calculate the distance from higher density points for each point in the dataset.

set_dists ()

Calculate the distance matrix for all points.

set_rhos (*typeFunc*='step')

Calculate the density for each point in the dataset.

Parameters **typeFunc** (*str*) – step function type (step, gaussian or logistic)

SimpSOM.densityPeak.densityPeak (*sample*, *show=False*, *printout=False*, *percent=0.02*)

Run the complete clustering algorithm in one go and returns the clustered indeces as a list.

Parameters

- **sample** (*array*) – The input dataset
- **show** (*bool*, *optional*) – Choose to display the decision graph.
- **printout** (*bool*, *optional*) – Choose to save the decision graph to a file.

Returns a list of lists containing the points indices belonging to each cluster

Return type clusters (list, int)

SimpSOM.densityPeak.dist (*p1*, *p2*, *metric='euclid'*)

Calculate the distance between two point objects in a N dimensional space according to a given metric.

Parameters

- **p1** (*point*) – First point object for the distance.
- **p2** (*point*) – Second point object for the distance.
- **metric** (*string*) – Metric to use. For now only euclidean distance is implemented.

Returns The distance between the two points.

Return type float)

SimpSOM.densityPeak.gaussian (*p1*, *p2*, *sigma*)

Gaussian function of the distance between two points scaled with sigma.

Parameters

- **p1** (*point*) – First point object for the distance.
- **p2** (*point*) – Second point object for the distance.
- **sigma** (*float*) – The scaling factor for the distance.

Returns value of the gaussian function.

Return type (float)

class **SimpSOM.densityPeak.pt** (*coordinates*)

Class for the points to cluster.

set_delta (*coll*)

Calculate the distance of the point from higher density points and set the nearest neighbour. [Deprecated]

Parameters **coll** (*collection*) – collection containing all the points of the dataset used to calculate the distance.

set_dist (*coll*)

Calculate the distances from all other points in a collection. [Deprecated]

Parameters **coll** (*collection*) – collection containing all the points of the dataset used to calculate the distances.

set_rho (*coll*, *typeFunc*='step')

Calculate the density of the single point for a given dataset. [Deprecated]

Parameters

- **coll** (*collection*) – collection containing all the points of the dataset used to calculate the density.
- **typeFunc** (*str*) – step function type (step, gaussian kernel or logistic).

`SimpSOM.densityPeak.sigmoid(p1, p2, sigma)`

Logistic function of the distance between two points scaled with sigma.

Parameters

- **p1** (*point*) – First point object for the distance.
- **p2** (*point*) – Second point object for the distance.
- **sigma** (*float*) – The scaling factor for the distance.

Returns value of the logistic function.

Return type (float)

`SimpSOM.densityPeak.step(p1, p2, cutoff)`

Step function activated when the distance of two points is less than the cutoff.

Parameters

- **p1** (*point*) – First point object for the distance.
- **p2** (*point*) – Second point object for the distance.
- **cutoff** (*float*) – The cutoff to define the proximity of the points.

Returns 1 if the points are closer than the cutoff, 0 otherwise.

Return type (int)

`SimpSOM.densityPeak.test()`

Run the complete clustering algorithm on a test case and print the clustered points graph.

2.1.3 Module contents

SimpSOM (Simple Self-Organizing Maps) v1.2.0 F. Comitani @2017

A lightweight python library for Kohonen Self-Organising Maps (SOM).

`SimpSOM.run_colorsExample()`

Example of usage of SimpSOM: a number of vectors of length three (corresponding to the RGB values of a color) are used to briefly train a small network. Different example graphs are then printed from the trained network.

class `SimpSOM.somNet(netHeight, netWidth, data, loadFile=None)`

Kohonen SOM Network class.

cluster (*array, type='qthresh', cutoff=5, quant=0.2, percent=0.02, savefile=True, filetype='dat', show=False, printout=True*)

Clusters the data in a given array according to the SOM trained map. The clusters can also be plotted.

Parameters

- **array** (*np.array*) – An array containing datapoints to be clustered.

- **type** (*str, optional*) – The type of clustering to be applied, so far only quality threshold (qthresh) algorithm is directly implemented, other algorithms require sklearn.
- **cutoff** (*float, optional*) – Cutoff for the quality threshold algorithm. This also doubles as maximum distance of two points to be considered in the same cluster with DBSCAN.
- **percent** (*float, optional*) – The percentile that defines the reference distance in density peak clustering (dpeak).
- **quant** (*float, optional*) – Quantile used to calculate the bandwidth of the mean shift algorithm.
- **savefile** (*bool, optional*) – Choose to save the resulting clusters in a text file.
- **filetype** (*string, optional*) – Format of the file where the clusters will be saved (csv or dat)
- **show** (*bool, optional*) – Choose to display the plot.
- **printout** (*bool, optional*) – Choose to save the plot to a file.

Returns A nested list containing the clusters with indexes of the input array points.

Return type (list of int)

data = None

Load the weights from file or generate them randomly.

diff_graph (*show=False, printout=True*)

Plot a 2D map with nodes and weights difference among neighbouring nodes.

Parameters

- **show** (*bool, optional*) – Choose to display the plot.
- **printout** (*bool, optional*) – Choose to save the plot to a file.

find_bmu (*vec*)

Find the best matching unit (BMU) for a given vector.

Parameters **vec** (*np.array*) – The vector to match.

Returns The best matching unit node.

Return type *bmu* (*somNode*)

nodes_graph (*colnum=0, show=False, printout=True*)

Plot a 2D map with nodes and weights values

Parameters

- **colnum** (*int*) – The index of the weight that will be shown as colormap.
- **show** (*bool, optional*) – Choose to display the plot.
- **printout** (*bool, optional*) – Choose to save the plot to a file.

project (*array, colnum=0, labels=[], show=False, printout=True*)

Project the datapoints of a given array to the 2D space of the SOM by calculating the bmus. If requested plot a 2D map with as implemented in `nodes_graph` and adds circles to the bmu of each datapoint in a given array.

Parameters

- **array** (*np.array*) – An array containing datapoints to be mapped.
- **colnum** (*int*) – The index of the weight that will be shown as colormap.
- **show** (*bool, optional*) – Choose to display the plot.
- **printout** (*bool, optional*) – Choose to save the plot to a file.

Returns bmu x,y position for each input array datapoint.

Return type (list)

save (*fileName='somNet_trained'*)

Saves the nodes weights to a file.

Parameters **fileName** (*str, optional*) – Name of file where the weights will be saved.

train (*epochs=5000, startLearnRate=0.01*)

Train the SOM.

Parameters

- **epochs** (*int*) – Number of training iterations.
- **startLearnRate** (*float*) – Initial learning rate.

update_lrate (*iter*)

Update the learning rate.

Parameters **iter** (*int*) – Iteration number.

update_sigma (*iter*)

Update the gaussian sigma.

Parameters **iter** (*int*) – Iteration number.

class `SimpSOM.somNode` (*x, y, numWeights, weiArray=[]*)

Single Kohonen SOM Node class.

get_distance (*vec*)

Calculate the distance between the weights vector of the node and a given vector.

Parameters **vec** (*np.array*) – The vector from which the distance is calculated.

Returns The distance between the two weight vectors.

Return type (float)

get_nodeDistance (*node*)

Calculate the distance within the network between the node and another node.

Parameters **node** (`somNode`) – The node from which the distance is calculated.

Returns The distance between the two nodes.

Return type (float)

update_weights (*inputVec, sigma, lrate, bmu*)

Update the node Weights.

Parameters

- **inputVec** (*np.array*) – A weights vector whose distance drives the direction of the update.
- **sigma** (*float*) – The updated gaussian sigma.
- **lrate** (*float*) – The updated learning rate.

- **bmu** (`somNode`) – The best matching unit.

INDICES AND TABLES

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

PYTHON MODULE INDEX

S

`SimpSOM`, [5](#)

`SimpSOM.densityPeak`, [3](#)