JustGlowing / **minisom** Public

master

| | |
|---|---|
| **JustGlowing** solution to rounding error issue documentation ✓ | |
| .github/workflows | newest python version added |
| examples | basic usage updated |
| .gitignore | fixed test with quantization error, extracted common code to |
| LICENSE | Create LICENSE |
| Readme.md | 400 citations update Readme.md |
| minisom.py | solution to rounding error issue documentation |
| setup.cfg | remove dash separate description file |
| setup.py | realease prep |


# MiniSom

## Self Organizing Maps

MiniSom is a minimalistic and Numpy based implementation of the Self Organizing Maps (SOM). SOM is a type of Artificial Neural Network able dimensional data items into simple geometric relationships on a low-dimensional display. Minisom is designed to allow researchers to easily bu

The project initially aimed for a minimalistic implementation of the Self-Organizing Map (SOM) algorithm, focusing on simplicity in features, de remains minimalistic by relying only on the numpy library and emphasizing vectorization in coding style.

Updates about MiniSom are posted on X.

Jump into using MiniSom via Google Colab: 


## Installation

Just use pip:

```
pip install minisom
```

or download MiniSom to a directory of your choice and use the setup script:

```
git clone https://github.com/JustGlowing/minisom.git
python setup.py install
```

Note that the commands above will install the latest version of MiniSom which might contain changes that are not parte of the stable release.

## How to use it

In order to use MiniSom you need your data organized as a Numpy matrix where each row corresponds to an observation or as list of lists like

```
data = [[ 0.80,  0.55,  0.22,  0.03],
        [ 0.82,  0.50,  0.23,  0.03],
        [ 0.80,  0.54,  0.22,  0.03],
        [ 0.80,  0.53,  0.26,  0.03],
        [ 0.79,  0.56,  0.22,  0.03],
```

```
        [ 0.75,  0.60,  0.25,  0.03],
        [ 0.77,  0.59,  0.22,  0.03]]
```

Then you can train MiniSom just as follows:

```
from minisom import MiniSom
som = MiniSom(6, 6, 4, sigma=0.3, learning_rate=0.5) # initialization of 6x6 SOM
som.train(data, 100) # trains the SOM with 100 iterations
```

You can obtain the position of the winning neuron on the map for a given sample as follows:

```
som.winner(data[0])
```

For an overview of all the features implemented in minisom you can browse the following examples: https://github.com/JustGlowing/minisom/

**Export a SOM and load it again**

A model can be saved using pickle as follows

```
import pickle
som = MiniSom(7, 7, 4)

# ...train the som here

# saving the som in the file som.p
with open('som.p', 'wb') as outfile:
    pickle.dump(som, outfile)
```

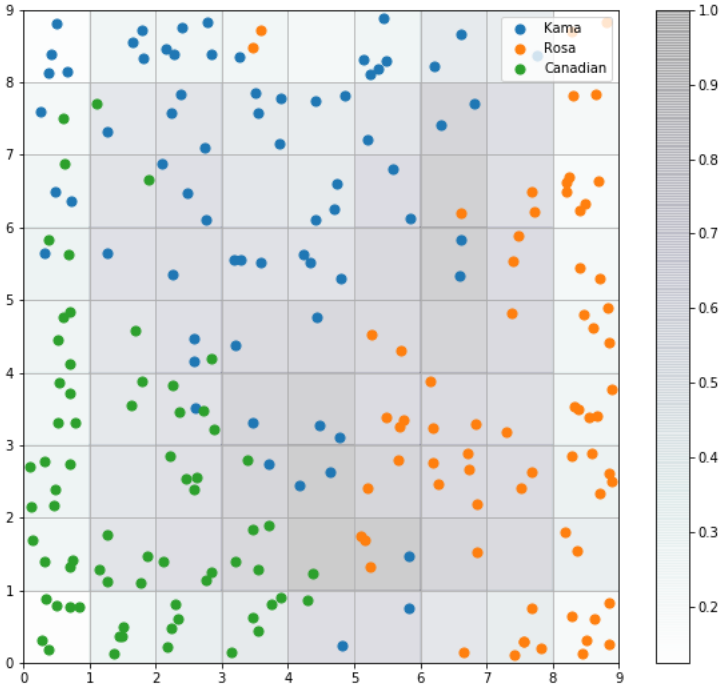
and can be loaded as follows

```
with open('som.p', 'rb') as infile:
    som = pickle.load(infile)
```

Note that if a lambda function is used to define the decay factor MiniSom will not be pickable anymore.

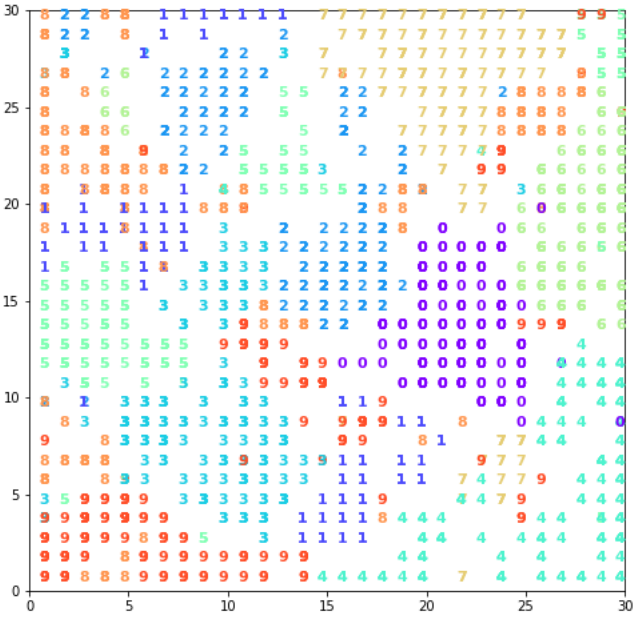
## Examples

Here are some of the charts you'll see how to generate in the examples:



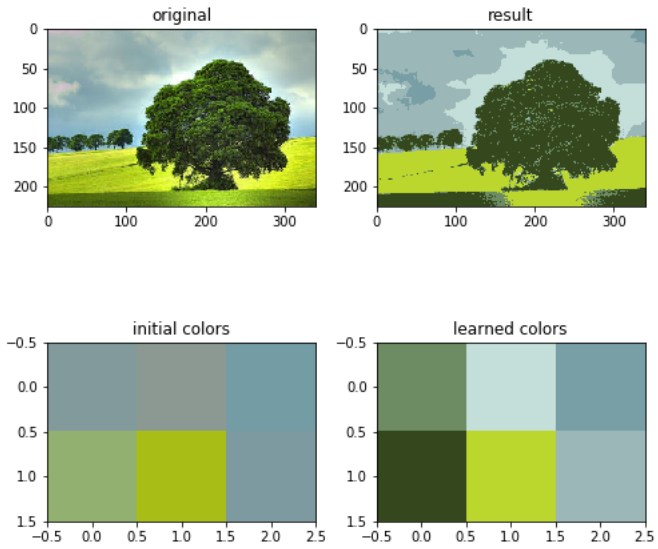

Seeds map

Class assignment



Handwritten digits mapping

Hexagonal Topolo




Color quantization

Outliers detection

## Other tutorials

- Self Organizing Maps on the Glowing Python
- How to solve the Travelling Salesman Problem from the book *Optimization Algorithms:Optimization Algorithms: AI techniques for design, plann*
- Lecture notes from the Machine Learning course at the University of Lisbon
- Introduction to Self-Organizing by Derrick Mwiti
- Self Organizing Maps on gapminder data [in German]
- Discovering SOM, an Unsupervised Neural Network by Gisely Alves
- Video tutorials made by the GeoEngineerings School: Part 1; Part 2; Part 3; Part 4
- Video tutorial Self Organizing Maps: Introduction by Art of Visualization
- Video tutorial Self Organizing Maps Hyperparameter tuning by SuperDataScience Machine Learning

## How to cite MiniSom

```
@misc{vettigliminisom,
  title={MiniSom: minimalistic and NumPy-based implementation of the Self Organizing Map},
  author={Giuseppe Vettigli},
  year={2018},
  url={https://github.com/JustGlowing/minisom/},
}
```

MiniSom has been cited more than 400 times, check out the research where MiniSom was used here.

## Guidelines to contribute

1. In the description of your Pull Request explain clearly what it implements or fixes. In cases that the PR is about a code speedup, report a re
2. Give your pull request a helpful title that summarises what your contribution does.
3. Write unit tests for your code and make sure the existing ones are up to date. `pytest` can be used for this:

```
pytest minisom.py
```

4. Make sure that there are no stylistic issues using `pycodestyle`:

```
pycodestyle minisom.py
```

5. Make sure your code is properly commented and documented. Each public method needs to be documented as the existing ones.