The Sky Metaphor, Stardinates, for Self-Organising Maps

SOM Exercise 3 : Coding

Group b2: Maximilian Holzmueller, 11770953 | Bijan Nikkhah, 11930506 | Gustavo Torres, 12122457

In [1]:
```
from SOMToolBox_Parse import SOMToolBox_Parse
from somtoolbox import SOMToolbox
from minisom import MiniSom
```

Sky Metaphor Visualization

The sky metaphor visualization is presented below. Controllers are available for a better interaction with the tool. The developed code is in file `SOMToolbox-main\visualizations\sky_metaphor.py`. Some changes were added in files `SOMToolbox-main\controls\controllers.py` and `SOMToolbox-main\somtoolbox.py`.

The code consists of three main functions: `getMapPosition`, `calculateExactPosition` and `calculatePointsPosition`. The first function calculates the position of the Best Matching Unit (BMU) of an input vector. The second function calculates the position of the input vector within the BMU. Finally, the last function combines the previous functions to get the exact position of all input vectors.

We based the development on the following paper: Latif, Khalid & Mayer, Rudolf. (2007). Sky-Metaphor Visualisation for Self-Organising Maps. Regarding the calculation of the exact position within the BMU, the group changed the formula to the following:
$p_{<x,y>} =< \lambda * \sum_{i=2}^{k} F_i * (U_{i<x>} - U_{1<x>}), \lambda * \sum_{i=2}^{k} F_i * (U_{i<y>} - U_{1<y>}) >$

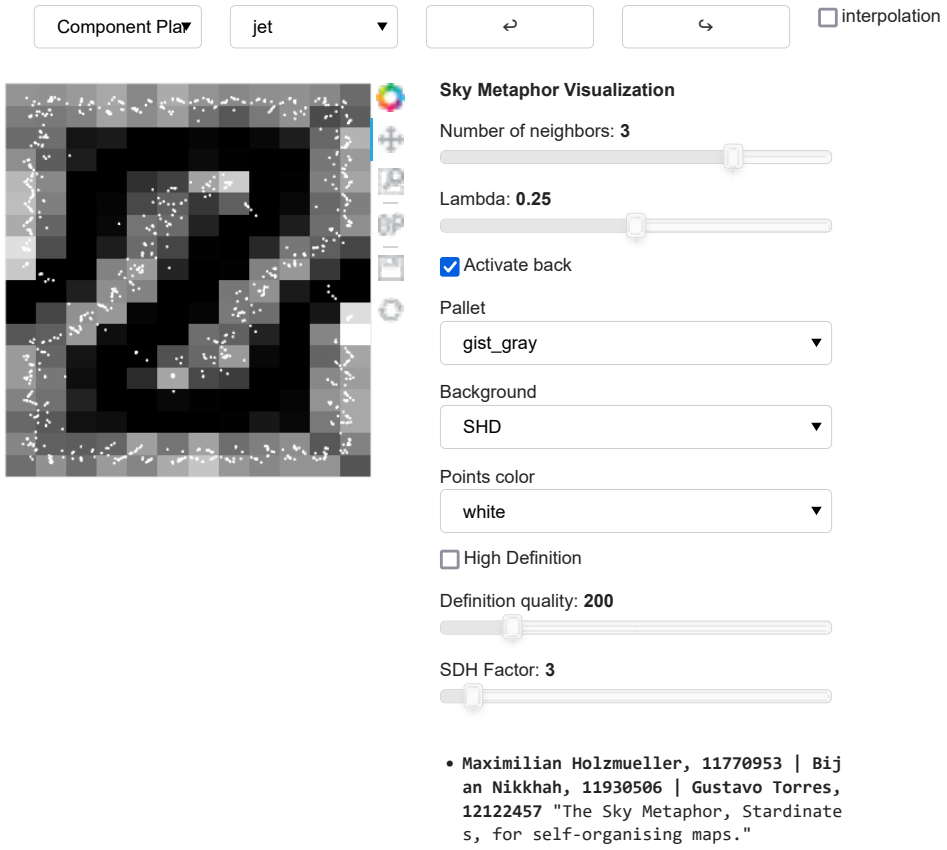
The force F_i controls how much the point goes towards the unit i. The displacement $U_{i<x>} - U_{1<x>}$ controls the direction where the point goes, as a vector. In the formula presented in the paper, if the points were close together the displacement would explode, so it made more sense to multiply and we got more consistent results with the paper this way.

In [2]:
```
idata = SOMToolBox_Parse("datasets\\10clusters\\10clusters.vec").read_weight_file()
weights = SOMToolBox_Parse("datasets\\10clusters\\10clusters.wgt").read_weight_file()
classes = SOMToolBox_Parse("datasets\\10clusters\\10clusters.cls").read_weight_file()
```

In [3]:
```
idata = SOMToolBox_Parse("datasets\\chainlink\\chainlink.vec").read_weight_file()
weights = SOMToolBox_Parse("datasets\\chainlink\\chainlink.wgt").read_weight_file()
classes = SOMToolBox_Parse("datasets\\chainlink\\chainlink.cls").read_weight_file()
```

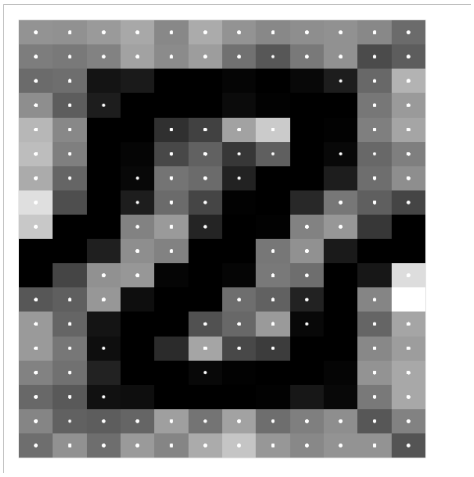
In [4]:
```
sm = SOMToolbox(weights=weights['arr'],m=weights['ydim'],n=weights['xdim'],
                dimension=weights['vec_dim'], input_data=idata['arr'],
               classes=classes['arr'], component_names=classes['classes_names'])
sm._mainview
```

Out[4]:

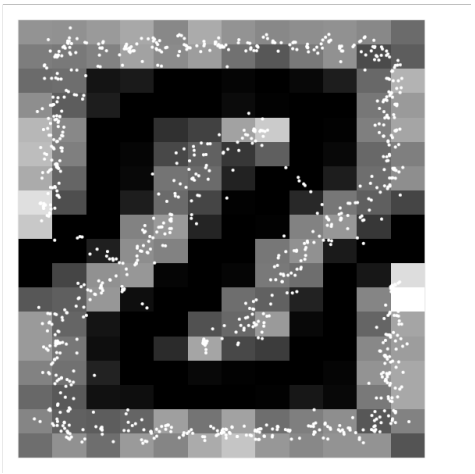


Lambda

The following figures show the effect of *Lambda* on the Sky Metaphor visualization. In the extreme case of *Lambda = 0* we can see that there is no displacement from the center of the cells, which is expected.



Chain Link Dataset (Lambda = 0)



Chain Link Dataset (Lambda = 0.5)

Training and Sky Metaphor Visualization

Chainlink Dataset

In [5]:
```
idata = SOMToolBox_Parse("datasets\\chainlink\\chainlink.vec").read_weight_file()
weights = SOMToolBox_Parse("datasets\\chainlink\\chainlink.wgt").read_weight_file()
classes = SOMToolBox_Parse("datasets\\chainlink\\chainlink.cls").read_weight_file()
```

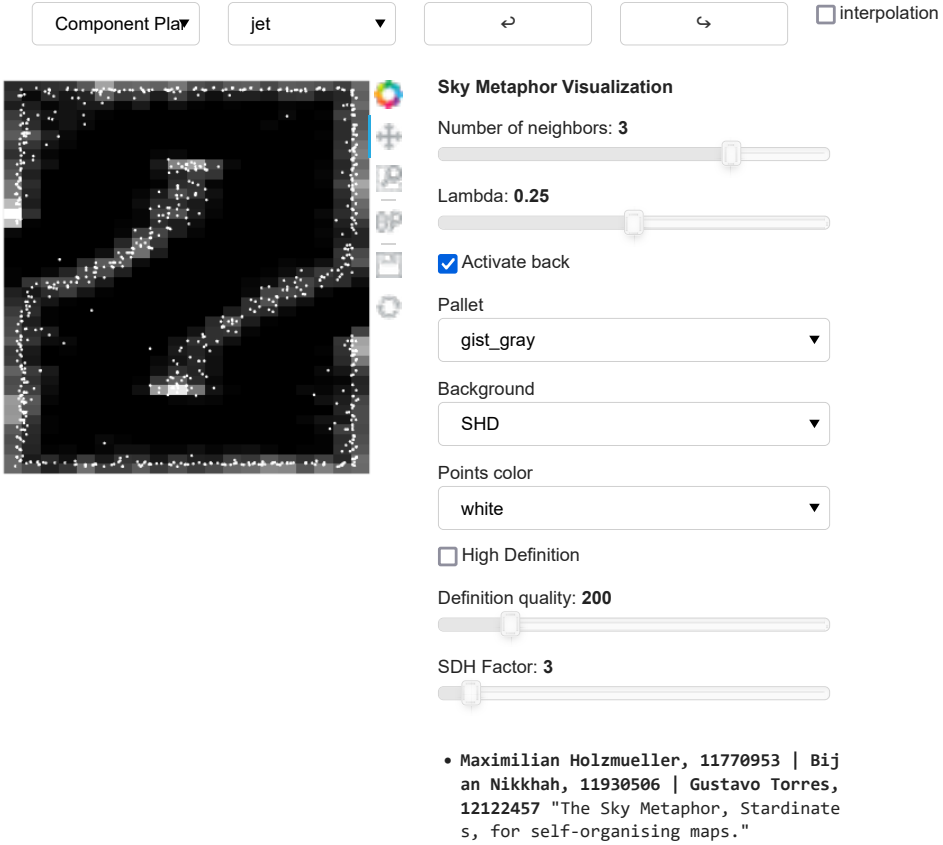
Small SOM (40 x 20)

In [6]:
```
m = 40
n = 20
dim = idata['vec_dim']

som = MiniSom(m, n, dim, sigma=10, learning_rate=0.7)
som.train(idata['arr'], 5000, True, False)

sm = SOMToolbox(weights=som._weights.reshape(-1,dim), m=m, n=n, dimension=dim, input_data=idata['arr'])
sm._mainview
```

Out[6]:



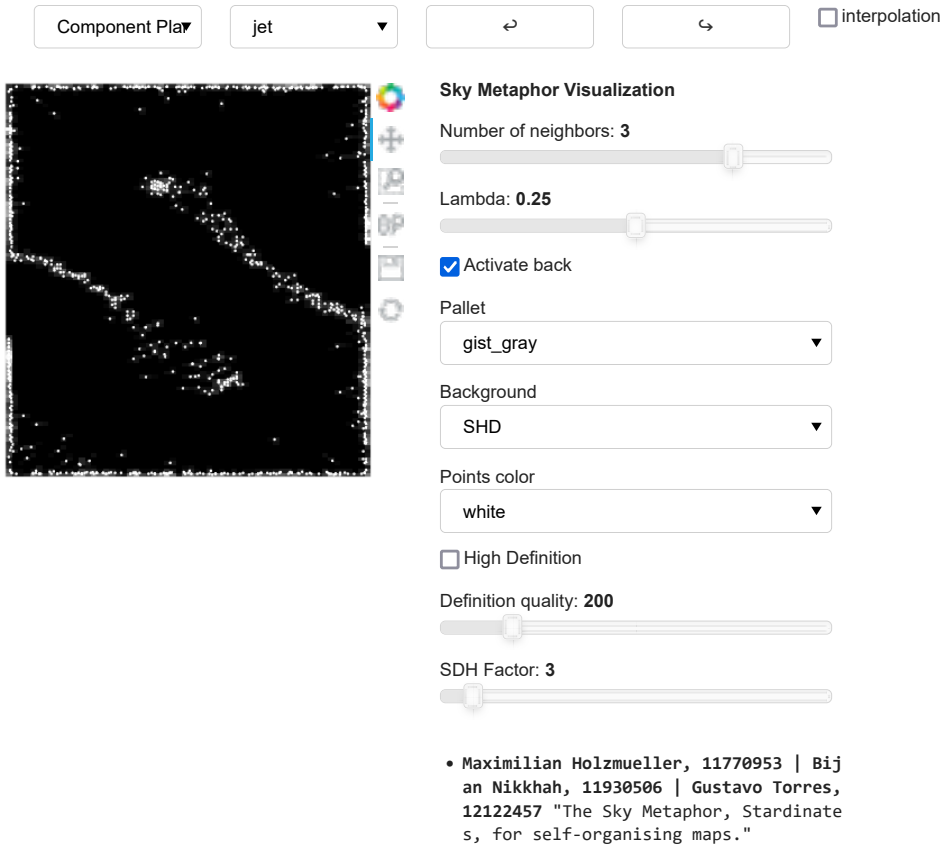
Large SOM (100 x 60)

In [7]:
```
m = 100
n = 60
dim = idata['vec_dim']

som = MiniSom(m, n, dim, sigma=30, learning_rate=0.5)
som.train(idata['arr'], 5000, True, False)

sm = SOMToolbox(weights=som._weights.reshape(-1,dim), m=m, n=n, dimension=dim, input_data=idata['arr'])
sm._mainview
```

Out[7]:



10 Clusters Dataset

In [8]:
```
idata = SOMToolBox_Parse("datasets\\10clusters\\10clusters.vec").read_weight_file()
weights = SOMToolBox_Parse("datasets\\10clusters\\10clusters.wgt").read_weight_file()
classes = SOMToolBox_Parse("datasets\\10clusters\\10clusters.cls").read_weight_file()
```

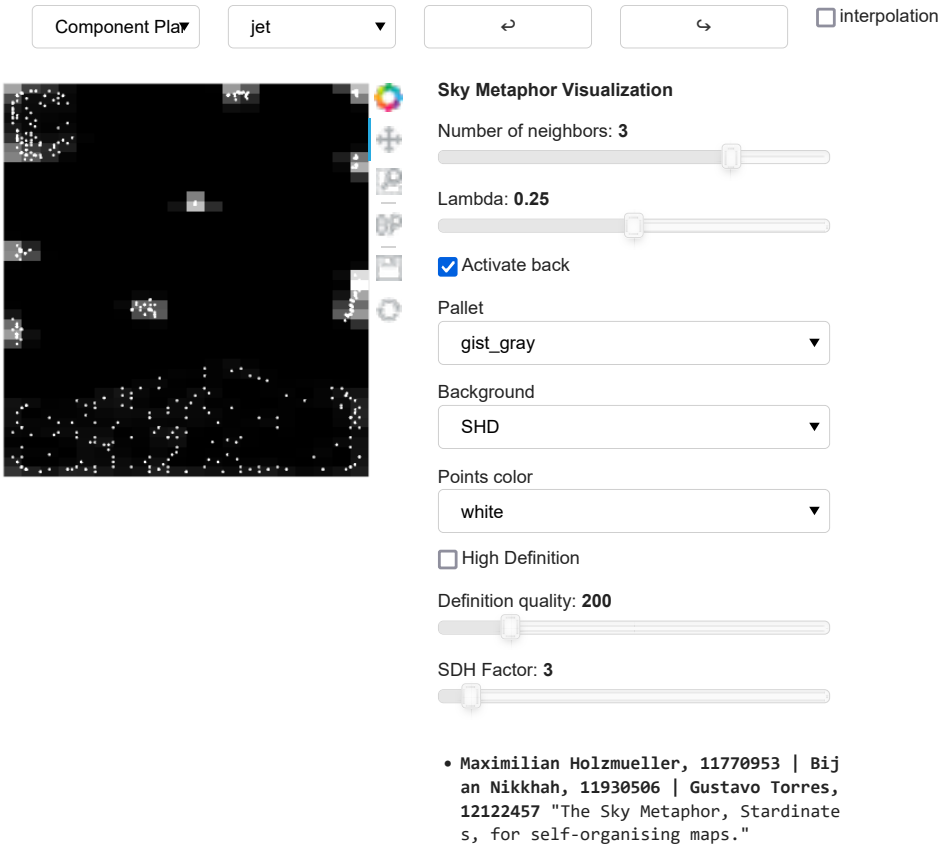
Small SOM (40 x 20)

In [9]:
```
m = 40
n = 20
dim = idata['vec_dim']

som = MiniSom(m, n, dim, sigma=10, learning_rate=0.7)
som.train(idata['arr'], 5000, True, False)

sm = SOMToolbox(weights=som._weights.reshape(-1,dim), m=m, n=n, dimension=dim, input_data=idata['arr'])
sm._mainview
```

Out[9]:



Large SOM (100 x 60)

In [10]:

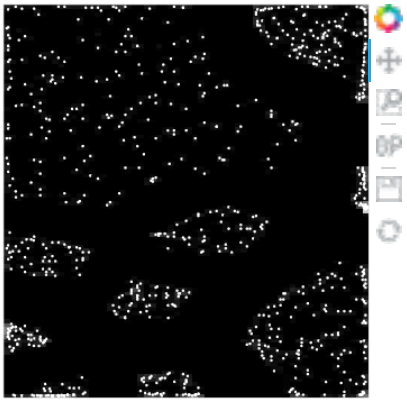
```
m = 100
n = 60
dim = idata['vec_dim']

som = MiniSom(m, n, dim, sigma=15, learning_rate=0.7)
som.train(idata['arr'], 5000, True, False)

sm = SOMToolbox(weights=som._weights.reshape(-1,dim), m=m, n=n, dimension=dim, input_data=idata['arr'])
sm._mainview
```

Out[10]:

Component Pla ▾ | jet ▾ | ↩ | ↪ | interpolation



Sky Metaphor Visualization

Number of neighbors: **3**

Lambda: **0.25**

Activate back

Pallet

gist_gray ▾

Background

SHD ▾

Points color

white ▾

High Definition

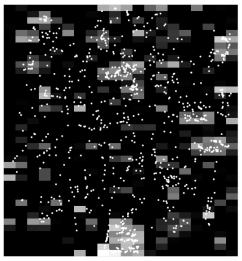
Definition quality: **200**

SDH Factor: **3**

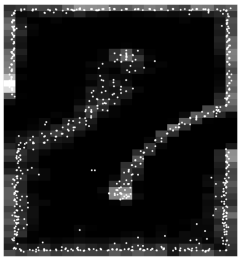
- **Maximilian Holzmueller, 11770953 | Bijan Nikkhah, 11930506 | Gustavo Torres, 12122457** "The Sky Metaphor, Stardinates, for self-organising maps."

Sigma

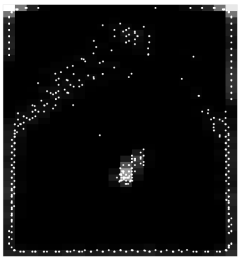
The effect of different values for *sigma* can be seen in the following figures. The other parameters are: *Learning Rate = 0.1, Iterations = 10000, Random Seed = 42*



Sigma = 1



Sigma = 10

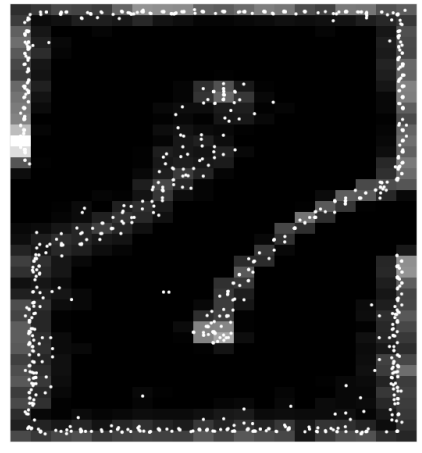


Sigma = 19

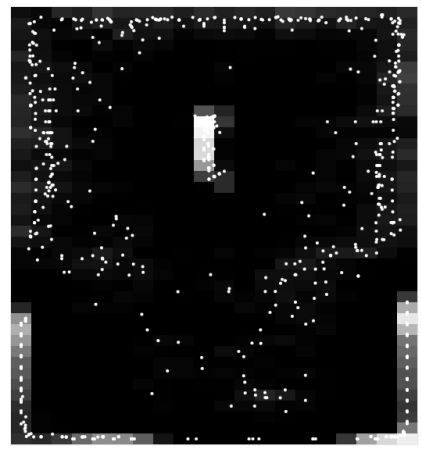
Learning Rate

The effect of different values for *learning rate* can be seen in the following figures. The other parameters are: *Sigma = 10, Iterations = 10000, Random Seed = 42*

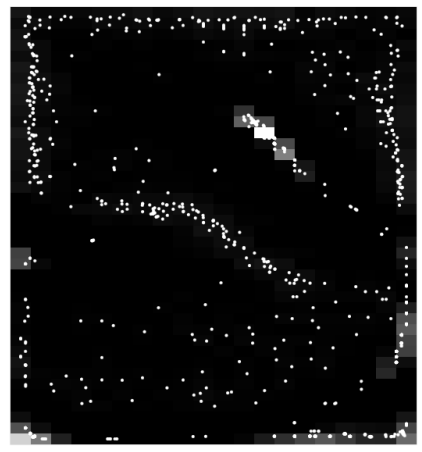
Learning Rate = 0.01



Learning Rate = 0.1



Learning Rate = 0.5



Learning Rate = 0.9

Comparison with Java SOM Toolbox

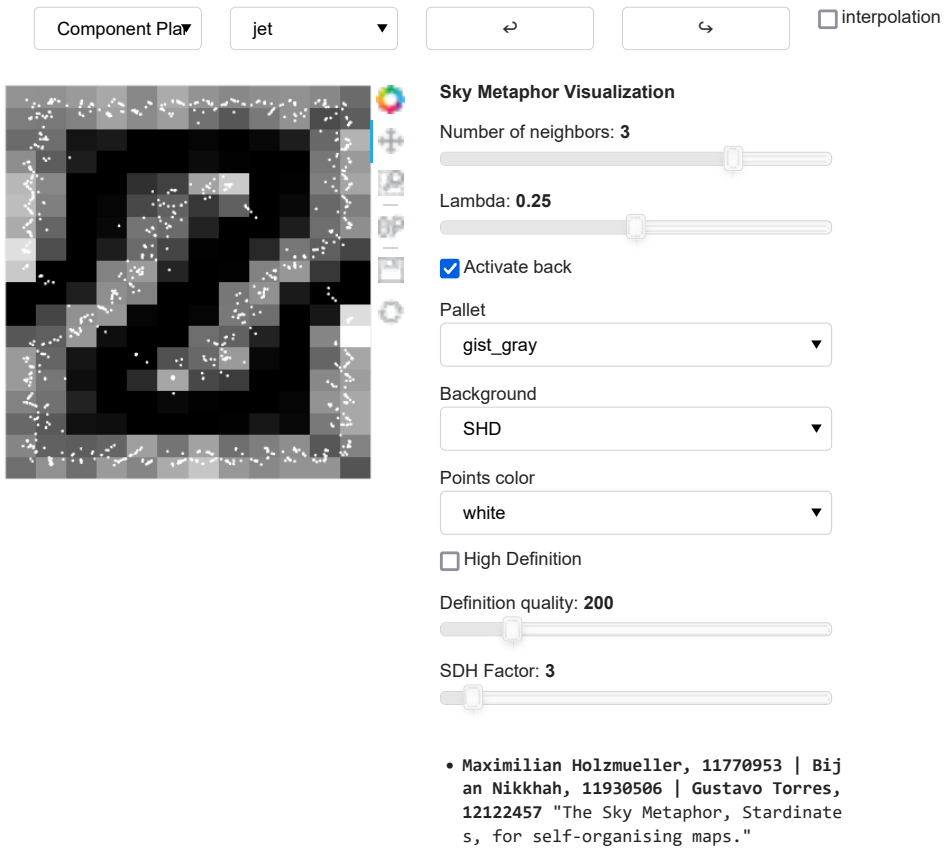
We had the task to compare our visualisation with the one from the Java SOM Toolbox. But no implementation of the Sky Metaphor could be found in the Java SOM Toolbox. We installed the newest release 0.7.5-4 svn4367 and also the "latest" build from November 2018. Additionally, we contacted the other team that got the same assignment and they ran into the same issue. Therefore we decided to not compare our visualisation with the one from the Java SOM Toolbox but use the pretrained SOM from the page of the Java SOM Toolbox and compare that with the pictures of the Chainlink and 10-clusters dataset that are part of the paper: "Sky-Metaphor Visualisation for Self-Organising Maps".

Chainlink Dataset

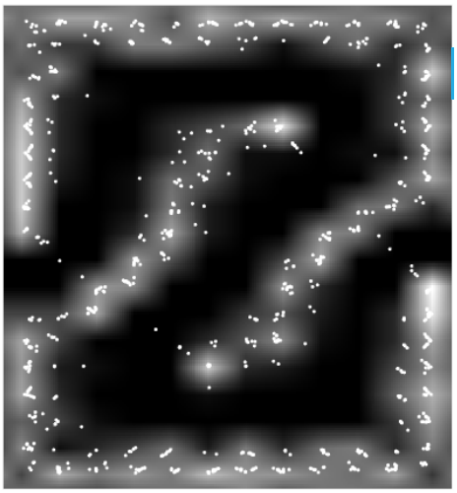
In [11]:
```
idata = SOMToolBox_Parse("datasets/chainlink/chainlink.vec").read_weight_file()
weights = SOMToolBox_Parse("datasets/chainlink/chainlink.wgt").read_weight_file()
classes = SOMToolBox_Parse("datasets/chainlink/chainlink.cls").read_weight_file()
```

In [12]:
```
sm = SOMToolbox(weights=weights['arr'],m=weights['ydim'],n=weights['xdim'],
                dimension=weights['vec_dim'], input_data=idata['arr'],
               classes=classes['arr'], component_names=classes['classes_names'])
sm._mainview
```

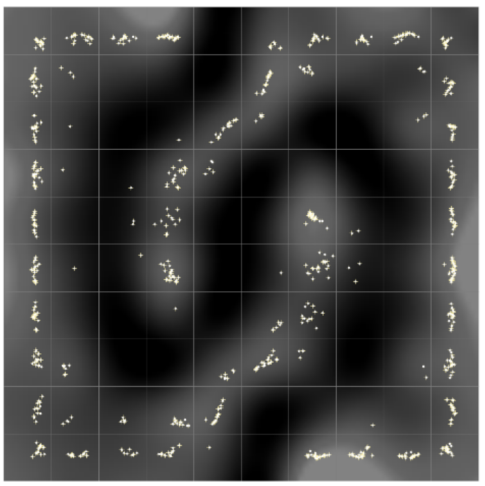
Out[12]:



Enabling High Definition, using a SDH Factor of 1, and a Lambda of 0.16 we got the following picture:



Java SOM Toolbox



Paper

Result

The most apparent differences between our visualisation and the picture of the paper are the following. Firstly, the pretrained SOM has dimensions of 12x18 whereas the SOM of the paper has dimensions of 10x10. Secondly, the rotation of the SOMs differs too. Both of those differences are not really a problem, but must be kept in mind, when analyzing both pictures.

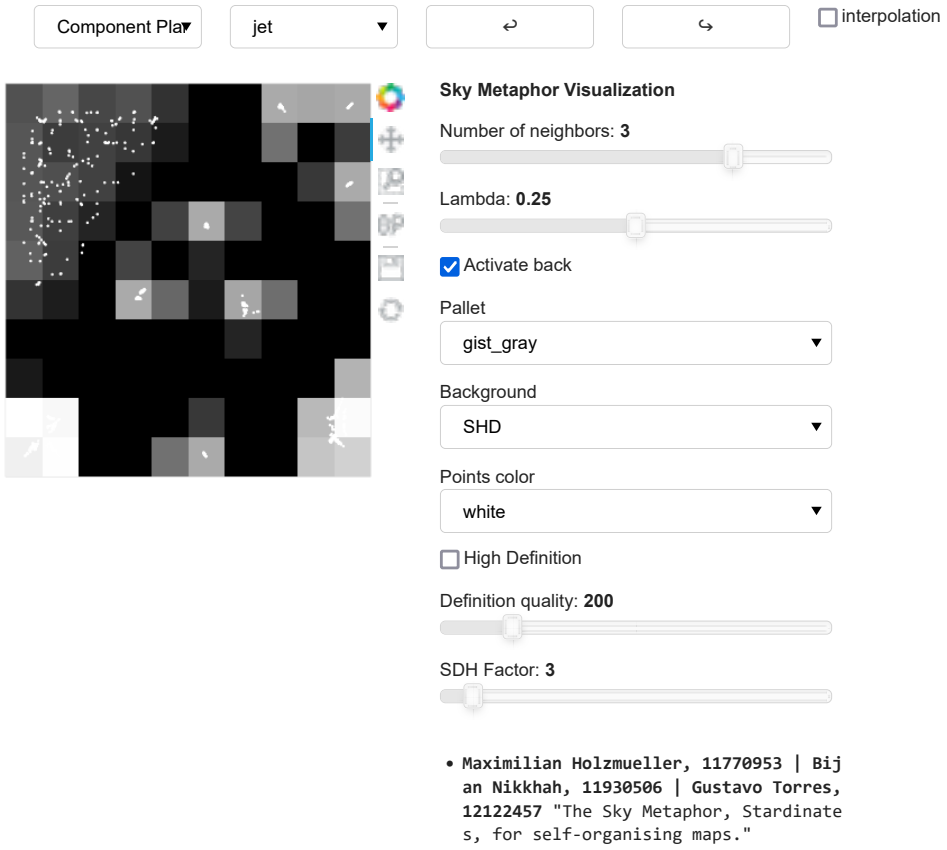
Overlooking those trivial differences, one can see that both pictures are quite similar to each other. Both show the two rings, which are "broken" at one position and cover the edge of the SOM on the other parts.

10 Clusters Dataset

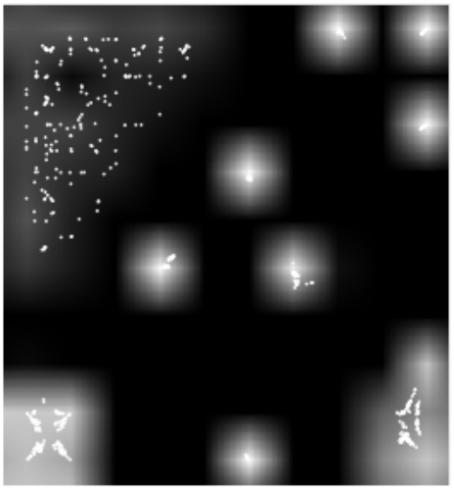
In [13]:
```
idata = SOMToolBox_Parse("datasets/10clusters/10clusters.vec").read_weight_file()
weights = SOMToolBox_Parse("datasets/10clusters/10clusters.wgt").read_weight_file()
classes = SOMToolBox_Parse("datasets/10clusters/10clusters.cls").read_weight_file()
```

In [14]:
```
sm = SOMToolbox(weights=weights['arr'],m=weights['ydim'],n=weights['xdim'],
                dimension=weights['vec_dim'], input_data=idata['arr'],
               classes=classes['arr'], component_names=classes['classes_names'])
sm._mainview
```

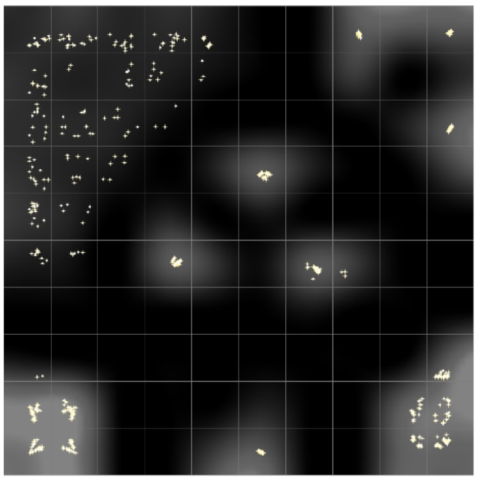
Out[14]:



Enabling High Definition, using a SDH Factor of 1, a Lambda of 0.26, and a Definition quality of 662 we got the following picture in comparison to the picture of the paper:



Java SOM Toolbox



Paper

Result

As one can see both pictures look very similar to each other. We can recognize the 10 clusters in both visualisations at the same positions. Furthermore even the structure of the points inside of one cluster seems to be very similar too (e.g. the cluster in the lower left corner contains four units and the data points are drawn from one to the others, resulting in a star like shape in both visualisations).

In [ ]: