Self Organizing Systems Exercise 3

Alexander Dobler 01631858
Thomas Kaufmann 01129115

January 31, 2021

1 Implementation

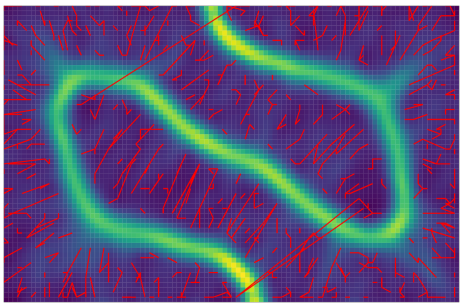
Explanation of the Implementation The implementation is described directly in the jupyter notebook. We implemented the k-NN-based and the radius-based neighborhood graphs. Both implementations can be found in the *SomViz* class under the methods *neighbourhood_knn* and *neighbourhood_radius*. The implementation is published here: https://github.com/tkauf15k/sos2020 in the directory *ex3*.

Adapting Parameters Adapting parameters is as easy as changing the parameter k for the function *neighbourhood_knn* and the parameter *radius* for *neighbourhood_radius*. We provide a section for each dataset, so one can try out different parameters for every dataset.

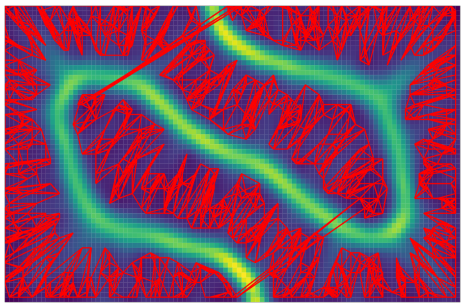
2 Evaluation

The neighborhood visualisations are projected onto the U-matrix visualisations.

Chainlink Dataset In Figure 1 and Figure 2 we can see the created visualisations for the chainlink data set for different parameters. The induced topology violations are clearly visible by the connections crossing the green interpolating aray. We can also see, how the parameter settings change the visualisations: Increasing the parameter k for the k-NN based neighborhood

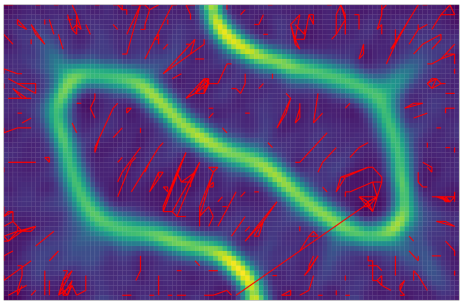


(a) $k = 1$

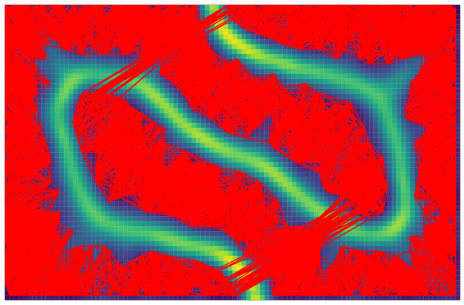


(b) $k = 5$

Figure 1: Chainlink dataset: K-NN visualisations



(a) $r = 0.05$



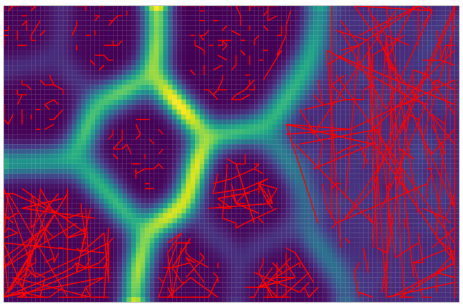
(b) $r = 0.2$

Figure 2: Chainlink dataset: Radius-based visualisations

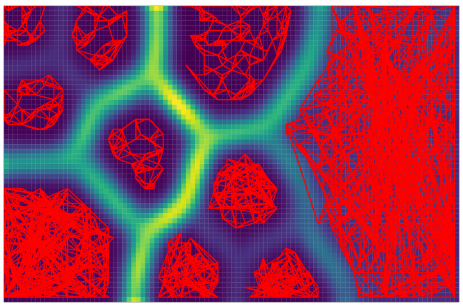
graph and the radius for the radius-based neighborhood graphs increases the number of connections between units. Setting the radius to a high value, we get somewhat of a connected red region, resembling cluster structures.

Obviously both visualisations can be used to identify density information and topology violations.

10 Clusters Dataset In Figure 3 and Figure 4 we can see the k-NN based and radius-based neighborhood graphs for the 10 clusters dataset projected onto the U-matrix visualisation. While the k-NN based neighborhood graph detects all 10 clusters, the radius-based neighborhood does not detect the clusters with big standard deviation to the right. Both visualisations clearly show density information: The k-NN based neighborhood graph detects all 10 clusters, while the radius-based neighborhood graph detects how *dense*

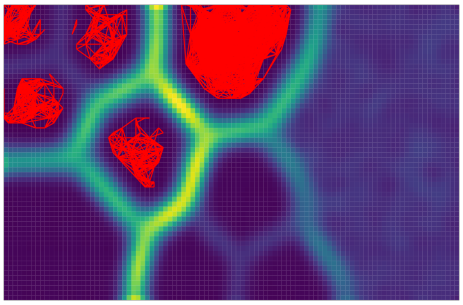


(a) $k = 1$

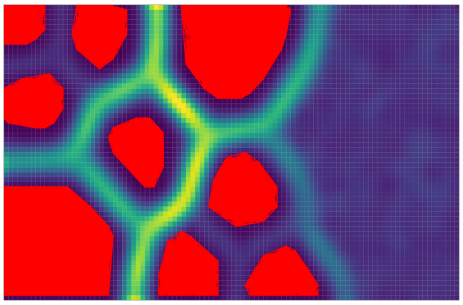


(b) $k = 5$

Figure 3: 10 clusters dataset: K-NN visualisations



(a) $r = 0.1$



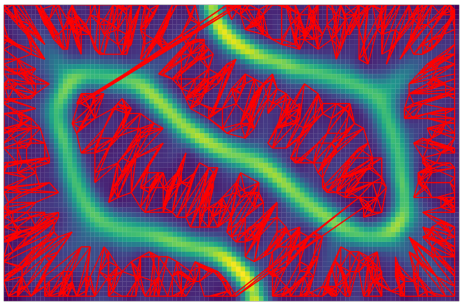
(b) $r = 1$

Figure 4: 10 clusters dataset: Radius-based visualisations

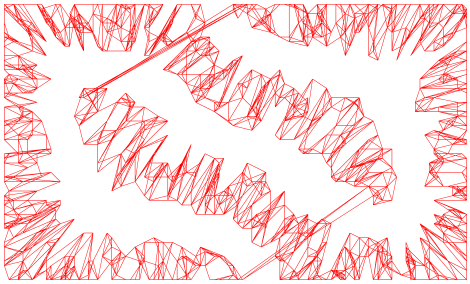
these clusters themselves are.

3 Comparison with SOM Toolbox

In Figures 5-8 we can see the comparison between visualisations generated by our implementation and the ones generated by SOM toolbox. Clearly they are identical, which means our implementation is correct.

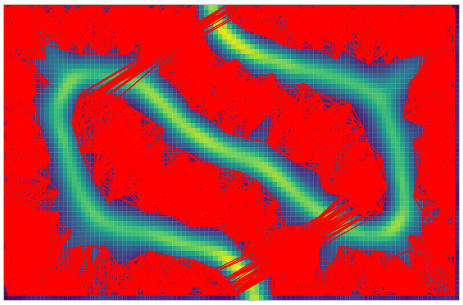


(a) Implementation

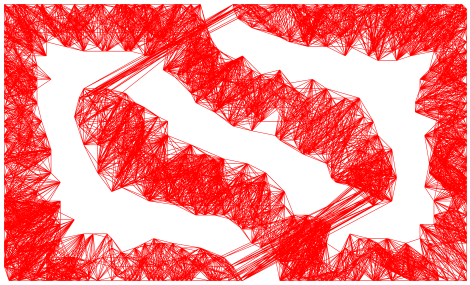


(b) SOM toolbox

Figure 5: Chainlink dataset: K-NN comparison

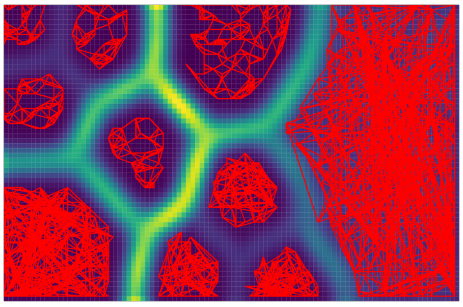


(a) Implementation

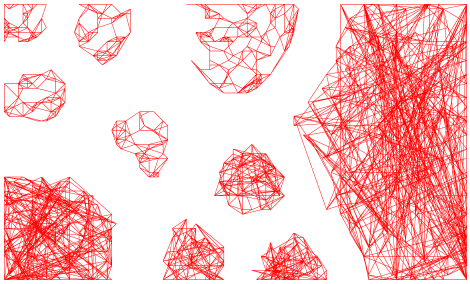


(b) SOM toolbox

Figure 6: Chainlink dataset: Radius-based comparison

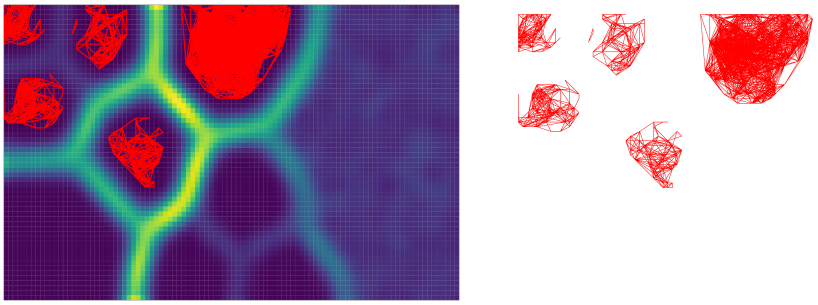


(a) Implementation



(b) SOM toolbox

Figure 7: 10 clusters dataset: K-NN comparison



(a) Implementation

(b) SOM Toolbox

Figure 8: 10 clusters dataset: Radius-based comparison

4 Conclusion

Having the weight and input vector files, generating the neighborhood graph is easy when using for example KD-trees for computing nearest neighbors. Even more, we showed that the visualizations allow to identify different characteristics of a dataset, like local densities (i.e. clusters), cardinality and density of clusters as well as topology violations.