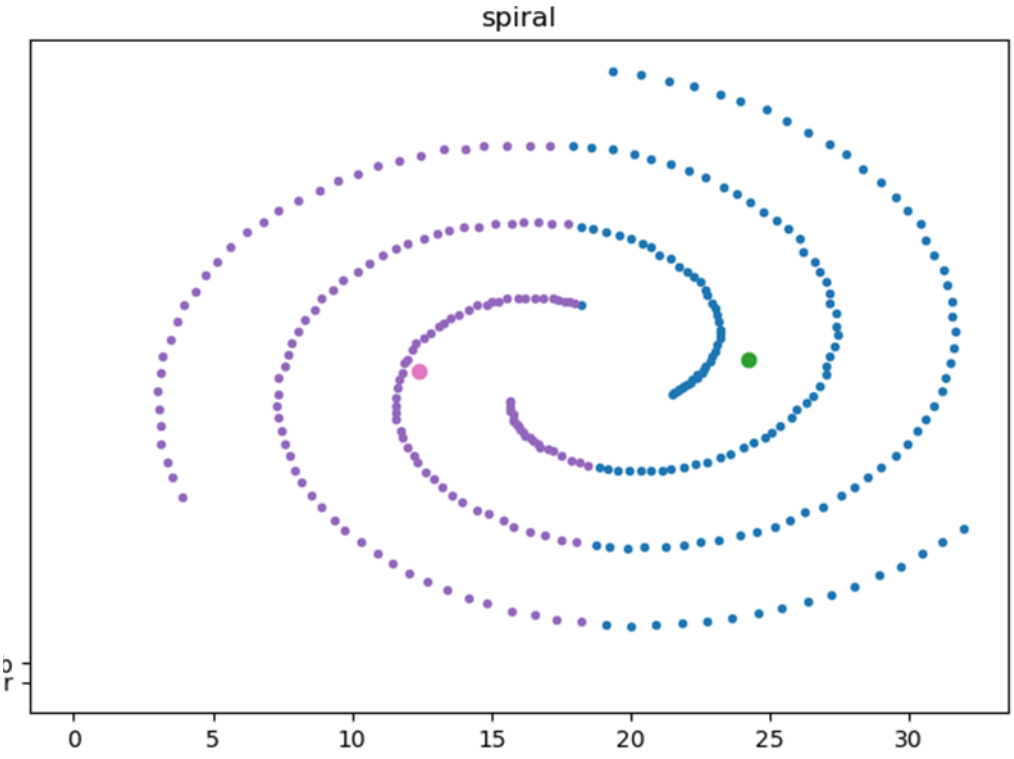
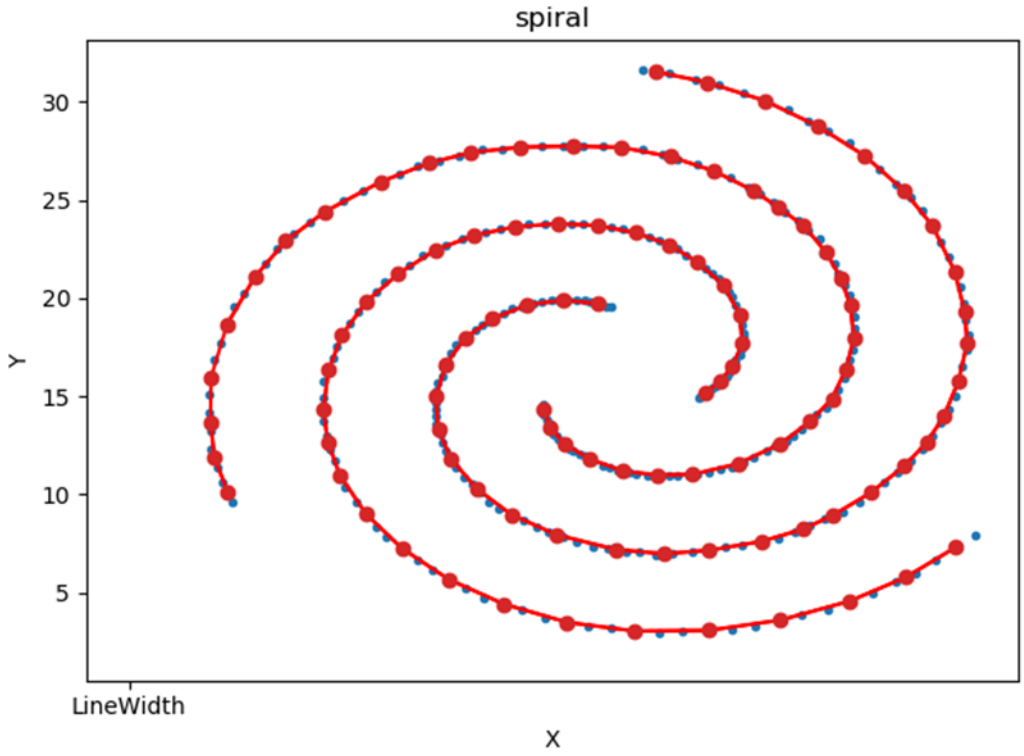
1-In this project, a neural network algorithm is implemented to solve the clustering problem in Python software. Neural networks are usually used to solve classification problems and fall under the category of supervised learning. The gas neural network algorithm is one of the competing neural networks that use the unobserved learning method to learn it, so it can be used to solve clustering, image segmentation, and so on. Its unique feature is learning the topology or shape of the dataset distribution. A neural network algorithm such as k-means algorithm is a type of vector quantization (LV) algorithm whose purpose is to reduce the data set and map it to a smaller data volume space (eg center of clusters in k-means algorithm) so that From them came the properties of the original dataset. In this project, we implement the gas network algorithm according to the reference in the docs folder in Python and run on the <spiral.mat> dataset included in the datasets folder, the results for the number of clusters 2 and 100 are as follows





As you can see in the first figure the center of two clusters (weight matrix in gas algorithm referred to as gas particles) with two solid circles and examples of each of the clusters with different colors Is. The second figure shows the topology learning feature of this algorithm when we multiply the number of gases (center of clusters) by 100 to learn the shape of the sample distribution in the dataset

Steps to implement neural network training algorithm:

1-Initialize:

We have this dataset : X={x1,x2,…,xm}

X Member R^n

N : Numbers of Neurons

Wi Member R^n

I=1,2,..,N

J=1,2,..,N

Cij=0 (means it has no link between I and j) where I,j=1,2,..,N

If Cij=1 (there is a link)

C is a Link , tij=0 where t is age(tij=0 means they just met)

2-An input named x is selected from the input data(Randomly,sorted,..)

3-Ranking:

Calculate the median distance x and the centers Wi

di is the distance

then Calculate Ki rank for each center (0<=Ki<=N-1)

4-Adaptation:

Wi(new)=Wi(old)+epsilon*exp^-Ki/lambda*(x-Wi(old))

For each i=1,2,..,N

5-Create a neighbor:

If there is no edge between the first-order neuron and the second-order neuron, go create

If there is nothing else

If Cij=0 → Cij=1

Id Cij=1 → Cij=1

Cij=cji=1 (Symmetrical)

6-Aging:

Assume Ki=0

The age of all edges increases

Only the edges that line the first-order neurons

Kij←tij + 1

7-Remove Old Links:

Let ki = 0 and i be the first rank

If there is a link between the neurons i and j for each i and j

Cij = 0 ← tij> T The relationship must be disconnected

Only the neurons that form each other remain their edges

8-If the termination conditions are not met, it will repeat from step one, otherwise the end

(This process is a recursive process)

*Parameters Definition for neural gas network:

Lambda is the amount of drop

Epsilon is the Maximum amount of learning

T is the amount of hardening

Lambda must be Decrease from Learning mode is equal to all neurons in the state WTA (Winner Takes All)

T must be increase (To make the edges longer)

Note:

Lambda_initial > Lambda_final

Epsilon_initial > epsilon_final

T_initial < T_final

*Calculate Parameteres:

Lambda=lambda_initial*(lambda_final/lambda_initial)^tt/tmax

Epsilon=epsilon_initial*(epsilon_final/epsilon_initial)^tt/tmax

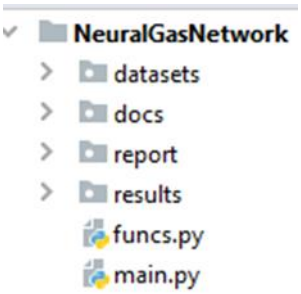
Lambda=T_initial*(T_final/T_initial)^tt/tmax

Where tt is a variable increases one unit time after pass the Adaptation step completely

Description of project code and files:

Content of the project

This project as shown below



It consists of four folders and two files. The folders are as follows: Datasets, References to the Neural Network Algorithm, Report File and Results of the Algorithm Implementation including Images. The main.py file is the executable of the project and it is used to execute the project and in the funcs.py file all the necessary functions such as plots and neural network algorithm are implemented.

To run the project, first install the necessary libraries like numpy, scipy, matplotlib, etc. and run the main.py file to save the results to the results folder after the execution.

The main steps of the project are as follows (main.py)

• Import the necessary libraries and functions written in the funcs.py file

```
from scipy.io import loadmat
from funcs import *
```

• Loading data

```
# ----- load dataset ------
dataset_path = 'datasets'
filename = 'spiral'
dataset = loadmat(dataset_path + '/' + filename+'.mat')

# ----- extract features from dataset
X = dataset['X']  # features
```

• Adjustment and Parameterization of the Neural Network Algorithm

```
# ---- initial NeuralGasNetwork parameters ----
NGN_params = {'N':100, #number clusters
              'MaxIt':100,
              'tmax':8000,
              'epsilon_initial':0.9,
              'epsilon_final':0.4,
              'lambda_initial':10,
              'lambda_final':1,
              'T_initial':5,
              'T_final':10,
              }
```

• Function call of the neural network algorithm for clustering

```
# ---- clustering --------
w,C = NeuralGasNetwork(X, NGN_params)
```

- Draw shapes and save them to the results folder

```
# ---- plot and save -----
fig1 = plt.figure()
PlotResults(X, w, C)
plt.title(filename)
plt.show()
figName1 = 'results/'+filename+'_1.png'
fig1.savefig(figName1)

if NGN_params['N']<=3:

    fig2 = plt.figure()

    PlotClustring(X,w)
    plt.title(filename)
    plt.show()
    figName2 = 'results/'+filename+'_2.png'
    fig2.savefig(figName2)
```

The dataset can be downloaded from the link below:

http://cs.joensuu.fi/sipu/datasets/