Self_organizing_map

January 21, 2020

1 Self Organizing Maps (SOM)

1.1 Unsupervised Clustering Algorithm

1.1.1 Algorithm Class: Competitive Learning

See also: (Kohonen 1990) : The Self-Organizing Map Proceedings of the IEEE, Vol.78, No. 9, September 1990 The Iris flower data set or Fisher's Iris data set is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his 1936 paper "The use of multiple measurements in taxonomic problems as an example of linear discriminant analysis".

The data set consists of 50 samples from each of three species of Iris (Iris setosa, Iris virginica and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals (Kelchblatt) and petals (Blütenblatt), in centimeters.

1.1.2 Iris Setosa

Iris Versicolor ### Iris Virginica

[1]:
```
#Load toy dataset (iris) from scikit learn library

from sklearn import datasets
from sklearn.utils import shuffle
import pandas as pd
import numpy as np

#Load data in matrix X for easier reference
iris = datasets.load_iris()
X = iris.data

#Normalize Data to be in range [0,1], alternative to standard normally␣
↪distributed data, i.e. mu = 0, sigma = 1
#If a feature has a variance that is orders of magnitude larger than others,
#it might dominate the objective function and make the estimator unable to␣
↪learn
#from other features correctly.
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X)
```

```python
X = scaler.transform(X)

#Create Pandas Dataframe for table display
df = pd.DataFrame(X)
df.columns = iris.feature_names
df['class'] = iris.target

#label class names
df['class_name'] = [iris.target_names[i] for i in df['class']]

#shuffle data
df = shuffle(df)
X = np.array(df.loc[:,'sepal length (cm)':'petal width (cm)'])


#Visualize data in table
df
```

[1]:

```
     sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
33            0.333333          0.916667           0.067797          0.041667
143           0.694444          0.500000           0.830508          0.916667
59            0.250000          0.291667           0.491525          0.541667
118           0.944444          0.250000           1.000000          0.916667
57            0.166667          0.166667           0.389831          0.375000
64            0.361111          0.375000           0.440678          0.500000
121           0.361111          0.333333           0.661017          0.791667
60            0.194444          0.000000           0.423729          0.375000
20            0.305556          0.583333           0.118644          0.041667
80            0.333333          0.166667           0.474576          0.416667
85            0.472222          0.583333           0.593220          0.625000
43            0.194444          0.625000           0.101695          0.208333
81            0.333333          0.166667           0.457627          0.375000
54            0.611111          0.333333           0.610169          0.583333
69            0.361111          0.208333           0.491525          0.416667
106           0.166667          0.208333           0.593220          0.666667
141           0.722222          0.458333           0.694915          0.916667
77            0.666667          0.416667           0.677966          0.666667
37            0.166667          0.666667           0.067797          0.000000
36            0.333333          0.625000           0.050847          0.041667
62            0.472222          0.083333           0.508475          0.375000
134           0.500000          0.250000           0.779661          0.541667
93            0.194444          0.125000           0.389831          0.375000
7             0.194444          0.583333           0.084746          0.041667
129           0.805556          0.416667           0.813559          0.625000
17            0.222222          0.625000           0.067797          0.083333
28            0.250000          0.583333           0.067797          0.041667
116           0.611111          0.416667           0.762712          0.708333
```

45	0.138889	0.416667	0.067797	0.083333
117	0.944444	0.750000	0.966102	0.875000
..	...	...	...	...
86	0.666667	0.458333	0.627119	0.583333
56	0.555556	0.541667	0.627119	0.625000
70	0.444444	0.500000	0.644068	0.708333
47	0.083333	0.500000	0.067797	0.041667
63	0.500000	0.375000	0.627119	0.541667
113	0.388889	0.208333	0.677966	0.791667
107	0.833333	0.375000	0.898305	0.708333
67	0.416667	0.291667	0.525424	0.375000
112	0.694444	0.416667	0.762712	0.833333
110	0.611111	0.500000	0.694915	0.791667
91	0.500000	0.416667	0.610169	0.541667
4	0.194444	0.666667	0.067797	0.041667
119	0.472222	0.083333	0.677966	0.583333
24	0.138889	0.583333	0.152542	0.041667
65	0.666667	0.458333	0.576271	0.541667
127	0.500000	0.416667	0.661017	0.708333
25	0.194444	0.416667	0.101695	0.041667
30	0.138889	0.458333	0.101695	0.041667
79	0.388889	0.250000	0.423729	0.375000
140	0.666667	0.458333	0.779661	0.958333
23	0.222222	0.541667	0.118644	0.166667
87	0.555556	0.125000	0.576271	0.500000
89	0.333333	0.208333	0.508475	0.500000
123	0.555556	0.291667	0.661017	0.708333
108	0.666667	0.208333	0.813559	0.708333
137	0.583333	0.458333	0.762712	0.708333
76	0.694444	0.333333	0.644068	0.541667
120	0.722222	0.500000	0.796610	0.916667
31	0.305556	0.583333	0.084746	0.125000
51	0.583333	0.500000	0.593220	0.583333

	class	class_name
33	0	setosa
143	2	virginica
59	1	versicolor
118	2	virginica
57	1	versicolor
64	1	versicolor
121	2	virginica
60	1	versicolor
20	0	setosa
80	1	versicolor
85	1	versicolor
43	0	setosa

```
81       1  versicolor
54       1  versicolor
69       1  versicolor
106      2   virginica
141      2   virginica
77       1  versicolor
37       0      setosa
36       0      setosa
62       1  versicolor
134      2   virginica
93       1  versicolor
7        0      setosa
129      2   virginica
17       0      setosa
28       0      setosa
116      2   virginica
45       0      setosa
117      2   virginica
..     ...         ...
86       1  versicolor
56       1  versicolor
70       1  versicolor
47       0      setosa
63       1  versicolor
113      2   virginica
107      2   virginica
67       1  versicolor
112      2   virginica
110      2   virginica
91       1  versicolor
4        0      setosa
119      2   virginica
24       0      setosa
65       1  versicolor
127      2   virginica
25       0      setosa
30       0      setosa
79       1  versicolor
140      2   virginica
23       0      setosa
87       1  versicolor
89       1  versicolor
123      2   virginica
108      2   virginica
137      2   virginica
76       1  versicolor
120      2   virginica
```

```
31        0      setosa
51        1  versicolor

[150 rows x 6 columns]
```

1.1.3 Little toolbox to explore SOM is delievered in the minisom package (use pip install minisom for quick download with Anaconda Distribution)

https://github.com/JustGlowing/minisom/blob/master/minisom.py

1.2 The self-organizing map algorithm

In general one can use SOM for two purposes:

1. Clustering of data, i.e. understand how many clusters might be in a specific data set

2. Dimensionality reduction, i.e. find out which features are truly relevant or can be neglected

1.2.1 The Algorithm:

- A number of neurons are aligned within a grid for instance rectangular or hexagonal
- Each neuron is assigned to a weight vector that has the same dimensionality as the training data
- Each node's weights are initialized, randomly.
- A vector is chosen at random from the set of training data.
- Every node (weight vector) is examined to calculate which one's weights are most like the input vector.
  - Here similiarity is represented as the euclidean distance between the node and the input vector
- The winning node is commonly known as the Best Matching Unit (BMU).
- Then the neighbourhood of the BMU is calculated. The amount of neighbors decreases over time.
- The winning weight is rewarded with becoming more like the sample vector. The neighbors also become more like the sample vector.
- The closer a node is to the BMU, the more its weights get altered and the farther away the neighbor is from the BMU, the less it learns.
- Repeat from step 2 for a number of iterations.
- Determin distances between neurons –> U-Matrix Unified Distance Matrix

1.2.2 Typical decisions to make when using SOM:

Learning rate: How close shall the BMU and its neighbours be pulled towards the data point

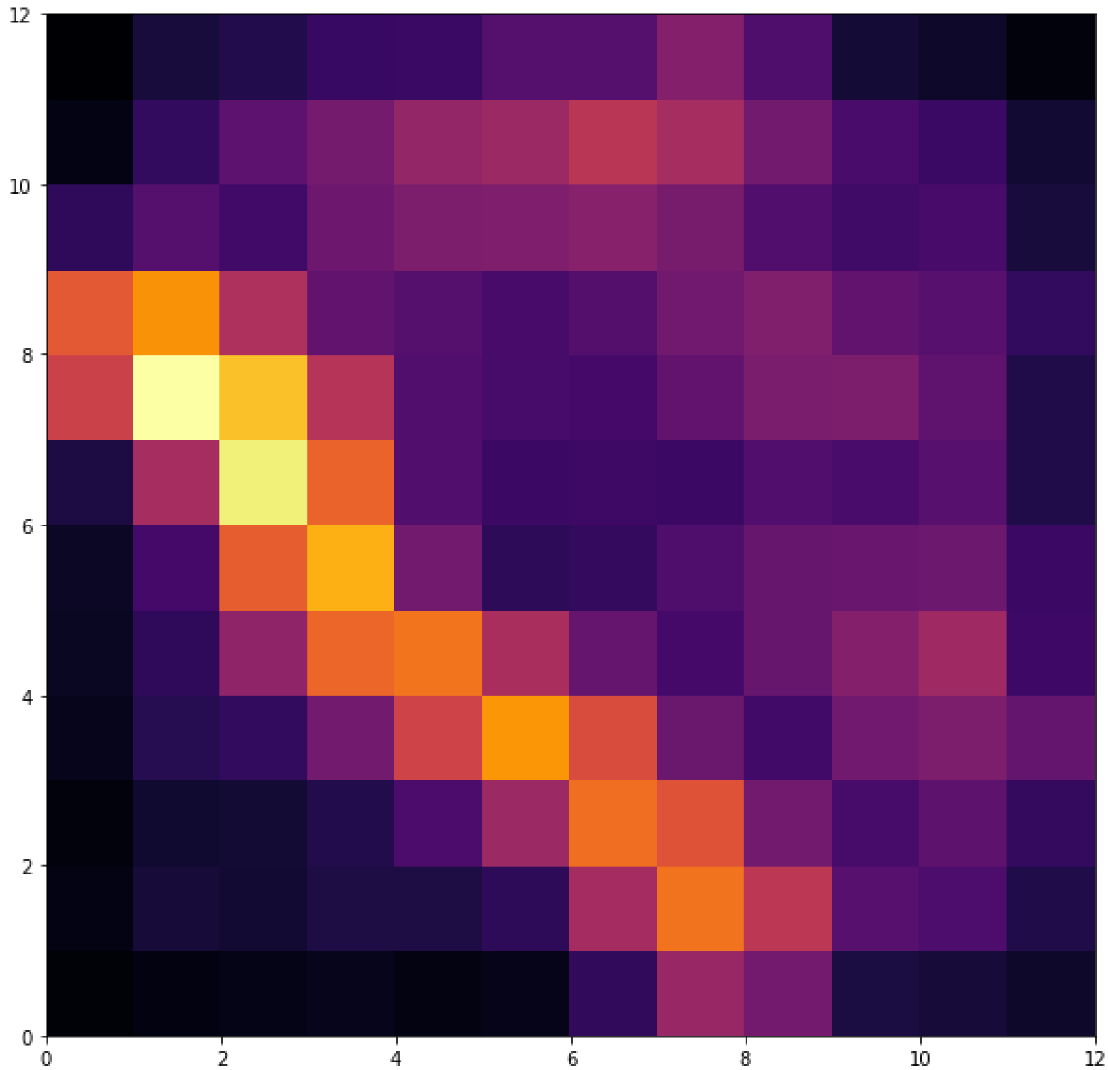
Neighbours: How many neighbours shall be affected from the BMU. Usually in the beginning the neighbourhood should be large and decrease monotonically over time

Number of Neurons

[2]:
```
from minisom import MiniSom

#Create object of class Minisom
#__init__(self, x, y, input_len, sigma=1.0, learning_rate=0.5,␣
↪decay_function=asymptotic_decay, neighborhood_function='gaussian',␣
↪random_seed=None)
#Rule of Thumb for choosing number of neurons. 5*srt(N)
som = MiniSom(12,12, 4, neighborhood_function = 'gaussian', sigma=1,␣
↪learning_rate= 0.5, random_seed = 10)

#Initializes the weights of the SOM
som.random_weights_init(data = X)

#Trains the SOM using all the vectors in data sequentially
som.train_batch(data = X, num_iteration = 10000, verbose = True)
```

```
[ 10000 / 10000 ] 100% - 0:00:00 left
quantization error: 0.04388275012315818
topographic error: 0.8666666666666667
```

[3]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.figure(figsize=(10,10))
plt.pcolor(som.distance_map(), cmap='inferno')
plt.show()
```



[4]:
```
neuron_weights = som.get_weights()
neuron_weights
```

[4]:
```
array([[[0.11575375, 0.66449164, 0.0220639 , 0.03724871],
        [0.14690438, 0.61978403, 0.07358308, 0.04853585],
        [0.14194492, 0.58313063, 0.11345687, 0.04730927],
        [0.11430388, 0.55050612, 0.08009162, 0.05679225],
        [0.10512983, 0.49831978, 0.0759001 , 0.04054646],
        [0.07279719, 0.45907515, 0.06790624, 0.03604977],
        [0.03372504, 0.3900913 , 0.05220335, 0.03688255],
        [0.04550598, 0.24331246, 0.05252024, 0.06421956],
        [0.14891086, 0.17302889, 0.25764912, 0.28796631],
        [0.19741773, 0.18250347, 0.36444621, 0.39526228],
        [0.19623231, 0.12757429, 0.38683182, 0.38227937],
        [0.19497866, 0.02998433, 0.41633878, 0.37606026]],
```

```
[[0.17454707, 0.65934994, 0.05922908, 0.03084316],
 [0.19848697, 0.62560749, 0.06424363, 0.05956384],
 [0.19443247, 0.58639618, 0.08292778, 0.04816405],
 [0.17185845, 0.53870691, 0.07473633, 0.04095875],
 [0.14663691, 0.4753136 , 0.07735759, 0.02356548],
 [0.13164948, 0.44051413, 0.07794649, 0.02792211],
 [0.10359574, 0.40742417, 0.07180432, 0.05648917],
 [0.1661834 , 0.28591613, 0.17031472, 0.16070253],
 [0.27271776, 0.18150552, 0.3813191 , 0.34626381],
 [0.25738542, 0.17391083, 0.41722765, 0.41145766],
 [0.27015042, 0.15565657, 0.45423654, 0.44741045],
 [0.2400297 , 0.14548782, 0.46064485, 0.45772466]],

[[0.23787155, 0.65951858, 0.07406739, 0.04398252],
 [0.23738271, 0.62247207, 0.07320888, 0.05455846],
 [0.238054  , 0.59122247, 0.07964783, 0.05205663],
 [0.21577965, 0.55579082, 0.08177416, 0.07199851],
 [0.18207098, 0.47328809, 0.06594154, 0.04170422],
 [0.17068049, 0.43223263, 0.08221602, 0.03962717],
 [0.22848153, 0.38008522, 0.18870102, 0.14909174],
 [0.37889349, 0.25289089, 0.44329908, 0.38280575],
 [0.35513271, 0.18798791, 0.46671027, 0.39086806],
 [0.35244354, 0.15597558, 0.48532478, 0.42606712],
 [0.32560995, 0.17158435, 0.50582735, 0.49667637],
 [0.26101331, 0.24089293, 0.51169317, 0.54818601]],

[[0.27391175, 0.70796122, 0.08954277, 0.05469279],
 [0.27830911, 0.64748124, 0.07642099, 0.06299765],
 [0.26994589, 0.59249806, 0.09378114, 0.08742671],
 [0.22433507, 0.56945711, 0.10505438, 0.12748592],
 [0.21223579, 0.50852048, 0.11640203, 0.12948059],
 [0.2496654 , 0.40342092, 0.23499988, 0.21254823],
 [0.37087538, 0.31233908, 0.48265799, 0.4225598 ],
 [0.40148626, 0.26776878, 0.50996138, 0.43241426],
 [0.39024557, 0.20856318, 0.49811032, 0.41562088],
 [0.44221139, 0.11916619, 0.51605985, 0.41808957],
 [0.38391072, 0.13813044, 0.55682259, 0.52764505],
 [0.22359228, 0.20563372, 0.57402523, 0.62607925]],

[[0.23549151, 0.74138732, 0.10892725, 0.08720154],
 [0.24864121, 0.71638933, 0.09145621, 0.11140644],
 [0.23439446, 0.6357928 , 0.0975969 , 0.15492151],
 [0.24115824, 0.56097288, 0.16703901, 0.20013052],
 [0.34708243, 0.42391045, 0.44829297, 0.4362184 ],
 [0.37479408, 0.35655929, 0.49826141, 0.49587057],
 [0.38765158, 0.32111378, 0.54983209, 0.49490409],
```

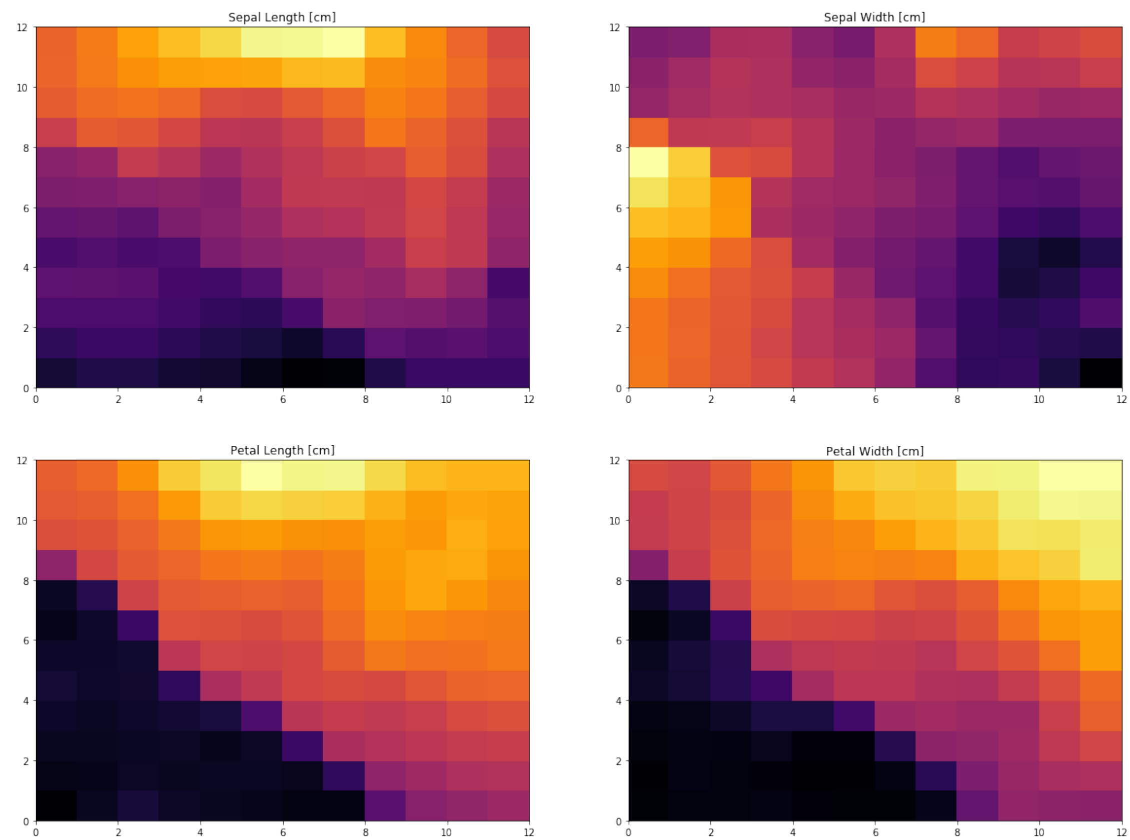
```
 [0.38560878, 0.2853665 , 0.55776933, 0.46467153],
 [0.43133797, 0.21365401, 0.55316378, 0.45913145],
 [0.52510528, 0.12549109, 0.58896428, 0.51260901],
 [0.49794447, 0.0972747 , 0.62275997, 0.57659507],
 [0.39017264, 0.14775238, 0.63294499, 0.65265247]],

[[0.28769504, 0.80017737, 0.08867761, 0.07826968],
 [0.29974952, 0.78087168, 0.09061878, 0.11640301],
 [0.27752656, 0.73071351, 0.10011871, 0.14861323],
 [0.34800125, 0.44644849, 0.48856868, 0.46115437],
 [0.37030653, 0.41321277, 0.54030883, 0.49855793],
 [0.40207659, 0.3826611 , 0.53334396, 0.50863241],
 [0.45522667, 0.34361211, 0.54816017, 0.50098484],
 [0.47192549, 0.32718576, 0.60563913, 0.48415729],
 [0.50213517, 0.26821213, 0.67034189, 0.55043737],
 [0.54680178, 0.20544669, 0.65474559, 0.59446412],
 [0.49531454, 0.18191171, 0.65759573, 0.66497005],
 [0.40590586, 0.2377499 , 0.67662561, 0.76412602]],

[[0.3495326 , 0.86664399, 0.06530127, 0.0437504 ],
 [0.35440301, 0.80181103, 0.09253395, 0.07863971],
 [0.37243905, 0.72811545, 0.19364248, 0.1922837 ],
 [0.38088563, 0.46661301, 0.57781775, 0.57279669],
 [0.35848846, 0.4192573 , 0.57563409, 0.56196489],
 [0.43040874, 0.40646855, 0.55995751, 0.55842733],
 [0.49834912, 0.38650775, 0.58321504, 0.5337281 ],
 [0.50436418, 0.34759338, 0.64560568, 0.54312733],
 [0.49862231, 0.28163331, 0.71592084, 0.59145447],
 [0.55147678, 0.25806515, 0.69778112, 0.67107028],
 [0.51128652, 0.25164319, 0.68604993, 0.74472694],
 [0.41286961, 0.29231024, 0.68389128, 0.76408673]],

[[0.36963726, 0.9349447 , 0.07805497, 0.08566203],
 [0.39653645, 0.82798034, 0.14686123, 0.13850289],
 [0.51084158, 0.57141113, 0.53234286, 0.5364919 ],
 [0.47677413, 0.55502365, 0.60019157, 0.62266475],
 [0.42000675, 0.4680544 , 0.61283635, 0.63394191],
 [0.46250087, 0.40760159, 0.62209885, 0.64395423],
 [0.49668594, 0.3727979 , 0.61540704, 0.59606168],
 [0.53093383, 0.34376901, 0.66824574, 0.58066162],
 [0.5461486 , 0.28214622, 0.7315569 , 0.61779028],
 [0.61626849, 0.24613088, 0.7685349 , 0.71854794],
 [0.56810348, 0.28652742, 0.73670223, 0.77704368],
 [0.45883052, 0.30535047, 0.70267155, 0.80322699]],

[[0.52223997, 0.62731611, 0.37977771, 0.36205986],
 [0.60988814, 0.49085466, 0.54670075, 0.52097802],
```

```
 [0.59397507, 0.49565004, 0.60357023, 0.5901396 ],
 [0.55323846, 0.51438284, 0.63188359, 0.63719235],
 [0.49096275, 0.46509158, 0.66905717, 0.69764401],
 [0.48847615, 0.41025565, 0.67881831, 0.70653939],
 [0.51941594, 0.3728253 , 0.65886202, 0.70135003],
 [0.57714083, 0.39371698, 0.68376599, 0.70868126],
 [0.67151395, 0.40803818, 0.74385327, 0.79906276],
 [0.62838025, 0.34024939, 0.76744036, 0.83398826],
 [0.57708989, 0.33638197, 0.77177007, 0.85632483],
 [0.48561146, 0.33555343, 0.72872717, 0.91208655]],

[[0.60954643, 0.39388865, 0.56971839, 0.51156737],
 [0.64812292, 0.43348223, 0.58258191, 0.54232065],
 [0.66487933, 0.45862561, 0.62265123, 0.58146727],
 [0.64017385, 0.44811583, 0.67224916, 0.65055982],
 [0.57158163, 0.43952928, 0.73325274, 0.70163672],
 [0.56116142, 0.40313328, 0.74451326, 0.71531978],
 [0.60384556, 0.41059472, 0.72560145, 0.76167763],
 [0.64240378, 0.46786537, 0.71919757, 0.8014339 ],
 [0.69432583, 0.44997556, 0.74606065, 0.84042601],
 [0.66931836, 0.42392083, 0.73684991, 0.89150856],
 [0.61379649, 0.40205212, 0.78059178, 0.88730762],
 [0.55571119, 0.41250465, 0.75353029, 0.9105351 ]],

[[0.62951737, 0.37251652, 0.60077886, 0.51842942],
 [0.67790092, 0.41976105, 0.61462808, 0.54496431],
 [0.72520049, 0.46729462, 0.6517012 , 0.5731754 ],
 [0.74858239, 0.44930728, 0.73973999, 0.63839911],
 [0.7572221 , 0.38965699, 0.83495055, 0.72523461],
 [0.76376155, 0.37162277, 0.85619844, 0.78761227],
 [0.79944646, 0.43073989, 0.84401042, 0.82986188],
 [0.80301131, 0.5631624 , 0.83808084, 0.83464216],
 [0.71632679, 0.52625166, 0.7865503 , 0.86614543],
 [0.70093994, 0.47163956, 0.74482596, 0.91177088],
 [0.64791874, 0.47556466, 0.77044732, 0.93858697],
 [0.58276722, 0.51581077, 0.7580228 , 0.93519028]],

[[0.62847335, 0.33994043, 0.61534754, 0.56428232],
 [0.68053833, 0.3499335 , 0.63728347, 0.55086365],
 [0.75665473, 0.44253023, 0.72315715, 0.60437729],
 [0.80976917, 0.44469879, 0.83261123, 0.67745261],
 [0.85406752, 0.36295206, 0.88504512, 0.73922213],
 [0.92053489, 0.32711701, 0.94830273, 0.84017888],
 [0.92606965, 0.45381722, 0.91686605, 0.85831817],
 [0.94667615, 0.67639436, 0.92230582, 0.84713025],
 [0.80955752, 0.63028543, 0.86013028, 0.92246329],
 [0.71086529, 0.510312  , 0.8037817 , 0.92704574],
```

```
[0.63511404, 0.53149496, 0.79264859, 0.96240585],
[0.55852234, 0.55830635, 0.79108005, 0.9609769 ]]])
```

[5]:
```
#Slice data matrix in four distince matrices to visualize individual feature␣
↪weight layers
sepal_length = neuron_weights[:,:,0]
sepal_width = neuron_weights[:,:,1]
petal_length = neuron_weights[:,:,2]
petal_width = neuron_weights[:,:,3]
```

[6]:
```
fig,((ax0, ax1), (ax2, ax3)) = plt.subplots(2,2, figsize = (20,15))
ax0.pcolor(sepal_length, cmap='inferno')
ax0.set_title('Sepal Length [cm]')
ax1.pcolor(sepal_width , cmap='inferno')
ax1.set_title('Sepal Width [cm]')
ax2.pcolor(petal_length, cmap='inferno')
ax2.set_title('Petal Length [cm]')
ax3.pcolor(petal_width, cmap='inferno')
ax3.set_title('Petal Width [cm]')
plt.show()
```



Petal Length and Petal Width are rather similar in terms of the magnitude of their weights - Petal Length or Petal Width can be removed from the data set since they are highly correlated