INT420: Big Data Analytics

INT491: Introduction to Data Analytics and Big Data

06: **Unsupervised Learning**
Self Organizing Map: SOM
& Cluster Evaluation

Niwan Wattanakitrungroj

School of Information Technology,
King Mongkut's University of Technology Thonburi

Semester 1 - 2022

Overview

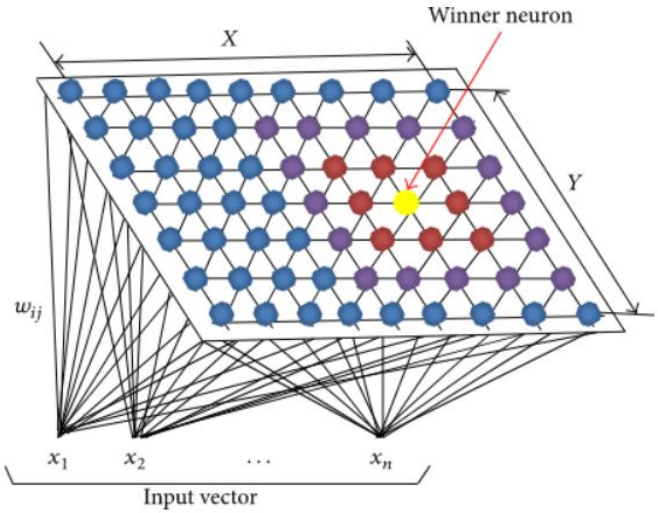
1. SOM:Self Organizing Map
2. BMU: Best-Matching Unit
3. BMU Neighborhood
4. Updating
5. Summary of the SOM Algorithm
6. Cluster Evaluation

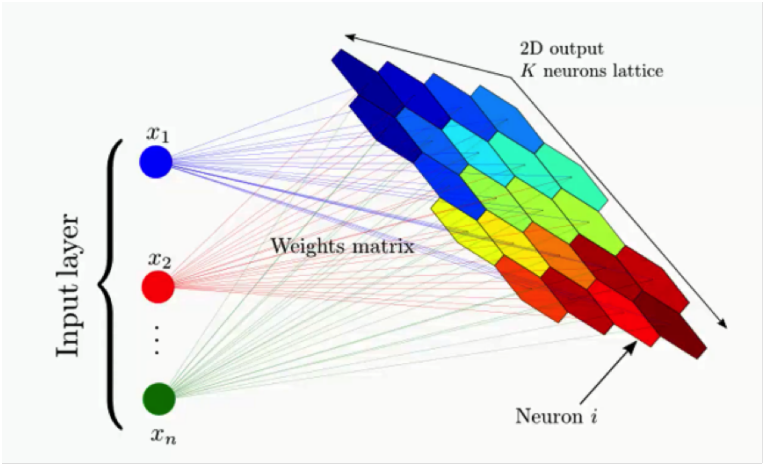
Outline

1. SOM:Self Organizing Map
2. BMU: Best-Matching Unit
3. BMU Neighborhood
4. Updating
5. Summary of the SOM Algorithm
6. Cluster Evaluation

SOM:Self Organizing Map

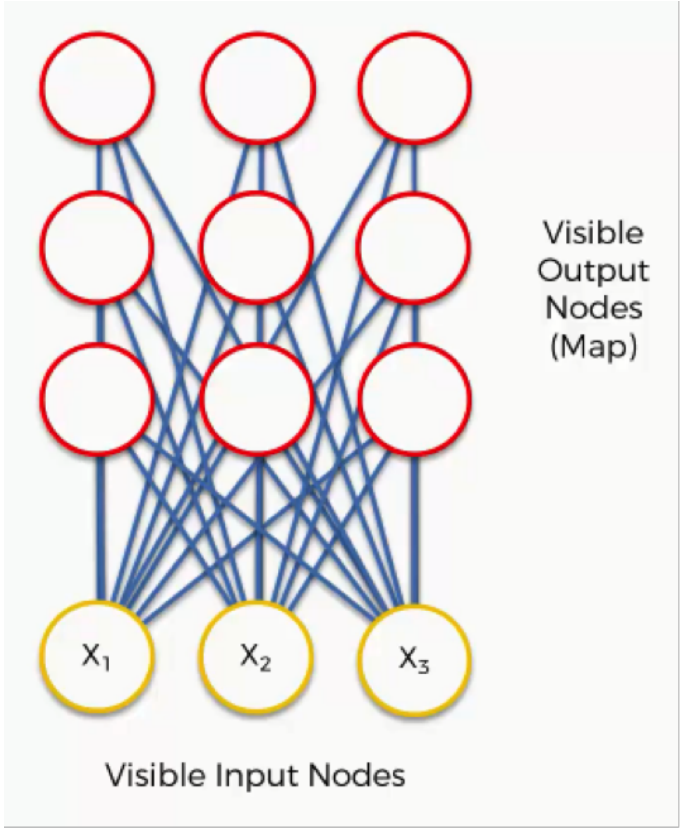
neuron $\approx$ processing unit

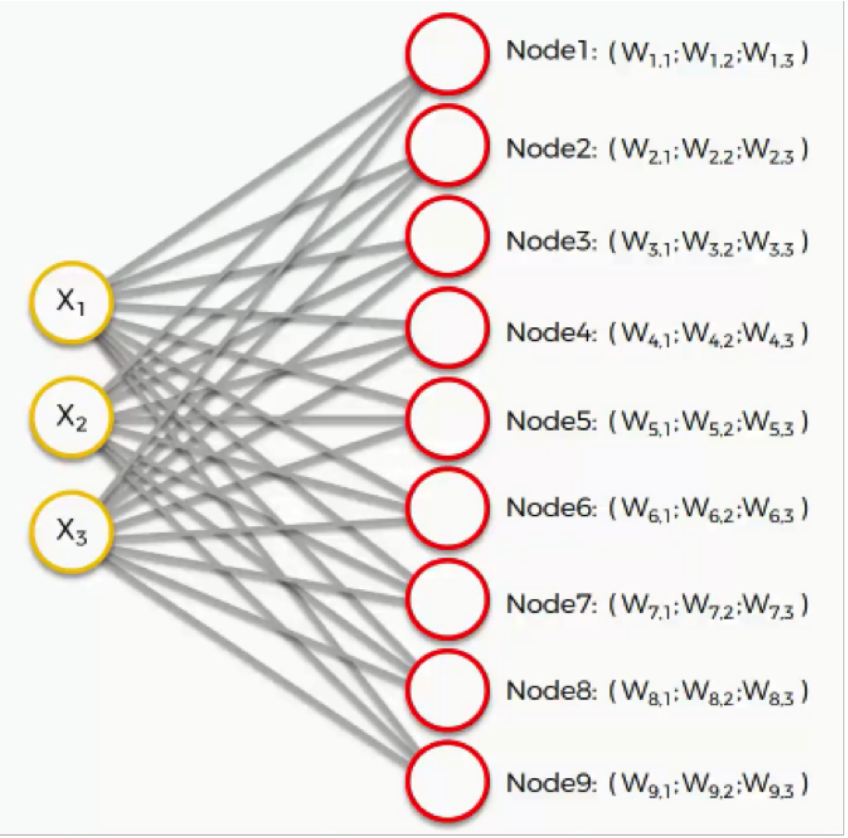




From: https://tinyurl.com/yf38snkk

https://tinyurl.com/2pkwvzem


Visible
Output
Nodes
(Map)
X1
X2
X3
Visible Input Nodes


X1
X2
X3
Node1: (W1,1;W1,2;W1,3)
Node2: (W2,1;W2,2;W2,3)
Node3: (W3,1;W3,2;W3,3)
Node4: (W4,1;W4,2;W4,3)
Node5: (W5,1;W5,2;W5,3)
Node6: (W6,1;W6,2;W6,3)
Node7: (W7,1;W7,2;W7,3)
Node8: (W8,1;W8,2;W8,3)
Node9: (W9,1;W9,2;W9,3)

Outline

1 SOM:Self Organizing Map

2 BMU: Best-Matching Unit

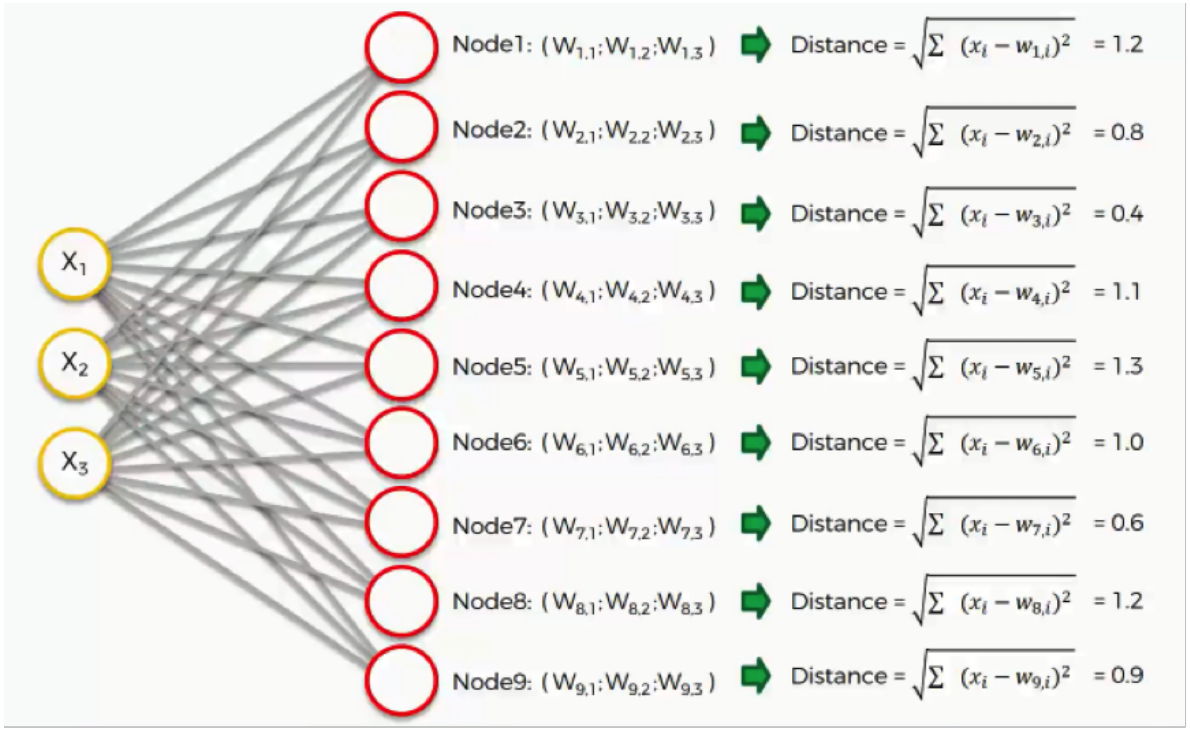
3 BMU Neighborhood

4 Updating

5 Summary of the SOM Algorithm

6 Cluster Evaluation

BMU: Best-Matching Unit)



- Node number 3 is the closest with a distance of 0.4.
- This node is called BMU (best-matching unit).

Outline

1 SOM:Self Organizing Map

2 BMU: Best-Matching Unit

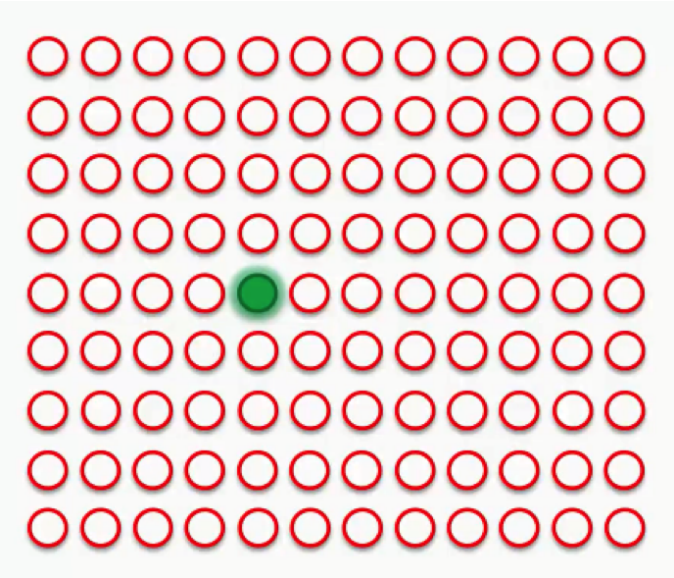
3 BMU Neighborhood

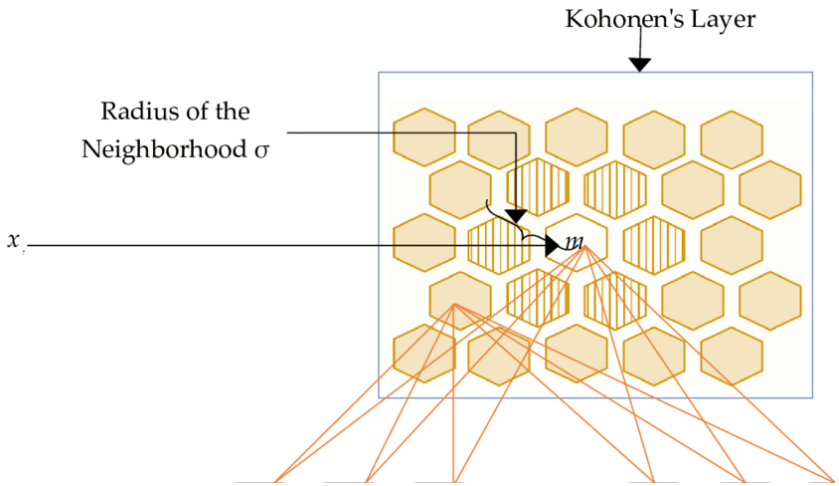
4 Updating

5 Summary of the SOM Algorithm

6 Cluster Evaluation

Neighborhood of BMU



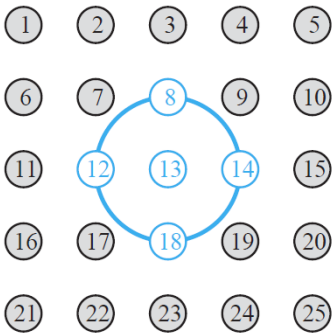


- The neighborhood $N_{i*}(r)$ contains the indices for all of the units that lie within a radius r of the BMU defined as i^* :

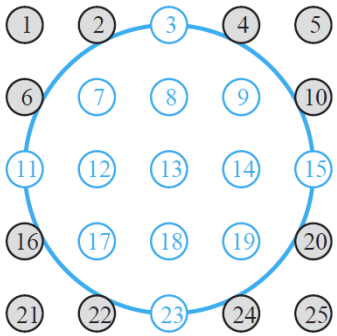
$$N_{i*}(r) = \{j, d_{j,i*} \leq r\}$$

$$d_{j,i*} = ||\mathbf{p}_j - \mathbf{p}_{i*}||$$

where the vector $\mathbf{p}_j$ defines the position of excited neuron j and $\mathbf{p}_{i*}$ defines the position of the winning neuron i^*.



$N_{13}(1)$



$N_{13}(2)$

$$N_{13}(1) = \{8, 12, 13, 14, 18\}$$

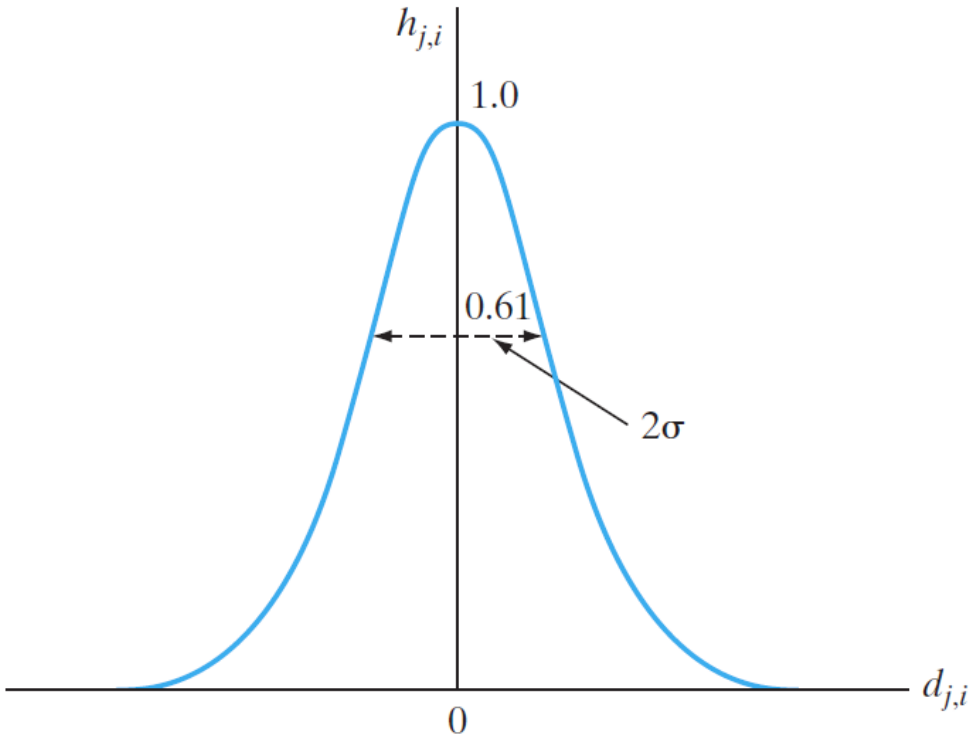
$$N_{13}(2) = \{3, 7, 8, 9, 11, 12, 13, 14, 15, 17, 18, 19, 23\}$$

- The best matching unit locates the center of a topological neighborhood of cooperating units.

$$h_{j,i^*} = exp(-\frac{d_{j,i^*}^2}{2\sigma^2})$$

- h_{j,i^*} is symmetric about the maximum point when $d_{j,i^*} = 0$.
- h_{j,i^*} decreases monotonically with increasing distance d_{j,i^*}, h_{j,i^*} closes to zero for $d_{j,i^*} \longrightarrow \infty$.

FIGURE 9.3 Gaussian neighborhood function.



Outline

1 SOM:Self Organizing Map

2 BMU: Best-Matching Unit

3 BMU Neighborhood

4 Updating

5 Summary of the SOM Algorithm

6 Cluster Evaluation

Updating

- Given an input vector $\mathbf{x}$.
- The weight vectors for all processing units within a certain neighborhood of the winning unit are updated using the Kohonen rule:
- By definition of neighborhood

$$\mathbf{w}_j^{(new)} = \mathbf{w}_j^{(old)} + \eta(\mathbf{x} - \mathbf{w}_j^{(old)}), \qquad j \in N_{i^*}(d)$$

- By Gaussian function

$$\mathbf{w}_j^{(new)} = \mathbf{w}_j^{(old)} + \eta h_{j,i^*}(\mathbf{x} - \mathbf{w}_j^{(old)})$$

where η is the learning rate parameter.

Outline

1 SOM:Self Organizing Map

2 BMU: Best-Matching Unit

3 BMU Neighborhood

4 Updating

5 Summary of the SOM Algorithm

6 Cluster Evaluation

Summary of the SOM Algorithm

1. **Initialization**: Choose random values for the initial weight vectors $\mathbf{w}_j$ for $j = 1, 2, ..., l$, where l is the number of neurons in the lattice.
2. **Sampling**: Get a sample $\mathbf{x}$ from the input space.
3. **Similarity matching**: Find the best-matching (winning) unit i^* by using the minimum-distance criterion

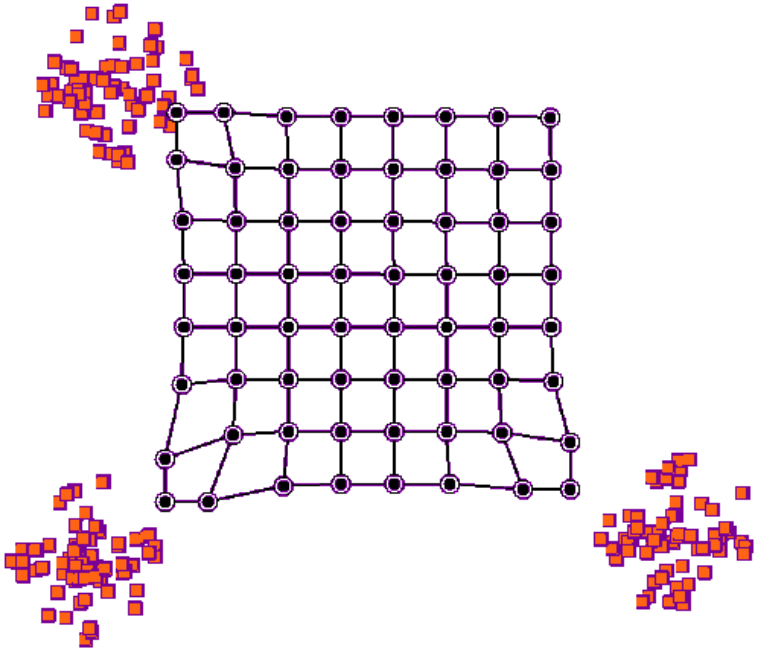
$$i^* = \arg\min_j ||\mathbf{x} - \mathbf{w}_j||, \quad j = 1, 2, ..., l$$

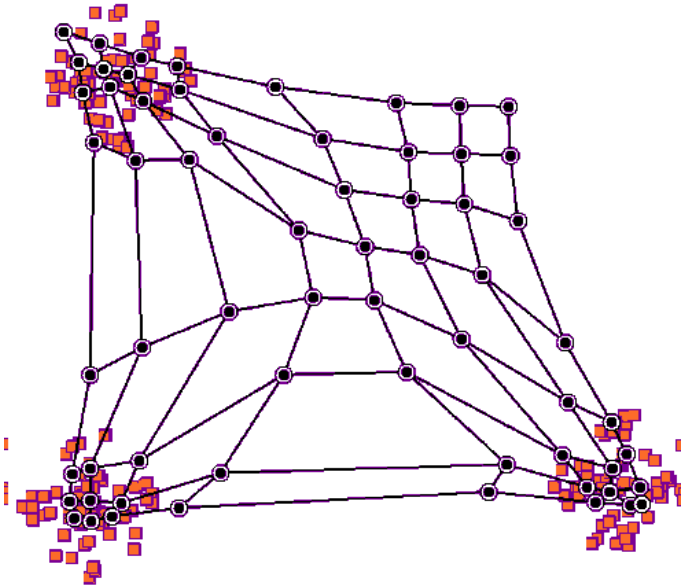
4. **Updating**: Adjust the synaptic-weight vectors of all excited units by using the update formula

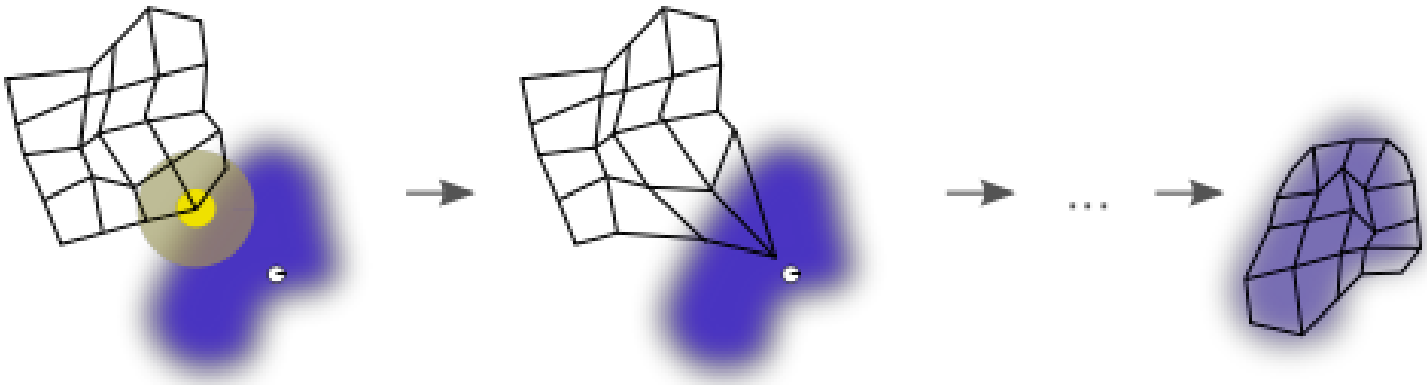
$$\mathbf{w}_j^{(new)} = \mathbf{w}_j^{(old)} + \eta(\mathbf{x} - \mathbf{w}_j^{(old)}), \qquad j \in N_{i^*}(d) \qquad \text{or}$$

$$\mathbf{w}_j^{(new)} = \mathbf{w}_j^{(old)} + \eta h_{j,i^*}(\mathbf{x} - \mathbf{w}_j^{(old)})$$

5. **Continuation**: Continue with step 2 until no noticeable changes in the feature map are observed.







https://en.wikipedia.org/wiki/Self-organizing_map

Example:

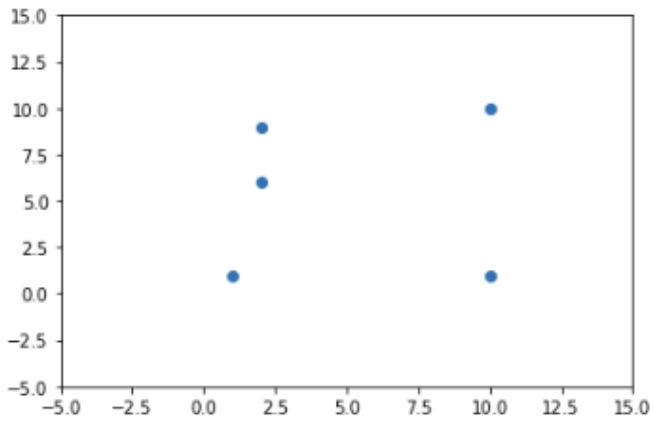
Gen 2D data

```
from sklearn_som.som import SOM
import numpy as np
import matplotlib.pyplot as plt

X = np.array([[1, 1],[2,6], [2,9], [10,1], [10,10]])
X
```

```
array([[ 1,  1],
       [ 2,  6],
       [ 2,  9],
       [10,  1],
       [10, 10]])
```

```
plt.scatter(X[:,0],X[:,1])
plt.xlim([-5,15])
plt.ylim([-5,15])
```

```
(-5.0, 15.0)
```



Initial SOM 1D

```
nrow = 1
ncol = 4
som1d = SOM(m=nrow, n=ncol, dim=2)
```

Overview the details of initiated som

```
som1d._locations   #Lattices/Location of processing units

```

```
array([[0, 0],
       [0, 1],
       [0, 2],
       [0, 3]], dtype=int64)
```

Initial SOM 2D

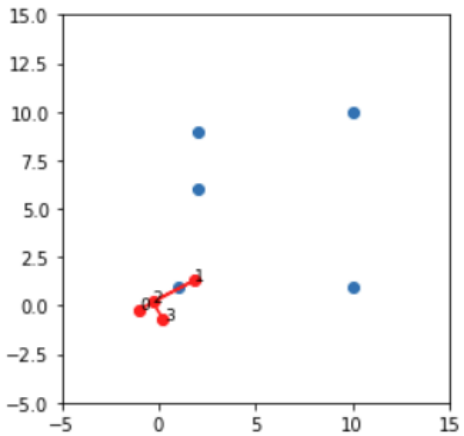
```
nrow = 3
ncol = 3
som2d = SOM(m=nrow, n=ncol , dim=2)
```

```
som2d._locations   #Lattices/Location of processing units
```

```
array([[0, 0],
       [0, 1],
       [0, 2],
       [1, 0],
       [1, 1],
       [1, 2],
       [2, 0],
       [2, 1],
       [2, 2]], dtype=int64)
```

SOM 1D

```
w = som1d.weights
plt.axes(aspect=1)
plt.scatter(X[:,0],X[:,1])
plt.xlim([-5,15])
plt.ylim([-5,15])


#plot weights/centers of processing units
plt.plot(w[:,0],w[:,1],'-ro')
for i in range(nrow*ncol):
    plt.text(w[i,0]+.03, w[i,1]+.03, i, fontsize=9)

```

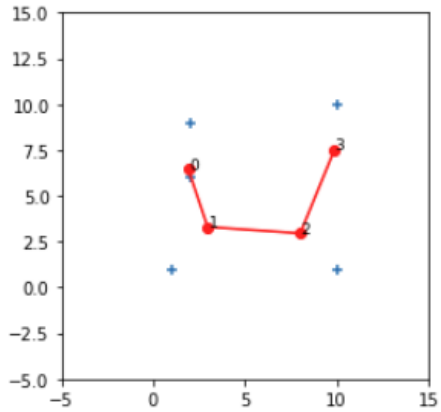


Updating/Learning

```
som1d.fit(X,epochs=100)
```

Visualization result of SOM 1D

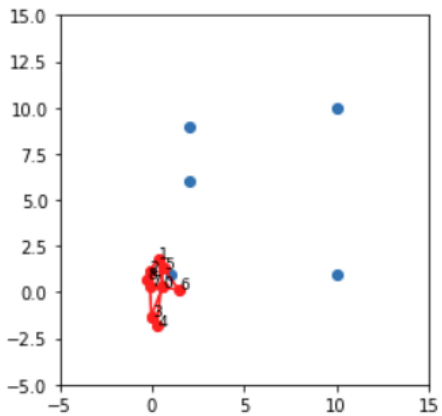
```
w = som1d.weights

plt.axes(aspect=1)
plt.scatter(X[:,0],X[:,1],marker='+')
plt.xlim([-5,15])
plt.ylim([-5,15])


#plot weights/centers of processing units
plt.plot(w[:,0],w[:,1],'-ro')
for i in range(nrow*ncol):
    plt.text(w[i,0]+.03, w[i,1]+.03, i, fontsize=9)

```



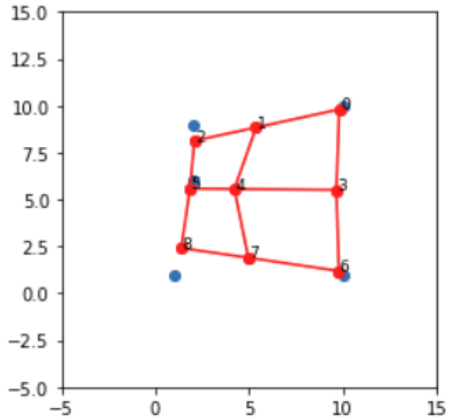
SOM 2D

```
w = som2d.weights
```

```
plt.axes(aspect=1)
plt.scatter(X[:,0],X[:,1])
plt.xlim([-5,15])
plt.ylim([-5,15])


#plot weights/centers of processing units
plt.plot(w[:,0],w[:,1],'-ro')
plt.plot(w[[0,3],0],w[[0,3],1],'-ro')
for i in range(nrow*ncol):
    plt.text(w[i,0]+.03, w[i,1]+.03, i, fontsize=9)

```




```
w = som2d.weights
plt.axes(aspect=1)
plt.scatter(X[:,0],X[:,1])
plt.xlim([-5,15])
plt.ylim([-5,15])


#plot weights/centers of processing units
plt.plot(w[0:3,0],w[0:3,1],'-ro')
plt.plot(w[3:6,0],w[3:6,1],'-ro')
plt.plot(w[6:9,0],w[6:9,1],'-ro')

plt.plot(w[[0,3,6],0],w[[0,3,6],1],'-ro')
plt.plot(w[[1,4,7],0],w[[1,4,7],1],'-ro')
plt.plot(w[[2,5,8],0],w[[2,5,8],1],'-ro')

for i in range(nrow*ncol):
    plt.text(w[i,0]+.03, w[i,1]+.03, i, fontsize=9)

```

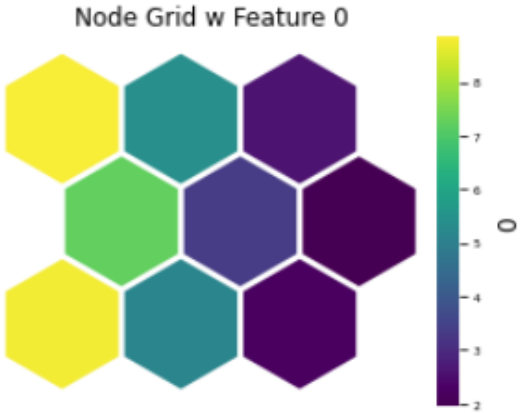


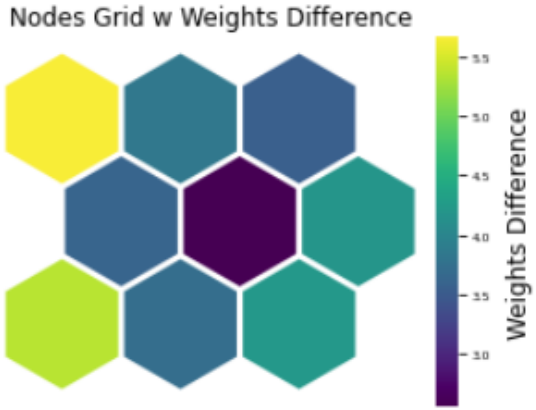
simpsom : https://pypi.org/project/simpsom/

```
1  import simpsom as sps
2
3  net = sps.SOMNet(3, 3, X, PBC= True,init='random')
4
```

```
1  net.train()
```

```
1  net.nodes_graph(colnum=0)
2  net.diff_graph()
```

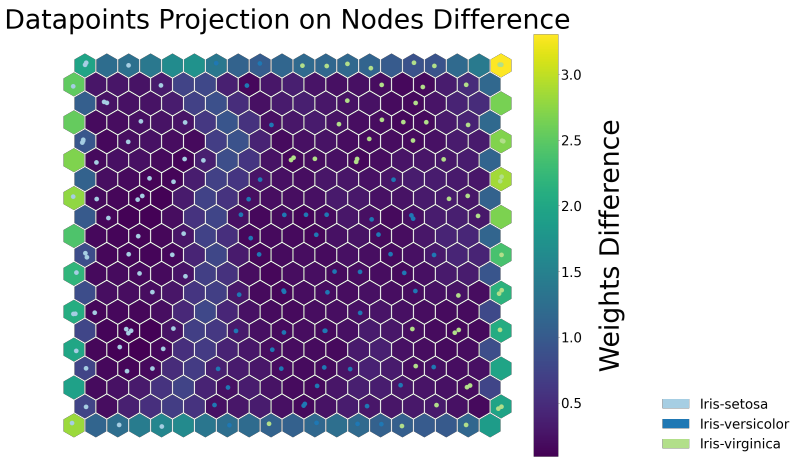




Iris dataset

```
Iris = pd.read_csv('Iris.csv')
labels = Iris['Species'].values

Iris = Iris.drop(Iris.columns[[0,-1]],axis=1)

scaler = StandardScaler()
scaler= scaler.fit(Iris.values)
train = scaler.transform(Iris.values)
#print(train.shape)

```

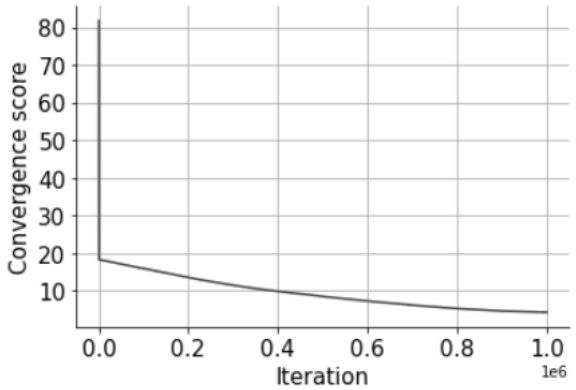
```
net = sps.SOMNet(20, 20, train, PBC=True)
net.train()
#Save the weights to file
net.save('som_iris_weights')

```



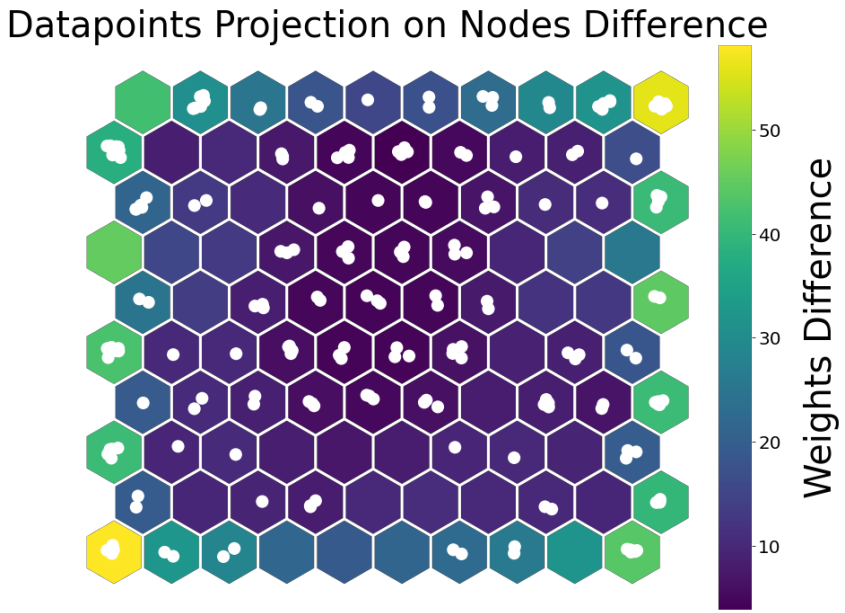
Customer data

```
train = df.values

net = sps.SOMNet(10, 10, train, PBC=True)
#net.train(start_learning_rate=0.1, epochs=10000)
net.train(start_learning_rate=0.1,epochs=1000000, early_stop='bmudiff',
      early_stop_patience=10, early_stop_tolerance=5e-10)
```

```
Periodic Boundary Conditions active.
The weights will be initialized with PCA.
The map will be trained with the batch algorithm.
Training SOM... done!
<Figure size 2160x720 with 0 Axes>
```



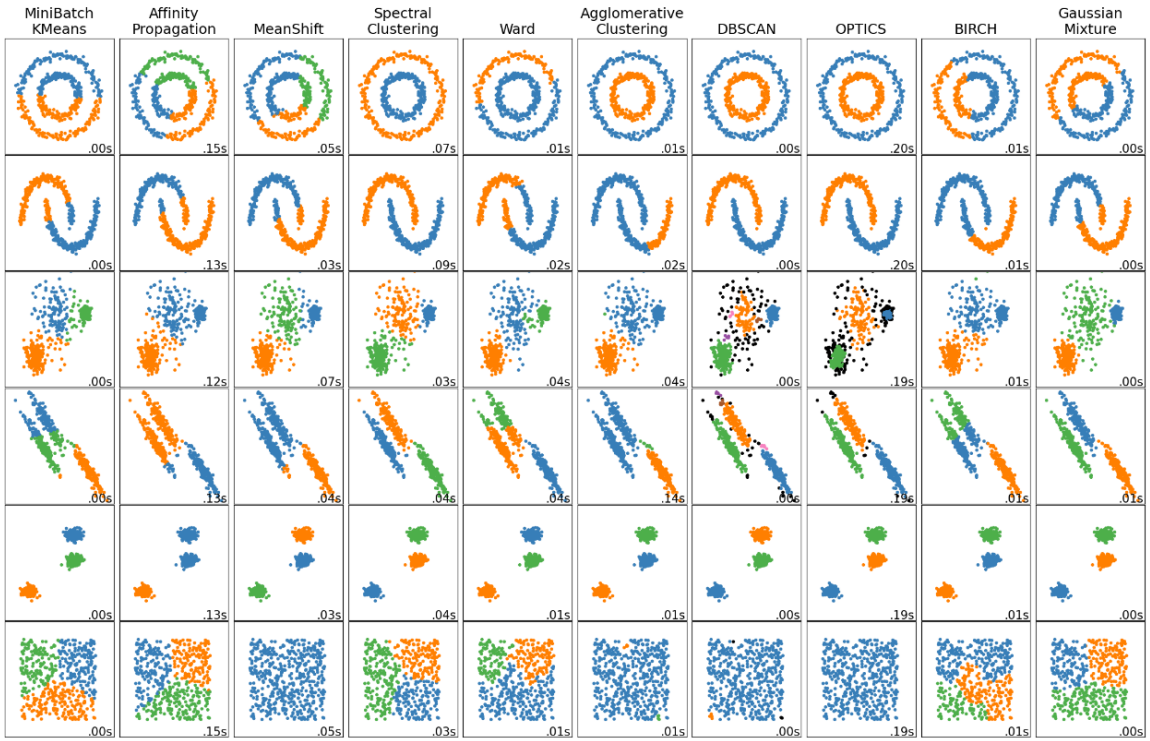

```
net.nodes_graph(colnum=1)
net.diff_graph()
```



```
cls = net.cluster(train, clus_type='KMeans') #'MeanShift') #
```

```
for c in range(len(cls)):
    plt.scatter(train[cls[c], 0],
                train[cls[c], 1], label='cluster='+str(c), alpha=.7)

plt.title("Clusters of Customers")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.legend(loc='center left', bbox_to_anchor=(1, 0.5));
```

Other Clustering Methods



Outline

1 SOM:Self Organizing Map

2 BMU: Best-Matching Unit

3 BMU Neighborhood

4 Updating

5 Summary of the SOM Algorithm

6 Cluster Evaluation

Cluster Evaluation

Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following two types.

- Supervised: Used to measure the extent to which cluster labels match externally supplied class labels.
  - Rand Index
  - Entropy
  - Purity
- Unsupervised: Used to measure the goodness of a clustering structure without respect to external information.
  - Sum of Squared Error (SSE)
  - Silhouette Coefficient

Rand index

- If C is a ground truth class assignment and K the clustering
- a, the number of pairs of elements that are in the same set in C and in the same set in K
- b, the number of pairs of elements that are in different sets in C and in different sets in K

$$\mathbf{RI} = \frac{a + b}{C_2^{n_{samples}}}$$

```
>>> from sklearn import metrics
>>> labels_true = [0, 0, 0, 1, 1, 1]
>>> labels_pred = [0, 0, 1, 1, 2, 2]
>>> metrics.rand_score(labels_true, labels_pred)
0.66...
```

https://scikit-learn.org/stable/modules/clustering.html#clustering-performance-evaluation

Entropy

- p_{ij} : the probability that a member of cluster i belongs to class j.

$$p_{ij} = \frac{m_{ij}}{m_i}$$

where m_i is the number of objects in cluster i and
m_ij is the number of objects of class j in cluster i.

- the entropy of each cluster i :

$$e_i = -\sum_{j=1}^{L} p_{ij} \log_2 p_{ij}$$

where L is the number of classes.

- The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster

$$e = \sum_{i=1}^{K} \frac{m_i}{m} e_i$$

where K is the number of clusters and m is the total number of data points.

Purity

- Using the previous terminology,

$$p_{ij} = \frac{m_{ij}}{m_i}$$

where m_i is the number of objects in cluster i and
m_ij is the number of objects of class j in cluster i.

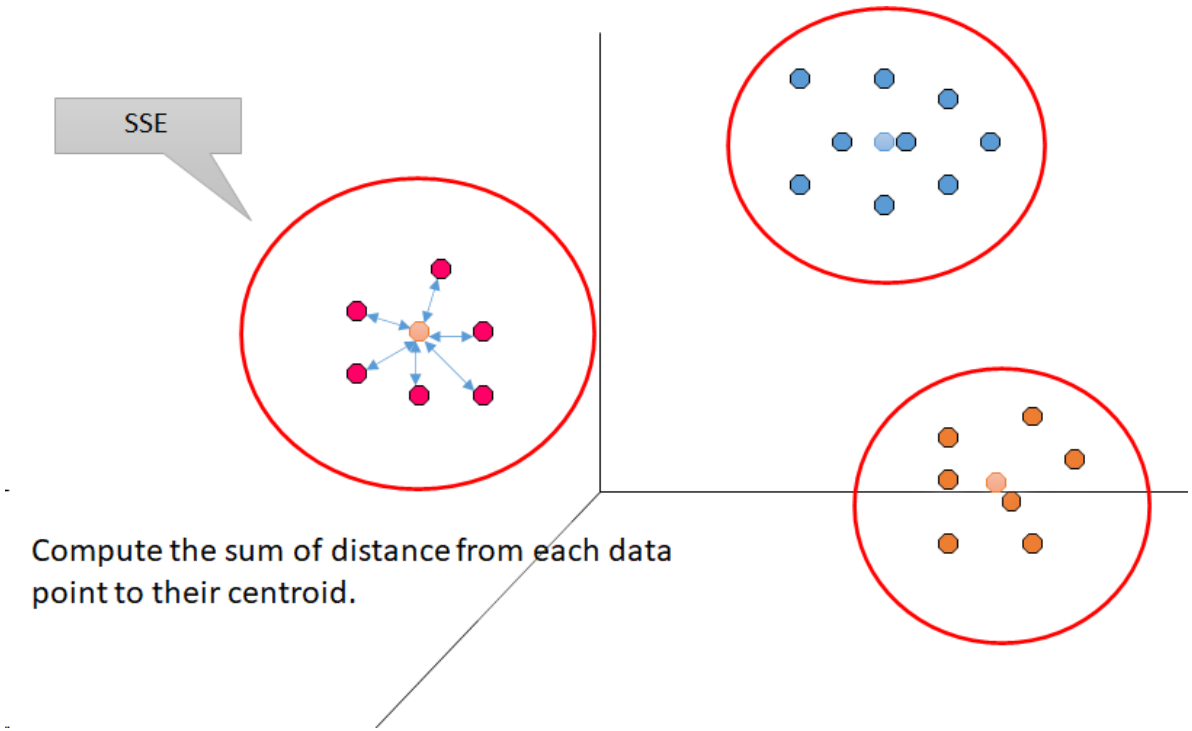
- Purity of cluster i

$$purity(i) = \max_j p_{ij}$$

- The overall purity of a clustering is

$$purity = \sum_{i=1}^{K} \frac{m_i}{m} purity(i)$$

Sum of Squared Error (SSE)

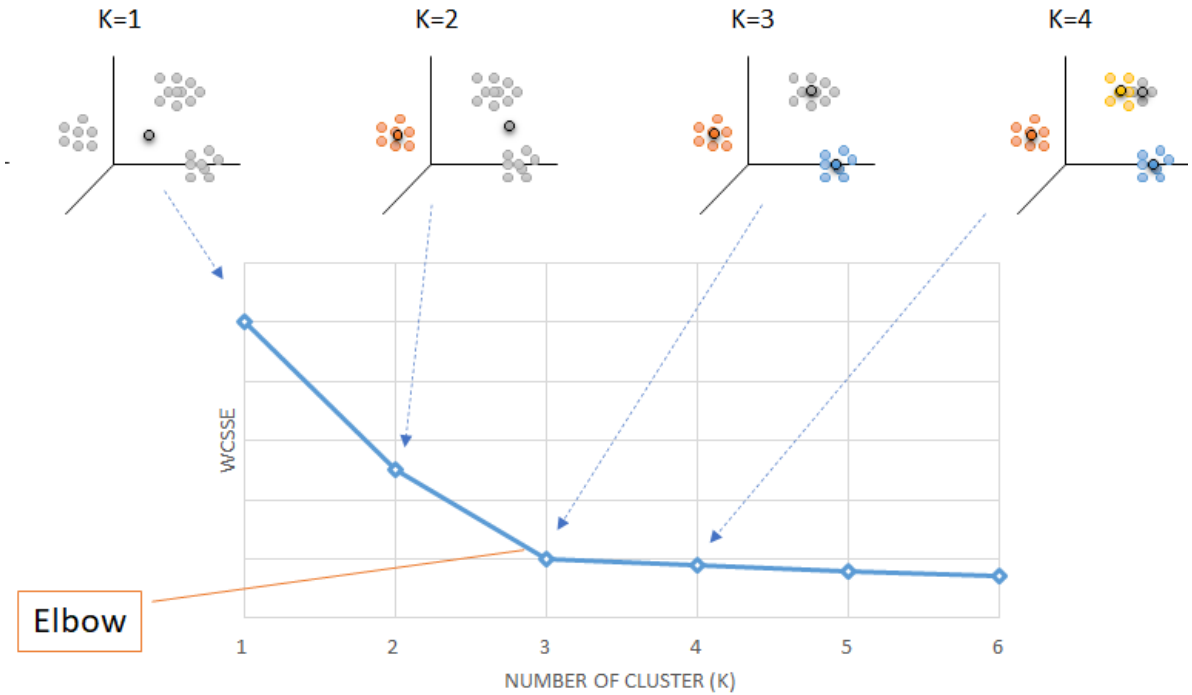


$$ClusterSSE = \sum_{\mathbf{x} \in C_i} dist(\mathbf{c}_i, \mathbf{x})$$

WCSSE is the Sum of the Squared differences (Error) between each observation and its group's mean (Within Cluster)

Selecting number of clusters

- Using elbow method to determine the optimal number of clusters



Silhouette Coefficient

- If the ground truth labels are not known, evaluation must be performed using the model itself.
- a : The mean distance between a sample and all other points in the same class.
- b : The mean distance between a sample and all other points in the next nearest cluster.
- The Silhouette Coefficient s for a single sample is then given as:

$$s = \frac{b - a}{\max(a, b)}$$

The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample.

```
>>> import numpy as np
>>> from sklearn.cluster import KMeans
>>> kmeans_model = KMeans(n_clusters=3, random_state=1).fit(X)
>>> labels = kmeans_model.labels_
>>> metrics.silhouette_score(X, labels, metric='euclidean')
0.55...
```

Example

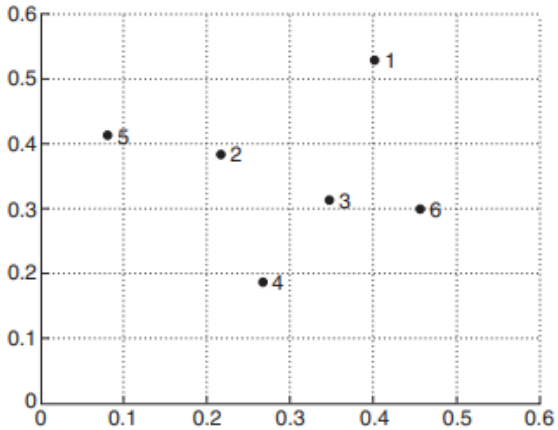


Figure 7.15. Set of six two-dimensional points.

Point	x Coordinate	y Coordinate
p1	0.4005	0.5306
p2	0.2148	0.3854
p3	0.3457	0.3156
p4	0.2652	0.1875
p5	0.0789	0.4139
p6	0.4548	0.3022

Table 7.3. xy-coordinates of six points.

	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Table 7.4. Euclidean distance matrix for six points.