Similarity-1

计算距离测度。一组均值为$[0,0]$和协方差矩阵为单位矩阵的数据，现有一个样本$x=[3,4]$，请问这个样本到原点的Mahalanobis距离是

多少？如果数据的协方差矩阵是$\begin{bmatrix}1 & 0.5\\0.5 & 1\end{bmatrix}$，还是这个样本，其与原点的Mahalanobis距离会发生变化吗？会是多少？

解：

1. 当协方差矩阵为$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$时：

$$D_M(x,y)=\sqrt{(x-y)^T\Sigma^{-1}(x-y)}=\sqrt{(\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix})^T\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}(\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix})}=\sqrt{[3,4]\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\begin{bmatrix}3\\4\end{bmatrix}}=5$$

2. 当协方差矩阵为$\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$时：

$$D_M(x,y)=\sqrt{(x-y)^T\Sigma^{-1}(x-y)}=\sqrt{(\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix})^T\begin{bmatrix}\frac{4}{3} & -\frac{2}{3}\\-\frac{2}{3} & \frac{4}{3}\end{bmatrix}(\begin{bmatrix}3\\4\end{bmatrix}-\begin{bmatrix}0\\0\end{bmatrix})}=\sqrt{[3,4]\begin{bmatrix}\frac{4}{3} & -\frac{2}{3}\\-\frac{2}{3} & \frac{4}{3}\end{bmatrix}\begin{bmatrix}3\\4\end{bmatrix}}=\sqrt{\frac{52}{3}}$$

3. 结论：

Σ是多维随机变量的协方差矩阵，若协方差矩阵是单位矩阵，即各维度独立同分布，则此时马氏距离就退化成了欧氏距离。

当样本点不变，而这组样本的协方差矩阵变化时，根据公式，最终两点之间的马氏距离也会变化。

Similarity-2

设两个统计独立的随机变量s_1和s_2的分布分别为：

$$p(s_i)=\begin{cases}\dfrac{1}{2\sqrt{3}} & \text{if}\,|s_i|\le\sqrt{3}\\0 & \text{otherwise}\end{cases}$$

若记$S=\begin{bmatrix}s_1\\s_2\end{bmatrix}$，请问经过下面的线性变换$X=AS$，其中$A=\begin{bmatrix}2 & 1\\1 & 3\end{bmatrix}$那么，$X=\begin{bmatrix}x_1\\x_2\end{bmatrix}$中的两个随机变量$x_1$和$x_2$是否还是统计独立

的？如果不是，请问它们的相关系数是多少？(实际上，典型的鸡尾酒会问题就是从观测数据X中恢复出统计独立信源S的问题，在实际问

题中有大量应用，解决这类问题的典型技术是ICA(independent component analysis，独立分量分析，即教材中的7.3节)

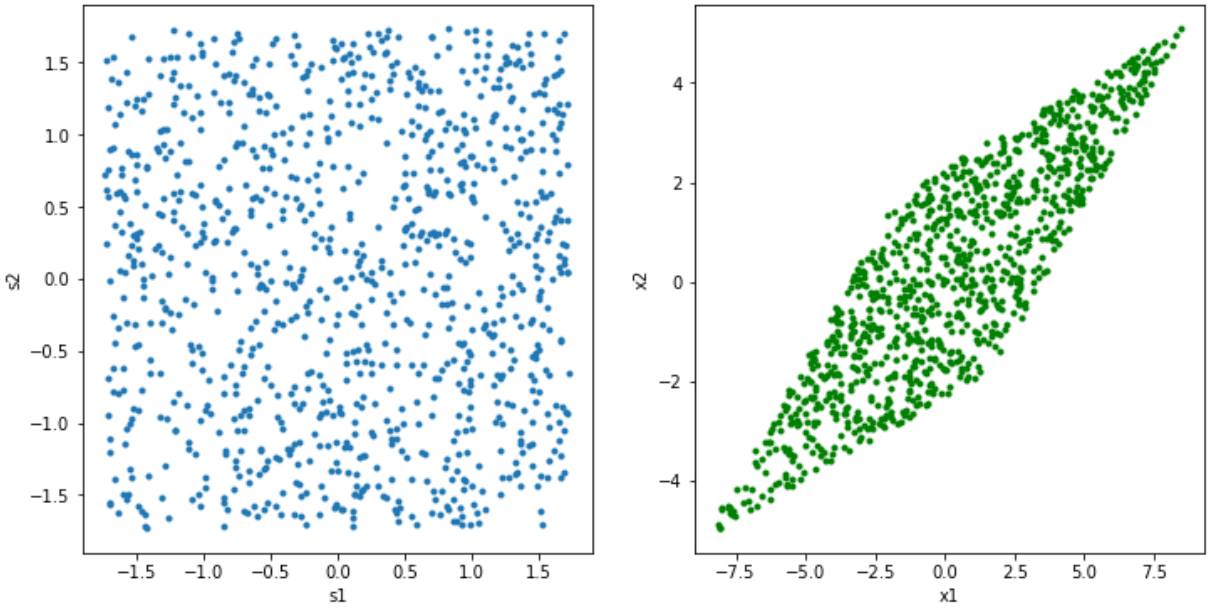
解：

可以利用python对变换前后的数据进行模拟，并计算出变换前后两个维度的相关系数。代码如下：

```python
import numpy as np
from numpy import random
import matplotlib.pyplot as plt

s1 = random.uniform(low=-np.sqrt(3), high=np.sqrt(3), size=1000)
s2 = random.uniform(low=-np.sqrt(3), high=np.sqrt(3), size=1000)
x1 = 2 * s1 + 3 * s2
x2 = 2 * s1 + 1 * s2
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(s1, s2, '.')
plt.xlabel('s1')
```

```
plt.ylabel('s2')
plt.subplot(1, 2, 2)
plt.plot(x1, x2, '.', color='green')
plt.xlabel('x1')
plt.ylabel('x2')

print('s1和s2的相关系数：{:.4f}'.format(np.corrcoef(s1, s2)[0,1]))
print('x1和x2的相关系数：{:.4f}'.format(np.corrcoef(x1, x2)[0,1]))
```

s_1, s_2和x_1, x_2的分布散点图如下：



变换前后两个维度的相关系数分别为：

```
s1和s2的相关系数：0.0160
x1和x2的相关系数：0.8720
```

结论：

原本两个离散随机变量s_1, s_2独立，服从各自的均匀分布，它们的相关系数为0.0160，表示s_1, s_2几乎无线性相关关系。但它们联合起来服从二维空间中的二维均匀分布，经过矩阵A的线性变换后得到的x_1, x_2之间的相关系数为0.8720，表示x_1, x_2之间呈显著的线性相关关系。由于"独立一定不相关"，可知相关一定不独立。而x_1与x_2显著线性相关，证明x_1与x_2不统计独立。

SOM-3

在SOM的学习过程中，与输入样本最为匹配的神经元，称为winning neuron, 那么请问，这里最为匹配是什么含义？可以用相似性/距离

测度来表示吗?

答：

最为匹配：计算竞争层（输出层）所有神经元的权向量与当前输入样本之间的相似度/距离，我们称该与输入样本最相近/相似的那个神经元为与当前输入样本最为匹配的神经元，这里的评估准则既可以使用相似性度量（如向量的余弦相似度等），也可以用距离度量（如欧氏距离等），只是要注意相似度越大两个向量越相近，距离越小两个向量越相近。

上机题

SOM-C-1 (Dataset comes from Duda's book, pp.681)

Here is a dataset. Use SOM for the clustering of the dataset. Discuss the effect of structure of the SOM, and number of neurons in the SOM, and visualize the learning process of the SOM by related matlab functions.

sample	x_1	x_2	x_3
1	-7.82	-4.58	-3.97
2	-6.68	3.16	2.71
3	4.36	-2.19	2.09
4	6.72	0.88	2.80
5	-8.64	3.06	3.50
6	-6.87	0.57	-5.45
7	4.47	-2.62	5.76
8	6.73	-2.01	4.18
9	-7.71	2.34	-6.33
10	-6.91	-0.49	-5.68

sample	x_1	x_2	x_3
11	6.18	2.81	5.82
12	6.72	-0.93	-4.04
13	-6.25	-0.26	0.56
14	-6.94	-1.22	1.13
15	8.09	0.20	2.25
16	6.81	0.17	-4.15
17	-5.19	4.24	4.04
18	-6.38	-1.74	1.43
19	4.08	1.30	5.33
20	6.27	0.93	-2.78

解：

先将上述数据写入xlsx表格，再导入Matlab中。利用Matlab自带的SOM工具箱对20个样本点进行SOM聚类分析。为了便于聚类结果的可视化，我选择将竞争神经元排布在二维拓扑空间中。为了探究不同的竞争层神经元排列结构和神经元个数对于聚类结果的影响，我分别设置竞争层结构为1x2、1x3、1x4、2x2、2x3、3x3并进行了相应的对比实验。总体脚本代码如下：

```
close all;clear all;clc;
[X,Y,~]=xlsread('data.xlsx');   % 读取文件
net = selforgmap([2 2]);        % 设置输出层神经元结构
net.trainparam.epochs = 20000;  % 设置迭代次数

net = train(net,X');     % 训练SOM网络
view(net)                % 查看网络结构
plotsompos(net,X')       % 绘制hit图
y = net(X');             % 输出y
classes = vec2ind(y);    % 将0 1向量转为索引
z=Y(2:end,1)';           % 将样本名称转为一行向量
d={classes,z};           % 将样本名称与索引对应

% 进行排序输出
k=unique(classes);       % k是类别个数
i=1;                     % 初始化i
while (i <= size(k,2))
    [~,n] = find(classes == k(i));
    disp(z(n));          % 输出每个类别有哪些样本
    disp('\n');
    i = i+1;             % i自增1
end
```

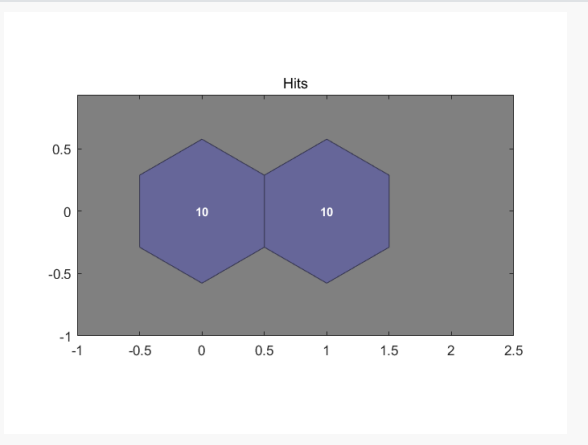
实验结果：

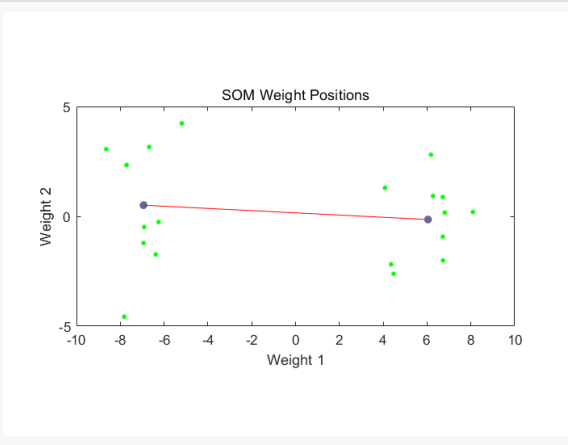
1. 竞争层神经元结构为1x2时：

聚类结果：

```
>> som
Class 1:    's1'    's2'    's5'    's6'    's9'    's10'    's13'    's14'    's17'    's18'

Class 2:    's3'    's4'    's7'    's8'    's11'    's12'    's15'    's16'    's19'    's20'
```

可视化结果：





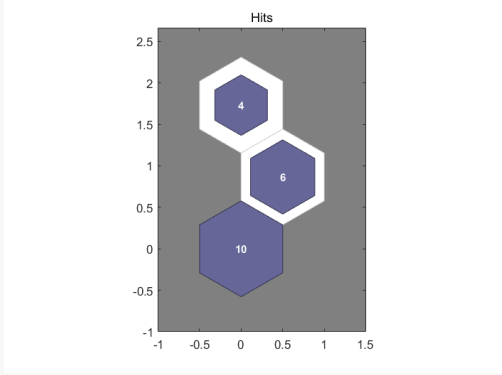
可以看到，20个样本被很好地均匀划分进了两个聚类类别中，这与二维平面上的散点分布情况也是一致的，即所有样本点差不多均匀分布在两个类别中。

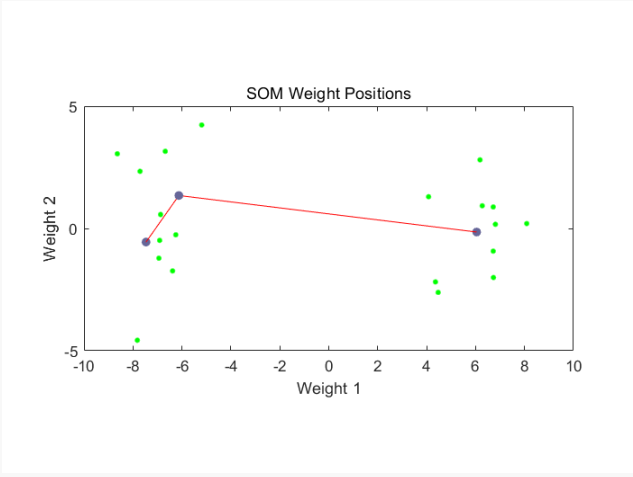
2. 竞争层神经元结构为1x3时：

聚类结果：

```
>> som
Class 1:    's3'    's4'    's7'    's8'    's11'    's12'    's15'    's16'    's19'    's20'

Class 2:    's2'    's5'    's13'    's14'    's17'    's18'

Class 3:    's1'    's6'    's9'    's10'
```

可视化结果：





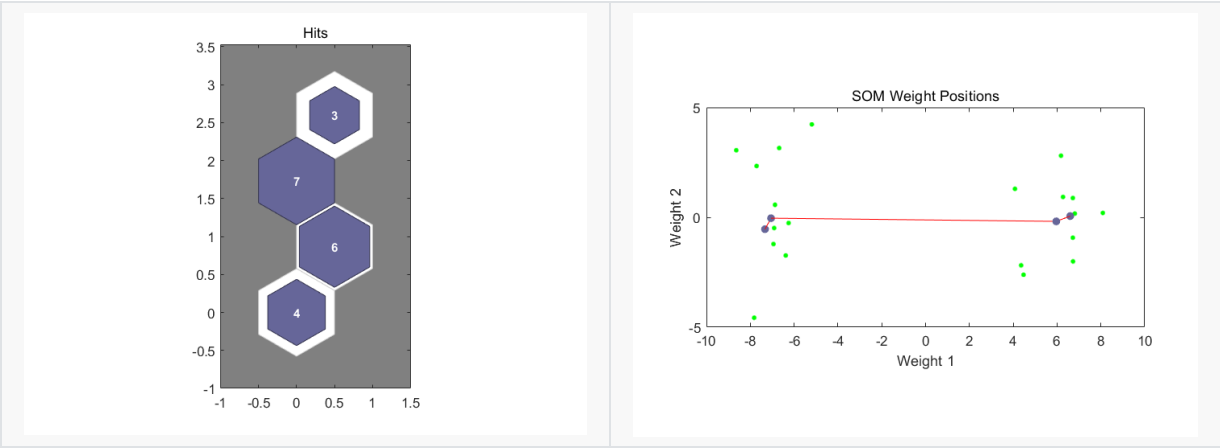
可以看到，当竞争神经元数量为3时，经过多次迭代后，20个样本被较好地划分入了3个聚类类别中。与前一次实验相比，这次有10个样本被划分为一类，而另外10个样本被划分入了两类，即类与类包含的样本数是不平衡的。但这同样符合人的直觉。为了把样本分为3类，必须把一个较大的簇拆分为两个簇。

3. 竞争层神经元结构为1x4时：

聚类结果：

```
>> som
Class 1:    's1'    's6'    's9'    's10'

Class 2:    's2'    's5'    's13'    's14'    's17'    's18'

Class 3:    's3'    's4'    's7'    's8'    's11'    's15'    's19'

Class 4:    's12'    's16'    's20'
```

可视化结果：



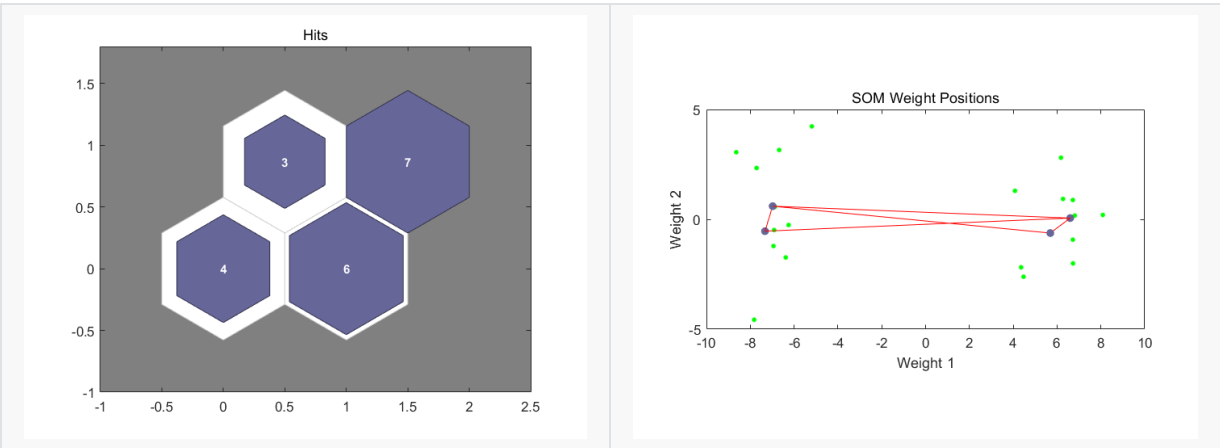
当竞争神经元数量为4，拓扑结构为1x4（即分布在空间的一串上）时，经过多次迭代后，20个样本被较好地划分入了4个聚类类别中。与前一次实验（1x3）相比，这次的结果中类与类包含的样本数较为平衡。

4. 竞争层神经元结构为2x2时：

聚类结果：

```
>> som
Class 1:    's1'    's6'    's9'    's10'

Class 2:    's2'    's5'    's13'    's14'    's17'    's18'

Class 3:    's12'    's16'    's20'

Class 4:    's3'    's4'    's7'    's8'    's11'    's15'    's19'
```

可视化结果：



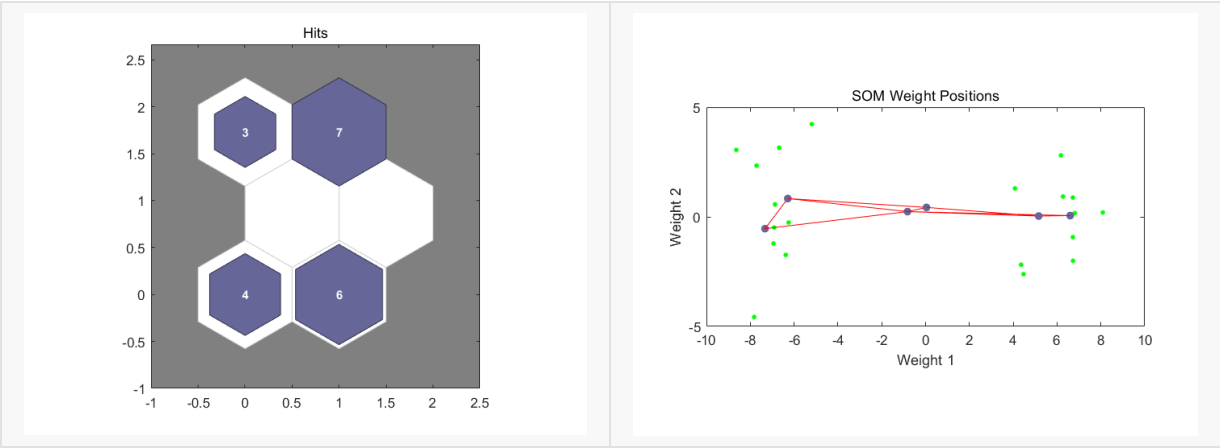
当竞争神经元数量为4，拓扑结构为2x2（Hexagonal型分布时）时，经过多次迭代后，20个样本被较好地划分入了4个聚类类别中。本次实验的结果与前一次实验（1x4）相比，是完全一致的。这说明当神经元数量相同，而拓扑排列机构不同时，聚类结果仍是相同的。

5. 竞争层神经元结构为2x3时：

聚类结果：

```
>> som
Class 1:    's1'    's6'    's9'    's10'

Class 2:    's2'    's5'    's13'    's14'    's17'    's18'

Class 5:    's12'    's16'    's20'

Class 6:    's3'    's4'    's7'    's8'    's11'    's15'    's19'
```

可视化结果：



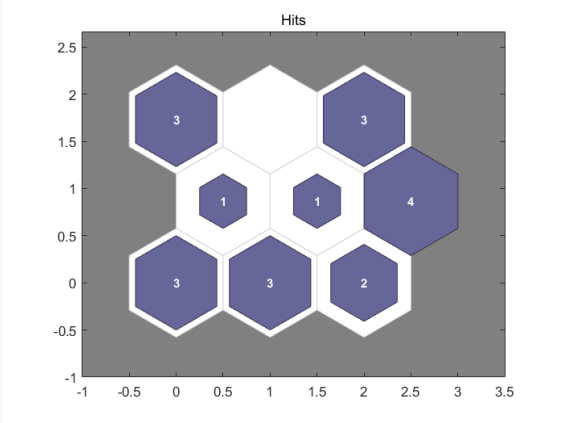
当竞争神经元数量为6，拓扑结构为2x3（Hexagonal型分布时）时，经过多次迭代后，20个样本被较好地划分入了4个聚类类别中。本次实验的结果与前一次实验（2x2）相比，是完全一致的。不难发现，“中间”那两个神经元至始自终没有竞争得过其他4个神经元，这导致它们两个成为了“死节点”，一直无法得到位置更新，从而也无法成为任何簇的聚类中心点。

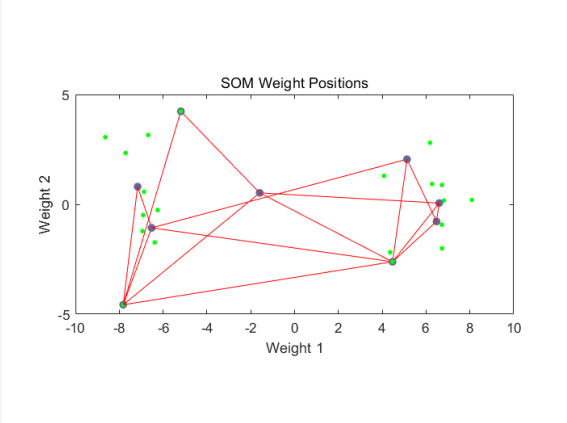
6. 竞争层神经元结构为3x3时：

聚类结果：

```
>> som
Class 1:    's6'    's9'    's10'

Class 2:    's13'    's14'    's18'

Class 3:    's11'    's19'

Class 4:    's1'

Class 5:    's7'

Class 6:    's3'    's4'    's8'    's15'

Class 7:    's2'    's5'    's17'

Class 9:    's12'    's16'    's20'
```

可视化结果：





当竞争神经元数量为9，拓扑结构为3x3（Hexagonal型分布时）时，经过多次迭代后，20个样本被划分到了8个聚类类别中。本次实验过程中，“中间”那个神经元(Class 8)至始自终没有竞争得过其他8个神经元，这导致它成为了“死节点”，一直无法得到位置更新，从而也无法成为任何簇的聚类中心点，聚类结果也证实了这一点，即Class 8中无任何样本点。

通过本次上机实验的若干对比实验，我加深了对于SOM算法应用于聚类的理解，并探究出了以下几点结论：

1. 当竞争层神经元数量确定时，即使神经元的拓扑结构发生变化（如从Hexagonal型分布时变为串型分布），也不会对聚类结果产生较大的改变，这体现出了SOM算法的“自组织”能力，也可以说明该算法鲁棒性较强，对于神经元排列结构不太敏感。

2. 当竞争层神经元的数量和拓扑排列结构都确定时，多次运行的结果也有可能不一致，推测这是因为在训练过程中是随机挑选样本进行输入，每次更新的权向量会有稍许不同，导致每次训练的最终结果也有所不同。

3. 当竞争层神经元数量增加时，SOM算法会倾向于将样本划分到更多的类中。对于像本实验这样的小样本数据集而言，一般我们选取竞争层神经元数量不宜过多。若神经元过多，会导致最终每个样本都被分为一类，失去了聚类的意义。

4. SOM网络拥有拓扑保持的能力，即：在原始输入空间中相距较近的样本也会被投影到相近的神经元。

5. SOM网络能将任意维度的输入在输出层映射成一维或二维图形,并保持原始拓扑结构不变。这一特性使得该算法能够实现数据降维，且便于人们对结果进行可视化分析。