

# 24

## Spectral Clustering

# 数据聚类

用谱聚类完成数据聚类



如果冬天来了，春天还会远吗？

*If Winter comes, can Spring be far behind.*

—— 雪莱 (Percy Bysshe Shelley) | 英国画家 | 1792 ~ 1822



- ◀ `sklearn.cluster.SpectralClustering()` 谱聚类算法
- ◀ `sklearn.datasets.make_circles()` 创建环形样本数据
- ◀ `sklearn.preprocessing.StandardScaler().fit_transform()` 标准化数据；通过减去均值然后除以标准差，处理后数据符合标准正态分布

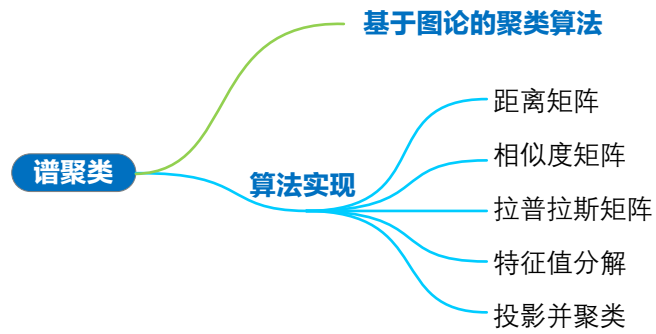
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)



## 24.1 数据聚类

本章介绍如何用图论完成**聚类** (clustering)。聚类是**无监督学习** (unsupervised learning) 中的一类问题。简单来说，聚类是指将数据集中相似的数据分为一类，以便更好地分析和理解数据。

如图 1 所示，删除鸢尾花数据集的标签，即 **target**，仅仅根据鸢尾花**花萼长度** (sepal length)、**花萼宽度** (sepal width) 这两个特征上样本数据分布情况，我们可以将数据分成两**簇** (clusters)。

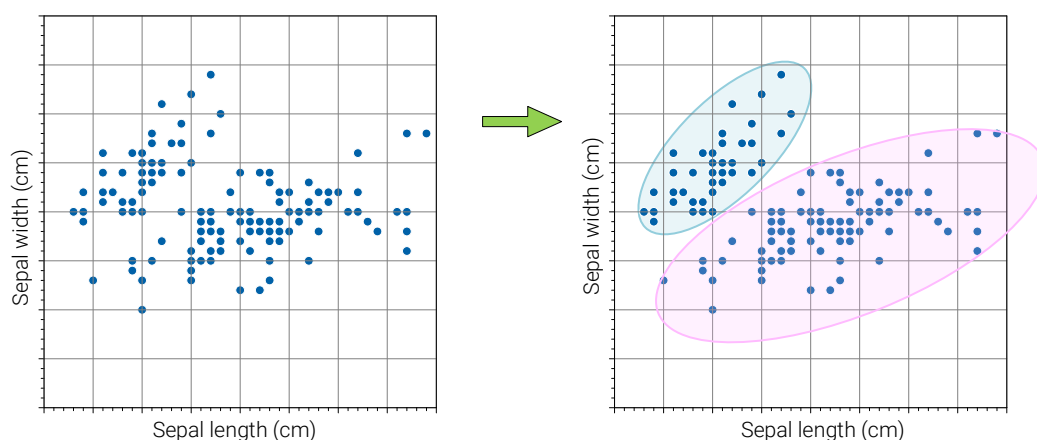


图 1. 用删除标签的鸢尾花数据介绍聚类算法

聚类算法有很多，下面要介绍的是**谱聚类** (spectral clustering)，它是一种基于无向图的聚类算法。

用无向图聚类思路很简单，切断无向图中权重值低的边，得到一系列子图。子图内部节点之间边的权重尽可能高，子图之间边权重尽可能低。

谱聚类算法流程如下。

- ▶ 首先，需要计算数据矩阵  $X$  内点与点的成对距离，并构造距离矩阵  $D$ 。
- ▶ 然后，将距离转换成权重值，即**相似度** (similarity)，构造**相似度矩阵** (similarity matrix)  $S$ ，利用  $S$  可以绘制无向图。
- ▶ 之后，将相似度矩阵转化成**拉普拉斯矩阵** (Laplacian matrix)  $L$ 。
- ▶ 最后，**特征值分解** (eigen decomposition)  $L$ ，相当于将  $L$  投影在一个低维度正交空间。
- ▶ 在这个低维度空间中，用简单聚类方法对投影数据进行聚类，并得到原始数据聚类。

图 2 所示为谱聚类的算法流程。

下面通过实例，我们一一讨论谱聚类这些步骤所涉及的技术细节。

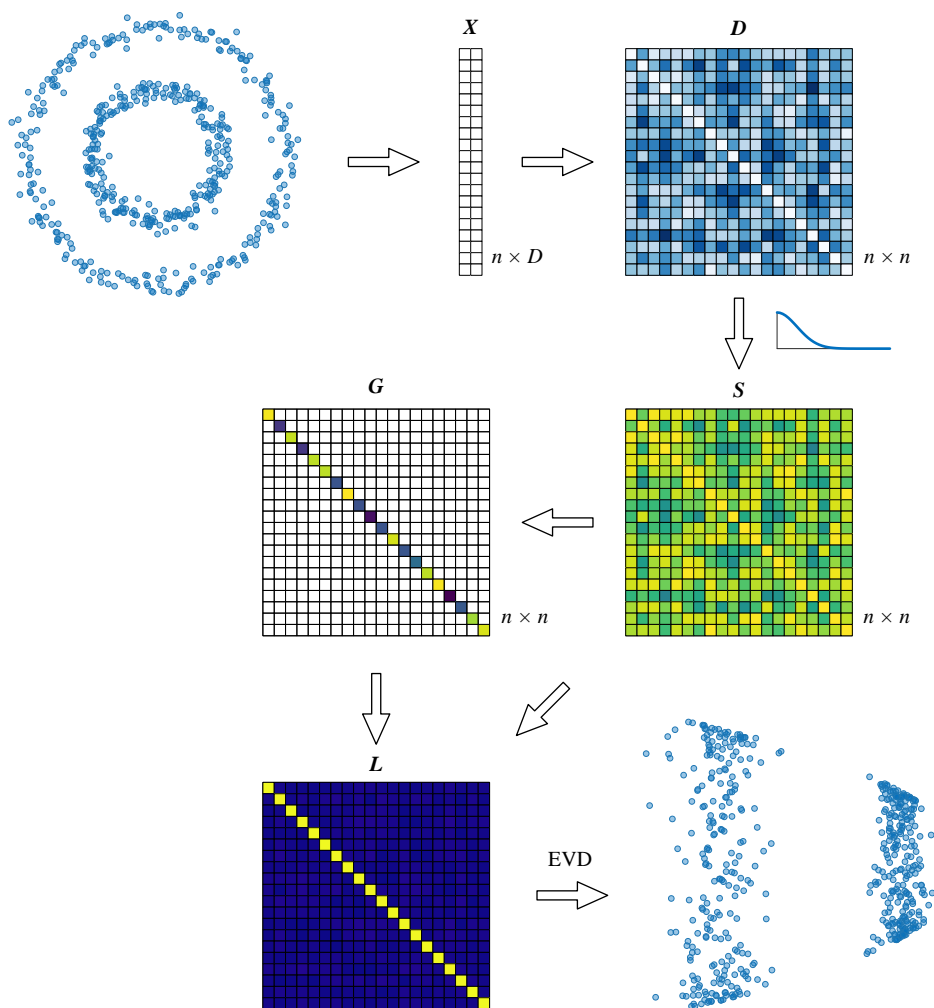
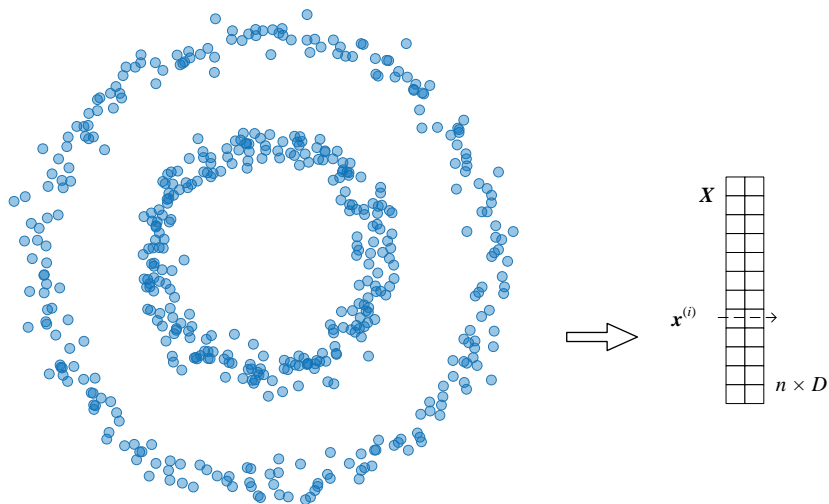


图 2. 谱聚类算法流程

## 23.2 距离矩阵

图 3 所示为样本数据 (500 个数据点) 在平面上位置, 我们可以发现这组数据有两个环; 谱聚类要做的就是尽量把大环、小环的数据分别聚成两簇。



本 PDF 文件为作者草稿, 发布目的为方便读者在移动终端学习, 终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有, 请勿商用, 引用请注明出处。

代码及 PDF 文件下载: <https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教, 本书专属邮箱: [jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

图 3. 样本点平面位置

代码 1 绘制图 3 散点图，下面聊聊其中关键语句。

- a 设定随机数种子，保证结果可复刻。
- b 用 `sklearn.datasets.make_circles()` 生成环形数据。
- c 用 `sklearn.preprocessing.StandardScaler.fit_transform()` 标准化特征数据。
- d 用 `matplotlib.pyplot.scatter()` 绘制散点图。

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from scipy.linalg import sqrtm as sqrtm

# 生成样本数据
a np.random.seed(0)

n_samples = 500;
# 样本数据的数量

b dataset = datasets.make_circles(n_samples=n_samples,
                                  factor=.5, noise=.05)

# 生成环形数据

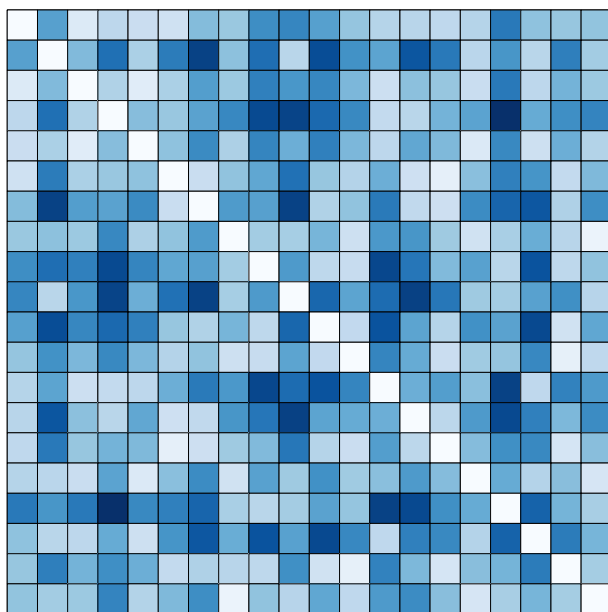
X, y = dataset
# X 特征数据, y 标签数据
c X = StandardScaler().fit_transform(X)
# 标准化数据集

# 可视化散点
fig, ax = plt.subplots(figsize = (6,6))

d plt.scatter(X[:,0],X[:,1])
ax.set_aspect('equal', adjustable='box')
plt.savefig('散点图.svg')
```

代码 1. 生成样本数据 |  Bk6\_Ch23\_01.ipynb

下面计算数据的成对距离矩阵  $D$ 。色块颜色越浅，说明距离越近；色块颜色越深，说明距离越远。注意，为了方便可视化，图 4 热图仅仅展示一个  $20 \times 20$  距离矩阵。

图 4. 成对欧氏距离矩阵  $D$ 

代码 2 绘制图 4，下面聊聊其中关键词句。

**a** 用 `numpy.linalg.norm()` 计算成对距离矩阵。其中，`numpy.newaxis()` 增加一个维度，这样相减时可以利用广播原则得到成对行向量之差。参数 `axis = 2` 保证计算范数时结果为二维数组。

大家也可以用 `scipy.spatial.distance_matrix()` 或 `sklearn.metrics.pairwise_distances()` 计算成对欧氏距离矩阵。

**b** 用 `seaborn.heatmap()` 绘制成对欧氏距离矩阵热图。

```
# 计算成对距离矩阵
a D = np.linalg.norm(X[:, np.newaxis, :] - X, axis=2)
# 请尝试使用
# scipy.spatial.distance_matrix()
# sklearn.metrics.pairwise_distances()

# 可视化成对距离矩阵
plt.figure(figsize=(8,8))
b sns.heatmap(D, square = True,
              cmap = 'Blues',
              # annot=True, fmt=".3f",
              xticklabels = [],
              yticklabels = [])
# plt.savefig('成对距离矩阵_heatmap.svg')
```

代码 2. 成对欧氏距离矩阵 |  Bk6\_Ch23\_01.ipynb

## 23.3 相似度

然后利用  $d_{ij}$  计算  $i$  和  $j$  两点的相似度  $s_{ij}$ ，“距离  $\rightarrow$  相似度”的转换采用高斯核函数：

$$s_{i,j} = \exp\left(-\left(\frac{d_{i,j}}{\sigma}\right)^2\right) \quad (1)$$

相似度取值区间为  $(0, 1]$ 。

两个点距离越近，它们的相似性越高，越靠近 1；反之，距离越远，相似度越低，越靠近 0。任意点和自身的距离为 0，因此对应的相似度为 1。

参数  $\sigma$  可调节，图 5 所示为参数  $\sigma$  对 (1) 高斯函数的影响。

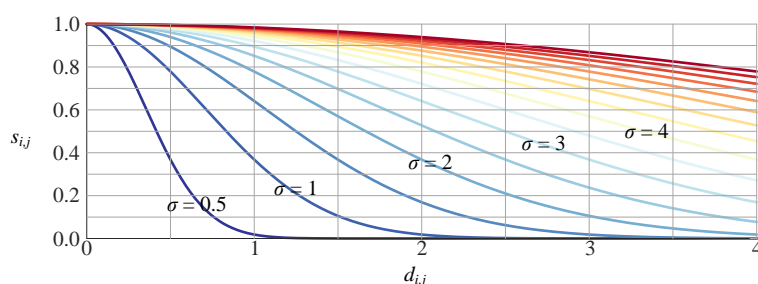


图 5. 参数  $\sigma$  对高斯函数的影响

图 4 所示成对距离矩阵转化为图 6 所示**相似度矩阵** (similarity matrix)  $S$ ；如果用相似度矩阵构造无向图的话，那么矩阵  $S$  就是**邻接矩阵** (adjacency matrix)。

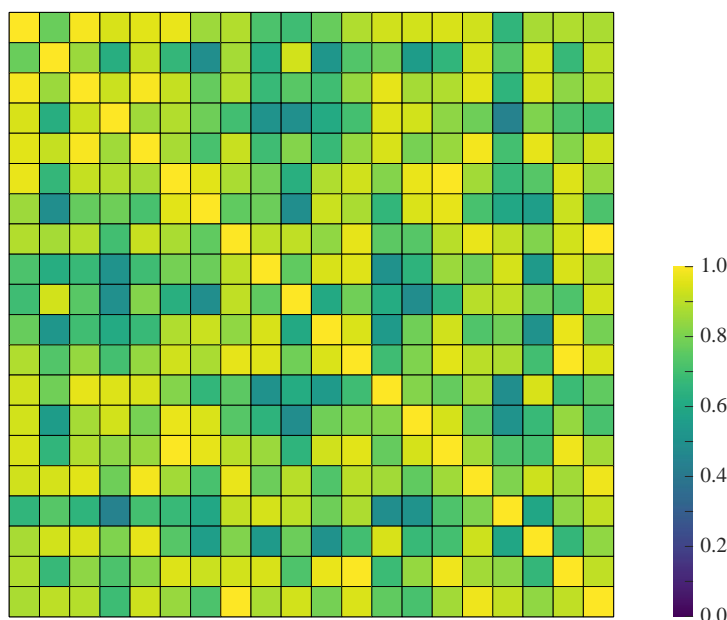


图 6. 成对相似度矩阵  $S$

代码 3 将欧氏距离矩阵转化成相似度矩阵，这个矩阵用作无向图的邻接矩阵。下面聊聊其中关键词句。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger：<https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

- a 自定义函数通过高斯函数将欧氏距离矩阵转化为相似度矩阵。
- b 调用函数，并将 `sigma` 设为 3（默认值为 1）。


```

# 自定义高斯核函数
a def gaussian_kernel(distance, sigma=1.0):
    return np.exp(-(distance ** 2) / (2 * sigma ** 2))

b S = gaussian_kernel(D, 3)
# 参数sigma设为3

# 可视化亲密度矩阵
plt.figure(figsize=(8,8))
sns.heatmap(S, square = True,
            cmap = 'viridis', vmin = 0, vmax = 1,
            # annot=True, fmt=".3f",
            xticklabels = [], yticklabels = [])
# plt.savefig('亲密度矩阵_heatmap.svg')

```

代码 3. 相似度矩阵（用作无向图的邻接矩阵） |  Bk6\_Ch23\_01.ipynb

## 23.4 无向图

图 7 为相似度矩阵  $S$  无向图。图中边的颜色越偏黄，说明两点之间的相似度越高，也就是两点距离越近。为了方便可视化，图中仅仅保留了 80 个节点和它们之间的边。

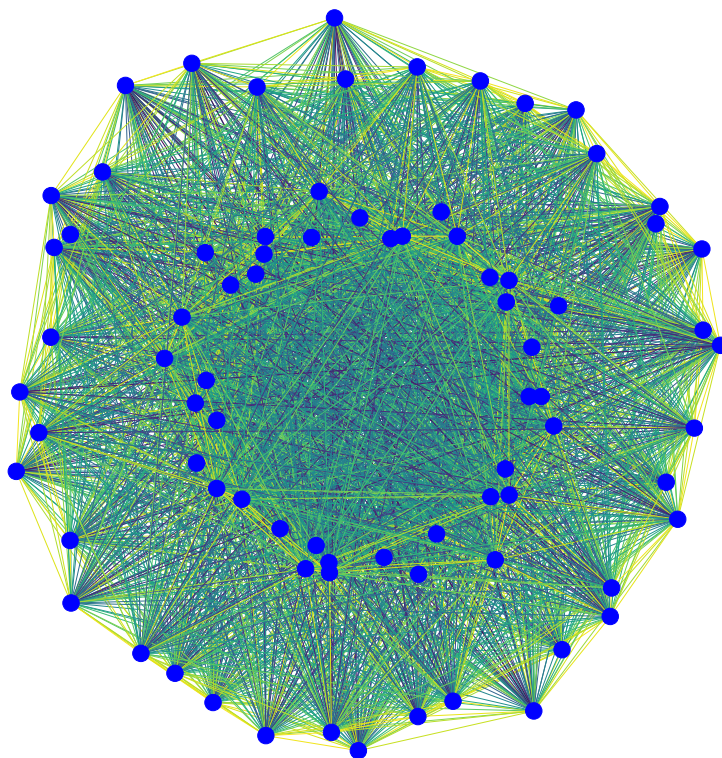


图 7. 相似度对称矩阵  $S$  无向图



代码 4 根据相似度矩阵创建无向图。

- a 用 `numpy.copy()` 创建相似度矩阵副本（不是视图）。
- b 用 `numpy.fill_diagonal()` 将相似度对角线元素置 0，不绘制自环。
- c 用 `networkx.Graph()` 基于相似度矩阵创建无向图。
- d 用 `add_node()` 方法增加节点位置信息。
- e 用 `networkx.get_node_attributes(G, 'pos')` 将节点位置信息取出。
- f 用 `networkx.draw_networkx()` 绘制无向图。根据边的权重值大小用颜色映射 'viridis' 渲染边。

```
# 创建无向图
a S_copy = np.copy(S)
b np.fill_diagonal(S_copy, 0)
c G = nx.Graph(S_copy, nodetype=int)
# 用邻接矩阵创建无向图

# 添加节点和边
d for i in range(len(X)):
    G.add_node(i, pos=(X[i, 0], X[i, 1]))

# 取出节点位置
e pos = nx.get_node_attributes(G, 'pos')

# 增加节点属性
node_labels = {i: chr(ord('a') + i) for i in range(len(G.nodes))}
edge_weights = [G[i][j]['weight'] for i, j in G.edges]

# 可视化图
f fig, ax = plt.subplots(figsize = (6,6))
nx.draw_networkx(G, pos, with_labels=False,
                 node_size=38,
                 node_color='blue',
                 font_color='black',
                 edge_cmap=plt.cm.viridis,
                 edge_color=edge_weights,
                 width=1, alpha=0.5)

ax.set_aspect('equal', adjustable='box')
ax.axis('off')
plt.savefig('成对距离矩阵_无向图.svg')
```

代码 4. 创建无向图 | Bk6\_Ch23\_01.ipynb

## 23.5 拉普拉斯矩阵

为了计算拉普拉斯矩阵，我们首先计算度矩阵。如图 8 所示，**度矩阵** (degree matrix)  $G$  是一个对角阵。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

注意，为了和成对距离矩阵  $D$  区分，本章度矩阵记作  $G$ 。

$G$  的对角线元素是对应相似度矩阵  $S$  对应列元素之和，即：

$$G_{i,i} = \sum_{j=1}^n s_{i,j} = \text{diag}(I^T S) \quad (2)$$

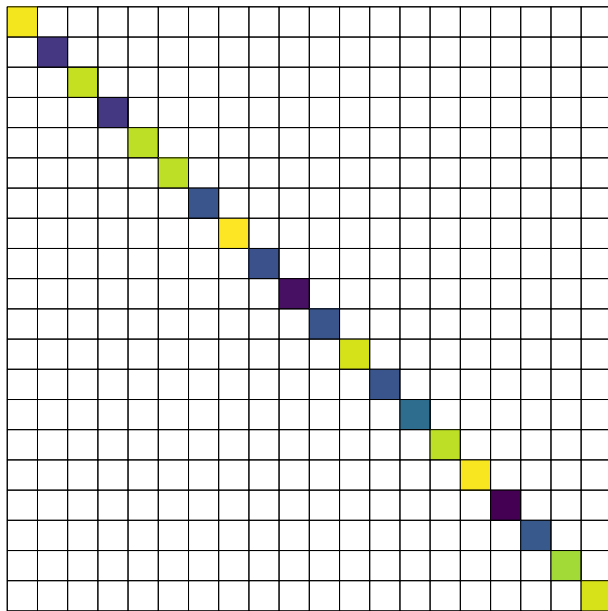


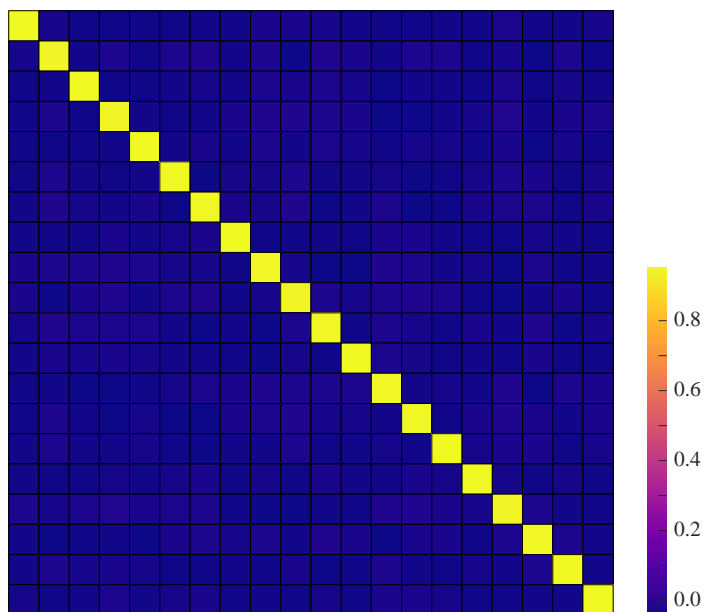
图 8. 度矩阵  $G$

然后构造拉普拉斯矩阵 (Laplacian matrix)  $L$ 。

本章采用的是归一化对称拉普拉斯矩阵 (normalized symmetric Laplacian matrix)，也叫做 Ng-Jordan-Weiss 矩阵，具体如下：

$$L_s = G^{-1/2} (G - S) G^{-1/2} \quad (3)$$

结果如图 9 所示。



本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

图 9. 归一化拉普拉斯矩阵  $L_s$ 

代码 5 计算度矩阵和归一化拉普拉斯矩阵，下面聊聊其中关键语句。


- a 邻接矩阵（相似度矩阵）列方向求和得到节点度数，然后再用 `numpy.diag()` 将其展成度矩阵（对角方阵）。
- b 先用 `numpy.linalg.inv()` 对度矩阵求逆，然后再用 `scipy.linalg.sqrtm()` 开平方。
- c 计算归一化拉普拉斯矩阵。大家也可以用 `networkx.normalized_laplacian_matrix()` 计算归一化拉普拉斯矩阵，并比较结果。

```
a G = np.diag(S.sum(axis = 1))
# 度矩阵

# 可视化度矩阵
plt.figure(figsize=(8,8))
sns.heatmap(G, square = True,
             cmap = 'viridis',
             # linecolor = 'k',
             # linewidths = 0.05,
             mask = 1-np.identity(len(G)),
             vmin = S.sum(axis = 1).min(),
             vmax = S.sum(axis = 1).max(),
             # annot=True, fmt=".3f",
             xticklabels = [], yticklabels = [])
# plt.savefig('度矩阵_heatmap.svg')

b G_inv_sqr = sqrtm(np.linalg.inv(G))
c L_s = G_inv_sqr @ (G - S) @ G_inv_sqr
# 计算归一化（对称）拉普拉斯矩阵

# 可视化拉普拉斯矩阵
plt.figure(figsize=(8,8))
sns.heatmap(L_s, square = True,
             cmap = 'plasma',
             # annot=True, fmt=".3f",
             xticklabels = [], yticklabels = [])
# plt.savefig('拉普拉斯矩阵_heatmap.svg')
```

代码 5. 计算度矩阵和归一化拉普拉斯矩阵 |  Bk6\_Ch23\_01.ipynb

## 23.6 特征值分解

对拉普拉斯矩阵  $L$  进行特征值分解：

$$L = V\Lambda V^{-1} \quad (4)$$

其中

$$A = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \\ & & & \lambda_{12} \end{bmatrix}, V = [v_1 \ v_2 \ \dots \ v_{12}] \quad (5)$$

图 10 所示为拉普拉斯矩阵  $L$  特征值分解得到的前 50 个特征值从小到大排序。

图 11 所示为前两个特征向量构成的散点图，容易发现大环、小环已经分成两簇。

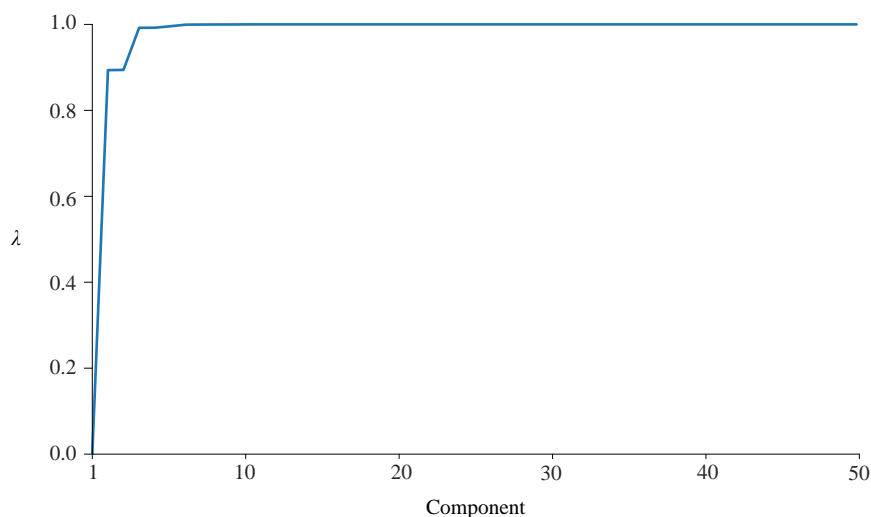


图 10. 拉普拉斯矩阵  $L$  特征值分解得到的特征值从小到大排序，前 50

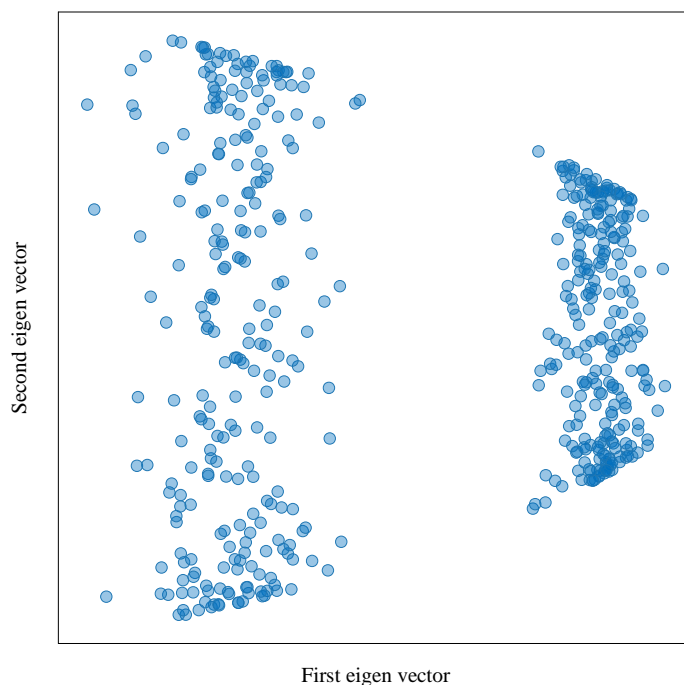


图 11. 前两个特征向量构成的散点图，特征值从小到大排序

代码 6 对归一化拉普拉斯矩阵进行特征值分解 (谱分解)，下面聊聊其中关键语句。

本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

版权归清华大学出版社所有，请勿商用，引用请注明出处。

代码及 PDF 文件下载：<https://github.com/Visualize-ML>

本书配套微课视频均发布在 B 站——生姜 DrGinger: <https://space.bilibili.com/513194466>

欢迎大家批评指教，本书专属邮箱：[jiang.visualize.ml@gmail.com](mailto:jiang.visualize.ml@gmail.com)

- a 用 `numpy.linalg.eigh()` 对归一化拉普拉斯矩阵进行特征值分解。
- b 按特征值从小到大排序得到排序索引。
- c 对特征值从小到大排序。
- d 对特征向量按对应特征值从小到大顺序排序。
- e 取出前两个特征向量绘制散点图。

```


a eigenValues_s, eigenVectors_s = np.linalg.eigh(L_s)
# 特征值分解

# 按特征值从小到大排序
b idx_s = eigenValues_s.argsort() # [::-1]
c eigenValues_s = eigenValues_s[idx_s]
d eigenVectors_s = eigenVectors_s[:,idx_s]

# 前两个特征向量的散点图
fig, ax = plt.subplots(figsize = (6,6))

e plt.scatter(eigenVectors_s[:,0], eigenVectors_s[:,1])
plt.savefig('散点图, 投影后.svg')

```

代码 6. 特征值分解 |  Bk6\_Ch23\_01.ipynb

本章介绍了一种基于图论的聚类方法——谱聚类。谱聚类用到了本书前文介绍的很多图论概念，比如邻接矩阵、无向图、度矩阵、拉普拉斯矩阵等等。

谱聚类将数据点视作为图中的节点，通过相似度函数构造图的边，形成相似度矩阵。接着，基于这个矩阵计算拉普拉斯矩阵，并对其进行特征分解，选取代表数据结构最重要特性的几个特征向量。最后，用这些特征向量的值作为新的特征空间，对数据点在这个空间中进行传统的聚类算法，以达到聚类目的。《机器学习》还会回顾这种聚类方法。