

19

Matrices of Pairwise Measures

成对度量矩阵

成对距离矩阵、亲密度矩阵、相关性系数矩阵，都是图



吸取昨天的教训，为今天而活，为明天的希望。重要的是不要停止提问。

Learn from yesterday, live for today, hope for tomorrow. The important thing is to not stop questioning.

—— 阿尔伯特·爱因斯坦 (Albert Einstein) | 理论物理学家 | 1879 ~ 1955



```

◀ sklearn.metrics.pairwise.euclidean_distances() 计算成对欧氏距离矩阵
◀ sklearn.metrics.pairwise_distances() 计算成对距离矩阵
◀ metrics.pairwise.linear_kernel() 计算线性核成对亲密度矩阵
◀ metrics.pairwise.manhattan_distances() 计算成对城市街区距离矩阵
◀ metrics.pairwise.paired_cosine_distances(X,Q) 计算 X 和 Q 样本数据矩阵成对余弦距离矩阵
◀ metrics.pairwise.paired_euclidean_distances(X,Q) 计算 X 和 Q 样本数据矩阵成对欧氏距离矩阵
◀ metrics.pairwise.paired_manhattan_distances(X,Q) 计算 X 和 Q 样本数据矩阵成对城市街区距离矩阵
◀ metrics.pairwise.polynomial_kernel() 计算多项式核成对亲密度矩阵
◀ metrics.pairwise.rbf_kernel() 计算 RBF 核成对亲密度矩阵
◀ metrics.pairwise.sigmoid_kernel() 计算 sigmoid 核成对亲密度矩阵

```

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



19.1 成对距离矩阵

看了上一章的内容，大家是否想到成对距离矩阵就可以看做是一个邻接矩阵？

完全图

图 1 给出 12 个样本数据在平面上的位置。相信大家还记得成对距离矩阵 (pairwise distance matrix) 这个概念。图 2 所示 12 个样本数据成对欧氏距离矩阵的热图。图 3 展示如何计算欧氏距离矩阵。

这个欧氏距离矩阵为一个对称矩阵。主对角线上元素为某点和自身的距离，显然距离为 0；非主对角线元素为成对距离。

图 4 所示为基于图 2 的无向图。而这个无向图还是一个完全图 (complete graph)。本书前文介绍过，一个完全图是指每一对不同的节点都有一条边相连，形成了一个全连接的图。换句话说，如果一个无向图中的每两个节点之间都存在一条边，那么这个图就是一个完全图。

下面让我们仔细观察图 4 这幅无向图。

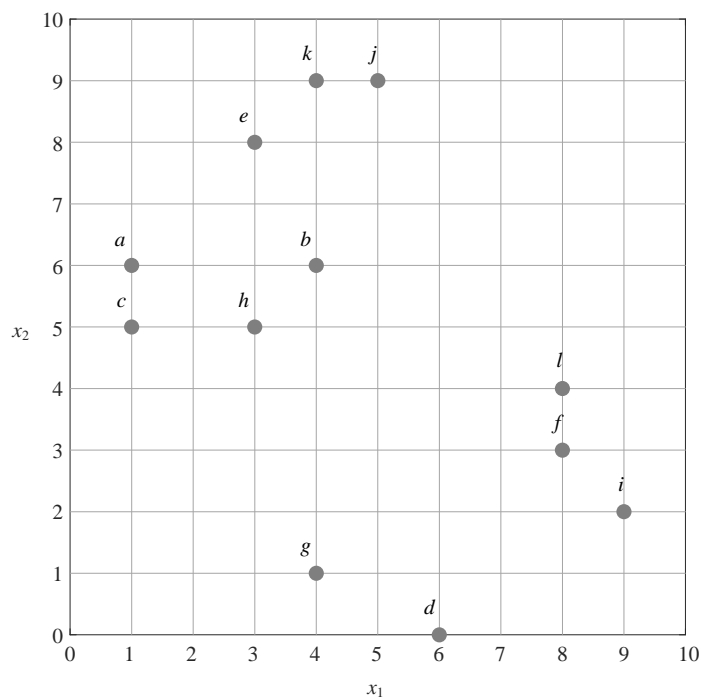


图 1. 12 个样本数据

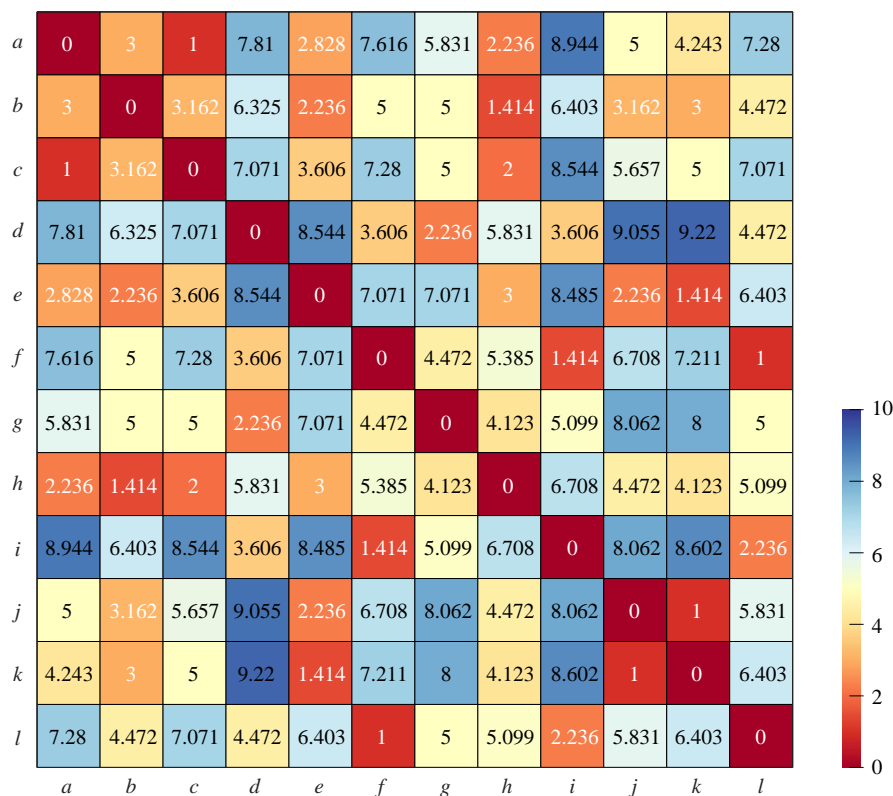


图 2. 12 个样本数据成对距离构成的方阵 heatmap

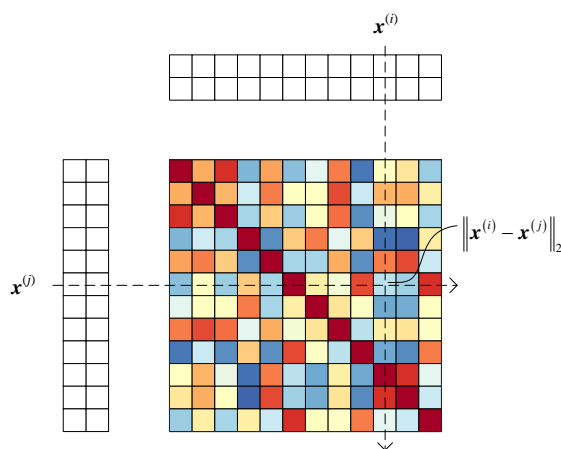


图 3. 计算成对欧氏距离矩阵

图 4 中所有边根据欧氏距离大小用红黄蓝颜色映射渲染。红色表示两点距离近，蓝色表示两点距离远。这幅图还标记了几个成对距离值。

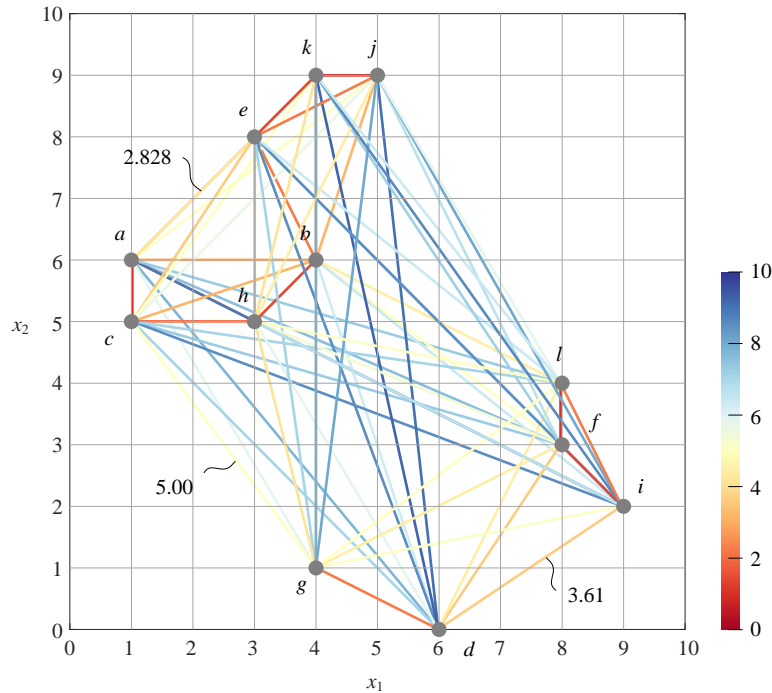


图 4. 基于成对距离矩阵的无向图，完全图

代码 1 绘制图 4，下面聊聊其中关键词句。

- a** 利用广播原则和 `numpy.linalg.norm()` 计算欧氏距离。大家也可以试着使用以下两个函数：`scipy.spatial.distance.pdist()` 和 `sklearn.metrics.pairwise_distances()`。
- b** 用 `seaborn.heatmap()` 绘制成对距离矩阵热图。
- c** 创建无向图实例。
- d** 利用两层 `for` 循环来添加节点、边。请大家思考如何简化这段代码。
- e** 用 `networkx.get_node_attributes()` 获取节点的属性，比如本例中的位置信息。
- f** 创建字典，将节点的整数索引标签转换为小写字母。
- g** 创建节点索引对 `(i, j)` 表示图的一条边，值是格式化的字符串，表示节点之间距离。
- h** 创建列表包含图中所有边的权重。
- i** 用 `networkx.draw_networkx()` 绘制图。其中，`pos` 包含节点位置信息。
`with_labels=True` 表示在绘图中显示节点标签。
`labels=labels` 是一个字典，指定每个节点的标签。它将节点索引与相应的小写字母关联起来。
`edge_vmin=0` 和 `edge_vmax=10` 参数定义了边的颜色的最小和最大值。在这种情况下，边的颜色基于 `edge_weights` 的值。
`edge_cmap=plt.cm.RdYlBu` 指定用于边缘着色的颜色映射。

```

import numpy as np
import networkx as nx
import matplotlib.pyplot as plt
import seaborn as sns

# 12个坐标点
points = np.array([[1,6],[4,6],[1,5],[6,0],
                   [3,8],[8,3],[4,1],[3,5],
                   [9,2],[5,9],[4,9],[8,4]])

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

plt.scatter(points[:,0],points[:,1])
ax.set_xlim(0,10); ax.set_ylim(0,10); ax.grid()
ax.set_aspect('equal', adjustable='box')

# 计算成对距离矩阵
a D = np.linalg.norm(points[:, np.newaxis, :] - points, axis=2)
# 请尝试使用
# scipy.spatial.distance.pdist()
# 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 = 'RdYlBu', vmin = 0, vmax = 10,
               xticklabels = [], yticklabels = [])

# 创建无向图
c G = nx.Graph()

# 添加节点和边
d for i in range(12):
    G.add_node(i, pos=(points[i, 0], points[i, 1]))
    # 使用pos属性保存节点的坐标信息
    for j in range(i + 1, 12):
        G.add_edge(i, j, weight=D[i, j])
    # 将距离作为边的权重
# 请思考如何避免使用 for 循环

# 增加节点/边属性
e pos = nx.get_node_attributes(G, 'pos')
f labels = {i: chr(ord('a') + i) for i in range(len(G.nodes))}
g edge_labels = {(i, j): f'{D[i, j]:.2f}' for i, j in G.edges}
h edge_weights = [G[i][j]['weight'] for i, j in G.edges]

# 可视化图
i fig, ax = plt.subplots(figsize = (6,6))
nx.draw_networkx(G, pos, with_labels=True,
                 labels=labels, node_size=100,
                 node_color='grey', font_color='black',
                 edge_vmin = 0, edge_vmax = 10,
                 edge_cmap=plt.cm.RdYlBu,
                 edge_color=edge_weights, width=1, alpha=0.7)

ax.set_xlim(0,10); ax.set_ylim(0,10); ax.grid()
ax.set_aspect('equal', adjustable='box')

```

代码 1. 将成对距离矩阵转化为完全图 | Bk6_Ch19_01.ipynb

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

设定阈值

图4这幅图的12个散点似乎可以分为两簇(cluster)。而欧氏距离大小就可以帮我们“切割”!

如图5所示为欧氏距离截断阈值设置为6的图;也就是说,超过6的边全部删除,保留不超过6的边。这幅图中两簇散点似乎还有点“藕断丝连”。图6所示为截断阈值对邻接矩阵的影响。

进一步将截断阈值收缩到4,我们便得到图7这幅图。这幅图中数据被分割成两簇。

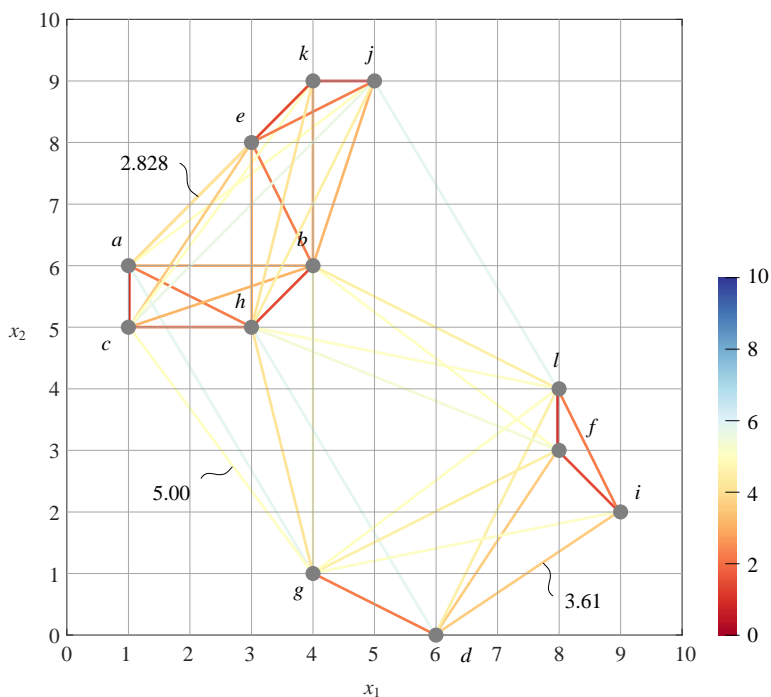


图5. 基于成对距离矩阵的无向图, 截断阈值 = 6

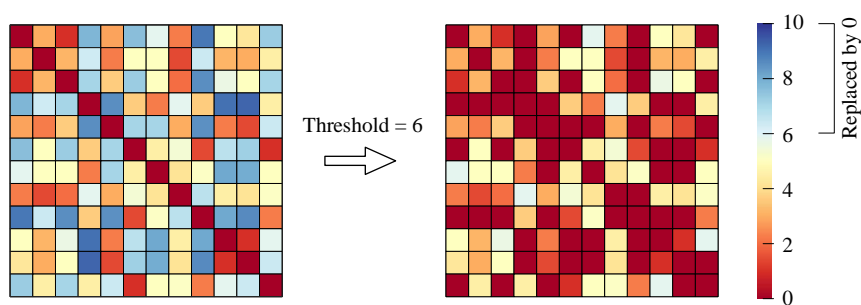


图6. 截断阈值为6对成对欧氏距离矩阵影响

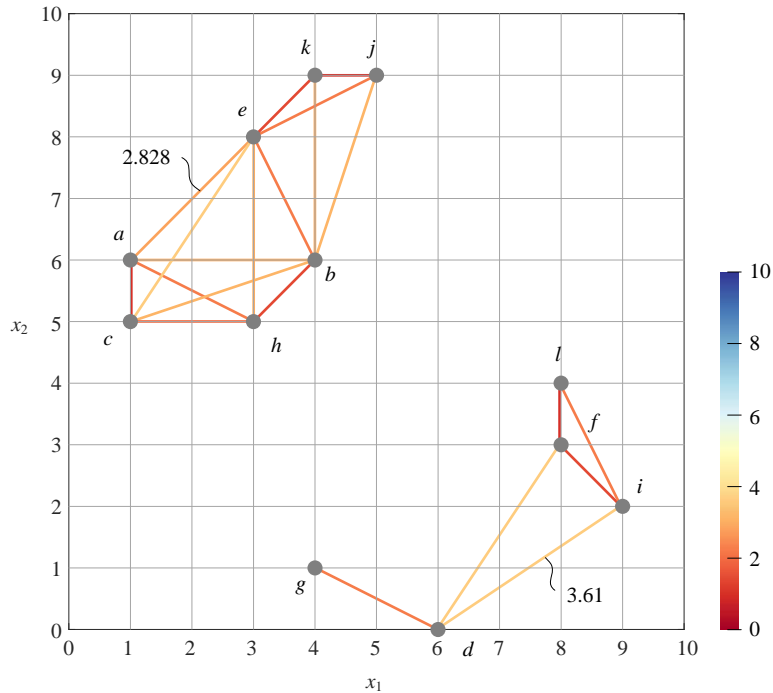


图 7. 基于成对距离矩阵的无向图，截断阈值 = 4

代码 2 对代码 1 稍微改进，请大家自行比较异同。

```
# 计算成对距离矩阵
a D = np.linalg.norm(points[:, np.newaxis, :] - points, axis=2)

# 设定阈值
threshold = 6
D_threshold = D
b D_threshold[D_threshold > threshold] = 0
# 超过阈值置零

# 创建无向图
c G_threshold = nx.Graph(D_threshold, nodetype=int)
# 用邻接矩阵创建无向图

# 添加节点和边
d for i in range(12):
    G_threshold.add_node(i, pos=(points[i, 0], points[i, 1]))

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

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

代码 2. 对距离矩阵设定阈值 |  Bk6_Ch19_02.ipynb

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

19.2 亲近度矩阵：高斯核函数

通过高斯核函数，我们可以很容易把距离转化为“亲近度”

$$\exp\left(-\frac{d_{i,j}^2}{2\sigma^2}\right) = \exp\left(-\frac{\|\mathbf{x}^{(i)} - \mathbf{x}^{(j)}\|_2^2}{2\sigma^2}\right) \quad (1)$$

图 8 所示为参数 σ 对高斯核函数的影响。

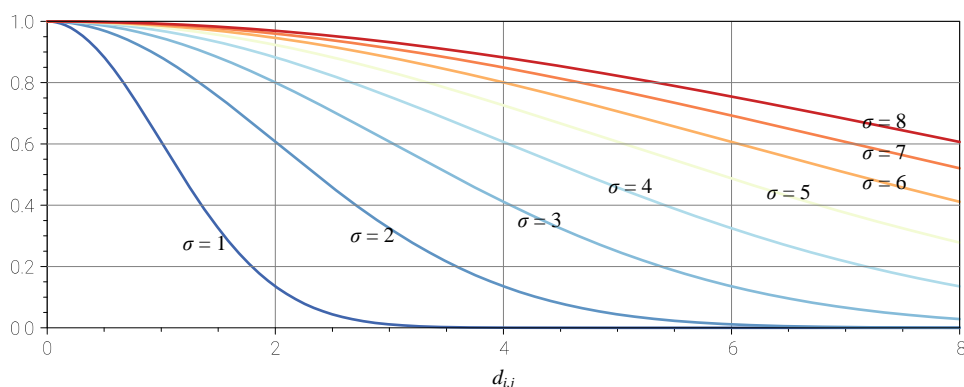


图 8. 将欧氏距离转化为亲近度

图 9 所示为利用高斯核函数将成对欧氏距离矩阵转化为亲近度矩阵。而这个亲近度矩阵可以作为创建无向图的邻接矩阵。

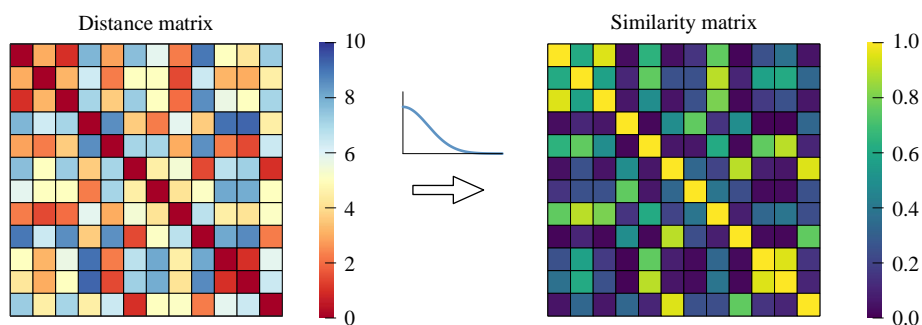


图 9. 成对欧氏距离矩阵转化为亲近度矩阵，高斯核

本例中，我们不绘制自环，因此将亲近度矩阵的对角线元素设置为 0，具体如图 10 所示。

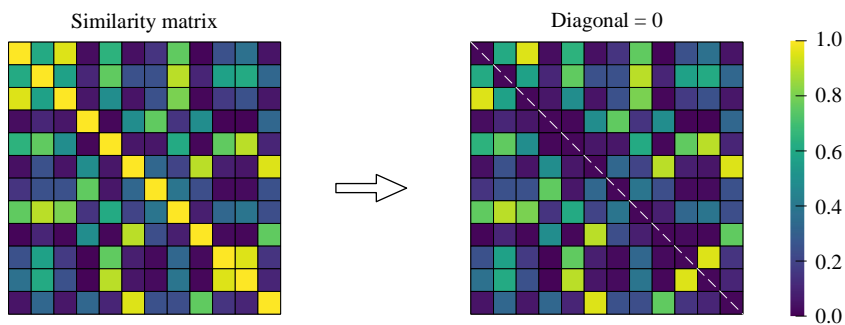


图 10. 亲密度矩阵对角线置 0，不绘制自环

图 11 所以为基于亲密度矩阵绘制的无向图。这幅图中，我们用不同的颜色映射渲染，代表边的权重。

类似上一节，通过设定阈值，我们可以利用亲密度矩阵来“分割”数据点，具体如图 12 和图 13 所示。

这也告诉我们类似成对距离矩阵、亲密度矩阵、协方差矩阵、相关性系数矩阵，都可以看做是无向图的邻接矩阵。请大家特别注意这一观察矩阵的全新视角。

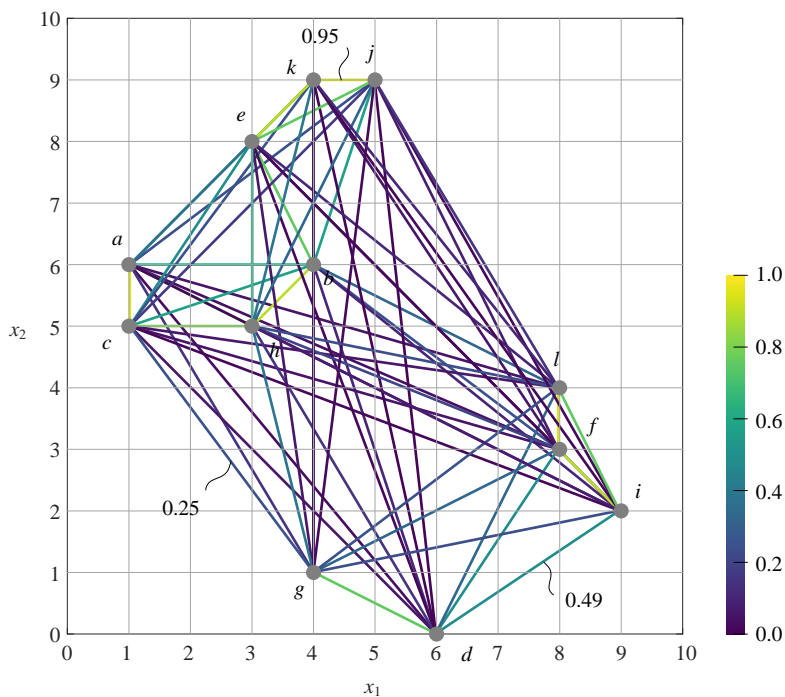


图 11. 基于亲密度矩阵的无向图

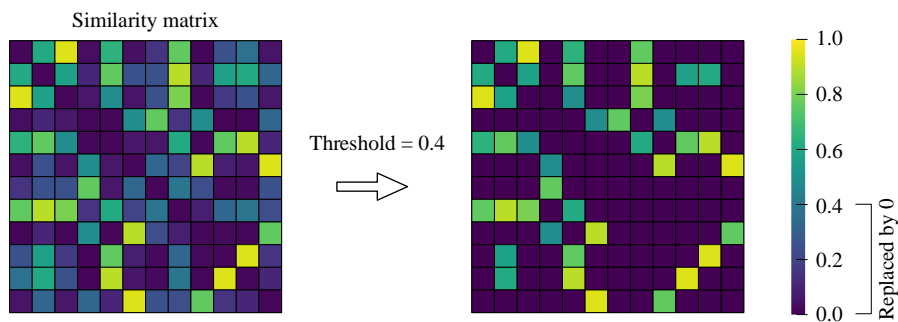


图 12. 阈值 0.4 对亲密度矩阵影响

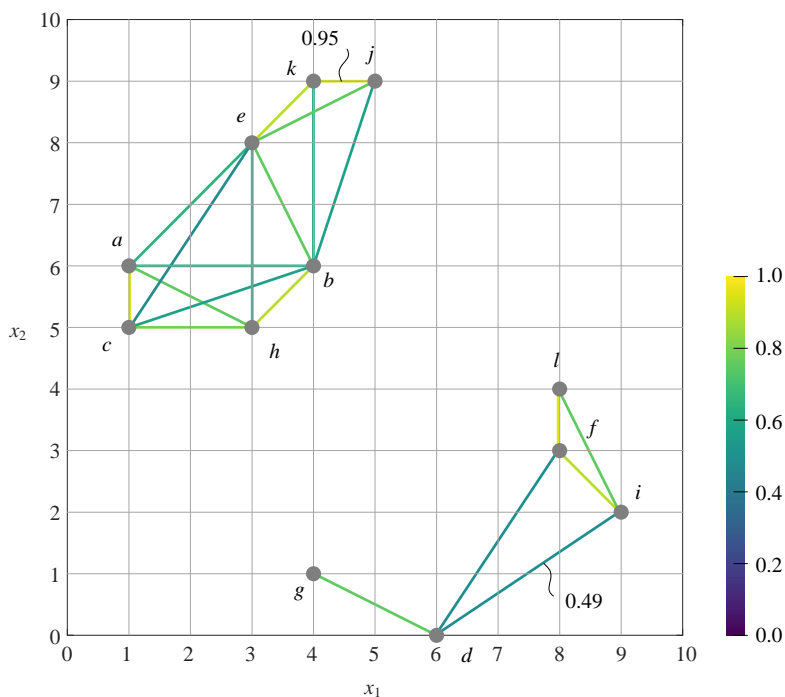


图 13. 基于亲密度矩阵的无向图，设置亲密度阈值为 0.4

Bk6_Ch19_03.ipynb 完成本节示例，下面聊聊代码 3 中关键语句。

- a 定义了高斯核函数，`sigma` 的默认值为 1。
- b 利用自定义高斯核函数将欧氏距离矩阵转换为亲密度矩阵。
- c 利用 `numpy.fill_diagonal()` 将亲密度矩阵对角线元素置 0，因为不需要自环。
- d 利用 `add_node()` 增加节点。
- e 提取节点位置信息。
- f 用 `numpy.copy()` 创建亲密度矩阵副本。
- g 亲密度矩阵中低于阈值的元素置 0。这和上节示例相反，请大家注意。
- h 基于以上亲密度矩阵（邻接矩阵）创建无向图。

```

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

# 计算成对距离矩阵
D = np.linalg.norm(points[:, np.newaxis, :] - points, axis=2)

# 距离矩阵转化为亲密度矩阵，高斯核
b K = gaussian_kernel(D, 3)
# 参数sigma设为3

c np.fill_diagonal(K, 0)
# 将对角线元素置0，不画自环

# 创建无向图
G = nx.Graph(K, nodetype=int)
# 用邻接矩阵创建无向图

# 添加节点和边
d for i in range(12):
    G.add_node(i, pos=(points[i, 0], points[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]
edge_labels = {(i, j): f'{K[i, j]:.2f}' for i, j in G.edges}

# 设定高斯核阈值
threshold = 0.4
f K_threshold = np.copy(K)
# 副本，非视图
g K_threshold[K_threshold < threshold] = 0
# 低于阈值置零，改为小于号


# 创建无向图
h G_threshold = nx.Graph(K_threshold, nodetype=int)
# 用邻接矩阵创建无向图

# 添加节点和边
for i in range(12):
    G_threshold.add_node(i, pos=(points[i, 0], points[i, 1]))

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

# 增加节点属性
node_labels = {i: chr(ord('a') + i) for i in
range(len(G_threshold.nodes))}
edge_weights = [G_threshold[i][j]['weight'] for i, j in G_threshold.edges]
edge_labels = {(i, j): f'{K_threshold[i, j]:.2f}' for i, j in G_threshold.edges}

```

代码 3. 基于亲密度矩阵的无向图 |  Bk6_Ch19_03.ipynb

利用同样的思路，根据亲近度矩阵，我们可以把鸢尾花数据（前两特征，不考虑标签）大致划分成两簇，结果如所示图 14。请大家自行学习 Bk6_Ch19_04.ipynb。

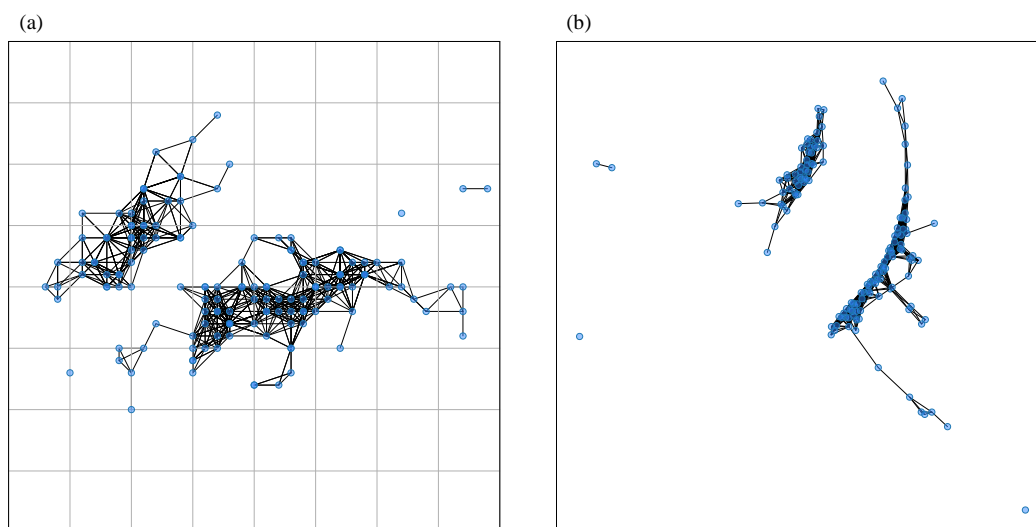


图 14. 用亲近度矩阵划分鸢尾花数据

19.3 相关性系数矩阵

受到前文内容启发，大家是否发现协方差矩阵（图 15）、相关性系数矩阵都可以看做邻接矩阵；也就是说，每个协方差矩阵，每个相关性系数矩阵，都是一副图！

而协方差矩阵、相关性系数矩阵都是特殊的格拉姆矩阵；推而广之，格拉姆矩阵也都可以看做是邻接矩阵，进而从图的视角来观察分析。

本节和大家讨论如何用相关性矩阵构造无向图。

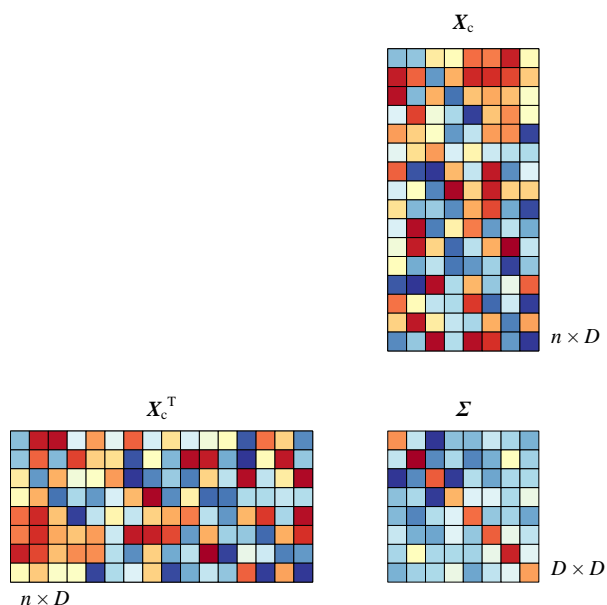


图 15. 协方差矩阵

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

本节采用的将相关性系数矩阵转换为邻接矩阵的规则很简单；举个例子，给定如下相关性系数矩阵

$$\begin{bmatrix} 1 & 0.7 & 0.9 & 0.85 \\ 0.7 & 1 & 0.65 & 0.5 \\ 0.9 & 0.65 & 1 & 0.92 \\ 0.85 & 0.5 & 0.92 & 1 \end{bmatrix} \quad (2)$$

设定阈值为 0.8；如果相关性系数小于 0.8，邻接矩阵对应位置置 0；如果相关性系数不小于 0.8，邻接矩阵相应位置置 1。由于不绘制自环，邻接矩阵对角线元素置 0。因此，(2) 对应的邻接矩阵为

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{bmatrix} \quad (3)$$

我们很容易根据上述邻接矩阵绘制对应的无向图。

Bk6_Ch19_05.ipynb 中加载 428 个有效股价数据；因此，邻接矩阵的大小为 428×428 。

图 16 (a) 所示为基于相关性系数矩阵创建的无向图；图 16 (b) 展示其中最大分量子图。

图 17 则展示其中前 4 大社区；这也相当于对股票的聚类。

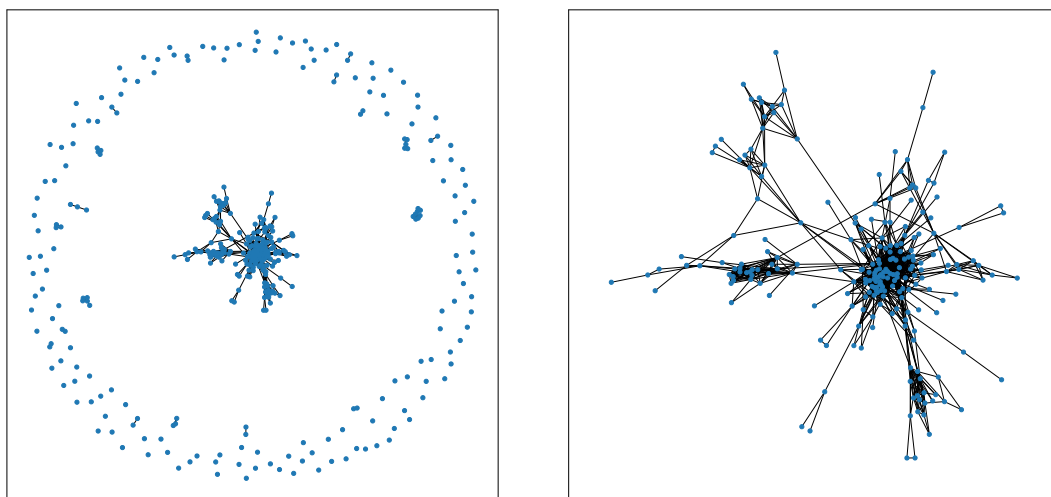


图 16. 基于相关性系数矩阵创建的无向图，阈值为 0.8

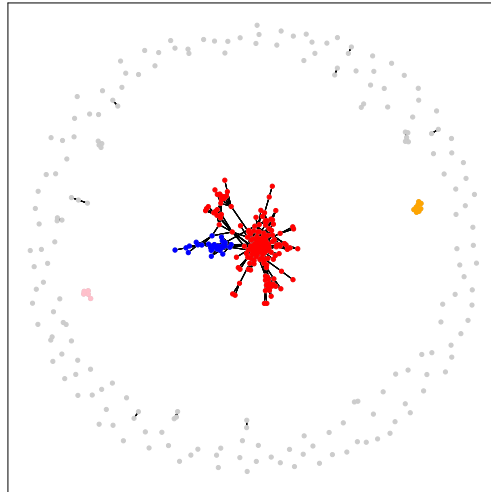


图 17. 无向图中前 4 大社区，阈值为 0.8

Bk6_Ch19_05.ipynb 有完整运算代码，下面仅仅聊聊代码 4 中关键语句。

- a 用 `pandas.read_pickle()` 加载 `.pkl` 数据，本书之前也用这个数据集。
- b 用 `pct_change()` 方法计算股票收盘价日收益率。
- c 用 `dropna()` 方法将整列、整行都为 `NaN` 的删除。
- d 用 `corr()` 方法计算日收益率的相关性系数矩阵。
- e 按前文介绍的映射规则，将相关性系数矩阵转化为邻接矩阵。
- f 将邻接矩阵对角线元素置 0，不画自环。
- g 用 `networkx.from_numpy_array()` 基于邻接矩阵创建无向图。
- h 用 `networkx.relabel_nodes()` 将非负整数的节点名称修改为股票代码。
- i 用 `networkx.connected_components()` 提取无向图中连通分量，将其中最大连通分量取出；然后用 `subgraph()` 方法构造子图。
- j 用 `networkx.algorithms.community centrality.girvan_newman()` 将图划分成社区。
- k 取出各个社区的节点，结果为嵌套列表，子列表元素为社区节点。
- l 根据子列表长度由大到小（社区由大到小）排列嵌套列表元素。

Bk6_Ch19_05.ipynb 有可视化函数，请大家自行学习。此外，请大家修改相关性系数阈值（比如 0.7、0.9）并观察无向图变化。

```

# 加载数据
a df = pd.read_pickle('stock_levels_df_2020.pkl')

# 计算日收益率
b returns_df = df['Adj Close'].pct_change()

# 整列、整行都为NaN的删除
c returns_df.dropna(axis = 1, how='all', inplace = True)
  returns_df.dropna(axis = 0, how='all', inplace = True)

# 计算相关性系数矩阵
d corr = returns_df.corr()

# 将相关性系数矩阵转换为邻接矩阵
A = corr.copy()

# 设定阈值
threshold = 0.7
# 低于阈值，置0
e A[A < threshold] = 0
# 超过阈值，置1
  A[A >= threshold] = 1

f A = A - np.identity(len(A))
# 将对角线元素置0，不画自环

# 创建图
g G = nx.from_numpy_array(A.to_numpy())

# 修改节点名称
h G = nx.relabel_nodes(G, dict(enumerate(A.columns)))


# 最大连通分量
i Gcc = G.subgraph(sorted(nx.connected_components(G),
                        key=len, reverse=True)[0])

pos_Gcc = {k: pos[k] for k in list(Gcc.nodes())}
# 取出子图节点坐标

# 划分社区
j communities = girvan_newman(G)
  node_groups = []
k for com in next(communities):
    node_groups.append(list(com))

# 按子列表长度（社区）由大到小排列
l node_groups.sort(key=len, reverse = True)

```

代码 4. 基于相关性系数矩阵创建的无向图 |  Bk6_Ch19_04.ipynb

成对距离矩阵、协方差矩阵、相关性系数矩阵可以看作无向图的邻接矩阵，其中邻接矩阵中的元素表示图中节点之间的关系。成对距离矩阵反映节点间的距离或相似度；协方差矩阵描述变量间的线性依

赖性；相关性系数矩阵进一步衡量变量间的关系强度和方向。这些矩阵通过节点间的关系强度，映射出无向图的结构，揭示数据间的内在联系。

矩阵就是图，图就是矩阵！

相信有了这章内容，大家更能领会到这句话的精髓。此外，本书后文将介绍更多和图有关的矩阵。