

13

Visualize Graphs Using NetworkX

图的可视化

用 NetworkX 可视化图



这些无限空间的永恒寂静，让我感到深深恐惧。

The eternal silence of these infinite spaces fills me with dread.

—— 布莱兹·帕斯卡 (Blaise Pascal) | 法国哲学家、科学家 | 1623 ~ 1662



```
networkx.draw_networkx_nodes() 绘制节点
networkx.draw_networkx_labels() 添加节点标签
networkx.draw_networkx_edges() 绘制边
networkx.draw_networkx_edge_labels() 添加边标签
networkx.spring_layout() 使用弹簧模型算法布局节点
networkx.complete_bipartite_graph() 创建完全二分图
networkx.dodecahedral_graph() 创建十二面体图
networkx.greedy_color() 贪心着色算法
networkx.star_graph() 绘制星型图
sklearn.datasets.load_iris() 加载数据
sklearn.metrics.pairwise.euclidean_distances() 计算成对欧氏距离矩阵
```

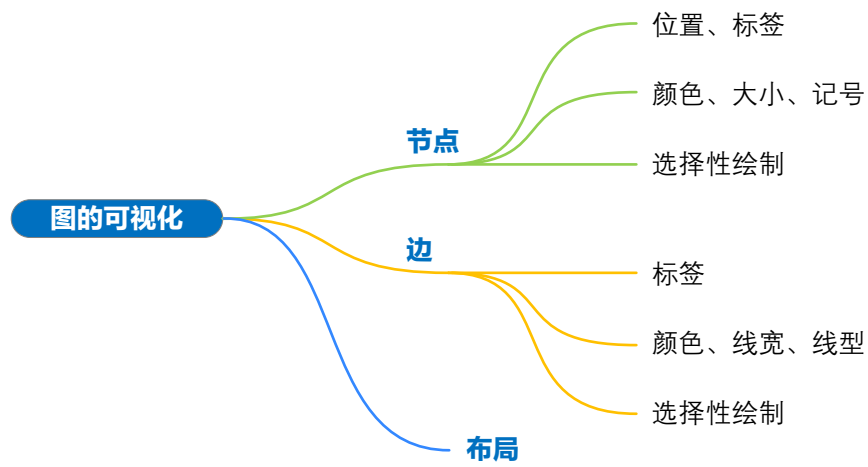
本 PDF 文件为作者草稿，发布目的为方便读者在移动终端学习，终稿内容以清华大学出版社纸质出版物为准。

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 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

13.1 节点位置

本章专门介绍如何用 NetworkX 可视化图。本书前文，我们用 `networkx.draw_networkx()` 绘制过有向图和无向图，本节简单回顾一下这个函数的基本用法。

用 `networkx.draw_networkx()` 绘制图时，我们可以利用参数 `pos` 指定节点位置布局。图 1 (a) 利用 `networkx.spring_layout()` 产生节点位置布局，这个函数使用弹簧模型算法将图的节点布局在平面上，模拟节点间的弹簧力和斥力关系。

图 1 (b) 则直接输入节点位置坐标的字典。这些位置坐标由随机数发生器生成。

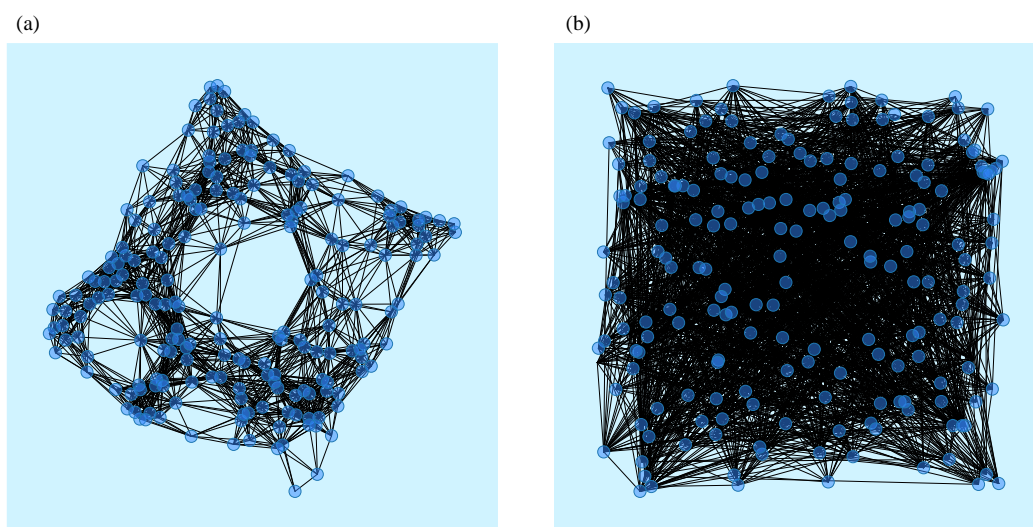


图 1. 使用 `networkx.draw_networkx()` 绘制图，节点位置布局

代码 1 绘制图 1，下面聊聊这段代码的关键语句。

a 用 `networkx.random_geometric_graph()` 创建无向图。第一个参数为节点数，第二个参数为半径大小。

b 利用 `networkx.spring_layout()` 产生节点位置布局。这个函数返回值是一个字典，包含了每个节点的序号 (key) 和平面坐标 (value)。

c 用 `networkx.draw_networkx()` 绘制图，如图 1 (a) 所示。参数 `pos` 控制节点位置。

d 用 `numpy.random.rand()` 生成随机数作为节点坐标点。

e 创建节点坐标字典。

f 再次用 `networkx.draw_networkx()` 绘制图，如图 1 (b) 所示。参数 `pos` 为 **e** 创建的节点坐标字典。

```

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

# 创建无向图
a G = nx.random_geometric_graph(200, 0.2, seed=888)

# 使用弹簧模型算法布局节点
b pos = nx.spring_layout(G, seed = 888)

# 可视化
c plt.figure(figsize = (6,6))
  nx.draw_networkx(G,
                    pos = pos,
                    with_labels = False,
                    node_size = 68)
  plt.savefig('节点布局, 弹簧算法布局.svg')

# 自定义节点位置
d data_loc = np.random.rand(200,2)
# 随机数发生器生成节点平面坐标

# 创建节点位置坐标字典
e pos_loc = {i: (data_loc[i, 0], data_loc[i, 1])
             for i in range(len(data_loc))}

# 可视化
f plt.figure(figsize = (6,6))
  nx.draw_networkx(G,
                    pos = pos_loc,
                    with_labels = False,
                    node_size = 68)
  plt.savefig('节点布局, 随机数.svg')

```


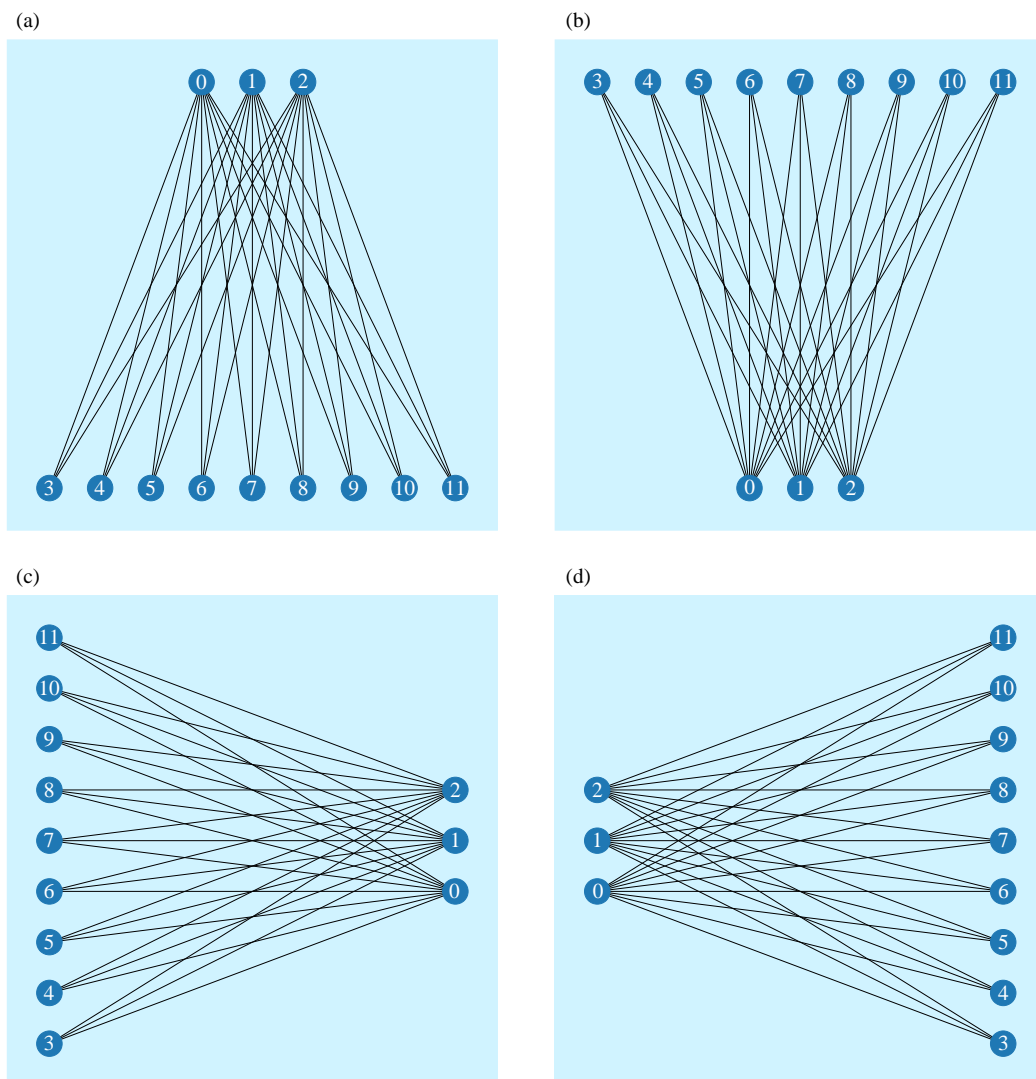
代码 1. 用 networkx.draw_networkx() 绘制图，布置节点 |  Bk6_Ch13_01.ipynb

图 2 所示为利用 networkx.draw_networkx() 绘制的**完全二分图** (complete bipartite graph)。绘制时，通过“节点-坐标”字典设置节点位置。Bk6_Ch13_02.ipynb 绘制图 2，代码相对简单，请大家自行学习。

完全二分图是一种特殊的图，下一章介绍相关内容。

图 2. 使用 `networkx.draw_networkx()` 绘制完全两分图，节点位置布局

类似 `networkx.spring_layout()`，NetworkX 还提供很多其他节点布局方案。图 3 所示为圆周布局和螺旋布局。请大家自行学习 `Bk6_Ch13_03.ipynb`。

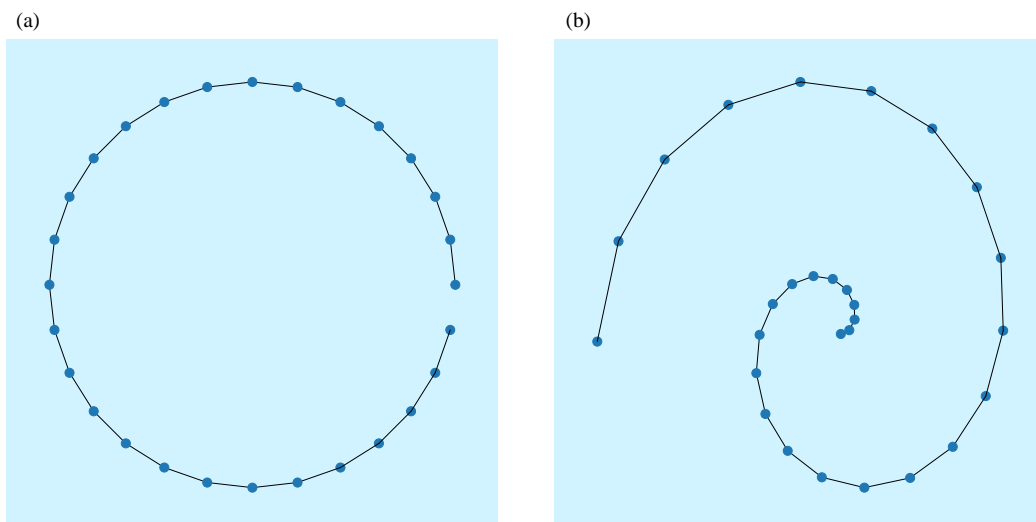


图 3. 圆周布局和螺旋布局

图 4 给出的这个例子，除了编号为 5 的节点之外，其余节点均为圆周布局。这个例子来自 NetworkX，下面聊聊关键语句。

- a 选定需要调整坐标位置节点编号，即编号为 5 的节点。
- b 获取剔除节点 5 的节点子集 `edge_nodes`。
- c 首先用 `G.subgraph(edge_nodes)` 构造子图，然后再用 `networkx.circular_layout()` 获取节点子集的圆周布局，结果为字典。
- d 在 `pos` 字典中增加节点 5 的坐标键值对。

Bk6_Ch13_04.ipynb 还给出另外一个调整节点坐标的方案，请大家对比学习。

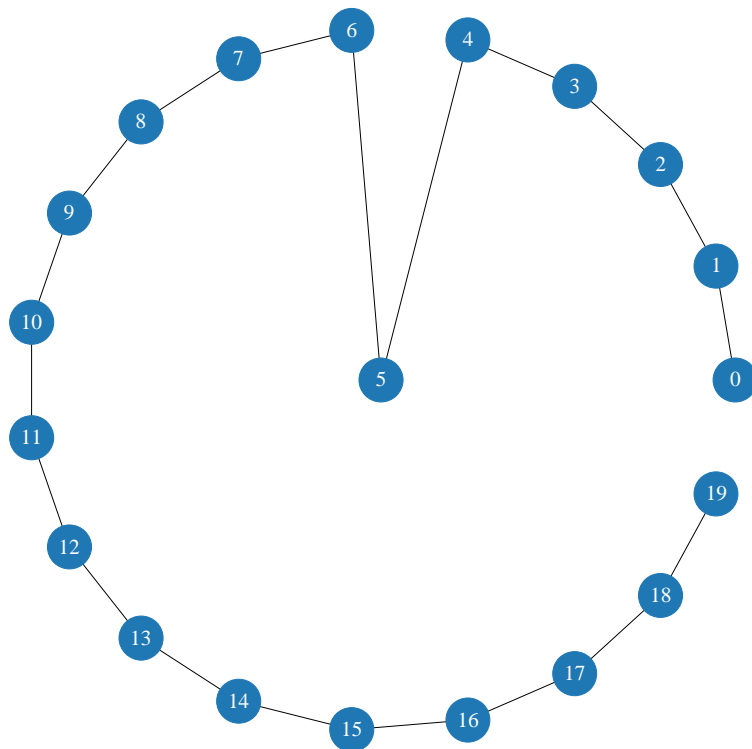


图 4. 调整个别节点位置

```

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

# 创建图
G = nx.path_graph(20)

# 需要调整位置的节点序号
a center_node = 5

# 剩余节点子集
b edge_nodes = set(G) - {center_node}
# {0, 1, 2, 3, 4,
# 6, 7, 8, 9, 10, 11, 12,
# 13, 14, 15, 16, 17, 18, 19}

# 圆周布局 (除了节点5以外)
c pos = nx.circular_layout(G.subgraph(edge_nodes))

# 在字典中增加一个键值对, 节点5的坐标
d pos[center_node] = np.array([0, 0])

# 可视化
plt.figure(figsize = (6,6))
nx.draw_networkx(G, pos, with_labels=True)
plt.savefig('调整节点位置.svg')

```

代码 2. 调整个别节点位置 | Bk6_Ch13_04.ipynb

13.2 节点装饰

图 5 所示为使用 `networkx.draw_networkx()` 绘制图时对节点进行装饰。

图 5 (a) 没有显示节点标签, 调整了节点大小和透明度。

图 5 (b) 调整了节点 marker 类型, 修改了节点颜色。

图 5 (c) 用颜色映射 'RdYlBu_r' 渲染节点颜色。

图 5 (d) 调整了用颜色映射 'hsv' 渲染节点颜色, 同时用随机数控制节点大小。

Bk6_Ch13_05.ipynb 绘制图 5, 代码相对比较简单, 请大家自行学习。

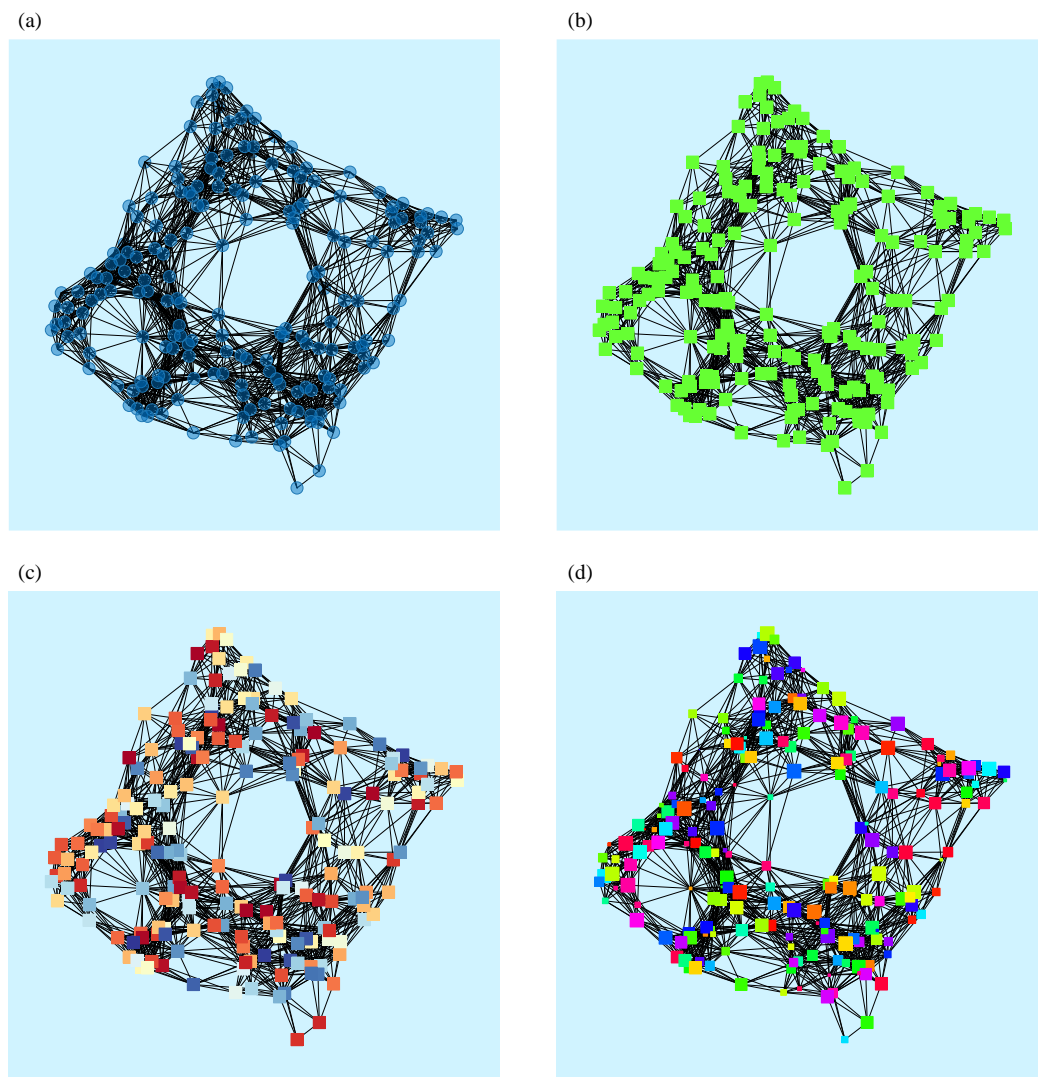


图 5. 使用 `networkx.draw_networkx()` 绘制图，节点装饰

图 6 所示为**十二面体图** (dodecahedral graph)，有 20 个节点、30 条边。《数学要素》介绍过**正十二面体** (dodecahedron) 是**柏拉图立体** (Platonic solid) 的一种；同理，十二面体图也是**柏拉图图** (Platonic graph) 的一种。

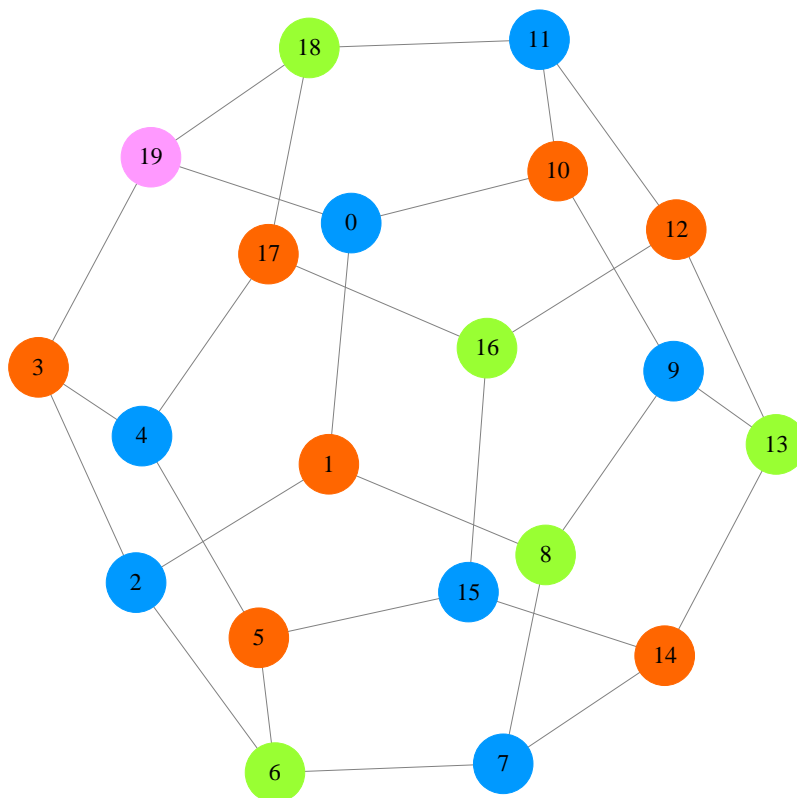


图 6. 使用 `networkx.draw_networkx()` 绘制十二面体图，图着色问题

图 6 所示的十二面体图每两个相邻节点的着色不同；整幅图一共采用了 4 种不同颜色。这幅图展示的实际上是图着色问题。这个问题起源于地图着色，用不同的颜色为地图不同区域着色，要求相邻区域颜色不同，并且整张地图所用颜色种类最少。图 6 这个例子来自 NetworkX 官方，本书对其代码稍作修改。下面聊聊代码 3 的关键语句。

- a 用 `networkx.dodecahedral_graph()` 创建十二面体图。
- b 用 `networkx.greedy_color()` 对十二面体图完成贪心算法着色。
- c 自定义颜色映射，0~3 整数分别对应不同颜色。
- d 完成每个节点的颜色映射，结果为一个列表。
- e 用 `networkx.draw_networkx()` 完成十二面体图的可视化。

```

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

# 创建十二面体图
a G = nx.dodecahedral_graph()

# 贪心着色算法
b graph_color_code = nx.greedy_color(G)

# 独特颜色, {0, 1, 2, 3}
unique_colors = set(graph_color_code.values())


# 颜色映射
c color_mapping = {0: '#0099FF',
                   1: '#FF6600',
                   2: '#99FF33',
                   3: '#FF99FF'}

# 完成每个节点的颜色映射
d node_colors = [color_mapping[graph_color_code[n]]
                 for n in G.nodes()]

# 节点位置布置
pos = nx.spring_layout(G, seed=14)

# 可视化
fig, ax = plt.subplots(figsize = (6,6))
e nx.draw_networkx(
    G,
    pos,
    with_labels=True,
    node_size=500,
    node_color=node_colors,
    edge_color="grey",
    font_size=12,
    font_color="#333333",
    width=2)
plt.savefig('十二面体图, 着色问题.svg')

```

代码 3. 用 `networkx.draw_networkx()` 绘制十二面体图, 图着色问题 |  Bk6_Ch13_06.ipynb

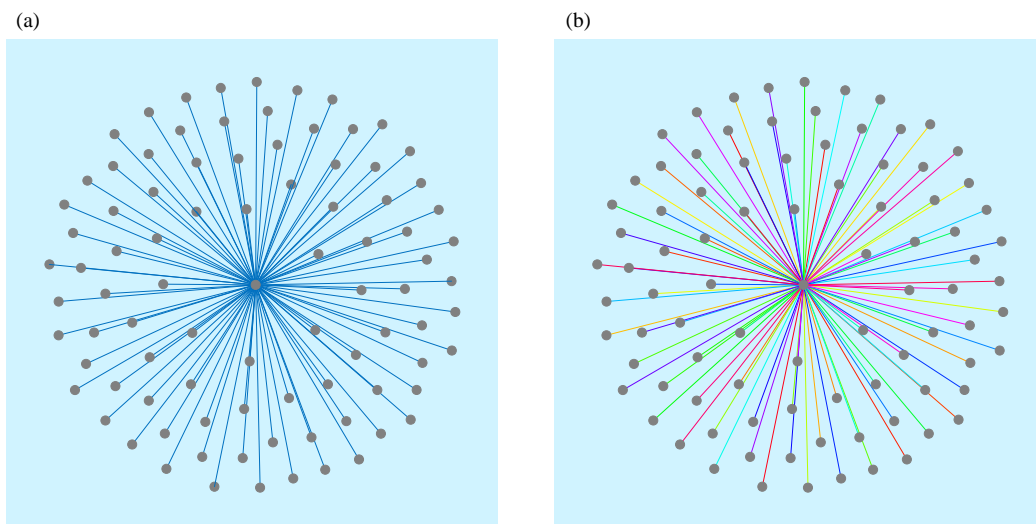
13.3 边装饰

图 7 所示为在用 `networkx.draw_networkx()` 绘制图时对边进行装饰。

图 7 (a) 修改了边的颜色和线宽。请大家自己参考技术文档, 修改边的线型。

图 7 (b) 用颜色映射 'hsv' 渲染边的权重。

Bk6_Ch13_07.ipynb 绘制图 7, 代码相对比较简单, 请大家自行学习。

图 7. 使用 `networkx.draw_networkx()` 绘制图，边装饰

选择性绘制边

图 8 所示为在用 `networkx.draw_networkx()` 绘制图时选择性地绘制边。

图 8 (a) 绘制了所有边，边的权重为两个节点之间的欧氏距离。欧氏距离大的边用暖色渲染，欧氏距离小的边用冷色渲染。

图 8 (b) 则仅仅保留部分边，欧氏距离大于 0.5 的边都被剔除。

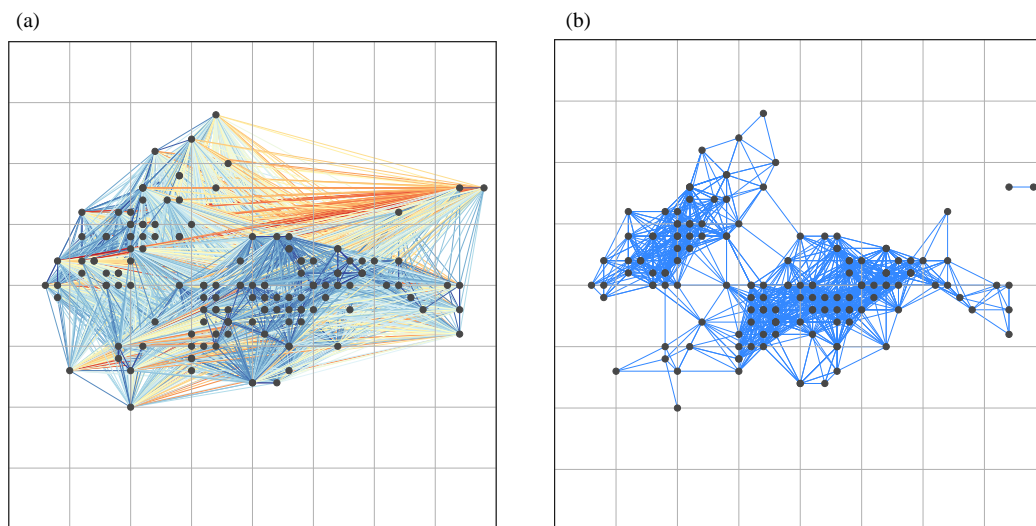


图 8. 鸢尾花数据欧氏距离矩阵的图

图 9 (a) 所示为鸢尾花数据前两个特征的成对欧氏距离矩阵。这幅矩阵便对应图 8 (a)。

图 9 (b) 所示为欧氏距离的直方图。图 9 (a) 这个欧氏距离矩阵行、列数都是 150，一共有 22500 元素；而主对角线元素都是 0，这是散点和自身距离。因此，真正有价值的距离值是刨除主对角线元素的上三角矩阵，或是刨除主对角线的下三角矩阵。而这部分元素一共有 11135 个，即 $150 \times 149 / 2$ 。

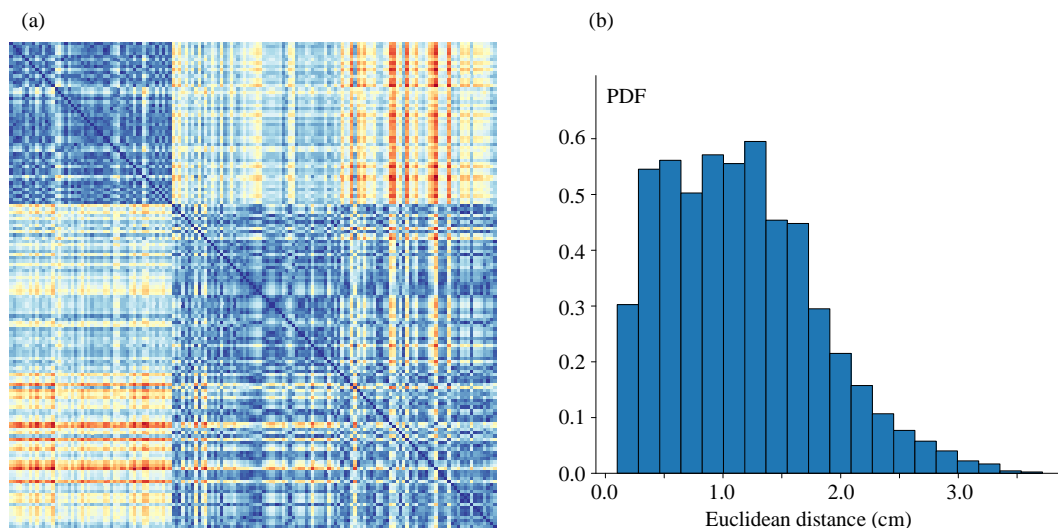


图 9. 欧氏距离矩阵 heatmap, 11135 个欧氏距离的直方图

Bk6_Ch13_08.ipynb 绘制图 8 和图 9，下面聊聊代码 4 中核心语句。

- a 利用 `sklearn.datasets.load_iris()` 加载鸢尾花数据。
- b 取出鸢尾花数据集前两特征。
- c 利用 `sklearn.metrics.pairwise.euclidean_distances()` 计算成对欧氏距离矩阵。
- d 用成对欧氏距离矩阵创建无向图，这是本书下一版块要重点介绍的内容。
- e 提取无向图边的权重，结果为 `list`。这个 `list` 一共有 11135 个元素。
- f 利用鸢尾花样本点在平面上的位置信息创建字典，代表图中节点的位置。
- g 用 `networkx.draw_networkx()` 绘制无向图。参数 `pos` 为节点位置字典；`node_color = '0.28'` 设置节点灰度值；`edge_color = edge_weights` 设置边颜色映射时用的权重值；`edge_cmap = plt.cm.RdYlBu_r` 设定边颜色映射；`linewidths = 0.2` 设定边宽度。
- h 选取需要保留的边，边的权重值（欧氏距离）超过 0.5 都剔除。这句话的结果是一个列表 `list`，列表元素是 `tuple`；每个 `tuple` 有两个元素，代表一条边的两个节点。
- i 用 `networkx.draw_networkx()` 绘制无向图，并通过设定 `edgelist` 保留特定边。

```

import networkx as nx
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.metrics.pairwise import euclidean_distances

# 加载鸢尾花数据集
a iris = load_iris()
b data = iris.data[:, :2]

# 计算欧氏距离矩阵
c D = euclidean_distances(data)
# 用成对距离矩阵可以构造无向图

# 创建无向图
d G = nx.Graph(D, nodetype=int)

# 提取边的权重, 即欧氏距离值
e edge_weights = [G[i][j]['weight'] for i, j in G.edges]

# 使用鸢尾花数据的真实位置绘制图形
f pos = {i: (data[i, 0], data[i, 1]) for i in range(len(data))}

# 绘制无向图, 所有边
g fig, ax = plt.subplots(figsize = (6,6))
nx.draw_networkx(G,
                  pos,
                  node_color = '0.28',
                  edge_color = edge_weights,
                  edge_cmap = plt.cm.RdYlBu_r,
                  linewidths = 0.2,
                  with_labels=False,
                  node_size = 18)

# 选择需要保留的边
h edge_kept = [(u, v)
               for (u, v, d)
               in G.edges(data=True)
               if d["weight"] <= 0.5]

# 绘制无向图, 剔除欧氏距离大于0.5的边
i fig, ax = plt.subplots(figsize = (6,6))
nx.draw_networkx(G,
                  pos,
                  edgelist = edge_kept,
                  node_color = '0.28',
                  edge_color = '#3388FF',
                  linewidths = 0.2,
                  with_labels=False,
                  node_size = 18)

```

代码 4. 用 networkx.draw_networkx() 绘制图, 保留部分边 | Bk6_Ch13_08.ipynb

13.4 分别绘制节点和边

本节将介绍如何利用如下几个函数完成更复杂的图的可视化方案。

- ▶ `networkx.draw_networkx_nodes()` 绘制节点；
- ▶ `networkx.draw_networkx_labels()` 添加节点标签；
- ▶ `networkx.draw_networkx_edges()` 绘制边；
- ▶ `networkx.draw_networkx_edge_labels()` 添加边标签。

本节很多例子都参考 NetworkX 范例，笔者对代码稍作修改。

不同边权重，不同线型、颜色

图 10 所示这幅无向图，对于权重小于 0.5 的边采用蓝色虚线，而权重大于 0.5 的边采用黑色实线。

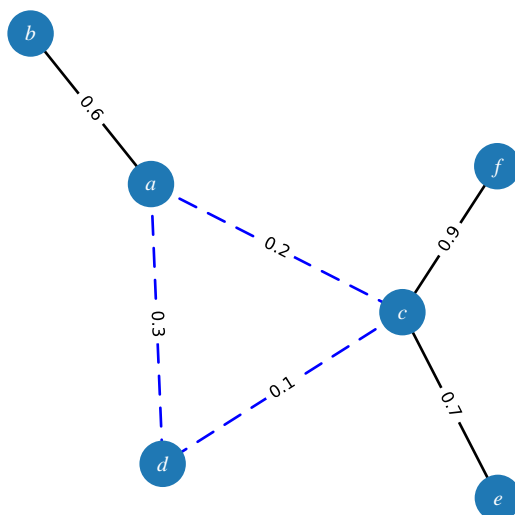


图 10. 不同边权重，不同线型、颜色

图 10 这个例子来自 NetworkX，下面让我们一起分析代码 5。

- a 采用列表生成式，选出权重大于 0.5 的边。
- 同样，b 也是采用列表生成式，选出权重不大于 0.5 的边。这样，我们就把图的所有边分成两组。
- c 用 `networkx.draw_networkx_nodes()` 绘制节点。
- d 用 `networkx.draw_networkx_labels()` 增加节点标签。
- e 用 `networkx.draw_networkx_edges()` 绘制第一组边（权重大于 0.5），颜色（黑色）、线型（实线）均为默认。
- f 用 `networkx.draw_networkx_edges()` 绘制第二组边（权重不大于 0.5），颜色修改为蓝色，线型为划线。
- g 用 `networkx.get_edge_attributes(G, "weight")` 提取图 G 所有边的权重。

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

h 用 `networkx.draw_networkx_edge_labels()` 添加边标签。

```
import matplotlib.pyplot as plt
import networkx as nx

G = nx.Graph()

# 添加边
G.add_edge("a", "b", weight=0.6)
G.add_edge("a", "c", weight=0.2)
G.add_edge("c", "d", weight=0.1)
G.add_edge("c", "e", weight=0.7)
G.add_edge("c", "f", weight=0.9)
G.add_edge("a", "d", weight=0.3)

# 将边分成两组
# 第一组：边权重 > 0.5
a elarge = [(u, v)
             for (u, v, d)
             in G.edges(data=True)
             if d["weight"] > 0.5]

# 第二组：边权重 <= 0.5
b esmall = [(u, v)
            for (u, v, d)
            in G.edges(data=True)
            if d["weight"] <= 0.5]

# 节点布局
pos = nx.spring_layout(G, seed=7)

# 可视化
plt.figure(figsize = (6,6))
# 绘制节点
c nx.draw_networkx_nodes(G, pos, node_size=700)

# 节点标签
d nx.draw_networkx_labels(G, pos,
                        font_size=20,
                        font_family="sans-serif")

# 绘制第一组边
e nx.draw_networkx_edges(G, pos,
                        edgelist=elarge,
                        width=1)

# 绘制第二组边
f nx.draw_networkx_edges(G, pos,
                        edgelist=esmall,
                        width=1,
                        alpha=0.5, edge_color="b",
                        style="dashed")

# 边标签
g edge_labels = nx.get_edge_attributes(G, "weight")
h nx.draw_networkx_edge_labels(G, pos,
                             edge_labels)

plt.savefig('不同边权重，不同线型.svg')
```

代码 5. 不同边权重，不同线型、颜色 | Bk6_Ch13_09.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

展示鸢尾花不同类别

图 11 在图 8 (b) 基础上还可可视化鸢尾花分类标签。

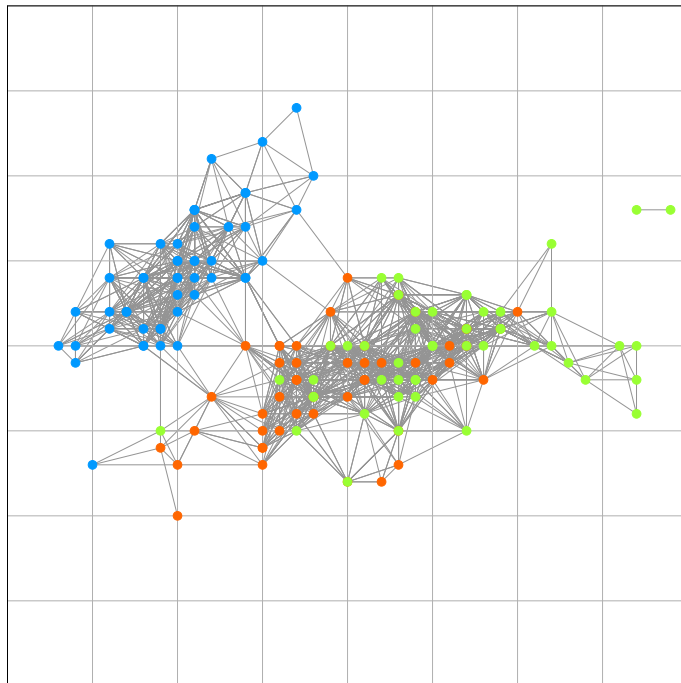


图 11. 鸢尾花数据欧氏距离矩阵的图，展示鸢尾花分类标签

下面聊聊代码 6 中关键词句。

- a** 值得反复强调，这句用成对距离矩阵创建图。这就是本书开篇提到的那句话——图就是矩阵，矩阵就是图！这是本书后续要介绍的重要知识点之一。
- b** 选择要保留的边，前文介绍过这句。
- c** 构造字典用于鸢尾花标签到颜色的映射。
- d** 完成每个节点的颜色映射。
- e** 用 `networkx.draw_networkx_edges()` 绘制保留下来的边。
- f** 用 `networkx.draw_networkx_nodes()` 绘制节点，参数 `node_color` 输入每个节点颜色的字典。


```

# 计算欧氏距离矩阵
D = euclidean_distances(data)
# 用成对距离矩阵可以构造无向图

# 创建无向图
a G = nx.Graph(D, nodetype=int)

# 提取边的权重，即欧氏距离值
edge_weights = [G[i][j]['weight'] for i, j in G.edges]

# 使用鸢尾花数据的真实位置绘制图形
pos = {i: (data[i, 0], data[i, 1]) for i in range(len(data))}

# 选择需要保留的边
b edge_kept = [(u, v)
               for (u, v, d)
               in G.edges(data=True)
               if d["weight"] <= 0.5]

# 节点颜色映射
c color_mapping = {0: '#0099FF',
                  1: '#FF6600',
                  2: '#99FF33'}

# 完成每个节点的颜色映射
d node_color = [color_mapping[label[n]]
               for n in G.nodes()]

# 绘制无向图，分别绘制边和节点

fig, ax = plt.subplots(figsize = (6,6))

e nx.draw_networkx_edges(G, pos,
                       edgelist=edge_kept,
                       width = 0.2,
                       edge_color='0.58')

f nx.draw_networkx_nodes(G, pos, node_size = 18,
                       node_color=node_color)

ax.set_xlim(4,8)
ax.set_ylim(1,5)
ax.grid()
ax.set_aspect('equal', adjustable='box')
plt.savefig('鸢尾花-欧氏距离矩阵-无向图，展示分类标签.svg')

```

代码 6. 展示鸢尾花分类 |  Bk6_Ch13_10.ipynb

有向图

图 12 这幅有向图案例来自 NetworkX，图中节点和边都分别绘制；而且还增加了颜色条，用来展示。

请大家自行学习 Bk6_Ch13_11.ipynb。

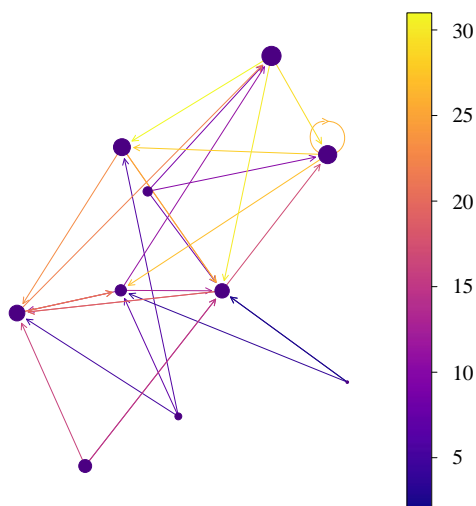


图 12. 有向图

节点标签

图 13 这幅图也是来自 NetworkX，图中每个节点都用 `networkx.draw_networkx_labels()` 增加了自己独特的标签。

请大家自行学习 Bk6_Ch13_12.ipynb。

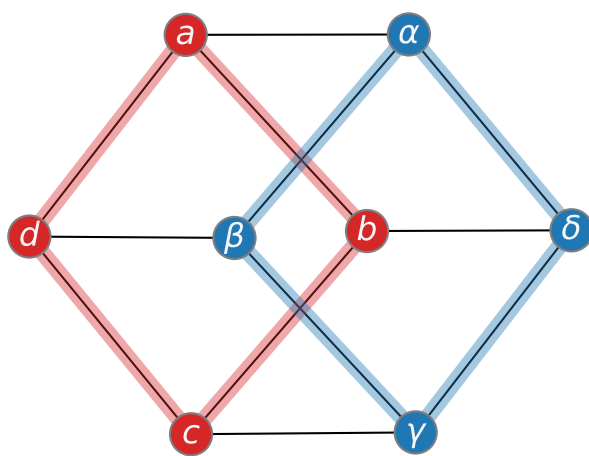


图 13. 节点标签

图 14 给出的图是根据度数大小用颜色映射渲染节点。

代码 7 绘制图 14，下面聊聊其中关键语句。

- a** 用 `networkx.get_node_attributes(G, "pos")` 获取图 G 的节点平面位置坐标信息。
- b** 根据节点度数大小排序。
- c** 将节点度数结果转化为字典。
- d** 创建自定义函数，用来从字典中提取满足特定条件（节点度数等于给定值）的键值对。
- e** 创建集合计算节点度数独特值。

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

- f 每个节点度数独特值对应颜色映射中一个颜色。
- g 用 `networkx.draw_networkx_edges()` 绘制图的边。
- h 中 for 循环每次绘制一组节点，这组节点满足特定节点度数。
- i 用 `networkx.draw_networkx_nodes()` 绘制一组节点，节点大小、节点颜色都和节点度数直接相关。

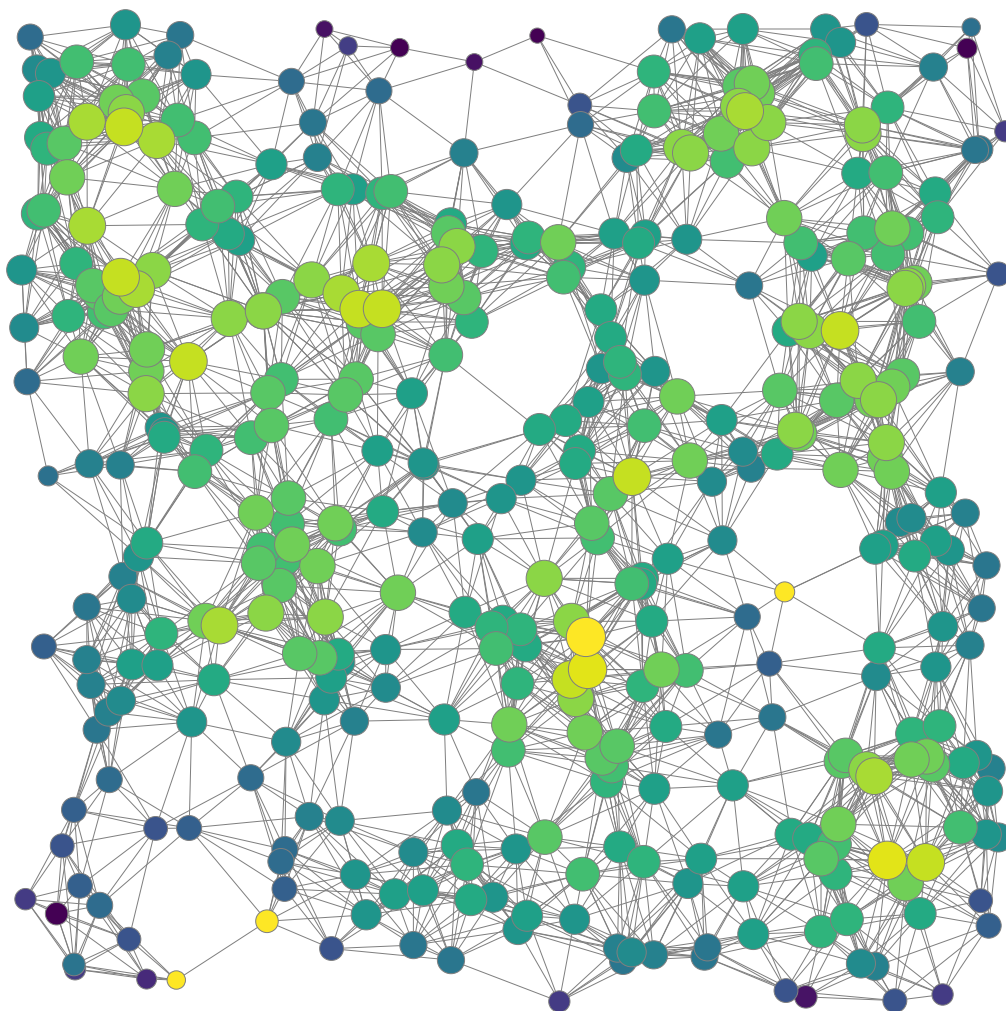


图 14. 根据度数大小用颜色映射渲染节点

```

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

# 产生随机图
G = nx.random_geometric_graph(400, 0.125)

# 提取节点平面坐标
a pos = nx.get_node_attributes(G, "pos")

# 度数大小排序
b degree_sequence = sorted((d for n, d in G.degree()),
                           reverse=True)

# 将结果转为字典
c dict_degree = dict(G.degree())

# 自定义函数，过滤dict
d def filter_value(dict_, unique):
    newDict = {}
    for (key, value) in dict_.items():
        if value == unique:
            newDict[key] = value
    return newDict

e unique_deg = set(degree_sequence)
# 取出节点度数独特值

f colors = plt.cm.viridis(np.linspace(0, 1, len(unique_deg)))
# 独特值的颜色映射

# 可视化
g plt.figure(figsize=(8, 8))
nx.draw_networkx_edges(G, pos, edge_color = '0.8')

# 根据度数大小渲染节点
h for deg_i, color_i in zip(unique_deg, colors):
    dict_i = filter_value(dict_degree, deg_i)
    nx.draw_networkx_nodes(G, pos,
                           node_list = list(dict_i.keys()),
                           node_size = deg_i*8,
                           node_color = color_i)

i plt.axis("off")
plt.savefig('根据度数大小用颜色映射渲染节点.svg')

```

代码 7. 根据度数大小用颜色映射渲染节点 | Bk6_Ch13_13.ipynb

这章介绍了如何用 NetworkX 完成复杂图的可视化。

这章特别引入了一个有趣的知识点——基于欧氏距离矩阵的图！也请大家试图理解这句话——图就是矩阵，矩阵就是图。