



南開大學
Nankai University

数据挖掘实验报告

学 号： 2213117

姓 名： 蔡 佳 良

年 级： 2022 级

学 院： 统计与数据科学学院

专 业： 统 计 学

完成日期： 2024 年 11 月 17 日

目录

1 第一次上机实验 (提取图像的纹理特征)	3
1.1 实验要求	3
1.2 数据分析与处理	3
1.3 实验步骤与原理	3
1.3.1 灰度共生矩阵	3
1.3.2 局部二值模式	4
1.4 实验结果与分析	4
1.5 实验代码	4
2 第二次上机实验	9
2.1 实验要求	9
2.2 数据分析与处理	9
2.3 实验步骤与原理	9
2.4 实验结论与分析	9
2.5 实验代码	9
3 第三次上机实验	10
3.1 实验要求	10
3.2 数据分析与处理	10
3.3 实验步骤与原理	10
3.4 实验结论与分析	10
3.5 实验代码	10
4 第四次上机实验	11
4.1 实验要求	11
4.2 数据分析与处理	11
4.3 实验步骤与原理	11
4.4 实验结论与分析	11
4.5 实验代码	11

5 第五次上机实验	12
5.1 实验要求	12
5.2 数据分析与处理	12
5.3 实验步骤与原理	12
5.4 实验结论与分析	12
5.5 实验代码	12

第一章 第一次上机实验 (提取图像的纹理特征)

1.1 实验要求

- 给定若干张图像，利用灰度共生矩阵特征或局部二值模式特征对这些图像进行特征提取
- 图象是 $W * H * 3$ 的矩阵
- 将最终提取到的特征通过 plot 的形式展示，直观对比不同纹理提取到的特征
- 使用 Python 编程实现

1.2 数据分析与处理

将图像转化为灰度矩阵

1.3 实验步骤与原理

1.3.1 灰度共生矩阵

灰度共生矩阵是从 $N \times N$ 的图像 $f(x, y)$ 的灰度为 i 的像素出发，统计与距离为 $\delta = (dx^2 + dy^2)^{1/2}$ ，灰度为 j 的像素同时出现的概率为，数学表达式为：

$$P(i, j, \delta, \theta) = \{[(x, y), (x + dx, y + dy)] | f(x, y) = i, f(x + dx, y + dy) = j\}$$

分别考虑 $\theta = 0^\circ, 45^\circ, 90^\circ, 135^\circ$,

当 $\theta = 0^\circ$ 时, $dx = 1, dy = 0$;

当 $\theta = 45^\circ$ 时, $dx = 1, dy = -1$;

当 $\theta = 90^\circ$ 时, $dx = 0, dy = -1$;

当 $\theta = 135^\circ$ 时, $dx = -1, dy = -1$ 。

1.3.2 局部二值模式

LBP 算子定义为在 3×3 的窗口内, 以窗口中心像素为阈值, 将相邻的 8 个像素的灰度值与其进行比较, 若周围像素值大于中心像素值, 则该像素点的位置被标记为 1, 否则为 0。这样, 3×3 邻域内的 8 个点经比较可产生 8 位二进制数 (通常转换为十进制数即 LBP 码, 共 256 种), 即得到该窗口中心像素点的 LBP 值, 并用这个值来反映该区域的纹理信息 (例如亮点和暗点), 数学表达式为:

$$LBP = \sum_{p=0}^{P-1} s(g_p - g_c) \cdot 2^p,$$

其中,

- $s(x)$ 是符号函数:

$$s(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$

- P 是邻域像素数 (通常为 8)

1.4 实验结果与分析

通过两种方法下生成的特征图可以看出, 两类图有明显不同的纹理特征。

1.5 实验代码

```
1 import os
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import matplotlib as mpl
5 from PIL import Image, ImageDraw, ImageFont
6
7 mpl.rcParams["font.sans-serif"] = ["SimHei"] # 中文
8 plt.rcParams["axes.unicode_minus"] = False # 正负号
9
10 # 更改当前工作目录为脚本所在目录
11 os.chdir(os.path.dirname(os.path.abspath(__file__)))
12 c_path = "./cloud/"
13 f_path = "./forest/"
```

```
14
15 os.makedirs("./output/glcm/", exist_ok=True)  # 创建输出文件夹
16 os.makedirs("./output/lbp/", exist_ok=True)
17
18 c_name = [c_path + i for i in os.listdir(c_path)]
19 f_name = [f_path + i for i in os.listdir(f_path)]
20
21 def show_grey(root=c_name):
22     for n, i in enumerate(c_name):
23         plt.subplot(2, 5, n + 1)
24         img = Image.open(i)
25         img = img.convert("L")  # 转为灰度图
26         plt.imshow(img, cmap="gray")  # 显示灰度图
27         plt.axis("off")  # 不显示坐标轴
28     plt.show()
29
30 def get_mat(root):
31     for n, i in enumerate(root):
32         # plt.subplot(2, 5, n + 1)
33         img = Image.open(i)
34         img = img.convert("L")  # 转为灰度图
35         img = np.array(img)  # 转为ndarray
36         yield i, img
37
38 # 计算灰度共生矩阵(gray-level co-occurrence matrix)
39 def get_glcm(img, angle=0, distance=1, gray_levels=256):
40     # 归一化到0-gray_levels灰度级
41     img = (img / (np.max(img) + 1e-5) * (gray_levels - 1)).
42         astype(int)
43     h, w = img.shape  # 获取图像的尺寸
44     glcm = np.zeros((gray_levels, gray_levels), dtype=np.
45         int64)
46
47     for i in range(h):
48         for j in range(w):
49             match angle:
50                 case 0:
```

```
49         if j + distance < w:
50             glcm[img[i, j], img[i, j + distance
51             ]] += 1
52         case 90:
53             if i + distance < h:
54                 glcm[img[i, j], img[i + distance, j
55                 ]] += 1
56         case 45:
57             if i + distance < h and j + distance <
58             w:
59                 glcm[img[i, j], img[i + distance, j
60                 + distance]] += 1
61         case 135:
62             if i + distance < h and j - distance >=
63             0:
64                 glcm[img[i, j], img[i + distance, j
65                 - distance]] += 1
66
67     glcm = glcm / np.sum(glcm) # 归一化
68     return glcm
69
70 # 绘制灰度共生矩阵
71 def get_glcms_fig(root, angles=[0, 90, 45, 135]):
72     for i, img in get_mat(root):
73         glcms = [] # 存储各角度的 GLCM 图像
74         for angle in angles:
75             glcm = get_glcms(img, angle=angle)
76             glcm = (glcm / glcm.max() * 255).astype(np.
77             uint8)
78             glcms.append((Image.fromarray(glcm), f"Angle: {
79             angle}°")) # 保存图像和标题
80
81     # 计算单张 GLCM 图像的大小
82     width, height = glcms[0][0].size
83     title_height = 20 # 为标题预留的高度
84
85     # 2x2 布局, 标题增加额外空间
```

```
78         canvas = Image.new("L", (2 * width, 2 * (height +
79             title_height)), "white")
80
81         draw = ImageDraw.Draw(canvas)
82         font = ImageFont.load_default()
83
84         for idx, (glcm, title) in enumerate(glcms):
85             x_offset = (idx % 2) * width
86             y_offset = (idx // 2) * (height + title_height)
87
88             text_bbox = draw.textbbox((0, 0), title, font=
89                 font)
90             text_width = text_bbox[2] - text_bbox[0]
91             text_x = x_offset + (width - text_width) // 2
92             draw.text((text_x, y_offset), title, fill="
93                 black", font=font)
94
95             canvas.paste(glcm, (x_offset, y_offset +
96                 title_height))
97
98         canvas.save(f"./output/glcm/{i}")
99
100 # 计算局部二值模式(local binary pattern)
101 def get_lbp(img):
102     h, w = img.shape # 获取图像的尺寸
103     lbp = np.zeros((h - 2, w - 2), dtype=np.int64) # 忽略边
104     # 缘像素
105     for i in range(1, h - 1):
106         for j in range(1, w - 1):
107             center = img[i, j]
108             neighbors = [
109                 img[i, j - 1],
110                 img[i + 1, j - 1],
111                 img[i + 1, j],
112                 img[i + 1, j + 1],
113                 img[i, j + 1],
114                 img[i - 1, j + 1],
```



```
110         img[i - 1, j],
111         img[i - 1, j - 1],
112     ]
113     code = 0
114     for idx, neighbor in enumerate(neighbors):
115         if neighbor > center:
116             code += 2**idx
117     lbp[i - 1, j - 1] = code
118     return lbp
119
120
121 # 绘制局部二值模式
122 def get_lbp_fig(root):
123     for i, img in get_mat(root):
124         lbp = get_lbp(img)
125         Image.fromarray(lbp.astype(np.uint8)).save(f"./
            output/lbp/{i}")
126
127 def main():
128     roots = [c_name, f_name]
129     for root in roots:
130         get_glcmm_fig(root)
131         get_lbp_fig(root)
132
133 if __name__ == "__main__":
134     main()
```

第二章 第二次上机实验

2.1 实验要求

2.2 数据分析与处理

2.3 实验步骤与原理

2.4 实验结论与分析

2.5 实验代码

第三章 第三次上机实验

3.1 实验要求

3.2 数据分析与处理

3.3 实验步骤与原理

3.4 实验结论与分析

3.5 实验代码

第四章 第四次上机实验

4.1 实验要求

4.2 数据分析与处理

4.3 实验步骤与原理

4.4 实验结论与分析

4.5 实验代码

第五章 第五次上机实验

5.1 实验要求

5.2 数据分析与处理

5.3 实验步骤与原理

5.4 实验结论与分析

5.5 实验代码