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

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 $\delta = (\mathrm{d}x^2 + \mathrm{d}y^2)^{1/2}$，灰度为 j 的像素同时出现的概率为, 数学表达式为：

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

分别考虑 $\theta = 0°, 45°, 90°, 135°$,
当 $\theta = 0°$ 时，$\mathrm{d}x = 1, \mathrm{d}y = 0$；
当 $\theta = 45°$ 时，$\mathrm{d}x = 1, \mathrm{d}y = -1$；
当 $\theta = 90°$ 时，$\mathrm{d}x = 0, \mathrm{d}y = -1$；
当 $\theta = 135°$ 时，$\mathrm{d}x = -1, \mathrm{d}y = -1$。

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

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl
from PIL import Image, ImageDraw, ImageFont

mpl.rcParams["font.sans-serif"] = ["SimHei"]  # 中文
plt.rcParams["axes.unicode_minus"] = False  # 正负号

# 更改当前工作目录为脚本所在目录
os.chdir(os.path.dirname(os.path.abspath(__file__)))
c_path = "./cloud/"
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
22 def show_grey(root=c_name):
23     for n, i in enumerate(c_name):
24         plt.subplot(2, 5, n + 1)
25         img = Image.open(i)
26         img = img.convert("L")  # 转为灰度图
27         plt.imshow(img, cmap="gray")  # 显示灰度图
28         plt.axis("off")  # 不显示坐标轴
29     plt.show()
30
31
32 def get_mat(root):
33     for n, i in enumerate(root):
34         # plt.subplot(2, 5, n + 1)
35         img = Image.open(i)
36         img = img.convert("L")  # 转为灰度图
37         img = np.array(img)  # 转为ndarray
38         yield i, img
39
40
41 # 计算灰度共生矩阵(gray-level co-occurrence matrix)
42 def get_glcm(img, angle=0, distance=1, gray_levels=256):
43     # 归一化到0-gray_levels灰度级
44     img = (img / (np.max(img) + 1e-5) * (gray_levels - 1)).
          astype(int)
45     h, w = img.shape  # 获取图像的尺寸
46     glcm = np.zeros((gray_levels, gray_levels), dtype=np.
          int64)
47
48     for i in range(h):
```

```
49          for j in range(w):
50              match angle:
51                  case 0:
52                      if j + distance < w:
53                          glcm[img[i, j], img[i, j + distance
        ]] += 1
54                  case 90:
55                      if i + distance < h:
56                          glcm[img[i, j], img[i + distance, j
        ]] += 1
57                  case 45:
58                      if i + distance < h and j + distance <
        w:
59                          glcm[img[i, j], img[i + distance, j
         + distance]] += 1
60                  case 135:
61                      if i + distance < h and j - distance >=
         0:
62                          glcm[img[i, j], img[i + distance, j
         - distance]] += 1
63
64      glcm = glcm / np.sum(glcm)  # 归一化
65      return glcm
66
67
68  # 绘制灰度共生矩阵
69  def get_glcm_fig(root, angles=[0, 45, 90, 135]):
70      for i, img in get_mat(root):
71          glcms = []  # 存储各角度的 GLCM 图像
72          for angle in angles:
73              glcm = get_glcm(img, angle=angle)
74              glcm = (glcm / glcm.max() * 255).astype(np.
        uint8)
75              glcms.append((Image.fromarray(glcm), f"Angle: {
        angle}°"))  # 保存图像和标题
76
77          # 计算单张 GLCM 图像的大小
```

```
78          width, height = glcms[0][0].size
79          title_height = 20  # 为标题预留的高度
80
81          # 2x2 布局，标题增加额外空间
82          canvas = Image.new("L", (2 * width, 2 * (height +
        title_height)), "white")
83
84          draw = ImageDraw.Draw(canvas)
85          font = ImageFont.load_default()
86
87          for idx, (glcm, title) in enumerate(glcms):
88              x_offset = (idx % 2) * width
89              y_offset = (idx // 2) * (height + title_height)
90
91              text_bbox = draw.textbbox((0, 0), title, font=
        font)
92              text_width = text_bbox[2] - text_bbox[0]
93              text_x = x_offset + (width - text_width) // 2
94              draw.text((text_x, y_offset), title, fill="
        black", font=font)
95
96              canvas.paste(glcm, (x_offset, y_offset +
        title_height))
97
98          canvas.save(f"./output/glcm/{i}")
99
100
101 # 计算局部二值模式(local binary pattern)
102 def get_lbp(img):
103     h, w = img.shape  # 获取图像的尺寸
104     lbp = np.zeros((h - 2, w - 2), dtype=np.int64)  # 忽略边
        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],
115                     img[i - 1, j],
116                     img[i - 1, j - 1],
117                 ]
118                 code = 0
119                 for idx, neighbor in enumerate(neighbors):
120                     if neighbor > center:
121                         code += 2**idx
122                 lbp[i - 1, j - 1] = code
123     return lbp
124
125
126 # 绘制局部二值模式
127 def get_lbp_fig(root):
128     for i, img in get_mat(root):
129         lbp = get_lbp(img)
130         Image.fromarray(lbp.astype(np.uint8)).save(f"./
        output/lbp/{i}")
131
132
133 def main():
134     roots = [c_name, f_name]
135     for root in roots:
136         get_glcm_fig(root)
137         get_lbp_fig(root)
138
139
140 if __name__ == "__main__":
141     main()
```

第二章　第二次上机实验（垂直平分分类器）

2.1　实验要求

- 由训练数据，训练一个垂直平分分类器
- 对测试数据进行分类
- 使用 Python 编程实现

2.2　数据分析与处理

测试集与训练集均为已分类的二维数据。

2.3　实验步骤与原理

训练集中两类数据的均值分别为记为 m_1 和 m_2，则垂直平分形式的线性分类器为

$$g(\boldsymbol{x}) = (m_1)'(\boldsymbol{x} - (m_1 - m_2)/2),$$

决策面方程为

$$g(\boldsymbol{x}) = 0,$$

其中 $\boldsymbol{x}$ 为向量。
当 $g(\boldsymbol{x}) > 0$ 时，决策为“1”类，否则为“2”类。

2.4　实验结论与分析

图 2.1 展示了训练集与决策面的可视化图，图 2.2 展示了测试集与决策面可视化图。

进而可以计算出模型的准确率（accuracy），精确率（precision），召回率（recall）和 F1 得分的值分别为 99.25%，99.00%，99.50%，99.25%（保留两位小数）。说明模型的预测性能表现优异。

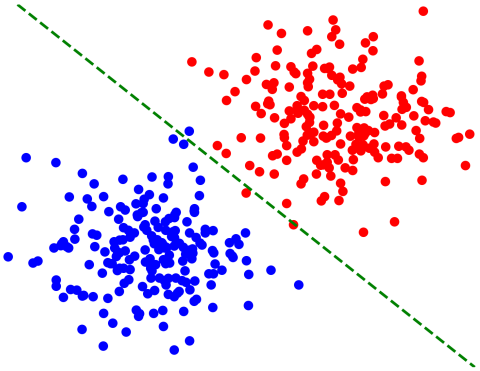

图 2.1: 训练集与决策面

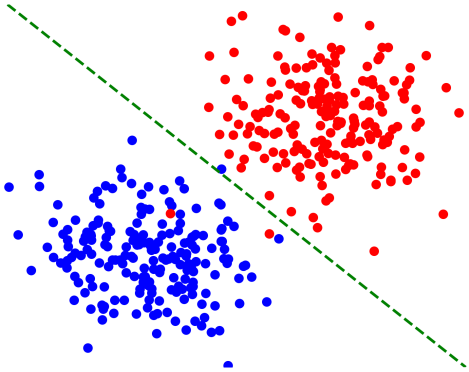

图 2.2: 测试集与决策面

2.5 实验代码

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib as mpl
import os

mpl.rcParams["font.sans-serif"] = ["SimHei"]  # 中文
plt.rcParams["axes.unicode_minus"] = False  # 负号
os.chdir(os.path.dirname(os.path.abspath(__file__)))  # 更改当前工作目录为脚本所在目录

train_path = "data/train.txt"
test_path = "data/test.txt"


def train(data):
    mean = data.groupby("Label").mean()
    m_1 = mean.loc[1]
    m_2 = mean.loc[2]
    point = (m_1 + m_2) / 2
    slope = m_1 - m_2
    return point, slope


```

```
24 def test(data, point, slope):
25     data["Predict"] = np.where(
26         (data.X - point.X) * slope.X + (data.Y - point.Y) *
           slope.Y > 0, 1, 2
27     )
28     data.to_csv("output/test_predict.csv")
29     return data
30
31
32 def plot(data, point, slope, filename):
33     plt.axline(
34         point,
35         slope=-slope.X / slope.Y,
36         color="green",
37         linewidth=2,
38         linestyle="--",
39         label="Perpendicular Bisector Classifier",
40     )
41     color = ["r", "b"]
42     for _, i in data.iterrows():
43         x, y, c = i.iloc[0], i.iloc[1], color[int(i.iloc
           [2]) - 1]
44         plt.scatter(x, y, color=c)
45     plt.xlim(-8, 8)
46     plt.ylim(-8, 8)
47     plt.axis("off")
48     plt.savefig(filename, dpi=300, bbox_inches="tight")
49     # plt.show()
50     plt.close()
51
52
53 def evaluate(data):
54     TP = 0
55     TN = 0
56     FP = 0
57     FN = 0
58     for _, i in data.iterrows():
```

```
59          if i.iloc[2] == i.iloc[3]:
60              if i.iloc[2] == 1:
61                  TP += 1
62              else:
63                  TN += 1
64          else:
65              if i.iloc[2] == 1:
66                  FP += 1
67              else:
68                  FN += 1
69      accuracy = (TP + TN) / (TP + TN + FP + FN)
70      precision = TP / (TP + FP)
71      recall = TP / (TP + FN)
72      F1 = 2 * precision * recall / (precision + recall)
73      print("accuracy:", accuracy)
74      print("precision:", precision)
75      print("recall:", recall)
76      print("F1 score:", F1)
77
78
79  def main():
80      train_data = pd.read_csv(train_path, index_col=0)
81      test_data = pd.read_csv(test_path, index_col=0)
82      point, slope = train(train_data)
83      # 绘制测试集与决策界
84      plot(train_data, point, slope, "./output/train.png")
85      # 绘制训练集与决策界
86      plot(test_data, point, slope, "./output/test.png")
87      test_data = test(test_data, point, slope)
88      # 评估测试结果
89      evaluate(test_data)
90
91
92  if __name__ == "__main__":
93      main()
```

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