Hw1 作业解答

学号:2430534　　姓名:杨赵山

1. 题目 1：

1.1 程序思路说明：

构建垂直线和水平线作为弱分类器。垂直线分类器的数学表达式如公式（1.1）所示，水平线分类器的数学表达式如公式（1.2）所示：

$$h(x,y)=\begin{cases}+1 & \text{if } x<c\\ -1 & \text{if } x\geq c\end{cases} \tag{1.1}$$

$$h(x,y)=\begin{cases}+1 & \text{if } y<c\\ -1 & \text{if } y\geq c\end{cases} \tag{1.2}$$

通过 AdaBoost 逐步训练，并选择最优的弱分类器。首先为每一个样本分配相同的初始化权重，如公式（1.3）所示：

$$w_i=\frac{1}{n} \tag{1.3}$$

在迭代过程中，计算分类器权重，如公式（1.4）所示。更新样本的权重，对错误分类的样本，增加其权重；被正确分类的样本，减小其权重。更新权重公式如公式（1.5）所示：

$$\alpha_t=\frac{1}{2}\ln\left(\frac{1-e_t}{e_t}\right) \tag{1.4}$$

$$w_i^{(t+1)}=w_i^{(t)}\exp\left(-\alpha_t l_i h_t(x_i,y_i)\right) \tag{1.5}$$

组合强分类器，将所有的弱分类器进行加权和，如公式（1.6）所示。

$$f(x,y)=\sum_t \alpha_t h_t(x,y) \tag{1.6}$$

1.2 程序运行结果

根据弱分类器的数量，AdaBoost 训练得到的强分类器，对于数据的预测能力不同，因此绘制出弱分类器数量与误差率的关系图，如图 1 所示。

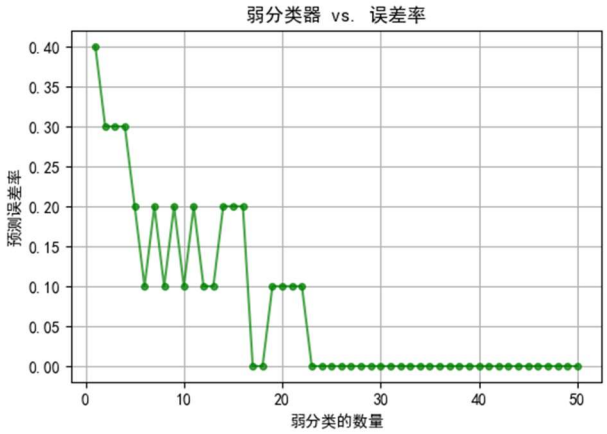

图 1 弱分类器数量与误差率的关系

从图中可知，到弱分类器的数量达到 23 个时，误差率在 0 处保持稳定，表示强分类器具有良好的性能。

2. 题目 2

2.1 问题 1

题目要求证明协方差矩阵 C 为半正定矩阵，证明过程如下：

1. 协方差矩阵的定义为：

$$C = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu)(x_i - \mu)^T \tag{1.7}$$

2. 考虑到任意向量 v，证明 $v^T Cv$ 的非负性，即如下所示。

$$v^T Cv = \frac{1}{n-1}\sum_{i=1}^{n} v^T (x_i - \mu)(x_i - \mu)^T v \tag{1.8}$$

3. 对表达式进行展开处理，可以得到如下表示：

$$v^T Cv = \frac{1}{n-1}\sum_{i=1}^{n} (v^T (x_i - \mu))^2 \tag{1.9}$$

4. 由于平方项的非负性，因此得到

$$v^T Cv \geq 0 \tag{1.10}$$

5. 得出结论，协方差矩阵为半正定矩阵，证明完成。

2.2 问题 2

题目要求证明在所有与最大特征值对应的特征向量α_1正交的方向中，方差最大化时的方向α_2是与协方差矩阵 C 的第二大特征值 α_2相关的特征向量。证明过程如下：

1. 设α_1是最大特征值λ_1的特征向量，即如下所示

$$C\alpha_1 = \lambda_1 \alpha_1 \tag{1.11}$$

2. 设α_2为和α_1正交的单位向量，即

$$\alpha_2^T \alpha_1 = 0 \quad 且 \quad \| \alpha_2 \| = 1 \tag{1.12}$$

3. 最大化投影到α_2的方差，即

$$v^T Cv \quad (其中 v = \alpha_2) \tag{1.13}$$

4. 利用 Rayleigh 商的性质，可以得到

$$\frac{v^T Cv}{v^T v} \tag{1.14}$$

5. 利用拉格朗日乘数法求解最大化方差时的约束条件，即

$$L(v, \lambda) = v^T Cv - \lambda(v^T v - 1) \tag{1.15}$$

进行对 v 进行求导，可以得到

$$Cv = \lambda v \tag{1.16}$$

6. 由于α_2是与α_1*正交的的单位向量，因此投影到*α_2的方差最大时，对应的特征值λ_2就是第二大的特征值。

7. 因此，方差最大化时的方向α_2是与协方差矩阵 C 的第二大特征值 α_2 相关的特征向量，证明完成。

3. 附录

题目 1 代码文件如下所示：

```
import numpy as np
import matplotlib.pyplot as plt

# 定义训练样本数据 (x, y, label)
samples = np.array([
    [80, 144, 1], [93, 232, 1], [136, 275, -1], [147, 131, -1],
    [159, 69, 1], [214, 31, 1], [214, 152, -1], [257, 83, 1],
    [307, 62, -1], [307, 231, -1]
])

X = samples[:, :2]  # 坐标 (x, y)
y = samples[:, 2]   # 标签 (+1, -1)

# 定义弱分类器：垂直线或水平线
def weak_classifier(x, y, line_pos, vertical=True, polarity=1):
    """
    垂直或水平线分类器
    :param x: 样本的 x 坐标
    :param y: 样本的 y 坐标
    :param line_pos: 分类线的位置
    :param vertical: 是否是垂直线 (True) 或水平线 (False)
    :param polarity: 分类极性 (+1 或 -1)
    :return: 分类结果 (+1 或 -1)
    """
    if vertical:
        return polarity * np.sign(line_pos - x)
    else:
        return polarity * np.sign(line_pos - y)

# 初始化样本权重
n_samples = len(y)
weights = np.ones(n_samples) / n_samples

# 存储弱分类器
classifiers = []
alphas = []
errors = []  # 记录每轮的误差

# 定义 AdaBoost 的训练过程
def adaboost_train(X, y, weights, num_classifiers=50):
    for t in range(num_classifiers):
        best_err = float('inf')
        best_clf = None
        best_polarity = 1
```

```
        best_line_pos = 0
        best_vertical = True

        # 尝试每个可能的垂直和水平线
        for i in range(n_samples):
            for vertical in [True, False]:
                for polarity in [1, -1]:
                    # 弱分类器根据每个样本位置（垂直线根据 x，水平线根据 y）
                    clf = weak_classifier(X[:, 0], X[:, 1], X[i, 0] if vertical else X[i, 1],
vertical, polarity)
                    err = np.sum(weights[y != clf])  # 计算加权错误率

                    if err < best_err:
                        best_err = err
                        best_clf = clf
                        best_polarity = polarity
                        best_line_pos = X[i, 0] if vertical else X[i, 1]
                        best_vertical = vertical

        # 计算该分类器的权重 alpha
        alpha = 0.5 * np.log((1 - best_err) / (best_err + 1e-10))
        alphas.append(alpha)
        classifiers.append((best_line_pos, best_vertical, best_polarity))

        # 更新样本权重
        weights *= np.exp(-alpha * y * best_clf)
        weights /= np.sum(weights)

        # 计算当前强分类器的误差并记录
          strong_clf = np.sign(np.sum([alpha * weak_classifier(X[:, 0], X[:, 1], line_pos,
vertical, polarity)
                                       for alpha, (line_pos, vertical, polarity) in zip(alphas,
classifiers)], axis=0))
        error_rate = np.mean(strong_clf != y)
        errors.append(error_rate)

# 训练 AdaBoost
adaboost_train(X, y, weights)

# 定义最终的强分类器
def strong_classifier(x, y):
    final_result = 0
    for alpha, (line_pos, vertical, polarity) in zip(alphas, classifiers):
        result = weak_classifier(x, y, line_pos, vertical, polarity)
```

```
        final_result += alpha * result
    return np.sign(final_result)

# 测试强分类器
predicted_labels = np.array([strong_classifier(x, y) for x, y in X])

# 可视化结果
plt.figure(figsize=(4, 4))
colors = ['red' if label == -1 else 'blue' for label in y]
markers = ['+' if label == -1 else '_' for label in y]
for i in range(n_samples):
    plt.scatter(X[i,0], X[i,1], color=colors[i], marker=markers[i] ,s=100, edgecolors='k')
#plt.show()

# 输出预测结果
print("Predicted labels:", predicted_labels)
print("True labels    :", y)

# 设置中文显示
plt.rcParams['font.sans-serif'] = ['SimHei']  # 黑体
plt.rcParams['axes.unicode_minus'] = False  # 解决负号'-'显示为方块的问题

# 绘制弱分类器数量与预测误差的关系图
plt.figure(figsize=(6, 4))
plt.plot(range(1, len(errors) + 1), errors,'g', marker='o', markersize=4, alpha=0.7)
plt.title('弱分类器 vs. 误差率')
plt.xlabel('弱分类的数量')
plt.ylabel('预测误差率')
plt.grid(True)
plt.show()
```