长安大学
CHANG'AN UNIVERSITY

学生实验报告

实验课名称：模式识别与机器学习

实验项目名称：编程实现贝叶斯分类器

专 业 名 称：人工智能

班　　级：　2022240401

学　　号：　2022905226 2022904484

学 生 姓 名：

教 师 姓 名：

实验地点：WX2102

实验日期：2024.10.13

一、实验名称：编程实现贝叶斯分类器

二、实验原理

本实验设计贝叶斯分类器实现目标数据集的分类。贝叶斯分类器是基于贝叶斯定理的一种统计学分类方法。其核心思想是通过计算给定特征条件下各类别的后验概率来进行分类决策。

2.1 贝叶斯定理

贝叶斯定理是贝叶斯分类器的基础，描述了条件概率之间的关系：

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

在分类问题中，我们关心的是给定特征 x 时，样本属于类别 ω_i 的概率，即后验概率 $P(\omega_i|x)$：

$$P(\omega_i|x) = \frac{P(x|\omega_i) \cdot P(\omega_i)}{P(x)}$$

其中： - $P(\omega_i|x)$ 是后验概率，即给定特征 x 时样本属于类别 ω_i 的概率。 - $P(x|\omega_i)$ 是似然概率，表示在类别 ω_i 条件下观察到特征 x 的概率。 - $P(\omega_i)$ 是先验概率，表示类别 ω_i 在总体中出现的概率。 - $P(x)$ 是证据因子，表示特征 x 出现的总概率。

2.2 朴素贝叶斯假设

朴素贝叶斯分类器假设特征之间相互独立。虽然这个假设在实际中往往不成立，但它大大简化了计算，并且在许多实际应用中仍然表现良好。在本实验中，**我们不直接使用这个假设，而采用多元正态分布来建模特征的联合分布**。

2.3 多元正态分布

在本实验中，假设每个类别的特征服从多元正态分布。多元正态分布的概率密度函数为：

$$p(x|\omega_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right)$$

其中：

- d 是特征的维度
- μ_i 是类别 ω_i 的均值向量
- Σ_i 是类别 ω_i 的协方差矩阵

- $|\Sigma_i|$ 是协方差矩阵的行列式
- Σ_i^{-1} 是协方差矩阵的逆矩阵

2.4 参数估计

为了使用多元正态分布，需要估计每个类别的均值向量和协方差矩阵。这些参数通过**最大似然估计（MLE）方法**从训练数据中估计得到：

均值向量的估计：

$$\hat{\mu}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} x_k$$

协方差矩阵的估计：

$$\hat{\Sigma}_i = \frac{1}{n_i}\sum_{k=1}^{n_i} (x_k - \hat{\mu}_i)\,(x_k - \hat{\mu}_i)^T$$

其中 n_i 是类别 ω_i 中的样本数量，x_k 是该类别中的第 k 个样本向量。

2.5 分类决策

在得到每个类别的参数估计后，对于一个新的样本 x，计算它属于每个类别的后验概率：

$$P(\omega_i|x) \propto p(x|\omega_i) \cdot P(\omega_i)$$

由于 $P(x)$ 对所有类别都是相同的，所以可以忽略，只比较 $p(x|\omega_i) \cdot P(\omega_i)$ 的值。

最终的分类决策是选择后验概率最大的类别：

$$\hat{\omega} = \arg\max_i P(\omega_i|x) = \arg\max_i p(x|\omega_i) \cdot P(\omega_i)$$

在实际实现中，使用 SciPy 库的 `multivariate_normal.pdf` 函数来计算多元正态分布的概率密度。这个函数接收样本点、均值向量和协方差矩阵作为输入，返回概率密度值。

对于不同维度的特征（1D、2D、3D），只需要调整均值向量和协方差矩阵的维度，而核心的分类算法保持不变。可以自然地处理多维特征，并且考虑了特征之间的相关性。通过上述方法，我们可以构建一个**能够处理多类别、多维特征的贝叶斯分类器**，并应用于给定的数据集进行分类任务。

三、问题回答

3.1　二分类问题的分类器设计

对于二分类问题，设计一个**基于多元正态分布的贝叶斯分类器**。分类器的主要步骤如下：

3.1.1参数估计

对于每个类别 ω_i，估计其**特征样本的均值向量** μ_i 和**协方差矩阵** Σ_i。估计公式：

$$\mu_i = (1/n_i) * \Sigma(x_k)$$

$$\Sigma_i = (1/n_i) * \Sigma\big((x_k - \mu_i)(x_k - \mu_i)^T\big)$$

- 当样本中的特征取一维时，均值是一个标量，协方差就是该特征的方差。
- 当样本中的特征取二维时，均值是一个二维向量，协方差是一个 2x2 的矩阵。
- 当样本中的特征取三维时，均值是一个三维向量，协方差是一个 3x3 的矩阵。

3.1.2概率密度计算

对于给定的样本点 x，计算它属于每个类别的**条件概率密度** $p(x|\omega_i)$。**假设每个类别的特征分布都服从多元正态分布**。使用多元正态分布的概率密度函数来计算条件概率：

$$p(x|\omega_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i)\right)$$

上述过程使用 SciPy 库的 `multivariate_normal.pdf` 函数来实现。此函数接收**某类别样本点数据**、**某类别样本点均值向量**和**某类别样本点协方差矩阵**就可以计算出某类别样本点数据在样本点均值向量和样本点协方差矩阵的多元正态分布下的概率密度。此概率密度就是样本点属于某类别的条件概率。

3.1.3后验概率计算和分类

根据贝叶斯定理，后验概率 $P(\omega_i|x)$ 正比于 $p(x|\omega_i) * P(\omega_i)$，其中 $P(\omega_i)$ 是先验概率。

$$P(\omega_i|x) = \frac{p(x|\omega_i) * P(\omega_i)}{\sum_j p\left(x|\omega_j\right) * P\left(\omega_j\right)}$$

其中 j 遍历所有类别。

处理决策就是选择使 $P(\omega_i|x)$ 最大的类别 i，也就是使 $p(x|\omega_i)*P(\omega_i)$ 最大的类别 i。其中，**先验概率 $P(\omega_i)$ 是已知的，所以处理决策就是选择使 $p(x|\omega_i)$ 最大的类别 i**。

3.1.4具体分类过程

3.1.4.1 使用 1 特征（x_1）的二分类

考虑数据集中的前两个类别（ω_1和ω_2），仅使用第一个特征（x_1）。

1. 参数估计
    - 对于ω_1:
        - 计算其类别下特征x_1的所有样本点的均值和协方差得：$\mu_1 = -0.54, {\sigma_1}^2 = 13.55$（对于一维数据，协方差就是样本点的方差）
    - 对于ω_2:
        - 计算其类别下特征x_1的所有样本点的均值和协方差得：$\mu_2 = -0.98, {\sigma_2}^2 = 25.33$
2. 针对特征x_1依次选取样本点（一维）,假设要分类的点是 $x = 1.08$。
3. 计算条件概率密度

$$p(x|\omega_1) = \left(1/\sqrt{(2\pi * 13.55)}\right) * exp\left(-\left(1.08-(-0.54)\right)^2/(2*13.55)\right) \approx 0.0986$$

$$p(x|\omega_2) = \left(1/\sqrt{(2\pi * 25.33)}\right) * exp\left(-\left(1.08-(-0.98)\right)^2/(2*25.33)\right) \approx 0.0639$$

4. 先验概率相等，即 $P(\omega_1) = P(\omega_2) = 0.5$，计算后验概率

$$P(\omega_1|x) \propto 0.0986 * 0.5 = 0.0493$$

$$P(\omega_2|x) \propto 0.0639 * 0.5 = 0.03195$$

5. 分类决策 由于 $P(\omega_1|x) > P(\omega_2|x)$，将该点分类为 ω_1（打上预测标签 1），这与实际情况相符。
6. 针对每个特征x_1样本点重复上述过程，直到所有样本点分类完毕。

3.1.4.2 使用 2 特征（x_1和x_2）的二分类问题

现在考虑使用前两个特征（x_1和x_2）进行分类。

1. 参数估计
    - 对于 ω_1:

        - 计算其类别下特征x_1和x_2的所有样本点的均值和协方差矩阵得：$\mu_1 = [-0.54, -2.15]$ $\Sigma_1 = \left[[13.55, 0.91], [0.91, 15.89]\right]$
    - 对于 ω_2：
        - 计算其类别下特征x_1和x_2的所有样本点的均值和协方差矩阵得：$\mu_2 = [-0.98, -0.91]$ $\Sigma_2 = \left[[25.33, -4.31], [-4.31, 12.98]\right]$
2. 针对特征x_1和x_2依次选取样本点（二维），假设要分类的点是 $x = [1.08, -5.52]$
3. 计算条件概率密度（使用 `multivariate_normal.pdf` 函数）：（函数内传入样本数据和特征均值向量与特征协方差矩阵即可）$p(x|\omega_1) \approx 0.0051$ $p(x|\omega_2) \approx 0.0016$
4. 先验概率相等，即 $P(\omega_1) = P(\omega_2) = 0.5$，计算后验概率：$P(\omega_1|x) \propto 0.0051 * 0.5 = 0.00255$ $P(\omega_2|x) \propto 0.0016 * 0.5 = 0.0008$
5. 分类决策 由于 $P(\omega_1|x) > P(\omega_2|x)$，将该点分类为 ω_1（打上预测标签 1），这与实际情况相符。

从上面两个计算过程中可以看到，分类器都正确地将样本点分类到了 ω_1 类别。但是发现**当从 1 特征扩展到 2 特征时，条件概率密度的值变小了**。实际上是因为在高维空间中，数据变得更加稀疏，再加上本身数据量就比较小，从结果上看，反而会造成分类效果变差了。

实际代码实现中使用 `multivariate_normal.pdf` 函数来计算条件概率密度。分类决策通过比较不同类别的后验概率（或等价地，条件概率密度与先验概率的乘积）来完成。

3.2 三分类问题的分类器设计

三分类问题的分类器设计原理与二类问题相同，只是扩展到了三个类别。主要区别在于：

1. 需要**估计三个类别的参数**（均值和协方差）。
    1. 估计参数的方式与二类完全相同。
2. 仍然需要用 `multivariate_normal.pdf` 函数来**计算条件概率密度**。将样本数据、某类别样本点均值向量和某类别样本点协方差矩阵传入参与计算即可。
3. 在分类时，计算**样本属于三个类别的后验概率，并选择最大的一个**。
4. 先验概率从二类问题的 [0.5, 0.5] 变为三类问题的 [1/3, 1/3, 1/3]。

决策原则仍然是选择**使 $p(x|\omega_i) * P(\omega_i)$ 最大的类别 i**。不同维度特征的处理方式通过调整均值向量和协方差矩阵的维度来实现。

- 对于一维特征，样本输入是一维，取三类三组样本，均值是标量，协方差是标量。
- 对于二维特征，样本输入是二维，取三类三组样本，均值是二维向量，协方差是 2x2 矩阵。
- 对于三维特征，样本输入是三维，取三类三组样本，均值是三维向量，协方差是 3x3 矩阵。

四、实验结果与分析

4.1 二分类问题

由于数据量少，取测试集即为训练集，“样本的经验训练误差”指用训练样本进行测试，分类错误的点的个数占总个数的百分比。

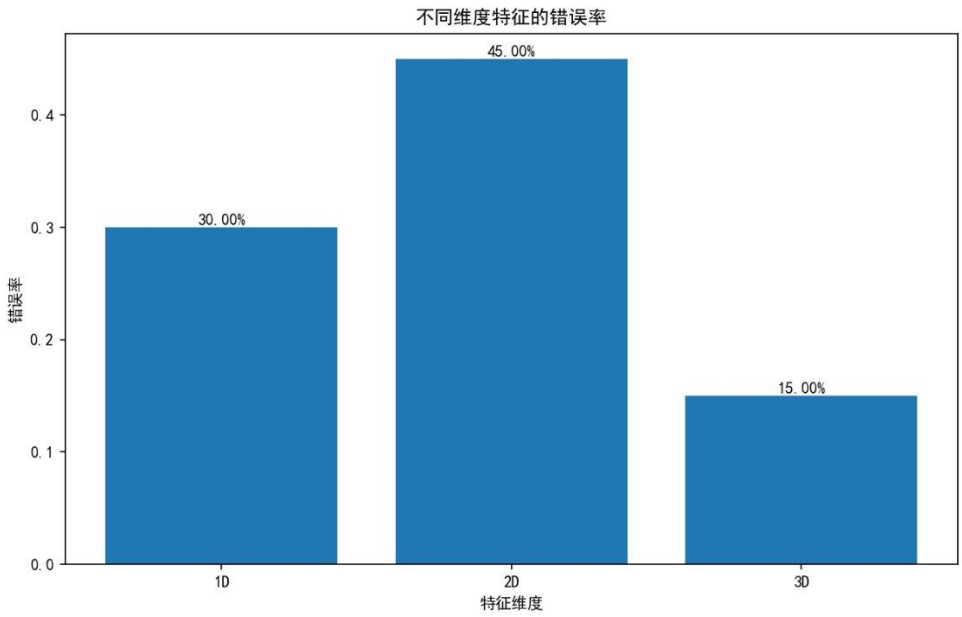

图 1：二分类不同维度特征错误率

1. 使用$x1$特征：样本的经验训练误差为 **30.00%**。（6/20）
2. 使用$x1$和$x2$特征：样本的经验训练误差为 **45.00%**。（9/20）
3. 使用所有三个特征：样本的经验训练误差为 **15.00%**。（3/20）

下面展示二分类下采用不同维度特征时的决策边界示意图。

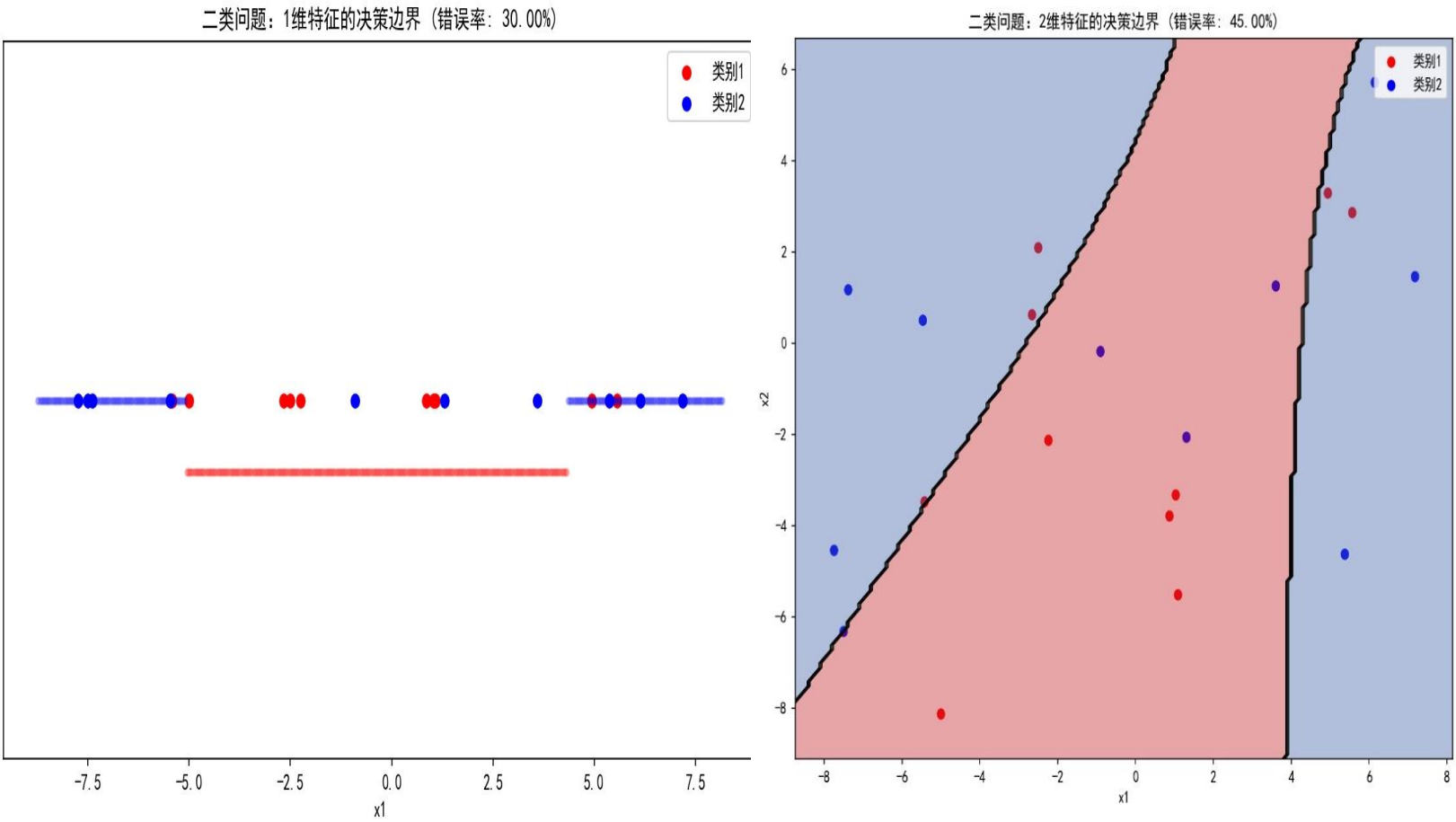

图 2：二分类一维、二维决策边界示意图

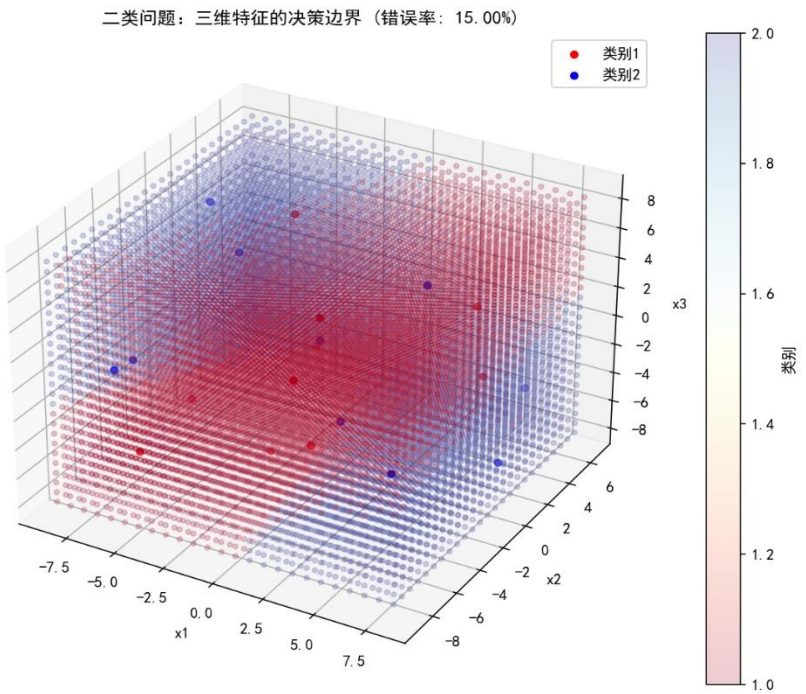

图 3：二分类三维决策边界示意图

4.2　三分类问题

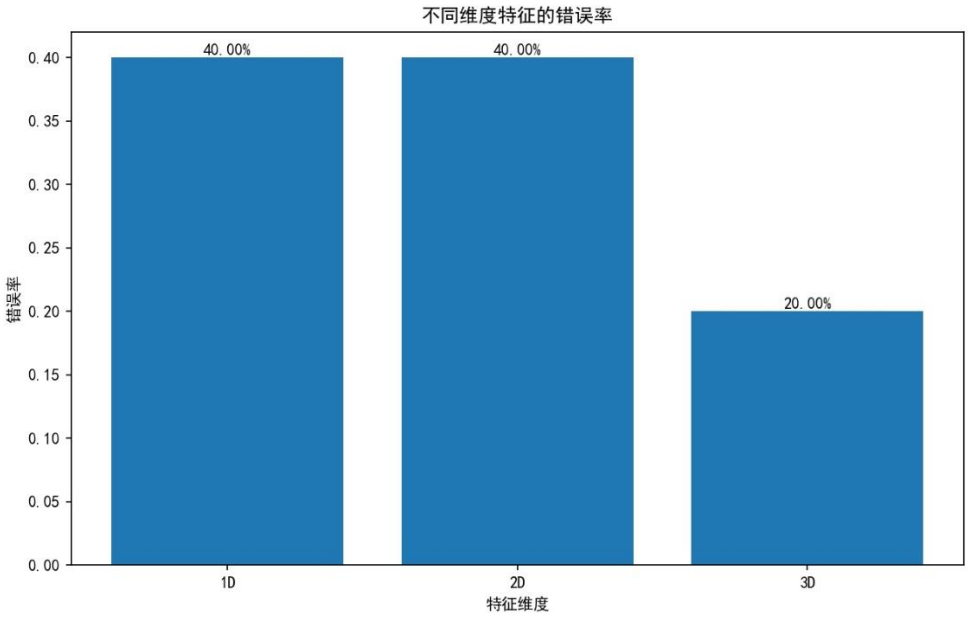

图 4：三分类不同维度特征错误率

1. 使用$x1$特征：样本的经验训练误差为 **40.00%**。（12/30）
2. 使用$x1$和$x2$特征：样本的经验训练误差为 **40.00%**。（12/30）
3. 使用所有三个特征：样本的经验训练误差为 **20.00%**。（6/30）

下面展示二分类下采用不同维度特征时的决策边界示意图。

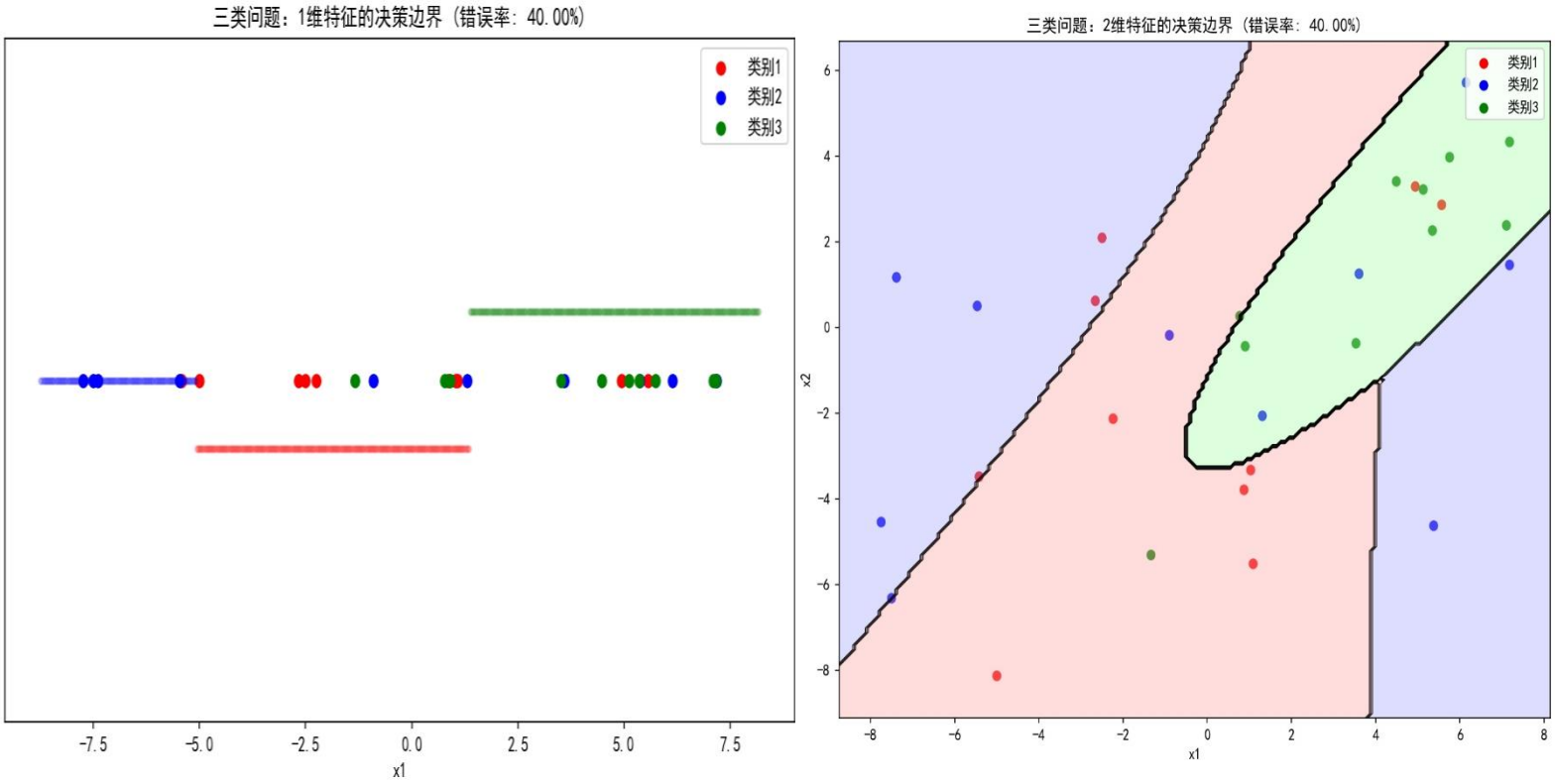

图 5：三分类一维、二维决策边界示意

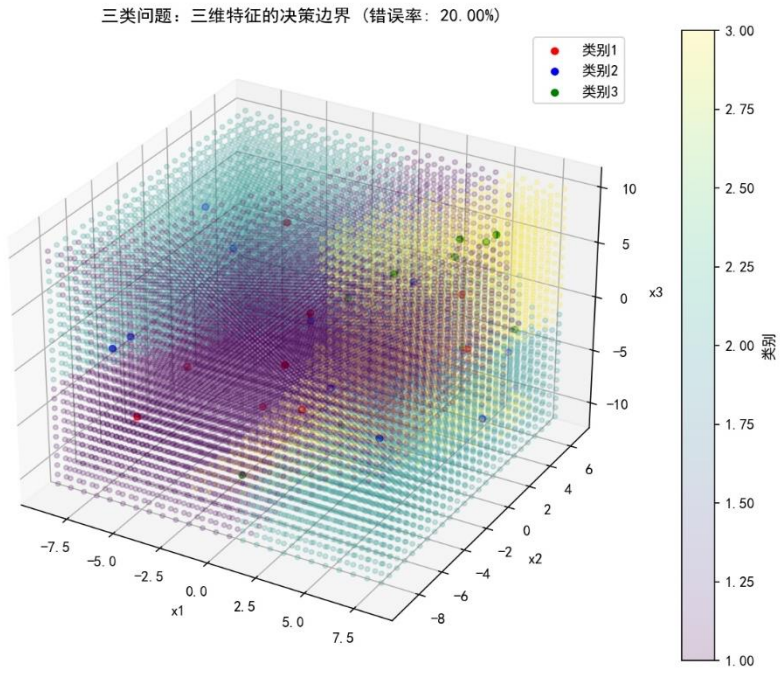

图 6：三分类三维决策边界示意图

4.3 结果分析

1. 对于二分类问题

    1. 观察到使用单一特征（$x1$）时，错误率为 30%。当增加到两个特征（$x1$和$x2$）时，**错误率反而上升到 45%**。可能是数据量较小，数据在二维空间中的分布更加复杂，或者$x2$特征引入了更多的噪声。所以，**对于一个有限的数据集，是完全有可能在更高的数据维数下经验误差会增加的。**
    2. 当使用所有三个特征时，错误率下降到 15%，表示**第三个特征提供了重要的分类信息**。
2. 对于三分类问题，使用$x1$特征和使用$x1$、$x2$两个特征时的错误率都是 40%，表示仅增加$x2$特征并没有提供额外的有用信息。
    1. 当使用所有三个特征时，错误率降低到 20%，**也表示第三个特征提供了重要的分类信息**。
3. 在二分类和三分类问题中，都观察到**使用更多特征并不总是会降低错误率**。但总的来说，对于实验中特定的数据集，使用所有三个特征能够获得最佳的分类性能。

五、部分代码

```
# 数据加载
data = np.array([
    [-5.01, -8.12, -3.68], [-5.43, -3.48, -3.54], [1.08, -5.52, 1.66],
    [0.86, -3.78, -4.11], [-2.67, 0.63, 7.39], [4.94, 3.29, 2.08],
    [-2.51, 2.09, -2.59], [-2.25, -2.13, -6.94], [5.56, 2.86, -2.26],
    [1.03, -3.33, 4.33], [-0.91, -0.18, -0.05], [1.30, -2.06, -3.53],
    [-7.75, -4.54, -0.95], [-5.47, 0.50, 3.92], [6.14, 5.72, -4.85],
    [3.60, 1.26, 4.36], [5.37, -4.63, -3.65], [7.18, 1.46, -6.66],
    [-7.39, 1.17, 6.30], [-7.50, -6.32, -0.31], [5.35, 2.26, 8.13],
    [5.12, 3.22, -2.66], [-1.34, -5.31, -9.87], [4.48, 3.42, 5.19],
    [7.11, 2.39, 9.21], [7.17, 4.33, -0.98], [5.75, 3.97, 6.65],
    [0.77, 0.27, 2.41], [0.90, -0.43, -8.71], [3.52, -0.36, 6.43]
])

# 将数据分为三类
class_1, class_2, class_3 = data[:10], data[10:20], data[20:]

def estimate_parameters(X):
    """
    参数估计
    @param X: 样本数据
    @return: 均值和协方差矩阵
    """
    return np.mean(X, axis=0), np.cov(X, rowvar=False)

def calculate_error_rate(true_labels, predicted_labels):
```

```
    """
    计算分类错误率
    @param true_labels: 真实标签
    @param predicted_labels: 预测标签
    @return: 错误率  这个错误率就是分类错误的样本数除以总的样本数
    """
    return np.mean(np.array(true_labels) != np.array(predicted_labels))
  # 计算错误率

def bayes_classifier(x, means, covs, priors):
    """
    贝叶斯分类器
    @param x: 样本数据
    @param means: 各类别的均值向量
    @param covs: 各类别的协方差矩阵
    @param priors: 各类别的先验概率
    @return: 预测的类别标签
    """
    probs = [multivariate_normal.pdf(x, mean=mean, cov=cov) * prior
             for mean, cov, prior in zip(means, covs, priors)]       #
用多元正态分布的概率密度函数计算每个类别的概率
    return np.argmax(probs) + 1  # 返回概率最大的类别的标签

def classify_samples(X, means, covs, priors):
    """
    对多个样本进行分类
    @param X: 样本数据
    @param means: 各类别的均值向量
    @param covs: 各类别的协方差矩阵
    @param priors: 各类别的先验概率
    @return: 预测的类别标签数组
    """
    return np.array([bayes_classifier(x, means, covs, priors) for x in
X])  # 对每个样本进行分类

class Visualizer:
    """
    可视化类，绘制决策边界和错误率
    """
    @staticmethod
    def plot_decision_boundary(X, y, means, covs, priors, error_rate, d
im):
        """
        绘制决策边界
        """
        plt.figure(figsize=(10, 6) if dim == 1 else (10, 8))  # 设置图像
```

```
大小
        colors = ['r', 'b', 'g']  # 颜色
        labels = ['类别 1', '类别 2', '类别 3']  # 标签

        y = np.array(y)

        if dim == 1:
            Visualizer._plot_1d(X, y, means, covs, priors, colors, labe
ls)
        else:
            Visualizer._plot_2d(X, y, means, covs, priors, colors, labe
ls)

        plt.title(f"{'二' if len(means) == 2 else '三'}类问题: {dim}维特
征的决策边界 (错误率: {error_rate:.2%})")
        plt.xlabel("x1")
        plt.ylabel("x2" if dim > 1 else "")
        plt.legend()
        plt.show()
    ....... 以下绘图代码省略

def run_classification(classes, dims, priors):
    """
    运行分类实验
    @param classes: 包含各个类别数据的列表
    @param dims: 要测试的特征维度列表
    @param priors: 各类别的先验概率
    """
    error_rates = []  # 错误率列表
    for dim in dims:
        X = np.vstack([cls[:, :dim] for cls in classes])  # 将所有类别的
数据合并为一个数组
        true_labels = [i+1 for i, cls in enumerate(classes) for _ in ra
nge(len(cls))]
        means = [estimate_parameters(cls[:, :dim])[0] for cls in classe
s]  # 计算每个类别的均值向量
        covs = [estimate_parameters(cls[:, :dim])[1] for cls in classes]
  # 计算每个类别的协方差矩阵

        predicted_labels = classify_samples(X, means, covs, priors)  #
对每个样本进行分类
        error_rate = calculate_error_rate(true_labels, predicted_labels)
  # 计算错误率
        error_rates.append(error_rate)  # 将错误率添加到列表中

        print(f"{'二' if len(classes) == 2 else '三'}类问题使用{dim}维特
```

```
征的分类器错误率: {error_rate:.2%}")  # 打印错误率
        if dim <= 2:
            Visualizer.plot_decision_boundary(X, true_labels, means, co
vs, priors, error_rate, dim)  # 绘制决策边界
        elif dim == 3:
            Visualizer.plot_decision_boundary_3d(X, true_labels, means,
 covs, priors, error_rate, len(classes))  # 绘制三维决策边界

    Visualizer.plot_error_rates(error_rates)  # 绘制错误率柱状图

def main():
    # 二类问题
    run_classification([class_1, class_2], [1, 2, 3], [0.5, 0.5])

    # 三类问题
    run_classification([class_1, class_2, class_3], [1, 2, 3], [1/3, 1/
3, 1/3])

if __name__ == "__main__":
    plt.rcParams['font.sans-serif'] = ['SimHei']  # 用来正常显示中文标签
    plt.rcParams['axes.unicode_minus'] = False    # 用来正常显示负号
    main()
```