南開大學
Nankai University

计算机学院

机器学习设计实验报告

模式识别与分类

姓名：周重天

学号：2311082

专业：计算机科学与技术

2025 年 11 月 6 日

目录

1 实验一：似然率测试规则（LRT）与最大后验概率规则（MAP）

1.1 实验目标

本实验旨在实现并验证两种贝叶斯分类规则：最大后验概率规则（MAP）和似然率测试规则（LRT）。MAP 规则的决策公式为 $C^* = \arg\max_{C_i}[\log P(X|C_i) + \log P(C_i)]$，LRT 规则的决策准则为 $\log P(X|C_1) - \log P(X|C_0) \gtrless \log P(C_0) - \log P(C_1)$。实验采用高斯朴素贝叶斯框架，在高分离度（class_sep=2.0）和低分离度（class_sep=0.5）数据集上评估分类性能。

1.2 核心代码实现

1.2.1 MAP 分类规则实现

MAP 分类的核心代码如下：

```
1  def predict(self, X):
2      # 1. 计算对数似然
3      log_likelihoods = self._calculate_log_likelihood(X)
4
5      # 2. 计算对数先验
6      log_priors = np.log(self.priors_)
7
8      # 3. 计算对数后验 = 对数似然 + 对数先验
9      log_joint = log_likelihoods + log_priors
10
11     # 4. 选择后验概率最大的类别
12     return np.argmax(log_joint, axis=1)
```

代码分析：第 1 行调用 _calculate_log_likelihood 方法计算每个样本在各类别下的对数似然 $\log P(X|C_i)$，返回形状为 (n_samples, n_classes) 的数组。第 2 行通过对先验概率取对数得到 $\log P(C_i)$。第 3 行利用 NumPy 广播机制将对数先验加到对数似然上，得到对数后验。第 4 行使用 np.argmax 沿 axis=1 找到每个样本的最大后验概率对应的类别索引。

1.2.2 LRT 分类规则实现

LRT 分类的核心代码如下：

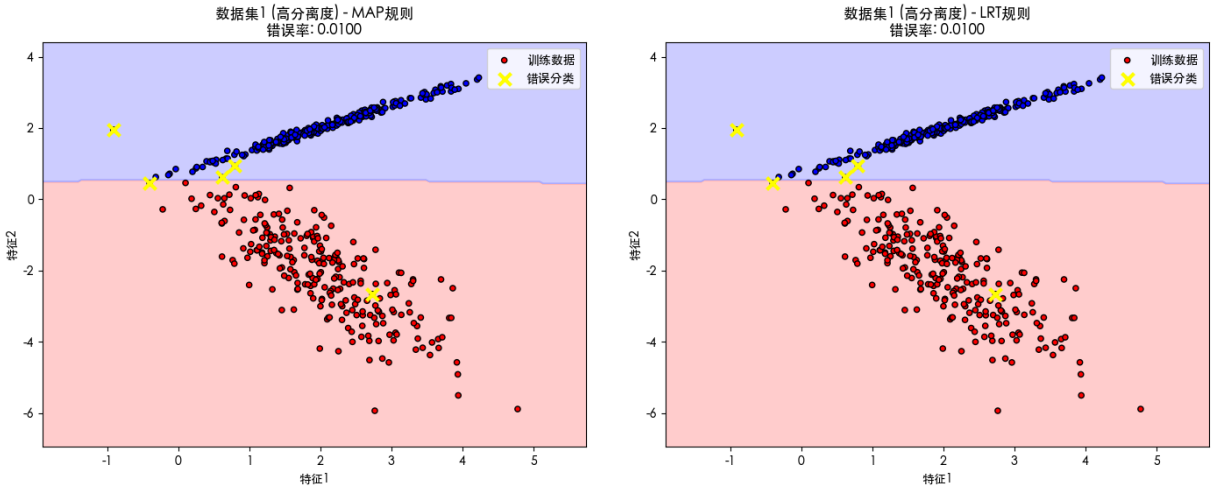
```
1  def apply_LRT_rule(model, X):
2      # 1. 计算对数似然
3      log_likelihoods = model._calculate_log_likelihood(X)
4
5      # 2. 提取两个类别的对数似然
6      log_like_c0 = log_likelihoods[:, 0]
7      log_like_c1 = log_likelihoods[:, 1]
8
9      # 3. 计算对数似然率（左边）
10     log_likelihood_ratio = log_like_c1 - log_like_c0
11
12     # 4. 计算阈值（右边）
```

```
13      log_priors = np.log(model.priors_)
14      log_threshold = log_priors[0] - log_priors[1]
15
16      # 5. 应用LRT规则
17      predictions = (log_likelihood_ratio > log_threshold).astype(int)
18
19      return predictions
```

代码分析：第 2-3 行通过切片操作提取类别 0 和类别 1 的对数似然向量。第 4 行计算对数似然率 $\log P(X|C_1) - \log P(X|C_0)$。第 6 行计算阈值 $\log P(C_0) - \log P(C_1)$。第 8 行通过布尔比较和类型转换实现决策：当似然率超过阈值时预测为类别 1，否则为类别 0。

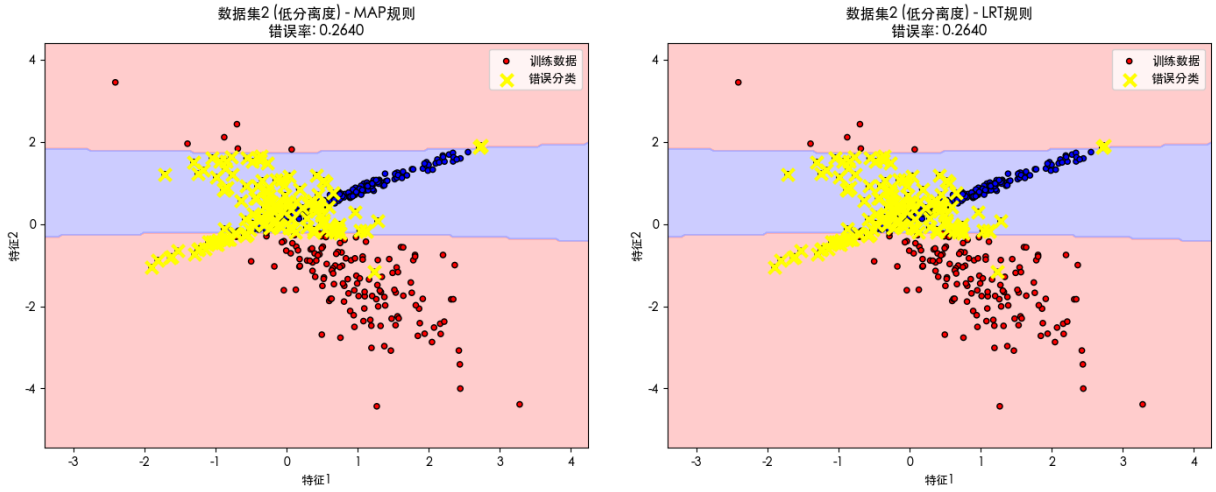
1.3 实验结果分析

1.3.1 高分离度与低分离度数据集对比



(a) 数据集 1 - MAP 规则 (b) 数据集 1 - LRT 规则

图 1.1: 数据集 1（高分离度）分类结果对比，错误率均为 0.0100



(a) 数据集 2 - MAP 规则 (b) 数据集 2 - LRT 规则

图 1.2: 数据集 2（低分离度）分类结果对比，错误率均为 0.2640

从图1.1可见，高分离度数据集中红色点（类别 0）和蓝色点（类别 1）明显分离，仅在决策边界附近有约 5 个黄色叉号标记的错误点，错误率为 1.00%。MAP 和 LRT 产生完全相同的决策边界和预测结果。

图1.2显示低分离度数据集中两类样本高度重叠，特别是在坐标原点附近混杂严重。决策边界两侧分布着大量黄色叉号，表明 132 个样本被错误分类，错误率达 26.40%。两种方法的决策边界和错误点分布完全一致，数值验证了算法等价性。

1.4 实验结论

本实验通过代码实现和实验验证，成功证明了 MAP 规则和 LRT 规则在高斯朴素贝叶斯框架下的理论等价性。两种方法在相同数据集上产生完全相同的错误率和决策边界。数据集的分离度对分类性能有决定性影响，高分离度数据集错误率仅为 1.00%，而低分离度数据集错误率达到 26.40%。高斯朴素贝叶斯在特征独立假设下产生线性决策边界，错误点主要分布在类别重叠区域。

2 实验二：核密度估计与 LRT 分类

2.1 实验目标

本实验旨在实现高斯核密度估计方法并结合 LRT 规则进行分类。核密度估计是一种非参数方法，通过核函数对数据进行密度估计：$p(x) = \frac{1}{N}\sum_{i=1}^{N} K(\frac{x-x_i}{h})$，其中 $K(u) = \exp(-0.5\|u\|^2)$ 为高斯核函数，h 为带宽参数。实验通过 K 折交叉验证寻找最优带宽 h，在高分离度和低分离度数据集上评估非参数密度估计的分类效果。

2.2 核心代码实现

2.2.1 核密度估计实现

核密度估计的核心代码如下：

```
def predict_log_density(self, X_new, h):
    log_densities = []
    for x in X_new:
        # 1. 计算距离
        distances = np.linalg.norm(self.X_train_ - x, axis=1)

        # 2. 应用高斯核
        kernel_vals = np.exp(-0.5 * (distances / h)**2)

        # 3. 求和并取对数
        density_sum = np.sum(kernel_vals) + self._epsilon
        log_density = np.log(density_sum)

        log_densities.append(log_density)

    return np.array(log_densities)
```

代码分析：第 1 行使用 np.linalg.norm 计算待估计点 x 到所有训练样本的欧氏距离，axis=1 表示按行计算向量范数。第 2 行应用高斯核函数 $K(u) = \exp(-0.5(d/h)^2)$，其中 d 为距离，h 为带宽参数，带宽越大核函数越平滑。第 3 行对所有核值求和并加上小常数 epsilon 防止 log(0)，然后取对数得到对数密度估计。这个循环对每个测试点重复上述过程。

2.2.2 基于核密度估计的 LRT 分类

LRT 分类的核心代码如下：

```
def predict(self, X_new, h):
    # 1. 计算两个类别的对数似然
    log_like_c0 = self.kde_c0_.predict_log_density(X_new, h)
    log_like_c1 = self.kde_c1_.predict_log_density(X_new, h)

    # 2. 计算对数似然率
    log_lr = log_like_c1 - log_like_c0

    # 3. 计算阈值
    log_threshold = self.log_prior_c0_ - self.log_prior_c1_

    # 4. 应用LRT规则
    predictions = (log_lr > log_threshold).astype(int)

    return predictions
```

代码分析：第 1-2 行分别调用两个类别的核密度估计器计算对数似然 $\log p(x|C_0)$ 和 $\log p(x|C_1)$。第 3 行计算对数似然率，这是 LRT 决策的核心量。第 4 行从先验概率计算决策阈值。第 5 行执行决策：当对数似然率超过阈值时预测为类别 1，否则为类别 0。

2.2.3 K 折交叉验证选择最优带宽

交叉验证的核心代码如下：

```
for h in h_values:
    fold_accuracies = []
    for train_idx, val_idx in kf.split(X):
        # 1. 分割数据
        X_train, X_val = X[train_idx], X[val_idx]
        y_train, y_val = y[train_idx], y[val_idx]

        # 2. 训练模型
        model_cv = MyKernelClassifier()
        model_cv.fit(X_train, y_train)

        # 3. 预测并计算准确率
        y_pred = model_cv.predict(X_val, h=h)
        acc = accuracy_score(y_val, y_pred)

        fold_accuracies.append(acc)

    mean_accuracy = np.mean(fold_accuracies)
```

代码分析：外层循环遍历候选带宽值 [0.1, 0.5, 1.0, 1.5, 2.0]。内层循环进行 K 折交叉验证，通过 train_idx 和 val_idx 索引分割训练集和验证集。对每个 fold 在训练集上拟合核密度估计器，在验证集上预测并计算准确率。最后对 K 折准确率求平均，选择平均准确率最高的带宽作为最优值。

2.3 实验结果分析

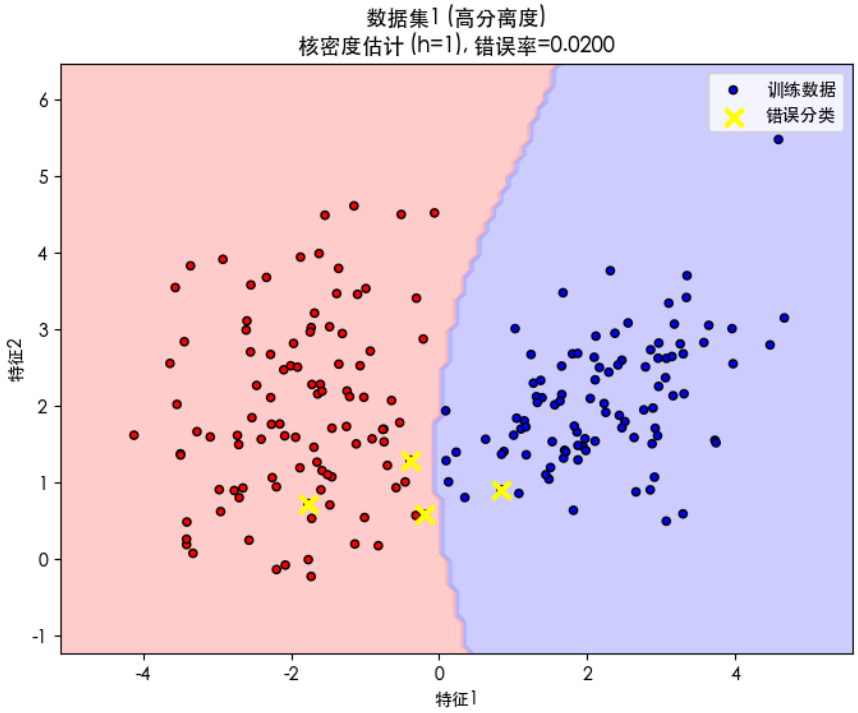


图 2.3: 数据集 1（高分离度）核密度估计分类结果，最优 h=1.0，错误率 =0.0200

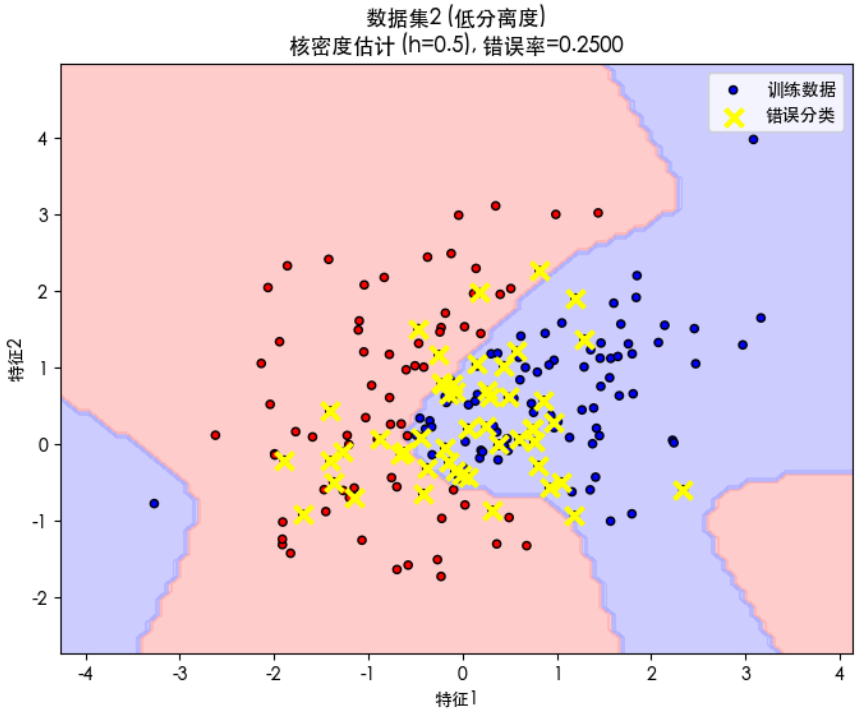


图 2.4: 数据集 2（低分离度）核密度估计分类结果，最优 h=0.5，错误率 =0.2500

从图2.3可见，高分离度数据集通过 5 折交叉验证选择的最优带宽为 h=1.0（平均准确率 98.00%）。决策边界相比实验一的线性边界更加平滑和曲线化，这是核密度估计非参数性质的体现。黄色叉号标

记的错误点约 4 个，错误率为 2.00%，略高于实验一的参数方法（1.00%），这是因为核密度估计需要估计整个密度函数而非仅估计参数，在小样本情况下可能存在过拟合风险。

图2.4显示低分离度数据集的最优带宽为 h=0.5（平均准确率 72.50%），小于高分离度数据集的最优带宽。决策边界呈现不规则的曲线形状，在两类重叠严重的区域形成复杂的分界。密集的黄色叉号表明约 50 个样本被错误分类，错误率为 25.00%，略优于实验一的参数方法（26.40%）。这说明核密度估计在数据分布复杂时可以通过非参数方式捕捉更灵活的决策边界。

2.4 实验结论

通过交叉验证发现，高分离度数据集的最优带宽 h=1.0 较大，对应平滑的决策边界；低分离度数据集的最优 h=0.5 较小，对应更复杂的决策边界。带宽参数需要在偏差和方差之间权衡。核密度估计产生非线性决策边界，在低分离度数据上性能略优于参数方法（25.00% vs 26.40%），但在高分离度数据上略逊（2.00% vs 1.00%），体现了非参数方法的灵活性。核密度估计的时间复杂度为 $O(N \times M)$，高于参数方法。

3 实验三：k-NN 密度估计

3.1 实验目标

本实验旨在实现 k 近邻（k-NN）密度估计方法。k-NN 密度估计也是一种非参数方法，其核心思想是基于局部密度估计：$p(x) = \frac{k}{N \times V}$，其中 k 是近邻数，N 是训练样本总数，V 是包含 k 个近邻的超球体体积。在二维空间中 $V = \pi r_k^2$，r_k 为第 k 个最近邻的距离。实验通过分析不同 k 值（k=1, 3, 5）对密度估计的影响，观察过拟合和欠拟合现象。

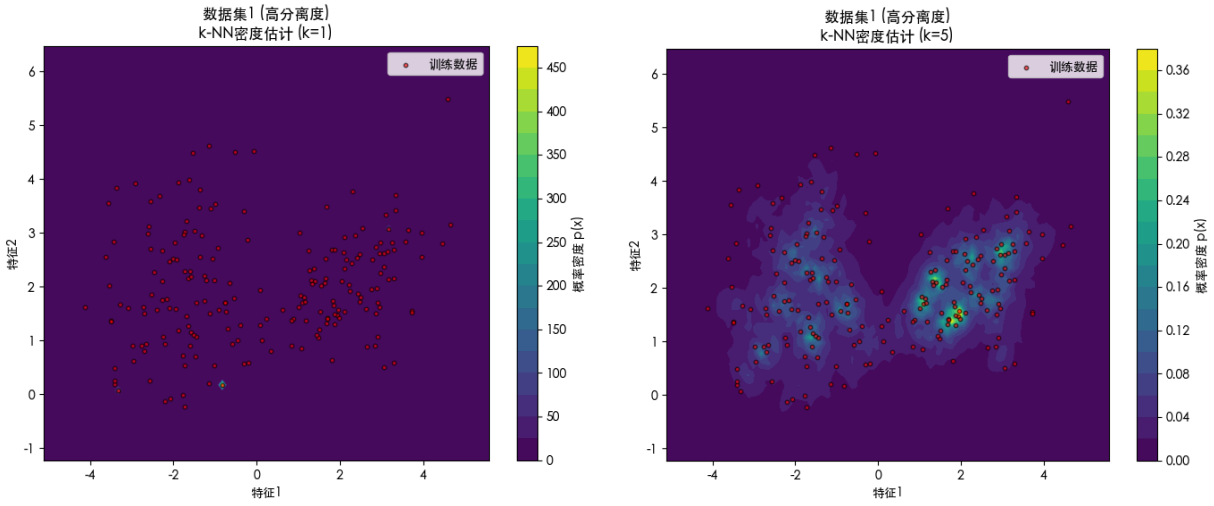
3.2 核心代码实现

k-NN 密度估计的核心代码如下：

```
def predict(self, X_new, k):
    densities = []
    for x in X_new:
        # 1. 计算距离
        distances = np.linalg.norm(self.X_train_ - x, axis=1)

        # 2. 排序并找到第k个距离
        distances_sorted = np.sort(distances)
        r_k = distances_sorted[k-1]  # 注意: k=1时索引是0

        # 3. 计算体积 (2D情况)
        volume = np.pi * r_k**2 + self._epsilon

        # 4. 计算密度
        density = k / (self.N_ * volume)

        densities.append(density)

    return np.array(densities)
```

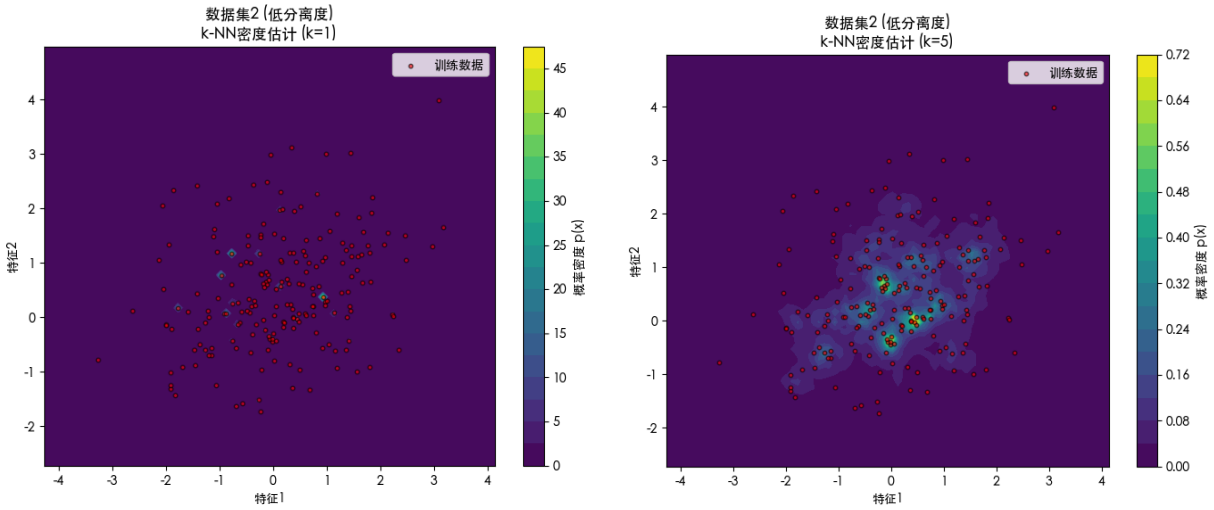
代码分析：第 1 行计算待估计点到所有训练样本的欧氏距离，与核密度估计相同。第 2-3 行对距离排序并提取第 k 个距离 r_k，注意索引从 0 开始所以使用 k-1。第 4 行根据二维圆形体积公式 $V = \pi r_k^2$ 计算包含 k 个近邻的区域体积，加 epsilon 防止除零。第 5 行应用 k-NN 密度估计公式 $p(x) = k/(N \times V)$ 计算密度值。与核密度估计不同，k-NN 方法固定近邻数 k，让体积 V 随局部密度自适应变化：密集区域 r_k 小，稀疏区域 r_k 大。

3.3 实验结果分析



(a) 数据集 1, k=1 (b) 数据集 1, k=5

图 3.5: 数据集 1（高分离度）不同 k 值的密度估计对比



(a) 数据集 2, k=1 (b) 数据集 2, k=5

图 3.6: 数据集 2（低分离度）不同 k 值的密度估计对比

从图3.5可见，高分离度数据集在 k=1 时呈现明显的过拟合特征。左图显示密度热力图几乎完全是深紫色（低密度），仅在训练样本点处出现极小的亮点，密度范围达到 0-450。这是因为 k=1 时只考虑最近的 1 个邻居，导致 r_1 非常小，体积 V 趋近于零，在非样本点处密度极低。右图 k=5 时密度分布

变得平滑，在两个聚类中心形成明显的高密度区域（黄绿色），密度范围降至 0-0.36，体积增大使密度估计更加稳定。两个类别的密度分布清晰可辨。

图3.6显示低分离度数据集的密度估计特性。左图 k=1 时同样出现严重过拟合，密度值仅在样本点处有响应，范围 0-45。右图 k=5 时密度分布平滑，在坐标原点附近形成连续的高密度区域（黄色），密度范围 0-0.72。由于两类样本重叠，密度热力图呈现单峰分布，难以区分两个类别中心。对比数据集 1 的双峰分布，说明 k-NN 密度估计能够反映数据的真实分布特征。

3.4 实验结论

k=1 时密度估计在每个样本点形成尖峰，呈现严重过拟合，密度值范围极大（0-450）。k 增大时密度分布逐渐平滑，k=5 时达到较好的平衡。k-NN 方法固定近邻数让体积自适应，与核密度估计固定体积让近邻数自适应形成对比。在密集区域体积小，稀疏区域体积大，这种自适应性使 k-NN 对不均匀分布的数据更加鲁棒。密度热力图直观展现了数据的分布结构，高分离度数据呈现双峰分布，低分离度数据呈现单峰分布。

4 实验四：贝叶斯分类器在训练集与测试集上的评估

4.1 实验目标

本实验旨在实现贝叶斯分类器并在训练集和测试集上进行评估。与前三个实验不同，本实验采用训练集-测试集划分策略，将数据按 7:3 比例划分为训练集（350 样本）和测试集（150 样本）。实验的核心目标是评估贝叶斯分类器的泛化能力，对比训练错误率和测试错误率，观察模型在未见数据上的表现。

4.2 核心代码实现

贝叶斯分类器的核心代码如下：

```
class GaussianNaiveBayes:
    def fit(self, X, y):
        n_samples = X.shape[0]
        self.classes_ = np.unique(y)
        n_classes = len(self.classes_)
        n_features = X.shape[1]

        self.means_ = np.zeros((n_classes, n_features))
        self.vars_ = np.zeros((n_classes, n_features))
        self.priors_ = np.zeros(n_classes)

        for idx, c in enumerate(self.classes_):
            X_c = X[y == c]
            self.means_[idx] = X_c.mean(axis=0)
            self.vars_[idx] = X_c.var(axis=0)
            self.priors_[idx] = X_c.shape[0] / n_samples

    def predict(self, X):
        log_priors = np.log(self.priors_)
```

```
20          log_likelihoods = []
21
22          for idx in range(len(self.classes_)):
23              mean = self.means_[idx]
24              var = self.vars_[idx]
25              log_likelihood = -0.5 * np.sum(np.log(2 * np.pi * var))
26              log_likelihood -= 0.5 * np.sum(((X - mean) ** 2) / var, axis=1)
27              log_likelihoods.append(log_likelihood)
28
29          log_likelihoods = np.array(log_likelihoods).T
30          log_posteriors = log_likelihoods + log_priors
31          return np.argmax(log_posteriors, axis=1)
```

代码分析：fit 方法在第 8-15 行计算每个类别的均值、方差和先验概率。第 11-15 行遍历每个类别，提取该类别的样本，计算统计量。predict 方法在第 17-30 行实现分类决策。第 24-25 行计算对数似然 $\log p(x|C_i) = -\frac{1}{2}\sum[\log(2\pi\sigma^2) + \frac{(x-\mu)^2}{\sigma^2}]$。第 28-29 行将对数似然与对数先验相加得到对数后验，选择最大值对应的类别。这个实现与实验一相同，但更加简洁。

4.3 实验结果分析

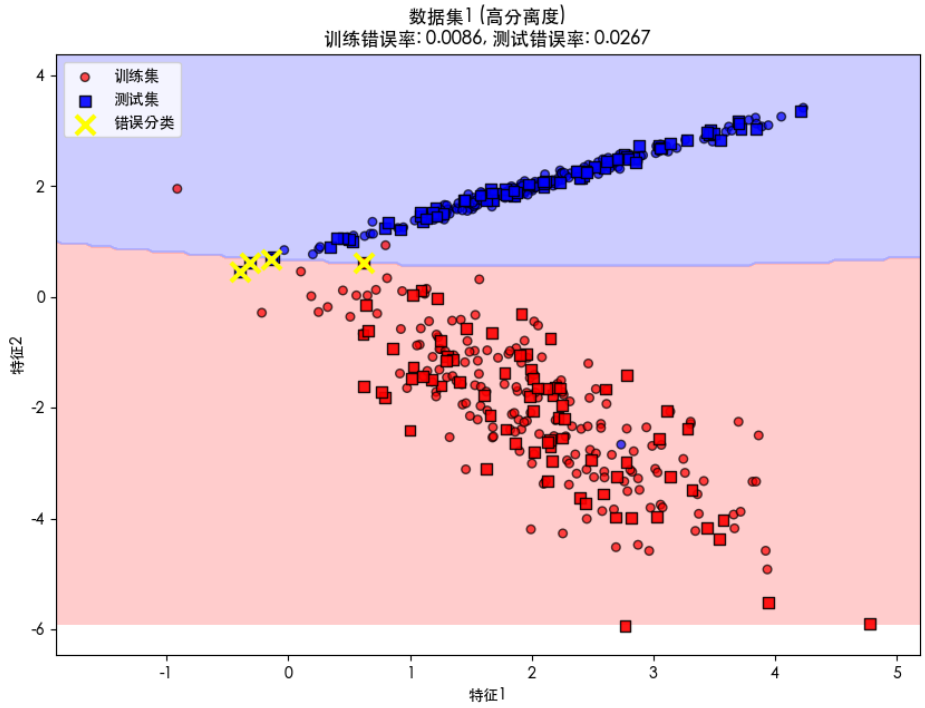


图 4.7: 数据集 1（高分离度）分类结果，训练错误率 =0.0086，测试错误率 =0.0267

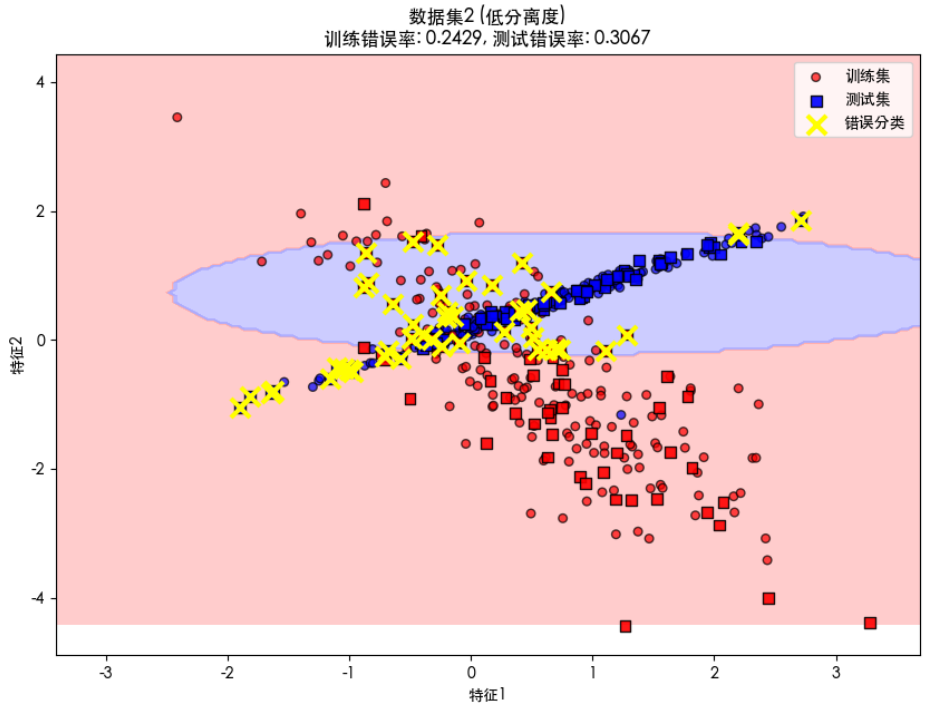


图 4.8: 数据集 2（低分离度）分类结果，训练错误率 =0.2429，测试错误率 =0.3067

从图4.7可见，高分离度数据集上训练错误率为 0.86%，测试错误率为 2.67%。图中红色圆点和蓝色方块分别表示训练集和测试集样本，决策边界清晰地将两类分开。黄色叉号标记的 4 个测试样本被错误分类，这些样本位于决策边界附近的重叠区域。测试错误率略高于训练错误率，差距为 1.81 个百分点，表明模型具有良好的泛化能力。

图4.8显示低分离度数据集上训练错误率为 24.29%，测试错误率为 30.67%。图中密集的黄色叉号分布在特征空间的中心区域，约 46 个测试样本被错误分类。决策边界呈线性形状，但由于两类高度重叠，大量样本处于不确定区域。测试错误率比训练错误率高 6.38 个百分点，这是正常的泛化差距，说明模型没有出现严重过拟合。

4.4 实验结论

通过训练集-测试集划分评估了模型的泛化能力。高分离度数据集的泛化差距为 1.81%，低分离度数据集的泛化差距为 6.38%，均在合理范围内，说明贝叶斯分类器具有较好的泛化性能。与实验一相比，本实验在测试集上错误率略高，这是因为训练样本减少（从 500 到 350）造成的。低分离度数据集的测试错误率（30.67%）是高分离度（2.67%）的 11.5 倍，再次证明数据的内在可分性是决定分类性能的关键因素。