机器学习-点集分类

韩琳
hanlin3309@163.com

Abstract

该报告使用Decision Trees, AdaBoost + DecisionTrees，SVM方法对月亮形状数据集进行分类，并且对于SVM方法尝试了三种核函数，最终Decision Trees, AdaBoost + DecisionTrees，SVM (rbf Kernel)得到较准确的分类结果，SVM (linear Kernel)、SVM (poly Kernel)分类结果不佳，体现了Decision Trees，SVM (rbf Kernel)两种方法在处理非线性数据时具有较好的效果

Introduction

二分类是模式识别的基础任务，广泛用于医疗诊断、欺诈检测等。Decision Trees, AdaBoost + DecisionTrees，SVM等算法是常见的二分类解决算法，决策树易于解释，AdaBoost提升分类性能，非线性核函数的SVM擅长处理高维数据，三者各有优势。

Methodology

一.Decision Tree

决策树是一种流行且功能强大的工具，用于机器学习和数据挖掘，用于分类和回归任务。它是一个树状结构模型，其中内部节点表示对属性的测试，分支表示这些测试的结果，叶节点表示决策结果或类标签。从根到叶的路径表示分类规则或回归路径。

1.核心概念

1.信息熵公式

信息熵（Entropy）是信息论中用来度量信息不确定性或混乱程度的一个指标，由香农（Claude Shannon）提出，其公式为：

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_b p(x_i)$$

1.1各项含义

- $H(X)$：随机变量 X 的信息熵，表示 X 的平均不确定性。
- $p(x_i)$：随机变量 X 取值 x_i 的概率。
- $\log_b$：对数函数的底数，常用底数有：
  - $b = 2$：单位为比特（bits）。
  - $b = e$：单位为纳特（nats）。
  - $b = 10$：单位为哈特莱（hartleys）。
- n：随机变量 X 可能取的不同值的个数。

1.2.公式解读

1. 概率 $p(x_i)$ 的作用：
   每个可能取值的概率决定了该值对整体不确定性的贡献。
2. 对数 $\log_b$ 的作用：
   计算信息量（Information Content），即某个值的“信息价值”。概率越低的信息，包含的信息量越高。
3. 负号的作用：
   保证信息熵是非负值，因为 $\log_b(p(x_i))$ 是负值（当 $0 < p(x_i) < 1$ 时）。

1.3.特殊情况

1. 完全确定性：
   如果 X 总是取同一个值（如 $p(x_1) = 1, p(x_2) = 0, \ldots$），那么信息熵为 0：
   $H(X) = -(1 \cdot \log_b 1 + 0 \cdot \log_b 0 + \ldots) = 0$
   表示没有不确定性。
2. 完全不确定性（均匀分布）：
   如果 X 的每个值出现的概率相等（如 $p(x_i) = \frac{1}{n}$），信息熵最大：
   $H(X) = -\sum_{i=1}^{n} \frac{1}{n} \log_b \frac{1}{n}$

1.4.示例

假设 X 是一个抛硬币实验的结果：

- X 有两个可能取值：正面（Head）和反面（Tail）。

- $p(\text{Head}) = 0.5$，$p(\text{Tail}) = 0.5$。

信息熵为：
$H(X) = -\left(0.5 \cdot \log_2 0.5 + 0.5 \cdot \log_2 0.5\right) = -\left(0.5 \cdot -1 + 0.5 \cdot -1\right) = 1$ 比特（bits）

2.节点类型：

- **根节点（Root Node）**：树的起点，包含整个数据集，并根据某一特征进行首次分裂。
- **内部节点（Internal Node）**：非叶子节点，表示进一步的分裂。
- **叶子节点（Leaf Node）**：最终的分类或预测结果。

3.分裂规则：

决策树的构建基于某种分裂规则，这些规则衡量每个特征的分裂效果。常用指标包括：

- **信息增益（Information Gain）**：基于熵（Entropy），用于衡量分裂前后数据的不确定性变化。

$$IG = \text{Entropy}(D) - \sum_{i=1}^{k} \frac{|D_i|}{|D|} \cdot \text{Entropy}(D_i)$$

- **基尼指数（Gini Index）**：衡量数据集的不纯度，基尼指数越小，分裂效果越好。

$$Gini(D) = 1 - \sum_{i=1}^{C} p_i^2$$

1. **停止条件**：
   - 达到预设的最大深度。
   - 当前节点的样本数量不足。
   - 所有特征均无法进一步分裂。
   - 信息增益或基尼指数小于阈值。
2. **剪枝（Pruning）**：
   为了防止过拟合，对树进行剪枝。分为两种：
   - **预剪枝（Pre-Pruning）**：在构建过程中提前终止生长。
   - **后剪枝（Post-Pruning）**：生成完全树后，通过验证集调整剪枝。

2.算法流程

1. **选择特征分裂点**：
   根据信息增益或基尼指数选择最优特征及分裂点。
2. **递归分裂**：
   将数据划分到各子节点，对每个子节点重复步骤1，直到满足停止条件。

3. **生成叶子节点**：
   对于无法进一步分裂的节点，将其转换为叶子节点，输出分类结果（分类问题）或预测值（回归问题）。

```
# 决策树分类器
dt_clf = DecisionTreeClassifier(random_state=42)
dt_clf.fit(X_train, y_train)
dt_pred = dt_clf.predict(X_test)
print("Decision Tree Performance:")
print(classification_report(y_test, dt_pred))
print("Accuracy:", accuracy_score(y_test, dt_pred))
# 绘制决策树结果
plot_predictions(X_test, y_test, dt_pred, 'Decision Tree Predictions')
```

3.实验结果

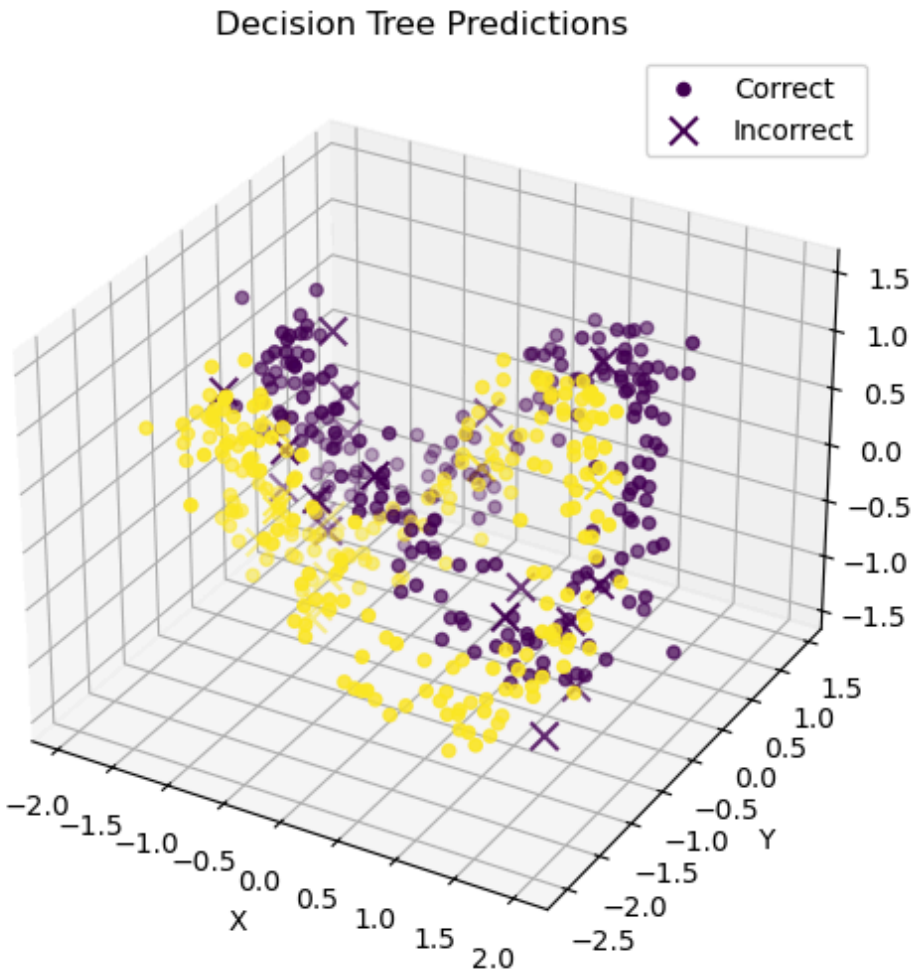
本实验调用**sklearn**库中的 **DecisionTreeClassifier**方法，选取random_state=42的随机等价划分保证结果的可重复性并且控制随即结果，训练结果在测试集上的Accuracy达到0.94

```
Decision Tree Performance:
              precision    recall  f1-score   support

         0.0       0.95      0.92      0.94       250
         1.0       0.93      0.96      0.94       250

    accuracy                           0.94       500
   macro avg       0.94      0.94      0.94       500
weighted avg       0.94      0.94      0.94       500

Accuracy: 0.94
```



二.Decision Trees+AdaBoost

AdaBoost（Adaptive Boosting） 是一种经典的 Boosting 集成学习方法，它通过**多轮训练弱分类器（如决策树）**，将它们组合成一个强分类器。

- **弱分类器（Weak Learner）**：通常是深度为1的决策树，称为“决策树桩”（Decision Stump）。
- **核心思想**：每一轮根据前一轮的分类错误调整样本权重，使分类器关注被错误分类的样本。

1、AdaBoost 算法流程（以二分类为例）

给定训练集：
$(x_1, y_1), \ldots, (x_n, y_n), \quad y_i \in \{-1, +1\}$

初始化样本权重：

$$w_i^{(1)} = \frac{1}{n}, \quad i = 1, 2, \ldots, n$$

对 $m = 1$ 到 M（总轮数）：

1. 训练弱分类器 $G_m(x)$，最小化加权错误率：

$$\varepsilon_m = \sum_{i=1}^{n} w_i^{(m)} \cdot \mathbb{I}(G_m(x_i) \neq y_i)$$

2. **计算弱分类器的权重**（反映其重要性）：

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_m}{\varepsilon_m} \right)$$

3. **更新样本权重**：

$$w_i^{(m+1)} = w_i^{(m)} \cdot \exp\left(-\alpha_m y_i G_m(x_i)\right)$$

然后对 $w_i^{(m+1)}$ 做归一化。

4、最终强分类器

将多个弱分类器加权组合形成最终分类器：

$$F(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

最终预测结果为：

$$\hat{y} = \text{sign}(F(x))$$

2、与普通决策树的对比

特征	决策树（单棵）	AdaBoost（集成）
模型复杂度	单一模型，易过拟合	多个弱模型组合，更鲁棒
偏差	偏差小但方差大	可降低偏差与方差
可解释性	可解释性强	可解释性下降
训练方式	一次训练完成	多轮迭代训练

3.实验结果

```
# AdaBoost + 决策树分类器
ab_clf =
AdaBoostClassifier(estimator=DecisionTreeClassifier(random_state=40),
n_estimators=50, random_state=42)
ab_clf.fit(X_train, y_train)
ab_pred = ab_clf.predict(X_test)
print("\nAdaBoost + Decision Tree Performance:")
print(classification_report(y_test, ab_pred))
print("Accuracy:", accuracy_score(y_test, ab_pred))
plot_predictions(X_test, y_test, ab_pred, 'AdaBoost Predictions')
```

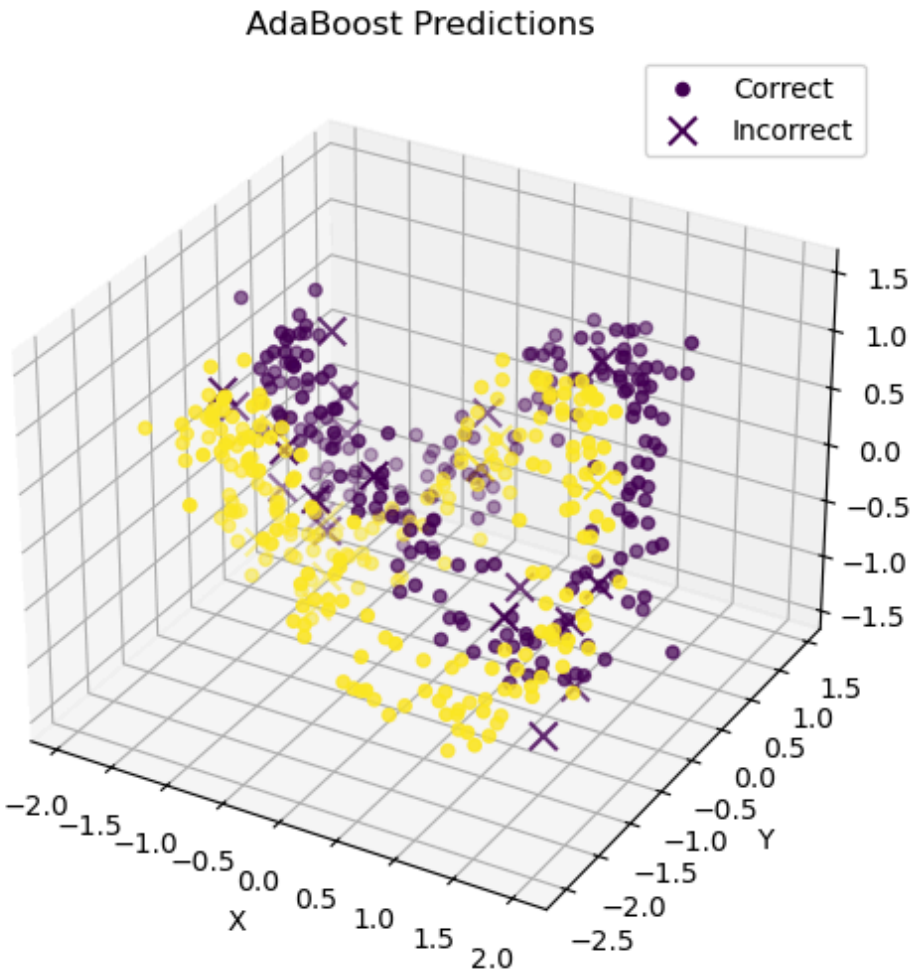
本实验调用**sklearn**库中的 **AdaBoostClassifier**方法，选取random_state=40的随机等价划分子决策树，选用选取random_state=42划分Adaboost本身，保证结果的可重复性并且控制随即结果，n_estimators取50表示使用50个决策树进行集成，训练结果在测试集上的Accuracy达到0.944，准确率略微高于Decision Trees,效果提升不明显

```
AdaBoost + Decision Tree Performance:
              precision    recall  f1-score   support

         0.0       0.97      0.92      0.94       250
         1.0       0.92      0.97      0.95       250

    accuracy                           0.94       500
   macro avg       0.95      0.94      0.94       500
weighted avg       0.95      0.94      0.94       500

Accuracy: 0.944
```



三.SVM

支持向量机（Support Vector Machine, SVM） 是一种经典的有监督机器学习算法，主要用于分类和回归任务。其核心思想是通过在特征空间中寻找一个 最大化间隔的超平面，将不同类别的数据点分开，从而实现对新样本的高效分类。

1、基本概念与思想

1.1 分类问题的目标

给定训练集：

$$(x_i, y_i), \quad i = 1, 2, \ldots, n, \quad x_i \in \mathbb{R}^d,\ y_i \in \{-1, +1\}$$

目标是构建一个最优超平面来最大程度地区分类别。

1.2 超平面定义

$$w^T x + b = 0$$

2、线性可分 SVM（Hard Margin）

2.1 分类约束条件

$$y_i(w^T x_i + b) \geq 1$$

2.2 间隔定义

几何间隔为：

$$\gamma = \frac{2}{\|w\|}$$

2.3 优化目标

$$\min_{w,b} \ \frac{1}{2}\|w\|^2$$
$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

3、线性不可分 SVM（Soft Margin）

引入松弛变量 ξ_i 以允许部分误分类：

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

优化问题变为：

$$\min_{w,b,\xi} \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} \xi_i$$
$$\text{s.t.} \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

4、核技巧（Kernel Trick）

利用核函数将数据映射到高维特征空间：

$$K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

常用核函数：

- 线性核：$K(x, x') = x^T x'$
- 多项式核：$K(x, x') = (x^T x' + c)^d$
- RBF 核（高斯核）：$K(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$
- Sigmoid 核：$K(x, x') = \tanh\left(\alpha x^T x' + c\right)$

5、对偶问题（Dual Problem）

对偶形式如下：

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \ \sum_{i=1}^{n} \alpha_i y_i = 0$$

决策函数：

$$f(x) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b \right)$$

6、SVM 的回归形式（SVR）

SVR 优化目标如下：

$$\min_{w,b} \ \frac{1}{2}\|w\|^2 + C \sum (\xi_i + \xi_i^*)$$

$$\text{s.t.} \quad \begin{cases} y_i - w^T x_i - b \leq \epsilon + \xi_i \\ w^T x_i + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

7.实验结果

```
# SVM 分类器（线性核）
svm_linear = SVC(kernel='linear', random_state=42)
svm_linear.fit(X_train, y_train)
svm_linear_pred = svm_linear.predict(X_test)
print("\nSVM (Linear Kernel) Performance:")
print(classification_report(y_test, svm_linear_pred))
print("Accuracy:", accuracy_score(y_test, svm_linear_pred))
plot_predictions(X_test, y_test, svm_linear_pred, 'SVM (linear Kernel)
Predictions')

# SVM 分类器（多项式核）
#多个degree值的多项式核SVM分类器
svm_poly = SVC(kernel='poly', degree=3, random_state=42)
svm_poly.fit(X_train, y_train)
svm_poly_pred = svm_poly.predict(X_test)
print("\nSVM (Polynomial Kernel) Performance:")
print(classification_report(y_test, svm_poly_pred))
```

```
print("Accuracy:", accuracy_score(y_test, svm_poly_pred))
plot_predictions(X_test, y_test, svm_poly_pred, 'SVM (poly Kernel)
Predictions')

# SVM 分类器（RBF核）
svm_rbf = SVC(kernel='rbf', random_state=42)
svm_rbf.fit(X_train, y_train)
svm_rbf_pred = svm_rbf.predict(X_test)
print("\nSVM (RBF Kernel) Performance:")
print(classification_report(y_test, svm_rbf_pred))
print("Accuracy:", accuracy_score(y_test, svm_rbf_pred))
plot_predictions(X_test, y_test, svm_poly_pred, 'SVM (rbf Kernel)
Predictions')
```

本实验调用**sklearn**库中的 **SVM**方法，选取random_state=42,并且使用linear、poly、rbf三种核函数，其中poly核函数调用了3次多项式进行拟合，得到三个核函数的预测结果，其中rbf核函数准确率最高，poly核函数其次，linear核函数方法准确率最低

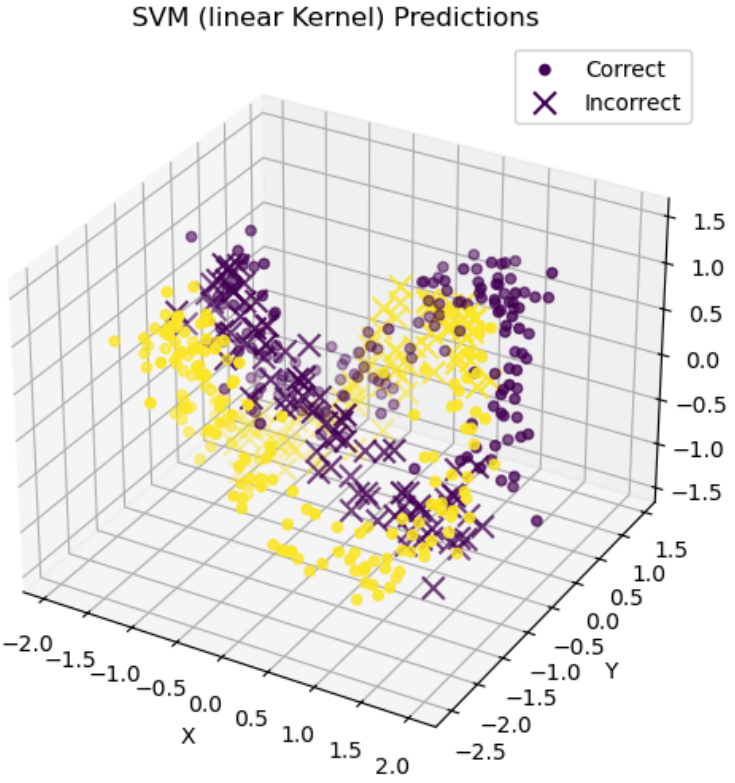
- **linear kernel**

```
SVM (Linear Kernel) Performance:
              precision    recall  f1-score   support

         0.0       0.67      0.66      0.66       250
         1.0       0.66      0.67      0.67       250

    accuracy                           0.67       500
   macro avg       0.67      0.67      0.67       500
weighted avg       0.67      0.67      0.67       500

Accuracy: 0.666
```



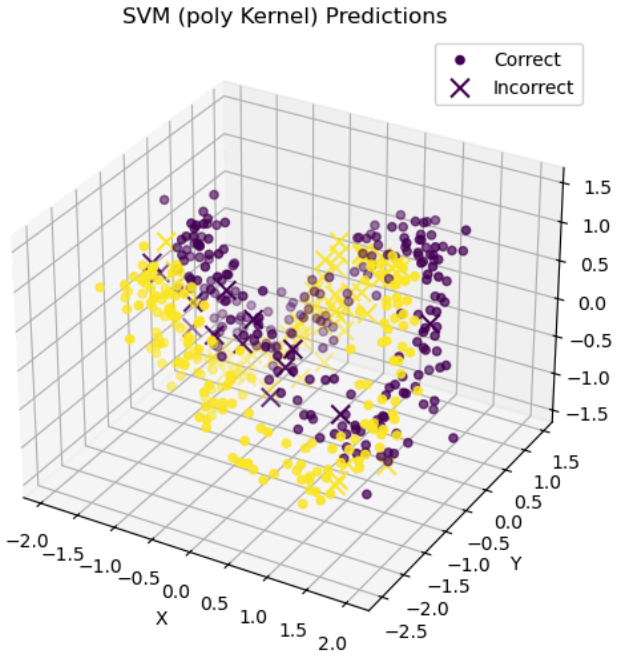
- **poly kernel**

```
SVM (Polynomial Kernel) Performance:
              precision    recall  f1-score   support

         0.0       0.79      0.95      0.86       250
         1.0       0.93      0.74      0.83       250

    accuracy                           0.85       500
   macro avg       0.86      0.85      0.84       500
weighted avg       0.86      0.85      0.84       500

Accuracy: 0.846
```



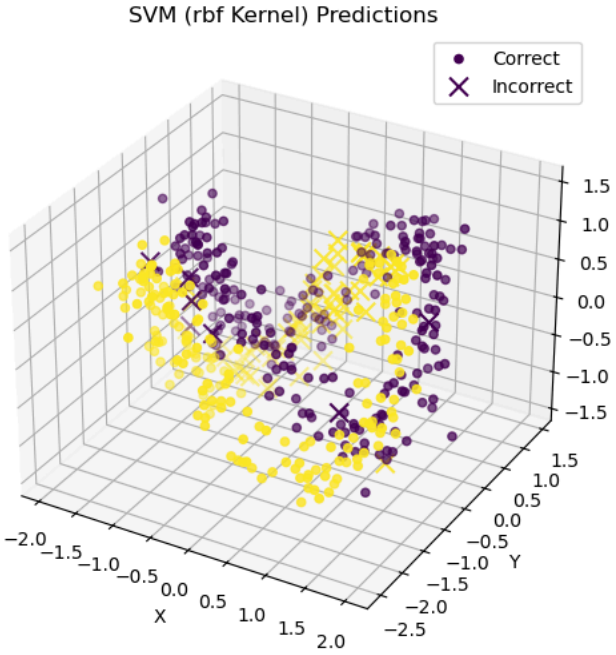
- **rbf kernel**

```
SVM (RBF Kernel) Performance:
              precision    recall  f1-score   support

         0.0       0.98      0.96      0.97       250
         1.0       0.96      0.98      0.97       250

    accuracy                           0.97       500
   macro avg       0.97      0.97      0.97       500
weighted avg       0.97      0.97      0.97       500

Accuracy: 0.972
```



Conclusions

本实验使用的几种分类方法中，Decision Trees, AdaBoost + DecisionTrees，RBF SVM分类准确率达到了0.94以上，其中RBF SVM方法分类效果最好，这体现了**高斯核SVM**在处理非线性数据二分类时的优势，能够在超平面高维度将数据准确划分，**多项式核方法以及线性核方法**准确率不佳，体现了这两种核函数在处理简单非线性数据时的局限性，启示我使用SVM方法时要根据数据的先验特征选取适合的核函数方法核参数，**决策树**算法在处理非线性问题时也同样体现出较好的性能，**AdaBoost** 通过集成弱决策树，在非线性拟合和泛化能力上优于单决策树。若数据接近线性，线性核 SVM 足够高效；若存在复杂非线性，高斯核 SVM 和集成方法（如 AdaBoost）更优。

方法	非线性拟合能力	泛化能力	参数复杂度	噪声鲁棒性	决策边界特性	适用数据类型
高斯核 SVM	最强（无限维）	强（间隔最大化）	低（2 参数）	强	连续平滑曲面	复杂非线性数据
AdaBoost 决策树	较强（集成分裂）	较强（降低方差）	中（基学习器参数）	中等	集成阶梯边界	中等非线性数据
决策树	中等（单棵分裂）	弱（易过拟合）	低（剪枝参数）	弱	单棵阶梯边界	简单非线性数据
多项式核 SVM	有限（固定多项式）	弱（高次过拟合）	高（3 参数）	弱	固定多项式曲面	明确多项式关系数据
线性核 SVM	无（仅线性）	中等（依赖 C）	最低（1 参数）	中等	线性超平面	严格线性可分数据