# 2. 3D 数据集上的分类算法比较

Author : Xinyu Zhao,Beihang University
Email:zxy2021@buaa.edu.cn



## 摘要

本实验旨在比较三种不同的机器学习算法——逻辑回归（Logistic Regression）、支持向量机（SVM）以及XGBoost——在3D数据集上的分类性能。我们首先生成了一个包含1000个数据点的3D数据集，并将其分为两大类：C0和C1。随后，我们使用这些数据进行训练，并生成了一个新的500个数据点的测试集，以评估不同算法的性能。实验结果表明，不同算法在该数据集上的表现存在显著差异。

## 1. 引言

在模式识别和机器学习领域，选择合适的分类算法对于提高分类任务的准确性至关重要，逻辑回归、支持向量机和XGBoost是三种常用的分类算法，它们各自具有不同的特性和优势。本实验通过在3D数据集上的实验，探讨这些算法的分类性能及其适用性。

## 2. 实验方法

### 2.1 数据集生成

我们使用了一个自定义函数`make_moons_3d`来生成数据集。该函数生成了两个半圆形的3D数据集（如下图所示），类似于二维空间中的“月亮”数据集，但增加了一个维度以增加复杂性。

```python
# Generating 3D make-moons data

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

def make_moons_3d(n_samples=500, noise=0.1):
    # Generate the original 2D make_moons data
```

```
    t = np.linspace(0, 2 * np.pi, n_samples)
    x = 1.5 * np.cos(t)
    y = np.sin(t)
    z = np.sin(2 * t)  # Adding a sinusoidal variation in the third dimension

    # Concatenating the positive and negative moons with an offset and noise
    X = np.vstack([np.column_stack([x, y, z]), np.column_stack([-x, y - 1, -z])])
    y = np.hstack([np.zeros(n_samples), np.ones(n_samples)])

    # Adding Gaussian noise
    X += np.random.normal(scale=noise, size=X.shape)

    return X, y

# Generate the data (1000 datapoints)
X, labels = make_moons_3d(n_samples=1000, noise=0.2)

# Plotting
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=labels, cmap='viridis', marker='o')
legend1 = ax.legend(*scatter.legend_elements(), title="Classes")
ax.add_artist(legend1)
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.title('3D Make Moons')
plt.show()
```

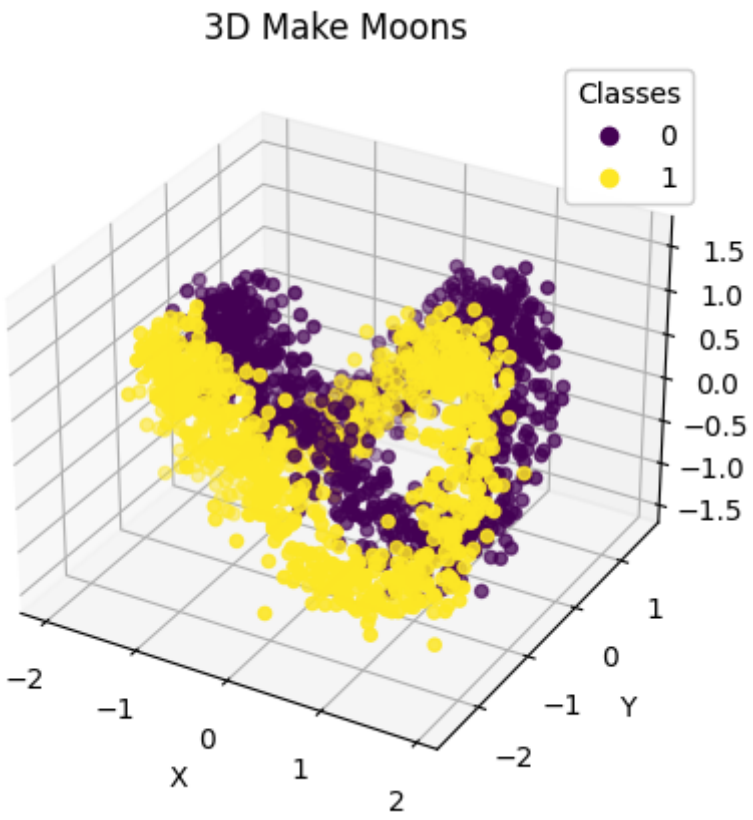


图1 make_moons_3d 数据分布

## 2.2 算法选择

我们选择了三种算法进行比较：

- **逻辑回归**：一种线性分类算法，适用于二分类问题。
- **支持向量机**：一种强大的分类算法，通过选择不同的核函数可以处理非线性问题。
- **XGBoost**：一种基于梯度提升决策树的集成学习算法，能够处理复杂的非线性关系。

## 2.3 实验设置

实验中，我们首先将数据集分为训练集和验证集，然后使用训练集对每种算法进行训练，并在验证集上进行测试。

```
# 生成额外的500个样本数据集
X_test, labels_test = make_moons_3d(n_samples=500, noise=0.2)

# 区分训练集和验证集
X_train, X_val, y_train, y_val = train_test_split(X, labels, test_size=0.2,
random_state=42)
```

# 3. 实验结果

## 3.1 逻辑回归

逻辑回归模型在测试集上的准确率为 0.679。

```
# 逻辑回归
lr_model = LogisticRegression()
lr_model.fit(X_train, y_train)
lr_pred = lr_model.predict(X_test)
lr_accuracy = accuracy_score(labels_test, lr_pred)
```

## 3.2 支持向量机

```
# 支持向量机
svm_kernels = ['linear', 'poly', 'rbf']
svm_models = []
for kernel in svm_kernels:
    svm_model = SVC(kernel=kernel)
    svm_model.fit(X_train, y_train)
    svm_models.append(svm_model)
```

我们测试了三种不同的核函数：线性、多项式和径向基函数（RBF）。每种核函数对应的准确率分别为 [0.679,0.866,0.981]。

## 3.3 XGBoost

```
# XGBoost
xgb_model = XGBClassifier()
xgb_model.fit(X_train, y_train)
xgb_pred = xgb_model.predict(X_test)
xgb_accuracy = accuracy_score(labels_test, xgb_pred)
```

XGBoost模型在测试集上的准确率为 0.98。

## 3.4 分类可视化

为保证可视化效果，我将高维数据投影到了二维平面，更加直观的展现出分类效果，如果点的颜色和区域颜色相同，证明分类正确，反之为分类错误。

```
# 绘图部分
def plot_decision_boundary(ax, model, X, y):
    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                         np.arange(y_min, y_max, 0.1))
    Z = model.predict(np.c_[xx.ravel(), yy.ravel(),
np.zeros_like(xx.ravel())])
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, alpha=0.4, cmap='viridis')
    ax.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis', marker='o')

fig, axs = plt.subplots(1, 5, figsize=(20, 5))

# Logistic Regression
plot_decision_boundary(axs[0], lr_model, X_test[:, :2], labels_test)
axs[0].set_title('Logistic Regression')
axs[0].set_xlabel('X')
axs[0].set_ylabel('Y')
```

```
# SVMs
for i, (model, kernel) in enumerate(zip(svm_models, svm_kernels), start=1):
    plot_decision_boundary(axs[i], model, X_test[:, :2], labels_test)
    axs[i].set_title(f'SVM ({kernel.capitalize()} Kernel)')
    axs[i].set_xlabel('X')
    axs[i].set_ylabel('Y')

# XGBoost
plot_decision_boundary(axs[4], xgb_model, X_test[:, :2], labels_test)
axs[4].set_title('XGBoost')
axs[4].set_xlabel('X')
axs[4].set_ylabel('Y')

plt.tight_layout()
plt.show()
```

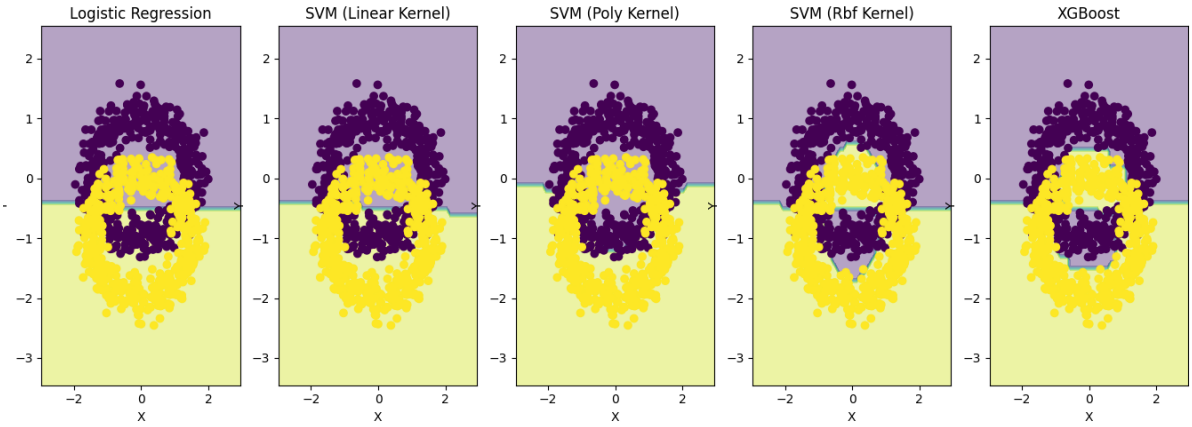


图2 五种分类方法结果可视化

# 4. 讨论

在本实验中，我们观察到XGBoost算法在测试集上的表现最佳，准确率达到了0.98。这一结果可能归因于XGBoost算法在处理复杂非线性关系时的高效性，以及其在集成学习框架下对过拟合的有效控制。相比之下，SVM算法在采用RBF核函数时表现出了最高的准确率0.981，这表明在非线性问题上，SVM能够有效地找到最优的决策边界。而多项式核函数的性能略低于RBF，这可能是因为多项式核函数在处理高维空间时容易受到维度灾难的影响。

逻辑回归模型的准确率为0.679，相对较低，这可能是由于逻辑回归模型的线性假设限制了其在非线性问题上的表达能力。然而，逻辑回归模型的简单性和计算效率使其在某些情况下仍然是一个有用的工具。

## 4.1 数据集特性分析

3D数据集的特性对于算法的选择具有重要影响。在本实验中，数据集的非线性特性使得基于线性决策边界的算法（如逻辑回归）在分类性能上不如基于非线性决策边界的算法（如SVM和XGBoost）。

## 4.2 算法性能对比

SVM和XGBoost在本实验中表现出了优越的性能，尤其是在采用合适的核函数或算法配置时。这表明在实际应用中，选择合适的算法并对其进行适当的调整是至关重要的。

## 4.3 结果讨论

尽管XGBoost在本实验中表现最佳，但在选择算法时还需要考虑其他因素，如模型的解释性、训练时间和资源消耗。逻辑回归模型虽然在本实验中表现不佳，但其模型的简单性和解释性在某些应用场景下可能更为重要。

# 5. 结论

本实验表明，在3D数据集上的分类任务中，XGBoost算法展现出了最高的准确率，而SVM在采用合适的核函数时也能取得很好的性能。逻辑回归模型虽然在本实验中表现不佳，但由于其简单易实现，在线性问题上仍然是一个有效的选择。未来的工作可以探索更多的算法和参数调整，以进一步提高分类性能。