西安电子科技大学

模式识别大作业

题　　目：SVM算法在两种经典数据集上的分类性能验证

姓　　名：X X X

学　　号：2300920XXXX

专　　业：

摘要

本实验旨在验证支持向量机（SVM）算法在不同类型数据集上的分类性能，选取 Iris 多分类数据集和 Sonar 二分类数据集作为实验对象，分别测试线性核、多项式核和高斯核（RBF）三种常用核函数对分类结果的影响。通过数据标准化预处理、模型训练与测试，以准确率、精确率、召回率和 F1 分数为核心评价指标，系统分析 SVM 算法及不同核函数的适用场景。实验结果表明：在低维、类别区分度高的 Iris 数据集上，高斯核函数表现最优，准确率达到 100%，线性核与多项式核准确率均为 96.67%；在高维、样本边界存在一定重叠的 Sonar 数据集上，多项式核与高斯核表现一致且最优，准确率均为 90.48%，显著高于线性核的 83.33%。本实验为 SVM 算法在多分类与二分类任务中的核函数选择提供了直接的实践依据。

关键词： 支持向量机；核函数；Iris 数据集；Sonar 数据集；分类性能

一．绪论

支持向量机（SVM）是一种基于统计学习理论的机器学习算法，通过寻找最优分离超平面实现样本分类，在小样本、高维数据分类任务中具有独特优势。其核心思想是将样本映射到高维特征空间，使原本线性不可分的数据变得线性可分，而核函数的引入巧妙规避了高维空间中的复杂计算，成为 SVM 算法的关键组成部分。

不同核函数决定了 SVM 的映射方式和分类能力：线性核适用于线性可分数据，计算效率高；多项式核通过引入高次项实现非线性分类，参数调节灵活；高斯核则能将数据映射到无穷维空间，对非线性关系的拟合能力最强。

Iris 数据集作为经典的多分类数据集，具有特征维度低、类别边界清晰的特点；Sonar 数据集为二分类数据集，特征维度较高且样本存在一定的重叠性，两类数据集的差异为验证 SVM 及不同核函数的适用性提供了良好场景。本次实验通过在两类数据集上的对比测试，深入分析核函数选择对 SVM 分类性能的影响，为实际分类任务中的模型参数调优提供实验依据。

二．算法介绍

2.1 SVM 算法基本原理

SVM 的核心目标是在特征空间中找到一个最优分离超平面，使得两类样本到超平面的最小距离（间隔）最大化，该间隔被称为 “最大间隔”，距离超平面最近

的样本点称为 “支持向量”。对于线性可分数据，最优超平面可通过求解凸二次规划问题得到；对于线性不可分数据，通过引入松弛变量和惩罚系数处理异常点，同时利用核函数将数据映射到高维特征空间，间接实现线性分离。

2.2 核函数类型及原理

本次实验选取三种常用核函数，其数学表达式及特点如下：

线性核（linear）：直接在原始特征空间中构建线性分离超平面，数学表达式为

$$K(x_i, x_j) = x_i \cdot x_j$$

计算效率最高，适用于线性可分或近似线性可分的数据。

多项式核（poly）：通过多项式映射实现非线性分类，数学表达式为

$$K(x_i, x_j) = (\gamma x_i \cdot x_j + r)^d$$

（其中γ为核系数，r为截距项，d为多项式次数）。

可通过调节参数适配不同复杂度的非线性关系，但参数选择对性能影响较大。

高斯核（rbf）：又称径向基核函数，通过指数函数实现非线性映射，数学表达式为

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$$

（其中γ为核系数）。

具有强大的非线性拟合能力，无需手动调节高次项参数，适用于特征维度高、样本分布复杂的数据。

2.3 评价指标

为全面评估分类性能，选取以下指标：

准确率：正确分类的样本数占总样本数的比例，反映模型整体分类效果。

精确率：预测为正类的样本中实际为正类的比例，衡量模型预测结果的准确性。

召回率：实际为正类的样本中被正确预测的比例，衡量模型对正类样本的捕捉能力。F1 分数：精确率和召回率的调和平均数，综合反映模型的分类稳定性。

三．实验过程

3.1 实验环境

编程语言：Python 3.x

核心库：scikit-learn（数据加载、模型构建、性能评估）、NumPy（数据处理）

硬件环境：普通 PC 机（CPU：Intel Core i5，内存：8GB）

3.2 数据集介绍与预处理

3.2.1 数据集说明

Iris 数据集：包含 3 类鸢尾花样本，共 150 个样本，每类 50 个样本，每个样本包含 4 个特征（花萼长度、花萼宽度、花瓣长度、花瓣宽度），属于低维、多分类数据集。

Sonar 数据集：包含岩石（Rock）和水雷（Mine）两类样本，共 208 个样本，每个样本包含 60 个特征（声纳信号的能量值），属于高维、二分类数据集。

3.2.2 数据预处理

数据加载：通过 scikit-learn 库直接加载 Iris 数据集，通过 fetch_openml 函数加载 Sonar 数据集。

数据标准化：由于 SVM 算法对特征尺度敏感，采用 StandardScaler 对两类数据集的特征进行标准化处理，使每个特征的均值为 0、方差为 1，消除特征尺度差异对模型的影响。

数据集划分：采用 train_test_split 函数将数据集按 8:2 的比例划分为训练集和测试集，其中随机种子设为 42，确保实验可重复性。

3.3 实验步骤

1. 数据集加载与预处理：完成两类数据集的加载、标准化和划分。
2. 模型构建与训练：分别构建基于线性核、多项式核和高斯核的 SVM 模型，使用训练集进行模型训练，保持随机种子为 42。
3. 模型测试：使用测试集对训练好的模型进行预测，记录预测结果。
4. 性能评估：计算各模型的准确率、精确率、召回率和 F1 分数，生成分类报告。
5. 结果分析：对比不同核函数在两类数据集上的性能差异，总结核函数选择规律。

3.4 核心代码实现

代码
import numpy as np from sklearn.datasets import load_iris, fetch_openml from sklearn.model_selection import train_test_split

```
from sklearn.svm import SVC

from sklearn.metrics import accuracy_score, classification_report

from sklearn.preprocessing import StandardScaler


# ---------------------- 1. 加载并预处理数据集 ----------------------

# 加载Iris数据集

iris = load_iris()

X_iris, y_iris = iris.data, iris.target

# 加载Sonar数据集

sonar = fetch_openml(name='sonar', version=1)

X_sonar, y_sonar = sonar.data, sonar.target


# 数据标准化

scaler_iris = StandardScaler()

X_iris_scaled = scaler_iris.fit_transform(X_iris)

scaler_sonar = StandardScaler()

X_sonar_scaled = scaler_sonar.fit_transform(X_sonar)


# 划分训练集和测试集（8:2）

X_iris_train, X_iris_test, y_iris_train, y_iris_test =
train_test_split(

    X_iris_scaled, y_iris, test_size=0.2, random_state=42
```

```
)

X_sonar_train, X_sonar_test, y_sonar_train, y_sonar_test =
train_test_split(

    X_sonar_scaled, y_sonar, test_size=0.2, random_state=42

)



# ---------------------- 2. 定义SVM模型并测试不同核函数 ---------------
-------

kernels = ['linear', 'poly', 'rbf'] # 三种核函数



# ===== Iris数据集（多分类）测试 =====

print("===== Iris数据集 SVM 分类结果 =====")

for kernel in kernels:

    svm_iris = SVC(kernel=kernel, random_state=42)

    svm_iris.fit(X_iris_train, y_iris_train)

    y_iris_pred = svm_iris.predict(X_iris_test)

    acc = accuracy_score(y_iris_test, y_iris_pred)

    print(f"核函数: {kernel}, 准确率: {acc:.4f}")

    print(classification_report(y_iris_test, y_iris_pred,
target_names=iris.target_names))

    print("-" * 50)



# ===== Sonar数据集（二分类）测试 =====
```

```
    print("===== Sonar数据集 SVM 分类结果 =====")

    for kernel in kernels:

        svm_sonar = SVC(kernel=kernel, random_state=42)

        svm_sonar.fit(X_sonar_train, y_sonar_train)

        y_sonar_pred = svm_sonar.predict(X_sonar_test)

        acc = accuracy_score(y_sonar_test, y_sonar_pred)

        print(f"核函数: {kernel}, 准确率: {acc:.4f}")

        print(classification_report(y_sonar_test, y_sonar_pred,
    target_names=['Rock', 'Mine']))

print("-" * 50)
```

四、实验结果与分析

4.1 Iris 数据集实验结果

核函数	准确率	精确率（加权）	召回率（加权）	F1 分数（加权）
线性核	0.9667	0.97	0.97	0.97
多项式核	0.9667	0.97	0.97	0.97
高斯核	1.0000	1.00	1.00	1.00

结果分析：

Iris 数据集特征维度低、类别边界清晰，三种核函数均取得了优异的分类效果。其中高斯核表现最优，准确率、精确率、召回率和 F1 分数均达到 100%，完美实现了三类样本的精准分类。线性核和多项式核的准确率均为 96.67%，略低于高斯核：

线性核在 versicolor 类别的召回率为 89%，存在 1 例样本分类错误；

多项式核在 virginica 类别的召回率为 91%，同样存在 1 例样本分类错误。

这一结果表明，对于低维、线性可分性强的数据，高斯核的非线性映射能力能够更好地捕捉样本间的细微差异，而线性核和多项式核在简单数据场景下的性能差异较小，仅在个别边界样本的分类上存在偏差。

4.2 Sonar 数据集实验结果

核函数	准确率	精确率（加权）	召回率（加权）	F1 分数（加权）
线性核	0.8333	0.86	0.83	0.84
多项式核	0.9048	0.91	0.90	0.91
高斯核	0.9048	0.91	0.90	0.91

结果分析：

Sonar 数据集特征维度高（60 维），且岩石与水雷的声纳信号特征存在部分重叠，属于非线性可分数据，核函数的选择对分类性能影响显著：

多项式核与高斯核表现一致且最优，准确率均达到 90.48%，精确率、召回率和 F1 分数均在 0.90 以上，能够有效捕捉数据中的非线性关系；

线性核的准确率为 83.33%，显著低于前两者，其在 Mine 类别的精确率仅为 71%，表明线性核难以处理高维数据中的复杂非线性关系，导致部分样本分类错误；

从类别细节来看，Rock 类别的精确率（0.96）高于 Mine 类别（0.83），这与数据集中 Rock 样本数量（26 例）多于 Mine 样本数量（16 例）有关，样本分布不

均衡对少数类别的分类性能产生了一定影响。

4.3 核函数性能对比总结

线性核：适用于低维、线性可分数据，计算效率高、模型简洁，但在高维非线性数据上性能受限，仅能捕捉样本间的线性关系。

多项式核：非线性拟合能力较强，在高维复杂数据上表现优异，但性能受多项式次数、核系数等参数影响较大，需通过调参进一步优化。

高斯核：通用性最强，无需手动设置高次项参数，在低维简单数据和高维复杂数据上均表现出色，尤其适合处理样本边界模糊、非线性关系显著的数据，是本次实验中综合性能最优的核函数。

五、实验结论与展望

5.1 实验结论

本次实验通过在 Iris 和 Sonar 两类不同特性的数据集上验证 SVM 算法及三种核函数的分类性能，得出以下结论：

SVM 算法在多分类（Iris）和二分类（Sonar）任务中均表现出良好的分类能力，尤其在低维清晰数据和高维复杂数据上分别通过合适的核函数实现了优异性能。

核函数的选择需紧密结合数据集特性：线性核优先用于低维线性可分数据；高斯核适用于高维、非线性数据；多项式核需明确数据多项式分布特性并调参后使用。

数据预处理对 SVM 算法至关重要，标准化处理能够消除特征尺度差异，显著提升模型分类性能。

附录：代码

```
import numpy as np

from sklearn.datasets import load_iris, fetch_openml

from sklearn.model_selection import train_test_split

from sklearn.svm import SVC

from sklearn.metrics import accuracy_score,
classification_report

from sklearn.preprocessing import StandardScaler



# ---------------------- 1. 加载并预处理数据集 ------------------
---

# 加载Iris数据集

iris = load_iris()

X_iris, y_iris = iris.data, iris.target

# 加载Sonar数据集

sonar = fetch_openml(name='sonar', version=1)

X_sonar, y_sonar = sonar.data, sonar.target



# 数据标准化

scaler_iris = StandardScaler()

X_iris_scaled = scaler_iris.fit_transform(X_iris)

scaler_sonar = StandardScaler()

X_sonar_scaled = scaler_sonar.fit_transform(X_sonar)
```

```
# 划分训练集和测试集（8:2）
X_iris_train, X_iris_test, y_iris_train, y_iris_test =
train_test_split(
    X_iris_scaled, y_iris, test_size=0.2, random_state=42
)
X_sonar_train, X_sonar_test, y_sonar_train, y_sonar_test =
train_test_split(
    X_sonar_scaled, y_sonar, test_size=0.2, random_state=42
)


# ---------------------- 2. 定义SVM模型并测试不同核函数 ---------
------------
kernels = ['linear', 'poly', 'rbf']  # 三种核函数


# ===== Iris数据集（多分类）测试 =====
print("===== Iris数据集 SVM 分类结果 =====")
for kernel in kernels:
    svm_iris = SVC(kernel=kernel, random_state=42)
    svm_iris.fit(X_iris_train, y_iris_train)
    y_iris_pred = svm_iris.predict(X_iris_test)
    acc = accuracy_score(y_iris_test, y_iris_pred)
    print(f"核函数: {kernel}, 准确率: {acc:.4f}")
```

```
    print(classification_report(y_iris_test, y_iris_pred,
target_names=iris.target_names))

    print("-" * 50)


# ===== Sonar数据集（二分类）测试 =====

print("===== Sonar数据集 SVM 分类结果 =====")

for kernel in kernels:

    svm_sonar = SVC(kernel=kernel, random_state=42)

    svm_sonar.fit(X_sonar_train, y_sonar_train)

    y_sonar_pred = svm_sonar.predict(X_sonar_test)

    acc = accuracy_score(y_sonar_test, y_sonar_pred)

    print(f"核函数: {kernel}, 准确率: {acc:.4f}")

    print(classification_report(y_sonar_test, y_sonar_pred,
target_names=['Rock', 'Mine']))

    print("-" * 50)
```