



南京理工大学

NANJING UNIVERSITY OF SCIENCE & TECHNOLOGY

# 机器学习实验报告

基于支持向量机的 wine 数据集分类算法设计与实现

# 基于支持向量机的 wine 数据集分类算法设计与实现

## 1 摘要

本研究旨在利用支持向量机（SVM）对 wine 数据集进行多分类。通过对数据进行标准化预处理，采用带有 RBF 核的 SVM 模型，并结合网格搜索进行超参数调优，最终在测试集上达到了 98% 的准确率。结果表明，SVM 在该分类任务上表现出色，具有良好的泛化能力。

## 2 相关理论与技术

本实验的实现，离不开背后坚实的机器学习理论支撑，以下将详细阐述本次实验所涉及的核心算法原理。

### 2.1 支持向量机理论

支持向量机是一种功能强大的二分类模型，其核心思想可以概括为：寻找一个最优分类超平面，该超平面不仅能将两类样本正确分开，而且能使两类样本中距离超平面最近的样本点到超平面的距离最大化。

#### 2.1.1 最优分类超平面与最大间隔

在一个二维特征空间中，我们可以将样本点想象成分布在平面上的点。一个线性分类器就是一条直线，它将平面分成两个区域。SVM 的目标不是找到任意一条能分开两类的直线，而是找到“最好”的那一条。

什么是“最好”？SVM 认为最好的直线应该具有最大的“间隔”，即直线两侧与离直线最近的同类点之间的距离最大。这些离直线最近的点，就是支持向量，它们支撑起了这个最优分类边界。

从数学上，一个分类超平面可以表示为：

$$w \cdot x + b = 0$$

其中， $w$  是超平面的法向量，这决定了超平面的方向， $b$  是偏置项，这决定了超平面的位置。

对于一个给定的样本点  $(x_i, y_i)$ ，其中  $y_i \in \{-1, +1\}$  是类别标签，它到超平面的距离为：

$$d_i = \frac{|w \cdot x_i + b|}{\|w\|}$$

SVM 的目标是最大化“间隔”，间隔被定义为两个不同类别的支持向量到超平面距离之和。由于支持向量满足  $|w \cdot x_i + b| = 1$ ，所以间隔为  $2/\|w\|$ 。因此，最大化间隔等价于最小化  $\|w\|^2$ 。

于是，SVM 的原始优化问题可以表述为：

$$\min_{w,b} \frac{\|w\|^2}{2}$$

约束条件为：

$$y_i(w \cdot x_i + b) \geq 1, i = 1, 2, \dots, n$$

## 2.1.2 支持向量的核心作用

在上述优化问题中，只有那些恰好落在间隔边界上（即 $y_i(w \cdot x_i + b) = 1$ ）的样本点对最终的解有贡献。这些点就是支持向量。

它们是定义最优超平面的“关键点”。如果移动任何一个非支持向量的点，只要它不穿过间隔边界，最优超平面就不会改变。但如果移动一个支持向量，超平面就会随之调整。

在后续的对偶问题求解中，我们会发现，只有支持向量对应的拉格朗日乘子 $\alpha_i$ 不为零。最终的权重向量 $w$ 完全由支持向量线性组合而成：

$$w = \sum_{i=1}^n \alpha_i y_i x_i$$

其中 $\alpha_i > 0$ 的点就是支持向量。这意味着模型的复杂度仅取决于支持向量的数量，而与总样本数无关。

## 2.1.3 从二分类到多分类

SVM 本身是一个二分类器。为了处理像本实验中葡萄酒这样的三分类问题，需要采用一定的策略将其扩展到多分类场景。常用策略有“一对一”和“一对多”，本实验中默认采用的是“一对一”策略。其原理如下：

对于 $K$ 个类别，一对一策略会为每一对类别构建一个二分类 SVM。因此总共需要构建 $K(K - 1)/2$ 个分类器。对于本实验的 3 个类别（类别 1,2,3），需要构建 $3 \times (3-1)/2 = 3$  个分类器：

- (1) 分类器 1：类别 1vs 类别 2
- (2) 分类器 2：类别 1vs 类别 3
- (3) 分类器 3：类别 2vs 类别 3

当对一个未知样本进行预测时，该样本会被输入到所有 3 个分类器中进行“投票”，每个分类器都会给出一个预测结果。最终，获得票数最多的那个类别，就是该样本的最终预测类别。

这种方法的优点是每个分类器都只在原始数据的一个子集上进行训练，计算效率相对较高，且分类器的决策边界更清晰。

## 2. 2 核函数方法

在前面的讨论中，我们隐含了一个假设：数据是线性可分的。即存在一个超平面可以将不同类别的样本完美分开。然而，在现实世界中，许多问题都是线性不可分的。

核技巧的提出，正是为了解决这类非线性问题。其核心思想非常巧妙：如果数据在原始的低维空间中线性不可分，我们可以通过一个非线性映射函数 $\phi(x)$ ，将数据映射到一个更高维的特征空间中，在这个高维空间中，数据变得线性可分了。

直接计算高维映射 $\phi(x)$ 通常计算成本极高，甚至维度是无限的。核技巧的精髓在于，我们无需显式地计算出 $\phi(x)$ ，只需要知道高维空间中任意两点内积的结果即可。

这个内积的计算，就是通过核函数 $K(x_i, x_j)$ 来完成的：

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

只要这个核函数满足默塞尔条件，它就对应某个高维空间中的内积。

本实验中使用的径向基函数核（RBF），也称为高斯核，是最常用的一种核函数。其数学形式为：

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

高斯核 SVM 的性能高度依赖于两个关键的超参数： $C$  和  $\gamma$ 。

### (1) $\gamma$ ——核系数

$\gamma$  控制了高斯函数的“陡峭程度”。当  $\gamma$  值较大时，高斯函数的“钟形曲线”非常窄和陡。每个支持向量只影响其周围一个非常小的区域。这会导致决策边界变得非常复杂、蜿蜒曲折，以适应每一个训练数据点。模型对训练数据非常敏感，容易过拟合。 $\gamma$  值较小时，高斯函数的“钟形曲线”非常宽和平缓。每个支持向量都有很大的影响范围。决策边界会变得非常平滑和简单。模型可能无法捕捉数据的复杂结构，导致欠拟合。

### (2) $C$ ——正则化参数

$C$  是原始优化问题中对误分类样本的惩罚系数。当  $C$  值较大时，模型对误分类的惩罚非常严厉，它会尽力将每一个样本都正确分类，即使这意味着创建一个极其复杂的决策边界，这同样会导致过拟合。当  $C$  值较小时，模型对误分类的容忍度较高，它允许一些样本被误分类，以换取一个更简单、更平滑的决策边界，如果  $C$  太小，模型可能会对数据的主要结构视而不见，导致欠拟合。

## 2. 3 模型评估指标

模型训练完成后，需要客观、全面地评估其性能。仅凭单一的“准确率”指标往往是不够的，尤其是在数据类别分布不均衡的情况下。

### 2.3.1 精确率、召回率与 F1 分数

这里先介绍一些基础概念：

TP：真实为正类，预测也为正类的样本数

FP：真实为负类，但预测为正类的样本数

TN：真实为负类，预测也为负类的样本数

FN：真实为正类，但预测为负类的样本数

为了更精细地评估模型性能，我们引入以下三个指标：

(1) 精确率：

$$Precision = \frac{TP}{TP + FP}$$

精确率表示，在所有被模型预测为“正类”的样本中，有多少是真正的“正类”，这能够衡量预测的“精确性”。高精确率意味着模型做出的“正类”预测是可靠的。

(2) 召回率：

$$Recall = \frac{TP}{TP + FN}$$

召回率表示，在所有真实为“正类”的样本中，有多少被模型成功预测出来了，这能够衡量模型的“查全能力”或“敏感性”。高召回率意味着模型擅长找出所有相关的正类样本。

(3) F1 分数：

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

F1 分数是精确率和召回率的调和平均数。它是一个综合性指标，同时考虑了精确率和召回率。当精确率和召回率都高时，F1 分数才会高。它特别适用于类别不均衡的数据集，因为在那些情况下，准确率可能会产生误导。

### 2.3.2 混淆矩阵

混淆矩阵是一个更直观、更详细的性能评估工具。它是一个  $N \times N$  的方阵 ( $N$  为类别数)，其中矩阵元素  $M_{ij}$  表示真实类别为  $i$  但被模型预测为类别  $j$  的样本数量。

对角线元素：代表正确分类的样本数。

非对角线元素：代表将  $i$  错误分类为  $j$  的样本数。

## 2.4 交叉验证与超参数调优

在机器学习模型的构建过程中，为了获得一个兼具良好泛化能力与高性能的模型，并对其性能进行可靠评估，仅依赖简单的训练集与测试集划分是远远不够的。这种单一划分方式的结果易受数据随机性的影响，可能导致评估结果的波动性较大。因此，我们引入了更为稳健和可靠的评估策略——交叉验证，它能够更科学地指导模型选择与超参数调优。

$K$ 折交叉验证（ $K$ -Fold Cross-Validation）是交叉验证中最常用的一种方法。其核心思想在于将训练数据集随机地划分为 $K$ 个大小相近且互斥的子集，通常称为“折”。在接下来的 $K$ 次迭代训练与验证中，每一次迭代都选取其中一个不重复的折作为验证集，而将剩余的 $K - 1$ 折数据合并作为训练集。模型在每次迭代中完成训练后，即在对应的验证集上进行性能评估，并记录下该次的评估指标。当所有 $K$ 次迭代完成后，我们将这 $K$ 次验证得到的性能指标取平均值，以此作为对该模型在此超参数配置下性能的最终、更为稳定的估计。

$K$ 折交叉验证的优势主要体现在三个方面。首先，它提供了更为可靠的性能估计。通过多次计算结果的平均，该方法有效降低了对单次数据划分随机性的依赖，使得模型性能的评估结果更加稳定和可信。其次，它实现了更充分的数据利用。在数据集规模有限的情况下， $K$ 折交叉验证确保了每一个样本都能参与 $K - 1$ 次模型的训练和 1 次验证，极大地提升了数据的使用效率。最后，在超参数调优阶段，交叉验证可以确保我们所选择的超参数组合是在多个不同的数据子集上均表现优异的参数，从而显著降低了模型因“过拟合”于某一特定验证集而选择到次优参数的风险。

然而，手动尝试不同的超参数组合不仅过程繁琐、效率低下，而且难以系统地探索所有可能性，容易遗漏最优解。为了解决这一问题，我们采用了网格搜索交叉验证来自动化这一调优过程。*GridSearchCV*的工作流程始于定义一个参数网格，用户为每个需要优化的超参数指定一个候选值列表。

随后，*GridSearchCV*会系统性地、无一遗漏地遍历这个网格中的每一种参数组合。对于每一种组合，它都会自动执行一次完整的 $K$ 折交叉验证，并计算出该组合下的平均性能分数。在所有参数组合都评估完毕后，*GridSearchCV*通过比较它们的平均交叉验证分数，自动识别并选出分数最高的那个组合作为最优参数。最后，它会使用这组最优参数，在全部的训练数据上重新训练一个最终的模型，这个模型将用于后续的预测任务。

### 3 实验设计与实现

本实验的算法实现与数据分析基于 Python 编程语言，核心的科学计算与机器学习任务主要依赖于 scikit-learn、numpy 及 matplotlib 等业界公认的开源库

```
import h5py
```

```
import numpy as np
import os
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix
import warnings

from utils.path import project_root

# 忽略一些版本更新可能产生的警告
warnings.filterwarnings("ignore")
```

实验加载 Wine 数据集作为研究对象。

```
# 定义数据集路径
# 获取项目根目录路径
PROJECT_ROOT = project_root().as_posix()
WINE_DATA_PATH = os.path.join(PROJECT_ROOT, "datasets",
"wine_data.mat")
WINE_LABEL_PATH = os.path.join(PROJECT_ROOT, "datasets",
"wine_label.mat")

# 从加载的数据中提取特征和标签
# 读取 wine_data.mat
with h5py.File(WINE_DATA_PATH, "r") as f:
    X = np.array(f["wine_data"]).T

# 读取 wine_label.mat
# 使用 .ravel() 将标签数组展平为一维数组，以符合 scikit-learn 的要求
with h5py.File(WINE_LABEL_PATH, "r") as f:
    y = np.array(f["wine_label"]).T.ravel()

print(f"数据加载成功。特征维度: {X.shape}, 标签维度: {y.shape}\n")
```

该数据集共包含 178 个样本，每个样本对应 13 个化学特征，并被划分为 3 个不同的类别。各类别样本数量分布相对均衡，为多分类任务的开展提供了良好的基础。

```
[14]     print(f"数据加载成功。特征维度: {X.shape}, 标签维度: {y.shape}\n")
...
...     数据加载成功。特征维度: (178, 13), 标签维度: (178,)
```

为确保模型评估的客观性与可复现性，数据集首先被划分为训练集与测试集。我们采用 7:3 的比例进行分割，并利用 `stratify` 参数确保训练集与测试集中的类别分布与原始数据集保持一致，避免了因数据划分不均引入的偏差。同时，通过

设置`random_state`参数，固化数据划分的随机种子，保证了实验结果的可重复性。

```
# 将数据集划分为训练集和测试集，测试集占 30%
# random_state 保证每次划分结果一致，便于复现
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.3, random_state=42, stratify=y
)
```

考虑到支持SVM算法对特征的尺度较为敏感，特征缩放是必不可少的预处理步骤。本实验采用标准化方法，具体通过`StandardScaler`实现。该过程对训练集中的每一个特征独立进行，计算其均值 $\mu$ 与标准差 $\sigma$ ，并将原始数据转换为均值为0、方差为1的新特征。另外，对测试集的标准化变换复用了训练集的均值与标准差，以避免数据泄露，确保了评估的有效性。

```
# 特征标准化：SVM 对特征缩放很敏感，标准化可以提升模型性能
# 创建一个标准化处理器
scaler = StandardScaler()
# 在训练集上计算均值和标准差，并对训练集进行转换
X_train_scaled = scaler.fit_transform(X_train)
# 使用相同的处理器（相同的均值和标准差）对测试集进行转换
X_test_scaled = scaler.transform(X_test)

print("数据预处理完成：已划分为训练集和测试集，并进行了标准化。\\n")
```

### 3.2 模型选择、构建与超参数优化

在模型选择阶段，本实验确定采用支持向量机作为核心分类算法。选择SVM的主要依据在于其在处理小样本、高维数据时表现出的优异性能，以及其决策函数对异常值具有较好的鲁棒性，非常适合本实验所用的 Wine 数据集特性。针对核函数的选择，我们采用了径向基函数核。`RBF`核能够将原始数据映射到高维空间，有效处理样本间的非线性关系，是解决复杂分类问题的常用且高效的选择。

```
# 定义要搜索的参数网格
# C 是正则化参数，控制模型的复杂度
# gamma 是 'rbf' 核函数的系数
param_grid = {
    "C": [0.1, 1, 10, 100],
    "gamma": [1, 0.1, 0.01, 0.001],
    "kernel": ["rbf"], # 这里我们专注于 RBF 核，因为它通常性能最好
}
```

为充分挖掘模型潜力，避免因超参数设置不当导致的性能欠佳或过拟合问题，我们采用网格搜索交叉验证进行系统性的超参数调优。调优的目标参数为SVM的关键超参数：正则化参数 $C$ 与`RBF`核参数 $gamma$ 。我们设定的搜索网格为 $C$ 在[0.1, 1, 10, 100]区间， $gamma$ 在[1, 0.1, 0.01, 0.001]区间。在评估每组参数组合的性能

时，结合 5 折交叉验证 ( $cv=5$ ) 策略，以获得对模型泛化能力更为稳健的估计，并最终筛选出性能最优的参数组合。

```
# 使用 GridSearchCV 进行网格搜索和交叉验证
# estimator=SVC() 是我们要优化的模型
# param_grid 是要搜索的参数
# cv=5 表示 5 折交叉验证
# verbose=2 会打印搜索过程
# n_jobs=-1 使用所有可用的 CPU 核心并行计算
grid_search = GridSearchCV(SVC(), param_grid, cv=5, verbose=2, n_jobs=-1)

print("开始进行网格搜索以寻找最优参数...")
# 在标准化的训练数据上进行拟合
grid_search.fit(X_train_scaled, y_train)

# 获取最优模型和最优参数
best_svm = grid_search.best_estimator_
print("\n网格搜索完成!")
print(f"找到的最优参数: {grid_search.best_params_}")
```

## 4 实验结果与分析

通过网格搜索交叉验证，我们确定了  $SVM$  模型的最优超参数组合为  $C = 1$ 、 $gamma = 0.1$ ，核函数为  $RBF$ 。在该参数配置下，模型在 5 折交叉验证中取得了约 0.9920 的最高平均准确率，这表明模型具有出色的稳定性，其性能对训练数据的微小划分不敏感，证明了所选参数组合的鲁棒性。

[16]

```
...      开始进行网格搜索以寻找最优参数 ...
Fitting 5 folds for each of 16 candidates, totalling 80 fits

网格搜索完成!
找到的最优参数: {'C': 1, 'gamma': 0.1, 'kernel': 'rbf'}
交叉验证最高准确率: 0.9920
```

在此基础上，我们在独立的测试集上对最终模型的性能进行了深入评估，其整体准确率达到了 0.98。具体而言，分类报告显示模型对各类别的识别均表现卓越：类别 1 的精确率、召回率和 F1 分数均达到完美的 1.00；类别 2 的召回率为 1.00，表明所有真实样本均被成功找出，但 0.95 的精确率意味着存在一个其他类别的样本被误判为类别 2；类别 3 的精确率为 1.00，但其 0.93 的召回率则揭示了一个真实的类别 3 样本被错误划分。本实验中模型在各项指标上均表现优异，F1 分数综合反映了其稳健性能。测试集的混淆矩阵  $[[18, 0, 0], [0, 21, 0], [0, 1, 14]]$  清晰地展示了这一结果，其中唯一的错误发生在位置(3, 2)，即一个真实类别为 3 的

样本被错误预测为类别 2，推测其原因，该样本很可能位于类别 2 和类别 3 的决策边界附近，其化学成分与类别 2 的典型特征高度相似，导致模型做出了将其归为类别 2 的决策。

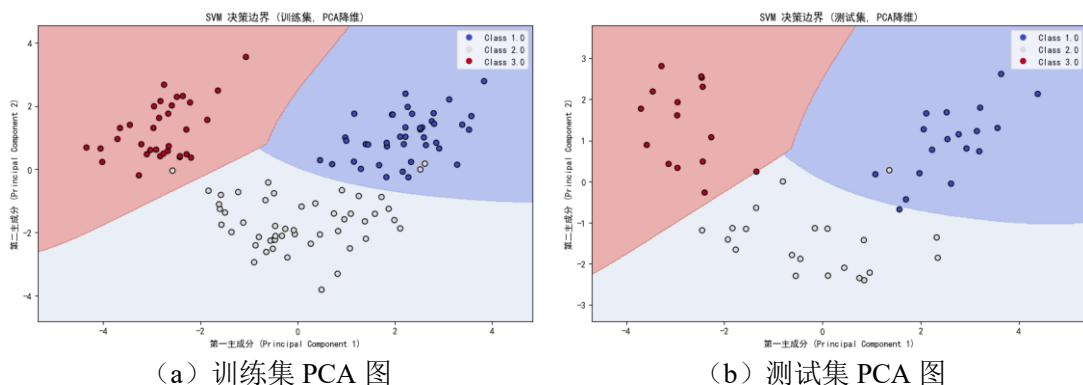
```
[17]
...
... 在测试集上评估最优模型性能 ...

分类报告:
precision    recall   f1-score   support
          1.0      1.00      1.00      1.00      18
          2.0      0.95      1.00      0.98      21
          3.0      1.00      0.93      0.97      15

accuracy           0.98
macro avg       0.98       0.98       0.98      54
weighted avg     0.98       0.98       0.98      54

混淆矩阵:
[[18  0  0]
 [ 0 21  0]
 [ 0  1 14]]
```

进一步地，通过PCA降维可视化分析，我们得以直观地洞察模型的决策机制。在PCA降维后的二维空间中，训练集上的决策边界呈现为平滑的非线性曲线，清晰地划分出三个独立的类别区域，各类别内部样本聚集紧密，区域间分离度良好。更重要的是，测试集上的决策边界与训练集边界在形状和位置上表现出高度的一致性，测试集中的样本点也准确地落在了根据训练集学习到的决策区域内，这种高度一致性强有力地证明了模型优秀的泛化能力，说明模型掌握了数据背后真实的分类模式而非简单记忆。



## 5 结论与展望

本研究成功构建了一个基于支持向量机的高精度葡萄酒分类模型。通过对 Wine 数据集进行系统性的预处理、模型构建、超参数调优与性能评估，我们最终在独立的测试集上取得了高达 98% 的分类准确率。这一卓越成果充分证明了 SVM 算法结合网格搜索交叉验证策略在处理此类多分类问题上的有效性与优越

性。整个研究流程不仅验证了理论方法的可行性，也为解决实际中的分类任务提供了一个可靠的范例。

然而，该模型也存在一些局限性。*SVM*本质上是一个“黑盒”模型，其决策过程缺乏直观的可解释性，这使得在需要追溯分类依据的场景中应用受限，此外，在面对大规模数据集时，本实验所依赖的超参数调优过程，尤其是网格搜索，虽然有效，但涉及较高的计算成本和时间成本，其训练过程相对耗时，计算效率成为瓶颈。

基于当前的研究成果与模型分析，未来的工作可从以下几个层面展开：

(1) 未来的研究可以探索不同的核函数，如多项式核或*Sigmoid*核，并与本实验采用的*RBF*核进行性能比较，以寻求更优的模型配置。此外，可以尝试将集成学习方法，如随机森林或*XGBoost*，应用于同一数据集，以评估其在 Wine 数据集分类任务上的潜力，并与*SVM*模型进行横向对比。

(2) 考虑引入不同的特征选择技术，例如基于递归特征消除或基于模型的特征排序，探究是否能通过筛选出更关键的特征子集来进一步提升模型性能或简化模型复杂度，从而增强模型的可解释性。