day02-机器学习02

上课日期	@June 14, 2025
主讲老师	李晓华
复习	@June 15, 2025
本课程关键词	NumPy｜向量化操作｜广播机制｜特征工程｜K 近邻算法｜标准化
课时	6

🔢 第一部分：NumPy 基本操作

1.1 数组创建

核心概念：NumPy 是 Python 科学计算的基础库，提供高效的多维数组对象和数学函数。

```
import numpy as np

# 从列表创建数组
scores = [random.randint(0, 100) for _ in range(300)]
arr = np.array(scores)

# 创建特殊数组
zeros_array = np.zeros(shape=(2, 3, 4, 5))  # 全零数组
ones_array = np.ones(shape=(2,))            # 全一数组
range_array = np.arange(10)                 # 0-9的数组
linspace_array = np.linspace(-10, 10, 100) # 等差数列
```

重要属性：

- `arr.shape`：数组形状
- `arr.size`：元素总数
- `arr.dtype`：数据类型

1.2 元素级运算（Element-wise Operations）

关键特性：NumPy 支持向量化操作，自动应用到每个元素上。

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# 基本运算
result_add = a + b    # [5, 7, 9]
result_mul = a * b    # [4, 10, 18]
result_pow = a ** b   # [1, 32, 729]

# 与标量运算
scalar_add = a + 1    # [2, 3, 4]
```

与 Python 列表对比：

- 列表：`[1,2,3] + [4,5,6] = [1,2,3,4,5,6]` (连接)
- NumPy：`array([1,2,3]) + array([4,5,6]) = array([5,7,9])` (元素级相加)

1.3 广播机制（Broadcasting）

定义：在不引起歧义的前提下，NumPy 自动调整数组形状以进行运算。

```
arr1 = np.array([1, 2, 3, 4, 5])  # shape: (5,)
arr2 = np.array([3])              # shape: (1,)
result = arr1 + arr2              # 广播成功: [4, 5, 6, 7, 8]

# 二维广播示例
arr2d = np.array([[1], [2]])      # shape: (2, 1)
result2d = arr1 + arr2d           # 结果 shape: (2, 5)
```

广播规则：

1. 从最后一个维度开始比较
2. 维度大小相等或其中一个为 1 时可以广播
3. 缺失维度视为 1

1.4 数学函数

```
arr = np.array([1, 2, 3, 4, 5])

# 三角函数
sin_values = np.sin(arr)
cos_values = np.cos(arr)

# 指数和对数
exp_values = np.exp(arr)
log_values = np.log(arr)
log2_values = np.log2(arr)
log10_values = np.log10(arr)
```

📐 第二部分：NumPy 向量化操作

2.1 向量空间基础

核心概念：从向量空间角度理解数据，涉及模长、夹角和内积。

2.2 向量模长计算

数学公式：对于向量$\mathbf{v} = [v_1, v_2, ..., v_n]$，模长为：

$$|\mathbf{v}| = \sqrt{v_1^2 + v_2^2 + ... + v_n^2}$$

```
v1 = np.array([1, 2, 3, 4, 5])

# 方法1：手动计算
norm_manual = (v1 ** 2).sum() ** 0.5

# 方法2：使用 NumPy 函数
norm_numpy = np.linalg.norm(v1)
```

2.3 向量内积与余弦相似度

内积公式：$\mathbf{v_1} \cdot \mathbf{v_2} = \sum_{i=1}^{n} v_{1i} \times v_{2i}$

余弦相似度公式：

$$\cos(\theta) = \frac{\mathbf{v_1} \cdot \mathbf{v_2}}{|\mathbf{v_1}| \times |\mathbf{v_2}|}$$

```
v1 = np.array([1, 2, 3, 4, 5])
v2 = np.array([6, 3, 1, 3, 6])

# 内积计算
dot_product_manual = (v1 * v2).sum()
dot_product_operator = v1 @ v2

# 余弦相似度
cosine_similarity = v1 @ v2 / (np.linalg.norm(v1) * np.linalg.norm(v2))
```

2.4 欧几里得距离

公式：$d(\mathbf{v_1}, \mathbf{v_2}) = \sqrt{\sum_{i=1}^{n} (v_{1i} - v_{2i})^2}$

```
# 欧几里得距离
euclidean_distance = ((v1 - v2) ** 2).sum() ** 0.5
# 或使用 NumPy
euclidean_distance = np.linalg.norm(v1 - v2)
```

2.5 矩阵运算

```
m1 = np.arange(12).reshape(3, 4)  # 3×4 矩阵
m2 = np.arange(12).reshape(4, 3)  # 4×3 矩阵

# 矩阵乘法（注意维度匹配）
result = m1 @ m2  # 结果为 3×3 矩阵
```

🔧 第三部分：特征工程

3.1 数据集介绍

使用 **乳腺癌威斯康辛数据集**：

- **样本数量**：569 个
- **特征数量**：30 个数值特征
- **目标变量**：二分类（恶性/良性）

```
from sklearn.datasets import load_breast_cancer

X, y = load_breast_cancer(return_X_y=True)
print(f"特征矩阵形状: {X.shape}")  # (569, 30)
print(f"目标向量形状: {y.shape}")  # (569,)
```

3.2 特征标准化

问题：不同特征的量纲和数值范围差异巨大，会影响算法性能。

解决方案：Z-score 标准化

数学公式：

$$x_{\text{标准化}} = \frac{x - \mu}{\sigma}$$

其中：

- μ 是特征均值
- σ 是特征标准差

```
# 计算统计量
mu = X.mean(axis=0)      # 各特征均值
sigma = X.std(axis=0)    # 各特征标准差

# 标准化
X_standardized = (X - mu) / sigma
```

标准化效果：

- 均值变为 0
- 标准差变为 1
- 消除量纲影响

3.3 其他规范化方法

Min-Max 规范化：

$$x_{\text{规范化}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

```
# Min-Max 规范化到 [0,1] 区间
X_minmax = (X - X.min(axis=0)) / (X.max(axis=0) - X.min(axis=0))
```

🎯 第四部分：K 近邻分类算法

4.1 算法原理

KNN 核心思想：根据样本在特征空间中的 K 个最近邻居的标签来预测其类别。

决策规则：对于分类问题，采用多数投票原则。

4.2 完整机器学习流程

```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

# 1. 加载数据
X, y = load_breast_cancer(return_X_y=True)

# 2. 数据切分
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, shuffle=True, random_state=0
)

# 3. 特征标准化（重要！）
mu = X_train.mean(axis=0)
sigma = X_train.std(axis=0)
X_train_scaled = (X_train - mu) / sigma
X_test_scaled = (X_test - mu) / sigma

# 4. 模型训练
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train_scaled, y_train)

# 5. 预测和评估
y_pred = knn.predict(X_test_scaled)
accuracy = (y_pred == y_test).mean()
print(f"准确率: {accuracy:.4f}")
```

4.3 关键注意事项

为什么需要标准化？

- KNN 基于距离计算
- 不同特征的数值范围差异会导致某些特征主导距离计算
- 标准化确保所有特征等权重参与计算

数据泄露防范：

- ✅ 正确做法：用训练集统计量标准化测试集
- ❌ 错误做法：用全体数据统计量标准化

📈 第五部分：K 近邻回归算法

5.1 数据集介绍

使用 **波士顿房价数据集**：

- **样本数量**：506 个
- **特征数量**：13 个房屋相关特征
- **目标变量**：房价（连续值）

```
import pandas as pd

# 加载数据
data = pd.read_csv("boston_house_prices.csv", skiprows=1)
X = data.loc[:, :"LSTAT"].to_numpy()  # 特征
y = data.loc[:, "MEDV"].to_numpy()    # 目标变量（房价）
```

5.2 回归算法流程

```
from sklearn.neighbors import KNeighborsRegressor

# 数据切分和标准化（与分类相同）
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0
)

mu = X_train.mean(axis=0)
sigma = X_train.std(axis=0)
X_train_scaled = (X_train - mu) / sigma
X_test_scaled = (X_test - mu) / sigma

# 回归模型
knn_reg = KNeighborsRegressor(n_neighbors=5)
knn_reg.fit(X_train_scaled, y_train)
y_pred = knn_reg.predict(X_test_scaled)
```

5.3 回归评估指标

平均绝对误差 (MAE)：

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

均方误差 (MSE)：

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

```
# 评估指标计算
mae = abs(y_pred - y_test).mean()
mse = ((y_pred - y_test) ** 2).mean()
rmse = mse ** 0.5

print(f"平均绝对误差: {mae:.4f}")
print(f"均方误差: {mse:.4f}")
print(f"均方根误差: {rmse:.4f}")
```

🔑 核心要点总结

NumPy 核心概念

- **向量化操作**：提高计算效率的关键
- **广播机制**：实现不同形状数组的运算
- **线性代数函数**：模长、内积、距离计算

特征工程要点

- **标准化必要性**：消除特征间的量纲差异
- **数据泄露防范**：始终用训练集统计量处理测试集
- **Z-score vs Min-Max**：根据数据分布选择合适方法

KNN 算法核心

- **距离度量**：欧几里得距离最常用

- **K 值选择**：影响模型复杂度和性能
- **分类 vs 回归**：投票机制 vs 平均值机制

机器学习标准流程

1. **数据加载** → 2. **数据切分** → 3. **特征预处理** → 4. **模型训练** → 5. **预测评估**

🚀 实践建议

1. **代码实践**：每个概念都要亲自编写代码验证
2. **参数调优**：尝试不同的 K 值，观察性能变化
3. **数据可视化**：用图表理解数据分布和算法效果
4. **扩展学习**：了解其他距离度量方法和预处理技术
