day03-机器学习03

上课日期	@June 18, 2025
主讲老师	李晓华
复习	@June 19, 2025
本课程关键词	机器学习｜数据预处理｜线性回归｜逻辑回归｜模型评估｜KNN算法
# 课时	2

📚 目录

🎯 机器学习基础概念

有监督学习流程

机器学习的标准流程包括以下几个关键步骤：

1. **数据收集与加载**
2. **特征工程与数据预处理**
3. **模型选择与训练**
4. **模型评估与优化**
5. **模型部署与保存**

回归 vs 分类

- **回归问题**：预测连续数值（如房价预测）
- **分类问题**：预测离散类别（如疾病诊断）

🔧 数据预处理与标准化

数据标准化的重要性

数据标准化是机器学习中的关键步骤，目的是消除不同特征之间量纲差异的影响。

标准化公式

$$X_{normalized} = \frac{X - \mu}{\sigma + \epsilon}$$

其中：

- μ 是均值
- σ 是标准差
- ε = 1e-9 是防止除零的小常数

代码实现

```
# 计算训练集的均值和标准差
mu = X_train.mean(axis=0)
sigma = X_train.std(axis=0) + 1e-9

# 对训练集和测试集进行标准化
X_train = (X_train - mu) / sigma
X_test = (X_test - mu) / sigma
```

> ⚠️ 注意：测试集必须使用训练集的均值和标准差进行标准化，避免数据泄露

📈 线性回归

理论基础

线性回归假设目标变量与特征变量之间存在线性关系：

$$y = w_0 + w_1x_1 + w_2x_2 + ... + w_nx_n + \epsilon$$

其中：

- y 是目标变量

- w_i 是权重参数
- x_i 是特征变量
- ε 是误差项

损失函数

线性回归使用均方误差（MSE）作为损失函数：

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

完整代码示例

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# 1. 数据加载
data = pd.read_csv("boston_house_prices.csv", skiprows=1)

# 2. 特征和标签分离
X = data.loc[:, :"LSTAT"].to_numpy()  # 特征
y = data.loc[:, "MEDV"].to_numpy()    # 标签

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

# 4. 数据标准化
mu = X_train.mean(axis=0)
sigma = X_train.std(axis=0) + 1e-9
X_train = (X_train - mu) / sigma
X_test = (X_test - mu) / sigma

# 5. 模型训练
lr = LinearRegression()
lr.fit(X=X_train, y=y_train)

# 6. 预测与评估
y_pred = lr.predict(X=X_test)
mse = ((y_pred - y_test) ** 2).mean()
print(f"均方误差: {mse}")
```

🧠 逻辑回归与分类

Sigmoid 激活函数

逻辑回归的核心是 Sigmoid 函数，它将线性输出映射到 (0,1) 区间，表示概率：

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Sigmoid 函数特性

- **输出范围**：(0, 1)
- **S型曲线**：平滑的概率转换
- **导数易计算**：$\sigma'(z) = \sigma(z)(1 - \sigma(z))$

代码实现

```
import numpy as np
import matplotlib.pyplot as plt

def sigmoid(x):
    """Sigmoid 激活函数"""
    return 1 / (1 + np.exp(-x))

# 可视化 Sigmoid 函数
x = np.linspace(-10, 10, 100)
plt.plot(x, sigmoid(x))
plt.grid()
plt.title('Sigmoid Function')
plt.xlabel('Input')
plt.ylabel('Output')
plt.show()
```

逻辑回归完整流程

```
from sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

# 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, random_state=0
)

# 3. 数据标准化
mu = X_train.mean(axis=0)
sigma = X_train.std(axis=0) + 1e-9
X_train = (X_train - mu) / sigma
X_test = (X_test - mu) / sigma

# 4. 模型训练
lr = LogisticRegression()
lr.fit(X=X_train, y=y_train)

# 5. 预测与评估
y_pred = lr.predict(X=X_test)
accuracy = (y_pred == y_test).mean()
print(f"准确率: {accuracy:.4f}")

# 查看模型参数
print(f"权重数量: {lr.coef_.shape}")
print(f"偏置项: {lr.intercept_}")
```

📊 模型评估

回归问题评估指标

均方误差 (MSE)

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

```
mse = ((y_pred - y_test) ** 2).mean()
```

分类问题评估指标

准确率 (Accuracy)

$$Accuracy = \frac{正确预测数量}{总预测数量}$$

```
accuracy = (y_pred == y_test).mean()
```

💾 模型保存与加载

序列化概念

- **序列化**：将内存中的对象转化为字节流，保存到硬盘
- **反序列化**：将硬盘上的文件读入，转化为内存中的对象

Python 序列化工具对比

工具	特点	适用场景
`pickle`	底层，操作相对复杂	通用Python对象序列化
`joblib`	上层，操作简便	专为大型NumPy数组和科学计算优化

代码示例

使用 pickle 保存单个模型：

```
import pickle

# 保存模型
with open("lr.pickle", "wb") as f:
    pickle.dump(obj=lr, file=f)

# 加载模型
with open("lr.pickle", "rb") as f:
    lr_loaded = pickle.load(file=f)
```

使用 joblib 保存多个模型：

```
import joblib

# 保存多个模型
joblib.dump(value=knn, filename="knn.joblib")
joblib.dump(value=[lr, knn], filename="models.joblib")

# 加载模型
knn_loaded = joblib.load(filename="knn.joblib")
lr_loaded, knn_loaded = joblib.load(filename="models.joblib")
```

📐 统计学概念：方差

方差的两种估计方法

统计学中，方差有两种不同的估计方法：

1. 总体方差（有偏估计）

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

2. 样本方差（无偏估计）

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

库函数差异

库	默认方法	参数控制
NumPy	有偏估计 (除以N)	`ddof=1` 使用无偏估计
PyTorch	无偏估计 (除以N-1)	`correction=1` 控制

代码示例

```
import numpy as np
import torch

ls = [1, 2, 3, 4, 5, 6]

# NumPy - 默认有偏估计
arr = np.array(ls)
biased_std = arr.std()        # 有偏
unbiased_std = arr.std(ddof=1) # 无偏

# PyTorch - 默认无偏估计
t = torch.tensor(ls, dtype=torch.float32)
unbiased_std = t.std(correction=1)  # 无偏
```

⚠️ 注意事项

1. **防止除零错误**：方差计算时加入小常数

   ```
   sigma = X_train.std(axis=0) + 1e-9
   ```

2. **选择合适的估计方法**：
   - 样本量较小时，使用无偏估计更准确
   - 深度学习中，通常使用有偏估计（计算效率更高）

🚀 KNN算法补充

K近邻算法特点

- **非参数方法**：不需要假设数据分布
- **懒惰学习**：训练阶段只存储数据，预测时才计算
- **适用于**：回归和分类问题

代码示例

```
from sklearn.neighbors import KNeighborsRegressor, KNeighborsClassifier
```

```
# 回归任务
knn_reg = KNeighborsRegressor(n_neighbors=5)
knn_reg.fit(X=X_train, y=y_train)

# 分类任务
knn_clf = KNeighborsClassifier(n_neighbors=5)
knn_clf.fit(X=X_train, y=y_train)
```

🎯 关键要点总结

✅ 最佳实践

1. **数据预处理必不可少**：标准化能显著提升模型性能
2. **合理划分数据集**：训练集用于学习，测试集用于评估
3. **选择合适的评估指标**：MSE用于回归，准确率用于分类
4. **模型保存很重要**：使用joblib保存训练好的模型

🔍 深入理解

1. **线性回归**：适用于线性关系明显的回归问题
2. **逻辑回归**：通过Sigmoid函数实现分类，本质上是线性分类器
3. **方差估计**：理解有偏与无偏的区别，根据场景选择

📈 进阶方向

- 正则化方法（Ridge、Lasso）
- 模型选择与交叉验证
- 特征工程技巧
- 集成学习方法
