数据预处理

—— 归一化与标准化实践工作册 ——

参 考 答 案 与 解 析

！ 本文档仅供教师参考，请勿向学生分发

第一部分：理论练习答案

练习 1.1 填空题

参考答案：

1. 归一化将数据映射到 [0, 1] 区间，而标准化将数据转换为均值为 0 、标准差为 1 的分布。
2. 归一化对 异常值（极端值） 敏感，而标准化对异常值相对 不敏感（鲁棒） 。
3. 在 scikit-learn 中，归一化使用 MinMaxScaler 类，标准化使用 StandardScaler 类。

练习 1.2 手动计算

给定数据集：X = [10, 20, 30, 40, 50]

(a) 归一化计算：

解题过程：

X_min = 10, X_max = 50

X_norm = (X - 10) / (50 - 10) = (X - 10) / 40

X=10: (10-10)/40 = 0.000

X=20: (20-10)/40 = 0.250

X=30: (30-10)/40 = 0.500

X=40: (40-10)/40 = 0.750

X=50: (50-10)/40 = 1.000

归一化结果：[0.0, 0.25, 0.5, 0.75, 1.0]

(b) 标准化计算：

☑ **解题过程：**

μ = 30，σ ≈ 14.14

X_std = (X - 30) / 14.14

X=10: (10-30)/14.14 ≈ -1.414

X=20: (20-30)/14.14 ≈ -0.707

X=30: (30-30)/14.14 = 0.000

X=40: (40-30)/14.14 ≈ 0.707

X=50: (50-30)/14.14 ≈ 1.414

标准化结果：[-1.414, -0.707, 0.0, 0.707, 1.414]

练习 1.3 场景分析

(a) KNN客户分类（年龄、收入、消费次数）：

☑ **参考答案：**

推荐使用标准化（StandardScaler）。

理由：KNN 基于距离计算，收入的值域（0-1000000）远大于年龄（0-100）和消费次数，如果不缩放，收入会主导距离计算。标准化对异常值更鲁棒，适合收入可能存在极端值的情况。归一化也可以，但如果数据有异常值则效果不佳。

(b) 图片像素值输入CNN：

☑ **参考答案：**

推荐使用归一化（Min-Max Normalization）。

理由：图片像素值范围固定为 0-255，分布均匀且无异常值。归一化后映射到 [0,1]，正好匹配神经网络激活函数（如 Sigmoid）的输入范围，有助于加速收敛。实际操作中通常直接除以 255 即可。

(c) PCA降维（数据含异常值）：

参考答案：

推荐使用标准化（StandardScaler）。

理由：PCA 基于协方差矩阵，需要数据中心化（均值为0）。标准化正好满足这一要求。同时，数据中存在异常值，标准化比归一化更鲁棒，不会因为极端值而过度压缩正常数据的范围。

第二部分：基础实践答案

练习 2.1 环境准备与数据加载

各特征统计信息表：

特征名	最小值	最大值	均值	标准差
sepal length (cm)	4.3	7.9	5.84	0.83
sepal width (cm)	2.0	4.4	3.06	0.44
petal length (cm)	1.0	6.9	3.76	1.77
petal width (cm)	0.1	2.5	1.20	0.76

☑ 各特征值域分析：

不同特征的值域明显不一致。例如 sepal length 范围为 4.3-7.9，而 petal width 范围为 0.1-2.5。

这会导致基于距离的算法（如 KNN、SVM）中，值域较大的特征会主导距离计算，导致模型偏向该特征，因此需要进行特征缩放。

练习 2.2 MinMaxScaler 归一化

☑ 完整代码：

```
from sklearn.preprocessing import MinMaxScaler
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# 创建 MinMaxScaler 对象
scaler_mm = MinMaxScaler()
```

```
# 选择数值特征列
features = df[iris.feature_names]

# 拟合并转换数据
features_normalized = scaler_mm.fit_transform(features)

# 转换为DataFrame
df_norm = pd.DataFrame(features_normalized,
                       columns=iris.feature_names)
print(df_norm.describe())
```

☑ 验证结果：

归一化后所有特征的 min = 0.0，max = 1.0。

运行 df_norm.min() 均为 0.0，df_norm.max() 均为 1.0，符合预期。

练习 2.3 StandardScaler 标准化

☑ 完整代码：

```
from sklearn.preprocessing import StandardScaler

# 创建 StandardScaler 对象
scaler_std = StandardScaler()

# 拟合并转换数据
features_standardized = scaler_std.fit_transform(features)

# 转换为DataFrame
df_std = pd.DataFrame(features_standardized,
                      columns=iris.feature_names)
print(df_std.describe())
```

☑ 验证结果：

标准化后所有特征的 mean ≈ 0（实际值接近 1e-15，浮点误差），std ≈ 1.0。

运行 df_std.mean() 约为 0，df_std.std() 约为 1.0，符合预期。

注意：pandas 默认使用 ddof=1（样本标准差），所以 std 可能显示为约 1.003，这是正常的。

第三部分：可视化对比答案

练习 3.1 分布直方图观察

☑ 观察结果 1 — 值域差异：

原始数据中，各特征值域差异显著：sepal length 集中在 4-8，petal length 分布在 1-7，petal width 在 0-2.5。直方图在 x 轴上的位置和范围差别很大。

☑ 观察结果 2 — 分布形状：

归一化和标准化都不会改变数据分布的形状，只改变了尺度和位置。

原因：归一化是线性变换 (ax+b)，标准化也是线性变换。线性变换不改变分布形状，只做平移和缩放。

归一化后所有值压缩到 [0,1]；标准化后数据以 0 为中心对称分布。

练习 3.2 箱线图对比

☑ 参考代码：

```
import matplotlib.pyplot as plt

fig, axes = plt.subplots(1, 3, figsize=(15, 5))

# 原始数据
df[iris.feature_names].boxplot(ax=axes[0])
axes[0].set_title('原始数据', fontsize=14)
axes[0].tick_params(axis='x', rotation=45)

# 归一化
df_norm.boxplot(ax=axes[1])
axes[1].set_title('Min-Max 归一化', fontsize=14)
axes[1].tick_params(axis='x', rotation=45)
```

```
# 标准化
df_std.boxplot(ax=axes[2])
axes[2].set_title('Z-Score 标准化', fontsize=14)
axes[2].tick_params(axis='x', rotation=45)


plt.tight_layout()
plt.savefig('boxplot_comparison.png', dpi=150)
plt.show()
```

☑ 异常值分析：

箱线图中的异常点（圆点）在三种情况下位置不同但数量相同。

归一化和标准化都是线性变换，不会消除异常值，只会改变它们的数值位置。

归一化后异常值被压缩到 [0,1] 内，可能显得不明显；标准化后异常值可能数值较大（绝对值 >2 或 >3），更容易识别。

第四部分：进阶实践答案

练习 4.1 fit 与 transform 的区别

☑ 完整代码：

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X_train, X_test = train_test_split(
    features, test_size=0.3, random_state=42)

scaler = StandardScaler()

# 对训练集 fit_transform
X_train_scaled = scaler.fit_transform(X_train)

# 对测试集只 transform
X_test_scaled = scaler.transform(X_test)

print('训练集均值:', X_train_scaled.mean(axis=0))
print('测试集均值:', X_test_scaled.mean(axis=0))
```

思考题答案：

☑ 问题 1：为什么测试集只能用 transform()?

fit() 会计算数据的统计参数（均值、标准差等）。测试集必须使用训练集的统计参数进行变换，这样才能保证训练集和测试集在同一尺度上。如果测试集重新 fit，就会用测试集自己的统计参数，破坏了一致性。

☑ 问题 2：测试集也用 fit_transform() 会怎样?

会导致数据泄露（Data Leakage）。具体表现：

1) 测试集和训练集使用不同的缩放参数，导致两者不在同一尺度上；

2) 模型在训练时“看到”了测试数据的信息，验证结果会虚高；

3) 模型部署到真实环境后，新数据的统计参数未知，不可能重新 fit。

练习 4.2 异常值对缩放的影响

☑ **输出结果参考：**

无异常值 - 归一化: [0.0, 0.111, 0.222, 0.333, 0.444, 0.556, 0.667, 0.778, 0.889, 1.0]

有异常值 - 归一化: [0.0, 0.001, 0.002, 0.003, 0.004, 0.005, 0.006, 0.007, 0.008, 1.0]

无异常值 - 标准化: [-1.566, -1.218, -0.870, -0.522, -0.174, 0.174, 0.522, 0.870, 1.218, 1.566]

有异常值 - 标准化: [-0.342, -0.339, -0.336, -0.332, -0.329, -0.326, -0.323, -0.319, -0.316, 3.063]

☑ **分析 1：归一化的影响**

归一化后，异常值 1000 占据了几乎整个 [0,1] 范围，而正常数据点（1-9）被压缩到 0.000-0.008 的极小范围内，几乎无法区分。这是归一化的重大缺陷。

☑ **分析 2：标准化的表现**

标准化后，正常数据点仍然分布在 -0.34 到 -0.32 之间，虽然范围也受影响，但相比归一化保留了更多的区分度。异常值被映射到约 3.06，清晰可识别。

☑ **建议：**

如果数据中存在异常值，推荐使用标准化（StandardScaler）。更鲁棒的替代方案是使用 RobustScaler，它基于中位数和四分位距（IQR）进行缩放，对异常值更不敏感。

练习 4.3 自定义缩放范围

(a) 缩放到 [-1, 1]：

☑ **完整代码：**

```
from sklearn.preprocessing import MinMaxScaler

scaler_custom = MinMaxScaler(feature_range=(-1, 1))
features_custom = scaler_custom.fit_transform(features)

df_custom = pd.DataFrame(features_custom,
                         columns=iris.feature_names)
print(df_custom.describe())
# 验证：min 应为 -1.0，max 应为 1.0
```

☑ (b) 思考题答案：

使用 [-1, 1] 范围的典型场景：

1) 某些神经网络使用 tanh 激活函数，输出范围为 (-1, 1)，输入匹配该范围效果更好；

2) 图像处理中某些模型（如 GAN）输入要求 [-1, 1]；

3) 当特征本身具有正负含义时（如温度变化、股价涨跌率），[-1,1] 能保留正负方向信息。

第五部分：综合挑战答案

练习 5.1 完整的数据预处理流程

完整参考代码：

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, StandardScaler
import matplotlib.pyplot as plt


# Step 1: 创建 DataFrame
data = {
    '数学': [85, 92, 76, 68, 95],
    '英语': [72, 88, 95, 81, 69],
    '物理': [90, 78, 85, 92, 71],
    '月生活费': [1500, 2200, 1800, 3500, 1200]
}
df = pd.DataFrame(data,
    index=['张三', '李四', '王五', '赵六', '孙七'])
print('原始数据:')
print(df)
print('\n统计信息:')
print(df.describe())


# Step 2: 归一化和标准化
mm_scaler = MinMaxScaler()
std_scaler = StandardScaler()


df_norm = pd.DataFrame(
    mm_scaler.fit_transform(df),
    columns=df.columns, index=df.index)


df_std = pd.DataFrame(
    std_scaler.fit_transform(df),
    columns=df.columns, index=df.index)


print('\n归一化结果:')
print(df_norm.round(3))
print('\n标准化结果:')
print(df_std.round(3))
```

```
# Step 3: 绘制对比图
fig, axes = plt.subplots(1, 3, figsize=(15, 5))

df.plot(kind='bar', ax=axes[0], title='原始数据')
df_norm.plot(kind='bar', ax=axes[1], title='Min-Max 归一化')
df_std.plot(kind='bar', ax=axes[2], title='Z-Score 标准化')

for ax in axes:
    ax.tick_params(axis='x', rotation=0)
    ax.legend(fontsize=8)

plt.tight_layout()
plt.savefig('student_comparison.png', dpi=150)
plt.show()
```

☑ Step 4 分析：

原始数据中，“月生活费”的值域（1200-3500）远大于成绩特征（68-95），如果直接用于距离计算，月生活费会完全主导结果。

归一化后，所有特征都在 [0,1] 内，可以公平比较；标准化后，所有特征以 0 为中心，特征间的相对差异保留得更好。

结论：当特征量纲差异巨大时，特征缩放是必不可少的预处理步骤。

练习 5.2 实际数据集处理（Wine）

完整参考代码：

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_wine
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt


# 1. 加载数据并查看统计信息
wine = load_wine()
df_wine = pd.DataFrame(wine.data, columns=wine.feature_names)
print(df_wine.describe())


# 2. 划分训练集和测试集
X_train, X_test = train_test_split(
    df_wine, test_size=0.2, random_state=42)


# 3. 归一化
mm_scaler = MinMaxScaler()
X_train_mm = mm_scaler.fit_transform(X_train)
X_test_mm = mm_scaler.transform(X_test)  # 注意：只 transform


# 标准化
std_scaler = StandardScaler()
X_train_std = std_scaler.fit_transform(X_train)
X_test_std = std_scaler.transform(X_test)  # 注意：只 transform


# 4. 绘制对比图
fig, axes = plt.subplots(1, 3, figsize=(18, 6))


X_train.boxplot(ax=axes[0], rot=90)
axes[0].set_title('原始数据', fontsize=14)


pd.DataFrame(X_train_mm, columns=wine.feature_names
    ).boxplot(ax=axes[1], rot=90)
axes[1].set_title('Min-Max 归一化', fontsize=14)


pd.DataFrame(X_train_std, columns=wine.feature_names
```

```
    ).boxplot(ax=axes[2], rot=90)
axes[2].set_title('Z-Score 标准化', fontsize=14)

plt.tight_layout()
plt.savefig('wine_comparison.png', dpi=150)
plt.show()
```

总结报告参考：

1) Wine 数据集各特征量纲差异极大，例如 proline 范围为 278-1680，而 magnesium 为 70-162，color_intensity 为 1.3-13.0。

2) 归一化后所有特征压缩到 [0,1]，但 proline 的异常值导致其他数据点集中在窄小范围内。

3) 标准化后各特征均值为 0、标准差为 1，保留了更多的数据分布信息。

4) 结论：对于 Wine 这样特征量纲差异大且含异常值的数据集，标准化通常是更好的选择。归一化适合数据均匀分布且无异常值的场景。