安徽大学《机器学习》实验报告 6

学号：WA2214014　　专业：人工智能　　姓名：杨跃浙

实验日期：24.12.30　　教师签字：　　成绩：

[实验名称] 神经网络分类任务综合实验

[实验目的]

1．熟悉和掌握机器学习的完整流程

2．熟悉和掌握神经网络的构建

[实验要求]

1．采用 Python、Matlab 等高级语言进行编程，推荐优先选用 Python 语言

2．代码可读性强：变量、函数、类等命名可读性强，包含必要的注释

3．提交实验报告要求：1）报告文件：命名为“学号—姓名—Lab6”； 2）提交时间截止：一周后

[实验原理]

神经网络原理概述

本实验所用的 BP 神经网络（Back—Propagation Neural Network）属于前馈式神经网络（Feedforward Neural Network），其核心思想是利用多层感知机（MLP，Multi—Layer Perceptron）对高维输入数据进行非线性映射与特征提取。神经网络一般由若干层的线性变换（全连接层）和非线性激活函数构成；在前向传播（Forward Propagation）过程中，数据从输入层逐层传递到输出层，每一层都会执行线性变换并通过 ReLU、Sigmoid 等激活函数将输出映射到相应区间，从而

提高网络对复杂边界的拟合能力。理论上，多层神经网络在隐层节点数足够时，可以逼近任何连续可测函数，这正是其在模式识别和分类任务中广泛应用的根源。

模型结构与训练过程

本实验设计了包含三层隐藏层的 BP 神经网络，输入层维度与样本的特征数相同，隐藏层维度分别设为 128、32、2，均使用 ReLU 函数 $\mathrm{ReLU}(x) = \max(0, x)$ 作为非线性激活；输出层则采用 1 维输出加 Sigmoid 函数，将网络输出映射到 $(0,1)$ 区间，表示二分类中"预测为正样本"的概率。

在前向传播中，设输入向量为 $x \in \mathbb{R}^d$，网络第一层的线性变换记作

$$z^{(1)} = W^{(1)}x + b^{(1)},\ h^{(1)} = \sigma(z^{(1)})$$

其中 $z^{(1)}$ 是线性变换后的结果，$h^{(1)}$ 是激活函数的输出。当数据依次经过后续隐藏层的线性变换与激活后，倒数第二层会输出一个 2 维的向量；最终输出层通过 Sigmoid 函数得到标量预测 $\hat{y}$ 。在测试阶段，可截取 2 维隐藏层输出（即网络的倒数第二层）来可视化测试样本的二维分布，以便直观分析模型学到的特征表示。

损失函数与反向传播优化

神经网络通过最小化损失函数来学习合适的权重与偏置，本实验采用二分类常用的交叉嫡损失（Cross Entropy Loss）。若单个样本的真实标签为 $y \in \{0,1\}$，网络输出的预测概率为 $\hat{y} \in (0,1)$，则其交叉嫡损失为

$$L(\hat{y}, y) = -[y\ln(\hat{y}) + (1-y)\ln(1-\hat{y})].$$

将所有样本的损失求平均，即得到训练集整体的损失值。为了不断减小此损失，需要用反向传播（Backpropagation）算法在每个参数上计算偏导数；依托链式法则，从输出层开始逐层向前回传梯度，再由优化器（如 SGD 或 Adam）沿负梯度方向更新各层权重 $W^{(l)}$ 和偏置 $b^{(l)}$ 。网络重复"前向传播 → 计算损失 → 反

向传播 → 参数更新" 的循环，使得预测值逐渐逼近真实标签，最终在训练集上收玫到较小误差。

五折交叉验证与 AUROC 评估

为了克服随机数据划分可能导致的偏差，并在有限样本量条件下尽量利用全部数据，本实验采用五折交叉验证（5—Fold Cross Validation）来评估神经网络。将 200 个正样本和 200 个负样本分别划分成 5 份（每份 40 个）；在第 i 折时，将"第 i 份正样本 + 第 i 份负样本"作为测试集，其余 4 份样本作为训练集。这样做可保证测试集中正负各 40 个，正负比为 1:1。每折训练结束后，针对该折测试集计算 AUROC（Area Under the ROC Curve），并在 5 折完成后对 AUROC 取平均值作为整体指标。AUROC 借助 ROC 曲线下的面积来衡量模型区分正负样本的能力，若 AUROC 越接近 1 则说明模型判别效果越好，若接近 0.5 则与随机猜测无异。

特征选择与特征提取可视化

实验还包含两个额外的可视化步骤：特征选择与特征提取。特征选择中，先计算每个特征的方差并挑选出方差最大的两列作为主坐标，将所有样本投影到这两个特征上绘制散点图，并以颜色区分正负样本；这是基于单维度方差大小的简单筛选，可直观检验哪些特征更能区分样本。特征提取采用 PCA（Principal Component Analysis）将原始数据降到二维，寻找令数据方差最大的正交方向作为主成分，再绘制相应的二维散点图，以比较与方差筛选方式下的区别。若样本在 PCA 二维投影后能形成相对分离的簇，说明数据具有一定可分性；若仍混杂，则可能表示该数据难以通过线性映射直接分割。通过与网络倒数第二层的二维输出对比，

可以进一步理解神经网络所学习的特征空间与原始降维结果之间的差异，从而在定量和定性两个层面评估模型的分类效果与数据本身的可分性。

[实验内容]

(一) 数据简介

采用数据集 “data/positive.csv”和“data/negative.csv”进行本次实验。

(二) 原始数据可视化

1. 调用可视化工具，将原始数据可视化输出至二维平面内，以颜色区分不同类别。（python 可使用 UMAP，*conda install umap-learn*；matlab 可使用 t-sne 函数）。

(三)模型设计

1. 本实验为二分类问题，正负样本各 200 个。
2. 使用 BP 神经网络进行二分类（Python 可使用 Pytorch，MATLAB 可使用神经网络工具箱 newff 函数。
3. 设计包含 3 个隐藏层的 BP 神经网络，输入层维度为样本维度；隐藏层维度分别是 128，32，2，隐藏层使用 relu 激活函数；输出层维度 1，使用 sigmoid 激活函数。

(四)模型训练与性能评估

1. 使用五折交叉验证（5-fold cross-validation）评估神经网络在该数据上的性能。
2. 五折交叉验证：将正负样本各分为数量相等的 5 份：1~40，41~80，81~120，121~160，161~200，并把 1 份正样本与 1 份负样本合并构成 1 个子集，则原数据分成了 5 个子集，且子集中正负样本比例 1：1 保持不变。对模型进行 5

次训练测试：

2.1. 第 i 次时，选择第 i 份子集作为测试集，其余 4 份数据子集作为训练集。使用训练集训练数据，并使用测试集验证/测试性能，计算测试集的 AUROC 值。

2.2. 模型倒数第 2 层维度为 2，导出测试集数据在这一层上的数值，输出成二维散点图，包含两个子图：子图 1 的标签为测试样本的预测标签（取决于模型输出，将输出大于 0.5 的样本预测为正样本，否则预测为负样本），以不同颜色区分不同的预测结果；子图 2 的标签为测试样本的真实标签，以不同颜色区分不同的真实结果。

2.3. 完成 5 次，使得每一个子集都作为测试集 1 次，作为训练集 4 次。计算整体的 AUROC。

3. 打断正负样本排列，重新划分训练集与测试集，重新训练模型，查看性能变化。

(五)特征选择和特征提取

1. 特征选择：

针对原始数据，计算每个特征的方差，选取两个方差最大的特征。将正负样本点输出为二维散点图，以颜色区分。

2. 特征提取：

针对原始数据，调用 PCA 降维函数将数据降到二维，将正负样本点输出为二维散点图，以颜色区分。

代码输出：

- 原始数据的散点图；

- 测试集的散点图；
- 经过特征选择后的散点图；
- 经过 PCA 降维后的散点图。

[实验代码和结果]

BPNet.py

```
import torch
from torch import nn
# BP 神经网络模型
class BPNetwork(nn.Module):
def __init__(self, input_dim):
super(BPNetwork, self).__init__()
self.model = nn.Sequential(
nn.Linear(input_dim, 128),
nn.ReLU(),
nn.Linear(128, 32),
nn.ReLU(),
nn.Linear(32, 2),
nn.ReLU(),
nn.Linear(2, 1),
nn.Sigmoid()
)
def forward(self, x):
return self.model(x)
```

Project1.py

```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import umap
import matplotlib.pyplot as plt
import torch
from torch import nn
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import KFold
```

```
from BPNet import BPNetwork

# 读取数据时指定没有表头
positive_data = pd.read_csv("data/positive.csv",
header=None)
negative_data = pd.read_csv("data/negative.csv",
header=None)
positive_data = positive_data.T
negative_data = negative_data.T

# 添加统一的列名
num_columns = positive_data.shape[1]
column_names = [f"Feature_{i}" for i in range(num_columns)]
positive_data.columns = column_names
negative_data.columns = column_names

# 拼接数据
data = pd.concat([positive_data, negative_data],
axis=0).reset_index(drop=True)
labels = np.concatenate([np.ones(len(positive_data)),
np.zeros(len(negative_data))])

# （一）UMAP 可视化（原始数据）
reducer = umap.UMAP(n_components=2, random_state=42)
reduced_data = reducer.fit_transform(data)

plt.figure(figsize=(8, 6))
plt.scatter(reduced_data[:200, 0], reduced_data[:200, 1],
label="Positive", alpha=0.7)
plt.scatter(reduced_data[200:, 0], reduced_data[200:, 1],
label="Negative", alpha=0.7)
plt.title("UMAP Visualization of Original Data")
plt.legend()
plt.tight_layout()
plt.savefig("figure/UMAP_OriginalData.png", dpi=300)
plt.show()


# （二）固定五折交叉验证（手动切分）
```

```
positive_indices = np.arange(0, 200)
negative_indices = np.arange(200, 400)

# 将正负样本各分成 5 份，每份 40 个样本
pos_splits = [positive_indices[i*40:(i+1)*40] for i in
range(5)]
neg_splits = [negative_indices[i*40:(i+1)*40] for i in
range(5)]
subsets = []
for i in range(5):
test_indices = np.concatenate([pos_splits[i],
neg_splits[i]])
train_indices = np.concatenate(
[pos_splits[j] for j in range(5) if j != i] +
[neg_splits[j] for j in range(5) if j != i]
)
subsets.append((test_indices, train_indices))

all_auroc_fixed = []

# 训练并测试（固定五折）
for fold, (test_indices, train_indices) in
enumerate(subsets):
train_data, test_data = data.iloc[train_indices],
data.iloc[test_indices]
train_labels, test_labels = labels[train_indices],
labels[test_indices]

# 转换为 Tensor
train_data = torch.tensor(train_data.values,
dtype=torch.float32)
train_labels = torch.tensor(train_labels,
dtype=torch.float32).unsqueeze(1)
test_data = torch.tensor(test_data.values,
dtype=torch.float32)
test_labels = torch.tensor(test_labels,
dtype=torch.float32).unsqueeze(1)

# 模型实例化
```

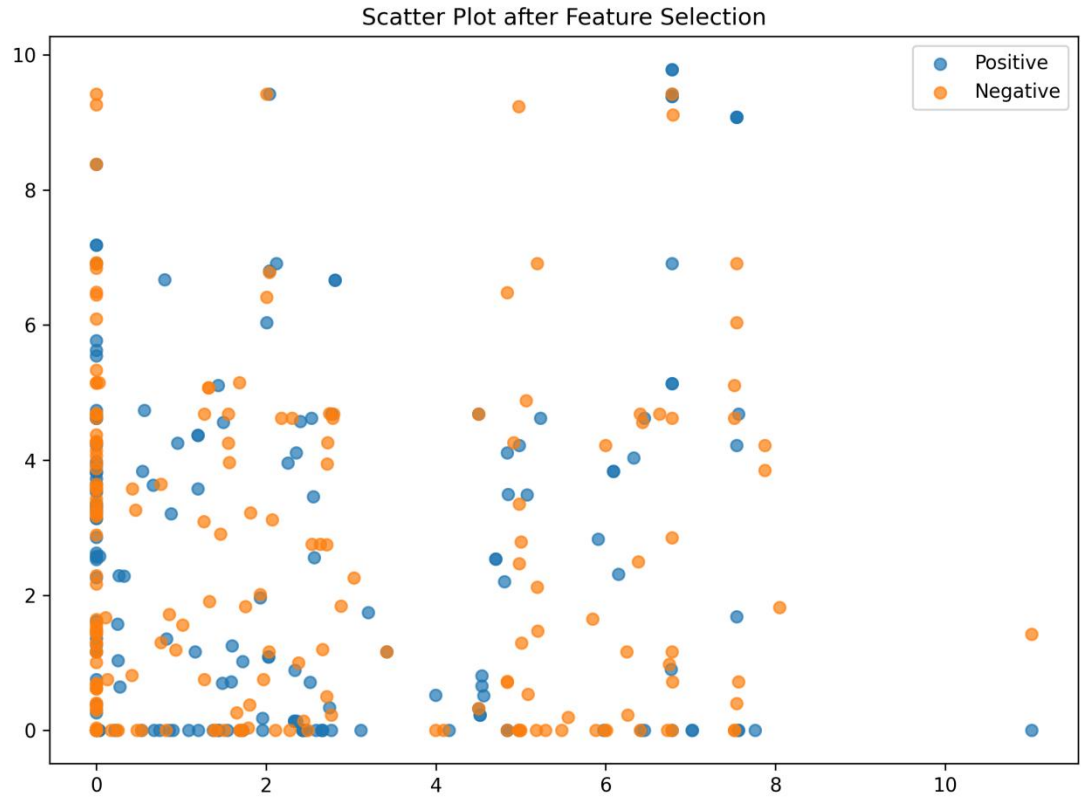
```
model = BPNetwork(input_dim=train_data.shape[1])
criterion = nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# 训练模型
for epoch in range(20):
model.train()
optimizer.zero_grad()
outputs = model(train_data)
loss = criterion(outputs, train_labels)
loss.backward()
optimizer.step()

# 测试模型
model.eval()
with torch.no_grad():
predictions = model(test_data).numpy()
auroc = roc_auc_score(test_labels.numpy(), predictions)
all_auroc_fixed.append(auroc)

# 倒数第二层的输出（2 维）
intermediate_layer =
nn.Sequential(*list(model.model.children())[:-2])
reduced_test_data = intermediate_layer(test_data).numpy()

# 绘制并保存测试集散点图
plt.figure(figsize=(12, 5))

# 子图 1：预测标签
plt.subplot(1, 2, 1)
plt.scatter(reduced_test_data[:, 0],
reduced_test_data[:, 1],
c=(predictions > 0.5).squeeze(),
cmap="coolwarm")
plt.title(f"Fold {fold+1}: Predicted Labels")

# 子图 2：真实标签
plt.subplot(1, 2, 2)
plt.scatter(reduced_test_data[:, 0],
```

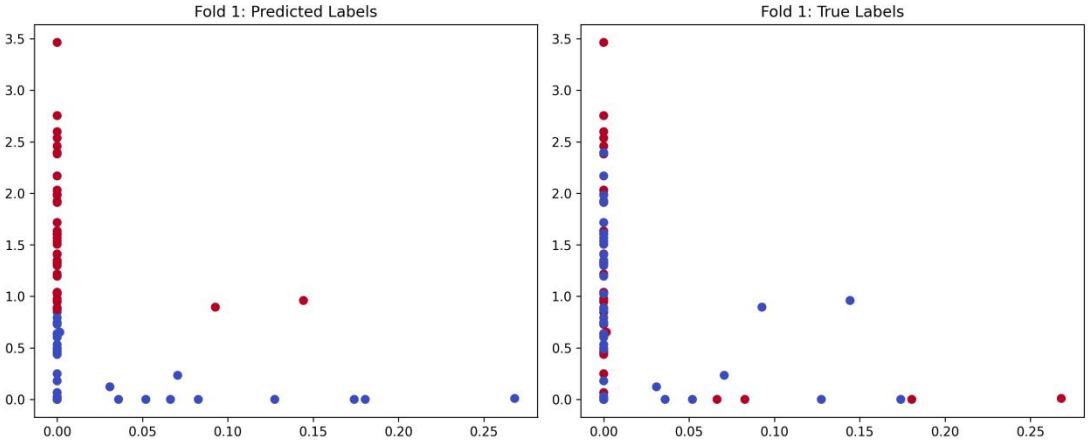
```
reduced_test_data[:, 1],
c=test_labels.numpy().squeeze(),
cmap="coolwarm")
plt.title(f"Fold {fold+1}: True Labels")

plt.tight_layout()
save_path = f"figure/Fold_{fold+1}_TestScatter.png"
plt.savefig(save_path, dpi=300)
plt.show()

fixed_mean_auroc = np.mean(all_auroc_fixed)
print(f"[固定五折] Average AUROC across folds:
{fixed_mean_auroc:.4f}")


# （三）特征选择（方差最大）
variances = data.var(axis=0)
top_features = variances.nlargest(2).index
selected_data = data[top_features]

plt.figure(figsize=(8, 6))
plt.scatter(selected_data.iloc[:200, 0],
selected_data.iloc[:200, 1],
label="Positive", alpha=0.7)
plt.scatter(selected_data.iloc[200:, 0],
selected_data.iloc[200:, 1],
label="Negative", alpha=0.7)
plt.title("Scatter Plot after Feature Selection")
plt.legend()
plt.tight_layout()
plt.savefig("figure/FeatureSelection_2D.png", dpi=300)
plt.show()

# （四）PCA 降维到二维
pca = PCA(n_components=2)
pca_data = pca.fit_transform(data)

plt.figure(figsize=(8, 6))
```

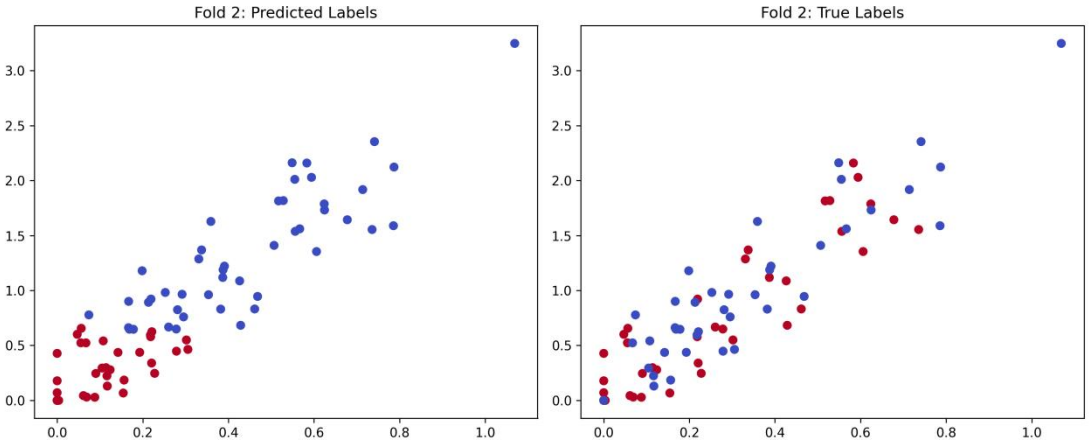
```
plt.scatter(pca_data[:200, 0], pca_data[:200, 1],
label="Positive", alpha=0.7)
plt.scatter(pca_data[200:, 0], pca_data[200:, 1],
label="Negative", alpha=0.7)
plt.title("Scatter Plot after PCA")
plt.legend()
plt.tight_layout()
plt.savefig("figure/PCA_2D.png", dpi=300)
plt.show()


# （五）多次随机划分的五折交叉验证（与固定划分作对比）
num_runs = 5 # 做 5 次随机划分
all_auroc_runs = [] # 存储每次随机五折的平均 AUROC

# 将 DataFrame 转成 numpy 方便 KFold 索引
data_np = data.values
labels_np = labels

for run_id in range(num_runs):
# 这里通过 KFold shuffle=True 并设置不同的 random_state
kf = KFold(n_splits=5, shuffle=True, random_state=run_id)
run_auroc_list = []
for fold_i, (train_idx, test_idx) in
enumerate(kf.split(data_np)):
# 拆分数据
X_train = torch.tensor(data_np[train_idx],
dtype=torch.float32)
X_test = torch.tensor(data_np[test_idx],
dtype=torch.float32)
y_train = torch.tensor(labels_np[train_idx],
dtype=torch.float32).unsqueeze(1)
y_test = torch.tensor(labels_np[test_idx],
dtype=torch.float32).unsqueeze(1)
# 创建 BPNetwork
model = BPNetwork(input_dim=X_train.shape[1])
criterion = nn.BCELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)
# 训练
```

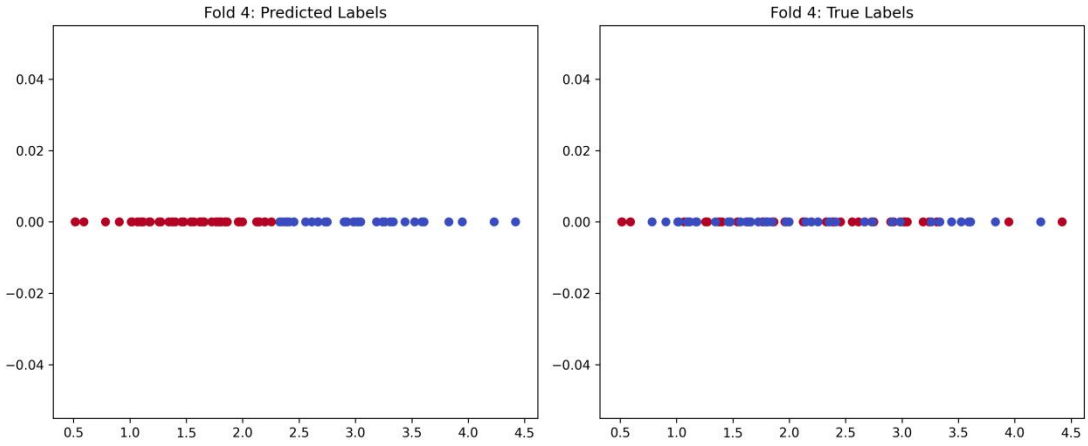
```
for epoch in range(20):
model.train()
optimizer.zero_grad()
outputs = model(X_train)
loss = criterion(outputs, y_train)
loss.backward()
optimizer.step()
# 测试
model.eval()
with torch.no_grad():
predictions = model(X_test).numpy()
auroc = roc_auc_score(y_test.numpy(), predictions)
run_auroc_list.append(auroc)
mean_auroc = np.mean(run_auroc_list)
all_auroc_runs.append(mean_auroc)
print(f"[随机划分第 {run_id+1} 次] 5-Fold Average AUROC:
{mean_auroc:.4f}")

print("随机多次的 AUROC 结果:", all_auroc_runs)
print("随机多次平均 AUROC:", np.mean(all_auroc_runs))
```

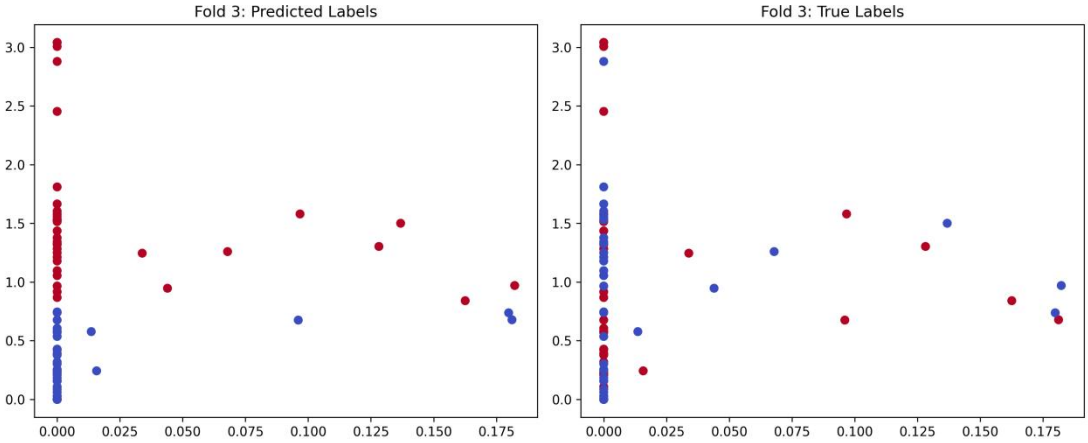
实验结果:

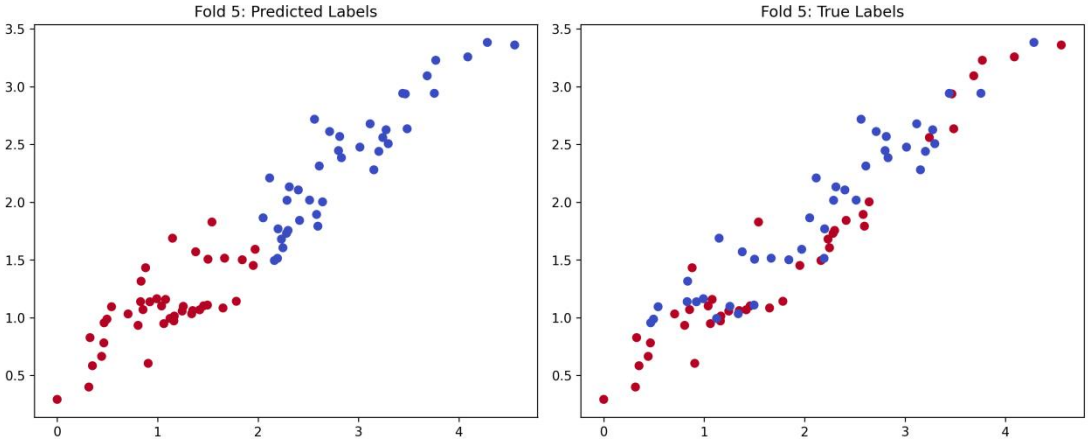

Scatter Plot after Feature Selection
Positive
Negative

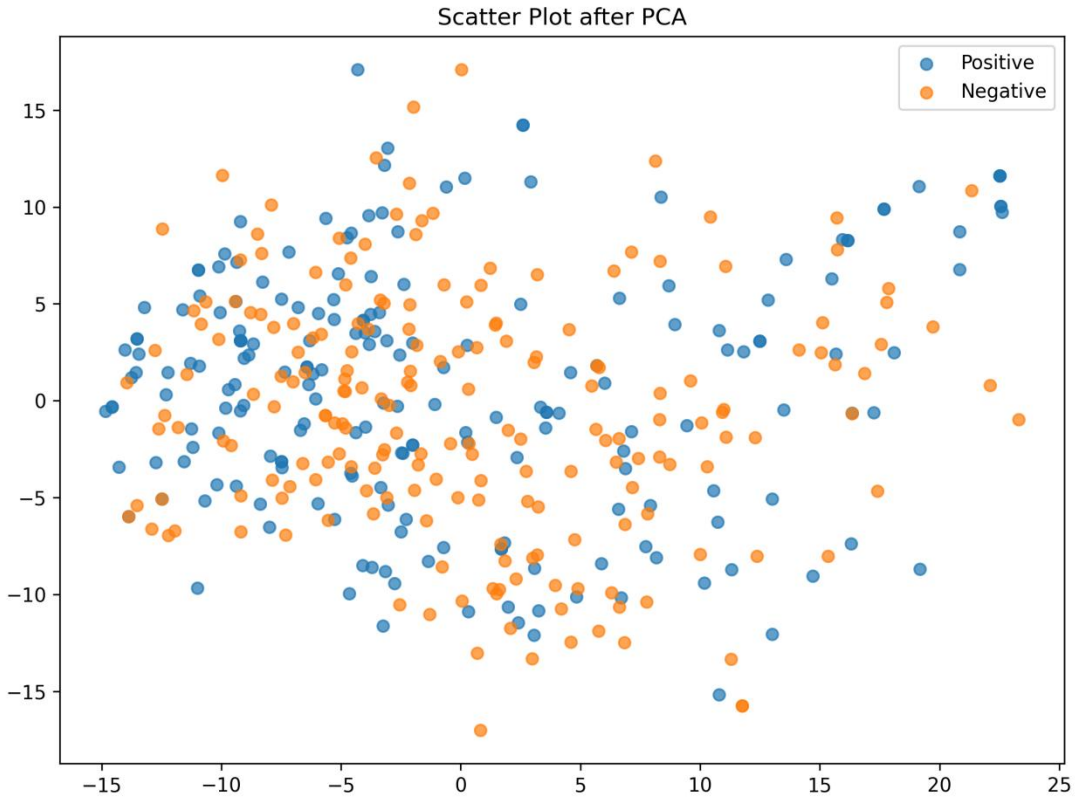

Fold 1: Predicted Labels
Fold 1: True Labels

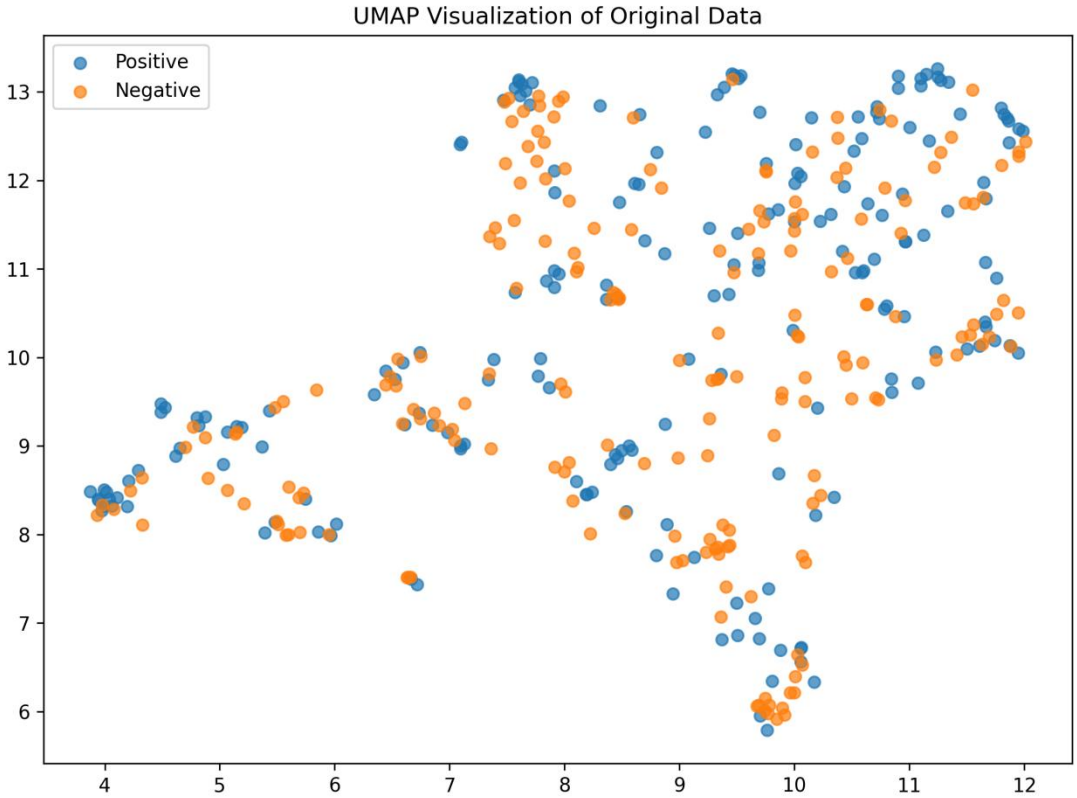

Fold 2: Predicted Labels
Fold 2: True Labels


Fold 4: Predicted Labels
Fold 4: True Labels


Fold 3: Predicted Labels
Fold 3: True Labels


Fold 5: Predicted Labels
Fold 5: True Labels





[固定五折] Average AUROC across folds: 0.5677
[随机划分第 1 次] 5-Fold Average AUROC: 0.5799
[随机划分第 2 次] 5-Fold Average AUROC: 0.6233
[随机划分第 3 次] 5-Fold Average AUROC: 0.5558

[随机划分第 4 次] 5-Fold Average AUROC: 0.5892
[随机划分第 5 次] 5-Fold Average AUROC: 0.6011
随机多次的 AUROC 结果: [0.5799046993076464, 0.6232542265455698, 0.5558180563148324, 0.5891858566301762, 0.6011141089116541]
随机多次平均 AUROC: 0.5898553895419758

[小结或讨论]

在本次实验中，我采用了一个由三个隐藏层构成的多层前馈神经网络来解决一个二分类问题，主要探索了固定五折和多次随机五折交叉验证的区别及其对模型性能的影响。通过详细分析不同验证方法下的模型表现，并结合原始数据的降维可视化，我尝试了深入理解模型性能的局限性和数据的特性。

实验设计与方法

实验中，首先我进行了固定五折交叉验证。该方法确保了测试集在每一折中正负样本的比例均衡，从而可以公平地评估模型在各个子集上的表现。结果显示，模型在所有折上的平均 AUROC 值接近随机猜测水平，这表明模型对于正负样本的区分能力极低。

为了进一步探索这一结果，我引入了随机五折交叉验证，旨在测试模型在不同随机样本组合下的泛化能力。这种方法通过在每次实验中打乱样本顺序，重新进行五折划分，理论上应提供对模型稳定性的更全面评估。然而，即便在多次随机化的条件下，模型的 AUROC 表现仍然没有显著提高。这种一致的低性能提示了数据特性可能是主导因素。

数据分析与可视化

通过 UMAP 对原始数据进行降维并可视化，我观察到正负样本在二维投影中高度重叠，几乎无法区分。这一发现是至关重要的，因为它直接指出了数据本身的复

杂性和模型在当前特征空间内学习有效决策边界的困难。此外，尽管我尝试了通过计算方差进行特征选择和使用 PCA 进行降维，希望能够找到更有区分力的特征表达，但这些方法同样未能显著改善样本的可分性。

实验结果分析

根据实验数据，固定五折交叉验证的平均 AUROC 为 0.5677，这表明模型的整体性能略高于随机猜测，但仍然不够理想。随后进行的多次随机五折交叉验证显示了不同的表现，平均 AUROC 值分别为：0.5799、0.6233、0.5558、0.5892 和 0.6011。这些结果揭示了模型在不同随机数据划分下的性能波动，其中第二次随机划分得到的 AUROC 最高为 0.6233，而第三次则降至 0.5558，显示出相对较大的变动性。

结果讨论

模型在随机五折交叉验证中的不同表现可能受多种因素影响。首先，不同的数据划分可能导致训练集和测试集的样本特征分布有所差异，从而影响模型的学习和泛化。特别是在数据本身可能存在噪声或样本分布不均的情况下，不同的样本组合可能会对模型性能产生显著影响。其次，神经网络作为一种高度非线性的模型，其性能对初始化权重和训练过程中的微小变化可能也非常敏感。

此外，尽管固定五折的平均 AUROC 较低，但随机划分的结果更能反映模型在不同数据组合下的泛化能力。这种波动也可能表明模型对数据中的某些特征过度敏感，或者学习到了一些在特定数据划分上有效但不具普遍性的规律。

总体而言，这次实验的结果表明，虽然神经网络能够从数据中学习到一定的分类规律，但其表现的稳定性和泛化能力还有待提高。

1．特征工程：进一步分析和筛选对分类任务更有帮助的特征，可能包括引入外部数据或使用更复杂的特征提取方法。

2．模型调优：调整网络结构，如增加或减少隐藏层、改变激活函数等；优化训练算法，如调整学习率、使用不同的优化器或引入正则化技术。

3．增强数据处理：实施更复杂的数据预处理策略，如特征标准化、数据增强等，以减少模型对特定数据划分的敏感性。

4．扩展验证方法：使用更加复杂的交叉验证策略，如分层交叉验证或重复随机划分交叉验证，以更准确评估模型的稳定性和泛化能力

这些结果使我认识到，单纯增加模型的复杂性或调整训练参数（如学习率和迭代次数）可能不足以解决问题。在未来的工作中，我需要更加关注数据预处理和特征工程的策略。例如，更深入地分析各特征对分类结果的影响，尝试引入新的数据表示或利用高级特征提取技术（如深度学习自动特征学习）。同时，考虑到当前模型的训练策略可能未充分优化，引入更复杂的正则化方法或采用不同的优化算法也可能有助于改善模型性能。