教程 一 : 基于神经网络的分类任务实现

提交日期：2024-07-18

提交人：张健豪

指导教师：秦科

目录

[**1.** **题目** 3](#_Toc172391304)

[**2.** **算法原理描述** 3](#_Toc172391305)

[2.1 神经网络模型 3](#_Toc172391306)

[**3.** **实验结果与分析** 4](#_Toc172391307)

[3.1 实验数据 4](#_Toc172391308)

[3.2 实验设置 4](#_Toc172391309)

[3.3 实验结果 5](#_Toc172391310)

[**4.** **总结** 6](#_Toc172391311)

[**5.** **参考文献** 6](#_Toc172391312)

[**6.** **完整代码** 7](#_Toc172391313)

# **题目**

基于神经网络的分类任务实现

# **算法原理描述**

## 2.1 神经网络模型

(1) 类初始化 (init):

input\_size: 输入层的节点数量（特征数量）。

hidden\_size: 隐藏层的节点数量。

output\_size: 输出层的节点数量（预测值）。

learning\_rate: 学习率，控制参数更新的步长。

num\_iterations: 迭代次数，决定模型训练的轮数。

(2) 激活函数:

ReLU: Rectified Linear Unit，处理非线性关系，避免梯度消失问题。

(3) 模型训练 (fit):

数据标准化处理，避免不同特征量级的影响。

前向传播：计算每层的线性组合 z 和激活值 a。

损失计算：使用均方误差损失函数 (MSELoss)。

反向传播：计算每层的梯度并进行参数更新。

调试信息：每 100 次迭代打印一次损失值。

(4) 预测函数 (predict):

前向传播：计算输入数据通过网络后的输出值。

(5) 数据准备和标准化:

StandardScaler: 对训练集和验证集进行标准化处理，使每个特征具有零均值和单位方差。

(6) 训练和验证:

model.fit(X\_train, y\_train): 使用训练数据训练神经网络。

evaluate\_model(model, X\_val, y\_val, criterion): 使用验证数据评估模型性能。

# **实验结果与分析**

## 3.1 实验数据

（1）训练集 (covid.train.csv)

行数: 2700

列数: 95

每一行表示一个数据样本，每一列表示一个特征或标签。

包含所有特征的描述统计信息。

（2）测试集 (covid.test.csv)

行数: 506

列数: 94

与训练集相似，包含所有特征但不包括标签。

## 3.2 实验设置

（1）数据预处理：

特征提取与标签：从训练集中移除id列和tested\_positive列，剩余的94列作为特征X；tested\_positive列作为标签y。

数据集划分：使用train\_test\_split函数将训练集划分为训练子集和验证子集，比例为80%训练子集和20%验证子集。设置random\_state=42以确保划分的可重复性。

特征标准化：使用StandardScaler对特征进行标准化，使每个特征的均值为0，方差为1。对训练子集进行拟合和变换，对验证子集和测试集进行变换。

（2）模型训练与评估：

使用 PyTorch 实现神经网络模型，定义 ReLU 激活函数、均方误差损失函数（MSELoss）和 Adam 优化器。

在训练子集上训练模型，并在验证子集上评估模型性能。

## 3.3 实验结果

神经网络训练过程

Iteration 0, Loss: 337.6845397949219

Iteration 100, Loss: 19.045970916748047

Iteration 200, Loss: 5.482507228851318

Iteration 300, Loss: 3.305593967437744

Iteration 400, Loss: 2.2610867023468018

Iteration 500, Loss: 1.709823727607727

Iteration 600, Loss: 1.3637670278549194

Iteration 700, Loss: 1.1305726766586304

Iteration 800, Loss: 0.952626645565033

Iteration 900, Loss: 0.8569190502166748

Iteration 1000, Loss: 0.7908409237861633

Iteration 1100, Loss: 0.7379313707351685

Iteration 1200, Loss: 0.6920252442359924

Iteration 1300, Loss: 0.6523813605308533

Iteration 1400, Loss: 0.6136134266853333

Iteration 1500, Loss: 0.5833501219749451

Iteration 1600, Loss: 0.556597113609314

Iteration 1700, Loss: 0.5342326164245605

Iteration 1800, Loss: 0.5131142139434814

Iteration 1900, Loss: 0.4923700988292694

Iteration 2000, Loss: 0.4762335419654846

Iteration 2100, Loss: 0.4598213732242584

Iteration 2200, Loss: 0.44369250535964966

Iteration 2300, Loss: 0.4288451373577118

Iteration 2400, Loss: 0.41488319635391235

Iteration 2500, Loss: 0.40167436003685

Iteration 2600, Loss: 0.39145851135253906

Iteration 2700, Loss: 0.3781084418296814

Iteration 2800, Loss: 0.36496031284332275

Iteration 2900, Loss: 0.3579322099685669

最终验证集损失

Final Validation Loss: 1.1026779413223267、

神经网络在训练过程中，损失逐渐减小，表明模型在不断学习和优化。从最终验证集的损失可以看出，模型在验证集上的表现较好。

# **总结**

从实验结果来看，神经网络模型在处理COVID-19数据集的回归任务时表现良好。模型在训练过程中逐步减少损失，并在验证集上取得了较低的均方误差。具体结论如下：

（1）非线性激活函数：使用 ReLU 激活函数，有助于模型捕捉数据中的非线性关系，提高模型的拟合能力。

（2）数据标准化：对特征进行标准化处理，避免了不同特征量级对模型训练的影响，确保了模型训练的稳定性。

（3）模型优化：Adam 优化器在训练过程中表现出色，加快了收敛速度。

# **参考文献**

【1】 花书第一部分：数学基础

【2】逻辑回归：https://zhuanlan.zhihu.com/p/74874291

【3】 BP神经网络推导：<https://zhuanlan.zhihu.com/p/74488184>

# **完整代码**

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

import torch

import torch.nn as nn

import torch.optim as optim

# 读取数据

train\_df = pd.read\_csv(r'D:\pythonproject\zhang\covid.train.csv')

test\_df = pd.read\_csv(r'D:\pythonproject\zhang\covid.test.csv')

# 提取特征和标签

X\_train = train\_df.drop(columns=['id', 'tested\_positive']).values

y\_train = train\_df['tested\_positive'].values

# 划分验证集

X\_train, X\_val, y\_train, y\_val = train\_test\_split(X\_train, y\_train, test\_size=0.2, random\_state=42)

# 标准化数据

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_val = scaler.transform(X\_val)

X\_test = scaler.transform(test\_df.drop(columns=['id']).values)

# 转换为 PyTorch 的张量

X\_train = torch.tensor(X\_train, dtype=torch.float32)

y\_train = torch.tensor(y\_train, dtype=torch.float32).view(-1, 1)

X\_val = torch.tensor(X\_val, dtype=torch.float32)

y\_val = torch.tensor(y\_val, dtype=torch.float32).view(-1, 1)

X\_test = torch.tensor(X\_test, dtype=torch.float32)

class NeuralNetwork(nn.Module):

def \_\_init\_\_(self, input\_size, hidden\_size, output\_size):

super(NeuralNetwork, self).\_\_init\_\_()

self.net = nn.Sequential(

nn.Linear(input\_size, 64),

nn.ReLU(),

nn.Linear(64, 16),

nn.ReLU(),

nn.Linear(16, 8),

nn.ReLU(),

nn.Linear(8, 4),

nn.ReLU(),

nn.Linear(4, 1)

)

def forward(self, x):

return self.net(x)

# 初始化模型、损失函数和优化器

input\_size = X\_train.shape[1]

hidden\_size = 64

output\_size = 1

learning\_rate = 0.001

num\_iterations = 3000

model = NeuralNetwork(input\_size, hidden\_size, output\_size)

criterion = nn.MSELoss() # 二分类任务使用 BCELoss

optimizer = optim.Adam(model.parameters(), lr=learning\_rate)

# 定义计算验证集分数的函数

def evaluate\_model(model, X\_val, y\_val, criterion):

model.eval()

with torch.no\_grad():

output = model(X\_val)

loss = criterion(output, y\_val)

return loss.item()

# 训练模型

for epoch in range(num\_iterations):

model.train()

optimizer.zero\_grad()

output = model(X\_train)

loss = criterion(output, y\_train)

loss.backward()

optimizer.step()

if epoch % 100 == 0:

print(f"Iteration {epoch}, Loss: {loss.item()}")

# 在验证集上进行预测

model.eval()

with torch.no\_grad():

# 最终验证集分数

final\_val\_loss = evaluate\_model(model, X\_val, y\_val, criterion)

print(f"Final Validation Loss: {final\_val\_loss}")