

**软件工程理论及应用**

**结课报告**

题 目： 人工神经网络的设计与实现

院 系：

班 级：

学 号：

学生姓名：

日期：2023年10月

**《软件工程理论及应用》结课报告**

**任 务 书**

**一、 目的与要求**

1．培养信息类专业学生对软件开发过程的全面理解。

2．熟练掌握面向对象设计的基本原则，如单一职责原则、开闭原则、里氏代换原则、依赖倒转原则、接口隔离原则、合成复用原则、迪米特法则等。

3．熟练掌握常用的创建型设计模式，如简单工厂模式、工厂方法模式、抽象工厂模式、单例模式、原型模式、建造者模式。

4. 熟练掌握常用的结构型设计模式，如享元模式、外观模式、适配器模式、桥接模式、组合模式、装饰模式、代理模式。

5．熟练掌握用类图表达类与类之间的关系。

**二、 主要内容**

1．对神经元进行抽象，封装诸如神经元的输入、输出和权值等常见属性。

2．对激活函数进行抽象，设计具体的阈值函数、符号函数、单极性Sigmoid函数、双极性Sigmoid函数。

3．对神经网络层进行抽象，表示该层神经元的集合。

4．对学习算法进行抽象，设计感知器学习规则、误差反向传播学习规则。

5．对神经网络进行抽象，表示神经元层的集合。

6．利用设计的单层感知器网络，解决逻辑与、逻辑或的运算问题。

7．利用设计的误差反向传播网络，解决语音特征信号识别问题。

**三、 实验成果要求**

1．基本要求：可对BP神经网络进行适当改进，使语音特征信号识别的准确率超过80%。

2．基本要求：可尝试利用单层感知器模型解决多分类问题。

3．基本要求：撰写结课报告。结课报告要求能够反映设计的思路和设计的过程，清楚记录调试中遇到的问题，参考资料真实有效。

4．基本要求：学生独立完成设计任务，严禁抄袭、找人代做等。

5．基本要求：结课报告必须提交电子版和打印版。结课报告必须符合格式要求。

6．基本要求：将源程序和实验报告电子版压缩后提交，并验收合格。

**四、 考核方式**

1．学生讲解设计思路。教师提问，学生回答。

2．教师审核学生实验报告。

**人工神经网络的设计与实现**

**1. 单层感知器网络的设计与实现**

**1.1 单层感知器网络原理**

其中σ为激活函数，常用的有RelU,Sigmoid等。

**1.2 单层感知器网络的实现**

在python中，可以用pytorch深度学习框架搭建一个单层感知器类。

import torch.nn as nn

class Net(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.fc1 = nn.Linear(in\_features=24,out\_features=4)

self.act1 = nn.ReLU()

def forward(self,x):

x = self.act1(self.fc1(x))

return x

**1.3 利用单层感知器解决逻辑运算问题**

初始化感知器中的参数，设定损失函数和学习法则，利用预测值与真实值之间的误差进行参数修正，使模型能够适应问题。

**2. 误差反向传播网络的设计与实现**

**2.1 误差反向传播网络的原理**

**2.2 误差反向传播网络的实现**

在python中，使用pytorch中的优化器可以实现反向传播和修改参数。

optimizer.zero\_grad()

loss.backward()

optimizer.step()

**2.3 误差反向传播网络解决语音特征信号识别问题**

数据表有四张，每张表格是一类数据，在读入数据之前，先在每张表格第一行添加一行，分别是标签label和24种特征从字母A到字母X。

首先读入表格，并将特征和标签分开。

import pandas as pd

def scan\_data(path):

df = pd.read\_csv(path)

features = df.drop(['label'],axis=1)

labels = df.loc[:,'label']-1

return features,labels

X1,y1 = scan\_data('./data/data1.csv')

X2,y2 = scan\_data('./data/data2.csv')

X3,y3 = scan\_data('./data/data3.csv')

X4,y4 = scan\_data('./data/data4.csv')

将数据转为张量，并划分训练集和测试集。共两千个数据，随机打乱并分为1600和400 ，分别为训练集和测试集。采用小批量学习，批量大小为16.

import numpy as np

import torch

X = np.array([X1,X2,X3,X4])

X\_tensor = torch.from\_numpy(X).float().reshape(-1,24)

y = np.array([y1,y2,y3,y4])

y\_tensor = torch.from\_numpy(y).reshape(-1)

from torch.utils.data import TensorDataset,DataLoader,random\_split

dataset = TensorDataset(X\_tensor,y\_tensor)

train\_size = 1600

test\_size = 400

train\_set,test\_set = random\_split(dataset,[train\_size,test\_size])

train\_load = DataLoader(train\_set,batch\_size=64,shuffle=True)

test\_load = DataLoader(test\_set,batch\_size=64,shuffle=True)

定义模型，神经网络类。

import torch.nn as nn

class Net(nn.Module):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.fc1 = nn.Linear(in\_features=24,out\_features=64)

self.act1 = nn.ReLU()

self.fc2 = nn.Linear(in\_features=64,out\_features=128)

self.act2 = nn.ReLU()

self.fc3 = nn.Linear(in\_features=128,out\_features=4)

def forward(self,x):

x = self.act1(self.fc1(x))

x = self.act2(self.fc2(x))

x = self.fc3(x)

return x

初始化模型，设定损失函数和超参数。

选用SGD优化器，学习率为0.05，损失函数为交叉熵损失函数。

model = Net()

optimizer = torch.optim.SGD(model.parameters(),lr=0.05)

loss\_fn = nn.CrossEntropyLoss()

num\_epochs = 200

设定学习规则，并进行训练。

def train\_model(model,epochs,loss\_fn,optimizer,train\_set):

for epoch in range(epochs):

loss\_train = 0.0

for X,y in train\_set:

y\_pre = model(X)

loss = loss\_fn(y\_pre,y)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

loss\_train += loss.item()

print('epoch: ',epoch,' loss: ',loss\_train/len(train\_set))

train\_model(model,num\_epochs,loss\_fn,optimizer,train\_load)

模型评价和保存模型。

def accuracy(dataset,model):

correct = 0

total = 0

with torch.no\_grad():

for X,y in dataset:

outputs = model(X)

\_,y\_pre = torch.max(outputs,dim=1)

total += y.shape[0]

correct += int((y\_pre == y).sum())

return correct/total

print('Train Accuracy: ',accuracy(train\_load,model))

print('Test Accuracy: ',accuracy(test\_load,model))

torch.save(model.state\_dict(),'./200epochs.pth')

**3. 总结**

在python中，我们可以在pytorch框架的基础上轻松的构建神经网络类，并可以根据具体问题选择优化器和损失函数进行参数学习。最后可以将模型的参数保存。

模型训练过程和结果如下，其中epoch为训练轮数，loss为损失率反映损失函数值大小：

**4. 参考文献**

Pytorch官方文档[torchtext — Torchtext 0.16 documentation (pytorch.org)](https://pytorch.org/text/stable/)

注意点：

（1）正文为宋体，小四号字行间距为20

（2）神经网络的原理部分要讲清楚算法的步骤。

（3）神经网络的设计部分要讲清楚每个类的作用，以及使用了哪些面向对象的设计原则和设计模式。

（4）神经网络的实现部分要列出关键代码。

（5）神经网络应用部分要列出关键结果。

（6）参考文献要符合格式要求。