



计算智能导论实验报告

|  |  |
| --- | --- |
| 名 称： | 人工神经网络的实现 |
| 学生姓名： | 刘易行 |
| 学 号： | 2020905896 |
| 专 业： | 人工智能 |
| 班 级： | 2020240401 |
| 指导教师： | 高婧洁 |
| 时 间： | 2022.11. |

# 一、实验目的

掌握人工神经网络的特性和应用范围，能够用人工神经网络解决

实际预测问题，能够根据应用需求，设计与实现人工神经网络，观测

神经网络的训练过程和输出。

# 二、实验内容

1、训练 BP 神经网络，使其能够拟合某一附加有白噪声的正弦

样本数据。最后将带有白噪声的正弦样本数据点、标准正弦曲线以及

拟合后的正弦曲线进行对比。

2、下表所示为某地区公路运力的历史统计数据表

（1）请根据该表，建立相应的 BP 神经网络预测模型，拟合历

年公路客运量曲线和历年公路货运量曲线。

（2）根据给出的 2010 年和 2011 年的数据，预测相应的公路客

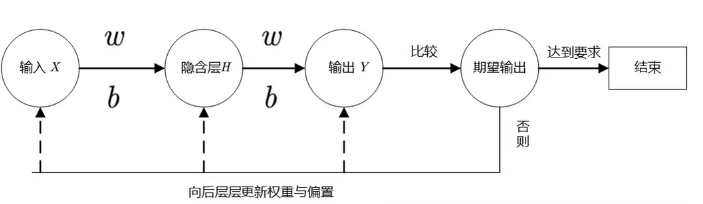
运量和货运量

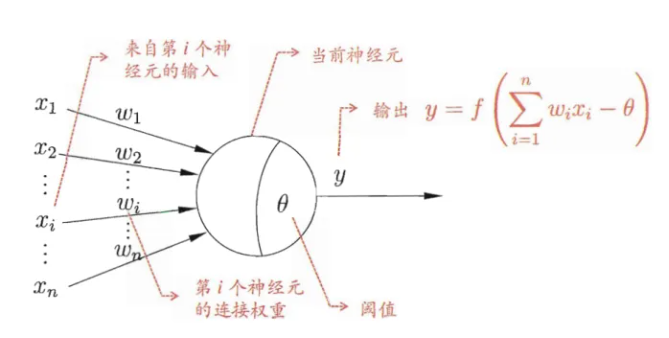
# 三、实验原理

本次实验的主题内容是人工神经网络，两道题目所使用的模型均为BP神经网络。

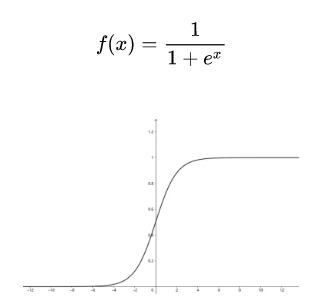
BP神经网络，BP是Back Propagation的简写，意味反向传播。

BP神经网络的主要特点是：信号是正向传播的，而误差是反向传播的。BP网络能学习和存贮大量的输入-输出模式映射关系，而无需事前揭示描述这种映射关系的数学方程。它的学习规则是使用最速下降法，通过反向传播来不断调整网络的权值和阈值，使网络的误差平方和最小。

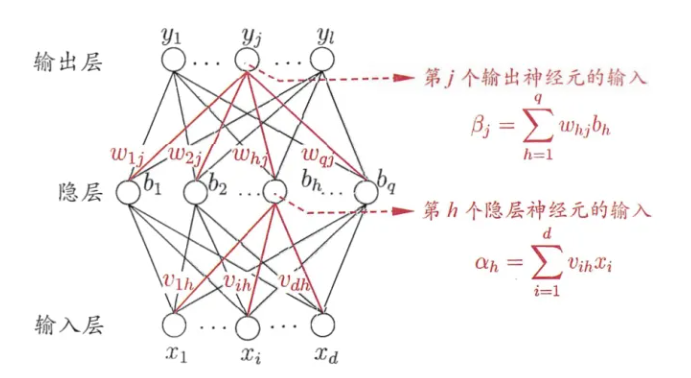




激活函数选用Sigmoid（logistic）函数：



BP网络由输入层、隐藏层、输出层组成。



# 四、实验步骤

## 题目一

* 题目一实现方法采用Python和Matlab两种语言实现，其中Python使用目前世界上最流行的人工智能框架Pytorch实现，Matlab则使用老师课堂上讲解的内置算法完成。
* 第一步：构造原始无噪声数据集并套用正弦函数，下述代码基于numpy
  + # 数据准备
  + data\_x = np.linspace(-2 \* np.pi, 2 \* np.pi, 400)
  + data\_y = np.sin(data\_x) # 套用正弦函数
* 第二步：为原始信号添加高斯白噪声构成噪声新信号
* 第三步：使用Pytorch构建BP神经网络模型，定义输入层、隐藏层、输出层及forward()函数
* 第四步：实例化网络模型、配置优化器、损失函数等
* 第五步：训练模型，此处训练1000次，训练过程中每100次打印误差
* 第六步：绘制拟合曲线，分析结果
* Matlab步骤类似，不予赘述

## 题目二

* 题目二实现方法采用Python和Matlab两种语言实现，其中Python使用目前世界上最流行的人工智能框架Pytorch实现，Matlab则使用老师课堂上讲解的内置算法完成。
* 第一步：读取数据，使用pandas中的readcsv()函数读取

dataFrame = pd.read\_csv('TrafficData.csv', encoding='utf-8')

print(dataFrame.head())

x = dataFrame[['人数', '机动车数', '公路面积']]

y = dataFrame[['公路客运量', '公路货运量']]

* 第二步：处理数据，数据归一化，使用sklearn中的MinMaxScaler实现

x\_scaler = MinMaxScaler(feature\_range=(-1, 1))

y\_scaler = MinMaxScaler(feature\_range=(-1, 1))

x = x\_scaler.fit\_transform(x)

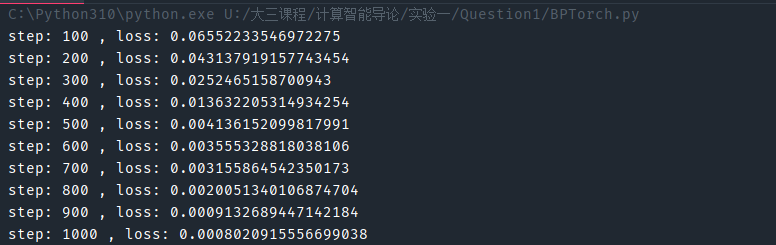
y = y\_scaler.fit\_transform(y)

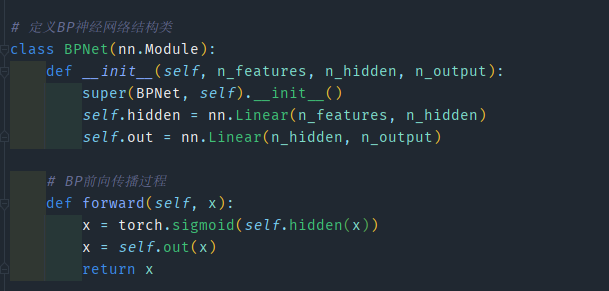
* 第三步：处理数据，样本转置
* 第四步：配置神经网络超参数
* 第五步：训练神经网络
* 第六步：获得神经网络训练后的参数
* 第七步：误差分析比对
* 第八步：曲线拟合结果分析
* 第九步：预测新数据并分析比对

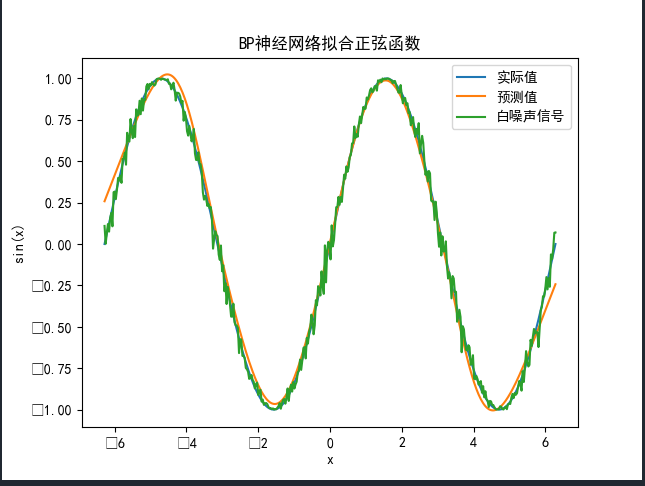
# 五、实验结果与分析

## 题目一

训练过程中迭代次数及其对应的loss值如下图：

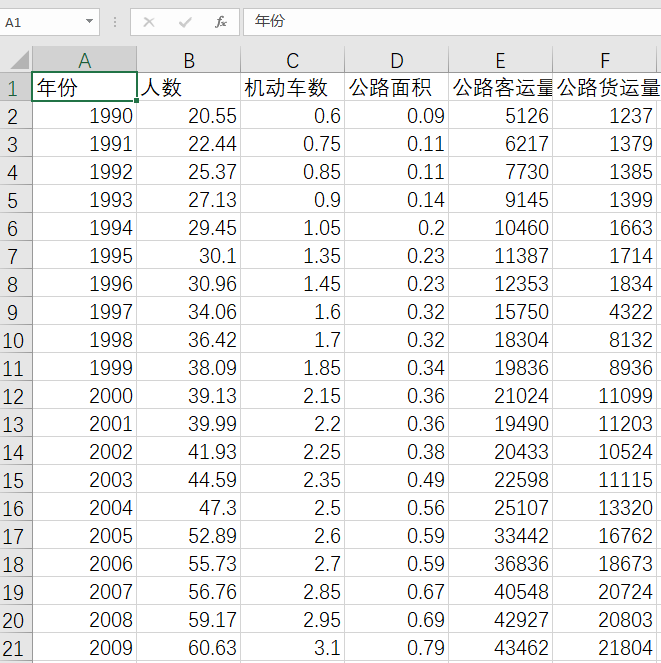




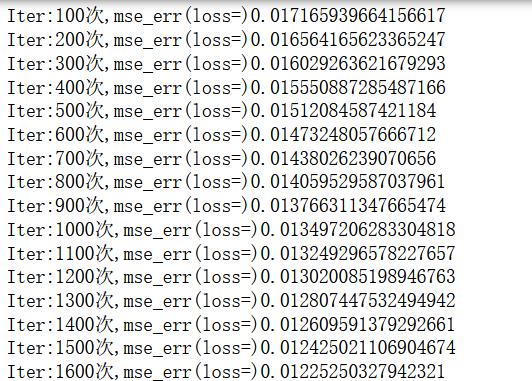


## 题目二

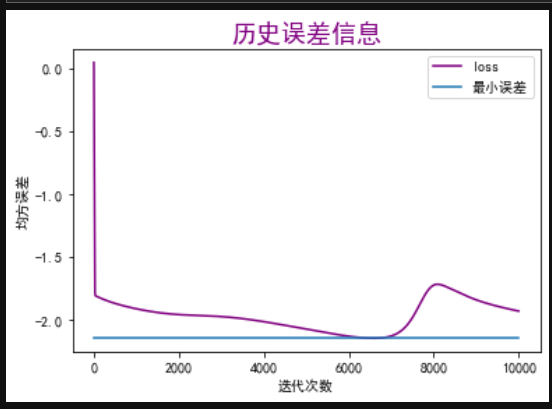
数据集如下：



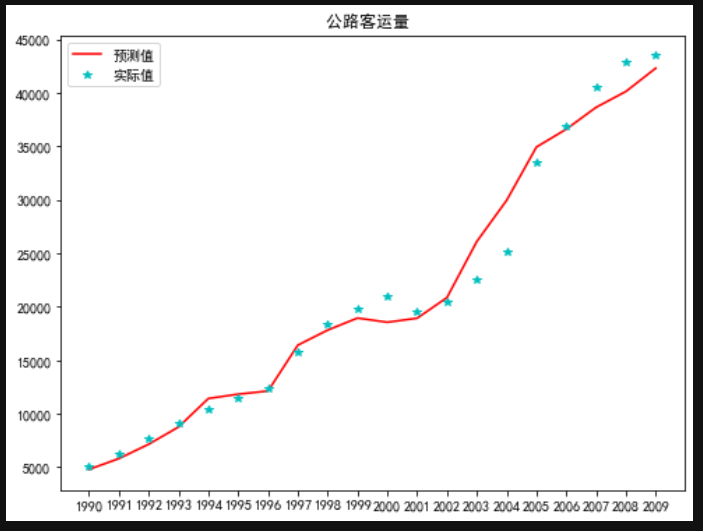
训练过程如下：



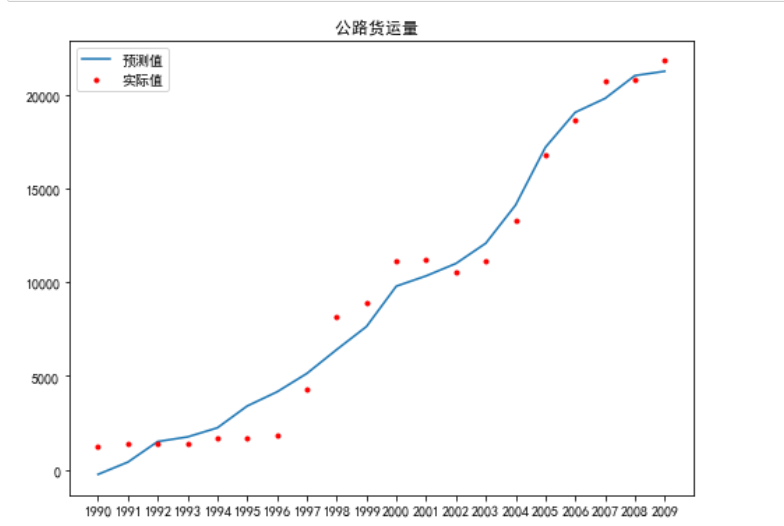
训练误差图：



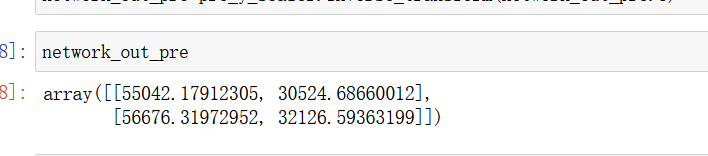
拟合图：



拟合图表二：



预测信息：



# 六、实验程序代码

## 题目一-BPSin.py

import torch.nn as nn

import numpy as np

import torch.nn.functional as Fun

import torch

from torchviz import make\_dot

from torch.utils.data import DataLoader

from torch.utils.data import TensorDataset

import matplotlib.pyplot as plt

# 数据准备

data\_x = np.linspace(-2 \* np.pi, 2 \* np.pi, 400)

data\_y = np.sin(data\_x) # 套用正弦函数

# print(data\_x)

XSet = np.expand\_dims(data\_x, axis=1)

YSet = data\_y.reshape(400, -1)

dataset = TensorDataset(torch.tensor(XSet, dtype=torch.float), torch.tensor(YSet, dtype=torch.float))

dataloader = DataLoader(dataset, batch\_size=100, shuffle=True)

def awgn(x, snr, seed=7):

'''

加入高斯白噪声 Additive White Gaussian Noise

:param x: 原始信号

:param snr: 信噪比

:return: 加入噪声后的信号

'''

np.random.seed(seed) # 设置随机种子

snr = 10 \*\* (snr / 10.0)

xpower = np.sum(x \*\* 2) / len(x)

npower = xpower / snr

noise = np.random.randn(len(x)) \* np.sqrt(npower)

return x + noise

# 定义BP神经网络结构类

class BPNet(nn.Module):

def \_\_init\_\_(self, n\_features, n\_hidden, n\_output):

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

self.hidden = nn.Linear(n\_features, n\_hidden)

self.out = nn.Linear(n\_hidden, n\_output)

# BP前向传播过程

def forward(self, x):

x = torch.sigmoid(self.hidden(x))

x = self.out(x)

return x

# 实例化网络模型

model1 = BPNet(n\_features=1, n\_hidden=100, n\_output=1)

# 配置优化器

optimizer = torch.optim.Adam(model1.parameters(), lr=0.01)

# 配置损失函数

loss\_func = nn.MSELoss()

# 训练网络模型

# 下面开始训练：

# 一共训练 1000次

for epoch in range(1000):

loss = None

for batch\_x, batch\_y in dataloader:

y\_predict = model1(batch\_x)

loss = loss\_func(y\_predict, batch\_y)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

# 每100次 的时候打印一次日志

if (epoch + 1) % 100 == 0:

print("step: {0} , loss: {1}".format(epoch + 1, loss.item()))

# 明确optimizer优化器的作用, 形象地来说，优化器就是需要根据网络反向传播的梯度信息来更新网络的参数，以起到降低loss函数计算值的作用

# 使用训练好的模型进行预测

predict = model1(torch.tensor(XSet, dtype=torch.float))

# 绘图展示预测的和真实数据之间的差异

data\_x\_noise = awgn(data\_x, 35)

data\_y\_noise = np.sin(data\_x\_noise)

plt.rcParams['font.sans-serif'] = [u'SimHei']

plt.plot(data\_x, data\_y, label="实际值")

plt.plot(data\_x, predict.detach().numpy(), label="预测值")

plt.plot(data\_x,data\_y\_noise,label='白噪声信号')

plt.title("BP神经网络拟合正弦函数")

plt.xlabel("x")

plt.ylabel("sin(x)")

plt.legend()

# plt.savefig(fname="result.png",figsize=[10,10])

plt.show()

# x = torch.randn(1, 1, 50, 1).requires\_grad\_(True)

# y = model1(x)

# MyConvnetis = make\_dot(y, params=dict(list(model1.named\_parameters()) + [('x', x)]))

# MyConvnetis.format = "png"

# MyConvnetis.directory = r"C:\2.png"

# MyConvnetis.view()

## 题目二-BPTraffic.py

import numpy as np

import pandas as pd

import matplotlib as mpl

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler # 用于数据归一化

def sigmoid(z):

return 1/(1+np.exp(-z))

def sigmoid\_delta(z): #sigmoid的偏导数

return 1/((1+np.exp(-z))\*\*2)\*np.exp(-z)

#读取数据

data = pd.read\_csv('TrafficData.csv', encoding='utf-8')

x = data[['人数', '机动车数', '公路面积']]

y = data[['公路客运量', '公路货运量']]

# 数据归一化处理

x\_scaler = MinMaxScaler(feature\_range=(-1, 1))

y\_scaler = MinMaxScaler(feature\_range=(-1, 1))

x = x\_scaler.fit\_transform(x)

y = y\_scaler.fit\_transform(y)

# 转置

sample\_in=x.T

sample\_out=y.T

#BP神经网络网络参数

max\_epochs=50000 #最大循环迭代次数

learn\_rate=0.02 #学习率

mse\_final=7.5e-4 #设置一个均方误差（MSE）的阈值，小于它则停止迭代

sample\_number=x.shape[0] #样本规模

input\_number=x.shape[1] #输入规模

output\_number=y.shape[1] #输出规模

hidden\_units=8 #隐含层（Hidden Layer）神经元个数

print(sample\_number,input\_number,output\_number)

#一层隐含层

#W1矩阵:M行N列，M等于该层神经元个数，N等于输入特征个数

W1=0.5\*np.random.rand(hidden\_units,input\_number)-0.1

b1=0.5\*np.random.rand(hidden\_units,1)-0.1

W2=0.5\*np.random.rand(output\_number,hidden\_units)-0.1

b2=0.5\*np.random.rand(output\_number,1)-0.1

mse\_history=[] #空列表，存储迭代的误差

for i in range(max\_epochs):

#训练

hidden\_out=sigmoid(np.dot(W1,sample\_in)+b1) #np.dot矩矩阵相乘,hidden\_out1结果为8行20列

network\_out=np.dot(W2,hidden\_out)+b2 #np.dot矩阵相乘,W2是2行8列，则output结果是2行20列

#误差

err=sample\_out-network\_out

mse\_err=np.average(np.square(err)) #均方误差

mse\_history.append(mse\_err)

if(i+1)% 100 == 0:

print("Iter:{}次,mse\_err(loss=){}".format(i+1,mse\_err))

if mse\_err<mse\_final:

print("Iter:{}次,mse\_err(loss=){}".format(i+1,mse\_err))

print("训练结束")

break

#BP

#误差向量

delta2=-err #最后一层的误差

delta1=np.dot(W2.transpose(),delta2)\*sigmoid\_delta(hidden\_out) #前一层的误差向量,这一层对hidden\_out用了sigmoid激活函数,要对hidden\_out求偏导数；注意最后一步是两个矩阵的点乘，是两个完全相同维度矩阵

#梯度：损失函数的偏导数

delta\_W2=np.dot(delta2,hidden\_out.transpose())

delta\_W1=np.dot(delta1,sample\_in.transpose())

delta\_b2=np.dot(delta2,np.ones((sample\_number,1)))

delta\_b1=np.dot(delta1,np.ones((sample\_number,1)))

W2-=learn\_rate\*delta\_W2

b2-=learn\_rate\*delta\_b2

W1-=learn\_rate\*delta\_W1

b1-=learn\_rate\*delta\_b1

print("w1={},b1={},w2={},b2={}".format(W1,b1,W2,b2))

#损失值画图

print(mse\_history)

loss=np.log10(mse\_history)

min\_mse=min(loss)

plt.plot(loss,label='loss')

plt.plot([0,len(loss)],[min\_mse,min\_mse],label='min\_mse')

plt.xlabel('迭代次数')

plt.ylabel('均方误差')

plt.title('历史误差信息',fontdict={'fontsize':18,'color':'red'})

plt.legend()

plt.show()

#模型预测输出和实际输出对比图

hidden\_out=sigmoid(np.dot(W1,sample\_in)+b1)

network\_out=np.dot(W2,hidden\_out)+b2

#反转获取实际值：

network\_out=y\_scaler.inverse\_transform(network\_out.T)

sample\_out=y\_scaler.inverse\_transform(y)

#解决图片中文无法显示

plt.rcParams['font.sans-serif'] = [u'SimHei']

plt.rcParams['axes.unicode\_minus'] = False

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

plt.plot(network\_out[:,0],label='预测值')

plt.plot(sample\_out[:,0],'r.',label='实际值')

x\_labels = range(1990,2010)

x\_ticks=range(0,20)

plt.xticks(ticks=x\_ticks, labels=x\_labels)

plt.title('公路客运量 ')

plt.legend()

plt.show()

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

plt.plot(network\_out[:,1],label='预测值')

plt.plot(sample\_out[:,1],'r.',label='实际值')

plt.title('公路货运量 ')

x\_labels = range(1990,2010)

x\_ticks=range(0,20)

plt.xticks(ticks=x\_ticks, labels=x\_labels)

plt.legend()

plt.show()

# 现在开始预测环节

PredictDataFrame = pd.read\_csv('DataUnknown.csv',encoding='utf-8')

Pre\_x=PredictDataFrame[['人数', '机动车数', '公路面积']]

Pre\_y=PredictDataFrame[['公路客运量', '公路货运量']]

pre\_x\_scaler = MinMaxScaler(feature\_range=(-1,1))

pre\_y\_scaler=MinMaxScaler(feature\_range=(-1,1))

pre\_x=pre\_x\_scaler.fit\_transform(Pre\_x)

pre\_y=pre\_y\_scaler.fit\_transform(Pre\_y)

#对样本进行转置，矩阵运算

pre\_in=pre\_x.T

pre\_out=pre\_y.T

#模型预测输出值

hidden\_out\_pre = sigmoid(np.dot(W1,pre\_in)+b1)

network\_out\_pre=np.dot(W2,hidden\_out\_pre)+b2

#翻转获取实际值

network\_out\_pre=pre\_y\_scaler.inverse\_transform(network\_out\_pre.T)