# 机器学习实验报告

## 神经网络

学院：航天学院

姓名： 石瑞河

学号：1180400510

**实验1：编程实现神经网络回归**

数据集：

文件名：NR-Train-20.csv，20个训练数据；NR-Test-20.csv，20个测试数据

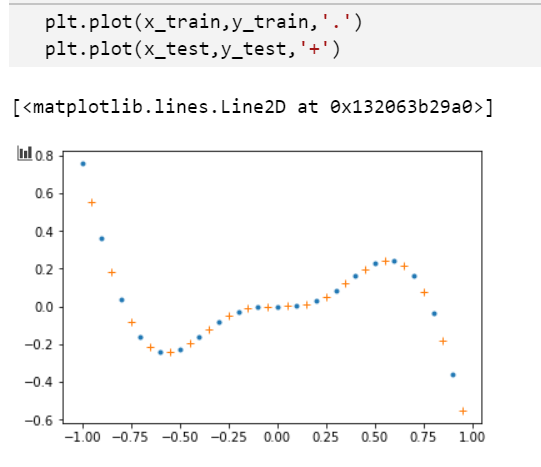
数据格式：第1列为输入，第2列为输出

数据生成：

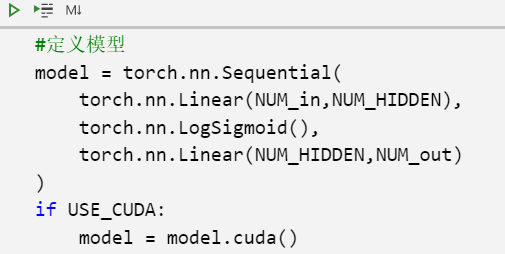
要求：

1. 编程实现神经网络回归；
2. 网络结构
   1. 输入层：1个神经元
   2. 隐含层：5个神经元，激活函数使用对数型或双曲正切型Sigmoid函数
   3. 输出层：1个神经元，激活函数使用线性函数
3. 用训练数据训练网络，并计算测试数据的输出；
4. 显示训练数据点和测试数据点；

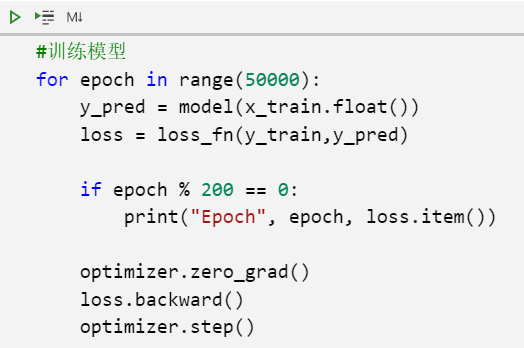
训练集数据点和测试集数据点

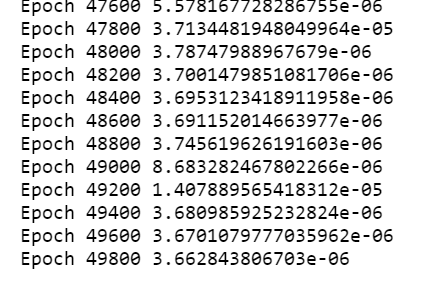


模型

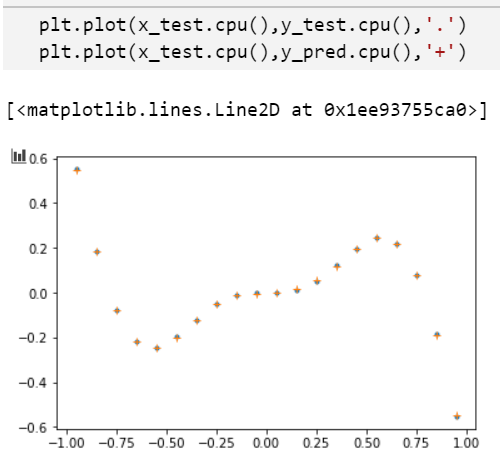


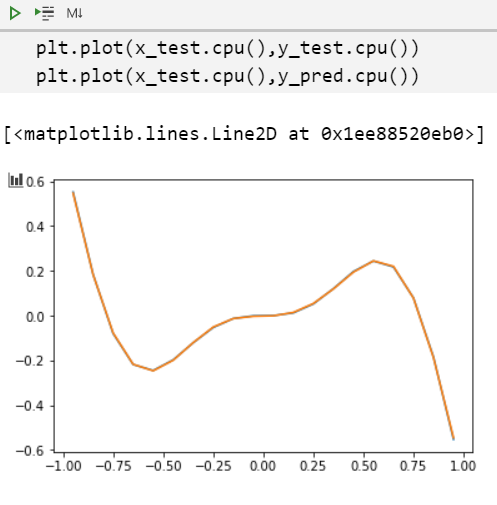
模型训练及损失





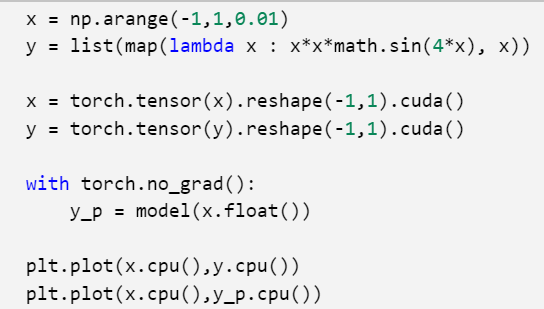
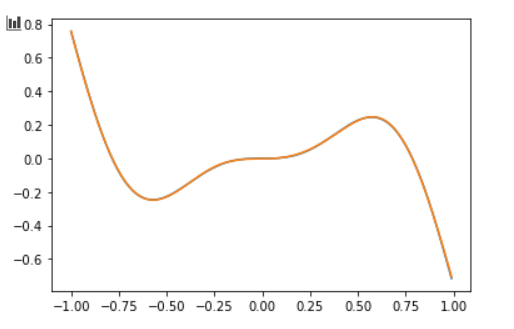
测试集上效果

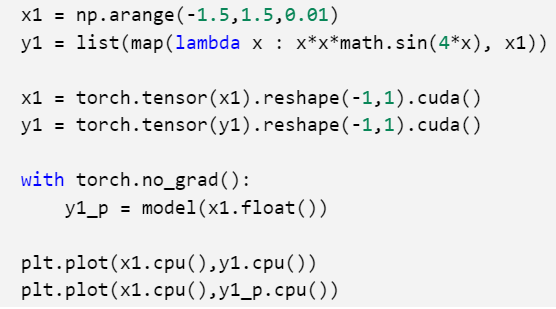
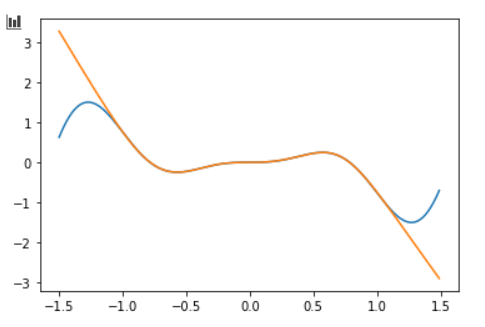




可以发现，训练出的模型在测试集上效果非常好

进一步，我们把训练模型与数据生成函数进行比较





在数据生成范围，即-1<x<1区间内，训练模型与数据生成函数高度接近

在|x|>1后，二者的区别迅速增加，分道扬镳

**实验2：编程实现神经网络分类**

训练数据：NC-Train-Data.csv，NC-Train-Label.csv，包含200个训练数据

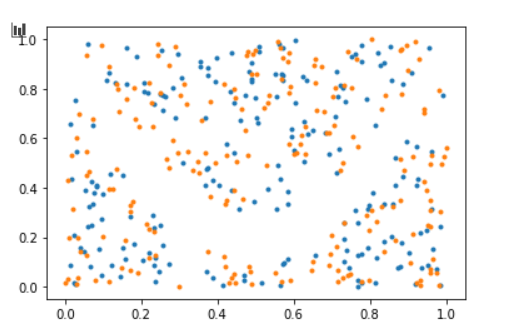
测试数据：NC-Test-Data.csv，NC-Test-Label.csv，包含200个训练数据

数据内容：二次函数可分的2维数据

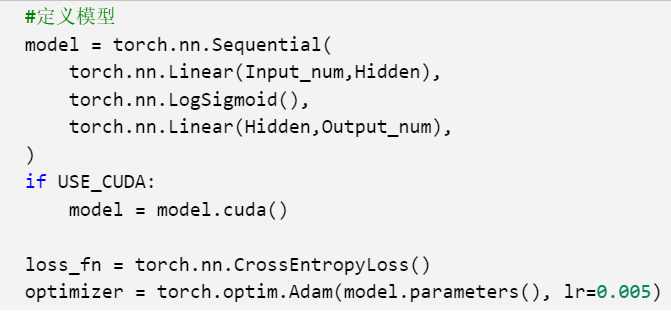
要求：

1. 编程实现神经网络分类器；
2. 网络结构：
   1. 输入层：2个神经元
   2. 隐含层：5个神经元，激活函数使用对数型或双曲正切型Sigmoid函数
   3. 输出层：1个神经元，激活函数使用Softmax函数
3. 以NC-Train-Data.csv和NC-Train-Label.csv为训练数据，训练神经网络；
4. 分类测试数据NC-Test-Data.csv，与NC-Test-Label.csv对比，显示分类的正确率；
5. 尝试不同的训练参数，观察对分类器学习结果的影响；

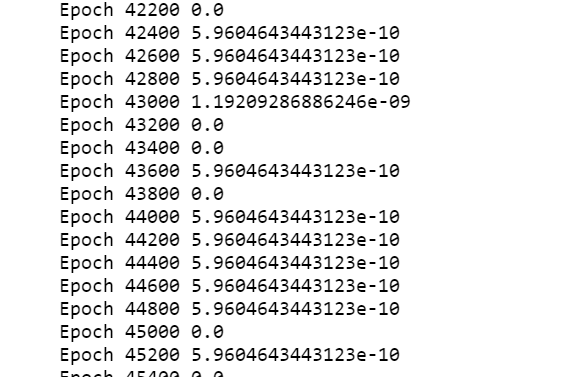
训练集和测试集数据点

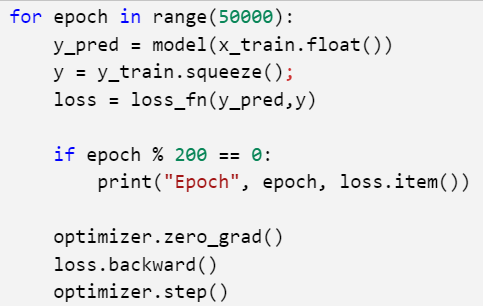


分类模型

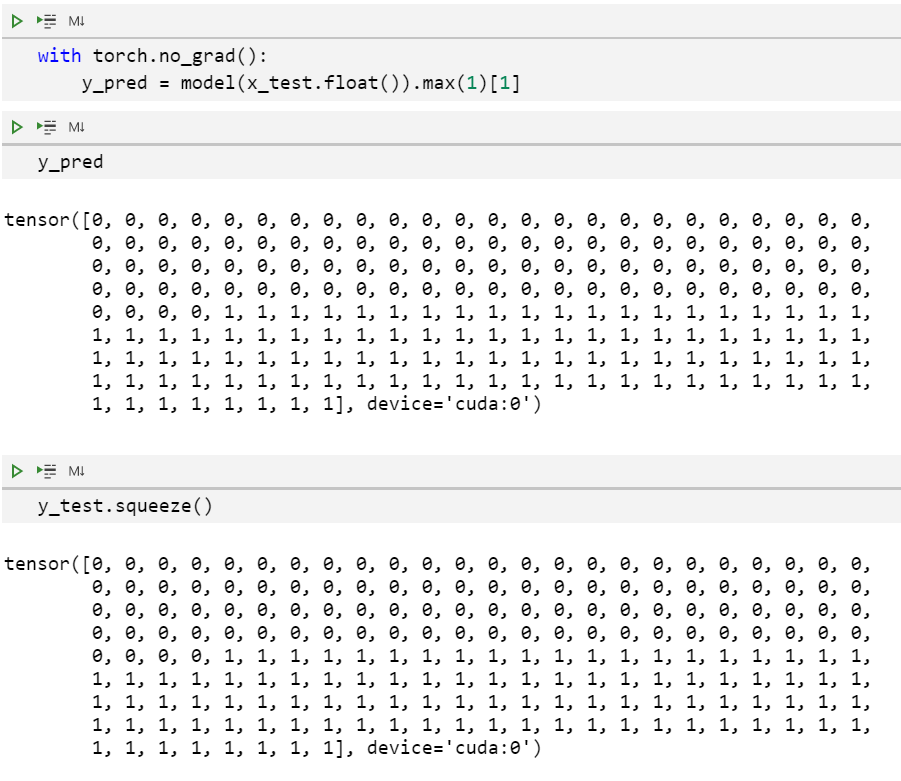


学习率0.005

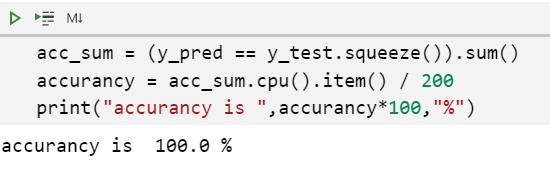
模型训练及训练结果



测试集上效果



测试集上准确率



分类正确率 100

**实验3：编程实现卷积神经网络分类（选做）**

训练数据：TrainSamples.csv，TrainLables.csv，包含30000个训练数据

测试数据：TestSamples.csv，TestLabels.csv，包含10000个测试数据；

数据内容：手写数字图像，图像大小，10个类别(0~9)

数据格式：

1. TrainSamples.csv和TestSamples.csv中，每行一个训练数据或测试数据的图像，784维属性，可以恢复成的图像；
2. TrainLabels.csv和TestLabels.csv中包含训练数据和测试数据的类别标记，每行对应一个训练数据的类别；

要求：

1. 编程实现卷积神经网络分类器；
2. 网络结构：使用讲义上“简化的LeNet”
3. 以TrainSamples.cvs和TrainLabels.csv为训练数据，训练卷积神经网络；
4. 分类测试数据TestSamples.csv，与TestLabels.csv对比，显示分类的正确率；
5. 尝试不同的训练参数，观察对分类器学习结果的影响；

**附注：**

实验环境：

Python3.8.5

VSCode

Jupyter notebook

Pytorch 1.6.0

Numpy pandas 等常用python包

代码附件单

实验1：LinearReg.ipynb Use\_LinearReg.ipynb （LinearReg.pth）

实验2：Classify.ipynb Use\_ Classify.ipynb （Classify.pth）