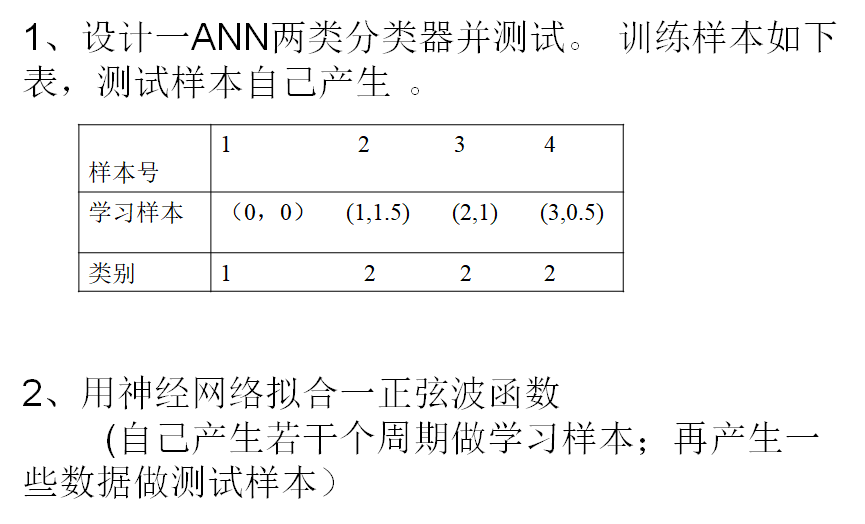
信号处理的智能化方法及应用 第六次作业

1. **分类器设计**



1. 实现环境：PyTorch Version 1.5 on Python 3.7
2. 实现方法：设计了一个神经网络，网络具有两层结构，一层为输入与输出维度一样的全连接网络，使用tanh函数作为激活函数，另一层也为全连接网络，使用softmax函数作为激活函数。网络输出为两个0-1变量，若前者更大，则认为类别为1，否则类别为0.
3. 程序代码

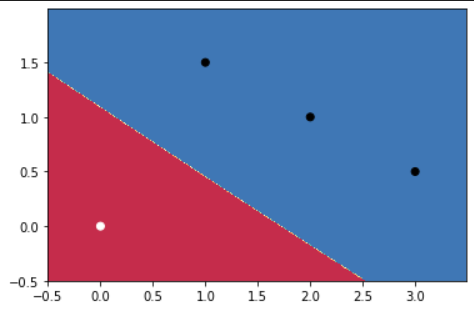
神经网络的 模型定义如下：

1. **class** MyNet(nn.Module):
2. **def** \_\_init\_\_(self, in\_dim, out\_dim):
3. super(MyNet, self).\_\_init\_\_()
4. self.fc1 = nn.Linear(in\_dim, in\_dim)
5. self.output = nn.Linear(in\_dim, out\_dim)
7. **def** forward(self, x):
8. x = self.fc1(x)
9. x = torch.tanh(x)
10. x = self.output(x)
11. **return** x
13. **def** predict(self, x):
14. pred = functional.softmax(self.forward(x), dim = 0)
15. ans = []
16. **for** t **in** pred:
17. **if** t[0]>t[1]:
18. ans.append(0)
19. **else**:
20. ans.append(1)
21. **return** torch.tensor(ans)
22. 运行结果

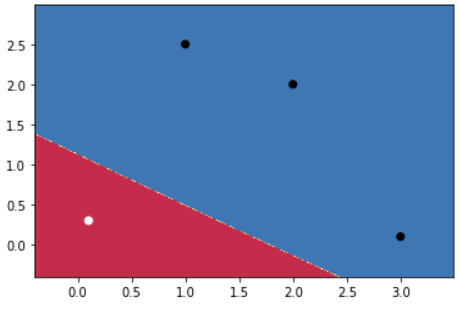
程序选用交叉熵作为损失函数，使用Adam优化器进行优化，学习率0.01。

训练了10000个epoch，最终误差低于10^-5。

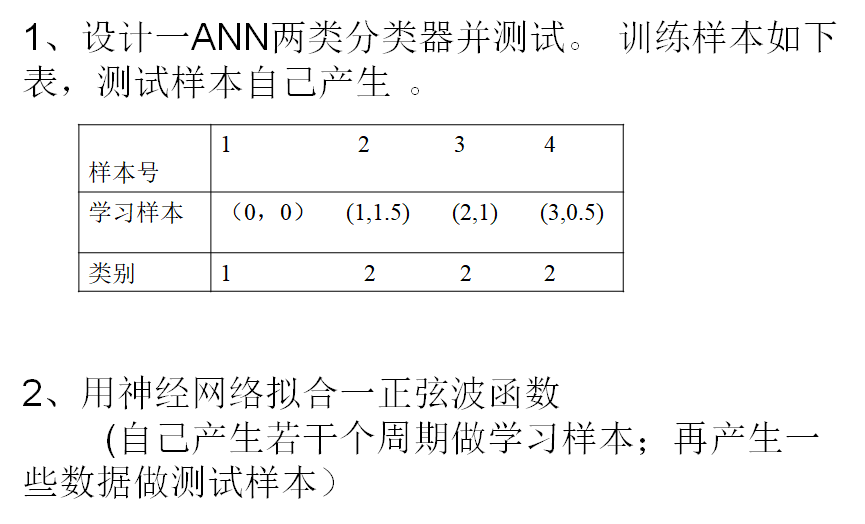
使用原先输入进行判断，结果如下：



使用随意挑选的几个新的点进行分类，分类器工作正常：



1. **函数拟合**



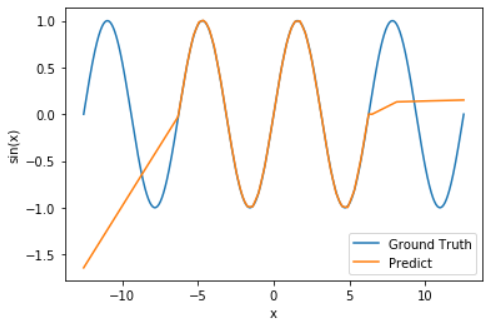
1. 实现环境：PyTorch Version 1.5 on Python 3.7
2. 实现方法：设计了一个神经网络，网络具有四层结构，输入输出维度分别是[1,8],[8,64],[64,8],[8,1]，除了输出层，其他层均使用ReLU作为激活函数
3. 程序代码

神经网络的 模型定义如下：

1. **class** MyNet(nn.Module):
2. **def** \_\_init\_\_(self, dim):
3. super(MyNet, self).\_\_init\_\_()
4. self.net = nn.Sequential(
5. nn.Linear(dim, 8 \* dim), nn.ReLU(),
6. nn.Linear(8 \* dim, 64 \* dim), nn.ReLU(),
7. nn.Linear(64 \* dim, 8 \* dim), nn.ReLU(),
8. nn.Linear(8 \* dim, dim)
9. )
11. **def** forward(self, x): # x:torch.FloatTensor
12. **return** self.net(x)
13. 运行结果

程序选用交叉熵作为损失函数，使用Adam优化器进行优化，学习率为0.01。

训练了10000个epoch，最终误差低于10^-4。



可以很明显的看出，在原有的训练集范围内，网络的预测性能很好，但是外推泛化时，网络预测的准确度急速下降，基本无法工作。

1. 改进

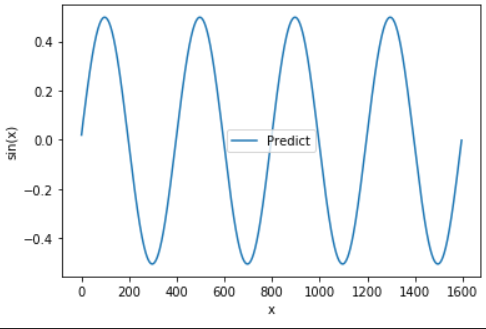
通过多次网络层数或维度的调整，网络外推泛化性能始终不理想，因此引入LSTM层加以改善：

1. **class** MyNet(nn.Module):
2. **def** \_\_init\_\_(self, input\_size, hidden\_size, output\_size=1, lstm\_layers=2):
3. super(MyNet, self).\_\_init\_\_()
4. self.rnn = nn.LSTM(input\_size, hidden\_size, lstm\_layers)
5. self.out = nn.Linear(hidden\_size, output\_size)
7. **def** forward(self, x): # x:torch.FloatTensor
8. x, \_ = self.rnn(x)
9. s, b, h = x.shape
10. x = x.view(s \* b, h)
11. x = self.out(x)
12. x = x.view(s, b, -1)
13. **return** x

网络中隐层为一个LSTM层，输出层为线性全连接层，LSTM输入维度为3，输出为10，同时使用3组值进行学习。

对测试集的预测效果如下：

（此题的两个测试集均为[-2Pi,2Pi]，而训练集为[-Pi,Pi]，由于LSTM需要对输入的变量进行特殊处理，而在画图时忘记了还原x变量的值，因此图像的x轴显示坐标与实际不符合，但实际上是正确的坐标）



1. **小结**

此次作业尝试不使用MATLAB完成，选用了PyTorch框架，本来的目的是使用GPU加速训练，不过由于网络很简单，事实上并没有起到帮助，同时GPU训练需要对数据集进行额外的处理，在这里我遇到了一些困难导致训练出现问题，由于最近有考试，时间比较紧张，就没有继续下去。

虽然使用老师提供的MATLAB代码训练无论是精简代码量还是训练结果都比Python更胜一筹，但是网络的总体训练结果还是较令人满意的，同时也加深了对于神经网络和相关框架的认识与掌握。

完整代码源文件请查看：

<https://git.nju.edu.cn/Minaduki/Intelligent_Signal_Processing>