

计算机视觉

课程实验报告

|  |  |
| --- | --- |
| 院系 | 计算机科学与技术 |
| 专业班级 | 本硕博2101 |
| 姓名 | 姚晨炫 |
| 学号 | U202115674 |
| 指导老师 | 杨卫 |

目录

[1 网络架构 3](#_Toc152921240)

[2 数据集 4](#_Toc152921241)

[3 训练 5](#_Toc152921242)

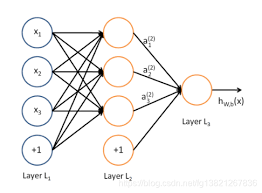
[4 实验结果 6](#_Toc152921243)

# 网络架构

我使用的网络是一个多层感知机MLP，有两个线性层，两个RELU激活函数和一个Sigmoid层，具体参数和代码如下

class MLP(nn.Module):  
 def \_\_init\_\_(self):  
 super(MLP, self).\_\_init\_\_()  
 self.fc1 = nn.Linear(2, 64)  
 self.relu1 = nn.ReLU()  
 self.fc2 = nn.Linear(64, 32)  
 self.relu2 = nn.ReLU()  
 self.fc3 = nn.Linear(32, 4)  
 self.sigmoid = nn.Sigmoid()  
  
 def forward(self, x):  
 out = self.fc1(x)  
 out = self.relu1(out)  
 out = self.fc2(out)  
 out = self.relu2(out)  
 out = self.fc3(out)  
 out = self.sigmoid(out)  
 return out

结构图如下



损失函数采用交叉熵损失函数，优化器采用SGD ，其中学习率为0.01，momentum为0.9

# 定义损失函数和优化器  
criterion = nn.CrossEntropyLoss()  
optimizer = torch.optim.SGD(model.parameters(),momentum=0.9, lr=0.001)

# 数据集

调用from torch.utils.data import TensorDataset, DataLoader 将CSV文件中的参数制作为数据集，使用one\_hot独热编码将不同的种类分开。然后调用from sklearn.model\_selection import train\_test\_split 进行数据集分割。对应代码和注释如下

# 加载数据集  
data = pd.read\_csv('dataset.csv')  
  
# 随机排序数据集  
data = shuffle(data)  
  
# 划分特征和标签  
X = data[['data1', 'data2']].values  
y = data['label'].values  
y = y-1  
  
#数据集分割  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.1)  
X\_train = torch.tensor(X\_train,dtype=torch.float32)  
X\_test = torch.tensor(X\_test,dtype=torch.float32)  
y\_train = torch.tensor(y\_train,dtype=torch.long)  
y\_test = torch.tensor(y\_test,dtype=torch.long)  
  
y\_train = F.one\_hot(y\_train)  
y\_test = F.one\_hot(y\_test)  
\_, y\_test\_labels = torch.max(y\_test, 1)  
  
# 创建TensorDataset和DataLoader用于批量训练  
train\_dataset = TensorDataset(X\_train, y\_train)  
train\_loader = DataLoader(train\_dataset, batch\_size=32, shuffle=True)  
test\_dataset = TensorDataset(X\_test, y\_test)  
test\_loader = DataLoader(test\_dataset, batch\_size=32, shuffle=True)

# 训练

采用标准的模型训练方式，Epoch设置为150.

num\_epochs = 150  
for epoch in range(num\_epochs):  
 for inputs, targets in train\_loader:  
 # Forward pass  
 outputs = model(inputs)  
 loss = criterion(outputs, targets.float())  
   
 # Backward pass and optimization  
 optimizer.zero\_grad()  
 loss.backward()  
 optimizer.step()

# 实验结果

实验环境为MacOS Apple Silicon M1 Pro，Python版本3.9，Pytorch版本2.0.0(MacOS版)

模型在120轮左右收敛，损失在0.83左右波动，测试集上的准确率为0.92，具体的训练Loss太长，在notebook中有完整呈现。

文本

描述已自动生成