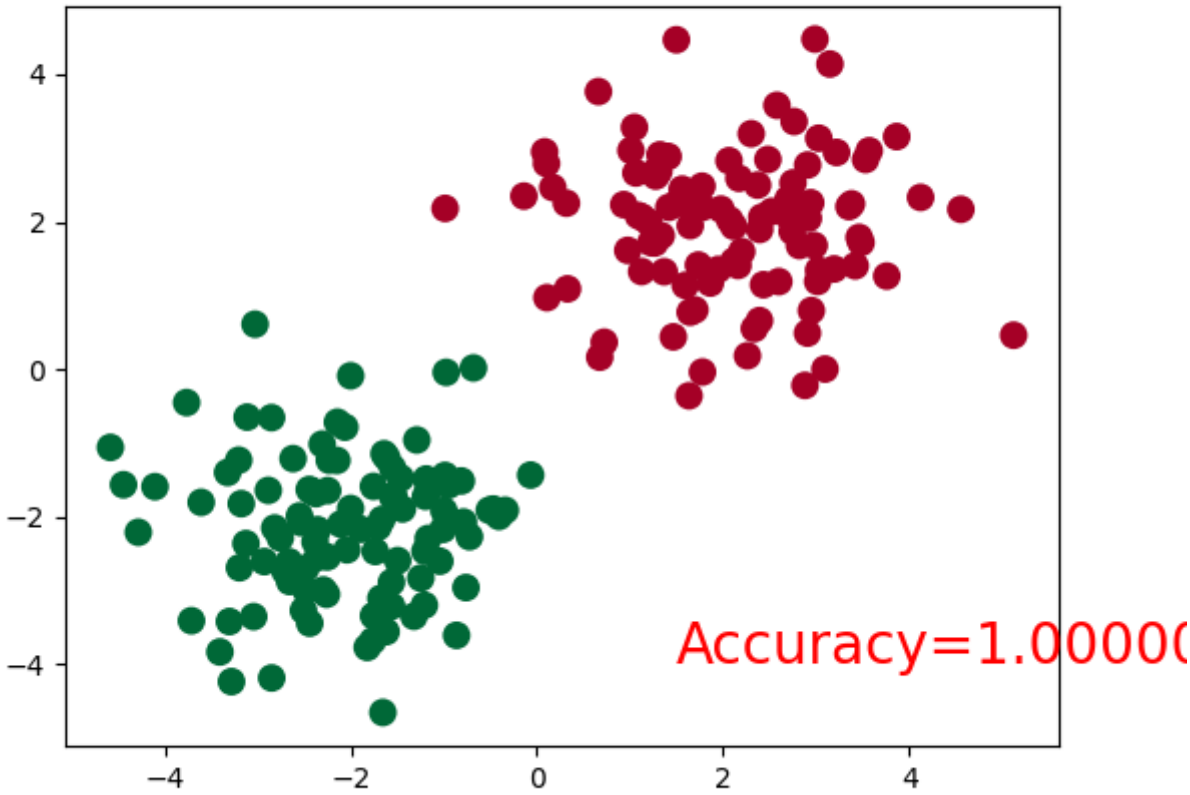
三、数据分类



姓名：游霄童

学号：21009200158

created：2024/04/28

---

1. 实验目的

本实验的主要目的是应用神经网络技术解决实际数据分类问题。通过构建和训练神经网络，我们将探索如何利用计算机视觉技术自动识别并分类具有随机性的散点数据。这项技术在机器学习领域具有广泛的应用，包括图像识别、自然语言处理等。

2. 实验数据

实验数据是通过Python的PyTorch库生成的。本次实验模拟了具有一定随机性的散点分布数据，这些数据被划分为两个类别，每个类别的数据都遵循不同的正态分布规律，模拟现实世界中分类任务的复杂性。

3. 实验环境

- **硬件环境：** Nvidia Jetson TX1 嵌入式系统，该系统配备了高性能的GPU，支持高速计算和实时数据处理，非常适合进行深度学习实验。
- **软件环境：** Python 3.8, PyTorch 1.8, Matplotlib，这些工具为数据处理和可视化提供了强大的支持。

4. 实验过程

4.1 数据准备

使用PyTorch生成两类不同的散点数据集：

```
#构建样本集
data = torch.ones(100, 2)
x0 = torch.normal(2*data, 1)
x1 = torch.normal(-2*data, 1)
y0 = torch.zeros(100)
y1 = torch.ones(100)
x = torch.cat((x0, x1), 0).type(torch.FloatTensor)
y = torch.cat((y0, y1), 0).type(torch.FloatTensor)
```

为每类数据标记标签，并合并

4.2 网络构建与训练

构建一个包含输入层、一个隐含层和输出层的简单前馈神经网络：

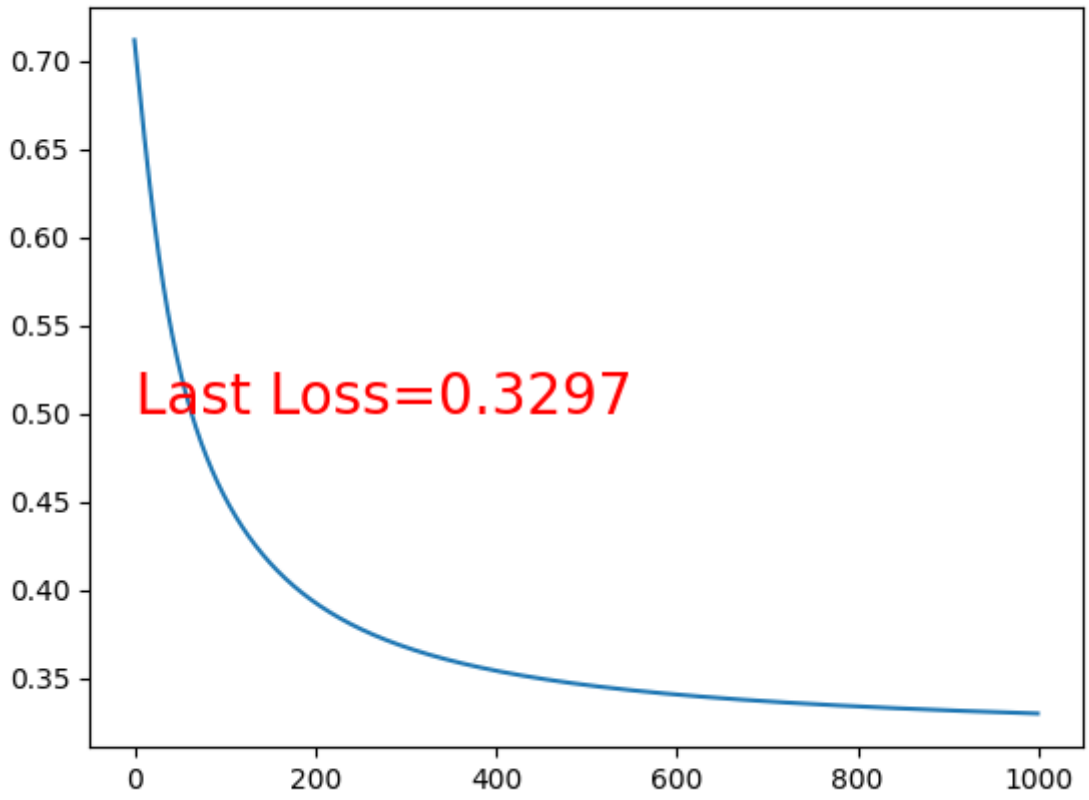
```
#构建网络
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()




        self.classify = nn.Sequential(
            nn.Linear(2, 15),
            nn.ReLU(),
            nn.Linear(15, 2),
            nn.Softmax(dim=1)
        )
    def forward(self, x):
        return self.classify(x)
```

进行网络训练：

```
#训练网络 使用SGD优化器 交叉损失函数 绘制损失函数曲线显示最终数值并保存图片
net = Net()
optimizer = optim.SGD(net.parameters(), lr=0.01)
loss_func = nn.CrossEntropyLoss()
losses = []
for epoch in range(1000):
    out = net(x)
    loss = loss_func(out, y.long())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    losses.append(loss.item())
plt.plot(losses)
plt.text(0, 0.5, 'Last Loss=%.4f' % loss.item(), fontdict={'size': 20,
'color': 'red'})
plt.savefig('loss.png')
plt.show()
```



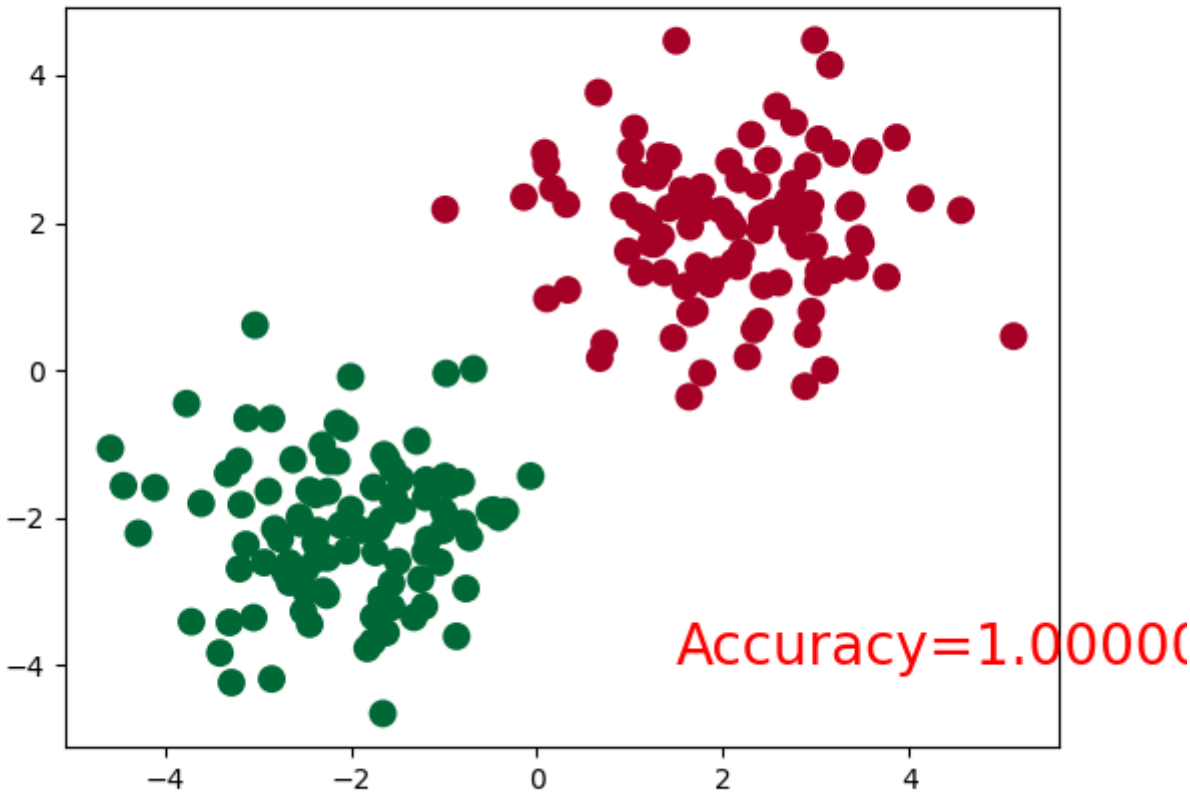
4.3 结果可视化与评估

绘制分类结果，并计算准确率：

```
#绘制分类结果并保存图片
classification = torch.max(out, 1)[1]
class_y = classification.data.numpy()
target_y = y.data.numpy()
# 绘制散点图 并显示准确率
plt.scatter(x.data.numpy()[:, 0], x.data.numpy()[:, 1], c=class_y, s=100,
lw=0, cmap='RdYlGn')
accuracy = sum(class_y == target_y)/200
plt.text(1.5, -4, 'Accuracy=%.5f' % accuracy, fontdict={'size': 20,
'color': 'red'})
plt.savefig('classification.png')
plt.show()
```

5. 实验结果

实验结果显示，所构建的神经网络模型能够有效地区分两类散点数据。



6. 实验心得

通过本次实验，我深刻理解了神经网络在数据分类问题中的应用。实验中涉及的数据预处理、模型构建、参数调优和结果评估等步骤，都大大增强了我的实践能力和理论知识。此外，实验也让我意识到，模型的性能强烈依赖于网络结构和超参数的选择，这需要在未来的学习中继续深入探索。

7.代码附录

```
#%%
#导入必要的库
import torch
import torch.nn as nn
import torch.optim as optim
import matplotlib.pyplot as plt
#%%
#构建样本集
data = torch.ones(100, 2)
x0 = torch.normal(2*data, 1)
x1 = torch.normal(-2*data, 1)
y0 = torch.zeros(100)
y1 = torch.ones(100)
x = torch.cat((x0, x1), 0).type(torch.FloatTensor)
y = torch.cat((y0, y1), 0).type(torch.FloatTensor)
#%%
#构建网络
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()




        self.classify = nn.Sequential(
            nn.Linear(2, 15),
            nn.ReLU(),
            nn.Linear(15, 2),
            nn.Softmax(dim=1)
        )
    def forward(self, x):
        return self.classify(x)
#%%
#训练网络 使用SGD优化器 交叉损失函数 绘制损失函数曲线显示最终数值并保存图片
net = Net()
optimizer = optim.SGD(net.parameters(), lr=0.01)
loss_func = nn.CrossEntropyLoss()
losses = []
for epoch in range(1000):
    out = net(x)
    loss = loss_func(out, y.long())
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
    losses.append(loss.item())
plt.plot(losses)
```

```
plt.text(0, 0.5, 'Last Loss=%.4f' % loss.item(), fontdict={'size': 20,
'color': 'red'})
plt.savefig('loss.png')
plt.show()

#%%
#绘制分类结果并保存图片
classification = torch.max(out, 1)[1]
class_y = classification.data.numpy()
target_y = y.data.numpy()
# 绘制散点图 并显示准确率
plt.scatter(x.data.numpy()[:, 0], x.data.numpy()[:, 1], c=class_y, s=100,
lw=0, cmap='RdYlGn')
accuracy = sum(class_y == target_y)/200
plt.text(1.5, -4, 'Accuracy=%.5f' % accuracy, fontdict={'size': 20,
'color': 'red'})
plt.savefig('classification.png')
plt.show()
#%%
```