lab1

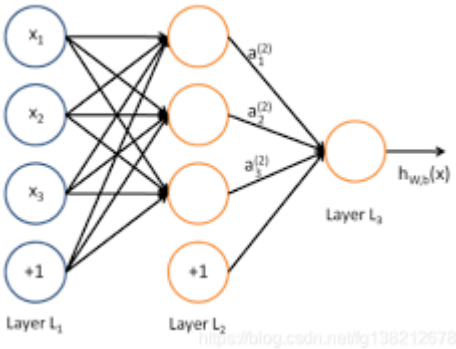
网络架构

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。

```
112   Epoch [112/150], Train_Loss: 0.8176, Test_Loss: 0.8377, Test Accuracy: 0.91
113   Epoch [113/150], Train_Loss: 0.8017, Test_Loss: 0.8373, Test Accuracy: 0.91
114   Epoch [114/150], Train_Loss: 0.8336, Test_Loss: 0.8371, Test Accuracy: 0.91
115   Epoch [115/150], Train_Loss: 0.8367, Test_Loss: 0.8367, Test Accuracy: 0.91
116   Epoch [116/150], Train_Loss: 0.8385, Test_Loss: 0.8362, Test Accuracy: 0.92
117   Epoch [117/150], Train_Loss: 0.8639, Test_Loss: 0.8360, Test Accuracy: 0.91
118   Epoch [118/150], Train_Loss: 0.8182, Test_Loss: 0.8357, Test Accuracy: 0.91
119   Epoch [119/150], Train_Loss: 0.8831, Test_Loss: 0.8354, Test Accuracy: 0.91
120   Epoch [120/150], Train_Loss: 0.8457, Test_Loss: 0.8354, Test Accuracy: 0.91
121   Epoch [121/150], Train_Loss: 0.8637, Test_Loss: 0.8350, Test Accuracy: 0.92
122   Epoch [122/150], Train_Loss: 0.8779, Test_Loss: 0.8348, Test Accuracy: 0.92
123   Epoch [123/150], Train_Loss: 0.7873, Test_Loss: 0.8345, Test Accuracy: 0.92
124   Epoch [124/150], Train_Loss: 0.8234, Test_Loss: 0.8342, Test Accuracy: 0.92
125   Epoch [125/150], Train_Loss: 0.9104, Test_Loss: 0.8340, Test Accuracy: 0.92
126   Epoch [126/150], Train_Loss: 0.7765, Test_Loss: 0.8337, Test Accuracy: 0.92
```