华中科技大学

2023

计算机视觉实验　课程设计报告

专　　业：	计算机科学与技术
班　　级：	计卓 2101 班
学　　号：	U202112071
姓　　名：	王彬
邮　　件：	wangbin2002@hust.edu.cn
完成日期：	2023.12.20

目　录

实验一：基于前馈神经网络的分类任务设计

1.1 任务描述

本实验需要设计一个前馈神经网络，对一组数据实现分类任务。

下载“dataset.csv”数据集，其中包含四类二维高斯数据和它们的标签。设计至少含有一层隐藏层的前馈神经网络来预测二维高斯样本$(data_1, data_2)$所属的分类$label$。这个数据集需要先进行随机排序，然后选取 90%用于训练，剩下的 10%用于测试。

1.2 实验内容

(一) 数据集获取与重排序

我们在获得 dataset.csv 后，将其导入模型中。这里我们使用 data.sample().reset_index()进行重排序，并调用 sklearn.model_selection.train_test_split()方法对数据的训练集与测试集进行划分。为了方便训练，我们将训练数据和标签封装为 DataLoader，这样对于每个 batch 都可以便于读取。我们的代码如下：

```
from torch.utils.data import Dataset, DataLoader
# 定义数据集类
class MyDataset(Dataset):
    def __init__(self, data, labels):
        self.data = data
        self.labels = labels
    def __len__(self):
        return len(self.labels)
    def __getitem__(self, idx):
        # print(self.data[idx])
        return self.data[idx], self.labels[idx]
```

```
if __name__=="__main__":
    # Load data
    data = pd.read_csv('dataset.csv')
    # shuffle
    data = data.sample(frac=1).reset_index(drop=True)
    # 将特征和标签分离
    X = data[['data1', 'data2']].values
    y = data['label'].values
    label_encoder = LabelEncoder()
    y = label_encoder.fit_transform(y)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=42)
    train_dataset = MyDataset(X_train, y_train)
    test_dataset = MyDataset(X_test, y_test)
    train_loader = DataLoader(train_dataset, batch_size=batch_size,
shuffle=True)
    test_loader = DataLoader(test_dataset, batch_size=batch_size,
shuffle=False)
```

(二) 构建神经网络模型

我们设计前馈神经网络模型。我们分别运行了多种激活函数的（Softmax、ReLU、Tanh 及 Softplus）和多种网络架构（变化网络层数及每层的神经元个数的）网络，而且我们分别尝试了 PyTorch 版本和 sklearn 版本的模型运行效果。这里我们仅介绍 PyTorch 的代码。

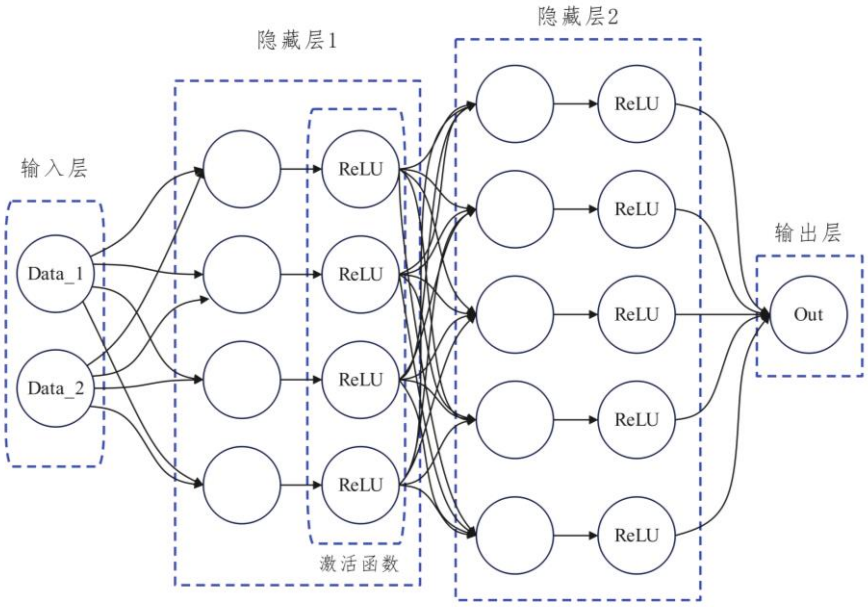


图 1 前馈神经网络

我们的模型定义为：

```
# 定义模型
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size, output_size)
        self.softmax = nn.Softmax(dim=1)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        out = self.softmax(out)
        return out
```

需要注意的是，对于不同的神经网络结构，我们会修改这里的 Net 模型。例如，

如果具有 3 层隐藏层，我们会增加 self.fc3=nn.Linear(hidden_size_2,output_size)。

(三) 训练与结果绘制

由于篇幅限制，训练代码详见附录中的代码文件（./train_pytorch.py），我们同时编写了 sklearn 版本的代码（./train.py），效果较好，在测试集上可达到 95%准确率，对于我们自己编写的 PyTorch 方法的准确率为 94.25%。

这里需要注意的是，我们在训练一定轮数（既一个 batch 后）就会对测试集的所有样本进行测试，并将测试集的结果写入输出列表。因此我们下文中所有准确率，均为实际测试样本的准确率。

对于图形绘制，我们调用 matplotlib.pyplot 画出 Loss 和 Acc 曲线。这里，绘图函数需要传入文件地址参数，以便对多个曲线统一绘制，并进行对比分析。

1.3 实验结果及其分析

(一) 网络结构对于训练结果的影响

我们首先增加网络层数。对于每 100 个 epoch 后，我们均计算一次测试集正确率，并将其统计为列表。

对于 10 层神经网络（9 隐藏层+1 输出层）和 3 层神经网络（2 隐藏层+1 输出层），我们分别测试了其损失值。对于深层神经网络，由于其待训练参数较多，在训练初期，其损失值下降不明显；较深层的神经元尚未学习到较多特征。之后会产生突降，并且训练效果保持相对由于浅层前馈网络。

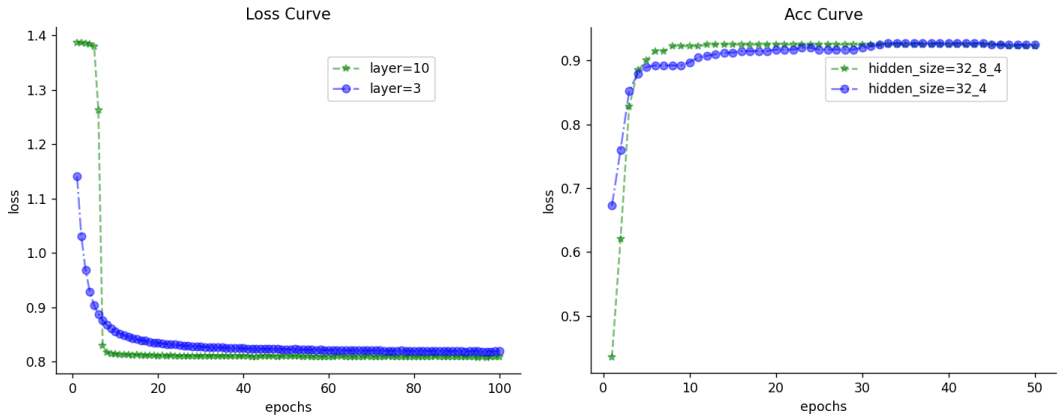


图 2 不同网络结构下的 Loss 及准确率变化

(二) 神经元个数对于训练结果的影响

首先，我们在三层前馈神经网络中修改隐藏层 1 的神经元个数。我们测试得到的层神经元个数分别是 128/8/4 的神经网络效果最好，它的准确率达到了 94.25%，而如果将第一层的神经元进一步增加，效果则产生了一定下滑。这是因为过多的参数使得该模型拟合效果欠佳，而适度增加神经元个数则可以提高其拟合度。

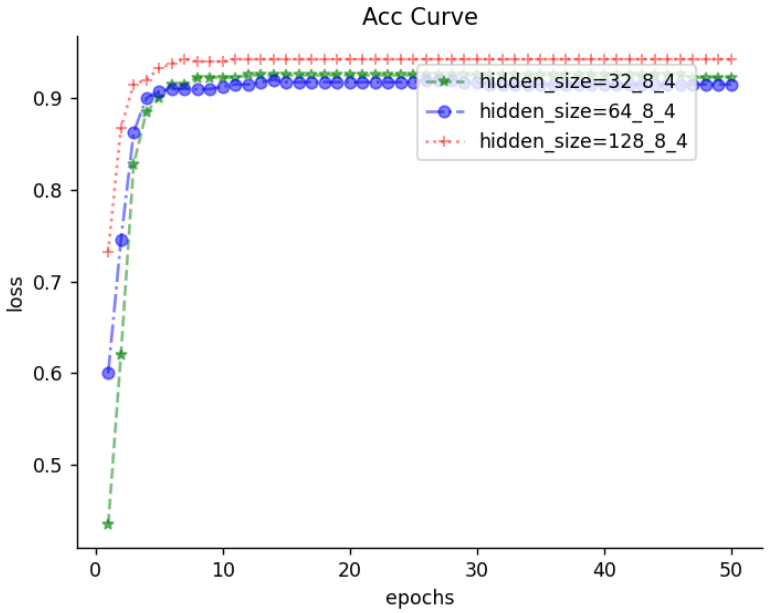
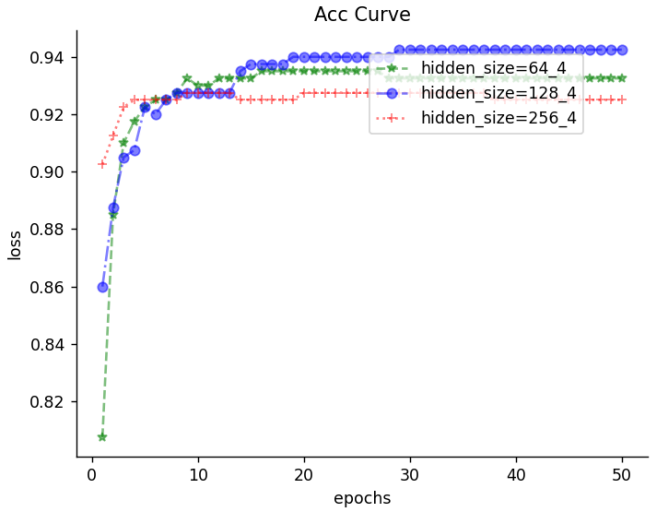


图 3 修改隐藏层 1 的训练效果

其次，我们运行了两层结构的前馈神经网络，我们注意到隐藏层 1 的神经元个数会影响损失的初始值，但在一定程度上，并不会过大影响其收敛值。



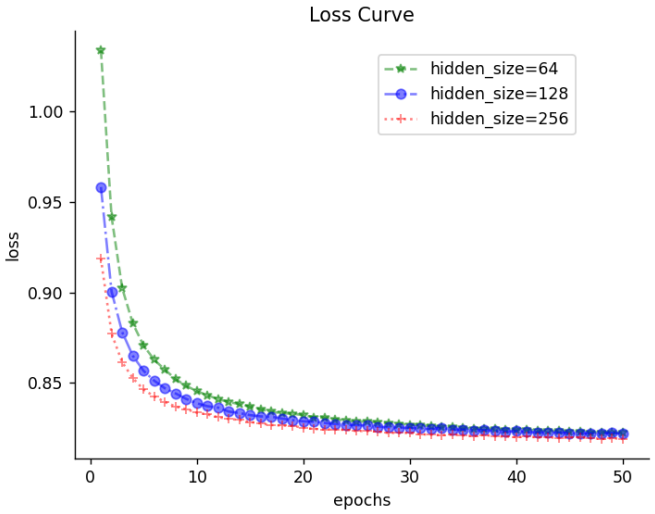


图 4 仅使用 1 层隐藏层的修改效果

(三) 激活函数对于训练结果的影响

我们使用不同的激活函数探讨其对训练的影响。由于算力受限，我们只运行到 50 轮，这里我们注意到 Softplus 的损失最低，其收敛速度远远超过 Sigmoid 和 Tanh。这是因为 Softplus 是对 ReLU 的平滑函数，其功能和 ReLU 相近，对于远程求导为 1，梯度消失的可能性会降低。而 Sigmoid 的导数最大值仅为 0.25，易于产生梯度消失，而且敏感区间短，因此其 Loss 值最大；Tanh 函数的敏感区间相对较大，因此居于中间。

随着训练的进行，其 Loss 将会趋同，达到与 Softplus 组相近的 0.2 水平。其余两组的损失将会进一步下降。

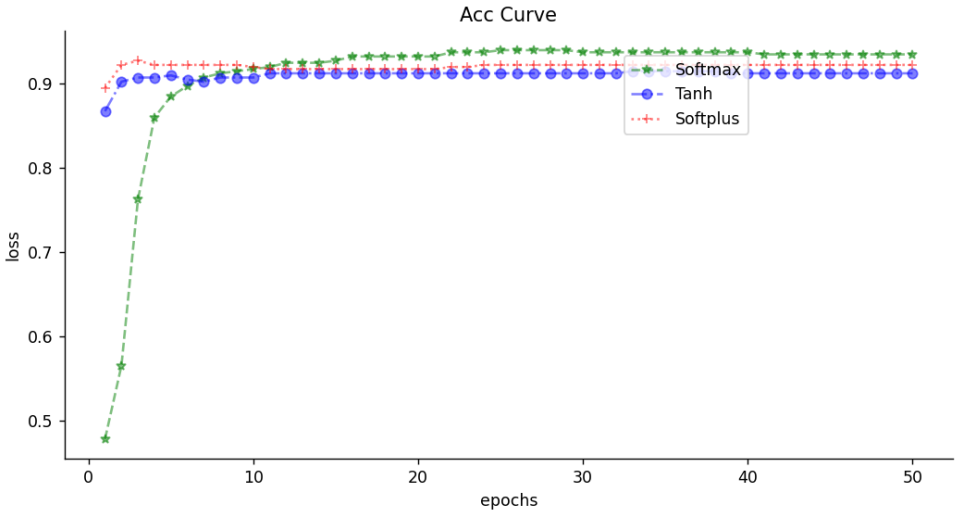


图 5 不同激活函数的对应准确率

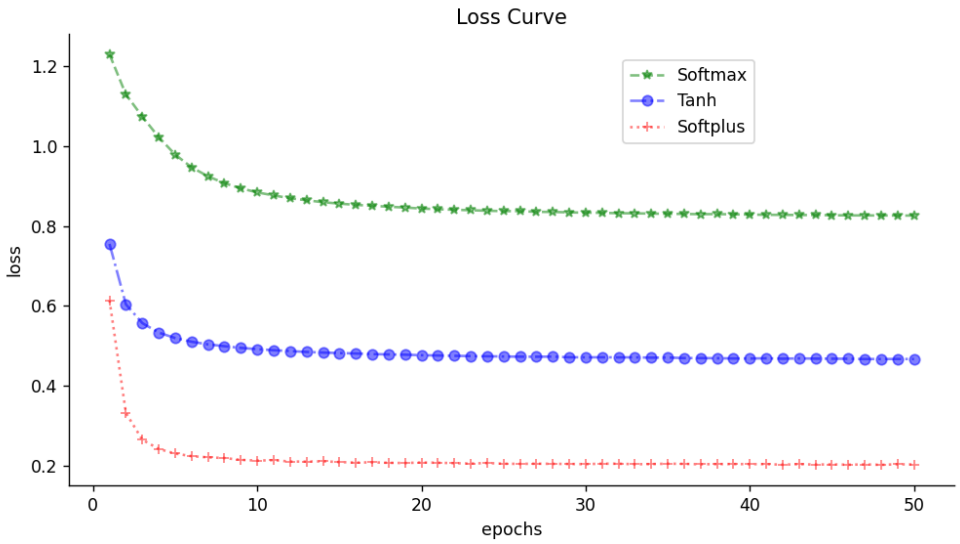


图 6 不同激活函数的训练 Loss 曲线

(四) 对于不同的库函数训练的效果

我们还尝试性地导入 sklearn 库进行比照实验。我们搭建了一个隐藏层分别为 16/8/4 的 Sequential()模型对于数据进行训练，其效果如图所示。

```
113/113 [==============================] - 0s 2ms/step - loss: 0.2152 - accuracy: 0.9231 - val_loss: 0.1797 - val_accuracy: 0.9475
Epoch 42/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2145 - accuracy: 0.9233 - val_loss: 0.1829 - val_accuracy: 0.9450
Epoch 44/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2129 - accuracy: 0.9239 - val_loss: 0.1868 - val_accuracy: 0.9450
Epoch 45/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2137 - accuracy: 0.9228 - val_loss: 0.1803 - val_accuracy: 0.9500
Epoch 46/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2134 - accuracy: 0.9250 - val_loss: 0.1782 - val_accuracy: 0.9500
Epoch 47/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2138 - accuracy: 0.9253 - val_loss: 0.1769 - val_accuracy: 0.9475
Epoch 48/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2138 - accuracy: 0.9228 - val_loss: 0.1790 - val_accuracy: 0.9475
Epoch 49/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2140 - accuracy: 0.9242 - val_loss: 0.1747 - val_accuracy: 0.9500
Epoch 50/50
113/113 [==============================] - 0s 2ms/step - loss: 0.2129 - accuracy: 0.9236 - val_loss: 0.1763 - val_accuracy: 0.9500
13/13 [==============================] - 0s 2ms/step - loss: 0.1763 - accuracy: 0.9500
测试集准确率:  0.949999988079071
```

图 7 使用 sklearn 库效果

我们在测试集上达到的正确率为 95%，而前面我们使用 PyTorch 相关模型测试时，正确率仅为 94.25%。这个原因可能是我们因为算力的限制，前面没有训练完全；如果运行更多轮，我们将会使得损失值进一步下降，从而使得模型拟合出更好的效果。

1.4 讨论

对于模型测试的准确率，很明显的变化是：随着训练的进行，准确率会先上升，后略为降低。这是因为模型在拟合过后产生了过拟合现象，过度适应训练集数据，导致对于测试集的分布则识别效果会降低。

我们在每个训练中，隐藏层的神经元个数均较多，因此表达能力相对较好。我在作业 1 中尝试在 Playground 中使用较少神经元的隐藏层，试图观察其效果。我发现隐藏层神经元个数过少时（比如一层只有一个神经元），其表达能力会大幅下降，因此我们在实验一不再进行更多的探讨。

激活函数会改变损失值的收敛速度。在我们进行的实验中，我们任务 Softplus 的效果最好，这是因为其敏感区间较大，一旦堆叠的隐藏层层数较多，也能较好地进行参数传递，梯度消失的情况略有缓解。