

哈爾濱工業大學

人工智能软件开发与实践
实验报告

题	目	多层感知机 MLP
学	院	计算机科学与技术
专	业	人工智能
学	号	
学	生	
任	课 教 师	武小荷

哈尔滨工业大学计算机科学与技术学院

2023.9

实验一：多层感知机 MLP

1、实验内容

搭建 Python 和 Pytorch 环境，并在 Iris 数据集上使用多层感知机实现数据的分类，主要包括三个部分：

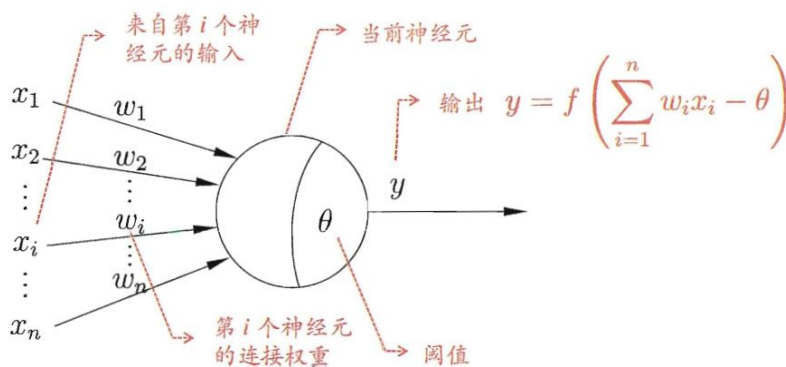
- 下载、预处理 Iris 数据集
- 使用 Pytorch 实现多层感知机(包含输入输出层>3 层)，多层感知机的层数、隐藏层维度，batch size 和 epoch 按需设置
- 在测试集上的准确率不低于 90%

2、算法简介及其实现细节

2.1 神经元模型

如图 1 所示，在这个模型中,神经元接收到来自 n 个其他神经元传递过来的输入信号，这些输入信号通过带权重的连接(connection)进行传递,神经元接收到的总输入值将与神经元的阈值（偏置）进行比较,然后通过“激活函数” (activation function) 处理以产生神经元的输出。

理想中的激活函数是阶跃函数 $sgn(\cdot)$ ，它将输入值映射为输出值“0”或“1”，显然“1”对应于神经元兴奋，“0”对应于神经元抑制。然而，阶跃函数具有不连续、不光滑等不太好的性质，实际常用 Sigmoid、ReLU 等函数作为激活函数把许多个这样的神经元按一定的层次结构连接起来,就得到了神经网络。



2.2 多层感知机模型

多层感知机(Multilayer Perceptron, MLP)是一种前向结构的人工神经网络，映射一组输入向量到一组输出向量，MLP 可以被看作是一个有向图，由多个的节点层所组成，每一层都全连接到下一层，除了输入节点，每个节点都是一个带有非线性激活函数的神经元。MLP 网络结构包含输入层、输出层及多个隐藏层，其中输入层神经元接收外界输入，隐藏层与输出层神经元对信号进行加工，最终结果由输出层神经元输出。3 层感知机的神经网络图如下所示：

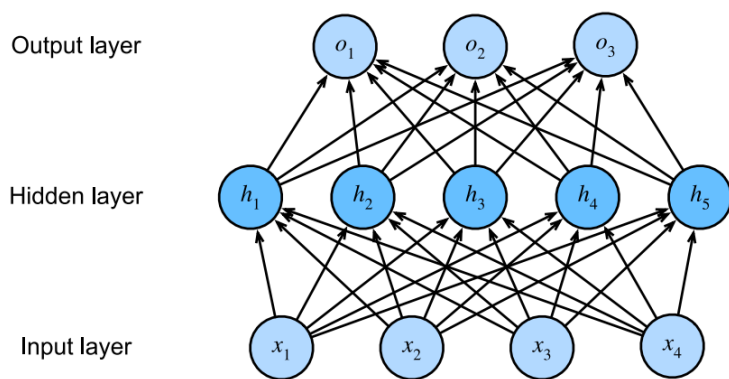


图 2 多层感知机图示

一个 MLP 可以视为包含了许多参数的数学模型，这个模型是若干个函数 $y_j = f(\sum_i w_{ij}b_i - \theta_j)$ 相互(嵌套)代入得到的。而对于给定由 d 个属性描述，输出为 l 维实值向量的训练集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}, x \in \mathbb{R}^d, y \in \mathbb{R}^l$ ，则隐藏层第 h 个神经元接收到的输入为 $\alpha_h = \sum_{i=1}^d v_{ih}x_i$ ，输出层第 j 个神经元接收到的输入为 $\beta_j = \sum_{h=1}^q w_{hj}b_h$ 。假设神经元激活函数为 $\sigma(\cdot)$ 。

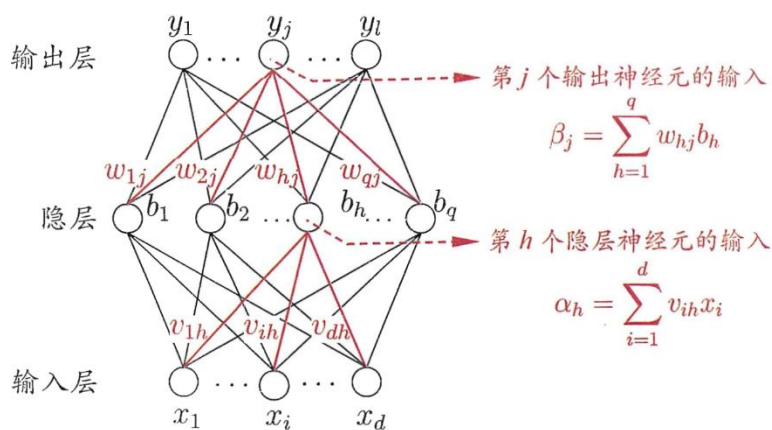


图 3 神经网络中变量符号

模型训练主要包括前馈传播和反向传播两个步骤，前馈传播负责计算模型的预测值，而反向传播负责计算梯度并更新模型的参数，降低损失函数，以便在训练中不断改进模型的性能。

具体而言，前馈传播是神经网络中的正向计算过程，它从输入层开始，沿着网络的层级顺序将数据传递到输出层，从而计算模型的预测值，但此过程并不涉及权重和偏差的更新。

反向传播则是使用前馈传播计算模型的输出，并将其与实际目标进行比较，计算损失（误差），而后从输出层开始基于链式法则计算损失对每个权重和偏差的梯度，使用 Adam、SGD 等优化算法来更新网络中的权重和偏差，以减小损失函数的值。通过反复迭代前馈传播和反向传播过程，多层感知机可以逐渐调整其权重和偏差，从而提高对输入数据的表示能力和泛化能力。

3、实验设置及结果分析（包括实验数据集）

3.1 Iris 数据集及数据处理

鸢尾花数据集(Iris dataset)包含了来自三个不同种类(Iris setosa, Iris versicolor, Iris virginica)的鸢尾花(Iris)的观测数据，每个种类有 50 个样本。每个样本由包括花萼长度(sepal length)、花萼宽度(sepal width)、花瓣长度(petal length)和花瓣宽度(petal width)四个特征描述。

在本实验中，通过 np.loadtxt 方法读取 iris.data 文件，而后通过 split 方法按字符','对特征及标签进行分割，并将标签通过字典映射为 0.1.2 的数字。同时将 Iris 数据集按 0.3 的划分比随机划分为训练集和验证集。

```
class iris_dataset(Dataset):
    def __init__(self, path='./iris.data'):
        super().__init__()
        iris = np.loadtxt(path, dtype=str)
        iris_data = [iris[i].split(',')[::-1] for i in range(len(iris))]
        self.iris = torch.from_numpy(np.array(iris_data, dtype=float)).type(torch.float32)
        self.iris = self.iris / self.iris.norm(dim=-1, keepdim=True)

        label = [iris[i].split(',')[::-1] for i in range(len(iris))]
        label_list = list(set(label))
        self.label_len = len(label_list)
        mapping = {label_list[i]: i for i in range(len(label_list))}
        self.label = [mapping[label[i]] for i in range(len(label))]

    def __len__(self):
        return self.iris.shape[0]

    def __getitem__(self, index):
        label = torch.zeros(self.label_len)
        label[self.label[index]] = 1
        return self.iris[index], label

    def get_splits(self, n_test=0.3):
        # determine sizes
        test_size = round(n_test * self.__len__())
        train_size = self.__len__() - test_size
        # calculate the split
        return random_split(self, [train_size, test_size])
```

图 4 数据处理代码

3.2 模型设计

在本次实验中，实现了包含输入输出层共 4 层的 MLP，其中激活函数选用 ReLU(·)，并通过 softmax 函数输出各类别的概率分布。同时使用 kaiming_uniform 和 xavier_uniform 方法对模型参数进行初始化，从而避免训练过程中梯度消失和梯度爆炸问题。

```

class MLP(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(MLP, self).__init__()

        self.fc1 = nn.Linear(input_dim, hidden_dim)
        kaiming_uniform_(self.fc1.weight, nonlinearity='relu')
        self.relu = nn.ReLU()

        self.fc2 = nn.Linear(hidden_dim, hidden_dim)
        kaiming_uniform_(self.fc2.weight, nonlinearity='relu')
        self.relu = nn.ReLU()

        self.fc3 = nn.Linear(hidden_dim, output_dim)
        xavier_uniform_(self.fc3.weight)
        self.softmax = nn.Softmax(dim=-1)

```

图 5 模型设计代码

3.3 实验结果

在本次实验中，使用交叉熵损失函数和 Adam 优化器，将隐藏层维度设为16，即 MLP 网络4层的维度分别为[4,16,16,3]，设置训练超参数 $epoch$ 为50，学习率 $learning\ rate$ 为0.01。训练过程中损失函数 $loss$ 的值和在测试集上的准确率变化如下图 6 所示。

实验发现，随训练过程的进行，损失函数不断降低，在测试集上准确率逐渐升高，最终测试正确率能够达到97.78%。测试准确率在最后阶段呈现波动态，可能原因是在局部最优点附近振荡。

同时，经过重复测试发现每次实验结果具有一定的随机性，分析可能是模型参数初始化和数据集划分的随机性导致的。

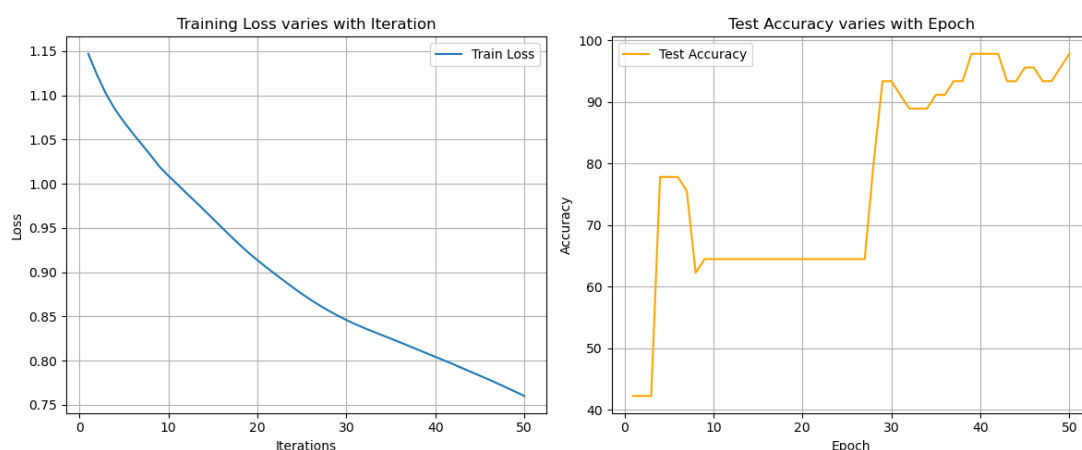


图 6 实验结果（左为损失变化，右为测试集上准确率）

4、实验结论

多层感知机模型是矩阵与向量的乘积的非线性变换的多次重复，能够学习和捕捉复杂的非线性关系，其基本结构较为简单，其具有较强的表达能力，可适应图像分类、识别等多种人工智能任务。

5、实验收获

在本次实验中，通过实际使用调试了解了现有深度学习框架 Pytorch，对其常用的接口如 `torch.utils`、`torchvision.datasets`、`torch.nn`、`torch.optim` 等有了进一步的学习，锻炼提高了我们的代码能力。

同时在自行搭建模型框架的过程中，熟悉了 MLP 前向推理和反向传播的过程，熟悉了多种激活函数的使用方法。

在代码实际编写过程中，会因为不注意维度的变化和处理导致程序出错，也会因为混淆数据在 CPU 还是在 GPU 上导致赋值或比较等操作出错，也会因为学习率设置的过大或者过小导致结果不收敛，在逐步调试代码的过程中也是对我们能力的锻炼。

6、参考文献

- [1] 李航. 统计学习方法[M]. 清华大学出版社, 2012.
- [2] 周志华. 机器学习[M]. 清华大学出版社, 2016.