# 实验1：使⽤Pytorch构建多层感知机报告

# 实验背景

**1.1实验数据**

ionosphere数据集（电离层数据集）

**1.2实验环境**

Windows 11，PyCharm，python 3.10

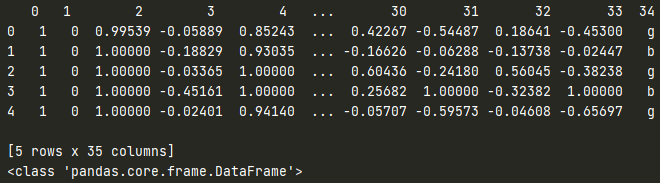
**1.3实验内容**

本次实验将使⽤Pytorch构建⼀个简单的神经⽹络：多层感知机，并完成⼀个简单的⼆分类任务。使用的数据集为ionosphere.csv（电离层数据集），是UCI机器学习数据集中的经典⼆分类数据集，常被用于训练和测试分类算法。它⼀共有351个观测值，包含34个⾃变量和1个因变量（雷达回波的类别），类别取值为g(good)和b(bad)。在ionosphere.csv文件中，共351⾏，前34列作为⾃变量（输入的X），最后⼀列作为类别值（输出的y）。

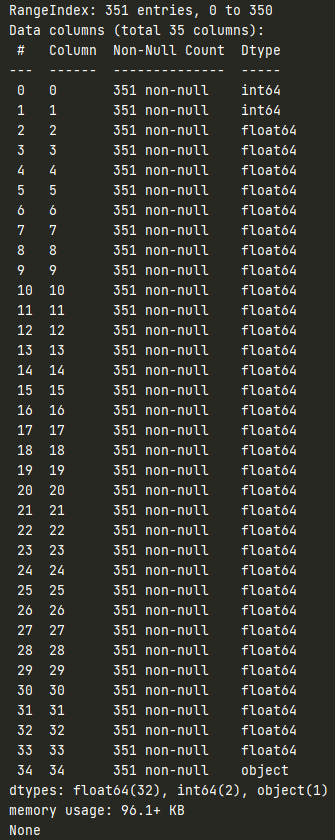
**1.4数据分析**

对该数据集进行简单的数据分析如下：

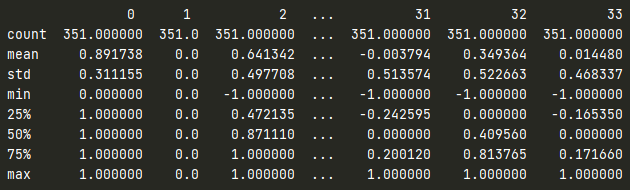
（1）前5行数据。前34列为自变量，最后一列为因变量类别。



（2）数据总体信息，数据类型及缺失值。数据总共351行（条目）、35列，前2列为int类型，中间32列为float类型，最后一列为object(char)类型。



（3）数值型特征的统计摘要。查看各列数据的均值、标准差、最值、四分位值等。



**1.5多层感知机多分类可行性分析**

多层感知机(MLP)由多个神经元层组成，包括输入层、一个或多个隐藏层以及输出层。每个神经元层都与其前一层全连接，即每个神经元都接收前一层的所有输出作为输入，并通过加权和与激活函数处理后产生输出。这种全连接的结构使得多层感知机能够捕捉输入数据之间的复杂关系，并对其进行有效的分类。

MLP通过多个隐藏层可以捕捉复杂的特征和模式，可以通过调整隐藏层的数量和每层中的神经元数量，以适应不同规模的数据集和分类任务，具有较高灵活性；隐藏层通常使用ReLU、Sigmoid等激活函数，这有助于提高网络的非线性表达能力；输出层使用Softmax函数将网络输出转换为类别概率，适合多分类任务；MLP通过多层非线性变换，能够捕捉输入数据之间的复杂关系，从而对复杂问题进行建模，具有强大表达能力；通过训练，MLP能够学习到输入数据之间的普遍规律，从而对未见过的数据进行准确的分类，泛用性强。

# 实验方案设计

**2.1数据集预处理**

使⽤sklearn和pandas库导入和处理数据集。定义一个CSVDataset类，继承自PyTorch的Dataset类，用于加载和处理存储在 CSV 文件中的数据集。

（1）使用pandas库的read\_csv函数加载CSV文件为Pandas的DataFrame，将前34列特征存储为输入值self.X，并将其转换为float32类型；最后一列类别存储为输出值self.y，并使用 LabelEncoder 对其进行标签编码（即将其转换为从0开始的整数），然后也转换为float32类型，并重塑为二维数组（每个样本一个标签），以匹配PyTorch的数据加载期望。

（2）定义获取数据集长度(\_\_len\_\_)和单个样本(\_\_getitem\_\_)的方法。

（3）定义获取训练和测试集的索引的get\_splits方法，结合random\_split库函数可以根据提供的比例(n\_test)将数据集随机分割为训练集和测试集，并返回两个数据集的索引或子数据集。

**2.2构建多层感知机**

定义一个MLP类，继承自troch.nn.Module，并对\_\_init\_\_和forward方法进行重写。设置两个隐藏层，使用ReLU函数作为激活函数；输出层使用Sigmoid函数作为激活函数。第一个隐藏层输入维度为数据的特征维度，即34，输出维度为18；第二个隐藏层输入维度为18，输出维度为8；输出层输入维度为8，输出维度为1。

在forward方法中，按照各层顺序进行前向传播，更新X向量，得到输出；后向传播在基类Module的默认方法backward中自动进行，计算导数和梯度。

**2.3超参数选择**

可以从构建只有一个隐藏层的网络开始，快速了解模型的基本表现和效果，如果模型在训练集上表现不佳（欠拟合），逐渐增加层数以学习更复杂的特征，达到最佳效果。同时隐藏层不宜过多，否则会增大计算量和复杂度，需要更多计算资源，模型性能提升却很有限，且可能有过拟合的风险。结合本次实验数据集大小和特征数量，选择设置2层隐藏层，并调整输入输出维度。

优化器选择随机梯度下降法，学习率lr从较小值（如0.01）开始，根据训练结果和运行效率进行调整，最终选择0.1作为学习率。

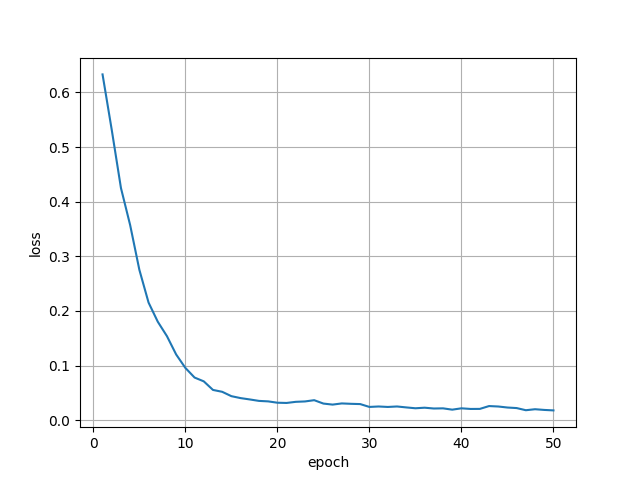
随着epoch数量的增加，权重更新迭代的次数增多，测试发现当epochNum为50时达到较好效果，若再增大则测试准确率降低，可能出现过拟合。

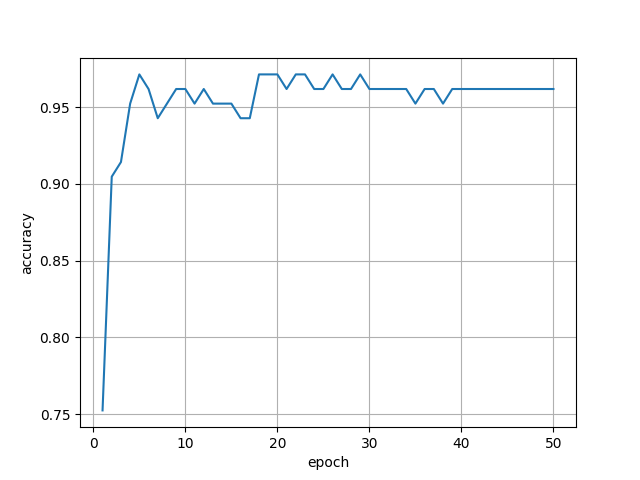
批处理大小（Batch Size）一般为2的幂，如32、64、128等，较大的batch\_size可以加快训练速度，但可能会导致模型精度下降；而较小的batch\_size可以提高模型精度，但会减慢训练速度。通过测试发现batch\_size为64时准确率最高。

两个隐藏层选择ReLU作为激活函数，其具有线性、非饱和的性质，在SGD中能够快速收敛，且计算复杂度低，不需要进行指数运算；输出层选择Sigmoid作为激活函数，输出一个0~1范围内的概率值，适合用于二分类的模型，同时梯度平滑，便于求导。

# 实验过程

根据训练过程中输出信息可以发现，随着训练迭代次数增加，loss值在前20代完整训练中迅速降低，在epoch超过20后缓慢降低；准确率迅速升高，在大约5代完整训练后已经达到95%以上，之后保持稳定。

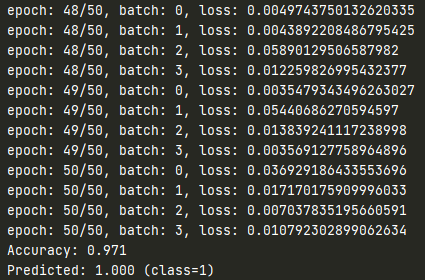




# 性能测试

随机划分数据集，70%作为训练集，30%作为测试集，训练后模型在测试集上的准确率为97.1%。

测试样例为row = [1, 0, 0.99539, -0.05889, 0.85243, 0.02306, 0.83398, -0.37708, 1, 0.03760, 0.85243, -0.17755, 0.59755, -0.44945, 0.60536, -0.38223, 0.84356, -0.38542, 0.58212, -0.32192, 0.56971, -0.29674, 0.36946, -0.47357, 0.56811, -0.51171,0.41078, -0.46168, 0.21266, -0.34090, 0.42267, -0.54487, 0.18641, -0.45300]，预测结果为1类。



# 实验习题

1. 写出一个单层MLP的数学表达式并求他对于输入的导数

以隐藏层作为当前层，规定以下变量：

|  |  |
| --- | --- |
| i | 输入层节点下标 |
| j | 当前层神经元下标 |
| k | 输出层神经元下标 |
| wij | 输入层各节点到当前层神经元的权重 |
| wjk | 当前层神经元到输出层各神经元的权重 |
| bj | 当前层神经元到输出层各神经元的偏置 |
| hj | 当前节点的所有输入加权之和 |
| aj | 隐藏层神经元的输出值 |
| g() | 激活函数 |
| M | 数据特征数量/输入维度 |

有，其中wojx0j即为偏置bj。

其对于输入的导数为

1. 根据你设置的超参数，写出你构建的MLP的数学表达式，画出他的计算图

隐藏层1：

隐藏层2：

隐藏层3：

计算图如下所示：

