|  |  |
| --- | --- |
| 成绩 |  |

装

订

线

网络空间安全与计算机学院实验报告

实验课程名称**：神经网络与深度学习实验**

实验项目名称： 基于前馈神经网络的二分类任务

学生姓名： 刘新媛

专 业： 人工智能

学 号： 20221205037

实验地点： C1-320

实验时间： 2024.10.16第7-8节

指导教师： 魏勇刚

预习报告部分

1. **实验目的：**实现一个基于前馈神经网络的二分类任务
2. **实验原理：**

1、前馈神经网络（FNN）：由输入层、隐藏层和输出层组成的神经网络，信息只在一个方向上流动，从输入层到隐藏层，再从隐藏层到输出层。

2、反向传播算法：通过链式法则计算损失函数对网络参数的梯度，从输出层逐层反向传播。

3、梯度下降优化：使用批量梯度下降优化器，根据计算得到的梯度更新模型的权重和偏置，以最小化损失函数。

1. **实验内容：**

**[作业链接](https://blog.csdn.net/qq_38975453/article/details/126492558)**

### 1. 数据集构建

使用Moon1000数据集，包含640条训练样本、160条验证样本和200条测试样本，每个样本有两个特征，目标是二分类。

### 2. 模型构建:定义每一层的算子，然后再通过算子组合构建整个前馈神经网络。

### 3. 损失函数:采用二分类交叉熵损失函数，衡量模型预测与真实标签之间的差距。

### 4. 模型优化:

* 前向计算：使用forward()方法计算输出。
* 反向计算：使用backward()方法计算梯度。
* 参数更新：在优化器中进行梯度下降更新。

### 完善Runner类：RunnerV2\_1

### 支持自定义算子的梯度计算，在训练过程中调用self.loss\_fn.backward()从损失函数开始反向计算梯度；每层的模型保存和加载，将每一层的参数分别进行保存和加载。

### 6. 模型训练

训练2000个epoch，使用训练集和验证集，评价指标为准确率（accuracy）。

### 7. 性能评价

使用测试集评价最优模型，观察准确率和损失值。

### 8.思考题

对比“基于Logistic回归的二分类任务”与“基于前馈神经网络的二分类任务”，谈谈自己的看法。

1. **实验所用设备：**pycharm、pytorch框架
2. **实验步骤：**1.数据预处理2.模型构建3.模型训练4.模型评价5.模型预测

实验报告部分

1. **实验步骤：**

#### 1 数据集构建

使用之前的[二分类](https://so.csdn.net/so/search?q=%E4%BA%8C%E5%88%86%E7%B1%BB&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/m0_57215376/article/details/_blank)数据集：Moon1000数据集，包含训练集640条、验证集160条、测试集200条，每个样本包含2个特征。这里直接**from data import make\_moons，**调用之前make\_moons函数来生成数据集。

#### 2 模型构建

网络的第 层的输入为第 层的神经元活性值，经过一个**仿射变换**()，得到该层神经元的净活性值，再输入到激活函数得到该层神经元的活性值。

为了提高模型的处理效率，将N个样本归为一组进行成批地计算。假设网络第 层的输入为其中每一行为一个样本，则前馈网络中第 层的计算如公式1-1、1-2所示，各个符号含义如表1-1所示。

(1-1)

(1-2)

**表 1-1**

|  |  |
| --- | --- |
| **符号** | **含义** |
|  | 一个矩阵，维度是。  表示N个样本第 层神经元的净活性值。 |
|  | 一个矩阵，维度是。  表示N个样本第 层神经元的活性值。 |
|  | 一个矩阵，维度是。  表示第 层的权重矩阵。 |
|  | 一个向量，维度是。  表示第 层的偏置。 |

实验中先定义每一层的算子，需用到的算子及定义的类如表1-2所示，然后再通过算子组合构建整个前馈神经网络。

**表 1-2**

|  |  |
| --- | --- |
| **算子/类** | **功能** |
| Op | 基类，提供统一的调用方式 |
| Linear | 线性层，实现线性变换和梯度计算 |
| Logistic | 执行Sigmoid激活以及激活值导数计算 |
| Optimizer | 优化器基类，实现梯度下降法中参数的更新 |
| BatchGD | 批量梯度下降优化器 |
| BinaryCrossEntropyLoss | 二分类交叉熵损失，计算损失和梯度。 |
| Model\_MLP\_L2 | 二层全连接神经网络，执行前向和反向传播 |

本节仅简单搭建前向传播的网络架构，不涉及反向传播以及梯度的计算。

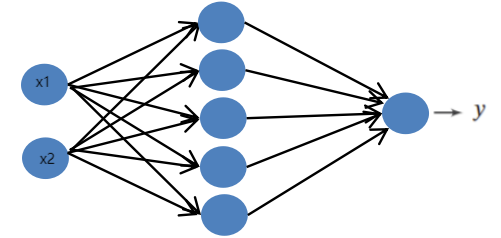
**（1）线性层算子：**用输入数据与权重矩阵相乘，并加上偏置，返回输出。

**（2）Logistic算子：**采用Logistic函数来作为公式1-2中的激活函数。

**（3）层的串行组合：**将不同的隐藏层、输入层、输出层之间串联起来，不断交叉重复使用它们来构建一个多层的神经网络。

这里构建网络传播过程为：输入数据 X 经过第一个线性层 fc1--->得到线性组合结果 z1---->z1 通过第一个激活函数 act\_fn1，得到激活值 a1----->a1 经过第二个线性层 fc2，得到第二个线性组合结果 z2---->z2 通过第二个激活函数 act\_fn2，得到最终的预测值 a2。

网络架构由一个输入层（2个神经元--每个样本有两个特征）、一个隐藏层（5个神经元，可调）、一个输出层（1个神经元--表示预测类别的概率），如图1-1所示。



**图 1-1 网络架构**

#### 3 损失函数

采用二分类交叉熵损失函数。

#### **4 模型优化 ※**

神经网络的层数通常比较深，其梯度计算和上一章中的线性分类模型的不同的点在于：线性模型通常比较简单可以直接计算梯度，而神经网络相当于一个复合函数，需要利用链式法则进行反向传播来计算梯度。

#### 反向传播算法

前馈神经网络的参数梯度通常使用误差反向传播算法来计算。

|  |
| --- |
| **误差反向传播算法：** |
| **Step1** **前向传播：**计算每一层的净活性值和激活值，直到最后一层；  **Step2** **反向传播：**从输出层向输入层传播误差，反向计算每一层的误差项。对于输出层误差项的计算如公式1-3所示，对于隐藏层误差项的计算如公式1-4所示。  （1-3）  （1-4）  其中，是损失函数,是激活函数的导数。  **Step3 更新参数：**通过计算每一层的梯度，来更新权重和偏置参数。 |

本实验中的前向计算，利用算子的forward()方法来实现；反向计算梯度，利用算子的backward()方法来实现；计算参数梯度也放到backward()中实现，更新参数放到另外的优化器（BatchGD）中专门进行。

按照反向的梯度传播顺序，**为每个算子添加backward()方法**，并在其中实现每一层参数的梯度的计算。

#### 损失函数

|  |
| --- |
| **二分类交叉熵损失函数,实现损失函数的backward():** |
| **Step1初始化\_\_call\_\_ ()**:创建类的实例，并传入模型。  **Step2前向传播forward()**：在每次训练迭代中，先进行前向传播计算模型的预测，再使用预测值和真实标签计算损失值。  **Step3反向传播backward()**：计算损失函数对预测值的梯度。将梯度传递给模型，模型根据梯度更新其参数。  **迭代**：重复步骤2和3，直到模型收敛或达到预定的迭代次数。 |

#### Logistic算子

为实现的Logistic算子增加反向函数backward()，在每次反向传播时，首先计算Sigmoid函数的导数，然后将这个导数与上游传递的梯度相乘，得到当前层输入的梯度。这个梯度将被传递到前一层，用于更新前一层的参数。

#### （4）线性层

为实现的线性层算子增加反向函数backward()，在每次反向传播时，首先计算权重和偏置的梯度，然后将这些梯度用于更新权重和偏置。计算得到的梯度将被传递到前一层，用于更新前一层的参数。

#### 整个网络

为之前搭建的网络Model\_MLP\_L2 增加反向函数backward()，从输出层开始，根据损失函数计算的梯度，依次通过各层的反向传播，更新模型的权重和偏置。

|  |
| --- |
| **Model\_MLP\_L2 --backward()：** |
| 1. 从输出层开始，接收上游传递的梯度loss\_grad\_a2。 2. 通过第二个激活函数层self.act\_fn2进行反向传播，计算第二层线性层的梯度 loss\_grad\_z2。 3. 通过第二个线性层self.fc2进行反向传播，计算第一层激活函数的梯度 loss\_grad\_a1。 4. 通过第一个激活函数层self.act\_fn1进行反向传播，计算第一层线性层的梯度 loss\_grad\_z1。 5. 最后通过第一个线性层self.fc1进行反向传播，更新模型的权重和偏置。 |

#### 优化器

计算好神经网络参数的梯度后，将梯度下降法中参数的更新过程实现在优化器中。此处的优化器需要**遍历每层**，对每层的参数分别做更新。

这里采用**批量梯度下降**是一种简单而有效的优化方法，它在每次更新参数时都使用整个训练集。通常会导致每次更新步骤比较慢，但可以保证使用所有数据来计算梯度，从而使得梯度的估计更加准确。

#### 5 完善Runner类：RunnerV2\_1

在之前的基础上添加的主要功能有：

1、支持自定义算子的梯度计算，在训练过程中调用self.loss\_fn.backward()从损失函数开始反向计算梯度；

2、每层的模型保存和加载，将每一层的参数分别进行保存和加载。

|  |
| --- |
| **完善后的RunnerV2\_1:** |
| **初始化：**设置模型、优化器、损失函数和评估指标，并记录训练和验证的得分与损失。  **训练：**执行多轮训练计算损失与得分，进行反向传播更新参数，并评估验证集表现。  **验证：**计算模型在验证集上的损失和评估指标。  **预测：**对输入数据进行预测。  **保存模型：**将每层的参数保存到指定路径。  **加载模型：**从指定目录加载模型参数。 |

#### 6 模型训练

基于RunnerV2\_1,使用训练集和验证集进行模型训练，共训练2000个epoch。评价指标为accuracy。

|  |
| --- |
| **模型训练过程：** |
| 1. 定义训练参数：设定训练轮数和**最优模型**保存目录。 2. 配置网络参数：指定输入层、隐藏层和输出层的维度。 3. 定义模型：创建多层感知机实例。 4. 设置损失函数：使用二元交叉熵损失适用于二分类任务。 5. 定义优化器：设置批量梯度下降优化器和学习率。 6. 选择评估指标：使用accuracy作为模型评估标准。 7. 实例化 RunnerV2\_1：传入模型、优化器、损失函数和评估指标。 8. 训练模型：调用 train 方法进行训练。 |

#### 7 性能评价

使用 load\_model 方法从指定目录加载先前训练好的模型参数，使用**测试集**对训练中的**模型**进行评价，通过 调用RunnerV2\_1的evaluate 方法在测试集上进行性能评估，返回模型的准确率和损失值。

#### 8【思考题】对比“基于Logistic回归的二分类任务”与“基于前馈神经网络的二分类任务”，谈谈自己的看法。

Logistic回归模型相对简单，适用于较简单的线性可分的数据。前馈神经网络较为复杂。通过隐藏层和激活函数，能够捕捉非线性关系，可以处理更复杂的数据模式，适用于特征较多或存在非线性关系的情况。通过两次实验对比，**前馈神经网络的最终score并没显示出多大的优越性**。我认为原因有二：

1、由于二分类问题较为简单，没有太大必要使用复杂的网络，使用简单的二分类模型也可以达到很好的效果。

2、网络的超参数设置不够好，或者激活函数的选择不恰当、优化器选择不恰当等等，总归无论如何神经网络肯定是更强大的。

【ps:这里我借助工具以及资料重新更改了一下网络结构，代码已在csdn中给出】

**二、 实验数据及结果分析：**

**[点击查看代码及结果分析](https://blog.csdn.net/qq_73704268/article/details/143062210?sharetype=blogdetail&sharerId=143062210&sharerefer=PC&sharesource=qq_73704268&spm=1011.2480.3001.8118)**

1. **实验结论：**

1、前馈神经网络能够有效地处理二分类问题

2、使用测试集对最优模型进行评估后，观察到模型具有较高的准确率和较低的损失值，表明模型具有良好的泛化能力。

3、尽管Logistic回归模型简单，但在本实验的简单二分类问题上，前馈神经网络并未表现出显著的优越性。这可能是由于问题本身的简单性，以及网络超参数设置和激活函数选择的不恰当。

1. **总结及心得体会：**

1、这次实验历时较长，主要是由于知识点的缺漏导致在调试代码bug的过程中耽误了，比如:

①标准正态分布初始化权重不能直接使用torch.stand\_normal,因为torch没有这个函数，需要使用torch.randn函数。也不能使用torch.normal函数，因为他是离散正态分布，并且还需要两个参数标准差和均值。



正确的方法应该是：

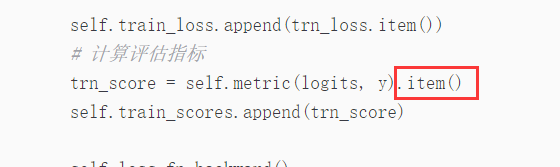
设置 weight\_init=**torch.randn**

self.params['W'] = torch.randn(size=(input\_size, output\_size))

②torch中没有cast函数，在paddle框架中cast函数可以将输入数据转换为其他类型数据，而在torch中需要用**torch.Tensor.to()**函数。

③torch中设置随机数种子不能像paddle一样直接用torch.seed()，而应该使用**torch.manual\_seed()**

2、有一件奇怪的事，paddlepaddle中封装的RunnerV2-1类有一句代码是这样写的：



运行报错显示



就是item() 只能用于 **PyTorch 的张量**对象。而self.metric 函数返回的是浮点数，这里我把item(）删掉就不报错了！！【啊哈，会不会是因为框架不同呢.....paddle框架下他返回的就是一个张量嘛？？】

3、这次实验不仅让我感受到了神经网络的强大之处，经过这次实验对类继承性的认识也进一步强化，也认识的到一些初始化方法的神奇，如call方法，如果一个对象需要处理输入数据并返回结果，使用 \_\_call\_\_ 可以避免显式调用一个名为 forward 的方法，使用起来更直观。这样，调用模型时可以直接使用 model(X)，而不需要写成 model.forward(X)。

|  |
| --- |
| **参考链接:** |
| [torch.cast](https://wenku.csdn.net/answer/de4ef3b7247e45baa49b116b3826db7c?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522E270A98C-1D21-43D0-AF7F-78B88124FFB2%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=E270A98C-1D21-43) |
| [paddle.cast](https://wenku.csdn.net/answer/c4553d429da445559a37227206bdc5cd?ops_request_misc=%257B%2522request%255Fid%2522%253A%25222E190DF1-12B2-4771-8D12-DC6640CCCB94%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fcommercial.%2522%257D&request_id=2E190DF1-) |
| [【Pytorch】torch.normal()函数](https://blog.csdn.net/weixin_41796265/article/details/131454547?ops_request_misc=%257B%2522request%255Fid%2522%253A%25225A4A61C2-225D-4B80-B9DE-8E7894C67337%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=5A4A61C2-225D-4B80-B9DE-8) |
| [关于numpy,torch中seed()方法的一些理解](https://blog.csdn.net/F845992311/article/details/123553715?ops_request_misc=%257B%2522request%255Fid%2522%253A%25221E867067-C369-45F8-916D-7E2A45ECDEC7%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=1E867067-C369-45F8-916D-7E2A45) |
| [paddle前馈神经网络理论解读](https://aistudio.baidu.com/projectdetail/8381167) |
| [Python当中类的\_\_call\_\_()方法、forward()方法以及\_\_getitem\_\_()方法作用是否会重叠](https://blog.csdn.net/m0_57317650/article/details/134855307?ops_request_misc=&request_id=&biz_id=102&utm_term=%E7%A5%9E%E7%BB%8F%E7%BD%91%E7%BB%9C%E7%9A%84call%E6%96%B9%E6%B3%95&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default) |
| [深度学习之前馈神经网络的入门学习（我觉得是全网最详细的）](https://blog.csdn.net/weixin_55189321/article/details/131193139?ops_request_misc=%257B%2522request%255Fid%2522%253A%25225D91F126-5D0E-4C4D-8863-F5950758B713%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=5D91F126-5D0E-4C4D-8863-F) |