**目录**

**[1.实验背景、目的、任务要求与思路 2](#_Toc11111)**

**[2.实验基本流程 3](#_Toc11111)**

**[3.心得体会 1](#_Toc11111)1**

**1.实验背景、目的、任务要求、思路：**

**1.1背景**

目前电信行业充满挑战，商业竞争激烈。因此，电信供应商的管理者们想通过挖掘历史数据信息分析客户群体流失的原因，即管理其客户群并减少客户流失。

数据介绍：

churn-bigml-80.csv：每一行表示一条用户记录，每一列表示用户的 20 种属性信息。该

数据包括自身属性数据，例如所在位置；客户活动数据，例如白天和黑夜呼叫的数量和长度

以及客户服务呼叫的数量；以及指定客户是否取消订阅的目标预测标签。

churn-bigml-20.csv：原始测试数据集。

**1.2目的：**

本次任务希望使用 Orange Telecom 的相关历史数据来开发一个健壮的客户流失预测模型，该模型可以识别导致客户流失的关键变量，并提醒电信运营商哪些客户可能会在未来退订其服务。

目的：根据客户过去的行为来预测客户的未来决策，即已知前 19 列信息预测第 20 列是否取消订阅。

**1.3任务要求：**

利用给出的 churn-bigml-80 数据集做模型训练，利用给出 churn-bigml-20 数据集做模型

测试（严禁在模型训练过程中使用 churn-bigml-20 中的任何信息）

具体训练+测试任务如下：

(1) 搭建课上学习的基本的 BP 神经网络，并实现模型训练，训练好之后在测试集上计

算预测效果，给出模型的预测准确率(Accuracy)、查准率(Precision)、查全率(Recall)。

(2) 尝试改进网络架构以提升模型在测试集上的准确率，并分析架构相关参数（如网络

层数，每层神经元个数，激活函数等等）对准确率的影响。

(3) 对以上表现最佳的模型尝试进一步优化，通过改变学习率, 训练轮数等模型训

练相关的超参数，以及尝试不同优化算法（比如 RMSProp，Adam 算法等）。

(4) 探索输入特征（即 19 列属性信息）是否对神经网络预测准确率有影响？减少特征个数是否可以提升预测准确率？通过训练+测试验证你的想法。

**1.4思路：**

分析可知，该问题属于二分类问题。

判断完问题的性质后，我们来采取合适的算法。关于二分类问题，我们从机器学习的课程中有学习到，可以使用逻辑回归、决策树、支持向量机等分类算法。而根据课程设计以及本次实验的要求，我们选择BP神经网络为模型。

**2.实验基本流程**

**2.1编程环境**

**IDE**： PyCharm community版本

**编程语言**： Python3.9

**深度学习框架:** Pytorch1.13.1

**2.2搭建网络模型**

1. **数据预处理：**

**1.1将文本数据统一转化为数字表示**

通过观察所给数据集可知，数据集中的第1列和第4，5列，以及最后一列标签均为字符串的数据类型。其中，第一列为所属国家缩写，而另外三列都是，Yes和No的二元组合。

通过数据处理经验可以知道，我们可以把Yes替换为1，No替换为0；但是在面对第一列的国家名称时，我有些一些疑惑，原因是：这样多个不同的属性不能简单地像之前那样分成0/1，并且在转化数据时，也不能将数据的特性给丢失掉。

我想到了两种处理办法：

1. 以国家名称为关键字，该国家在训练集中出现的总次数为键值，生成一个字典，然后通过字典，将训练集中的国家名称替换为其对应的键值；

**理论：**

这种方法转化出来的数据对数据集的数据分布有依赖，再不能确保训练集和测试集的数据分布是相似或相同的情况下，训练出来的模型泛化能力比较差，不能很好地进行预测。

**事实：**

采用这种处理方法，最终的实验结果的准确率相比采用第二种方法的较低

1. 按照国家与数字一一映射的关系，将国家名称一一替换为其对应的数字，这也是契合之前的替换思想；

经过最终实验结果的呈现，选择第二种方式能提高预测的准确率，或者说更能保

留数据的内在性质。

**1.2数据归一化：**

数据归一化可以将数据缩放到一个特定的范围内，以消除不同维度数据的量纲以及量纲单位，使得不同维度的特征具有可比性，从而提高模型的精度和稳定性，同时加快模型的训练速度。

**1.3ndarry转化为tensor类型，用于网络传播：**

tensor类型可以更好地与神经网络进行交互，PyTorch框架中的神经网络是基于tensor类型进行搭建的。

1. **定义神经网络：**

在搭建神经网络时，首先继承已有的nn.module基类，再在其上作网络层数和神经元个数的修改，其中包括输入层的input\_size、隐藏层的hidden\_size和输出层的output\_size三个变量。

在这一部分，还需要定义激活函数的种类，常见的有sigmoid,relu和tanh三类函数。在后续的实验结果中，这三种激活函数在效果上会有所区别。

1. **训练：**

**3.1定义训练轮数（Epoch）:**

适当的训练轮数能帮助神经网络更好地学习数据集中的特征，提高模型的准确性。但是，训练轮数过多可能会导致过拟合以及训练时间的消耗，训练轮数过少可能会导致欠拟合，使得准确率降低。我首先采用了epoch = 100，后续再不断调整。

**3.2定义优化器种类（optimizer）：**

最基础的优化器选择有SGD，但是SGD的收敛速度较慢，并且容易陷入局部最优解。还有另外一个更优的Adam优化器，其可以自适应地调整学习率，并且收敛速度快，但是容易过拟合。在实验过程中，我首先采用了SGD，发现效果不如Adam，并且准确率一直上不了0.9。并且，在最后我还加入了正则性，防止过拟合。

**3.3学习率（learning rate）：**

适当的学习率能够使得模型能够优化得更快，这里我首先采用的是0.01的学习率，并且从实验结果来看，速度和效果也比较好。

**3.4损失函数(loss function)：**

二分类问题的损失函数一般采用BCE\_loss函数，这里我也是首选了BCE\_loss作为损失函数，相比其他函数也达到了比较好的实验结果。

**3.5训练过程：**

1. **梯度清零**
2. **前向传播**
3. **计算损失函数**
4. **后向传播**
5. **更新参数**
6. **测试：**

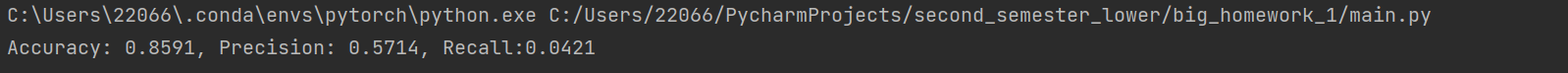
将测试集的数据放入训练好的模型中训练，得到结果，并调用函数计算正确率，查准率和查全率。

1. **结果分析：**

至此，**任务一已经完成**，在如下参数的选取下

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 3 | 50 | BCE\_loss | 500 | 0.01 | Leaky\_relu、sigmoid | Adam |

实验的结果为：



可以看到，实验结果并不是非常地好，这个正确率同全部预测为0的正确率相差无几，

说明网络还未真正地学到数据中带有的潜在规律。

**下面根据任务2**，尝试改进网络架构以提升模型在测试集上的准确率，并分析架构相关参数（如网络层数，每层神经元个数，激活函数等等）对准确率的影响。

1. **修改网络层数：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 50 | BCE\_loss | 500 | 0.01 | Leaky\_relu、sigmoid | Adam |
|  | | | | | | |

在提升网络层数之后，准确率明显上升，并且查准率和查全率都有很大的提高。其中原因是，神经网络的层数越多，就越能够学到数据的更为复杂的特征，使得其预测的能力更好。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 100 | BCE\_loss | 500 | 0.01 | Leaky\_relu、sigmoid | Adam |
|  | | | | | | |

**2.修改隐含层神经元个数：**

在将网络隐含层神经元个数提高之后，准确率提高到0.91，又是较大的进步，此时查准率和查全率分别有所下降和上升，这是合理的，因为查准率和查全率本身就是一对矛盾的量，两者不可能都很高。神经元个数的增加，会使得网络对数据的表达能力增强，更加充分地利用数据的特征，从而找到数据之间的内在联系。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 500 | BCE\_loss | 500 | 0.01 | Leaky\_relu、sigmoid | Adam |
|  | | | | | | |

可以看到，再次修改隐含层神经元个数到500时，准确率不增反降，查准率相比提高了，而查全率又降低了。推测分析原因，当神经元的个数超过合理范围后，部分神经元其实在“偷懒”，无作为，其对最终结果的输出没有起到作用。就像老师在课堂上讲的那样，在一个team中，成员个数多了之后，部分成员就会开始摸鱼，使得整个team效果不好，而当神经元个数在合理范围内时，每个神经元各司其职，充分发挥，就能够使得整个网络的输出效果较好。

**3.修改激活函数：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 500 | BCE\_loss | 500 | 0.01 | Tanh、sigmoid | Adam |
|  | | | | | | |

在修改relu函数为tanh函数之后，正确率又有了一定的提升，而查准率和查全率又是降一升一。sigmoid函数用于处理最后一层，使得结果映射到0-1之间，最后再设定阈值来进行判断。采用relu函数中的leakyrelu能够较好的解决神经元死亡的问题，这里将relu改为tanh函数，尝试一下是否会使得准确率提高，结果如上所示。在准确率上有了更好的提升。

**混合使用激活函数：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 500 | BCE\_loss | 500 | 0.01 | Tanh、Relu、sigmoid | Adam |
|  | | | | | | |

通过对激活函数的混合使用，但是最终的结果与之前相比并未有较大的提升。

**下面根据任务3**，对以上表现最佳的模型尝试进一步优化，通过改变学习率, 训练轮数等模型训练相关的超参数，以及尝试不同优化算法（比如 RMSProp，Adam 算法等）。

1. **改变学习率：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 500 | BCE\_loss | 500 | 0.001 | Tanh、Relu、sigmoid | Adam |
|  | | | | | | |

可以看到，在改变学习率，从0.01-0.001的过程中，正确率有略微的提升，查准率和查全率又是降一升一，总体变化不大，但是我注意到，此时的优化时间比较慢，原因即步长减小，下降优化的速度就会相对较慢。

1. **训练轮数：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 500 | BCE\_loss | 1000 | 0.001 | Tanh、Relu、sigmoid | Adam |
|  | | | | | | |

可以看出，在训练轮数达到1000轮之后，准确率接近0.93，查准率和查全率降一升一。分析可知，训练的轮数越多，使得模型能够更好地优化参数，从而更好地学习了数据的内在特征，将预测能力很好地泛化到新数据。

1. **尝试不同的优化算法：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 500 | BCE\_loss | 1000 | 0.001 | Tanh、Relu、sigmoid | SGD |
|  | | | | | | |

可以看到，在更改优化器算法之后，实验效果大打折扣，推测原因可能是：在优化过程中，SGD优化陷入了某个局部最小值，但是未能跳出该局部最小值，导致全部预测结果均为0，正确率回到了初始状态，使得效果比较糟糕。

**下面根据任务4**，探索输入特征（即 19 列属性信息）是否对神经网络预测准确率有

影响？减少特征个数是否可以提升预测准确率？通过训练+测试验证你的想法。

1. 减少输入特征个数，探索是否对神经网络的预测准确率有影响：

将国家名称属性去掉，只使用18列属性信息，其他相关参数如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 4 | 500 | BCE\_loss | 1000 | 0.001 | Tanh、Relu、sigmoid | Adam |
|  | | | | | | |

可以看到，在输入特征为18列属性信息后，准确率相比之前并未有较大的提升。以下尝试多种选择，减少数据维度。

删去前两列的数据，观察准确率的情况（其他参数均一致）：

|  |
| --- |
|  |

删去最后一列数据，观察准确率的情况（其他参数均一致）：

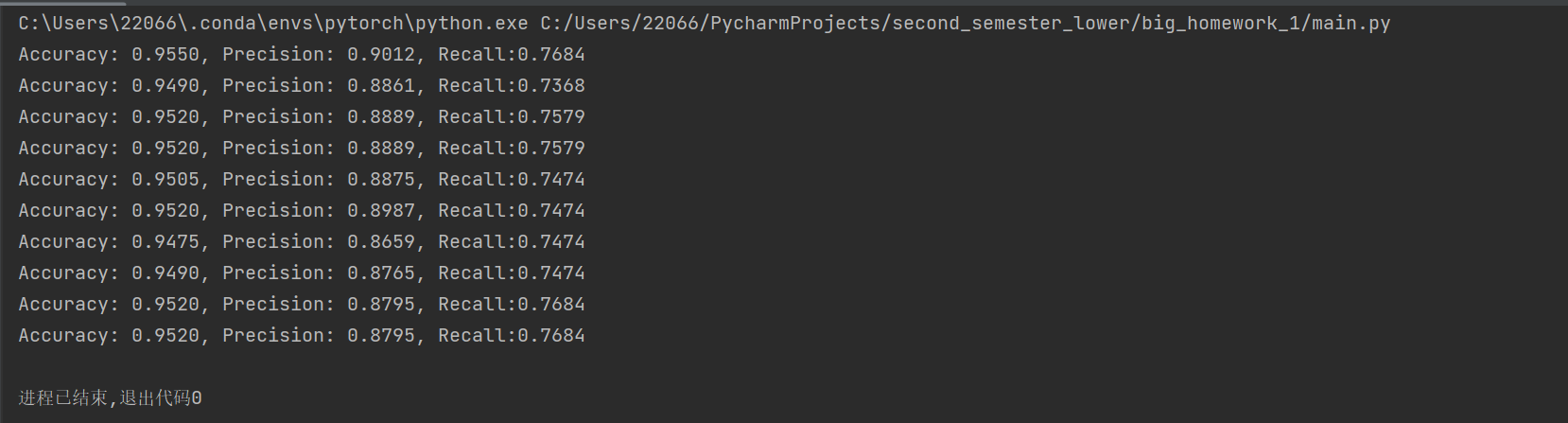
|  |
| --- |
|  |

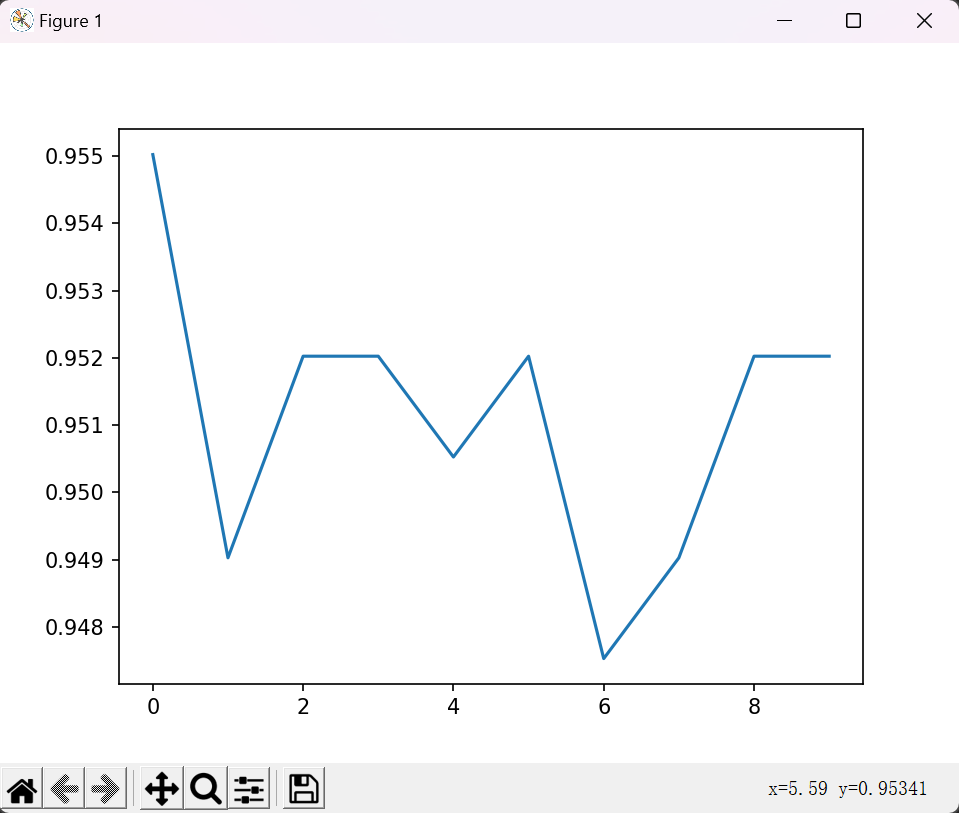
删去最后两列数据，观察准确率的情况（其他参数均一致）：

|  |
| --- |
|  |

可以看出，通过减少数据维度，对于准确率的影响并不是很大，但在一定程度上可以简化计算，让网络更加简单。

**最后，通过不断修改网络层，以及混合使用激活函数，我找到了一个较为稳定且高准确率的模型，其正确率在0.95上下变动,如图所示**



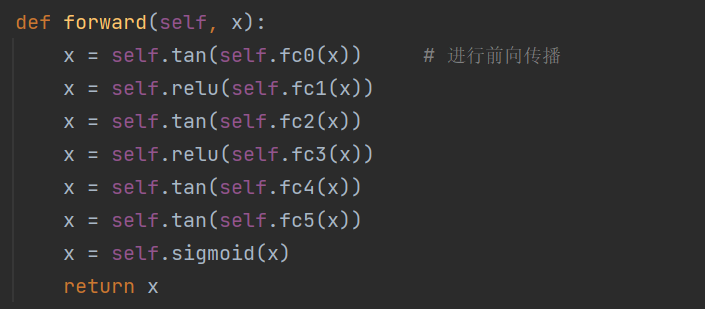


其参数如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 网络层数 | 隐层神经元个数 | 损失函数 | 训练轮数 | 学习率 | 激活函数 | 优化算法 |
| 6 | 60 | BCE\_loss | 1000 | 0.001 | Tanh、Relu、sigmoid | Adam |

此外还加入了正则项：weight\_decay=0.001,

其中，前向传播的设计如下：



**3.心得体会：**

在不断优化代码的过程中，让代码更加地简洁，高效，也更加感受到Python对于模块化设计的思想。

同时，对于数据的预处理占了本次实验的非常大的一部分。数据的处理会直接影响模型的稳定以及输出效果等。其中如何转化转化标准则又是一个问题，通过检验，我发现了一些转化的标准不能提高模型的准确性，反而会有负面的影响。

并且，通过观察得到，各个属性之间的数据的大小分布存在不均衡的特点，因此，在未归一化之前，我训练出来的模型效果很糟糕，几乎全部都预测为0。通过其他同学的善意提醒，我才意识到对数据的归一化非常重要。在归一化之后，模型输出的效果相对来说就优化了许多。