**武汉大学国家网络安全学院**

**实验报告**

**课程名称 舆情分析**

**专业年级 2019**

**姓 名**

**学 号**

**协 作 者 无**

**实验学期 2021-2022　 学年 第一 学期**

**课堂时数 课外时数**

**填写时间 2021 年 10 月 20 日**

|  |
| --- |
| **实验介绍** |
| **【实验名称】：舆情分析实验1 文本主题分类** |
| **【实验目的】：**   1. 熟悉简单的文本处理流程和神经网络文本分类模型 2. 初步了解深度学习框架pytorch的使用   **【实验环境】：**   1. Windows 10 家庭中文版 2. Anaconda Navigator 1.9.12 3. conda 4.8.3 4. Python 3.8.11 5. torch 1.9.1 6. torchtext 0.10.1 7. matplotlib 3.4.3   **【参考文献】：** [1] pytorch中的relu,sigmiod,tanh等激励函数(激活函数） <https://blog.csdn.net/weixin_44912159/article/details/104994863>  **[2]** 写给程序员的机器学习入门 (三) - 线性模型，激活函数与多层线性模型  https://www.cnblogs.com/zkweb/p/12761743.html |
| **实验内容** |
| **【实验方案设计】：**  实验步骤：   1. 访问原始数据集   首先通过torchtext库访问原始数据集，从而获得标签与文本的元组。标签表示该文本所属的主题。比如下图文本属于标签3，即属于Business主题类。     1. 准备数据处理管道   对原始数据集进行分词处理，然后构建词汇表，并将文本字符串转换成使用整数序列。如下图所示，序列中每个整数表示文本字符串中相应词在词汇表中的序号。     1. 数据批处理函数和加载器   这一步骤中使用了pytorch中的一个重要接口DataLoader，DataLoader根据batch大小、是否shuffle以及collatte\_fn等设置，将传入的数据集封装成可供训练的张量。     1. 定义模型   在pytorch中，自定义的模型一般继承自torch.nn.Module类，并且需要对其中的\_\_init\_\_和forward方法进行自定义。\_\_init\_\_定义了模型中的每个层，forward实现了前向传播。在模型中不同节点对输出的影响不同，因此每个节点应该拥有相应的权重，即对输出的影响程度，在实验中init\_weights方法就是用于对模型权重进行初始化的。下图为一个单层线性模型的定义。     1. 固定随机数种子   默认情况下模型的参数是随机初始化的，但在随机的情况下结果会有较大不确定性，因此需要通过固定随机数种子来保证每次初始化的参数是一致的。     1. 启动一个实例   初始化维度等参数，构建模型。     1. 定义用于训练模型和评估结果的函数 2. 拆分数据集并运行模型   原始数据集中没有显式的验证集，因此需要将原始数据集拆分为训练集和验证集，训练集与验证集之比为0.95:0.05。   1. 使用测试数据评估模型 2. 测试一条新闻 3. 使用LSTM模型   长短期记忆网络，即LSTM，是循环神经网络的一种。LSTM能够记住较长期的依赖关系，在文字语义的理解这种需要联系上下文的工作上有着较好的效果，因此该模型常用于自然语言处理领域中。整个模型的定义、初始化、训练以及测试步骤与上述线性模型类似，关键在于定义模型的时候加入了LSTM层。如下图所示，LSTM模型中加入了LSTM层。     1. 将教程中的单层线性模型改成双层线性模型，并比较两者的准确率   下图为单层线性模型的定义    下图为双层线性模型带tanh激活函数的定义    两个模型的准确率折线图如下所示    绘图所用代码    在测试集上的准确率：单层线性模型为0.908，双层线性模型为0.899。   1. 比较三组激活函数   这里选择比较tanh、relu、sigmoid三种激活函数。  首先是tanh层，位于第一层线性层后方，模型的第二层线性层没有使用激活函数。如图所示为添加tanh激活函数后的模型定义。    relu激活函数和sigmoid激活函数都与tanh激活函数的使用方法类似，如下图所示，分别为添加了relu激活函数和sigmoid激活函数后的模型定义。      然后分别训练模型并得到每个epoch的准确率数据，再将这些数据用折线图的方式绘制出来，以比较三种激活函数的优劣，与上文中比较单双层线性模型使用的方法一致。   1. 比较四组超参数   首先比较超参数epoch，分别取值5、10、15、20，对比其正确率。再比较超参数学习率，分别取值1、2、5、8，同样对比其正确率。最后比较超参数batch大小，分别取取值4、16、32、64，对比正确率。比较结果请见实验结果分析内容中。   1. 比较self.embedding.weight.requires\_grad不同取值   self.embedding.weight.requires\_grad=False意味着不需要对embedding参数进行更新，不对模型中词向量的权重进行调整，这样可能会使得模型性能下降。比较结果请见实验结果分析内容中。  **【实验结果分析】：**  激活函数可以让模型学习非线性数据，以满足一些线性模型所无法满足的情况。在没有激活函数的情况下，多层线性模型与单层线性模型没有明显区别，使用激活函数可以使得模型更加符合现实情况。总体而言，在没有激活函数之前，输入输出呈线性关系，而添加了激活函数之后就变为非线性关系，这增加了模型的性能。实验中所用到的tanh、relu、sigmoid激活函数图像如下图所示。    实验中这三种激活函数的比较如下图所示。其中可以发现，最后收敛时sigmoid激活函数效果较好，并且tanh激活函数中间起伏较大，相对而言relu函数较为平稳。整体而言在收敛时三种激活函数差别不大，sigmoid和tanh激活函数几乎没有区别。    下图所示是超参数epoch分别取值5、10、15、20时得到模型训练的正确率，可以发现总体而言差别不大，正确率基本维持在0.91左右，相对而言epoch取值为10有着较好的效果。    下图所示是超参数学习率分别取值1、2、5、8时得到的正确率数据，可以发现模型收敛时当学习率为5或者8时正确率较高，学习率为5的波动相对而言更小一点，因此总体而言认为学习率为5较优。    下图所示是超参数batch大小分别取值4、16、32、64时的正确率数据，可以发现模型收敛时batch为4时正确率较高，不过batch越小数据集分得越细，训练模型所需时间也越长。    在本实验中，self.embedding.weight.requires\_grad作用较大，该参数为True或者False决定了embedding参数是否在训练过程中得到更新，通过实验对比认为在训练时更新embedding参数能够得到性能较优的模型。 |
| **【实验总结】：** |
| **评语及评分（指导教师）** |
| **【评语】：**      **评分：**  **日期：** |

**附件2：**

**实验报告说明**

**1．实验名称：**要用最简练的语言反映实验的内容。

**2．实验目的**：目的要明确，要抓住重点。

**3．实验环境**：实验用的软硬件环境（配置）。

**4．实验方案设计（思路、**步骤和方法等**）**：这是实验报告极其重要的内容。包括概要设计、详细设计和核心算法说明及分析，系统开发工具等。应同时提交程序或设计电子版。

对于**设计型和综合型实验**，在上述内容基础上还应该画出流程图、设计思路和设计方法，再配以相应的文字说明。

对于**创新型实验**，还应注明其创新点、特色。

**5．实验结果分析：**即根据实验过程中所见到的现象和测得的数据，进行对比分析并做出结论（可以将部分测试结果进行截屏）。

**6．实验总结：**对本次实验的心得体会，所遇到的问题及解决方法，其他思考和建议。

**7．评语及评分：**指导教师依据学生的实际报告内容，用简练语言给出本次实验报告的评价和价值。