**中山大学计算机学院**

**人工智能**

**本科生实验报告**

**（2022学年春季学期）**

课程名称：Artificial Intelligence

|  |  |  |  |
| --- | --- | --- | --- |
| 教学班级 | **计科2班** | 专业（方向） | **计算机科学与技术** |
| 学号 | **20337263** | 姓名 | **俞泽斌** |

# 实验题目

在给定文本数据集完成文本情感分类训练，在测试集完成测试， 计算准确率。

要求

• 文本的特征可以使用TF或TF-IDF （可以使用sklearn库提取特征）

• 设计合适的网络结构，选择合适的损失函数利用训练集完成网络训练，

并在测试集上计算准确率

• 需提交实验报告+代码

• 实验报告应包含损失的可视化展示，以及学习率对准确率影响的可视化展示

# 实验内容

1. 算法原理

首先是读入数据的部分，这里主要调用的是re的库，主要操作就是逐行

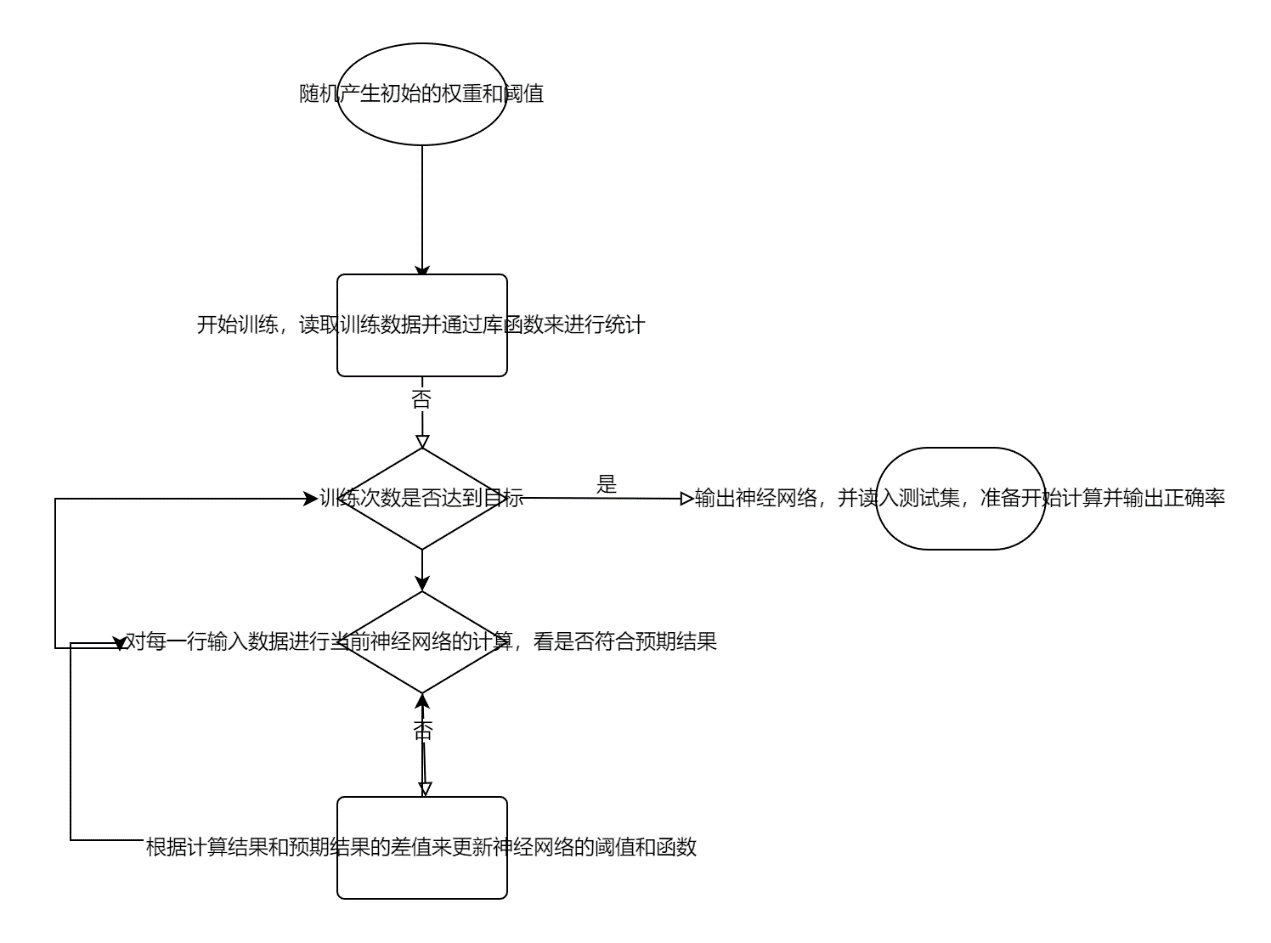
读入数据，然后对于每行数据按照相同的格式进行分类，加入到两个列表中，分别是整体语句以及情感程度。

然后到了通过向量的方式来展示语句的方面上来了，具体的也是调用了sklearn里的TfidfVectorizer（）函数中的fit和transform函数，可以通过这两个函数将语句统计并标准化。

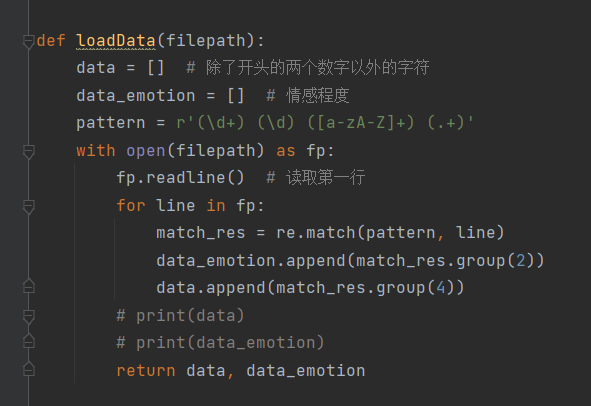
至于神经网络，该算法主要的原理就是感知器的理解上了， BNN神经网络，是由很多个单个神经元所组成，每一个神经元都是通过输入，然后进行一些操作，与阈值相比较，然后得出输出，这次的实验中我还设计了一个隐层，增加了神经网络的复杂性，希望能得到更好的结果。

这里对于这个神经网络，我们一开始的各项的权重包括阈值之类的都是通过随机数来生成的，然后通过train函数不断读取训练集中的数据，不断地修改相关的权重和阈值，最终得到我们新的神经网络，然后读入test集合，对于每句话进行神经网络的判断，与一开始的第一项的情绪数字进行比较，从而得出是否相同，然后记录正确数，计算正确率。

1. 伪代码



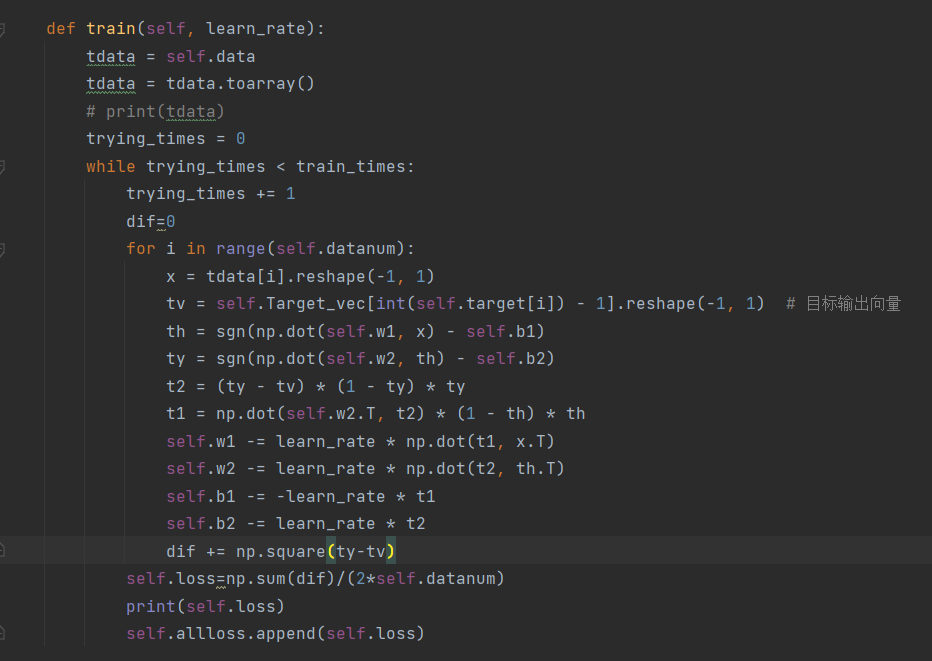
1. 关键代码展示（带注释）



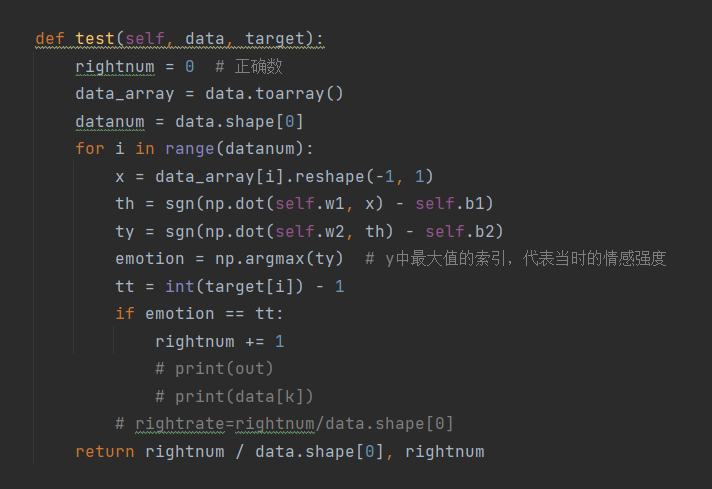
一开始是一个关于读入数据的方面，这里调用的是re这个库里的match函数，因为一开始的数据都是一串字符，需要先进行统计之类的操作，然后才能交给sk-learn里面的库进行操作，得到需要的向量之类的数据。



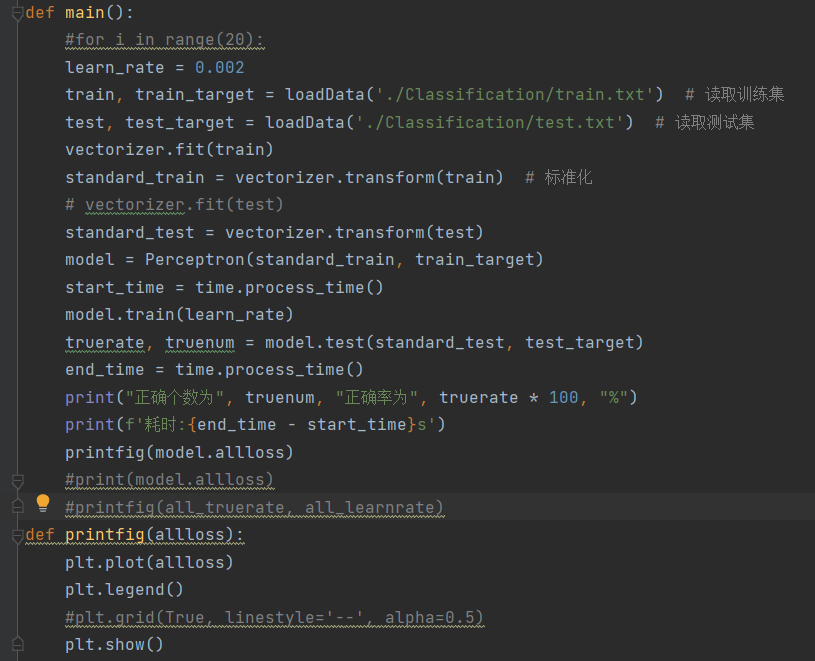
首先定义了一个Perceptron的类，来保存我的模型，模型用的是两层的神经元，然后保存了一下两层神经元的分别的权重和阈值。



接着是具体的有关训练时的操作，其实主要参考的还是ppt上的训练操作，先把得到的数据全都放到-1到1的范围下，然后通过向量的操作来得到对应的解，也就是当前神经网络的输出，与目标的向量来进行比较，并且通过学习率等操作来修改更新我们两个权值和阈值。至于损失函数，我在这里用的是mse，也就是均方误差的方法，将当前神经网络的输出，与目标的向量的平方差值通过numpy库内的sum函数来得到并且/2n得到相对的数值，方便之后输出

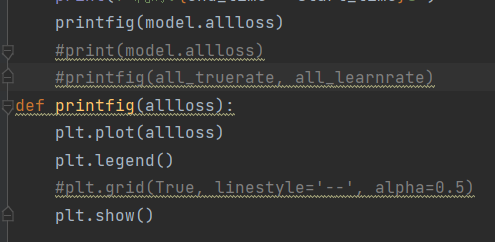


这是测试函数，也就是对测试集来工作的函数，主要的思想其实类似于训练集了，就是通过一些操作来将输入的数据经过神经网络得到输出，然后与本来的数据内进行比对，如果比对成功，就说明该神经网络在当前的输入为正确的，正确数目加一，同时最后得出正确率来输出。

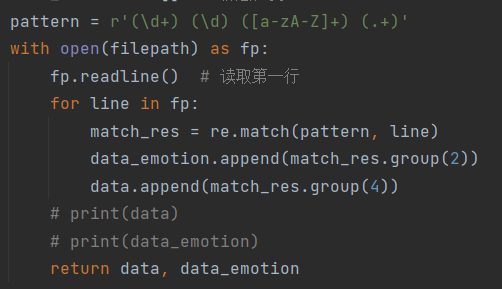


这里就是主题的main函数了，其实有了上面的基础，main函数中最大的功能可能就是sk-learn这个库里的函数调用了，毕竟通过输入得到的都是字符，需要sk-learn库中的函数进行由字符到向量的转化。然后通过产生一个model对象来先通过训练集进行训练模型，完善了里面的权值和阈值，然后用测试集来测试，并且统计里面的正确数和正确率。

1. 创新点&优化（如果有）



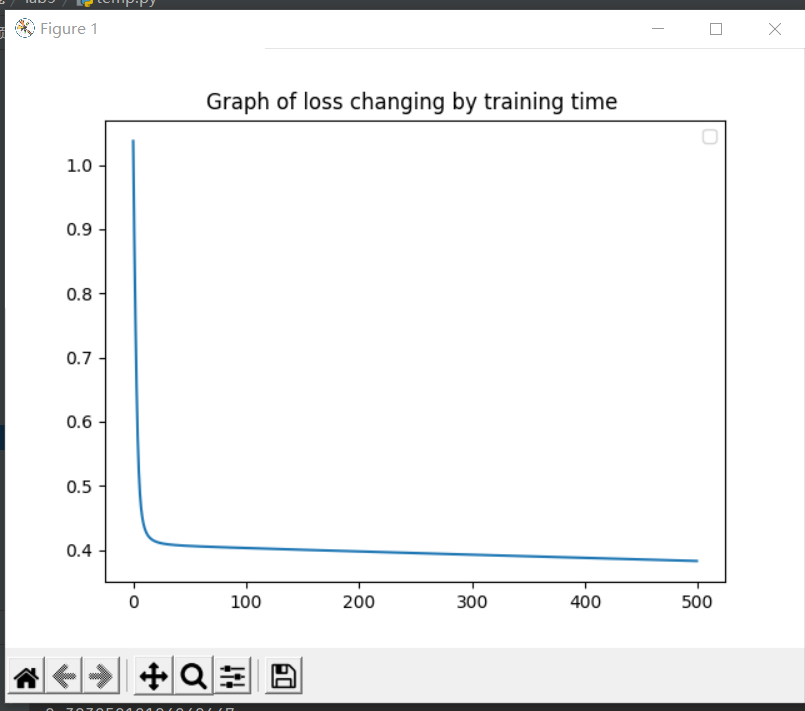
调用了plt库来实现具体的关于损失的图像实现，能够更直白地看出随着训练次数的增加，损失的变化。



采用re库来对数据进行预处理，使得数据可以先行统计里面的参数和大小，方便之后调用sk-learn库以及后续操作

# 实验结果及分析

1. 实验结果展示示例（可图可表可文字，尽量可视化）



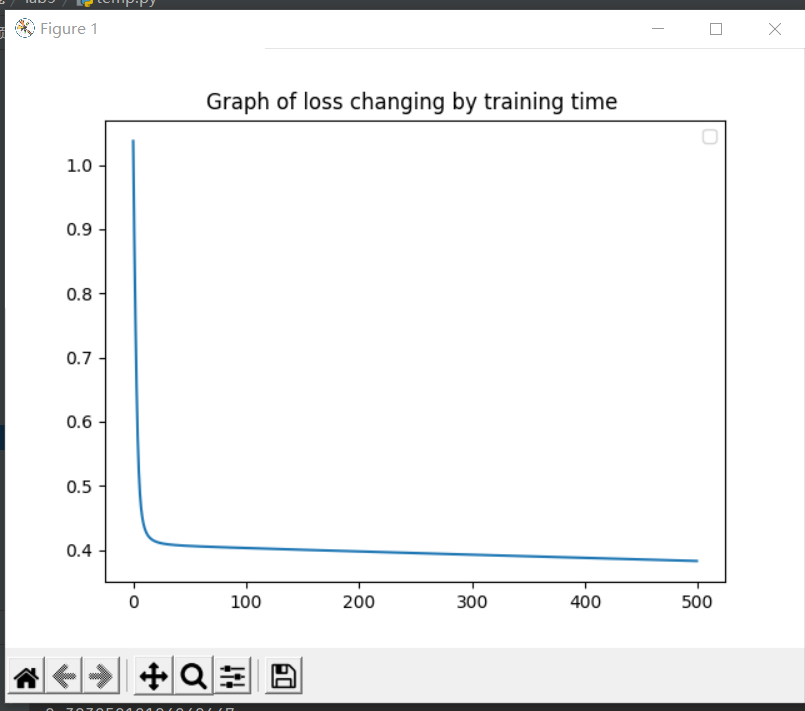
首先是损失关于训练次数的增长而减小的图，可以看到，在较短的训练时间内，损失已经开始慢慢收敛，收敛到基本在0.4左右。



并且当时的正确率基本在34左右，测试的为测试集里的数据。此时学习率为0.002

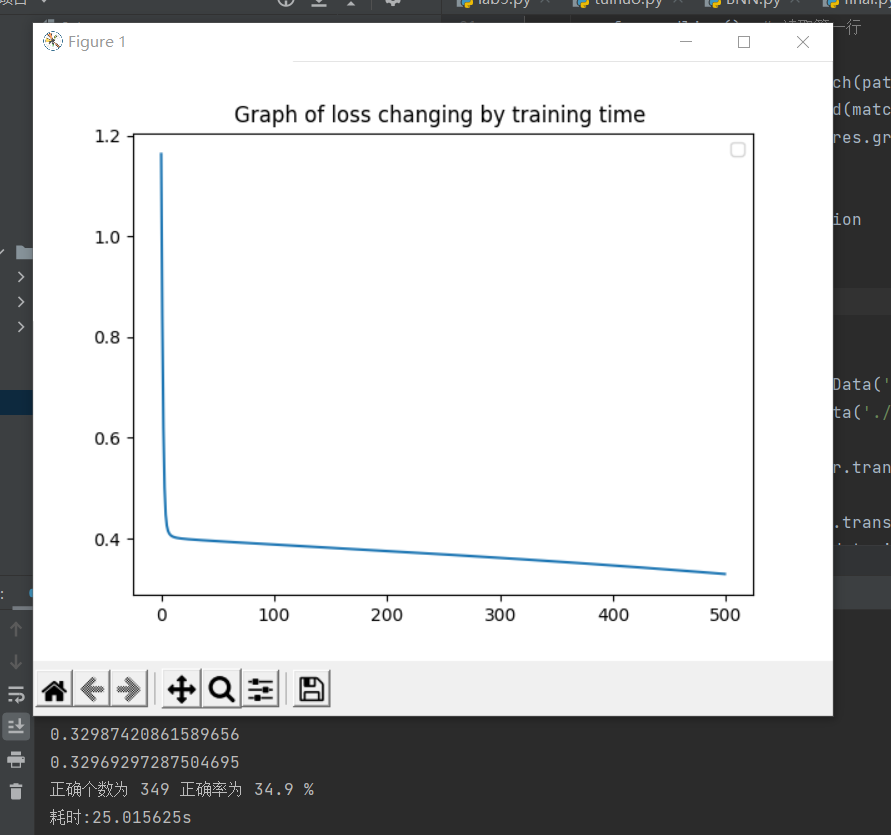
2. 评测指标展示及分析（机器学习实验必须有此项，其它可分析运行时间等）

当学习率为0.002的时候



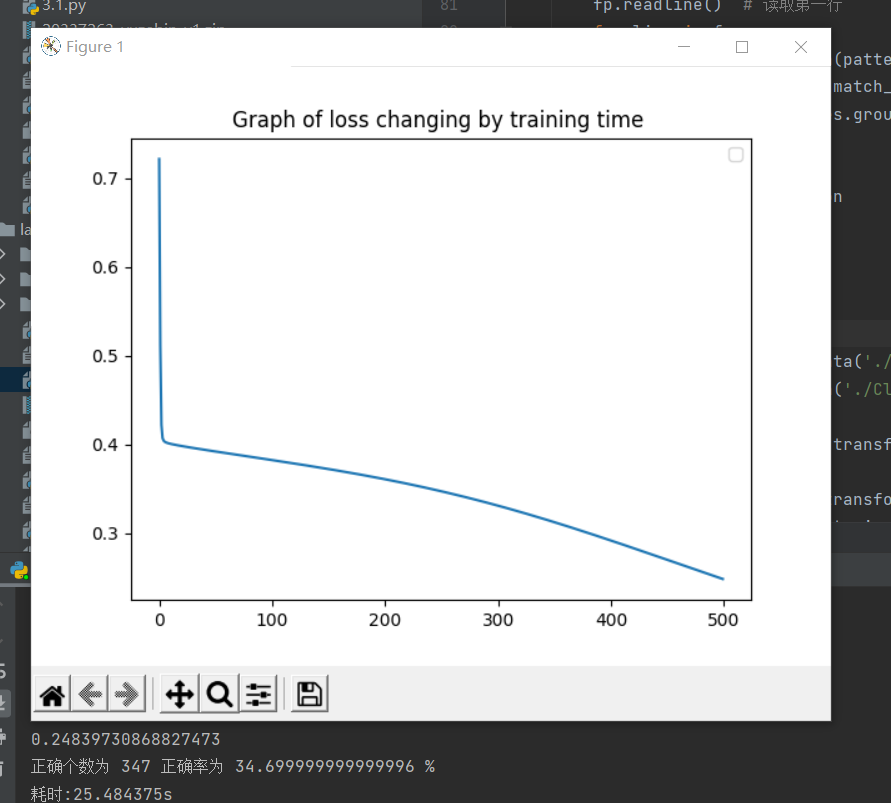


当学习率为0.005的时候



可以看到，收敛的更快，而且正确率有一点点的增长。

当学习率为0.01的时候



正确率又有一点点增长，且收敛变慢，可能需要进一步的训练。

所以综上，在学习率较低的情况下，增长一点学习率可能会提高一点点正确率

**|-----------如有优化，请重复1，2，分析优化后的算法结果-----------------------|**

# 参考资料

**https://blog.csdn.net/qq\_37373203/article/details/82791200?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522165415628416781432996654%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request\_id=165415628416781432996654&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~sobaiduend~default-2-82791200-null-null.142^v11^control,157^v13^control&utm\_term=python%E5%8D%95%E5%B1%82%E6%84%9F%E7%9F%A5%E5%99%A8%E5%AD%A6%E4%B9%A0%E7%8E%87&spm=1018.2226.3001.4187**