人工智能导论实验三.实验报告

2018011324

尤艺霖

1.简介

实验要求利用给定的4570篇新闻文章进行情感分类的实验。我选择实现“分类问题”的版本。

1.1.提交文件结构

data目录下存放了验证集、测试集、训练集的原始数据

model目录下存放了训练好的模型，并附带了足以测试的数据和代码

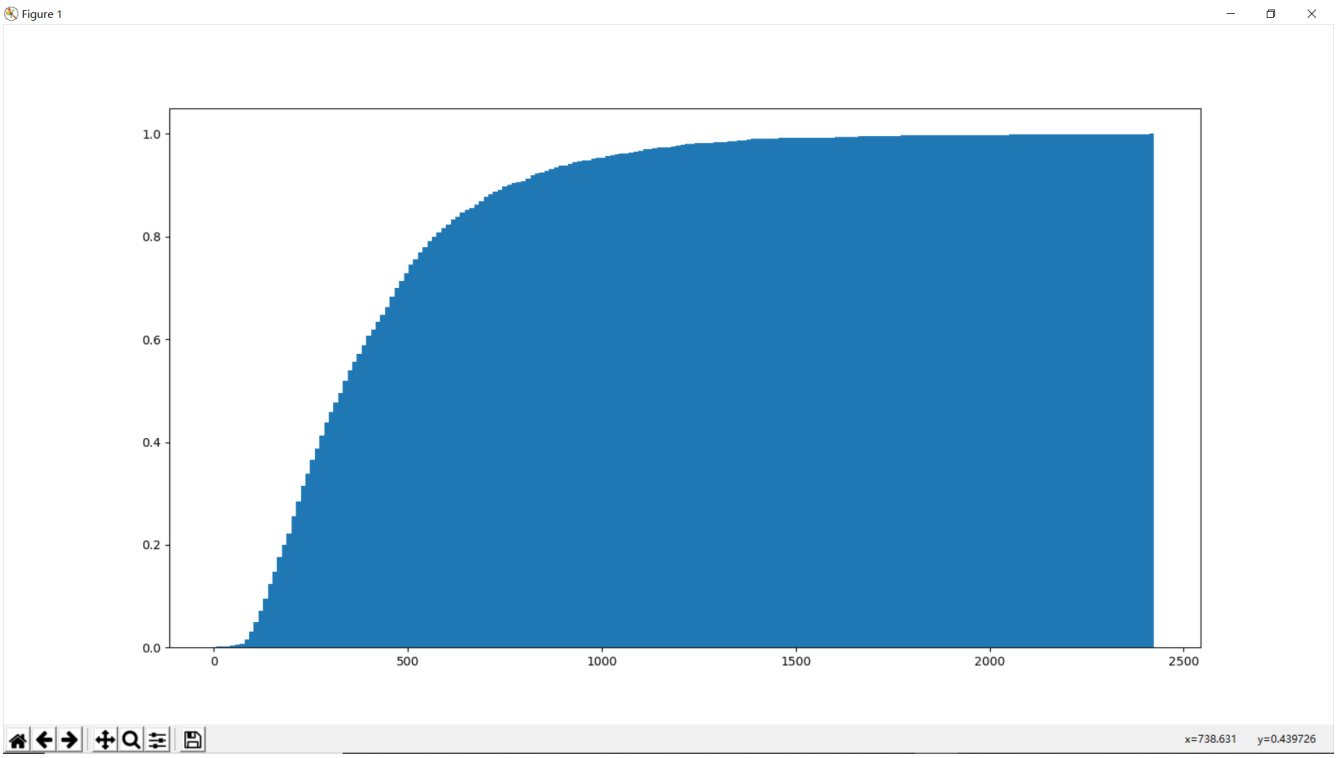
src目录下存放了所有源文件

如果需要完整运行整个训练流程，需要将这三个目录的内容合并到同一个目录下

2.数据处理

首先下载**pdf**中网址上的一个词向量（mixed-large-word 300d），对于所有不在词向量中的词，直接删除

然后观察每条新闻的长度，从而选择用于实验的合适长度。训练集中的词统计结果如下：



从统计结果可以看出，**80%**的新闻长度小于**500**词，故选择**500**作为模型的输入长度，少则补零，多则截断。

然后将所有的新闻（训练集和测试集）全部预处理成词向量形式存储，分类也全部处理成**one-hot**形式。

取出测试集和训练集各自的前**300**条组成交叉验证集。经统计，验证集、训练集、测试集的类别分布是接近的。

3.模型设计

3.1.CNN

采用论文中给出的模型，从卷积层之后先经过激活函数，然后经过池化层，然后通过Dropout和全连接层得到输出

卷积核选择和论文中一致的安排，(3,300)/(4,300)/(5,300)三种卷积核各100个。

激活函数选择ReLu，Dropout的参数选择和论文一致的0.5

3.2.RNN

选择直接使用**pytorch**中给出的双向LSTM作为Rnn层，然后用一个全连接层转到输出。

由于词向量为300d，隐藏层大小选择150

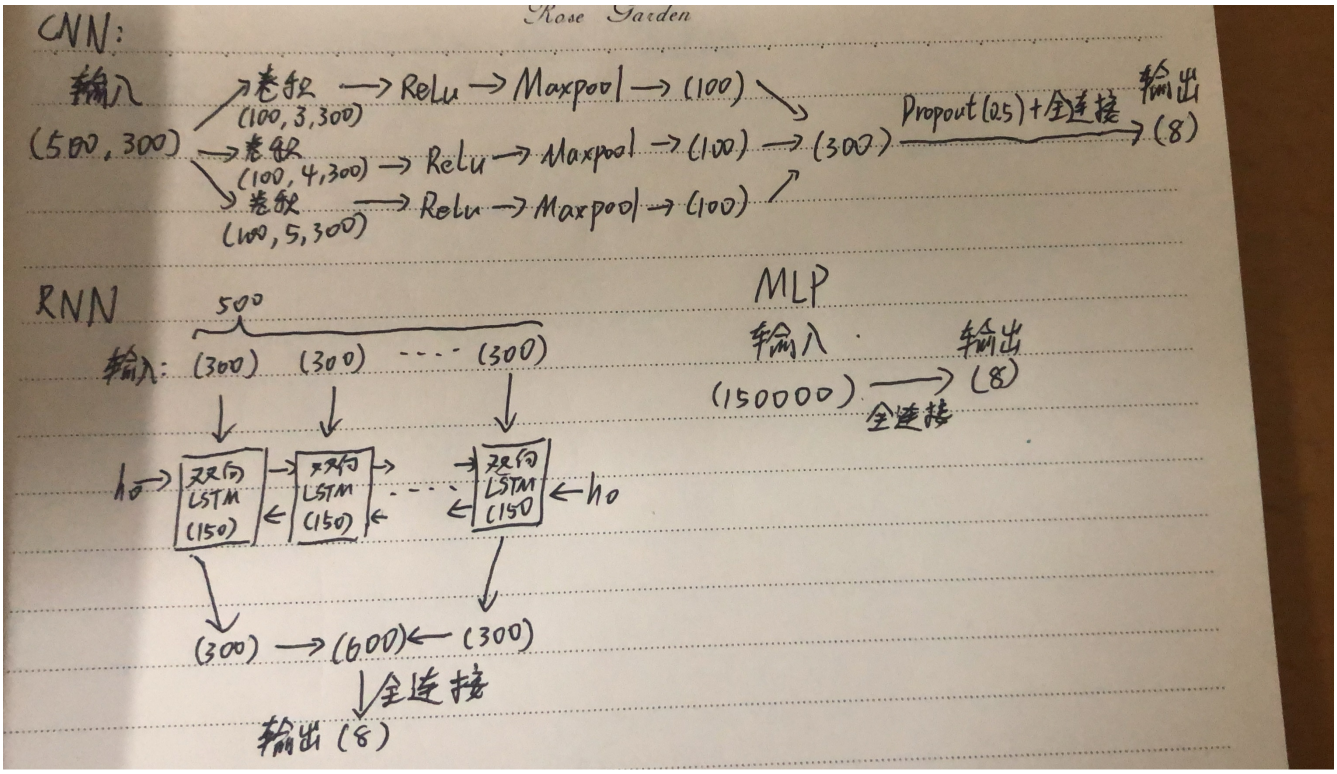
由于是双向的，全连接之前把头尾两个输出先拼起来

3.3.MLP

MLP作为baseline，只包含一个单独的全连接层，将输入的词向量平摊在一维上，通过全连接转成长度8的输出。

3.4.三种模型结构图

三种模型中具体的尺度和一些确定的系数在图中可以体现。



4.实现细节

首先考虑到本机显卡支持，加上**Pytorch**上使用显卡很容易，所有的训练都在显卡上完成。

模型的共同细节如下：**loss**选择**交叉熵**，优化器选择**Adam**，**weight_decay**选择**1e-5**，**batch_size**选择**45**

同一模型的不同版本改变的只有**学习率**这一项，其他参数都是固定的

判停选择**early-stopping**，如果连续**12**个Epoch没有使得验证集**loss**降低**1%**则停止训练

保存的策略是每当有新模型能使验证集**loss**降低**1%**就保存，取最后一个作为结果

最终评价的时候选择的是宏平均的F1值。而相关系数在分类问题上作用不大

5.实验结果

5.1.实验环境

```
显卡：GTX1060 6G
内存：16G
```

```
Python 3.7.5
Torch 1.5.0
Numpy 1.18.1
Scipy 1.4.1
Matplotlib 3.1.3
Cuda 10.2.150
```

5.2.实验步骤

批阅者如果需要运行代码，请参见**README.txt**，这里仅列出我自己实验的过程

首先下载pdf中网站上的词向量，下载其中**Mixed-large 300d**，下载下来的文件名为**sgns.merge.word**

然后从**sinanews.train**和**sinanews.test**中各取出**300**条，合成**sinanews.valid**作为验证集

然后运行**vectortodict.py**，将词向量文件转成python的字典，存放在**wordvec.json**中

然后分别运行**test_label.py**，**train_label.py**，**valid_label.py**，进行数据预处理

然后分别用不同的学习率去运行**main_cnn.py**，**main_rnn.py**，**main_mlp.py**，得到不同模型

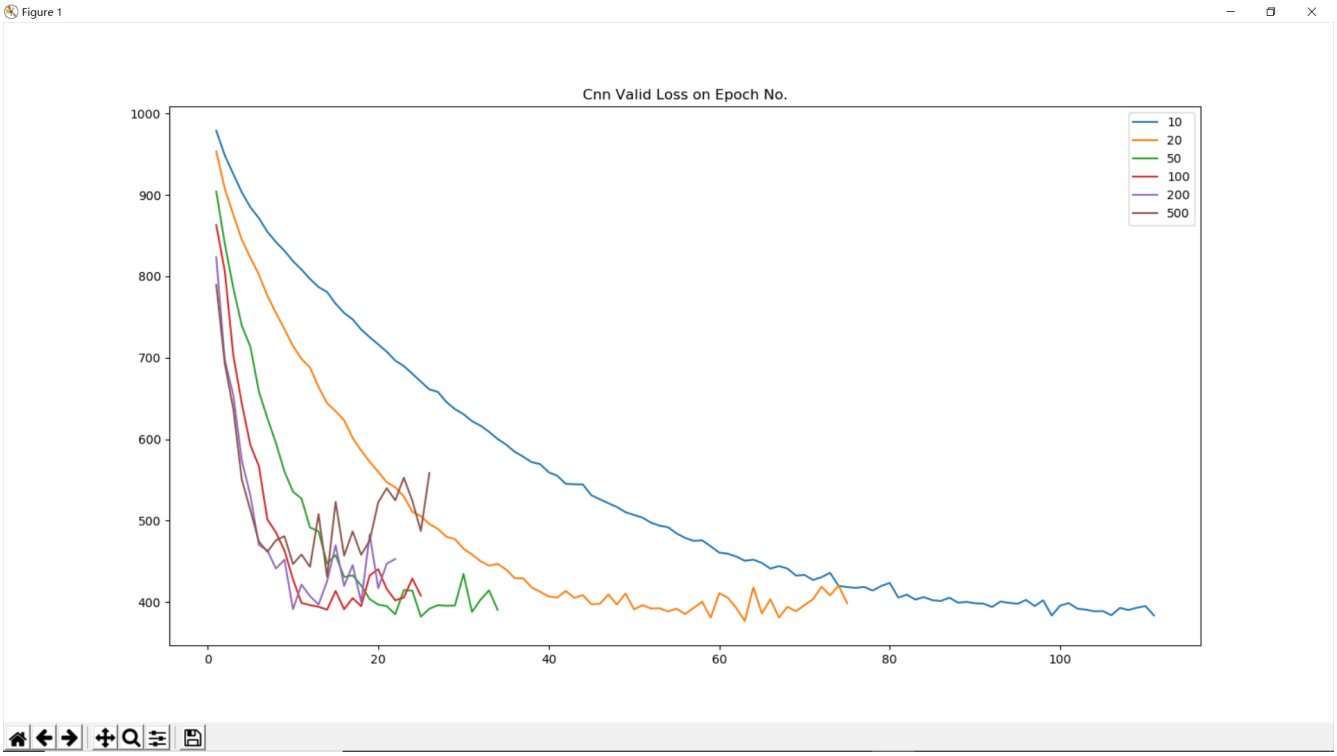
然后在每个训练出的模型上运行**load_and_test.py**，得到模型在测试集上的表现。

5.3.实验结果

5.3.1.CNN

学习率	正确率	Macro F1 Score	相关系数
0.0001	0.5983	0.2958	0.5060
0.0002	0.5893	0.2933	0.4975
0.0005	0.5916	0.2998	0.5005
0.001	0.5961	0.2953	0.4894
0.002	0.5794	0.3223	0.4761
0.005	0.5575	0.2936	0.4487

对于不同的学习率，保留了训练过程中验证集的**loss**，作折线图如下：



其中不同学习率对应的折线标注在右上角（单位：1e-5）

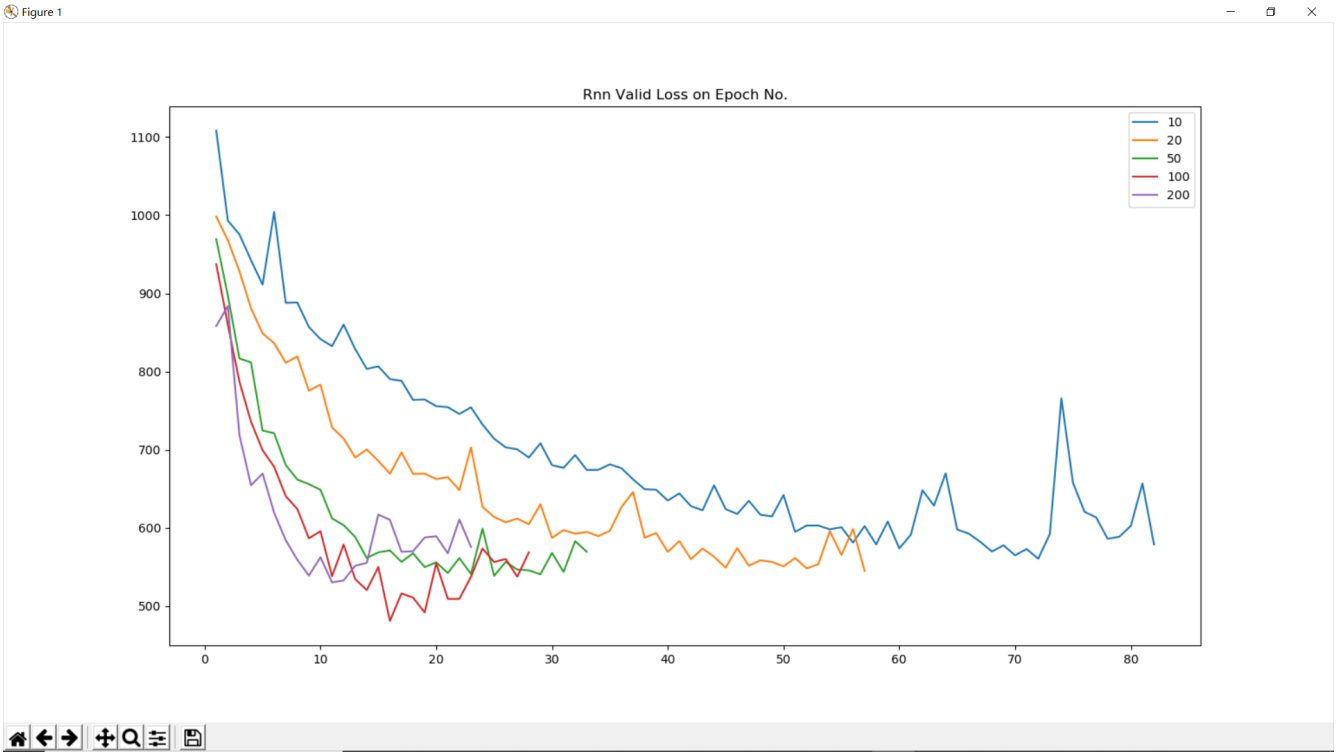
注：保存的**loss**包括了最优网络到终止训练的12个Epoch，故可以看出过拟合趋势，以下的两个折线图也一样。

CNN的折线图很符合正常训练的情况，棕色线也很明显的体现出了过高学习率的问题

5.3.2.RNN

学习率	正确率	Macro F1 Score	相关系数
0.0001	0.5575	0.3103	0.4684
0.0002	0.5377	0.3030	0.4636
0.0005	0.5202	0.3141	0.4355
0.001	0.5561	0.3192	0.4517
0.002	0.5013	0.2948	0.4315

对于不同的学习率，保留了训练过程中验证集的**loss**，作折线图如下：

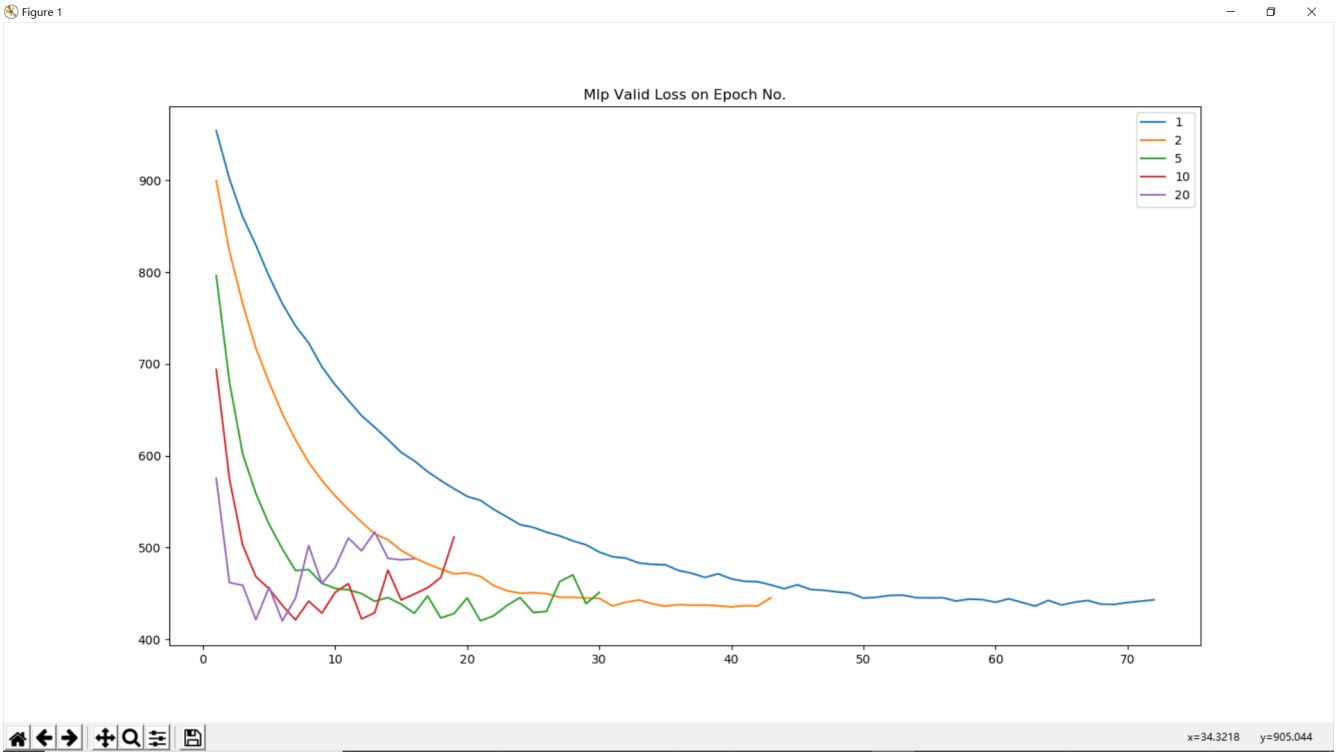


RNN的折线图震荡非常厉害，但是整体仍然符合正常训练的规律

5.3.3.MLP

学习率	正确率	Macro F1 Score	相关系数
0.00001	0.5736	0.2113	0.4356
0.00002	0.5790	0.2141	0.4421
0.00005	0.5691	0.2117	0.4348
0.0001	0.5718	0.2247	0.4263
0.0002	0.5651	0.2171	0.4339

对于不同的学习率，保留了训练过程中验证集的**loss**，作折线图如下：



可以观察到MLP最明显的一点就是需要很小的学习率才可以让**loss**得到一个好结果。已经实验过更高的学习率往往会直接使得**loss**无法收敛到理想的情况。

6.问题思考

6.1.训练停止相关

我采用了**early-stopping**的策略来判断是否停止训练，具体为如果连续12个Epoch无法让验证集的**loss**比之前的最优解降低**1%**，则停止，并使用之前保存的最优解模型作为输出。同时我还设定了200的Epoch上限，但实验过程中没有到达过。我认为防止过拟合这样是必须的，因为这次给出的数据集太小，而网络的参数却很多，如果让网络训练到固定次数停止，几乎一定是会过拟合的，必须要用一些条件来控制停止。

6.2.初始化相关

我没有手动初始化，因为Pytorch对于大部分模块都自带了初始化，通常是**kaiming-uniform**

对于LSTM的初始值，我直接用了默认的零向量。

6.3.防止过拟合

在模型的角度来看，防止过拟合一方面需要在优化器中加入**weight_decay**，另一方面在模型本身也可以加入一层Dropout。而从训练的角度来看，选择合适的学习率和合适的停止策略，可以更好的防止过拟合。

6.4.三个模型优缺点

首先选择三个模型中各自正确率最高的一组进行对比：

模型	学习率	正确率	Macro F1 Score	相关系数
CNN	0.0001	0.5983	0.2958	0.5060
RNN	0.0001	0.5575	0.3103	0.4684
MLP	0.00002	0.5790	0.2141	0.4421

相关系数这一栏在分类问题上意义不大，只是走个流程，不比较。

准确率方面，从高到低依次是CNN，MLP，RNN

从F1 Score方面来看，从高到低依次是RNN，CNN，MLP。且MLP与另外两者有较大差距

这两个参数的排名上出现较大差距的原因，我认为是由于数据本身分布导致的。在下发的数据集中，各个分类占比非常不合理，而F1 Score可以体现出在占比较小的类别上，RNN是做的更好的。而MLP更倾向于在各种情况下都输出占比较大的分类从而获得更高准确率。可以推测，MLP学到了更多的数据集分布，而RNN学到了更多的新闻本身的特征。而CNN既有较高正确率，也有不低的F1 Score，在本测试集上发挥的确实更好。但如果更换成一个更健康的数据集，那么RNN的潜力是更大的。

7.心得体会

在本次实验中我只调节了学习率这一个参数，其他都是固定的。从上面几张折线图可以看出来，学习率越大，终止的就越早；但学习率太大就无法收敛到合适的位置；学习率越小，折线看起来越正常，但需要更多的Epoch。

同时本次实验给出的数据集并不理想，一方面是“愤怒”几乎占到了总数的一半，另一方面也存在个别分类数量极少。同时数据集的大小虽然偏小，但为了在个人电脑上运行也不好再扩大。如果能优化一下类别结构，最终的效果可能会更好。