情感分类大作业实验报告

孙迅　2019011292

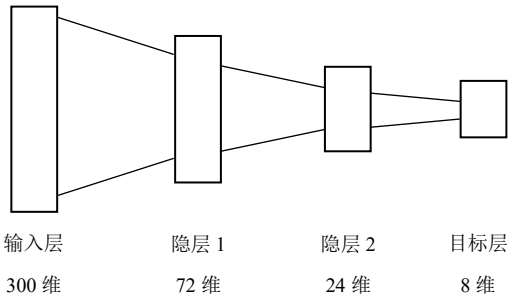
一、模型概述

在本次大作业中，我使用 PyTorch 平台，分别采用了全连接神经网络、卷积神经网络、循环神经网络和朴素贝叶斯分类为模型进行实验。在代码层面，我针对大作业任务的特点在 `interface.py` 中封装了一个 `Model` 抽象类，从而大量减少重复代码，并提高调试效率。

在这一部分，我将对所采用的模型进行简要概述。

全连接神经网络

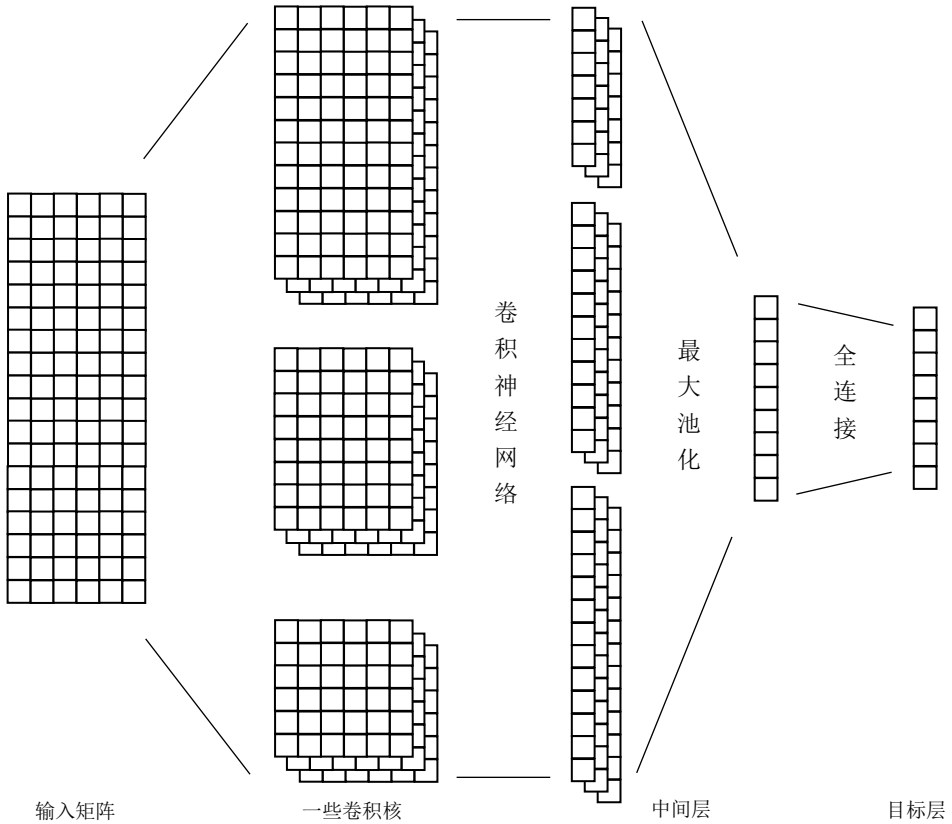
全连接神经网络的结构如下图所示：



对每一份样本，将其中的词语转化成词向量后取平均值，得到该样本的特征向量，传入神经网络并进行训练。其中，每个全连接层均用 tanh 函数进行激活。

卷积神经网络

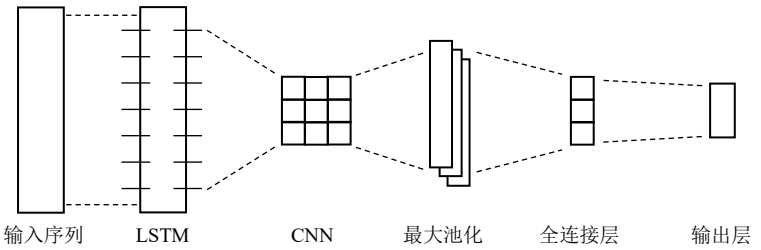
卷积神经网络的结构如下图所示：



对每一份样本，将其中的词语转化成词向量后，拼接为一个输入矩阵，传入神经网络进行训练。其中，卷积层和最后的全连接层均使用了 tanh 函数进行激活。

循环神经网络

这里使用 LSTM 实现循环神经网络，结构如下图所示：



将文本表示为词向量序列后，传入双向 LSTM 网络，转换得到输出序列。再将这个输出序列传入 CNN 中进行特征提取，经最大池化后传入全连接层，转换得到最终的输出层。

朴素贝叶斯分类

作为对照，我还采用朴素贝叶斯分类的模型进行了实验。

文本编码采用词袋模型，用词语的词频表示文本。

训练时，将某类样本出现的频率作为先验概率，某个词在某类样本中出现的频率作为后验概率。计算后验概率时采用拉普拉斯平滑，具体而言，设词w在样本i中出现n次，样本i共有N个词，词汇表中共有V个词，则词w在样本i中出现的概率为

$$p(w|i) = \frac{n+1}{N+V}$$

分类时，从测试样本中滤去所有词汇表以外的词语后，用

$$P(i)\prod P(w_j|i)$$

估计预测为第i类的可能性。

在实际处理过程中，将所有概率值取对数后进行运算，以防出现下溢问题。

二、流程分析

整体的流程为：读取→编码→训练→测试→评价。

在这一部分，我将对流程中的一些具体细节进行阐述。

文本编码

在本次任务中，我制定了三种编码策略：转化为词向量序列（适用于 CNN、LSTM）、转化为词向量后取平均值（适用于 MLP）以及转化为词汇表中的序号（适用于 Bayes）。其中，词向量使用了由[1]训练好的结果。

此外，在 CNN 和 LSTM 中，考虑到统一句子长度的需求，针对这两种模型设置了句子长度的参数，并采用过长截去、不足补 0 的方法进行了处理。

损失函数

对于每份样本的标签分布，使用 softmax 归一化后，将其作为训练的目标。因此，采用*均方误差*作为损失函数。

学习速率

为了达到更好的收敛效果，在本次实验中对学习速率进行了动态调整。具体而言，每轮迭代完成后让学习速率乘以一个常数，从而使学习速率指数下降，达到先快速收敛后逐步逼近的效果。

三、结果展示

指标		MLP	CNN	RNN(LSTM)	Bayes
准确率		60.5%	56.1%	58.6%	58.3%
F1	宏平均	0.346	0.201	0.313	0.289
	微平均	0.605	0.561	0.586	0.583
	加权平均	0.570	0.463	0.545	0.522
相关系数		0.611	0.572	0.586	0.556

其中，F1 值采用宏平均、微平均和加权平均分别进行评估。在当前实验的测试集上，由于各类别样本分布较为不均，加权平均的 F1 值具有相对较高的参考价值。

四、效果对比

横向比较

LSTM 比 CNN 取得了相对更好的成绩，可见在长文本分类上，LSTM 能够保留更多的语序信息，因此还是略胜一筹。

然而，MLP 取得了总体上最好的成绩——这是出乎预料的。

在整个实验过程中，我也曾多次对 CNN 和 LSTM 的实现进行过调整，不过均未见显著提升，可能是我什么地方一直没有写好。

调整全连接神经网络的参数

以下均用准确率代表模型效果。

dropout	**0**	0.2	0.4	0.6
acc	**60.5%**	59.9%	59.6%	57.9%
初始学习速率	0.1	0.5	**0.9**	1.2
acc	59.3%	59.1%	**60.5%**	59.8%
学习速率下降比率	0.4	0.6	**0.95**	1
acc	57.6%	58.1%	**60.5%**	58.7%

我们注意到，**增加 dropout 并没有起到提高准确率的效果**。这可能是因为，在当前数据集上，模型尚未达到过拟合的状态，增加 dropout 只会适得其反。

初始学习速率对模型最终的效果也有一定的影响。**学习速率偏低，则收敛缓慢，或容易收敛于局部最优值；学习速率偏高，则容易震荡，同样会削弱模型的效果**。

学习速率下降比率对准确率的影响与初始学习速率类似，**学习速率减少得过快，易造成收敛不充分，而若不减少（比率为 1），则容易在最优值附近震荡**。

调整卷积神经网络的参数

每份样本词数	100	**150**	200	250
acc	54.4%	**56.1%**	55.9%	55.3%
总通道数	12	24	**36**	48
acc	54.3%	55.5%	**56.1%**	55.0%

当减少样本词数时，准确率也相应提高。**可见 CNN 在相对较短的文本上能够更有效地提取文本特征**。当然，词数过少时原文信息也已丢失大半，准确率自然也随之下降。

随着总通道数，即卷积核个数的增加，准确率也呈上升趋势，不过也有一定的限度。

调整循环神经网络的参数

每份样本词数	120	180	**240**	320
acc	58.4%	58.5%	**58.6%**	58.6%

隐层维数	10	20	**40**	60
acc	55.6%	57.5%	**58.6%**	57.3%

调整样本词数并没有显著影响正确率，可见 LSTM 对句子的长度并不敏感。

另一方面，**增加 LSTM 隐层的维数，可以增加网络能记住的特征数量，因而能在一定程度上提高准确率**。不过随着维数的进一步提高，准确率的提升也不再明显。

五、问题思考

停止训练的时机

在本次实验中，我主要通过观察训练集的 loss 值确定迭代次数。具体而言，当发现 loss 无明显降低，或有反弹趋势时，即可停止训练。

作为辅助，我还从测试集中随机抽取 20%的样本作为验证集，并观察验证集上准确率的变化，在准确率开始显著下降之前停止训练。

两种方式相比，**固定迭代次数的方式较为简便，但可能陷入过拟合的问题，适合在实验确定后的生产部署中使用；通过验证集调整的方式能够在一定程度上避免过拟合现象，不过需要占用更多时间和内存资源，且需要人工参与，因而适合在实验过程中使用**。

参数的初始化

对于全连接层，我使用高斯分布初始化；对于卷积层，我使用正交初始化。

零均值初始化可以让初始参数在正负区域均有分布，从而防止 ReLu 和 Sigmoid 等输出恒正的激活函数造成梯度下降过程的失效[2]。

更进一步，高斯分布初始化是一种较为通用的初始化方式，它有助于使输入和输出的参数均得到良好的分布。

正交初始化可以使卷积核更紧凑，有助于卷积核的学习，适合在卷积神经网络中运用[3]。

过拟合的应对

本次实验主要通过引入验证集和添加 *dropout* 层的方式避免过拟合。

本次实验还尝试通过加入正则项的方式来减少过拟合，不过效果并不显著。

除此之外，扩大样本容量也是一种较为朴素但有效的避免过拟合的方式。

模型优缺点比较

模型	MLP	CNN	LSTM
优点	搭建简便，训练较快	能够较充分地提取特征	在长文本上表现较好
缺点	在本次实验中对文本直接取了均值，可能丢失细节信息	不擅长对长文本的处理，容易忽略语序信息	搭建难度较大，耗时长，资源占用高，且不能根除梯度消失问题[4]

六、心得体会

由于新闻类文本的特殊性，样本的前一两百个词已经能够涵盖文本的大部分信息。

张量的形状和维度要匹配准确，尤其是 LSTM 中，batch 被~~反人类地~~放到了 1 号维度。

情感之间其实并不总是能够划分明确的界限，例如感动和温馨，即使是人类也常常难以给出准确的判断。

七、参考资料

[1] 中文词向量：https://github.com/Embedding/Chinese-Word-Vectors

[2] 零均值初始化：https://www.zhihu.com/question/327435793

[3] 初始化方法比较：https://zhuanlan.zhihu.com/p/71644688

[4] LSTM 局限性：https://blog.csdn.net/weixin_41803874/article/details/100554276