# Introduction to AI
# Sentimental Classification Report

姓名: 任一

学号:2018011423

ry18@mails.tsinghua.edu.cn

2020 年 5 月 27 日

| 实验环境 | |
|---|---|
| 操作系统: | Windows10 家庭版 18362.72 |
| Python 版本: | Python 3.7.3 64-bit |

# 1 实验概述

在本实验中，我使用了神经网络的方法实现了文本的情感分类，具体来说我尝试了 MLP, TEXT CNN, RNN 和我自己设计的 Simple CNN 这 4 种方法对文本进行了分类。在文本向量化方面，我尝试了 Bag of Words 和 word2vec 方法。

# 2 实验思路

## 2.1 文本向量化

文本分类任务不可避免地需要需要文本向量化。常见的文本向量化方法有 Bag of Words, TF-IDF, word2vec 等等。在本次实验中我使用了 Bag of Words 和 word2vec 方法进行文本向量化。

对于 Bag of Words, 我调用了 sklearn 中的 CountVectorizer 模块，对训练集中输入文本的词汇进行计数。同时，我还去掉了文本中的停用词，并只选取出现频数最多的 1000 个词作为特征词汇。去除停用词可以排除诸如" 的、地、得" 等没有实际含义的虚词的影响，有利于对语义的捕捉。选取出现频数最多的 1000 个词，一方面可以降低模型复杂度，加快训练速度，另一方面也可以排除出现频数低的词汇造成的干扰。下图则为我在 MLP 上，设置不同的选词数量，得到的测试集准确率数据。

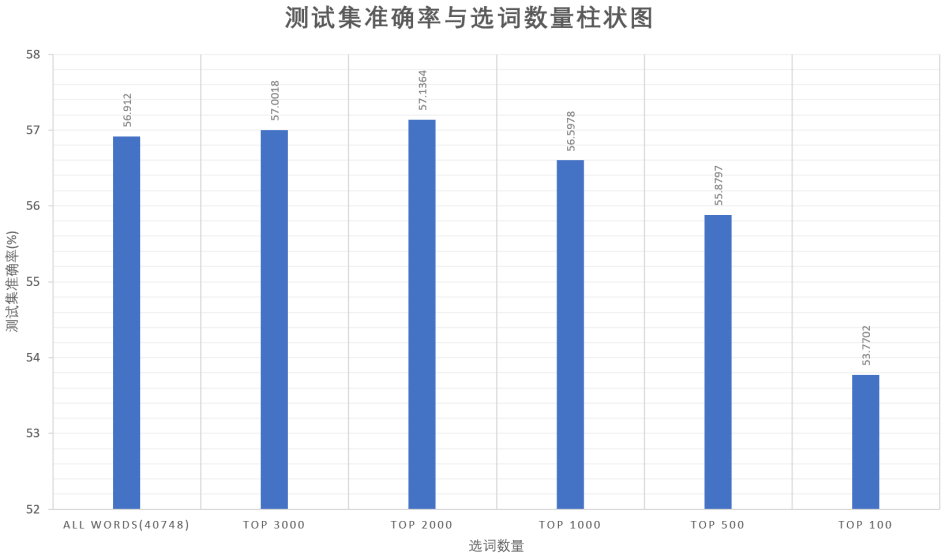



图 1: 测试集准确率与选词数量的关系

从图中可以看出，选择 2000 个词的测试集准确率最高，选词过多或过少都会一定程度上影响准确率，这也与上述分析相符。

对于 word2vec, 我使用了助教文档中的词向量参考链接[1]。我分别尝试了不使用预训练词向量和使用预训练词向量的方法。下图为在 Text CNN 上，测试上述两种方法，得到的准确率柱状图。

从图中可以看出，加载了预训练词向量的 Text CNN 准确率较为明显地高于未加载预训练词向量的 Text CNN. 我认为这是因为，已经在大语料上训练好的词向量已经具有了一定的语义特征，在

[1]词向量链接如下: `https://github.com/Embedding/Chinese-Word-Vectors`

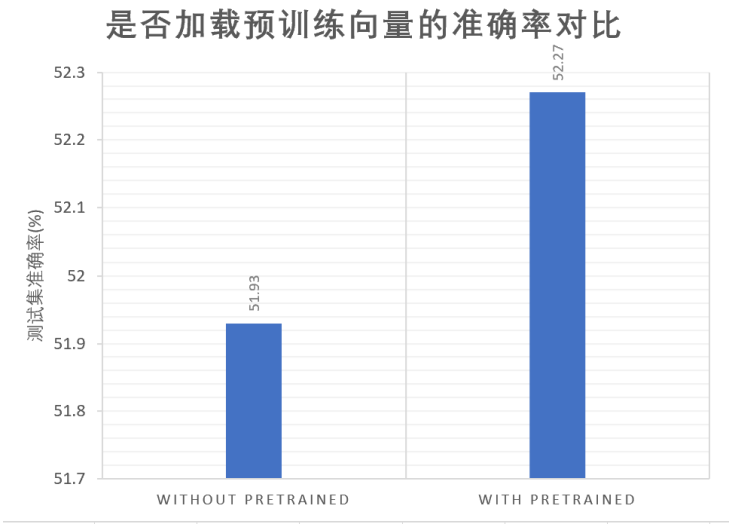



图 2: 是否加载预训练词向量的准确率对比

当前任务下会有更好的表现。相比之下，未加载预训练词向量时，词向量从随机初始化开始训练，可能对词的语义特征捕捉就不太充分，因此效果也略微差一些。

## 2.2 网络结构

### 2.2.1 MLP

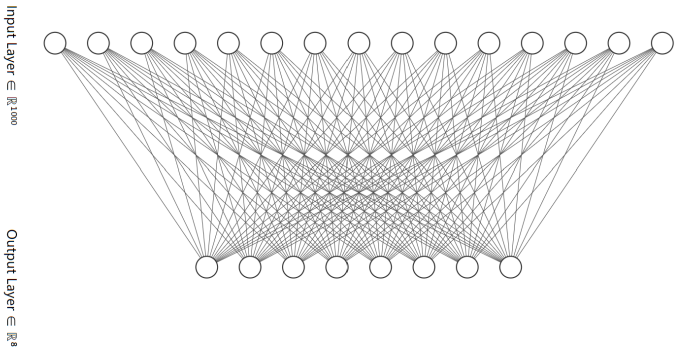
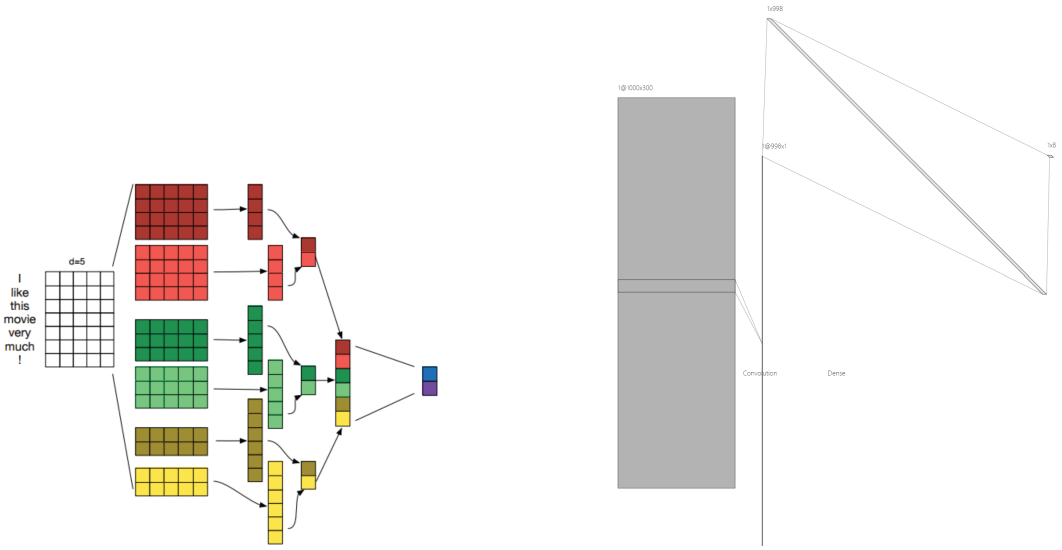



图 3: MLP 网络结构

我实现的 MLP 结构较为简单，只经过了一层全连接，将每个句子中对 1000 个特征词的计数作为输入，输出即为 8 个标签，输出结果经过 softmax 归一化后，即可得到每个标签的概率值，选取概率最大的标签作为最终的输出结果。

### 2.2.2 Text CNN 和 Simple CNN

在这部分，我实现了两种 CNN 结构，分别是 Text CNN 和我自己设计的 Simple CNN 如下图所示。




(a) Text CNN 网络结构　　(b) Simple CNN 网络结构

图 4: 我实现的两种 CNN

图 6(a) 中，设一句话的长度为 $m$, 词向量维数为 $d$, 则一句话的 tensor 大小为 $[m \times k]$. 分别使用大小为 $[3 \times d], [4 \times d], [5 \times d]$ 的卷积核对句子 tensor 做卷积，每个卷积核的输出通道数为 2. 对每个卷积核卷积后的结果经过 relu 激活函数后取 max, 将这 6 个 max 值拼为一个 $[1 \times 6]$ 的 tensor, 再经过 Full Connection 以及 softmax, 得到 8 个标签的概率值，选择概率最大的标签作为最终结果。

图 6(b) 中，我限定每句话的长度为 1000, 词向量维数为 300，则每句话的 tensor 大小为 $1000 \times 300$. 使用大小为 $3 \times 300$ 的卷积核进行卷积，得到 $[1 \times 998]$ 的 tensor，再经过 $[998 \times 8]$ 的 Full Connection 和 softmax, 得到 8 个标签的概率值，选择概率最大的标签作为最终结果。

### 2.2.3 RNN

为了解决传统 RNN 中梯度消失和长距离信息依赖的问题，我采用了 LSTM 模型。结构图如下：

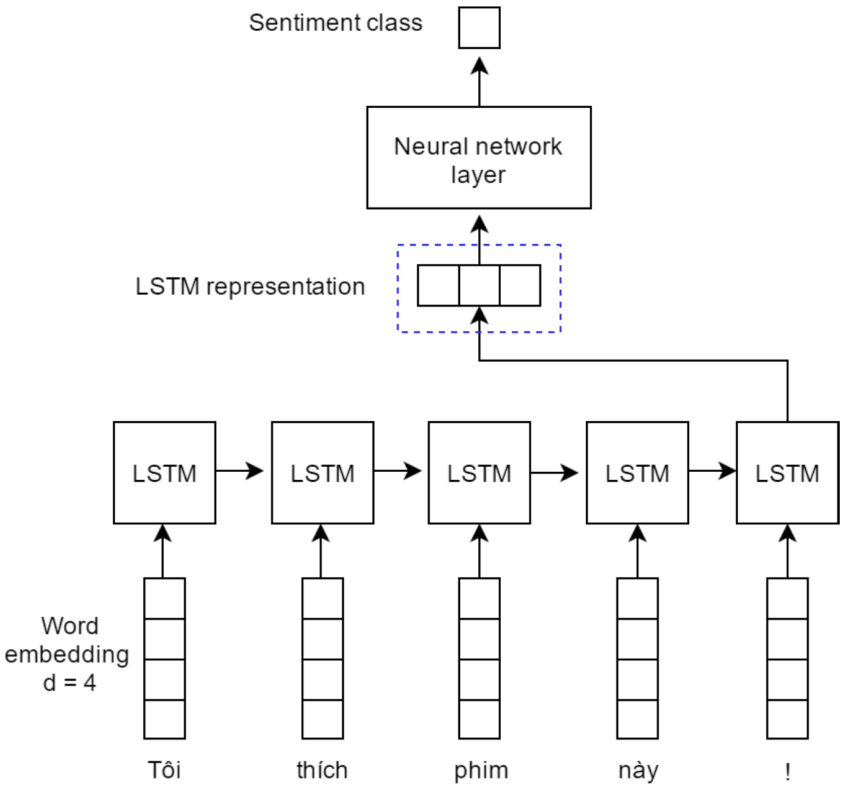



图 5: LSTM 网络结构

在这个 LSTM 网路中，句子中的每个词的词向量都作为一个 LSTM 单元的输入。取最后一个 LSTM 单元的输出，并经过 Full Connection 层和 softmax 层, 得到 8 个标签的概率值，选择概率最大的标签作为最终结果。

# 3 实验结果

## 3.1 指标展示及分析

表 1: 不同模型各指标数据表

| | Accuracy | Macro-F1 | Micro-F1 | Correlation |
|---|---|---|---|---|
| MLP | 0.5682 | 0.169 | 0.568 | 0.523 |
| TextCNN | 0.5727 | 0.169 | 0.568 | 0.391 |
| SimpleCNN | 0.5570 | 0.163 | 0.557 | 0.371 |
| LSTM | 0.4776 | 0.081 | 0.478 | 0.334 |

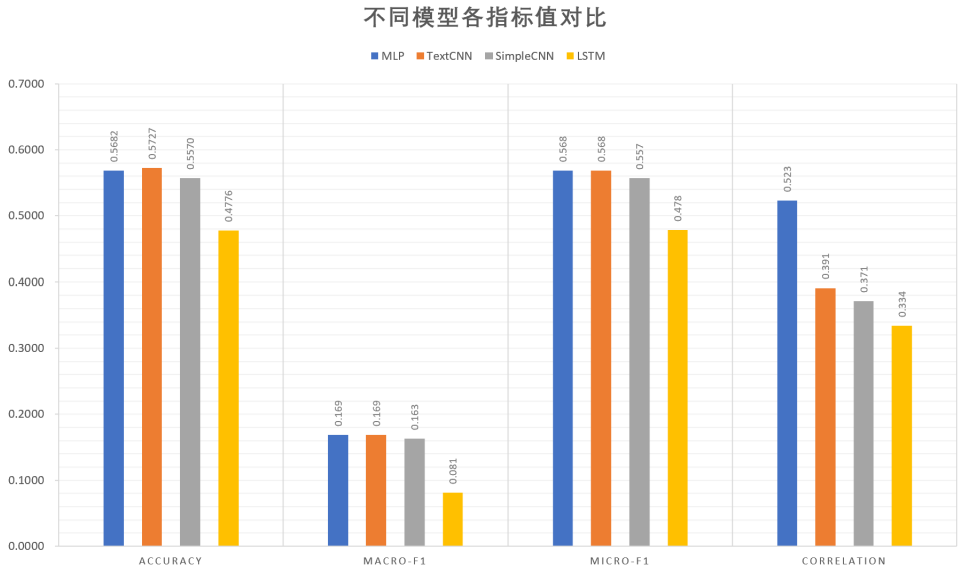


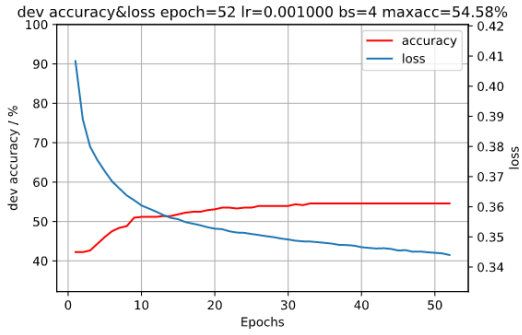

图 6: 不同模型各指标数据

从上面的图表可以看出，MLP 和 Text CNN 以及我设计的 Simple CNN 在各指标上能够取得更好的结果。LSTM 的效果一般。

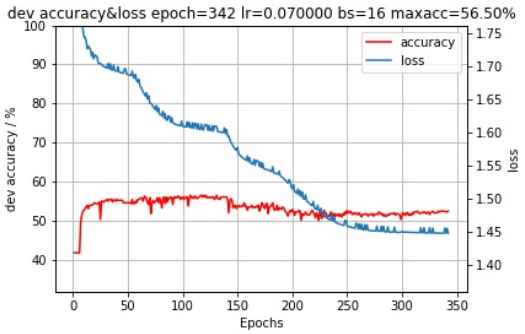
为了更加细致地分析各模型训练的效果，我画出了各模型训练时的 loss 和验证集的 Accuracy 随训练轮数变化的折线图 (下面简称该折线图为训练曲线) 如下：

从图 7(a) 中可以看出，MLP 的训练曲线较为合理，即 loss 下降速度由快到慢且 loss 下降到较低的值，验证集准确率上升速度由快到慢，最终基本收敛于一个值。我认为 MLP 能够取得较好结果的原因是，MLP 把出现频次高的词的频数作为神经元的输入，只经过一层全连接就映射到输出的 8 个标签，我认为这有很好的解释性。例如" 悲伤" 作为一个常用词，就很有可能作为神经元的输入，这个输入神经元与" 难过" 这一标签对应的神经元之间的权值就可能很大，而与" 搞笑" 这一标签对应的神经元之间的权值就可能很小。这样的特征词汇与标签之间可能存在的强相关性，可能是 MLP 效果较好的一个原因。此外 MLP 也具有结构简单、参数较少的特点，这一特点也有利于模型的快速收敛，有利于取得不错的训练结果。

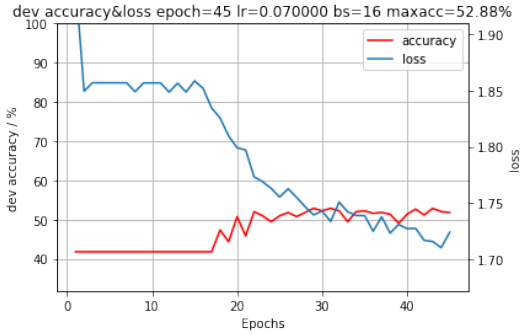
从图 7(b)(c) 中可以看出，Text CNN 和 Simple CNN 的训练曲线也基本正常，随着 loss 的下降，验证集准确率也在上升。但二者的 loss 均下降不明显，相较 MLP 来说没有下降到一个很低的值。我



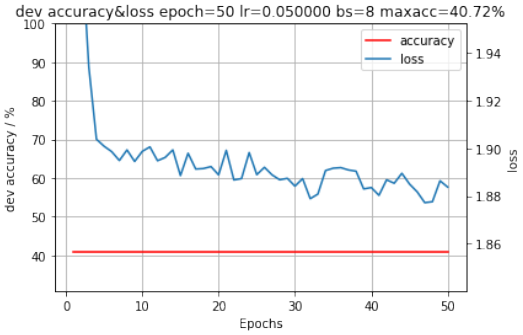

(a) MLP




(b) Text CNN




(c) Simple CNN




(d) LSTM

图 7: 各模型训练曲线

认为 CNN 能够在情感分类中取得较好任务的原因是，CNN 通过卷积的方式 (卷积核大小为 $[k, d]$, $d$ 是词向量维数，$k$ 是较小的常数例如 3) ，能够较好地捕捉每个词的语义以及词与词之间的联系。例如" 非常", " 开心", " 快乐" 这 3 个词放在一起出现，就很有可能表达了积极的语义，卷积核也能在一次运算中把这 3 个词的语义同时捕捉到。捕捉词之间的语义联系，有利于更好地掌握整句话的语义，从而得出较为正确的预测结果。

从图 7(d) 中可以看出，LSTM 的训练曲线就较为异常，主要体现在 LSTM 的验证集准确率一直保持在第一个 epoch 训练后的值，在后续的训练中并没有上升，此外 loss 下降较为波动并且没有收敛到很低的值。LSTM 这样的异常训练曲线，其实和 Simple CNN 刚开始的训练曲线很相似，只是 Simple CNN 在开始的平台期过后，loss 进一步下降，验证集准确率进一步上升，而 LSTM 在训练过程中却没有进一步的改善。我认为这可能与 LSTM 的参数有关，需要对 LSTM 模型进行进一步的调参以达到更好的效果。此外我也认为，LSTM 效果不佳，可能与句子长度太长有关。在对 LSTM 进行训练时，我将训练和测试的句子，都裁剪到了 1000 的长度，句子中每个词都会作为一个 LSTM 单元的输入，这样的网络就有 1000 个 LSTM 单元，而每个单元内部都有若干的运算，其中包含的参数很多。因此 LSTM 效果不佳，可能也与句子太长、参数太多有关。

## 3.2 不同参数下效果分析

在这一部分，我将分析不同的 learning rate 和 batch size 对模型测试集准确率的影响。为了运行的方便，我将使用 MLP 进行这一部分的实验。

### 3.2.1 Acc-Learning Rate

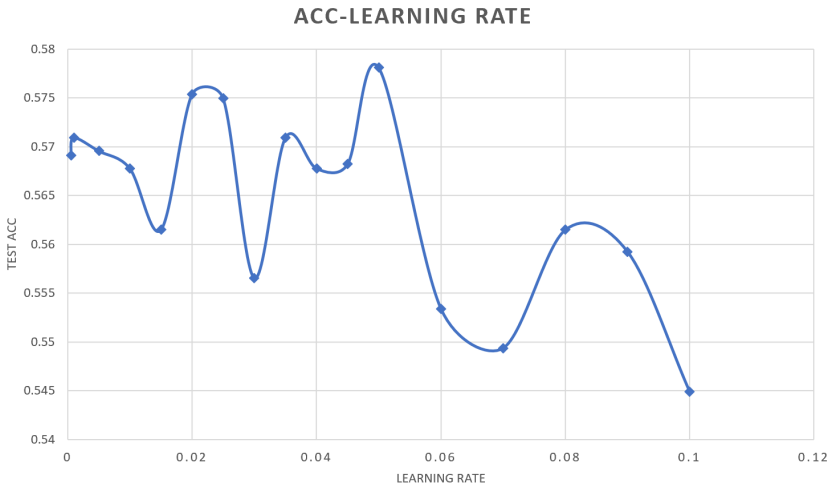



图 8: 测试集准确率与学习率关系 (batch_size = 4)

从图中可以看出，当学习率较小时 (约在 0.06 以下)，测试集 Accuracy 变化较为波动，但总体较高。当学习率较大时，测试集 Accuracy 呈波动下降趋势。这可以解释为，当学习率太大时，模型难以有效收敛于全局最优，而当学习率较小时，尽管训练所需时间可能会更多一些，但收敛到全局最优的可能性会更大一些。

### 3.2.2 Acc-Batch Size

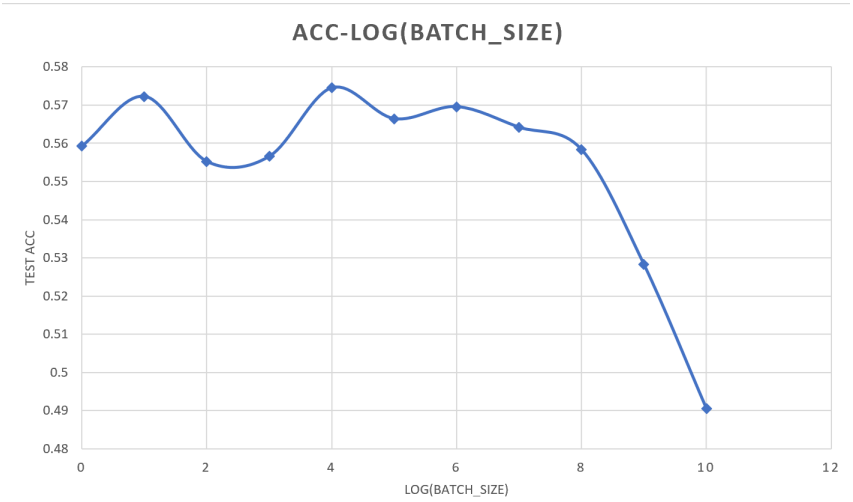



图 9: 测试集准确率与 Batch Size 关系 (learning_rate=0.01)

从图中可以看到，当 batch_size<$2^6$ 时，模型的准确率基本都很高，但高于 $2^6$ 时，准确率明显下降。batch size 的大小可以解释为一次使用多少样本来更新梯度. batch size 较小时，使用的样本少，可以为模型提供更多随机下降的机会，能够在一定程度上避免陷入局部最优值，但这样也可能导致梯度方向更新不准确. batch size 较大时，可以提供较为准确的梯度方向，但是更有可能陷入局

部最优值. 因此我认为，图中 batch size 过大时，可能是陷入了局部最优，从而没能达到很好的预测准确率。

# 4 问题思考

## 4.1 何时停止训练

理论上，当模型出现过拟合，即训练 loss 不断下降而验证集 Accuracy 不再上升或开始下降时，应该停止训练。我的实现方法是，当最近得到的 10 个验证集准确率的标准差小于 0.001 时停止训练，或验证集准确率连续 3 次下跌 1% 以上，停止训练。

若比较固定迭代次数和使用验证集调整的方法，我认为使用验证集调整的优点在于，可以在训练过程中实时把握模型的准确率，也方便根据验证集的结果调整模型参数等等，但缺点在于，每个 epoch 后对验证集做测试，会占用一定的时间，可能会对训练速度有影响。固定迭代轮数有利于提升训练速度，但是无法实时掌握模型的准确率。

## 4.2 实验参数的初始化

在本实验中，我的模型参数初始化主要是均匀分布初始化. [2]

正交初始化在 RNN 中较为常用，由于正交矩阵的行列式为 1，正交矩阵中每一个向量的长度也都为 1 且彼此正交，因此在 BP 算法中，可以较好地缓解梯度消失或梯度爆炸的问题.

高斯分布初始化也是很常用的一种初始化方法. 具体来说即为神经网络中的参数值采自某个高斯分布总体. 零均值初始化实现方法与高斯分布初始化也类似。但是这样的初始化方法可能导致的问题是隐藏层方差过大，若经过 Sigmoid 或 Tanh 这样的非线性函数时，梯度消失现象会较为明显.

## 4.3 如何解决过拟合问题

过拟合是深度学习中常见的问题，我对该问题的解决方法主要有一下几种.

### 4.3.1 降低网络复杂度

降低网络复杂度在一定程度上可以降低模型的拟合能力，从而降低对训练集过拟合的风险。例如 MLP 这样简单的模型，过拟合风险可能就会小一些。

### 4.3.2 Early Stopping

当训练集 loss 仍在下降，但验证集准确率提升不明显或已经开始下降时，就很有可能出现了过拟合。此时就应该及时停止训练，防止训练时间过长导致模型对训练集过拟合。

### 4.3.3 Dropout

Dropout 技术即为在训练过程中，以一定的概率，随机选择某些神经元使之输出为 0. 在预测时则不需要做 Dropout. Dropout 可以在一定程度上缓解训练时的过拟合现象。

[2]均匀分布为 Pytorch 默认的初始化方法.

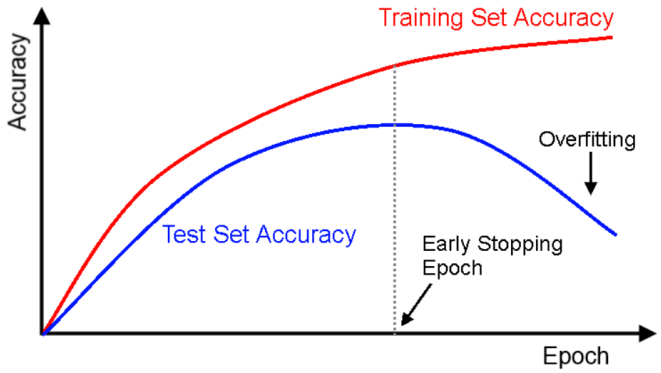



图 10: Early Stopping 示意图

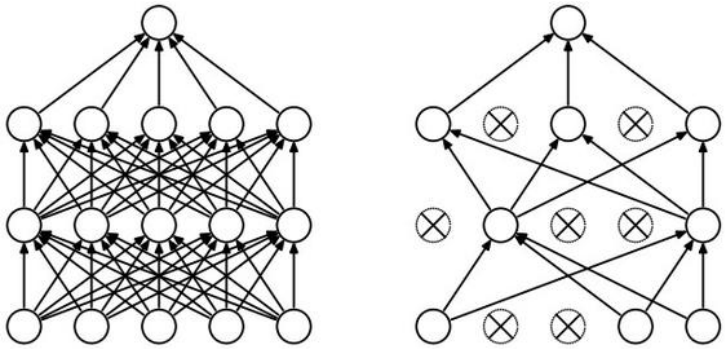



图 11: Dropout 示意图

#### 4.3.4 Regularization

Regularization 即正则化，简单来说就是在 loss function 中加入正则项 (或" 惩罚" 项)，以对较为复杂的模型给予一定的" 惩罚". 其数学表示为:

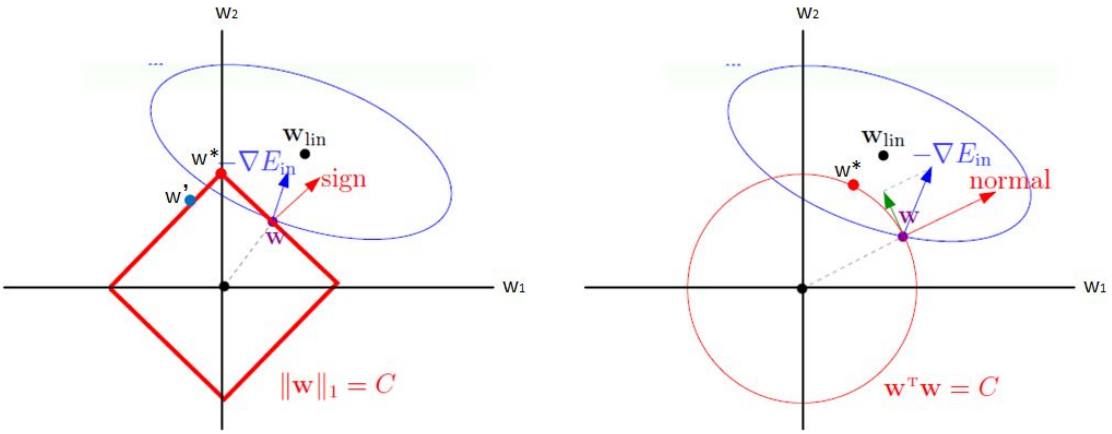
$$\tilde{J}(w; X, y) = J(w; X, y) + \alpha\Omega(w)$$

正则化也有两种形式，分别是 L1 Regularization 和 L2 Regularization, 其数学表示如下:

$$L_1 : \Omega(w) = \|w\|_1 = \sum_i |w_i|$$
$$L_2 : \Omega(w) = \|w\|_2^2 = \sum_i w_i^2$$

#### 4.3.5 Ensemble Learning

集成学习在一定程度上能够缓解过拟合. 例如 Bagging 方法，在产生训练集时会使用 Bootstrapping 方法，使得训练数据有一些扰动，同时在最后集成时也会对各个基学习器的结果取平均，从而在一定程度上缓解过拟合.




(a) L1 Regularization (b) L2 Regularization

图 12: 两种正则化示意图

## 4.4 MLP, CNN, RNN 优缺点

MLP 的优点在于，网络结构简单，训练速度快。缺点在于表达能力有限，对于文本序列信息、图像信息处理能力不强. 此外当 MLP 较深时，易陷入局部最优，且梯度消失现象较明显。

CNN 的优点在于，卷积核的权值共享可以降低一些参数量，同时卷积也可以较好地捕捉局部特征。缺点在于，CNN 的卷积操作忽略了局部与整体的关系. CNN 较深时，靠近输入层的梯度较小. 此外 CNN 中也有较多参数需要调整，例如卷积核大小、通道数等等.

RNN 的优点在于，可以有效地处理文本序列信息. 缺点在于，易产生梯度消失现象，对长距离语义的依赖难以处理 (可以考虑使用 LSTM 或 GRU)。此外 RNN 模型的参数量也较大，模型复杂。

## 4.5 训练集与测试集数据分布

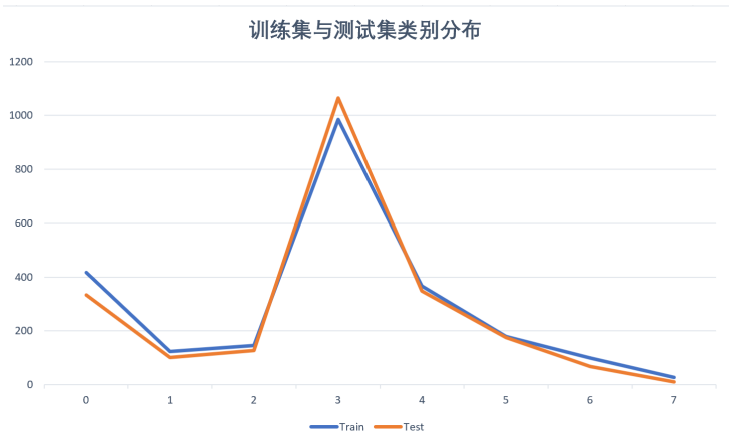



图 13: 训练集与测试集数据分布折线图

从图表中可以看出，训练集和测试集基本是同分布的。但很明显的问题是，类别分布非常不均匀，第 3 类 (即" 愤怒") 占比基本是其他单个标签的两倍及以上。这对训练也会造成很大影响。经过我的输出，我发现我使用的 LSTM 将所有的测试集样例都预测为了" 愤怒"，一部分原因可能就是" 愤怒" 占比太高，使得学习器对" 愤怒" 这个标签较为敏感，对其他的标签学习并不充分.

# 5 心得体会

在本次实验中，我尝试了用多种神经网络做情感分类。在这个过程中，我学习了如何对文本进行向量化，如何设计并实现神经网络，并掌握了简单的调参方法。同时我也感受到了神经网络的神奇和调节神经网络的不易。

此外我也在实验中进行了充分的分析，用自己的思考解释实验结果，提出猜想并通过实验验证或否定猜想。这样理论与实践相结合的实验方法，使我感到收益良多。

感谢老师和助教在本次实验中给予我们的悉心指导！