人工智能导论 Lab-2 实验报告

褚轩宇 2023012461

2024 年 5 月 8 日

代码网盘链接：https://cloud.tsinghua.edu.cn/d/9d641aa08b67446fa45a/

1 实验环境

操作系统: macOS Sonoma Version 14.1.2

开发语言: Python 3.11.7

2 MLP 模型 (Baseline)

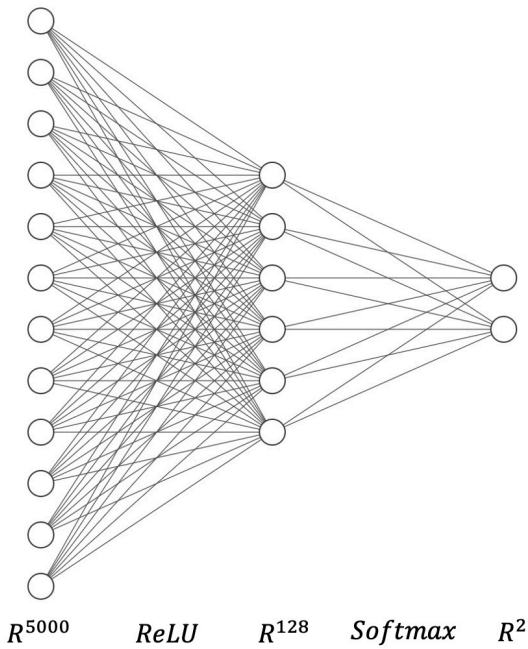
2.1 数据处理

对 train.txt, test.txt validation.txt 进行如下处理:

设置变量 text_size = 100，将所有句子的长度设置 text_size（超出部分截断，不足部分补 0），使用 Word2Vec 将每个词转化为长为 50 的向量，无法识别的设为 0 向量。将代表情感正负的 0, 1 分别转化为 $[1, 0], [0, 1]$ 的 One-Hot 编码。

2.2 模型结构图及流程分析

2.2.1 模型结构图



2.2.2 流程分析

输入层：大小为 text_size ×50，即 5000。

隐藏层：大小为 hidden_layer = 128，激活函数为 ReLU。

输出层：大小为 2，激活函数为 Softmax。

损失函数：经过 Softmax 激活函数后，使用交叉熵损失函数来计算 loss。

2.3 实验结果

在取 epoch = 20, batch_size = 16, learning_rate = 0.05, text_size = 100，hidden_layer = 128 时：

Accuracy：72.90%，F-score：72.52%。值得一提的是，在迭代 20 次后模型在训练集上的 loss 已经达到了 10^{-3} 级别。

3 CNN 模型

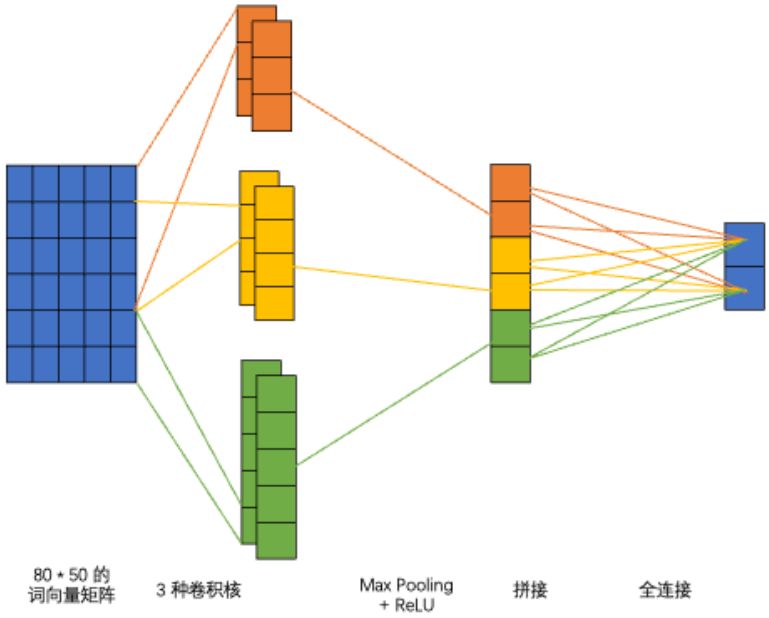
3.1 数据处理

对 train.txt, test.txt validation.txt 进行如下处理：

设置变量 text_size = 80，将所有句子的长度设置 text_size（超出部分截断，不足部分补 0），使用 Word2Vec 将每个词转化为长为 50 的向量，无法识别的设为 0 向量，组成 text_size ×50 的词向量矩阵。将代表情感正负的 $0, 1$ 分别转化为 $[1, 0], [0, 1]$ 的 One-Hot 编码。

3.2 模型结构图及流程分析

3.2.1 模型结构图



3.2.2 流程分析

卷积层：选取大小为 3×50, 4×50, 5×50 的卷积核各 kernel_num $= 100$ 个。

池化层：采用 max pooling，对于一个卷积核，取其输出元素中全局的最大值为当前卷积核经过 max pooling 的结果，得到长为 $3\times$ kernel_num $= 300$ 的向量。

全连接层：使用 300×2 的全连接层。

激活函数：在经过池化层后，使用 ReLU 激活函数；经过全连接层后，使用 Softmax 激活函数。

损失函数：经过 Softmax 激活函数后，使用交叉熵损失函数来计算 loss。

3.3 实验结果

在取 epoch $= 20$, batch_size $= 16$, learning_rate $= 0.05$, kernel_num $= 100$, text_size $= 80$ 时：

Accuracy：83.47%，F-score：83.10%。

4 RNN 模型

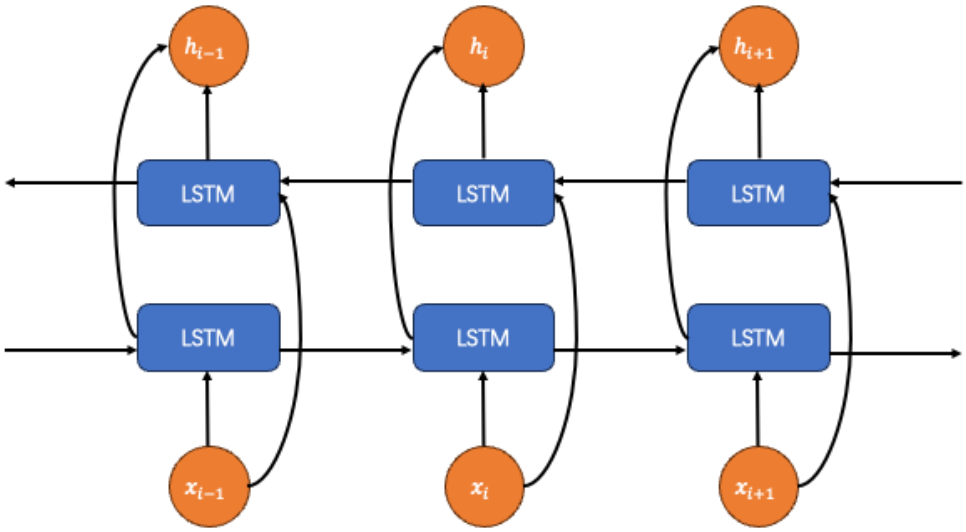
4.1 数据处理

对 train.txt, test.txt validation.txt 进行如下处理：

设置变量 text_size = 64，将所有句子的长度设置 text_size（超出部分截断，不足部分补 0），使用 Word2Vec 将每个词转化为长为 50 的向量，无法识别的设为 0 向量。将代表情感正负的 0, 1 分别转化为 $[1, 0], [0, 1]$ 的 One-Hot 编码。

4.2 模型结构图及流程分析

4.2.1 模型结构图



4.2.2 流程分析

LSTM 模块：使用双向 LSTM 模型，num_layers=2，hidden_dim = 32。

全连接层：由于使用了双向 LSTM 模型，将正反向输出结果拼接后，使用 (hidden_dim ×2) ×2，即 64 × 2 的全连接层。

损失函数：经过 Softmax 激活函数后，使用交叉熵损失函数来计算 loss。

4.3 实验结果

在取 epoch = 20, batch_size = 16, learning_rate = 0.3,
text_size = 64, num_layers = 4，hidden_dim = 32 时：
Accuracy：85.09%，F-score：85.86%。

5 参数效果对比

5.1 batch size 与 learning rate

在 CNN 模型中，分别取不同的 batch_size(batch) 与 learning_rate(lr)，对 epoch = 10 后的 loss 与训练时长进行统计：

batch \ lr	0.1	0.01	0.001
8	0.14 / 67s	0.05 / 76s	0.41 / 73s
32	0.04 / 50s	0.30 / 48s	0.52 / 48s
128	0.29 / 37s	0.46 / 37s	0.64 / 37s

当选取的 batch size 越大，意味着反向传播过程的进行次数越少，这也就对应着训练时长的降低。然而，更大的 batch size 也可能导致数据中一些较小的特征被忽略。

当选取的 learning rate 越大，意味着单次反向传播梯度的大小越大，对应着 loss 的减小速度越快。但如果 learning rate 过大，可能会导致模型在训练后期单次梯度过大，无法进行更细微的调整。

统计数据以及尝试的结果表明，更大的 batch size 往往需要配合更大的 learning rate，这也与更大的 batch size 对应更少的反向传播次数相符合。

5.2 CNN 中卷积核的选取

在上述 CNN 模型中，我们选择了 $k \times 50$，$k = 3, 4, 5$ 的卷积核各 100 个。

若卷积核的种类太少，模型可能只能学到小颗粒度（或大颗粒度）的特征；若卷积核个数太少，模型可能难以学到句子中充足的特征。但是，若卷积核的种类、个数太多，可能出现训练时长过长的问题，过大的模型也更容易出现过拟合的情况。

5.3 RNN 中 text size 的选取

在上述 RNN 模型中，我们设置 text_size=64，即将所有句子的长度均视作 64。

当 text size 过小，句子的大部分内容可能都被截取了，模型更难以判断句子的真实情感；而当 text size 过大，使得多数句子都需要在末尾补 0，这可能导致 0 被模型视作某种特征，从而影响前面内容的情感判断，与此同时，更大的 text size 也意味着更长的训练时长。

因此，应当选取一个与大多数句子长度相近的 text size。

6 效果差异

MLP 在测试集上的 Accuracy 与 F-Score 基本都收敛到了约 75%，而 CNN 与 RNN 的 Accuracy 与 F-Score 较为依赖超参数的选取，在适当选取参数的情形下均可以达到约 83%，明显优于 MLP。

可以看出，MLP 模型在情感分类问题上，由于可能会忽略文本之间关联的特征等原因，而效果不如 CNN 与 RNN；同时，MLP 模型在情感分类问题上能够很好的拟合训练集，但泛化能力不如 CNN 与 RNN。

7 问题思考

7.1 训练的停止

上述模型均在经过固定迭代次数后停止。

7.1.1 固定迭代次数：

优点：实现方法简单，且不需要额外的计算来判断是否停止训练。

缺点：没有考虑模型是否已经收敛，可能导致过拟合或者欠拟合；对于不同的数据集需要多次设置参数，不够灵活。

7.1.2 通过验证集调整：

优点：能够动态判断模型是否收敛，一定程度上避免过拟合与欠拟合的情况。

缺点：需要额外的计算与一定强度的验证集。

7.2 实验参数的初始化

上述模型的参数初始化均通过 pytorch 库中默认的初始化完成。

零均值初始化：如果将所有参数的初值均设置为 0，那么在训练过程中，多个神经元可能会得到完全相同的梯度，学到完全相同的内容。

高斯分布初始化：避免了不同神经元学到相同内容的情况，同时高斯分布初始化也能够在一定程度上避免模型在训练初期陷入局部最优解。

7.3 过拟合的解决方法

1. 在损失函数中加入正则化项，从而降低模型的复杂度；

2. 降低模型本身的复杂度；

3. 加入验证集，在适当的时间停止模型的训练，避免模型对训练集的过拟合；

4. 在训练过程中每次随机 dropout 一些节点，强制模型学习更为普遍、共性的特点；

5. 通过旋转、剪切等方法处理训练数据，使得训练集更加丰富。

7.4 MLP, CNN, RNN 的优缺点比较

7.4.1 MLP

优点：结构简单，灵活性高，可以通过添加隐藏层的个数来适应不同问题；

缺点：参数量过大，训练效率不高；在序列问题与图像问题上，都忽略了特征之间的关系（序列、图像上的相邻等），可能会影响训练的效果。

7.4.2 CNN

优点：擅长于解决图像类任务，能够通过选取不同大小的卷积核学习到图像中不同大小颗粒度的特征；使用卷积核来减少了参数量，从而加速了计算。

缺点：对一些没有明显图像特征的任务较为乏力；对于一些整体特征的获取不够充分；难以解决一些长期依赖的问题。

7.4.3 RNN

优点：擅长于处理序列类的任务，且能够记忆之前的信息；可以较好地关联上下文的特征，解决一些长期依赖的问题。

缺点：在处理较长的序列时，容易出现梯度消失的问题。

8 心得体会

本次 Lab 是我第一次学习并实践神经网络相关的知识，在实现 MLP, CNN 与 RNN 网络的过程中，我逐渐意识到模型结构建立、超参数的选取，甚至是算力对当今 AI 训练的决定性作用，在逐步尝试的过程中加深了对神经网络的理解，并期待能在往后的学习中进一步探索相关的内容。