情感分析实验报告

计01　容逸朗　2020010869

代码相关

网盘地址：https://cloud.tsinghua.edu.cn/d/069615800fb34548aeee/

模型介绍

模型建构

对于每一个要分析的句子，我们可以按照字或词的方式来断句，由于本次实验给出了词向量库，因此我们偏向于以词为单位的断句。

实验给出的训练材料与测试材料都已经按照词的方式分开了，为此我们只需要遍历句子中的每一个词，然后将每一个词向量化即可，此时每个句子可以被表示为一个 $W \times L$ 的矩阵，其中 $L = 50$ 为词向量的长度，W 是句子的词数。

为了训练时的便利，我实现的模型会将所有句子的词数统一为 B。具体来说，就是将长度不足的句子以多个**随机的50维向量**填充至 B，而长度过长的句子则会捨去大于 B 的部分。

需要注意的是，并不是所有的词都可以在词向量库中找到，因此我们规定找不到的词会统一用**随机的50维向量**填充。

CNN 模型

本次我实现的 CNN 模型主要可以分为 3 个部分，第一部分是卷积层，第二部分是池化层，最后一部分为全连接层，下面将具体分析各部分的实现方法。

流程图

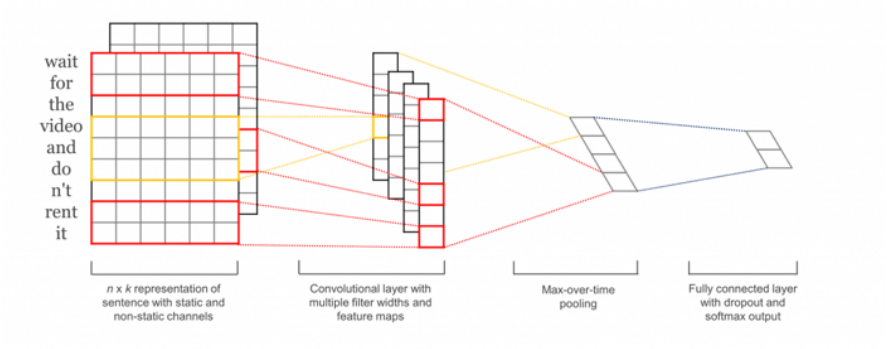

图1　CNN 流程图

流程分析

1. 输入

   输入是一个 3 维的矩阵（但实现时会额外加上维度为 1 的一维，这是为了配合 `nn.Conv2d` 的输入所导致的），第一维的大小是每批次的句子数（`BATCH_SIZE`，下记为 `B`），第二维是句子的词数（`SENTENCE_SIZE`，下记为 `SS`），第三维是词向量的长度，在本次实验中为 50（下记为 `L`）。

2. 卷积层

   这里卷积核分为 3 类，大小分别为 $2 \times L$, $3 \times L$, $4 \times L$，每一类有 `HIDDEN_SIZE`（指令：`-hs`）个。

   对于大小为 $B \times SS \times L$ 的输入，其输出分别是大小为 $B \times (SS - 1)$，$B \times (SS - 2)$, $B \times (SS - 3)$ 的矩阵。

   完成操作后，还需要对每一个输出做激活操作，这里选择了 `ReLU`，即负值取零，其余不变的方法。

3. 池化层

   对于步骤 2 的输出，我们可以利用池化操作将输出的大小统一为 $B \times 1$ 的矩阵，具体来说，就是将大小为 $B \times (SS - 1)$ 中第二维的最大值提取出来。

   由于我们有 $3 \times HS$ 个卷积核，所以池化后的结果共有 $3HS$ 个大小为 $B \times 1$ 的矩阵。我们可以将这些输出拼接为一个大小为 $B \times 3HS$ 的矩阵。

4. 全连接层

   最后，我们可以通过一个简单的线性变换（对矩阵的第二维左乘一个大小为 $2 \times 3HS$ 的矩阵并加上偏移量）得到二分类的结果。

   在输出前，可以增加一个 `dropout` 层减慢过拟合的速度。

   最后可以增加一个 `softmax` 层使得输出的结果满足概率分布。

RNN 模型

流程图

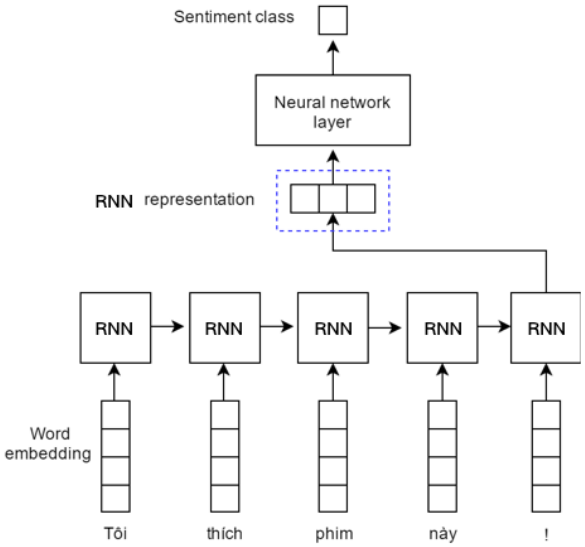

图2　RNN 流程图

流程分析

RNN 模型的结构如上图所示，RNN 会顺序读入句子中的词向量，然后这层的信息会保序在隐藏层中，并与下一步的组合通过各种操作得到一个结果。不断重复上述过程，直至最后一位输入后，可将此位置给出的输出（大小 $SS \times HS$ 的矩阵）和隐藏层（大小为 HS 的向量）的结果。

最终，我们可以通过线性变换和 softmax 得到二分类的结果。

注意：可以改变 RNN 模型网络的类型来得到不同的结果。（如 LSTM, GRU 等）

MLP 模型

直接将输入通过线性变换映射到一个隐藏层上，然后将结果通过线性变换映射到长度为 2 的向量上（二分类任务），并利用 dropout 与 softmax 层使输出的结果满足概率分布。

实验结果

下面的实验统一设置隐藏层的维度为 64，最大 epoch 数为 50，Batch Size = 64，句子词数 64，优化器为 Adam，drop rate = 0.5，学习率 lr = 0.001。

其中 CNN 中每一种卷积核的个数均为 50 个，合计 150 个。

模型	准确率(%)	F-Score
CNN	85.908	0.86316
RNN-LSTM	84.553	0.85271
RNN-GRU	85.366	0.85714
MLP	80.488	0.81443

从上表可见，CNN 模型的效果最好，但是 RNN 模型与 CNN 模型的差距不大，猜想是因为数据集的规模足够大，以及任务足够简单（二分类），因此两者的效果相若。

除此之外，RNN 在这一问题上的效果从理论上来看还有提升空间，因此需要对超参数做进一步的讨论以获得更好的效果。

整体而言，两种模型的效果都比 baseline (MLP) 来得更好。

参数分析

CNN 模型

参数	准确率(%)	F-Score
初始值	85.908	0.86316
$hs = 1$	76.423	0.75766
$hs = 10$	82.656	0.82609
$hs = 20$	84.824	0.84530
$hs = 100$	85.095	0.85255
$hs = 150$	88.347	0.88410
$hs = 200$	86.450	0.86188
$hs = 500$	85.908	0.85792
$lr = 0.1$	50.678	0.67266
$lr = 0.01$	82.927	0.83200
$lr = 0.0001$	80.759	0.80548

LSTM 模型

参数	准确率(%)	F-Score
初始值	84.553	0.85271
$hs = 1$	74.797	0.73504
$hs = 4$	81.301	0.80115
$hs = 16$	84.824	0.84946
$hs = 32$	84.282	0.84574
$hs = 128$	85.366	0.85864
$hs = 256$	85.637	0.85154
$lr = 0.1$	69.648	0.69892
$lr = 0.01$	85.366	0.84916
$lr = 0.0001$	83.469	0.84156

GRU 模型

参数	准确率(%)	F-Score
初始值	85.366	0.85714
$hs = 1$	79.675	0.79675
$hs = 4$	80.488	0.80952
$hs = 16$	84.553	0.84298
$hs = 32$	84.282	0.85052
$hs = 128$	85.908	0.86096
$hs = 256$	84.824	0.84530
$lr = 0.1$	50.678	0.67266
$lr = 0.01$	85.095	0.84150
$lr = 0.0001$	84.011	0.84833

对于 CNN 模型，更多的卷积核可以得到更好的效果，在卷积核的个数达到 150*3=450 个时，准确率可达 88%，但卷积核的个数继续增加时，准确率却会有所下降；

对于 RNN 的 LSTM 与 GRU 模型，更多的隐藏层可以带来更好的性能，但是当隐藏层数量大于一定数值后，准确率则有所下降。

若学习率大于 0.001，则性能会有所降低，当学习率为 0.1 时，训练的效果已是毫无作用；相反，若学习率小于 0.001 （如 0.0001），此时模型收敛速度过慢，在 100 次迭代后仍然不能达到一个理想的结果。因此，学习率以 0.001 为最佳。

整体而言，CNN 和 RNN 模型在不同参数下取到最优值，所以统一设置的参数不一定可以反映模应的最佳效能，但由于句子补长时使用的是随机向量，因此训练出的模型会根据句子中随机向量的不同而有着细微的差别。

问题思考

> 实验训练什么时候停止是最合适的？简要陈述你的实现方式，并试分析固定迭代次数与通过验证集调整等方法的优缺点。

当模型在训练集的 loss 之差小于一定数值（模型收敛）或验证集上的 loss 开始上升时，便可以结束训练。具体而言，我选取了固定最大迭代次数为 50，若模型连续 10 次与上一次的损失之差的绝对值小于 0.001 时，则停止训练。同时在训练过程中，我保留了在测试集上效果最好的时刻为最终采用的模型，这样可以保证验证集上的结果相对好。

固定迭代次数则需要对每一个模型花费大量的时间确定最优的迭代次数，不利于训练。而通过验证集调整的方法，则可能导致模型对测试集的数据产生过拟合的现象。

> 实验参数的初始化是怎么做的？不同的方法适合哪些地方？（现有的初始化方法为零均值初始化，高斯分布初始化，正交初始化等）

高斯分布初始化/零分布初始化

使用一个均值为 μ，方差为 σ^2 的高斯分布 $N(\mu, \sigma^2)$ 对每个参数进行随机初始化。通常情况下，我们更希望均值为零（零均值初始化），然后并对生成的数乘上一个小数，并把权重初始化为很小的随机数。

不过若随机数的值很大（即权重过大），会导致激活函数 `sigmoid` 或 `tanh` 停留在平坦的边缘位置上，不利模型训练。相反，若权重过小，则会导致梯递消失的问题。

为了解决上述问题，我们需要设置一个适中的方差 σ^2 ，且时在模型中设置 `softmax` 层降低方差带来的影响。

正交初始化

由于正交矩阵的特征值的绝对值等于 1，而且正交矩阵的积仍然是正交矩阵，故正交矩阵初始化权重得到的矩阵既不会爆炸也不消失。这样可以使梯度更有效地反向传播而不用担心梯度的问题。

> 过拟合是深度学习常见的问题，有什么方法可以防止训练过程陷入过拟合。

1. 增大初始数据集的规模；
2. 增加 `dropout` 层可以降低模型复杂度（适时减少参数的数量）可以防止过拟合；
3. 利用 `Early Stopping` 的方式可以避免权值过大，导致过拟合的现象；
4. 正则化（L1, L2）可以降低输出与样本间的误差，同时控制权值的大小，防止过拟合；

> 试分析CNN，RNN，全连接神经网络（MLP）三者的优缺点。

CNN

- **优点**
  - 可以通过控制卷积核的大小来提取局部的连续特征；
  - 参数数量少，但仍有不错的效果。
- **缺点**
  - 池化操作会忽略大部分有效数据；（如本次实验中 59 个数据只有一个被使用）
  - 梯度下降算法有可能令训练结果收敛至局部最小值，而非全局最小值。

RNN

- **优点**
  - 输入长度无限制，训练更灵活；
  - 对较短的序列有记忆功能，因此十分适合处理短文本信息。
- **缺点**
  - 容易出现梯度爆炸、消失等问题；
  - 不能监测矩离较远的数据之间的关系；
  - 梯度下降算法有可能令训练结果收敛至局部最小值，而非全局最小值。

MLP

- **优点**
  - 拟合能力强；

    - 收敛速度快；
    - 可以并行计算；（只涉及矩阵运算）
- **缺点**
    - 有可能令训练结果收敛至局部最小值，而非全局最小值；
    - 容易出现模型过拟合的问题；
    - 隐藏层无明显规律，只能暴力调参。

心得体会

本次实验，加强了我对一些简单的深度学习模型的了解和认识，特别是在模型对比与分析的部分，使我明白到各个模型的优缺点。

同时，在实现模型的过程中也让我学会了如何使用深度学习框架 **Pytorch**。

总体而言，我获益良多，感谢助教和老师的付出！