情感分类

2018011359 计84 乐阳

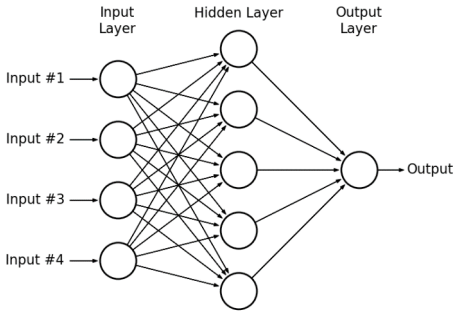
简介

本次实验完成了一个简单的中文文本的情感分类任务。选择pytorch深度学习平台，尝试了不同类型的神经网络模型，包含MLP、DAN、RNN、CNN以及RCNN，调整参数并比较性能。文本预处理时，取每篇文章的前512个中文词汇（无其他符号）作为训练数据，词汇嵌入使用预训练词向量`sgns.sougou.char`。运行说明见`README.md`。

模型结构

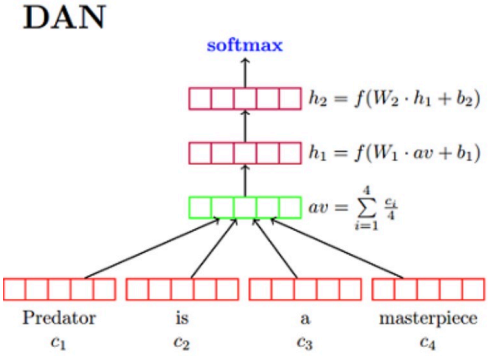
MLP

全连接网络（Multilayer Perceptron）是最简单的神经网络模型。实验中采用带有一个隐层的两层网络，以ReLU为激活函数，添加Dropout层来做正则化。每篇文章的词向量矩阵展成一维后输入网络。该模型参数数量大，加之本次训练的数据集规模较小，很容易过拟合。



DAN

DAN（Deep Average Network）是有较深网络层次的词袋模型（BOW）。其最显著的特点在于直接对一篇文章的所有词向量求平均，再输入到全连接网络中去学习，如下图中绿色的层次所示。相比朴素的MLP而言，DAN的网络规模和参数数目明显更小，训练更快且泛化能力更强。

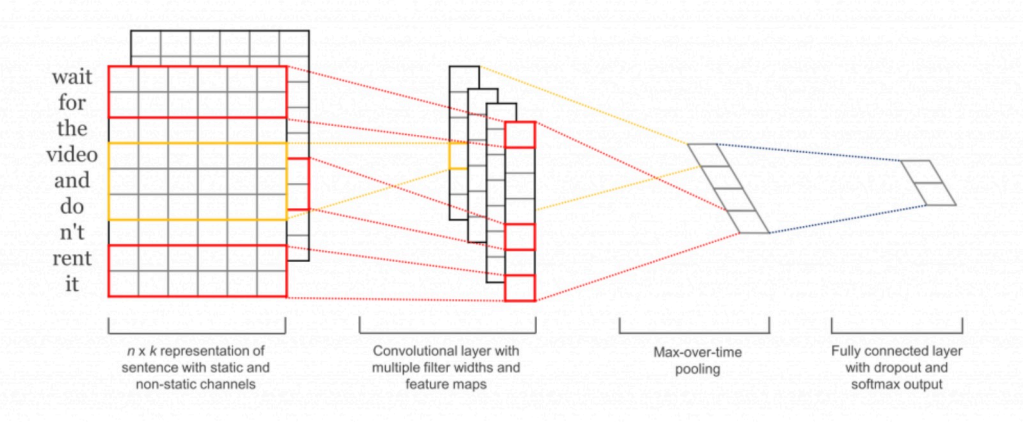


此外，还可以采用Word Dropout策略，即在求平均前随机使某些词语失效，只对未失效的词求平均。该策略可能使模型的性能有所提升。一方面，与网络结构的Dropout策略一样，会带来数据增强或类似集成学习的作用；此外一篇文本的非关键的、中立的词汇占大多数，因此丢弃一些词是可行的。

$$r_w \sim \text{Bernoulli}(p)$$
$$\hat{X} = \{w | w \in X \quad \&\& \quad r_w > 0\}$$
$$\text{average}(v) = \frac{\sum_{w \in \hat{X}} v_w}{|\hat{X}|}$$

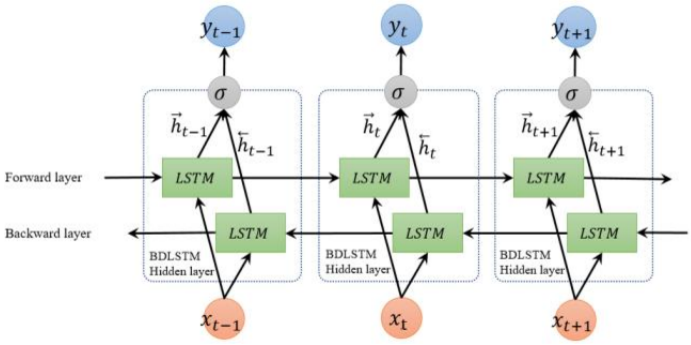
CNN

卷积神经网络（Convolutional Neural Network）使用不同大小的卷积核与词向量做卷积，再将各核的卷积结果连接在一起，最后利用全连接网络输出分类信息。使用卷积的优势在于提取局部信息，每次卷积同时检查相邻的几个词语，相当于实现了一个N-gram。针对卷积网络的防止过拟合的方法有Dropout，Batchnorm, Pooling等等。



RNN

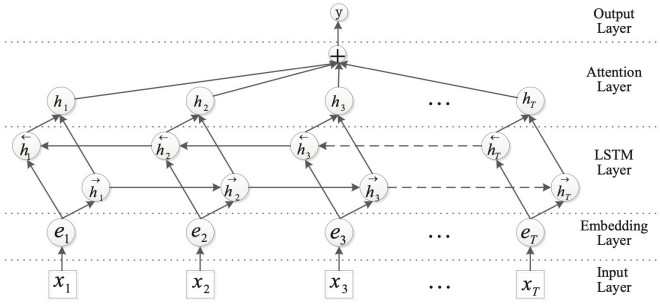
循环神经网络（Recurrent Neural Network）是一类可以接受序列输入的神经网络，常见的模型有LSTM、GRU等等。本次实验中首先尝试了双向LSTM模型，将两个方向的输出向量（最终状态）连接在一起作为文本特征交给全连接网络。经过实验，这种架构的RNN收敛很慢且效果不佳，有可能是因为文本序列较长，不考虑各中间状态的输出会损失一定的信息。



一种改进的策略是采用注意力机制（Attention），该机制综合利用了每个状态的输出。具体来讲，设LSTM的全部输出构成矩阵$H = [h_1, \cdots, h_T]$，此外还有一个模型参数w，则按照如下过程计算文本的表示r：

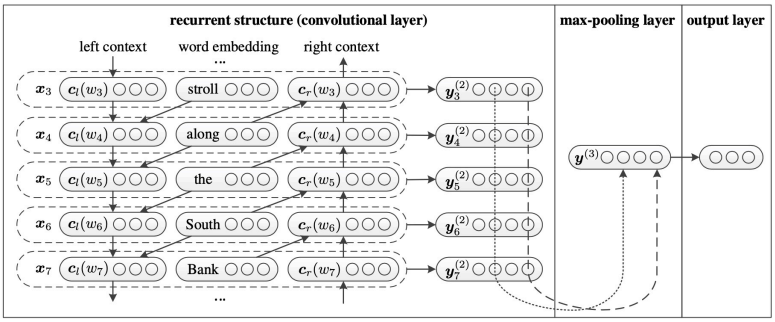
$$M = \tanh(H)$$
$$\alpha = \text{softmax}(w^T M)$$
$$r = H\alpha^T$$

该流程可表示为下图



RCNN

RCNN是一种结合RNN与CNN优势的模型。RNN提取全局特征，CNN捕捉局部特征，二者相结合能够做到优势互补。但也注意到这个模型相对前述模型变得更加复杂、参数更多，在本实验的小规模训练集上效果不一定更优。



该模型的关键一步操作是利用双向LSTM获取上下文信息的能力，将词向量与对应状态的前向、后向输出连接在一起。然后再做卷积操作，由于连接后的向量已经获得了上下文信息，因此只需要设置卷积核的大小为1即可，相当于是对每个x_i做同样的线性变换。

$$x_i = [c_l(w_i); e(w_i); c_r(w_i)]$$
$$y_i^{(2)} = \tanh(W^{(2)} x_i + b^{(2)})$$

上式中c_l, c_r为前向、后项的输出，$e(w_i)$为词向量。随后再经过MaxPooling和全连接得到输出。

$$y^{(3)} = \max_{i=1}^{n} y_i^{(2)}$$
$$y^{(4)} = W^{(4)} y^{(3)} + b^{(4)}$$

实验结果

实验中将`sinanews.test`的前1000条数据作为测试集，后1000条作为验证集。下面的测试中模型的参数已调整完毕。

单次训练的性能有一定的上下浮动，因此对以上每种模型各训练5次，对结果的准确率、F-Score、相关系数取平均值以得到性能的准确反映。其中F-Score的计算方法为宏平均（Marco）。更多的测试结果以及参数设置可见`log/test.txt`。

Model	Accuracy	Marco F1	Correlation
MLP	59.3372	22.3285	0.5946
DAN	63.8402	32.1285	0.5904
CNN	**64.3665**	**33.6428**	0.6581
RNN (Without Att)	54.9513	19.7955	0.5181
RNN (With Att)	63.8012	32.5017	0.6261
RCNN	63.4698	30.2243	**0.6659**

可见CNN在准确率和F-Score方面表现最优，RCNN在相关系数方面表现最优。DAN和MLP本质上都是全连接结构，但DAN明显更高效。加入注意力机制让RNN的性能有大幅提升。

参数调整

本次实验的几个模型的主要超参数有学习率、训练轮数、网络规模（隐层数、通道数）以及正则化强度。正则化方法包括Dropout、DAN的Word Dropout以及Adam优化器的权重衰减选项。下面对部分参数的调整分别举例。

学习率

学习率是每次迭代中参数更新的步幅，在调参过程中一般最先考虑。如果步幅过小则收敛缓慢，步幅过大则可能导致跳过最优解、很难收敛。在训练过程中也可以看出学习率过小时（训练集）正确率增长缓慢，学习率过大时正确率迅速增大到一定幅度后就不再增加。

下面是CNN在不同学习率下的性能对比，可见对于CNN而言1e-3的学习率是更好的选择。

Lr	Acc	F1	Corr
1e-4	59.8246	16.8074	0.5783
1e-3	64.3665	33.6428	0.6581
1e-2	61.9688	30.3732	0.6087

Dropout

Dropout即随机失活，往往能给模型带来额外的性能提升。它能够一定程度上缓解复杂网络的过拟合，同时加快训练速度。对于复杂网络结构的惩罚迫使模型去学习更鲁棒的数据特征；随机失活的大网络相当于很多小规模的网络的组合，起到”从人群中学习“（集成学习）的作用。

经验上讲Dropout率的默认值为0.5，在该条件下（从概率意义上）可以获得最多的网络结构。下表展示了CNN模型中Dropout率对性能的影响。本次任务中，Dropout率为0.5时性能略好，但是不同Dropout率的影响并不显著。

Dropout Rate	Acc	F1	Corr
0.1	64.2495	32.4895	0.5979
0.3	64.1520	31.3941	0.6606
0.5	64.3665	33.6428	0.6581
0.7	64.2495	32.2472	0.6639

Word Dropout

在DAN模型求平均之间，可以随机丢弃一些词汇，只对剩余词汇求平均。DAN作为词袋模型对语言的顺序关系没有要求，而且扔掉的大多是中性的、非关键的词语。与前述的网络节点Dropout类似，资料显示其对性能有所提升。

下表展示了不同的Word Dropout概率下的DAN性能，在0.5的丢弃概率下性能略有提升。

Word Dropout Rate	Acc	F1	Corr
0	63.5478	31.6139	0.5862
0.3	63.4893	31.9507	0.5660
0.5	63.8402	32.1285	0.5904

网络规模

对于神经网络而言，网络规模（隐层大小、卷积核通道数）决定了其复杂度。隐层更大的模型表示训练集数据的能力更强，但是也更容易过拟合，消耗的计算资源也更多。一个经验准则是”只有需要的时候，才使用复杂的模型“。

本次实验的数据集中，训练集与测试集的规模相等且数目不大，而且训练集与测试集的分布似乎有所差异。在实际操作过程中，往往发现网络在前10个迭代轮次里就基本将训练集拟合完毕，而测试集的正确率持续挣扎。在这种条件下使用大规模的网络并不是一个好选择。

下表中展示了CNN在不同的卷积核通道数下的性能。可见通道数最少的模型反而性能最佳。

num_filters	Acc	F1	Corr
128	64.3665	33.6428	0.6581
256	64.0741	32.6130	0.6626
512	63.9181	31.3795	0.6595

模型比较

全连接网络

全连接网络的实质即词袋模型，不考虑文本的排列顺序。朴素的MLP模型参数量巨大、训练迟缓且效果平庸。但是对数据先做一定处理，比如DAN中直接求平均词向量，再输入神经网络就能获得一个简单而快速的模型。

从统计数据可以看出DAN的性能CNN和RNN所差无几，而且实际测试中DAN是所有模型中训练最快的（只处理平均词向量，网络规模小）。此外，DAN还能有效防止过拟合，训练集acc不过高的情况下，验证集就有较好的效果。

卷积网络

卷积的优势在于提取局部信息，在NLP领域是相邻几个词的关系，CV领域则是一小块图像。这种观察让模型对数据有了更深层次的认知。每次卷积同时检查相邻的几个词语，相当于实现了一个N-gram。一层卷积不能像RNN那样获得全局信息，使用更深的卷积网络可以增加神经元的视野（本次没有实现）。

在本次实验中，CNN是表现最优的模型，且性能波动较小。不过，CNN与RNN(Att)和RCNN在本次实验中都会很快过拟合训练集（几乎百分之百），因此需要限制训练的轮数。

循环网络

循环神经网络能够处理流式的输入和输出，这个特性让它在很多文本或音频任务中有独特的地位。循环网络可以直接获取到整个序列的特征，是CNN无法比拟的。对于较长的文本序列，只采用最后一个状态的输出会丢失很多信息，需要引入Attention机制。在实验中，RNN相比其他模型训练时间更长，加入Attention机制能显著提高性能。

RCNN是RNN和CNN优势的结合，综合了CNN获取局部信息的能力和RNN获取全局信息的能力。但在本次实验中RCNN并没能体现出明显的优势。

问题思考

1. 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，并分析固定迭代次数与通过验证集调整等方法的优缺点。

   答：本次我使用的是固定迭代次数的方式，选择每轮迭代中在测试集正确率高的模型保存，期间也尝试了不同的迭代轮数以提高训练效率。固定迭代次数消耗计算资源更多，

2. 参数初始化是怎么做的？不同的方法适合哪些地方？

   答：本次实验没有使用特殊的初始化方式，直接采用Pytorch的默认方法。均匀分布初始化适用于卷积层、全连接层，这也是Pytorch的默认策略。正交初始能够防止梯度消失和梯度爆炸，常用于RNN。高斯初始化应用范围广，适用于各种需要随机初始化的模型。

3. 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？

答：Dropout通过随机失活节点防止过拟合，迫使网络去学习更鲁棒的特征。对权重添加L2正则化项可以惩罚数值过大的权重，降低模型对噪声的敏感性。数据增强（从原始数据中抽样构成新数据集）、利用多个不同模型学习等方式也能够防止过拟合。

心得体会

本次实验是我第一次接触NLP领域，见识了很多经典的模型，并对Pytorch有了更熟练的掌握。