IMDB文本情感分析实验报告

庞博琛

北京大学信息科学技术学院

1701214092

实验环境

本机的实验主要在笔记本上运行,随后在实验室服务器上运行进行GPU加速.

笔记本win10, i5-8U,no GPU

Python3.6

Pytorch 0.4.0

Visdom pytorch推荐使用的可视化库,和tensorboard有相似的作用.

实验要求

使用神经网络进行文本情感分类,数据集为斯坦福IMDB数据集

实验原理

使用Pytorch神经网络框架实现了不同的网络结果,并在IMDB数据集中进行情感分类实验,并且和前两次实验一样是使用visdom进行可视化展示。

文本情感分类可以看作是一个文本embedding+feature\_extraction+分类任务。IMDB电影评论文本分类和之前的图片分类不同，图片的数据有明确的实际意义，每张图片可以看作为shape为（w，h，channel）的三维数组，数组中每个值的取值范围是0~255之间的RGB值，具有明确的实际意义。同时表示图片的三维数组可以直接作为神经网络的输入。

**Embedding：**

原始的文本数据为字符串类型，要将其转化为神经网络需要的数值类型则需要进行Word Embedding操作，Embedding操作可以将一个字符token转换为（embedding\_dim）长度的向量，即将(len\_seq)长度的文本序列转换为(len\_seq, embedding\_dim)的二维数组，常见的Word Embedding操作包括：

1. one-hot embedding：即将word转换为长度为vocabulary\_size的向量，第idx维设为1，其余位均为0
2. 隐主题embedding：结合LDA，PLSA等主题方法得到word在不同隐主题上的分布，并得到embedding vector
3. 基于矩阵方法的embedding：比如glove模型
4. 基于网络的方法：比如word2vec和神经网络方法

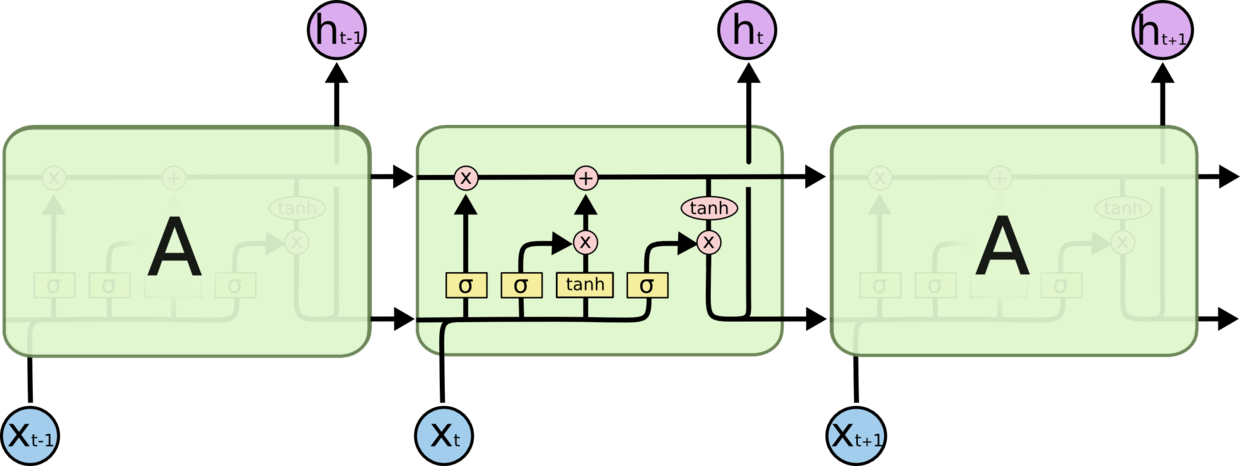
在本次实验中，我先使用one-hot embedding方法，辅以nn.Embedding()网络层，最后采取预训练好的Glove模型来提高文本编码性能。

**Feature extraction:**

经过Embedding之后，将(batch\_size, len\_seq)的评论batch字符串集转换为(batch\_size, len\_seq, embedding\_dim)的数组，可以作为神经网路的输入。

可以将不同的神经网络模型看作是文本特征提取器，发掘表示本文语义的特征向量。本次实验中使用RNN循环神经网路如lstm，bilstm，此外还就CNN文本分类网络进行实验。

LSTM（Long Short-Term Memory）是长短期记忆网络，是一种循环神经网络。



**网络结构**

LSTM通过对网络结构的设计避免了上述长期依赖（Long-Term Dependencies）问题。LSTM和标准RNN由相似的结构，但是重复的模块具有不同的结构，不同于单一神经网络层，这里是有四个层级结构，并以一种非常特殊的方式进行交互。

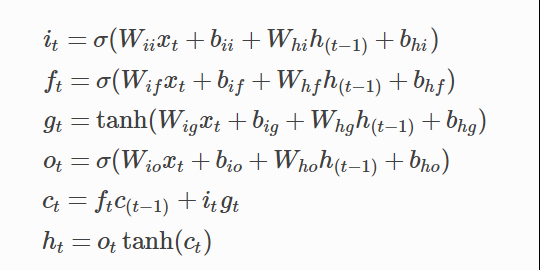
LSTM精心设计了“门”结构来给细胞状态去除或者增加信息。LSTM 拥有三个门，来保护和控制细胞状态。门是一种让信息选择式通过的办法，模拟了信息记忆个遗忘的过程。实现中"门"结构包含了一个 sigmoid 神经网络层和一个 pointwise 乘法操作。sigmoid层输出0至1之间的数值，代表每个部分由多少量可以通过。0代表全部不通过，1则指全部通过。

BiLSTM则是双向LSTM网路，是对LSTM的一种扩展

CNN网络结构在前面的实验中已经实现过。有区别的一点是CNN网络需要固定输入数据的长度（比如在图像分类中，需要在数据预处理中将图片转换为大小相同的尺寸），因此文本数据预处理时需要将序列进行截取或者padding的操作。

此外GRU也是常用的RNN网络结构，本次实验没有深入实现。

此外，LSTM模型可能有许多变种，本次实验中使用的LSTM结构为



**特征分类器**

本次实验将文本情感分类任务看作是embedding，feature extraction，feature classification三部分任务。神经网络中最后的线性层就可以看作是一个线性分类器。

此外还有一种做法：即将神经网络和传统机器学习方法结合在一起，将网络的中间层作为传统分类器的输入。本次实验在这方面也进行了尝试。

实验过程与结果

数据预处理，对原始的文本文件进行预处理，其中特殊的一步就是word embedding，将字符串转换为网络能够处理的数值形式。一开始我使用one-hot编码，但是由于模型比较简单，使用one-hot无法取得较好的结果，因此随后使用pretrained glove model进行训练。

之后构建测试数据集和训练数据集，并结合pytorch框架构建dataset\_iterator,每次返回(batch\_size, seq\_len, input\_dim)的数据进行实验

模型加载，根据不同的模型设置参数，进行训练

训练过程可视化：和之前的实验相同，使用visdom对中间的实验过程进行可视化。

模型评估：在测试集评估模型性能。

分类模型的扩展：将神经网络和传统分类器结合起来，进行对比。

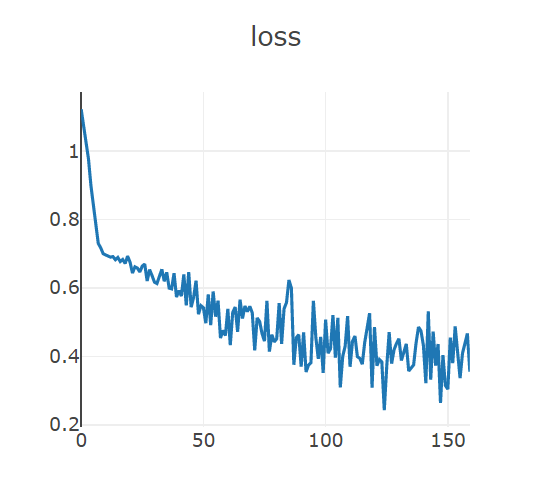
**LSTM模型**

使用LSTM.py中的模型，采用glove embedding model进行word embedding，使用pretrained glove weights。

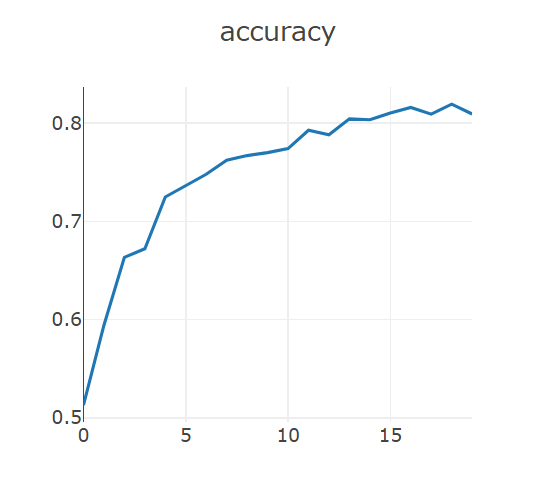
模型的参数设置为

[0613\_093536]env lstm   
[0613\_093536]hidden\_dim 128   
[0613\_093536]max\_seq\_len 200   
[0613\_093536]batch\_size 64   
[0613\_093536]embedding\_dim 100   
[0613\_093536]grad\_clip 0.1   
[0613\_093536]learning\_rate 0.01   
[0613\_093536]model lstm   
[0613\_093536]dataset imdb   
[0613\_093536]keep\_dropout 0.8   
[0613\_093536]max\_epoch 20   
[0613\_093536]embedding\_file glove.6b.300   
[0613\_093536]embedding\_training False   
[0613\_093536]lstm\_mean mean   
[0613\_093536]lstm\_layers 1

训练过程中每个50个batch的loss情况，采用cross\_entropy\_loss



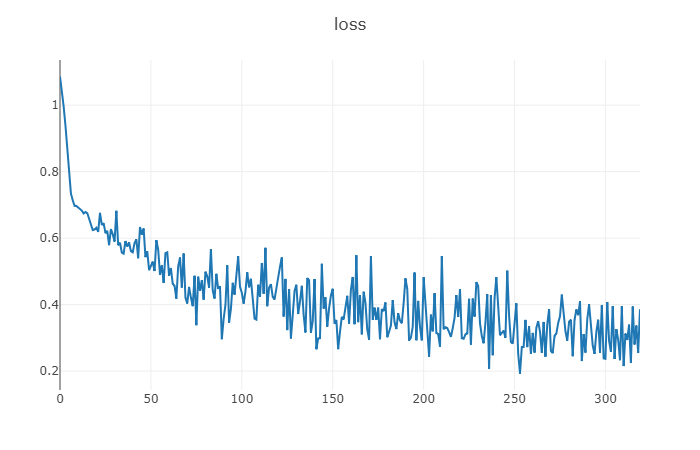
每个training epoch后模型在测试集上的分类准确性

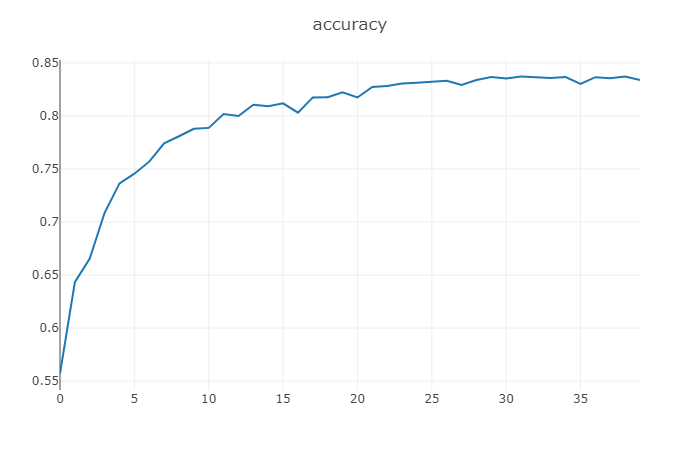


在测试集上最好的准确率达到了0.819

可以看到模型的准确率还有提升的空间，随后我将max\_epoch增加到40

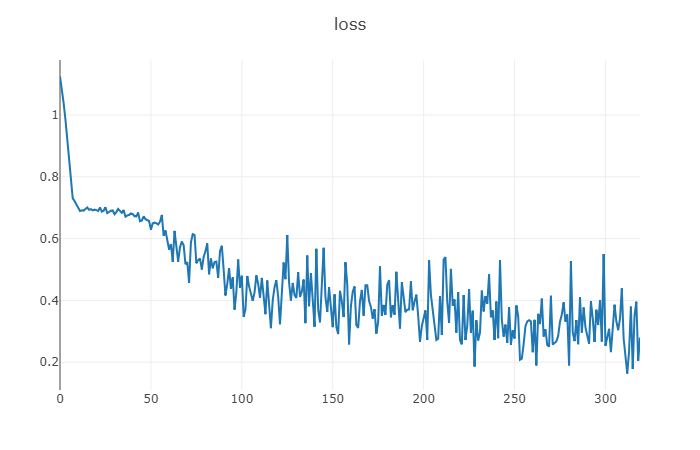
准确率提升到0.837

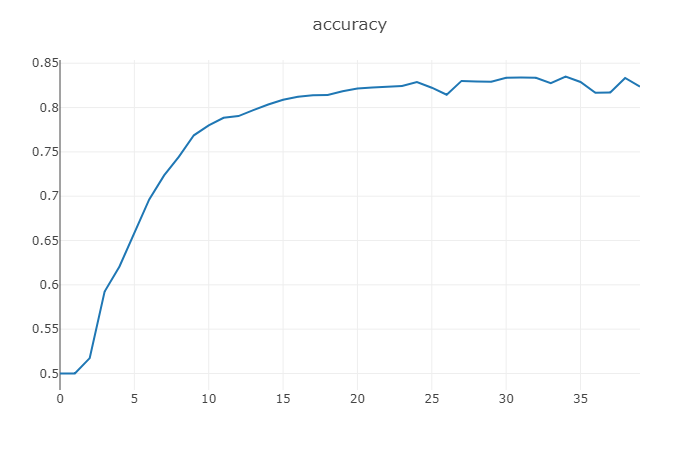




**BiLSTM模型**

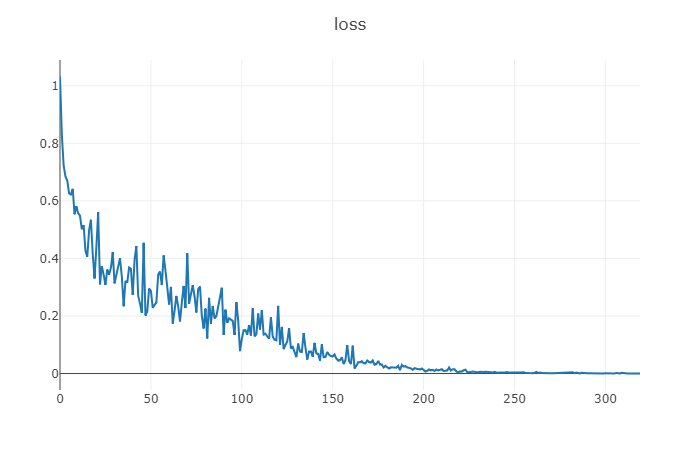
模型最好的准确率达到了0.84

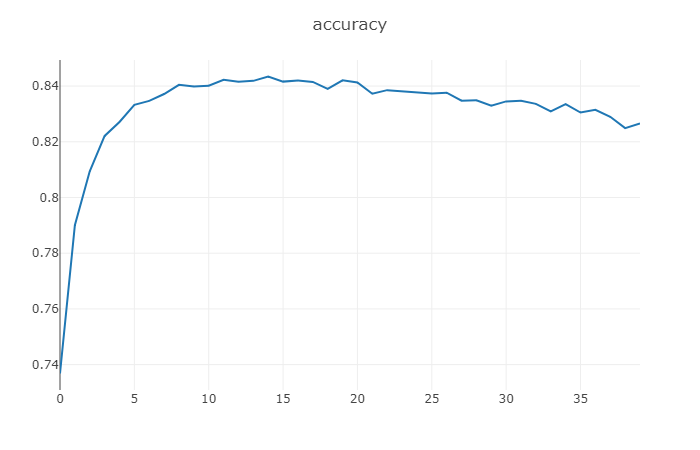




CNN模型

模型在测试集上最好的准确率为0.8423，可以看到，在后面epoch中模型出现了过拟合的情况。





将神经网络的模型和传统的分类模型结合在一起，实验效果是否有提升。

具体的做法为：

本次实验选择LSTM神经网络模型和SVM分类器进行组合，将文本序列的lstm层的输出的均值作为序列文本的特征向量，随后将其作为SVM分类器的输入进行分类。

中间遇到一个问题：即将网络模型的中间层提取出来。

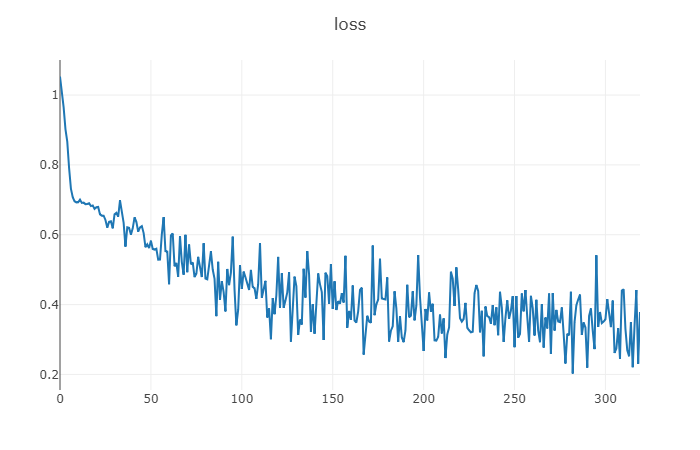
由于我使用的是pytorch框架，主要有以下三种方法：

1. 在forward函数中将网络结果和中间层数据一并返回，这个是最简单的做法，但是会改变原有的模型
2. 新建一个模型，以lstm层为输出层，这个方法很灵活
3. 使用hook方法，结合全局变量机制完成这个任务。

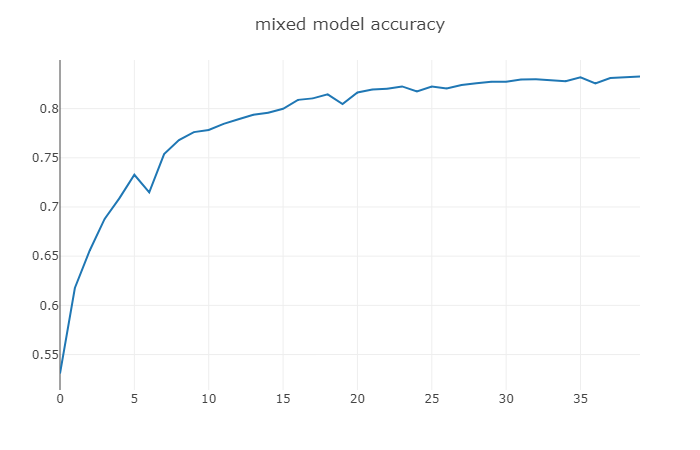
这次实验中我尝试了方法3）这个新的方法，即模型的forward\_hook方法，由于hook函数中不能return，因此使用全局变量完成文本序列特征向量提取的过程。通过和使用方法1）进行对比，我觉得对于提取中间层输出这一个任务，方法1）更加安全合适一些。

训练过程中的loss情况

（loss为LSTM网络的loss）



每轮epoch在svm模型的分类准确率



这种方法相比于原始lstm分类模型，分类准确率为0.83，模型的准确率有部分的提升