FDU-nlp-task

Task 1

实验要求：实现基于softmax / logistic regression的分类。并比较不同损失函数，特征，学习率对分类模型的影响。

实验内容：

- 数据集的划分
- 词袋模型和n-gram
- 随机梯度下降
- softmax和交叉熵
- SVM和hinge loss

数据集的划分

task1使用的数据集来自kaggle竞赛Sentiment Analysis on Movie Reviews，已经被划分为训练集和测试集，从训练集156060条数据中，随机选择16060个样本作为验证集用以调参。

词袋模型和n-gram

将文本看做是词的集合，不考虑词序，语法的影响，按词频从高到低赋予一个标签或索引，从而将文本数据转化为独热编码的形式。

n-gram可以看做是特殊的词袋模型，通过大小为n的滑动窗口截取长度为n的字词序列，每一个序列称为gram，进而形成文本的向量特征空间。

随机梯度下降

在训练模型的过程中，每次迭代取一个训练样本，计算损失并更新参数。在实验中，采用的是小批度随机梯度下降，每次迭代随机抽取一部分样本计算loss，并更新参数。。

softmax函数和交叉熵

对于一个线性分类器，采用不同的损失函数意味着不同的机器学习模型。交叉熵是信息论中的一个定义，可以度量两个分布的距离。而softmax函数可以把输出转化为一个概率分布，通过交叉熵计算与真实分布的距离，指导模型学习优化。所以softmax和交叉熵函数通常是绑定的。

SVM和hinge loss function

SVM全称support vector machine（支持向量机），是一个最优化算法，它的思想就是最大化类间最小间隔，类别间最小间隔越大说明分类器性能越好。而对于多分类问题，svm的损失函数是hinge loss function。

实验结果与分析

参数设置

task1实验使用不同的特征组合和不同的学习率来进行grid search；N1，N2，N3分别表示n-gram中的unigram，bi-gram，tri-gram，因为内存空间的缘故，各选取词频最高的1000个词组作为文本特征；学习率的组合为0.1,0.03,0.01,0.003,0.001；随机梯度下降的batch size为200，迭代次数为4000；分类模型包括softmax和svm。

实验结果

用训练集数据训练模型，在验证集进行实验，选择在验证集上表现最好的一个参数组合参与测试集的预测。 softmax在验证集分类结果如下：

features	1e-1	3e-2	1e-2	3e-3	1e-3
N1,N2,N3	0.546887	0.532690	0.522354	0.511395	0.510212
N2,N3	0.519738	0.512267	0.510212	0.512012	0.510212
N1,N3	0.545890	0.530262	0.520486	0.511083	0.510212
N1,N2	0545455	0.532379	0.522790	0.511208	0.510212

SVM在验证集分类结果如下：

features	1e-1	3e-2	1e-2	3e-3	1e-3
N1,N2,N3	0.567746	0.543960	0.526588	0.510523	0.510212
N2,N3	0.525280	0.514633	0.510149	0.512012	0.510212
N1,N3	0.563823	0.541096	0.523412	0.510212	0.510212
N1,N2	0.565131	0.541594	0.526961	0.510274	0.510212

横向比较，更大的学习率往往意味着更好的表现，说明对于n-gram特征的参数空间，小学习率可能使模型陷入鞍点，降低分类精度；纵向比较，对模型分类准确率的影响，N1>N2>N3，但在学习率较小时，所有的特征组合分类结果都是0.510212，一方面是因为鞍点的影响，模型陷入局部最优，另一方面task1的数据集是由一万多条电影评论拆分成150000条样本数据，有大概率unigram，bi-gram，tri-gram的特征空间具有很高的相关性，所以四个特征组合有可能陷入了同一个局部最优点。采用不同的损失函数，svm的分类精度比softmax要高一些。

最终，softmax和svm在测试集上的得分为0.55050和0.56533

Task 2

实验要求：用Pytorch重写《任务一》的分类器；随机embedding的初始化方式，用glove 训练出来的文本初始化；实现CNN、RNN的文本分类；

实验内容

- 文本数据序列化
- 词向量的加载
- TextCNN和RNN

文本数据序列化

文本数据序列化的目的是向量化文本，为embedding做好准备。实现文本数据序列化分为以下几个步骤：

- 数据预处理：大小写转化，去标点符号，分词
- 构建词表：文本数据分词后，遍历所有样本，把不重复的单词放入到一个词表中，然后赋予每个独立单词一个数字标签
- 序列化：再次遍历样本，把样本看做词的集合，把词语映射为词表中对应的标签，实现文本的向量化

词向量的加载

随机初始embedding不需要其他的操作，加载glove词向量，需要根据文本序列化中构建的词表，建立词向量的嵌入矩阵，字词的数字标签对应词向量矩阵的下标。实验使用的词向量是glove.twitter.27B.50d.txt，cnn，rnn的embedding层的权重参数加载词嵌入向量。

TextCNN和RNN

TextCNN用一维卷积核来代替视觉任务中的二维卷积核，然后通过不同大小的卷积核来提取类似于n-gram形式的文本特征，Alexnet等卷积神经网络是通过“串联”叠加卷积层，把低级的特征加以组合衍化成高级复杂的特征，TextCNN则是把网络“并联”起来，统合没有显著依赖性的各种文本特征。

循环神经网络是一种具有记忆能力的深度模型，可以通过历史信息和当前输入计算当前输出。相比于前馈网络类似于树型的数据结构，循环神经网络的神经元可以接受上一层神经元以及自身的输入，形成了环型的网络结构。实验中使用的是GRU网络，因为对于这个任务pytorch的RNN基本不学习。

实验结果与分析

参数设置

文本最大序列长度为50，词向量维度为17821*50，epoch为20，初始学习率为1e-3，优化方法为Adam，batch_size为256，TextCNN卷积核大小为3,4和5，卷积核数为100，GRU的隐藏层维度为50

实验结果

模型	得分
CNN	0.55952
CNN(pre)	0.61764
CNN(pre+fre)	0.61757
GRU	0.61087
GRU(pre)	0.64226
GRu(pre+fre)	0.64196

CNN，GRU表示模型使用的是随机初始化的embedding，(pre)用的是glove预训练词向量，(pre+fre)在使用glove embedding的基础上，冻结了embedding层的参数，即词向量参数不随训练更新。从上表中的结果可以看出，深度学习模型的效果比task1中传统的机器学习模型效果更好。其次，迁移训练好的词向量可以使分类精度进一步上升。在TextCNN的论文《ConvolutionalNeuralNetworksforSentenceClassification》中，作者提到，在预训练词向量的基础上进一步fine tune嵌入层的参数可以提升模型的性能，但在task2的实验中并没有体现出这一点。

Task3

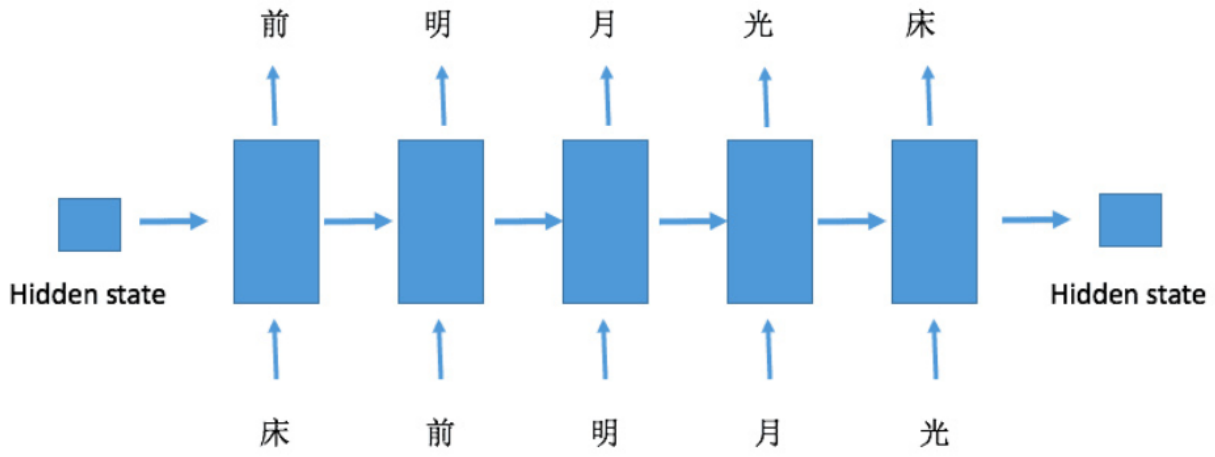
实验要求：用LSTM、GRU来训练字符级的语言模型，计算困惑度并生成文本

实验内容：

- RNN模型的训练
- 文本生成
- 困惑度的计算

RNN模型的训练

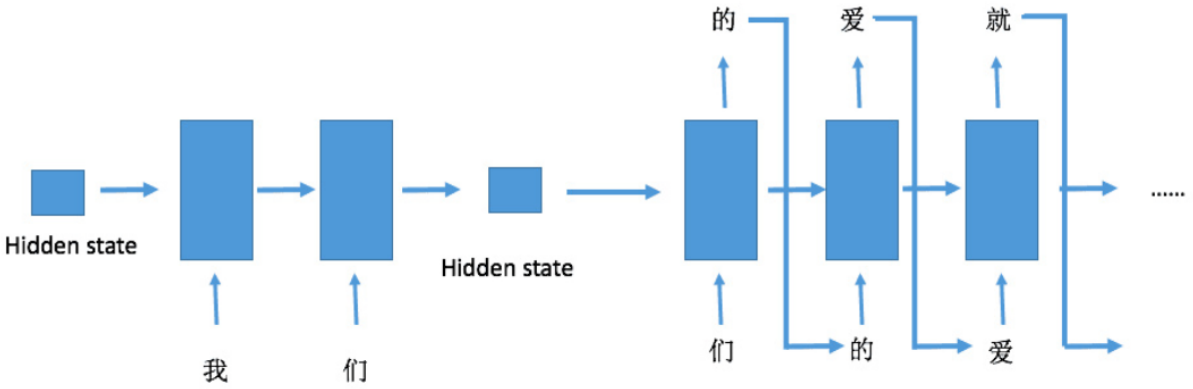
文本的生成分为两个阶段，一是构造训练数据进行有监督学习，然后再进行文本的一系列生成，RNN模型训练，其具体过程如下图所示：



实质上是一个多对多类型的RNN，输入一句唐诗，每一个字符的输出是其在输入中的下一个字符，输出的最后一个字符对应输入的第一个字符

文本生成

文本生成的过程如下图所示：



先输入一段文本对模型进行预热，生成拥有记忆效果的隐含层状态，再利用保留的隐含层状态进行不断的文本生成。

困惑度的计算

模型困惑度的定义为：

$$2^{H(pr,p\theta)} = 2^{-\frac{1}{N}\sum_{n=1}^{N} log_2 p\theta(x^{(n)})}$$

简单来说就是2的交叉熵次方

实验结果与分析

参数设置

词向量的维度为128，隐含层的维度为128，epoch为25，初始学习率为0.01，优化方法为Adam。

实验结果

GRU在经过25次epoch的训练后困惑度为5.751，输入“夜卧高丘梦神”，生成文本为：夜卧高丘梦神神知妾天漏下下南，隅寒蒲城阙百衔树临临临流萦玉薄过楼楼珠争玉

LSTM训练后的困惑度为4.203，输入“荆王猎时逢暮”，生成文本为：荆王猎时逢暮何古人世问平白草平知，日世事平朝，时东开，平朝云平十高高十古

Task 4

实验要求：用LSTM+CRF来训练序列标注模型

实验内容：

- 数据分析和预处理
- CRF

数据分析和预处理

task4使用的数据集是conll 2003，其数据的基本格式为： JAPAN NNP B-NP B-LOC 每一行代表一个识别样例，第一项为单词，第二项为pos标签，第三项为chunk tag，最后一项为命名实体标签。而命名实体分为4种：PER,LOC,ORG,MISC，分别代表人物，地点，组织和其他命名实体。原先的conll 2003数据集采用的是IOB标注法，为了方便后续实验，本任务使用的数据集是改用BIO标注的conll 2003。“train.txt”对应的是原数据集的“eng.train.txt”，“testa.txt”对应“eng.testa.txt”，“testb.txt”对应“eng.testb.txt”。 数据特征只考虑根据文本加载的glove词向量。

CRF

ner问题常在LSTM后加上一层CRF，CRF在最后一层应用进来可以考虑到概率最大的最优label路径，可以提高指标。

对于LSTM的输出，CRF层的输入X，其对应的输出tag为y，定义得分为：

$$S(X,y)=\sum_{i=0}^{n}A_{y_{i+1},y_i}+\sum_{i=1}^{n}p_{i,y_i}$$

$A_{i,j}$ 代表的是tag间的转移概率，$p_{i,j}$ 代表词i到tag j的非归一化概率。利用softmax函数，为正确的标签y定义一个概率:

$$P(y\mid X)=\frac{e^{S(x,y)}}{\sum_{\tilde{y}\in Y_X}e^{S(X,\tilde{y})}}$$

损失函数为上式的负对数似然：

$$-log(P(y\mid x))=log\sum_{\tilde{y}\in Y_X}e^{S(X,\tilde{y})}-S(x,y)$$

上式右侧第一项表示输入到输出全路径的分数，第二项是正确标签的得分，比较容易计算。而全局路径分数可以通过动态规划的方法计算。

计算过程可以参考这个博客： CRF layer

代码的实现可以参考pytorch官方教程： pytorch BiLSTM+cRF

实验结果

经过训练后，LSTM+CRF在testa，testb两个数据集的预测结果为：

dataset	precision	recall	f1
testb	0.814164	0.699310	0.752379

实验结果没有达到预期，一方面可能是数据特征太单一，只使用了文本信息，另一方面模型不够完善。可以尝试在特征中加入n-gram，pos tag等信息，用BiLSTM代替LSTM。

此外，模型的实现调用了pytorch-crf，刚开始的时候我是用了pytorch官方的那个模板，想要实现自己的一个CRF模型，可是那个模板一次只能处理一个样本，我花了很多时间把这个改成向量化实现，就是能够处理批数据的形式，主要的几个函数都改好了，只剩下Viterbi算法解码的模块。接下来的工作可能就是参考pytorch-crf的源码完成这个模型。