2021.11

黄元通

20195063

自然语言处理

实验报告

目录

[基础内容 搭建LSTM网络 2](#_Toc87479056)

[一、题目 2](#_Toc87479057)

[二、网络结构设计 2](#_Toc87479058)

[1、计算公式 2](#_Toc87479059)

[2、程序实现 3](#_Toc87479060)

[三、传入、传出及其他细节 4](#_Toc87479061)

[1、输入参数及维度 4](#_Toc87479062)

[2、运行设备统一 5](#_Toc87479063)

[3、输出参数及维度 5](#_Toc87479064)

[4、未传state时的默认初始化 7](#_Toc87479065)

[四、结果 8](#_Toc87479066)

[提高内容 搭建双层LSTM网络 10](#_Toc87479067)

[一、题目 10](#_Toc87479068)

[二、整体设计 10](#_Toc87479069)

[1、编译环境 10](#_Toc87479070)

[2、注释 10](#_Toc87479071)

[附录：源码 11](#_Toc87479072)

[一、八皇后 11](#_Toc87479073)

# 基础内容 搭建LSTM网络

## 一、题目

尝试自己搭建LSTM网络

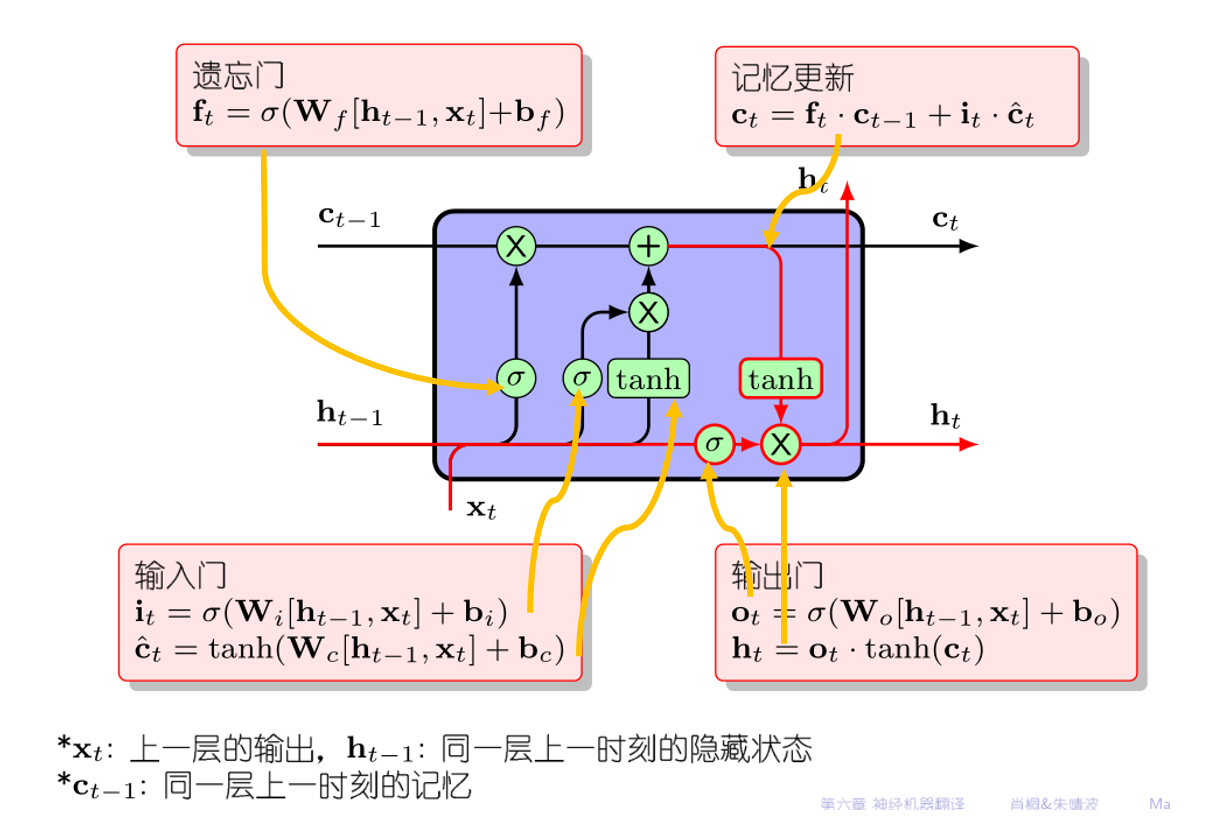
* 1. 不能调用nn.LSTM、nn.LSTMCell，可以使用nn.Linear、nn.Parameter等等搭建网络
  2. 可以参考torch.nn.LSTM的计算公式、可以仿照其输入输出，官方文档： https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html#torch.nn.LSTM
  3. 数据加载、模型训练的代码都是现成的，只需要完成模型搭建

## 二、网络结构设计

### 1、计算公式

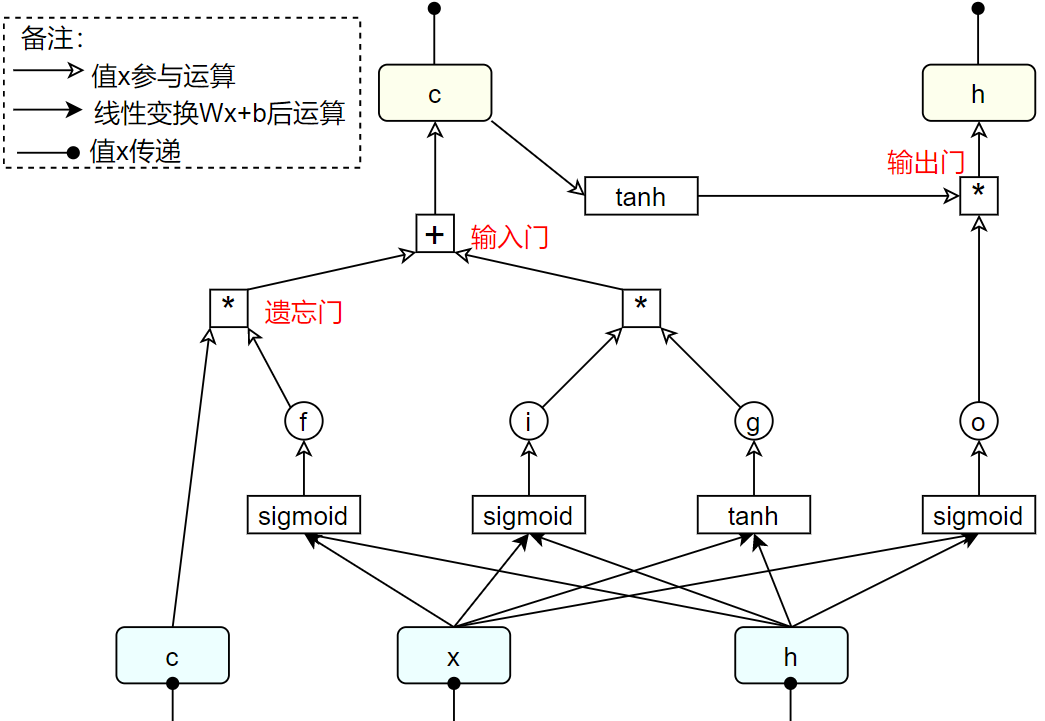
参考torch.nn.LSTM，其计算公式如下：

其中，为遗忘门，（新记忆占比）和（新记忆）共同组成输入门，为输出门，为输入（或上一层输出），为记忆。其方程与LSTM结构图对应如下：



### 2、程序实现

通过torch.nn中的Linear、Sigmoid、Tanh函数实现上述功能，则其网络结构图如下：

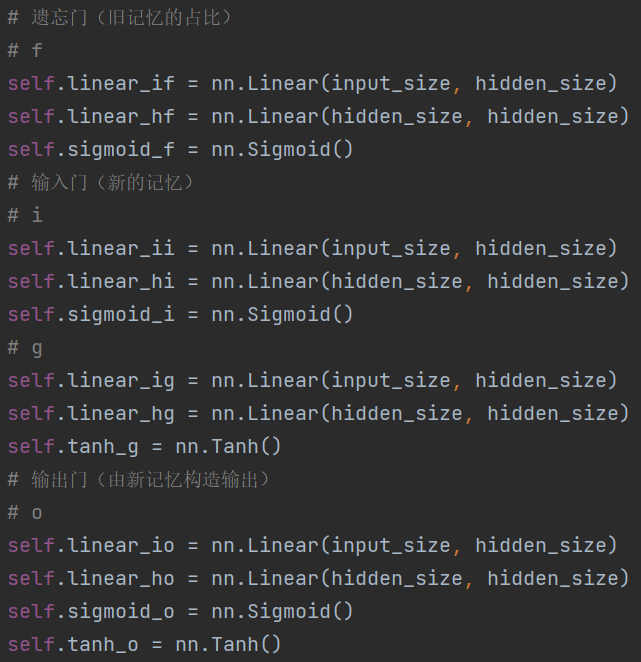


因此网络非线性，无法使用nn.Sequential( )进行封装，因此选择在自实现类LSTM时，通过在forward( )函数中手动指定计算流程。

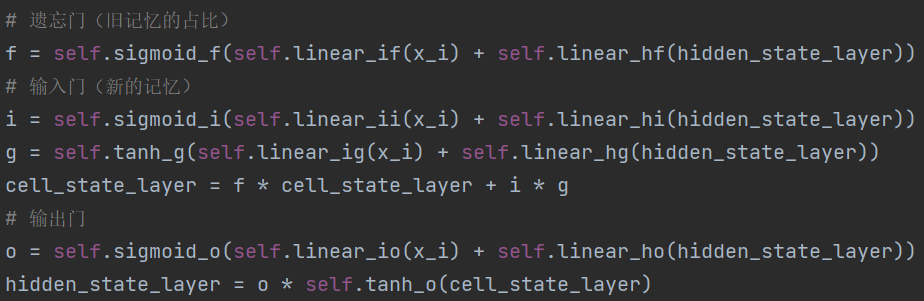
所有运算的线性层：输入输出维度分别为：input\_size、hidden\_size；

所有运算的线性层：输入输出维度都别为：hidden\_size。类的代码实现如下：

1. 在\_\_init\_\_函数中添加如下图所示的函数组件：



1. 在forward( )中，计算过程如下图所示：

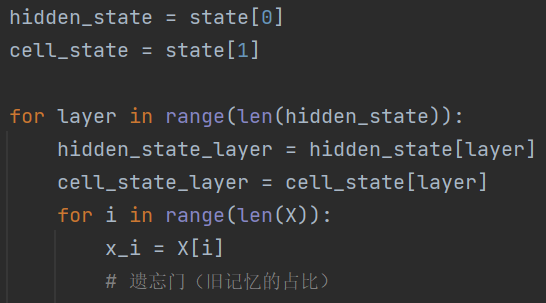


## 三、传入、传出及其他细节

### 1、输入参数及维度

在主程序（LSTMLM.py）中，通过以下语句进行LSTM调用：

outputs, (\_, \_) = self.LSTM(X, (hidden\_state.to(device), cell\_state.to(device)))

1. 参数含义：
   1. X为输入数据
   2. hidden\_state为外界定义的LSTM胞体初始化h值
   3. cell\_state为外界定义的LSTM胞体初始化c值
2. 各参数的shape，及在本实验中的值：
   1. X：[n\_step=5, batch\_size=128, embeding size=256]
   2. hidden\_state：[num\_layers=1 \* num\_directions=1, batch\_size=128, n\_hidden=256]
   3. cell\_state：[num\_layers=1 \* num\_directions=1, batch\_size=128, n\_hidden=256]
3. 各维度含义：
   1. n\_step一层LSTM的循环个数，即每次使用的句子长度（单词个数为5）
   2. embeding size为外界定义的LSTM胞体中间参数的维度，即LSTM的hidden\_size
   3. batch\_size为批训练时，一次所用数据个数
   4. num\_layers为LSTM层数
   5. num\_directions标识LSTM方向，等于1时为单向，等于2时为双向
4. 数据类型分别为(tensor, tuple(tensor, tensor))，因此，在自实现的LSTM类中，其forward( )函数进行如下处理：

### 2、运行设备统一

因为在主程序中可能会将网络及各参数运行设备设置为cpu（即CPU）或cuda:0（即GPU），因此自实现的该LSTM网络也应该运行在相同的设备上。

1. pytorch的选取原则：

首先使用nn.LSTM进行测试。

因为在主程序中hidden\_state、cell\_state的运行设备为CPU，而X的运行设备为GPU（实验所用电脑为支持cuda运行的环境，代表更广泛的情景），通过测试发现，若使用如下语句进行运行会发出设备不统一的报错。

outputs, (\_, \_) = self.LSTM(X, (hidden\_state, cell\_state))

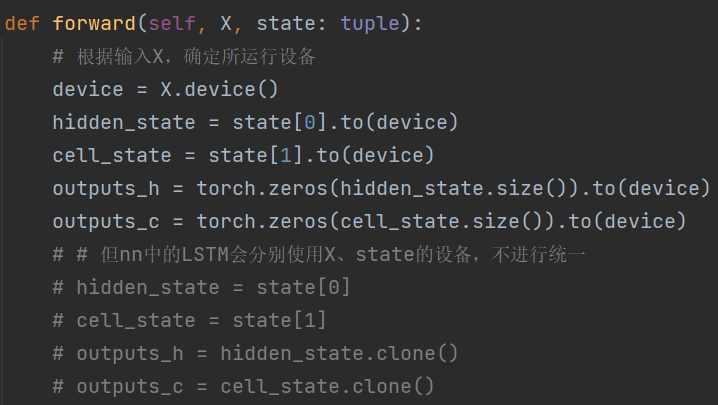
也即pytorch官方默认使用X、hidden\_state、cell\_state他们各自传入时的设备。

1. 自实现的选取原则：

但考虑到X、hidden\_state、cell\_state若有运行设备不统一的情况无法运行，这样需要编程人员手动显式地在调用前将设备统一。

因此为了降低编程人员的负担，在自实现LSTM时决定在胞体内将所有变量都设置到X所在的设备上。

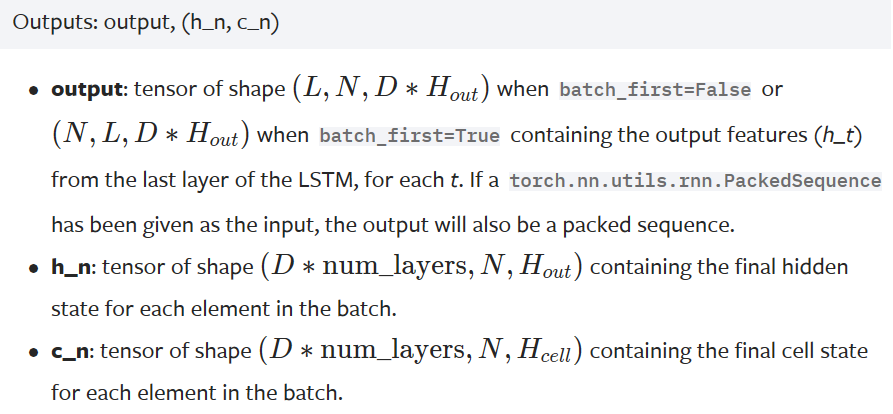
forward( )函数更新为下：



### 3、输出参数及维度

1. 参数含义：

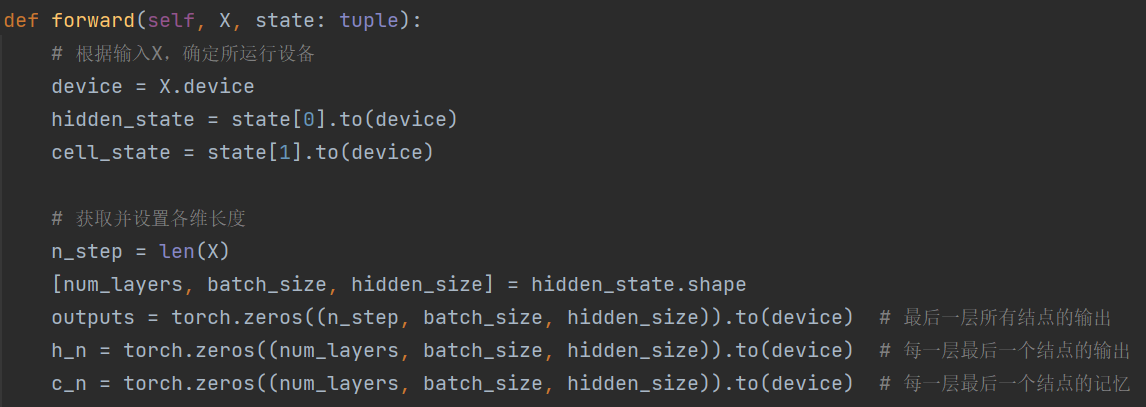
参考pytorch官网给出的LSTM网络的输出：



即：

* 1. output保存了最后一层，每个time step的输出h，如果是双向LSTM，每个time step的输出h = [h正向, h逆向] (同一个time step的正向和逆向的h连接起来)。第一维表示序列长度，第二维表示一批的样本数(batch)，第三维是 hidden\_size(隐藏层大小) \* num\_directions。
  2. h\_n保存了每一层，最后一个time step的输出h，如果是双向LSTM，单独保存前向和后向的最后一个time step的输出h。
  3. c\_n与h\_n一致，只是它保存的是c的值。

1. 在自定义的LSTM结点实现时，使用outputs、h\_n、c\_n分别存储上述参数output、h\_n、c\_n。n\_step、num\_layers、 batch\_size、hidden\_size值获取过程，以及outputs、h\_n、c\_n定义如下更新后代码所示：



1. outputs、h\_n、c\_n值的获取，如下更新后代码所示：



### 4、未传state时的默认初始化

在主程序（LSTMLM.py）中，还可通过以下语句进行LSTM调用：

outputs, (\_, \_) = self.LSTM(X)

1. 因此需要添加在未传入LSTM胞体默认状态state时，实现默认初始化，因此将forward( )函数state的默认值设为None
2. 此时num\_layers、hidden\_size因为未传入state，其值的获取途径也需要做相应改变。可放入初始化函数\_\_init\_\_当中获取
3. num\_layers的值也因此需要添加在\_\_init\_\_的参数列表，默认值为1

修改后的代码如下：



## 四、结果

1. 使用自定义的LSTM层时，使用以下命令均可成功运行，如图所示：

outputs, (\_, \_) = self.LSTM(X) （图1.4.1）

outputs, (\_, \_) = self.LSTM(X, (hidden\_state, cell\_state)) （图1.4.2）

图1.4.1 使用self.LSTM(X)，使用默认初始化方法

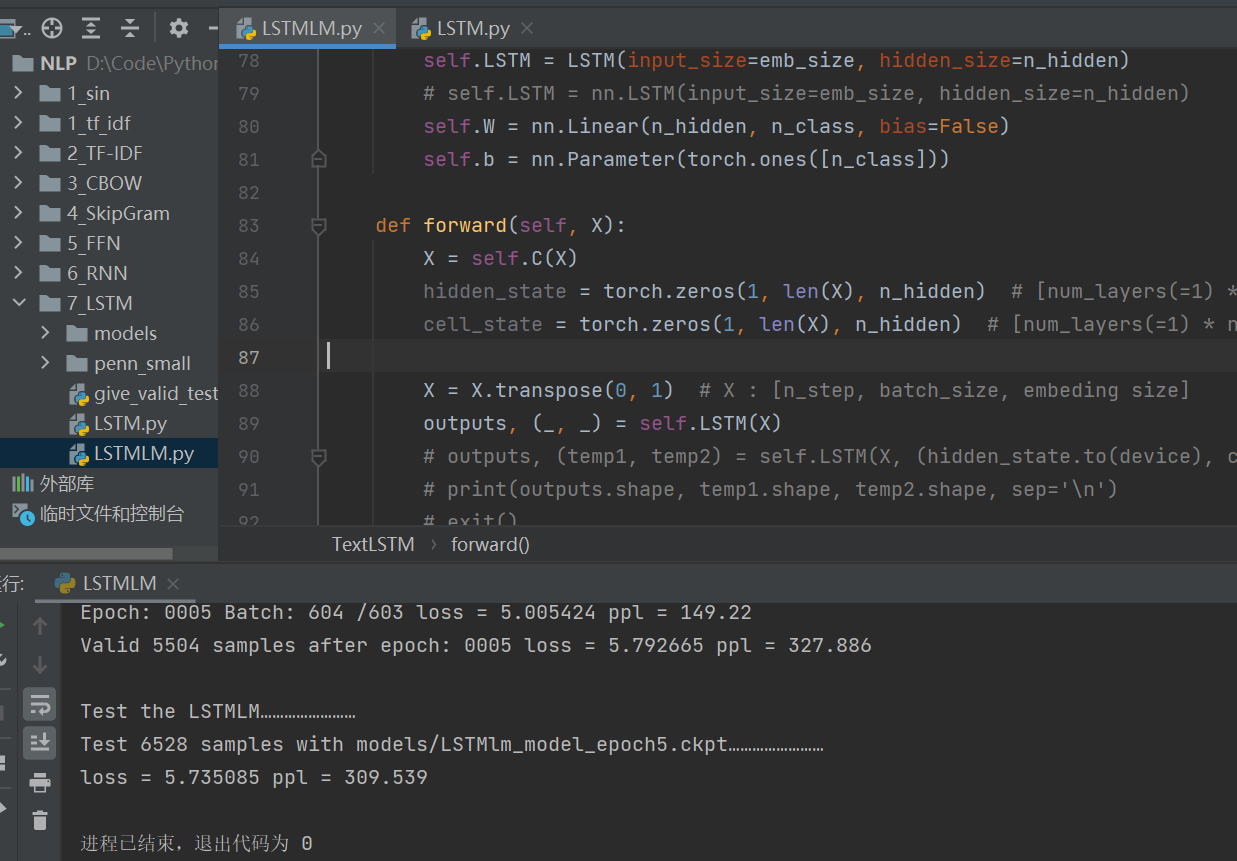
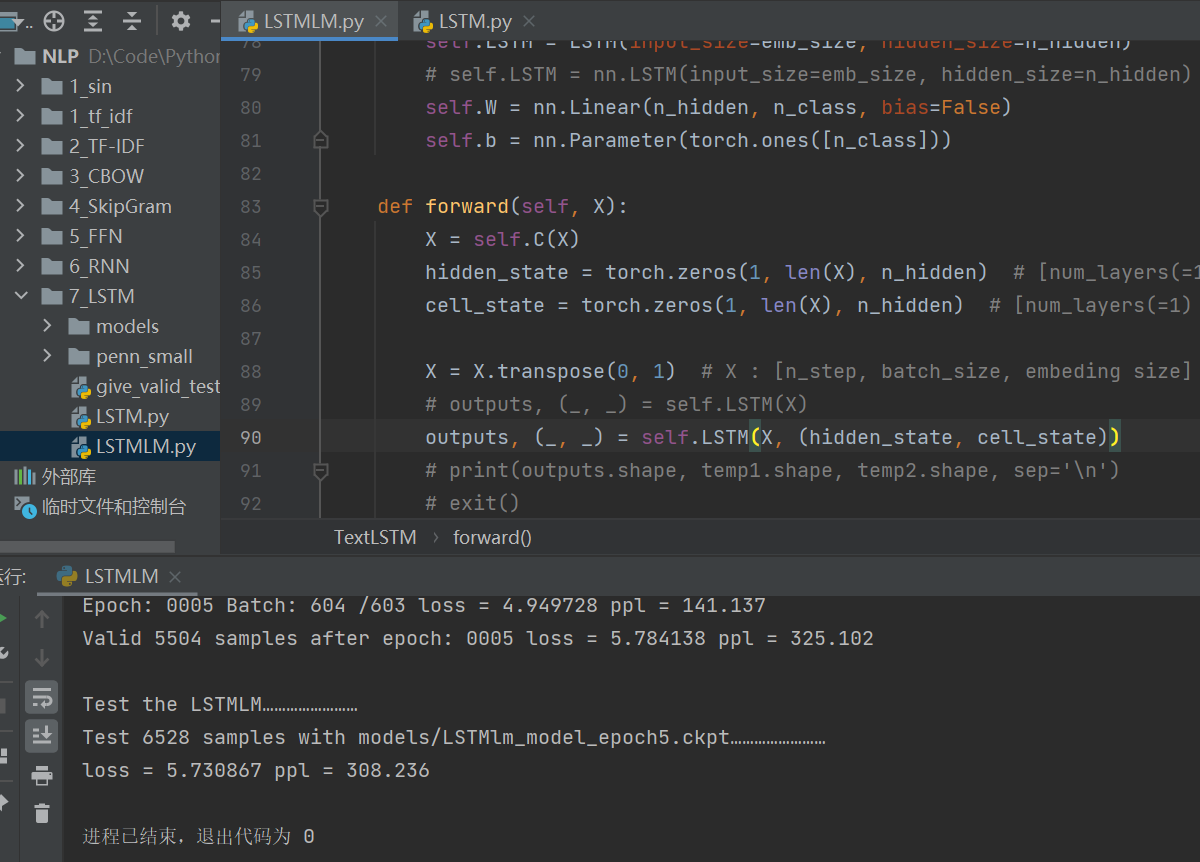


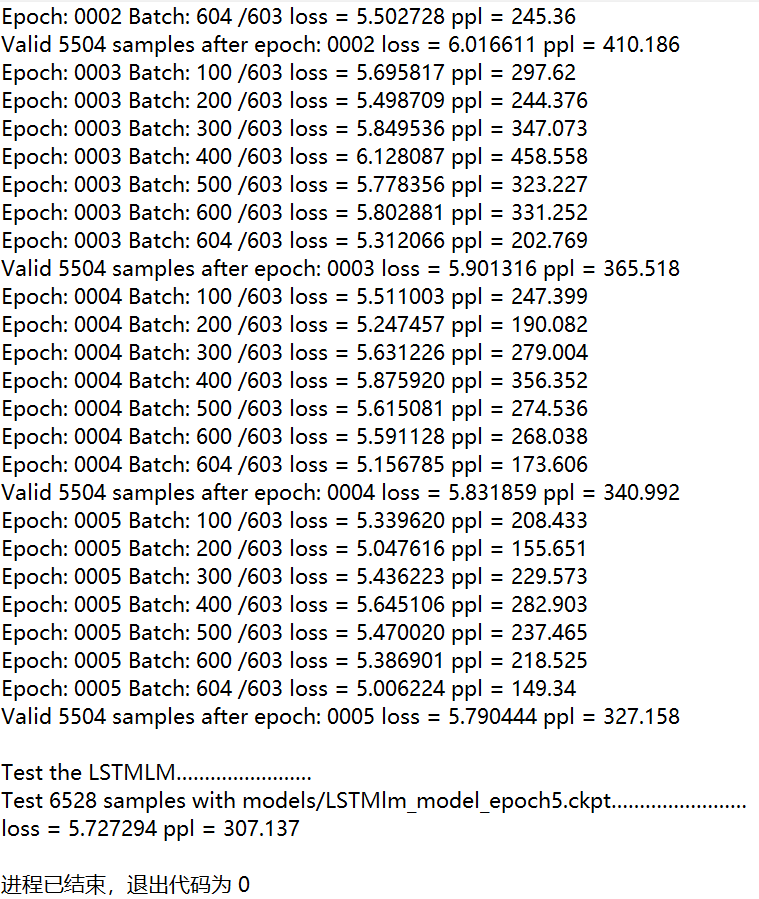
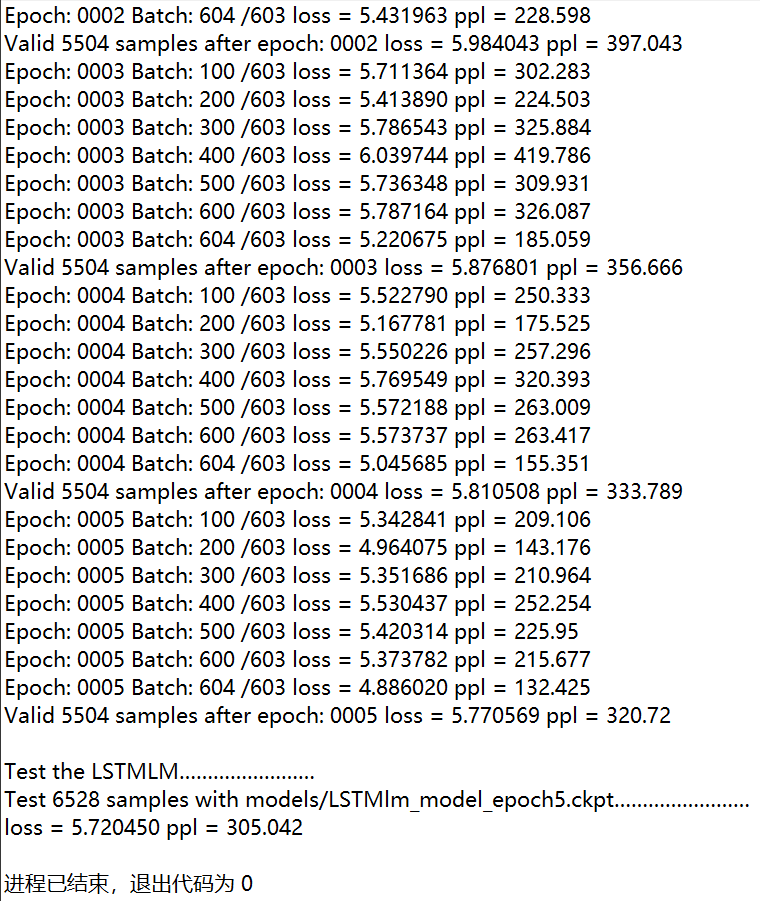
图1.4.1 使用self.LSTM(X, (hidden\_state, cell\_state))，使用外部参数



1. 通过多次运行测试，将使用自实现的LSTM和nn.LSTM的两种情况进行比较，发现自实现的LSTM和nn.LSTM在最终（epoch=5的训练情况下）运行后，考虑到波动性将多次测试所得值进行综合计量，其ppl值基本一致。在此训练情况下均在大致区间内。

如下图所示，左侧图1.4.3为使用nn.LSTM进行测试最终所得结果，右侧图1.4.4为使用自定义的LSTM进行测试最终所得结果。

图1.4.3 使用nn.LSTM所得结果 图1.4.4 使用自定义的LSTM所得结果



# 提高内容 搭建双层LSTM网络

## 一、题目

在自己搭建出来的LSTM网络基础上，实现双层LSTM网络。

## 二、整体设计

### 1、编译环境

1. 使用语言：C++
2. 使用IDE：Microsoft Visual Studio Community 2019，版本 16.10.0
3. 本报告中图片作图软件：draw.io.exe

### 2、注释

1. 程序.h文件：Data.h、Iris.h

# 附录：源码

## 一、八皇后

#### main.cpp

#include <iostream>

#include <stdio.h>

#include <Windows.h>

#include "Climb.h"

#include "Heredity.h"

using namespace std;

int main()