实验报告

目的

利用线性回归模型和One-Hot Encoding完成命名实体识别的BO二分类任务。

环境与工具

```
python 3.8.8
pytorch 1.8.1
numpy
seaborn
```

源代码结构

- `/main.py`
  调用工具类完成数据加载、数据格式化、模型训练与模型评估
- `/utils`
  工具类与工具函数
  - `/utils/DataAgent.py`
    文本数据加载、清洗、管理与生成模块
    - `class DataAgent`
      从原始数据文件加载带标注的文本，进行初步分割与处理。为下文中 `Encoder` 批量数据生成提供基础。
  - `/utils/Encoder.py`
    数据向量化编码模块
    - `class EncInterface`
      提供 `encode()` 抽象接口。
      因One-Hot编码特征导致接口函数效率较低，因此训练时实际采用的是下文继承本接口的 `class OneHotEncoder` 提供的 `para_fast_enc()` 。
    - `class OneHotEncodedPara`
      用于组装向量的工厂类。
      One-Hot编码结果为稀疏矩阵，直接在编码函数生成稀疏矩阵不利于功能分离，且会导致函数可读性下降。本类接收label来初始化，在填充完毕后后调用 `.data()` 展开成为稀疏矩阵并返回。
    - `class BIOLabeledPara`
      可根据修饰过的词标注信息生成BIO标签向量的工厂类。
    - `class OneHotEncoder`
      编码器核心

接收DataAgent来初始化。在初始化时根据数据库动态生成指定词数的高频词词典用于编码。提供以单个词为最小单位的基础编码函数 `encode()` 和以段落为最小单位的快速编码函数 `para_fast_enc()` 。
`para_fast_enc()` 针对滑动窗口的编码方式进行了优化，较基础版本显著提升了编码效率。

    - `/utils/util_func.py`
      用于容纳一些无法归类的工具性函数。
- `/plot.py`
  读取训练日志并绘制折线图的简易绘图程序。
- `/data`
  用于存放原始文本数据的文件夹。
- `/model`
  用于存放训练模型参数和训练日志的文件夹。

算法与参数选择

运算方式

单热编码的稀疏矩阵在转化为线性回归模型要求的float tensor后空间占用极大。如采用cuda加速需要通过分batch方式将GPU运存占用缩减至数GB每批来避免抛出异常。这势必会带来极大的数据复制与对象构造开销。
而线性回归模型决定了其运算量较传统模型运算量较小，cuda加速相对cpu直接计算带来的运算时间节省并不多。
因此在本次实验中未使用cuda来加速训练。

优化器与学习率

本次实验我选用了较为出色的Adam优化器。
不过，尽管优化器本身通过设计避免了很多优化陷阱，学习率作为重要的一个超参仍对训练结果有很大的影响。
因此，我选用了 `ReduceLROnPlateau` 来实现学习率的动态调节。

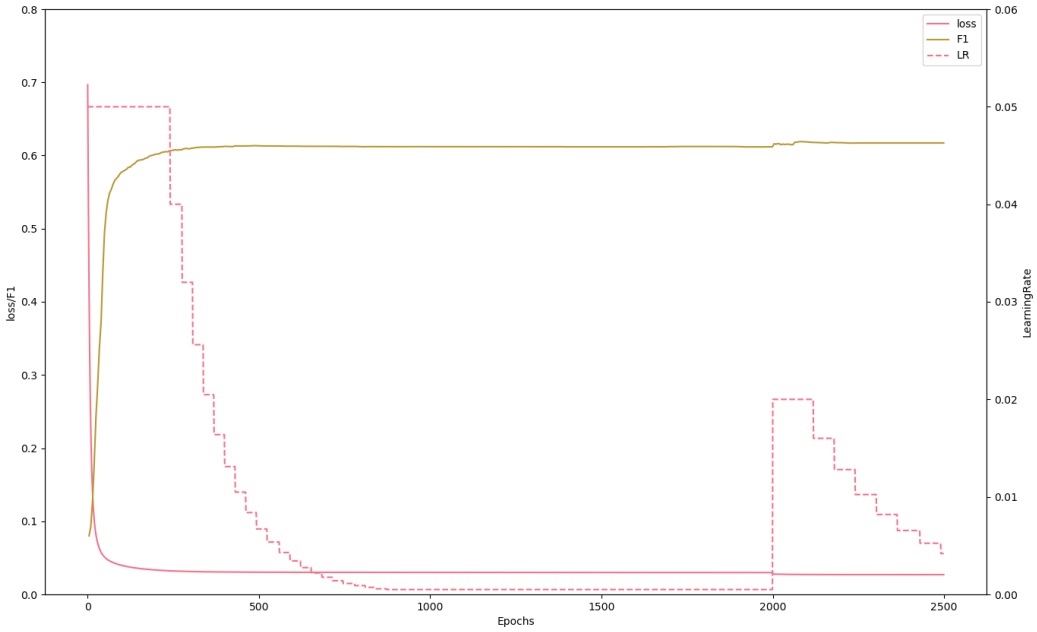
数据加载模式

因未使用cuda加速，全量数据加载成为了可能。经测试，全量数据加载到内存的情况下，主流桌面cpu每epoch耗时仅2s左右。
但是，这也带来了极高的内存占用。随内存空间缩小越过某个界限后，可能因为频繁内存交换导致耗时激增。
因此，本加载方式不适用于所有运行情况。

实验结果分析

图中可见，在训练前期loss迅速下降，F1随之上升，学习率保持初始设定的较高水平。
进入稳定期后二者变化速度趋于0，在动态调节作用下学习率开始被自动调小。

在epoch计数2000时，我手动重置了学习率。伴随这次重置loss和F1的表现均出现了小幅提升，随后再次稳定。



结论

本实验成功实现了一个简单的命名实体识别二分类器。

但是，对于自然语言识别这个复杂的问题来说，窗口长度为3的单热编码不能充分高效地提取文本特征，同时线性回归也不能有效地学习这种特征。可以预见到在后续实验中更换更合适的编码方式和网络模型后，能取得更好的效果。