阶段二思路

第一阶段我使用的是 MATLAB 滑动平均，在本人电脑上单纯滑动平均法来说用 MATLAB 速度比 Python 块一倍，但是第二阶段可能需要搭建模型等，就改用 Python 编程解决问题。

将 train_step2.csv 与 test_step2.csv 和下面所编写的.py 代码放置在同一文件夹下，若存在./dataset/文件夹则应与该文件夹所在目录为同一目录，若不存在./dataset/文件夹下面代码运行过程中也会自动生成该文件夹。

Step1:数据处理(清洗)

1.数据修补

观察数据集发现训练集中有很多测试集中没有出现的传感器数据，因此先将它们从训练集中剔除。

观察数据集发现训练集中有 22145 个 0，3 个大于 100 的 value，为了方便后续处理，将 0 设置为 10^{-8}，大于 100 的数设置为 nan 在下一步进行插补。去除为 0 的数值之外最小的 value 为 5×10-8。

阶段 2 相比于阶段 1 多了很多缺失值 nan，因此缺失值的补全是一项必要的工作，尝试过很多想法这里就不赘述了，**首先**，采用了赛题组老师点拨的用一周(7 天)之前的数据填补缺失值，但是有很多缺失值在一开始就存在，所以我又从后往前若一周之前没有数据就采用一周之后的数据补全，如此多次迭代即可填充大部分缺失数据，但是仍有部分数据缺失值过多，该方法依然无法补全缺失值，**其次**，我又采取**随机森林回归**方法填补剩余数据，由于题目所给日期和 ID 不构成有效训练集，所以又将每周的周几这个特征提取出来作为训练集一部分，没有在这个阶段提取更多特征是因为我认为其它的特征不是主要特征(其实还可以对时间这个特征有更深挖掘，如是否节假日、前一天是否节假日、后一天是否节假日，月、年、季等)，最终除去传感器训练集数据全为 nan 的情况外均补齐了所有缺失值，本人也考虑过是否根据 80%原则将 nan 占比过大的数据集保留不做缺失值填补，后来又想了想算了(尽管可以利用规则使用 nan 折中不好预测的数据，但是赛题组老师也说过实际中根本没有 nan 得 40 分这种自欺欺人的情况，因而能预测就都预测)。

以上步骤直接运行 **Clean_Data.py(在子文件 CNN 卷积中)**即可得到清洗且补充 nan 之后的数据 dataset/train_clean.csv。

2.提取特征、归一化、one-hot 编码

然后就是数据归一化，因为数据之间的标度差别过大，所以需要将所有 value 归一化再进行下一步处理，凑巧的是数据最小值为 10^{-8} 级别，最大值 100，取以 10 为底对数刚好数值相差 10，为此选用如下归一化公式:

$$\mathrm{norm}_y = 0.1log_{10}(y) + 0.8$$

刚好 10^{-8} 归一化到 0，100 归一化到 1。数据在同一量级，方便后续使用。

由于 7 天一周的周期特征比较明显，因此除去 value 本身外，同时又取滑动平均 2、3、4、5、6、7 步得到的数据作为额外特征，滑动平均会产生一定 nan，对这些 nan 用后面滑动平均得到的第一个数据进行填补，数据量小于 7 的数据则单独分析，滑动平均到数据量个数为止，更高次数的滑动平均直接等于数据个数的滑动平均。

开始考虑过不对数据量过少的样本处理，用移动平均法简单分析后发现若将数据量过少的直接取平均得分是比置为 nan 高的，滑动平均法的实现代码为 Average_Model.py，在数据修补这个过程处理得到 dataset/train_clean.csv 文件夹之后运行，数据量过少直接取平均值可以得到 62.4 的跑分，而置为 nan 的话得分则为 62.3，其它参数没有调整。因此数据

量虽然少，但是依然是有用信息。

再将星期数进行 one-hot 编码，保留原有星期数目同时保存。再将归一化的特征数据取 5 位小数保存,若按照原有精度保存的化占用内存量是现在两倍。

这个步骤直接运行 DataFrame.py(在子文件 CNN 卷积中)可以得到这个汇总了所有要提取并且归一化之后的特征数据 dataset/All_dataset.csv。

3.划分训练集、测试集

初步分析可以将数据中连续 1 年的数据作为训练输入未来 91 天 value 作为输出，以此为基础设计训练集。

读取 All_dataset.csv，遍历每个传感器将全为 nan，数据量小于 91 的数据 pass 不作为训练集,剩余的数据中数据量大于 365+91=456 的直接取最后 91 个数据的 value 作为预测标签 Train_Y,倒数 456-倒数 91 共 365 个数据作为训练输入 Train_X，最终得到大小为(23820, 365, 14) 和(23820, 91)的两个 numpy 数组形式数据，并保存为.npy 格式备用。

除了这个方式取训练集外还可以设置滑窗在数据量较大的传感器上多次取数据，但是这种方式可能会导致最终训练的模型会更倾向于提供数据量多的传感器,若后续实验发现所制备的训练集数据量不够用再统一向前滑窗取值添加到该训练集中。

该过程实现在 Make_DataSet.py(在子文件 CNN 卷积中)中实现。测试集不建议从以上方式得到的训练集中瓜分，可以向前滑窗用 Make_DataSet.py 微调的代码实现。

Step2:训练模型

这方面模型有 WaveNet 模型，多步预测的基本思路就是先预测一个值,再将该值返回模型预测下一个值,即自回归特点，由于时间仓促该方法目前暂时没有尝试，其中代码实现细节我也会在赛后学习实现。LSTM 等循环神经网络结构可能会出现误差累加，有一定趋势等现象，暂时不考虑，并不代表不合适，仅为主观认知。

结合第一阶段的思路可以大致了解到一定步长的滑动平均法可以得到较好的结果,说明序列的局部(或者某部分)特征对于预测作用比较大，而滑动平均又与卷积神经网络 CNN 中的卷积核有着相似之处，若单看一维序列，它是有可能学习出滑动平均的权重，而滑动平均法预测后 91 天数据其实就是已有数据基础上数据权重不同的线性组合，如以下例子：

序列[1,2,3]滑动平均窗口为 3 预测后 2 个数据，可以得到以下预测结果:

$$y1=\frac{(1+2+3)}{3}=\frac{1}{3}*1+\frac{1}{3}*2+\frac{1}{3}*3$$

$$y2=\frac{(2+3+y1)}{3}=\frac{1}{3}*1+\frac{4}{3}*2+\frac{4}{3}*3$$

可以看出预测的未来两步归根结底均为历史 3 个数据的不同权重线性组合，为此大胆构思网络结构为若干 CNN 卷积层+若干全连接层，输出层之前为提取出的少量重要信息，如果是 5 步移动平均法可以尝试把神经元个数设置为 5，输出为 91，每个输出对应权重可以尝试训练出来。对标 5 步移动平均法可以将网络设计为[conv,…,Dence(5),Dence(91)],当然我是假定神经网络比较聪明，倒数第二层可以学到类似序列后 5 个数据大小的数据特征，最后一层即为前面分析的不同权重组合预测未来 91 个值。

选定了大致网络结构之后，因为老师提示了可以提取一些特征再加以引导训练，因为 CNN 比较难发掘时间维度的特征，所以就需要提取出时间信息让网络学习，观察数据可以发现大致以 1 周为周期，因此将星期数进行 one-hot 编码后期待网络可以学习出时间层面的信息，此外还提取了滑动平均等特征作为辅助，结合前面分析就可以不断尝试调整网络参数期待得到较为满意的结果了。

该模型的训练可以直接运行 Train_CNN_Model.py(在子文件 CNN 卷积中)在/dataset 文件夹下得到 train_model.h5 格式的模型文件，对于测试集的预测则运行(在子文件 CNN 卷积中)Per_CNN_Model.py 可以直接在/dataset 文件夹下得到最终预测结果 result.txt，其中预测细节为判定全为 nan 的数据预测值也置为 nan，其它包含数值的数据数据量大于 365 个则直接取最后 365 个数据进行预测，小于 365 个数据的先用 0 补全为 365 时序大小数据再带入模型预测。

初步设置的网络模型有 206079 个参数，是相对比较多的，因为本人电脑只有 CPU，初步尝试预测 10 个 epoch 测试代码可行性，最终得到的结果是可以正常在打分系统中运行的，10 个 epoch 大概运行时间为 15-20 分钟左右，得分 7.252 分，距离初赛结束还有大概 20 小时，斟酌之后决定使用 300 个 epoch 训练模型，模型训练相当长时间(6 小时)之后效果相比于 10 个 epoch 有了很大提升，得分为 31.384，虽说相比于滑动平均来说效果依然不是很好，但是该模型也被证明是可行的，需要不断调整参数与模型架构以得到更好的结果。

分析原因粗略估计可优化的地方有，一个是模型参数两较大，训练集数目可能不足以支撑，还有就是观察模型中 conv 层展平到 fc 层时候参数量依然非常大，连接到下一层神经元之间的权重必然会非常小，所以需要对卷积展平层数据量作进一步优化是个方向。

尽管该模型已经搭建完成，但是综合考量可行性是不太高的，毕竟训练一次需要以小时为单位，所以在模型训练期间尝试学习其它模型。

Step3:其它模型

结合群中大佬的反馈大致可以知道，死磕平均也是可以得到较好的结果，不过这种方法对于目前的我来说意义不大，还有一种方法是提取多组特征使用 lgb 模型也可以得到比较好的结果，鉴于已经提取出一些特征，尝试搜集 lgb 有关资料学习该模型并试着实现。

参考这篇博文 https://blog.csdn.net/hellozhxy/article/details/82461757

大致读了下这篇文章讲的是饭店流量的预测，它与我们这次比赛最大的区别是它已知了很多特征，训练时候以特征为输入，value 为输出，输入输出数目相同，包括它的测试集也是已知特征来预测值，但是本题特征非常少，而特征挖掘也需要在原有 value 基础上挖掘，这就导致测试集缺少该特征，仅有时间特征的预测显然不行，所以需要想办法解决。

目前想到的办法是错位预测，训练集的输入 x 取时间序列前一段，输出取后一段的 value，预测时以已知数据中特征为输入预测未来 91 天数据，这个思路模仿博文实现起来并不困难，可能需要一点时间，现在我在思考的问题是数据量比较小的传感器怎么错位划分，还没有想明白，暂时思路是这些数据滑动平均，其它的用 Lgb。关于该模型的原理我了解并不多，仅仅是主观上泛泛的理解，可能需要一段时间系统学习。不过很遗憾没能在初赛实现该方法，虽然遗憾，但是并不后悔，因为在其它方面也学到了很多知识。

Step4:个人感受

参加中兴这次比赛让我学到了很多东西，这也是我第一次参加这样的比赛，像随机森林,lgb,TCN,Transformer 等自己之前还没了解过的模型在这次比赛中都有了深入的了解，以及之前仅简单学习过的 tensorflow 框架在这次比赛中得到了应用实现，更为珍贵的是在比赛中处理数据编写程序时遇到了非常多的 bug 一步一步解决之后发现自己对 python 编程更为得心应手了，对于我本人来说这次比赛过程中学习到的东西非常有价值，至少在我看来是高于比赛结果的。或许我可以为了得分而不断优化平均算法，不过对于已经研究生的我来说时间已经不允许我这么浪费，对于非科班出身的我来说要学习的东西还有很多，没有必要过于执着排名，更重要的是多学多动手，调参这种东西更注重经验，我相信有了这次开始，后面我也会不断参加类似比赛，不断积累经验，会摸索出一条适合自己的路。

总之，很感谢中兴给我带来的这场由浅入深的启蒙比赛，让之前一直在比赛门前因学艺不精望而却步的我开始尝试，并发现打比赛是一条更高效的学习路线，一场比赛下来我能从基本编程到机器学习算法再到深度学习中的大部分模型都有深入了解，更为可贵的是赛题组人员发出的论文让我可以开始面对读论文这个选项。

无论今年结果如何，明年中兴的算法比赛，我会以更加饱满的姿态迎接，不再像今年这般查漏补缺狼狈不堪。