樊赫铮实验报告2——TASK1.2 将股票数据带入ResNet

1.代码地址

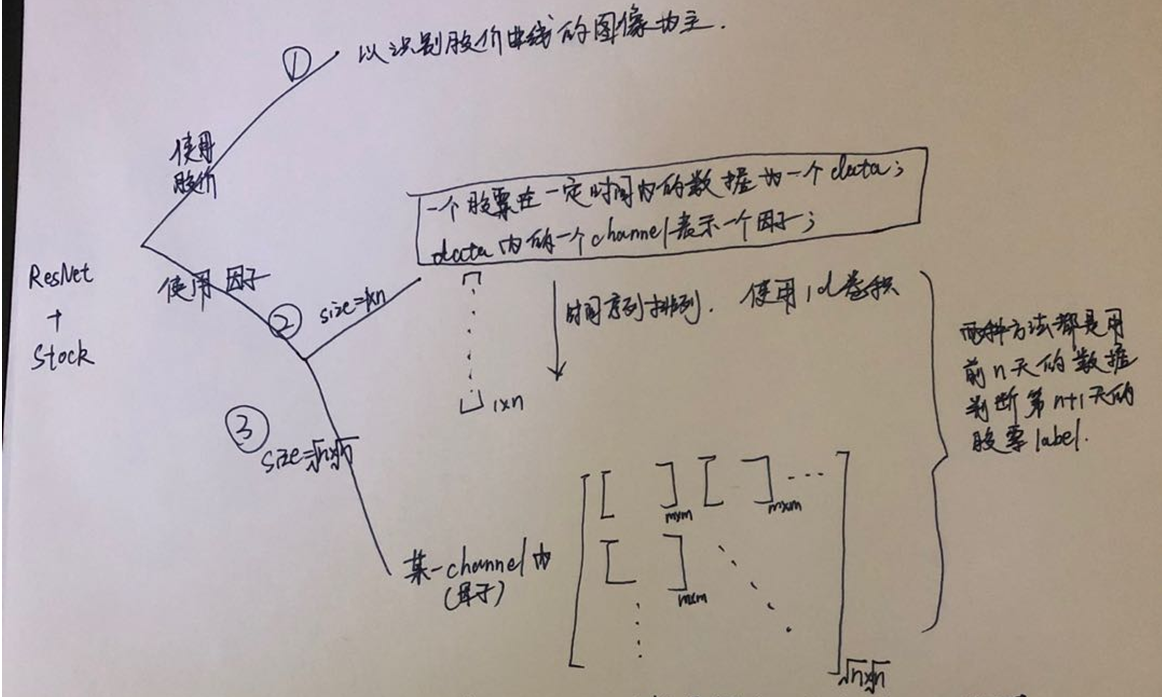
[http://124.251.74.84:42206/user/fanhezheng/notebooks/Task1.2\_Stock\_Prediction\_ResNet\_Multi-Factor\_Strategy.ipynb#](http://124.251.74.84:42206/user/fanhezheng/notebooks/Task1.2_Stock_Prediction_ResNet_Multi-Factor_Strategy.ipynb)

2.实验目标

将已有的股票数据处理成合适的数据结构带入ResNet，期望可以使网络学习到因子规律，最终进行预测得到相对于市场指标超额收益的买入卖出指令。

3.实验思路

（1）寻找合适的股票数据结构



经过讨论、思考、上网查阅，主要发现了以上三种带入方式。

不考虑方法1：我认为市场可以达到弱势有效。

方法2和方法3考虑先实现方法3尝试一下。但是预计数据处理的工作量很大。

Label: 将沪深三百指数看作基准。label选用相对于沪深三百指数该股票第n+1日有超额收益为正的买入，第n+1日超额收益为负的卖出。目前先不考虑如何设计买入卖出多少手，只判断要不要买入和卖出。

模型背后的逻辑还有待进一步思考和完善。

（2）模型采用resnet34

2.实验内容：

（1）导入数据

首先导入数据，选用‘factor\_data\_13\_19.csv’数据集作为feature数据集，‘return\_data\_13\_19.csv’为label数据集，其中column：car\_hs300\_b30\_d001为label，大于0的表明有正超额收益，赋值1；小于0的表明超额收益为负，赋值0。

（2）数据处理

利用feature集和label集中"stock\_code","trade\_date"作为key进行内连接（inner），这种处理方式很快。

Feature集中共有42个factor。训练数据的排布方式选择为[n, num\_factor, 8, 8], 对应[数据数，channel数，height，width]。具体如何处理代码中有详细实例。本次实验中数据处理采用的是循环->变成矩阵->reshape。循环时速度较慢。提高速度的方法：先完全变成array再循环处理；变成array用np.take。但这两种方法对于处理能力要求比较高且由于原表中含有较多有用但不能添加到training矩阵中的index信息，需要不断地drop，因此用DataFrame较为清晰容易debug。

8\*8是某一股票连续64天的相关factor信息。

Training\_set、validation\_set、test\_set都必须都要取连续时间的数据。（实验中先不取test\_set, 用validation找到合适的参数后再加入test\_set）。

需要注意：每个股票的数据时间长度是参差不齐的，同时存在一定的NaN，需要有适当的数据处理。

（3）带入模型训练

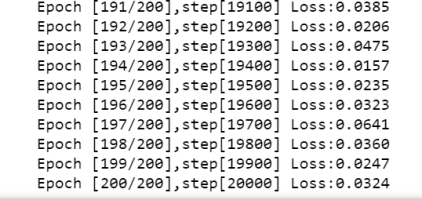
注：第一遍做的时候服务器出bug了必须要把模型代码完全复制才可以跑通。之后再测试时可以用py文件导入。

（4）模型训练好后进行测验

3.实验表现

（1）NaN用均值替代

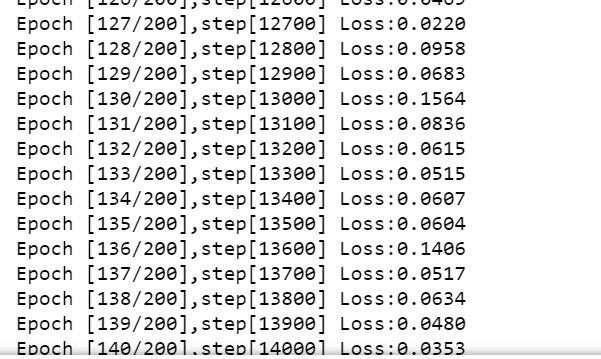
训练效果：



测试效果：

51.51740830385403%

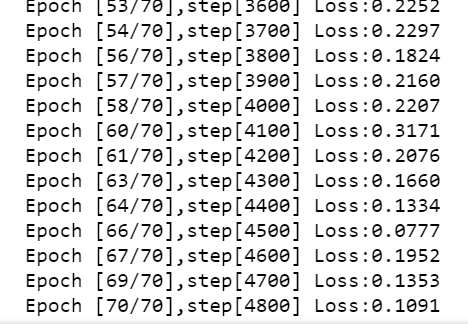
调整至20个factor



测试结果：53.025507354310186%

（2）彻底删除NaN：

结果依然类似上方

测试集：51.9117325759231%

4.实验结论

当前的数据结构和网络不能有效的进行预测。

代码经过检查应该没有问题。

网络在进行训练前对测试集进行预测和网络在训练后进行预测准确率都在50左右，同时利用不同的epoch进行训练，结果也大致相同，说明问题出在数据结构和网络结构的选择上，而不是过拟合的问题。

后续再进行研究应该尝试其他数据排列的方法，或者调整网络结构。或在有具体因子信息及其含义的情况下根据市场意义选择feature组合。