樊赫铮实验报告3——TASK2 CNN/RNN分类模型在多种股票数据组合方式上的测试

实验时间

7.1–7.24

1. 代码地址

本段小实验的参考代码地址见每段开头处。

2. task1问题总结及task2中需要注意的问题

经过讨论总结，task1主要的测试效果差的问题可能在于：

1.卷积方法的使用。尽管每一个卷积channel内都符合是一定的时间序列，但是构建resnet直接套用的图像处理的conv2d，并没有体现出时间序列自身的意义。故task中所有涉及到卷积的部分均使用conv1d。

2.处理数据的方法。task1中使用了13-19的全部数据，但可能因为因子有效性的问题而数据选取时间过长导致股票数据的分布有变化故模型效果不佳。task2中要注意这个问题。

3.模型评价指标。task1中模型的评价指标用的accuracy。其问题在于有可能在validation set中模型大多数超额收益预测均为正（做多）或负（做空），依然可以保证一定的准确率。为了更好的评价模型，task2中选取预测为正（做多）的precision为主要评价指标。尽管前述问题依然存在，但precision可以更具体的评价。若有时间则尝试引入confusion matrix做辅助评价指标。

3. 主要使用的模型：

1.LSTM
2.Conv1D + LSTM
3.ResNet（conv1d）+ LSTM
模型参考代码：http://124.251.74.84:42206/user/fanhezheng/edit/classification_models.py
为了方便修改网络结构做测试，task2中不使用import方法导入模型而是在测试中直接定义，但import方法是可行的。

4. 模型应用于13-19年全部数据

此类数据处理方法测试模型效果不好，所以简略总结。

1. 数据清洗方法

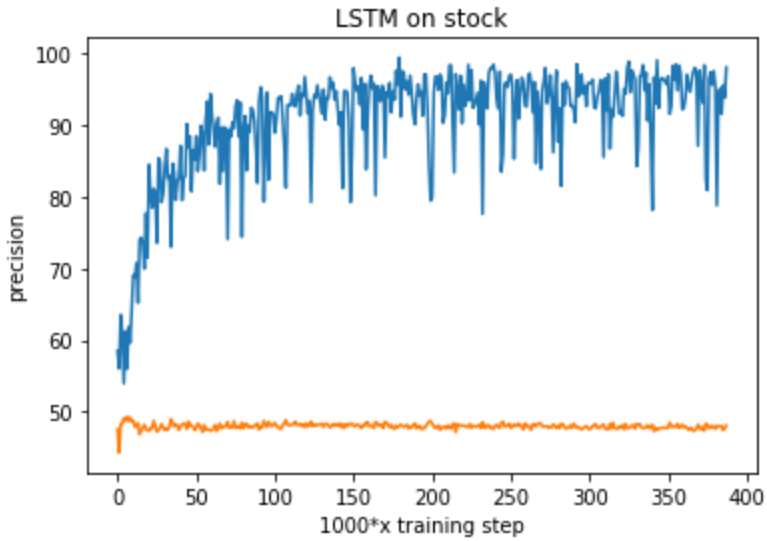
代码参考：
http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.1_cleaning_classification_data.ipynb

将编号‘000820.SZ’（受限于时间只能清洗出这么多数据）前的数据留连续150个交易日数据作为validation set，其余全部清洗为训练集。
使用的结构为：每个样本包括全部42个factor数据作为features，连续十个交易日为sequence。
数据量大小为（清理nan后）：
training：397353
validation：48055

2. LSTM测试结果

代码参考：
http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.2_Classification_Models.ipynb

先使用两层hidden layer的LSTM



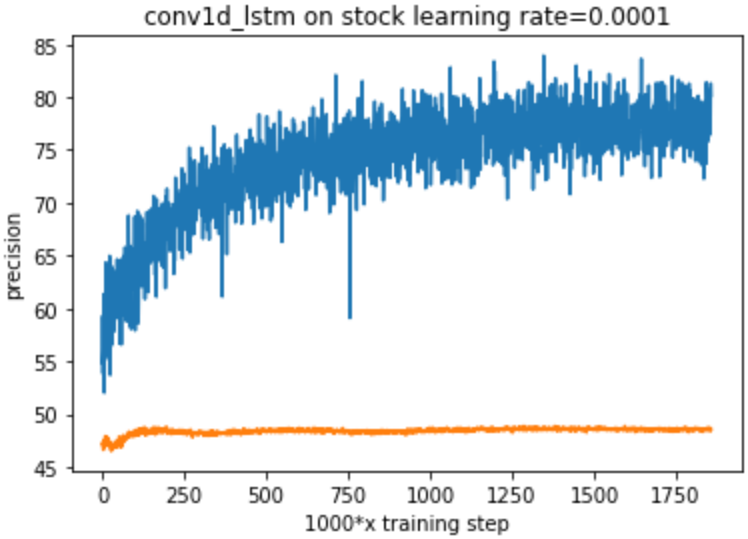
precision基本稳定保持在0.5以下，换大一些的batch_size:

依旧效果不佳。

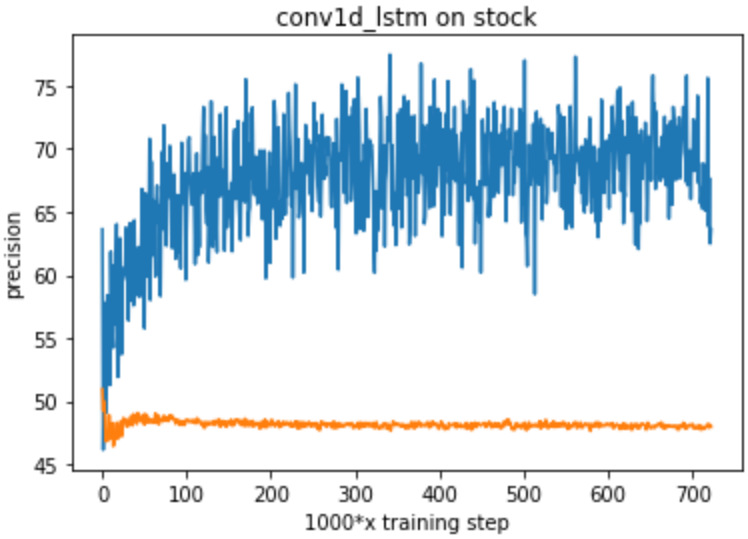
3. Conv1D+LSTM测试结果

代码参考：
http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.2_Classification_Models.ipynb

出现的问题：
刚开始时尝试将learning_rate调为0.1和0.01，皆出现training loss diverged问题，将learning_rate调到极小解决了diverge问题但又出现了training loss无法缩小的问题：



在做实验之前就已考虑过此问题，故所有卷积激活函数都用的leakyrelu。此时换回relu尝试：

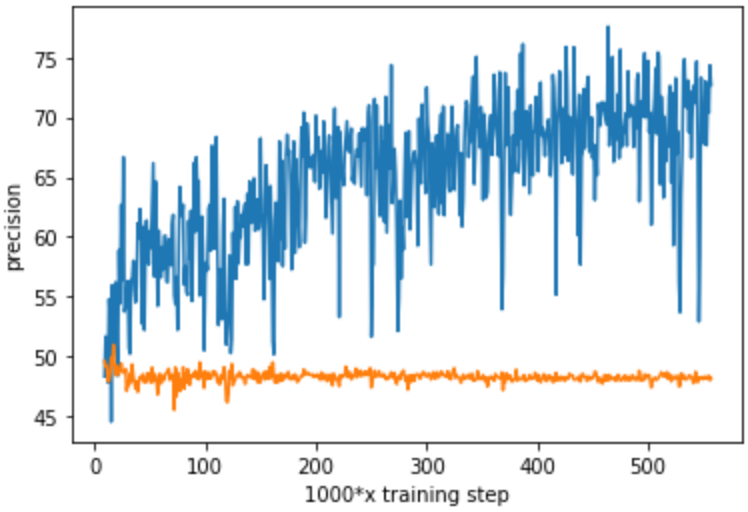


不出意外地training更加糟糕，故leakyrelu是正确的选择。之后又尝试了更换优化算法等方式，皆无法解决此问题。（注：之后大模块5（三级标题）LSTM中可以看到解决此问题的方法）

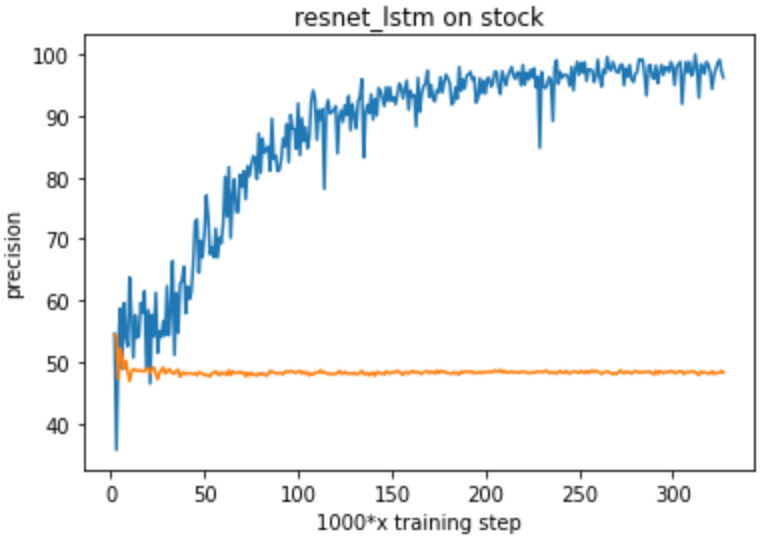
4. ResNet+LSTM测试结果

代码参考：http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.2_resnet_lstm.ipynb

首先采用ResNet34+LSTM（2层hidden layer）的结构，出现了梯度消失问题：



认为是由于模型过于复杂导致的，经过多次尝试简化模型和调整超参数解决了gradient vanishing问题：



但是validation precision依旧很差。还进行了多种其他尝试，validation precision都比较相似在此不赘述。

5. 总结

13–19年的数据长度上测试出的模型validation set precision都在0.48左右，难以令人满意。经过交流我认为是时间选取过长导致出现了因子失效或者数据分布发生变化的问题，下一大节使用新的数据处理方式进行模型训练和评价。

5.模型应用于19年11月和19年12月的数据

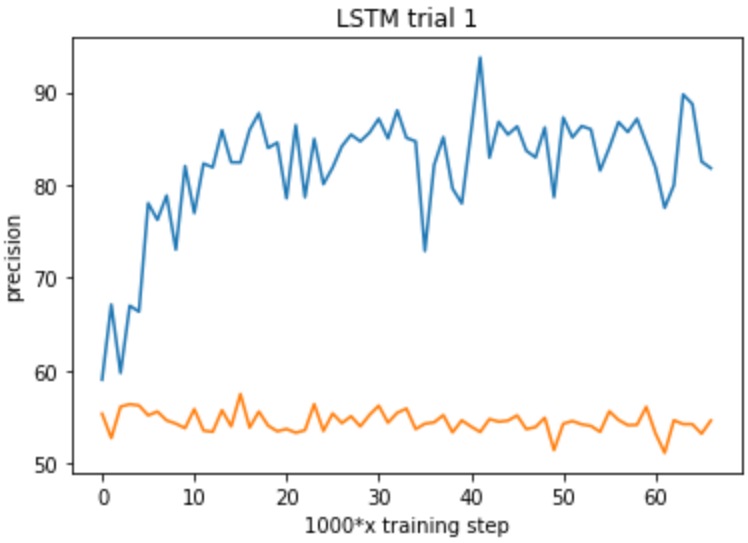
1. 数据清洗方法

代码参考：
http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.3_data_cleaning.ipynb

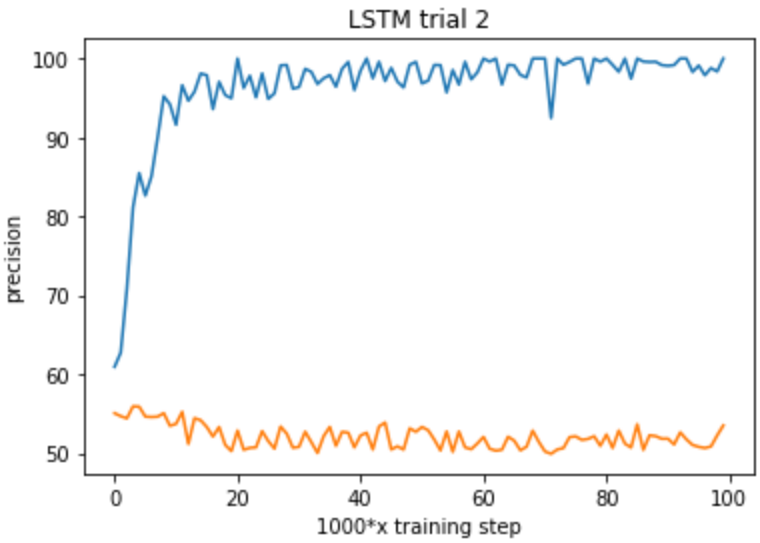
吸取上一节的经验本次数据采用时间轴上较为相近的数据作为training set和validation set。以11月全月的数据作为training（约4w数据点），12月的数据（12.2–12.19）作为validation set（约1.5w数据点）。

2. LSTM测试结果

代码参考：http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.3_LSTM.ipynb



#trial 1 结果分析：
#随着训练步数的增加模型loss无法继续收敛，可能是由于梯度消失。
#下一步，增加num_epochs并缩小learning_rate和batch_size



#trial 2 结果分析：

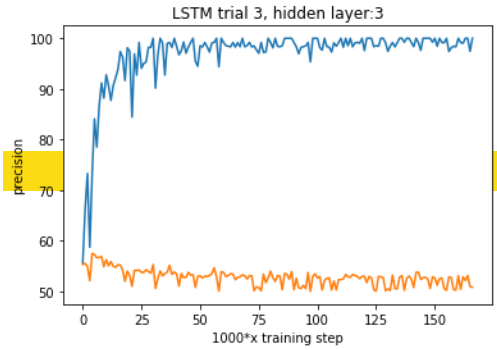
#上述提出的解决训练过程中梯度爆炸问题的方法是有效的，训练集已经可以达到接近100%的预测准确率

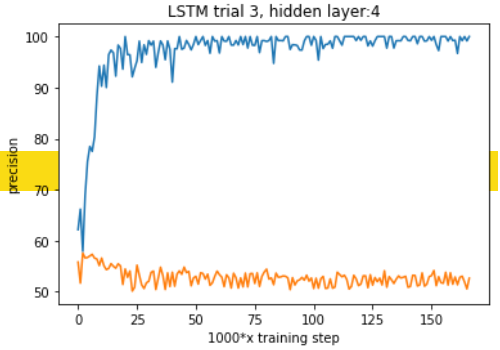
#但是在valid_data中很早就到达了过拟合点（12000 steps 左右），valid precision在0.54-0.56左右

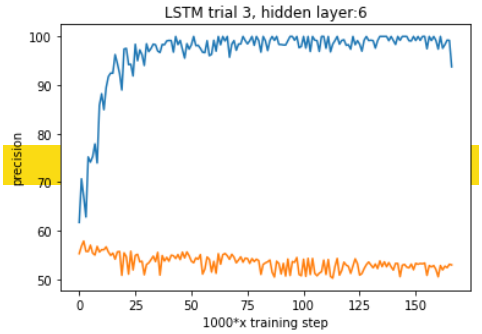
#故认为在2层hidden layer LSTM下，此precision范围已经是极限，下面需要尝试其他hidden layer层数的LSTM

#batch_size 和 learning_rate 在接下来的实验中不变

至此寻找到了防止梯度消失的参数组合：learning_rate=0.001,batch_size=256(后续实验证明延续使用这个参数组合也没有再出现过loss无法下降的问题）







#trial 3 结果分析:

#增加layer层数最终的valid precision并没有太大变化，过拟合前集中在0.555-0.575（略好于2layer结构）

但是过拟合依然十分严重，之后的LSTM实验围绕着防止过拟合进行。

解决过拟合的尝试：

（1）L2 penalty ——无效

（2）增加dropout——无效

（3）换优化算法——无效

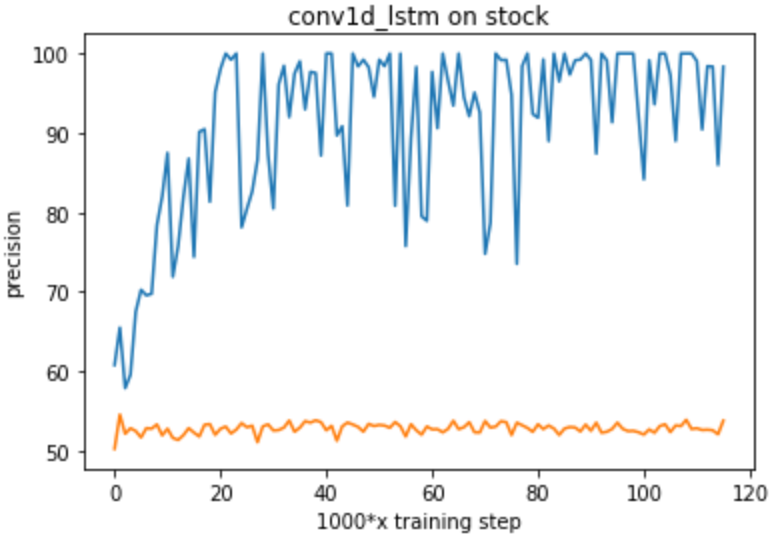
（4）后来想到因为整个数据集里有42个factor，可能是由于factor过多太复杂导致的过拟合，于是尝试从中多次任取四个factor和多次任取十个factor进行训练尝试，无一例外均比采用完整42个factor时的LSTM模型表现差。

至此尝试了所有能想到的延迟过拟合、提升模型表现的方法，均不能成功。

3. Conv1D+LSTM模型测试结果

代码参考：

http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.3_Conv1D_LSTM.ipynb



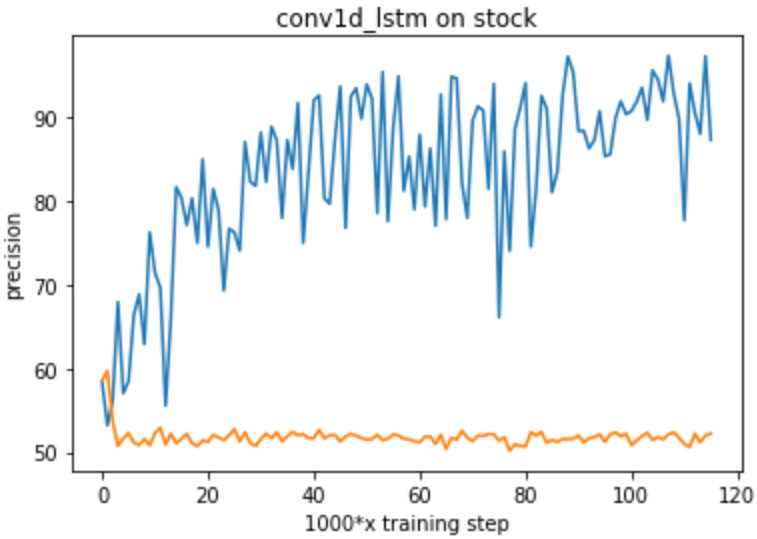
#Conv1D_LSTM trial 1 总结:

#batch_size，learning_rate等参数全部继承LSTM实验中筛选出的最优参数

#整体效果并不好，在44000-55000step中可以稳定保证valid precision在0.53-0.54间，之后由于过拟合 valid precision整体下降

#吸取LSTM实验中的教训，目前没有找到特别好的方法减小过拟合

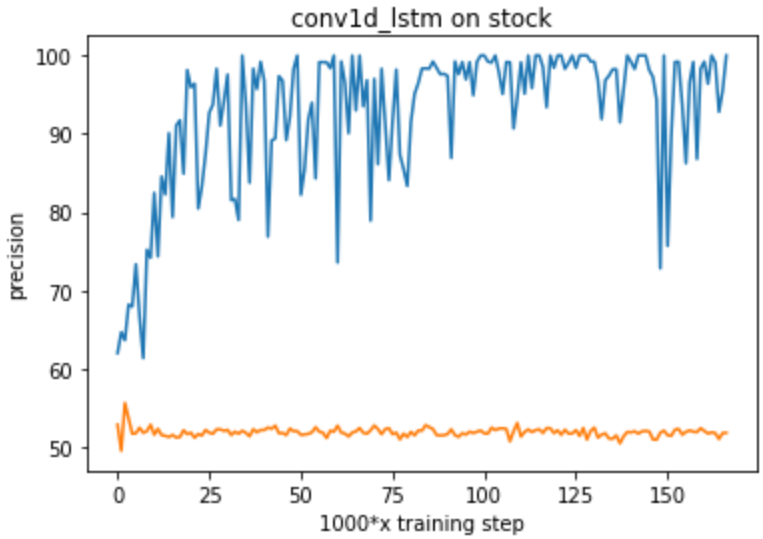
#故下一步试验改变网络结构，将LSTM的hidden_layer缩小到一层进一步尝试



#Conv1D_LSTM trial 2 总结:在将LSTM缩减为一层后模型在valid数据集的表现下降了一个档次，没有任何一段可以稳定超过0.53

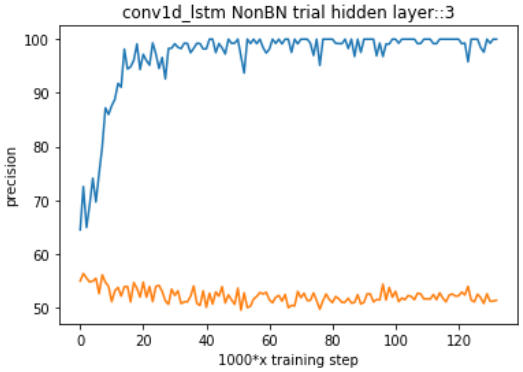
#由于LSTM hidden layer是唯一变量，而再将其减小后模型表现变差了，故怀疑是LSTM的比重减小是原因

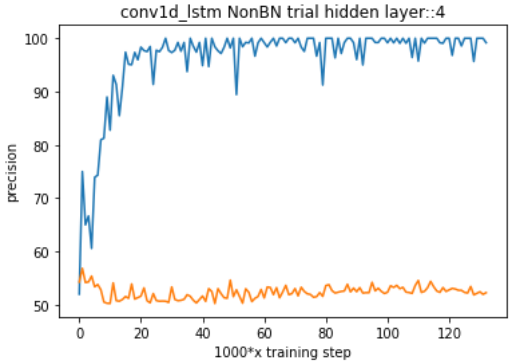
#下一步试验将LSTM hidden layer增加为2再尝试

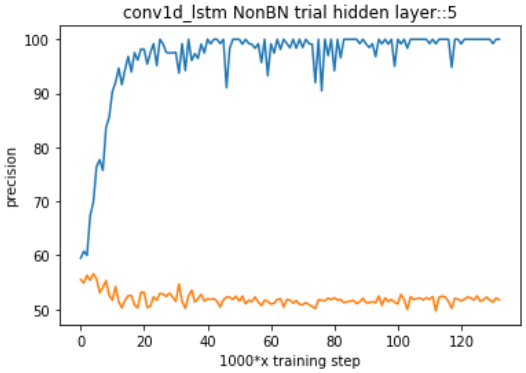


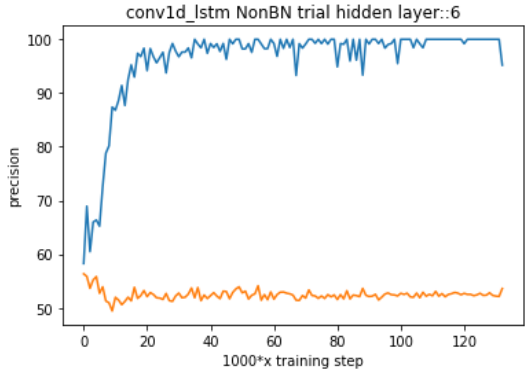
#Conv1D_LSTM trial 3 总结：没有太大改观

#经过思考，决定下一步尝试取消batch_norm进行尝试（normalization消除了conv产生的数据间距）







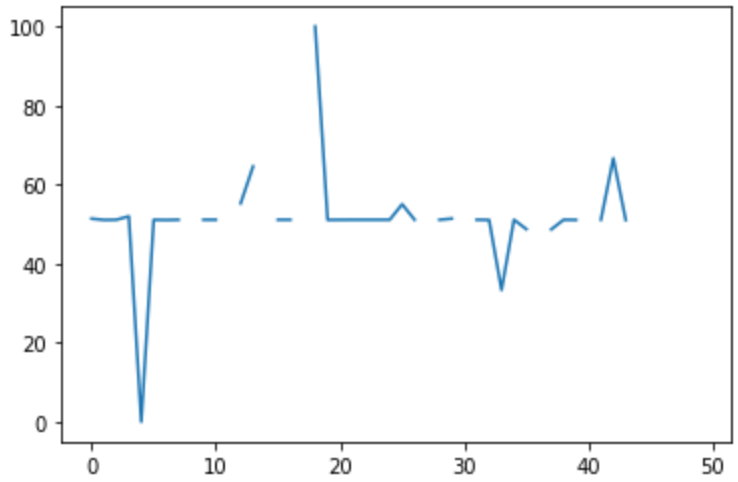


#Conv1D_LSTM trial 4 总结：

有意思的发现：conv1d+lstm一开始确实显得conv1d很拖后腿，稳定低至少4个百分点。后来琢磨为啥差这么多，毕竟都有lstm，感觉是增加了batch_norm的原因（因为之前看到的所有做图像识别的都加batch_norm了所以在我心里就变成了default choice），因为卷积基本上就是在增大数据（类比于图像的每个pixel）间的间距让不同的data point有差距才能有效果的预测。但normalization一定程度上就把卷积创造的差距打回原形了。确实norm可以加快gradient descent，但是对于股票这种复杂的数据来说（相对于简单的猫狗识别）目前连效果都没有就也谈不上加速了。所以后来把norm去了，然后真的有效果，接近了lstm之前的预测效果且非常明显的比加上norm的最好的precision还要好。

#最高的validation precision明显有所提高，但是往往都在训练epoch较小时，所以对模型的真实性有所怀疑

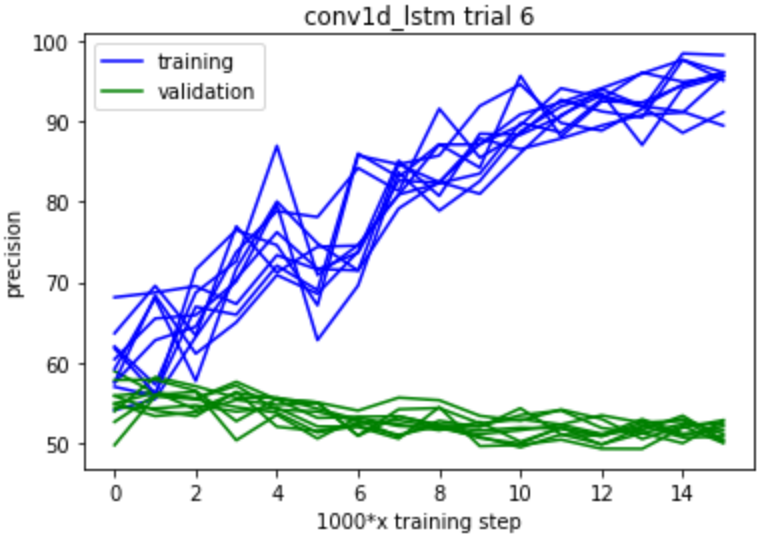
#下一步试验，不训练模型直接测试和用极少epoch测试效果



#Conv1D_LSTM trial 5 总结: 不训练就测试效果十分随机。

#接下来利用极少的训练次数进行训练。

#Conv1D_LSTM trial 6



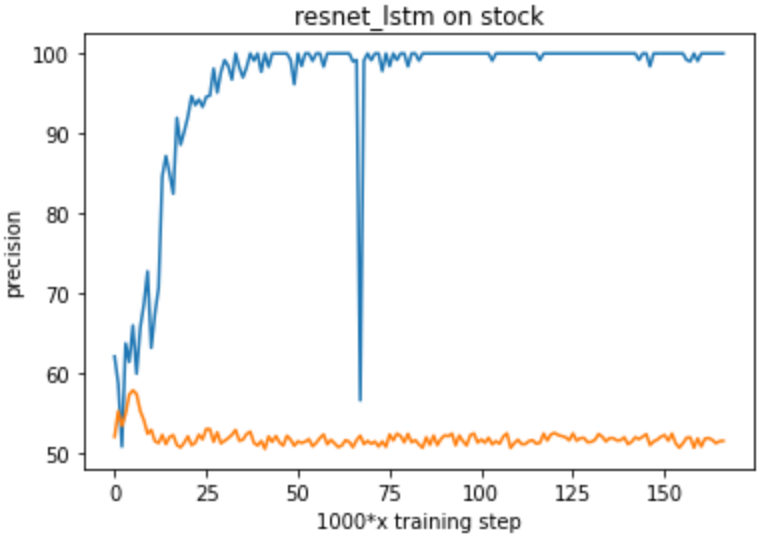
总结: 在使用Conv1D_LSTM时用较小的epoch可以达到比较好的效果，并且的确是由训练得出的

conv1d出现的值得关注的现象总结：不训练模型直接预测时出来的结果其实是随机的（循环做了很多次），因为在卷积层的初始化参数用的he initialization，所以其实每次都不一样不训练就预测只是运气问题（不太清楚pytorch lstm的参数初始化是不是也是随机正态）。然后又用较小的epoch训练了一下求了求precision。之前出现在刚开始训练时train precision<valid precision的原因想了一下，应该是因为我在算precision时，train set只是用的当前step的那个batch，而又因为为了解决梯度消失问题我把batch size调的比较小（256），而validset算precision时我用的整个validation set（1w5），所以在刚开始阶段某个step后train precision<valid precision其实是非常正常的。对此又做了多个循环，基本就是不训练太多时效果最好，而并非是随机的。所以综上觉得虽然在训练较少时就达到了valid precision的最高点然后过拟合了train precision平均下来也没有比valid高多少，但这确实是有意义的，能够反应模型的效果，过拟合的问题是数据本身的分布和模型造成的。

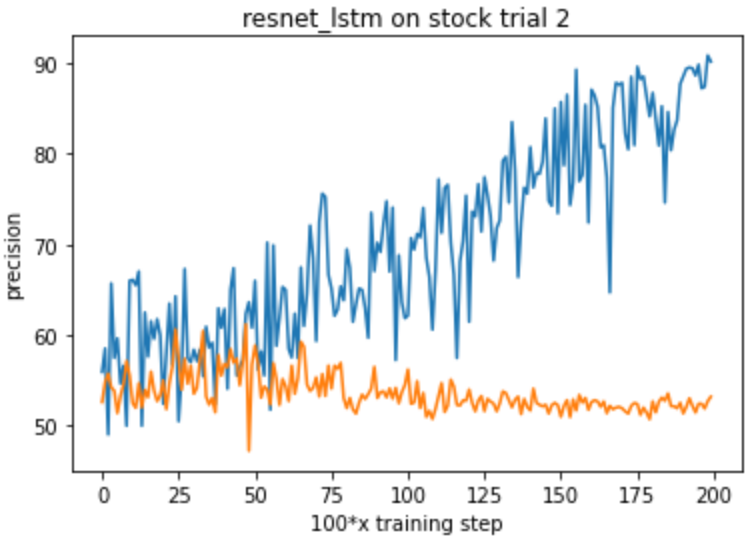
#下一步试验：验证LSTM和Conv1D_LSTM之间的模型的信号相关性
train集precision相关性系数为0.7814648697721939
valid集precision相关性系数为-0.23722368976954009
#目前不太清楚原因和背后含义（train的相关性系数可以理解，因为同有lstm）

4. ResNet+LSTM测试结果

代码参考：
http://124.251.74.84:42206/user/fanhezheng/notebooks/Task2.3_ResNet_LSTM.ipynb

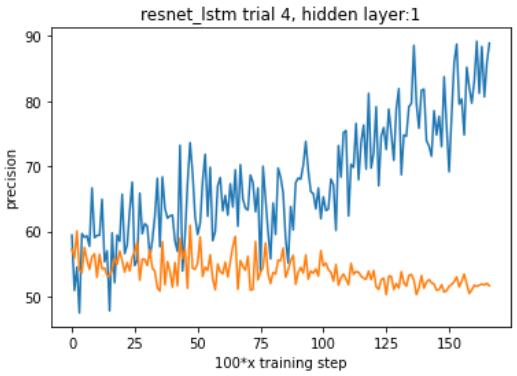


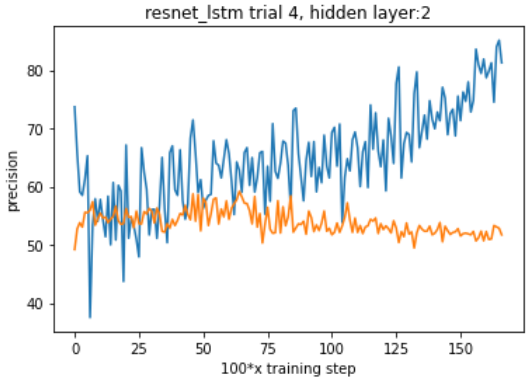
#resnet_lstm trial 1 总结:
#比较快就达到了过拟合点，下一步每100 training step就绘一次图，并减小epoch数

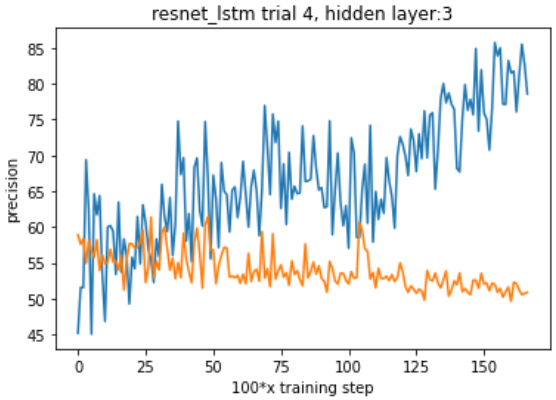


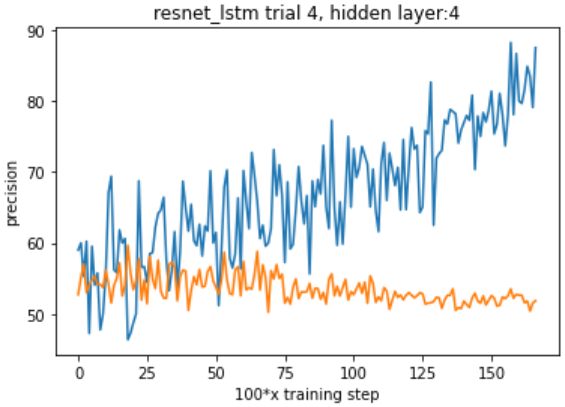
#resnet_lstm trial 2 总结:
#效果在一定区间内达到甚至超过了简单LSTM，但过拟合也很快，下一步调整参数继续试验，尝试不同取消maxpooling的resnet-lstm。

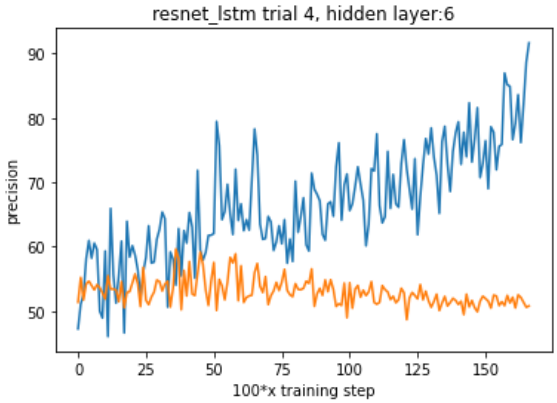
#resnet_lstm trial 3（no maxpooling）总结：效果并不如原版resnet-lstm
#故放弃这个想法以后的实验还是基于原版resnet-lstm
#下一步试验：用原版resnet-lstm测试不同hidden layer层数的效果

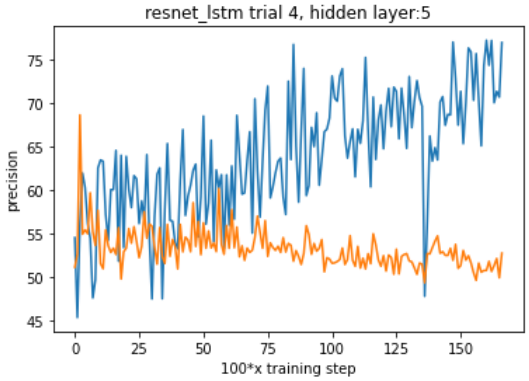












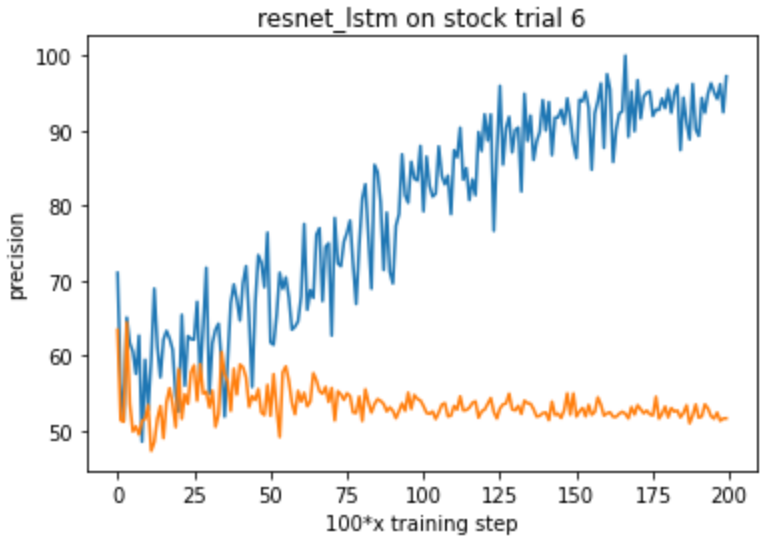
#resnet_lstm trial 4 总结：效果和之前没有太大差别

#故放弃这个想法以后的实验还是基于原版resnet-lstm

#下一步试验：减小resnet部分的深度，减小模型复杂度

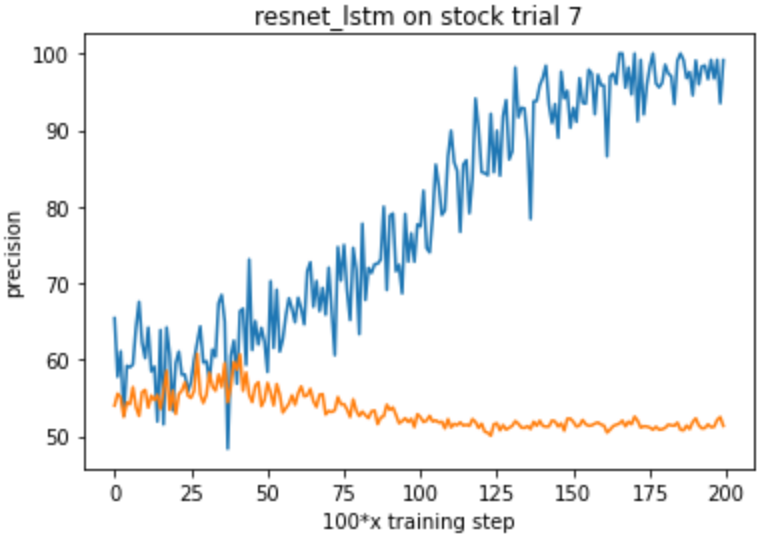
#resnet_lstm trial 5 总结：效果和之前没有太大差别

#下一步试验：将batch_norm删除看效果，其余参数与trial5保持一致

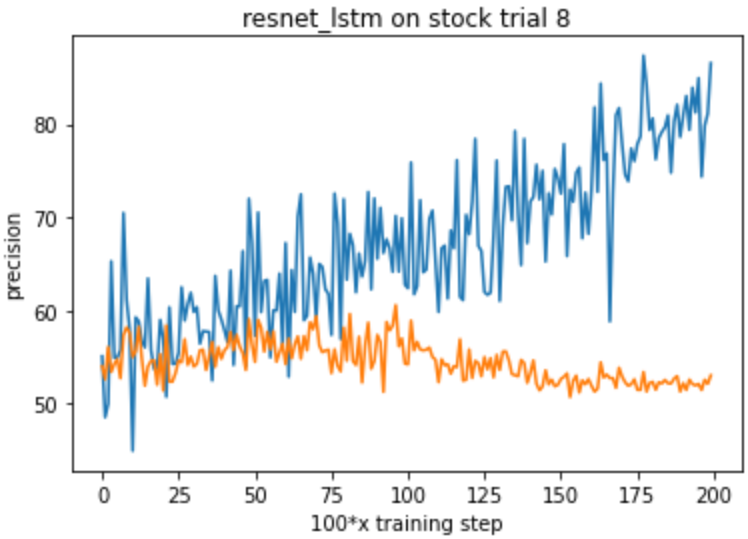


#resnet_lstm trial 6 总结：batch_norm去掉后效果和之前没有太大差别（与conv1d+lstm不同）

#下一步试验：保留原来的batch_norm，改为将第三类resnet block删除



#resnet_lstm trial 6 总结：效果和之前没有太大差别
#下一步试验：增加dropout尝试解决过拟合



#resnet_lstm trial 8 总结：稳定性和参数模型完全相同的trial5比要好一些
#下一步试验：在dropout基础上增加l2正则化
#结果：无效。
整体而言，ResNet+LSTM效果比简单lLSTM要好，可以在一定训练epoch区间内找到valid precision在0.56-0.6的结果，但是相比于之前的LSTM，模型稳定性不好（validation precision波动较大），产生的原因可能在于网络层数比较深参数多，难以训练。dropout可以一定程度上解决这个问题但无法进一步提升validation precision。

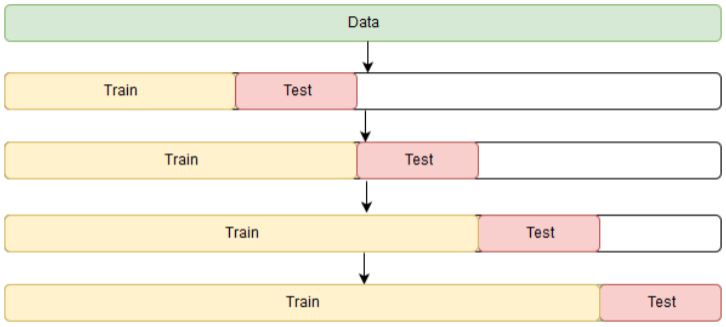
5. 总结

数据选取的时间对于模型的训练十分重要，目前看来可以呈现出一定效果的至少是本节所选用的用上一月的数据作为训练然后用本月数据作为validation可以呈现出效果。LSTM和ResNet（[1,2,1]，dropout）

+LSTM(3-hidden layer)，两个模型效果最好。

6. 下一步研究方向

本次实验中已经筛选出了两种有一定效果的模型，下一步采用rolling window训练的方式测试LSTM和ResNet+LSTM两个模型在13-19年数据中的可靠性。



争取下一段代码有比较好的复用性。