**Grid2dLSTM 构建高抽象的多因子股票时间序列预测模型**

[猫狗大战](https://www.zhihu.com/people/Uekyoung)

[**猫狗大战**](https://www.zhihu.com/people/Uekyoung)

退乎保平安

37 人赞了该文章

**1前置假设**

这里假设股票市场的对数收益率在泛泛的情况下假想为一个弱平稳时间序列。

* Stacked LSTM

Stacked LSTM通过堆砌多层LSTM的抽取下层LSTM的hidden\_state的信息获得实例训练算法，这个取得了很好的效果提升，但是受限于tanh，这里使用的LSTM模型在超过5层之后的需要消耗的计算资源过大。我这里使用一个3层的Stacked LSTM模型进行。

* Grid2dLSTM

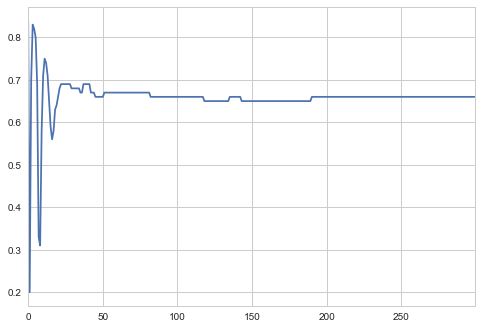
Grid LSTM 是吸收Stacked LSTM和Multidimensional LSTM两种LSTM单元间网络结构形成的新算法设计。相对于Stacked结构，Grid2d使用维度概念代替Stacked里面的深度概念(层数)，将深度构建为另一个维度的序列，即转换多层LSTM为x，y维度的多维度LSTM网格结构。实验显示，Grid相对于Stacked效果有明显提升。由于计算资源有限，我只测试了3层Tied 和 3层No Tied的网络结构。

**2 模型数据**

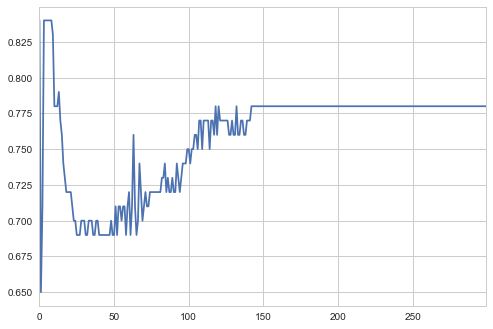
对HS300成份股使用连续59交易日多因子数据进行many to one 训练。

**3 数据对比**

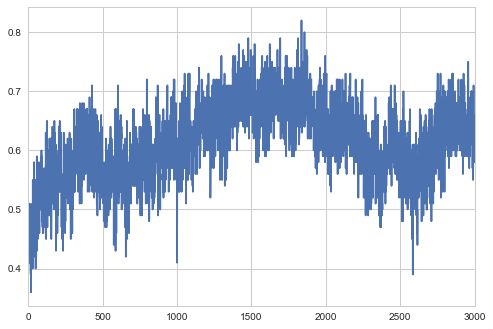
1) 单层LSTM的样本外准确率为0.66



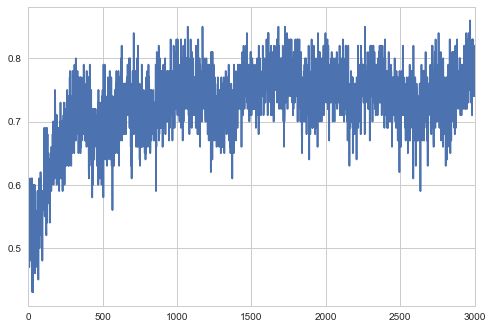
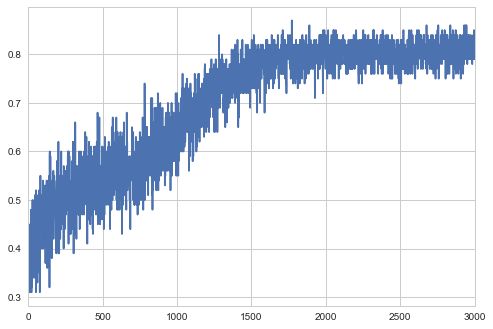
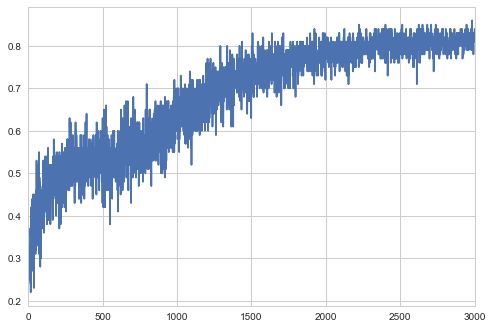
2) 3层Stacked LSTM的样本外准确率为0.78



3) 3 层Grid2dLSTM样本外准确率约为为 0.7 (由于使用CPU这里模型迭代只进行了3000次，未达到最终收敛状态)



4) 3层Tied Grid2dLSTM模型的样本外准确率为 0.86 (由于使用CPU这里模型迭代只进行了3000次，未达到最终收敛状态)

5) 6层  
6）9层

[运行示例](https://link.zhihu.com/?target=https%3A//www.ricequant.com/community/topic/2753/)

**def** **LSTM**(H\_upper, H\_lower, m, omega):

H **=** tf**.**concat([H\_upper, H\_lower], axis**=**1)

g\_u **=** tf**.**sigmoid(tf**.**add(tf**.**matmul(a**=**H, b**=**omega['W\_u']), omega['b\_u']))

g\_f **=** tf**.**sigmoid(tf**.**add(tf**.**matmul(a**=**H, b**=**omega['W\_f']), omega['b\_f']))

g\_o **=** tf**.**sigmoid(tf**.**add(tf**.**matmul(a**=**H, b**=**omega['W\_o']), omega['b\_o']))

g\_c **=** tf**.**tanh(tf**.**add(tf**.**matmul(a**=**H, b**=**omega['W\_c']), omega['b\_c']))

m\_new **=** tf**.**add(tf**.**multiply(m,g\_f), tf**.**multiply(g\_c,g\_u))

h\_new **=** tf**.**tanh(tf**.**multiply(m\_new, g\_o))

**return** h\_new, m\_new

**def** **scan\_Grid2dLSTM**(initializer, elems):

x\_i **=** elems

*# layer 1*

I **=** initializer[0]

h\_1\_A\_prev **=** initializer[1]

m\_1\_A\_prev **=** initializer[2]

m\_1\_B\_prev **=** initializer[3]

I\_x\_i **=** tf**.**matmul(x\_i, I)

h\_1\_A, m\_1\_A **=** LSTM(I\_x\_i, h\_1\_A\_prev, m\_1\_A\_prev, W\_1\_A)

h\_1\_B, m\_1\_B **=** LSTM(I\_x\_i, h\_1\_A, m\_1\_B\_prev, W\_1\_B)

*# layer 2*

h\_2\_A\_prev **=** initializer[4]

m\_2\_A\_prev **=** initializer[5]

h\_2\_A, m\_2\_A **=** LSTM(h\_1\_B, h\_2\_A\_prev, m\_2\_A\_prev, W\_2\_A)

h\_2\_B, m\_2\_B **=** LSTM(h\_1\_B, h\_2\_A, m\_1\_B, W\_2\_B)

*# layer 3*

h\_3\_A\_prev **=** initializer[6]

m\_3\_A\_prev **=** initializer[7]

h\_3\_A, m\_3\_A **=** LSTM(h\_2\_B, h\_3\_A\_prev, m\_3\_A\_prev, W\_3\_A)

h\_3\_B, m\_3\_B **=** LSTM(h\_2\_B, h\_3\_A, m\_2\_B, W\_3\_B)

**return** (I, h\_1\_A, m\_1\_A, m\_1\_B, h\_2\_A, m\_2\_A, h\_3\_A, m\_3\_A, h\_3\_B)