

1. 前言: 使用LSTM, 或同作業二採技術指標即可?

答案: 以CP值而論

很明顯本期末專題是時序性資料預測, 一定想到用回饋式神經網路RNN或長短期記憶網路LSTM訓練。

神經網路固然強大, 卻可能存在一個致命缺陷: **過擬合**

(Overfitting): 在initail submission很好的分數, 有可能在最後14天評測中, 大大落敗 (而14天也較短, 無法看出以多年資料作訓練集從長計議的優勢, 例如模型以半年至一年的長線scale有賺頭, 但在未來14天評測, 因時間長度不夠, 短線而言可能無法給出好結果)。

或許只要layer by layer fine-tuning, 搭配Drop, pruning即可避免。但會耗不少時間, 效果亦未知。

期末繁忙下, 以類似作業二方式實作, 會是較高投資報酬率 (指所付出的時間心力和所獲得的成果) 的方法。

2. 何種技術指標? 如何優化?



由上台指期走勢圖得知，指數雖沒有如亞馬遜公司在近幾年大幅竄升，卻是穩穩成長(納入股市型態學考量)。選擇最基本技術指標MA、RSI即可。

第二次作業，我用RSI作指標，搭配短期RSU數值限制在某範圍配合各支股票fine tuning，而獲得了高達5.5萬分的高分成績位居全班第八。

```
rsi_s = float((up + 1) / (up + down + 1))
if rsi_s > rsi_l or (rsi_s > a and rsi_s < au):
    action = 1
elif rsi_s < rsi_l or (rsi_s < b and rsi_s > bl):
    action = -1
else:
    action = 0
```

MA不採用乃因測試數次後效果皆不如RSI (MA單純採用**移動平均**，易受到**極端值**影響；然RSI是**量化的結果**，可清楚看出股票**"潛在勢能"**)。

本期末專題捨棄短期RSI額外設上下界(上圖的 $rsi_s > a$ and $rsi_s < au$)，以免過擬合，純粹採用 $rsi_s > rsi_l$ 作為買的指標，再暴力搜尋s、l的參數(天數)即可。

又在HackMD作搜尋結果log，分別記錄以分鐘和以天為單位兩個csv檔參數搜尋結果，避免重複找參數區間而顯冗贅。

 CHANGED 7 DAYS AGO

 Watch

NTU Intro. to Fintech Fall 2019 Final Project report

Optimization log - Initial submission

(The last 3 column represents the score, x means untuned or unevaluated)

date	l	s	a	b	local	init	final
1221 daily [0, 100]	49	47	x	x	0.040	508	399
1222 daily [100, 200]	126	124	x	x	0.042	535	426
1222 daily [200, 300]	253	240			0.038		
1226 daily [300, 400]	324	305			0.038	515	406
1222 daily [1000, 1200]							
1222 minly [0, 100]	68	63	x	x	0.036	473	365
1222 minly [200, 300]	234	225	x	x	0.040	505	396
1226 minly [300, 800]	441	431	x	x	0.045	557	448
1228 minly [800, 1000]	916	903	x	x	0.038		

最後，不論initial 或 final submission，我都獲得了第一名，可見該參數模型具有一定的股價評判能力 (final submission 到正式開始評判前，已有超過 2019/12/16的資料，完全不存在助教給的csv中，可視為testing set概念，而得以客觀評價) 程式碼既短、快速又高分。

405

94

91

Leader Board

	Account	Verdict	Score	Len		Date
1	r08922024	16.6 s, 343 MB	448	866 B	Python 3.5.2	2019/12/27 10:18:32
2	b06902084	17.0 s, 338 MB	448	1 KB	Python 3.5.2	2019/12/23 22:29:42
3	r07221012	16.8 s, 343 MB	427	703 B	Python 3.5.2	2019/12/25 11:31:13
4	r08922051	17.7 s, 363 MB	427	11 KB	Python 3.5.2	2019/12/26 19:47:43
5	r08944029	16.9 s, 338 MB	412	3 KB	Python 3.5.2	2019/12/24 12:20:52

3. 如何加速參數搜尋？以達最少時間最大效果

暴搜參數相當曠日廢時，用python multithread，各自搭配 `os.system('python main.py a b')`，搜尋 [a, b)區間傳入 argv，既不用煩惱critical section問題，又能自動化，一舉數得。(詳細程式碼請見 main.py, main_multithread.py)

```
search between [880, 980] type_eval 1
search between [840, 860] type_eval 1

1 main.py
2 import numpy as np
3 import pandas as pd
4
5 '''
6 Progress log note:
7
8 12/20: Try using the brute force search like HW2, but trying with daily scale approach
9
10 '''
11 min_l = int(sys.argv[1])
12 max_l = int(sys.argv[2])
13 type_eval = int(sys.argv[3])
14 print('search between [%d, %d] type_eval %d' % (min_l, max_l, type_eval))
15 f_name = 'search_' + str(min_l) + '_' + str(max_l) + '.txt'
16
17 with open(f_name, 'w') as myfile:
18     myfile.write(f_name)
19
20 myfile.close()
21
22 NORMAL master > main.py
23 search hit BOTTOM, continuing at TOP

1 main_multithread.py
2
3 class mythread(threading.Thread):
4     def __init__(self, num):
5         threading.Thread.__init__(self)
6         self.num = num
7
8     def run(self):
9         cnt = self.num
10        exe = 'python3 main.py ' + str(min_search + cnt * scale) + ' ' + str(min_search + (cnt + 1) * scale) + ' ' + str(type_eval)
11        os.system(exe)
12
13 threads = []
14 for i in range(THREAD_CNT):
15     threads.append(mythread(i))
16     threads[i].start()
17
18 for i in range(THREAD_CNT):
19     NORMAL master > main_multithread.py
20     unix < utf-8 < python 9% 14:17
```

```
1 [|||||]100.0% 9 [|||||]100.0% 17 [|||||]100.0% 25 [|||||]100.0%
2 [|||||]100.0% 10 [|||||]100.0% 18 [|||||]100.0% 26 [|||||]100.0%
3 [|||||]100.0% 11 [|||||]100.0% 19 [|||||]100.0% 27 [|||||]100.0%
4 [|||||]100.0% 12 [|||||]100.0% 20 [|||||]100.0% 28 [|||||]100.0%
5 [|||||]100.0% 13 [|||||]100.0% 21 [|||||]100.0% 29 [|||||]100.0%
6 [|||||]100.0% 14 [|||||]100.0% 22 [|||||]100.0% 30 [|||||]100.0%
7 [|||||]100.0% 15 [|||||]100.0% 23 [|||||]100.0% 31 [|||||]100.0%
8 [|||||]100.0% 16 [|||||]100.0% 24 [|||||]100.0% 32 [|||||]100.0%

Mem[|||||] 20.8G/126G Tasks: 244, 718 thr; 32 running
Swap[|||||] 0K/8.00G Load average: 19.97 6.79 3.46
Uptime: 10 days, 23:38:39

PID/USER PRI NI VIRT RES SHR S CPU% MEM% TIME Command
3125 alfonso32 20 0 51556 8804 4432 S 0.0 0.0 0:01.14 -zsh
15903 alfonso32 20 0 57928 8976 4416 S 0.0 0.0 0:07.62 -zsh
15441 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15442 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15443 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15444 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.09 /usr/bin/python /home/alfonso329/.vim/bundle
15445 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15446 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.05 /usr/bin/python /home/alfonso329/.vim/bundle
15447 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15448 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15449 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15450 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15451 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15452 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.08 /usr/bin/python /home/alfonso329/.vim/bundle
15453 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15454 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15455 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15456 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15457 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15458 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15459 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15460 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15461 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15462 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.05 /usr/bin/python /home/alfonso329/.vim/bundle
15463 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15464 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15465 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.05 /usr/bin/python /home/alfonso329/.vim/bundle
15466 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15467 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15468 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.05 /usr/bin/python /home/alfonso329/.vim/bundle
15469 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15470 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.07 /usr/bin/python /home/alfonso329/.vim/bundle
15471 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.06 /usr/bin/python /home/alfonso329/.vim/bundle
15472 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.05 /usr/bin/python /home/alfonso329/.vim/bundle
15478 alfonso32 20 0 2540M 61692 27212 S 0.0 0.0 0:00.00 /usr/bin/python /home/alfonso329/.vim/bundle
File: /home/alfonso329/.vim/bundle/Python3.5.2/Scripts/python.exe
```

4. 評估(如果有 1/10的結果再補寫這裡)

5. 未來展望

以上方法跟知名金融科技公司模型預測比可能小巫見大巫，但短期交易(炒短線)也算能派上用場。

不久將來，若以程式作自動交易、股價追蹤，則RSI MA KD三個最基本指標，會是除LSTM等方法最需要納入模型中，甚至某些情況以上三個指標的表現可能不亞於神經網路。

但完全靠程式發大財，我認為不可能，影響股價的因素太多了：預期心理、天然災害、政治戰爭情勢、甚至有權有勢的川普、普丁、Elon Musk 一篇推文，都牽一髮而動全身。這時加入人工修正就相當重要，以人工為主、程式為輔，會是最適切的組合。

6. 心得

雖然目前仍未公布總體排名，但不論好壞，直接下市場廝殺是投資學相關科目最好的課程指導老師。

結果好則繼續精進、差強人意也無須氣餒，至少自己**學到相當多金融科技的知識**，無論從學期初的期貨與選擇權、再到學期中的機器人理專，以及最後的比特幣交易，都是收穫滿滿。每一次模型修正、參數精進，則讓我理解原來刻板的程式運作，居然**能跟造福人類的金融體系結合，這些是金錢與分數無法衡量的無價瑰寶！**

股票市場起伏如人生，本就有許多不確定因素，一如知名俗諺【三分天註定，七分靠打拼】或誠意伯劉伯溫所云【豈能盡如人意，但求無愧於心】，在能力所及之內，做好自己應盡本份，我想這就是無愧於心了。