Fintech2019 Final report
NTU CSIE R08922174 黃昱仁

Overview

這次的final，我使用過的兩個方法如下：

1. 使用**LSTM**，將資料從2週開始，逐漸append歷史資料進去，並每次預測接下來兩週的股價數值，並相減以獲得漲跌資訊，然後與答案做比較，整個訓練過程大概花費15分鐘，但可惜的是，所訓練出來的model在本地端這邊做預測沒有很好的結果，在漲跌上正確率大概只有一半一半。
2. 使用**RSI**，我修改了HW2的一些東西來跑這次 final 的這檔股票模擬，使用`exhaustive search`方法，試圖找出適合這檔股票的最佳4個parameters，分別是**長期windows**，**短期windows**，**floor**，**ceiling**，並且只使用了`TX_daily.csv`這份資料，然後用3個threads下去跑 for loop 的不同區間，加速程式碼跑的速度，整份模擬大概需要1小時，跑出來的結果平平，於是加入了random一起做考量，發現結果還不錯。

> 最後是使用了方案2，因為judge系統無法從外面load model還有準確率不佳等等因素，不使用方案一。
> 所有訓練都只有納入open price，因為myStrategy()的第三個argument只能是open price，所以沒有探討將別的價格納入考量。

Code Review

1. 首先是**mainThread_open.py**，這是將RSI search best parameters這塊加速的檔案，會在這份py檔裡spawn 4 threads並且分別呼叫另一份**myStrategy_open.py**檔裡的**training()**函式。

```
def job(end_index, q):
    print("End index:", end_index)
    time.sleep(2)
    result_param = myStrategy_open.training(end_index)
    q.put(result_param)

q = multiprocessing.Queue()
process = []
for i in range(1, 4):
    process.append(multiprocessing.Process(target=job, args=(i,q,)))
    process[i-1].start()
for i in range(3):
    process[i].join()
```

2. 接下來是**myStrategy_open.py**這份檔案，在這邊4個 for loop 裡，最裡面兩個就是跑 `ceiling & floor` 的可能性，也就是如果短期RSI值大於 ceiling 值，那麼就該賣掉，反之若小於 floor值，則該買進。最外圍兩個迴圈就是長期 / 短期的 window 視窗大小該取多少才是最好，也就是我們要盡可能在最短時間內試過所有`window_size`的組合，而我的方法就是利用`multiprocessing`，將每個 threads 的

代號以`end_index`變數表示，三個 threads 分別跑長短期相差**5, 10, 15**天，也就是差了三週以內的組合都會被我們試試看。

```
windowSizeMin=5; windowSizeMax=20;
ceilingMin=50; ceilingMax=100
floorMin=0; floorMax=50
windowSizeRange = np.arange(5, 25, 5)
windowSizeBest_1 = windowSizeBest_2 = floorBest = ceilingBest =
returnRateBest = 0
for j in range(windowSizeRange[end_index]-4, windowSizeRange[end_index]+1):
    for windowSize_1, windowSize_2 in zip(range(windowSizeMin,
windowSizeMax+1), range(windowSizeMin+j, windowSizeMax+j)):
        for ceiling in range(ceilingMin, ceilingMax+1):
            for floor in range(floorMin, floorMax+1):
                print("EndIndex=%d, WindowSize_1=%d, WindowSize_2=%d,
Floor=%d, Ceiling=%d" %(end_index, windowSize_1, windowSize_2, floor,
ceiling), end="")
                returnRate=computeReturnRate(dailyOhlcv, windowSize_1,
windowSize_2, ceiling, floor)
                print(" ==> returnRate=%f " %(returnRate))
                if returnRate > returnRateBest:
                    windowSizeBest_1=windowSize_1
                    windowSizeBest_2=windowSize_2
                    floorBest = floor
                    ceilingBest = ceiling
                    returnRateBest=returnRate
```

3. 最後是我提交的**myStrategy.py**檔案，每回合被呼叫時會將 open price column 取出來之後再 append 新的 open price value 進去，接著後面就是傳統的**RSI**過程。

```
if dataLen >= windowSize_2:
    windowedData_1=openVec[-windowSize_1:]
    windowedData_2=openVec[-windowSize_2:]
    up = down = 0
    for i in range(1, len(windowedData_1)):
        temp = windowedData_1[i] - windowedData_1[i-1]
        if temp >= 0:
            up += temp
        else:
            down += abs(temp)
    rsi_1 = (up/(up+down))*100
    up = down = 0
    for i in range(1, len(windowedData_2)):
        temp = windowedData_2[i] - windowedData_2[i-1]
        if temp >= 0:
            up += temp
        else:
            down += abs(temp)
    rsi_2 = (up/(up+down))*100
```

```
    if rsi_1 >= ceiling:
        action = -1
    elif rsi_1 < floor:
        action = 1
    else:
        if rsi_1 >= rsi_2:
            action = 1
        else:
            action = -1
```

Suggestion

希望不是一次就預測兩週下去，總覺得兩週變數太大，也不能補上新的股價接著做預測會太難預測。