Final Programming Exam. (part 1)
Deep Learning, 2022

姓名：吳佳展　　學號：110318517

- Project Title：

結合趨勢的深度強化學習股票交易策略

- Introduction：

基於強化學習方法將預測股價和投資動作結合在一起，直接以投資收益目標作為優化目標。當狀態與動作連續時，動作價值表空間過大，導致查表狀態和動作過於復雜，因而提出了使用神經網絡擬合狀態動作價值表，即深度強化學習。首次使用深度強化學習方法用於金融市場，深度學習自動感知動態市場條件，提取信息特徵，強化學習模塊與環境互動並做出交易決策，為進一步提高市場穩定性，引入模糊學習來減少輸入數據不確定性，實證結果較好。將深度強化學習用於股票交易策略和股價預測，比較 DQN、Double DQN 和 Dueling DQN 三種不同的算法，均取得不錯的投資收益。設計了基於在線策略和離線策略的離散狀態和動作模型以最大化投資回報和微分夏普比率。針對交易數據不足的問題，提出遷移學習融合 Q-learning 處理高波動金融數據帶來的過擬合問題。將強化學習的動作選擇與趨勢跟踪方法相結合，通過趨勢指標直接影響代理動作，結果顯示方法帶來較高的收益效果。

基於以上討論，本文提出了一種結合趨勢的深度強化學習股票交易模型尋找最佳交易策略。主要步驟如下：

1) 將股票開盤價和閉盤價相結合代表股票市場狀態。
2) 選取根據趨勢指標 RSI 指數調整後特定條件下的利潤作為獎勵函數。
3) 在股票市場中使用 DQN 做出交易決策。
4) 在股票數據集上的實驗結果顯示。

本文提出的模型評價指標均優於其他基準算法，使用趨勢指標調整獎勵函數明顯的有效改善模型的表現及效果。

- Problem statement：

股票交易是指在不斷變化的股票市場環境選擇股票並做出不同的交易動作從

而改變資金在市場中的分配比例，最大化投資回報率並降低風險的過程。在強化學習交易模型中，通過深度強化學習網絡，以股票的歷史數據作為狀態，實現總收益最大化為目標，在每個交易時刻之前輸出一個交易動作，並為每個交易動作提出一個獎勵，通過自動交易將資金調整到最優，不斷進行計算和自我學習，從而實現優化的股票交易模型。

● Data description：

1. 實驗數據集

驗證本文提出的模型，在實驗中，從中證 100 指數中隨機選取 3 隻股票中信證券、保利發展和水泥進行實證分析，股票代碼如下：600030, 600048, 600585。交易數據是從 Tushare 金融社區下載的每日股價數據。實驗數據為訓練週期為 2010 年 1 月 1 日至 2017 年 12 月 31 日，測試期間從 2018 年 1 月 1 日至 2021 年 12 月 31 日。

2. 實驗環境

使用語言為 Python3.6.8，並採用 Pytorch1.10.1 為運行環境。

● Reinforcement learning and neural networks structure

狀態s 由連續多個交易日開盤價和收盤價的變化情況構成。從市場中獲得每個交易日的股價波動信息，股價波動由兩部分組成，當前交易日開盤價較前一交易

$$s_t = \left(r_{t-T+1}^{oc}, r_{t-T+1}^{co}, r_{t-T+2}^{oc}, r_{t-T+2}^{co}, \cdots, r_t^{oc}, r_t^{co} \right), T \geq 2$$

$$r_t^{oc} = \frac{PO_t - PC_{t-1}}{PC_{t-1}}$$

$$r_t^{co} = \frac{PC_t - PO_t}{PO_t}$$

日閉盤價的變化率r_t^{oc}和當前交易日的閉盤價較當前交易日的開盤價的變化率r_t^{co}，其中， PO_t為t 時刻股票的開盤價， PC_t 為t 時刻股票的閉盤價。開盤價在一定程度上代表著市場消息面因素，收盤價則直接反映股價波動情況，這兩個變化率分別代表著股票在休市和開市期間的股市信息。經實驗結果表明，當時間窗口T取30時，模型效果最好。因此在本文中，$T = 30$。

趨勢使用技術分析指標來計算，比如相對強弱指數(Relative Strength Index, RSI)，順勢指標(Commodity Channel Index, CCI)等。如果趨勢向上，股價估計將上

漲，相反，如果趨勢向下，股價估計將下跌，應結束之前的多頭頭寸。因中國股市不允許賣空，這裡不考慮空頭頭寸。在本文中，使用相對強弱係數RSI決定趨勢，相對強弱係數是一個動量指標，它提供了一個超買或者超賣的信號。該指標的取值範圍為0 到100，如果其值低於30，表示超賣，其值高於70時，表示超買。相對強弱係數是根據股票前14個交易日的波動情況計算出來的，其中，average-gain 是14天內閉盤價上漲數之和的平均值，average-loss 是14天內閉盤價下跌數之和的平均值。為了規避風險，當市場處於超買情況時，股價極有可能發生反轉下跌，此時應抑制代理的買入和持有行為，鼓勵賣出行為，且下一天閉盤價下跌越多，對賣出行為的鼓勵越高；反之，當市場處於超賣情況時，股價極有可能發生反轉上漲，此時應抑制代理的賣出和持有行為，鼓勵買入行為，且下一天閉盤價上漲越多，對買入行為的鼓勵越高。通過調整特定條件下的獎勵函數值，造成對代理行為鼓勵或抑制的影響，這種調整通過在原先的獎勵值上乘以特定的影響係數m來完成，特定條件如表1所示。

RSI<30			RSI>70		
動作a_t	r_{t+1}^c	影響係數m	動作a_t	r_{t+1}^c	影響係數m
1~10	0.05~0.1	1.3	1-10	-	0.8
1~10	0.03~0.05	1.2	0	-	0.9
1~10	0.01~0.03	1.1	-10~1	-0.01~0.03	1.1
0	-	0.9	-10~1	-0.03~0.05	1.2
-10~1	-	0.8	-10~1	-0.05~0.1	1.3

表1. 不同條件下調獎勵值的影響係數

$$RSI = 100 - \frac{100}{1 + \frac{average - gain}{average - loss}}$$

$$r_t = \begin{cases} m \times r_t^{profit}, 滿足表一某條件 \\ m \times r_t^{profit}, 其他 \end{cases}$$

- Simulations

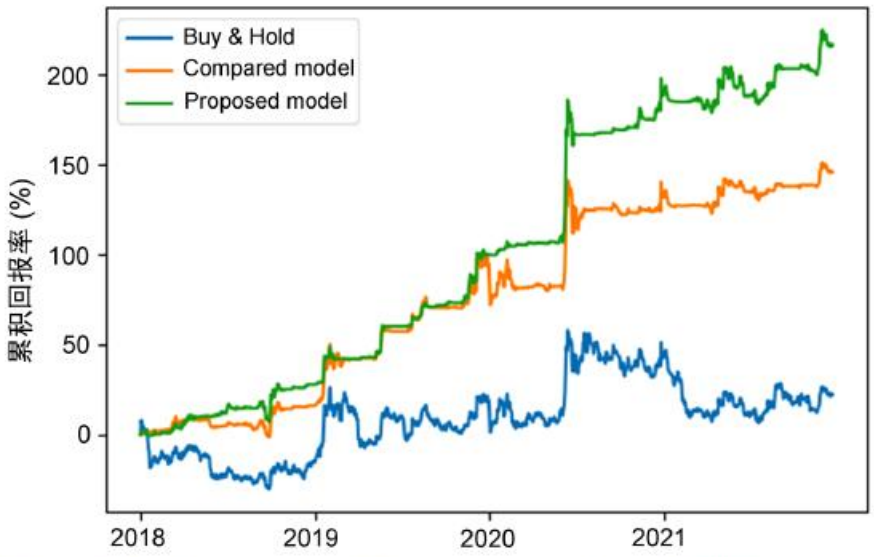

Figure 2. Cumulative returns of three investment models on 600030

圖 2. 中信證券上三種投資模型的累積收益率

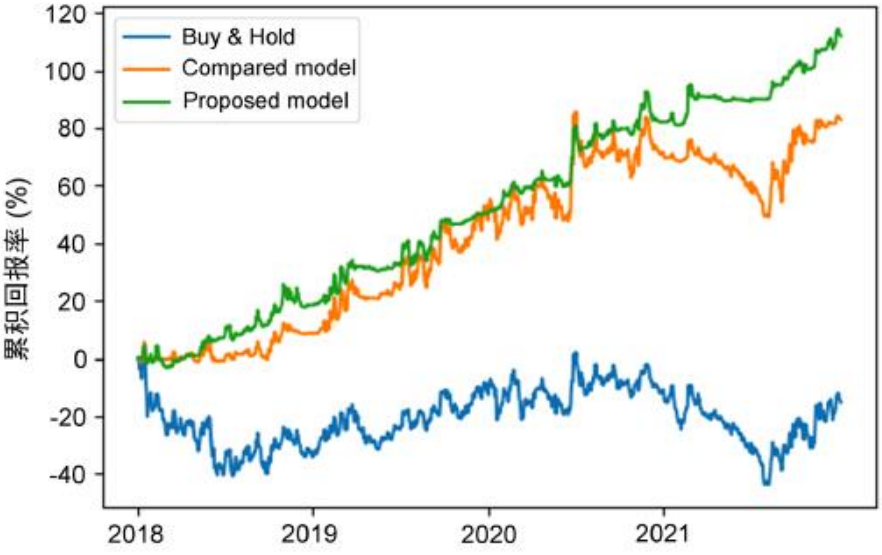

Figure 3. Cumulative returns of three investment models on 600048

圖 3. 保利發展上三種投資模型的累積收益率

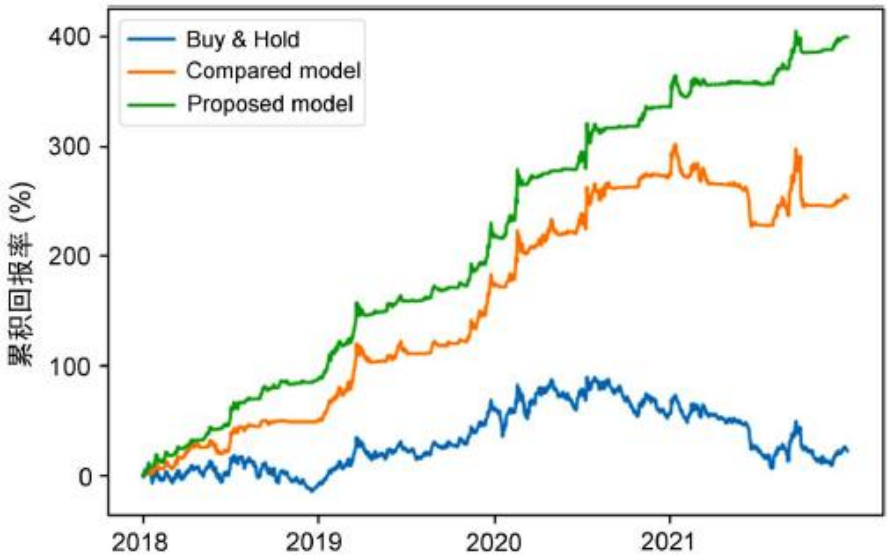

Figure 4. Cumulative returns of three investment models on 600585

圖 4. 海螺水泥上三種投資模型的累積收益率

投資模型	平均年回報(%)	年波動率(%)	夏普比率(%)
Buy & hold	5.1	35.32	19.62
Compared model	25.19	19.72	101.12
Proposed model	33.33	16.22	197.68

表 2. 中信證券上三種投資模型的評價指標對比

投資模型	平均年回報(%)	年波動率(%)	夏普比率(%)
Buy & hold	-4.04	37.46	-15.12
Compared model	16.31	19.8	121.89
Proposed model	20.65	12.9	514.62

表 3. 保利發展上三種投資模型的評價指標對比

投資模型	平均年回報(%)	年波動率(%)	夏普比率(%)
Buy & hold	-4.04	37.46	-15.12
Compared model	16.31	19.8	121.89
Proposed model	20.65	12.9	514.62

表 4. 海螺水泥上三種投資模型的評價指標對比

由上方表 2、表 3、表 4 展示了三種投資模型在中信證券、保利發展和海螺水泥上的評價指標對比情況。從表中可以看出基於傳統獎勵函數的強化學習投資模型可以獲得可觀的回報，並且其年波動率和夏普比率不錯。與前者相比，本文構建的投資模型的平均年回報率提升了 30%，年波動率降低 25%，夏普比率提升 50%以上。

● Conclusions

將股價趨勢與強化學習方法中的獎勵函數設定相結合，通過模型行動和股價在不同條件下的影響係數調整獎勵函數，使之構建成新的深度強化學習股票交易模型並應用於股票交易。本文在中國股票市場中選擇了 3 支股票進行投資實驗，實驗結果顯示，本文的模型表現優於其他對照組，在實驗期間的平均年回報更高，年波動率更低，且夏普比率更好，表明了在股票交易上的有效性，有較好的應用價值。但是本文的模型是基於一些假設進行的，不符合市場中實際投資者的投資方式。例如當交易量較大對股價造成影響時，本文的模型不適用，因此有待進一步研究與探索。

● Code

```
def m2_initialize_bigquant_run(context):
    context.pre_act = context.options['data'].read_df()
    context.set_commission(PerOrder(buy_cost=0.0003, sell_cost=0.0013, min_cost=5))
    stock_count = 5
    context.stock_weights = T.norm([1 / math.log(i + 2) for i in range(0, stock_count)])
    context.max_cash_per_instrument = 0.2
    context.hold_days = 5
    return stock_count

def m2_handle_data_bigquant_run(context, data):
    sid = symbol('000001.SZA')
    action = context.pre_act[context.pre_act['date'] == data.current_dt.strftime('%Y-%m-%d')]
    cur_position = context.portfolio.positions[sid].amount
    if len(action['pre_action'])>0:
        if int(action['pre_action'])==0 and  cur_position == 0:
            context.order(sid, 100)
        elif int(action['pre_action'])==1 and cur_position > 0:
            context.order_target_percent(sid, 0)

m7 = M.instruments.v2(
    start_date='2012-01-01',
    end_date='2016-01-01',
    market='CN_STOCK_A',
    instrument_list="000001.SZA",
    max_count=0)


m8 = M.advanced_auto_labeler.v2(
    instruments=m7.data,
    start_date='',
    end_date='',
    benchmark='000300.SHA',
    drop_na_label=True,
    cast_label_int=True)
```

圖 5. 執行程式使用 functin

```
m12 = M.join.v3(
    data1=m8.data,
    data2=m10.data,
    on='date,instrument',
    how='inner',
    sort=False
)

m14 = M.instruments.v2(
    start_date='2016-01-01',
    end_date='2017-01-01',
    market='CN_STOCK_A',
    instrument_list='000001.SZA',
    max_count=0
)

m19 = M.advanced_auto_labeler.v2(
    instruments=m14.data,
    start_date='',
    end_date='',
    benchmark='000300.SHA',
    drop_na_label=True,
    cast_label_int=True
)
```

圖 6. 執行程式使用 functin

```
m16 = M.derived_feature_extractor.v3(
    input_data=m15.data,
    features=m17.data,
    date_col='date',
    instrument_col='instrument',
    drop_na=False,
    remove_extra_columns=False
)

m18 = M.join.v3(
    data1=m19.data,
    data2=m16.data,
    on='date,instrument',
    how='inner',
    sort=False
)

m4 = M.dropnan.v1(
    input_data=m18.data
)

m1 = M.DQN_model.v9(
    data=m12.data,
    data_2=m4.data
)

m3 = M.dropnan.v1(
    input_data=m1.data_1
)
```

圖 7. 執行程式使用 functin

```
m17 = M.input_features.v1(
    features="6"
)

m15 = M.general_feature_extractor.v7(
    instruments=m14.data,
    features=m17.data,
    start_date='',
    end_date='',
    before_start_days=20
)

m16 = M.derived_feature_extractor.v3(
    input_data=m15.data,
    features=m17.data,
    date_col='date',
    instrument_col='instrument',
    drop_na=False,
    remove_extra_columns=False
)

m18 = M.join.v3(
    data1=m19.data,
    data2=m16.data,
    on='date,instrument',
    how='inner',
    sort=False
)
```

圖 8. 執行程式使用 functin

```
m1 = M.DQN_model.v9(
    data=m12.data,
    data_2=m4.data
)

m3 = M.dropnan.v1(
    input_data=m1.data_1
)

m2 = M.trade.v4(
    instruments=m14.data,
    options_data=m3.data,
    start_date='',
    end_date='',
    initialize=m2_initialize_bigquant_run,
    handle_data=m2_handle_data_bigquant_run,
    prepare=m2_prepare_bigquant_run,
    before_trading_start=m2_before_trading_start_bigquant_run,
    volume_limit=0.025,
    order_price_field_buy='open',
    order_price_field_sell='close',
    capital_base=100000,
    auto_cancel_non_tradable_orders=True,
    data_frequency='daily',
    price_type='后复权',
    product_type='股票',
    plot_charts=True,
    backtest_only=False,
    benchmark='')
```

圖 9. 執行程式使用 functin

References

[1] Jeong, G. and Kim, H.Y. (2019) Improving Financial Trading Decisions Using Deep Q-Learning: Predicting the Number of Shares, Action Strategies, and Transfer Learning. Expert Systems with Applications, 117, 125-138.

[2] Chakole, J. and Kurhekar, M. (2020) Trend Following Deep Q-Learning Strategy for Stock Trading. Expert Systems, 37.

[3] Leem, J. and Kim, H.Y. (2020) Action-Specialized Expert Ensemble Trading System with Extended Discrete Action Space Using Deep Reinforcement Learning. PLoS ONE, 15, e0236178.