Deep Q-Networks For Multi-Asset Trading Decisions

Shen Gao | shengao@stanford.edu

Motivation

- Computers are replacing highly paid traders on Wall Street. Trading in its essence is a decision making process in an environment of information flow.
- Unlike many other problems, financial data is very fluid. Relationships and patterns are short-lived and hard to generalize in particular.
- Deep Q-Network (DQN) by definition trains an agent that acts upon observation of sequential information. Promising for trading problems.

References

[1] Kavukcuoglu K. Silver D. et al Mnih, V. Human-level control through deep reinforcement learning. Nature, 518:529–533, 01 2015.

[2] Gordon Ritter. Machine learning for trading. SSRN Electronic Journal, 01 2017.

[3] Xiang Gao. Deep reinforcement learning for time series: playing idealized trading games. Arxiv Quant, 03 2018.

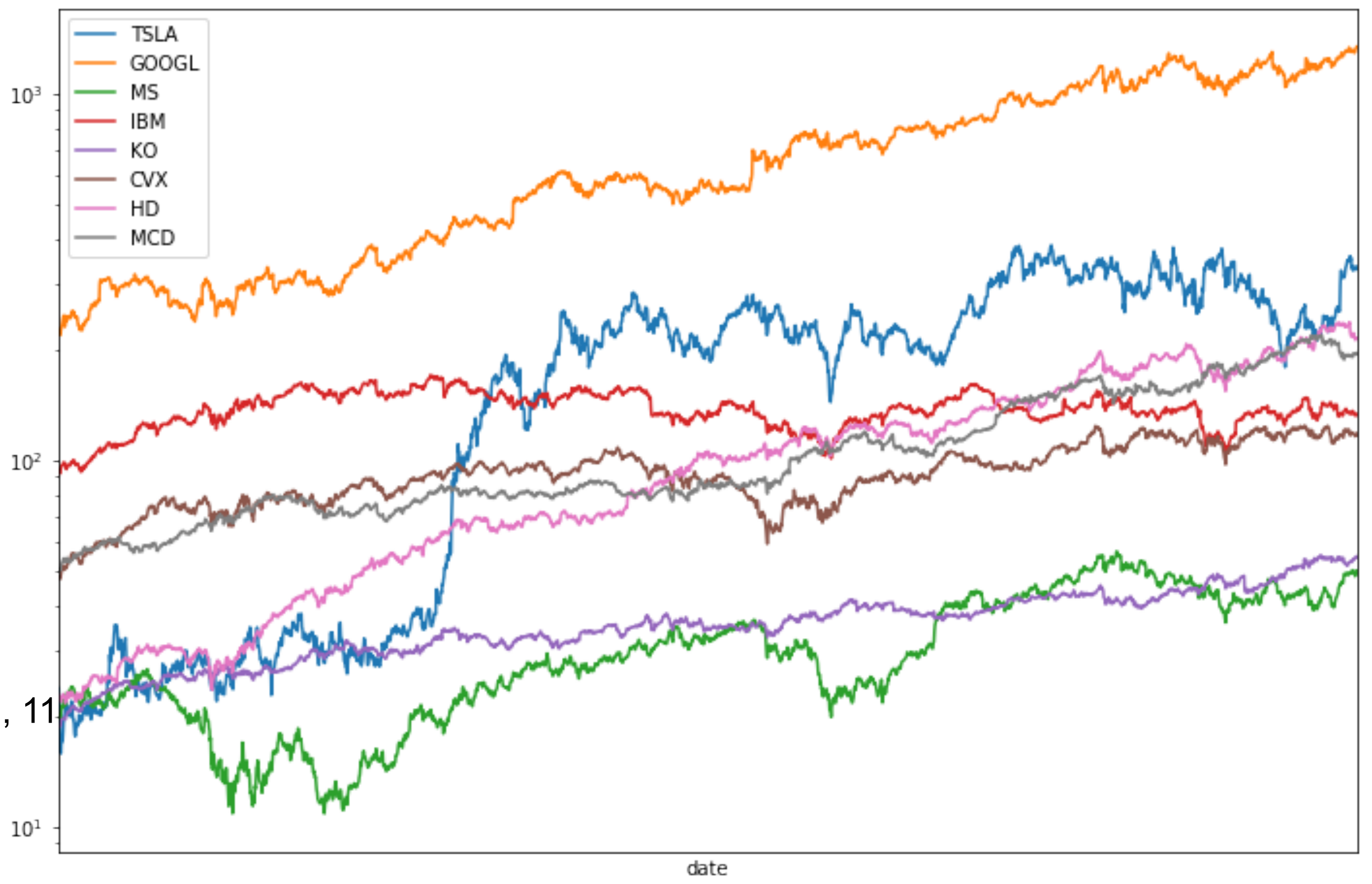
[4] Chien Huang. Financial Trading as a Game: A Deep Reinforcement Learning Approach , 07 2018.

[5] Yang Wang, et el. Q-trading, CSLT Technical Report, 01 2017.

[6] Taewook Kim and Ha Kim. Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries. Complexity, 2019:1–20, 11 2019.

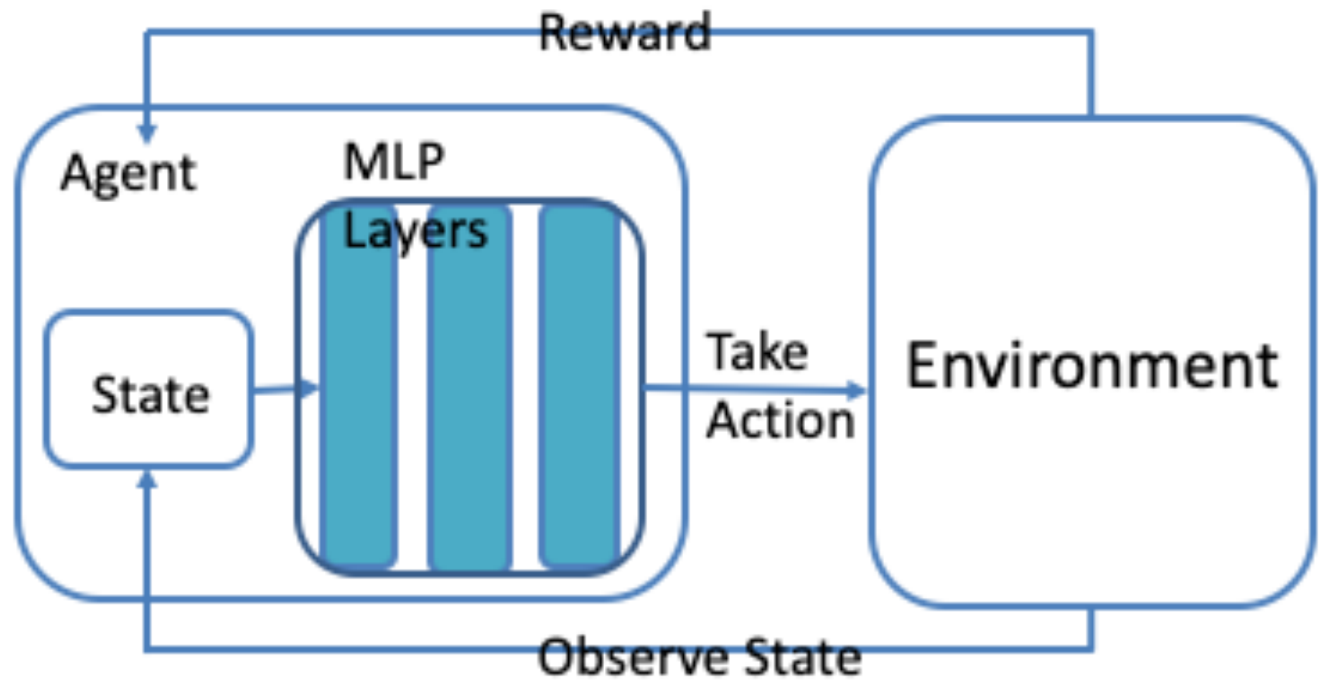
Dataset and Feature

- Daily adjusted close price time series for 8 stocks from alpha-vantage API: Tesla, Google, Morgan Stanley, IBM, Coca Cola, Chevron, Home Depot, Mcdonald's
- Training set: 2010/06/29 – 2016/6/13
- Testing set: 2016/6/14 – 2019/12/01
- Reward is not directly evaluated on the raw time series, but rather using exponentially weighted average price for reduced noise in volatility.



Method

```
Algorithm 1 DQN Trader
  Initialize environment E, recurrent Q-network, $Q_\theta$, agent A with $Q_\theta$ as input
  Initialize Simulator with inputs E and A for training and testing
  for  each episode do
    Observe initial state s from E
    while  each time step in E do
      Select greedy action a w.r.t $Q_\theta$ and apply to E
      Receive reward r and next state s' from env E
      Store memory (s, a, r, s') to A's memory M
      Agent A replays memory M up until current time step:
      Randomly sample a batch of memories from M
      for each batch i do
        Get (s, a, r, s') from batch i
        Train $Q_\theta$ based on the Bellman Equation via TensorFlow
        $L(\theta) = E_{s,a,r,s'}[||r + \gamma Q_\theta(s', argmax_{a'} Q_\theta(s', a')) - Q_\theta(s, a)||^2]$
        where $Q_{\theta'}$ is the Q-network output given the next state s' and the next action a'
        $\theta = \theta - \alpha \nabla_\theta L(\theta)$
      end for
      Update s = s'
    end while
    if Exploration threshold $\epsilon$ greater than 0.1 then
      Reduce $\epsilon$ by multiplying a decay factor 0.995
    end if
  end for
```



State Space (S): (20-day lookback time series, 8 stocks). State is normalized by window mean and standard deviation.

Action Space (A): {-1000, 100, 0, 100, 1000}, 8 stocks)

Reward Function: $(P_{t+1} - P_t)N_t - c|N_t - N_{t+1}|$

where

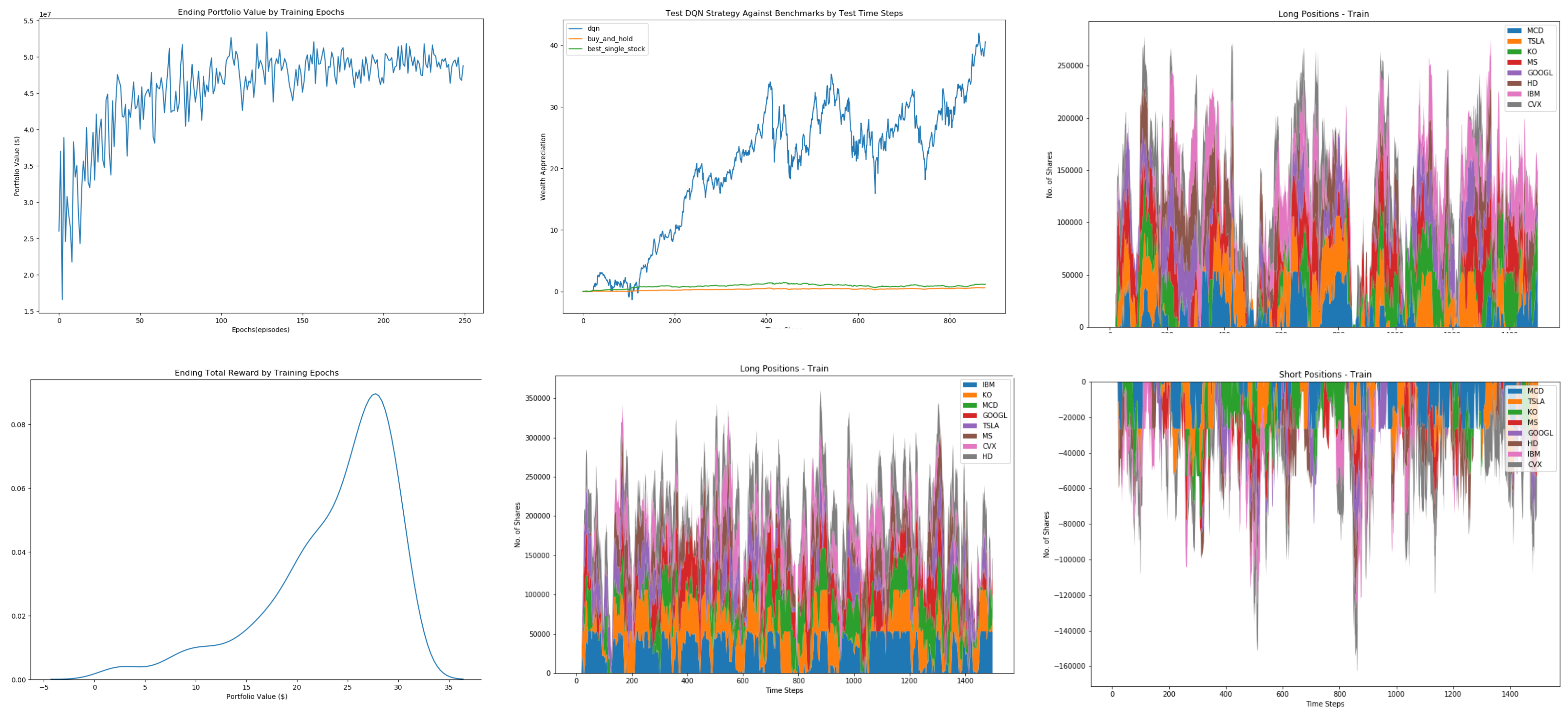
P_t, P_{t+1}: stock prices at current state and next state

N_t, N_{t+1}: stock positions (in shares) at current state and next state

c: fixed transaction price per share traded (buy or sell)

Reward is normalized by initial cash. For long only strategy, allocate cash by buy cost across assets. Negative reward is penalized 10% more.

Results



- Model performs greatly and outperforms simple buy and hold strategy by orders of magnitude.
- Judging from Fig 2, the DQN seems to have learned a strategy that makes money most of the time and lose money occasionally. It's surprisingly good considering both transaction costs and risk aversion further penalizes rewards.
- DQN network tends to overfit to training environment. We use experience replay and random exploration to reduce it.
- When short trade and cash borrow is allowed in Fig 5&6, the model seems to take full advantage of the extra leverage to deliver outsized gains.

Fig top to bottom, then left to right:
1) Portfolio value over training set (model metric)
2) Average reward distribution (training set)
3) Portfolio value over test period, compared with simple hold strategy
4) Long only strategy trades over time (color for assets)
5) Long-short strategy buy trades over time (color for assets)
6) Long-short strategy sell trades over time (color for assets)

Discussion & Future Work

- Further tests needed to show if the strategy is robust in a bear market condition since data is drawn from last decade of bull market.
- It's interesting to use other metrics to assess model performance, such as Sharpe Ratio, Max Drawdown, etc.
- There might be survivorship bias as the stocks selected for testing are all well-known large cap stocks. It's interesting to test against worse-performing assets, assets significantly affected by random events and other asset classes.
- Test other neural network structure other than MLPs, such as GRU, LSTM, attention layers, etc.
- Explore other deep reinforcement learning frameworks such as double q-learning, policy gradient, etc.