Financial Assets Optimization with Reinforcement Learning Inference

Junwoong Yoon (junwoony) [1] **Muhammed Shuaibi (mshuaibi)** [1] **Mingjie Liu (mingjie1)** [1]

1. Introduction

Financial institutions are constantly in search of stock trading strategies that can produce higher returns. Such strategies enable these institutions to either directly profit from such trades or profit on the high fees they charge to their clients. Decades of experience has been spent trying to better understand how these markets work, with the core question being: when and what stocks do I buy or sell?

Recent developments in machine learning and AI has opened the way to more profitable trading optimization in the complex and dynamic stock market. Machine learning based methods have the ability to recognize patterns and/or critical factors from given data and make predictions based on what they learn from the data. However predicting the stock market is very challenging due to the high noise to signal ratio and various latent factors that affect stock prices. Therefore, more sophisticated machine learning methods have been explored to forecast overall trends of stock prices in the pool of uncertainty.

One of the most intuitive supervised learning approaches for trading decision making is the Long Short-Term Memory (LSTM) neural network. However, despite its intuitiveness and that it could quickly adapt to the new stock price by incorporating new training data, many studies show that LSTM tends to predict the stock price for the next day to be very close to the closing price of the previous day. In that sense, the prediction from LSTM will not be useful for generating any profit. On the other hand, reinforcement learning, whose formulation does not rely on merely matching the stock price, could potentially alleviate this problem and make the prediction more useful.

The development of more sophisticated reinforcement learning models (Ng et al., 2000; Argall et al., 2009) has enabled researchers to examine more complex problems within AI. Of these includes financial market transactions, specifically, the stock market. Models deployed that behave greedily and seek to exploit a new trading strategy may not consider the greater repercussions that come in leading to a crash in the market (Frye). This work proposes to employ a class of RL models that seeks to model decision making as a probabilistic inference problem, namely - RL as inference. Throughout this project we will explore how modeling financial assets as an inference problem can potentially enhance our performance and more so, empower financial institutions with answers to future financial queries.

Reinforcement learning (RL) applications to robotics (Singh et al., 2019), video games (Silver et al.), and financial markets (Meng & Khushi, 2019; Xiong et al., 2018; Pendharkar & Cusatis, 2018) have shown great success in recent years in demonstrating the capabilities of such models. These models seek to learn a set of actions for any given state, namely, a policy function, that maximizes some reward functions that measure the environments response to such actions.

Xiong et al. (2018) have incorporated a deep reinforcement learning algorithm, namely Deep Deterministic Policy Gradient (DDPG), into stock trading strategy optimization to maximize return in the stock market. DDPG is an improved version of Deterministic Policy Gradient (DPG) model that combines the Q-learning and policy gradient frameworks. Unlike DPG, DDPG uses neural networks as function approximators. The DDPG consists of two main components: (i) an actor-critic framework in which the actor network maps states to actions while critic network computes the value of the action. (ii) an experience replay that improves the usage of data by removing correlations between sample transactions drawn from the state-action history. The DDPG reports higher return compared to the traditional min-variance portfolio allocation and Dow Jones Industrial Average (DJIA) but the scalability to larger data and the robustness to dynamic environment are still questionable.

(Pendharkar & Cusatis, 2018) have proposed and tested several RL agents for trading in financial markets for a personal portfolio. They proposed and compared an on-policy state-action-reward-state-action (SARSA), an off-policy Q learning algorithm and an online gradient descent method with TD(λ) algorithm for learning continuous agent actions. Also, they compared the results for using static knowledge agent and adaptive knowledge agent. They found that the gradient descent method with adaptive knowledge agent is able to get most profits compared to the other methods. It is able to beat the $S\&P500$ and AGG

[1]Carnegie Mellon University, Pittsburgh, PA 15213, USA.

10708 Class Project, Spring, 2019.

portfolio by a large magnitude.

The examples cited above seem very promising for stock trading. Unfortunately, those RL methods seek to maximize the reward function without the need of inference. Without inference, if we deploy the RL agents on lager scales, they could lead to catastrophic results. For example, for stock trading, it could potentially cause a flash crash while the RL agent is exploring trading strategies (Frye & Feige, 2019). The lack of inference is a common problem in traditional RL. We can foresee a scenario such that our agent learns to exploit the reward function in an unpremeditated way (Amodei et al.). "Reward hacking", as this scenario has commonly been referred to, has become a major concern for reinforcement learning and AI as designing a reward or loss function is not always obvious (Amodei et al.). To address some of these concerns, RL models casted as graphical models, referred to as RL as inference, offer some insight to addressing these issues. (Levine, 2018) presents a detailed discussion and overview of how, rather than defining a, sometimes arbitrary, reward function, RL as inference uses rewards to induce distributions in which the optimal policy seeks to match. This is achieved by the introduction of binary optimality variables characterized by a distribution over the rewards. In casting RL as an inference problem, answers to a variety of decision making queries are also available, including (Xing et al.):

- Inferring optimal sequences given rewards;
- Inferring prior rewards and actions given optimal sequences;
- Inferring a policy function given rewards

(Fu et al.) have proposed Variational Inverse Control with Events (VICE), an RL as inference framework that seeks to maximize the probability of an event happening as opposed to maximizing the cumulative reward. This is achieved by replacing rewards with log-probabilities of events. The proposed model directly addresses the issue of reward hacking by using examples of desired outcomes instead of actual demonstrations. In doing so, the authors have developed a framework that does not utilize agent observations but rather probabilities of events occurring, a more real world like representation. Their framework can best be put in context by the following examples: i) In a traditional RL scenario, a cleaning robot rewarded for picking up crumbs could game the model by repeatedly scattering the crumbs and picking them up again; ii) In the VICE framework, a cleaning robot estimates from its observations if its responsibilities have been completed, never receiving direct feedback whether the room is fully cleaned. It is worth noting that although inverse reinforcement learning, a field of RL that attempts to learn the reward function from expert behavior of an agent, does not involve explicit reward specification, it requires expert demonstrations on how to perform the task. VICE was shown to significantly outperform a variety of RL-based algorithms in a variety of tests including the Pusher, Ant, and Maze tasks. As a result, in this project, we want to explore the potential of reinforcement learning as inference for managing financial assets.

In this report, we present two different RL models that learn stock trading strategies to improve profits. The first model is based on Deep Q-Network (DQN) (Mni) algorithm to train a policy that tries to maximize cumulative reward. The second model uses policy gradient methods for the same objective. We then expand both baseline models to incorporate soft gradient optimality to encourage model exploration. Soft Q-Learning and soft policy gradients were benchmarked against their standard counterparts. The remainder of this report will introduce our proposed models, summarize the results, and provide an outlook for future work.

2. Methods

Baseline models

MODEL 1

We first propose a Deep Q-Network (DQN) model to learn the trading strategy necessary to maximize our profits. In this problem formulation, our agent must decide, for each day, whether a share of a stock is to be bought, sold, or held onto. Such decisions must be made based off a rolling window horizon - providing access to the last N_h trading days. The action space of this model is $3N_s$, with N_s being the number of stocks available to choose from in our portfolio. We formulate our DQN model with the following variables:

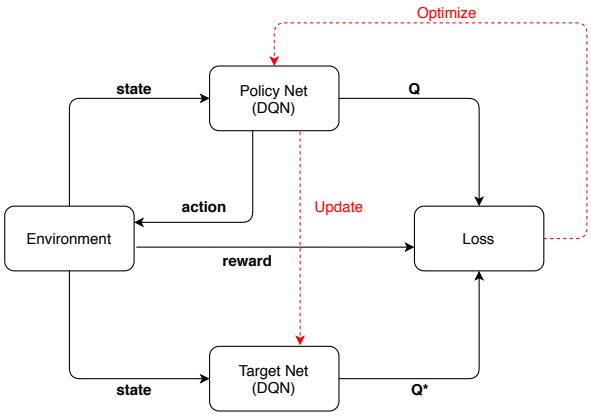


Figure 1. Overall workflow of model 1 with DQN

State: The states, s, consist of the per-day differences in

closing prices for the past training window N_h. Our window size regulates the number of historic stock prices our model has access to before taking an action.

Action: The actions, *a*, in our model include: buy, sell, and hold for each stock in our portfolio. In order to reduce the complexity surrounding this already large action space, we establish that in any particular day we are only able to buy or sell 1 share of a stock.

Rewards: The reward, *r*, is the sum of the individual day rewards over the complete trading time horizon. For a particular day, r is defined for a variety of actions:

- buy: $r = 0$
- hold: $r = 0$
- sell (with inventory > 0): r = close price - buy price
- sell (with inventory = 0): $r = -1$

Under this reward structure our agent seeks to maximize large profit transactions. Tuning these definitions can lead to the model optimizing towards certain policies over others. For example, if selling with inventory > 0 was rewarded with $r = 1$ then our agent seeks to learn a policy that will generate any profit rather than the greatest.

Algorithm 1 DQN with Experience Replay

Initialize policy model
Initialize target model Allocate memory M to experience replay H
for episode = 1, ..., N **do**
  Initialize trading environment
  for t=1, ..., T **do**
    Randomly draw ϵ
    if $\epsilon \leq$ threshold **then**
      Select a random action a_t
    else
      $a_t = \arg\max_a Q(s, a; \theta)$
    end if
    Execute action a_t and observe r_t
    Push (s_t, a_t, s_{t+1}, r_t) into replay H
    Set $s_{t+1} = s_t$
    Sample random minibatch from H
    Compute expected Q value:

$$Q^* = r + \gamma V(s')$$

    Compute Loss(Q, Q^*) and Optimize DQN
    Update model parameters:

$$\theta \leftarrow \theta + \alpha \nabla_\theta Q_\theta(s, a)\big(r(s, a) + \gamma V(s') - Q_\theta(s, a)\big)$$

    For every C steps, reset $\hat{Q} = Q$
  end for
end for

The underlying driving principle of our DQN comes from learning $Q(s, a)$ a state-action pair that tells the agent what action is optimal at a given state. Unlike traditional Q-learning where these values get tabulated through experiences, our DQN algorithm trains a neural network to approximate $Q(s, a)$. Our DQN seeks to then optimize the following loss function:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')}[(r + \gamma V(s') - Q(s, a; \theta)] \quad (1)$$

$$Q(s, a) = \max_\pi \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 + r_{t+2} + \dots] \quad (2)$$

For Q-learning,

$$V(s') = \max_{a'} Q(s', a'; \theta^*) \quad (3)$$

For Soft Q-learning,

$$V(s') = \log \int \exp Q(s', a'; \theta^*) da'. \quad (4)$$

Where θ^* and θ correspond to the network parameters of the Q-network and target network, respectively. To overcome some of the limitations that come with moving targets in RL problems - we also incorporate experience replay (Mni) to help with the stability of our model. The full algorithm of our DQN with experience replay as inspired by (Mni)is shown in 1.

MODEL 2

The second model we propose is different from the first model in the trading mechanism. Instead of giving the agent options of buy, sell or hold at each trading day, we give the agent a portfolio of cash and stocks, and the agent can distribute the total balance it has to those properties at each trading day. We are interested in maximizing the profit, and this usually corresponds to spending all money in the most promising stock, if we do not consider risk management. Therefore, we first formulate the RL variables as follows:

State: The state, s, is defined as the percentage price changes of all the properties in the previous N_h trading days, where N_h is a hyperparameter that defines how much history the agent can get access to.

Action: The action, a, defines which property the agent will spend money on for the the next trading day. The property could be cash or stock.

Rewards: The reward, r, is calculated in each trading day. It is equal to the difference of the balance of the agent one trading day after the action.

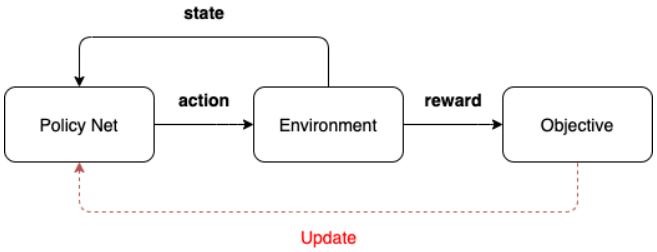


Figure 2. Overall workflow of model 2 with policy gradient

Basically, our goal is to choose the action that maximize the reward at each state. To learn the actions, we model it as a function of the state, which is the policy function $\pi_\theta(a|s)$ in our model. Specifically, $\pi_\theta(a|s)$ follows a deep neural network structure with a softmax output such that it represents the probability of each action in the given state. Therefore, in this model, the objective function is:

$$\begin{aligned} J(\theta) &= \sum_{s\in S} V_\pi(s) \\ &= \sum_{s\in S} \mathbb{E}_\pi[G_t|s_t = s] \\ &= \sum_{s\in S} \mathbb{E}_\pi[\sum_{k=0}^{T} \gamma^k r_{t+k+1}|s_t = s] \end{aligned}$$

Through some derivation, we can get that:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[G_t \nabla_\theta \ln \pi_\theta(a_t|s_t)] \tag{5}$$

Then, we can use the REINFORCE (Monte-Carlo policy gradient) method to optimize the objective function (maximize the total reward). The detailed algorithm is listed below:

Algorithm 2 REINFORCE (Monte-Carlo policy gradient)

Initialize policy model
for episode = 1, ..., N **do**
  Generate M trajectories on policy π_θ: s_1^m, a_1^m, r_2^m, s_2^m, a_2^m, ..., s_T^m
  for t=1, ..., T **do**
    Compute $G_t^m = \sum_{k=0}^{T-t} r_{t+k+1}$
    Update policy parameters:

$$\theta \leftarrow \theta + \frac{1}{M} \sum_m \alpha \gamma^t G_t^m \nabla_\theta \ln \pi_\theta(a_t^m|s_t^m)$$

  end for
end for

In the algorithm 2, α is the learning rate and γ is the decay factor. The number of trajectory is introduced in order to reduce the variance of the learning process.

RL as Inference

One concern with the baseline models is that they will be stuck in the local optimal by exploitation and not able to achieve the real optimality. as a result, we also proposed to implement the inference counterparts of the baseline models shown above. Specifically the soft Q-learning and the soft policy gradient algorithms.

Soft Q-Learning

To encourage our agent to explore we can define a non-zero probability across all actions for our Q function : $\pi(a|s) \propto \exp Q(s,a)$. Stemming from a Boltzmann distribution, this corresponds to a negative energy Q-function. A policy defined through the energy form can then be shown to be a solution to the maximum-entropy objective:

$$\pi^*_{MaxEnt} = argmax_\pi \mathbb{E}\left[\sum_{t=0}^{T} r_t + \mathbb{H}(\pi(*|s_t))\right]$$

A solution to the optimal maximum entropy objective can be done by introducing the soft Bellman equation:

$$Q^*_{soft}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1}\sim p_s}\left[V^*_{soft}(s_{t+1})\right]$$
$$V^*_{soft}(s_t) = \alpha \log \int_{\mathbb{A}} \exp\left(\frac{1}{\alpha} Q^*_{soft}(s_t, a')\right) da'$$

Our update equation remains the same, except for our target value V(s'):

$$\theta \leftarrow \theta + \alpha \nabla_\theta Q_\theta(s,a)\big(r(s,a) + \gamma V(s') - Q_\theta(s,a)\big)$$
Q-Learning: $V(s') = max_{a'} Q_\theta(s', a')$
Soft Q-Learning: $V(s') = softmax_{a'} Q_\theta(s', a')$
$$softmax_{a'} Q_\theta(s', a') = \log \int \exp\big(Q_\theta(s', a')\big) da'$$

Soft Policy Gradient

In soft policy gradient, instead of optimizing the expected reward at each time step, it optimizes the KL divergence of the real optimal trajectory and the trajectory following the policy function, as shown below:

$$
\begin{aligned}
J(\theta) &= -D_{KL}(\hat{p}(\tau)||p(\tau)) \\
&= \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim\hat{p}(s_t,a_t)}[r(s_t,a_t)] \\
&+ \mathbb{E}_{(s_t)\sim\hat{p}(s_t)}[H(\pi(a_t|s_t))]
\end{aligned}
$$

Through similar derivation as that in the baseline model, we can get that:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi[\sum_{t'=t}^{T}(r(s_{t'},a_{t'}) - \ln \pi_\theta(a_{t'}|s_{t'}))\nabla_\theta \ln \pi_\theta(a_t|s_t)] \tag{6}$$

Therefore, the update rules in Algorithm 2 becomes:

$$\theta \leftarrow \theta + \frac{1}{M}\sum_m \alpha\gamma^t(\sum_{k=0}^{T-t} r_{t+k+1} - \log \pi_\theta(a^m_{t+k+1}|s^m_{t+k+1}))) \\ \nabla_\theta \log \pi_\theta(a^m_t|s^m_t)$$

3. Experiments and Results

Dataset Information

We used Kaggle's open source dataset "Huge Stock Market Dataset" that contains full historical daily price and volume information for all US-based stocks and ETFs tradning on NYSE and NASDAQ. This dataset consists of date, prices (open, high, low, and close), volume, and open interest. We constructed two datasets we thought to be correlated and uncorrelated. Dataset A includes Facebook (FB), Apple (AAPL), Amazon (AMZN), Netflix (NFLX), Google (GOOGL), and Microsoft (MSFT). Dataset B includes Boeing (BA), Gilead (GILD), Microsoft (MSFT), Oil ETFs (USO), Gold ETFs (GLD), and Treasury Bonds (TLT). From these datasets we hope to examine the performance of our model trained on certain stocks and inferring on others.

For our preliminary work, we began by establish baselines through only one stock. We selected stock price history of Apple Inc. (AAPL) for the baseline experiment to estimate prospective profits from the two different models. For training both models, we used one-year price history of apple, starting from "2012-01-02" to "2012-12-31". Our models were then evaluated on our test set - AAPL stock prices for the consecutive year: "2012-12-31" to "2013-12-31". In both scenarios the 'Close' price was used for that day's stock price.

Results

Figures 3 and 4 shows the predicted day by day trading strategies for the DQN and REINFORCE models. In all our preliminary runs we have demonstrated our models' ability to successfully predict profitable transactions. It is worth nothing that the two models cannot be directly compared as their underlying trading strategies differ - the DQN model is limited to buying/selling one stock per day while the REINFORCE will spend all of the equity on one stock in each trading day.

Despite the successes, a few obstacles still remain. The DQN model ran into issues of not being able to consistently demonstrate the apparent results. This is further supported when examining the corresponding Loss/Reward vs Episode plots - as the loss is cyclical and the rewards are not always increasing. A careful hyperparameter tuning will be necessary including # of episodes, target model update frequency, Q-network parameters, etc.

For the REINFORCE model, it is able to gain significant amount of profit from the training data, indicating that it is able to exploit the structure of the stock prices in the training set. However, it can be found that the profits on the test set is much smaller.Also, from the trading strategies plots, we can see that in the test set, the agent sometimes buy at the peak and sell at the bottoms, resulting in the loss of equity. There are two main reasons for worse performance in the test set. First, the AAPL stock in the test set does not go up as much as that in the train set. Second, the shapes of the stock prices is different in the period of time during the training set and the test set. The model is overfitted to the structure in the training set and could not generalize to the test set. Despite the overfitting problem, the profit in the test set is still higher than spending 50% money on cash and 50% money on AAPL and hold it for the whole time, which corresponds to $ 201.77 profits, indicating that the model can successfully generate profits. Then, we use the same model for testing on other stocks, including AMZN, BA and FB. We find that for all of the stocks, our trading agent is bale to generate profits. However, since the price of every stock is increasing over the period in the test data, the profits are really small . For example, for FB, the trading agent is able to generate a profit of $ 275.74, which corresponds to 27.5% increase. However, the stock price is about doubled from the beginning. As a result, the trading bot is not able to generate more profits than random guessing. The results for the RL as inference is also shown for those stocks. The performance of RL as inference is very similar to those from the baseline model. As a result, it is not clear that RL as inference can significantly improve the performance of our model.

4. Conclusion and Future Works

At our current stage, we present two different baseline RL models: DQN and REINFORCE. DQN uses neural networks that predict Q values to guide selecting optimal ac-

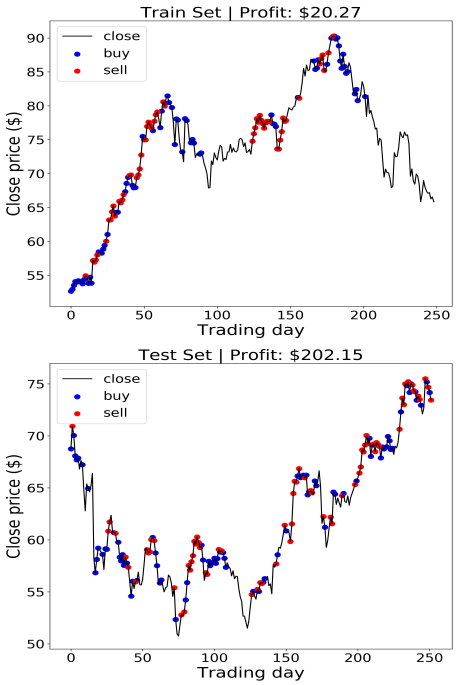


Figure 3. DQN trading strategies and corresponding profits on the train and test sets given a starting budget of $ 1,000.

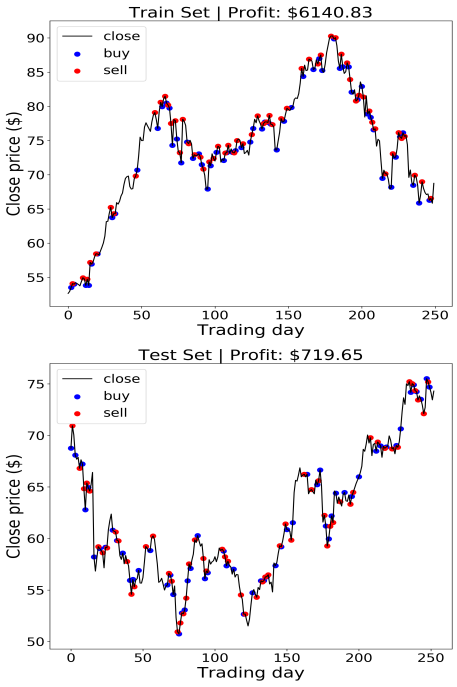


Figure 4. REINFORCE trading strategies and corresponding profits on the train and test sets given a starting budget of $ 5,000.

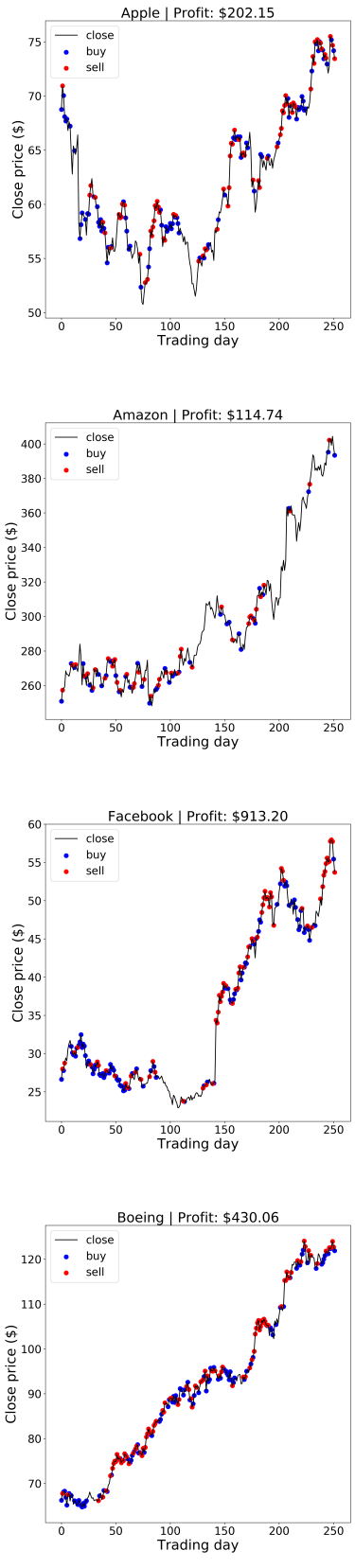
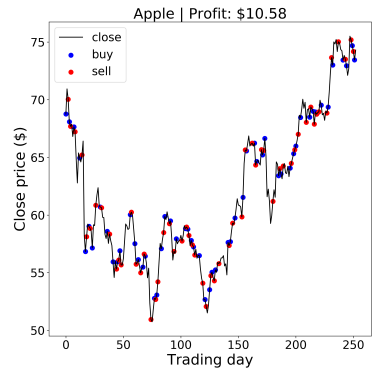
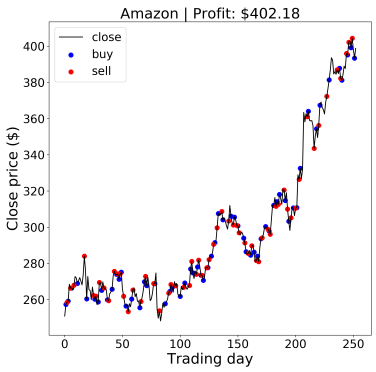
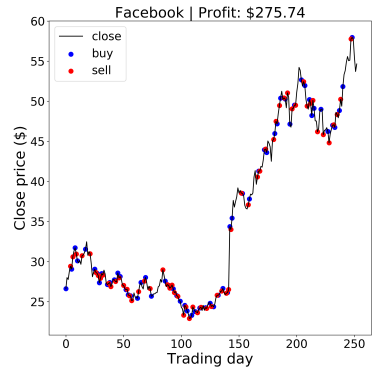


Figure 5. Baseline DQN test results







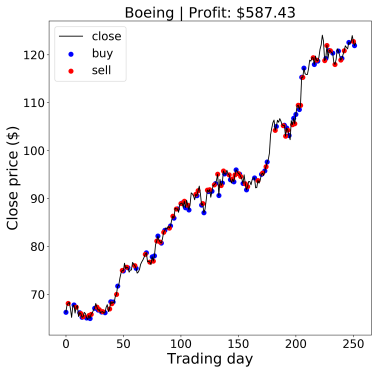
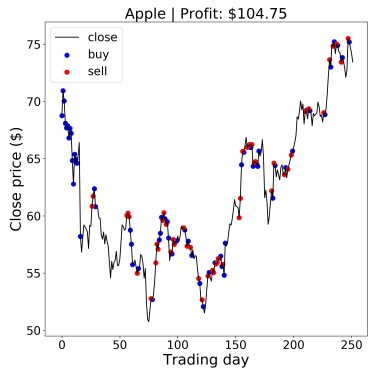
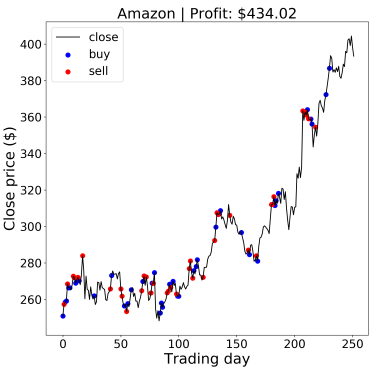
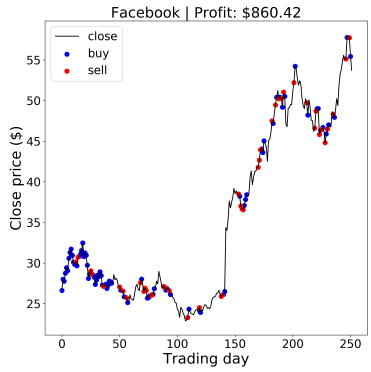


Figure 6. Baseline REINFORCE test results







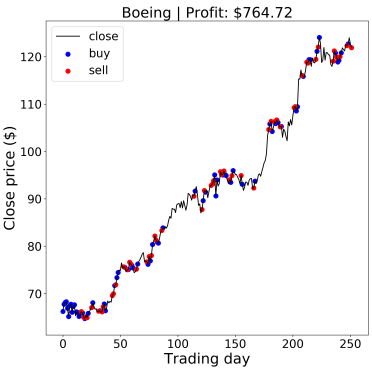


Figure 7. DQN with soft Q-learning test results

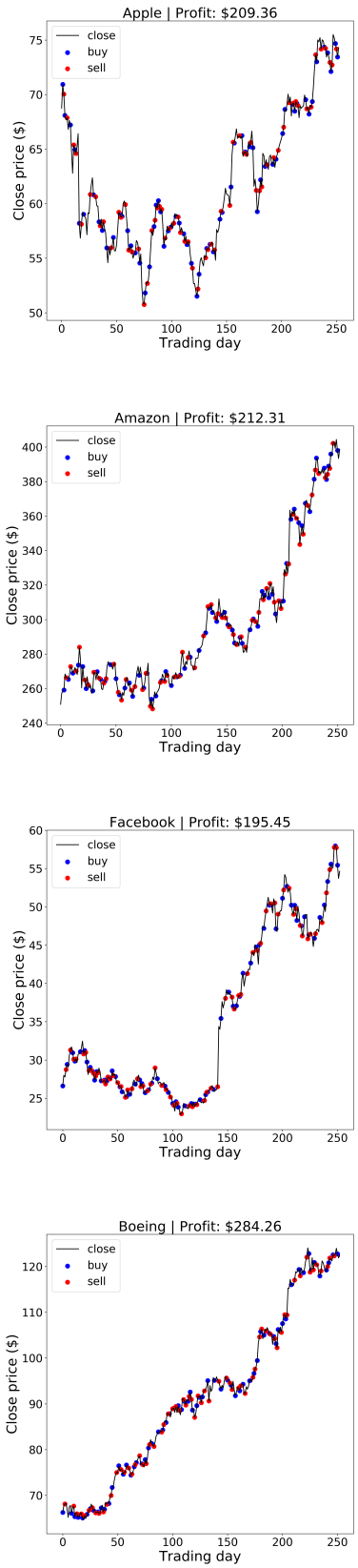


Figure 8. Baseline REINFORCE test results

tions for maximum reward, while REINFORCE optimizes neural networks to maximize reward using policy gradient method. Both models were trained on AAPL stock price history to learn trading strategies that can maximize profits. DQN was trained to maximize reward which is the difference between the buying and selling prices of an AAPL stock. DQN could generate about $202 on test set while being limited to buying or selling one stock per day and given an initial budget of $1,000. REINFORCE could generate about $700 on the test set, given an initial budget of $5,000, but it should be noted that the risk is much larger in this model as the agent spends all of its equity in every single trading step.

Both baseline models are able to generate profits in the test set. However, there is no measure in our models to indicate how confident we are about the new predictions. Each trial in both models results in different profits. In worse cases, we observed that DQN did not make any transactions throughout the entire test set because it thought it will lose money. Additionally, our baseline models do not demonstrate their ability to successfully profit on other correlated and uncorrelated stocks.

Moving forward, we first plan to fine tune our existing models to ensure we have reliable baselines to compare against. Convergence metrics will need to be established and presented to verify that our models are not just stochastically making decisions but learning actual policies. Our preliminary results tells us that is the case as we observe reasonable buy/sell points. Our baselines will then be used to predict on a variety of correlated and uncorrelated stocks.

We had previously proposed to incorporate RL as inference into this work with very little understanding of what that entailed. Once we become confident of our baseline results, we will incorporate soft Q-learning algorithms into our models to improve exploration and prevent entropy collapse that can possibly impede the training process. For the REINFORCE model, we will incorporate the entropy term into the objective function and compare the result with the baseline model.

References

Human-level control through deep reinforcement learning. doi: 10.1038/nature14236.

Amodei, D., Olah, C., Brain, G., Steinhardt, J., Christiano, P., Schulman, J., Dan, O., and Google Brain, M. Concrete Problems in AI Safety. Technical report.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483,

2009. doi: 10.1016/j.robot.2008.10.024. URL www.elsevier.com/locate/robot.

Frye, C. The dangers of reinforcement learning in the real world - Faculty : Faculty.

Frye, C. and Feige, I. Parenting: Safe reinforcement learning from human input. 02 2019.

Fu, J., Singh, A., Ghosh, D., Yang, L., and Levine, S. Variational Inverse Control with Events: A General Framework for Data-Driven Reward Definition. Technical report.

Levine, S. Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. Technical report, 2018.

Meng, T. L. and Khushi, M. Reinforcement Learning in Financial Markets. *Data*, 4(3):110, jul 2019. ISSN 2306-5729. doi: 10.3390/data4030110. URL https://www.mdpi.com/2306-5729/4/3/110.

Ng, A. Y., Ng, A. Y., and Russell, S. Algorithms for Inverse Reinforcement Learning. *IN PROC. 17TH INTERNATIONAL CONF. ON MACHINE LEARNING*, pp. 663—-670, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.7513.

Pendharkar, P. C. and Cusatis, P. Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, 103:1 – 13, 2018. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.02.032. URL http://www.sciencedirect.com/science/article/pii/S0957417418301209.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. Mastering the Game of Go with Deep Neural Networks and Tree Search. Technical report.

Singh, A., Yang, L., Hartikainen, K., Finn, C., and Levine, S. End-to-End Robotic Reinforcement Learning without Reward Engineering. apr 2019. URL http://arxiv.org/abs/1904.07854.

Xing, E., Ancha, S., Liu, L., and Kim, J. 10-708 PGM — Lecture 20: Reinforcement Learning & Control Through Inference in GM. URL https://www.cs.cmu.edu/{~}epxing/Class/10708-19/notes/lecture-20/.

Xiong, Z., Liu, X.-Y., Zhong, S., Yang, H., and Walid, A. Practical deep reinforcement learning approach for stock trading, 2018.