Department of Computer Science, Georgia Institute of Technology

QLearning Trader

Joseph Su

January 6, 2019

1 Introduction

This paper considers a strategy learner that employs a model-free, ϵ-greedy Q-learning to predict the optimal policy on a stock portfolio over time. We build and test the learner against multiple test cases to ascertain solution convergence, to measure the learner's performance.

Computing Facilities This work makes use of various libraries, toolsets, and modules in Python 2.4 to arrive at its studies and results: `numpy`, `IPython`, and `pandas`. All of the experiments are conducted on a 2015 Mac OSX 2.5 GHz Intel Core i7 with 16GB DDR3.

1.1 Q-Learning

Q-learning is a model-free technique in reinforcement learning that focuses on finding optimal policy, $\pi^* : S \rightarrow A$, in any Markov Decision Process (MDP), which is a discrete-time stochastic process involving taking an action $a \in A$ at any given state, $s \in S$, with a transition probability $T(s,a,s')$ of moving from s to the next state, s'. An immediate reward, $r(s,a,s')$ is observed. We assume a Markovian property that the effect of an action taken in s depends only on that state, not on its prior history [1]. Namely $T(s,a,s') = T(s,a)$. Also, the ordering preference of states are independent of when they occur, making rewards cumulative over the visited states. Rewards become less important in the distant future; we model such characteristic by discounting them with a discount factor that is $0 < \gamma < 1$:

$$U(s_0, s_1, s_2, \cdots) = r(s_0) + \gamma r(s_1) + \gamma^2 r(s_2) + \cdots \tag{1.1}$$

Q-learning is an alternative Temporal Difference (TD) method [2] in that a TD agent does not require $T(s'|s,a)$ in learning or selecting actions. The technique relates to utility by way of $U(s) = \max_a Q(s,a)$, with the following TD constraint:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{1.2}$$

where α is the learning rate.

2 METHODOLOGY

We apply Q-learning to obtain an optimal policy for each day based on adjusted closing price of the target stock. Model training involves assessing the following technical indicators:

- Bollinger Bands (BB)
- Relative Strength Index (RSI)
- Derivative of Daily Returns (DDR)

Taining windows of 20-day and 14-day are used for BB and RSI, respectively. Indicator values, X, are discretized into valued of 0 through 9 over the in-sample period. For each trading day, we convert the 3 indicator values into a unique state value for day i, $S[i]$, according to the following:

$$S[i] = X[:,0] * 25 + X[:,1] * 5 + X[:,2] \tag{2.1}$$

We initialize our holdings and cash portfolio with the first day's trading data. Reward is based on a modified daily return by shifting its y-intercept by a value of -10.0. This shifts the original daily return close to 0.0 into a more negative range, ameliorating the model's predicability for shorts over a market downtrend. Conversely, during market uptrend, we adjust reward with a 1.0 or more to amplify the magnitude of influence for longs. The convergence criteria are as follows:

- Running time: maximum clock time of 25 secs for each test case
- Number of consecutive (5 times) cumulative returns within 90% of all time max profit

For training, we apply Q-learning to arrive at a portfolio and record its cumulative return during the in-sample period. We repeat this process until either one of the above mentioned convergence criteria is met. The model takes one of the 3 actions daily: 200 shares of long, short, or nothing, while maintaining a maximum holding of 200 shares at all time. The starting cash is $100,000.

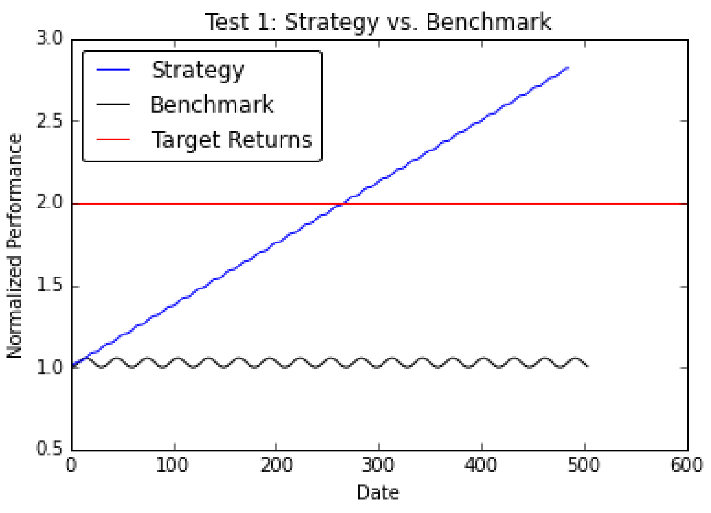
3 RESULTS

The results for the following test cases are as follows:

- For ML4T-220, the trained policy should provide a cumulative return greater than 100% in sample (Fig. 3.1).
- For ML4T-220, the trained policy should provide a cumulative return greater than 100% out of sample (Fig. 3.1).
- For AAPL, the trained policy should significantly outperform the benchmark in sample (Fig. 3.2).
- For SINE_FAST_NOISE, the trained policy should provide a cumulative return greater than 200% in sample (Fig. 3.2).
- For UNH, the trained policy should significantly outperform the benchmark in sample (Fig. 3.3).

4 DISCUSSIONS

EXPLORATION VS EXPLOITATION We observe tradeoffs between exploration and exploitation. ϵ-greediness is based on the random action decay rate ($radr$), defaulting to 0.99 at a starting random action rate of 50%. These values are used to encourage maximization of the model's reward by exploring its surrounding 50% of the time initially, decaying that randomness to 0% at a discount rate of 99% until the agent only exploits its surrounding all the time.



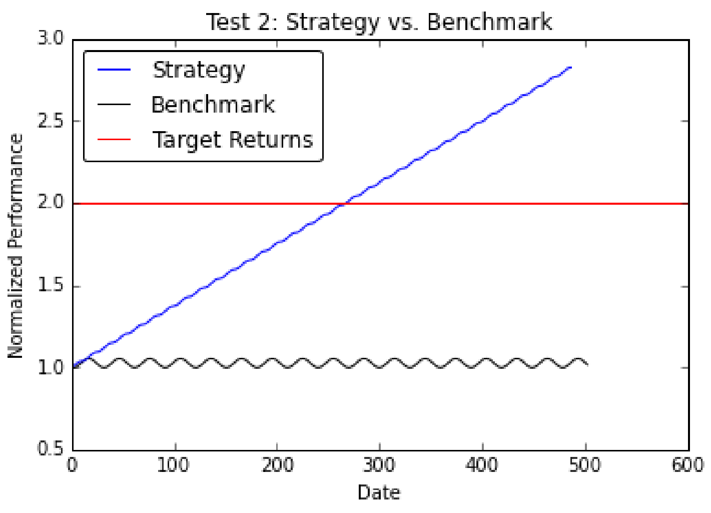
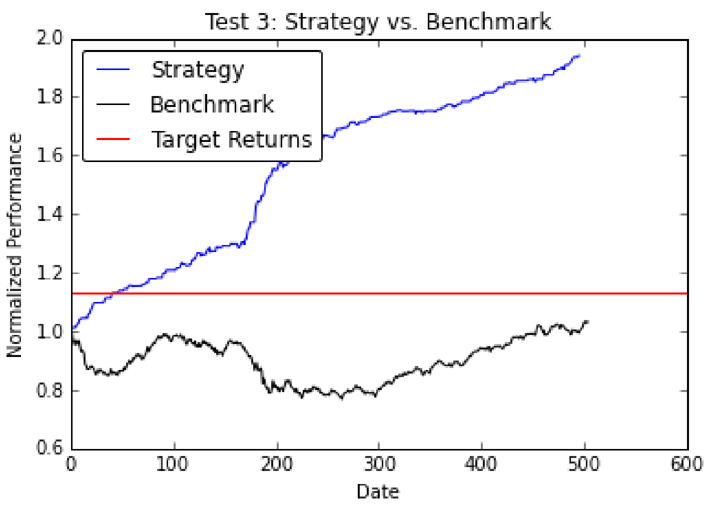


Figure 3.1: Test 1 (ML4T-220, in-sample) and Test 2 (ML4T-220, out-of-sample)



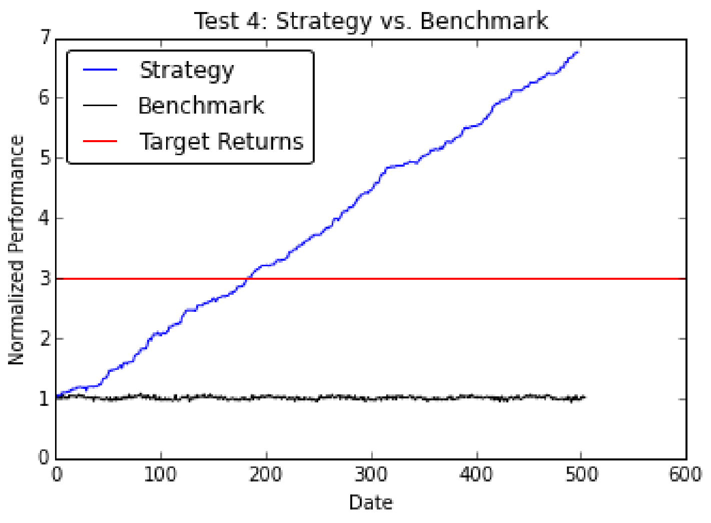


Figure 3.2: Test 3 (AAPL, in-sample, benchmark) and Test 4 (SINE_FAST_NOISE, in-sample)

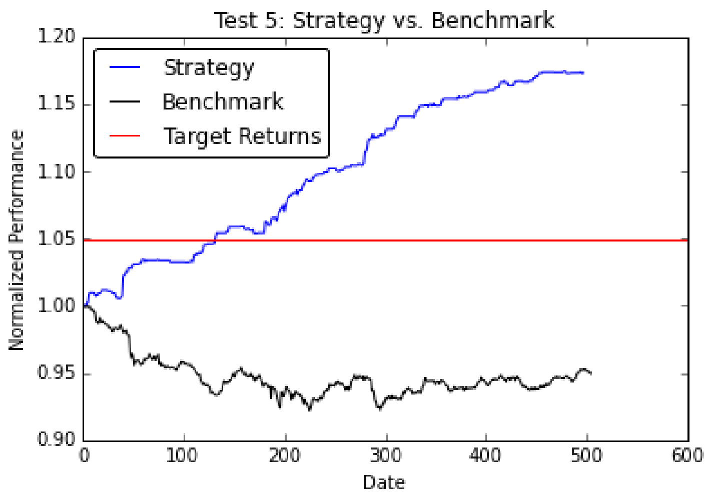


Figure 3.3: Test 5 (UNH, in-sample, benchmark)

CONVERGENCE CRITERIA The aforementioned convergence criteria yield, on average, 5,000 to 6,000 iterations for each test case. Note for future work, we will measure policy loss as a function of $|Q - Q'|^2$ where Q is the current $Q(s, a)$ and Q' the improved version, over all iterations:

$$Q\prime = r + \gamma max_a(Q[s', :]) \tag{4.1}$$

5 CONCLUSION

Q-learning thrives on learning in the absence of a model, paying no attention to any actual policy in its iteration. The model works extremely well for a sine model with fixed regularity. Our empirical study shows that such off-policy algorithm is slow and yields no long-term benefit to MDPs where the state space is big and there is no apparent controlled policy to guide its agent through. Therefore, for a large trading window, we are better off learning a model and a utility function with either policy or value iterations, rather than relying blindly on Q-learning to perform local updates with no consistency. For a smaller trading window, the Q-learner shows moderately good results.

REFERENCES

[1] Norvig, P and Russell, S. *Artificial Intelligence: A Modern Approach.* Third Edition. 1994.

[2] Sutton, R. Learning to predict by the methods of temporal differences. *Machine Learning 3.1*, 9-44.