Evan Knight and Finn Bergquist

Q Trader Report

Our Q trader works by creating discretized states given a handful of technical indicators and a number of shares held at that time. While there isn't any action the trader can take to influence prices or indicators in the future, the trader *can* influence its holdings day to day. Because the reward function used to calculate the utility of each state uses the number of shares held to calculate the reward, the trader is able to interact with and learn from its environment.

For our indicators, we used a two day rsi, a 14 day bollinger band percentage, and a MACD histogram (MACD - signal). To discretize our states, we bucketed each indicator into a number 0-4 and then added them together into a base 5 number. The value 5 was decided on because it created a state space less than the number of training data points we had, thereby avoiding having states we haven't encountered in the training data. When bucketing our indicators, we first divided all the data for each indicator over the training window into equally weighted fifths using the numpy.nanpercentile function. We did this to make our bucketing better represent the distribution of data.

The testing on our code is on the standard date range: 2018-01-01 to 2019-12-31, and we use the DIS stock. The trading costs are set to defaults of 0.005 and 9.95.

Experiment 1:

Baseline trading strategy(2018-01-1 to 2019-12-31):

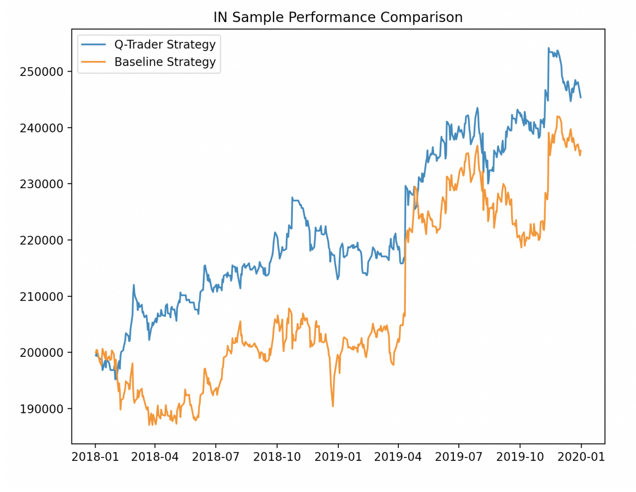
Sharpe Ratio: 0.7417836176554774

Volatility: 0.007662005123031563

Average Daily Return: 0.0003580299891180514

Cumulative Return: 0.1795

End value: **235900.0**



Here you can see that the q trader outperforms the baseline in-sample

In sample learned strategy(2018-01-1 to 2019-12-31):

Sharpe Ratio: 1.06707400368931

Volatility: 0.00634959345828907

Average Daily Return: 0.0004268155063680611

Cumulative Return: 0.22316174999999783

End value: **244632.34999999957**

In sample technical trading strategy(2018-01-1 to 2019-12-31):

Sharpe Ratio: 0.9133105857051127

Volatility: 0.02417766570289625

Average Daily Return: 0.0013910174860192108

Cumulative Return: 0.6619977500000003

End value: **332399.6**

These results came from passing the q-learner the exact same technical indicators that we based our trading strategy off of in the last project. These included bollinger bands, MACD, and RSI. Both the **learned and technical strategy outperform the baseline strategy**, but the **technical strategy vastly outperforms the learned strategy** which we tailored to this specific data. The trader performance fluctuates slightly(see graph below).

This does not surprise us because given our limitation of states due to the restriction of 500 data points, we were forced to bucket the technical indicators into relatively general groups(5 per indicator). In our technical strategy, we had these types of conditions :

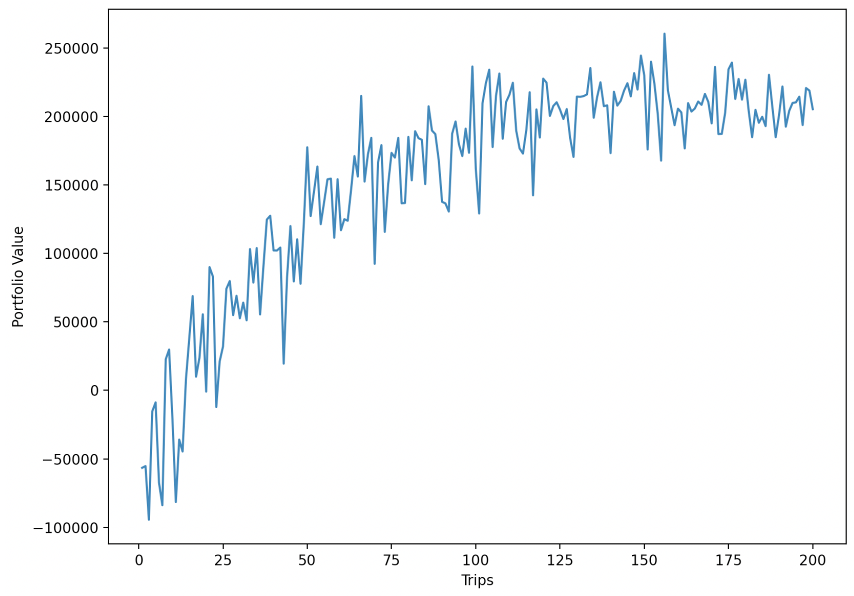
```
        (macd > signal) & (rsi < 15)] = 1
```

and

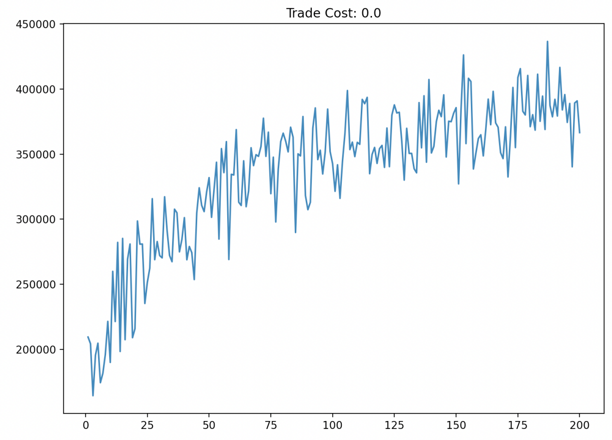
```
        (macd < signal) & (rsi > 85) & (bbp > 0.5)]
```

The q-trader could not make decisions about the technical indicators to the same level of precision as the technical strategy. We expected this to make the technical indicators almost useless. On the other hand, q-trader would improve more over time due to the number of trips it would perform and the versatility of our reward function. We perform 200 trips before the learner stops improving which takes quite a while, and the reward function combines short term rewards(daily return) with long term rewards(cumulative return from enter to exit date of a short or long position). Over time the learner could utilize short and and long term trends in the data to maximize the profit. A way we could improve the q-learner would be to improve the state representation so that it could account for smaller changes in the technical indicators.

Trading Strategy Performance over 200 Trips(default trading costs)



Interestingly, when both strategies have trade costs set to 0, the q-trader outperforms the technical strategy in-sample. The technical strategy does not change much because the trading decisions are not affected by the costs, but the q-traders decisions are affected by this cost. Here is how well the q-trader does with no trading costs. We explore this in the next experiment.

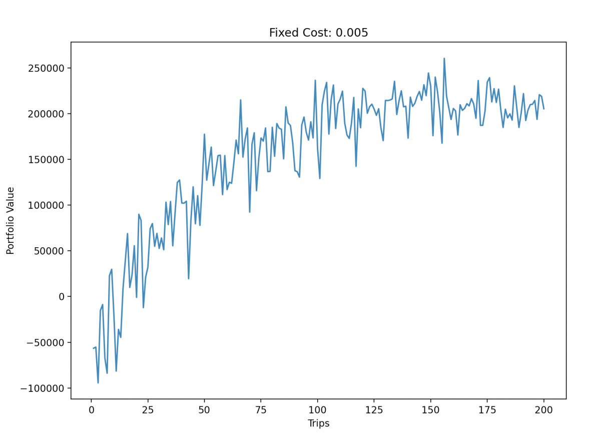


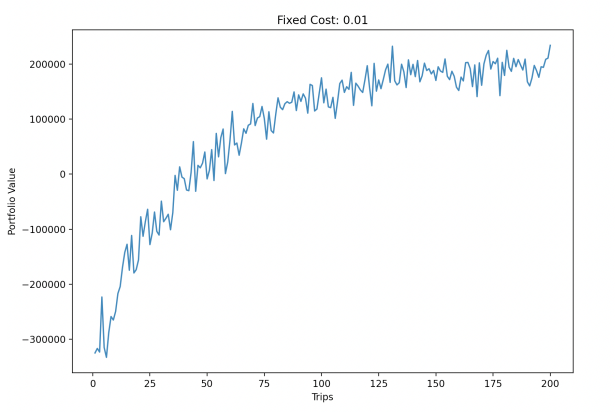
Experiment 2:

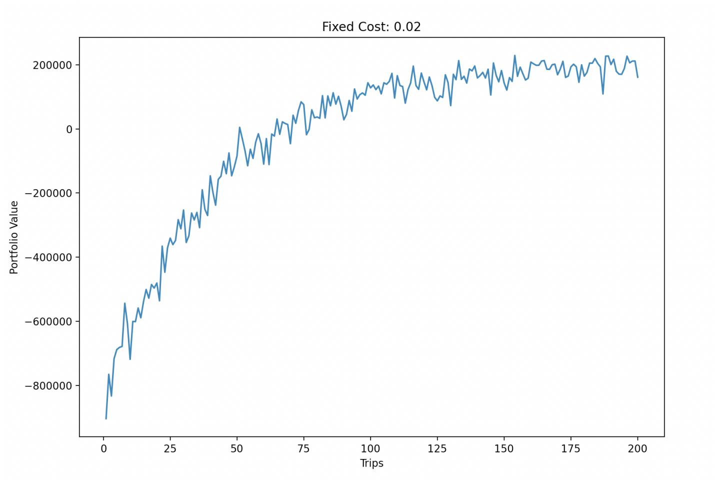
Hypothesis for how changing the floating cost value should affect the Q-Learner:

We expect that changing or increasing the floating cost will ultimately teach the Q learner to make fewer trades and likely marginally reduce profits.

Experiment: track the q-learner cumulative returns over 200 trips using the DIS stock at different floating cost values:







We observe that at a higher floating cost, the cumulative return of our learner does not improve as much over the 200 trips. This does not yet prove that the reason for this is related to the number of trades decreasing because it could continue to make as many trades and simply get charged more at higher trade cost rates. We decided to track the number of trades made at the different floating_costs as well.

0.005 and 9.95: NUMBER OF TRADES: 117

0.010 and 9.95: NUMBER OF TRADES: 41

0.020 and 9.95: NUMBER OF TRADES: 13

This confirms our hypothesis that higher floating cost values result in fewer trades made and a less profitable q-learner.

Out of Sample Results: 2020-01-01 to 2021-12-31

Q-Trader Out of Sample Results:

Sharpe Ratio: -0.9591718896989779

Volatility: 0.019262517812766895

Average Daily Return: -0.001163882733646994

Cumulative Return: -0.4948757500000044

End value: 101024.84999999912

Technical Trading Strategy Out of Sample Results:

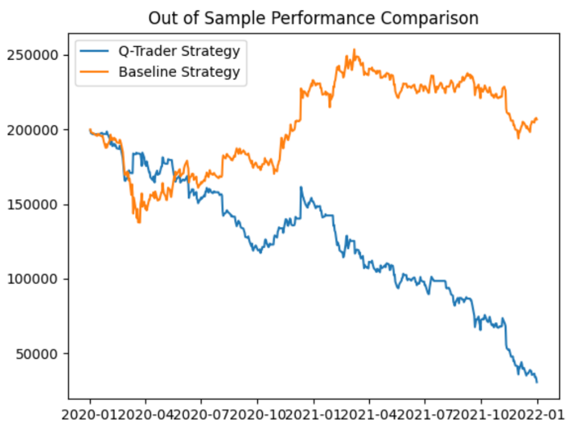
Sharpe Ratio: 0.7662005191958465

Volatility: 0.019189293798768232

Average Daily Return: 0.0009261922949277346

Cumulative Return: 0.38345249999999775

End value: 276690.49999999953



It isn't surprising the technical strategy out performs our q-trader, because it makes sense for our q-learner to be even more overfitted to the in-sample data, so we did not expect it to do very well out of sample. Here is a graph to show its performance when tested vs the baseline.