Machine Learning For Trading
Project 8 Strategy Trader Report
Jie Lyu
903329676

Why Q Learner

I chose to use Q learning for this project because I think trading is a perfect problem for reinforcement learning. We have 3 discrete actions: LONG, CASH and SHORT, and the indicators can be discretize to small world. Unlike chess problems, the reward is immediate and easy to compute. Dyna is also a reason I chose Q learning because I don't want the training process to be expensive.

Discretize the states

I implemented 3 indicators in Project 6: EMA (with a parameter for window), MACD and TSI. So in this project I used EMA 20, EMA 30, EMA 50, MACD and TSI. Because Q learning need discrete states, I discretized each indicator to output a 0 or 1. For example, if EMA > Price, it will give a 0; if MACD > Signal, it will give a 1. In total there are 96 permutations (2 for each of the five indicators, 3 for holding position), so in the Q learning world, there are 96 possible states. I kept the world (number of states) relatively small by binarizing each indicator instead of discretizing it to more than two states because I believe a small world will give better performance.

Determine the rewards

If the bot went short yesterday and the price rises the today, the reward will be negative in proportion to the amount of price risen. Vice versa. If the bot took CASH action, the reward will be 0 until it is takes LONG or SHORT.

Tune hyperparameters

I found a relatively low random action rate (0.9) and a higher random action decay rate (0.99) gives better performance. Dyna iteration is set to 100 to keep the early trainings relatively cheap. Learning rate and future reward discount rate is set to 0.2 and 0.9 for the best performance. Sometimes one configuration will give great result for one stock data and terrible result for another. In my model the parameters are set to be balanced and well performed for the five stocks I tested.

Experiment 1

Configuration

- QLearner parameters: num_states=96, num_actions = 3, alpha = 0.2, gamma = 0.9, rar = 0.9, radr = 0.99, dyna = 100, verbose=False
- impact = 0

Result

[Manual Trader]
Cumulative return: 0.2246999999999999
Stdev of daily returns: 0.01074570841256326
Mean of daily returns: 0.0004597904343384854 4

[Strategy Learner]
Cumulative return: 0.0024999999999999467
Stdev of daily returns: 0.014096792695730696
Mean of daily returns: 0.00010429369834201319

Interpretation

Manual trader outperformed Q learning bot with a higher cumulative return and lower standard deviation of daily return. In my opinions it is understandable. The manual trader is specifically tuned for the in sample period by hand, while the configuration is not set for the max performance for JPM for the in sample period. I also tested the same configuration and date on AAPL. Q learning bot significantly outperformed manual trader.

Experiment 2

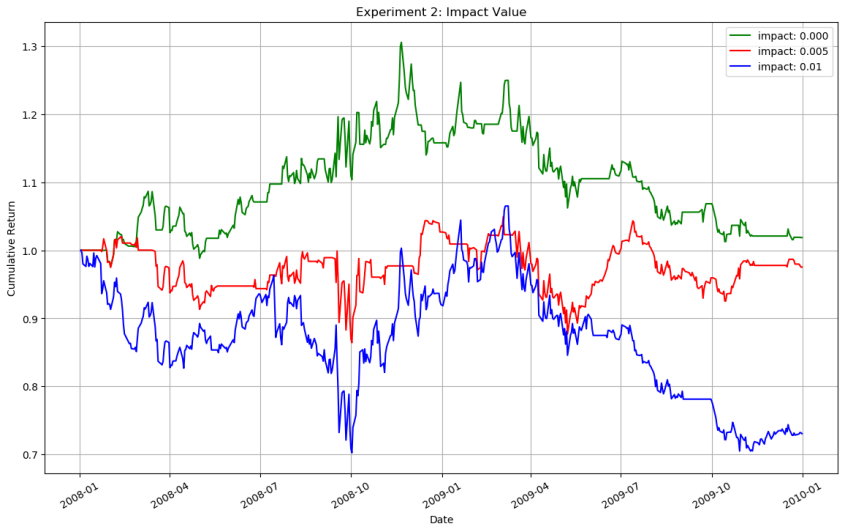
Configuration

- QLearner parameters: num_states=96, num_actions = 3, alpha = 0.2, gamma = 0.9, rar = 0.9, radr = 0.99, dyna = 100, verbose=False

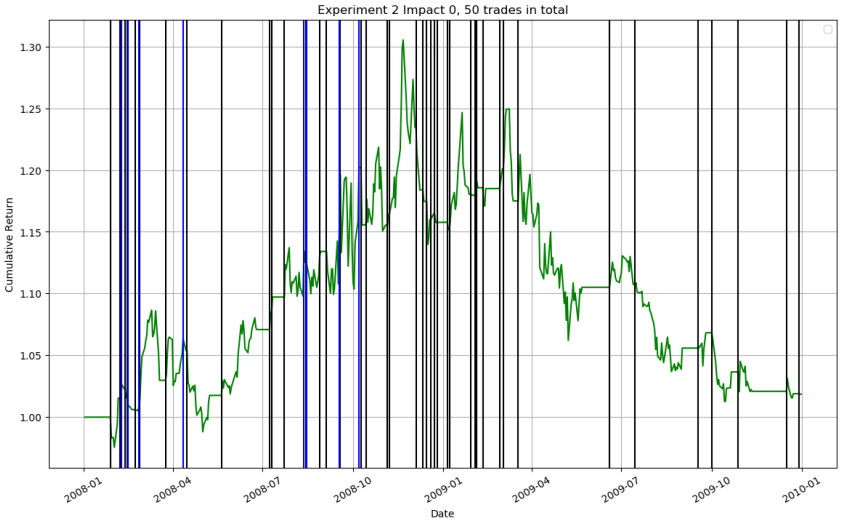
Hypothesis

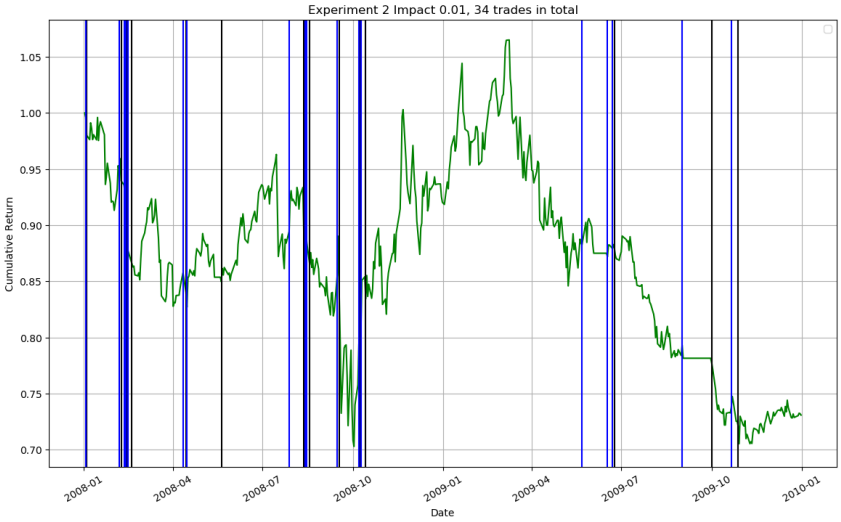
With an impact > 0, profitability will decrease and fewer trades will be executed.

Tests



As we can see from the figure above, with an impact of 0, Q learning bot achieved the best performance. As the impact increases from 0 to 0.005 and 0.01, the cumulative return drops. This supports the first point in the hypothesis.





In the two figures above, SHORT entries are plotted as black vertical lines and LONG entries are plotted as blue vertical lines. With an impact of 0, 50 SHORT or LONG actions are taken during the in sample period. With an impact value of 0.01, only 34 SHORT or LONG actions are taken. This supports the second point in the hypothesis.

Conclusion

The three figures combined prove the hypothesis right. With a big impact, the price floats up when the bot takes LONG and drops when it takes SHORT, making each trade less profitable. For the Q learning bot, the change is reflected on the reward it receives from an action.