Corey Caskey
CS 7646: ML for Trading
Strategy Learner

Overview

For this project, I chose to frame the trading problem as a reinforcement learning problem by using my Q Learner as the backbone structure of my Strategy Learner. The Q Learner relies on an agent that exists within a discrete set of states that we have defined, and this agent is allowed to perform a set of discrete actions at each state to move to another state. Each (state, action, new state) tuple is accompanied by a reward. To convert this to a finance-related problem, as in our Strategy Learner, we need to provide the Q Learner with discrete states defined by the financial indicators we used.

My Strategy Learner used the same three financial indicators that my Manual Strategy did: (1) Simple Moving Average with a 14-day lookback, (2) Bollinger Band % with a 14-day lookback, and (3) Moving Average Convergence Divergence with the required 12- and 26-day lookback periods. As a note of functionality, I wanted to avoid NaN values in the indicator dataframes. In order to do so, I accounted for the largest lookback period (26 days) and pushed the start date parameter back by 50 days. As a result, the lookback period was accounted for in these "droppable" dates — these dates were later removed from the prices dataframe after calculating the indicator values.

Since these indicators have continuous values and the Q Learner uses only discrete states, I needed to convert these indicators to discrete values as well. This required me to perform discretization on each of the three indicator dataframes in which I separated the range of values for that indicator into some *x* number of bins. For my implementation, I used the **qcut** method defined by numpy to split the values of each indicator into 10 bins — returning a new dataframe for each indicator where every date index was now paired with some discrete value between 0 and 9. As I mentioned, these indicators would indicate the specific state of the Q Learning agent; therefore, I concatenated the three numbers to form a three-digit number. For example, if the bins for **datetime(1, 2, 2008)** were 4, 0, 4 for Simple Moving Average, Bollinger Band %, and Moving Average Convergence Divergence, respectively, the resultant state would be 404, which would then be passed to the Q Learner.

This means that the agent has a total of 1,000 possible states that it could exist in (e.g. 000, 001, …, 998, 999). Also, because I'm using my Q Learner to determine the orders to execute on a specific stock, there are three possible actions that the agent can take at any state: BUY (long), SELL (short), or NOTHING (hold). These actions are returned by my Q Learner as integers of 0, 1, or 2. My implementation connected 0 to BUY, 1 to SELL, and 2 to NOTHING. The reward for

each action is the daily return for that date minus the net holdings at that date multiplied by the impact value. In my experiments, if the impact is 0.0, I pass the Strategy Learner an impact of 0.002 so that during the training phase the agent can take this into account when determining whether the impact is too high to trade. Otherwise, I provide the Strategy Learner the same impact passed in as a parameter.

In short, the agent starts at the first state defined by the indicators for the first trading day of the sample period. For each date in the data, we determine the daily return (modified with the impact as described above) and the new state from the indicators and pass those to the Q Learner as the reward and new state, respectively. After we get the returned action, we perform the corresponding action. This process is repeated until it has converged (i.e. where the previous iteration's trades dataframe is the same as the update one) or when the number of iterations exceeds 400.

In the test policy method, we are not worried about training and converging, so we just perform the date loop once to determine the best possible actions to take based on the model we've created during training. As mentioned above, I maintained the same three indicators for my Strategy Learner as my Manual Strategy.

Experiment 1

In this experiment, I focused on comparing the in-sample performance of both the Strategy Learner and the Manual Strategy by looking at the normalized portfolio values over the course of the date range of the in-sample period. Since the project guidelines marked commission as always being $0.00, the only assumption I made for this experiment was that the impact was 0.00, although I passed the Strategy Learner an impact value of 0.002 during the training phase. Therefore, the movement of the portfolio value over time is solely determined by the return on the buying and selling trades made. The parameter values for this experiment were the symbol of the stock, which is specified to be JPM for these experiments; the start date of the in-sample period; the end of the in-sample period; and impact value, which, again, was set to 0.00. The experiment captured the trade dataframes for both the Manual Strategy and Strategy Learner and then used the **compute_portvals()** method defined in marketsim to provide the data displayed below.

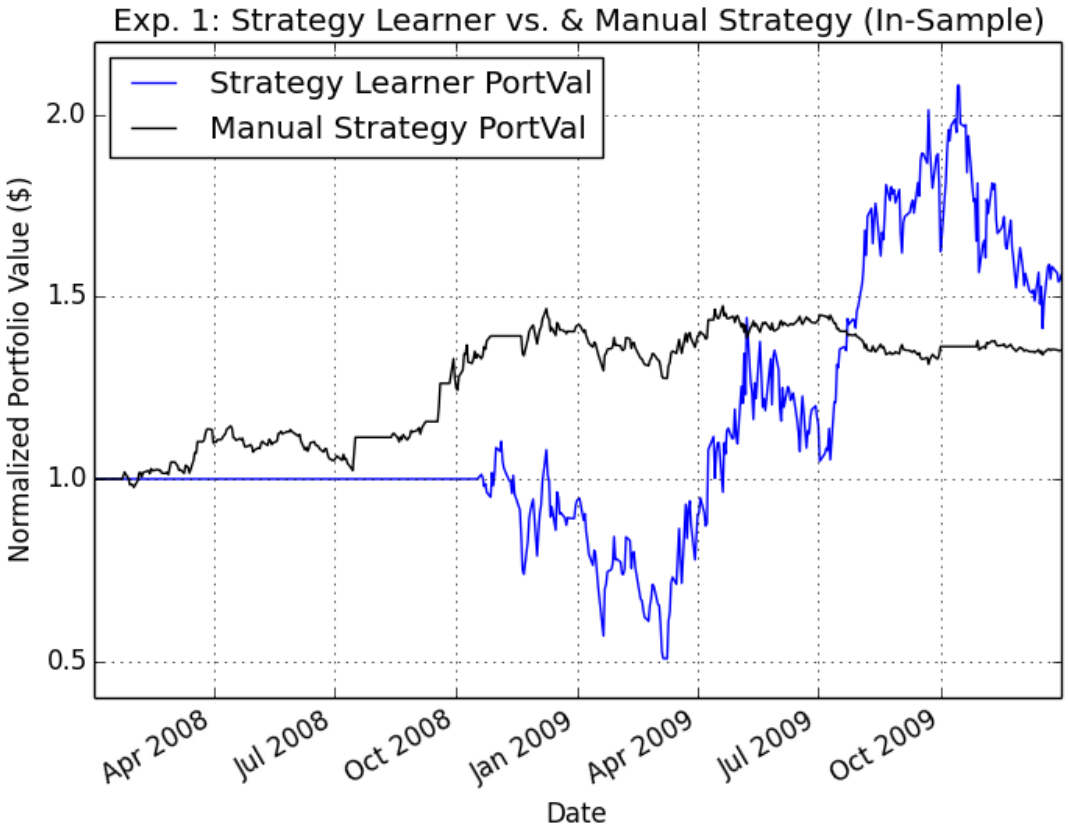

Figure 1: Comparing Strategy Learner and Manual Strategy In-Sample (Impact = 0.0)

As shown above in Figure 1, the Strategy Learner ultimately performed better than the Manual Strategy. This makes sense because the Strategy Learner is taking a "learning" approach in which it dynamically adjusts its actions based on the indicators whereas the Manual Strategy relies on

the same set of static conditions for the indicators at every date. As a result, the Strategy Learner seeks the most optimal action and creates a model for it while the Manual Strategy passively chooses an action from some met conditions for the indicators that provide a good estimate of future price fluctuation. It also makes sense for the Strategy Learner to perform well because it's testing on the same data it trained on, meaning it should fit the model fairly well — in fact, it should meet it perfectly because I set the random action rate (rar) of my Q Learner to 0.0.

The corresponding data from this experiment is:

	Manual Strategy	**Strategy Learner**
Cumulative Return	0.3537	0.5597
Standard Deviation	0.0111067005895	0.048717197397
Average Daily Return	0.000662280797922	0.0020381761447
Sharpe Ratio	0.946580098354	0.66414007724
Number of Trades	19	24

Experiment 1 also provided the following statistics shown in the table above. The two approaches are relatively similar in their quantitative statistics. Ultimately, the Strategy Learner provides a larger cumulative return and a larger daily return, the Manual Strategy has a smaller standard deviation and a higher Sharpe Ratio. The green highlights indicate the better/larger value between the two columns, depending on the statistic present. The overall takeaway from these statistics also makes sense. The conditions set for the Manual Strategy are static and are not trying to change over time to learn and get a better outcome. These means that the gains and losses will likely be more averaged out and stable over time than the Strategy Learner, which, on the other hand, is taking a learning approach and may end up performing actions that are suboptimal.

I would expect the same type of outcome every time for the in-sample data. For Manual Strategy, there is no learning/training aspect. It simply computes BUY and SELL orders based on the same static conditions for the three indicators. Since these don't change each time the method is called, neither will the portfolio values. For the Strategy Learner, while there is a learning aspect, I set the random action rate (rar) to 0.0, as mentioned above. This prevents the likelihood of the Q Learning agent from choosing a random action rather than the one that is most optimal. As a result, the same actions will be taken by the learner every time.

Experiment 2

In this experiment, I created two hypotheses, as outlined below. In order to test these hypotheses, I manipulated the impact level from 0.002 to 0.01 in increments of 0.002. I excluded the impact of 0.0 from this experiment because the values are easily transferable from the previous table. I subsequently passed the impact value into the Strategy Learner for the Q Learner to use in its training phase. Again, the project guidelines marked commission as always being $0.00. Since the impact is greater than 0.0 in each trial, the movement of the portfolio value over time is determined by the return on the buying and selling trades made in addition to the impact value. The parameter values for this experiment were the symbol of the stock, which is specified to be JPM for these experiments; the start date of the in-sample period; the end of the in-sample period; and impact value, which was set to 0.00 by default.

Hypothesis 1: As impact increases, I expect a lower cumulative return for the trading strategies.

Since the impact is calculated in marketsim while computing the portfolio values and adds to the BUY cost or subtracts from the SELL payout, the return for each order will be lower. This obviously correlates to a lower portfolio value overall.

Hypothesis 2: As impact increases, I expect the same number of trades to be made.

With impact being used in the training phase of the Q Learner, this value has an effect on the reward of each action. While it affects the decisions of the Q Learner, it doesn't necessarily reduce the number of trades the Q Learner will make. A lower reward means there's less motivation for the agent to go to a particular state but, again, it doesn't necessarily correlate to fewer trades made overall.

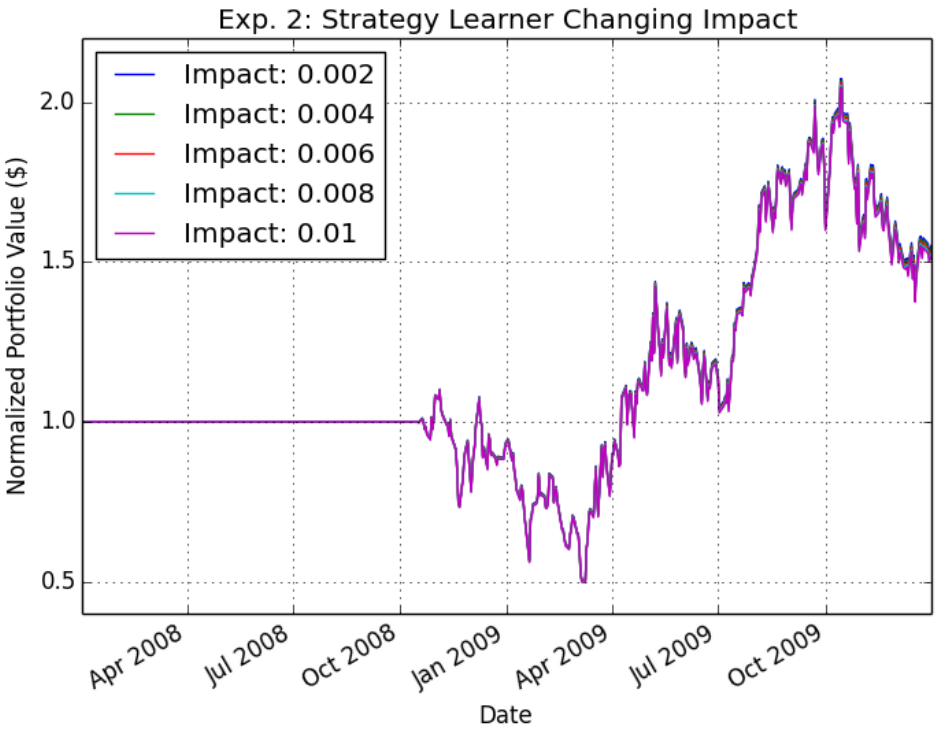

Figure 2: Comparing In-Sample Portfolio Values Across Impact Values

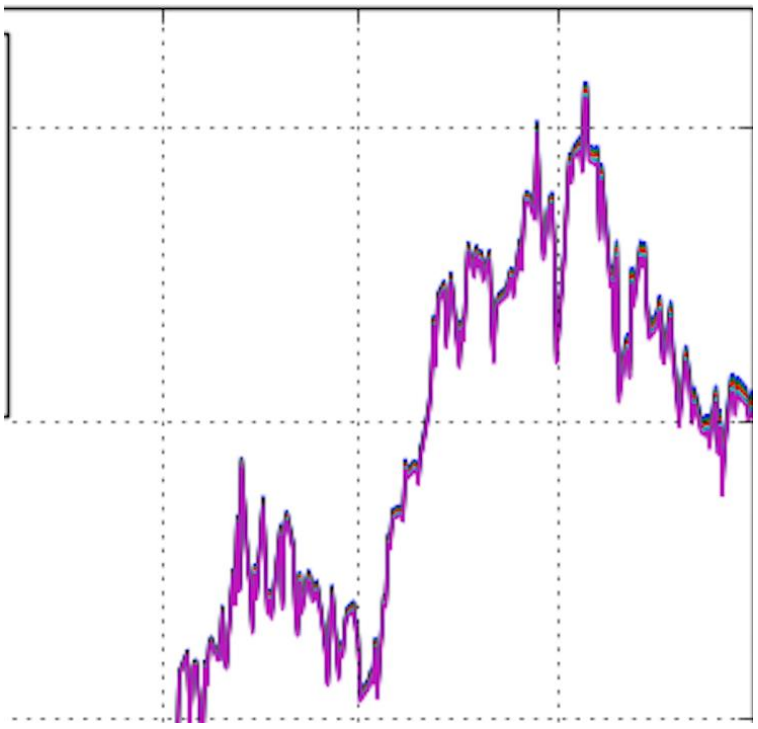

Figure 3: Zoom-In on Portfolio Difference Between Impacts

The corresponding data from this experiment is:

	Impact = 0.00	**Impact = 0.002**	**Impact = 0.004**
Cumulative Return	0.5597	0.5512914	0.5428828
Standard Deviation	0.048717197397	0.0488613331195	0.0490065375208
Average Daily Return	0.0020381761447	0.00203421335772	0.00203026215067
Sharpe Ratio	0.66414007724	0.660893469077	0.657655368286
Number of Trades	24	24	23

	Impact = 0.006	**Impact = 0.008**	**Impact = 0.01**
Cumulative Return	0.5344742	0.5260656	0.517657
Standard Deviation	0.0491528206381	0.0493001926563	0.0494486639103
Average Daily Return	0.00202632279947	0.00202239558525	0.00201848079447
Sharpe Ratio	0.654425866155	0.651205054124	0.66414007724
Number of Trades	24	24	24

These tables clearly depict that as the impact value increases, the overall cumulative return decreases, supporting my first hypothesis. Additionally, the change in impact had no effect on the number of trades made by the Strategy Learner, supporting my second hypothesis. Figure 3 is a screenshot taken of Figure 2 to allow you to better see the minute differences in the portfolio value lines in which each one is shifted down further as the impact increases.