Strategy Learner

Strategy Learner Implementation.

The strategy learner is implemented with a Q-learning-based strategy using the Q-Learner that I have implemented for the Q-learning robot project. The strategy learner class has two methods, the addEvidence method and testPolicy method.

In the constructer of the strategy learner, the Q-learner will be initiated with 800 number of states, 3 actions, alpha = 0.2, gamma = 0.9, rar =0.5, radr = 0.95 and dyna = 0. The Q-learner will be trained in addEvidence method and the Q-table (with 800 states and 3 actions) will be used to predict trading actions in the testPolicy method.

The trading problem is framed in the addEvidence method. The same three indicators as manual strategy are used for this strategy learner. They are the ratio of price and simple moving average (p/SMA), the Bollinger Band® (BB indicator = $\frac{price-SMA}{2*std}$), and the stochastic oscillator ($\%K = 100 * \frac{C-L}{H-L}$). By using the pandas.cut() function, the p/SMA indicator is discretized to 8 bins with digital label (0,1,2,…,7), the BB indicator is discretized to 10 bins (0,1,2,…,9), the stochastic oscillator K is also discretized to 10 bins (0,1,2,…,9). Then, the discretized indicators are combined as 100 * p/SMA_bins + 10 * K_bins + BB_bins to form the 800 states, which is used to construct the Q-table. Since the indicators are discretized into digital number 0 to 9, there is no need for further standardization. I note that the pandas.cut() function defines the number of equal-width bins in the range of indicator values rather than bins with equal number of samples. The reason that I implement in this way is the values of the indicators close to critical point like 1.0 is more meaningful to trigger an action.

The main structure of the strategy learner in the addEvidence method is as follows:

```
Load the price table of the stock symbol.
Compute the indicators using the prices.
Discretize the indicators and combine them to form states.
While not converged and count < max_iterations:
    Instantiate the Q-learner: action = self.learner.querysetstate(states[0])
    Based on this initial action, calculate the holding state on the first day.
    For each day (from the second day) in the training data:
        Compute the reward for the last action.
        Query the learner with the current state and reward to get an action
        Use the action that the learner returned (LONG, CASH, SHORT) and the current
        holding state to compute the trade amount.
            if (action == 0): #long
                if netHolding == -1000:
                    df_trades.loc[day] = 2000
                elif netHolding == 0:
                    df_trades.loc[day] = 1000
            elif (action == 2): #Short
                if netHolding == 1000:
                    df_trades.loc[day] = -2000
                elif netHolding == 0:
                    df_trades.loc[day] = -1000
        Update the holding state.
    Update portfolio value using the trades dataframe.
    Check whether the convergence condition reached. If true, stop the while loop; otherwise,
    continue training, increment count.
```

The reward is defined as the daily return taking the impact into account:

*if holding > 0: Reward = (daily_return – impact) * holding*

*if holding <= 0: Reward = (daily_return + impact) * holding*

The actions are represented by integers 0, 1, 2, where 0 represents LONG, 2 represents SHORT and 1 represents CASH.

The convergence condition is defined as: the difference between the portfolio value of the current day and the average portfolio value of the 10 previous days is smaller than 50 dollars, and the count number of the while loop is at least 25 times (the learning at least runs 25 times). The maximum allowed loop number is 500, but never reached in my experiments.

Experiment 1.

Experiment details.

The same three indicators (p/SMA, BB and stochastic oscillator K) are used for strategy learner and manual strategy. The strategy learner is trained and tested on stock "JPM" with starting cash at $100,000 using the prices and indicators in the in-sample period from January 1, 2008 to December 31 2009. The commission and impact are all set to 0.0. The same parameters are used for manual strategy and benchmark for comparison.

The parameters to construct the Q-learner of the strategy learner are discussed in the ***Strategy Learner Implementation*** part, without additional changes.

Since the initial actions of the Q-learning is partially random selected, the learning results are not the same from time to time. Therefore, to report the meaningful performance of the strategy learner, I run the experiment 7 times, and find the experiment whose cumulative return is the median of the 7 cumulative returns. The daily portfolio value and other metric from this median performance experiment are shown in Figure 1 and Table 1.

Results of Experiment 1.

As shown in Figure 1, the portfolio value (red curve) of strategy learner is compared with the portfolio values of the manual strategy (black) and the benchmark (blue) for the in-sample period. The strategy learner has significantly higher performance than the manual strategy and benchmark. The portfolio value keeps increasing with the strategy learning except for small vibrations. Since the strategy learner does not have big decreases, its volatility is also relatively smaller than that of manual strategy and benchmark. The final portfolio value, cumulative return, standard deviation and mean of daily return are summarized in Table 1, where we can see the cumulative return of strategy learner is about three times of that of manual strategy.

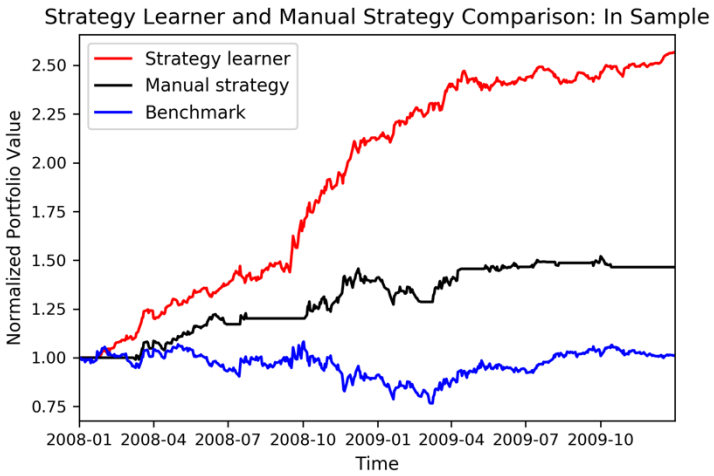

Figure 1

Table 1. The in-sample performance of the strategy learner, manual strategy and benchmark on "JPM".

	Start value	Final value	Cumulative return	Stdev of daily returns	Mean of daily returns
Benchmark	100000	101230	0.0123	0.017004	0.000168
Manual strategy	100000	146450	0.4645	0.009222	0.000800
Strategy learner	100000	256730	1.5673	0.008995	0.001913

Discussion about experiment 1: Would you expect this relative result every time with in-sample data?

The relative result will be different in different experiment. The reason is that in Q-learner the process to generate action is partially random, as shown in the code snippet below. Especially when the learner is initialized the rar value is larger, the higher chance that the action is random.

```
if rand.uniform(0.0, 1.0) <= rar:
    action = rand.randint(0, num_actions-1)
else:
    action = np.argmax(Q_table[s_prime])
rar = rar * radr
```

Therefore, the learning route will be different from experiment to experiment and the resulted Q-table is different from time to time. We thus get different portfolio value in different experiments for the strategy learner. However, the manual strategy does not have any random process implemented, so we get constant results from manual strategy. Therefore, the relative result between the two strategies are different every time.

Experiment 2.

Hypothesis.

As the value of impact factor increases, the in-sample trading frequency and the cumulative return of the portfolio will decrease.

Experiment details.

The same three indicators (p/SMA, BB and stochastic oscillator K) are used for strategy learner. The strategy learner is trained and tested on stock "JPM" with starting cash at $100,000 using the prices and indicators in the in-sample period from January 1, 2008 to December 31 2009. The commission is set to 0.0. The impact values 0, 0.002, 0.004, 0.006, 0.008, 0.01, 0.02, 0.05, 0.1, 0.3 and 0.6 have been used for the experiment.

The parameters to construct the Q-learner of the strategy learner are discussed in the ***Strategy Learner Implementation*** part, without additional changes.

Results of Experiment 2.

The daily portfolio values of in-sample period with different impact values are shown in Figure 2, from which we can see that the impact value affect the full range of the learner performance. When the impact value close to 0.01, the final portfolio value is barely higher than the starting portfolio value.

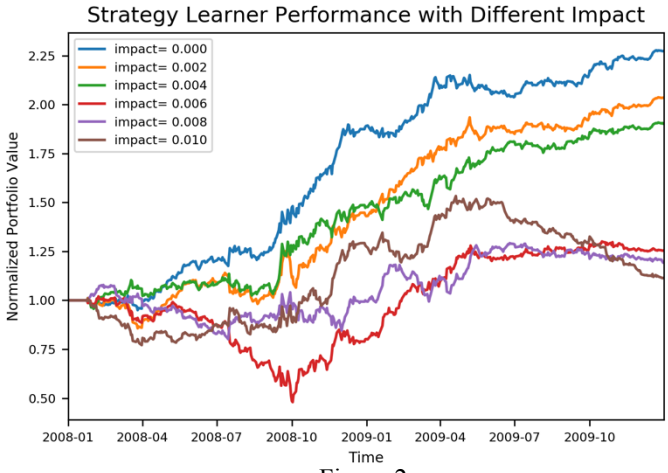

Figure 2

The number of trades at different impact values is shown in Figure 3. As the impact value increases, the number of trades decreases dramatically. The only outlier is impact = 0.01, with which the strategy learner trades for 166 times greater than the trading times with impact = 0.004, 0.006 and 0.008. The reason is that: the goal of the strategy learner is to maximize reward, which does not necessarily mean that the number of trading times has to be smaller with higher impact value. It is possible that with impact = 0.01, the reward value of each day is optimal with 166 trades rather than other trade times. Even though it is possible to have special cases like impact = 0.01, the general trend of the number of trades with increasing impact value should be decreasing. Because when the impact value becomes large, the profit from trading cannot compensate the cost of impact, in this situation the strategy learner will try to avoid trading. As shown in Figure 2 and Table 2, when impact = 0.3, only 2 trades happened; when impact = 0.6, no trades happened.

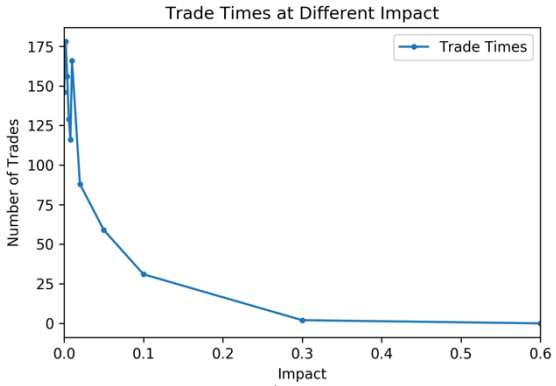

Figure 3

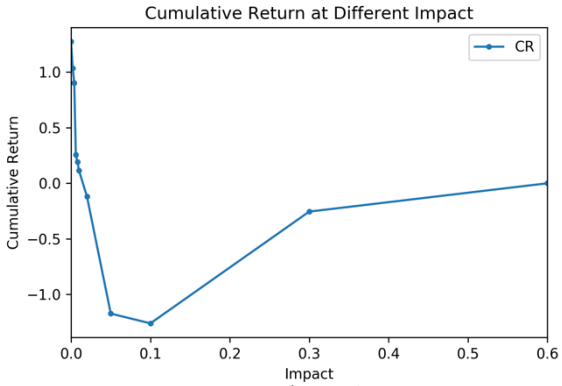

Figure 4

The cumulative return with different impact is shown in Figure 4. When impact < 0.1, the cumulative return keeps decreasing as the impact value increases. With impact = 0.05 and 0.1, the cumulative return even become smaller than -1.0. When impact > 0.1, the cumulative return increases to around 0. This result is consistent with the trading behavior with increasing impact values. When the impact values are small, the learner tends to trade frequently to gain profit. When the impact values are too big, the learner tend to stop trading to avoid the expensive trading cost. However, in some magnitude scale of the impact values, these values are harmful for the overall performance, but not large enough for the learner to know this adverse effect when it computes the reward based on daily returns. For this experiment on "JPM" in-sample period, the impact values between 0.02 to 0.3 are such cases. Though the number of trades with these impact values is much less than the number of trades at smaller impact values, but these trades still make the final portfolio value decrease.

Moreover, the standard deviations of daily returns with impact = 0.05 and 0.1 are also very large (0.53420 and 3.62204, respectively), as shown in Table 2. This high volatility indicates the strategy learner performs worst with impact values in this range.

Table 2. The in-sample performance of the strategy learner on "JPM" at different impact values.

Impact	Trade times	Final portvals	Cumulative return	Mean of daily returns	Stdev of daily returns
0	146	227430	1.2743	0.00169	0.01043
0.002	178	203592.26	1.0359	0.00149	0.01229
0.004	156	190329.16	0.9033	0.00135	0.01178
0.006	129	125508.66	0.2551	0.00066	0.02053
0.008	116	119346.72	0.1935	0.00047	0.01548
0.01	166	111495.1	0.1150	0.00033	0.01526
0.02	88	88143.6	-0.1186	-0.00011	0.01658
0.05	59	-17185.5	-1.1719	0.03597	0.53420
0.1	31	-25953	-1.2595	-0.22893	3.62204
0.3	2	74543	-0.2546	-0.00026	0.02485
0.6	0	100000	0.0000	0.00000	0.00000