USE BANDIT FOR STOCK SELECTION

Weitao CHEN, Lingfang XU

HKUST MAFM 2021 Intake

1. PROBLEM DEFINITION

Our goal is to find the optimal hedge position for one stock under the assumption of binomial stock price model.

With binomial stock price models, the assumptions are that there are two possible outcomes. With a pricing model, the two outcomes are a move up, or a move down. The major advantage of a binomial option pricing model is that they're mathematically simple. And we assume that the hedge action we take will not affect the stock price process.

We propose to use Q-learning to learn the best action positions between 0, 0.5, 1. Q-learning does not require a model of the environment. However under this circumstances, we have full knowledge about the environment which is the stock price process. The algorithm is expected to select the best action very quickly after some exploration.

2. MODEL IMPLEMENTATION

2.1. Binomial Model

We constructed a binomial stock price model of 15 steps. Parameter set is shown below: volatility σ 0.007, riskfree interest rate r 0.002. For state price s, the upper price u can be calculated as $e^{\sigma} * s$, the lower price d can be calculated as $e^{-\sigma} * s$, the growth a can be calculated as $e^{r} * s$. Hence, we can calculate the probability of going up as 0.6412 and the probability of going down as 0.3588 using the formula below.

$$prob\_up = \frac{a-d}{u-d}, prob\_down = 1 - prob\_up$$

2.2. Q-learning Algorithm

For basic elements of Q-learning algorithm, we define current stock price s as state, the hedge position $[0, 0.5, 1]$ as actions, the expected vale of the opposite absolute value of the price difference between now and then, $E[-abs(s_t - s_{t-1}) * (1 - H)]$, as rewards. Here, H denotes the hedge position and we use the notion of expectation since at every state, except the last state, we have two possible future state. And we use the negative absolute value since, whether we receive profits or receive losses, it's opposite to our goal of perfectly hedging.

After defining basic elements, for the learning process, we propose to use Epsilon-Greedy algorithm as the basic learning algorithm where $\epsilon = 0.1$ and learning rate equals to 0.1. We divide the whole learning process into different episodes. For each episode, we generate a stock price path and make our Epsilon-Greedy model learning the path and update our estimation of Q. At each step within one episode, we update our estimation of Q according to the formula below.

$$Q(S, a) = (1 - \alpha) * Q(S, a) + \alpha * (Reward + Q(S', a'))$$

where α is the learning rate, S' is the next state, a' is the action of next state chosen by Epsilon-Greedy formula.

2.3. Performance Measure

For each episode, after we updated the Q-function using the previous episode, we roll over the episode again by greedy formula. At the same time, we record the number of each action we take and the risk we faced during this episode. Risk is calculated by formula: $abs(s_t - s_{t-1}) * (1 - H)$.

3. CONCLUSION AND PROBLEM

3.1. Rough Conclusion

For the experiment, we ran 50000 episodes and in each episode, we ran through 15 states (steps). Since our choice of reward is negative which makes it more like 'punishment', for states that are not fully explored (reached less than 3 times), our algorithm tends to choose actions that have not been tried.

Below is graph of the cumulative risk which is expected to flatten since the best risk one action can get is zero under this circumstance.

As the plot shows, the cumulative risk converges after running 2000 episodes approximately which proves the convergence of this Q-learning model. To see which actions the model took at different episodes, we draw a picture showing the number of each action taken in all episodes.

We can tell from the plot shown above that, after roughly 2000 episodes, the algorithm would always choose to short 1 stock in order to avoid risks.

The final updated Q function of different state is stored in Q.csv file. The final updated Q function also shows for those states that are fully explored, the Q function of Action 3(short 1) is the biggest. So we should always choose to short 1 in order to achieve perfect hedging.

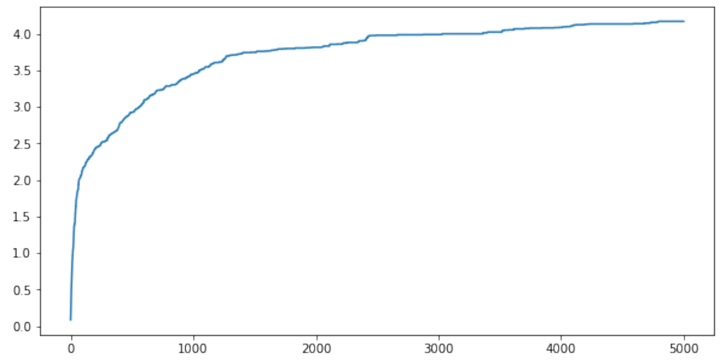

Fig. 1. The cumulative risk

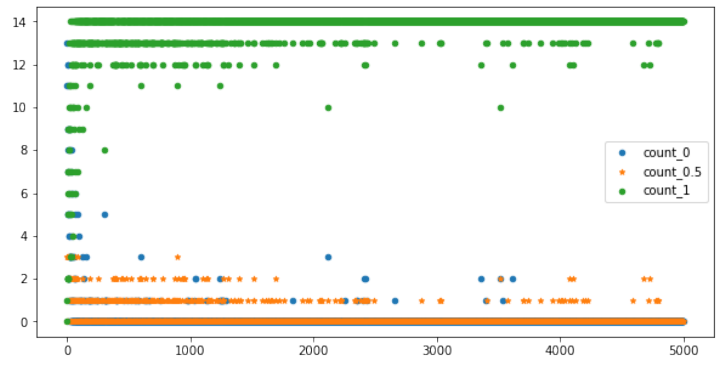

Fig. 2. The cumulative risk

3.2. Problems and Potential Improvements

One problem is that even we learn 50000 episodes which is a relatively large number for home-used computer and this problem, we didn't reach lot's of the state since the probability of some of the states appearing is so small. It's hard to estimate this in a stochastic and 'Monte Carlo' way. One way to solve this problem is that we should try more deterministic methods such as Sarsa, since we have full knowledge of the whole environment. Another way is that we could use sampling techniques to elevate the probability of reaching those rare events, like importance sampling.

Another problem is that, in our previous examples, we mostly use positive functions as rewards. This time, theoretically, the negative function increased our chance of trying new actions and make our performance evaluation methods different. It may causes other problems that we didn't notice.

4. GROUP DIVISION

Weitao CHEN: Model formulation and writing report.

Lingfang XU: Coding and model formulation.

5. CODES

Codes are in ipynb file in repo below. Github address: Github Repo.