|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | | Algorithmic trading is a subject undergoing intense study in machine learning. Collated to other methods, Reinforcement Learning (RL), peculiarly Qlearning, can learn decision rules directly with reasonable reward, and therefore is suitable for learning trading strategies. Recently, Q-learning based on deep neural models, also known as Deep Q-learning, has been successfully applied to some challenging tasks like game playing and robot motion. In this project, we propose to employ Deep Q-learning to build an Algorithmic Cryptocurrency Trading system which can automatically determine what position to hold at each trading time. | | |  | | --- | | Methodology   1. Retrieve the feature vector that defines the state, i.e the delta price zt - zt\_l of the bitcoin price chart. 2. Use the artificial neural network to estimate the Q values for each action. The output is the Target Q vector. 3. Generate a number randomly between 0 and 1. If the number is greater or equal to the Epsilon, the exploration factor, choose an action randomly. If the number is smaller than the Epsilon, choose the action with the largest Q value from the Target Q vector. 4. Apply the action from the previous step to the portfolio and retrieve the reward. 5. Move one step forward through the environment. 6. Retrieve the feature vector that defines the state. 7. Estimate the Q values from the feature vector from the previous step. 8. Apply a future value discount to the maximum Q value from the previous step and add it to the reward. 9. Update the Q value that corresponds to the action taken from the Target Q vector with the reward from previous step. 10. Train the deep neural network with the new Target Q vector and State incrementally. 11. Decrease the Epsilon by a decay function and repeat these steps till convergence. | | |  | | --- | | Results  Epoch:  1  Epoch:  10  800  After one and ten epoch respectively, this was the result of the system suggesting trades at each time step. Observe that the system initially executed random actions according to the exploration and exploitation strategy.  Epoch: 20 Epoch: 30  800  400  Time  With experience, the system learned to suggest profitable actions resulting in good trades. | |
| |  | | --- | | Deep Q-Learning Algorithm    Algorithm 1 Deep Q-learning with Experience Replay    Initialize replay memory D to capacity N  Initialize action-value function Q with random weights for episode 1, M do  Initialise sequence Sl = {ml } and preprocessed sequenced 41 = ) for t = 1, T do  With probability select a random action at otherwise select at = maxa Q\* a; O)  Execute action at in emulator and observe reward rt and image a;t+l  Set st+l st, at, :rt+l and preprocess (þt+l = (þ(st+l)  Store transition (Ot, at, rt, OH-I ) in D  Sample random minibatch of transitions , rj , ) from D  for terminal  Set yj — rj + max-a' , a'; O) for non-terminal cþj+l  Perform a gradient descent step on (yj — Q(4j, aj; O))2 according to equation 3 end for end for | |
| |  | | --- | | Implementation Overview & Model Evaluation | | We considered a simple trading task that operates on a single security, and at each trading day t, only one action was allowed. The action at had three options: hold, buy, or sell, and a reward rt was obtained. Our task was to learn a deep Q-function Q(s, a) that maximized the long-term accumulated profit -t q, No transaction cost was considered in this project.  The environment and was coded from scratch by mimicking the OpenAl Gym infrastructure. The Deep Q-Learning agent consisted of the ANN written in Keras. Pandas and Numpy was used for data preprocessing. Data visualization was done by the matplotlib library. The model was evaluated based on parameters like loss and reward generated during training.  Model Loss Model Reward   |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | |  | |  |  | | --- | --- | |  | LOSS | |  | |  | |  | |  |   Epoch Epoch  Figure 1. Model Loss vs number of epochs Figure 2. Model Reward vs number of epochs | |
| |  | | --- | | Architecture of the model    The architecture of the Deep Q-network involved four layers in total (two are hidden), with the number of units set to 90, 100, 100 and 3 respectively. Input: The input units (features) were composed by the delta price zt - zt\_l of the bitcoin price chart.  Output: The output units correspond to the three actions in trading namely:  Hold, Buy and Sell.  The learning rate for Q-network was 10-3, and the training stopped after 30 iterations. The interpolation factor was set to 0.125, discount factor was set to 0.95 and the minibatch involved training examples of the past 32 days. | |
| |  | | --- | | Conclusions & Future Work  The Accumulated Return was 13.548 0/0.  From the positions held by the system, it seems that it has learned how to take different actions in different market situations. Compared to existing methods, deep Q-Trading is able to detect market status from raw and noisy data, and pays attention to long-term returns.  Despite these interesting results, the project is still in a preliminary stage. In future work, I will investigate the contributions of other features derived from financial research and take care of other aspects like position sizing and latency. | |
| |  | | --- | | Acknowledgement  This research was supported by Academia Sinica and the National Chengchi University. I would further like to thank Dr. Yuh-Jong Hu who provided me with cutting edge technology and the best resources available. His insight and expertise greatly assisted the research and made the experience worthwhile. | |

References

|  |
| --- |
| Intern's name: Yash Nisar  Advisor: Dr. Yuh-Jong Hu  Program: Social Networks and  Human Centered  Computing |

1. Richard S Sutton and Andrew G Barto. Reinforcement Learning: An Introduction. MIT Press, 1998.
2. Ian Goodfellow, Yoshua Bengio, and Aaron Courveille. Deep Learning. MIT Press, 2016 .
3. Yang Wang, Dong Wang, Shiyue Zhang, Yang Feng, Shiyao Li and Qiang Zhou. "Deep Q-Trading." 2017
4. Gabriel Molina. "Stock Trading with Recurrent Reinforcement Learning (RRL)." 2003