National University of Singapore
NUS Investment Society

RESEARCH REPORT

Zhihao LIU

Dueling Deep Q-Network for Portfolio Construction

Department: Quantitative Research
Group: Deep Learning

2017

Introduction

The core idea in reinforcement learning is to create an abstract "agent" that maps an action-value function using a neural network. The agent interacts with a defined environment, which in this case are the financial markets and a universe of investible assets. At step t the agent observes state s_t, (e.g. portfolio position, relative performance over an index etc.) and enacts action a_t, e.g. buy long, sell short etc. The aforementioned mapping is the policy $\pi(a|s)$, and the action value function is $Q^\pi(s,a) = \mathbb{E}[R_t|s_t = s, a_t = a, \pi]$, where R is the sum of discounted rewards over time. The Q function is iterated recursively using the Bellman Equation, represented over a neural net architecture which can be thought of as solving a dynamic programming problem defined over stochastic controls.

Given that asset pricing is a sub-problem of asset selection, it follows that the mapping of e.g. dataset inputs of securities' fundamentals to their price has a Markovian structure. The minimizing control process is found by standard machine learning loss optimization. One caveat is that the scope of this report will not cover the standard machine learning optimization concepts, finance theory on contingent claims and Fundamental Analysis, or rigorously prove the network architecture's capability of approximating any Borel-measurable functions.

Double Deep Q-Network

The sum of discounted rewards R_t sought to be maximised by the agent, where the discount is the trade-off between immediate and future rewards, is:

$$R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau, \qquad \gamma \in [0,1]. \tag{1}$$

For an agent behaving according to stochastic policy π, the state-action pair value is:

$$Q^\pi(s,a) = \mathbb{E}[R_t|s_t = s, a_t = a, \pi]. \tag{2}$$

The state value is:

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(s)}[Q^\pi(s,a)]. \tag{3}$$

The state-action Q function computed recursively:

$$Q^\pi(s,a) = \mathbb{E}'_s[r + \gamma \mathbb{E}_{a'\sim\pi}(s')[Q^\pi(s',a')]|s,a,\pi]. \tag{4}$$

Taking the optimal $Q^*(s,a) = \max_\pi Q^\pi(s,a)$, the Bellman Equation is satisfied by:

$$Q^*(s,a) = \mathbb{E}_s[r + \gamma \max_{a'} Q^*(s',a')|s,a]. \tag{5}$$

The advantage function is the value of acted actions over simply being in the current state, and has $\mathbb{E}_{a'\sim\pi}[A^\pi(s,a)] = 0$:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s). \tag{6}$$

The Q network with parameters θ optimized using Huber Loss over iteration i:

$$L_i(\theta_i) = \begin{cases} \frac{1}{2}(y_i^{DDQN} - Q(s,a;\theta_i))^2 & \text{for } |y - Q| \leq \delta, \\ \delta|y_i^{DDQN} - Q(s,a;\theta_i)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases} \tag{7}$$

The architecture maintains 2 Q-networks, with another "target" network with parameter θ^-, receive updates over N iterations (i.e. θ^- is copied over from previous episodes). In addition, the max operator evaluating actions is separate from the state evaluation:

$$y_i^{DDQN} = r + \gamma Q(s', \underset{a'}{\text{argmax}}\, Q(s', a', \theta_i); \theta^-). \tag{8}$$

At time t of the agent's interaction with the environment, it stores the experience tuple $e_t = (s_t, a_t, r_t, s_{t+1})$ into a dataset $\mathcal{D}_\sqcup = (e_1, ..., e_t)$. Mini-batches are sampled uniformly from $\mathcal{D}$ at random and the optimization for each mini-batch is:

$$\nabla_{\theta i} L_i(\theta_i) = [y_i^{DDQN} - Q(s,a;\theta_i)\nabla_{\theta i} Q(s,a;\theta_i)]. \tag{9}$$

Dueling Deep Q-Network

It turns out that Fundamental Analysis and pricing of derivatives do not occupy every step t and iteration i for predicting the security's price in the interests of portfolio construction. For some state V, the action advantage A should be separately estimated, whilst sharing a common convolutional feature learning module, and recombined at the end to obtain the Q value. In general, we are faced with two different paradigms of attacking the problem of noise in policy $\pi(a|s)$. One, assume a stochastic policy whereby the cost-minimized optimal control for Q^* is non-anticipative of diffusion and drift of relevant stochastic elements in the state space $\mathcal{S}$. If we characterize the optimization and training process for some units of time, due to the Markov property $\forall s \in \mathcal{S}$, every state element is "reinitalized" for the next episode. Two, and in the interest of the DDQN model discussed in this report, we use a Temporal Difference (TD) policy which updates the views on "old" states given the observation of immediate rewards. By (8), the separation of the action evaluation can operate by sampling all possible actions, independent of the Q estimation policy until the aforementioned N iterations. Q corresponding to s_t updates according to s_{t+1}.

At recombination of both action and state "streams", with α and β being the parameters for action and state values respectively:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + (A(s,a;\theta,\alpha) - \max_{a' \in |\mathcal{A}}(A(s,a;\theta,\alpha)). \tag{10}$$

References

[1] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen, S. Legg, V. Mnih, K. Kavukcuoglu, D. Silver, **Massively Parallel Methods for Deep Reinforcement Learning**, Google Deepmind, London (2015)

[2] K. Ross, **Stochastic Control in Continuous Time** , Stanford University Department of Statistics (2008)

[3] T. Shaul, J. Quan, I. Antonoglou, D. Silver **Prioritized Experience Replay**, Google Deepmind (2016)

[4] A. Wang, T. Shaul, M. Hessel, H. van Hasselt, M. Lanctot, N. de Freitas, **Dueling Network Architectures for Deep Reinforcement Learning**, Google Deepmind, London (2016)