March 29, 2019

1 Methodology

1.1 Overview

Before delving into the specifics of employing reinforcement learning to the problem of automated trading, it will be informative to discuss the general theory and its underpinning principles.

Reinforcement learning aims to maximise a given reward signal by undertaking certain actions (a restricted space). In this framework, an agent must take the environment state as input and take actions that alter the future state. A measurable goal related to the environment is also necessary in the problem formulation. Beyond this, each reinforcement learning problem contains four subelements: a *policy*, a *reward signal* and a *value function.*[1]

The policy defines the agent's actions in different environment states. The reward signal defines the goal and should be maximised over the course of the learning process.The value function maps the current state to a value so the agent can make optimal longer run decisions. It can be seen as the expected total future reward that can be obtained beginning from that state. Most of the challenges associated with the implementation of reinforcement learning derive from the estimation of the value function.

Formally, we construct a Markov Decision Process (MDP) In an ideal situation, we would have access to the value function directly in tabular form when we have a tractable action and state space.

At a sequence of discrete time steps $t = 0, 1, 2, 3...$, the agent interacts with the environment. At each step t, the agent receives state information $S_t \in \mathcal{S}$ and performs an action

---

[1]sutton book

$A_t \in \mathcal{A}(s)$. As a consequence of the action, the agent receives a reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and transitions to a new state S_{t+1}.

In the context of a *Markov* decision process, the future rewards (R_t) and states (S_t) only depend on the previous state and action.

The general reinforcement learning paradigm involves find an optimal policy π to maximise the expected discounted return. The discount factor is required to ensure that rewards in the distant future are less valuable than current rewards.

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Value and action-value functions allow the actions of the agent to be assessed under the implementation of a certain policy. The value function and action-value functions respectively are defined below:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t | S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right], \text{ for all } s \in \mathcal{S}$$

$$q_\pi(s, a) \doteq \mathbb{E}_\pi \left[G_t | S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right]$$

Ideally, the value function is decomposed into the following (known as the *Bellman's Equation*):

$$\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi \left[G_t | S_t = s \right] \\
&= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} | S_t = s \right] \\
&= \sum_a \pi(a|s) \sum_{s'} \sum_r p\left(s', r | s, a\right) \left[r + \gamma \mathbb{E}_\pi \left[G_{t+1} | S_{t+1} = s' \right] \right] \\
&= \sum_a \pi(a|s) \sum_{s',r} p\left(s', r | s, a\right) \left[r + \gamma v_\pi\left(s'\right) \right], \quad \text{for all } s \in S
\end{aligned}$$

Both expressions relate to a specific state and action taken at any time t.

A reinforcement learning problem principally involves pursuing the optimal policy π which is is said to maximise the value and action-value functions:

$$v_*(s) \doteq \max_\pi v_\pi(s)$$

$$q_*(s, a) \doteq \max_\pi q_\pi(s, a)$$

In the most simple cases where the value and action-value functions are specified, dynamic programming can be used to derive the optimal policy π.

In the algorithmic trading problem, the value function cannot be ascertained easily in this way. To deal with these situations and arbitrarily large state space, approximate solution methods must be used. This is known as a partially observable markov decision process as the state is only observed indirectly and cannot be fully known (we cannot know the trading behaviour of other agents for example) and we do not have access to the transition probabilities between states.

Q-Learning is a technique whereby the value functions are repeatedly estimated based on the rewards of our actions and assumes no prior model specification.

1.2 Q-Learning

Q-learning attempts to estimate q_* (optimal action-value function) without any regard for the policy followed.From a high level perspective, The Q-Learning algorithm proceeds by randomly initialising q, perform actions, measure reward and update q accordingly. The final output after a training period should be a stable approximation of the q_* table.

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \delta^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

Initialize S

Loop for each step of episode:

Choose A from S using policy derived from $Q($ e.g. $, \varepsilon$ -greedy $)$

Take action A, observe R, S'

$Q(S, A) \leftarrow Q(S, A) + \alpha\left[R + \gamma \max_a Q\left(S', a\right) - Q(S, A)\right]$

$S \leftarrow S'$

Notice the Bellman equation appearing in the update phase of the algorithm.

1.2.1 Deep Q Learning

An extension of this idea (which we aim to employ) is to use Neural Networks to approximate the Q-function.

A Neural Network is an appropriate tool for our use case due to the infinite nature of the state space. Estimating a table for each possible state would be excessive in regards to memory requirements.

To train the Neural Network on the state space, we must define a loss function. As The Bellman Equation defines the optimal result, we can use this to calculate our loss as follows:

$$\hat{Q}(s, a) = R(s, a) + \gamma \max_{a' \in A} Q(s, a)$$

$$\text{Loss} = \|Q - \hat{Q}\|_2$$

1.3 Problem formulation - Algorithmic Trading

Now we must formualate the trading problem as a Markov Decision Process and define the states, actions and rewards.

- State: $\mathcal{S} = \{prices, holdings, balance\}$

- Actions: $\mathcal{A} = \{buy, sell, hold\}$

- Rewards: $R_t \in \mathcal{R}$ can be defined as the change in the portfolio value due to an action A_t

- Policy: π which is governs the trading strategy at state S

- Action-value function: $q_\pi(s, a)$ as defined above. The expected reward we obtain by following policy π, choosing action A while in state S.

-