RL FOR OPTIMAL EXERCISE OF AMERICAN OPTIONS

ASHWIN RAO (STANFORD UNIVERSITY)

In this technical note, we explain how to solve the problem of Optimal Exercise of American Options using Reinforcement Learning. We start by showing how to solve this problem with a simple linear-approximation RL algorithm known as Least Squares Policy Iteration (LSPI). The LSPI algorithm is customized to the specific nuances of this Optimal Exercise problem. Finally, we show to solve this problem with Deep Q-Learning and Experience Replay.

1. Review of LSPI

Let us first consider the general LSPI algorithm. In each iteration, we are given:

$$Q(s,a) = \sum_i w_i \cdot \phi_i(s,a)$$

and deterministic policy π (known as the target policy for that iteration) is given by:

$$\pi(s) = \arg\max_a Q(s,a)$$

The goal in the iteration is to solve for weights $\{w'_i\}$ such that we minimize

$$\sum_{s,a,r,s'} (r + \gamma \cdot Q'(s', \pi(s')) - Q'(s,a))^2$$

over a data-set comprising of a sequence of 4-tuples (s,a,r,s') with:

$$Q'(s,a) = \sum_i w'_i \cdot \phi_i(s,a)$$
$$Q'(s',\pi(s')) = \sum_i w'_i \cdot \phi_i(s',\pi(s'))$$

Therefore, we solve for $\{w'_i\}$ that minimizes:

$$\sum_{s,a,r,s'} (r + \gamma \cdot Q'(s', \pi(s')) - \sum_i w'_i \cdot \phi_i(s,a))^2$$

So we calculate the gradient of the above expression with respect to $\{w'_j\}$ and set it to 0 (semi-gradient). This gives:

$$\sum_{s,a,r,s'} (r + \gamma \cdot Q'(s', \pi(s')) - \sum_i w'_i \cdot \phi_i(s,a)) \cdot \phi_j(s,a) = 0 \text{ for all } j \tag{1}$$

2. LSPI Customization

Now we customize LSPI to the specific nuances of our problem of Optimal Exercise of American Options. Our coverage is based on this paper by Li, Szepesvari, Schuurmans. We have two actions: $a = c$ (continue the American Option) and $a = e$ (exercise the American Option). We consider a function approximation for $Q'(s,a)$ only for the case of $a = c$ since we know the exact expression for the case of $a = e$, given by the option payoff function (call it $g : \mathcal{S} \rightarrow \mathbb{R}$). Therefore,

$$Q'(s,e) = g(s)$$

We write the function approximation for $Q'(s,c)$ as:

$$Q'(s,c) = \sum_i w'_i \cdot \phi_i(s,c) = \sum_i w'_i \cdot x_i(s)$$

for feature functions $x_i : \mathcal{S} \to \mathbb{R}$ (i.e., functions of only state and not action).

Since we are learning the Q-Value function for only $a = c$, our experience policy μ is a constant function $\mu(s) = c$. Also, for American Options, the reward for $a = c$ is 0. Thus, when considering the 4-tuples (s,a,r,s') for training experience, we always have $a = c$ and $r = 0$. So the 4-tuples are $(s,c,0,s')$ and so, we might as well simply consider 2-tuples (s,s') for training experience (since $a = c$ and $r = 0$ are locked). This means Equation (1) reduces to:

$$\sum_{s,s'} (\gamma \cdot Q'(s',\pi(s')) - \sum_i w'_i \cdot x_i(s)) \cdot x_j(s) = 0 \text{ for all } j \tag{2}$$

Now we need to consider two cases for $Q'(s',\pi(s'))$.

Case 1 (which we will denote as condition $C1$): If s' is a non-terminal state and $\pi(s') = c$ (this happens when $\sum_i w_i \cdot x_i(s') \geq g(s')$). In this case, we substitute $\sum_i w'_i \cdot x_i(s')$ for $Q'(s',\pi(s'))$ in Equation (2).

Case 2 (which we will denote as condition $C2$): If s' is a terminal state or $\pi(s') = e$ (this happens when $g(s') > \sum_i w_i \cdot x_i(s')$). In this case, we substitute $g(s')$ for $Q'(s',\pi(s'))$ in Equation (2).

Thus, we can now rewrite Equation (2), accounting for both of the above cases, as follows (note the $C1$ and $C2$ conditioning in the equation below):

$$\sum_{s,s'|C1} (\gamma \cdot \sum_i w'_i \cdot x_i(s') - \sum_i w'_i \cdot x_i(s)) \cdot x_j(s) + \sum_{s,s'|C2} (\gamma \cdot g(s') - \sum_i w'_i \cdot x_i(s)) \cdot x_j(s) = 0 \text{ for all } j$$

Swapping the summations over i and over (s,s'), we get:

$$\sum_i w'_i \cdot [\sum_{s,s'|C1} x_j(s) \cdot (x_i(s) - \gamma \cdot x_i(s')) + \sum_{s,s'|C2} x_j(s) \cdot x_i(s)] = \sum_{s,s'|C2} x_j(s) \cdot \gamma \cdot g(s') \text{ for all } j$$

Using the notation $\mathbb{1}_{C1}$ and $\mathbb{1}_{C2}$ to denote the indicator function for conditions $C1$ and $C2$ respectively, we can wewrite the above equation as:

$$\sum_i w'_i \cdot [\sum_{s,s'} x_j(s) \cdot (x_i(s) - \gamma \cdot \mathbb{1}_{C1} \cdot x_i(s'))] = \gamma \cdot \sum_{s,s'} x_j(s) \cdot \mathbb{1}_{C2} \cdot g(s') \text{ for all } j$$

This equation can be written more compactly in vector-matrix notation as $\mathbf{A} \cdot \mathbf{w'} = \mathbf{b}$ where the rows of the matrix $\mathbf{A}$ are indexed by j, the columns of $\mathbf{A}$ are indexed by i ($\mathbf{w'}$ is a column vector indexed by i), and the column vector $\mathbf{b}$ is indexed by j. We express $\mathbf{A}$ as $\mathbf{X}^T \cdot \mathbf{Y}$ and $\mathbf{b}$ as $\mathbf{X}^T \cdot \mathbf{v}$ where the rows of the matrix $\mathbf{X}$ are indexed by the 2-tuples (s,s'), the columns of $\mathbf{X}$ are indexed by j (an element in $\mathbf{X}$ is $x_j(s)$), the rows of matrix $\mathbf{Y}$ are indexed by the 2-tuples (s,s'), the columns of $\mathbf{Y}$ are indexed by i (an element in $\mathbf{Y}$ is $x_i(s) - \gamma \cdot \mathbb{1}_{C1} \cdot x_i(s')$), and the column vector $\mathbf{v}$ is indexed by the 2-tuples (s,s') (an element in $\mathbf{v}$ is $\gamma \cdot \mathbb{1}_{C2} \cdot g(s')$).

3. Solving with Deep Q-Learning and Experience Replay

Although the above LSPI algorithm is data-efficient and computationally-efficient as well, it is limited by the fact that our function approximation needs to be linear in features. Linearity is a significant constraint on this problem. In order to use a more general function approximation (such as a deep neural network), we will need to look into traditional incremental RL algorithms. A straightforward choice is

Q-Learning. We will employ Q-Learning together with Experience Replay by storing 2-tuples of (s, s') in a buffer and drawing from the buffer randomly (the random draws serve as training experience).

We use the same notation as we used above for LSPI. Let $\hat{f}$ denote a deep neural network function approximation for the Q-Value function when the action is to continue. Therefore,

$$Q(s, c) = \hat{f}(s; w)$$
$$Q(s, e) = g(s)$$
$$\pi(s) = \begin{cases} c & \text{if } \hat{f}(s; w) \geq g(s) \\ e & \text{otherwise} \end{cases}$$

where w denotes the weights of the deep neural network and $g(\cdot)$ is the option payoff function.

Noting that in the experience 4-tuples (s, a, r, s'), $a = c$ and $r = 0$ (as explained earlier), the Q-Learning update for each 2-tuple (s, s') from the Experience Replay is as follows:

$$w \leftarrow w + \alpha \cdot (\gamma \cdot Q(s', \pi(s')) - \hat{f}(s; w)) \cdot \nabla_w \hat{f}(s; w)$$

where α is the learning rate.

When s' is a non-terminal state, the update is:

$$w \leftarrow w + \alpha \cdot (\gamma \cdot \max(g(s'), \hat{f}(s'; w)) - \hat{f}(s; w)) \cdot \nabla_w \hat{f}(s; w)$$

When s' is a terminal state, the update is:

$$w \leftarrow w + \alpha \cdot (\gamma \cdot g(s') - \hat{f}(s; w)) \cdot \nabla_w \hat{f}(s; w)$$