Model-based Reinforcement Learning

Shivaram Kalyanakrishnan

August 24, 2018

Abstract

The Reinforcement Learning (RL) *problem* requires an agent that is interacting with an MDP to (eventually) start taking actions according to an optimal policy. The difference with the planning problem is that the transition and reward functions of the MDP are unknown to the agent—they only manifest themselves indirectly through (state, action, reward, next state) tuples. In this note, we outline a procedure for an agent to solve the RL problem by maintaining estimates of the transition and reward functions. These estimates (the *model*), which are refined based on experience, eventually converge to the true quantities. An agent that explores sufficiently, while also acting greedily with respect to its model, will eventually perform optimal action-selection.

1 Learning Algorithm

Assume an agent interacts with an MDP (S, A, T, R, γ). We shall assume that S, A, and γ are known to the agent: it is only T and R that are not directly available. However, if the agent takes action $a \in A$ from state $s \in S$, we may assume that the next state $s' \in S$ is drawn with probability $T(s, a, s')$, and the corresponding reward r is drawn from a real-valued distribution with mean $R(s, a, s')$.

If our agent must eventually figure out an optimal policy, it needs to be able to visit every state in the MDP. To ensure it does so, we make the convenient assumption that our MDP is such that the probability of reaching any state from any state, following any policy, is positive.

The algorithm shown on the next page is a model-based algorithm that estimates T and R through experience. Its estimates—$\hat{T}$ and $\hat{R}$—constitute a *model* of the environment. If each state-action pair is visited infinitely often, $\hat{T}$ and $\hat{R}$ (estimated based on empirical frequencies) will converge to T and R, respectively. An agent that takes actions that are optimal for the MDP $(S, A, \hat{T}, \hat{R}, \gamma)$, will, eventually, act optimally for the true MDP (S, A, T, R, γ), as well. To balance exploration and exploitation, our agent follows action-selection that is Greedy in the Limit with Infinite Exploration. One way to do so is to follow an ϵ_t-greedy policy with respect to the model, where $\epsilon_t = \frac{1}{t}$.

The exploration strategy can be further optimised; our primary interest at the moment is to establish sufficiency. The algorithm shown serves as a proof that the RL problem can indeed be solved. Observe that the space complexity of the algorithm is $\Theta(|S|^2|A|)$. Indeed *model-free* algorithms can solve the RL problem with only $\Theta(|S||A|)$ space. Hence, estimating the model is not a necessary step for solving the RL problem.

Model-based Reinforcement Learning

```
For $s, s' \in S, a \in A : \hat{T}[s][a][s'] \leftarrow 0$; $\hat{R}[s][a][s'] \leftarrow 0$.
$modelValid \leftarrow False$.

For $s, s' \in S, a \in A : totalTransitions[s][a][s'] \leftarrow 0$; $totalReward[s][a][s'] \leftarrow 0$.
For $s \in S, a \in A : totalVisits[s][a] \leftarrow 0$.

Assume that the agent is born in state $s^0$.
For $t = 0, 1, 2, \ldots :$
    If $modelValid$:
        $\pi^{opt} \leftarrow MDPPlan(S, A, \hat{T}, \hat{R}, \gamma)$.
        $a^t \leftarrow \begin{cases} \pi^{opt}(s^t) & \text{with probability } 1 - \epsilon_t, \\ UniformRandom(A) & \text{with probability } \epsilon_t. \end{cases}$
    Else:
        $a^t \leftarrow UniformRandom(A)$.

    Take action $a^t$, obtain reward $r^t$ and next state $s^{t+1}$.
    $UpdateModel(s^t, a^t, r^t, s^{t+1})$.
```

UpdateModel$(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}')$

```
    $totalTransitions[s][a][s'] \leftarrow totalTransitions[s][a][s'] + 1$.
    $totalVisits[s][a] \leftarrow totalVisits[s][a] + 1$.
    $totalReward[s][a][s'] \leftarrow totalReward[s][a][s'] + r$.

    For $s'' \in S$ :
        $\hat{T}[s][a][s''] \leftarrow \frac{totalTransitions[s][a][s'']}{totalVisits[s][a]}$.

    $\hat{R}[s][a][s'] \leftarrow \frac{totalReward[s][a][s']}{totalTransitions[s][a][s']}$.

    If $\neg modelValid$:
        If $\forall s'' \in S, \forall a'' \in A : totalVisits[s''][a''] \geq 1$:
            $modelValid \leftarrow True$.
```