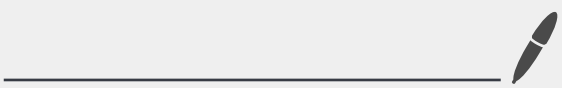
Model Based Reinforcement Learning

Yanjie Ze @CS19 SJTU

Mar 25, 2021



Model Based Reinforcement Learning

What's the difference between model-free and model-based reinforcement learning?

To answer this question, lets revisit the components of an MDP, the most typical decision making framework for RL.

An MDP is typically defined by a 4-tuple (S, A, R, T) where

S is the state/observation space of an environment

A is the set of actions the agent can choose between

$R(s, a)$ is a function that returns the reward received for taking action a in state s

$T(s'|s, a)$ is a transition probability function, specifying the probability that the environment will transition to state s' if the agent takes action a in state s.

Our goal is to find a policy π that maximizes the expected future (discounted) reward.

Now if we know what all those elements of an MDP are, we can just compute the solution before ever actually executing an action in the environment. In AI, we typically call computing the solution to a decision-making problem before executing an actual decision planning. Some classic planning algorithms for MDPs include Value Iteration, Policy Iteration, and whole lot more.

But the RL problem isn't so kind to us. What makes a problem an RL problem, rather than a planning problem, is the agent does *not* know all the elements of the MDP, precluding it from being able to plan a solution. Specifically, the agent does not know how the world will change in response to its actions (the transition function T), nor what immediate reward it will receive for doing so (the reward function R). The agent will simply have to try taking actions in the environment, observe what happens, and somehow, find a good policy from doing so.

So, if the agent does not know the transition function T nor the reward function R, preventing it from planning a solution out, how can it find a good policy? Well, it turns out there are lots of ways!

model based

One approach that might immediately strike you, after framing the problem like this, is for the agent to learn a model of how the environment works from its observations and then plan a solution using that model. That is, if the agent is currently in state $s1$, takes action $a1$, and then observes the environment transition to state $s2$ with reward $r2$, that information can be used to improve its estimate of $T(s2|s1,a1)$ and $R(s1,a1)$, which can be performed using supervised learning approaches. Once the agent has adequately modelled the environment, it can use a planning algorithm with its learned model to find a policy. RL solutions that follow this framework are model-based RL algorithms.

model free

As it turns out though, we don't have to learn a model of the environment to find a good policy. One of the most classic examples is Q-learning, which directly estimates the optimal Q-values of each action in each state (roughly, the utility of each action in each state), from which a policy may be derived by choosing the action with the highest Q-value in the current state. Actor-critic and policy search methods directly search over policy space to find policies that result in better reward from the environment. Because these approaches do not learn a model of the environment they are called model-free algorithms.

So if you want a way to check if an RL algorithm is model-based or model-free, ask yourself this question: after learning, can the agent make predictions about what the next state and reward will be before it takes each action? If it can, then it's a model-based RL algorithm. if it cannot, it's a model-free algorithm.

This same idea may also apply to decision-making processes other than MDPs.

Denote

时间范围 H，State Space S，Action Space A

策略 $\pi : S \times [H] \to A$

价值函数 $V_h^\pi(x)$

动作价值函数 $Q_h^\pi(x,a)$

} $\Rightarrow$ 下标加上 k 表明第 k 个 episode

最优价值函数 $V_h^*(x)$

最优动作价值函数 $Q_h^*(x,a)$

deterministic immediate reward $r_h^\pi(x) = R(x, \pi(x,h))$ $R(x,a) \in [0,1]$

Regret up to time K $R_k = \sum_{k=1}^{K} V_1^*(x_{k,1}) - V_1^{\pi_k}(x_{k,1})$

未知的 environment dynamics $P_h^\pi(y|x) = P(y|x, \pi(x,h))$

预估的 environment dynamics $\hat{P}_{k,h}(y|x,a)$

对未来价值的期望 $PV(x,a) = \sum_{y \in S} P(y|x,a) V(y) = \langle P(y|x,a), V(y) \rangle$

到第 k 轮第 h 步的 (x,a) 采样次数 $N_{k,h}(x,a)$

到第 k 轮第 h 步的 (x,a,y) 采样次数 $N_{k,h}(x,a,y)$

探索过 w 次及以上的 (x,a) pair 的集合 $K = \{(x,a) \in S \times A, N_{k,h}(x,a) > 0\}$

一个常数 C

Algorithm : Upper Confidence Bound Value Iteration

Algorithm 1: UCBVI

Initialize data buffer $H = \varnothing$;
for *epoch* $k = 1, 2, ...K$ **do**
  $Q_{k,h} =$ **UCBQ**(H) **for** *step* $h = 1,2,...,H$ **do**
    - Take action $a_{k,h} = \underset{a \in A}{argmax}\ Q_{k,h}(x_{k,h}, a)$
    - Sample $x_{k,h+1}$ from the dynamics of the Environment.
    - $H = H \cup (x_{k,h}, a_{k,h}, x_{k,h+1})$

  end
end

Algorithm 2: UCBQ

Data: Data H
Result: Q-value $Q_{k,h}$
Initialize $V_{k,H+1}(x) = 0$ for all $x \in S$;
Estimate $\hat{P}_{k,h}(y|x,a) = \begin{cases} \frac{N_{k,h}(x,a,y)}{N_{k,h}(x,a)}, & \text{for } (x,a) \in K \\ 0, & \text{otherwise} \end{cases}$; 根据采样次数估算 dynamics
Calculate "Bonus" $b_{k,h}(x,a) = \begin{cases} CH\sqrt{\frac{\ln(\frac{|S||A|T}{\delta})}{N_{k,h}(x,a)}}, & \text{for } (x,a) \in K \\ H, & \text{otherwise} \end{cases}$; explore 越少 bonus 越大 若没有explore过, bonus 为最大, H
Estimate $Q_{k,h}$ and $V_{k,h}$;
for $h = H, H\text{-}1, ..., 1$ **do**
  for $(x,a) \in S \times A$ **do**
    - $Q_{k,h} = \begin{cases} R(x,a) + (\hat{P}_{k,h}V_{k,h+1})(x.a) + b_{k,h}(x,a) & \text{for } (x,a) \in K \\ b_{k,h}(x,a), & \text{otherwise} \end{cases}$ bonus for exploration
    - $V_{k,h} = \max_{a \in A} Q_{k,h}(x,a)$

  end
end
return $Q_{k,h}$

Why not: $V_{k,h} = \sum_{a \in A} \pi(a|s) * Q^{\pi}_{k,h}(s,a)$?

Prof of the Upper Bound of Regret

首先定义 Regret：

$$R_K = \sum_{k=1}^{K} V_{k,1}^{*}(x) - V_{k,1}^{\pi_k}(x)$$

其中 $V_{k,1}^{\pi_k}(x)$ 是在第 k 轮的策略 π_k 下，h=1 时的 value

1. Backward Induction 后向归纳法

2. Union Bound 布尔不等式

证明：

$n=2$: $P(A_1 \cup A_2) = P(A_1) + P(A_2) - P(A_1 \cap A_2)$

$\leq P(A_1) + P(A_2)$

$n=k$: ✓

$n=k+1$ $P(A_1 \cup A_2 \cdots A_{k+1})$

$\leq P(A_1 \cup \cdots A_k) + P(A_{k+1})$ 归纳

$\leq P(A_1) + \cdots + P(A_{k+1})$

3.

Theorem 2: Hoeffding Inequality

Let $Z_1, Z_2, \ldots, Z_n$ be independent bounded random variables with $Z_i \in [a, b]$ for all i. Then we have

$$\mathbb{P}\left(\left|\frac{1}{n}\sum_{i=1}^{n}(Z_i - E[Z_i])\right| \geq t\right) \leq 2e^{-\frac{2nt^2}{(b-a)^2}}$$