Reinforcement Learning

Tateyama Kaoru

February 14, 2025

1 Continous RL

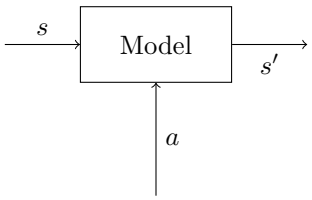
Two pratical ways to impliment continous RL algorithm are discretization and modeling RL.

1.1 Discretization

We can discretize state space and action space so we'll turn the problem as we do in discretized case.But a clear downside of discretization is that when we have a lot of states and actions,the time complexity to impliment this algorithm will goes up really quickly.For example,say we need n parameters to determine a stete and m parameters to determine an action,now we have k states in total so to fully solve the system,the time complexity for states and actions will be k^n and k^m respectively.If we discretize the system into too many pieces,we have a large k resulting in unacceptable time complexity.

1.2 Modelinig RL

In this part,the approach is that we will set up a model for RL.A model means a mapping $S \times A \to S$,where S denotes the state space and A denotes the action space.The image of this mapping can be a fixed value or a random variable up to whether we're building a deterministic or stochastic model.



For the sake of convenience,still,we discretize actions.To fit the model,we collect data by human control or other approaches and gather them together

$$s_0^{(1)} \xrightarrow{a_0^{(1)}} s_1^{(1)} \xrightarrow{a_1^{(1)}} s_2^{(1)} \longrightarrow \cdots \xrightarrow{a_{T-1}^{(1)}} s_T^{(1)}$$

$$\vdots$$

$$s_0^{(m)} \xrightarrow{a_0^{(m)}} s_1^{(m)} \xrightarrow{a_1^{(m)}} s_2^{(m)} \longrightarrow \cdots \xrightarrow{a_{T-1}^{(m)}} s_T^{(m)}$$

Then we'll use linear approximation to build the model.

1.3 Deterministic

We build the model by setting

$$s_{t+1} = As_t + Ba_t$$

where s_{t+1} represents the state at step $t+1$ and s_t and a_t represent state and action taken at step t.A and B are matrixes.

1.4 Stochastic

We tend not to build too complicated stochastic model for RL,so we just put

$$s_{t+1} = As_t + Ba_t + w_t$$

where $w_t \sim N(0, \Sigma)$.A and B are matrixes.

The optimization goal are the same

$$\min_{A,B} \sum_{i=1}^{m} \sum_{t=0}^{T} ||s_{t+1}^{(i)} - (As_t^{(i)} + Ba_t^{(i)})||^2$$

Now we will set model for $V(s)$.We define $\phi(s)$ to be a vector function of s and set

$$V^*(s) = \theta^T \phi(s)$$

We need to generate data set of $(s, V^*(s))$ so that we can fit the above equation.To do this,we sample $\{s^{(1)}, s^{(2)} \cdots s^{(m)}\}$ randomly and impliment the fowl-

lowing looop(called fitted value iteration)

$$
\begin{aligned}
&\textit{Loop}\\
&\quad\{\textit{for } i = 1, 2 \cdots m\\
&\quad\{\quad \textit{for } a_j \textit{ in action space } A\\
&\qquad\{\\
&\qquad\quad \textit{create set } \{s'_{j1}, s'_{j2} \cdots s'_{jk}\} \sim P_s^{(i)} a_j\\
&\qquad\qquad \textit{let } q(a_j) = \frac{1}{k}\sum_{l=1}^{k}[R(s^{(i)}) + \gamma V(s'_{jl})]\\
&\qquad\}\\
&\qquad\quad \textit{let } y^{(i)} = \max_a q(a)\\
&\quad\}\\
&\}
\end{aligned}
$$

The loop will end until outputs will live under tolerance.The blue part is simulating

$$R(s) + \gamma \sum P_{s^{(i)}a} V(s^{(i)})$$

and the orange part is simlulating

$$\max_a \left[R(s) + \gamma \sum P_{s^{(i)}a} V(s^{(i)})\right]$$

When we go through the loop for many rounds enough,$y^{(i)}$ will be reasonably close to $V^*(s^{(i)})$.Therefore,instead,we will use $(s^{(i)}, y^{(i)})$ to fit

$$V^*(s) = \theta^T \phi(s)$$

The optimization goal is nothing different from what we have in linear regression

$$\min_\theta \frac{1}{2}\sum_{i=1}^{m}(y^{(i)} - \theta^T \phi(s^{(i)}))^2$$

When we've done everything about fitting,we can determine the optimal policy via

$$\pi^*(s) = \operatorname*{argmax}_a E_{s' \sim P_{s,a}}[V^*(s')]$$

where we've already known how to calculate $V^*(s')\ \forall\ s'\ \in S$

2 Generalized Reward

2.1 State Action Reward

In a lot of occasions,Reward function is also dependent of action but not only state.Formally,we say Reward function R is a mapping

$$R : S \times A \to \mathbb{R}$$

Now Bellman's equation should be

$$V^*(s) = \max_a [R(s,a) + \gamma \sum P_{s,a}(s')V^*(s')]$$

Almost identical to common Bellman's equation except that the optimal goal is added up with a new term.

2.2 Finite Horizon MDP

We now consider a MDP where the machine works in given and finite steps.It doesn't try to get to destination as soon as possible but tries to get more profit in given time or steps.Therefore,we don't need penalty constant γ here and the payoff is now

$$\sum_{t=1}^{T} R(s_t, a_t)$$

Since we have only finite steps,the value function $V(s)$ and policy $\pi(s)$ will also depend on step t and we change the notation into $V_t(s)$ and $\pi_t(s)$.The optimazation goal is

$$V_t^*(s) = \max_a [R(s,a) + \sum^{s'} P_{s,a}(s')V_{t+1}^*(s')]$$

$$\pi_t^*(s) = \operatorname*{argmax}_a [R(s,a) + \sum^{s'} P_{s,a}(s')V_{t+1}^*(s')]$$

With initial condiction

$$V_T^*(s) = \max_a R(s,a)$$

We can slove $V_t^*(s)$ $\forall$ t *and* s along solving the sequance

$$V_T^*, V_{T-1}^* \cdots V_0^*$$

Afterwards,it will be simple to solve optimal policy as well.

3 Linear Quadratic Regularization