Successor representation (SR)

cyril regan

January 2021

1 Introduction

This objective of this report is to sum up the mathematical assumptions of the Successor Representation (Dayan, 1993; Gershman, 2018) with the formalism of Reinforcement Learning, as introduced in (Sutton-Barto, 1998).

2 MDP

A reinforcement learning task that satisfies the Markov property is called a Markov Decision Process, or MDP. A particular finite MDP is defined by its state and action sets and by the one-step dynamics of the environment. Hence, the probability of transitioning to $s_{t+1} = s$, given the previous 'history', is fully captured by conditioning only on the current state $s_t = s$ and action $a_t = a$. In other words, the future is independent of the past, given the present. This is described by the transition function:

$$T(s'|s,a) = \mathcal{P}^a_{ss'} = \mathbb{P}[s_{t+1} = s'|s_t = s, a_t = a] \tag{1}$$

Similarly, given any current state and action, s and a, together with any next state s', the expected value of the next reward is :

$$\mathcal{R}^a_{ss'} = \mathbb{E}[r_{t+1}|s_t = s, a_t = a, s_{t+1} = s'] \tag{2}$$

This characterization of the reward dynamics of an MDP in terms of $\mathcal{R}^a_{ss'}$ is slightly unusual. It is more common in the MDP literature to describe the reward dynamics in terms of the expected next reward given just the current state and action, i.e., by

$$r_{t+1} = \mathcal{R}^a_s = r(s,a) = \mathbb{E}[r_{t+1}|s_t = s, a_t = a] = \sum_{s'} \mathcal{P}^a_{ss'}\mathcal{R}^a_{ss'} \tag{3}$$

2.0.1 Policy

At any given state, the agent undergoes a decision-making process in order to select an action. The policy is a function that maps the state of an agent to an action. This can either be deterministic :

$$a = \pi(s) \tag{4}$$

or stochastic :

$$\pi(a|s) = \mathbb{P}[a_t = a|s_t = s] \tag{5}$$

2.0.2 Value functions

Value functions on policy : Reinforcement learning is concerned with the estimation of the *state-value function* $V(s)$ or *action-value function* $Q(s, a)$, the total reward an agent expects to earn in the future, with short-term rewards weighed more highly than long-term rewards. Of course the rewards the agent can expect to receive in the future depends on what actions it will take. Accordingly, value functions are defined with respect to particular policies. The state-value function is defined as the expected discounted future return following the policy π (Sutton-Barto, 1998):

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s] \\ &= \mathbb{E}_{\pi}[\sum_{k=0}^{+\infty} \gamma^k r_{t+k}|s_t = s] \end{aligned} \tag{6}$$

where γ is a discount factor that captures a preference for proximal rewards. Similarly the action-value function is defined by :

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[\sum_{k=0}^{+\infty} \gamma^k r_{t+k}|s_t = s, a_t = a] \tag{7}$$

State-value function can be written recursively with Bellman equation :

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}_{\pi}\left[\sum_{k=0}^{+\infty}\gamma^{k} r_{t+k} | s_t = s\right] \\ &= \mathbb{E}_{\pi}[r_{t+1} + \gamma \sum_{k=0}^{+\infty}\gamma^{k} r_{t+k+1} | s_t = s] \\ &= \sum_{a}\pi(s,a)\left[r_{t+1} + \sum_{s'}\mathcal{P}^{a}_{ss'}\gamma\,\mathbb{E}_{\pi}\left[\sum_{k=0}^{+\infty}\gamma^{k} r_{t+k+1} | s_{t+1} = s'\right]\right] \\ &= \sum_{a}\pi(s,a)\left[r_{t+1} + \gamma\sum_{s'}\mathcal{P}^{a}_{ss'}V^{\pi}(s')\right] \\ &= \sum_{a}\pi(s,a)\left[r(a,s) + \gamma\sum_{s'}\mathcal{P}^{a}_{ss'}V^{\pi}(s')\right] \end{aligned} \tag{8}$$

Value functions on optimal policy : A policy π is defined to be better than or equal to a policy π' if its expected return is greater than or equal to that of π' for all states. In other words, $\pi \geq \pi'$ if and only if $V^{\pi}(s) \geq V^{\pi'}(s)$ for all state $s \in \mathcal{S}$

All the optimal policies by denoted by π^* share the same state-value function V^*.

$$V^*(s) = \max_{\pi} V^{\pi}(s) \qquad \forall s \in \mathcal{S} \tag{9}$$

The optimal policies share also the same action-value function Q^* :

$$Q^*(s,a) = \max_{\pi} Q^{\pi}(s) \qquad \forall s \in \mathcal{S} \text{ and } a \in \mathcal{A}(s) \tag{10}$$

Thus, we can write Q^* in terms of V^* as follows:

$$Q^*(s,a) = \mathbb{E}[r_{t+1} + \gamma V^*(s_{t+1}) | s_t = s, a_t = a] \tag{11}$$

Intuitively, the Bellman optimality equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state:

$$\begin{aligned} V^*(s) &= \max_{a} Q^*(s,a) \\ &= \max_{a}\left[r(a,s) + \gamma\sum_{s'}\mathcal{P}^{a}_{ss'}V^*(s')\right] \end{aligned} \tag{12}$$

$$Q^*(a,s) = r(a,s) + \gamma \sum_{s'} \mathcal{P}^a_{ss'} \max_{a'} Q^*(s',a') \tag{13}$$

2.1 Model-based algorithms

Model-based decision algorithms are explicitly based on the underlying "model" (i.e., the reward function r and the transition function $\mathcal{P}^a_{ss'}$), which is either given a priori or learned. As the model is known, an estimate of the value functions can be updated by :

$$\hat{V} \leftarrow r(s) + \gamma \sum_{s'} \max_{a} \mathcal{P}^a_{ss'} \hat{V}(s') \tag{14}$$

$$\hat{Q}(s,a) \leftarrow r(s,a) + \gamma \sum_{s'} \mathcal{P}^a_{ss'} \max_{a'} \hat{Q}(s',a'), \tag{15}$$

Where $r(s) = \max_a r(s,a)$.

Drawback: this architecture is computationally expensive, because value estimation must iterate over the entire state space each time the model is updated. *Advantages* : agent endowed with a model requires only a small amount of experience to adapt to such changes.

2.2 Model-free algorithms

Model-free algorithms directly estimate the value function V from experienced transitions and rewards, without explicitly using or learning a model. In a deterministic setting, the estimated value functions $\hat{V}$ or $\hat{Q}$ can be represented by a lookup table storing the estimates for each state (resp state-action) while physically interacting with the environment, through temporal difference (TD) learning.

$$\hat{V}(s) \leftarrow r(s) + \gamma \hat{V}(s'), \tag{16}$$

$$\hat{Q}(s,a) \leftarrow r(s,a) + \gamma \max_{a'} \hat{Q}(s',a'), \tag{17}$$

Where s' corresponds to the successor state following the policy π of the Model-free algorithms.

The TD error δ is :

$$\delta = r(s) + \gamma \hat{V}(s') - \hat{V}(s) \tag{18}$$

$$\delta = r(s,a) + \gamma \max_{a'} \hat{Q}(s',a') - \hat{Q}(s,a) \tag{19}$$

And the update of the value function is done by :

$$\hat{V}^{new}(s) = \hat{V}^{old}(s) + \alpha.\delta, \tag{20}$$

$$\hat{Q}^{new}(s,a) = \hat{Q}^{old}(s,a) + \alpha.\delta, \tag{21}$$

where α is the learning rate.

Drawback: inflexibility because a local change in the transition or reward functions will produce nonlocal changes in the value function. *Advantages*: computationally efficient.

2.3 The successor representation

The successor representation M represent the "state occupancy". In fact, one can also see the successor representation as a "timeless" state representation. Indeed, the M matrix is a kind of map representing the occupancy of the actual state s and the projection of all future state occupancy (discounting by the γ factor) which could depend also of the action a taken at the actual state s. The construction of the successor representation M with the transition state function $\mathcal{P}^a_{ss'}$ is meaningful :

$$M(s',s,a) = 1_{s,s'} + \gamma \mathcal{P}^a_{ss'} + \gamma^2 (\mathcal{P}^a_{ss'})^2 + \gamma^3 (\mathcal{P}^a_{ss'})^3 + ... \tag{22}$$

Here, $M(s',s,a)$ represents the occupancy of state s' knowing the agent is presently in the state s and takes the action a. Let's take a look of the first term of the equation :

$$\begin{aligned} 1_{s,s'} &= 1 \text{ if } s' = s \\ 1_{s,s'} &= 0 \text{ if } s' \neq s \end{aligned} \tag{23}$$

$1_{s,s'}$ represents trivially the state occupancy of the "present time" which is null for all the states different from the actual.

Then the state occupancy of the first upcoming time is the probability to be in the state s' knowing the agent is in the state s and takes the action a. It is by definition the probability expressed by the state transition matrix discounting by the γ factor :

$$\gamma \mathcal{P}^a_{ss'} \tag{24}$$

But we can go on and project the state occupancy of the next upcoming time :

$$\gamma^2(\mathcal{P}^a_{ss'})^2 \tag{25}$$

To sum up, the successor representation is a cumulative probability to be in a state from the actual and all future projected times. To say it differently, M is defined as the discounted occupancy of state s, averaged over all possible trajectories initiated in state s. The SR can intuitively be thought of as a predictive map that encodes each state in terms of the other states that will be visited in the future.

Of course, the future state occupancy is only a "projection" of the present knowledge of the world expressed by the transition matrix $\mathcal{P}^a_{ss'}$. As the manipulation of a sum is not easy in a computational exploitation, one can express the sum with a straightforward limited development :

$$M(s', s, a) = \frac{1}{1 - \gamma\mathcal{P}^a_{ss'}} \tag{26}$$

or in a matrix way :

$$M(a) = (1 - \gamma\mathcal{P}^a)^{-1} \tag{27}$$

The successor representation is also popular because of its nice property which represents any value function as :

$$V(s) = \sum_{s'} M(s, s')r(s') \tag{28}$$

$$Q(s, a) = \sum_{s'} M(s, s', a)r(s', a) \tag{29}$$

Moreover, it is not necessarily to calculate directly the transition matrix $\mathcal{P}^a_{ss'}$ to get the successor representation M. An evaluation $\hat{M}$ can be learned via TD-learning for example with the δ_t error :

$$\delta_t(s') = \mathbb{I}[s_t = s'] + \gamma\hat{M}(s_{t+1}, s') - \hat{M}(s_t, s') \tag{30}$$

Notice that unlike the temporal difference error for value learning, the temporal difference error for SR learning is vector-valued, with one error for each successor state.

Advantage on flexibility : changes in the reward function will immediately propagate to all the action-value function. *Drawback* : this is not true of changes in the transition function

References

P. Dayan. *Improving Generalization for Temporal Difference Learning: The Successor Representation.* Computational Neurobiology Laboratory, The Salk Institute, P.0. Box 85800, Sun Diego, CA 92186-5800 USA, 1993.

S. J. Gershman. *The Successor Representation: Its Computational Logic and Neural Substrates.* The Journal of Neuroscience, 2018. ISBN 9781417642595.

Sutton-Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998 A Bradford Book, 1998.