AIMS Course 5: Reinforcement Learning

Answers To Questions About Tabular Methods

Jake Levi

November 2022

1 Introduction

We consider Markov Decision Processes (MDPs) [3], which generally consist of:

- A state space $\mathcal{S}$
- An action space $\mathcal{A}$
- A dynamics function $p(s', r|s, a)$ which specifies the probability after observing state $s \in \mathcal{S}$ and performing action $a \in \mathcal{A}$ of transitioning to state $s' \in \mathcal{S}$ and receiving reward $r \in \mathbb{R}$ (note that from the dynamics function it is straightforward to derive the expected reward $r(s, a)$ when action a is taken in state s, and the transition probability $p(s'|s, a)$ of transitioning to state s' after performing action a in state s)
- A discount factor $\gamma \in [0, 1]$
- A distribution over initial states (often this is implicit, and does not need to be specified or estimated)

The return R_t at time t is equal to the discounted sum of all rewards received during and after time t, and can be expressed recursively:

$$\begin{aligned} R_t &= r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots \\ &= \sum_{k=0}^{\infty} \left[\gamma^k r_{t+k}\right] \\ &= r_t + \gamma R_{t+1} \end{aligned}$$

A policy $\pi(a|s)$ specifies the probability of taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$. In general, the action taken during one time step can affect the rewards received (and states observed) on all future time steps, and the objective of reinforcement learning is to find an optimal policy $\pi_*(a|s)$ which maximises the expected discounted return from any state s. The expected return when following policy π in state s is known as the state value function $V^\pi(s)$:

$$\begin{aligned} V^\pi(s) &= \mathbb{E}_{p,\pi}\left[R_t \middle| s_t = s\right] \\ &= \sum_{R_t} \left[R_t p(R_t|s_t = s)\right] \end{aligned}$$

Similarly, the expected return when performing action a in state s and thereafter following policy π is known as the state-action value function $Q^\pi(s, a)$:

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}_{p,\pi}\left[R_t \middle| s_t = s, a_t = a\right] \\ &= \sum_{R_t} \left[R_t p(R_t|s_t = s, a_t = a)\right] \end{aligned}$$

The state value function and state-action value function can both be expressed in terms of each other:

$$
\begin{aligned}
V^\pi(s) &= \sum_{R_t} \Big[R_t p(R_t|s_t = s) \Big] \\
&= \sum_{R_t} \left[R_t \sum_{a \in \mathcal{A}} \Big[p(R_t, a|s_t = s) \Big] \right] \\
&= \sum_{R_t} \left[R_t \sum_{a \in \mathcal{A}} \Big[p(R_t, |s_t = s, a_t = a) \pi(a|s) \Big] \right] \\
&= \sum_{a \in \mathcal{A}} \left[\pi(a|s) \sum_{R_t} \Big[R_t p(R_t, |s_t = s, a_t = a) \Big] \right] \\
&= \sum_{a \in \mathcal{A}} \Big[\pi(a|s) Q^\pi(s, a) \Big] \\
Q^\pi(s, a) &= \mathbb{E}_{p,\pi} \Big[R_t \Big| s_t = s, a_t = a \Big] \\
&= \mathbb{E}_{p,\pi} \Big[r_t + \gamma R_{t+1} \Big| s_t = s, a_t = a \Big] \\
&= \mathbb{E}_{p,\pi} \Big[r_t \Big| s_t = s, a_t = a \Big] + \gamma \mathbb{E}_{p,\pi} \Big[R_{t+1} \Big| s_t = s, a_t = a \Big] \\
&= r(s, a) + \gamma \sum_{R_{t+1}} \Big[R_{t+1} p(R_{t+1}|s_t = s, a_t = a) \Big] \\
&= r(s, a) + \gamma \sum_{R_{t+1}} \left[R_{t+1} \sum_{s' \in \mathcal{S}} \Big[p(R_{t+1}, s'|s_t = s, a_t = a) \Big] \right] \\
&= r(s, a) + \gamma \sum_{R_{t+1}} \left[R_{t+1} \sum_{s' \in \mathcal{S}} \Big[p(R_{t+1}|s_{t+1} = s') p(s'|s, a) \Big] \right] \\
&= r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \left[p(s'|s, a) \sum_{R_{t+1}} \Big[R_{t+1} p(R_{t+1}|s_{t+1} = s') \Big] \right] \\
&= r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|s, a) V^\pi(s') \Big]
\end{aligned}
$$

This leads to the recursive Bellman equations for $V^\pi(s)$ and $Q^\pi(s, a)$:

$$
\begin{aligned}
V^\pi(s) &= \sum_{a \in \mathcal{A}} \left[\pi(a|s) \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|s, a) V^\pi(s') \Big] \right) \right] \\
Q^\pi(s, a) &= r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \left[p(s'|s, a) \left(\sum_{a \in \mathcal{A}} \Big[\pi(a|s) Q^\pi(s, a) \Big] \right) \right]
\end{aligned}
$$

2 Policy Evaluation

The Bellman equation for the state value function $V^\pi(s)$ can be expressed in matrix form as follows (note that here the expected reward $R(s', s, a)$ when transitioning to state s' after performing action a in state s is used instead of the expected reward $r(s, a)$ when action a is taken in state s, which is not conditioned on the next state):

$$
\begin{aligned}
\underbrace{V^\pi(s)}_{v_s^\pi} &= \sum_{a\in\mathcal{A}}\left[\pi(a|s)\sum_{s'\in\mathcal{S}}\left[p(s'|s,a)\Big(R(s',s,a)+\gamma V^\pi(s')\Big)\right]\right] \\
&= \underbrace{\sum_{a\in\mathcal{A}}\left[\pi(a|s)\sum_{s'\in\mathcal{S}}\left[p(s'|s,a)R(s',s,a)\right]\right]}_{r_s^\pi} + \gamma\sum_{a\in\mathcal{A}}\left[\pi(a|s)\sum_{s'\in\mathcal{S}}\left[p(s'|s,a)V^\pi(s')\right]\right] \\
&= r_s^\pi + \gamma\sum_{s'\in\mathcal{S}}\left[\underbrace{\sum_{a\in\mathcal{A}}\left[\pi(a|s)p(s'|s,a)\right]}_{P_{s,s'}^\pi}\underbrace{V^\pi(s')}_{v_{s'}^\pi}\right] \\
&= r_s^\pi + \gamma\sum_{s'\in\mathcal{S}}\left[P_{s,s'}^\pi v_{s'}^\pi\right] \\
&= r_s^\pi + \gamma(P^\pi v^\pi)_s \\
\Rightarrow \quad v^\pi &= r^\pi + \gamma P^\pi v^\pi
\end{aligned}
$$

$$
\text{where} \quad \begin{cases} v_i^\pi &= V^\pi(i) \\ r_i^\pi &= \sum_{a\in\mathcal{A}}\left[\pi(a|i)\sum_{s'\in\mathcal{S}}\left[p(s'|i,a)R(s',i,a)\right]\right] \\ P_{i,j}^\pi &= \sum_{a\in\mathcal{A}}\left[\pi(a|i)p(j|i,a)\right] \\ &= p(s_{t+1}=j|s_t=i) \end{cases}
$$

If the value of any or all states is unknown, but the transition probabilities and expected rewards are known, then the dynamic programming technique of policy evaluation can be used to estimate the values of unknown states. Policy evaluation refers to iteratively updating an estimate for the value of each state as follows:

$$
\begin{aligned}
v_{k+1}^\pi(s) &= \mathbb{E}_\pi\left[r_t + \gamma v_k^\pi(s_{t+1})\Big|s_t = s\right] \\
&= \sum_{a\in\mathcal{A}}\left[\pi(a|s)\left(r(s,a) + \gamma\sum_{s'\in\mathcal{S}}\left[p(s'|s,a)v_k^\pi(s')\right]\right)\right]
\end{aligned}
$$

For $\gamma \in [0, 1)$ it can be proved that policy evaluation converges. The error in the estimate $v_k^\pi(s)$ of the state value function $V^\pi(s)$ of state s during iteration k of policy evaluation is denoted by $\varepsilon_k(s)$:

$$
\begin{aligned}
(\forall s \in \mathcal{S}) \qquad & v_k^\pi(s) = V^\pi(s) + \varepsilon_k(s) \\
\Rightarrow (\forall s \in \mathcal{S}) \qquad & v_{k+1}^\pi(s) = \sum_{a \in \mathcal{A}} \left[\pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \left[p(s'|s,a) \Big(V^\pi(s') + \varepsilon_k(s') \Big) \right] \right) \right] \\
& \qquad = \sum_{a \in \mathcal{A}} \left[\pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|s,a) V^\pi(s') \Big] + \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|s,a) \varepsilon_k(s') \Big] \right) \right] \\
& \qquad = V^\pi(s) + \gamma \sum_{a \in \mathcal{A}} \left[\pi(a|s) \sum_{s' \in \mathcal{S}} \Big[p(s'|s,a) \varepsilon_k(s') \Big] \right] \\
& \qquad \leq V^\pi(s) + \gamma \sum_{a \in \mathcal{A}} \left[\pi(a|s) \sum_{s' \in \mathcal{S}} \Big[p(s'|s,a) \max_{\bar{s}} \Big[\varepsilon_k(\bar{s}) \Big] \Big] \right] \\
& \qquad = V^\pi(s) + \gamma \max_{\bar{s}} \Big[\varepsilon_k(\bar{s}) \Big] \\
(\forall s \in \mathcal{S}) \qquad & v_{k+1}^\pi(s) = V^\pi(s) + \varepsilon_{k+1}(s) \\
\Rightarrow (\forall s \in \mathcal{S}) \qquad & V^\pi(s) + \varepsilon_{k+1}(s) \leq V^\pi(s) + \gamma \max_{\bar{s}} \Big[\varepsilon_k(\bar{s}) \Big] \\
\Rightarrow (\forall s \in \mathcal{S}) \qquad & \varepsilon_{k+1}(s) \leq \gamma \max_{\bar{s}} \Big[\varepsilon_k(\bar{s}) \Big] \\
\Rightarrow \qquad & \max_{\bar{s}} \Big[\varepsilon_k(\bar{s}) \Big] \leq \gamma \max_{\bar{s}} \Big[\varepsilon_k(\bar{s}) \Big]
\end{aligned}
$$

This implies that the upper bound across states of the error in the estimated state value function decreases at least by a factor of γ on every iteration, and therefore decreases at least exponentially (similar reasoning can be used to derive the lower bound). As a result, a desired tolerance in the estimated state value function will always be reached if policy evaluation is run for sufficiently many iterations.

3 Policy And Value Iteration

The policy improvement theorem states that a new policy which is greedy with respect to the value function of an existing policy has equal or better value than the existing policy in every state. We start off by proving that a policy σ which is greedy with respect to the value of a single state under π (and is equal to π in every other state) has value in every state which is greater than or equal to the value of the same state under π, and then apply similar reasoning to prove the full policy improvement theorem. To begin with, say we are using a (possibly stochastic) policy π, and consider a policy σ, which is identical to π in all states except $\bar{s}$, in which σ chooses action $\bar{a}$ deterministically:

$$
\sigma(a|s) = \begin{cases} \pi(a|s) & s \neq \bar{s} \\ 1 & s = \bar{s}, a = \bar{a} \\ 0 & s = \bar{s}, a \neq \bar{a} \end{cases}
$$

Is the value of choosing action $\bar{a}$ in state $\bar{s}$ higher under the new policy? We can express the difference in the value of this state-action pair under the two policies in terms of the change in value of every state:

$$
Q^\sigma(\bar{s}, \bar{a}) - Q^\pi(\bar{s}, \bar{a}) = \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|\bar{s}, \bar{a}) \Big(V^\sigma(s') - V^\pi(s') \Big) \Big]
$$

The change in value of every state besides $\bar{s}$ can be expressed recursively:

$$
\begin{aligned}
(\forall s \neq \bar{s}) \quad V^\sigma(s) - V^\pi(s) &= \gamma \sum_{a \in \mathcal{A}} \left[\pi(a|s) \sum_{s' \in \mathcal{S}} \Big[p(s'|s,a) \Big(V^\sigma(s') - V^\pi(s') \Big) \Big] \right] \\
&= \gamma \sum_{s' \in \mathcal{S}} \left[\underbrace{\sum_{a \in \mathcal{A}} \Big[\sigma(a|s) p(s'|s,a) \Big]}_{P^\sigma_{s,s'}} \Big(V^\sigma(s') - V^\pi(s') \Big) \right]
\end{aligned}
$$

The change in the value of state $\bar{s}$ can be expressed as follows:

$$\begin{aligned}
V^\sigma(\bar{s}) &= \sum_{a\in\mathcal{A}}\Big[\sigma(a|\bar{s})Q^\sigma(\bar{s},a)\Big] \\
&= Q^\sigma(\bar{s},\bar{a}) \\
\Rightarrow \quad V^\sigma(\bar{s}) - V^\pi(\bar{s}) &= Q^\sigma(\bar{s},\bar{a}) - V^\pi(\bar{s}) \\
&= \Big(Q^\sigma(\bar{s},\bar{a}) - Q^\pi(\bar{s},\bar{a})\Big) + \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big) \\
&= \left(\gamma\sum_{s'\in\mathcal{S}}\Big[p(s'|\bar{s},\bar{a})\Big(V^\sigma(s') - V^\pi(s')\Big)\Big]\right) + \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big) \\
&= \gamma\sum_{s'\in\mathcal{S}}\left[\underbrace{\sum_{a\in\mathcal{A}}\Big[\sigma(a|\bar{s})p(s'|\bar{s},a)\Big]}_{P^\sigma_{\bar{s},s'}}\Big(V^\sigma(s') - V^\pi(s')\Big)\right] + \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big)
\end{aligned}$$

These expressions can now be assembled in matrix form using the basis vector $e_{\bar{s}}$ for state $\bar{s}$:

$$\begin{aligned}
v^\sigma - v^\pi &= \gamma P^\sigma(v^\sigma - v^\pi) + \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big)e_{\bar{s}} \\
\text{where} \quad (e_{\bar{s}})_i &= \begin{cases} 1 & i = \bar{s} \\ 0 & i \neq \bar{s} \end{cases} \\
\Rightarrow \quad v^\sigma - v^\pi &= \gamma P^\sigma\left(\gamma P^\sigma(v^\sigma - v^\pi) + \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big)e_{\bar{s}}\right) + \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big)e_{\bar{s}} \\
&= \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big)\Big(I + \gamma P^\sigma\Big)e_{\bar{s}} + \gamma^2(P^\sigma)^2(v^\sigma - v^\pi) \\
&\vdots \\
&= \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big)\sum_{n=0}^{N-1}\Big[\gamma^n(P^\sigma)^n\Big]e_{\bar{s}} + \gamma^N(P^\sigma)^N(v^\sigma - v^\pi) \\
&= \Big(Q^\pi(\bar{s},\bar{a}) - V^\pi(\bar{s})\Big)\sum_{n=0}^{\infty}\Big[\gamma^n(P^\sigma)^n\Big]e_{\bar{s}}
\end{aligned}$$

$$\begin{aligned}
&(\forall s, s' \in \mathcal{S})(\forall n \in \mathbb{N}) \quad (\gamma^n(P^\sigma)^n)_{s,s'} \geq 0 \\
&\qquad \Rightarrow \quad \Big(Q^\pi(\bar{s},\bar{a}) > V^\pi(\bar{s})\Big) \Rightarrow \Big((\forall s \in \mathcal{S}) \quad V^\sigma(s) \geq V^\pi(s)\Big)
\end{aligned}$$

This completes the proof that greedification with respect to a single state $\bar{s}$ leads to policy improvement. We now turn to the full policy improvement theorem, which states that greedification with respect to all states leads to policy improvement. Consider the same original policy π, and a new policy σ which in every state s chooses action a which is greedy with respect to $Q^\pi(s,a)$ (ties in the argmax operator are broken arbitrarily):

$$(\forall s \in \mathcal{S})(\forall a \in \mathcal{A}) \quad \sigma(a|s) = \begin{cases} 1 & a = \underset{a'}{\operatorname{argmax}}\Big[Q^\pi(s,a')\Big] \\ 0 & \text{otherwise} \end{cases}$$

Denote the action chosen by σ in state s by $\bar{a}_s$:

$$(\forall s \in \mathcal{S}) \quad \bar{a}_s = \underset{a'}{\operatorname{argmax}}\Big[Q^\pi(s,a')\Big]$$

The difference in the state-action value function of choosing action $\bar{a}_s$ in state s under policies σ and π can be expressed in terms of the differences in state value functions of subsequent states:

$$Q^\sigma(s,\bar{a}_s) - Q^\pi(s,\bar{a}_s) = \gamma\sum_{s'\in\mathcal{S}}\Big[p(s'|s,\bar{a}_s)\Big(V^\sigma(s') - V^\pi(s')\Big)\Big]$$

Note that the state-action value function $Q^\sigma(s, \bar{a}_s)$ of σ choosing action $\bar{a}_s$ in state s is equal to the state value function $V^\sigma(s)$ of s under σ (because σ chooses $\bar{a}_s$ deterministically):

$$\begin{aligned} V^\sigma(s) &= \sum_{a \in \mathcal{A}} \Big[\sigma(a|s) Q^\sigma(s, a) \Big] \\ &= Q^\sigma(s, \bar{a}_s) \end{aligned}$$

Also note that the state-action value function $Q^\pi(s, \bar{a}_s)$ of π choosing action $\bar{a}_s$ in state s is related to the state value function $V^\pi(s)$ of s under π by the advantage function $A^\pi(s, \bar{a}_s)$, which in this case is necessarily non-negative:

$$\begin{aligned} (\forall s \in \mathcal{S}) \quad V^\pi(s) &= \sum_{a \in \mathcal{A}} \Big[\pi(a|s) Q^\pi(s, a) \Big] \\ &\leq \sum_{a \in \mathcal{A}} \Big[\pi(a|s) \max_{a'} \Big[Q^\pi(s, a') \Big] \Big] \\ &= \max_{a'} \Big[Q^\pi(s, a') \Big] \\ &= Q^\pi(s, \bar{a}_s) \\ \Rightarrow \quad V^\pi(s) &\leq Q^\pi(s, \bar{a}_s) \\ A^\pi(s, \bar{a}_s) &= Q^\pi(s, \bar{a}_s) - V^\pi(s) \\ \Rightarrow (\forall s \in \mathcal{S}) \quad A^\pi(s, \bar{a}_s) &\geq 0 \end{aligned}$$

The equation for the difference in state-action value functions can now be expressed purely in terms of differences between state value functions and the advantage function, and this equation can further be expressed as a recursive matrix equation:

$$\begin{aligned} V^\sigma(s) - \Big(V^\pi(s) + A^\pi(s, \bar{a}_s) \Big) &= \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|s, \bar{a}_s) \Big(V^\sigma(s') - V^\pi(s') \Big) \Big] \\ \Rightarrow \quad V^\sigma(s) - V^\pi(s) &= A^\pi(s, \bar{a}_s) + \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|s, \bar{a}_s) \Big(V^\sigma(s') - V^\pi(s') \Big) \Big] \\ \Rightarrow \quad v^\sigma - v^\pi &= \beta + \gamma P^\sigma (v^\sigma - v^\pi) \\ \text{where} \quad & \begin{cases} v^\sigma_s &= V^\sigma(s) \\ v^\pi_s &= V^\pi(s) \\ \beta_s &= A^\pi(s, \bar{a}_s) \\ P^\sigma_{s,s'} &= p(s'|s, \bar{a}_s) \end{cases} \\ \Rightarrow \quad v^\sigma - v^\pi &= \beta + \gamma P^\sigma \left(\beta + \gamma P^\sigma (v^\sigma - v^\pi) \right) \\ &= (I + \gamma P^\sigma) \beta + \gamma^2 (P^\sigma)^2 (v^\sigma - v^\pi) \\ &\vdots \\ &= \sum_{n=0}^{N-1} \Big[\gamma^n (P^\sigma)^n \Big] \beta + \gamma^N (P^\sigma)^N (v^\sigma - v^\pi) \\ &= \sum_{n=0}^{\infty} \Big[\gamma^n (P^\sigma)^n \Big] \beta \\ (\forall s, s' \in \mathcal{S})(\forall n \in \mathbb{N}) \quad (\gamma^n (P^\sigma)^n)_{s,s'} &\geq 0 \\ (\forall s \in \mathcal{S}) \quad \beta_s &\geq 0 \\ \Rightarrow (\forall s \in \mathcal{S}) \quad \left(\sum_{n=0}^{\infty} \Big[\gamma^n (P^\sigma)^n \Big] \beta \right)_s &\geq 0 \\ \Rightarrow (\forall s \in \mathcal{S}) \quad V^\sigma(s) &\geq V^\pi(s) \end{aligned}$$

This concludes the proof of the policy improvement theorem.

Value iteration is a dynamic programming approach which attempts to find the optimal state value function using the following iterative update:

$$V_{k+1}(s) = \max_a \left[r(s, a) + \gamma \sum_{s' \in \mathcal{S}} \Big[p(s'|s, a) V_k(s') \Big] \right]$$

It should be possible to prove that value iteration converges to the optimal state value function using a combination of the arguments used to prove the convergence of policy iteration and the policy improvement theorem, with actions being selected so as to maximise the current *estimate* of the state-action value functions, which includes some unknown error ε_k. A further challenge in proving that value iteration converges to the optimal state value function is proving that there is a *unique* state value function which is a fixed point of value iteration (which is to say that there is no state value function which is locally optimal across all states but globally sub-optimal).

4 Control Without A Model

Policy evaluation, policy iteration and value iteration are dynamic programming approaches which all assume access to an exact model of the environment, however in the general reinforcement learning problem such a model is not available. Two simple approaches to control without a model are constant-α Monte Carlo (MC) and Temporal Difference (TD) methods. MC estimates the return from a given state using samples of the reward from that state and all subsequently observed states of an episode, whereas TD(0) estimates the return from a given state using the sample of the reward from that state, and the "bootstrapped" estimate of the return from the next state. The differences between how these algorithms estimate return leads to differences in the bias and variance of the updates that these algorithms perform.

MC methods have high variance, because the estimate of the return at each time step depends on many rewards which are sampled stochastically from the environment. MC methods also have low bias, because sampled rewards come from the true distribution of rewards in each state, and not an estimate of the reward distribution based on previous samples.

Conversely, TD(0) methods have low variance, because the estimate of the return at each time step only depends on a single stochastic sample of the reward. However, TD(0) methods also have high bias, because each estimate of the return depends on a previous estimate of the return from the subsequent state, and in general this estimate will not be fully accurate.

In summary, MC methods tend to have lower bias, but TD(0) methods tend to have lower variance.

When using such a method in practise, an ϵ-greedy policy is useful because it ensures that in any given state, every action has a non-zero probability of being chosen. As a result, if these algorithms run for long enough, every state-action pair will be sampled enough times to reach a desired level of certainty in its value. If a policy was used in which some state-action pairs were never sampled, it could be the case that some states were never visited, so their value could not be accurately estimated, and the algorithm might never converge to an optimal policy. In general, a behaviour policy is the policy used when interacting with the environment, and the target policy is the policy whose value is being estimated, and these two policies need not be the same (for example in off-policy methods). To guarantee convergence of the *target* policy to an optimal policy, it is necessary for the *behaviour* policy to sample all state-action pairs with non-zero probability.

Q-learning is an alternative to MC and TD(0) methods, which estimates the optimal expected return when choosing any action in any given state, using the following update equation:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left(r_t + \gamma \max_a \Big[Q(s_{t+1}, a) \Big] - Q(s_t, a_t) \right)$$

Q-learning can be interpreted as estimating the value of a state-action pair according to a policy that only chooses optimal actions during future time steps, and that never chooses sub-optimal actions during future timesteps. Therefore, Q-learning can be interpreted as estimating the state-action value function of an optimal *target* policy. In general, the behaviour policy will choose all actions in a given state with non-zero probability (for example using ϵ-greedy choices) in order to guarantee convergence, which in general will be different to (and have lower value than) the target policy. Q-learning can therefore be considered an off-policy method because the policy whose value it estimates (the target policy) is different to the policy it follows in the environment (the behaviour policy). An advantage of off-policy methods is that an explorative policy can be used as the behaviour policy, allowing estimation of every state action pair and guaranteeing eventual convergence, while learning a target policy which is an optimal policy.

5 Approximate Q-learning

Tabular Q-learning (estimating a table of values for each state-action pair) does not scale well to problems with large state spaces and/or large action spaces, motivating the use of approximate Q-learning methods. Approximate Q-learning methods use a parameterised estimate $Q(s, a, \theta)$ of the state-action value function, which ideally allows generalisation between state-action pairs, and optimise the parameters θ in order to find the best approximation to the state-action value function of an optimal policy. In [1], this problem is approached by minimising the expected squared Bellman error using stochastic gradient descent:

$$\mathcal{L}(\theta_i) = \mathbb{E}\left[\Big(y_i - Q(s, a, \theta_i)\Big)^2\right]$$
$$\text{where} \quad y_i = \mathbb{E}\left[r + \gamma \max_{a'}\Big[Q(s', a', \theta_{i-1})\Big]\Big|s, a\right]$$

The optimisation of $\mathcal{L}$ is not performed with respect to the target y_i (although y_i technically depends on the parameters θ), and therefore this is known as a semi-gradient method.

In certain cases (for example if the approximate state-action value function is linear with respect to its parameters), semi-gradient methods can be proven to converge. In practise, learning with such methods can be unstable. One reason for this is that convergence of the linear semi-gradient method assumes convergence of certain quantities to their expected values, which assumes conditions on the learning rate which are not satisfied if the learning rate is constant. Another reason for the instability is that although generalisation between state-action pairs is advantageous because it allows generalisation to state-action pairs which have not been sampled, it has the downside that changes to the parameters that improve the accuracy of one state-action pair can cause changes in estimates of the value of other state-action pairs such that they become less accurate, which further can distort the calculation of $\max_{a'}\Big[Q(s', a', \theta_{i-1})\Big]$, and therefore distort the parameter updates for subsequent state-action pairs.

A solution proposed by [2] is to use a frozen target network for estimating y_i with parameters θ^-, with θ^- only being updated with the newest set of parameters θ_i every C time steps, for some C (for example 100). This is equivalent to minimising the following loss function:

$$\mathcal{L}(\theta_i) = \mathbb{E}\left[\Big(r + \gamma \max_{a'}\Big[Q(s', a', \theta^-)\Big] - Q(s, a, \theta_i)\Big)^2\right]$$

This approach can improve stability of the learning process, but instability can still be observed in practise because inaccuracies in the frozen target network weights θ^- can still distort the calculation of $\max_{a'}\Big[Q(s', a', \theta^-)\Big]$.

A solution known as double Q-learning proposed by [4] is to use the online network weights θ_i to choose the approximately optimal action to take in the next time step, and use the frozen target network weights θ^- to approximate the value of that action, which is equivalent to minimising the following loss function:

$$\mathcal{L}(\theta_i) = \mathbb{E}\left[\Big(r + \gamma Q(s', \operatorname*{argmax}_{a'}\Big[Q(s', a', \theta_i)\Big], \theta^-) - Q(s, a, \theta_i)\Big)^2\right]$$

Of the approaches outlined so far, double Q-learning tends to be the most stable, because it avoids the instability caused by using the max operator. The max operator can be very sensitive to changes in parameters (of either the online network or the frozen target network), leading to overly optimistic estimates of state-action values, which derail future parameter updates and destabilise learning. Therefore the success of double Q-learning can arguably be attributed to its avoidance of using the max operator.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[3] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[4] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.