Markov decision process

Program	Agent	environment	action	state

,

Markov decision process (MDP):

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, ...$$

Reward is a signal from the environment

Policy function $\pi(a \mid s) = $ probability

return $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + ...$

discount factor: $0 \leq \gamma \leq 1$

Monte Carlo method

world models: $p(s', r \mid s, a)$

An episode describes a single run through of the process, with state and return reset after each epsidoe.

Value functions: keep track of average return (G_t) expected when following a certain policy (π) in a certain state (s) or state + actions (s, a)

State-value: $V_\pi(s)$, expected return when in state s while follow policy π, or the average expected return for $Q_\pi(s, a)$ accross all possible a.

Action-value: $Q_\pi(s, a)$, expected return when in state s, pick a certain action a, and then follow policy (π).

Optimal functions: $V_*(s)$, $Q_*(s, a)$.

Three methods of RL:
1. Policy gradient method
    - Issues with variance (different averages)
    - More dependent on policy function
2. Policy gradient method, with value function
    - Accounts for variance, which is good
    - Balance of using policy & value function
3. Just use action-value (Q) function
    - Policy becomes dependent on action-value function
    - More dependent on value function

Developing Q following method 3

Epsilon-greedy balance:

Epsilon (ε) describes how much the policy picks a random value as opposed to the optimal value.

Epsilon usually starts off high (because the agent is not much better than random), and becomes lower and lower ask training progresses. This ensures sufficient exploration while retaining effective exploitation.

Temporal Difference

Credit assignment problem: figuring out the impact of nidividual actions within a sequence of many actions.

Temporal difference breaks down evaluation to an action by action basis, rather than episode by episode.

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots$$

$$G_t = r_t + \gamma(r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots)$$

$$G_t = r_t + G_{t+1}$$

Learning **action-value (Q)** function

SARSA:

$$Q(s_t, a_t) \to r_t + \gamma Q(s_{t+1}, a_{t+1})$$

Expected SARSA:

$$Q(s_t, a_t) \to r_t + \gamma \sum_{i=1}^{n} \pi(a, s_{t+1}) Q(s_{t+1}, a)$$

Q-Learning:

$$Q(s_t, a_t) \to r_t + \gamma \max(a) Q(s_{t+1}, a)$$

Learning **state-value (V)** function:

$$V(s_t) \to r_t + \gamma V(s_{t+1})$$

Actual math for learning at episode e:

$$Q_{e+1}(s_t, a_t) = Q_e(s_t, a_t) + \alpha[r_t + \gamma Q_e(s_{t+1}, a_{t+1}) - Q_e(s_t, a_t)]$$

Bellman's Principle of Optimality: The overall best policy must involve choosing the best action at every point.

Deep Q Networks