Damoun Ayman
Github repository

Reinforcement Learning
November 15, 2020

Dynamic Programming
(Value Iteration, Policy Iteration, Q-Learning)

0.1 Recall

- Reinforcement learning is the filed of learning to make decisions.
- Agents can learn a **policy**, **value fonction** and/or a **model**.
- Decisions affect the **reward**, the agent state and environment state.
- The Q value allows us to find the value of an action in each state.

$$Q(s, a) = Transition\ probability * (Reward\ probability + gamma * value\ of\ next\ state)$$

0.2 Dynamic programming

Dynamic programming (**DP**) is a technique for solving complex problems. Instead of solving complex problems directly, we break the problem into simple sub-problems. DP helps in significantly minimizing the computation time. We use this approach to solve the **belman equation** usin to powerful algorithms.

- Value iteration
- Policy iteration

0.3 Value Iteration

Value Iteration start with a random value function and update it to improved value function. Once the optimal value function is found, we can easily derive an optimal policy from it.

Algorithm 1: Value Iteration

Input: MDP, small positive number θ
Output: policy $\pi \approx \pi_*$
Initialize V arbitrarily (e.g., $V(s) = 0$ for all $s \in \mathcal{S}^+$)
repeat
    $\Delta \leftarrow 0$
    for $s \in \mathcal{S}$ **do**
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s, a)(r + \gamma V(s'))$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end
until $\Delta < \theta$;
$\pi \leftarrow$ **Policy_Improvement**(MDP, V)
return π

- $p(s', r|s, a)$: probability of next state s' and reward r, given current state s and current action a ($\mathbb{P}(S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a)$)

Algorithm 2: Policy Improvement

Input: MDP, value function V
Output: policy π'
for $s \in \mathcal{S}$ **do**
  for $a \in \mathcal{A}(s)$ **do**
    $Q(s,a) \leftarrow \sum_{s' \in \mathcal{S}, r \in \mathcal{R}} p(s', r|s,a)(r + \gamma V(s'))$
  end
  $\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}(s)} Q(s,a)$
end
return π'

0.4 Policy Iteration

In contrast to value iteration, in policy iteration, we start with the random policy, then we find the value function of that policy; if the value function is not optimal, we find the new improved policy. We repeat this process until we find the optimal policy.

Algorithm 3: Policy Iteration

1. Initialisation
$V(s) \in \mathbb{R}$, (e.g $V(s) = 0$) and $\pi(s) \in A$ for all $s \in S$,
$\Delta \leftarrow 0$
2. Policy Evaluation
while $\Delta \geq \theta$ *(a small positive number)* **do**
  foreach $s \in S$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum\limits_{a} \pi(a|s) \sum\limits_{s',r} p(s', r|s,a) \left[r + \gamma V(s')\right]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end
end
3. Policy Improvement
policy-stable $\leftarrow$ *true*
foreach $s \in S$ **do**
  old-action $\leftarrow \pi(s)$ $\pi(s) \leftarrow_a \sum\limits_{s',r} p(s', r|s,a) \left[r + \gamma V(s')\right]$
  policy-stable $\leftarrow$ *old-action* $= \pi(s)$
end
if *policy-stable* return V $\approx V_*$ and $\pi \approx \pi_*$, else go to 2

0.5 Implementation

in this part the two previously mentioned algorithms are applied for the Cliff Walking problem. The cliff walking MDP consists of the following:

- States: Set of states. Here we have 4x12 states (each little square box in the grid).
- Actions: Set of all possible actions (left, right, up, down; these are all the four possible actions our agent can take in our environment).
- Transition probabilities: The probability of moving from one state to another state by performing an action a.

- Rewards probabilities: Each time step incurs -1 reward, and stepping into the cliff incurs -100 reward and a reset to the start. An episode terminates when the agent reaches the goal.

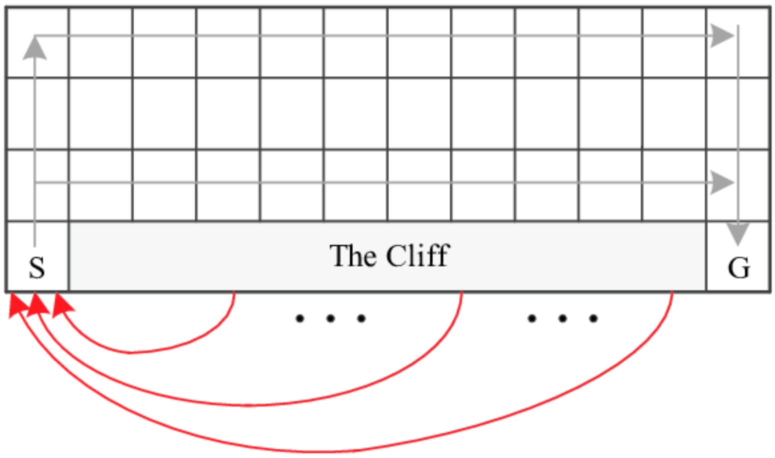

Figure 1: the Cliff Walking Environment Adapted from [1]

Bibliography

[1] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: `http://incompleteideas.net/book/the-book-2nd.html`.