



Reinforcement learning Course

Instructor: Prof. Mohammad Hossein Rohban

2025

Recitation second

Presenter: Reza Ghaderi

Table of contents

- Markov Process
- Markov Reward Process
- Markov Decision Process
- Graphical Models
- Bellman Equations
- Value Iteration
- Policy Iteration
- First-Visit MC
- Every-Visit MC
- Bootstrapping
- TD Error
- N-Step TD
- Epsilon-Greedy Exploration
- SARSA
- Q-Learning

Markov Process (MP) & Markov Property

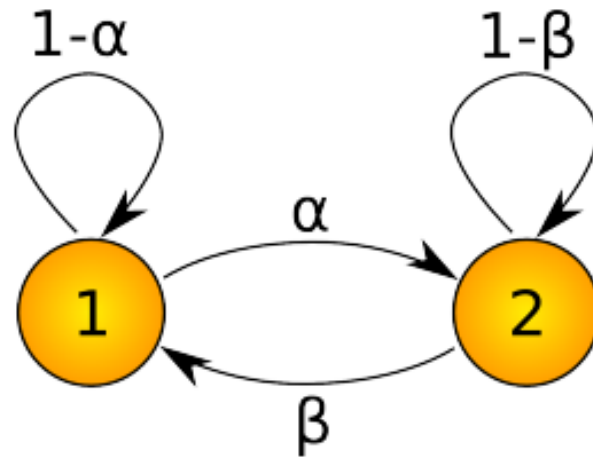
- Markov Process: Set of random variables(states) that have Markov Property
- Markov Property: Future states only depend on the current state.
- Example:

if today was rainy, tomorrow is sunny

- In this statement, Only current state(today) is considered for predicting the future.
- A better statement would be: if today was sunny, with the probability of 0.8 tomorrow is sunny and with probability of 0.2 is rainy. And:
- if today was rainy, with the probability of 0.9 tomorrow is rainy and with probability of 0.1 is sunny.

Transition Probability

- The mentioned chances in the previous examples are called transition probability.
- A better definition for MP: A set of states the have Markov property and with a transition probability they transform to each other



Markov Reward Process (MRP)

- In MP we are only monitoring, and we have no effect on the world.
- If we set a reward for moving from a state to another state (still monitoring, we don't make the move), now this is called MRP.

Markov Decision Process(MDP)

- Now if we make the decisions to make the actions in each state, It is called MDP.
- From a state, we take an action, move to a next state and receive a reward.
- receiving reward could happen after taking the action, or after taking action and moving to a certain next state.

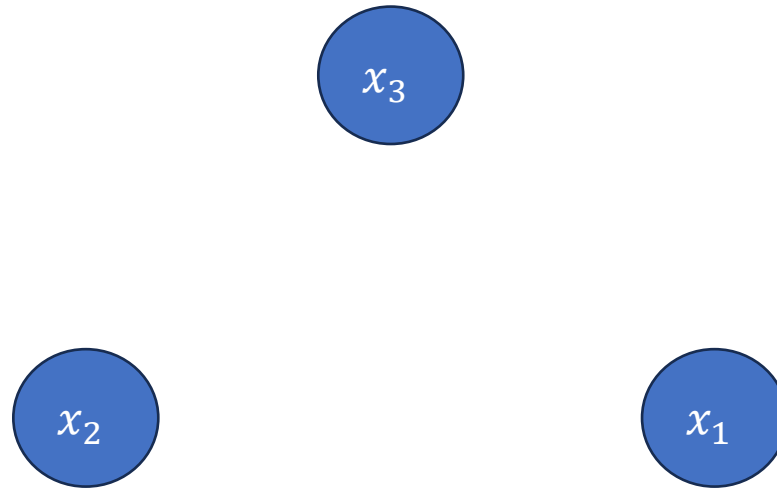
Graphical Models

- Graphical Model is a visualization tool to represent the probabilistic relation between random variables.
- Assume x_1, x_2, x_3
- $P(x_1, x_2, x_3) = P(x_1|x_2, x_3)P(x_2|x_3)P(x_3)$

How to Draw its Graphical Model?

$$P(x_1, x_2, x_3) = P(x_1|x_2, x_3)P(x_2, x_3) = P(x_1|x_2, x_3)P(x_2|x_3)P(x_3)$$

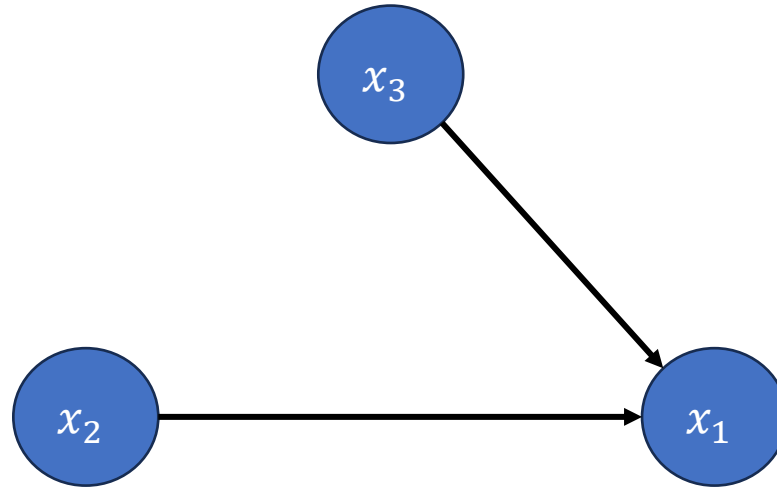
- First draw the random variables as nodes nodes:



How to Draw its Graphical Model?

$$P(x_1, x_2, x_3) = P(x_1|x_2, x_3)P(x_2, x_3) = P(x_1|x_2, x_3)P(x_2|x_3)P(x_3)$$

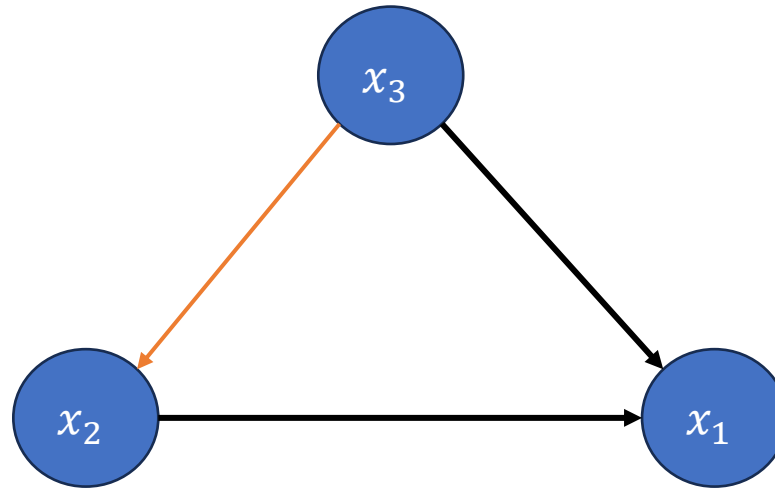
- $P(x_1|x_2, x_3)$ since x_1 is conditioned on x_2 and x_3 , we draw two edges from x_3 and x_2 to x_1 .



How to Draw its Graphical Model?

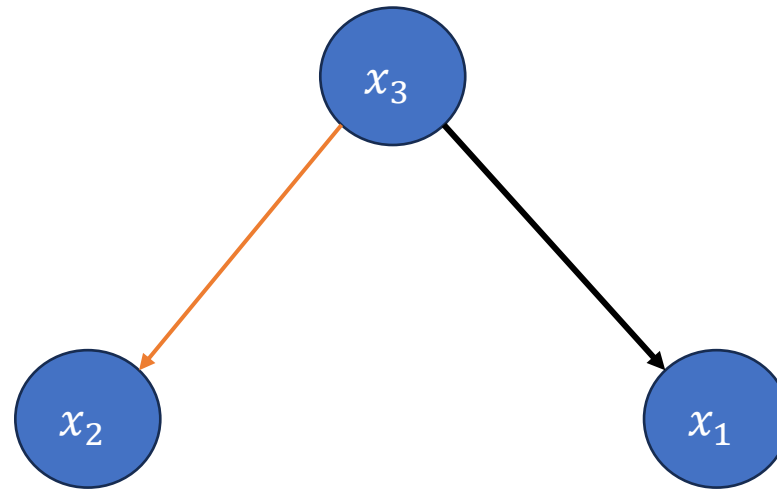
$$P(x_1, x_2, x_3) = P(x_1|x_2, x_3)P(x_2, x_3) = P(x_1|x_2, x_3)P(x_2|x_3)P(x_3)$$

- $P(x_2|x_3)$, since x_2 is conditioned on x_3 , we draw two edges from x_3 and x_2 .



What if x_1 is independent of x_2 given x_3 ?

Then we have:



Visualizing MDPs using Graphical Models

- $P(s_0, a_0, s_1, a_1, s_2) = P(s_2 | s_0, a_0, s_1, a_1)P(a_1 | s_0, a_0, s_1)P(s_1 | s_0, a_0) \times P(a_0 | s_0)P(s_0)$

Due to Markovian Property: Next State only depends on current state, We have:

$$P(s_2 | s_0, a_0, s_1, a_1) = P(s_2 | s_1, a_1)$$

Taking an action only depends on current state not the past states, We have:

$$P(a_1 | s_0, a_0, s_1) = P(a_1 | s_1)$$

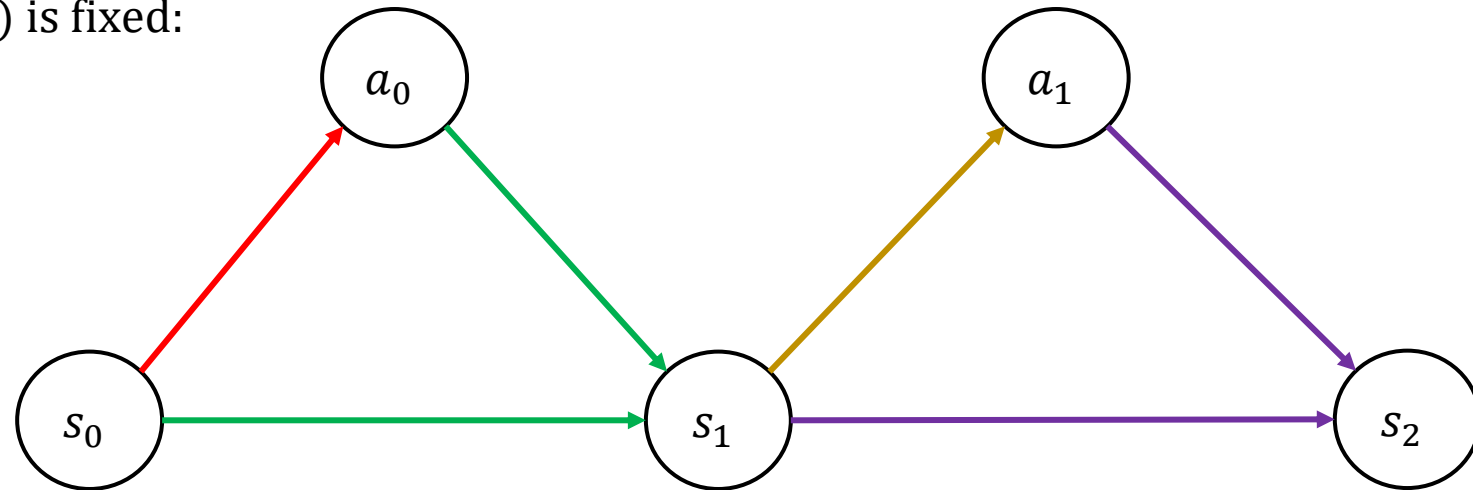
Visualizing MDPs using Graphical Models

- Thus we have:

$$P(s_0, a_0, s_1, a_1, s_2) = P(s_2 | s_1, a_1) P(a_1 | s_1) P(s_1 | s_0, a_0) P(a_0 | s_0) P(s_0)$$

Rearranging and using $\pi(a|s)$ instead of $P(a|s)$: $P(s_0) \pi(a_0 | s_0) P(s_1 | s_0, a_0) \pi(a_1 | s_1) P(s_2 | s_1, a_1)$

Assuming that the $P(s_0)$ is fixed:

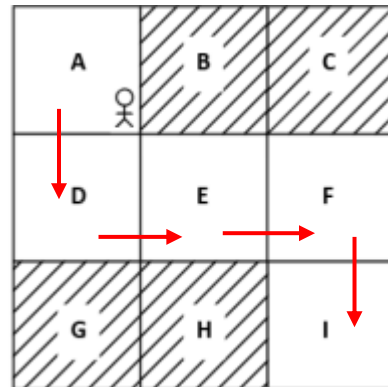


State Value Functions

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

Example: In State A, With the probability of 0.8 we go right and 0.2 we go down

For the first trajectory we have:



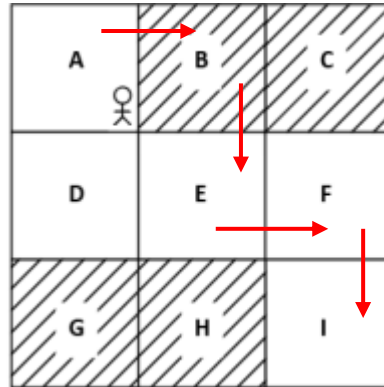
$$V(A) = R(\tau_1) = 1 + 1 + 1 + 1 = 4$$



State Value Functions

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s]$$

For the Second trajectory we have:



$$V(A) = R(\tau_2) = -1 + 1 + 1 + 1 = 2$$



State Value Functions

Calculating the $V(A)$ according to the given Trajectories:

$$\begin{aligned}V^\pi(A) &= \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = A] \\&= \sum_i R(\tau_i) \pi(a_i | A) \\&= R(\tau_1) \pi(\text{down} | A) + R(\tau_2) \pi(\text{right} | A) \\&= 4(0.8) + 2(0.2) \\&= 3.6\end{aligned}$$

Optimal State Value Functions

(i) The optimal state-value function ... $\mapsto v_*(s) = \max_{\pi} v_{\pi}(s), \forall s \in S$ (a) ... is the state-value function with the highest value across all policies.

Summary



SHOW ME THE MATH

The state-value function V

(1) The value of a state $s \dots$ (3) \dots is the expectation over $\pi \dots$

(2) \dots under policy $\pi \dots$ $v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s]$ (5) \dots given you select state s at time step t .

(4) \dots of returns at time step $t \dots$

(6) Remember that returns are the sum of discounted rewards. \rightarrow

$$v_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

(7) And we can define them recursively like so. \rightarrow

$$v_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma v_{\pi}(S_{t+1}) | S_t = s]$$

State-Action value function



SHOW ME THE MATH

The action-value function Q

(1) The value of action a in state s under policy π ...

(2) ... is the expectation of returns, given we select action a in state s and follow policy π thereafter.

$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[G_t | S_t = s, A_t = a]$$

(3) We can define this equation recursively like so:

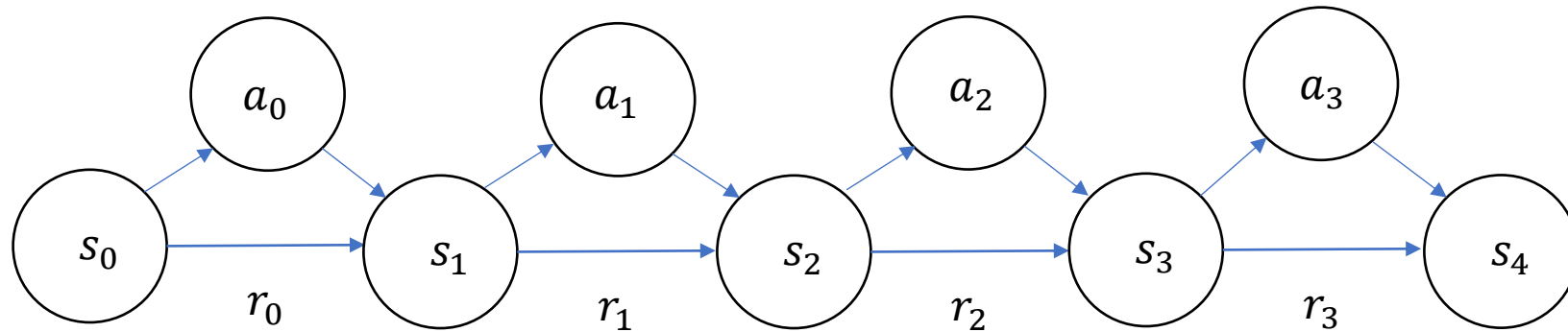
$$q_{\pi}(s, a) = \mathbb{E}_{\pi}[R_t + \gamma G_{t+1} | S_t = s, A_t = a]$$

(3) Likewise, the optimal action-value function is the action-value function with the highest values.

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a), \forall s \in S, \forall a \in A(s)$$

Bellman equations- Value Function

$$V(s) = R(s, a, s') + \gamma V(s')$$



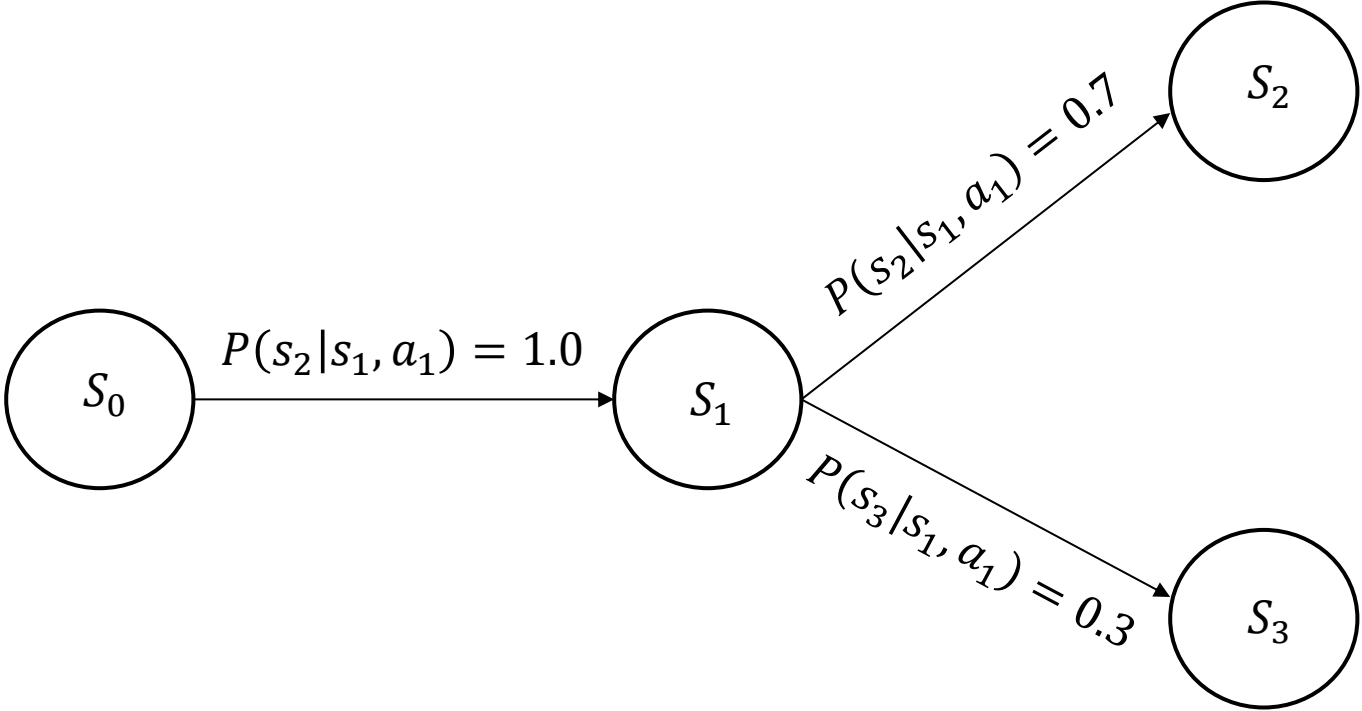
$$V(s_2) = R(s_2, a_2, s_3) + \gamma V(s_3)$$

Bellman equations- Value Function

- What if the Environment is Stochastic?

$$V^\pi(s) = \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma * V^\pi(s')]$$

How to Calculate values if we have this?



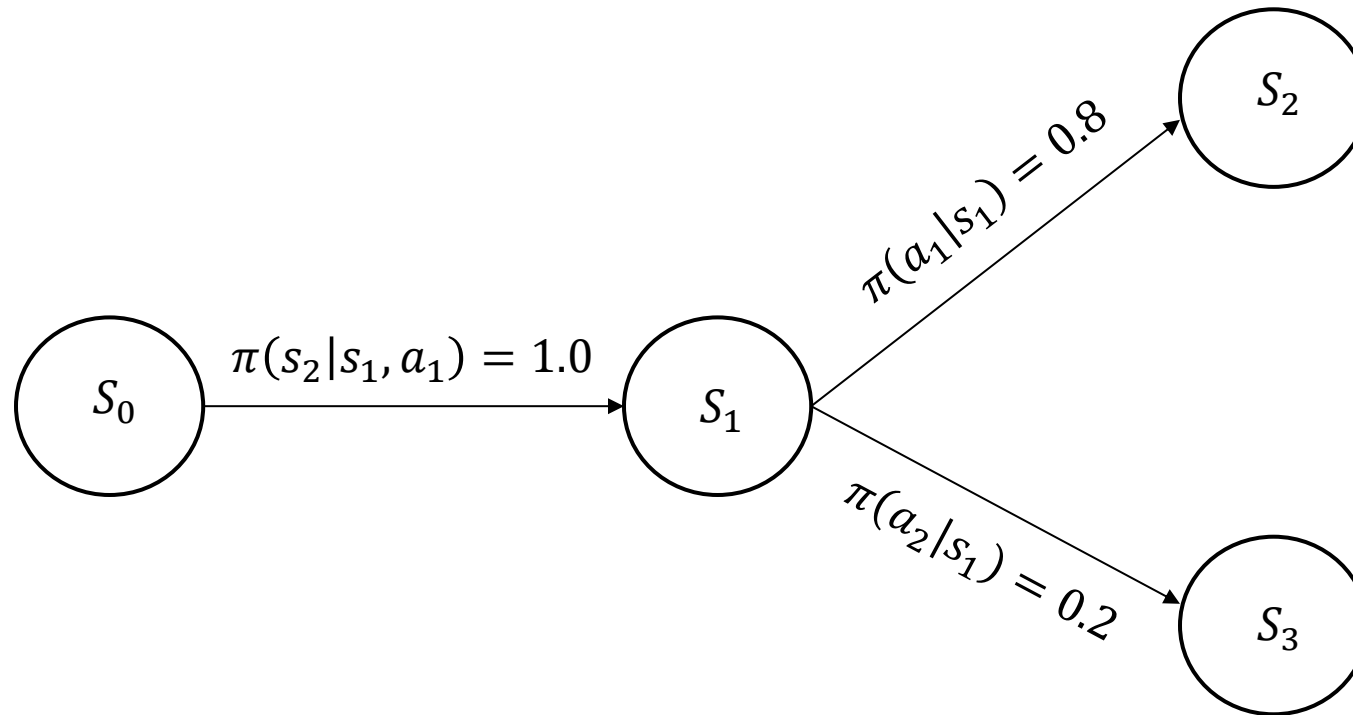
Then we have

$$V(s_1) = 0.7[R(s_1, a_1, s_2) + \gamma * V(s_3)] + 0.3[R(s_1, a_1, s_3) + \gamma * V(s_3)]$$

$$V(s_1) = P(s_2|s_1, a_1)[R(s_1, a_1, s_2) + \gamma * V(s_3)] + P(s_3|s_1, a_1)[R(s_1, a_1, s_3) + \gamma * V(s_3)]$$

What if the policy is stochastic?


- Write the bellman equation for this MDP



Take another Expectation!

$$V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma V^\pi(s')]$$

Summary

 **SHOW ME THE MATH**
The state-value function V

(1) The value of a state $s \dots$ (3) \dots is the expectation over $\pi \dots$

(a) \dots under policy $\pi \dots$ $v_{\pi}(s) = \mathbb{E}_{\pi}[G_t | S_t = s]$ (5) \dots given you select state s at time step t .

(4) \dots of returns at time step $t \dots$

(6) Remember that returns are the sum of discounted rewards. \rightarrow

$$v_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots | S_t = s]$$

(7) And we can define them recursively like so. \rightarrow $v_{\pi}(s) = \mathbb{E}_{\pi}[R_{t+1} + \gamma G_{t+1} | S_t = s]$

(8) This equation is called the Bellman equation, and it tells us how to find the value of states.

$$v_{\pi}(s) = \sum_a \pi(a|s) \sum_{s', r} p(s', r | s, a) [r + \gamma v_{\pi}(s')], \forall s \in S$$

(9) We get the action (or actions, if the policy is stochastic) prescribed for state s and do a weighted sum. \rightarrow

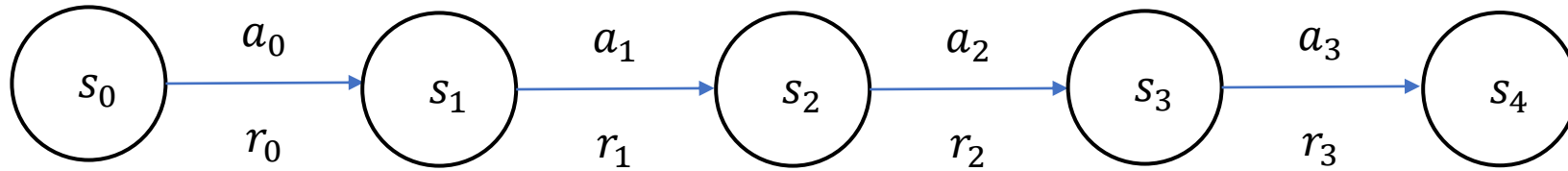
(10) We also weight the sum over the probability of next states and rewards. \rightarrow

(11) We add the reward and the discounted value of the landing state, then weight that by the probability of that transition occurring. \rightarrow

(12) Do this for all states in the state space. \rightarrow

Bellman Equations Q-Functions

$$Q(s, a) = R(s, a, s') + \gamma * Q(s', a')$$

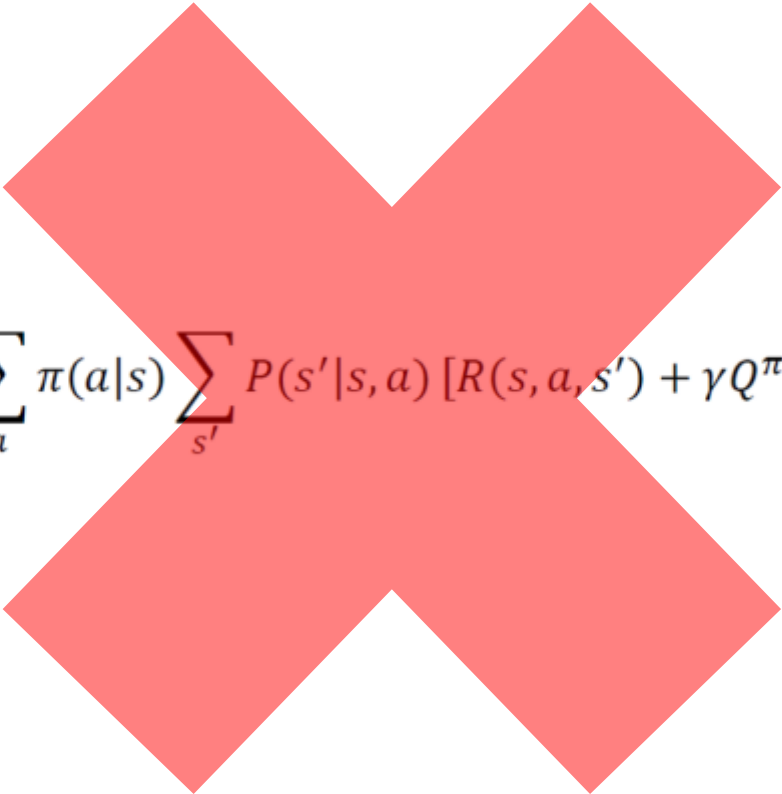


$$Q(s_2, a_2) = R(s_2, a_2, s_3) + \gamma * Q(s_3, a_3)$$

What if We have Stochastic Environment

$$Q^\pi(s, a) = \sum_{s'} P(s'|s, a)[R(s, a, s') + \gamma Q^\pi(s', a')]$$

Stochastic Policy?

$$Q^\pi(s, a) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a) [R(s, a, s') + \gamma Q^\pi(s', a')]$$


But We need Another Expectation

- Two Points
 - When We choose the action, Since the action is fix, The Expectation is not needed. (we don't take Expectation w.r.t a)
 - But the a' is Stochastic So we have to take Expectation

Q function for Stochastic Environment and Policy

$$Q^\pi(s, a) = \sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma \sum_{a'} \pi(a'|s') Q^\pi(s', a') \right]$$

$$Q^\pi(s, a) = \mathbb{E}_{s' \sim P} [R(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi} Q^\pi(s', a')]]$$

Summary



SHOW ME THE MATH

The Bellman optimality equations

(1) The optimal state-value function ...

$$v_*(s) = \max_{\pi} v_{\pi}(s), \forall s \in \mathcal{S}$$

(2) ... is the state-value function with the highest value across all policies.

(3) Likewise, the optimal action-value function is the action-value function with the highest values.

$$q_*(s, a) = \max_{\pi} q_{\pi}(s, a), \forall s \in \mathcal{S}, \forall a \in A(s)$$

(4) The optimal state-value function can be obtained this way.

$$v_*(s) = \max_a \sum_{s', r} p(s', r | s, a) [r + \gamma v_*(s')]$$

(5) We take the max action ...

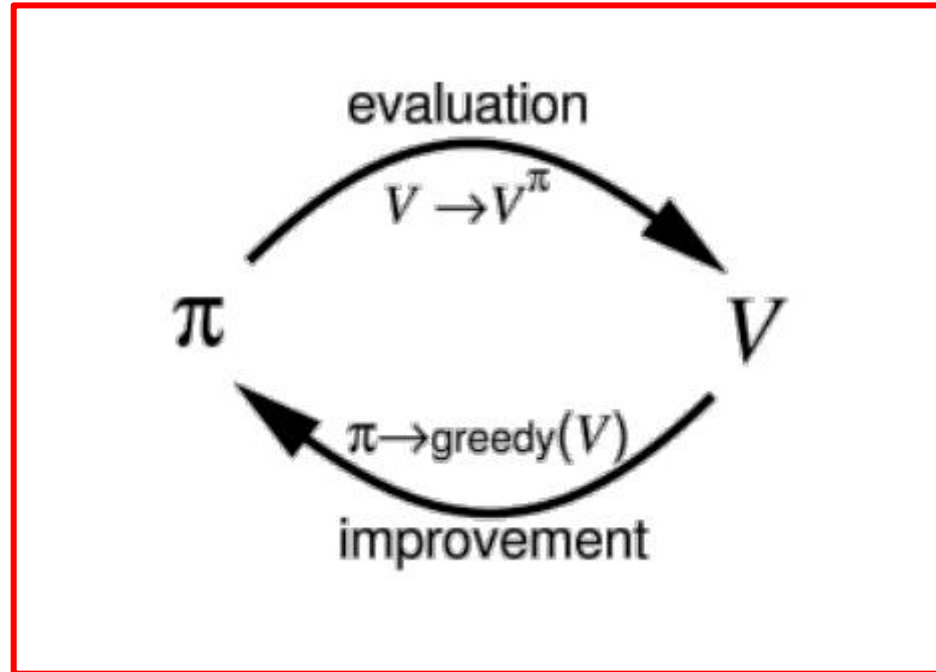
(6) ... of the weighted sum of the reward and discounted optimal value of the next state.

(7) Similarly, the optimal action-value function can be obtained this way.

$$q_*(s, a) = \sum_{s', r} p(s', r | s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]$$

(8) Notice how the max is now on the inside.

Policy Iteration



Policy Iteration Pseudocode

1. **Initialize a policy** π_0 (for example, pick actions at random).
2. **Policy Evaluation:** Given the current policy π_k , compute the value function V^{π_k} . In a tabular setting, this means either

- solving the linear system for V^{π_k} , or
- iterating the Bellman equation for the current policy until convergence:

$$V^{\pi_k}(s) \leftarrow \sum_{s'} P(s' | s, \pi_k(s)) [R(s, \pi_k(s), s') + \gamma V^{\pi_k}(s')].$$

3. **Policy Improvement:** Use the newly computed V^{π_k} to get

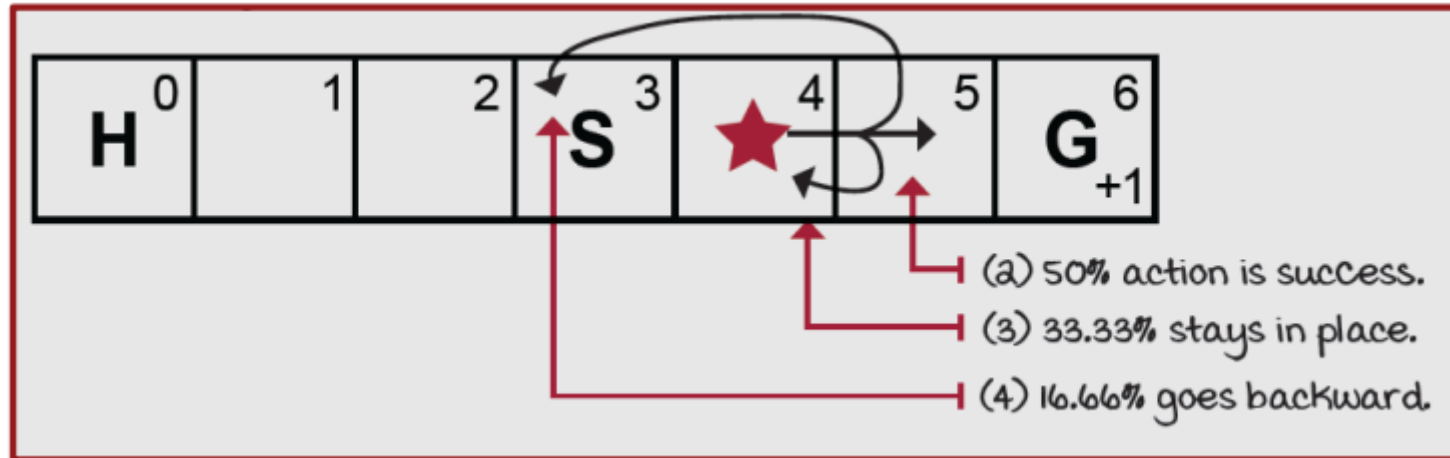
$$Q^{\pi_k}(s, a) = \sum_{s'} P(s' | s, a) [R(s, a, s') + \gamma V^{\pi_k}(s')],$$

and then update the policy:

$$\pi_{k+1}(s) = \arg \max_a Q^{\pi_k}(s, a).$$

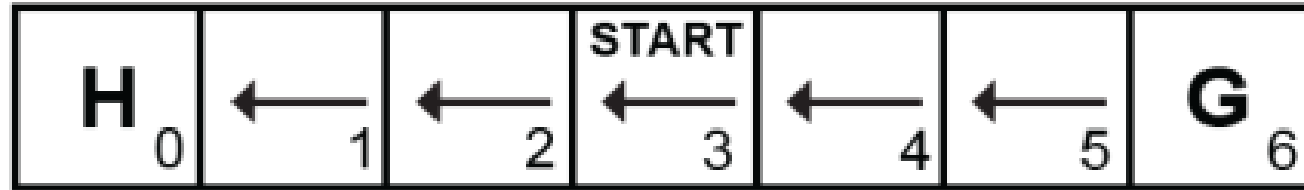
4. **Repeat** until the policy no longer changes (i.e., $\pi_{k+1} = \pi_k$).

SWF Environment



Example

- Given the following policy, Calculate State 3 value.




- Which State Will get updated first?

Policy Evaluation

$$\begin{aligned}v_1^{\pi}(5) &= p(s' = 4 \mid s = 5, a = \text{Left}) * [R(5, \text{Left}, 4) + v_0^{\pi}(4)] + \\ & \quad p(s' = 5 \mid s = 5, a = \text{Left}) * [R(5, \text{Left}, 5) + v_0^{\pi}(5)] + \\ & \quad p(s' = 6 \mid s = 5, a = \text{Left}) * [R(5, \text{Left}, 6) + v_0^{\pi}(6)]\end{aligned}$$

$$v_1^{\pi}(5) = 0.50 * (0+0) + 0.33 * (0+0) + 0.166 * (1+0) = 0.166$$

(4)  Yep, this is the value of state 5 after 1 iteration of policy evaluation ($v_1^{\pi}(5)$).

Policy Evaluation

$$V(4) = P(s' = 3, a = L)[r + V(3)] + P(s' = 4, a = L)[r + V(4)] + P(s' = 5, a = L)[r + V(5)]$$

$$V(4) = 0.5[r + 0] + 0.3333[0 + 0] + 0.1666[0 + 0] = 0.0$$

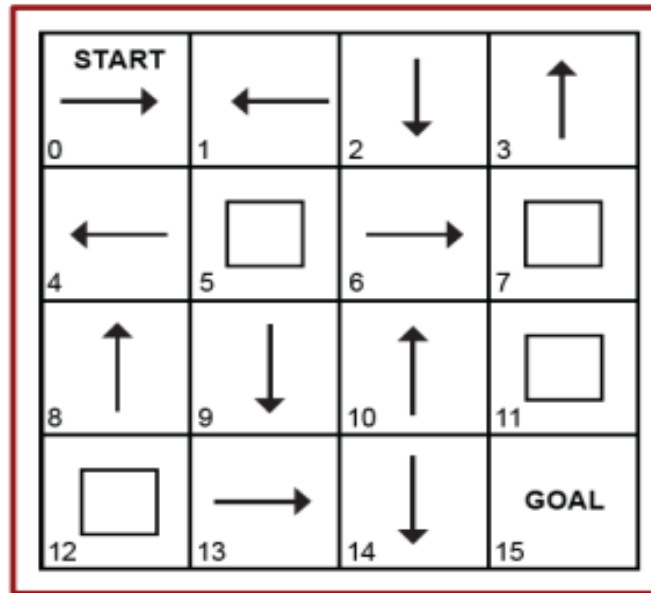
Note: the values will get updated after a full iteration over the Environment so the value of the other states remains Zero.

Convergence After 104 iterations!

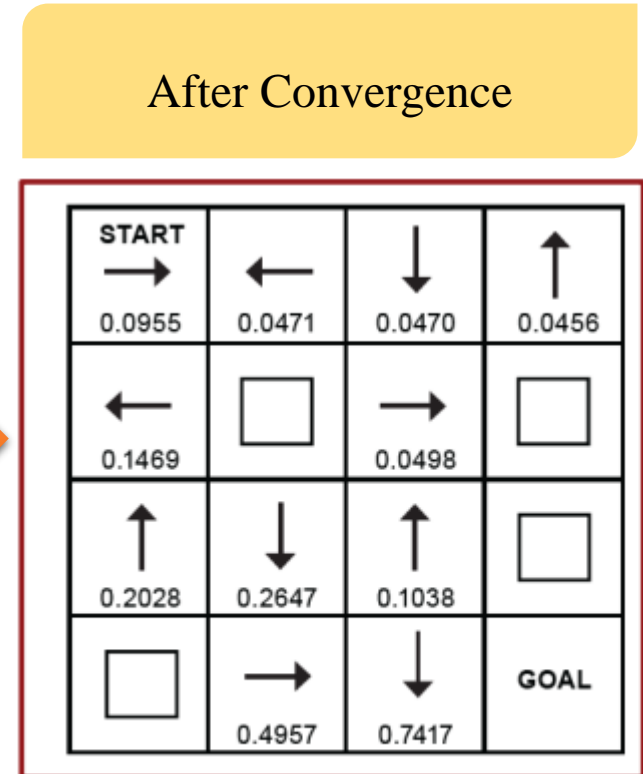
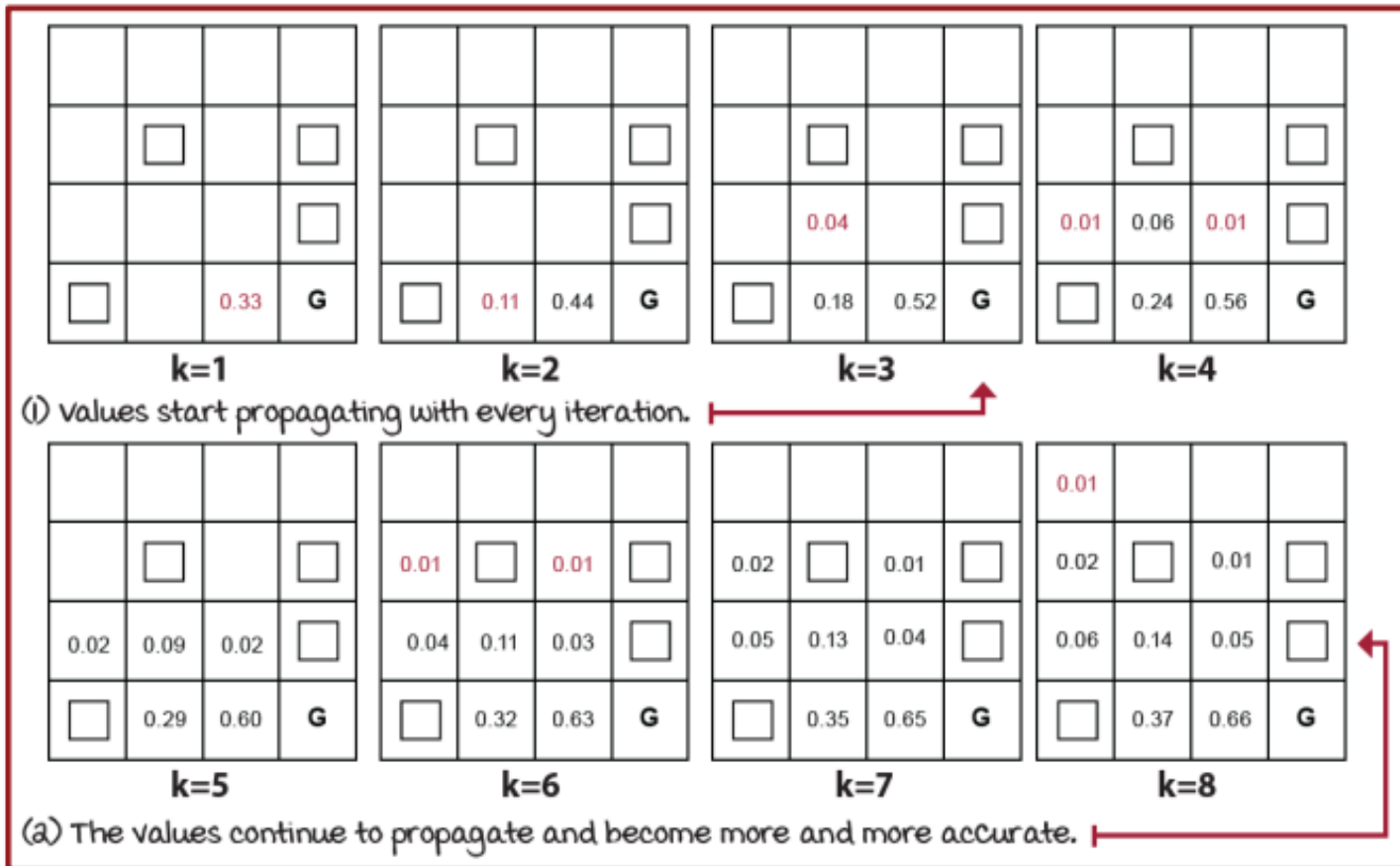
| k | $V^n(0)$ | $V^n(1)$ | $V^n(2)$ | $V^n(3)$ | $V^n(4)$ | $V^n(5)$ | $V^n(6)$ |
|-----|----------|----------|----------|----------|----------|----------|----------|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0.1667 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0.0278 | 0.2222 | 0 |
| 3 | 0 | 0 | 0 | 0.0046 | 0.0463 | 0.2546 | 0 |
| 4 | 0 | 0 | 0.0008 | 0.0093 | 0.0602 | 0.2747 | 0 |
| 5 | 0 | 0.0001 | 0.0018 | 0.0135 | 0.0705 | 0.2883 | 0 |
| 6 | 0 | 0.0003 | 0.0029 | 0.0171 | 0.0783 | 0.2980 | 0 |
| 7 | 0 | 0.0006 | 0.0040 | 0.0202 | 0.0843 | 0.3052 | 0 |
| 8 | 0 | 0.0009 | 0.0050 | 0.0228 | 0.0891 | 0.3106 | 0 |
| 9 | 0 | 0.0011 | 0.0059 | 0.0249 | 0.0929 | 0.3147 | 0 |
| 10 | 0 | 0.0014 | 0.0067 | 0.0267 | 0.0959 | 0.318 | 0 |
| ... | ... | ... | ... | ... | ... | ... | ... |
| 104 | 0 | 0.0027 | 0.011 | 0.0357 | 0.1099 | 0.3324 | 0 |

Another Example

- Given The following policy, Which State Gets updated First?



Policy Evaluation



Policy Improvement

- Take the action that makes the $Q(s,a)$ the maximum:

$$\pi(s, a) = \begin{cases} 1, & \text{if } a = \arg \max_a Q(s, a) \\ 0, & \text{otherwise;} \end{cases}$$

How to Calculate Q? We only have Values!

Write the Q in terms of V



$$Q(s, a) = \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V(s')]$$

| | | | | |
|-------|---|---|---|---|
| START | ← | ↑ | ↑ | ↑ |
| ← | □ | ↑ | □ | |
| ↑ | ↓ | ← | □ | |
| □ | → | → | G | |

| | | | |
|---------------------------|---------------------------|---------------------------|-------------------|
| START | 0.38 | 0.35 | 0.34 |
| 0.39 0.41 0.40 0.40 | 0.26 0.24 0.25 | 0.28 0.27 0.28 | 0.23 0.23 0.23 |
| 0.27 0.42 0.28 0.29 | □ | 0.12 0.26 0.26 0.14 | □ |
| 0.45 0.29 0.30 0.31 | 0.29 0.34 0.34 0.48 | 0.2 0.43 0.27 0.39 | □ |
| □ | 0.39 0.35 0.59 0.43 | 0.67 0.57 0.71 0.76 | GOAL |

| | | | | |
|-------|---|---|---|---|
| START | ← | ↑ | ↑ | ↑ |
| ← | □ | ← | □ | |
| ↑ | ↓ | ← | □ | |
| □ | → | ↓ | G | |

How to Calculate Q? We only have Values!

Write the Q in terms of V



$$Q(s, a) = \sum_{s'} P_{SS'}^a [R_{SS'}^a + \gamma V(s')]$$



$Q(s,a)$

1st iteration

| | | | | | | | | | | | | | |
|--------|-----|-----|-----|-----|-------|-----|-----|-----|-----|-----|------|------|--------|
| H 0 | 0.0 | 0.0 | 0.0 | 0.0 | START | | 0.0 | 0.0 | 0.0 | 0.0 | 0.17 | 0.56 | G 6 |
| | | | | | 0.0 | 0.0 | | | | | | | |

2nd iteration

| | | | | | | | | | | | | | |
|--------|-----|-----|-----|-----|-------|-----|-----|-----|-----|------|------|------|--------|
| H 0 | 0.0 | 0.0 | 0.0 | 0.0 | START | | 0.0 | 0.0 | 0.0 | 0.01 | 0.18 | 0.58 | G 6 |
| | | | | | 0.0 | 0.0 | | | | | | | |

...

104th iteration

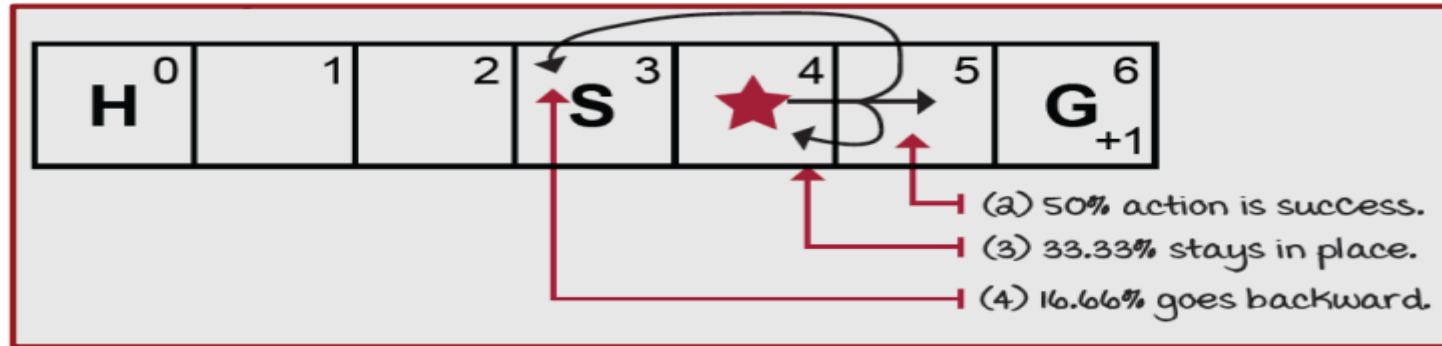
| | | | | | | | | | | | | | |
|--------|-----|-----|-----|-----|-------|------|------|------|------|------|------|------|--------|
| H 0 | 0.0 | 0.0 | 0.0 | 0.0 | START | | 0.01 | 0.01 | 0.03 | 0.04 | 0.24 | 0.63 | G 6 |
| | | | | | 0.01 | 0.01 | | | | | | | |

Policy iteration in a Nutshell

$$\pi_0 \rightarrow V^{\pi_0} \rightarrow \pi_1 \rightarrow V^{\pi_1} \rightarrow \pi_2 \rightarrow V^{\pi_2} \rightarrow \pi_3 \rightarrow V^{\pi_3} \rightarrow \dots \rightarrow \pi^* \rightarrow V^{\pi^*}$$

Value Iteration

- SWF environment



Original Method-Mentioned in the class

Algorithm:

Start with $V_0^*(s) = 0$ for all s .

For $k = 1, \dots, H$:

For all states s in S :

$$V_k^*(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma V_{k-1}^*(s'))$$

$$\pi_k^*(s) \leftarrow \arg \max_a \sum_{s'} P(s'|s, a) (R(s, a, s') + \gamma V_{k-1}^*(s'))$$

This is called a **value update** or **Bellman update/back-up**

For SWF Environment

- Initial $V(s) = 0$
- First iteration
- $V_k^*(s) \leftarrow \max_a \sum_{s'} P_{ss'} (r + \gamma V(s'))^*_{k-1}$

$$P_{56}^R (r + V_0(6)) + P_{54}^R (r + V_0(4)) + P_{55}^R (r + V_0(5)) = 0.5(1 + 0) + 0.1666(0 + 0) + 0.3333(0 + 0) = 0.5$$

$$P_{56}^L (r + V_0(6)) + P_{54}^L (r + V_0(4)) + P_{55}^L (r + V_0(5)) = 0.1666(1 + 0) + 0.5(0 + 0) + 0.3333(0 + 0) = 0.1666 \cong 0.17$$



$$V_1^*(5) = \max(0.5, 0.17)$$

$$V_1^*(5) = 0.5$$

For SWF Environment

$$\pi^*(s) \leftarrow \operatorname{argmax}_a \sum_{s'} P_{ss'}(r + \gamma V(s')^*_{k-1}) \quad \longrightarrow \quad \pi_1^*(5) = \textit{Right}$$

$$P_{45}^R(r + V_0(5)) + P_{44}^R(r + V_0(4)) + P_{43}^R(r + V_0(3)) =$$

$$0.5(0 + 0) + 0.3333(0 + 0) + 0.1666(0 + 0) = 0$$

$$P_{45}^L(r + V_0(5)) + P_{44}^L(r + V_0(4)) + P_{43}^L(r + V_0(3)) =$$

$$0.1666(0 + 0) + 0.3333(0 + 0) + 0.3333(0 + 0) = 0$$

From this point we observe no change till the next iteration

Second Iteration

$$P_{56}^R(r + V_1(6)) + P_{54}^R(r + V_1(4)) + P_{55}^R(r + V_1(5)) = 0.5(1 + 0) + 0.1666(0 + 0) + 0.3333(0 + 0.5) = 0.6666$$

$$P_{56}^L(r + V_1(6)) + P_{54}^L(r + V_1(4)) + P_{55}^L(r + V_1(5)) = 0.1666(1 + 0) + 0.5(0 + 0) + 0.3333(0 + 0.5) = 0.33325 \cong 0.3333$$



$$V_2^*(5) = \max(0.6666, 0.3333)$$

$$V_2^*(5) = 0.6666$$

$$\pi_2^*(5) = \textit{Right}$$

$$P_{45}^R(r + V_1(5)) + P_{44}^R(r + V_1(4)) + P_{43}^R(r + V_1(3)) = 0.5(0 + 0.5) + 0.3333(0 + 0) + 0.1666(0 + 0) = 0.25$$

$$P_{45}^L(r + V_1(5)) + P_{44}^L(r + V_1(4)) + P_{43}^L(r + V_1(3)) = 0.1666(0 + 0.5) + 0.3333(0 + 0) + 0.3333(0 + 0) = 0.08333$$



$$V_2^*(4) = \max(0.25, 0.08333)$$

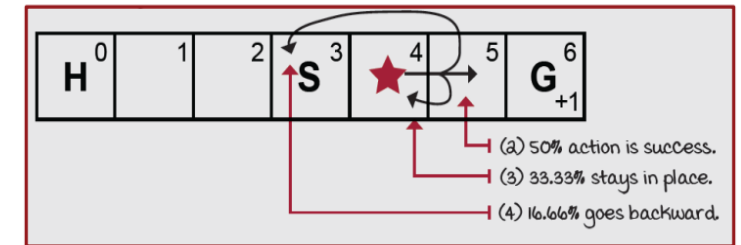
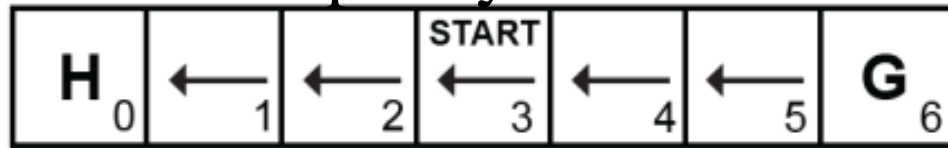
$$V_2^*(4) = 0.25$$

$$\pi_2^*(4) = \textit{Right}$$

A little bit different in terms of notation

- Initial $V_0(s) = 0$

For an initial policy:



$$Q_1(s = 5, a = R) = P_{56}^R(r + V_0(6)) + P_{54}^R(r + V_0(4)) + P_{55}^R(r + V_0(5)) =$$

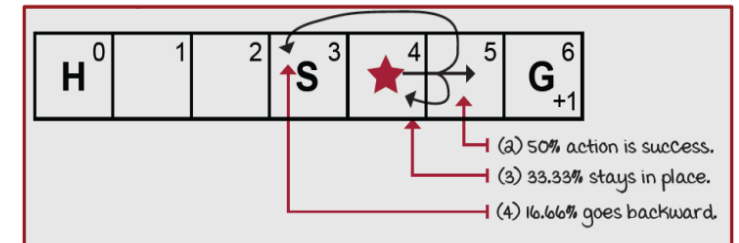
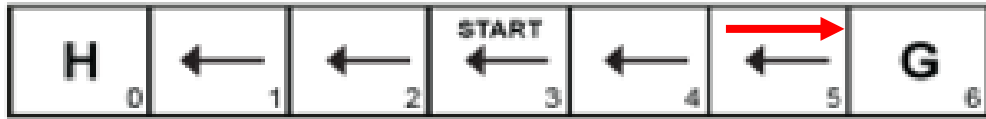
$$0.5(1 + 0) + 0.1666(0 + 0) + 0.3333(0 + 0) = 0.5$$

$$Q_1(s = 5, a = L) = P_{56}^L(r + V_0(6)) + P_{54}^L(r + V_0(4)) + P_{55}^L(r + V_0(5)) =$$

$$0.1666(1 + 0) + 0.5(0 + 0) + 0.3333(0 + 0) = 0.1666 \cong 0.17$$

Second Iteration

- Updating the Policy based on Q values
- Recalculating V using new Policy



$$V_1(5) = P(s' = 6, a = R)[r + V_0(6)] + P(s' = 5, a = R)[r + V_0(5)] + P(s' = 4, a = R)[r + V_0(4)]$$

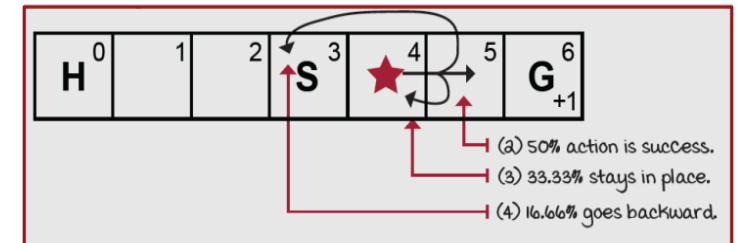
$$V_1(5) = 0.5[1 + 0] + 0.3333[0 + 0] + 0.1666[0 + 0] = 0.5$$

$$V_1(4) = P(s' = 5, a = L)[r + V_0(5)] + P(s' = 4, a = L)[r + V_0(4)] + P(s' = 3, a = L)[r + V_0(3)]$$

$$V_1(4) = 0.5[0 + 0] + 0.3333[0 + 0] + 0.1666[0 + 0] = 0.0$$

Second Iteration

- Updating Q Values Based on new V



$$Q_2(s = 5, a = R) = P_{56}^R(r + V_1(6)) + P_{55}^R(r + V_1(5)) + P_{54}^R(r + V_1(4)) =$$

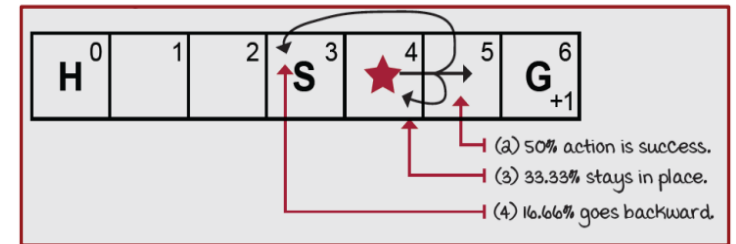
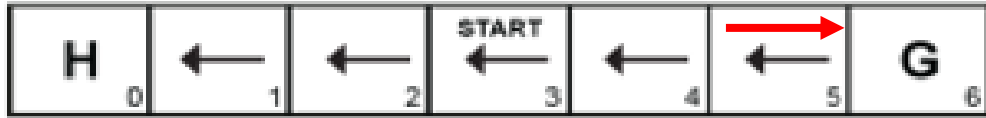
$$0.5(1 + 0) + 0.3333(0 + 0.5) + 0.1666(0 + 0) = 0.6665 \cong 0.67$$

$$Q_2(s = 5, a = L) = P_{56}^L(r + V_1(6)) + P_{54}^L(r + V_1(4)) + P_{55}^L(r + V_1(5)) =$$

$$0.1666(1 + 0) + 0.5(0 + 0) + 0.3333(0 + 0.5) = 0.3332 \cong 0.33$$

Second Iteration

- Updating Q Values Based on new V



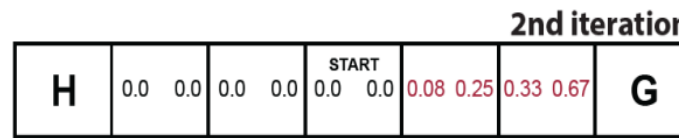
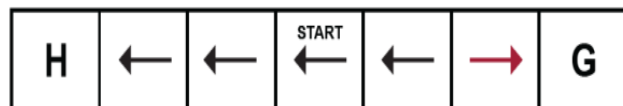
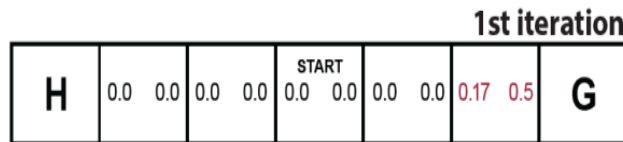
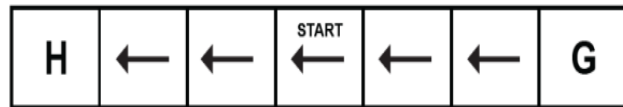
$$Q_2(s = 4, a = R) = P_{45}^R(r + V_1(5)) + P_{44}^R(r + V_1(4)) + P_{43}^R(r + V_1(3)) =$$

$$0.5(0 + 0.5) + 0.3333(0 + 0) + 0.1666(0 + 0) = 0.25$$

$$Q_2(s = 4, a = L) = P_{45}^L(r + V_1(5)) + P_{44}^L(r + V_1(4)) + P_{43}^L(r + V_1(3)) =$$

$$0.1666(0 + 0.5) + 0.3333(0 + 0) + 0.3333(0 + 0) = 0.0833 \cong 0.08$$


- Key note: in value iteration there is no need to reach the optimal Value to update the policy.
- You can update the policy with each iteration.



Sampling and Bootstrapping Methods

- Monte Carlo (MC)
 1. First visit
 2. Every visit
- Temporal Difference Learning (TD Learning)
 1. SARSA
 2. Q Learning

Monte Carlo (MC)

| | | | | |
|------------------|----------|---|----------|-----------------------|
| Start $s = 1$ | $s = 2$ | $s = 3$ | $s = 4$ | $s = 5$ |
| $s = 6$ | $s = 7$ | $s = 8$ | $s = 9$ | $s = 10$ |
| $s = 11$ | $s = 12$ | | $s = 13$ | $s = 14$ |
| $s = 15$ | $s = 16$ | | $s = 17$ | $s = 18$ |
| $s = 19$ | $s = 20$ | $R = -10$ $s = 21$  | $s = 22$ | $R = +10$ $s = 23$ |
| | | | | Goal |

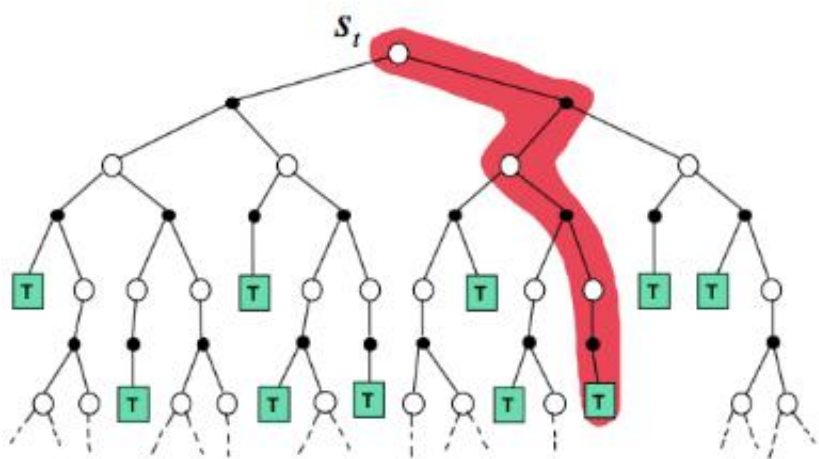
Bootstrapping

- Bootstrapping doesn't wait till the end of the episodes to update the values.
- It calculates the new immediately after receiving the reward.
- Which one of the DP and MC Does bootstrap?

Bootstrapping

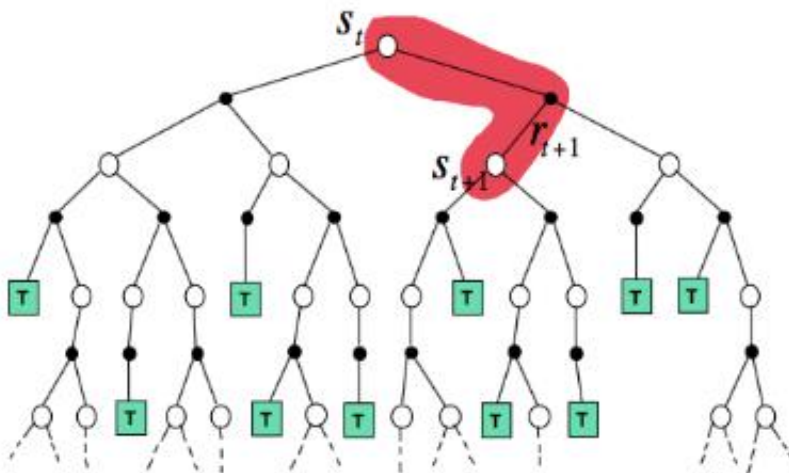
Monte-Carlo

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$



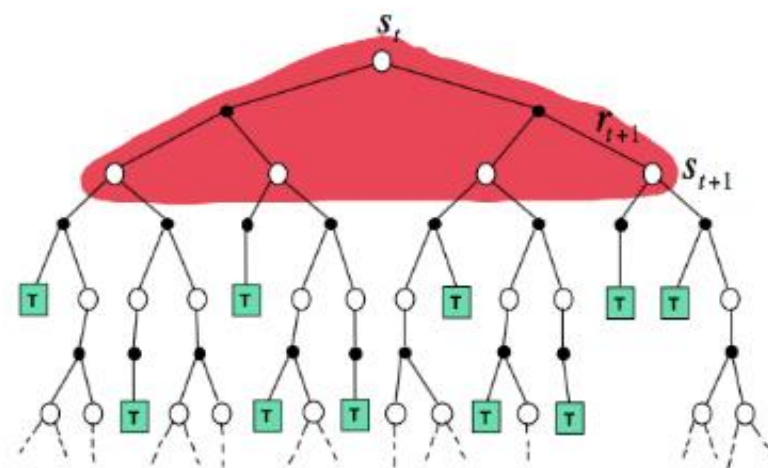
Temporal-Difference

$$V(S_t) \leftarrow V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



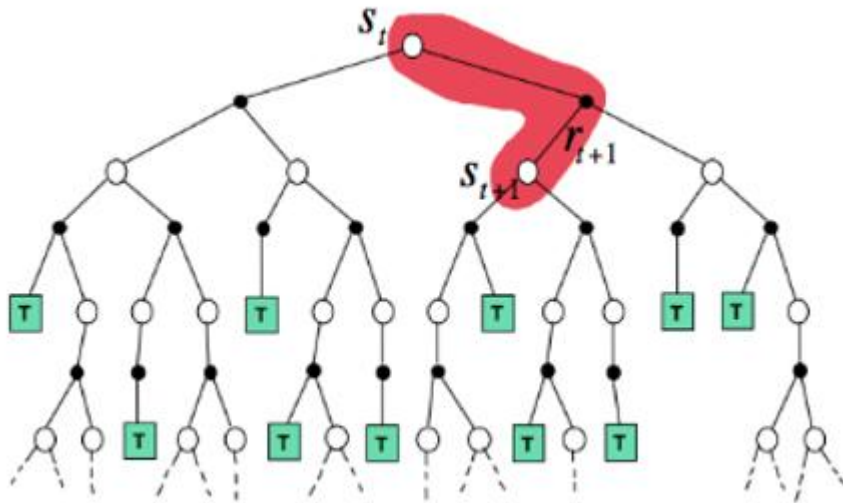
Dynamic Programming

$$V(S_t) \leftarrow \mathbb{E}_\pi [R_{t+1} + \gamma V(S_{t+1})]$$



TD Methods

- Value update formula:



$$V^{new}(s_k) = V^{old}(s_k) + \alpha \underbrace{(r_k + \gamma V^{old}(s_{k+1}) - V^{old}(s_k))}_{\text{TD Error}}$$

TD Target Estimate

TD Methods

- The Same goes for Q functions

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha [R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)]$$

Epsilon Greedy Policy

$$\pi(a|S_t) \leftarrow \begin{cases} 1 - \varepsilon + \frac{\varepsilon}{|A(S_t)|} & \text{if } a = A^* \\ \frac{\varepsilon}{|A(S_t)|} & \text{if } a \neq A^* \end{cases}$$

SARSA

Sarsa (on-policy TD control) for estimating $Q \approx q_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Loop for each step of episode:

 Take action A , observe R, S'

 Choose A' from S' using policy derived from Q (e.g., ε -greedy)

$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$

$S \leftarrow S'; A \leftarrow A';$

 until S is terminal

Q Learning

Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$

Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

Loop for each episode:

 Initialize S

 Loop for each step of episode:

 Choose A from S using policy derived from Q (e.g., ε -greedy)

 Take action A , observe R, S'

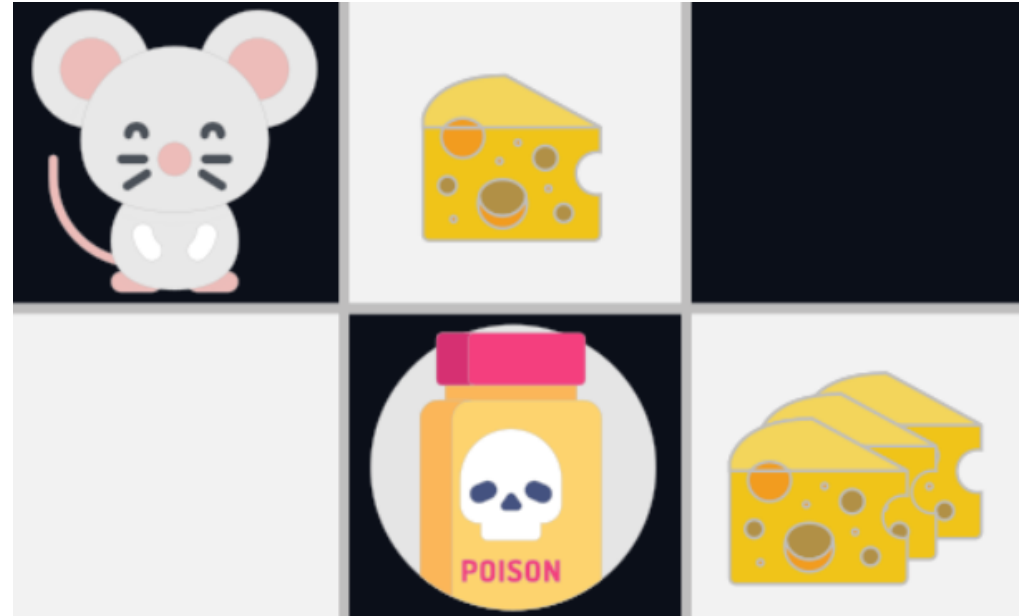
$Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$

$S \leftarrow S'$

 until S is terminal











Q Learning Example

- **+0:** Going to a state with no cheese in it.
- **+1:** Going to a state with a small cheese in it.
- **+10:** Going to the state with the big pile of cheese.
- **-10:** Going to the state with the poison and thus dying.
- **+0:** If we take more than five steps.



Step 1

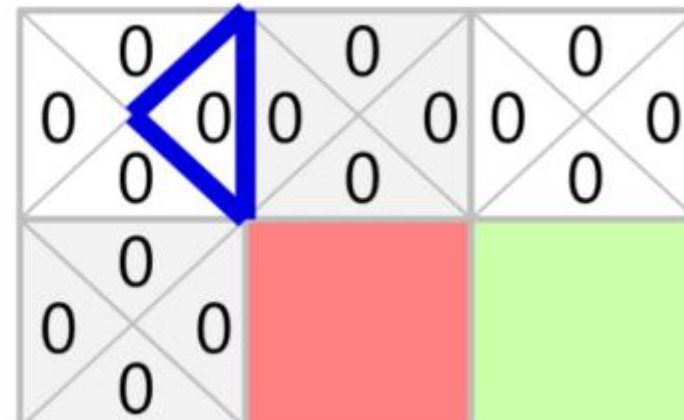
Initialize Q arbitrarily (e.g., $Q(s, a) = 0$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$, and $Q(\text{terminal-state}, \cdot) = 0$)

| |  |  |  |  |
|---|--|---|---|---|
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |

Step 2: Choose an action using the Epsilon Greedy Strategy

- Because epsilon is big (= 1.0), I take a random action. In this case, I go right.

Choose action A_t using policy derived from Q (e.g., ϵ -greedy)













Step 3: Perform action A_t , get R_{t+1} and S_{t+1}



Step 4: Update $Q(S_t, A_t)$

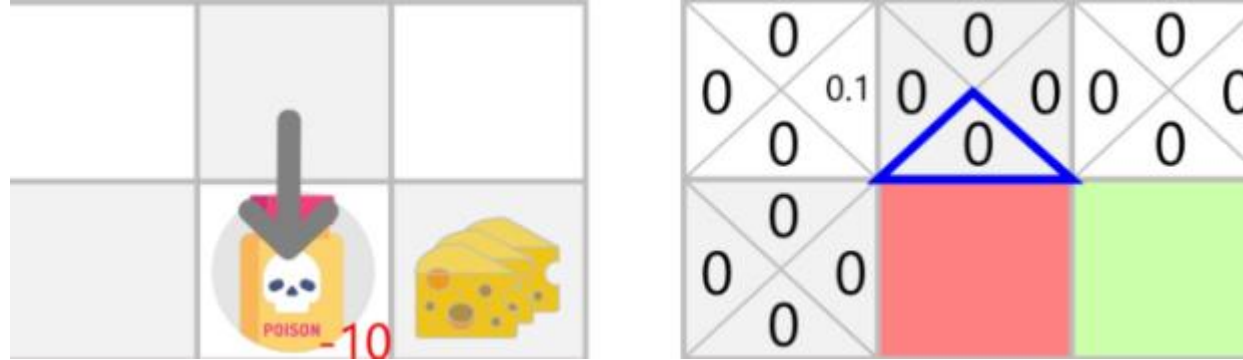
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

$$Q(\text{initial state, Right}) = 0 + 0.1 * (1 + 0.99 * 0 - 0) = 0.1$$

| |  |  |  |  |
|---|--|---|---|---|
|  | 0 | 0.1 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |

Time step 2- Step 2: Choose an action using the Epsilon Greedy Strategy

- I take a random action again, since $\epsilon=0.99$ is big. (Notice we decay epsilon a little bit because, as the training progress, we want less and less exploration).
- I took the action 'down'. This is not a good action since it leads me to the poison.



Step 3: Perform action A_t , get R_{t+1} and S_{t+1}











- Because I ate poison, I get $R_{t+1} = -10$, and I die.



Step 4: Update $Q(S_t, A_t)$

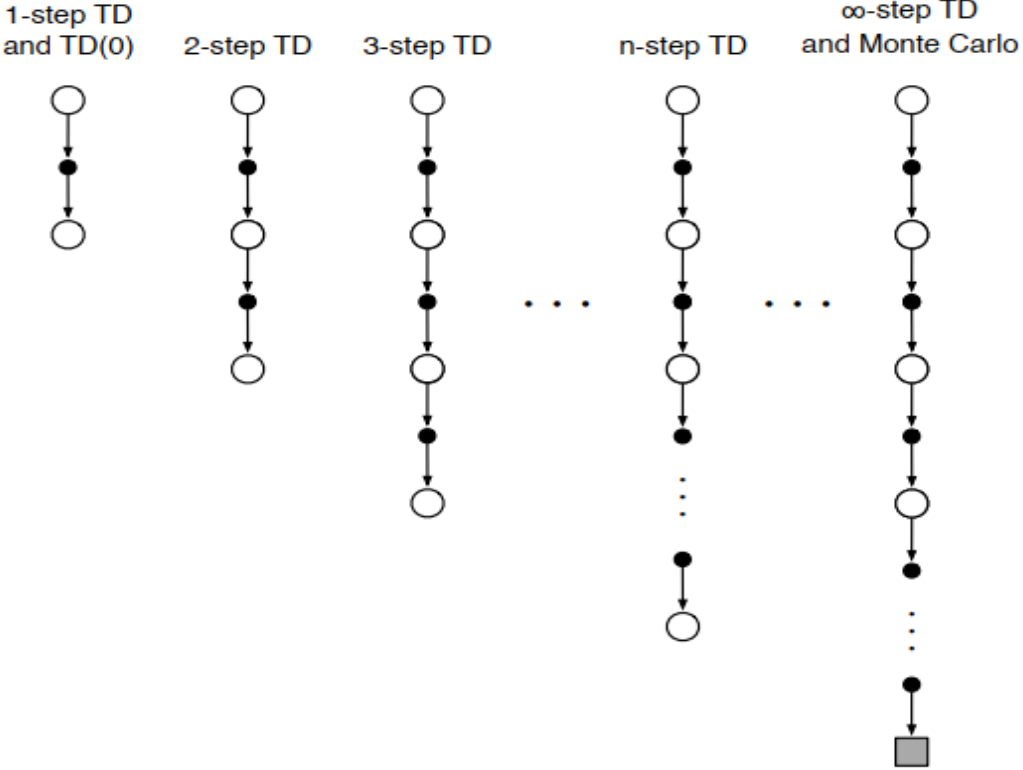
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$$

$$Q(\text{State 2, Down}) = 0 + 0.1 * (-10 + 0.99 * 0 - 0) = -1$$

| |  |  |  |  |
|---|---|---|---|---|
|  | 0 | 0.1 | 0 | 0 |
|  | 0 | 0 | 0 | -1 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |
|  | 0 | 0 | 0 | 0 |

N Step TD

- Instead of only using the reward of a single state-action, you can take multiple actions
- To do that you need to calculate the N step Returns



N Step Returns

- Consider the following n -step returns for $n = 1, 2, \infty$:

$$n = 1 \quad (TD) \quad G_t^{(1)} = R_{t+1} + \gamma V(S_{t+1})$$

$$n = 2 \quad G_t^{(2)} = R_{t+1} + \gamma R_{t+2} + \gamma^2 V(S_{t+2})$$

\vdots

$$n = \infty \quad (MC) \quad G_t^{(\infty)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-1} R_T$$

- Define the n -step return

$$G_t^{(n)} = R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{n-1} R_{t+n} + \gamma^n V(S_{t+n})$$

- n -step temporal-difference learning

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t^{(n)} - V(S_t) \right)$$

N Step SARSA

n-step Sarsa for estimating $Q \approx q_*$ or q_π

Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$

Initialize π to be ε -greedy with respect to Q , or to a fixed given policy

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer n

All store and access operations (for S_t , A_t , and R_t) can take their index mod $n + 1$

Loop for each episode:

 Initialize and store $S_0 \neq$ terminal

 Select and store an action $A_0 \sim \pi(\cdot|S_0)$

$T \leftarrow \infty$

 Loop for $t = 0, 1, 2, \dots$:

 If $t < T$, then:

 Take action A_t

 Observe and store the next reward as R_{t+1} and the next state as S_{t+1}

 If S_{t+1} is terminal, then:

$T \leftarrow t + 1$

 else:

 Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$

$\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)

 If $\tau \geq 0$:

$G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$

 If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$ ($G_{\tau:\tau+n}$)

$Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha [G - Q(S_\tau, A_\tau)]$

 If π is being learned, then ensure that $\pi(\cdot|S_\tau)$ is ε -greedy wrt Q

 Until $\tau = T - 1$

References

- Sutton & Barto Book: Reinforcement Learning
- Robotch Academy: Slides of Reinforcement Learning Course
- Grokking Deep Reinforcement Learning Book
- Hugging Face RL course