Markov Decision Processes

What is a Markov decision process?

- **MDP** (Environment evaluation)

- Andrey Markov (1856-1922)

- “Markov” generally means that given the **present state**, the **future** and **independent**

- F

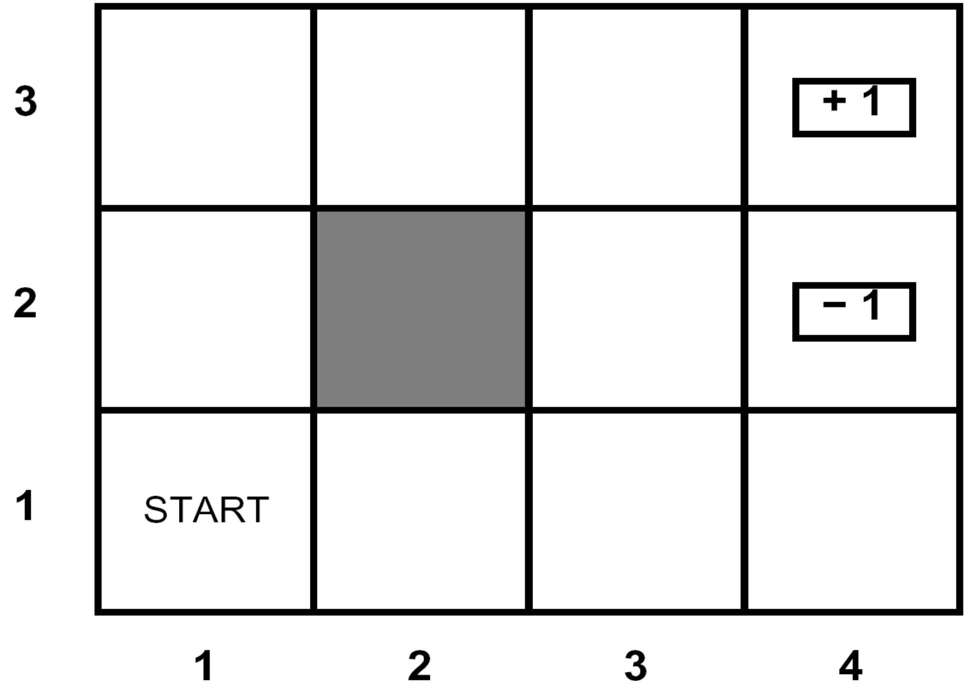
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0)$$

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

- A **mathematical** representation of a sequential decision making problem

Markov Decision Process

- An MDP is defined by 5 tuple (S,A, γ, {Psa},R)
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s,a,s')
    - Probability that a from s leads to s'
    - i.e., P(s' | s,a)
    - Also called the model
  - A reward function R(s, a, s')
    - Sometimes just R(s) or R(s')
  - A start state (or distribution)
  - Maybe a terminal state
  - A discount factor: γ

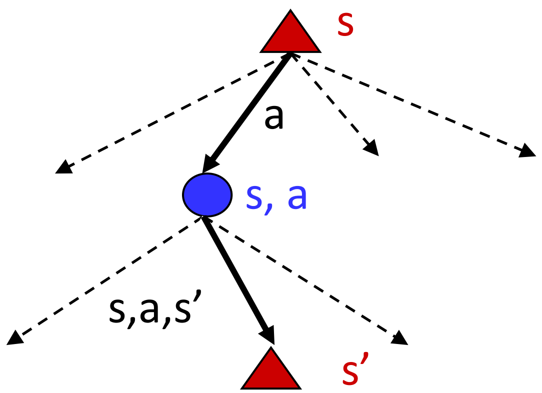


- MDPs are a family of non-deterministic search problems
  - Reinforcement learning: MDPs where we don't know the transition or reward functions

Optimal Utilities

- Fundamental operation: compute the optimal values of states s

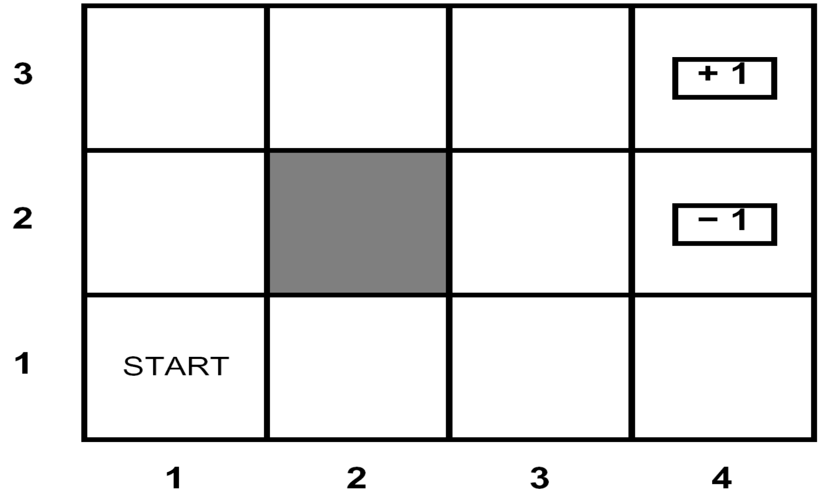
- Optimal values ($V^*(s)$) define optimal policies($\pi^*(s)$)



- Define the value of a state s:
  $V^*(s)$ = expected utility starting in s and acting optimally

- Define the value of a q-state (s,a):
  $Q^*(s,a)$ = expected utility starting in s, taking action a and thereafter acting optimally

- Define the optimal policy:
  $\pi^*(s)$ = optimal action from state s



Optimal Value function (Bellman equations)

- Given a fixed policy π, its value function V π satisfies the

Bellman equations:

$$\underbrace{V^{\pi}(s)=R(s)}_{\text{Immediate reward}}+\underbrace{\gamma\sum_{s'\in S}P_{s\pi(s)}(s')V^{\pi}(s')}_{\text{expected sum of future discounted rewards}}$$

- We start in some state s_0, and get to choose some action $a_0 \in A$
- As a result of our choice, the state of the MDP randomly transitions to some successor state s_1, drawn according to $s_1 \sim Ps0a0$
- Then, we get to pick another action a_1

$$s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} \ldots$$

The Basic Setting for Learning

- Training data: n finite horizon trajectories, of the form

$$\{s_0, a_0, r_0, \ldots, s_T, a_T, r_T, s_{T+1}\}.$$

- Deterministic or stochastic policy: A sequence of decision rules

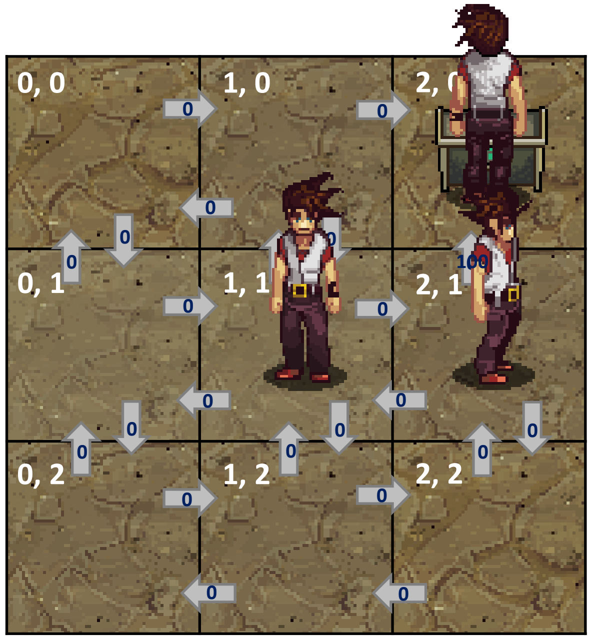
$$\{\pi_0, \pi_1, \ldots, \pi_T\}.$$

- Each π maps from the observable history (states and actions) to the action space at that time point.

Example of Q learning (episode 1)

- Initialize $\hat{Q}$ to 0
- Random initial state = $< 1,1 >$
- Random action from $A_{<1,1>} = east$
  - $s' =< 2,1 >$
  - $R_a(s, s') = 0$
- Update $\hat{Q}(< 1,1 >, east) = 0$
- Random action from $A_{<2,1>} = north$
  - $s' =< 2,0 >$
  - $R_a(s, s') = 100$
- Update $\hat{Q}(< 2,1 >, north) = 100$
- No more moves possible, start again…

$$\hat{Q}(s, a) = R_a(s, s') + \gamma \max_{a'} \hat{Q}_{n-1}(s', a')$$



Example of Q learning

$$\hat{Q}(s,a) = R_a(s,s') + \gamma \max_{a'} \hat{Q}_{n-1}(s',a')$$

$$\gamma = 0.5$$

- Random Initial State $< 0,0 >$
- Update $\hat{Q}(< 1,1 >, east) = 50$
- Update $\hat{Q}(< 1,2 >, east) = 25$