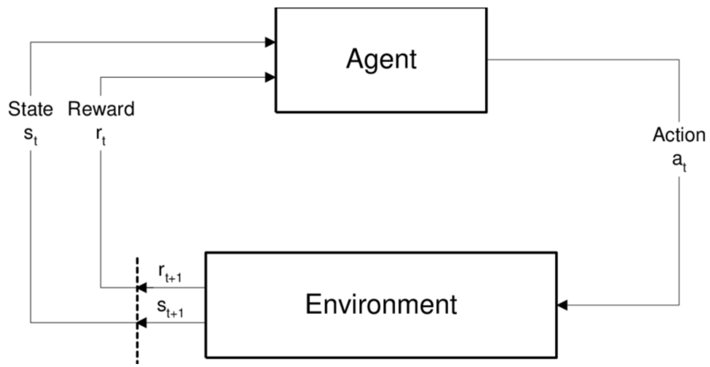
Last time ...

- 

$S_0, A_0, R_1, S_1, A_1, R_2, \ldots$

trajectory

- Discounted Future rewards

$$\sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

γ: discount rate $\in [0, 1]$

- value function

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

intuitively: given a state s, tells me the expected discounted future rewards if I follow policy π.

- action-value function

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

intuitively: given a state s AND action a, tells me the expected discounted future rewards if I follow policy π.

Recall MDP (Markov Decision Process).

Formally: MDP is a tuple $\langle S, A, P, R, \gamma \rangle$

S : finite set of states

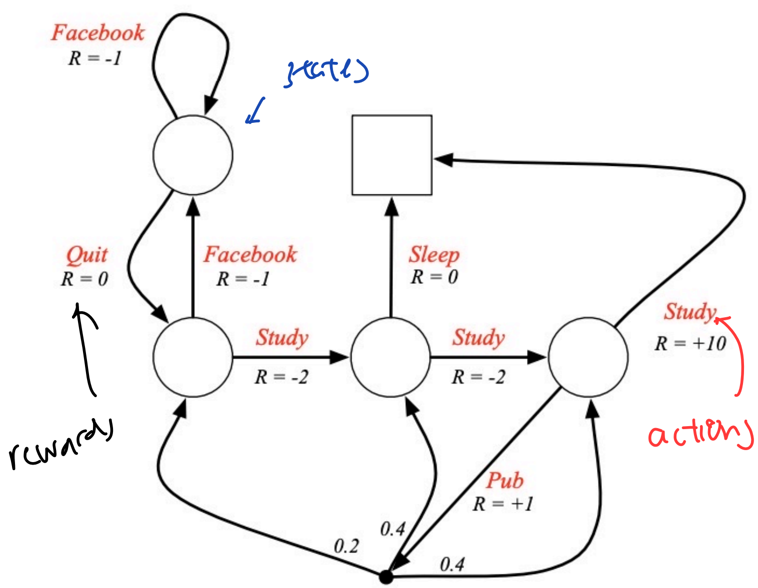
A : finite set of actions

P : transition probability btwn. states : $P(S_{t+1} = s' \mid S_t = s, A_t = a)$

R : Reward function : $\mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$

γ : discount factor $\gamma \in [0,1]$

Example:



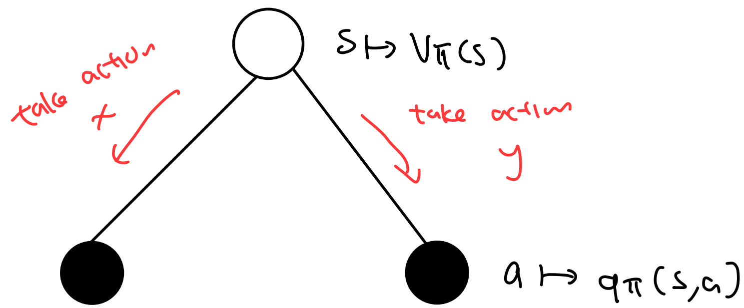
Student MDP

State-value function:

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

Can be expressed as

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



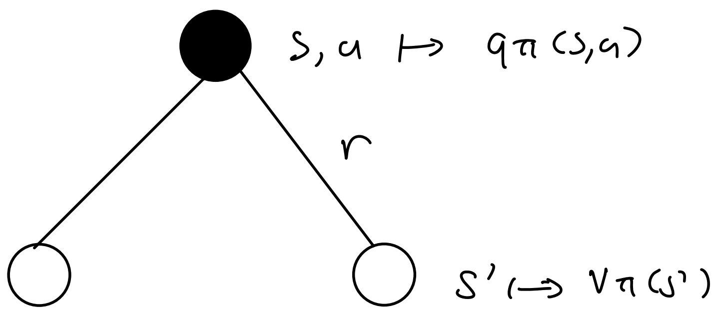
$$v_\pi(s) = \sum_{a \in A} \pi(a \mid s)\, q_\pi(s,a)$$

taking the average across all actions.

Similarly for $q_\pi(s,a)$ . . .

$$q_\pi(s,a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] + \sum_{s' \in S} \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a) v_\pi(s')$$

$$= \mathcal{R}_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V_\pi(s')$$



Recall, the Bellman Equation for v_π is given by

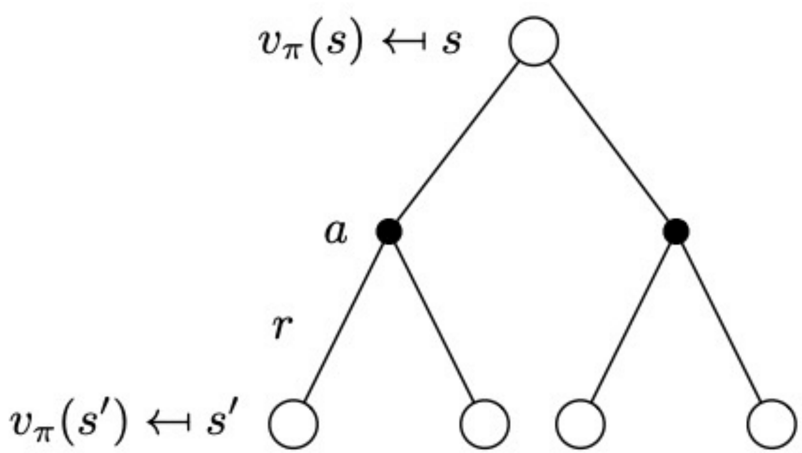
$$v_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s',r|s,a)\left[r + \gamma v_\pi(s')\right] \quad \forall s \in S$$

- we sum over a, s', r
- can be treated as an expected value
- For each triple (a, s', r), we compute its probability $\pi(a|s)\,p(s',r|s,a)$
  - → weighted by $[r + \gamma v_\pi(s')]$
  - → Sum over all possibilities → expected value

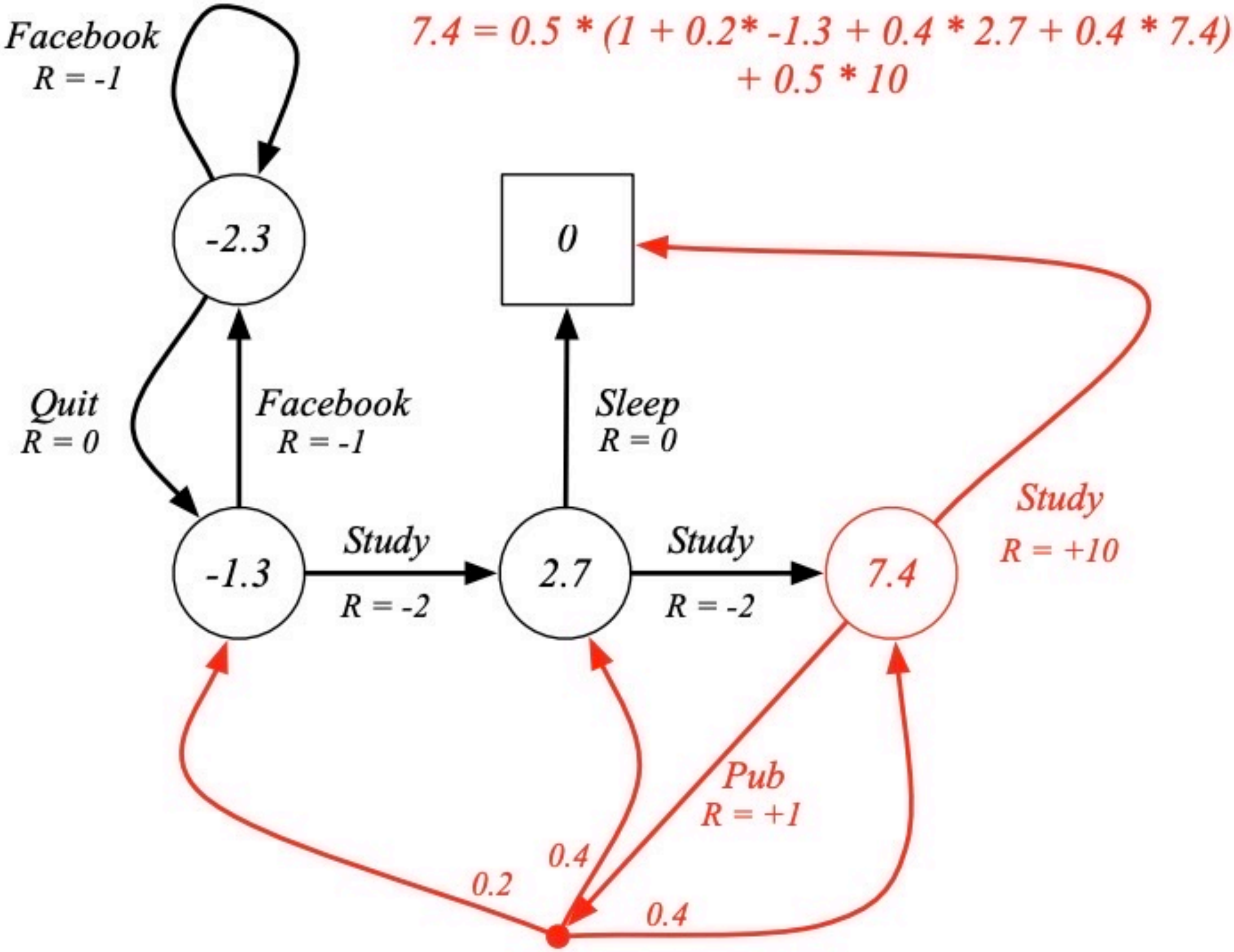
expresses the relationship between the value of a state and values of successor states.

Intuitively: Value of start state = discounted value of expected next state + reward along the way

To solve a reinforcement problem = finding $v_*(s) = \max_\pi v_\pi(s)$ and $q_*(s,a) = \max_\pi q_\pi(s,a)$



Example



7.4 = 0.5 * (1 + 0.2* -1.3 + 0.4 * 2.7 + 0.4 * 7.4)
+ 0.5 * 10
Facebook
R = -1
-2.3
0
Quit
R = 0
Facebook
R = -1
Sleep
R = 0
Study
R = +10
-1.3
Study
R = -2
2.7
Study
R = -2
7.4
Pub
R = +1
0.4
0.2
0.4

To solve for this $v*(s)$ and $q*(s,a)$, we can use:

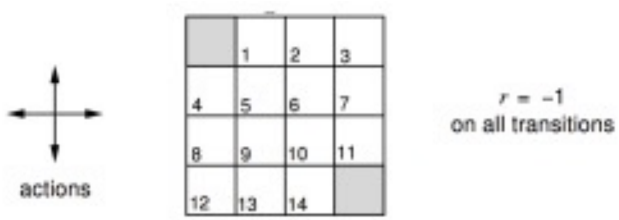
- value iteration
- policy iteration
- Q-learning
- Sarsa

Today: DP & policy evaluation / iteration

Dynamic Programming

DP: Algorithms used to find optimal policies which have complete knowledge ($p(s', r \mid s, a)$). → not realistic assumption

Example:



- Undiscounted episodic MDP ($\gamma = 1$)
- Nonterminal states 1, ..., 14
- One terminal state (shown twice as shaded squares)
- Actions leading out of the grid leave state unchanged
- Reward is -1 until the terminal state is reached
- Agent follows uniform random policy

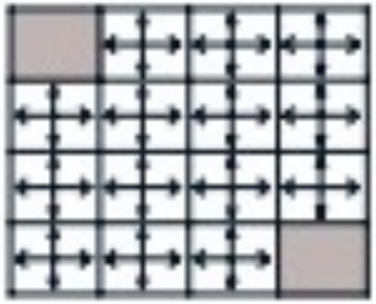
$$\pi(n|\cdot) = \pi(e|\cdot) = \pi(s|\cdot) = \pi(w|\cdot) = 0.25$$

POLICY ITERATION

v_k for the Random Policy | Greedy Policy w.r.t. v_k

$k = 0$

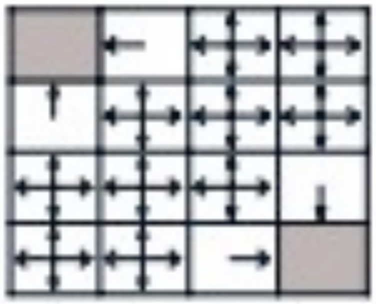
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0



random policy

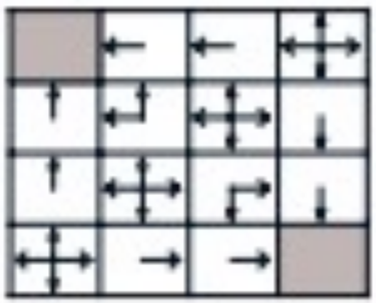
$k = 1$

0.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	-1.0
-1.0	-1.0	-1.0	0.0



$k = 2$

0.0	-1.7	-2.0	-2.0
-1.7	-2.0	-2.0	-2.0
-2.0	-2.0	-2.0	-1.7
-2.0	-2.0	-1.7	0.0



$k = 3$

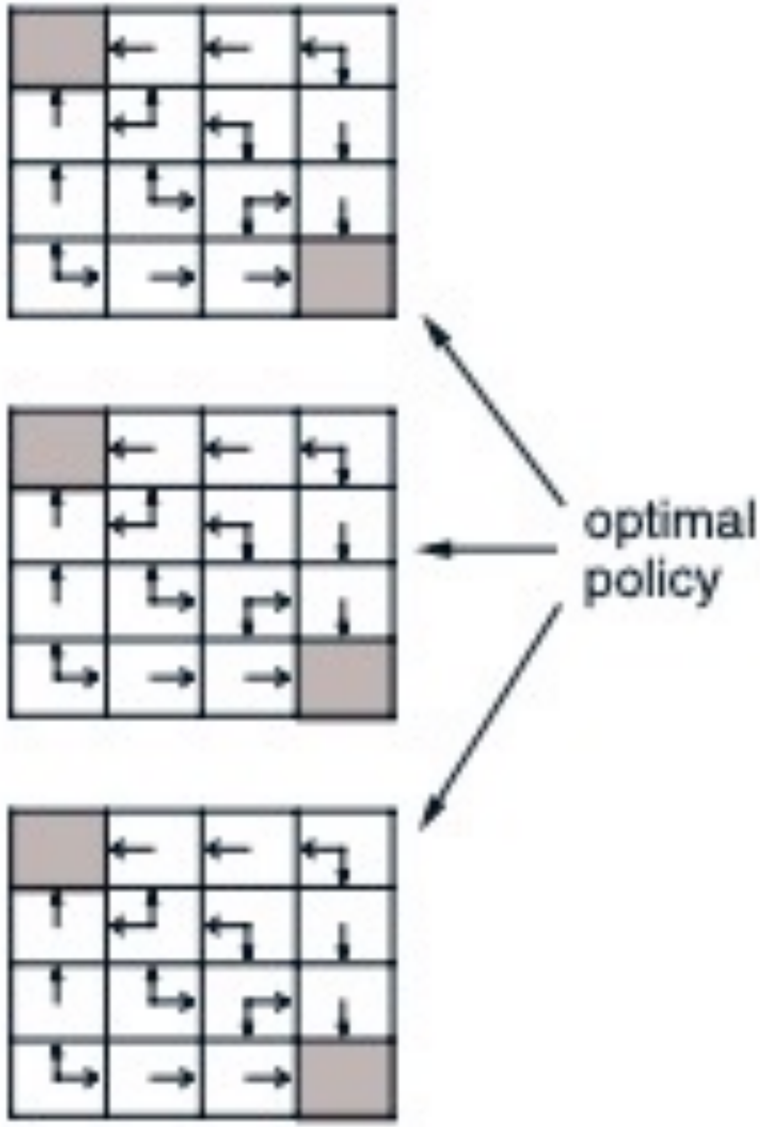
0.0	-2.4	-2.9	-3.0
-2.4	-2.9	-3.0	-2.9
-2.9	-3.0	-2.9	-2.4
-3.0	-2.9	-2.4	0.0

$k = 10$

0.0	-6.1	-8.4	-9.0
-6.1	-7.7	-8.4	-8.4
-8.4	-8.4	-7.7	-6.1
-9.0	-8.4	-6.1	0.0

$k = \infty$

0.0	-14.	-20.	-22.
-14.	-18.	-20.	-20.
-20.	-20.	-18.	-14.
-22.	-20.	-14.	0.0



How do we improve a policy?

A: Given a policy π,

1. Evaluate the policy

$$V_\pi(s) = \mathbb{E}[G_t \mid S_t = s]$$

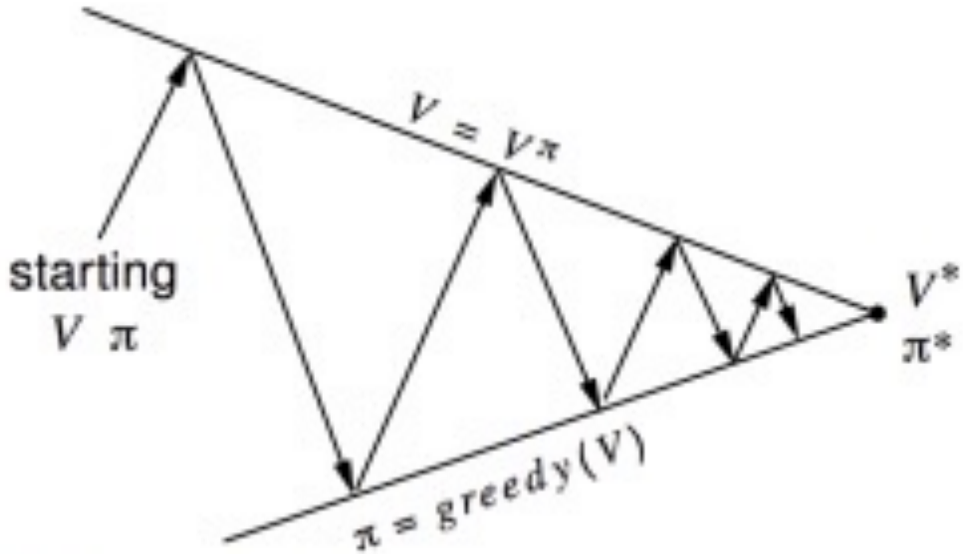
2. Improve policy by acting greedily w.r.t. v_π

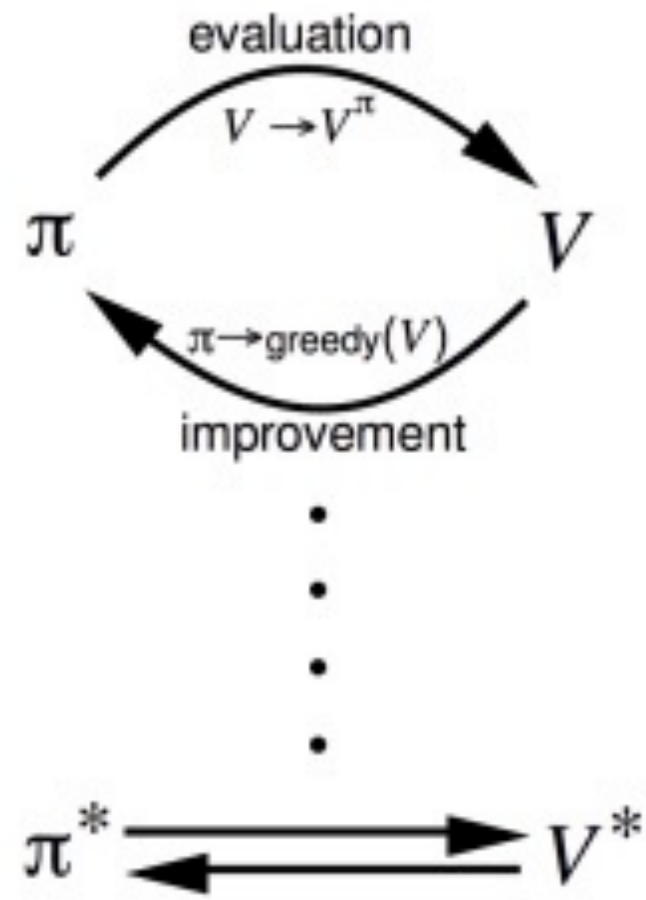
$$\pi' = \text{greedy}(v_\pi)$$

What is greedy? → Taking the maximum rewards / value

NOTE: POLICY ITERATION ALWAYS converges to π^*

In general...





Policy evaluation Estimate v_π

Any policy evaluation algorithm

Policy improvement Generate $\pi' \geq \pi$

Any policy improvement algorithm

Next time:

- Value iteration
- Code in notebook
- Start thinking about projects