SAPIENZA
UNIVERSITÀ DI ROMA

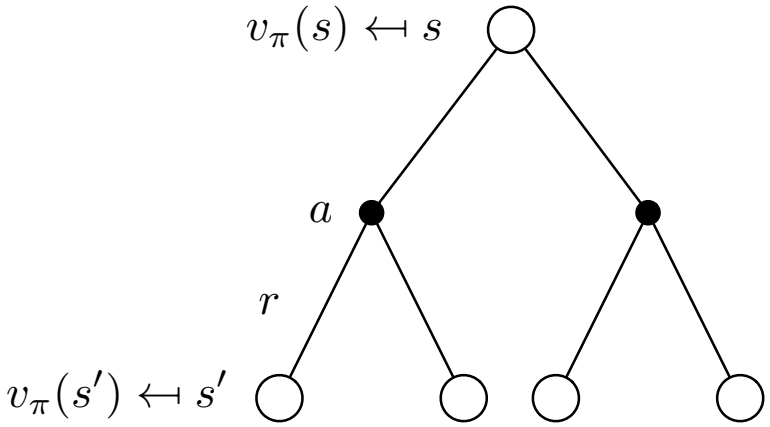
Autonomous Networking

Gaia Maselli
Dept. of Computer Science

Today's plan

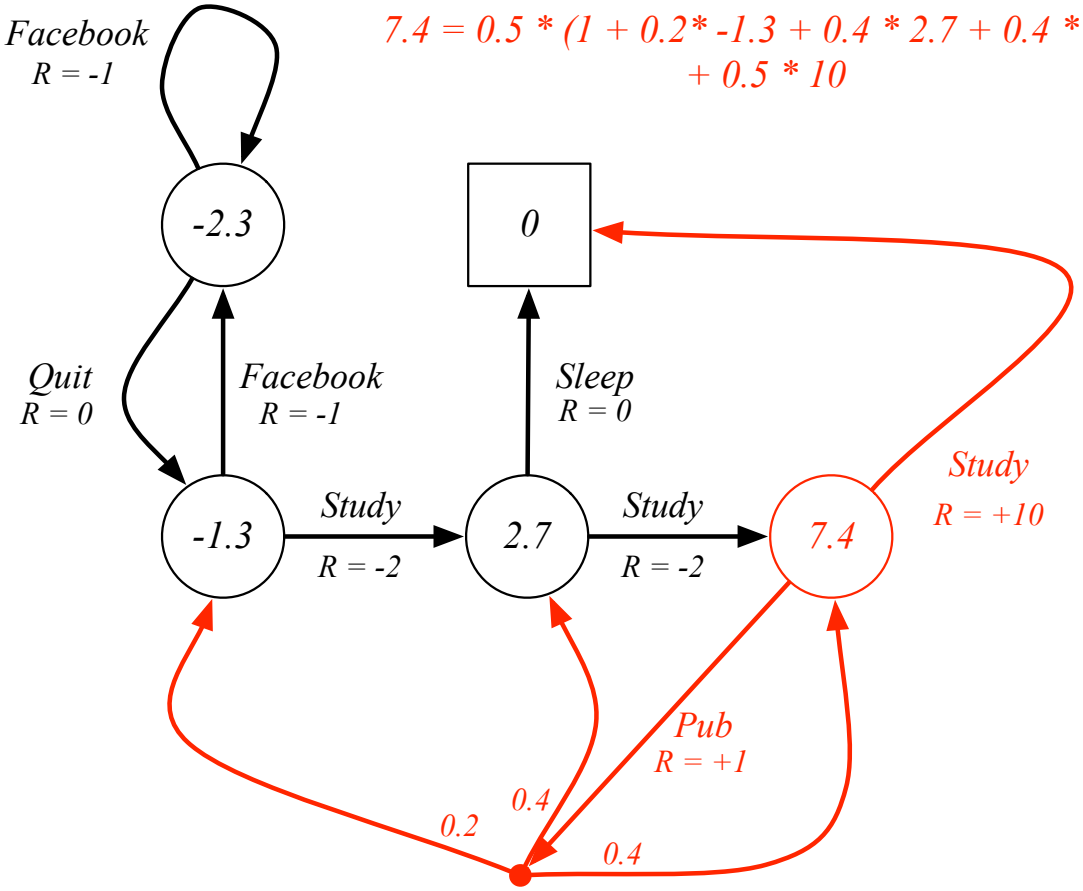
- Example of value function calculation
- Optimal policy
- Q-learning

Bellman Expectation Equation for v_π (2)

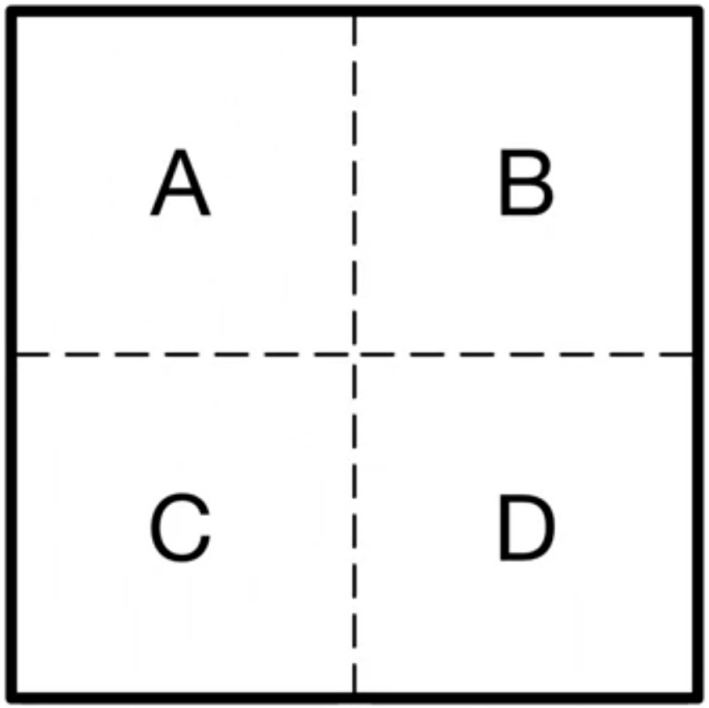


$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s') \right)$$

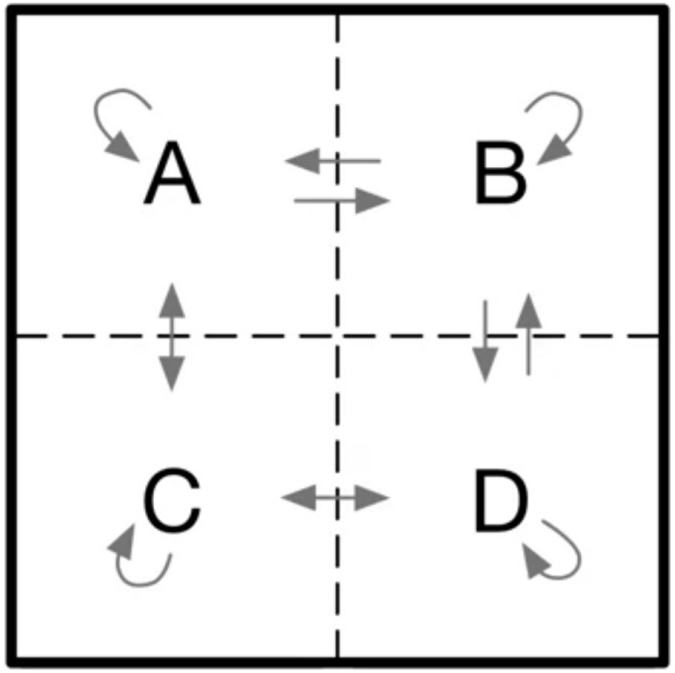
Example: Bellman Expectation Equation in Student MDP

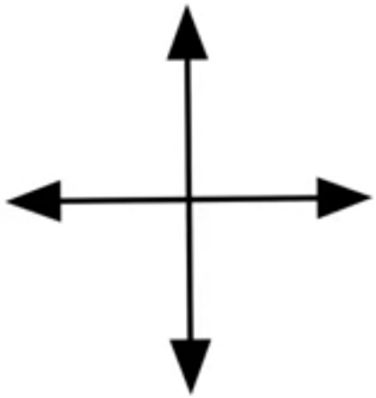


Example: Gridworld

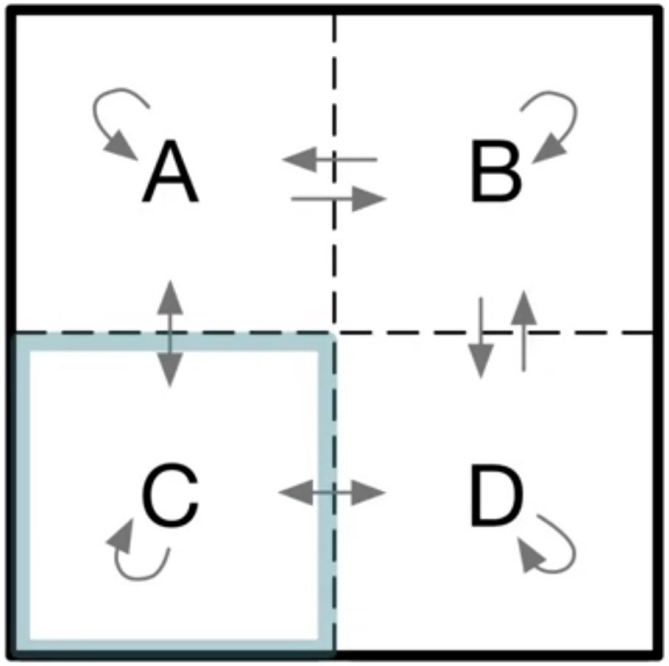


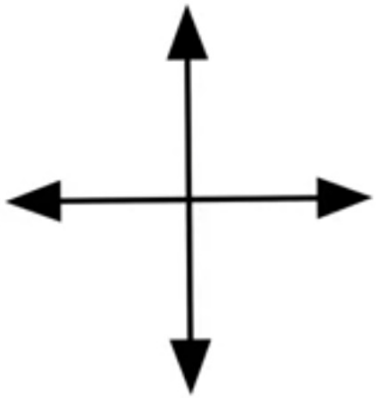
Example: Gridworld



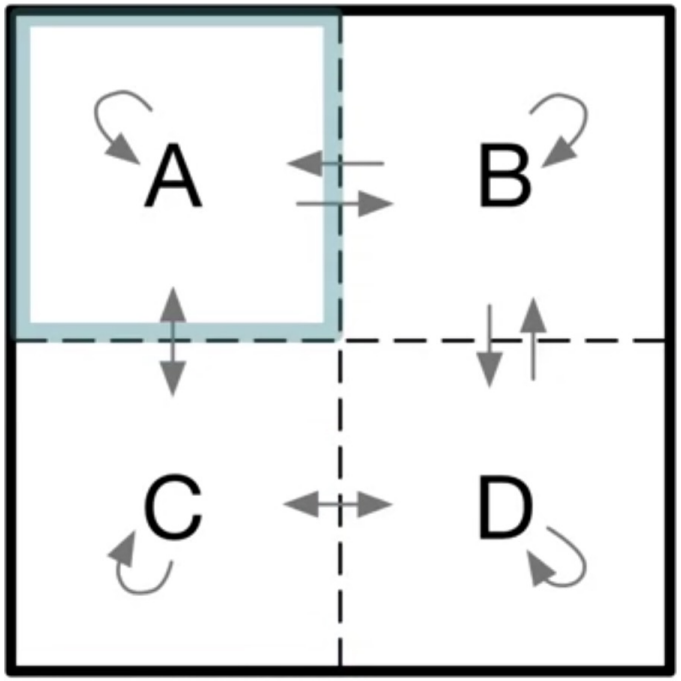


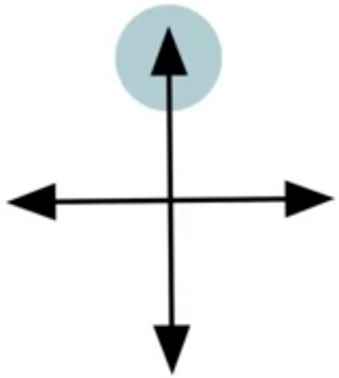
Example: Gridworld



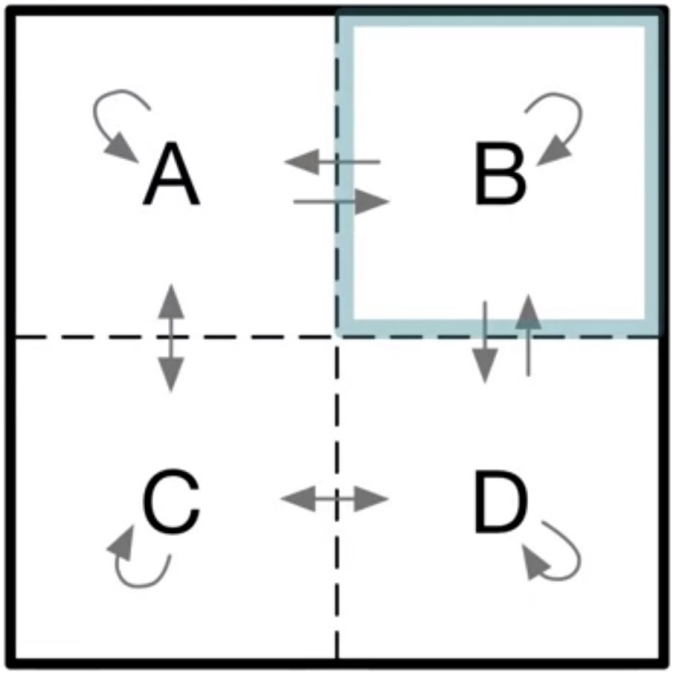


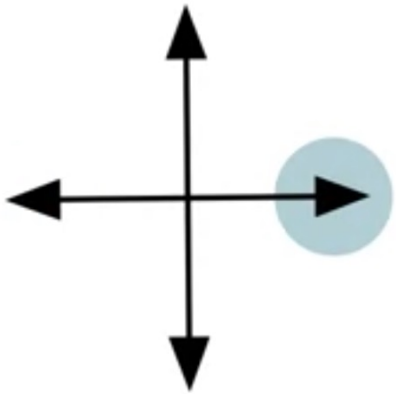
Example: Gridworld



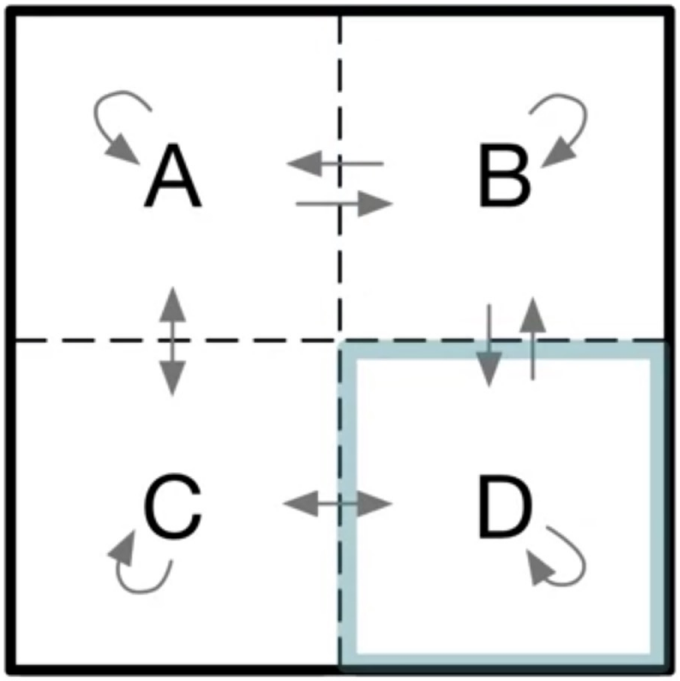


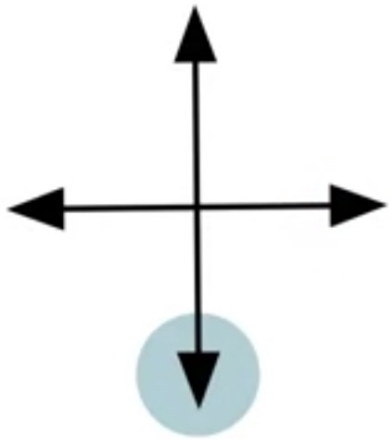
Example: Gridworld



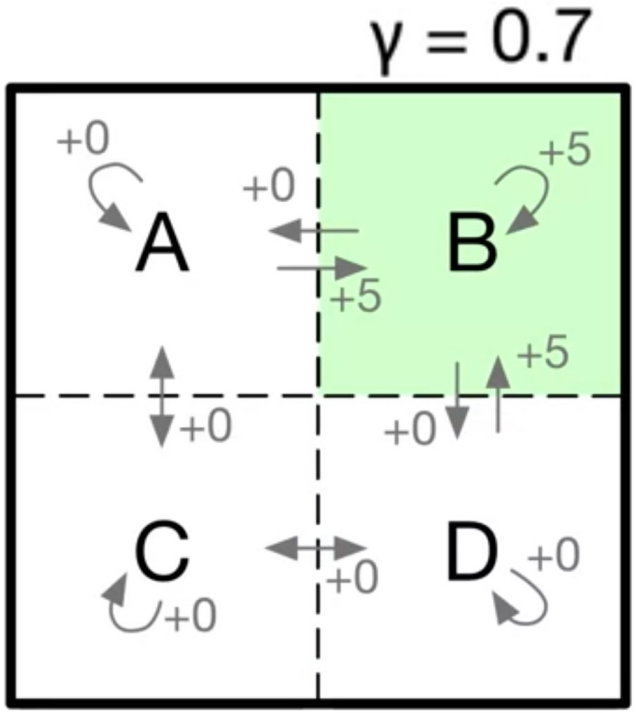


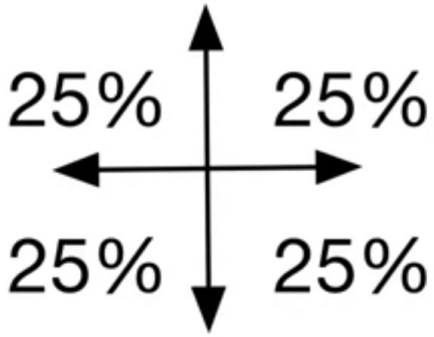
Example: Gridworld



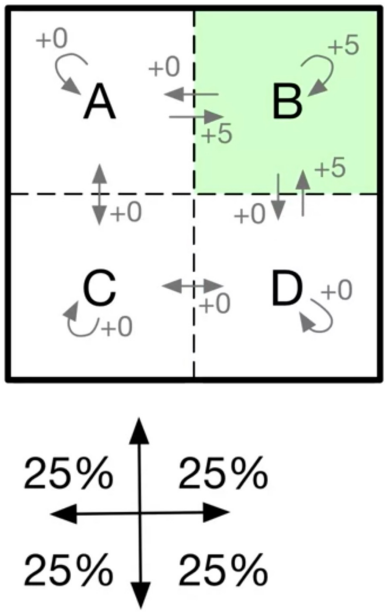


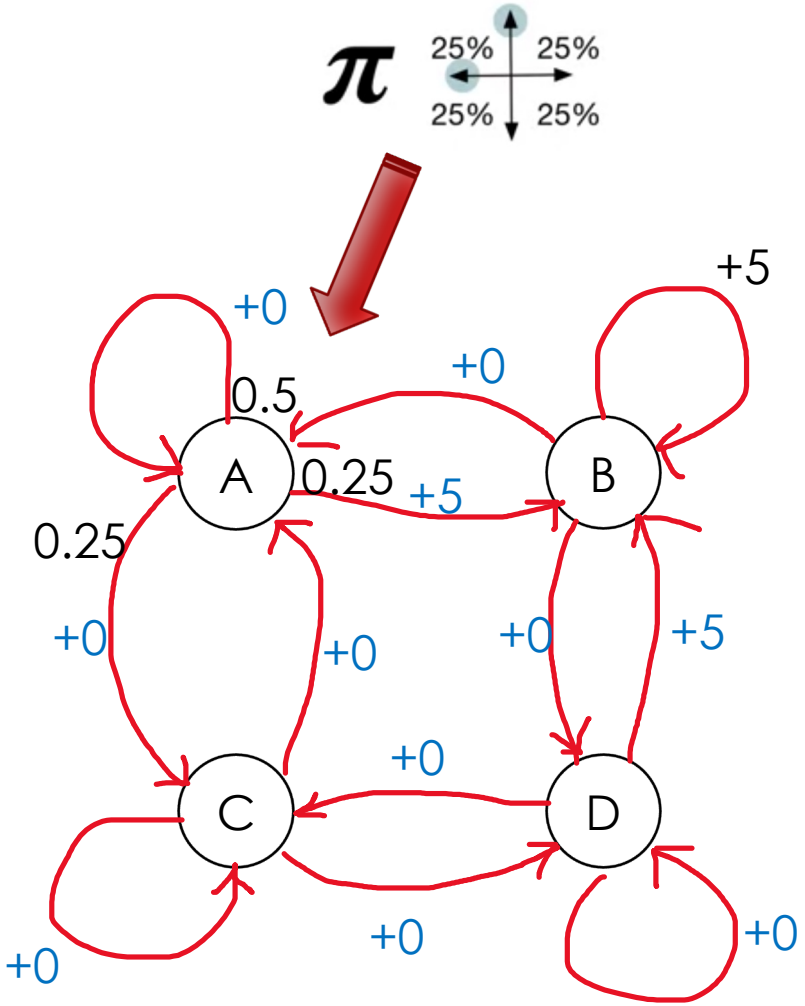
Example: Gridworld





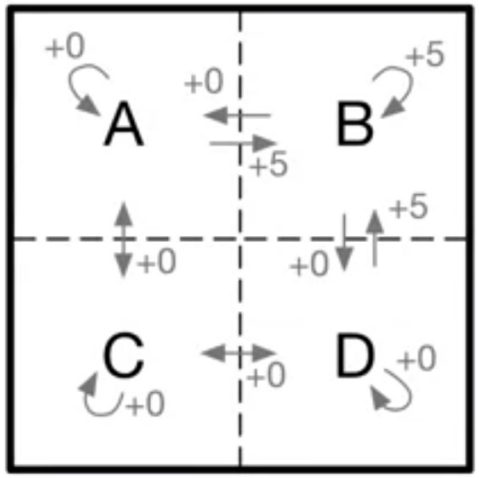
Example: Gridworld



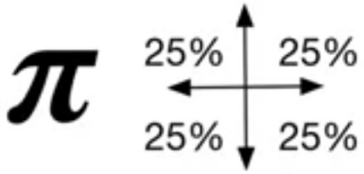


Example: Gridworld

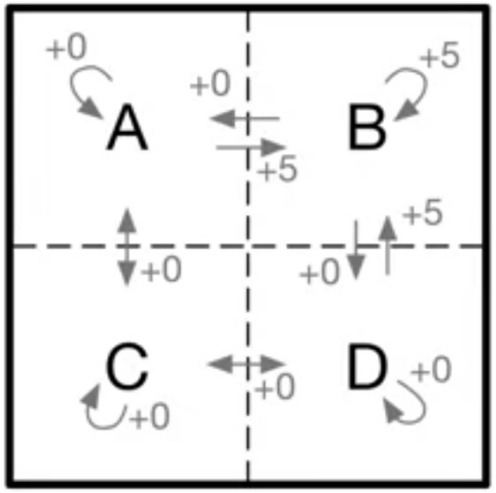
γ=0.7



$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

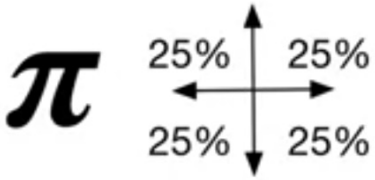


Example: Gridworld

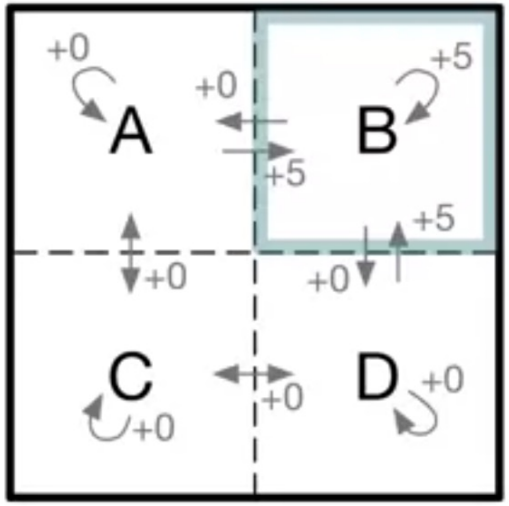


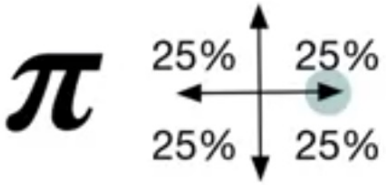
$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

$$V_\pi(A) = \sum_a \pi(a|A)\left(r + 0.7 V_\pi(s')\right)$$



Example: Gridworld



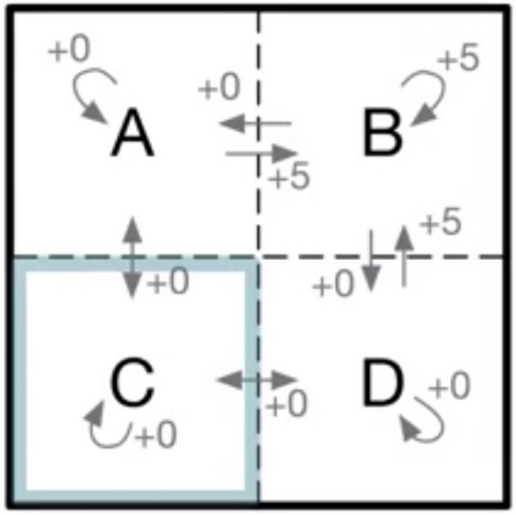


$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

$$V_\pi(A) = \sum_a \pi(a|A)\left(r + 0.7V_\pi(s')\right)$$

$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B))$$

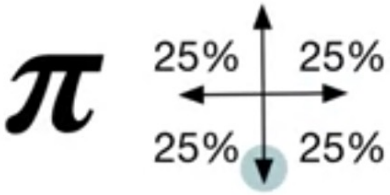
Example: Gridworld



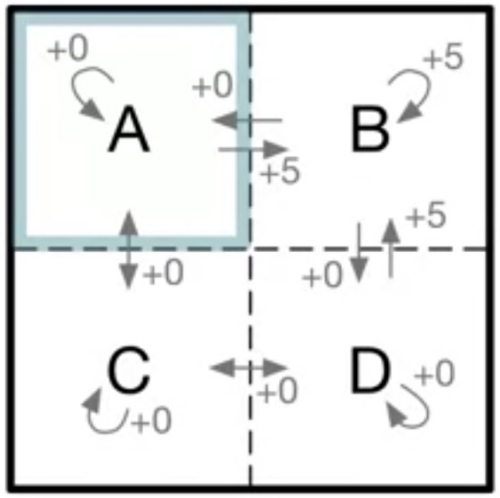
$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

$$V_\pi(A) = \sum_a \pi(a|A)\left(r + 0.7 V_\pi(s')\right)$$

$$V_\pi(A) = \frac{1}{4}(5 + 0.7 V_\pi(B)) + \frac{1}{4} 0.7 V_\pi(C)$$



Example: Gridworld

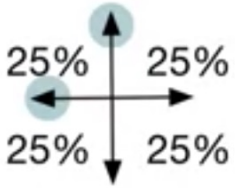


$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

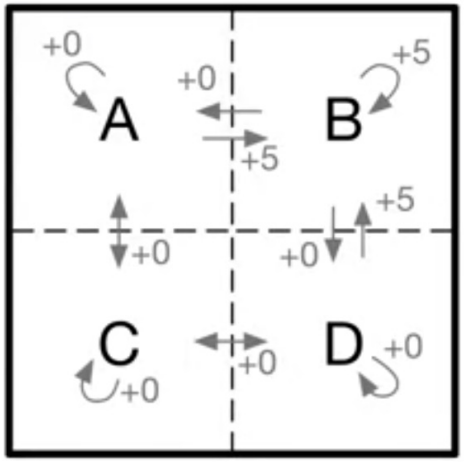
$$V_\pi(A) = \sum_a \pi(a|A)\left(r + 0.7V_\pi(s')\right)$$

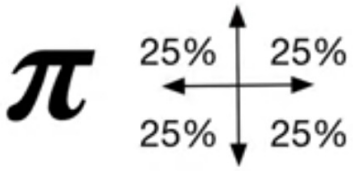
$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(A)$$

π



Example: Gridworld

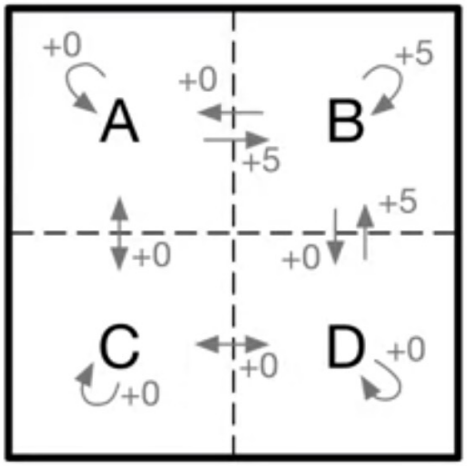


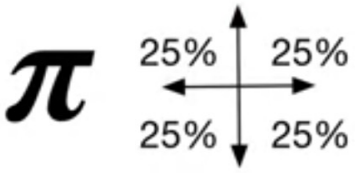


$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(A)$$

Example: Gridworld



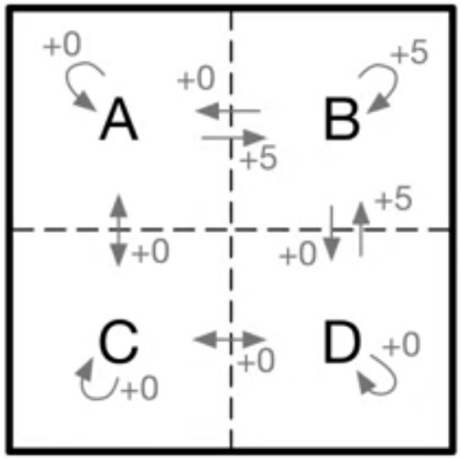


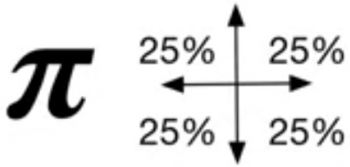
$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(A)$$

$$V_\pi(B) = \frac{1}{2}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(A) + \frac{1}{4}0.7V_\pi(D)$$

Example: Gridworld





$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

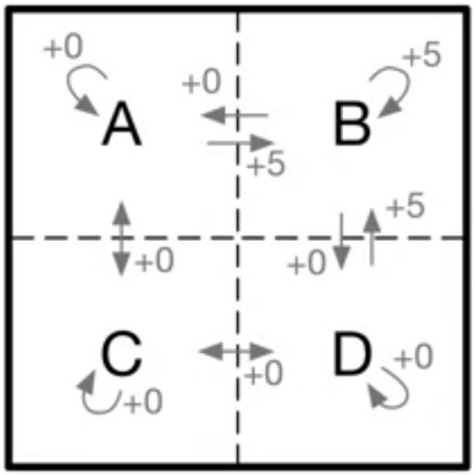
$$V_\pi(A) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(A)$$

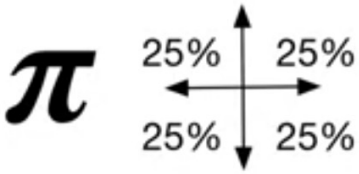
$$V_\pi(B) = \frac{1}{2}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(A) + \frac{1}{4}0.7V_\pi(D)$$

$$V_\pi(C) = \frac{1}{4}0.7V_\pi(A) + \frac{1}{4}0.7V_\pi(D) + \frac{1}{2}0.7V_\pi(C)$$

$$V_\pi(D) = \frac{1}{4}(5 + 0.7V_\pi(B)) + \frac{1}{4}0.7V_\pi(C) + \frac{1}{2}0.7V_\pi(D)$$

Example: Gridworld





$$V_\pi(s) = \sum_a \pi(a|s) \sum_r \sum_{s'} p(s', r|s, a)[r + \gamma V_\pi(s')]$$

$$V_\pi(A) = 4.2$$

$$V_\pi(B) = 6.1$$

$$V_\pi(C) = 2.2$$

$$V_\pi(D) = 4.2$$

Policies

- Up to this point, we've generally talked about a policy as something that is given.
- The policy specifies how an agent behaves.
- Given this way of behaving, we then aim to find the value function.
- But the goal of reinforcement learning is not just to evaluate specific policies.
- Ultimately, *we want to find a policy that obtains as much reward as possible in the long run*

How to find the best possible solution to MDP

How to find the optimal policy

Optimal value function

- To define an optimal policy, we first have to understand what it means for one policy to be better than another

Definition

The **optimal state-value function $v_*(s)$** is the maximum value function over all policies

$$V_*(s) = \max_\pi V_\pi(s)$$

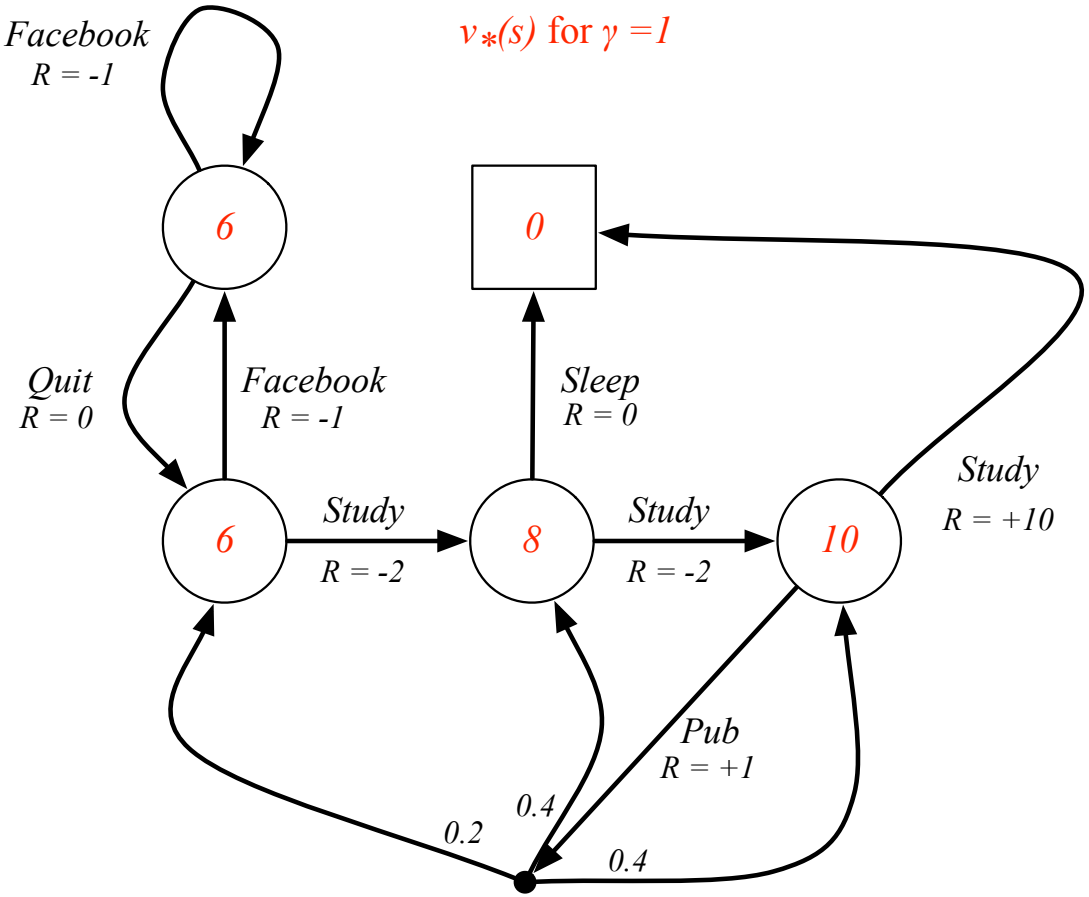
Definition

The **optimal action-value function $q_*(s,a)$** is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

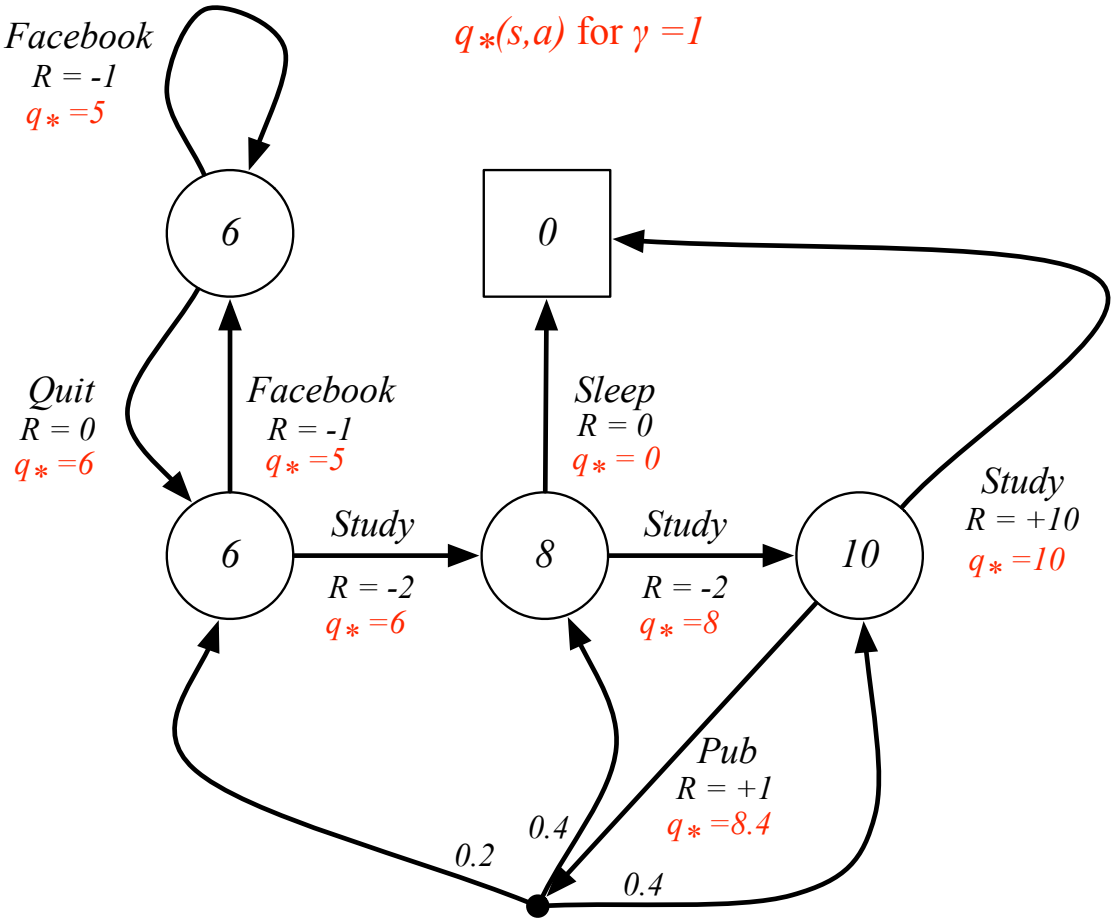
- The optimal value function specifies the best possible performance in the MDP
- An MDP is "solved" when we know the optimal value fn.

Example: Optimal Value Function for Student MDP



- V* says how good is to be in each state
- it does say how to behave

Example: Optimal Action-Value Function for Student MDP



Optimal policy

- Define a partial ordering over policies

$$\pi \geq \pi' \text{ if } V_{\pi}(s) \geq V_{\pi'}(s), \forall s$$

Theorem

For any Markov Decision Process

- There exists an optimal policy $\pi*$ that is better than or equal to all other policies, $\pi * \geq \pi, \forall \pi$
- All optimal policies achieve the optimal value function, $V_{\pi_*}(s) \geq V_*(s)$
- All optimal policies achieve the optimal action-value function, $q_{\pi_*}(s,a) \geq q_*(s,a)$

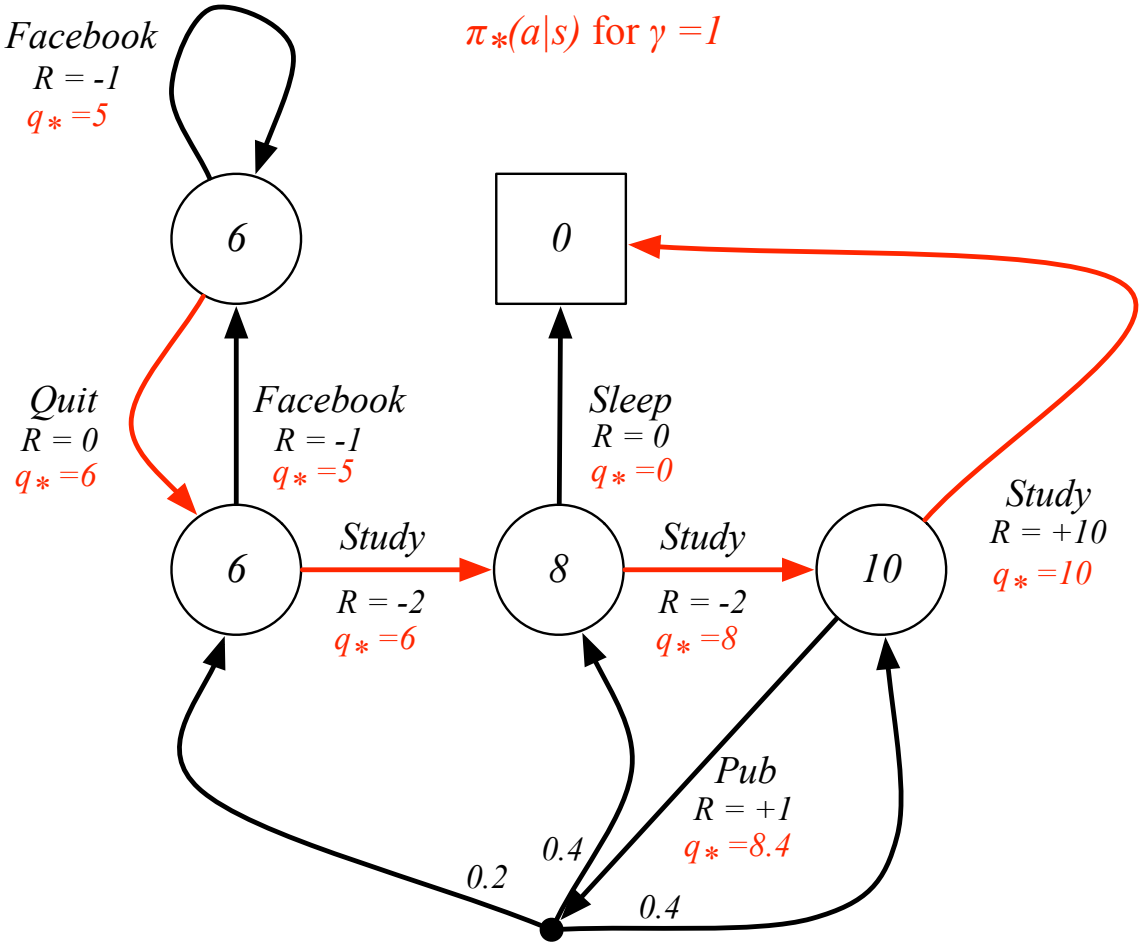
Finding an optimal policy

- An optimal policy can be found by maximising over q∗(s,a)

$$\pi_*(a|s) = \begin{cases} 1 & \text{if } a = \underset{a \in \mathcal{A}}{\text{argmax}}\ q_*(s,a) \\ 0 & \textit{otherwise} \end{cases}$$

- There is always a deterministic optimal policy for any MDP
- If we know q∗(s,a), we immediately have the optimal policy

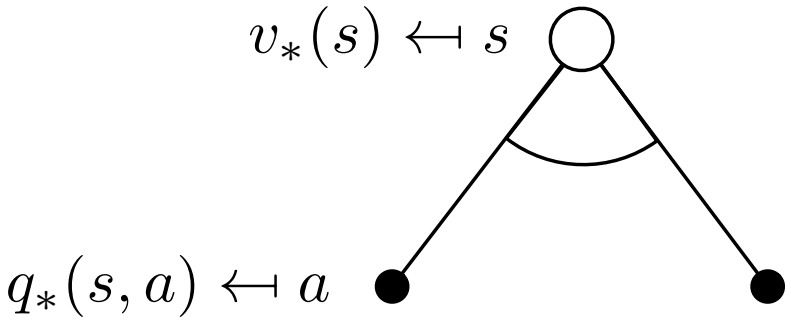
Example: Optimal Policy for Student MDP



How do we get q_* values?

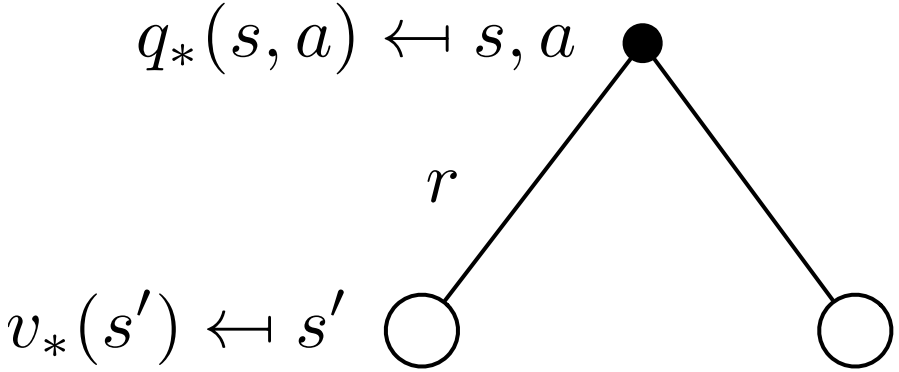
Bellman Optimality Equation for v_*

- The optimal value functions are recursively related by the Bellman optimality equations:



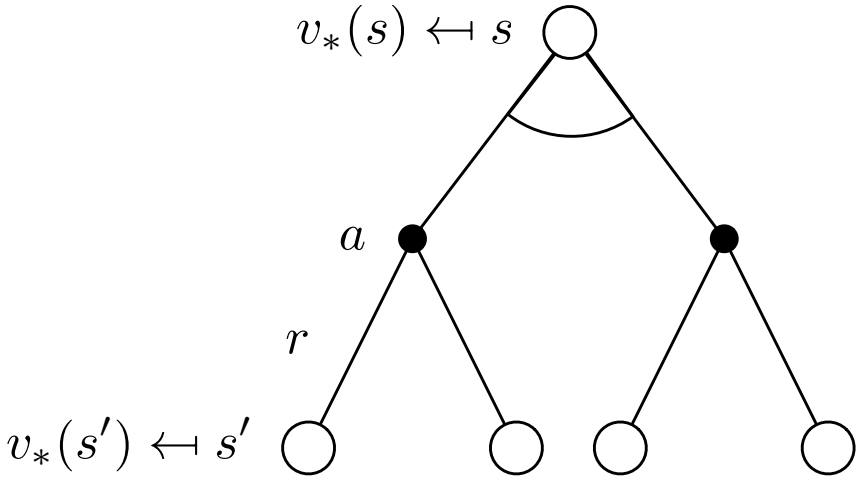
$$v_*(s) = \max_a q_*(s,a)$$

Bellman Optimality Equation for Q*



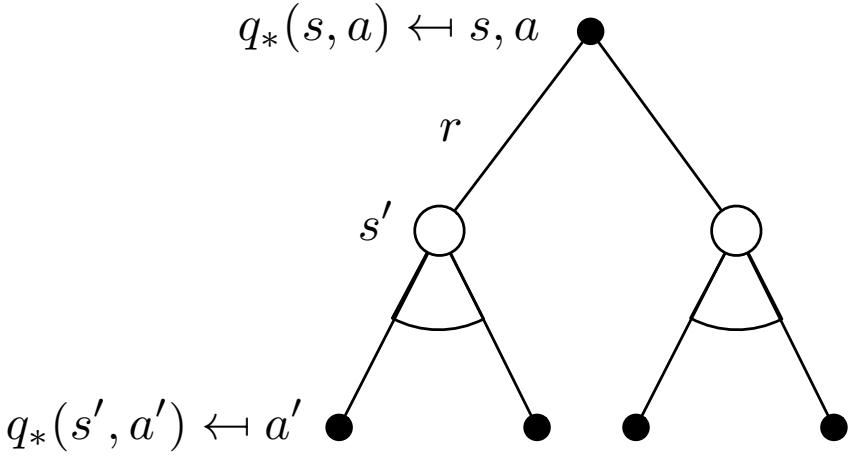
$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for V∗ (2)



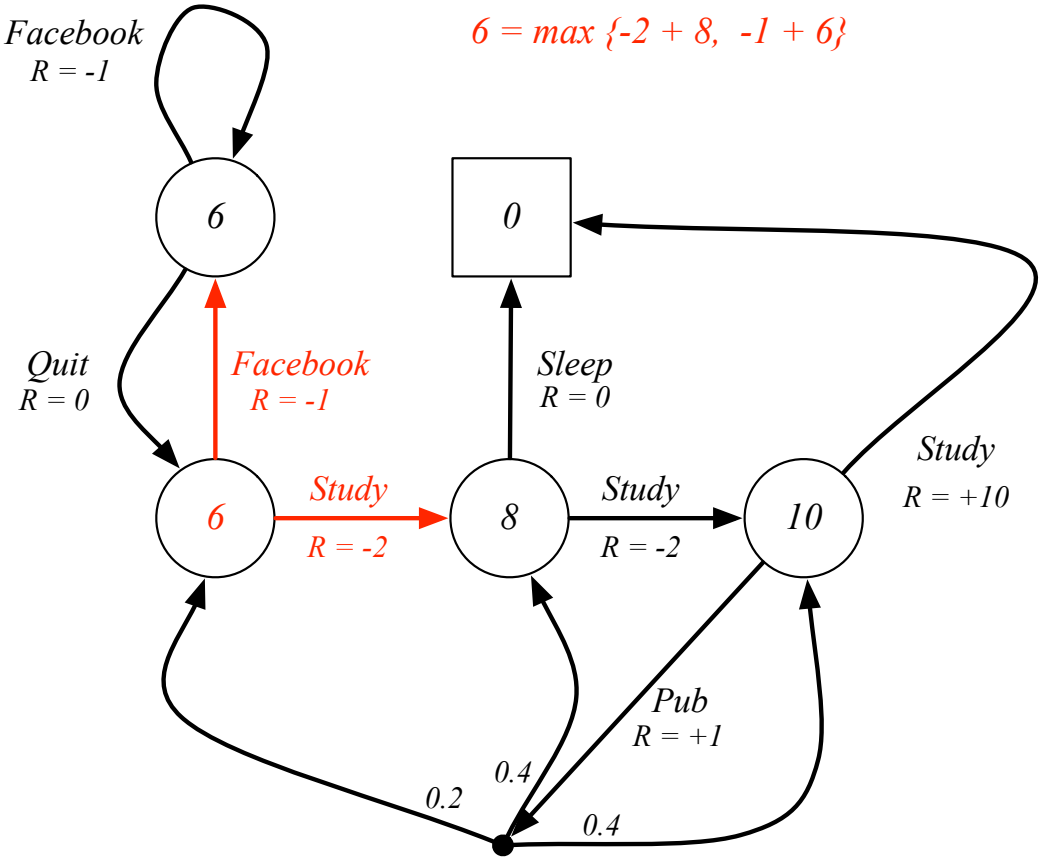
$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Bellman Optimality Equation for Q* (2)



$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s',a')$$

Example: Bellman Optimality Equation in Student MDP



Solving the Bellman Optimality Equation

- Bellman Optimality Equation is non-linear
- No closed form solution (in general)
- Many iterative solution methods
  - Value Iteration
  - Policy Iteration
  - **Q-learning**
  - Sarsa

Q-learning

Temporal Difference (TD) Learning

Off-policy

Q-learning (off-policy TD control)

Temporal Difference(TD) learning

- TD methods learn directly from episodes of experience (learns online after every step)
- TD is **model-free**: no knowledge of MDP transitions / rewards
- TD learns from incomplete episodes, by bootstrapping
- TD updates a guess towards a guess

TD example: driving home

- Each day as you drive home from work, you try to predict how long it will take to get home
- As you wait in traffic, you already know that your initial estimate of 30 minutes was too optimistic. Must you wait until you get home before increasing your estimate for the initial state?

State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

On and Off-Policy Learning

- On-policy learning
  - "Learn on the job"
  - Learn about policy π from experience sampled from π

- Off-policy learning
  - "Look over someone's shoulder"
  - Learn about policy π from experience sampled from μ

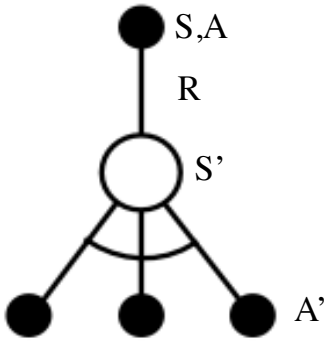
Off-policy

- Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$
- While following behaviour policy $\mu(a|s)$

$$\{S1, A1, R2, \ldots, ST\} \sim \mu$$

- Why is this important?
- Learn from observing humans or other agents
- Re-use experience generated from old policies $\pi_1, \pi_2, \ldots, \pi_{t-1}$
- Learn about *optimal* policy while following *exploratory* policy
- Learn about *multiple* policies while following one policy

Q-Learning Control Algorithm



$$Q(S,A) \leftarrow Q(S,A) + \alpha \left(R + \gamma \max_{a'} Q(S',a') - Q(S,A) \right)$$

- Q-learning control converges to the optimal action-value function, $Q(s, a) \rightarrow q*(s, a)$

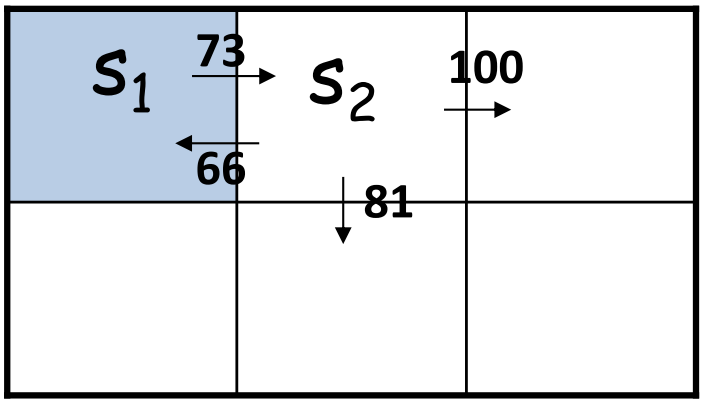
Q-learning algorithm

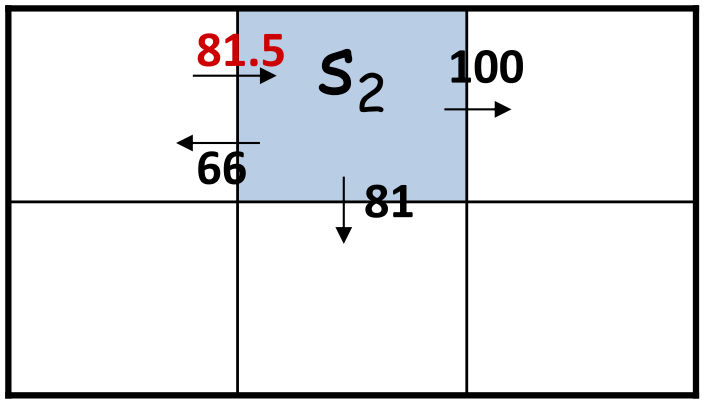
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

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 $Q(terminal, \cdot) = 0$

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

Example





$\gamma = 0.9, \quad \alpha = 0.5, \quad r = 0$ for non-terminal states

$$Q(s_1, right) = Q(s_1, right) + \alpha\left(r + \gamma \max_{a'} Q(s_2, a') - Q(s_1, right)\right)$$
$$= 73 + 0.5(0 + 0.9 \max\{66,81,100\} - 73)$$
$$= 73 + 0.5(17)$$
$$= 81.5$$