COMS 4030A/7047A
Adaptive Computation and Machine Learning

Hima Vadapalli

Semester I, 2022

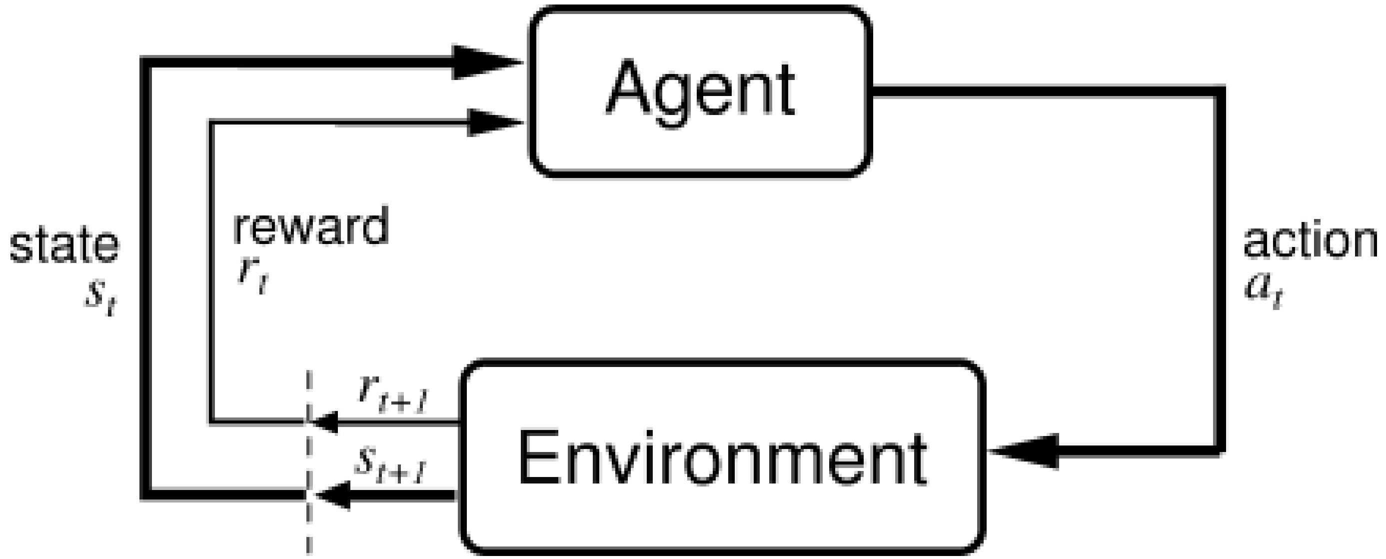
So far:
Intro to RL

Today:
MDPs, value iteration, policy iteration

Slides based heavily on course material by David Silver and Eric Eton

Reinforcement Learning

- Basic idea:
  - Receive feedback in the form of rewards
  - Agent's utility is defined by the reward function
  - Must (learn to) act so as to maximize expected rewards



RL agent components

Policy:

$$a = \pi(s)$$

$$\pi(a|s) = \mathbb{P}[A_t = a | S_t = s]$$

Reward Function:

$$\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

Value Function:

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

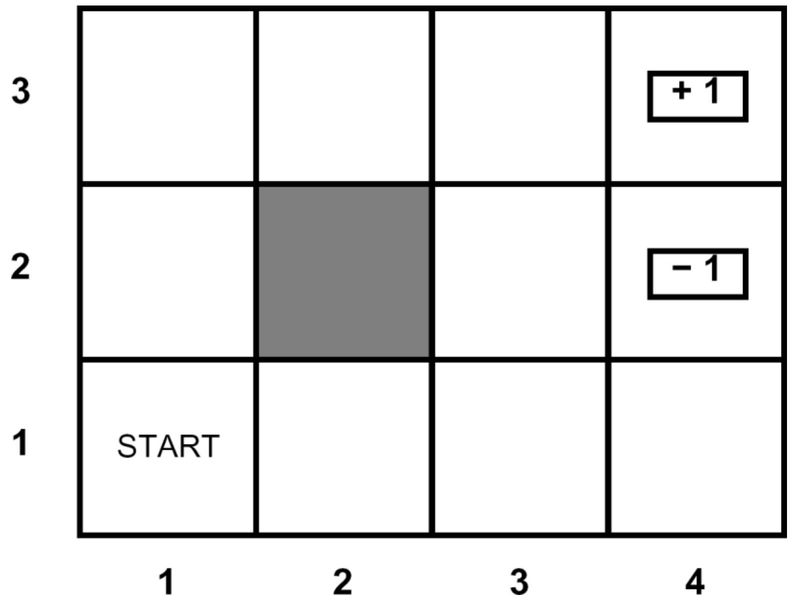
Model:

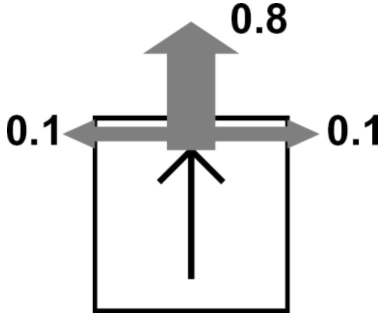
$$\mathcal{P}_{ss'}^a = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} \mid S_t = s, A_t = a\right]$$

Grid World

- The agent lives in a grid
- Walls block the agent's path
- Small "living" reward each step
- Big rewards come at the end
- Goal: maximize sum of rewards*
- The agent's actions do not always go as planned:
  - 80% of the time, the action North takes the agent North
    - (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put





Markov Decision Processes

- Markov decision processes: (S, A, P, γ , R)
  - States s $\in$ S
  - Actions a $\in$ A
  - State transition function P
  - Discount factor γ
  - Reward Function R

RL problem :

Goal :

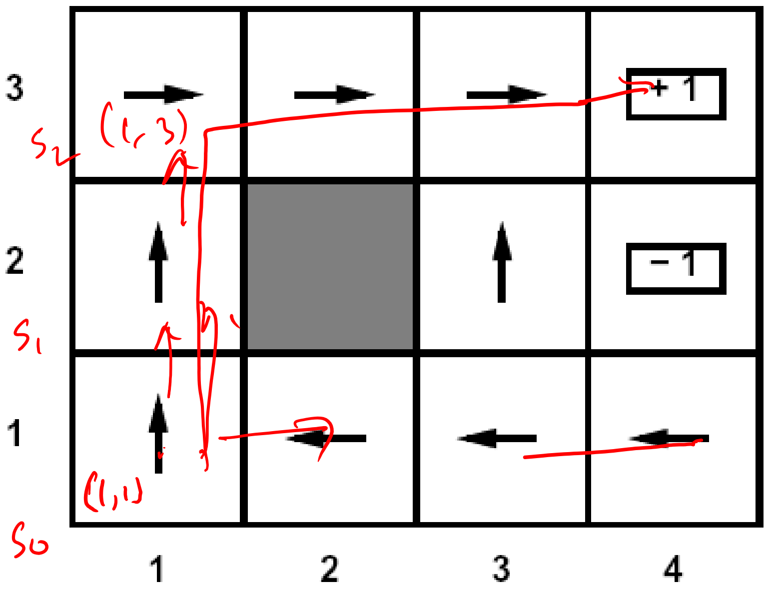
max $V_\pi(s)$

by using policy π : s →a

Solving MDPs

- In an MDP, we want an optimal policy $\pi^*: S \rightarrow A$
  - A policy π gives an action for each state
  - An optimal policy maximizes expected utility if followed

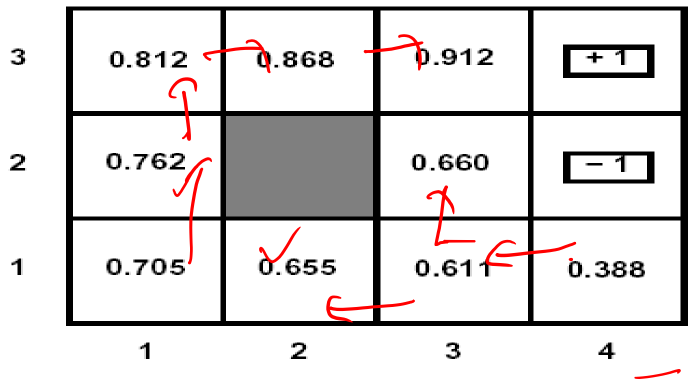
Optimal policy when $R(s, a, s') = -0.03$ for all non-terminals s

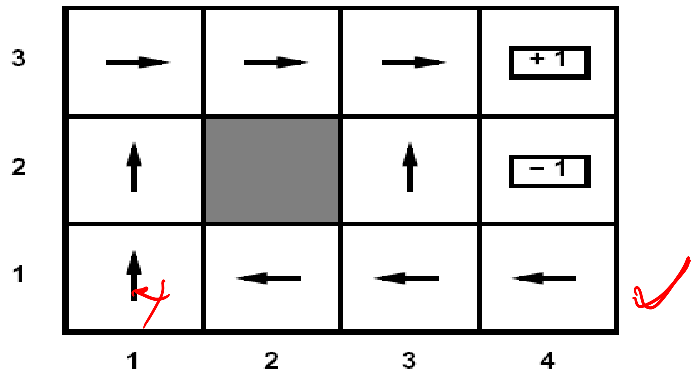


Optimal Policy and Value function

- Define the optimal value function:
  $V^*(s)$ = expected reward starting in s and acting optimally

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





The Bellman Equations

- Value function can be given as
  - Immediate reward
  - Discounted value of successor state

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

$$\begin{aligned}
v_\pi(s) &= \mathbb{E}_\pi \left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \dots\right) \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right]
\end{aligned}$$

The Bellman Equations

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

$S_0 \rightarrow R(S_0)$

$\gamma V_\pi(S_1)$

$P^{a}_{S_0 S_1}$ $\pi(S_0)$

- Can be rewritten as

$$v_\pi(s) = R(s) + \gamma \sum_{s'} P_{ss'}^{\pi(s)} \; v_\pi(s')$$

s s'

Solving for $v_\pi(s)$ given π, we get a linear system of equations in terms of $v_\pi(s)$

The Bellman Equations

- Optimal value function $V^*(s)$ is the maximum value function over all policies

  - $v^*(s) = \max_{\pi} \; v_\pi(s)$

    $v^*(s) = R(s) + \max_{a} \gamma \sum_{s'} P_{ss'}^{\pi(s)} v^*(s')$

The Bellman Equations

- Optimal policy π^*

$$\pi^*(s) = \underset{\pi}{argmax} \;\; v_\pi(s)$$

$$= \underset{a}{\text{argmax}}\, \gamma \;\; \textstyle\sum_{s'} P_{ss'}^{\pi(s)}\, v^*(s')$$

- Strategy for finding optimal policy
  - Find v^*
  - Use argmax to find π^*

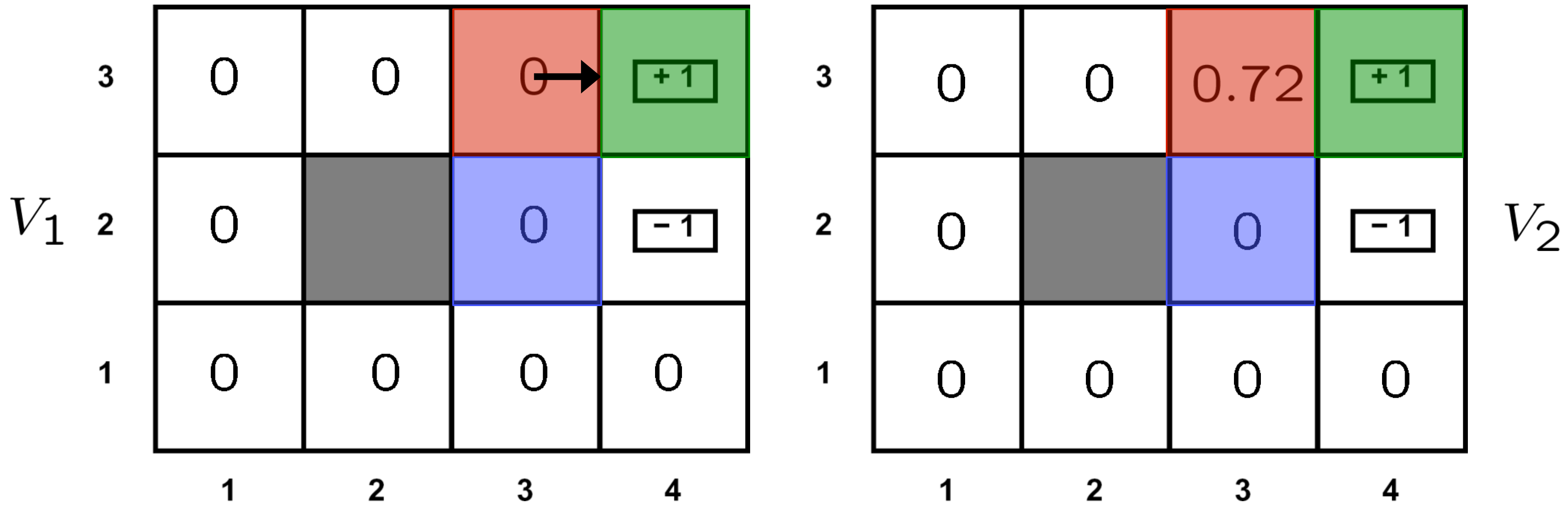
Compute v^* (value iteration algorithm)

- Initialise v(s) := 0 for all states s
- For every state s, update

$$v_\pi(s) = R(s) + \gamma \sum_{s'} P_{ss'}^{\pi(s)} \; v_\pi(s')$$

- Can be done in synchronous /asynchronous
- Value iteration algorithm allows $v\,(S)$ to converge to $v^*(S)$

Example: Bellman Updates



Example: γ=0.9, living reward=0,

$$= 0.9\,[0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0]$$

Example: Value Iteration

V_2

3	0	0	0.72	+1
2	0		0	−1
1	0	0	0	0
	1	2	3	4

V_3

3	0	0.52	0.78	+1
2	0		0.43	−1
1	0	0	0	0
	1	2	3	4

Information propagates outward from terminal states and eventually all states have correct value estimates

Compute π^* (policy iteration algorithm)

- Initialise π randomly
- Repeat:
  - Solve $v := v_\pi$ (i.e. solve Bellman's equ)
  - Set $\pi(s) := \underset{a}{\text{argmax}}\, \gamma \sum_{s'} P_{ss'}^{\pi(s)} v^*(s')$

- Converges to optimal policy

When state transition probability *P* is unknown

- $Estimate\ P_{ss'}^{a}$
- $P_{ss'}^{a} = \dfrac{\text{no. of times agent took action a in state } s \text{ and got to } s'}{\text{no. of times agent took action a in state } s}$

Or

= 1/\s\ if above is "0"/"0" (no such action was taken)

Putting it all together

- Repeat{
  - Take action wrt π to get experience in MDP
  - Update estimates of p_s^a
  - Solve Bellman’s equation using value iteration to get v
  - Update $\pi(s)$

  }

When Reward is unknown

- Use the previous algorithm to estimate both state transition prob. And rewards
- But the alg might end up in local optima
  - Solution: Use exploration vs exploitation