COMS W4701: Artificial Intelligence

Lecture 5: Sequential Decision Problems

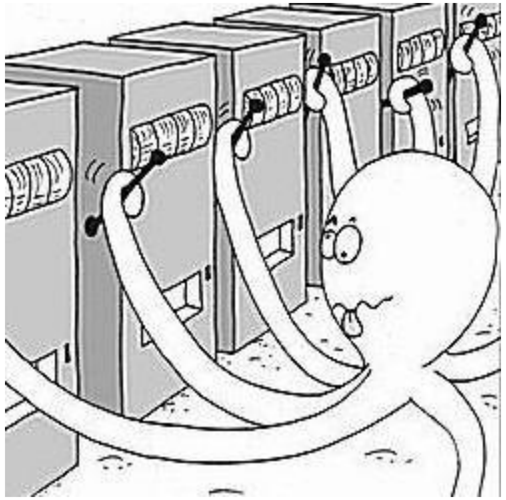
Tony Dear, Ph.D.
Department of Computer Science
School of Engineering and Applied Sciences

Today

- Multi-armed bandit problems
- Exploration vs exploitation
- UCB action selection

- Markov decision processes
- Utilities, discounting, values, and policies
- Bellman equations

Multi-Armed Bandits



- Thought experiment: Suppose we have several slot machines
- Each machine has different rewards and different odds
- We can only find out by trying different machines

- Tradeoff between **exploration** and **exploitation**
  - Gather more information or maximize best rewards so far?
  - How to determine when current knowledge is good enough?

- Applications: Resource allocation for maximizing productivity, clinical trials to explore different treatments, financial portfolio design

Action Values

- Choosing a slot machine is choosing an *action* a
- A_t is the action chosen and R_t is the resultant reward at time t
- **Action value** is the *expected* reward of an action: $Q^*(a) = E[R_t | A_t = a]$

- If we know all Q^*, optimal strategy would be to pick a with highest Q^*
- Idea: Build *estimates* of Q^* by trying different actions and recording results

- First initialize all $Q_0(a)$, e.g. by trying each action once and recording reward

- *Sample-averaging*: $Q_t(a) = \dfrac{\text{sum of rewards from taking } a \text{ prior to } t}{\text{number of times taking } a \text{ prior to } t}$

Action Selection

- How to select actions to try while building up Q estimates?
- *Greedy* action selection always exploits: $A_t = \mathrm{argmax}_a\, Q_t(a)$

- Problem: We would never explore new actions
- What if there are better options out there?

- ***ε*-greedy**: Select greedy action *most* of the time, but with small probability ε, pick a random action to *explore* instead

- In the limit, estimates Q_t will converge to true Q^*

Controlling Exploration

- While exploration is *necessary* to learn action values, it is usually **not** optimal
- Too much exploration will lower total rewards received
- Example: Bandits with deterministic rewards only require one round of exploration

- Even for stochastic problems, we probably don't need to explore *forever*
- Better idea: Explore more beginning (higher ε), then gradually taper off (lower ε)

- Another problem: When exploring, ε-greedy picks actions completely randomly
- Can we do targeted exploration, e.g. pick actions that are more promising?

Upper Confidence Bound

- Idea: Target actions that have high *potential* for being optimal

- The *confidence interval* of a value estimate specifies range of true values
- Interval (estimate uncertainty) shrinks as we try an action more often

- **Upper confidence bound** action selection: $A_t = \text{argmax}_a \left[Q_t(a) + \underbrace{c\sqrt{\ln t \,/ N_t(a)}}_{UCB_t(a)} \right]$
  - $N_t(a) =$ number of times a has been taken prior to t
  - $\sqrt{1/N_t(a)}$ proportional to standard deviation
  - c is a parameter that controls the amount of exploration

- As $N_t(a)$ increases, $UCB_t(a)$ approaches 0 due to increased confidence

Sequential Decision Problems

- Suppose we require a *sequence* of decisions in a *stochastic* environment

- Like stochastic games, we will have a *stochastic* transition model
- Unlike in games, rewards can be received at any time (not just the end)

- The problems will be single-agent, sequential, stochastic, fully observable

- A step-by-step *plan* no longer works due to stochasticity
- Instead, we form a *policy*: a mapping from states to actions

Markov Decision Processes

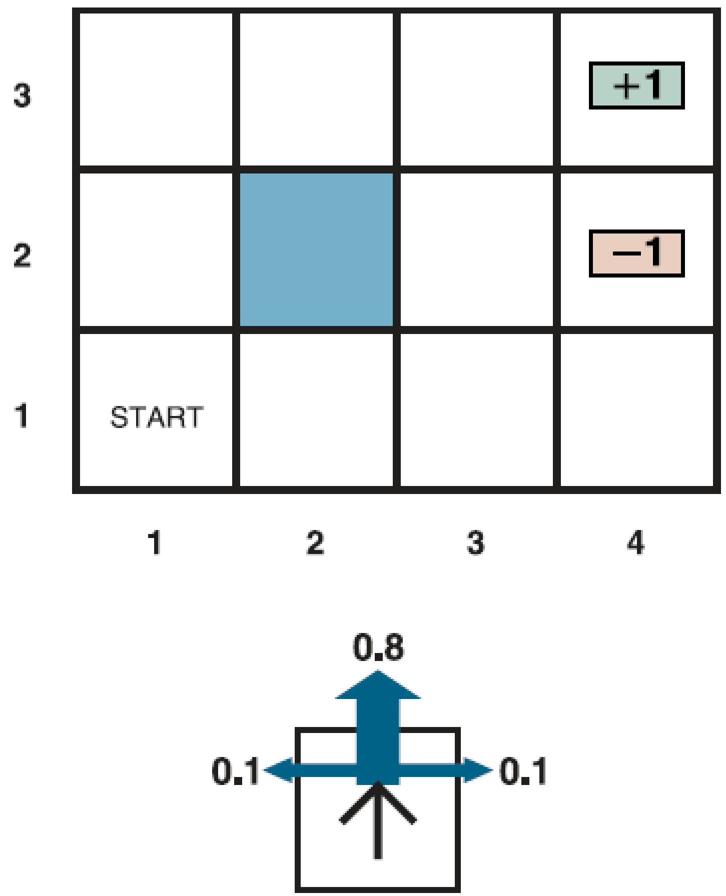
- Bandit problems are decision problems without states
- In **Markov decision processes**, agents take actions, move from state to state according to a transition function, and receive rewards along the way

- Set of states S and set of actions A
  - S may include initial and/or terminal states
- Transition function $T: S \times A \times S \to [0,1]$, where $T(s, a, s') = \Pr(s' | s, a)$
  - Also called the *model* or *dynamics* of the problem
- Reward function $R: S \times A \times S \to \mathbb{R}$, written as $R(s, a, s')$

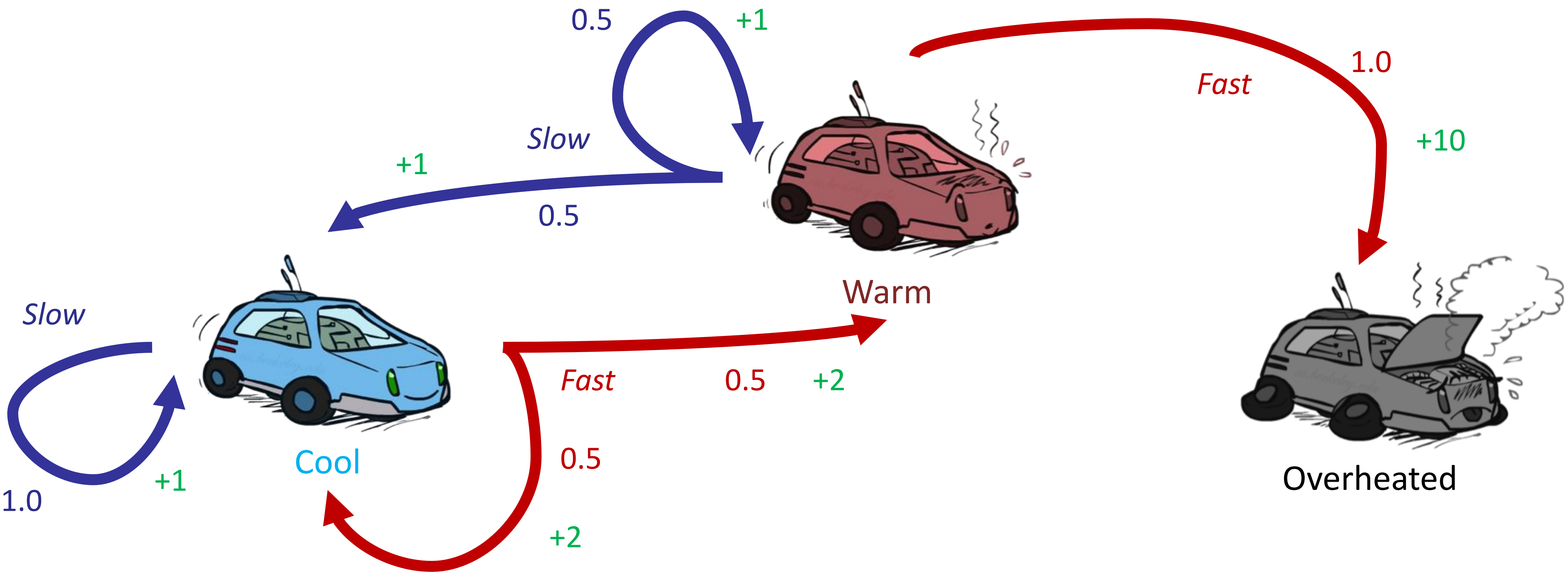
- MDPs are **Markovian** in that we only need to track current, not past, states

Example: Gridworld

- States: Grid locations (11 in total)
  - First two states in column 4 are **terminal states**
- Actions: North, south, east, west (4 for each state)
  - May stay in original state if moving into wall
- Transition function: *Intended* direction with prob 0.8; otherwise, slip left or right with prob 0.1, respectively
- Rewards: +1 or -1 for moving into terminal states; small negative reward otherwise (*living reward*)



Example: Race Car MDP



MDPs in Practice

- Agriculture
  - S: Soil condition and precipitation forecast. A: Whether or not to plant a given area.
- Water resources and energy generation
  - S: Water levels and inflow. A: How much water to use to generate power.
- Inspection and maintenance
  - S: System age and probability failure. A: Whether to test / restore / repair a system.
- Inventory
  - S: Inventory levels and commodity prices. A: How much to purchase.
- Finance and investment
  - S: Holding or capital levels. A: How much to invest.

- Many, many more (D. J. White 1993)

Utilities

- A sequence of states and actions can be quantified by its **utility**
- *Rational* agent seeks to maximize its utility over time

- **Finite-horizon** MDPs: Process ends after some finite time T
  - Equivalent to entering a *terminal state* s_T
  - One way to define utility of a state-action sequence: sum up individual rewards

$$V_h([s_0, a_0, s_1, a_1, \dots, a_{T-1}, s_T]) = \sum_{t=0}^{T-1} R(s_t, a_t, s_{t+1})$$

- This won't work for **infinite-horizon** MDPs!

Discounted Utilities

- Which reward sequence is better? $R_1 = (1,1,1)$ vs $R_2 = (0,0,3)$

- Sums are equal, but R_1 is preferable if rewards *now* are better than rewards *later*
- Supported by psychology, economics; also due to uncertainty in obtaining rewards

- Idea: Apply a **discount factor** $0 < \gamma < 1$ to diminish future rewards
- Utility is computed using **additive discounted rewards**

$$V_h([s_0, a_0, s_1, a_1, \ldots, a_{T-1}, s_T]) = \sum_{t=0}^{T-1} \gamma^t R(s_t, a_t, s_{t+1})$$

Discounting for Infinite-Horizon MDPs

- Additive utilities for finite-horizon MDPs use $\gamma = 1$

- For infinite-horizon MDPs, $\gamma < 1$ allows utility sequences to converge
  - The closer γ is to 0, the less we value the future
  - Rewards far in the future are effectively zero

$$V_h([s_0, a_0, s_1, \dots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \leq \frac{R_{\max}}{1 - \gamma}$$

- Ex: Utility of infinite sequence of $+1$ rewards with $\gamma = 0.8$:

$$V_h = 1 + (0.8)1 + (0.8)^2 1 + \cdots = \frac{1}{1 - 0.8} = 5$$

Policies and Value Functions

- Solving MDP means finding a **policy**—mapping from states to actions
- $\pi: S \rightarrow A$ tells agent what to do in any state

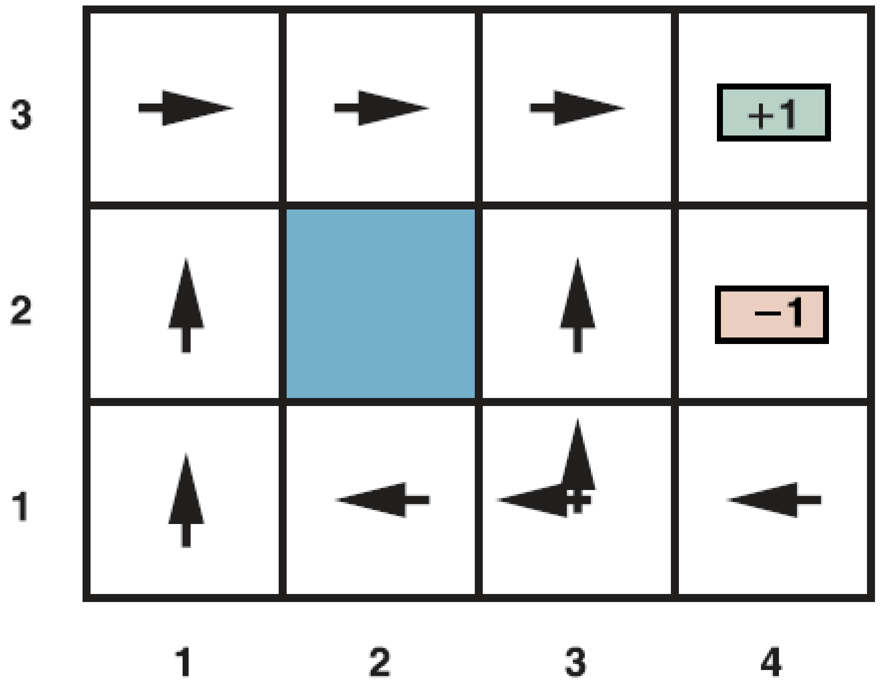
- Policies can be quantified by **value functions**
- $V^{\pi}: S \rightarrow \mathbb{R}$ is *expected* utility of following π starting from given state

$$V^{\pi}(s) = E\left[\sum_{t=0} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right], s_0 = s$$

- **Optimal** policy and value function:

$$\pi^* = \operatorname{argmax}_{\pi} V^{\pi} \qquad V^* = \max_{\pi} V^{\pi}$$

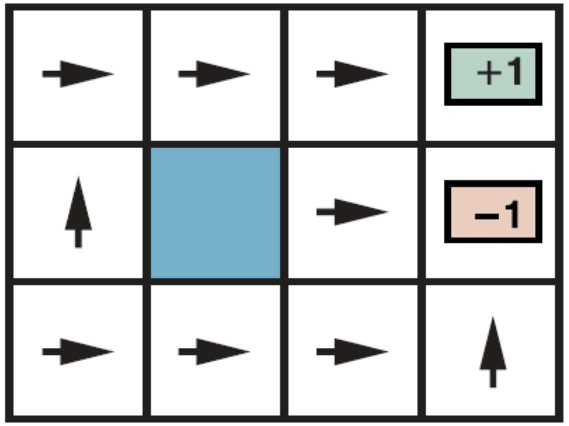
Gridworld Policy and Value Function



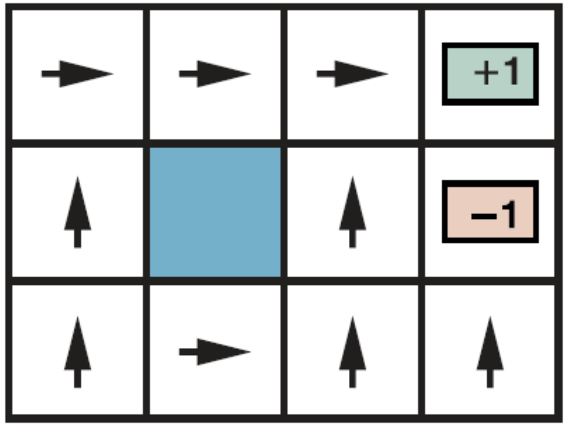
	1	2	3	4
3	0.8516	0.9078	0.9578	+1
2	0.8016		0.7003	−1
1	0.7453	0.6953	0.6514	0.4279

$R(s) = -0.04$ for nonterminal states
$\gamma = 1$ (no discounting)

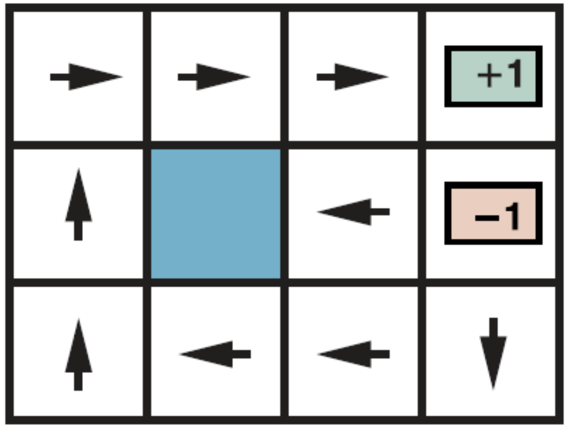
Rewards and Policies



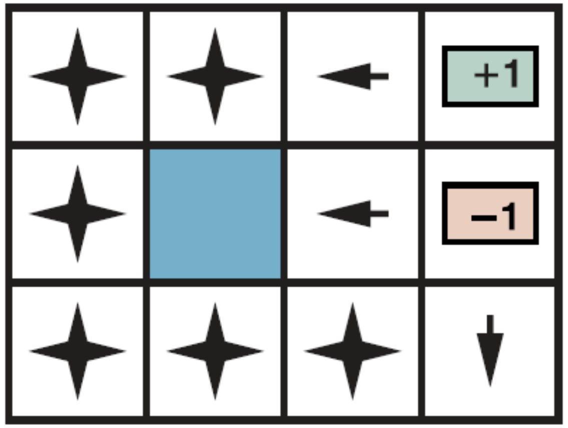
$r < -1.6497$



$-0.7311 < r < -0.4526$



$-0.0274 < r < 0$



$r > 0$

Recursive Relationship

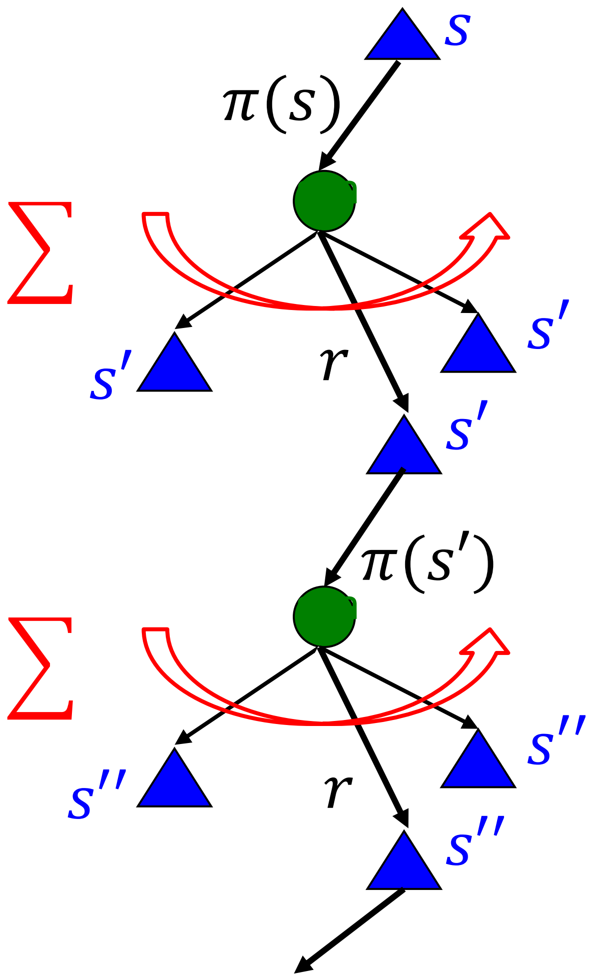
$$V^{\pi}(s) = E\left[\sum_{t=0} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right]$$

- This is a function on every state in the state space

- For a given state s, we can alternatively write $V^{\pi}(s)$ as a *recursive* function of *successor state values*

$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

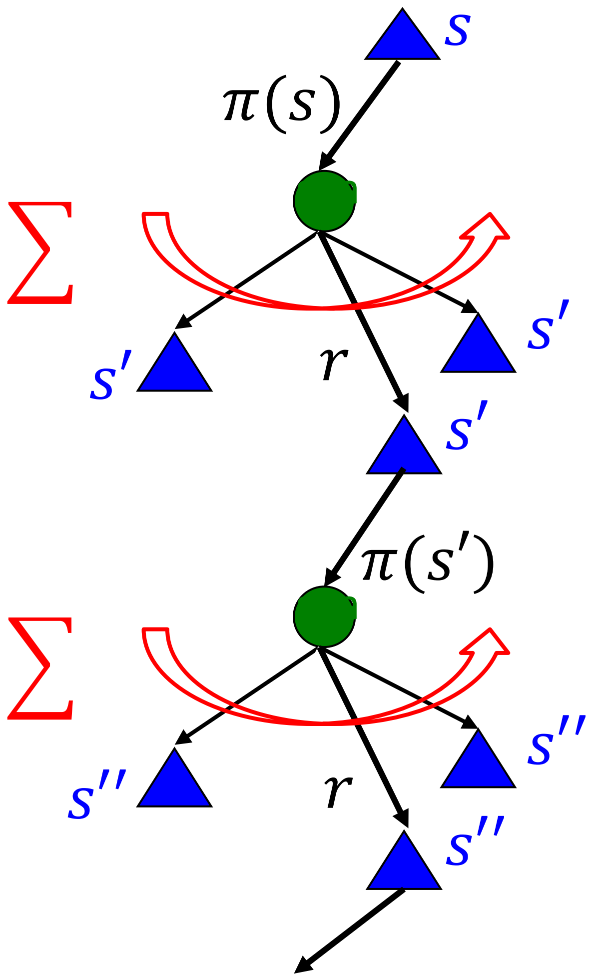
- $V^{\pi}(s)$ is a *weighted average* of (immediate reward plus discounted successor state values)



Solving for Values

$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

- Suppose we know the model (T, R), discount γ, and a fixed policy π
- Then the above is a system of $|S|$ *linear* equations in the $|S|$ unknowns $V^{\pi}(s)$
- Linear solvers: ~$O(|S|^3)$ time, can find $V^{\pi}(s)$



Example: Mini-Gridworld

$$V^\pi(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^\pi(s')]$$

- States: A, B, C; actions: left, right; $\gamma = 0.5$
- Policy: $\pi(s) = \text{left} \ \forall\ s$
- Rewards: $R(s, a, A) = +3, R(s, a, B) = -2, R(s, a, C) = +1$
- Transitions: $\Pr(\text{intended direction}) = 0.8, \Pr(\text{opposite direction}) = 0.2$

+3	-2	+1
A	B	C

- V^π can be found by solving 3 linear equations:

$$V^\pi(A) = 0.8\big(3 + 0.5V^\pi(A)\big) + 0.2(-2 + 0.5V^\pi(B))$$

$$V^\pi(B) = 0.8\big(3 + 0.5V^\pi(A)\big) + 0.2(1 + 0.5V^\pi(C))$$

$$V^\pi(C) = 0.8\big(-2 + 0.5V^\pi(B)\big) + 0.2(1 + 0.5V^\pi(C))$$

Bellman Optimality Equations

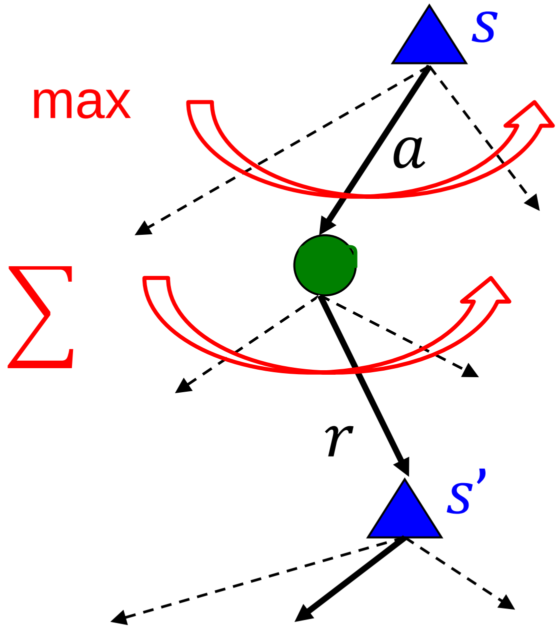
- Our goal is to find an **optimal policy** or **optimal value function**

$$V^\pi(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$$

$$\pi^* = \text{argmax}_\pi V^\pi$$
$$V^* = \max_\pi V^\pi$$

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

$$\pi^*(s) = \underset{a}{\text{argmax}} \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$



- **Bellman optimality equations** are nonlinear!

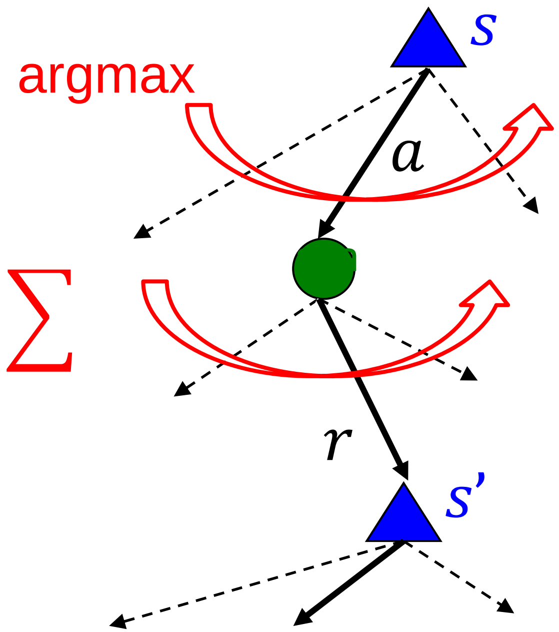
Value Function to Policy

- We don't know (yet) how to solve for V^* from scratch
- We *do* know how to find V^* given π^* (solve linear system)

- Bellman equation tells us how to find π^* given V^*

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

- Everything on the RHS is known!
- Solving for complete policy takes $O(|S|^2|A|)$ time



Example: Mini-Gridworld

$$\pi^*(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$$

+3	-2	+1

- States: A, B, C; actions: left, right
- Transitions and rewards same as before

- Given $V^*(A) = 4.06, V^*(B) = 4.36, V^*(C) = 1.39$
- Find $\pi^*(B)$:

$\gamma = 0.5$

$$\pi^*(B) = \operatorname{argmax} \begin{cases} 0.8\big(3 + 0.5V^*(A)\big) + 0.2\big(1 + 0.5V^*(C)\big) & \text{Left} \\ 0.8\big(1 + 0.5V^*(C)\big) + 0.2(3 + 0.5V^*(A)) & \text{Right} \end{cases}$$

Summary

- In sequential decision problems, an agent must take actions over and over
- Multi-armed bandits: No states, multiple actions; agent must often contend with exploration vs exploitation

- More general decision problems: Markov decision processes
  - States, actions, transitions, rewards, (discount factor)

- Goals -> rewards -> utilities -> value functions and policies
- Bellman optimality equations: Recursive and nonlinear