Decision Making and Reinforcement Learning

Module 3: Markov Decision Processes

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

Topics

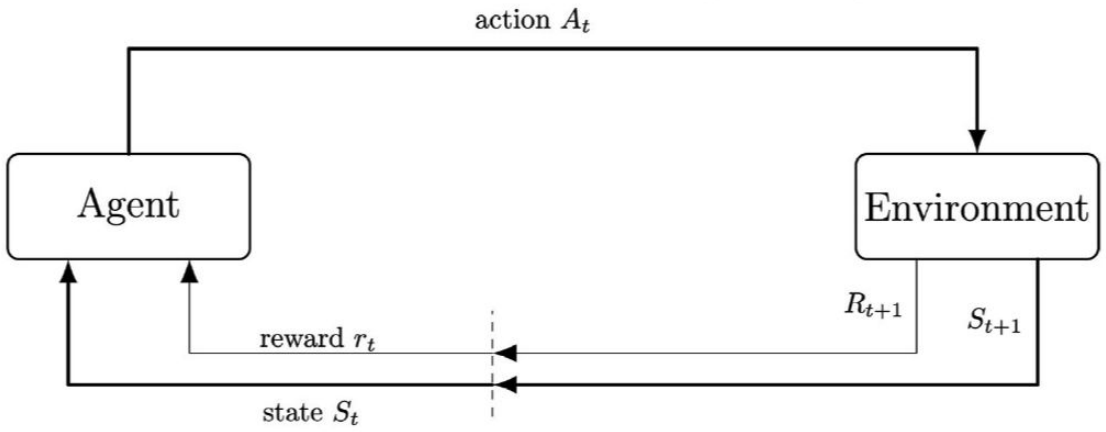
- Markov decision process framework
- Rewards, utilities, and discounting
- Policies and value functions
- Bellman equations

Learning Objectives

- **Model** a sequential decision problem as a MDP
- **Predict** and **explain** effects of rewards and discounts in MDPs
- **Define** value and policy functions for a MDP
- **Write** the Bellman equations for optimal values and policies

Agent-Environment Interface

- An **agent** can interact with its **environment** by performing *actions*
- The result of the action may change the **state** of the system
- The agent can receive feedback in the form of *rewards*



Markov Decision Processes

- A **Markov decision process (MDP)** is a mathematical model for a sequential decision problem with uncertainty
- It can be used to help quantify optimal decision making

Markov Decision Processes

- A **Markov decision process (MDP)** is a mathematical model for a sequential decision problem with uncertainty
- It can be used to help quantify optimal decision making

- **Components of a MDP**:

- State space S and action space $A(s)$ for each state

Markov Decision Processes

- A **Markov decision process (MDP)** is a mathematical model for a sequential decision problem with uncertainty
- It can be used to help quantify optimal decision making

- **Components of a MDP**:

- State space S and action space $A(s)$ for each state

- *Transition function* $T: S \times A \times S \to [0,1]$, where $T(s, a, s') = \Pr(s'|s, a)$

Markov Decision Processes

- A **Markov decision process (MDP)** is a mathematical model for a sequential decision problem with uncertainty
- It can be used to help quantify optimal decision making

- **Components of a MDP**:

- State space S and action space $A(s)$ for each state

- *Transition function* $T: S \times A \times S \rightarrow [0,1]$, where $T(s, a, s') = \Pr(s'|s, a)$

- *Reward function* $R: S \times A \times S \rightarrow \mathbb{R}$, written as $R(s, a, s')$

Uncertainty and the Markov Property

- The transition function captures *uncertainty* and *stochasticity*
- Since $T(s, a, s') = \Pr(s'|s, a)$, we have that $\sum_{s'} T(s, a, s') = 1$

Uncertainty and the Markov Property

- The transition function captures *uncertainty* and *stochasticity*
- Since $T(s, a, s') = \Pr(s'|s, a)$, we have that $\sum_{s'} T(s, a, s') = 1$

- **Markov property**: Transitions depend on a finitely many previous states

- This extends to decision making, where current actions should not depend on the history of states

Uncertainty and the Markov Property

- The transition function captures *uncertainty* and *stochasticity*
- Since $T(s, a, s') = \Pr(s'|s, a)$, we have that $\sum_{s'} T(s, a, s') = 1$

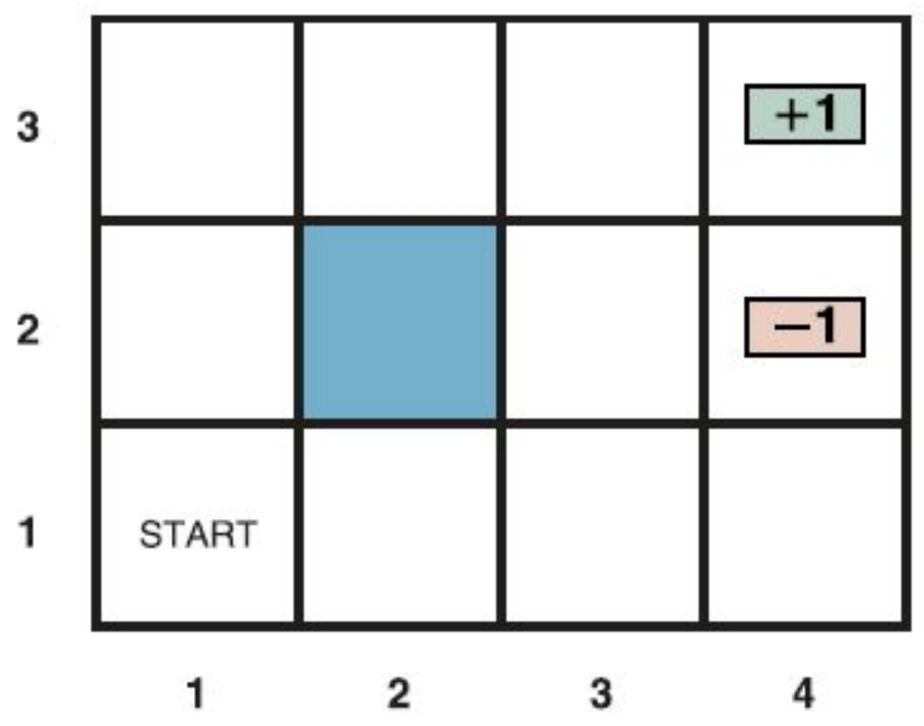
- **Markov property**: Transitions depend on a finitely many previous states

- This extends to decision making, where current actions should not depend on the history of states

- The triplet (s, a, s') may be collectively referred to as a *transition*
- Starting state, action taken, successor state

Gridworld Example

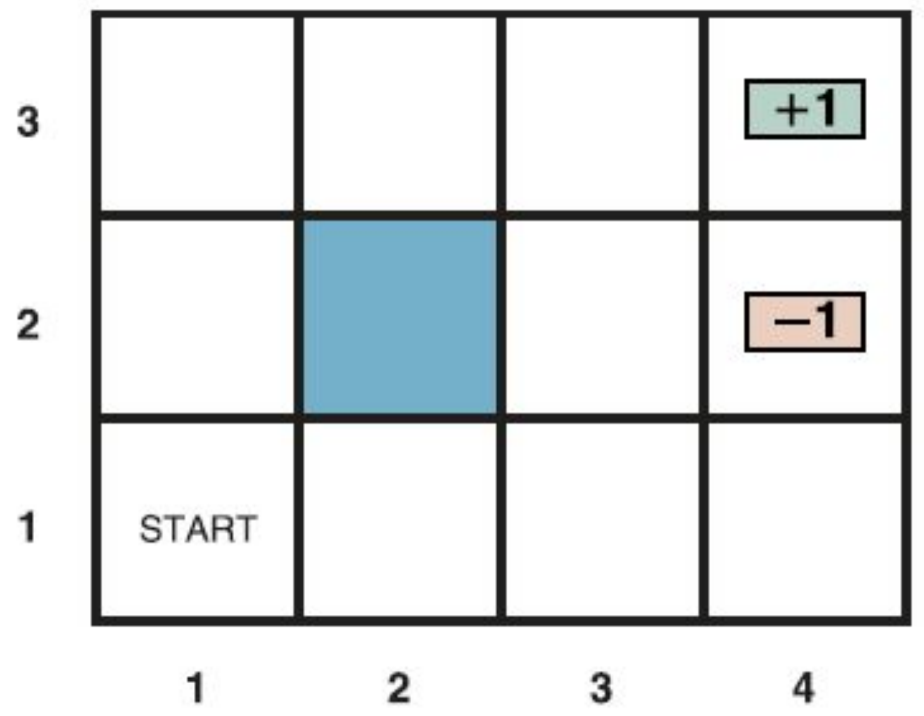
- States: Grid locations (11 in total)
- Cell (2,2) is a "wall" and is not achievable



Gridworld Example

- States: Grid locations (11 in total)
- Cell (2,2) is a "wall" and is not achievable

- Actions: North, south, east, west
- Available in most states, except…

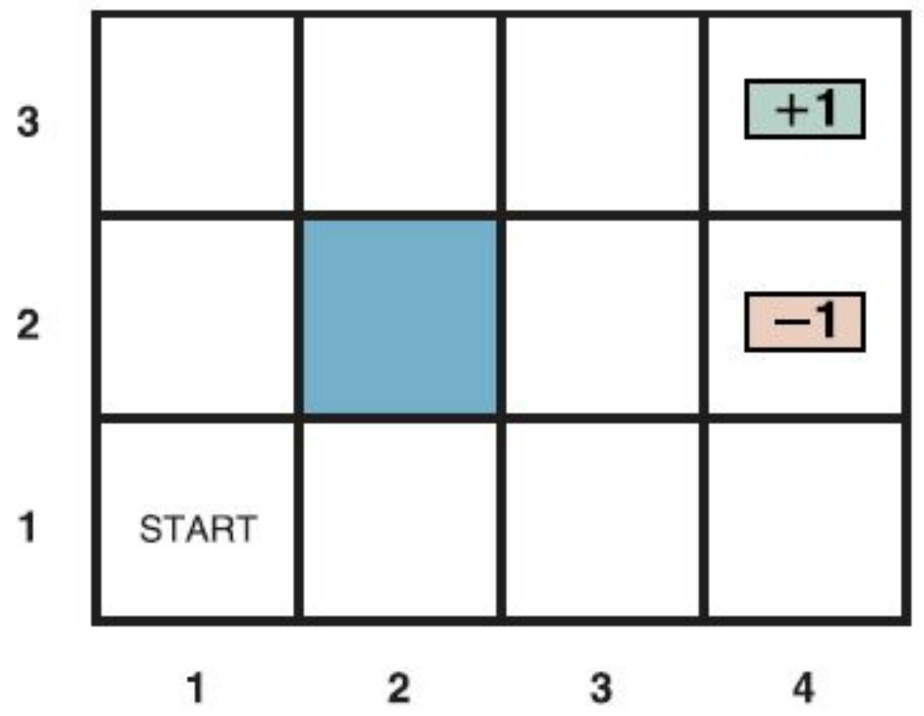


Gridworld Example

- States: Grid locations (11 in total)
- Cell (2,2) is a “wall” and is not achievable

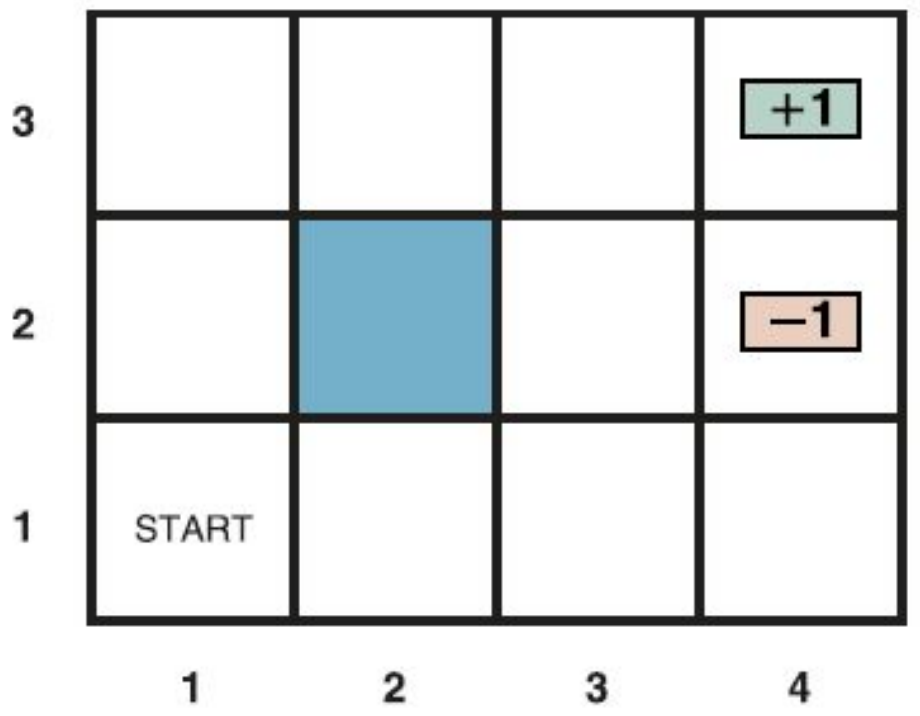
- Actions: North, south, east, west
- Available in most states, except…

- *Terminal* states (4,2) and (4,3)
- No actions from either state



Gridworld Example

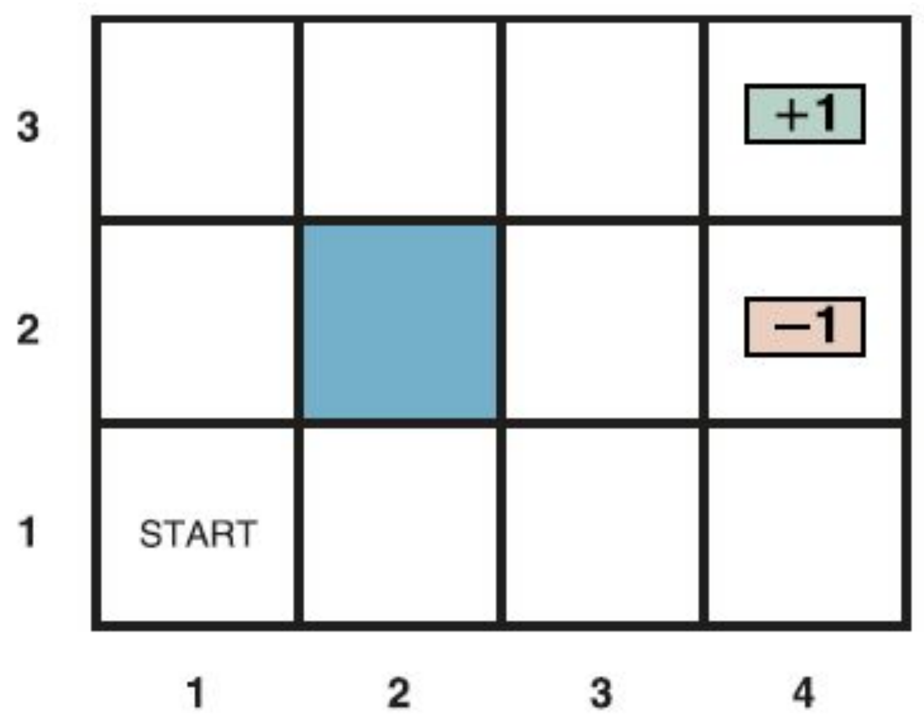
- Transition function: Agent ends up in the "expected" successor state *most* of the time

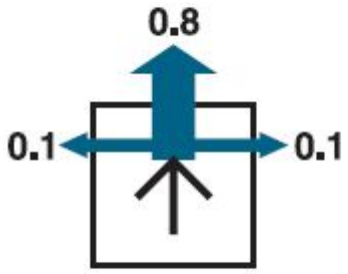


Gridworld Example

- Transition function: Agent ends up in the “expected” successor state *most* of the time

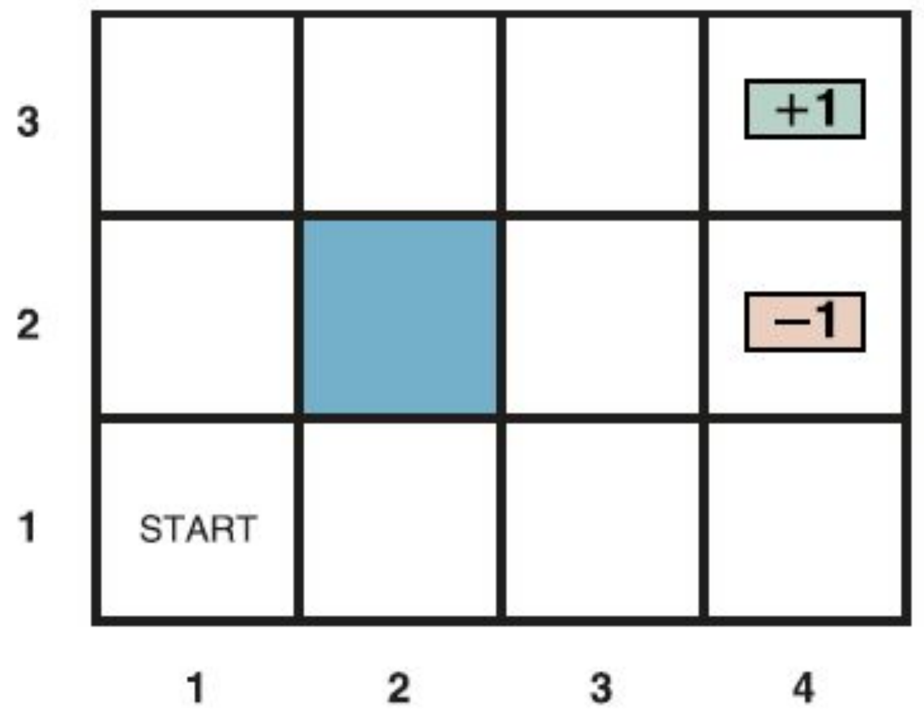
- Small probability that the agent ends up in the state to the “side” of the expected successor

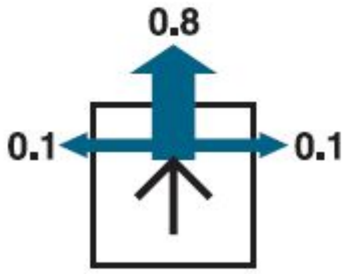




Gridworld Example

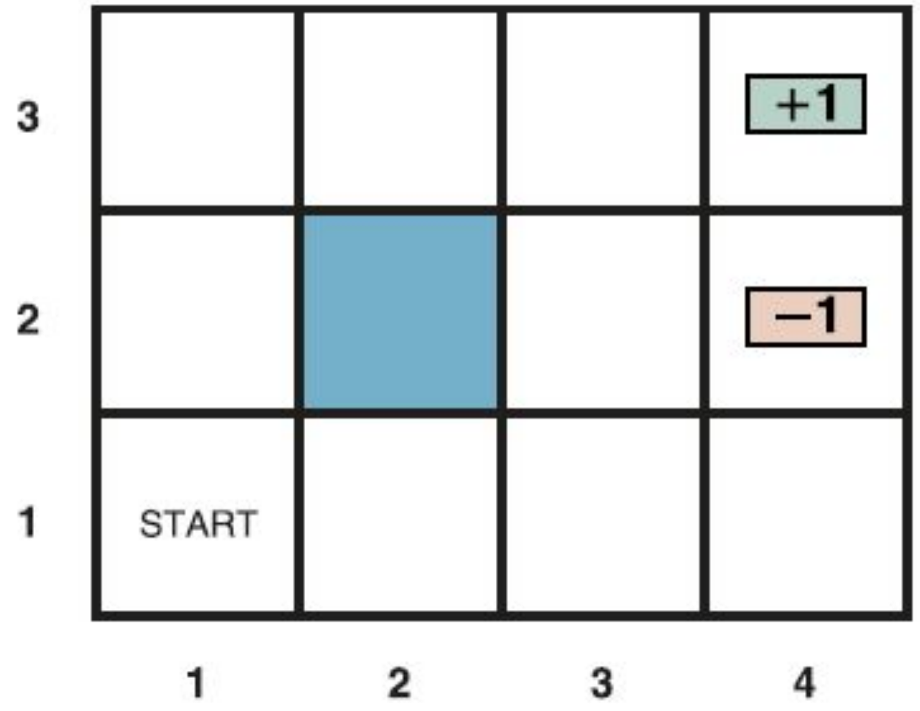
- Transition function: Agent ends up in the "expected" successor state *most* of the time
- Small probability that the agent ends up in the state to the "side" of the expected successor
- If the successor state is outside gridworld limits, agent simply remains in original state

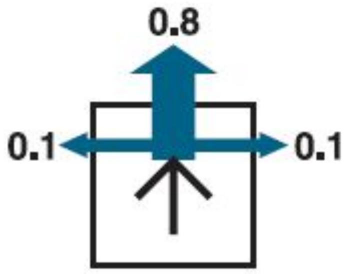




Gridworld Example

- Transition function: Agent ends up in the "expected" successor state *most* of the time
- Small probability that the agent ends up in the state to the "side" of the expected successor
- If the successor state is outside gridworld limits, agent simply remains in original state
- Reward function: ± 1 for entering respective terminal states; **living reward** received for all other transitions





Checkpoint: Gridworld Example

- Given the problem description, what are each of the following?

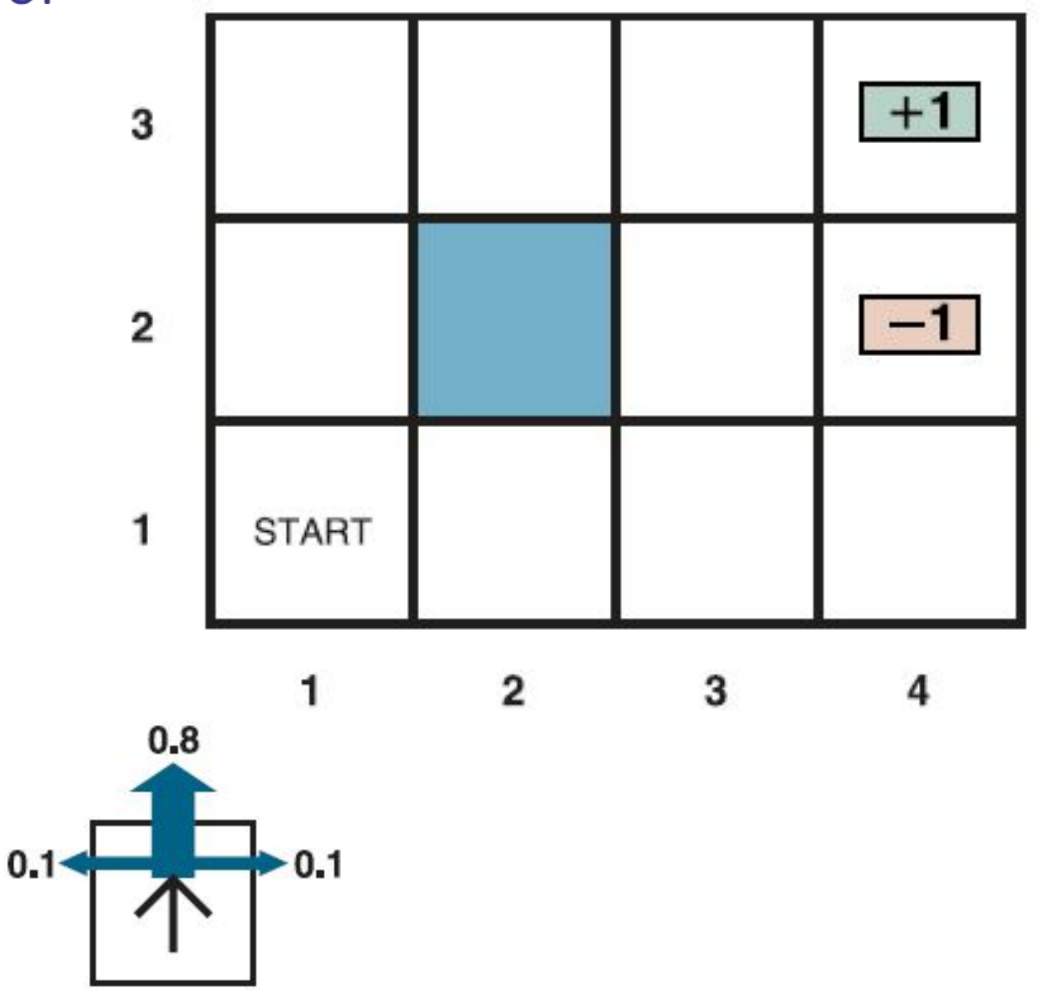
- $T\big((3,3), E, (4,3)\big) = ?$

- $T\big((3,2), S, (4,2)\big) = ?$

- $T\big((4,1), N, (4,1)\big) = ?$

- $R\big((3,3), N, (3,3)\big) = ?$

- $R\big((3,3), N, (4,3)\big) = ?$



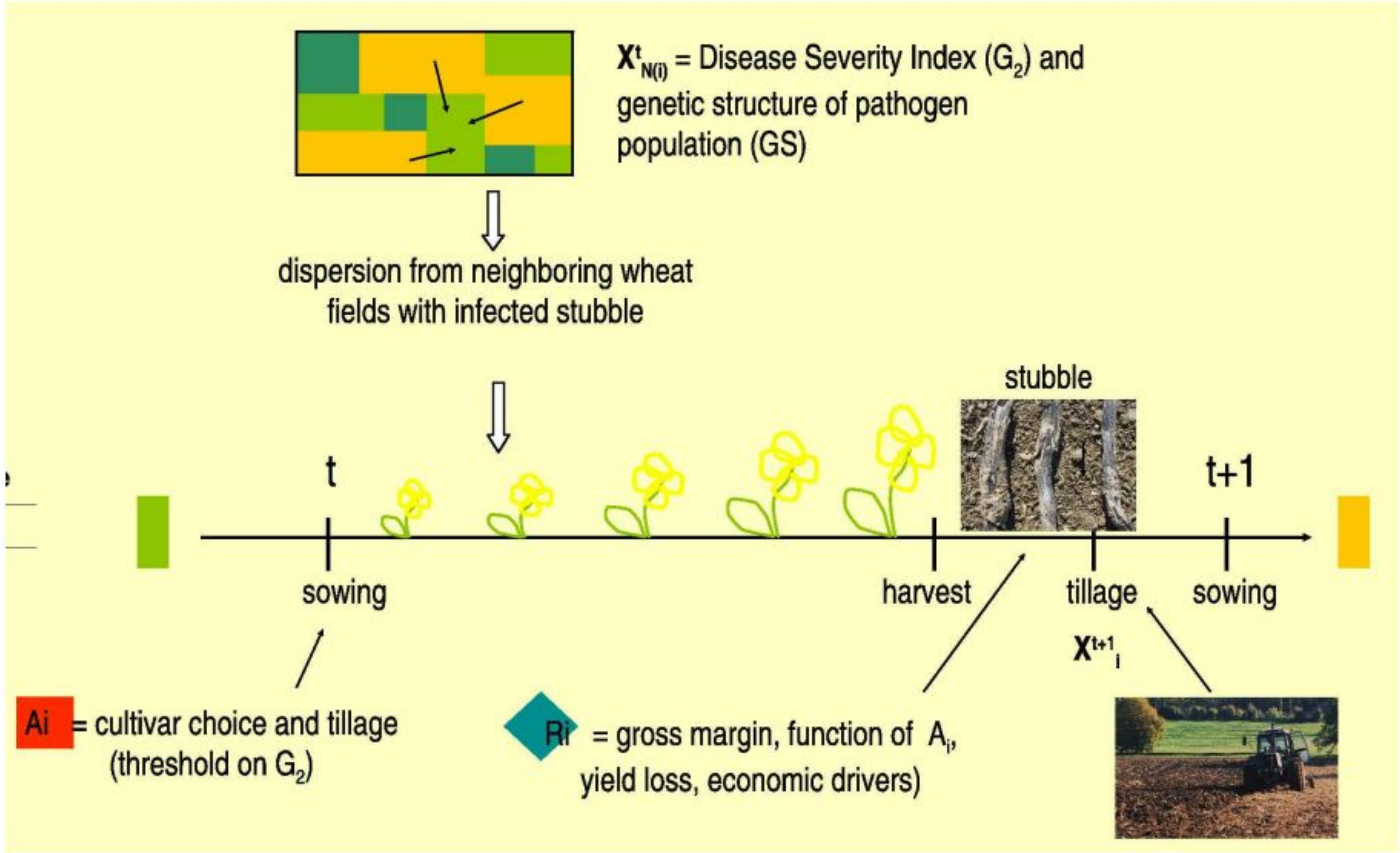
MDPs in Practice

- 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)

Example: Agricultural Disease Management

- Peyrard et al., 2007



Utilities

- We can define a utility for any given sequence of states and actions
- A *rational* agent seeks to maximize its utility

Utilities

- We can define a utility for any given sequence of states and actions
- A *rational* agent seeks to maximize its utility

- **Finite-horizon** MDP: Process ends after some finite time T
- Equivalent to entering a terminal state s_T

Utilities

- We can define a utility for any given sequence of states and actions
- A *rational* agent seeks to maximize its utility

- **Finite-horizon** MDP: Process ends after some finite time T
- Equivalent to entering a terminal state s_T

- One definition of utility of a state-action sequence: Sum all rewards!

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

Discounted Utilities

- Utilities can also account for the *timing* of when rewards are received

Discounted Utilities

- Utilities can also account for the *timing* of when rewards are received

- Example: Sums of reward sequences $R_1 = (1,1,1)$ and $R_2 = (0,0,3)$ are equal, but R_1 is preferable if rewards *now* are better than rewards *later*

Discounted Utilities

- Utilities can also account for the *timing* of when rewards are received

- Example: Sums of reward sequences $R_1 = (1,1,1)$ and $R_2 = (0,0,3)$ are equal, but R_1 is preferable if rewards *now* are better than rewards *later*

- Idea: Apply a **discount factor** $0 < \gamma < 1$ to *diminish* future rewards
- We can now compute a utility based on **additive discounted rewards**

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

Example: Additive Discounted Rewards

- Suppose we have an infinite sequence of rewards all equal to 2
- What is the **utility** of this sequence using a discount factor $\gamma = 0.8$?

Example: Additive Discounted Rewards

- Suppose we have an infinite sequence of rewards all equal to 2

- What is the **utility** of this sequence using a discount factor $\gamma = 0.8$?

$$V = \sum_{t=0}^{\infty} \gamma^t R = 0.8^0(2) + 0.8^1(2) + 0.8^2(2) + \cdots$$

Discounting for Infinite-Horizon MDPs

- Additive rewards with no discounting simply use $\gamma = 1$

Discounting for Infinite-Horizon MDPs

- Additive rewards with no discounting simply use $\gamma = 1$
- If we have infinitely many transitions, we *must* use a discount factor $\gamma < 1$ so that the rewards sum do not become unbounded

Discounting for Infinite-Horizon MDPs

- Additive rewards with no discounting simply use $\gamma = 1$

- If we have infinitely many transitions, we *must* use a discount factor $\gamma < 1$ so that the rewards sum do not become unbounded

- The *choice* of γ determines how myopic or forward-looking our agent is
- We can compute an upper bound on the additive reward as follows:

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

Policies and Value Functions

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

Policies and Value Functions

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

- We can quantify policies using **value functions**
- $V^{\pi}: S \to \mathbb{R}$ is the *expected* utility of following π starting from a given state

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

Policies and Value Functions

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

- We can quantify policies using **value functions**
- $V^{\pi}: S \rightarrow \mathbb{R}$ is the *expected* utility of following π starting from a given state

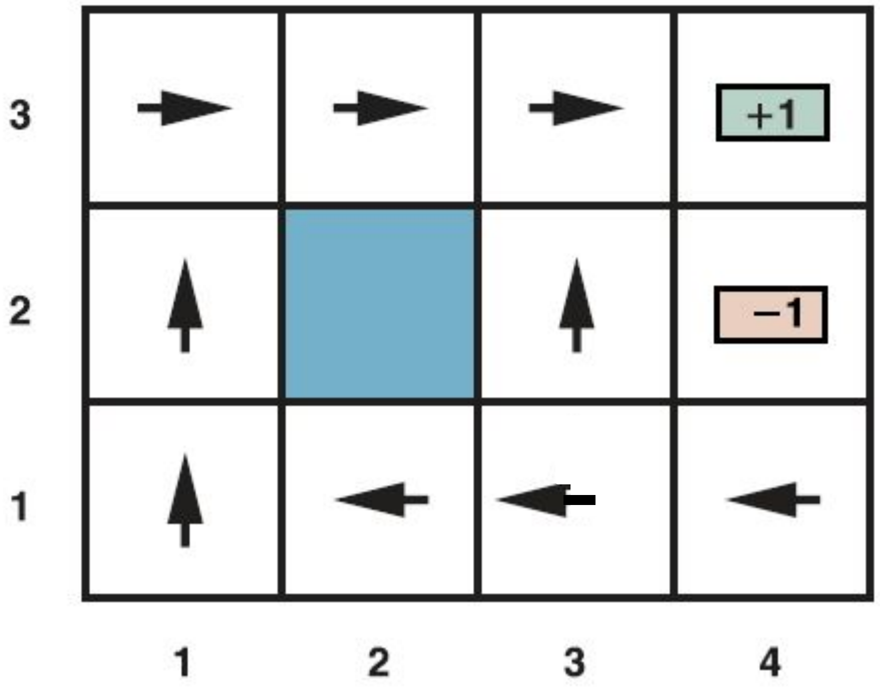
$$V^{\pi}(s) = E\left[\sum_{t=0} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right], s_0 = s$$

- We are generally interested in the **optimal** policy and value functions:

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

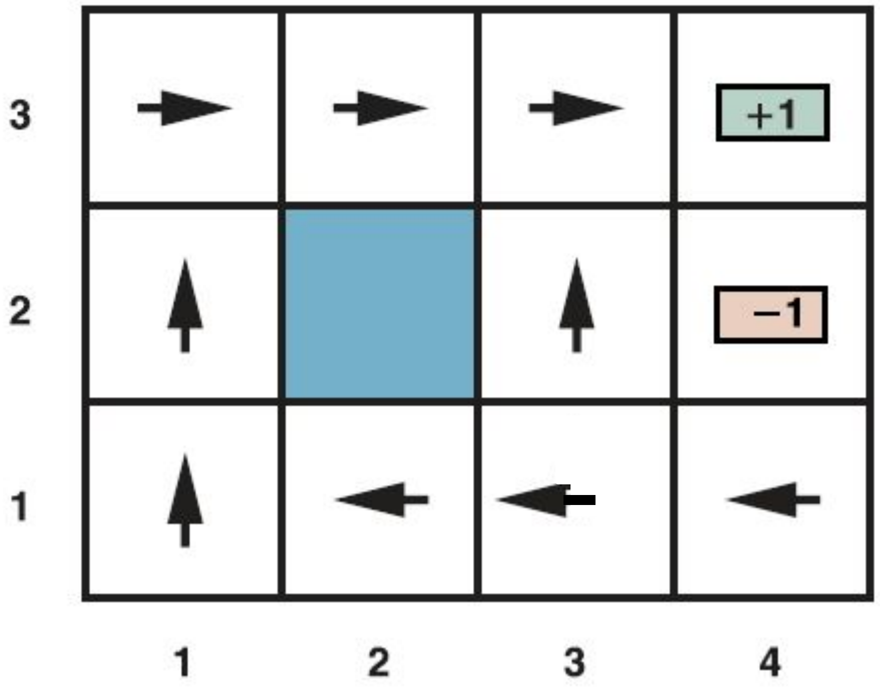
Gridworld Policy and Value Functions

Example policy:

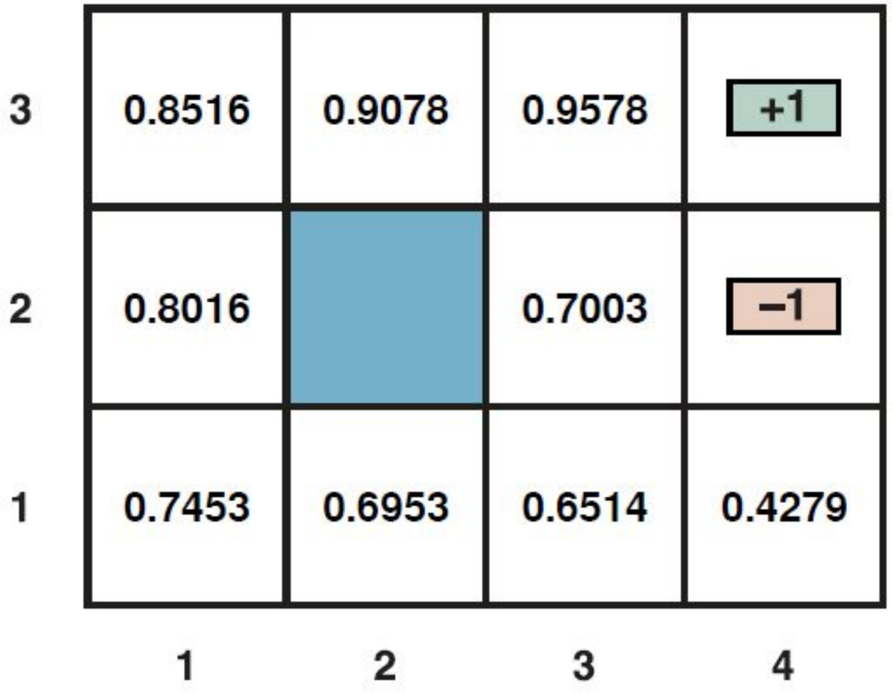


Gridworld Policy and Value Functions

Example policy:

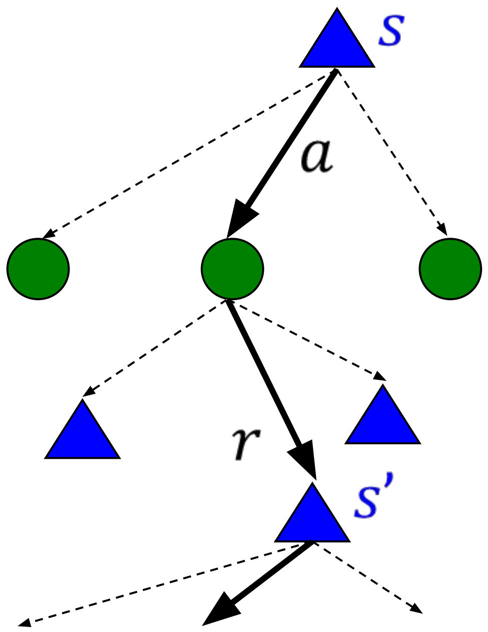


Example value function:



Backup Diagrams

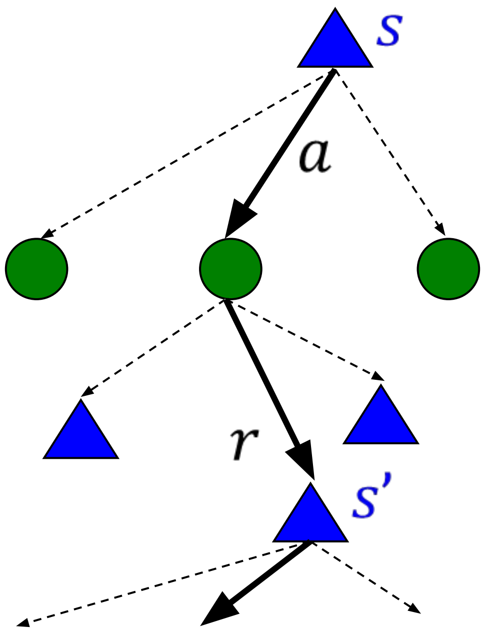
- We can visualize all states and actions visited by an agent using a tree called a **backup diagram**



Backup Diagrams

- We can visualize all states and actions visited by an agent using a tree called a **backup diagram**

- States are denoted by triangular nodes
- Each edge coming out of state corresponds to a possible action

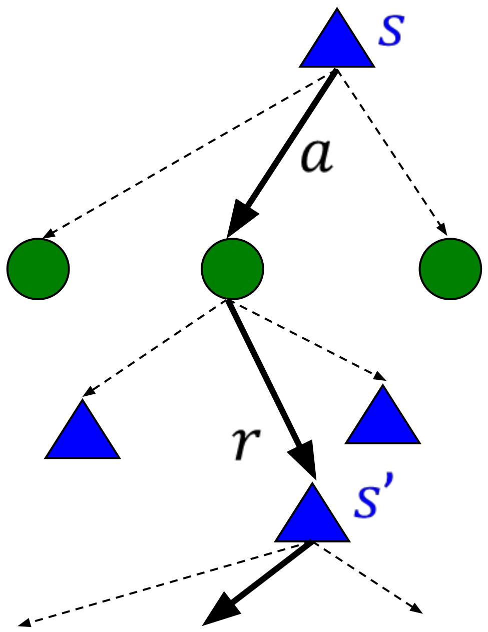


Backup Diagrams

- We can visualize all states and actions visited by an agent using a tree called a **backup diagram**

- States are denoted by triangular nodes
- Each edge coming out of state corresponds to a possible action

- Edges point to circular nodes, which represent "decisions" taken by the environment to account for stochasticity



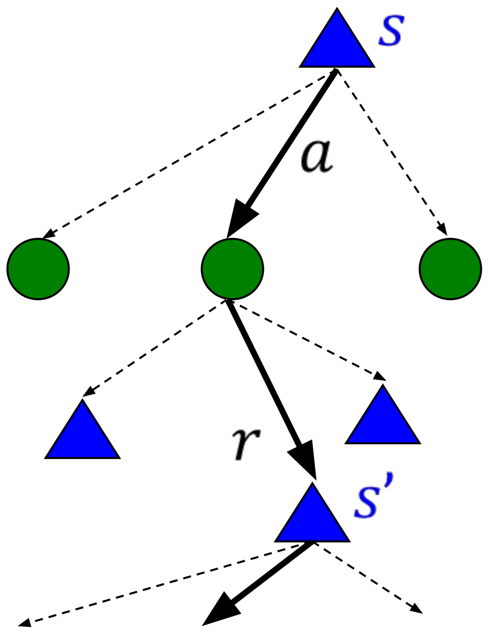
Backup Diagrams

- We can visualize all states and actions visited by an agent using a tree called a **backup diagram**

- States are denoted by triangular nodes
- Each edge coming out of state corresponds to a possible action

- Edges point to circular nodes, which represent "decisions" taken by the environment to account for stochasticity

- Upon action resolution, the agent proceeds to a new successor state and receives the associated reward



Recursive Definition

- Recall the definition of a value function V^{π}:

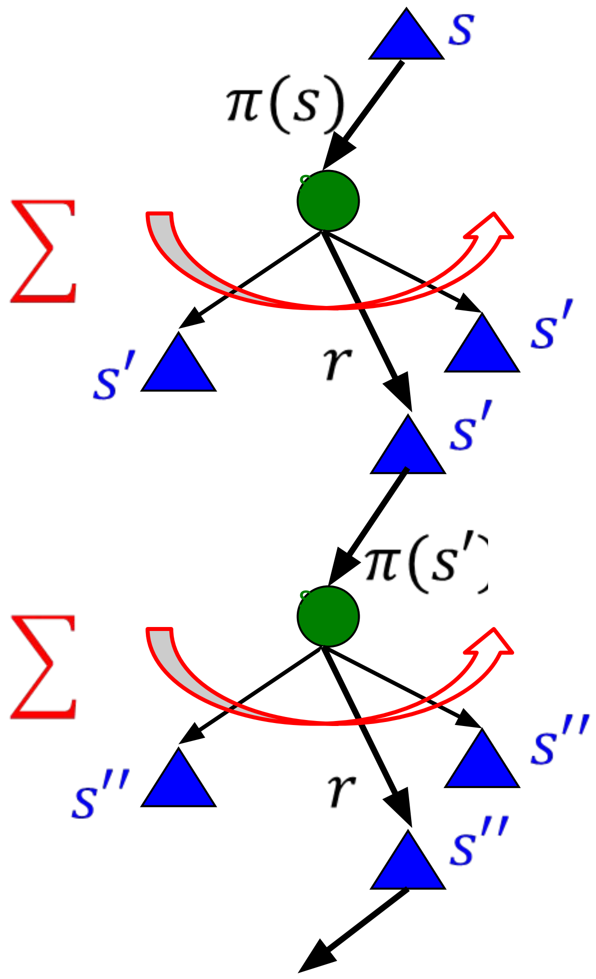
$$V^{\pi}(s) = E\left[\sum_{t=0} \gamma^{t} R(s_t, \pi(s_t), s_{t+1})\right]$$

Recursive Definition

- Recall the definition of a value function V^π:

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

- We have a fixed action at each state due to π
- The *expected* value entails summing over values of successors



Recursive Definition

- Recall the definition of a value function V^π:

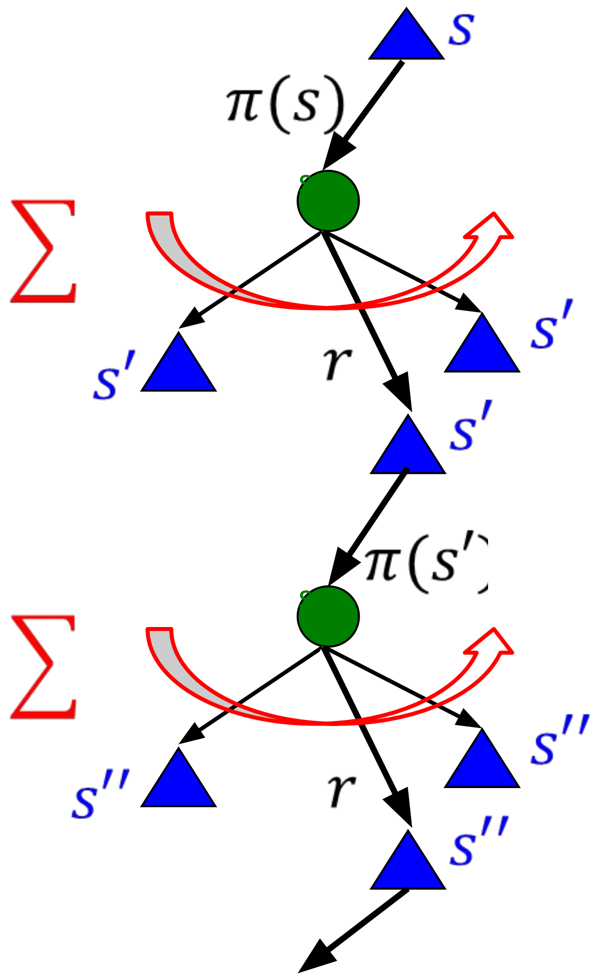
$$V^\pi(s) = E\left[\sum_{t=0} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right]$$

- We have a fixed action at each state due to π
- The *expected* value entails summing over values of successors

- We can write a *recursive* definition of the value function:

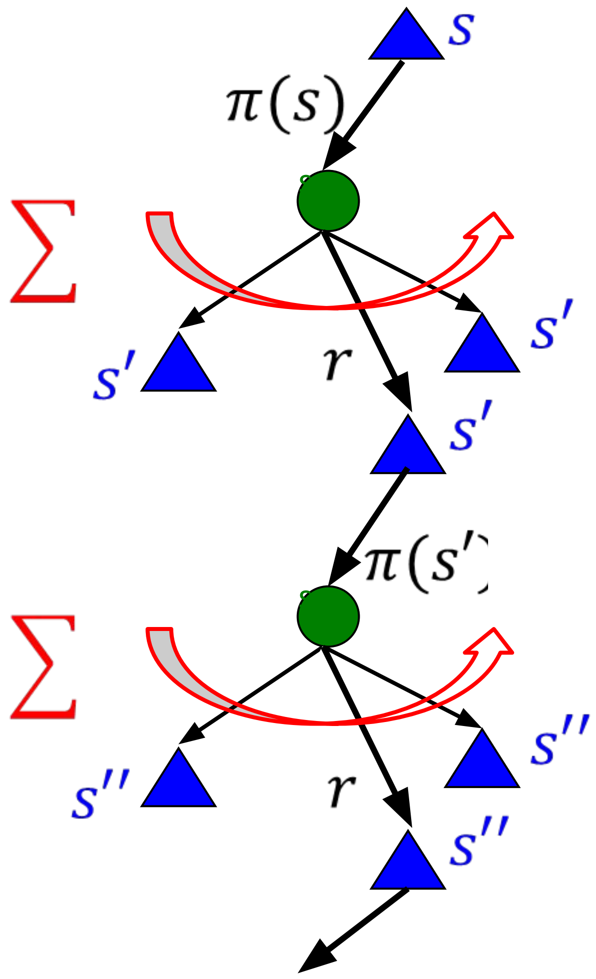
$$V^\pi(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^\pi(s')]$$

- Def of expected value: sum(probability × individual value)



Checkpoint: Recursive Value Function

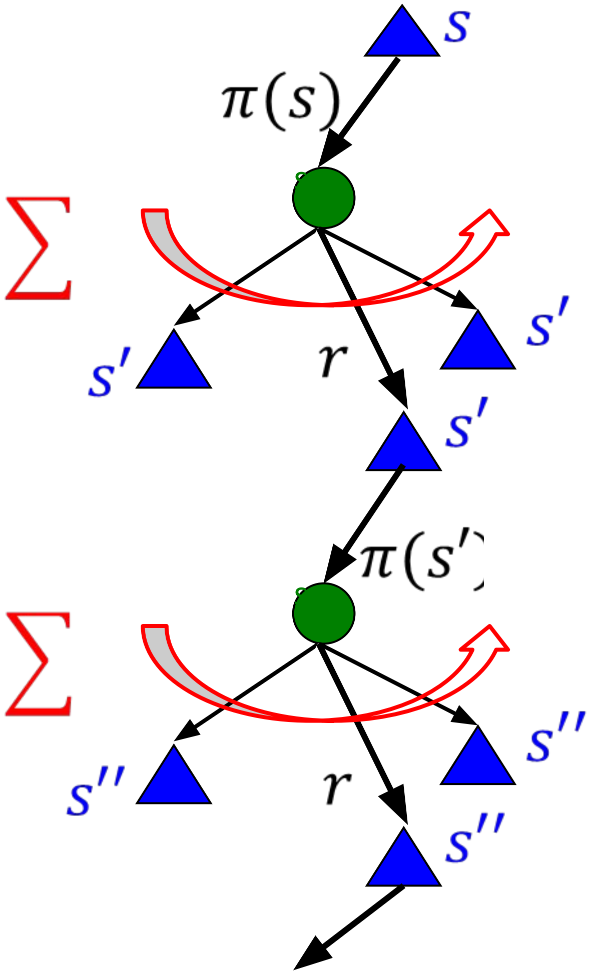
- $$V^\pi(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^\pi(s')]$$
- How does the equation above simplify given the following?
- Only one successor state s' to state s:
- Discount factor is $\gamma = 0$:



Policy Evaluation

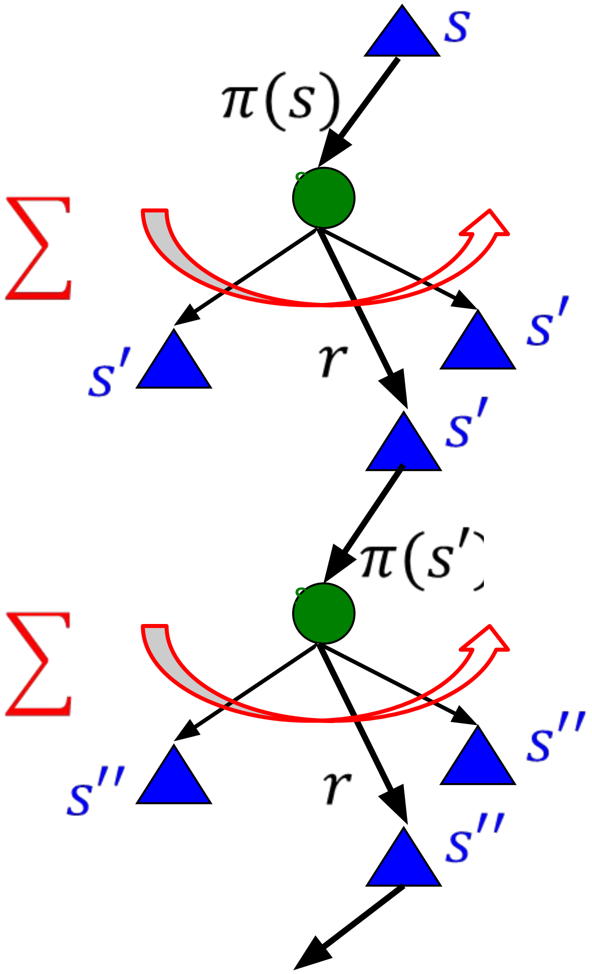
$$V^{\pi}(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^{\pi}(s')]$$

- This equation gives a way of solving for the value function given a fixed policy



Policy Evaluation

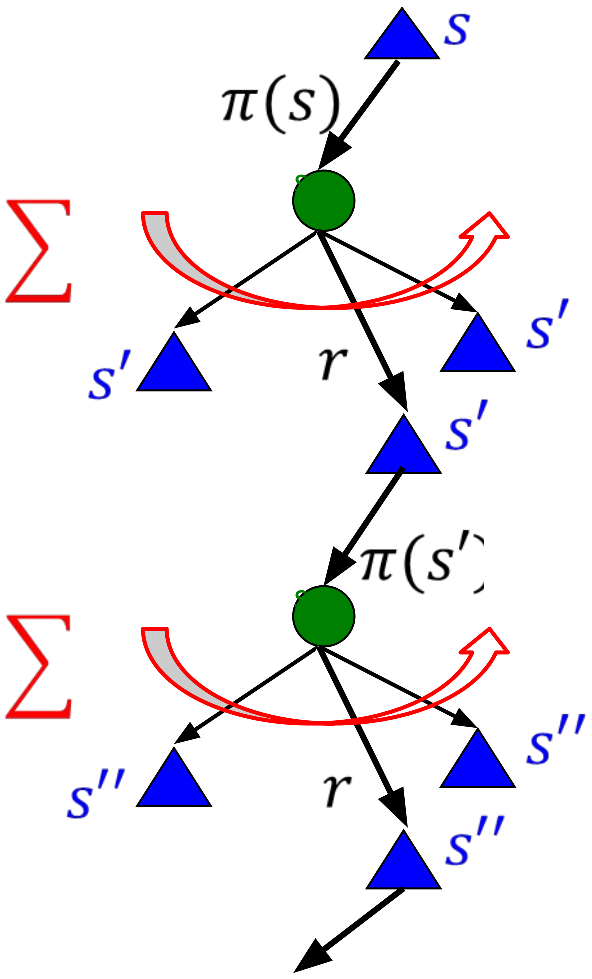
$$V^{\pi}(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^{\pi}(s')]$$



- This equation gives a way of solving for the value function given a fixed policy
- Assume that we know all transition probabilities, rewards received, as well as discount factor

Policy Evaluation

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



- This equation gives a way of solving for the value function given a fixed policy
- Assume that we know all transition probabilities, rewards received, as well as discount factor
- Result: A set of $|S|$ linear equations in the $|S|$ unknowns $V^{\pi}(s)$
- Linear solvers can solve them in $O(|S|^3)$ time

Example: Mini-Gridworld

- Consider a mini-gridworld with states A, B, C
- No terminal states!

- From each state, we can take action L or R

+3	-2	+1
A	B	C

Example: Mini-Gridworld

- Consider a mini-gridworld with states A, B, C
- No terminal states!

A	B	C
+3	-2	+1

- From each state, we can take action L or R

- Reward function: $R(s, a, A) = 3, R(s, a, B) = -2, R(s, a, C) = 1$

Example: Mini-Gridworld

- Consider a mini-gridworld with states A, B, C
- No terminal states!

+3	-2	+1
A	B	C

- From each state, we can take action L or R
- Reward function: $R(s, a, A) = 3, R(s, a, B) = -2, R(s, a, C) = 1$
- Transition function: $\Pr(\text{intended direction}) = 0.8$, $\Pr(\text{opposite direction}) = 0.2$; $s' = s$ if outside grid boundaries

Example: Mini-Gridworld

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

+3	-2	+1
A	B	C

- Suppose we are given the policy $\pi(s) = L \ \forall s$
- Suppose we use the discount factor $\gamma = 0.5$

Example: Mini-Gridworld

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

+3	-2	+1
A	B	C

- Suppose we are given the policy $\pi(s) = L\ \forall s$
- Suppose we use the discount factor $\gamma = 0.5$

- We can form a system of three equations, one for each state:

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

Example: Mini-Gridworld

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

+3	-2	+1
A	B	C

- Suppose we are given the policy $\pi(s) = L \ \forall s$
- Suppose we use the discount factor $\gamma = 0.5$

- We can form a system of three equations, one for each state:

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

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

Example: Mini-Gridworld

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

+3	-2	+1
A	B	C

- Suppose we are given the policy $\pi(s) = L \; \forall s$
- Suppose we use the discount factor $\gamma = 0.5$

- We can form a system of three equations, one for each state:

$$V^\pi(A) = 0.8\big(3 + 0.5V^\pi(A)\big) + 0.2(-2 + 0.5V^\pi(B))$$
$$V^\pi(B) = 0.8\big(3 + 0.5V^\pi(A)\big) + 0.2(1 + 0.5V^\pi(C))$$
$$V^\pi(C) = 0.8\big(-2 + 0.5V^\pi(B)\big) + 0.2(1 + 0.5V^\pi(C))$$

$$\Rightarrow \quad V^\pi = \begin{pmatrix} 4.04 \\ 4.25 \\ .333 \end{pmatrix}$$

Example: Mini-Gridworld

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

+3	-2	+1
A	B	C

- Now suppose we have a different policy $\pi' = R\ \forall s$

- What does the system of linear equations look like?

Example: Mini-Gridworld

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

+3	-2	+1
A	B	C

- Now suppose we have a different policy $\pi' = R\ \forall s$

- What does the system of linear equations look like?

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

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

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

Bellman Optimality Equations

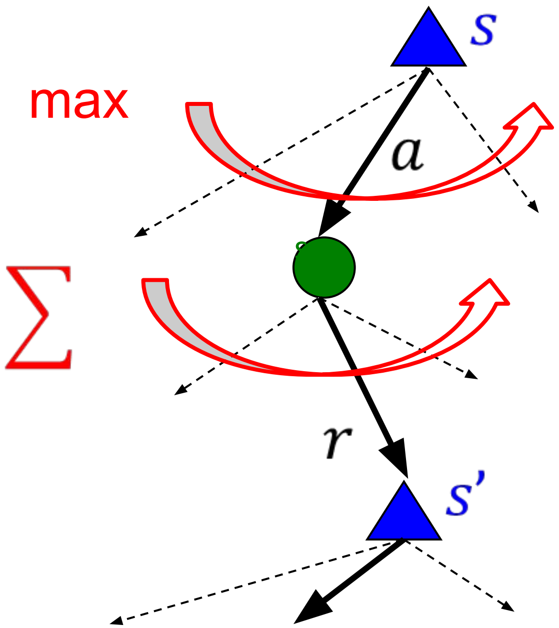
- Generally, we want to find an **optimal policy** or **optimal value function**

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

Bellman Optimality Equations

- Generally, we want to find an **optimal policy** or **optimal value function**

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



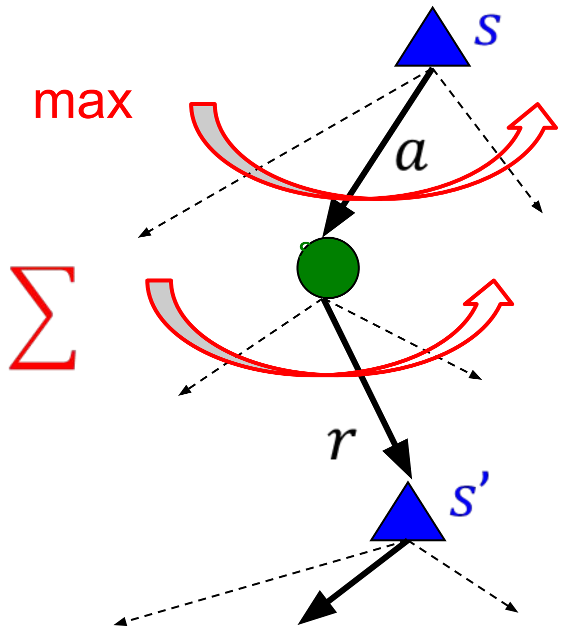
Bellman Optimality Equations

- Generally, we want to find an **optimal policy** or **optimal value function**

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

$$\Downarrow$$

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



Bellman Optimality Equations

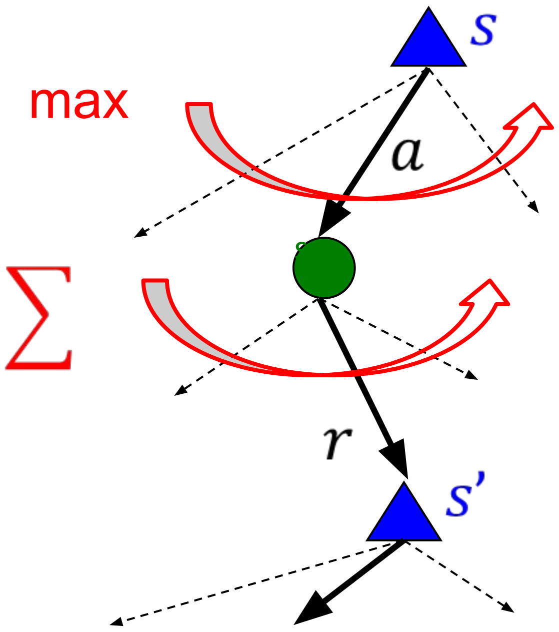
- Generally, we want to find an **optimal policy** or **optimal value function**

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

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

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

Bellman optimality equations



Bellman Optimality Equations

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

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

- The Bellman optimality equations are *nonlinear*
- We cannot solve a system of linear equations to find an optimal policy

Bellman Optimality Equations

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

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

- The Bellman optimality equations are *nonlinear*
- We cannot solve a system of linear equations to find an optimal policy

- Assuming we *can* solve for V^*, it is feasible to find π^* using a brute force search over all actions at each state and taking the argmax

Summary

- Sequential decision problems can be modeled as MDPs
  - Key components: States, actions, transitions, rewards
  - Derived concepts: Utilities, policies, value functions

- Discounting can apply diminishing weights to future rewards and allow utilities of infinite sequences to converge

- Policies and value functions describe what an agent can do
- The Bellman optimality equations are recursive and nonlinear