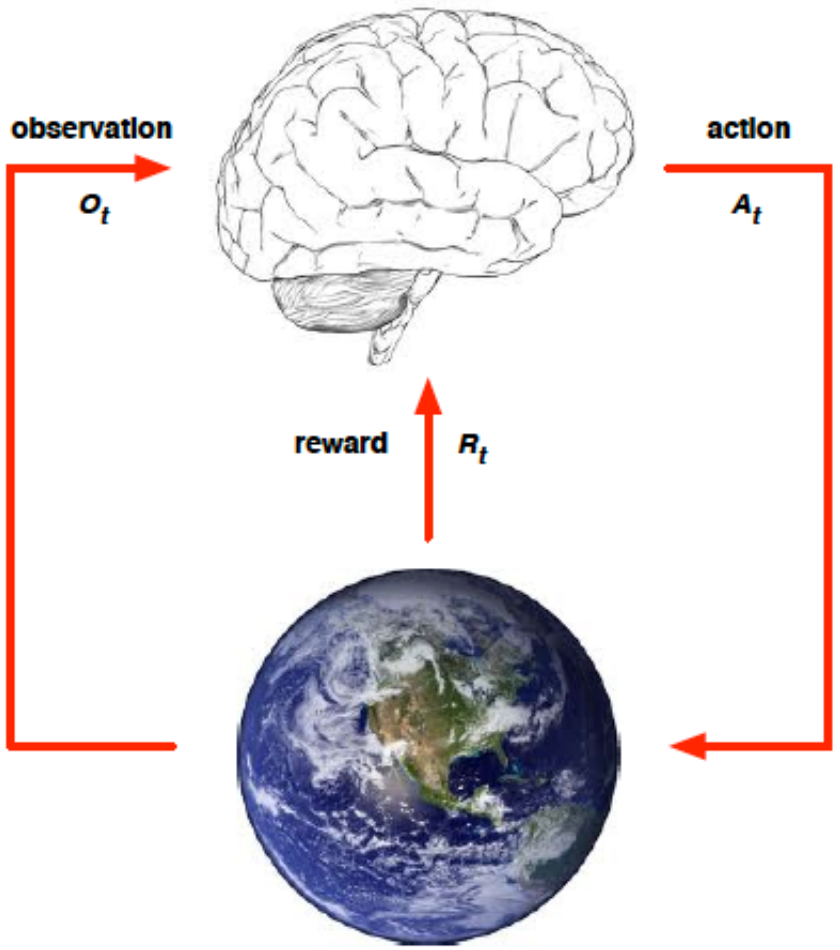
Finite Markov Decision Processes

CMPS 4660/6660: Reinforcement Learning

Acknowledgement: slides adapted from David Silver's RL course and Stanford CS234

Agent and Environment



Goals and Rewards

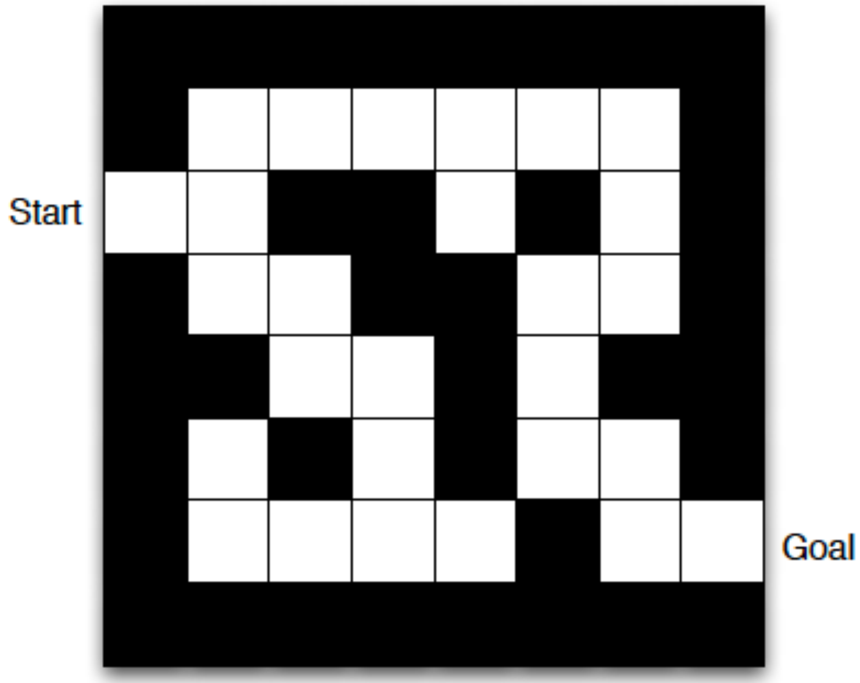
- A reward R_t is a scalar feedback signal
  - Indicates how well agent is doing at step t
- The agent's job is to maximize cumulative reward

Reward Hypothesis: *All* goals can be described by the maximization of expected cumulative reward

- Do you agree with the statement?

Maze Example

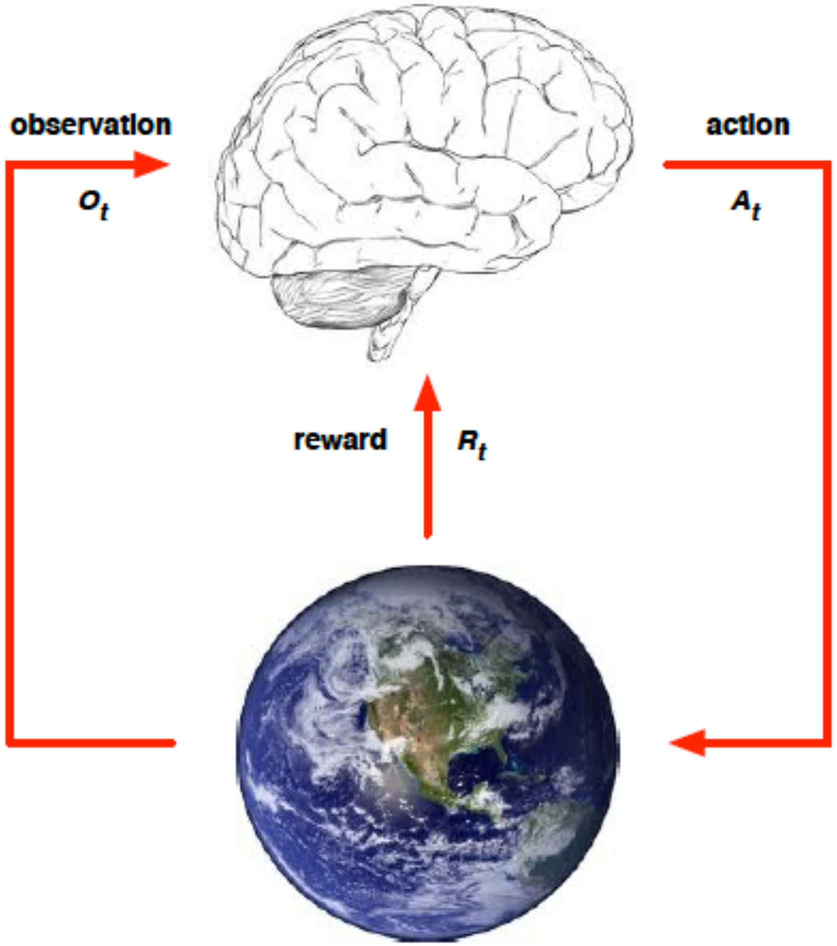
- Goal: escaping from the maze as soon as possible
- Assume R_t = 1 for escaping and R_t = 0 otherwise
  - Does this work?
  - What is a better way to assign reward?



Chess Example

- Goal: win the game
- What is a good assignment of rewards?
  - +1(win), -1(lose), 0(draw)
  - reward only for actually winning, not for achieving subgoals, e.g., taking opponent's pieces or gaining control of the center of board
  - *Delayed* reward
- reward signal is your way of communicating to the robot what you want it to achieve, not how you want it achieved

Agent and Environment

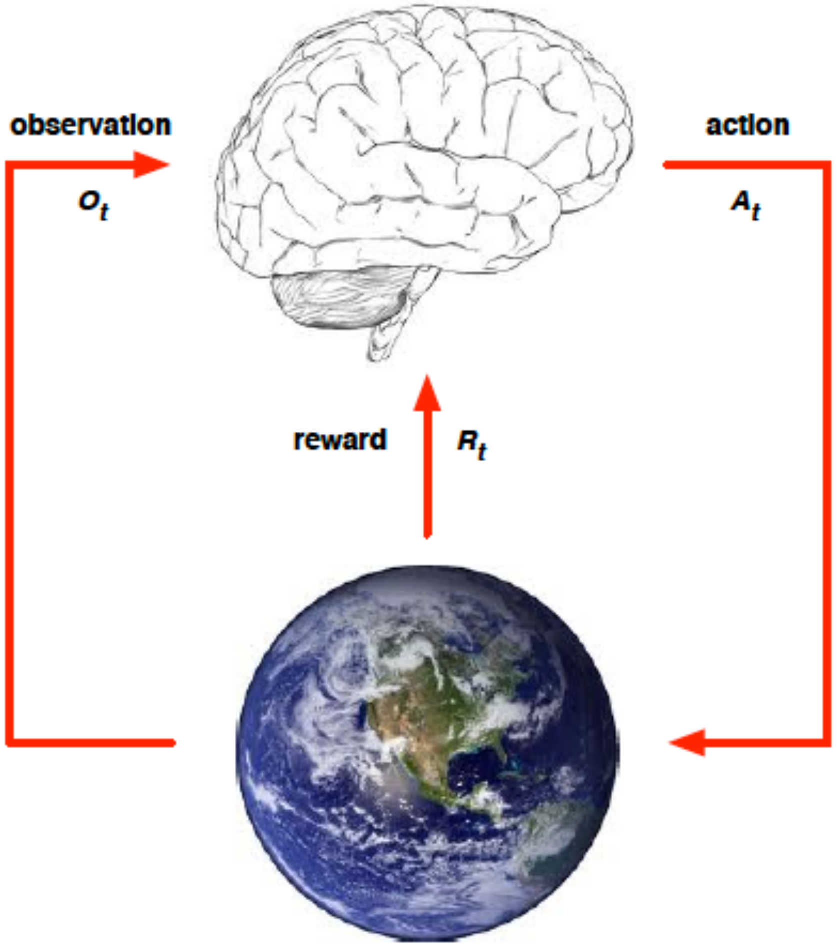


- At each step t the agent:
  - Executes action A_t
  - Receives observation O_{t+1}
  - Receives scalar reward R_{t+1}
- t increments at env. step

History and State

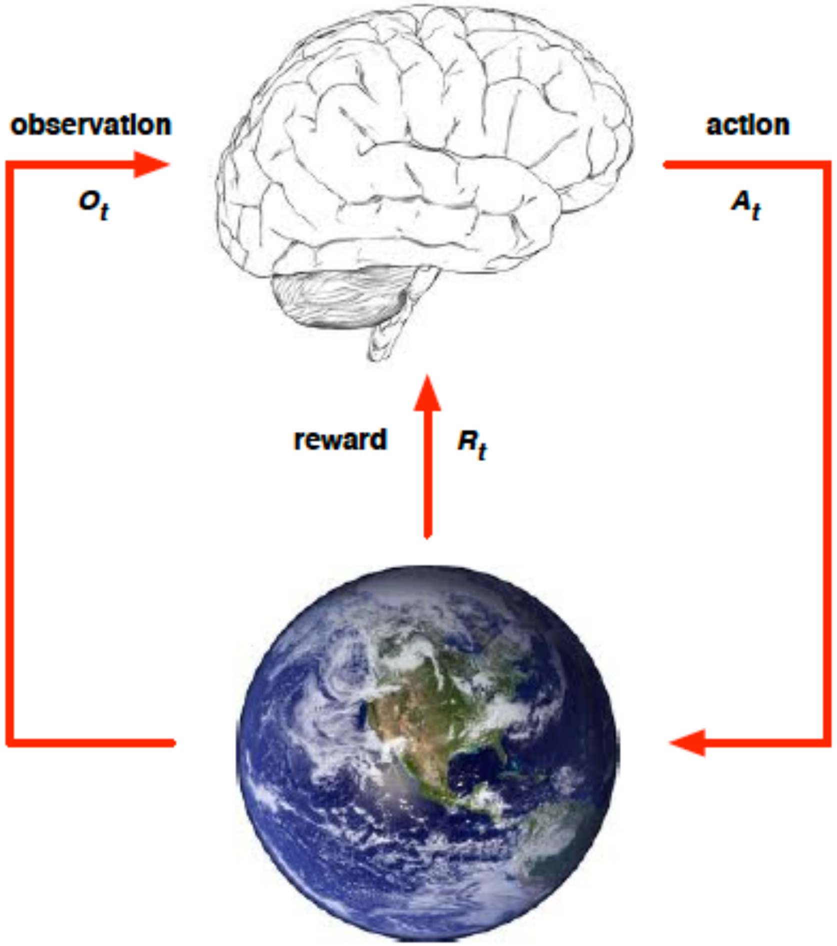
- The history is the sequence of observations, actions, rewards
  - $H_t = O_0, A_0, R_1, O_1, A_1, R_2, \dots, R_t, O_t$
  - i.e. all observable variables up to time t
  - e.g. the sensorimotor stream of a robot or embodied agent
- What happens next depends on the history:
  - The agent selects actions
  - The environment selects observations/rewards
- State is the information used to determine what happens next
- Formally, state is a function of the history: $S_t = f(H_t)$

Environment State



- The environment state S_t^e is the environment's private representation
- i.e. whatever data the environment uses to pick the next observation/reward
- The environment state is not usually visible to the agent
- Even if S_t^e is visible, it may contain irrelevant information

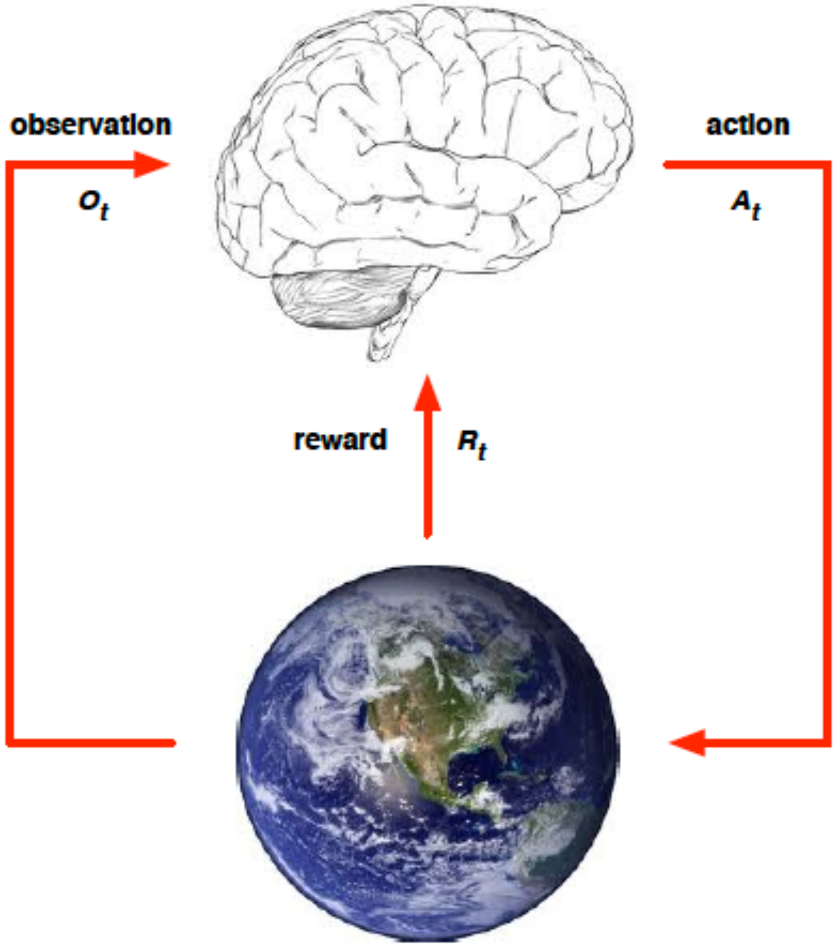
Agent State



- The agent state S_t^a is the agent's internal representation
- i.e. whatever information the agent uses to pick the next action
- i.e. it is the information used by reinforcement learning algorithms
- It can be any function of history:

$$S_t^a = f(H_t)$$

Fully Observable Environments



- Full observability: agent directly observes environment state

$$O_t = S_t^e$$

- In this case, we may simply set

$$S_t^a = O_t$$

  - This is not always the best choice though
  - We also want the states to be “Markov” (to be defined shortly)

Partially Observable Environments

- Partial observability: agent indirectly observes environment:
    - A robot with camera vision isn't told its absolute location
    - A trading agent only observes current prices
    - A poker playing agent only observes public cards

- Agent must construct its own state representation S_t^a , e.g.
    - Complete history: $S_t^a = H_t$
    - Beliefs of environment state: $S_t^a = (\Pr[S_t^e = s^1], \ldots, \Pr[S_t^e = s^n])$
    - Recurrent neural network: $S_t^a = \sigma(S_{t-1}^a W_s + O_t W_o)$

Episodic and Continuing Tasks

- Time Horizon (T) = the number of time steps in each episode

- Episodic tasks: agent-environment interaction naturally breaks into episodes
  - E.g., plays of games
  - Each episode ends in a special terminal state
  - T is typically random (varies from episode to episode)

- Continuing tasks: agent–environment interaction does not break naturally into identifiable episodes, but goes on continually without limit
  - E.g., a robot with a long life span
  - Infinite horizon

Markov Decision Processes

- Markov chains
- Markov reward processes
- Markov decision processes
- Value functions
- Bellman equations

Introduction to Markov Decision Processes

- Markov decision processes formally describe an environment for reinforcement learning
- Where the environment is fully observable
  - i.e. The current state completely characterizes the process

- Almost all RL problems can be formalized as MDPs, e.g.,
  - Optimal control primarily deals with continuous MDPs
  - Partially observable problems can be converted into MDPs
  - Bandits are MDPs with one state

Markov Property

- “The future is independent of the past given the present”
- A state is S_t is *Markov* if

$$\Pr(S_{t+1} = s' \mid S_t = s, A_t = a, S_{t-1} = s_{t-1}, A_{t-1=} a_{t-1}, \dots, S_0 = s_0)$$

$$= \Pr(S_{t+1} = s' | S_t = s, A_t = a)$$

- The state captures all relevant information from the history
- Once the state is known, the history may be thrown away
- i.e. The state is a sufficient statistic of the future

Andrey Markov
(1856-1922)

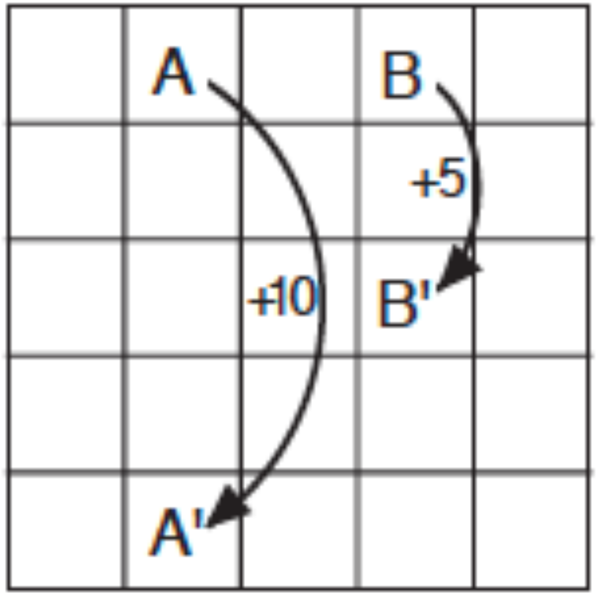
Markov Chains

- A discrete time Markov chain is a memoryless random process, i.e., a sequence of random states $S_0, S_1, \ldots$ with the Markov property.

- A finite *Markov Chain* is a tuple $\langle \mathcal{S}, P \rangle$
  - $\mathcal{S}$ is a (finite) set of states
  - P is a state transition probability matrix, $P_{ss'} = \Pr\{S_t = s' | S_{t-1} = s\}$

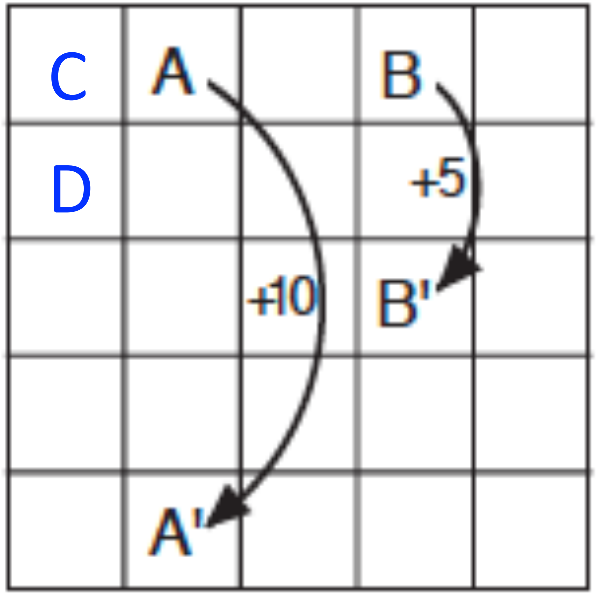
Example: Gridworld

- States: Agent's location
- Actions: N, E, S, W
- Actions that would take the agent off the grid leave its location unchanged
- From state A, all four actions take the agent to A0
- From state B, all actions take the agent to B0



Example: Gridworld

- Consider the equiprobable random policy
- We have a Markov chain $\langle \mathcal{S}, P \rangle$
  - $\mathcal{S}$ is the set of all locations, $|\mathcal{S}| = 25$
  - P can be easily derived, e.g., $P_{CA} = 0.25$, $P_{CD} = 0.25, P_{CC} = 0.5, P_{AA'} = 1$



$$\Pr(S_2 = A' | S_0 = C)$$

$$= \Pr(S_2 = A' | S_1 = A) \cdot \Pr(S_1 = A | S_0 = C)$$

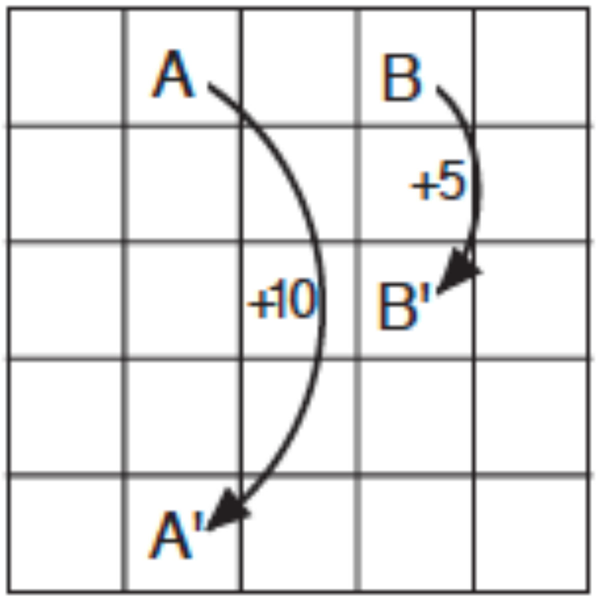
$$= P_{AA'} \cdot P_{CA} = 1 \cdot 0.25 = 0.25$$

Markov Reward Process

- Markov reward process is a Markov chain + rewards
- A finite *Markov Reward Process* is a tuple $\langle \mathcal{S}, P, r, \gamma \rangle$
  - $\mathcal{S}$ is a finite set of states
  - P is a state transition probability matrix, $P_{ss'} = \Pr\{S_t = s' | S_{t-1} = s\}$
  - r is a reward function, $r(s) = \mathbb{E}[R_t | S_{t-1} = s]$
  - γ is a discount factor, $\gamma \in [0,1]$
- Note: no actions

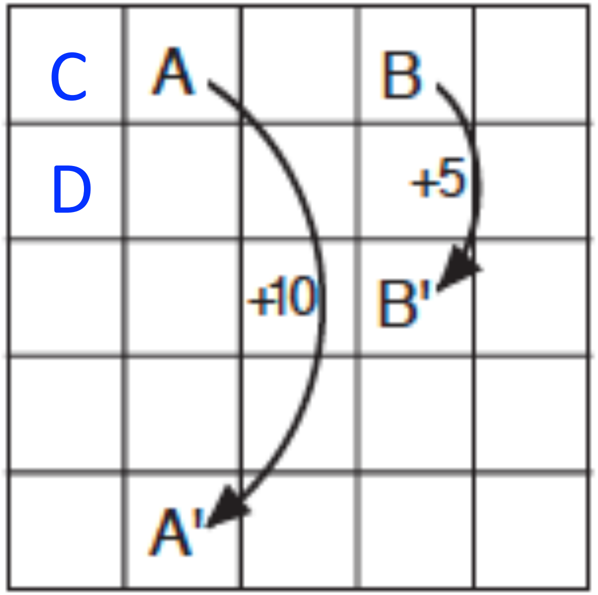
Example: Gridworld

- States: Agent's location
- Actions: N, E, S, W
- Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of −1.
- From state A, all four actions yield a reward of +10 and take the agent to A0.
- From state B, all actions yield a reward of +5 and take the agent to B0
- Other actions result in a reward of 0



Example: Gridworld

- Consider the equiprobable random policy
- We have a Markov reward process $\langle \mathcal{S}, P, r, \gamma \rangle$
  - $\mathcal{S}$ is the set of all locations, $|\mathcal{S}| = 25$
  - P can be easily derived
  - r can be derived for each state, e.g.,
    - $r(C) = (-1) \cdot 0.5 + 0 \cdot 0.5 = -0.5$
    - $r(A) = \ 10$



Return

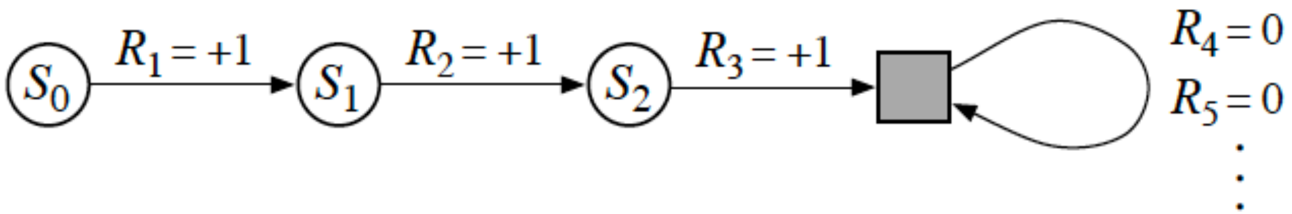
- The *return* G_t is the total discounted reward from time-step *t* to horizon.

- Episodic tasks: $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$
  - Finite horizon: terminate at a fixed time T
  - Termination is inevitable (to be made precise shortly)

- Continuing tasks: $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
  - $\gamma \in [0,1]$
  - If $\gamma < 1$ and $|R_t| \leq R_{max}$ for $\forall t$, then $|G_t| \leq \frac{1}{1-\gamma} R_{max}$
  - $\gamma = 0$: Only care about immediate reward
  - $\gamma = 1$: Future reward is as beneficial as immediate reward
  - Another common approach: average rewards (difficult to analyze)

Unified notation for episodic and continuing tasks

- Introduce a terminal (absorbing) state s^* where $P_{s^*s^*} = 1$ and $R_t = 0$ if $s_{t-1} = s^*$



- Then for both episodic and continuing tasks

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

- $\mathcal{S}^+ = \mathcal{S} \cup \{s^*\}$

Why discount?

- Mathematically convenient (avoid infinite returns and values)
- Uncertainty about the future may not be fully represented
- Model the reality
  - If the reward is financial, immediate rewards may earn more interest than delayed rewards
  - Animal/human behavior shows preference for immediate reward

Stationary Preferences

- Theorem: if we assume stationary preferences:

$$[a_1, a_2, \ldots] \succ [b_1, b_2, \ldots]$$
$$\Updownarrow$$
$$[r, a_1, a_2, \ldots] \succ [r, b_1, b_2, \ldots]$$

- Then: there are only two ways to define utilities
  - Additive utility: $U([r_0, r_1, r_2, \ldots]) = r_0 + r_1 + r_2 + \cdots$
  - Discounted utility: $U([r_0, r_1, r_2, \ldots]) = r_0 + \gamma r_1 + \gamma^2 r_2 \cdots$

Markov Decision Process

- A Markov decision process (MDP) is a Markov reward process with decisions. It is an environment in which all states are Markov.
- A finite *Markov Decision Process* is a tuple $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$
  - $\mathcal{S}$ is a finite set of states
  - $\mathcal{A}(s)$ is a finite set of actions available at state s
  - P is a state transition probability matrix,

    $$P_{ss'}(a) = \Pr\{S_t = s' | S_{t-1} = s, A_{t-1} = a\}$$
  - r is a reward function, $r(s, a) = \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a]$
  - γ is a discount factor, $\gamma \in [0,1]$

Markov Decision Process

$$
\left.\begin{aligned}
P_{ss'}(a) &= \Pr\{S_t = s' | S_{t-1} = s, A_{t-1} = a\} \\
r(s,a) &= \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a]
\end{aligned}\right\}
$$

Let $\mathcal{R}$ be a finite set of rewards

$$p(s', r | s, a) = \Pr\{S_t = s', R_t = r | S_{t-1} = s, A_{t-1} = a\}$$

$$P_{ss'}(a) = ? \qquad \sum_{r \in \mathcal{R}} p(s', r | s, a)$$

$$r(s,a) = ? \qquad \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r | s, a)$$

Policy (1)

- A policy = any rule for choosing actions
  - A policy fully defines the behavior of an agent, and
  - can be history dependent and/or randomized
- A stochastic stationary policy π is a distribution over actions given states,

$$\pi(a|s) = \Pr\{A_t = a|S_t = s\} \in [0,1] \text{ and } \sum_{a\in\mathcal{A}(s)} \pi(a|s) = 1$$

- Stationary policies depend on the current state (not the history)
- Deterministic stationary policy: $A_t = \pi(S_t)$
- A fundamental question: *For a given optimality criterion, under what conditions is it optimal to use a deterministic stationary policy?*

Policy (2)

- Given an MDP $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$ and a stationary policy π
- The state and reward sequence S_0, R_1, S_1, R_2 ... is a Markov reward process $\langle \mathcal{S}, P^\pi, r^\pi, \gamma \rangle$, where

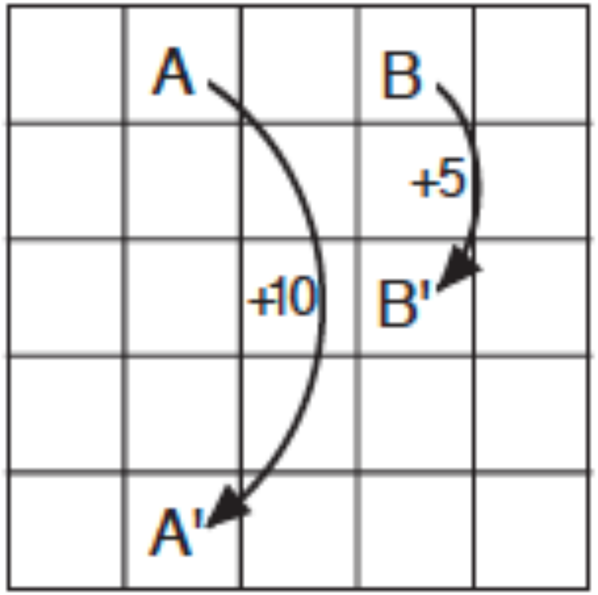
$$P^\pi_{s,s'} = \sum_{a \in \mathcal{A}(s)} \pi(a|s) \, P_{ss'}(a)$$

$$r^\pi(s) = \sum_{a \in \mathcal{A}(s)} \pi(a|s) r\,(s, a)$$

- The state sequence S_0, S_1, ... is a Markov chain $\langle \mathcal{S}, P^\pi \rangle$

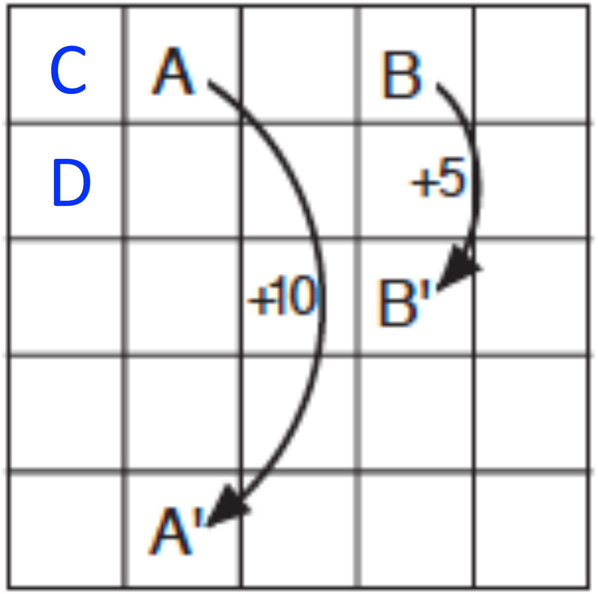
Example: Gridworld

- States: Agent's location
- Actions: N, E, S, W
- Actions that would take the agent off the grid leave its location unchanged, but also result in a reward of −1.
- From state A, all four actions yield a reward of +10 and take the agent to A0.
- From state B, all actions yield a reward of +5 and take the agent to B0
- Other actions result in a reward of 0



Example: Gridworld

- A Markov decision process $\langle \mathcal{S}, \mathcal{A}, P, r, \gamma \rangle$
  - $\mathcal{S}$ is the set of all locations, $|\mathcal{S}| = 25$
  - $\mathcal{A}(s) = \{\text{N, E, S, W}\}$
  - P and r can be easily derived from the game rule
    - $P_{CC}(N) = 1, P_{CA}(E) = 1, P_{CD}(S) = 1,\ P_{CC}(W) = 1$
    - $r(C, N) = r(C, W) = -1, r(C, E) = r(C, S) = 0$
  - A stationary policy π picks an action depending on the current location
  - E.g., equiprobable random policy



Return

- The *return* G_t is the total discounted reward from time-step *t* to horizon.
- Episodic tasks: $G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \cdots + R_T$
  - Finite horizon: terminate at a fixed time T
  - Termination is inevitable (to be made precise shortly)
- Continuing tasks: $G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$
  - $\gamma \in [0,1]$
  - If $\gamma < 1$ and $|R_t| \leq R_{max}$ for $\forall t$, then $|G_t| \leq \frac{1}{1-\gamma} R_{max}$
  - $\gamma = 0$: Only care about immediate reward
  - $\gamma = 1$: Future reward is as beneficial as immediate reward
  - Another common approach: average rewards (difficult to analyze)

State-Value Function

- The *state-value* function $v_\pi(s)$ of an MDP is the expected return starting from state s, then following policy π

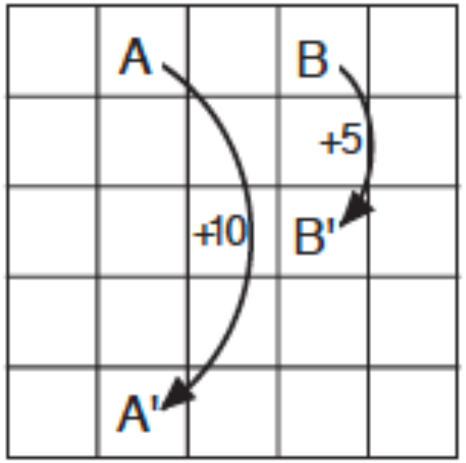
$$v_\pi(s) = \mathbb{E}_\pi(\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s)$$

$$= \mathbb{E}_\pi\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s\right) \quad \text{(if } \pi \text{ is stationary)}$$

$$= \mathbb{E}_\pi(G_t | S_t = s) \quad \text{(if } \pi \text{ is stationary)}$$

Example: Gridworld

- Consider an equiprobable random policy
- $\gamma = 0.8$



Actions

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

state-value function

- Why $v(A) < 10$ and $v(B) > 5$?
- How to find the state-value function?
- Can we do better?

Action-Value Function

- The *action-value* function $q_\pi(s,a)$ is the expected return starting from state s, taking action a, then following policy π

$$q_\pi(s,a) = \mathbb{E}_\pi(\textstyle\sum_{t=0}^{\infty} \gamma^t R_{t+1} | S_0 = s, A_0 = a)$$

$$= \mathbb{E}_\pi\left(\textstyle\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a\right) \quad \text{(if } \pi \text{ is stationary)}$$

$$= \mathbb{E}_\pi(G_t | S_t = s, A_t = a) \quad \text{(if } \pi \text{ is stationary)}$$

Value Functions

$$v_\pi(s) = \sum_a \pi(a|s)\, q_\pi(s,a)$$

$$q_\pi(s,a) = r(s,a) + \gamma \sum_{s'} P_{ss'}(a)\, v_\pi(s')$$

Example: Inventory Management

- An inventory of capacity M
- A_t: the number of items ordered in the evening of day t for day $t+1$
- D_t: the demand on day t (independent and identically distributed with a known distribution)
- Payoff on day t
  - purchasing cost: $K\mathbb{I}_{\{A_t>0\}} + cA_t$
  - holding cost: h per item
  - selling price: p per item sold

Example: Inventory Management

- State X_t: the size of the inventory in the evening of day t
- Action A_t: the number of items ordered in the evening of day t
- R_t: reward on day t
- Transition probabilities and reward function can be derived from:

$$X_{t+1} = (\min(X_t + A_t, M) - D_{t+1})^+$$

$$\begin{aligned} R_{t+1} &= -K\mathbb{I}_{\{A_t>0\}} - c\,(\min(X_t + A_t, M) - X_t) \\ &\quad -hX_t + p\,(\min(X_t + A_t, M) - X_{t+1})^+ \end{aligned}$$

MDP with a terminal state

Question: For episodic tasks, when are the return/value functions well defined?

Assumption (termination is Inevitable Under All Policies)

There exists an integer m such that regardless of the policy used and the initial state, there is a positive probability that the terminal state will be reached after no more than m stages; i.e., for all admissible policies π we have

$$\rho_\pi = \max_{s \in S} \Pr_\pi\{S_m \neq s^* | S_0 = s\} < 1$$

E.g., this holds when at the end of each time step, there is a positive probability that the process ends ($m = 1$).

MDP with a terminal state

Lemma

Let $\rho = \max_{\pi} \rho_{\pi}$, the maximum probability of not researching s^* within m stages over all starting states and policies (not necessarily stationary). We have

(1) ρ<1;

(2) $\Pr_{\pi}\{S_{km} \neq s^* | S_0 = s\} \leq \rho^k$ for any π.

Proof sketch: since ρ_π depends only on the first m components of π and there are finite number of actions, there can be only finite number of distinct ρ_π.

(see DB p. 180 for a complete proof)

MDP with a terminal state

Lemma

The total reward in the m stages between km and $(k+1)m-1$ is bounded by $\rho^k B$ where $B = m \max_{s \in \mathcal{S}, a \in \mathcal{A}(s)} |r(s,a)|$

Theorem

$$|v_\pi(s)| \leq \frac{1}{1-\rho} B, \qquad \forall s, \pi$$

(see DB p. 212 for proofs)

Major Components of an RL Agent

- Policy: agent's behavior function
- Value function: how good is each state and/or action
- Model: agent's representation of the environment

Policy

- A policy is the agent's behavior
- It is a map from state to action, e.g.
  - Deterministic (stationary) policy: $a = \pi(s)$
  - Stochastic (stationary) policy: $\pi(a|s) = \Pr\{A_t = a | S_t = s\}$

Valuation

- Value function is a prediction of future reward
- Used to evaluate the goodness/badness of states
- And therefore to select between actions, e.g.

$$v_\pi(s) = \mathbb{E}_\pi(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots | S_t = s)$$

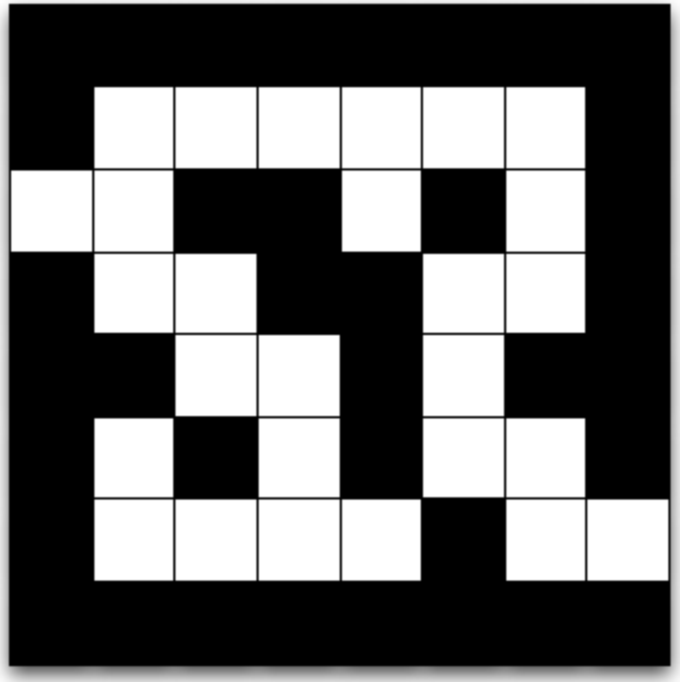
Model

- A model predicts what the environment will do next
- P predicts the next state
- r predicts the next (immediate) reward, e.g.

$$P_{ss'}(a) = \Pr\{S_t = s' | S_{t-1} = s, A_{t-1} = a\}$$

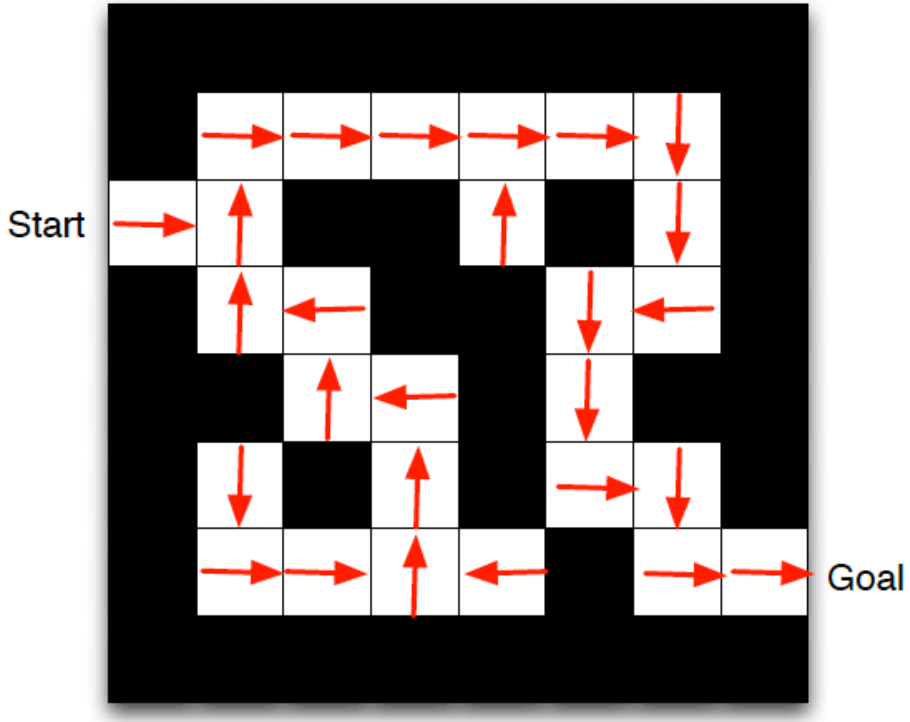
$$r(s, a) = \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a]$$

Maze Example: Model



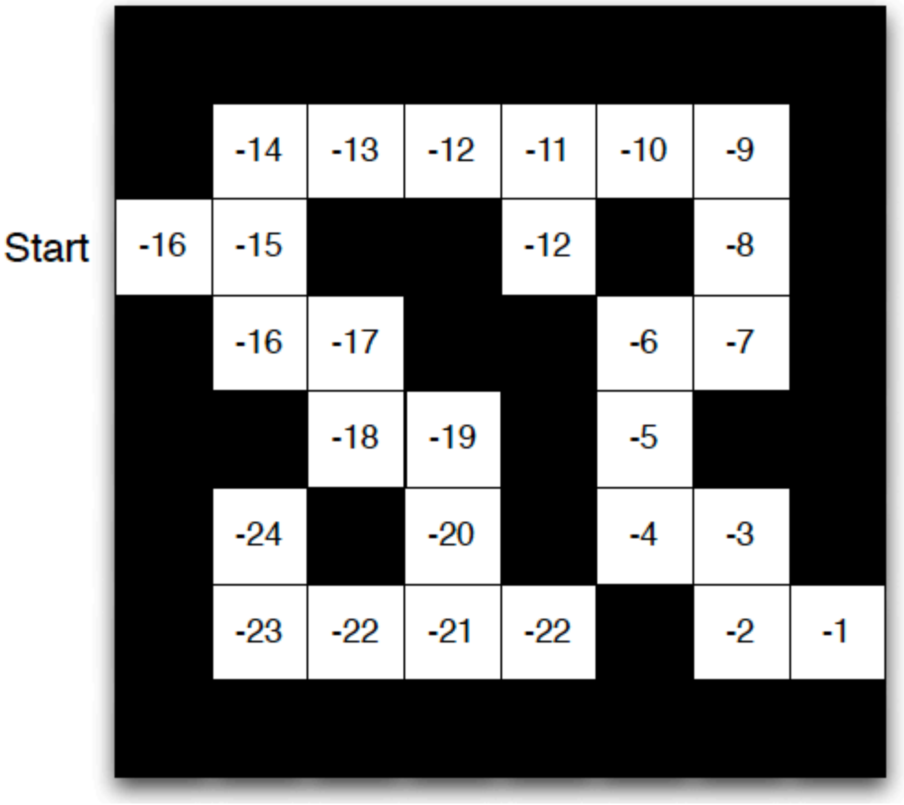
- Goal: escaping from the maze as soon as possible
- Rewards: -1 per time-step
- Actions: N, E, S, W
- States: Agent's location

Maze Example: Policy



- Arrows represent policy $\pi(s)$ for each state s

Maze Example: Value Function

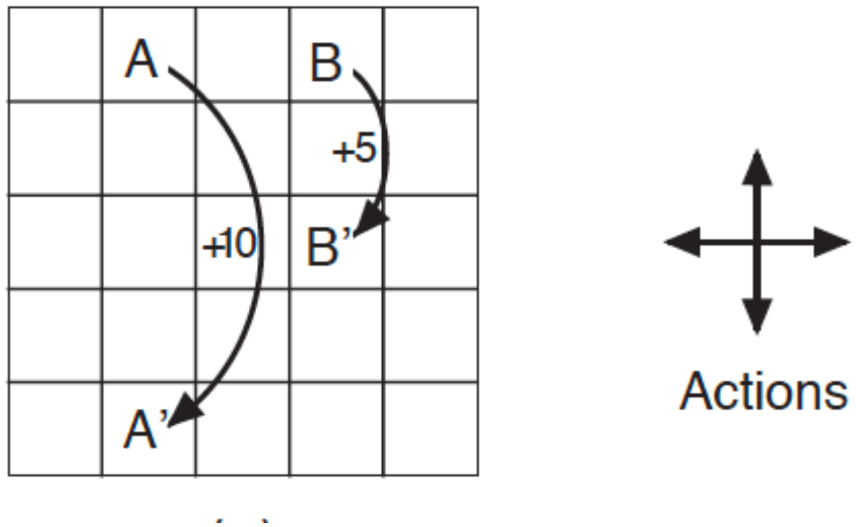


- Numbers represent value $v(s)$ of each state s

Prediction and Control

- Prediction: evaluate the future
  - Given a policy
- Control: optimize the future
  - Find the best policy

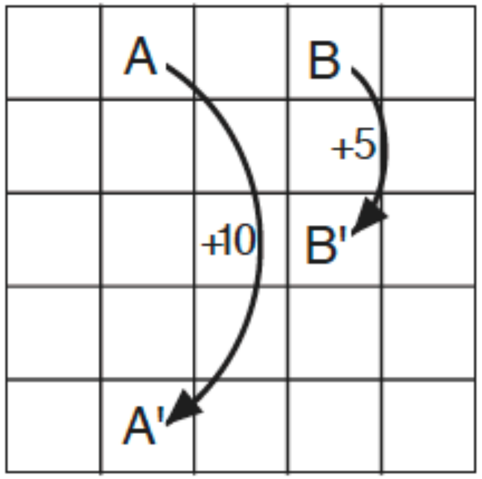
Gridworld Example: Prediction



3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

What is the value function for the uniform random policy?

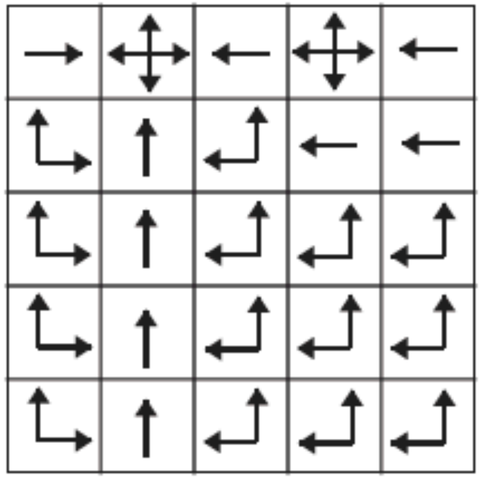
Gridworld Example: Control



Gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

v_*



π_*

- What is the optimal value function over all possible policies?
- What is the optimal policy?

Prediction and Control

- Prediction: evaluate the future
  - Given a policy
  - Bellman equations
- Control: optimize the future
  - Find the best policy
  - Bellman optimality equations

Bellman Equation for MDP

Consider a stationary policy π

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi(G_t | S_t = s) \\
&= \mathbb{E}_\pi(R_{t+1} + \gamma G_{t+1} | S_t = s)
\end{aligned}
$$

$$
\begin{aligned}
G_t &\doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \\
&= R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \cdots) \\
&= R_{t+1} + \gamma G_{t+1}
\end{aligned}
$$

Bellman Equation for MDP

Consider a stationary policy π

$$
\begin{aligned}
v_\pi(s) &\doteq \mathbb{E}_\pi(G_t | S_t = s) \\
&= \mathbb{E}_\pi(R_{t+1} + \gamma G_{t+1} | S_t = s) \\
&= \sum_a \pi(a|s) \left(r(s,a) + \gamma \sum_{s'} P_{ss'}(a) \, \mathbb{E}(G_{t+1} | S_{t+1} = s') \right) \\
&\boxed{= \sum_a \pi(a|s) \left(r(s,a) + \gamma \sum_{s'} P_{ss'}(a) \, v_\pi(s') \right)}
\end{aligned}
$$

Bellman Equation for MDP

$$v_\pi(s) = \sum_a \pi(a|s)\, r(s,a) + \gamma \sum_{s'} \sum_a \pi(a|s) P_{ss'}(a)\, v_\pi(s')$$

$$= \underbrace{r^\pi(s)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s'} P^\pi_{ss'} v_\pi(s')}_{\text{Discounted sum of future rewards}}$$

Richard Bellman
(1856-1922)

Bellman Equation in Matrix Form

- For finite state MDP, we can express $v_\pi(s)$ using a matrix equation

$$\begin{bmatrix} v(s_1) \\ \vdots \\ v(s_N) \end{bmatrix} = \begin{bmatrix} r(s_1) \\ \vdots \\ r(s_N) \end{bmatrix} + \gamma \begin{bmatrix} P_{s_1 s_1} & \cdots & P_{s_1 s_N} \\ \vdots & \ddots & \vdots \\ P_{s_N s_1} & \cdots & P_{s_N s_N} \end{bmatrix} \begin{bmatrix} v(s_1) \\ \vdots \\ v(s_N) \end{bmatrix}$$

$$v_\pi = r^\pi + \gamma P^\pi v_\pi$$

Analytic Solution for Value of MDP

- Bellman equation: $v_\pi = r^\pi + \gamma P^\pi v_\pi$

Theorem
The Bellman equation has a unique solution (to be proved)

- Analytic solution

$$v_\pi = r^\pi + \gamma P^\pi v_\pi$$
$$v_\pi - \gamma P^\pi v_\pi = r^\pi$$
$$(I - \gamma P^\pi) v_\pi = r^\pi$$
$$v_\pi = (I - \gamma P^\pi)^{-1} r^\pi$$

Solving directly requires taking a matrix inverse $\sim O(|\mathcal{S}|^3)$

Direct solution only possible for small MDPs

Iterative methods for large MDPs, e.g.

- Dynamic programming
- Monte-Carlo evaluation
- Temporal-Difference learning

Bellman Equation for MDP

- The action-value function can similarly be decomposed,

$$
\begin{aligned}
q_\pi(s,a) &= r(s,a) + \gamma \sum_{s'} P_{ss'}(a)\, v_\pi(s') \\
&= r(s,a) + \gamma \sum_{s'} P_{ss'}(a) \sum_{a'} \pi(a'|s')\, q_\pi(s',a')
\end{aligned}
$$

Optimal Policy and Optimal Value Functions

- We say that $\pi \geq \pi'$ ("π is better than π'") if $v_\pi \geq v_{\pi'}$ i.e., $v_\pi(s) \geq v_{\pi'}(s), \forall s$
- There is a policy π_* that is better than any other policy (including non-stationary ones), which is an optimal policy (to be proved)
- All optimal policies share the same value functions

$$v_*(s) = \sup_\pi v_\pi(s)\,, \forall s$$

$$q_*(s,a) = \sup_\pi q_\pi(s,a)\,, \forall s, a$$

Bellman Optimality Equation

Theorem

The optimal value function $v_*(s)$ satisfies the following equation

$$v_*(s) = \max_{a \in \mathcal{A}(s)} \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right], \forall s \in \mathcal{S}$$

Remark 1: We will show that v_* is the unique solution to the optimality equation.

Remark 2: if v_* is known, any policy that is greedy with respect to v_* is optimal. In particular, there is a deterministic stationary policy that is optimal.

Bellman Optimality Equation

- A Markov policy is a sequence of mappings $\pi = (\mu_0, \mu_1, \ldots)$, one for each time step, where each μ_t is a (randomized) mapping from state to action.

Lemma

Given any history-dependent policy and starting state, there exists a Markov policy with the same value.

(see Theorem 5.5.1 in "Markov Decision Processes" by Puterman)

Proof of Bellman Optimality Equation

Step 1: $v_*(s) \leq \max_a \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right]$

Let $\pi = (\mu_0, \mu_1, \mu_2, \ldots)$ be an arbitrary Markov policy and $\pi' = (\mu_1, \mu_2, \ldots)$. Then

$$v_\pi(s) = \sum_a \mu_0(a|s) \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_{\pi'}(s') \right].$$

Since $v_{\pi'}(s') \leq v_*(s')$ for all s', we have

$$\begin{aligned} v_\pi(s) &\leq \sum_a \mu_0(a|s) \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right] \\ &\leq \sum_a \mu_0(a|s) \max_a \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right] \\ &= \max_a \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right]. \end{aligned}$$

As this holds for any Markov policy, we have $v_*(s) \leq \max_a \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right]$.

Proof of Bellman Optimality Equation

Step 2: $v_*(s) \geq \max_a \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right]$

Let $a_0 = \operatorname{argmax}_a \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right]$.

Let $\pi_{s'}$ be a policy such that $v_{\pi_{s'}}(s') \geq v_*(s') - \epsilon$.

Let π be the policy that chooses a_0 at time 0, and, if the next state is s', then view the process as originating in state s', following the policy $\pi_{s'}$. Then

$$v_\pi(s) = r(s,a_0) + \gamma \sum_{s'} P_{ss'}(a_0) v_{\pi_{s'}}(s') \geq r(s,a_0) + \gamma \sum_{s'} P_{ss'}(a_0) v_*(s') - \gamma\epsilon.$$

Thus, $v_*(s) \geq r(s,a_0) + \gamma \sum_{s'} P_{ss'}(a_0) v_*(s') - \gamma\epsilon \;= \max_a \left[r(s,a) + \gamma \sum_{s'} P_{ss'}(a) v_*(s') \right] - \gamma\epsilon$.

The result follows by making ϵ arbitrarily small.

Bellman Optimality Equation

Corollary

The optimal value function $q_*(s,a)$ satisfies the following equation

$$q_*(s,a) = r(s,a) + \gamma \sum_{s'} P_{ss'}(a) \boxed{\max_{a' \in \mathcal{A}(s)} q_*(s',a')}, \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$$

Remark: Given $q_*(s,a)$, an optimal deterministic stationary policy can be easily obtained as $\pi(s) = \arg\max_{a} q_*(s,a)$.