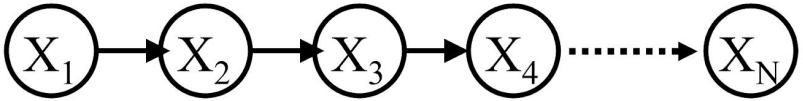
Markov Model,
Hidden Markov Model

Recall

Markov Models		Do we have control over the state transitons?	
		NO	**YES**
Are the states completely observable?	**YES**	**Markov Chain**	**MDP** Markov Decision Process
	NO	**HMM** Hidden Markov Model	**POMDP** Partially Observable Markov Decision Process

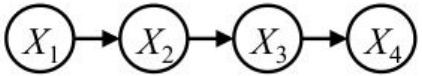
Markov Models (Markov Chains)



- A Markov model includes:
  - Random variables X_t for all time steps t (the state)
  - Parameters: called transition probabilities or dynamics, specify how the state evolves over time (also, initial probs)

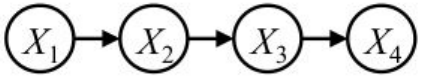
$$P(X_1) \quad \text{and} \quad P(X_t|X_{t-1})$$

Implied Conditional Independencies



- We assumed: $X_3 \perp\!\!\!\perp X_1 \mid X_2$ and $X_4 \perp\!\!\!\perp X_1, X_2 \mid X_3$
- Do we also have $X_1 \perp\!\!\!\perp X_3, X_4 \mid X_2$?

Implied Conditional Independencies



- We assumed: $X_3 \perp\!\!\!\perp X_1 \mid X_2$ and $X_4 \perp\!\!\!\perp X_1, X_2 \mid X_3$
- Do we also have $X_1 \perp\!\!\!\perp X_3, X_4 \mid X_2$?

$$P(X_1, X_2, \ldots, X_T) = P(X_1) \prod_{t=2}^{T} P(X_t | X_{t-1})$$

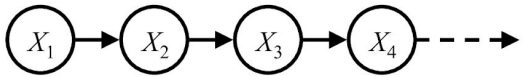
Stationary Distributions

- If we simulate the chain long enough:
  - What happens?
  - Uncertainty accumulates
  - Eventually, we have no idea what the state is!

- Stationary distributions:
  - For most chains, the distribution we end up in is independent of the initial distribution
  - Called the stationary distribution of the chain
  - Usually, can only predict a short time out

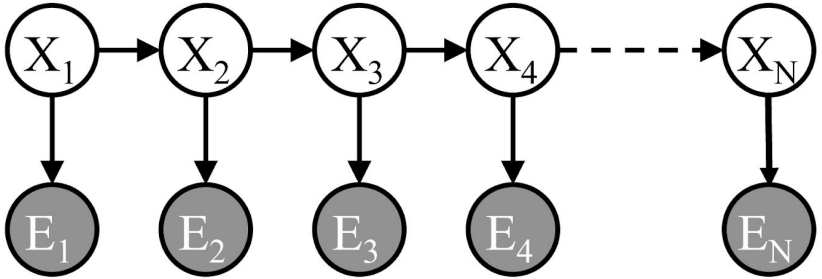
Hidden Markov Models

- Markov chains not so useful for most agents
  - Eventually you don't know anything anymore
  - Need observations to update your beliefs

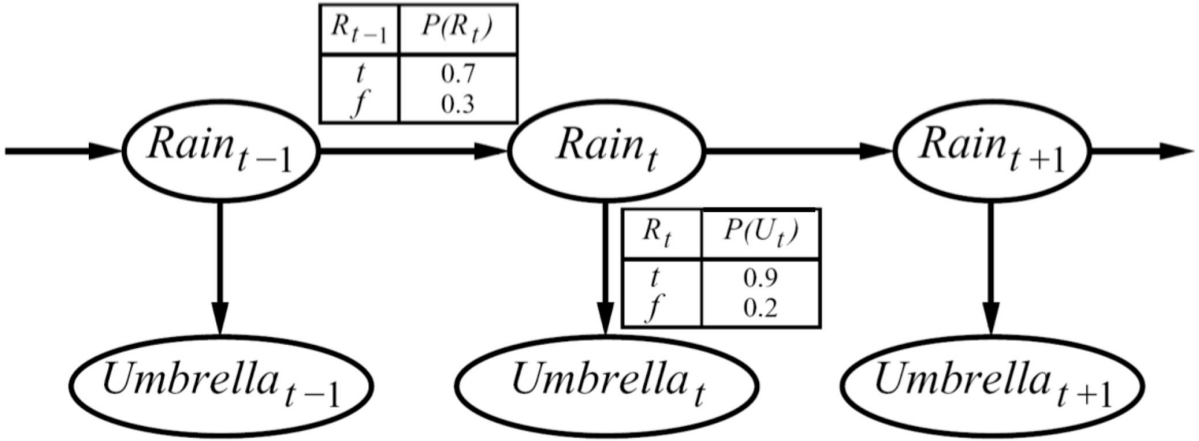


$$P(X_1) \qquad P(X|X_{-1})$$

- Hidden Markov models (HMMs)
  - Underlying Markov chain over states S
  - You observe outputs (effects) at each time step

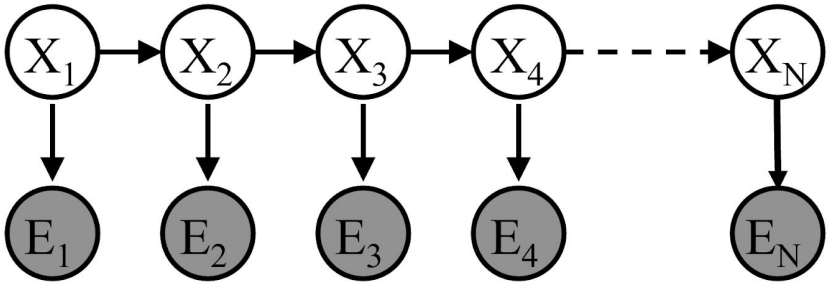


Example:



- An HMM is defined by:
  - Initial distribution: $P(X_1)$
  - Transitions: $P(X_t|X_{t-1})$
  - Emissions: $P(E|X)$

HMM



- Defines a joint probability distribution:

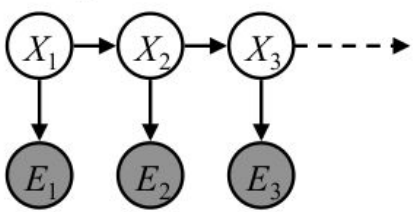
$$P(X_1, E_1, X_2, E_2, X_3, E_3) = P(X_1)P(E_1|X_1)P(X_2|X_1)P(E_2|X_2)P(X_3|X_2)P(E_3|X_3)$$

$$P(X_1, \ldots, X_n, E_1, \ldots, E_n) =$$

$$P(X_{1:n}, E_{1:n}) =$$

$$P(X_1)P(E_1|X_1)\prod_{t=2}^{N} P(X_t|X_{t-1})P(E_t|X_t)$$

Implied Conditional Independencies

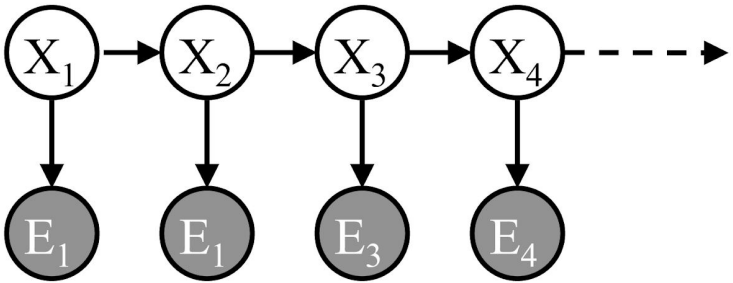


- Many implied conditional independencies, e.g.,

$$E_1 \perp\!\!\!\perp X_2, E_2, X_3, E_3 \mid X_1$$

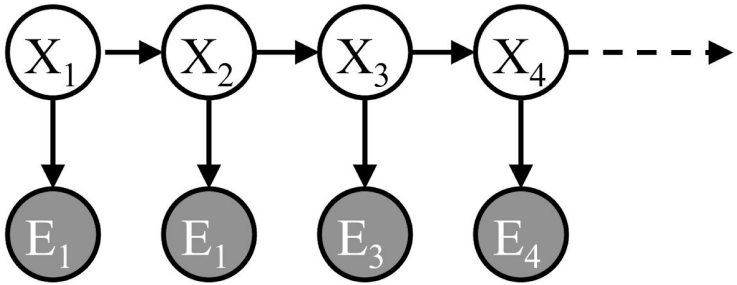
HMM Examples

- Speech recognition HMMs:
  - Observations are acoustic signals (continuous valued)
  - States are specific positions in specific words (so, tens of thousands)



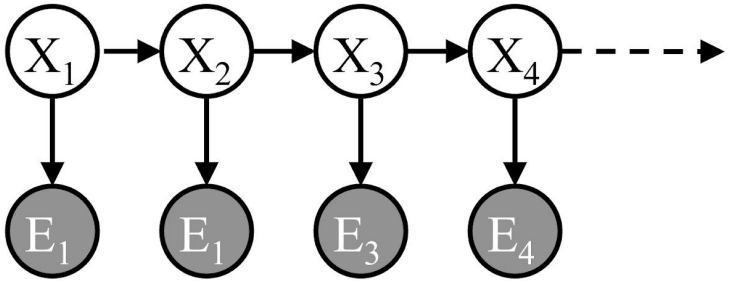
HMM Examples

- Machine translation HMMs:
  - Observations are words (tens of thousands)
  - States are translation options



HMM Examples

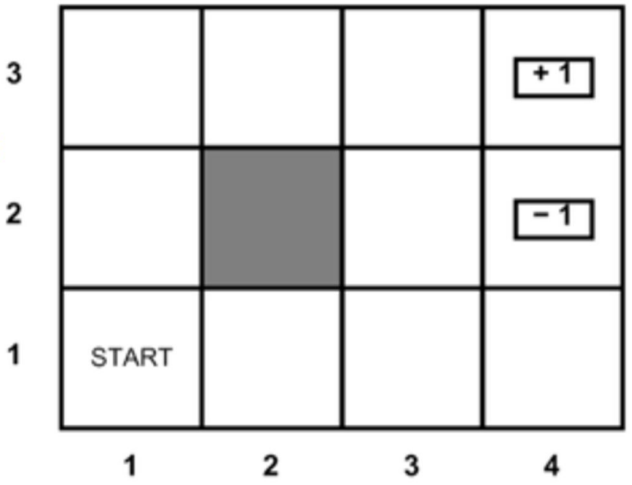
- Robot tracking:
  - Observations are range readings (continuous)
  - States are positions on a map (continuous)

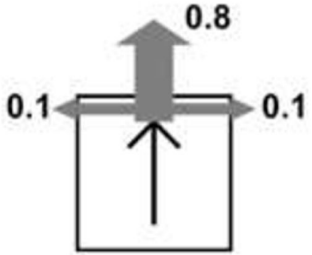


MDP and POMDP

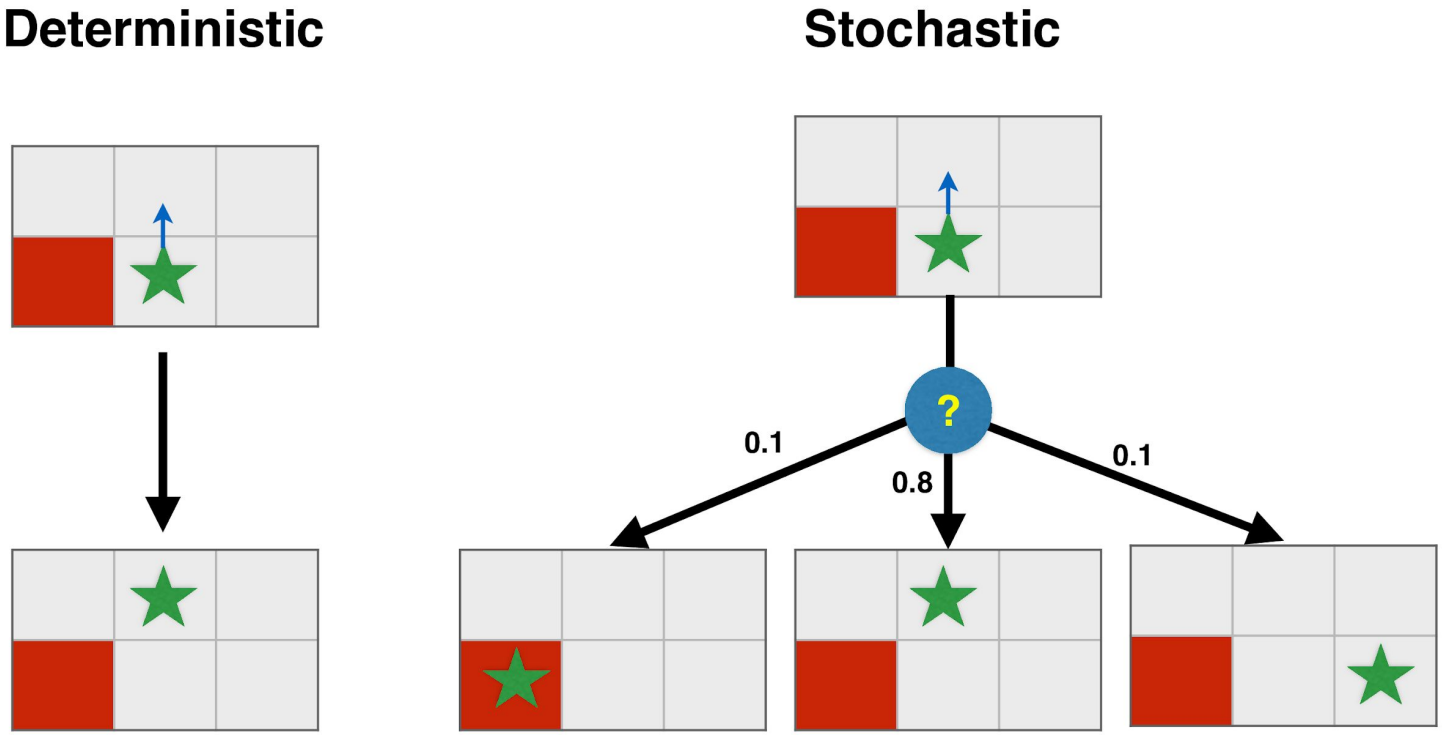
Example: Grid World

- A maze-like problem:
  - The agent lives in a grid
  - Walls block the agent's path
- 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
- Agent receives rewards each time step:
  - Small "living" reward each step
  - Big rewards come at the end
- Goal: maximize sum of rewards



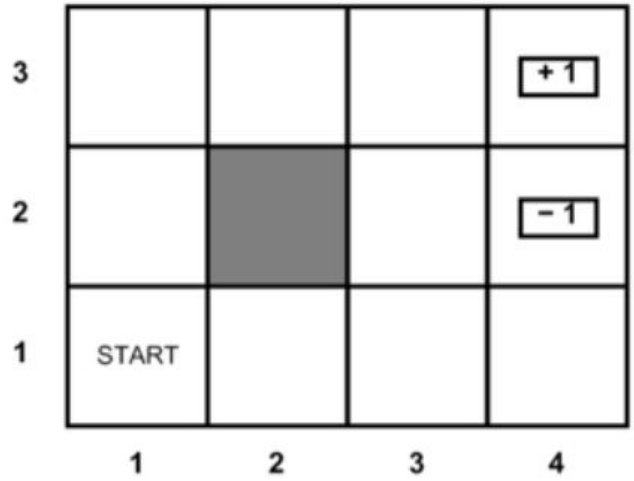


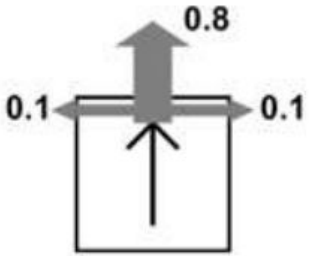
Grid World Actions



Markov Decision Processes

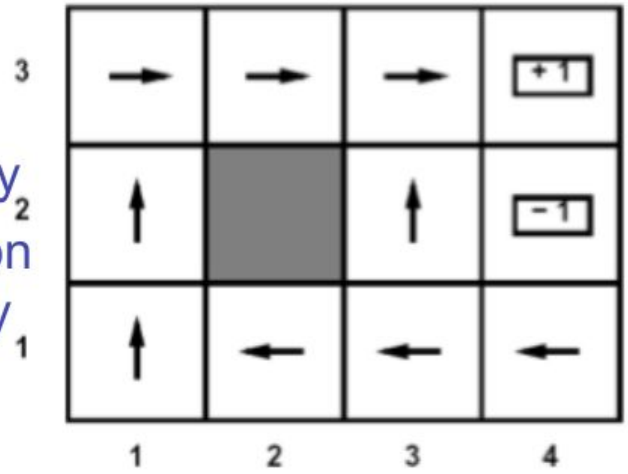
- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s,a,s')
    - Prob that a from s leads to s'
    - i.e., P(s' | s,a)
    - Also called the model
  - A reward function R(s, a, s')
    - Sometimes just R(s) or R(s')
  - A start state (or distribution)
  - Maybe a terminal state





Solving MDPs

- In deterministic single-agent search problems, want an optimal plan, or sequence of actions, from start to a goal
- 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
  - Defines a reflex agent
- Expectimax didn't compute the entire policy
  - It computed the action for a single state only



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

Episode

Each attempt of agent for the completing the task is referred to as an episode.

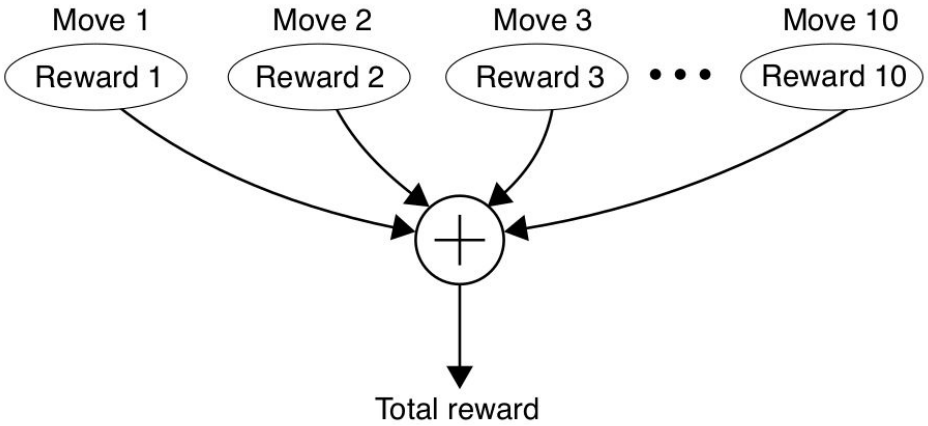
Episode is defined as

$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_n, a_n, r_n)$$

where the episode proceeds through time-steps t = 1, 2, …., n

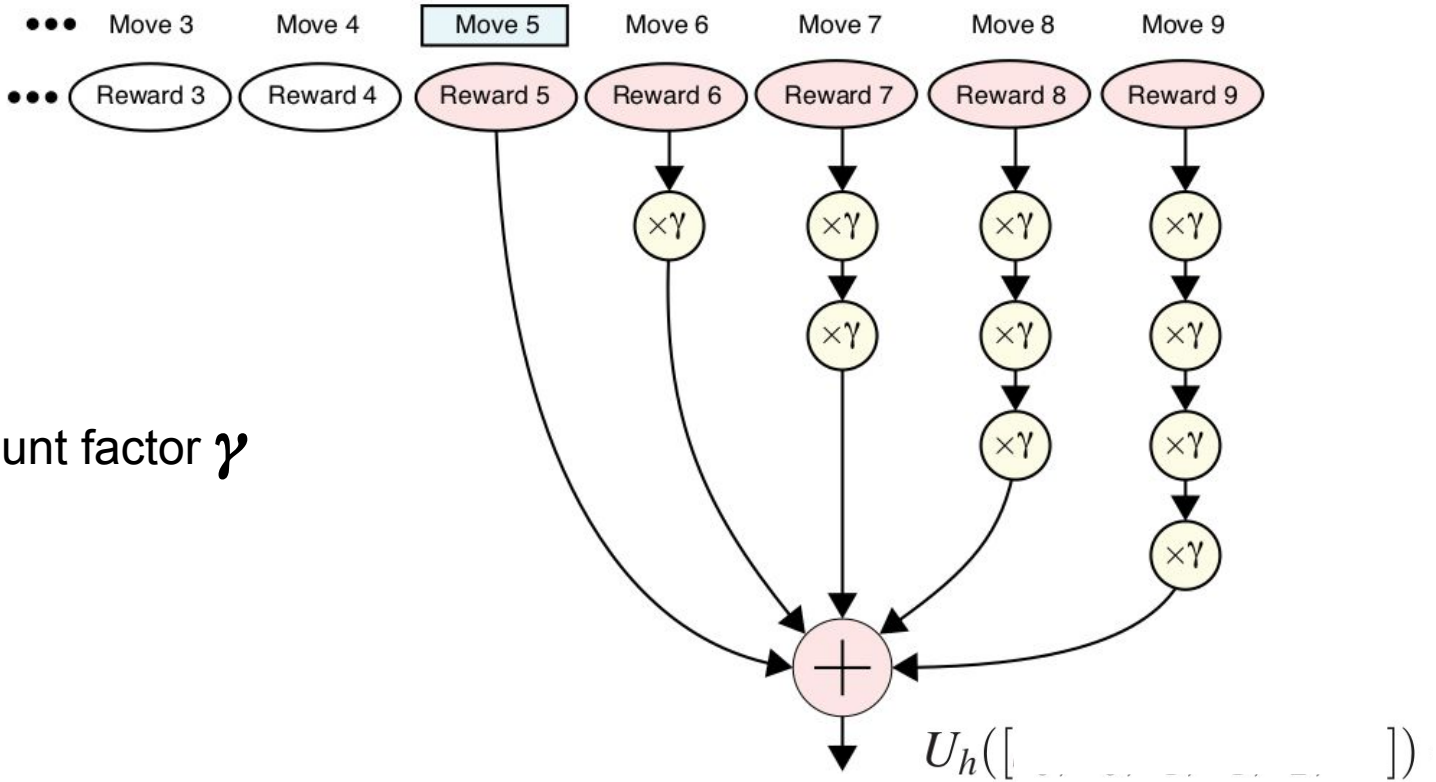
Rewards

- The only goal of an intelligent agent is to maximize cumulative reward (utility) across an episode.
- The cumulative reward earned across an episode can be defined as



Additive discounted rewards

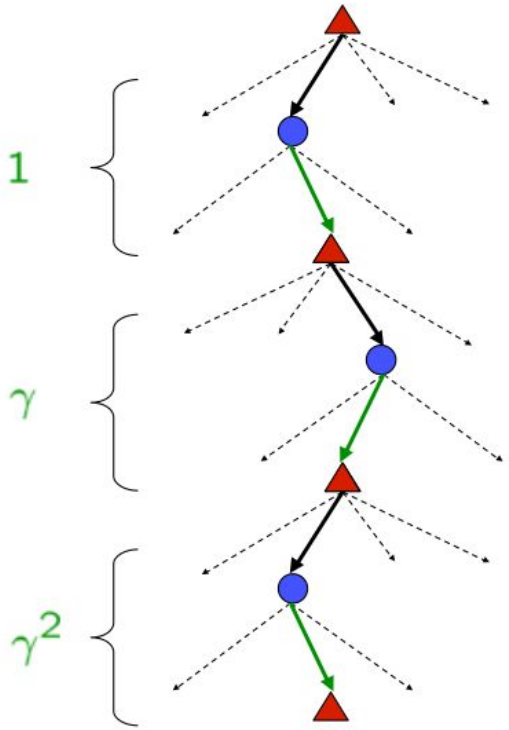
$$U_h([s_0, a_0, s_1, a_1, s_2, \ldots]) = R(s_0, a_0, s_1) + \gamma R(s_1, a_1, s_2) + \gamma^2 R(s_2, a_2, s_3) + \cdots,$$



discount factor γ

Discounting

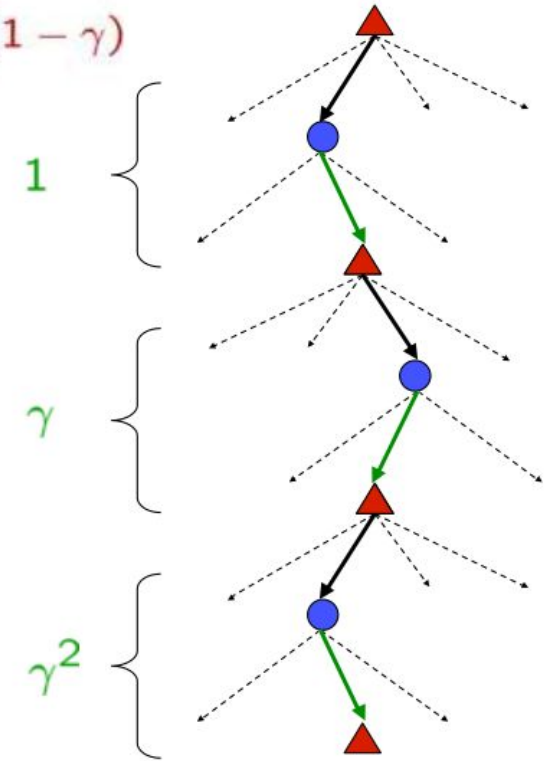
- How to discount?
  - Each time we descend, we multiply in the discount once
- Why discount?
  - Sooner rewards probably do have higher utility than later rewards
  - Also helps our algorithms converge



Discounting

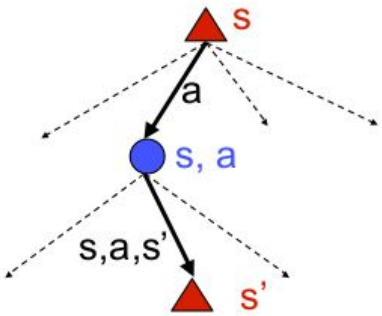
$$U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\max}/(1-\gamma)$$

- Typically discount rewards by $\gamma < 1$ each time step
  - Sooner rewards have higher utility than later rewards
  - Also helps the algorithms converge



Solving MDPs

- Markov decision processes:
  - States S
  - Start state s_0
  - Actions A
  - Transitions P(s'|s,a) (or T(s,a,s'))
  - Rewards R(s,a,s') (and discount γ)

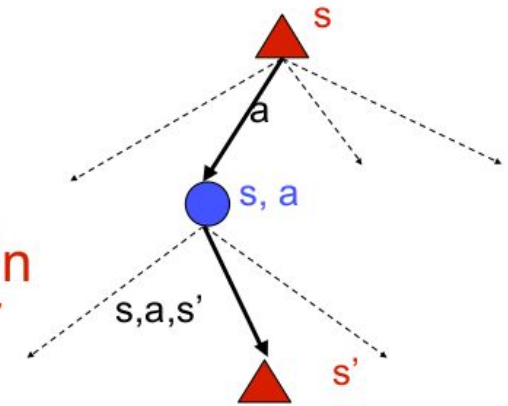


- MDP quantities so far:
  - Policy = Choice of action for each state
  - Utility (or return) = sum of discounted rewards

Optimal Utilities

- Define the value of a state s:
  $V^*(s)$ = expected utility starting in s and acting optimally

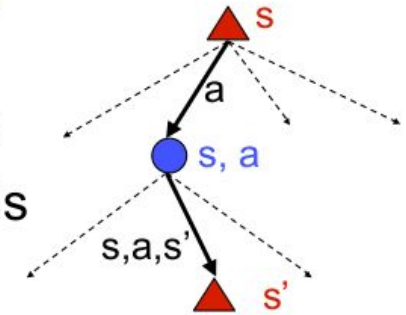
- Define the value of a q-state (s,a):
  $Q^*(s,a)$ = expected utility starting in s, taking action a and thereafter acting optimally



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

The Bellman Equations

- Definition of "optimal utility" leads to a simple one-step lookahead relationship amongst optimal utility values:
  - Expected utility under optimal action
  - Average sum of (discounted) rewards
  - This is just what expectimax does



- Formally:

$$V^*(s) = \max_a Q^*(s,a)$$

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

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

Solving MDPs

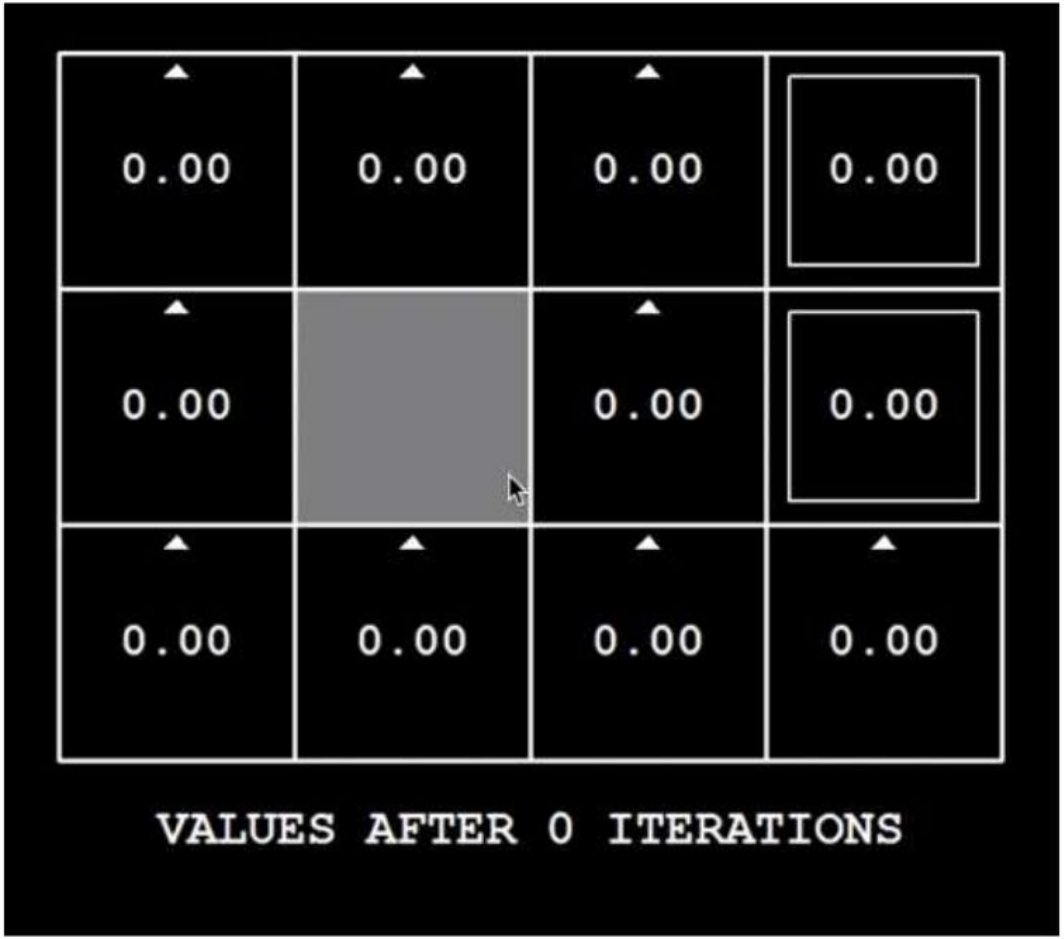
- Find V*(s) for all the states in S
  - |S| non-linear equations with |S| unknown

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

- Our proposal:
  - Dynamic programming
  - Define V*i(s) as the optimal value of s if game ends in i steps
  - V*0(s)=0 for all the states

$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

Example: γ=0.9, living reward=0, noise=0.2

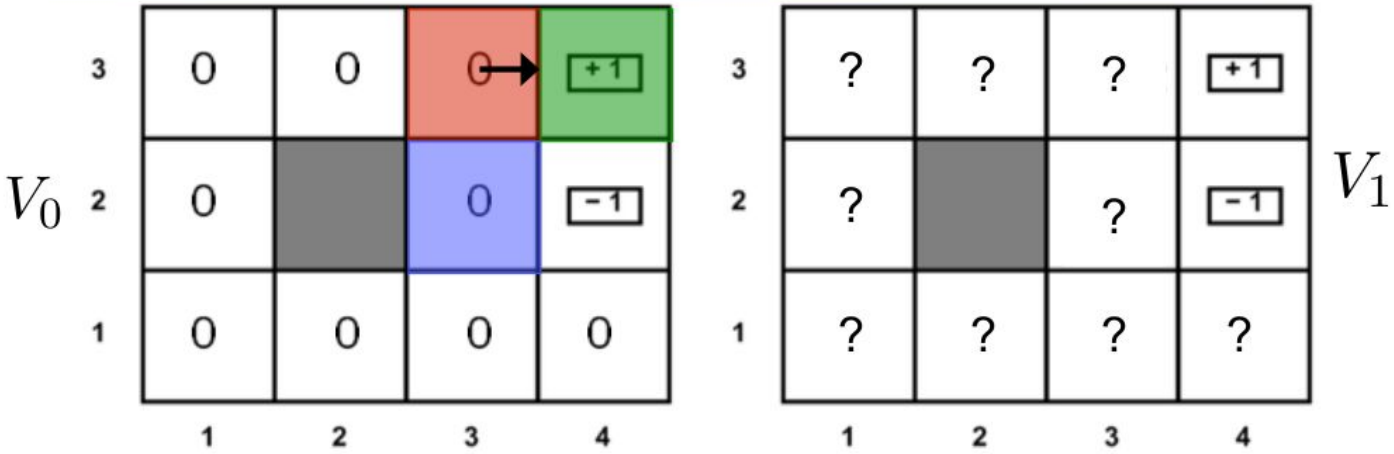



0.00
0.00
0.00
1.00
0.00
0.00
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 1 ITERATIONS


0.00
0.00
0.72
1.00
0.00
0.00
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 2 ITERATIONS

Example: Bellman Updates

Example: γ=0.9, living reward=0, noise=0.2

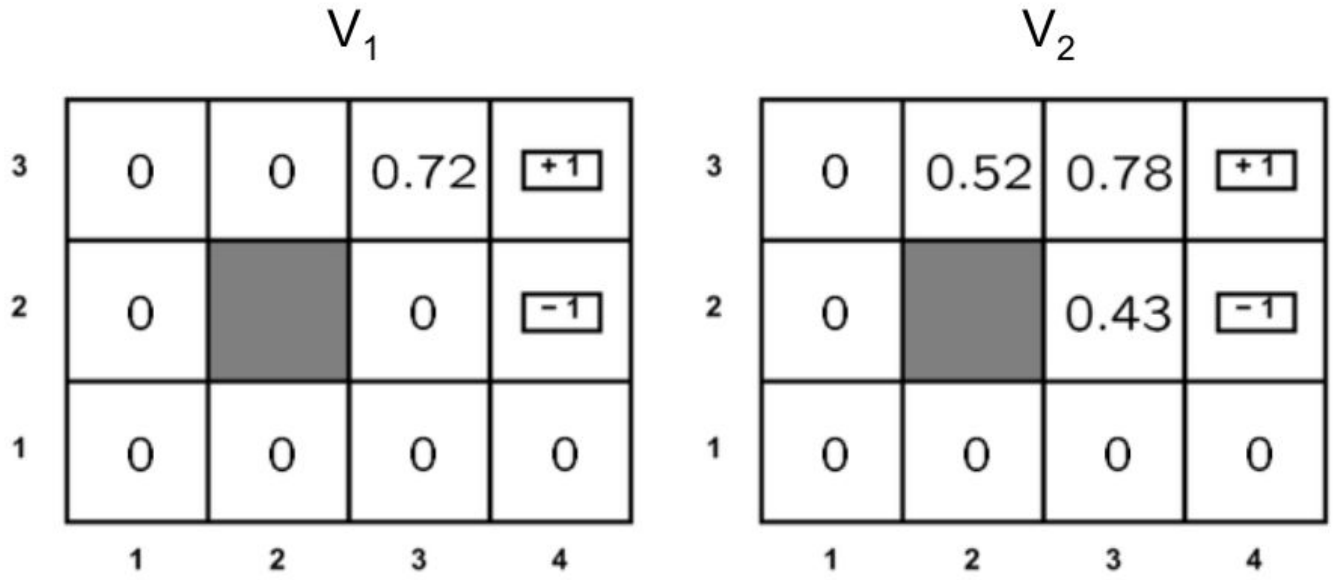


$$V_{i+1}(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s')\right] = \max_a Q_{i+1}(s,a)$$

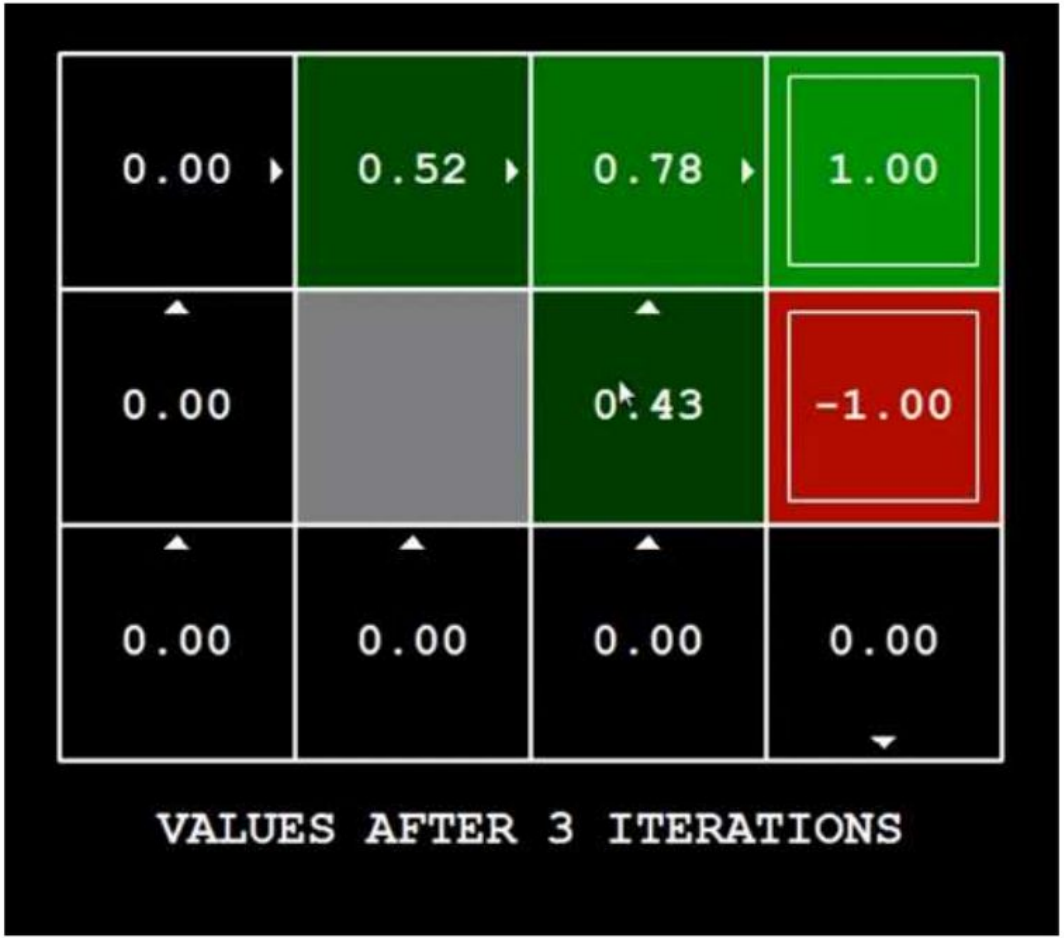
$$Q_1(\langle 3,3\rangle, \text{right}) = \sum_{s'} T(\langle 3,3\rangle, \text{right}, s') \left[R(\langle 3,3\rangle, \text{right}, s') + \gamma V_i(s')\right]$$

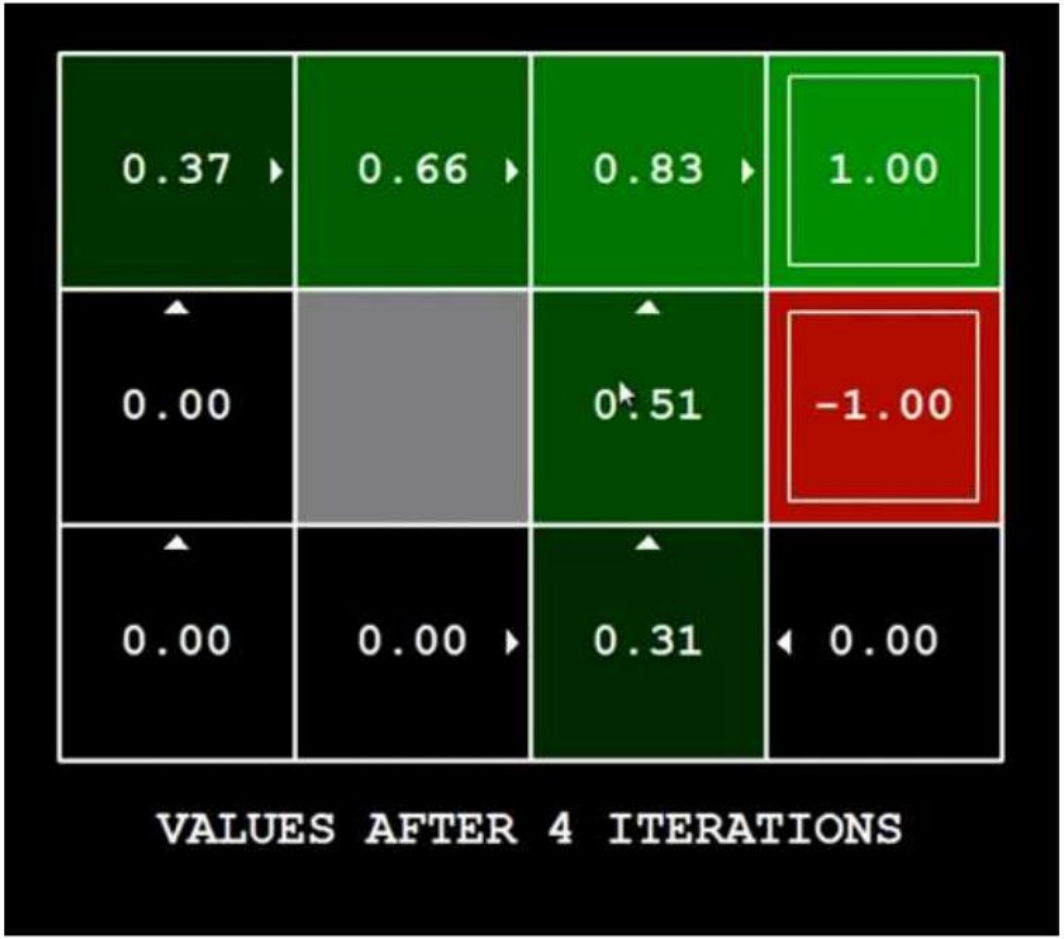
$$= 0.8 * [0.0 + 0.9 * 1.0] + 0.1 * [0.0 + 0.9 * 0.0] + 0.1 * [0.0 + 0.9 * 0.0]$$

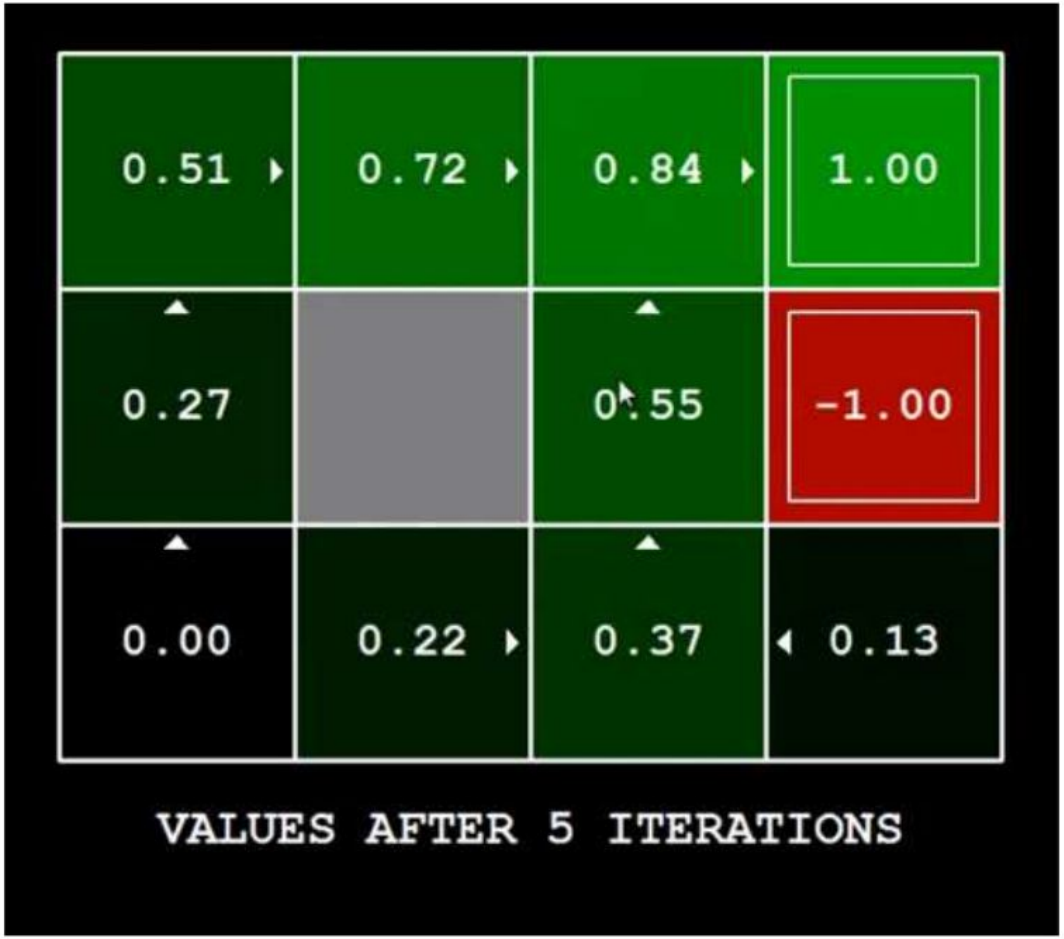
Example: Value Iteration



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


0.00
0.52
0.78
1.00
0.00
0.43
-1.00
0.00
0.00
0.00
0.00
VALUES AFTER 3 ITERATIONS


0.37
0.66
0.83
1.00
0.00
0.51
-1.00
0.00
0.00
0.31
0.00
VALUES AFTER 4 ITERATIONS


0.51
0.72
0.84
1.00
0.27
0.55
-1.00
0.00
0.22
0.37
0.13
VALUES AFTER 5 ITERATIONS


0.59
0.73
0.85
1.00
0.41
0.57
-1.00
0.21
0.31
0.43
0.19
VALUES AFTER 6 ITERATIONS


0.62
0.74
0.85
1.00
0.50
0.57
-1.00
0.34
0.36
0.45
0.24
VALUES AFTER 7 ITERATIONS

Value Iteration

- Idea:
  - Start with $V_0^*(s) = 0$, which we know is right (why?)
  - Given V_i^*, calculate the values for all states for depth i+1:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

  - This is called a value update or Bellman update
  - Repeat until convergence

- Theorem: will converge to unique optimal values
  - Basic idea: approximations get refined towards optimal values
  - Policy may converge long before values do

Value Iteration Complexity

- Problem size:
  - |A| actions and |S| states
- Each Iteration
  - Computation: $O(|A| \cdot |S|^2)$
  - Space: $O(|S|)$

Practice: Computing Actions

- Which action should we chose from state s:
  - Given optimal values Q?

$$\arg\max_a Q^*(s,a)$$

  - Given optimal values V?

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

  - Lesson: actions are easier to select from Q's!

Aside: Q-Value Iteration

- Value iteration: find successive approx optimal values
  - Start with $V_0^*(s) = 0$
  - Given V_i^*, calculate the values for all states for depth i+1:

$$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_i(s') \right]$$

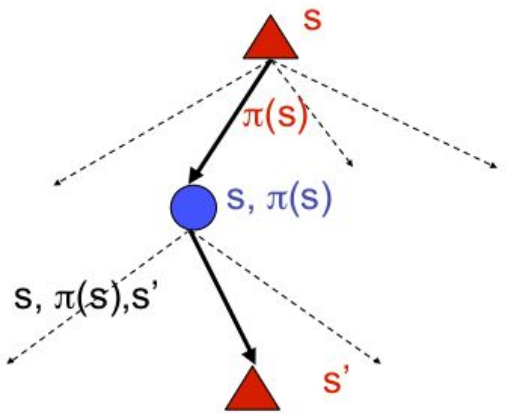
- But Q-values are more useful!
  - Start with $Q_0^*(s,a) = 0$
  - Given Q_i^*, calculate the q-values for all q-states for depth i+1:

$$Q_{i+1}(s,a) \leftarrow \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma \max_{a'} Q_i(s',a') \right]$$

Utilities for Fixed Policies

- Another basic operation: compute the utility of a state s under a fix (general non-optimal) policy
- Define the utility of a state s, under a fixed policy π:

  $V^{\pi}(s)$ = expected total discounted rewards (return) starting in s and following π
- Recursive relation (one-step look-ahead / Bellman equation):



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

- How do we calculate the V's for a fixed policy?
- Idea one: modify Bellman updates

$$V_0^\pi(s) = 0$$

$$V_{i+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_i^\pi(s')]$$

- Idea two: it's just a linear system, solve

Policy Iteration

- Problem with value iteration:
  - Considering all actions each iteration is slow: takes |A| times longer than policy evaluation
  - But policy doesn't change each iteration, time wasted
- Alternative to value iteration:
  - Step 1: Policy evaluation: calculate utilities for a fixed policy (not optimal utilities!) until convergence (fast)
  - Step 2: Policy improvement: update policy using one-step lookahead with resulting converged (but not optimal!) utilities (slow but infrequent)
  - Repeat steps until policy converges

Policy Iteration

- Policy evaluation: with fixed current policy π, find values with simplified Bellman updates
  - Iterate until values converge

$$V_{i+1}^{\pi_k}(s) \leftarrow \sum_{s'} T(s, \pi_k(s), s') \left[R(s, \pi_k(s), s') + \gamma V_i^{\pi_k}(s') \right]$$

  - Note: could also solve value equations with other techniques

- Policy improvement: with fixed utilities, find the best action according to one-step look-ahead

$$\pi_{k+1}(s) = \arg\max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V^{\pi_k}(s') \right]$$

Policy Iteration

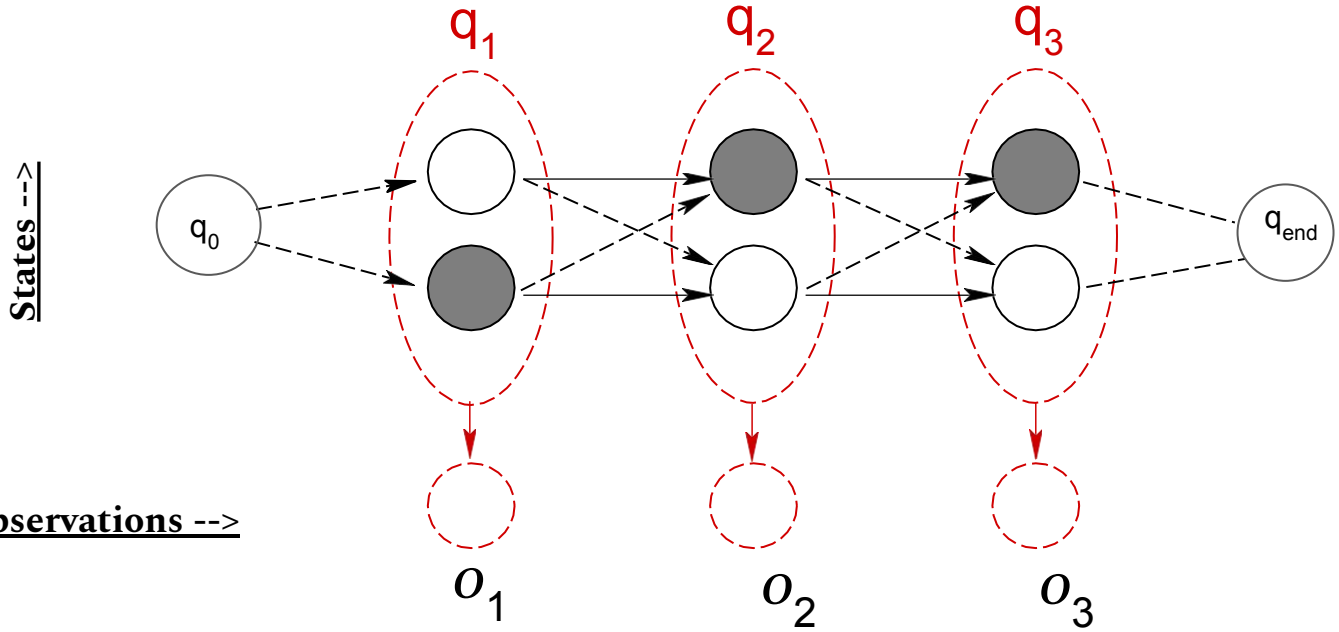
- Problem size:
  - |A| actions and |S| states
- Each Iteration
  - Computation: $O(|S|^3 + |A| \cdot |S|^2)$
  - Space: O(|S|)
- Num of iterations
  - Unknown, but can be faster in practice
  - Convergence is guaranteed

Comparison

- In value iteration:
  - Every pass (or “backup”) updates both utilities (explicitly, based on current utilities) and policy (possibly implicitly, based on current policy)
- In policy iteration:
  - Several passes to update utilities with frozen policy
  - Occasional passes to update policies

Partially Observable Markov decision process (POMDP)

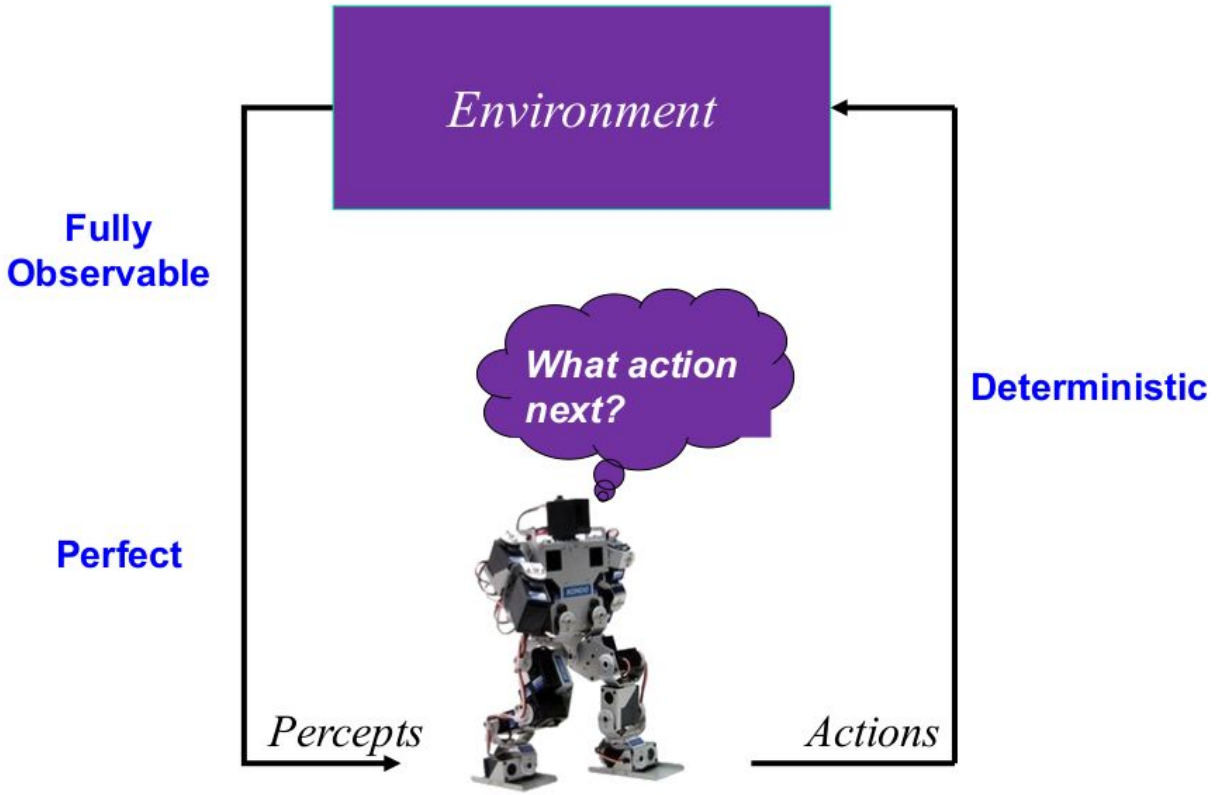
Hidden Markov Model (recall)



Conditional Probabilities

$$\underset{q_1, q_2, q_3}{\text{argmax}} \; P(q=q_1, q_2, q_3 | O = o_1, o_2, o_3)$$

Classical

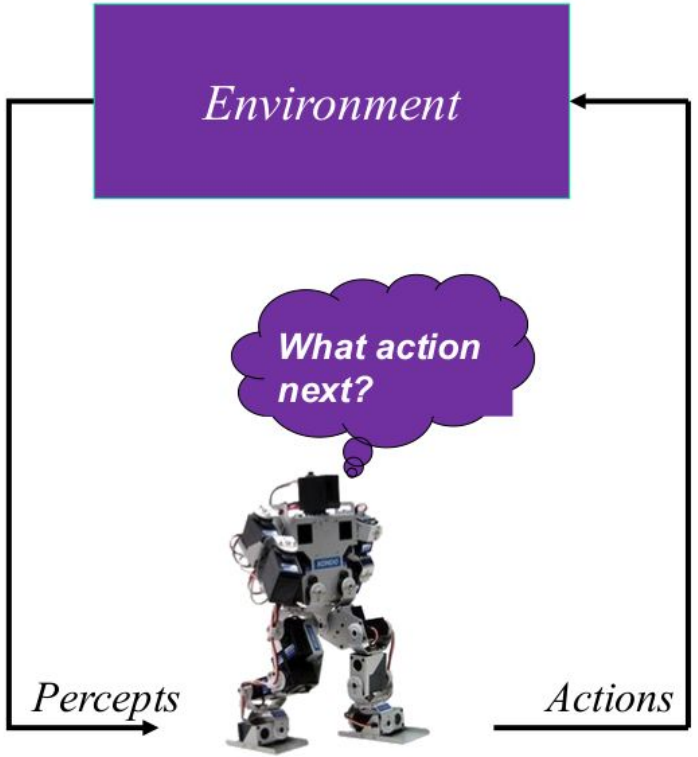

Environment
Fully Observable
What action next?
Deterministic
Perfect
Percepts
Actions

Stochastic (MDP)

- **S**: set of states
- **A**: set of actions
- **Pr**(s'|s,a): transition model
- **R**(s,a,s'): reward model
- γ: discount factor
- s_0: start state

Fully Observable

Perfect



Stochastic
T(s, a, s')

Partially-Observable Stochastic (POMDP)

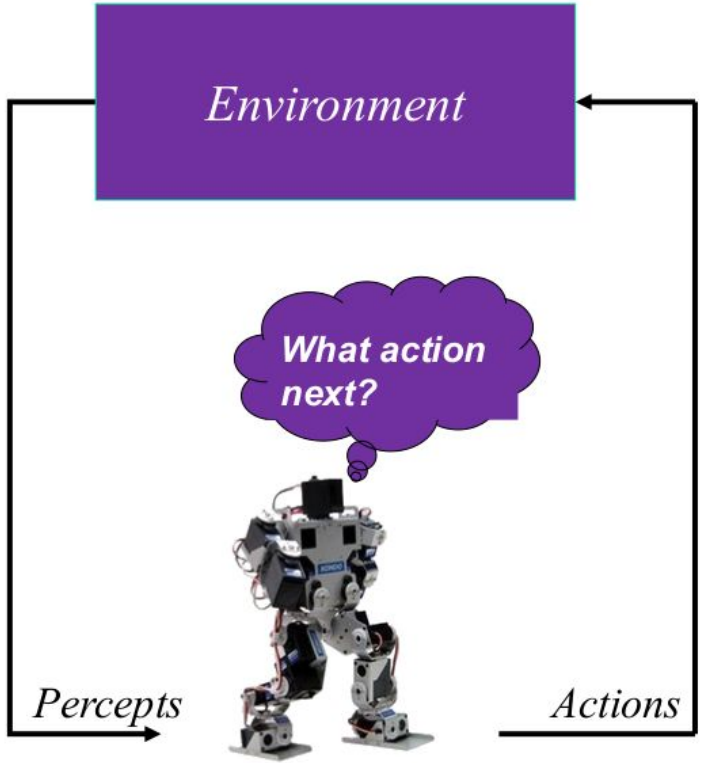
- **S**: set of states
- **A**: set of actions
- **Pr(s'|s,a)**: transition model
- **R(s,a,s')**: reward model
- **γ**: discount factor
- s_0: start state
- **E** set of possible evidence

(observation)

- **Pr(e|s)**

Partially Observable

Noisy



Stochastic T(s, a, s')

Objective of a Fully Observable MDP

- Find a policy

  π: **S** → **A**

- which maximizes expected discounted reward
  - given an infinite horizon
  - assuming full observability

Objective of a POMDP

- Find a policy

  π: BeliefStates($\mathbf{S}$) → $\mathbf{A}$

  A belief state is a ***probability distribution*** over states

- which maximizes expected discounted reward

  - given an infinite horizon
  - assuming ***partial*** & ***noisy*** observability

POMDPs

- In POMDPs we apply the very same idea as in MDPs.
- Since the state is not observable, the agent has to make its decisions based on the ***belief state*** which is a ***posterior distribution over states***.

  π : beliefs → actions

POMDP

Decision cycle of a POMDP agent can be broken down into following 3 steps:

1. Given the current belief state b, execute the action $a = \pi^*(b)$.

2. Receive percept e.

3. Set the current belief state to $b' = \text{FORWARD}(b, a, e)$ and repeat.

Transition model for belief-states

- Let’s calculate the probability that an agent in belief state b reaches belief state b' after executing action a.

$$P(b' \mid b, a) = P(b'|a, b) = \sum_e P(b'|e, a, b) P(e|a, b)$$

Probability of b’, given the percept e

Probability of perceiving ***e***, given that action ***a*** was performed in b

Sensor model

probability of perceiving ***e***, given that ***a*** was performed starting in belief state b, is given by summing over all the actual states s' that the agent might reach:

$$P(e|a,b) = \sum_{s'} P(e|a,s',b)P(s'|a,b)$$

Probability of percept e, given the state s'

Probability of state s', given action *a* on belief state b

Sensor model

probability of perceiving ***e***, given that ***a*** was performed starting in belief state b, is given by summing over all the actual states s' that the agent might reach:

$$P(e|a,b) = \sum_{s'} P(e|a,s',b)P(s'|a,b)$$

$$= \sum_{s'} P(e \mid s')P(s'|a,b)$$

Probability of percept e, given the state s'

Probability of state s', given action *a* on belief state b

POMDP

probability of perceiving ***e***, given that ***a*** was performed starting in belief state b, is given by summing over all the actual states s' that the agent might reach:

$$
\begin{aligned}
P(e|a,b) &= \sum_{s'} P(e|a,s',b)P(s'|a,b) \\
&= \sum_{s'} P(e \mid s')P(s'|a,b) \\
&= \sum_{s'} P(e \mid s') \sum_{s} P(s' \mid s,a)b(s) \ .
\end{aligned}
$$

Probability of percept e, given the state s'

Probability of s', given s and action a with probability of state s

Transition model (belief-state space)

Let us write the probability of reaching b' from b, given action a, as P (b' | b, a)). Then that gives us

$$P(b' \mid b, a) = P(b'|a, b) = \sum_e P(b'|e, a, b)P(e|a, b)$$

$$= \sum_e P(b'|e, a, b) \sum_{s'} P(e \mid s') \sum_s P(s' \mid s, a)b(s)$$

Probability of b', given percept e

Probability of percept e, given the state s'

Probability of s', given s and action a with probability of state s

above can be viewed as defining a transition model for belief-state space.

Reward function

We can also define a reward function for belief states (i.e., the expected reward for the actual states the agent might be in):

$$\rho(b) = \sum_s b(s)R(s)$$

Together, P(b' | b, a) and ρ(b) define an observable MDP on space of belief states.