CS 440
Introduction to Artificial Intelligence

Lecture 16:

Markov Decision Processes (MDPs)

March 10, 2020

Markov Processes

- Consider an environment
  - Environment may transition to different states
    - Due to actions selected by agent
    - Due to things outside of agent's control
- Example: Autonomous car
  - State of road changes based on what agent does as well as what other agents do
    - Agent may turn, change lanes, accelerate/decelerate, ect.
    - Other cars may move, change lanes, cut you off, ect.
- Very complicated
  - Impossible to predict exactly
- Given state of environment possible to estimate future state
  - Give you position of cars with current speed
    - Predict likely position of cars after certain amount of time has passed

Markov Processes

- **Markovian assumption**
  - **Future state only depend on current state**
  - **Only matters where cars currently are on road**
    - **Doesn't matter what maneuvers they took to get there**
  - **Assumption makes solving problems a lot easier**
    - **Only need to keep track of current state**
      - **As opposed to history of previous states**
    - **Only need to reason over current state**

Types of Markov Processes

- Discrete vs continuous
- Passive vs active
  - Active process: The agent's actions influence process
- Observable state vs partially observable state
  - Observable state: agent can observe state directly
  - Partially observable state:
    - Example: Wumpus world

	Observable State	Hidden State
Passive	Markov Chain	Hidden Markov Model
Active	Markov Decision Process (MPD)	Partially Observable Markov Decision Process (POMPD)

Markov Chain

- May be discrete or continuous
- Passive - state of environment does not depend of actions of agent
- Fully observable
- May be discrete or continuous
  - Discrete Markov chains can be represented as finite state machines
- Example: Bit flip
  - String of bits
    - State: string of 1s and 0s
  - At each time step each bit has a p probability of flipping

Solutions to Markov Chain

- Given an environment in a known state
  - What could the environment look like after n steps?
  - Probability of being in each state
- Solve inductively
  - Assume we know the probability you are in each state s at step i
    - $p_{s,i}$ for all $s \in S$
  - Compute probability for each state at step i+1
    - $p_{s',i+1}$ for all $s' \in S$
  - Probability we will transition from state s to s' at step i+1 equal to probability we are in state s at step i times the transition probability from s to s'
    - $p_{s',i+1} = p(s'|s) * p_{s,i}$
  - Probability we will be in state s' at step i+1
    - $p_{s',i+1} = \sum_{s' \in S} p(s'|s) * p_{s,i}$

Example: Bit Flipping

- **Example: Bit flip**
  - **String of bits 2**
    - **States 00, 01, 10, 11**
    - **Transitions: Each bit has a p probability of flipping**

T =

	00	01	10	11
00	$(1-p)^2$	$p(1-p)$	$p(1-p)$	p^2
01	$p(1-p)$	$(1-p)^2$	p^2	$p(1-p)$
10	$p(1-p)$	p^2	$(1-p)^2$	$p(1-p)$
11	p^2	$(1-p)^2$	$(1-p)^2$	$(1-p)^2$

  - **Probability at step n**

n	00	01	10	11
0	1.0	0	0	0
1	$(1-p)^2 =$ .81	$p(1-p) =$ .09	$p(1-p) =$ .09	$p^2 =$ .01
2	.6724	.1476	.1476	.0324

Can we formulate as a matrix multiplication problem?

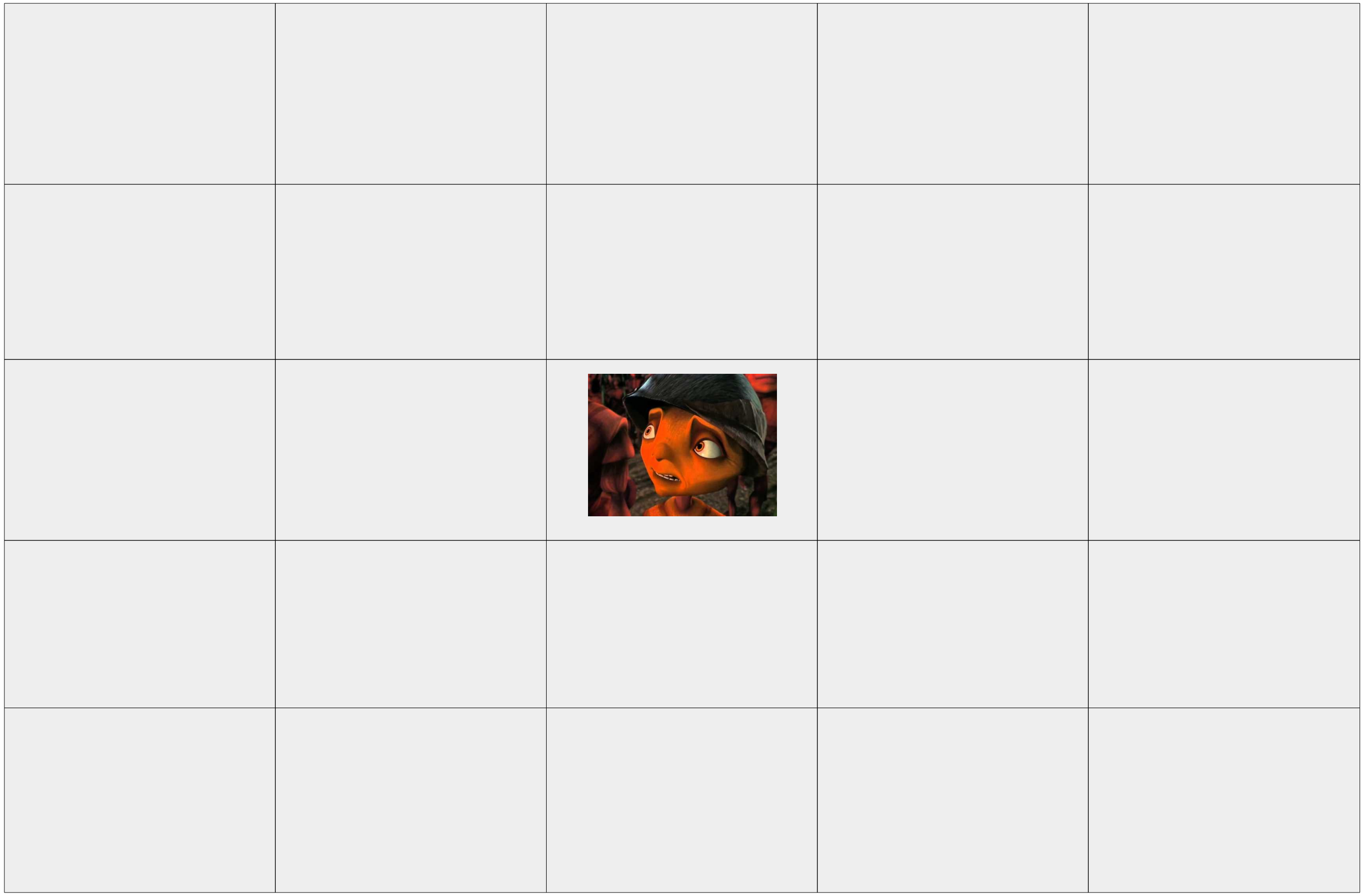
Solve By Matrix Multiplication

- Let $P_i = \{P_{s1,i}, P_{s2,i}, P_{s3,i}, \ldots\}$
- Let $P_{i+1} = \{P_{s1,i+1}, P_{s2,i+1}, P_{s3,i+1}, \ldots\}$
- Let T be a transition function
- $P_{i+1} = T * P_i^T$
- Compute P_{i+1} by repetitively multiplying by T
  - $P_{i+1} = T^{i+1} * P_0^T$
- Can use parallel computing to expedite these computations

- $P_{i+1} = T^{i+1} * P_0^T$ OR $P_{i+1} = P_0^T * T^{i+1}$
  - Depends how you define T
    - p(y|x)
    - X row and Y column ->
    - X row and Y column ->
  - Doesn't matter for this example because T is symmetric

Markov Chains Example

- **And moves up/down/left/right with probability p**
  - **stays in same cell with probability 1-p**
- **We don't know what move the ant will make**



Markov Decision Problem

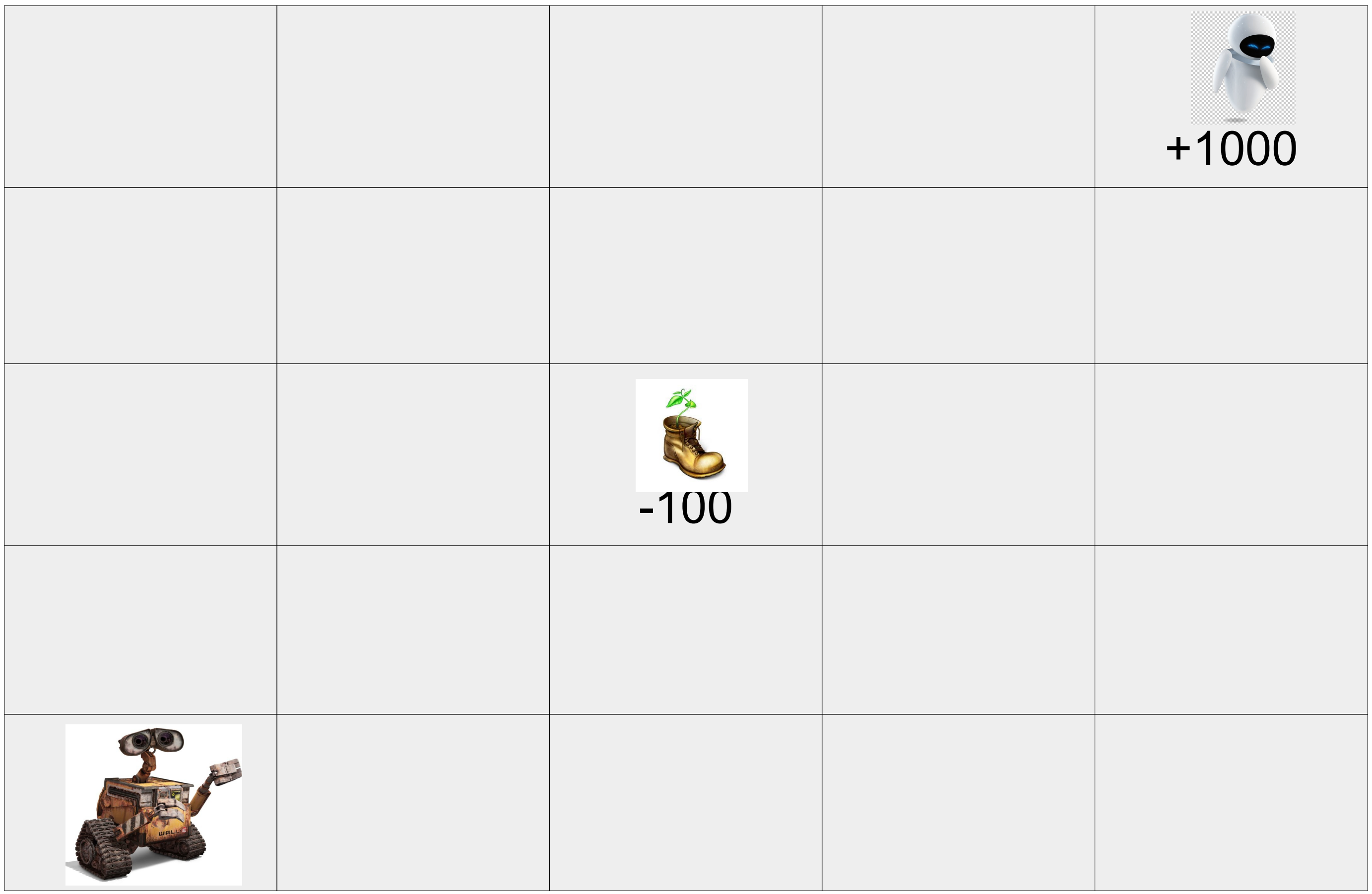
- **May be discrete or continuous**
- **Active - agents actions effect state**
- **Fully observable**
- **May be discrete or continuous**

Formalization

- State space S
- Set of actions A
- Transition function T(s,a,s')
  - T(s,a,s') = p(s'|s',a)
    - Probability that you will end up in state s' if you take action a while in state s
    - Defined for all combinations of $s \in S$, $a \in A$, $s' \in S$
- Reward function R
  - Could define reward of being in a state, R(s)
  - Could define reward of performing action while in state R(s,a)
  - Could be reward of performing action that ends up in a particular state R(a,s')
- Immediate objective: Determine the best action to take given your current state
  - Action that maximizes expected future reward

Markov Chains Example

- **Robot in a grid with noisy actions**
  - **Robot can choose Left/Right/Up/Down**
  - **Actions may bring robot to wrong cell**



Possible Solutions

- Objective: find best action
- Search
  - Branching faction equal to number of actions
  - For each node in search tree need probability for each state
  - Need to compute for every state
  - Can blow up quickly

Policy

- **Policy is a mapping of states to actions**
  - **$\Pi(s) \rightarrow a$**
- **Policies are solutions to MDPs**
- **Optimal policy is a policy that maps each state to the action which maximizes expected future reward.**

Computing Policies

- **Ideas?**

Computing Policies

- Construct a policy that is optimal for next n moves
  - Define Π_n to be a policy that is optimal for n steps
  - Define R_n to be the expected reward for this policy
- Construct inductively
  - Assume you have a policy Π_i that is optimal over i steps
  - $R_{i+1}(s) = \max_{a \in A}(R(a,s) + \sum_{s \in S} p(s'|s)\, R_i(s'))$
  - $\Pi_{i+1}(s) = \mathrm{argumax}_{a \in A}(R(a,s) + \sum_{s \in S} p(s'|s)\, R_i(s'))$
- What can you say if $R_{i+1}(s) = R_i(s)$ and $\Pi_{i+1}(s) = \Pi_i(s)$?

Computing Policies

- Construct a policy that is optimal for next n moves
  - Define Π_n to be a policy that is optimal for n steps
  - Define R_n to be the expected reward for this policy
- Construct inductively
  - Assume you have a policy Π_i that is optimal over i steps
  - $R_{i+1}(s) = \max_{a \in A}(R(a,s) + \sum_{s \in S} p(s'|s) R_i(s'))$
  - $\Pi_{i+1}(s) = \mathrm{argumax}_{a \in A}(R(a,s) + \sum_{s \in S} p(s'|s) R_i(s'))$
- What can you say if $R_{i+1}(s) = R_i(s)$ and $\Pi_{i+1}(s) = \Pi_i(s)$?
  - Policy won't change for all future iterations
  - $\Pi_{i+1}(s)$ is an optimal policy

Computing Policies

- Idea: iteratively compute $R_{i+1}(s)$ and $\Pi_{i+1}(s)$ from $R_{i+1}(s)$ until it converges to optimal
  - do
    - For all $s \in S$, $a \in A$, $s' \in S$
      - $R_{i+1}(s) = \max_{a \in A}(R(a,s) + \sum_{s \in S} p(s'|s) R_i(s'))$
      - $\Pi_{i+1}(s) = \text{argumax}_{a \in A}(R(a,s) + \sum_{s \in S} p(s'|s) R_i(s'))$
  - Until $R_{i+1}(s) = R_i(s)$ and $\Pi_{i+1}(s) = \Pi_i(s)$
- Problem
  - Does not take into account number of steps to get to goal
    - Sequence of n moves to goal yields same reward as single move to goal

Discounting

- **Multiply reward of future steps by discounting factor α**
- **do**
  - **For all $s \in S$, $a \in A$, $s' \in S$**
    - $R_{i+1}(s) = \max_{a \in A}(R(a,s) + \alpha \sum_{s \in S} p(s'|s) R_i(s'))$
    - $\Pi_{i+1}(s) = \text{argumax}_{a \in A}(R(a,s) + \alpha \sum_{s \in S} p(s'|s) R_i(s'))$
- **Until $R_{i+1}(s) = R_i(s)$ and $\Pi_{i+1}(s) = \Pi_i(s)$**

Markov Chains Example

- **Robot in a grid with noisy actions**
  - **Robot can choose Left/Right/Up/Down**
  - **Actions may bring robot to wrong cell**

Types of Markov Processes

- **Discrete vs continuous**
- **Passive vs active**
  - **Active process: The agent's actions influence process**
- **Observable state vs partially observable state**
  - **Observable state: agent can observe state directly**
  - **Partially observable state:**
    - **Example: Wumpus world**

	Observable State	Hidden State
Passive	Markov Chain	Hidden Markov Model
Active	Markov Decision Process (MPD)	Partially Observable Markov Decision Process (POMPD)

Hidden Markov Model

- **May be discrete or continuous**
- **Passive - agents actions don't effect state**
- **PArtially observable**
- **May be discrete or continuous**

Hidden Markov Model: Example

- One an ordinary day Bob has a .7 probability of being in his office
- If Bob is busy working on a paper deadline he has a .9 probability of being in his office
- If he is sick and there is no paper deadline he has a .1 probability of being in his office
- If he is sick but there is a paper deadline he has a .6 probability of being in his office
- John does not know if Bob has a paper deadline of if he is sick
  - He only knows if Bob is in his office
  - Can he infer the probability Bob has a paper deadline or is sick based on these observations?
  - Can he predict if Bob will be in his office?

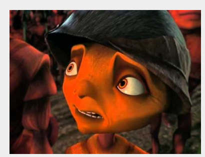
Hidden Markov Model: Example

	Not Sick	Sick
Paper	.9	.6
No Paper	.7	.1

- **Also need to know transition probability of paper deadline and being sick**
  - **If Bob has a paper deadline at time step i there is a .9 probability he will have a paper deadline at step i+1**
  - **If Bob is sick at time step i there is a .8 probability he will be sick at step i+1**

Hidden Markov Model Example 2

- **Food at corner of grid**
  - **Don't know where food is**
  - **And has .3 probability of moving towards food and .1 probability of moving away**

		.1		
	.1		..3	
		.3