EECS 391: Introduction to AI

Soumya Ray

http://vorlon.case.edu/~sray/eecs391_sp12/index.html

Email: sray@case.edu

Office: Olin 516

Office hours: Tues 2:30-4:00 or by appointment

Today

- Sequential Decision Making Under Uncertainty (Read Chapter 17.1 to 17.3)

Taking Actions

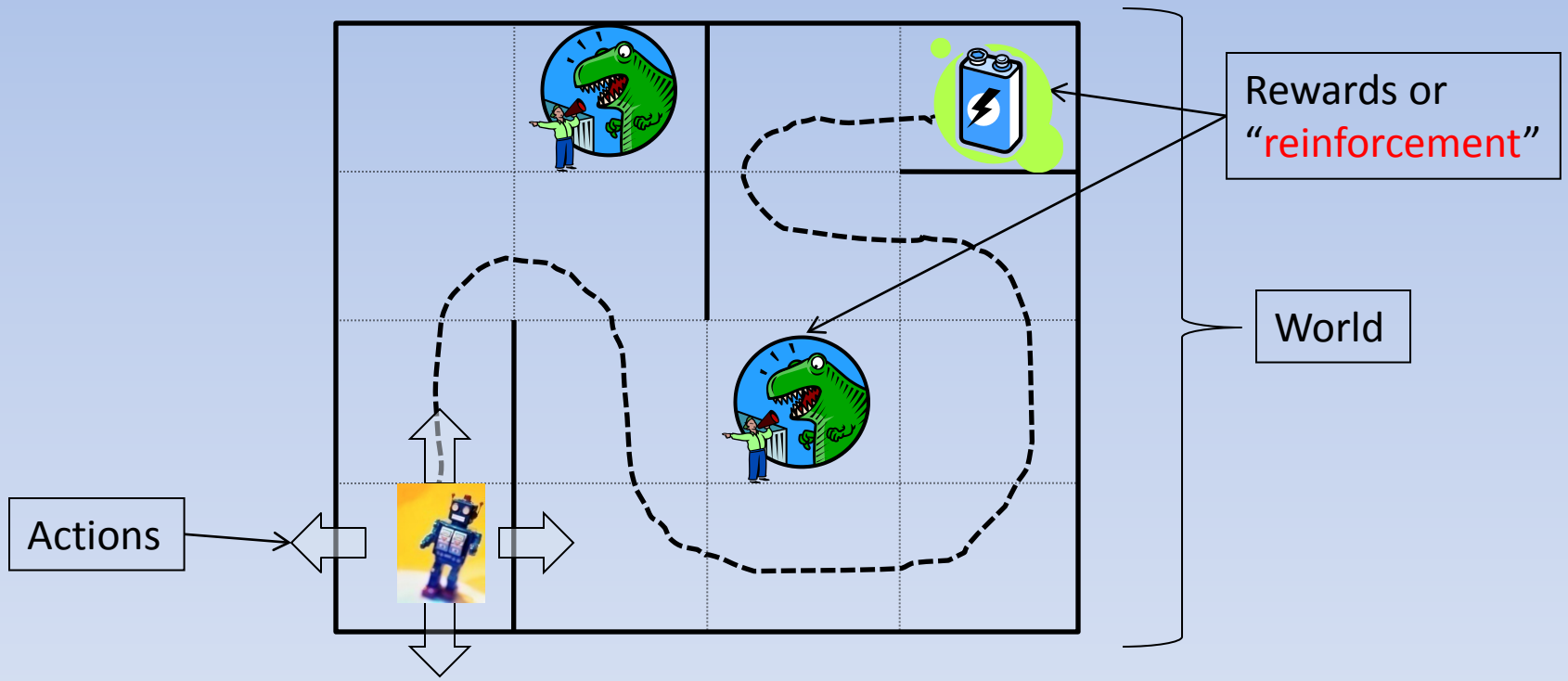
- So far
  - General probabilistic reasoning
  - Machine Learning for “one-shot” decision problems

- Now: how to *take actions* based on information about the world, “while living in it”

Extending Automated Planning

- If we possess detailed knowledge about the environment, we can use logic-based planning to take actions
- But what if the knowledge we have is incomplete or uncertain?
- Or what if what the agent wants to optimize is more complex than the shortest cost path?

Sequential Decision Making Under Uncertainty



Goal: Find a sequence of actions that maximizes "expected future reward"

Sequential Decision Making

- Agent starts in some state of the world
- Repeat until done:
  - Agent takes an action based on current knowledge
  - The world provides some feedback about the action (maybe)
  - The state of the world changes (and agent gets more knowledge about world)
- "Done": Some sort of terminal or goal state is reached

Sequential Decision Making

- A very general framework

- If we want to deploy agents in the real world, they will invariably need to tackle problems of this sort

SDM and Supervised Learning

- SDMs are *sequential,* supervised learning is not
  - Each example in supervised learning is considered to be independent of other examples
- SDMs have no “teacher”/oracle, just some intermittent feedback
- Generally SDMs are harder than supervised learning (often much harder)

SDM and Classical Planning

- Classical planning is a type of SDM, with some restrictions

SDM and Classical Planning

Reinforcement Learning

- Agent starts with no initial knowledge
- Stochastic Worlds/Actions
- Produces “policy”: optimal action for each state
- Propositional only
- Optimize Utility

Classical Planning

- Agent starts with detailed structured knowledge
- Deterministic Worlds/Actions
- Produces “plan”: optimal action sequence from initial state
- Can be extended to first-order worlds
- Goal-Directed

Issues in Sequential Decision Making

- Credit Assignment
  - Suppose the agent performs a sequence of actions, and then the world gives it a reward (or penalty)
  - Which action(s) in the sequence were really responsible for this reward (or penalty)?

Issues in Sequential Decision Making

- Exploration versus Exploitation
  - Generally, the agent will not start off by knowing the characteristics of the world it is in, specifically, how to get to the high utility states
    - It has to discover these by *exploring* the world
  - Suppose it has explored a bit and found some sequence of actions that looks good
    - Should it just follow (*exploit*) this sequence or explore some more and possibly find an even better sequence?

SDM Formalization

- A formal model for an SDM is defined via a Markov Decision Process (MDP)
- An MDP has six components:
  - A set of states, S, representing possible states of the world
  - A set of actions, A, representing possible actions of the agent
  - A transition function, T
  - A reward function, R
  - An initial state distribution, P_0
  - A "discount factor", $0 \leq \gamma \leq 1$

Transition Function

- The transition function maps a state and action to a probability over the next state: $T: S \times A \times S \rightarrow [0,1]$
  - $T(s,a,s') = Pr(s'|s,a)$

Markov property: The next state only depends on the current state and action.

- Actions in the real world aren't necessarily deterministic
  - For a deterministic domain, $T(s,a,s') = 1$ for one next state s' and zero elsewhere

Reward Function

- The reward function maps a state and action to a real number: $R: S \times A \rightarrow \Re$
  - $R(s,a)$

Markov property: The reward only depends on the current state and action.

- If there is no feedback from the environment when the agent carries out an action, this will be zero

Policy

- Given a Markov Decision Process, an agent follows a (deterministic) "policy" π: $S \rightarrow A$
  - $\pi(s)$ is the action the agent will execute in state s

- An optimal policy, π^*, is a policy that maximizes the expected future reward from any state
  - This is what the agent needs to learn

Optimality Criterion

- Suppose the agent, following policy π, visits a state sequence $s_0, s_1, s_2, \ldots$

- We will measure the goodness or utility of this sequence as the *discounted cumulative reward:*

$$U([s_0, s_1, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)$$

Polynomial discount factor

Discounting

- Two reasons to use this criterion:
  - Behavioral
  - Mathematical
- Behavioral: People and animals appear to prefer short term rewards over long term rewards
- Mathematical: Since visit sequences can be infinitely long, if we just add up the rewards, the sum is not well defined

Visit Distribution

- Since actions are stochastic, if we start at some state s_0 and follow π, we will generate many state sequences, each with some probability (product of the transition functions)
  - Call this the visit distribution

Value of a policy

- We define the value of a policy as the *expected utility,* where expectation is with respect to the visit distribution

- Then the optimal policy is the policy that maximizes this expected utility:

$$\pi^* = \arg\max_{\pi} E\left(\sum_t \gamma^t R(s_t, \pi(s_t))\right)$$

Value of a state under a policy

- We define the value of a state s under a policy π as the value of the policy given that we start at s:

$$V^{\pi}(s) = E\left(\sum_{t} \gamma^{t} R(s_t, \pi(s_t)) \mid s_0 = s \right)$$

- This is called the "value function"

Rewriting the value function

- For a Markov Decision Process, we have:

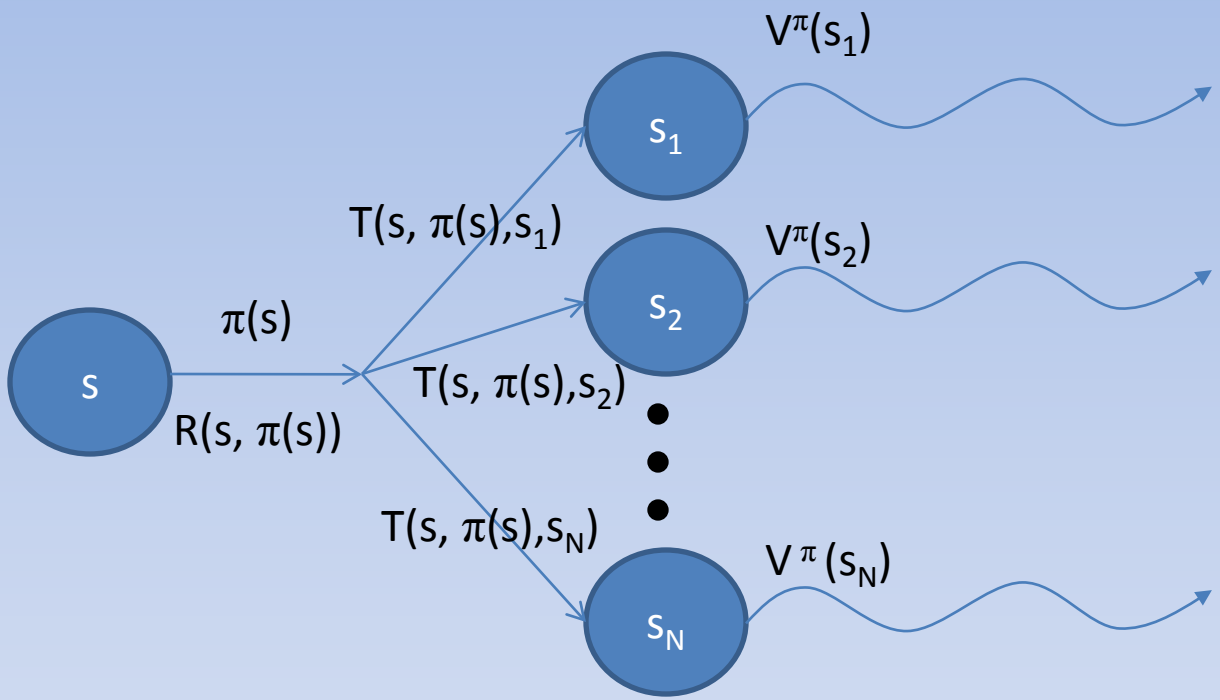
$$V^{\pi}(s) = E\left(\sum_{t=0}^{\infty} \gamma^{t} R(s_t, \pi(s_t)) \mid s_0 = s \right)$$

$$= R(s, \pi(s)) + \gamma E\left(\sum_{t=1}^{\infty} \gamma^{t-1} R(s_t, \pi(s_t)) \right)$$

$$= R(s, \pi(s)) + \gamma \sum_{s'} \Pr(s' \mid s, \pi(s)) \left[E\left(\sum_{t=1}^{\infty} \gamma^{t-1} R(s_t, \pi(s_t)) \mid s_1 = s' \right) \right]$$

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

Picture



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

Bellman equation

Finding the optimal policy

- The optimal policy is the policy with the largest value function:

$$\pi^* = \arg\max_{\pi} V^{\pi}(s) \text{ for all } s$$

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

Bellman Optimality Criterion

- As a consequence of the previous slide, the value function of a state under π^* can be shown to satisfy:

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

Bellman Optimality Criterion (Bellman 1957)
Necessary *and* sufficient!

Finding the optimal policy

- The Bellman optimality criterion gives us a way to find the optimal policy
- If we can find a value function that satisfies it, that determines the optimal policy
- Unfortunately, this system of equations is nonlinear (because of the max), so we can't solve it directly, but a dynamic programming procedure works

Value Iteration

- Start with an arbitrary value function V_0
- Do

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

- Until $|V_{i+1}(s) - V_i(s)|$ is zero

- Then

$$\pi^*(s) = \arg\max_a R(s,a) + \gamma \sum_{s'} T(s,a,s')V_{final}(s')$$

Convergence of value iteration

- It can be shown that at each step of value iteration, the error of the current value function decreases by a factor of (at least) γ

- For small discount factor, convergence is rapid