18734 Recitation

Course Project

MDP

Course Project

- Teams finalized?
- 9 teams (17 students) on the doc.

- Project Idea
- Related readings

Project Proposal

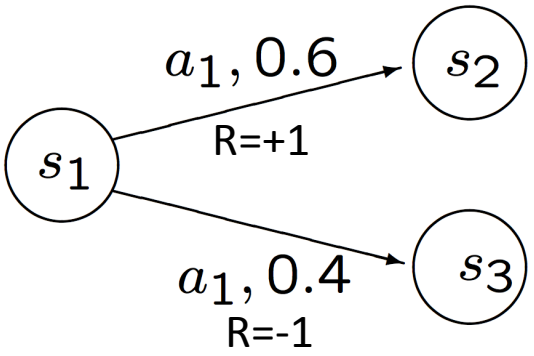
- Pdf document (1-2 pages):
  - Team members
  - Motivation & Problem Statement
  - Approach
  - Deliverables & Timeline

- In-class presentation by members

Markov Decision Process

MDP: Formally

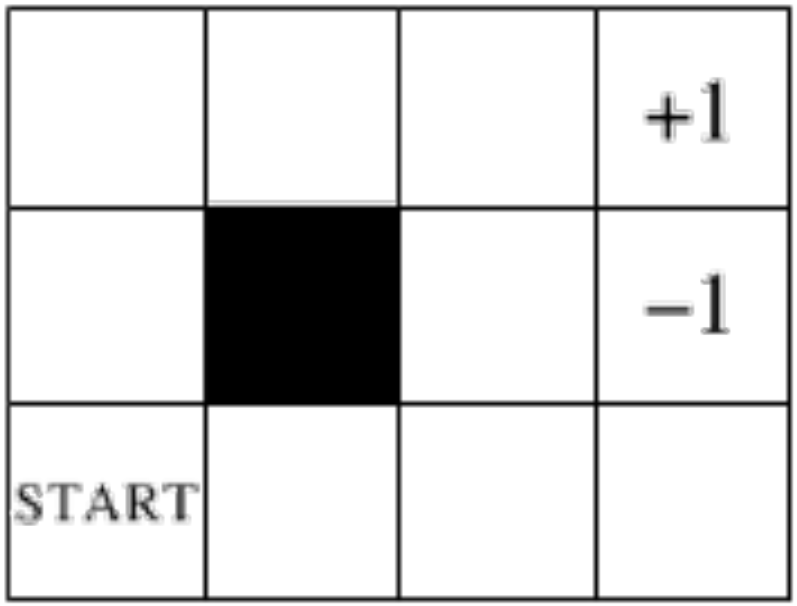
- MDP is a tuple <S, D, A, T, R>, where:
- S is a finite state space
- D is a sequence of discrete time steps/decision epochs (1,2,3, ... , L), L may be ∞
- A is a finite action set
- T: S x A x S -> [0, 1] is a transition function
- R: S x A x S -> Real is a reward function



Finite MDP

Typical Example

- Actions: Left, Right, Up, Down
- We are looking for a good policy/strategy – that maximizes reward.



Policy

- Policy – what action to take?
  - Stationary: $\pi: S \rightarrow A$

- For infinite MDP, there exists an optimal stationary policy (with certain conditions on rewards)

Several notions of what we might want a learned strategy to optimize:

- Expected reward per time step.
- Expected reward in first t steps.
- Expected discounted reward: $r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots$ $(\gamma < 1)$

We will focus on this last one.

Slide: Avrim Blum

Q-values

Goal is to maximize discounted reward:

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots,$$

Define: $Q(s, a) =$ expected discounted reward if perform a from s and then follow optimal policy from then on.

Define: $V(s) = \max_a Q(s, a)$.

Equivalent defn: $V(s) = \max_a [R(s, a) + \gamma \sum_{s'} \Pr(s') V(s')]$.

($R(s, a) =$ expected reward for doing action a in state s.)

Slide: Avrim Blum

How to solve for Q-values?

Suppose we are given the transition and reward functions. How to solve for Q-values? Two natural ways:

1. Dynamic Programming. Start with guesses $V_0(s)$ for all states s. Update using:

Value Iteration $V_i(s) = \max_a \left[\mathbf{E}[R(s,a)] + \gamma \sum_{s'} \Pr(s') V_{i-1}(s') \right]$

Get ϵ-close in $O(\frac{1}{\gamma} \log \frac{1}{\epsilon})$ steps. In fact, if initialize all $V_0(s) = 0$, then $V_t(s)$ represents max discounted reward if game ends in t steps.

2. Linear Programming. Replace "max" with "$\geq$" and minimize $\sum_s V(s)$ subject to these constraints.

Slide: Avrim Blum

Q-learning

Start off with initial guesses $\hat{Q}(s,a)$ for all Q-values. Then update these as you travel. Update rules:

- Deterministic world: in state s, do a, go to s', get reward r:

$$\hat{Q}(s,a) \longleftarrow r + \gamma \max_{a'} \hat{Q}(s',a')$$

- Probabilistic world: on the tth update of $\hat{Q}(s,a)$:

$$\hat{Q}(s,a) \longleftarrow (1-\alpha_t)\hat{Q}(s,a) + \alpha_t[r + \gamma \max_{a'} \hat{Q}(s',a')]$$

Idea: dampen the randomness.

$\alpha_t = 1/t$ or similar. With $\alpha_t = 1/t$, you get a fair average of all the rewards received for doing a in state s. If you make more slowly decreasing, you favor more recent r's.

Slide: Avrim Blum

Policy Iteration

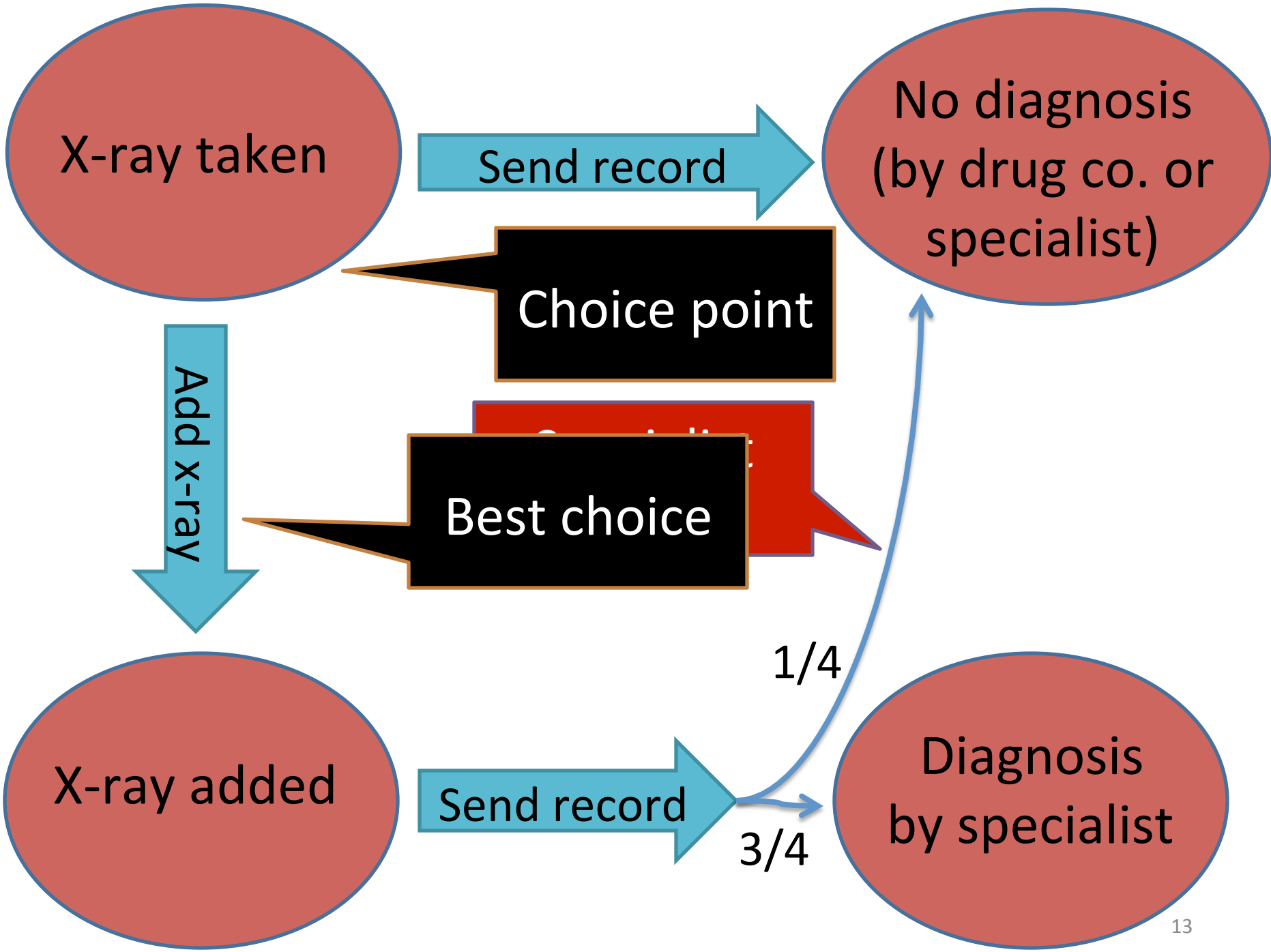
- Two steps

$$\pi(s) := \arg\max_a \left\{ \sum_{s'} P_a(s,s') \left(R_a(s,s') + \gamma V(s') \right) \right\}$$

$$V(s) := \sum_{s'} P_{\pi(s)}(s,s') \left(R_{\pi(s)}(s,s') + \gamma V(s') \right)$$

In policy iteration, step one is performed once, and then step two is repeated until it converges.

Then step one is again performed once and so on.


X-ray taken
Send record
No diagnosis (by drug co. or specialist)
Choice point
Add x-ray
Best choice
1/4
X-ray added
Send record
3/4
Diagnosis by specialist

Was that an infinite MDP?

- How can you make it one?
  - Add a stop action that loops at the end

Audit Algorithm

AUDIT($m = \langle \mathcal{S}, \mathcal{A}, t, r, \gamma \rangle$, $b = [s_1, a_1, \ldots, s_n, a_n]$):

```
01  if (IMPOSSIBLE(m, b))
02      return true      //  behavior impossible for NMDP
03  V*_m := SOLVEMDP(m)
04  for (i := 1; i ≤ n; i++):
05      if (Q*(V*_m, s_i, a_i) < V*_m(s_i)):
06          return true      //  action suboptimal
07      if (Q*(V*_m, s_i, a_i) ≤ 0 and a_i ≠ Stop):
08          return true      //  action redundant
09  return false
```

Figure 2. The algorithm AUDIT