Using Markov Decision Processes for Policy Enforcement

September 15, 2017

Automated enforcement of privacy policies

- Healthcare industry has had numerous high profile privacy incidents
- Government is increasingly cracking down on such breaches
- Privacy is an *ongoing process*
- Problem: Manual audit log reviews are tedious and expensive

Verifying an organization obeys privacy policies

- Privacy policies restrict the purpose for which some protected information can be used for
- Types of restrictions
  - Prohibitive rules
    - An organization does *not* use certain information *for* a purpose
    - Example: Yahoo!'s privacy policy
  - Exclusivity rules
    - An organization users certain information *only for* a given list of purposes
    - Example: HIPAA

Enforcing privacy policies

- Problem:
  - Tools for enforcement assume that actions they audit are accurately labeled with the purpose
  - Tools do not analyze the meaning of *purpose*
- Solution:
  - Need a way to determine if an action could be for a purpose

Purpose in policy enforcement

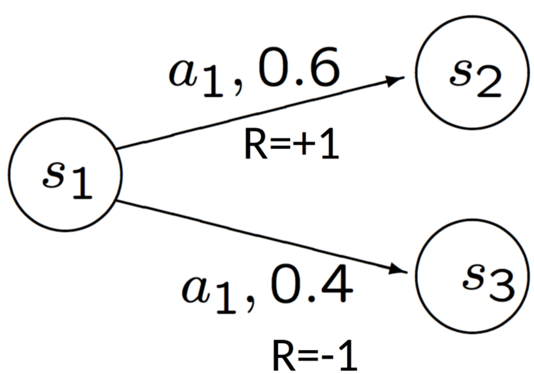
- Planning is central to purpose
  - Cognitive psychology: Purpose is the central determinant of behavior
- Formalizing the relationship between planning and purpose
  - Use MDP to construct all possible behaviors an exclusivity privacy policy allows
  - Auditing an organization:
    - MDP model of allowed behaviors VS
    - Behaviors recorded in the given audit log

Planning for a purpose

- Use an MDP to model the environment the organization being audited operates in
- Why MDP?
  - Good for planning with probabilistic systems
  - Reward function quantifies the degree of progress towards a purpose that taking an action from a state results in
- If the organization being audited logs actions that only help to achieve a particular purpose:
  - Actions correspond to an optimal plan for that MDP
  - Actions are for that purpose

MDP: Formally

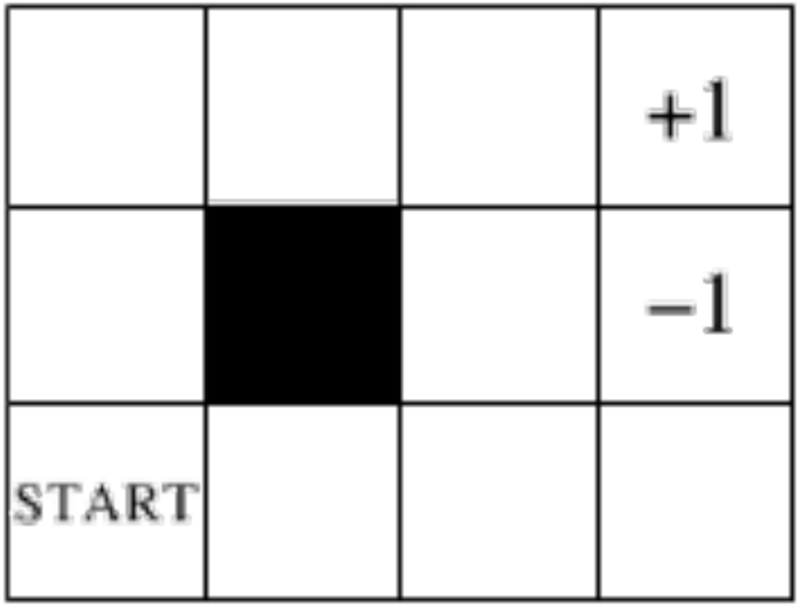
- *MDP* is a tuple <*S*, *A*, *T*, *R*, γ>
  - *S* is a finite state space
  - *A* is a finite action set
  - $T: S \times A \rightarrow D(S)$ is a transition function that given a state and an action, maps to a distribution over states D(S)
  - *R*: $S \times A \rightarrow \mathbb{R}$ is a reward function
  - γ: a discount factor such that $0 < \gamma < 1$



Finite MDP

Example process modeled by MDP

- Actions: Left, Right, Up, Down
- Entering top-right corner gives reward of +1 and then takes you to a random state.
- Finding good strategy that maximizes reward.



Goal of an agent using MDP to plan its actions

- *MDP = <S, A, T, R, >*
- ***Maximize its expected total discounted reward***
  - i is a natural number ranging over time modeled as discrete steps
  - is the reward at time i
  - Expectation taken over probabilistic transitions
  - Discount factor accounts for preference of people to receive rewards sooner, not later

What is a good strategy?

- What we want a strategy to optimize:
  - Expected reward / time step
  - Expected reward in the first t steps
  - *Maximizing the expected discounted reward*

Maximizing the discounted reward

- $$Q(s,a) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots$$

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

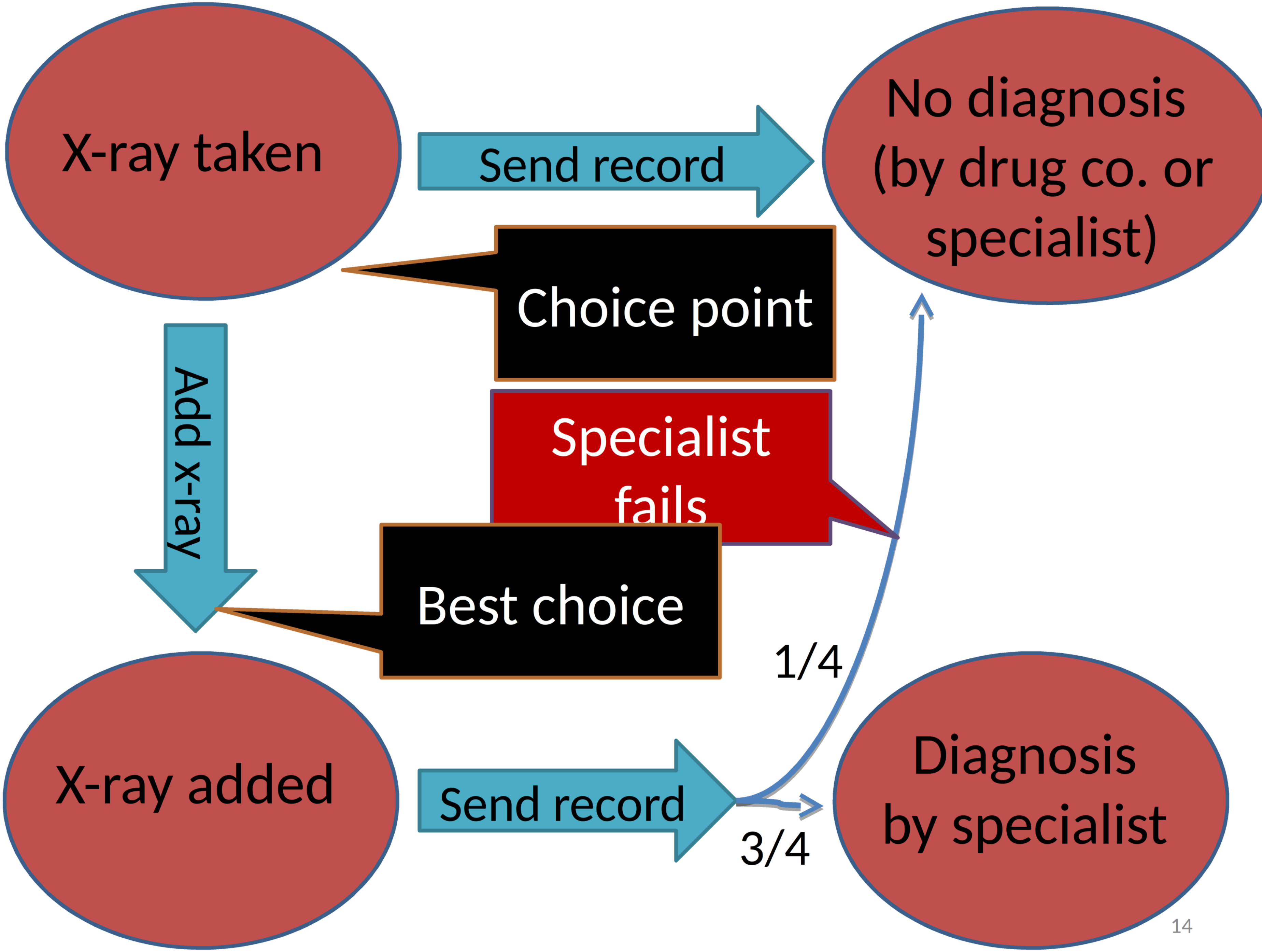
$$V(s) = \max_a [R(s,a) + \gamma \sum_{s'} P(s')V(s')]$$

Solving for the maximum discounted award

- Given:
  - Transition function
  - Reward function
- Possible Solutions
  - Dynamic programming
    - Start with guesses for all states s
    - Update using $$V_i(s) = \max_a \left[\mathbf{E}[R(s,a)] + \gamma \sum_{s'} \Pr(s') V_{i-1}(s') \right]$$
  - Linear programming
    - Replace “max” with “” and minimize subject to the constraints

Infinite MDP

There exists an optimal stationary policy, with certain conditions on rewards.


X-ray taken
Send record
No diagnosis (by drug co. or specialist)
Choice point
Add x-ray
Specialist fails
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

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

if ($\textsc{Impossible}(m, b)$)

    return true // behavior impossible for NMDP

$V^*_m := \textsc{SolveMDP}(m)$

for ($i := 1$; $i \leq n$; i++):

    if ($\mathtt{Q}^*(V^*_m, s_i, a_i) < V^*_m(s_i)$):

        return true // action suboptimal

    if ($\mathtt{Q}^*(V^*_m, s_i, a_i) \leq 0$ and $a_i \neq \mathsf{Stop}$):

        return true // action redundant

return false

Figure 2. The algorithm $\textsc{Audit}$

Questions on the homework?