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 $\infty$
  - $A$ is a finite action set
  - $T: S \times A \times S \rightarrow [0, 1]$ is a transition function
  - $R: S \times A \times S \rightarrow \text{Real}$ is a reward function

![Finite MDP Diagram]

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.
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:

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

Get $\epsilon$-close in $O\left(\frac{1}{\gamma} \log \frac{1}{\epsilon}\right)$ steps. In fact, if initialize all $V_0(s) = 0$, then $V_i(s)$ 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.
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 \( r \):
  \[
  \hat{Q}(s,a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s',a') \\
  \]

- **Probabilistic world**: on the \( t \)th update of \( \hat{Q}(s,a) \):
  \[
  \hat{Q}(s,a) \leftarrow (1 - \alpha_t)\hat{Q}(s,a) + \alpha_t [r + \gamma \max_{a'} \hat{Q}(s',a')] \\
  \]

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 \( \alpha_t \) more slowly decreasing, you favor more recent \( r \)’s.
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.
Was that an infinite MDP?

• How can you make it one?
  – Add a stop action that loops at the end
Audit Algorithm

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

01 \quad \text{if (IMPOSSIBLE}(m, b))
02 \quad \quad \text{return true} \quad \quad \text{// behavior impossible for NMDP}
03 \quad V_m^* := \text{SOLVEMDP}(m)
04 \quad \text{for (i := 1; i \leq n; i++):}
05 \quad \quad \text{if } (Q^*(V_m^*, s_i, a_i) < V_m^*(s_i)):
06 \quad \quad \quad \text{return true} \quad \quad \text{// action suboptimal}
07 \quad \quad \text{if } (Q^*(V_m^*, s_i, a_i) \leq 0 \text{ and } a_i \neq \text{Stop}):
08 \quad \quad \quad \quad \text{return true} \quad \quad \text{// action redundant}
09 \quad \quad \text{return false}

Figure 2. The algorithm AUDIT