3 handouts: slides, asst4, asst3 reference solution
730W blog entries were due
Solving MDPs

Definition
What to do?
Value Iteration
Stopping
Sweeping
Break
Policy Iteration
Policy Evaluation
Summary

Solving MDPs
initial state: $s_0$

transition model: $T(s, a, s') = \text{probability of going from } s \text{ to } s' \text{ after doing } a.$

reward function: $R(s)$ for landing in state $s$.

terminal states: sinks = absorbing states (end the trial).

objective:

**total reward:** reward over (finite) trajectory:  
$R(s_0) + R(s_1) + R(s_2)$

**discounted reward:** penalize future by $\gamma$:  
$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) \ldots$

find:

policy: $\pi(s) = a$

optimal policy: $\pi^*$

proper policy: reaches terminal state
What to do?

\[ \pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s') U^\pi^*(s') \]

\[ U^\pi(s) = E[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi, s_0 = s] \]

The key:

\[ U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s') \]

(Richard Bellman, 1957)
Repeated Bellman updates:

Repeat until happy

\[
U'(s) \leftarrow R(s) + \gamma \max_a \sum' T(s, a, s') U(s')
\]
\[
U \leftarrow U'
\]

For infinite updates, guaranteed to reach equilibrium. Equilibrium is unique solution to Bellman equations!
\[ \| U_{i+1} - U_i \| = \text{max difference between corresponding elts} \]

if \[ \| U_{i+1} - U_i \| < \epsilon (1 - \gamma) / \gamma \] then \[ \| U_{i+1} - U^* \| < \epsilon \]
Stopping

\[ ||U_{i+1} - U_i|| = \text{max difference between corresponding elts} \]

if \[ ||U_{i+1} - U_i|| < \epsilon(1 - \gamma)/\gamma \] then \[ ||U_{i+1} - U^*|| < \epsilon \]

if \[ ||U_{i+1} - U^*|| < \epsilon \] then \[ ||U^{\pi_{i+1}} - U^{\pi*}|| < 2\epsilon\gamma/(1 - \gamma) \]
Stopping

\[ ||U_{i+1} - U_i|| = \max \text{difference between corresponding elts} \]

if \[ ||U_{i+1} - U_i|| < \epsilon \frac{(1 - \gamma)}{\gamma} \] then \[ ||U_{i+1} - U^*|| < \epsilon \]

if \[ ||U_{i+1} - U^*|| < \epsilon \] then \[ ||U_{\pi i+1} - U^{\pi*}|| < 2\epsilon \frac{\gamma}{(1 - \gamma)} \]

loss \[ < \frac{2(\max-update)\gamma}{1 - \gamma} \]

Until \( max-update \leq \text{loss} - \text{bound} \frac{(1-\gamma)}{2\gamma} \)

for each state \( s \)

\[ U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s') \]
Stopping

Solving MDPs
- Definition
- What to do?
- Value Iteration
- Stopping
- Sweeping
- Break
- Policy Iteration
- Policy Evaluation
- Summary

RL

Wheeler Ruml (UNH)
Prioritized Sweeping

- Definition
- What to do?
- Value Iteration
- Stopping

**Sweeping**
- Break
- Policy Iteration
- Policy Evaluation
- Summary

**Solving MDPs**
- Definition
- What to do?
- Value Iteration
- Stopping

**RL**

concentrate updates on states whose value changes!

to update state $s$ with change $\delta$ in $U(s)$:

update $U(s)$

priority of $s \leftarrow 0$

for each predecessor $s'$ of $s$:

  priority $s' \leftarrow \max$ of current and $\max_a \delta \hat{T}(s', as')$
Solving MDPs
■ Definition
■ What to do?
■ Value Iteration
■ Stopping
■ Sweeping
■ Break
■ Policy Iteration
■ Policy Evaluation
■ Summary

asst 3: solution
asst 4: simulators
final projects
office hours by appointment
repeat until $\pi$ doesn’t change:
  given $\pi$, compute $U^{\pi}(s)$ for all states
  given $U$, calculate policy by one-step look-ahead

If $\pi$ doesn’t change, $U$ doesn’t either.
We are at an equilibrium ($\equiv$ optimal $\pi$)!
Policy Evaluation

Solving MDPs
- Definition
- What to do?
- Value Iteration
- Stopping
- Sweeping
- Break
- Policy Iteration
- Policy Evaluation
- Summary

RL

computing $U^\pi(s)$:

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

linear programming ($N^3$) or
Policy Evaluation

computing $U^\pi(s)$:

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

linear programming ($N^3$) or simplified value iteration:

do a few times:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} T(s, \pi(s), s') U(s')$$

(simplified because we are given $\pi$, no max over $a$)
value iteration: compute $U^\pi^*$

policy iteration: compute $U^\pi$ using

- linear algebra
- simplified value iteration
- a few updates (modified PI)
Reinforcement Learning
build a policy based on experience $(s, a, s', r)$

objective:

**finite horizon:** $R(s_0) + R(s_1) + R(s_2)$

**infinite discounted reward:** $R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$
‘model-based’. active vs passive
learn \( T \) and \( R \) as we go, calculating \( U(s) \) using MDP methods

Until \( \text{max-update} \leq \text{loss} - \text{bound} \frac{(1-\gamma)^2}{2\gamma^2} \)

for each state \( s \)

\[
U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')
\]

\[
\pi(s) = \arg\max_a \sum_{s'} T(s, a, s') U(s')
\]

problem:
‘model-based’. active vs passive
learn $T$ and $R$ as we go, calculating $U(s)$ using MDP methods

Until $\text{max-update} \leq \text{loss} - \text{bound} \frac{(1-\gamma)^2}{2\gamma^2}$

for each state $s$

$$U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$

$$\pi(s) = \arg\max_a \sum_{s'} T(s, a, s') U(s')$$

problem: greedy (local minima)
be sure to explore!
Exploration vs Exploitation

Solving MDPs

RL
- Definition
- ADP
- Bandits
- EOLQs

\[ U^+(s) \leftarrow R(s) + \gamma \max_a f \left( \sum_{s'} T(s,a,s')U^+(s'), N(a,s) \right) \]

where \( f(u,n) = R_{\text{max}} \) if \( n < k \), \( u \) otherwise
What question didn’t you get to ask today?
What’s still confusing?
What would you like to hear more about?

Please write down your most pressing question about AI and put it in the box on your way out.

*Thanks!*