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

Solving MDPs

Markov Decision Process (MDP)

initial state: s_0
transition model: $T(s, a, s')$ = probability of going from s to s' 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 γ:
$R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) \ldots$

find:

policy: $\pi(s) = a$
optimal policy: π^*
proper policy: reaches terminal state

What to do?

$$\pi^*(s) = \operatorname*{argmax}_a \sum_{s'} T(s,a,s') U^{\pi^*}(s')$$

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

The key:

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

(Richard Bellman, 1957)

Value Iteration

Repeated Bellman updates:

Repeat until happy
  for each state s
    $U'(s) \leftarrow R(s) + \gamma \max_a \sum_s' T(s, a, s')U(s')$
  $U \leftarrow U'$

For infinite updates, guaranteed to reach equilibrium.
Equilibrium is unique solution to Bellman equations!

Stopping

$||U_{i+1} - U_i|| =$ max difference between corresponding elts

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

Stopping

$||U_{i+1} - U_i||$ = 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 difference between corresponding elts

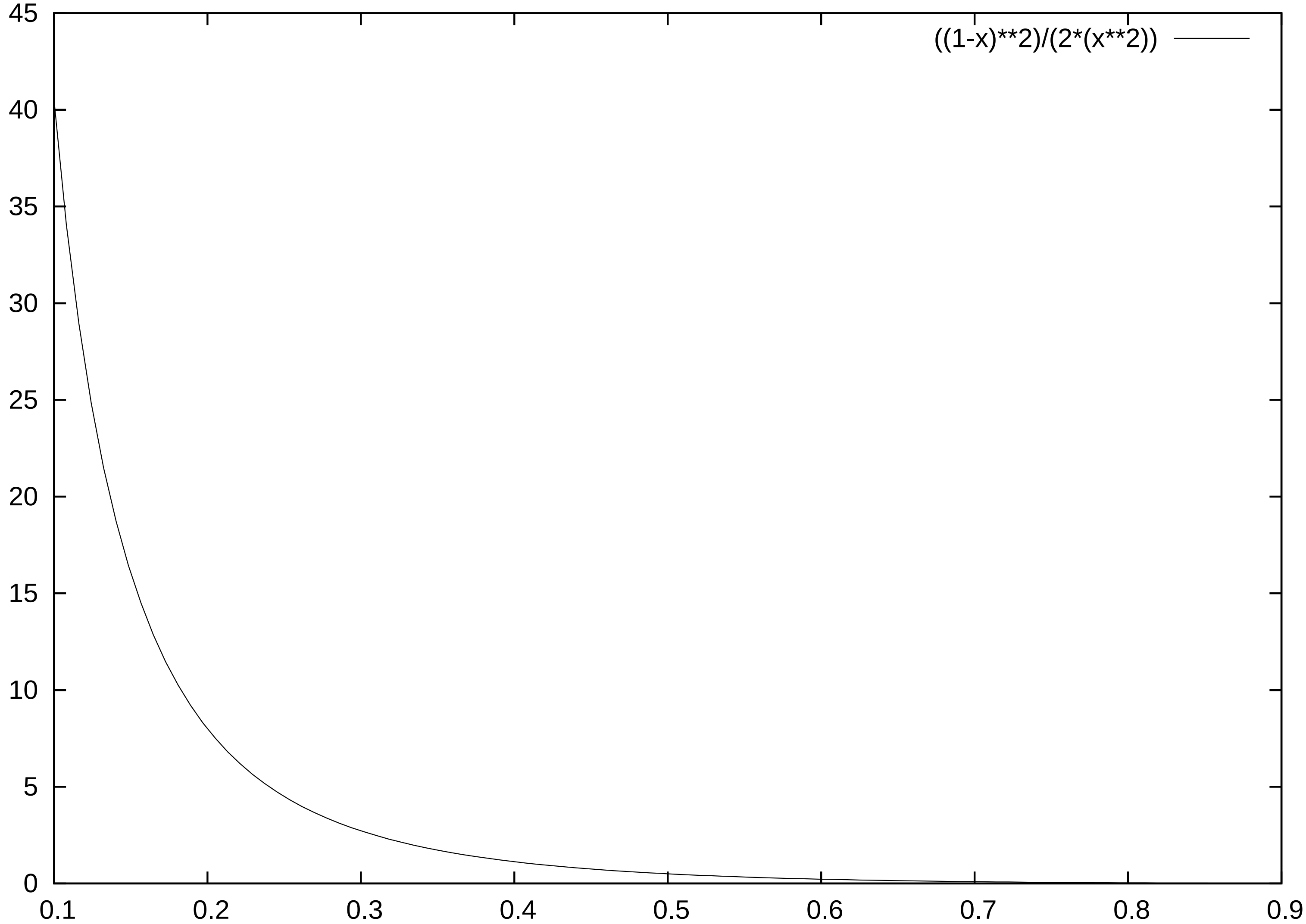
if $||U_{i+1} - U_i|| < \epsilon(1-\gamma)/\gamma$ then $||U_{i+1} - U^*|| < \epsilon$

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

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

Until *max-update* $\leq loss - bound\frac{(1-\gamma)}{2\gamma}$
 for each state s
  $U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$

Stopping



Prioritized Sweeping

concentrate updates on states whose value changes!

to update state s with change δ in $U(s)$:
  update $U(s)$
  priority of $s \leftarrow 0$
  for each predecessor s' of s:
    priority $s' \leftarrow$ max of current and $\max_a \delta\hat{T}(s', as')$

Break

- asst 3: solution
- asst 4: simulators
- final projects
- ofice hours by appointment

Policy Iteration

repeat until π doesn't change:
  given π, compute $U^{\pi}(s)$ for all states
  given U, calculate policy by one-step look-ahead

If π doesn't change, U doesn't either.
We are at an equilibrium ($=$ optimal π)!

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

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 π, no max over a)

Summary of MDP Solving

- value iteration: compute U^{π^*}
- policy iteration: compute U^{π} using
  - linear algebra
  - simplified value iteration
  - a few updates (modified PI)

Reinforcement Learning

Reinforcement Learning (RL)

build a policy based on experience (s, a, s', r)

objective:

finite horzon: $R(s_0) + R(s_1) + R(s_2)$
infinite discounted reward: $R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$

Adaptive Dynamic Programming

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

Until *max-update* $\leq loss - bound\frac{(1-\gamma)^2}{2\gamma^2}$
for each state s
$U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s,a,s')U(s')$

$$\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')U(s')$$

problem:

Adaptive Dynamic Programming

'model-based'. active vs passive

learn T and R as we go, calculating $U(s)$ using MDP methods

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

for each state s

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

$$\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s'} T(s,a,s')U(s')$$

problem: greedy (local minima)

be sure to explore!

Exploration vs Exploitation

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

where $f(u, n) = R_{\max}$ if $n < k$, u otherwise

EOLQs

- 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!