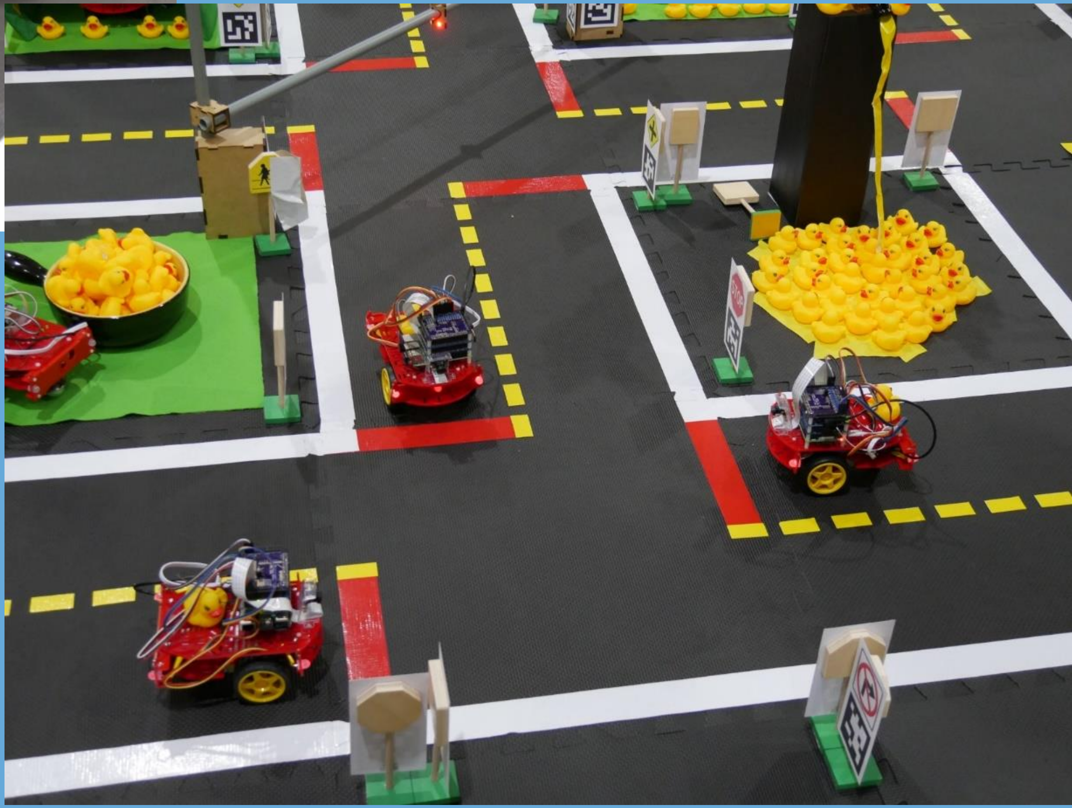
CS 3630 Reinforcement Learning: 2



Markov Decision Processes,
Approaches to
Reinforcement Learning

Policies

$$E[r_h] = E\left[\sum_{i=0}^{h} \gamma^i R(s_i) \mid a_1, \ldots, a_h\right]$$

Def: A policy $\pi : S \to A$ specifies $a = \pi(s)$, the action to take in state s.

For a policy π

$$V^\pi(s) = E[r_\infty(s) \mid \pi]$$

$$= E\left[\sum_{i=0}^{\infty} \gamma^i R(s_i) \mid s_0 = s, \pi\right]$$

$$= E\left[R(s_0) + \sum_{i=1}^{\infty} \gamma^i R(s_i) \mid s_0 = s, \pi\right]$$

$$= R(s) + E\left[\gamma \sum_{i=1}^{\infty} \gamma^{i-1} R(s_i) \mid \pi\right]$$

$i = j+1$
$j = i-1$

$$= R(s) + \gamma \underbrace{E\left[\sum_{j=0}^{\infty} \gamma^j R(s_{j+1}) \mid \pi\right]}_{\text{Expected return under } \pi \text{ from state } s_{j+1}}$$

$$V^\pi(s) = R(s) + \gamma \underbrace{\sum_{s'} T(s, \pi(s), s') V^\pi(s')}_{\text{Expected return for executing } a = \pi(s) \text{ from state } s.}$$

We want the optimal policy, π^*.

$$\pi^* = \arg\max_{\pi} V^\pi(s)$$

Def The value function

$V^* (= V^{\pi^*})$, $V^* : S \to \mathbb{R}$

is the value for π^*.

$$\pi^*(s) = \arg\max_{a \in A} \underbrace{\sum_{s'} T(s, a, s') V^*(s')}_{\text{Expected return for executing action } a \text{ in state } s}$$

Bellman Equation

$$\pi^* = \arg\max_a \sum T(s,a,s')V^*(s')$$

$$V^*(s) = R(s) + \gamma \max_a \sum_{s' \in S} T(s,a,s') V^*(s')$$

Bellman Eqn

How to find V^* ??

Suppose we have an estimate $\underline{V}^k$ of V^* and we want to iteratively improve s.t. $V^k \xrightarrow[k\to\infty]{} V^*$, the unique soln to Bellman eqn.

Value Iteration

$$V^{k+1}(s) = \underbrace{R(s)}_{\text{Truth}} + \underbrace{\gamma \max_a \sum_{s' \in S} T(s,a,s')\underline{\underline{V^k}}(s')}_{\text{what we would have, if } V^k = V^*}$$

Best guess at V^* at k^{th} iteration

If Works

A	+1	.3	-.05
B	+1	.3	0.1
C	+1	.3	.25
D	+1	.3	0.1
E	+1	1.5	1.15
F	+1	.3	0.1
G	+1	.3	.25
H	+1	.3	0.1
I	+1	.3	-.05
J	+1	.3	-.05
K	+1	.3	-.05
L	+1	.3	-.05
	V^0	V^1	V^2

R ↑

L ↓

$\boxed{\gamma = 0.5}$

$$V^2(s) = R(s) + 0.5 \max_{a \in \{R,L\}} T(s,a,s')V^1(s')$$

For A, I, J, K, L

$$V^2(s) = -0.2 + 0.5 \max_{a \in \{R,L\}} \{0.25 \times .3 + 0.5 \times .3 + 0.25 \times .3,\ 0.25 \times 3 + 0.5 \times .3 + 0.25 \times .3\}$$

$$= -0.05$$

$S = B$

$$V^2(B) = -0.2 + 0.5 \max \{\overset{R}{0.3},\ \underbrace{0.25 \times .3 + 0.5 \times 1.5 + 0.25 \times .3}_{0.6}\}$$

$$= -0.2 + 0.5(0.6)$$

$$= 0.1$$

$$\boxed{V^{k+1}(s) = R(s) + \gamma \max_a \sum_{s'} T(s,a,s')V^k(s)^*}$$

$$V^1(s) = R(s) + 0.5 \sum_a T(s,a,s')V^0(s)$$

For $S \neq E$, $R(S) = -.2$

$$V^1(S) = -.2 + 0.5 \max_{R,L} \{0.25 \times 1 + 0.5 \times 1 + 0.25 \times 1,\ 0.25 \times 1 + 0.5 \times 1 \times 0.25 \times 1\}$$

$$= -.2 + .5 \times 1 = 0.3$$

Policy Iteration

Let π^i be approximation to π^*
at i^{th} iteration.

$$\pi^{i+1}(s) = \arg\max_{a \in A} \sum_{s' \in S} T(s, \underbrace{\pi^i(a)}_{\text{current guess for } \pi^*}, s') \underbrace{V^i(s)}_{\text{current guess for } V^*}$$

$$V^{i+1}(s) = R(s) + \gamma \underbrace{\sum_{s' \in S} T(s, \pi^{i+1}(s), s') V^i(s')}_{\text{looks like expected return under } \pi}$$

Reinforcement Learning

we have done a lot of work:

→ MDP = (S, A, T, R, γ)

→ Bellman Eqn + Value Function

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

→ Optimal Policy

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

For Machine Learning

* We don't know T.
  ↳ world model

* We don't know R(s)

R.L. Two Approaches

Passive: The agent has a fixed policy π, and it can learn about T and/or V^π by executing π.

Active: The policy is not fixed. The agent learns about V^* and/or T, while learning to act optimally.

Direct Utility Estimation

$$V^{\pi}(s) = E\left[\sum_{i=0}^{\infty} \gamma^{i} R(s_i) \,\middle|\, \pi, s_0 = s \right]$$

Idea execute π from s many times. The average of the returns is a good approximation for $V^{\pi}(s)$.

Can't execute ∞ actions, so execute over a finite horizon, length h.

Error in approximation

$$r_{\infty}(s_0, \cdots) = \sum_{i=0}^{\infty} \gamma^{i} R(s_i)$$

$$= \sum_{i=0}^{h} \gamma^{i} R(s_i) + \sum_{i=h+1}^{\infty} \gamma^{i} R(s_i)$$

Approximation of $V^{\pi}(s)$

$$\text{Error} = \sum_{i=h+1}^{\infty} \gamma^{i} R(s_i) \leq \sum_{i=h+1}^{\infty} \gamma^{i} R_{max} = \sum_{j=0}^{\infty} \gamma^{j+h+1} R_{max} = \gamma^{h+1} \sum_{j=0}^{\infty} \gamma^{j} R_{max}$$

*

$$= \frac{\gamma^{h+1} R_{max}}{1-\gamma} \Leftarrow \text{Bound on the error}$$

Adaptive Dynamic Programming (ADP)

Recall:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T(s, \pi(s), s')\, V^{\pi}(s') \qquad (*)$$

Idea at each stage of execution, execute $a = \pi(s)$ in state s and arrive to state s' $\Longrightarrow$ s, a, s', we observe $R(s')$

At each stage

new estimate of T

- For each $\bar{s}$, update $\hat{T}(s, a, \bar{s}) = \dfrac{N_{sa\bar{s}}}{N_{sa}}$

- Update $\underbrace{\hat{V}^{\pi}(s)}_{\text{new estimate for } V^{\pi}(s)} \leftarrow R(s) + \gamma \sum_{s'} \hat{T}(s, a, s')\, \underbrace{\hat{V}^{\pi}(s')}_{\text{prev. estimate of } V^{\pi}}$

$N_{sa\bar{s}}$ = # of times we experience $s, a, \bar{s}$

N_{sa} = # of times we execute action a from state s

↳ Keep in a table

USE D.P. to implement this

Temporal Difference Learning - (TD Learning) ← Model-Free Approach.

Execute a in state s ~~t~~ and reach s'. $\Rightarrow s, a, s'$

$$\cancel{R(s) + \gamma \pi'} \quad R(s) + \gamma V^{\pi}(s') \rightsquigarrow \text{after experience } s, a, s'$$

Before experiencing s, a, s', the best guess for return is $V^{\pi}(s)$

$$\underbrace{[R(s) + \gamma V^{\pi}(s')] - V^{\pi}(s)}_{\text{Temporal Difference.}}$$

$\boxed{\text{No Model of } T}$

Updating Scheme:

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha \left[R(s) + \gamma V^{\pi}(s) - V^{\pi}(s)\right]$$

TD equation

Learning rate

ADP vs. TD

ADP: V^{π} is made to agree with all past experience.

TD: V^{π} is made to agree only with current experience.

Active Learning –

Approximate

- If we have a model, T, we can use T to build an approximation of V^*, and thus π^*. [e.g., using Value or Policy Iteration]

Q Given $\hat{T} \Rightarrow \hat{V}^*, \hat{\pi}^*$ should we

a. execute $\hat{\pi}^*(s)$ ← Exploitation

b. Try some other action, a, ← Exploration
maybe finding better $\hat{V}^*$.

→ Key problem: Balancing Exploration vs. Exploitation

Many available Algorithms to do this---

Q-Learning: For active learning, we need to explicitly consider the action a when deciding what to do.

> $V^*(s)$ does not explicitly consider the action to be performed.

- $V^*(s) = R(s) + \gamma \underbrace{\max_a}_{\text{action appears.}} \sum_{s'} T(s,a,s') V^*(s')$

$$= \max_a \underbrace{\left[R(s) + \gamma \sum_{s'} T(s,a,s') V^*(s') \right]}_{Q(a,s) \,=\, Q \text{ function}}$$

- $V^*(s) = \max_a Q(a,s)$

$$Q(a,s) = R(s) + \gamma \sum T(s,a,s') \overbrace{V^*(s')}$$

constraint eqn for Q function

$$\boxed{Q(a,s) = R(s) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q(a',s')}$$

TD Q-Learning

Given Q at some stage of learning.
We execute action a in state s, arrive to s'

$\Rightarrow$ s, a, s'

$$Q(a,s) \leftarrow \underbrace{Q(a,s)}_{\substack{\text{current} \\ \text{estimate} \\ \text{of } Q}} + \alpha \Big[\underbrace{R(s) + \gamma \max_{a'} Q(a',s')}_{\substack{\text{estimate of } Q \\ \text{based on having} \\ \text{seen } s,a,s'}} - \underbrace{Q(a,s)}_{\substack{\text{estimate of } Q \\ \text{before seeing} \\ s,a,s'}} \Big]$$

Learning rate.

Note: We never build a model for T $\Rightarrow$ Model-Free.

END