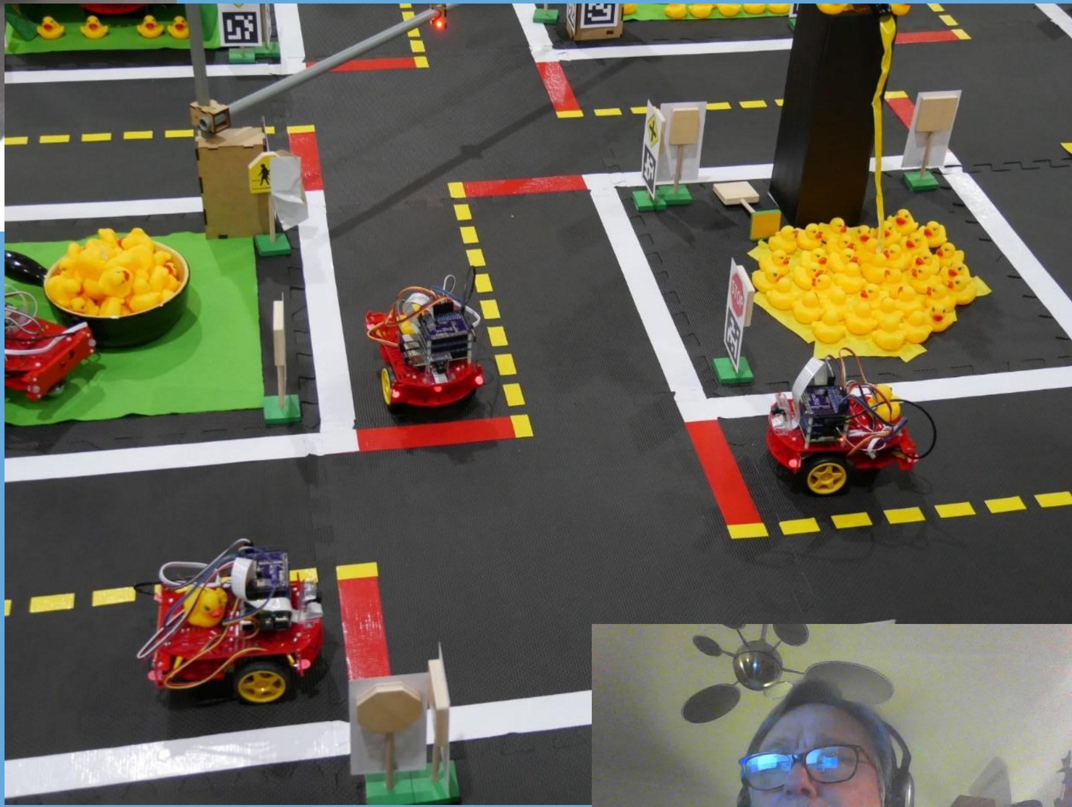
CS 3630
Reinforcement Learning



Reinforcement Learning

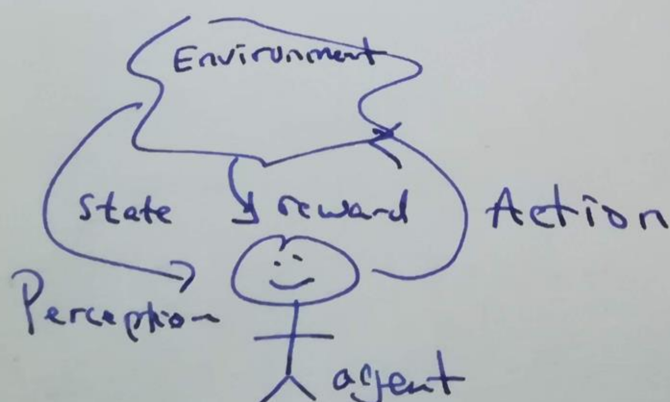
"Typical" Machine Learning

- Lots of Data
- passive Learner
- Learning = pattern analysis
  function approx.

Examples

* Find photos of cats.
* product recommendation
* Text Translation

Robots are Not Like This!



RL:

RL is a process of modifying behavior by rewarding desired outcomes.

- Dogs – Treats, Learn tricks
  – punishment → negative reward
- Kids – 1$/A on report
- students: grades
  joy, satisfaction
  starting selary

1. Robot senses world
2. Robot decides & executes action
3. World state changes
4. Robot Receives a reward. Robot senses world
5. Robot update its "strategy"
6. Go to 2

Questions:

- Mathematical Model
  - states
  - actions
  - uncertainty (!)
- Mathematical formalism for reward
- Given the above How to compute an Optimal strategy/policy
- Now... suppose we don't know the models for world, actions... How can we learn them?

- Markov Process
- Markov Decision Process
- Value function & how to compute it
- R.L.

Markov Processes

Simplest case:

- discrete time, $k = 0, 1, 2, \dots$
- discrete states, S set of states.
- A set of transition probabilities

$$T : S \times S \to \underbrace{[0,1]}_{\text{probability}}$$

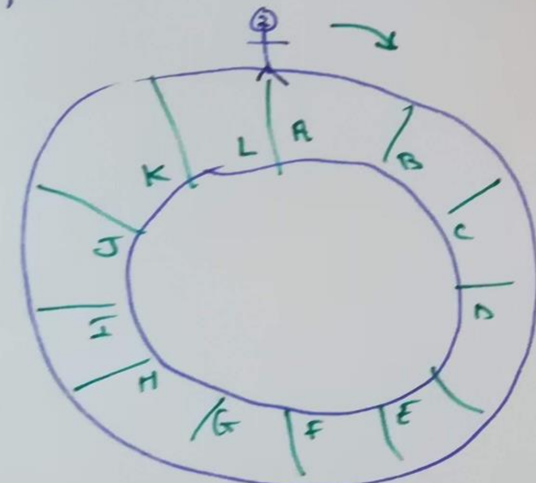
$$P\{s_{k+1} \mid s_k\} = T(s, s') = T(s_k, s_{k+1})$$

$T(s, s') = \text{prob}\{$arriving to state s', given we are now in state $s\}$

$\hookrightarrow$ M.P. evolves "autonomously"

Example:

Sisyphus has the job of, every day, throwing a large rock, on a circular track.



L_k = distance of throw on k^{th} day

$$P\{L_k = 1\} = 0.25$$

$$P\{L_k = 2\} = 0.5$$

$$P\{L_k = 3\} = 0.25$$

$$S = \{A, B, \cdots L\}$$

$T(A,B) = 0.25$ $\quad T(A,C) = 0.5$ $\quad T(A,D) = 0.25$

$\vdots$

$T(J,K) = 0.25$ $\quad T(J,L) = 0.5$ $\quad T(J,A) = 0.25$

$T(K,L) = 0.25$ $\quad T(K,A) = 0.5$ $\quad T(K,B) = 0.25$

$T(s,s') = 0$ otherwise

Properties:

- Stationary: T does not change over time

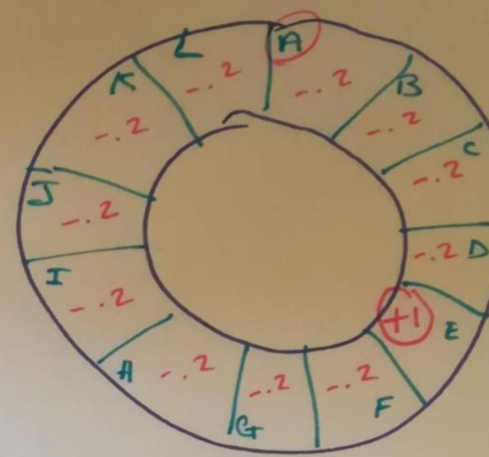
- $\boxed{P\{S_{k+1} \mid S_0, S_1, \cdots S_k\} = P\{S_{k+1} \mid S_k\}}$

Markov Property

$$P\{S_3 = E \mid S_0 = A, S_1 = C, S_2 = D\} = P\{S_3 = E \mid S_2 = D\}$$

$$P\{S_3 = E \mid S_0 = A, S_1 = B, S_2 = D\} =$$

Markov Decision Processes (MDPs)



Markov Processes evolve <u>autonomously</u>.

Let's give Sisyphus a tiny bit of free will:

R : Throw rock counterclockwise

L : Throw rock clockwise

Same throwing abilities for L + R

~~P{L_k}~~ $P\{L_k\}$ unchanged

$\Rightarrow T(A, \underline{L}, B) = 0.25 \quad T(A, \underline{L}, C) = 0.5 \quad T(A, \underline{L}, D) = 0.25$

$\hookrightarrow T(s, a, s') = P\{s' \mid s, a\}$

$T : S \times A \times S \rightarrow [0,1]$

$\uparrow$ set of possible actions.

⋮ L = action

$\Rightarrow T(A, R, L) = 0.25 \quad T(A, R, K) = 0.5 \quad T(A, R, J) = 0.25$

⋮

<u>Rewards</u> : $R : S \rightarrow \mathbb{R}$

For Sisyphus: $R(E) = +1$

$R(s) = -0.2$ for $s \neq E$

Suppose Sisyphus executes two actions

$a_1 = L, \quad a_2 = L$

$k=1 \qquad k=2$

Expectation

Suppose a random variable X takes values from the set $\{c_1, c_2, \cdots, c_n\}$. The expected value of X is defined as

$$E[X] = \sum_{i=1}^{n} P\{X = c_i\} \times c_i$$

Example: roll one die, X shows

$$E[X] = \underbrace{\sum_{i=1}^{6} \frac{1}{6}}_{\text{all values equally likely.}} \times i = \underline{3.5}$$

Intuition: Perform this many times

$$\text{Average of } X\text{'s} \longrightarrow E[X]$$

Generalization

$$E[X_1 + X_2] = \sum_i \sum_j P\{\underline{X_1 = c_i, X_2 = c_j}\}(c_i + c_j)$$

Two dice

$$E[X_1 + X_2] = \sum \sum \frac{1}{36}(i+j) = 7$$

$$E[\sum X_i] = \sum P\{X_0 = c\} \sum (c_i)$$

For Sisyphus: $E[R(s_0) + R(s_1) + R(s_2)]$ given $a_1 = L$, $a_2 = L$??

We know

$$\begin{cases} R(s_0) = -0.2 \text{ because } S_0 = A \\ R(s_1) = -0.2,\ S_1 \in \{B, C, D\} \\ R(s_2) = \begin{cases} +1 & S_2 = E \\ -0.2 & \text{Else} \end{cases} \end{cases}$$

Expected return

Define return $r_h = \sum_{i=0}^{h} R(s_i)$. ← $*$

r_2 for sisyphus $= R(s_0) + R(s_1) + R(s_2)$

$E[r_2] = E[R(s_0) + R(s_1) + R(s_2)]$

$= \overset{P\{S_3 = E\}}{P\{r_h = .6\}} \times .6 + \overset{P\{S_3 \neq E\}}{P\{r_h = -.6\}} \times -.6$ $*$

because r_h has only two possible values

$-0.2 + -0.2 + -0.2 = -0.6 \Leftarrow S_2 \neq E$

$-0.2 + -0.2 + 1 = 0.6 \Leftarrow S_2 = E$

To arrive $S_2 = E$

S_0	S_1	S_2	Probability
A	B	E	$(0.25) \times (0.25) = 0.0625$
A	C	E	$(0.5) \times (0.5) = 0.25$
A	D	E	$(0.25) \times (0.25) = 0.0625$

~~0.0312~~

$P(S_3 = E) = 0.375$

$P\{S_3 \neq E\} = 1 - P\{S_3 = E\}$
$= 0.625$

This is great for finite horizons (i.e. only consider a finite # of stages).

Suppose $h \to \infty$??

To deal with this, use <u>discounted rewards</u>.

$r_h = \sum_{i=0}^{h} \gamma^i R(s_i)$ ← discount factor

for $0 < \gamma < 1$.

$\lim_{h \to \infty} \sum_{i=0}^{h} \gamma^i R(s_i) \leq \sum_{i=0}^{\infty} \gamma^i R_{max}$

$\sum_{i=0}^{\infty} \gamma^i = \frac{1}{1-\gamma}$ $\quad 0 < \gamma < 1$

$\sum_{i=0}^{\infty} \gamma^i R(s_i) \leq \frac{R_{max}}{1-\gamma}$

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

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

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

Bellman Eqn

How to find V^* ??

Suppose we have an estimate 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

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

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

If Works

	V^0	V^1	V^2
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

R ↑

L ↓

$\gamma = 0.5$

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

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

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

<u>Idea</u> execute π from s <u>many</u> 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.

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

Error in approximation

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

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

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

*

$$= \frac{\gamma^{h+1} R_{max}}{1-\gamma} \Leftarrow$$

Bound on the error

Adaptive Dynamic Programming (ADP)

Recall:

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

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 $\underline{a}$ in state $\underline{s}$ and reach $\underline{s'}$. $\Rightarrow$ s, a, s'

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

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

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

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---

<u>Q-Learning</u>: 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.

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

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

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

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

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'

$$\Rightarrow \underline{\underline{s, a, s'}}$$

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

Learning rate.

<u>Note</u> We never build a model for T $\Rightarrow$ Model-Free.

<u>END</u>