Reinforcement Learning
Introduction to Reinforcement Learning

→ Subdomain of ML
→ Train models (Agents in RL)
→ Sequential decisions.

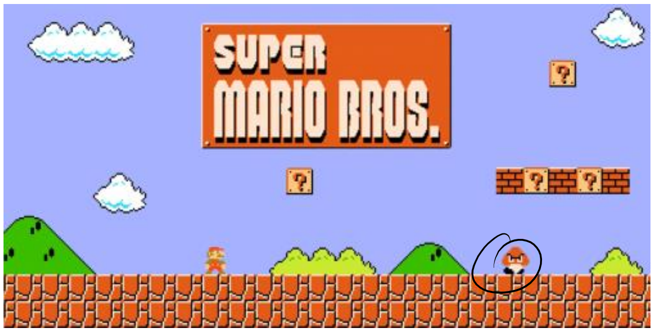
States in RL

Self driving tech.

Waymo
↓
Alphabet

techniques from RL

Super Mario



Actions in RL

Alpha Go
Alpha Star

Go

Star craft

Supervised

Supervised

$X \rightarrow Y$

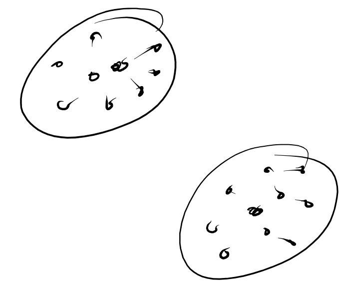
features — target variable.

model.

$x' \rightarrow y$

Unsupervised

X



Reinforcement

Situations → Actions

States → Actions

What to do??

Interacting

Learning from interaction.

General discussion on RL → MDP → (i) Tabular
→ DP
→ MC
→ TD
(ii) Approx.

→ DRL → Papers.

(c) App.

→ Quizzes
→ Coding.

Capstone project.

→ ML
→ Stats/prob
→ Python

"Reinforcement Learning : An introduction"
→ Sutton & Barto.

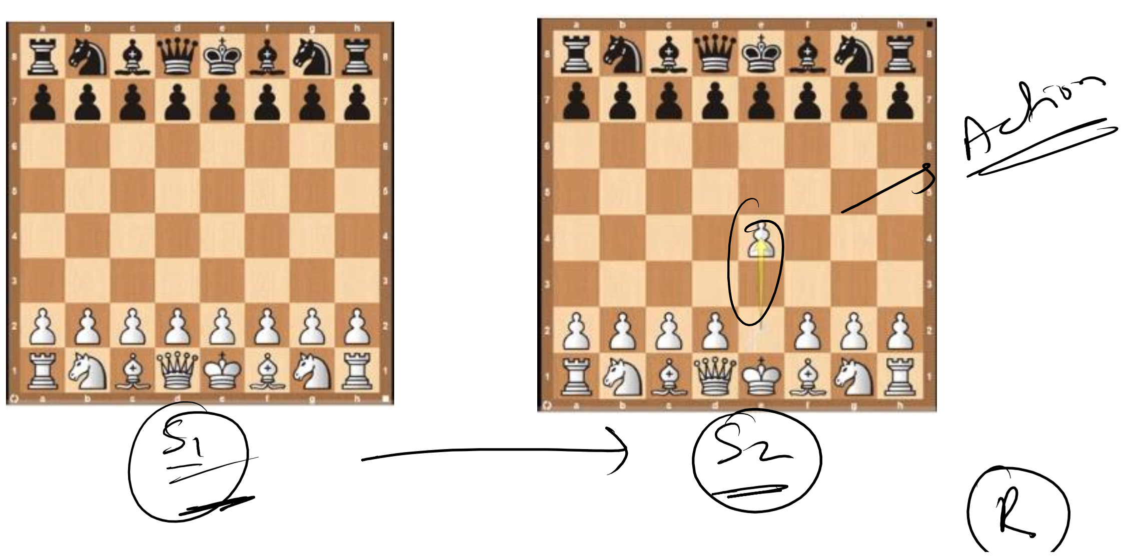
Reinforcement Learning
Building blocks Reinforcement Learning

→ Agent → "the model"

→ Environment & States → Snapshot of env.

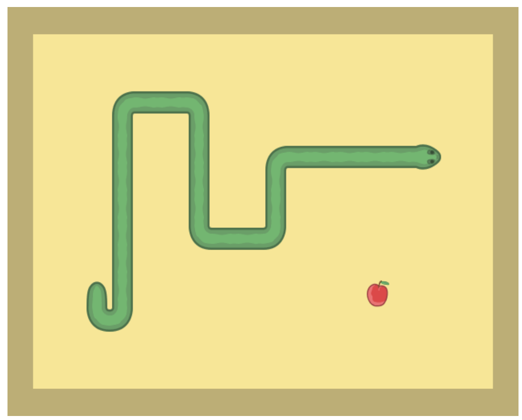
↳ Representation of the problem.

→ Actions → "moves"



R

→ **Reward**: → "numerical feedbacks"



+1

+3

maximize the total reward.

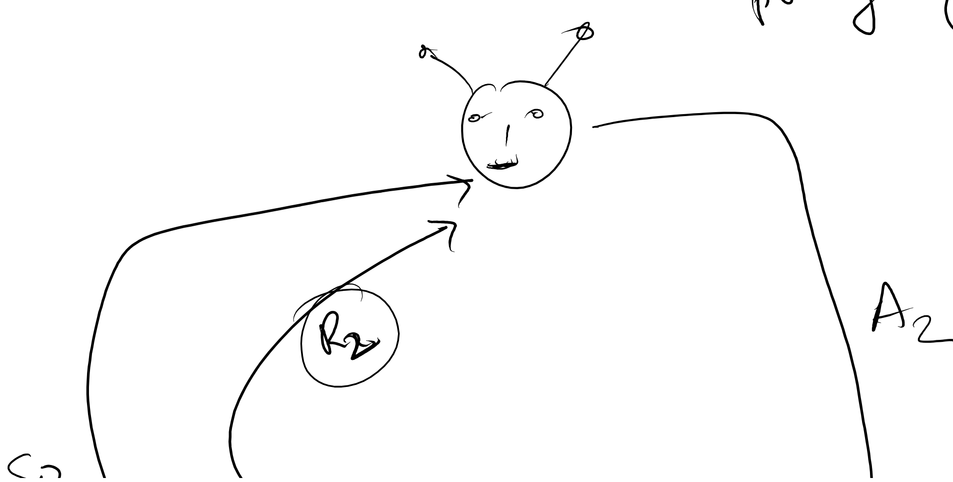
→ **Value function**: → State

→ Action.

↑ Rewards

→ **Policy**: – "how to behave?"

policy → π



So
S2
Env.

Reinforcement Learning
Building blocks Reinforcement Learning - Examples

Env. , agent, state, action, values, policies.

→ Self driving car in a simulation

Env :- Simulator
State :- Front, rear, side
speed, accel, road angle.

policy :- well trained agent

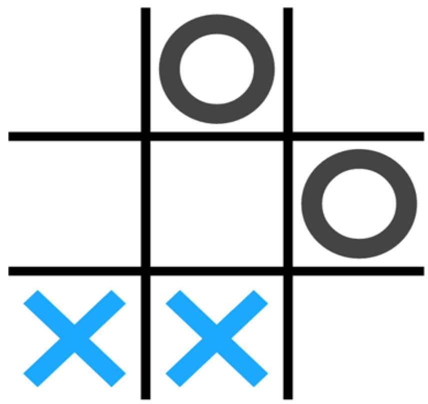
Reward :-
-1 each passing timestep
-5 go off the lane
-10 jump off the signal
-100 hit someone

→ Tic-tac-toe

:- Game board

→ Tic-tac-toe

Optimal policy.



Env :- Game board along with the opp. playing (X)

State :- X's or O's on board.

Reward :- +1 → win
0 → loose

Action :- placing O.

Policy → Random.

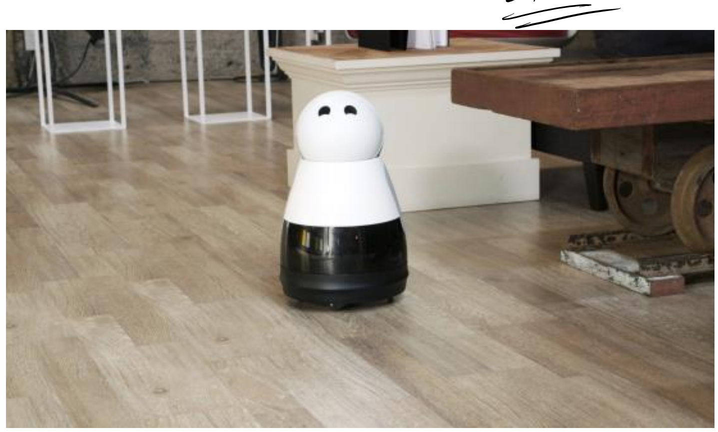
→ Mobile rag picking robot.

Actions :- Go ahead, return back to charging station.

Policy :- Go ahead if battery > 20%

Env :- The complete floor.

State :- front, side, rear, current battery %, space in rag bag.

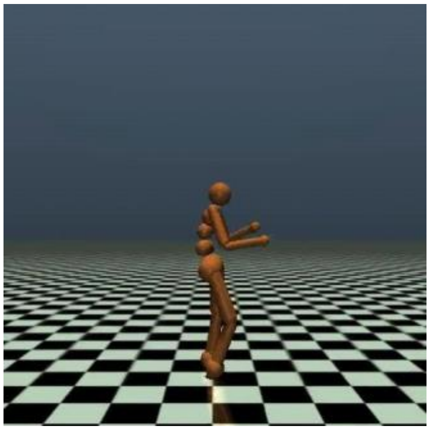


Reward :-
+1 for rag collected.
-100 : stuck.

→ Robot learn to walk

Envi :- Simulator.

State :- Pos. of joint, ang. vel.
of joints, joint position
x, y vel.



Reward :- +1 → forward
-100 → falling.

Actions :- Torque of
joints.

policy :- optimal.

Reinforcement Learning
Introduction to the K-Armed Bandit Problem

$X \rightarrow N(\mu, \sigma^2)$

$$P(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

(σ^2, μ)

k=5

μ_1, σ_1^2 μ_2, σ_2^2 μ_3, σ_3^2 μ_4, σ_4^2 μ_5, σ_5^2

μ_1^* μ_2^*

Solving K-armed bandit - Greedy

Monday, January 13, 2020 9:27 PM

Reinforcement Learning

Solving K-armed bandit - Greedy method

K actions

Action values

μ, σ^2

true value of action

$q_*(a)$

t, A_t, R_t

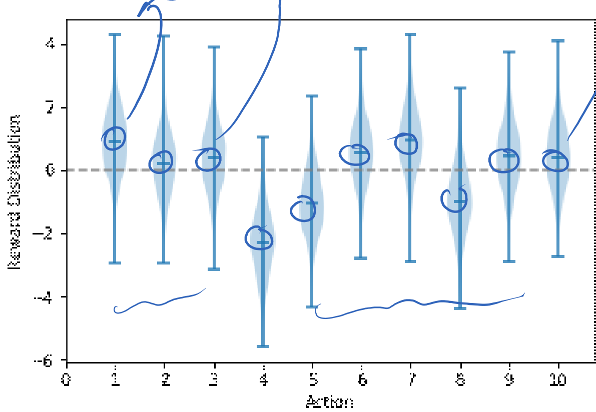
$$q_*(a) = \mathbb{E}[\,R_t \mid A_t = a\,] = \mu_a$$

$q_*(a) = E[R_t \mid A_t = a]$

$q_*(a)$ $E[R \mid a]$

$Q_t(a) \approx q_*(a)$

$q_*(1)$ $q_*(3)$ $q_*(10)$



$K=10$

μ

$\sigma^2 = 1$

Calculating action value estimates

Calculating action value estimates

$K = 3$
Randomly.
A, B, C

t	$Q_t(A)$	$Q_t(B)$	$Q_t(C)$	A	B	C
1	0	0	0	2		
2	2	0	0		1	
3	2	1	0	1		
4	1.5	1	0			3
5	1.5	1	3			-1
6	1.5	1	1			

$$\frac{2+1}{2}$$

$$Q_t(a) = \frac{\sum \text{Reward, choosing 'a', } t-1}{\text{no. of times 'a' was chosen before 't'}}$$

A, B, C
Greedily.

$$\frac{2+(-5)}{2} \quad 2+5$$

t	$Q_t(A)$	$Q_t(B)$	$Q_t(C)$	A	B	C
1	0	0	0	-1		
2	-1	0	0		2	
3	-1	2	0		-5	
4	-1	-1.5	0			1
5	-1	-1.5	1			1
6	-1	-1.5	1			

Calculating action value estimates incrementally

action 'a'

$(n+1) \rightarrow \quad R_1, \quad R_2, R_3, R_4 \; --- \; R_n$

$$Q_{n+1} = \frac{R_1 + R_2 + --- R_n}{n}$$

$$= \frac{1}{n} \sum_{i=1}^{n} R_i$$

$$\sum^{n-1} R_i \Big]$$

$$= \frac{1}{n} \sum_{i=1} R_i$$

$$= \frac{1}{n}\left[R_n + \sum_{i=1}^{n-1} R_i\right]$$

$$= \frac{1}{n}\left[R_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} R_i\right] \rightarrow Q_n = \frac{1}{n-1}\sum_{i=1}^{n-1} R_i$$

$$= \frac{1}{n}\left[R_n + (n-1) Q_n\right]$$

$$= \frac{1}{n}\left[R_n + nQ_n - Q_n\right]$$

$$\Rightarrow Q_{n+1} = Q_n + \frac{1}{n}\left[R_n - Q_n\right] \quad (0,1]$$

$$\Rightarrow \text{New Estimate} = \text{Old Estimate} + \text{Step Size}\left[\text{Target} - \text{Old Estimate}\right]$$

0.5 [0,1]

O N T

<u>**Greedy Algorithm for K-Armed bandit**</u>

No of Bandits.

$Q(a|s)$

> *Initialize, for* $a = 1$ *to* k:
> $Q(a) \leftarrow 0$
> $N(a) \leftarrow 0$
>
> *Repeat forever*:
> $A \leftarrow argmax_a Q(a)$
> $R \leftarrow bandit(A)$
> $N(A) \leftarrow N(A) + 1$
> $Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\,[R - Q(A)]$

Solving K-armed bandit - Epsilon Greedy

Monday, January 13, 2020 9:27 PM

Reinforcement Learning
Solving K-armed bandit - Epsilon Greedy method

Exploration and exploitation

- Exploiting → Best use of current knowledge
- Explore → Non-optimal move

Exploit ⟷ Explore

$\epsilon = 5\%$

$(1-\epsilon) = 95\%$

Epsilon Greedy Algorithm

$Initialize, for\ a = 1\ to\ k$:

$\quad Q(a) \leftarrow 0$

$\quad N(a) \leftarrow 0$

$Initialize\ \epsilon$

$Repeat\ forever$:

$$A \leftarrow \begin{cases} \mathrm{argmax}_a Q(a) & \text{with probability } (1-\epsilon) \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

$\quad R \leftarrow bandit(A)$

$\quad N(A) \leftarrow N(A) + 1$

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$$

$[0,1] = e$

$e \leq \epsilon$

$e > \epsilon$

Decaying - Epsilon Greedy Algorithm

$Initialize, for\ a = 1\ to\ k$:

$\quad Q(a) \leftarrow 0$

$\quad N(a) \leftarrow 0$

$Initialize\ \epsilon$

$Repeat\ forever$:

$$A \leftarrow \begin{cases} \mathrm{argmax}_a Q(a) & \text{with probability } (1-\epsilon) \\ \text{a random action} & \text{with probability } \epsilon \end{cases}$$

$\quad R \leftarrow bandit(A)$

$\quad N(A) \leftarrow N(A) + 1$

$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}\left[R - Q(A)\right]$$

$\quad Decrease\ value\ of\ \epsilon$

Reinforcement Learning
Solving K-armed bandit - Upper Confidence Bounds

Upper Confidence Bounds:

$$\underbrace{Q_t(a)}_{} + \underbrace{U_t(a)}_{\text{uncertainty}} \geq q_*(a)$$

UCB1

$$U_t(a) = \sqrt{\frac{2 \ln t}{N_t(a)}}$$

$t \rightarrow$ total time elapsed.
$N_t(a) \rightarrow$ No. of times action 'a' is taken.

$N_t(a) \uparrow \longrightarrow U_t(a) \downarrow$

$t \uparrow \longrightarrow U_t(a) \uparrow$

$U_t(a) \downarrow$

$Q_t(a)$

$U_t(a) \uparrow$

UCB1 Algorithm:

$Initialize, for\ a = 1\ to\ k:$

$Q(a) \leftarrow 0$

$N(a) \leftarrow 0$

$Q_t(a)$

$U_t(a)$

$Repeat\ forever:$

$$A \leftarrow argmax_a[\,Q(a) + \sqrt{\frac{2 \ln t}{N(a)}}\,]$$

$R \leftarrow bandit(A)$

$N(A) \leftarrow N(A) + 1$

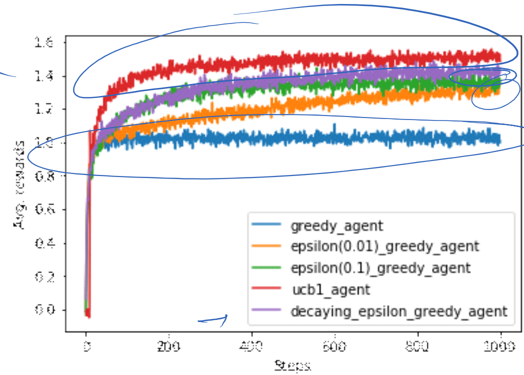
$$Q(A) \leftarrow Q(A) + \frac{1}{N(A)}[R - Q(A)]$$

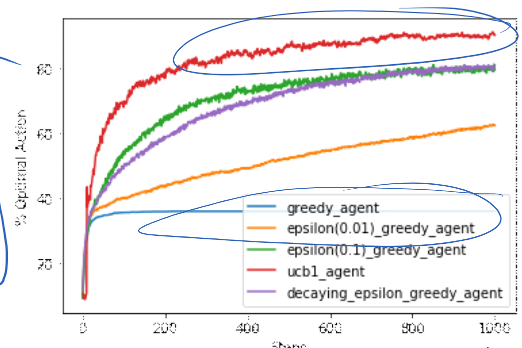
Performance comparison:

2000 → K=10

K=10

K=10





1000