Introduction to Reinforcement Learning (RL)

AppliedAICourse.com

Agenda:

- Applications
- Problem Setup
- Terminology
- Bellman Equation
- Markov Decision Process
- Q-learning
- Temporal-Difference
- Deep Q-Learning [intuition]

Assumptions:

- Comfortable with math in ML & DL

Applications:

- Robotics
- Game-play
- Trading Systems
- Self-driving Cars

④ Modern RL = Deep RL

⇓

DL + RL

Problem Formulation:

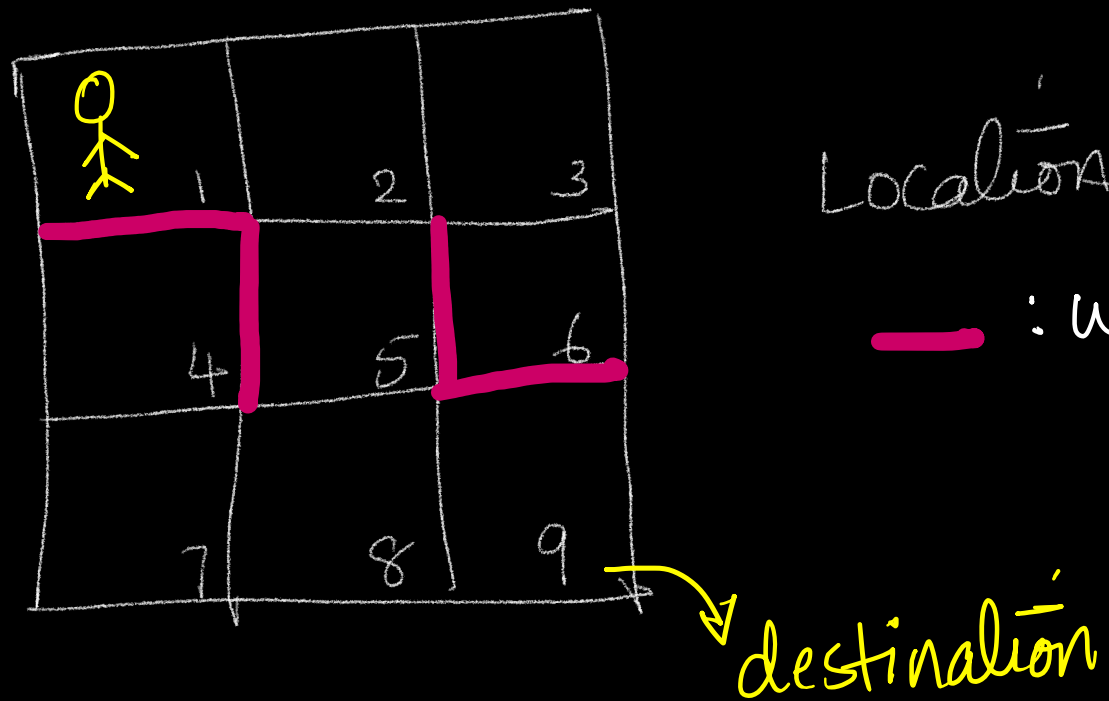
Agent : 

environment :

States : $L_1 - L_9$

Actions : $\rightarrow \leftarrow \downarrow \uparrow$

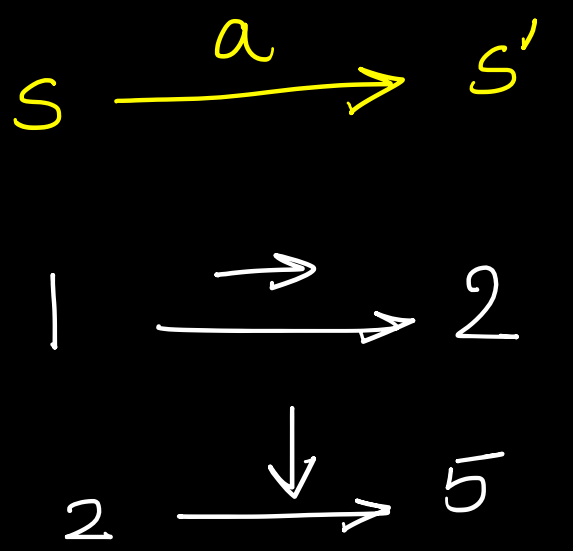
Reward : $R(s, a)$

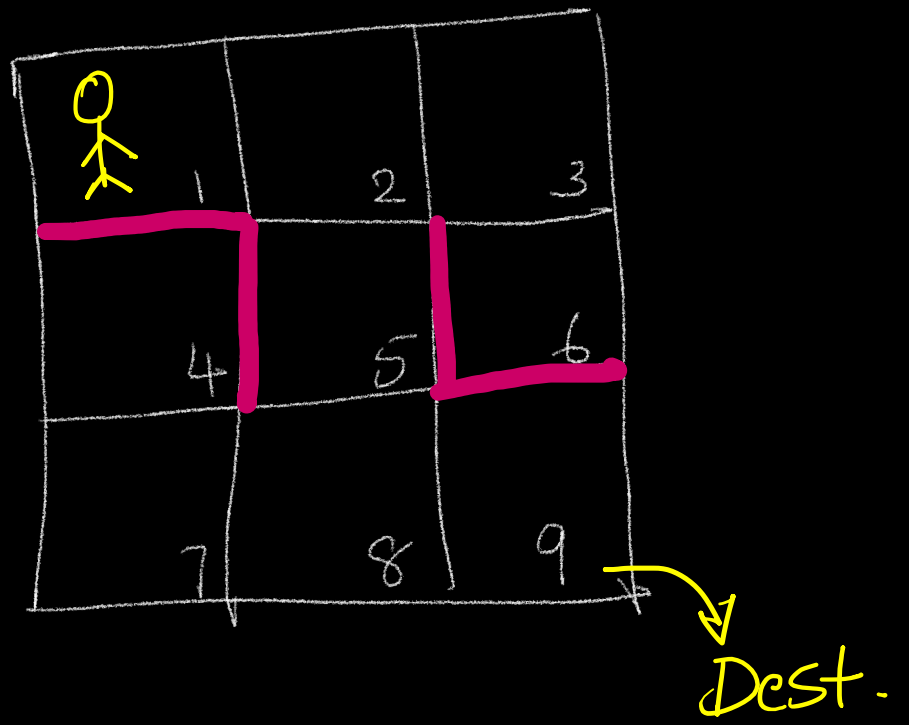


Locations 1-9

— : walls

Value - function: V(s)





$$s \xrightarrow{a} s'$$

$$V(s) = \max_a \left(R(s,a) + \gamma \cdot V(s') \right)$$

Bellman - Equation

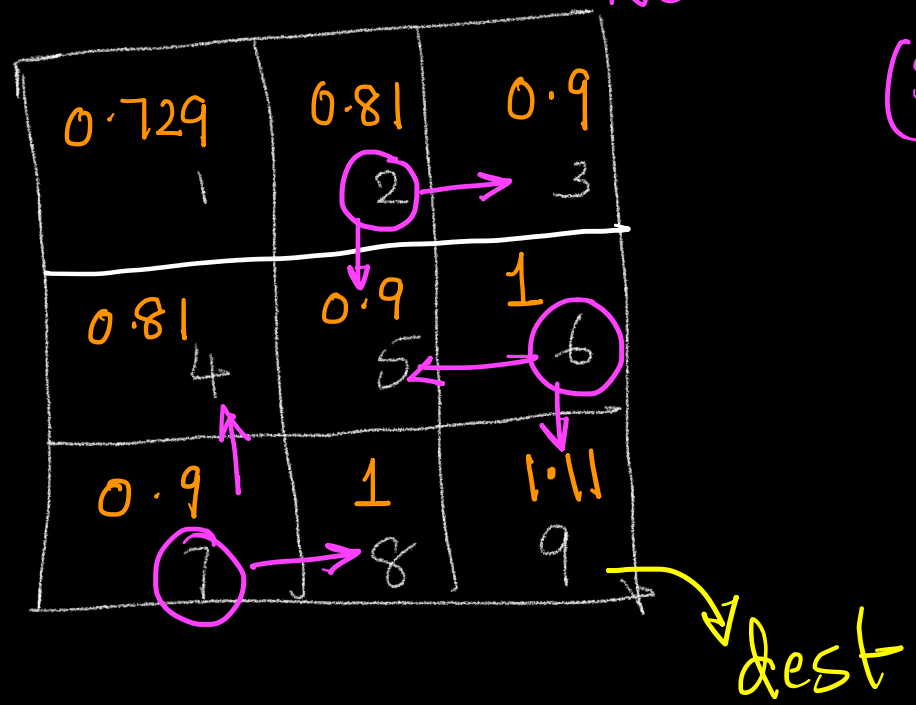
$0 \leq$ discounting factor ≤ 1

Let $\gamma = 0.9$

$$V(S) = \max_{a} \left(R(S,a) + \gamma V(S') \right)$$

Let $\gamma = 0.9$

$R(S,a) = 0$

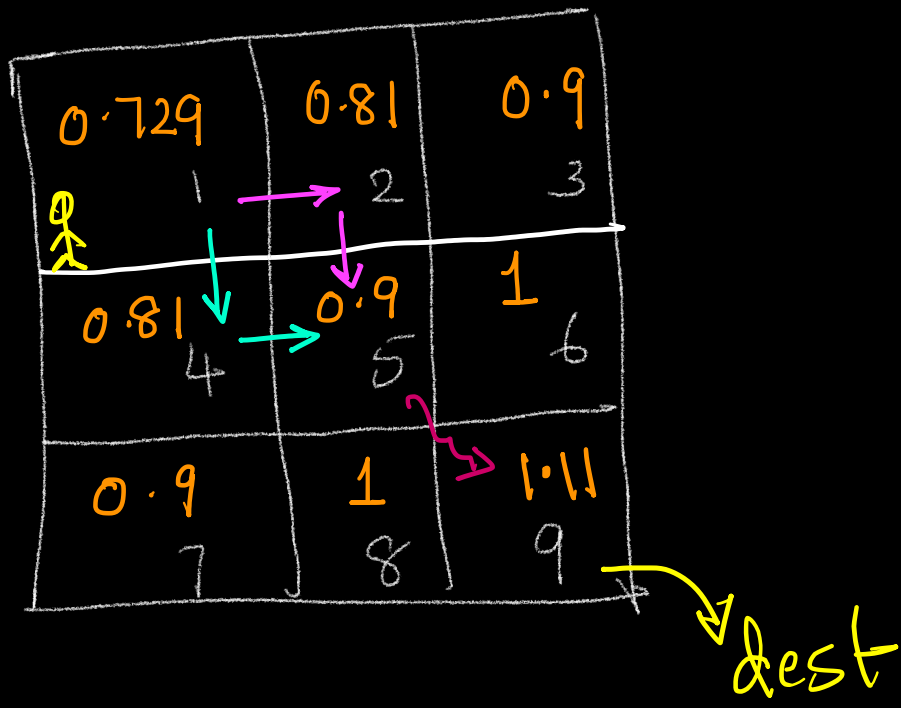


No-walls environment (simpler)

APPLIED ROOTS

Dynamic Programming in CS:

- recursion
- overlapping Sub-problems



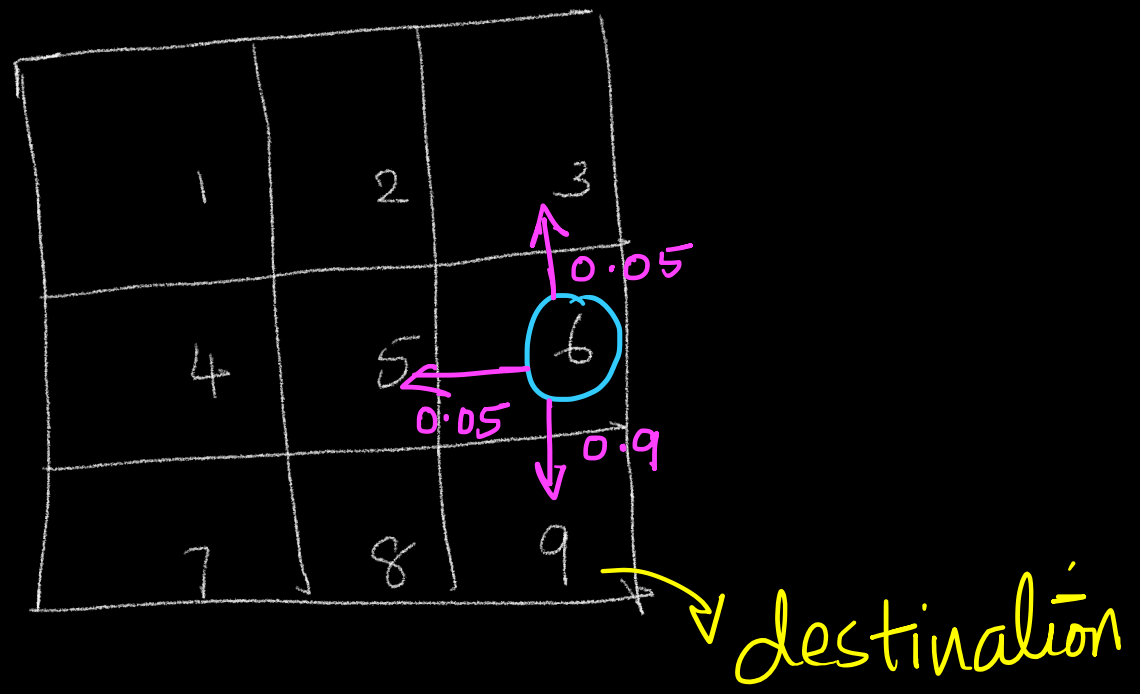
$$V(s) = \max_{a} R(s,a) + \gamma \cdot V(s')$$

Dynamic Programming → by. Bellman

(Q) why not solve it trivially ?

Stocastic - process: Markov Decision process [MDP]

Roomba:

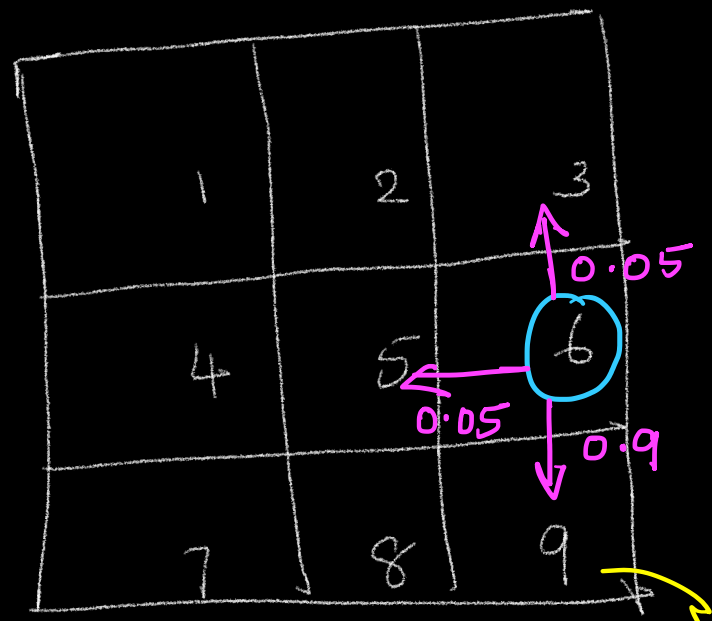


MDP:

↓

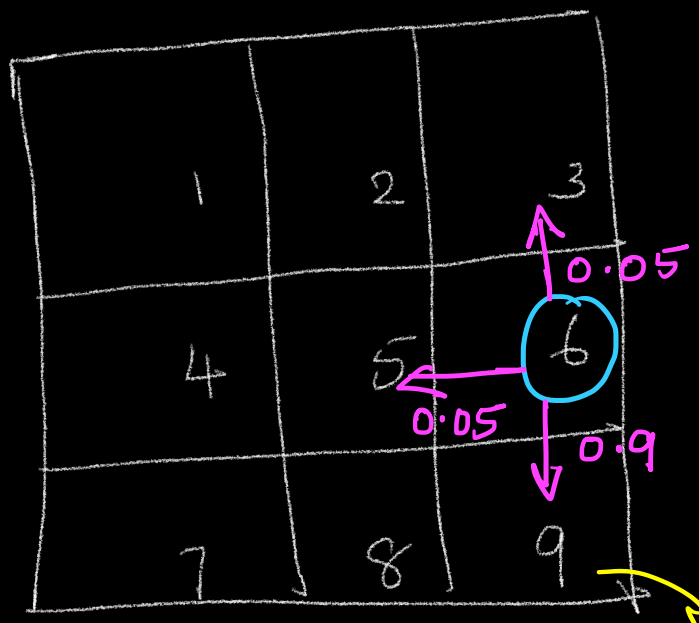
partly stochastic / random

& mostly based on decision made



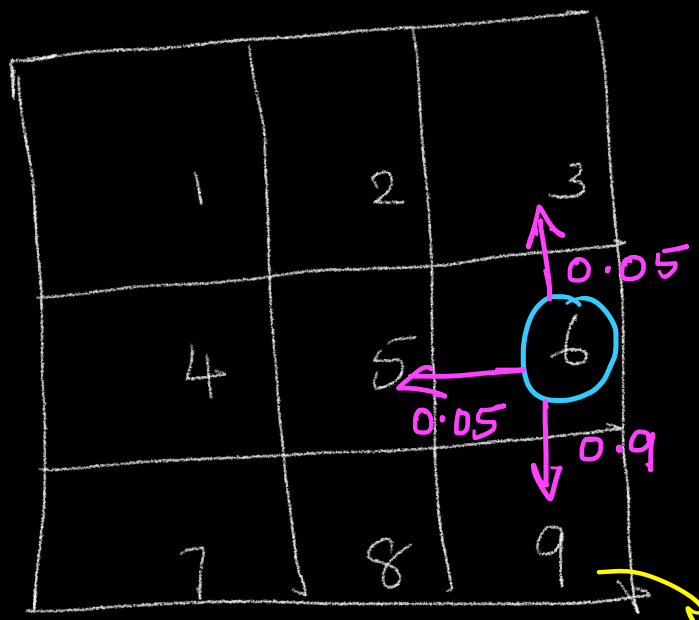
MDP:

$$V(s) = \max_a \left[R(s,a) + \gamma V(s') \right]$$



$$V(s) = \max_a \; R(s,a) + \gamma \cdot \sum_{s'} P(s,a,s') \; V(s')$$

MDP:



$$V(s) = \max_a R(s,a) + \gamma \cdot \sum_{s'} P(s,a,s')\, V(s')$$

Expected value of $V(s')$

$$V(s) = \max_a \left[R(s,a) + \gamma \sum_{s'} p(s,a,s') \cdot V(s') \right]$$

$Q(s,a)$: Q-function

$$\Rightarrow V(s) = \max_a Q(s,a)$$

Q-function:

$$Q(s,a) = R(s,a) + \gamma \cdot \sum_{s'} p(s,a,s') \cdot V(s')$$

$$V(s') \rightarrow \max_{a'} Q(s',a')$$

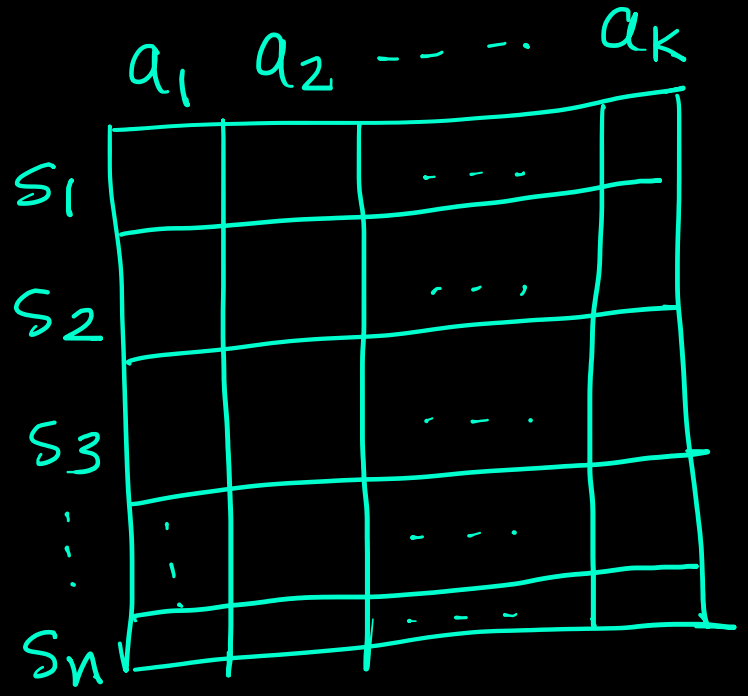
$$Q(s,a) = R(s,a) + \gamma \cdot \sum_{s'} \left(P(s,a,s') \cdot \max_{a'} Q(s',a') \right)$$

Intuition:

$$Q(s,a) = R(s,a) + \gamma \cdot \sum_{s'} \left(p(s,a,s') \cdot \max_{a'} Q(s',a') \right)$$

Let = 1

Q:



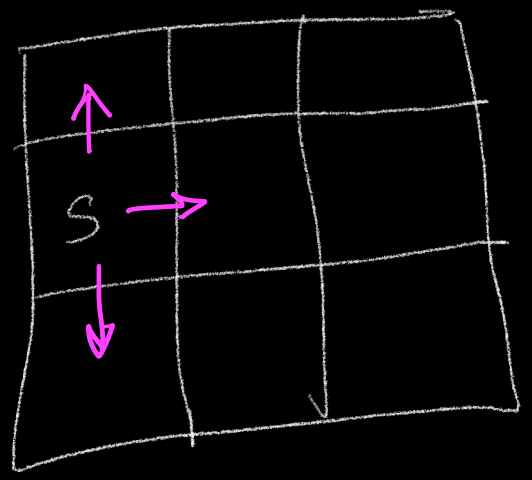
Temposal-difference : TD(s,a)

what if the environment is also changing

Q(s,↑)

Q(s,→)

Q(s→↓)

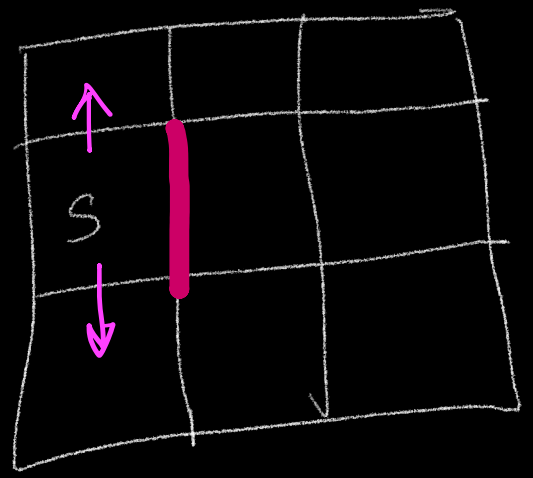


Time t-1:

Q(s,↑)

Q(s,→)

Q(s→↓)



Time t

APPLIED ROOTS

$$Q_{t-1}(s,a) \quad \text{vs} \quad Q_t(s,a)$$

$$TD(s,a) = Q_{new}(s,a) - Q_{old}(s,a)$$

$$= \left[R(s,a) + \gamma \sum_{s'} p(s,a,s') \max_{a'} Q(s',a') \right] - Q_{t-1}(s,a)$$

$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha \cdot TD_t(a,s)$$

↓

how quickly can the robot adapt to env. change

a.k.a. learning rate

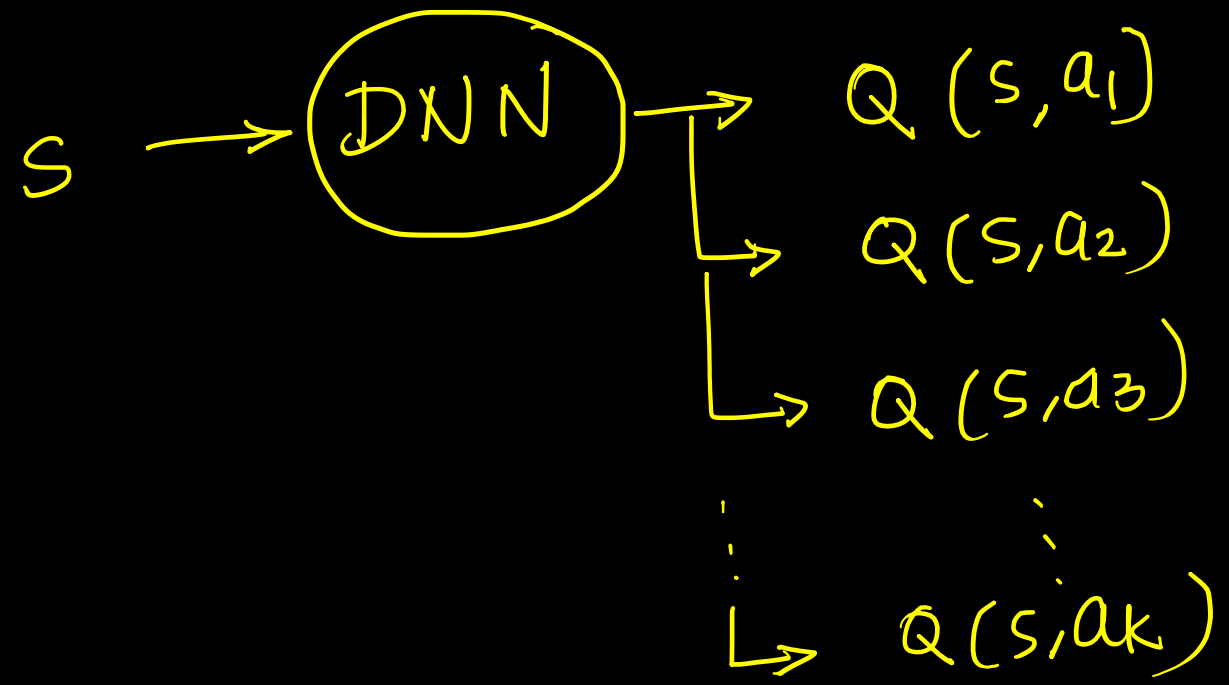
$$Q_t(s,a) = Q_{t-1}(s,a) + \alpha \cdot \left(R(s,a) + \gamma \cdot \sum_{s'} p(s,a,s') \max_{a'} Q(s',a') - Q_{t-1}(s,a) \right)$$

Simplified form:

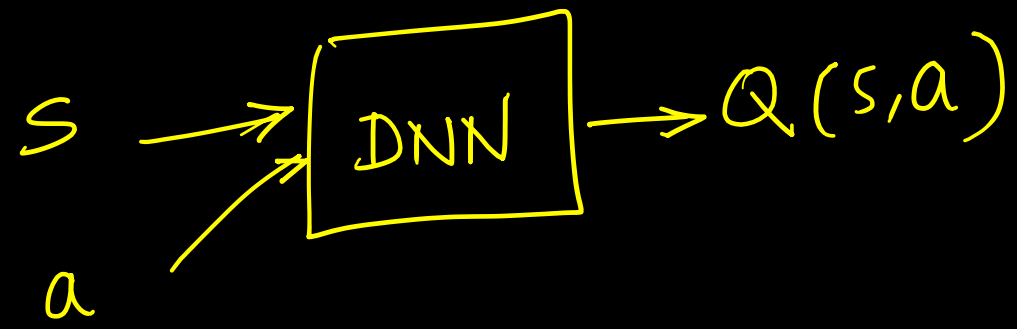
$$Q(s,a) = R(s,a) + \gamma \cdot \max_{a'} Q(s',a')$$

Deep-Q-learning : Intuition

Q(s,a) : Table



not



inefficient

Next Session : DQNN + Code

APPLIED ROOTS

Q & A

APPLIED ROOTS