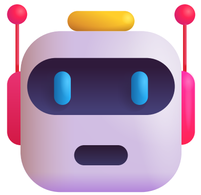


GDG Algiers AI Bootcamp

Reinforcement Learning

By **Kamel Brouthen**

Kamel Brouthen

- Development Core Team Member (AI)
- Ex-Comanager (Multimedia)

Machine Learning
(Regression & Classification)

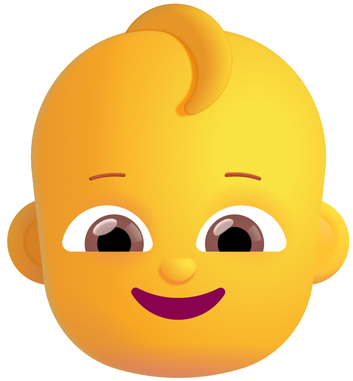
Reinforcement Learning
(The most exciting!)

AI Bootcamp

Generative AI
(LLMs & GANs)

Deep Learning
(CV & NLP)

Building Intuition

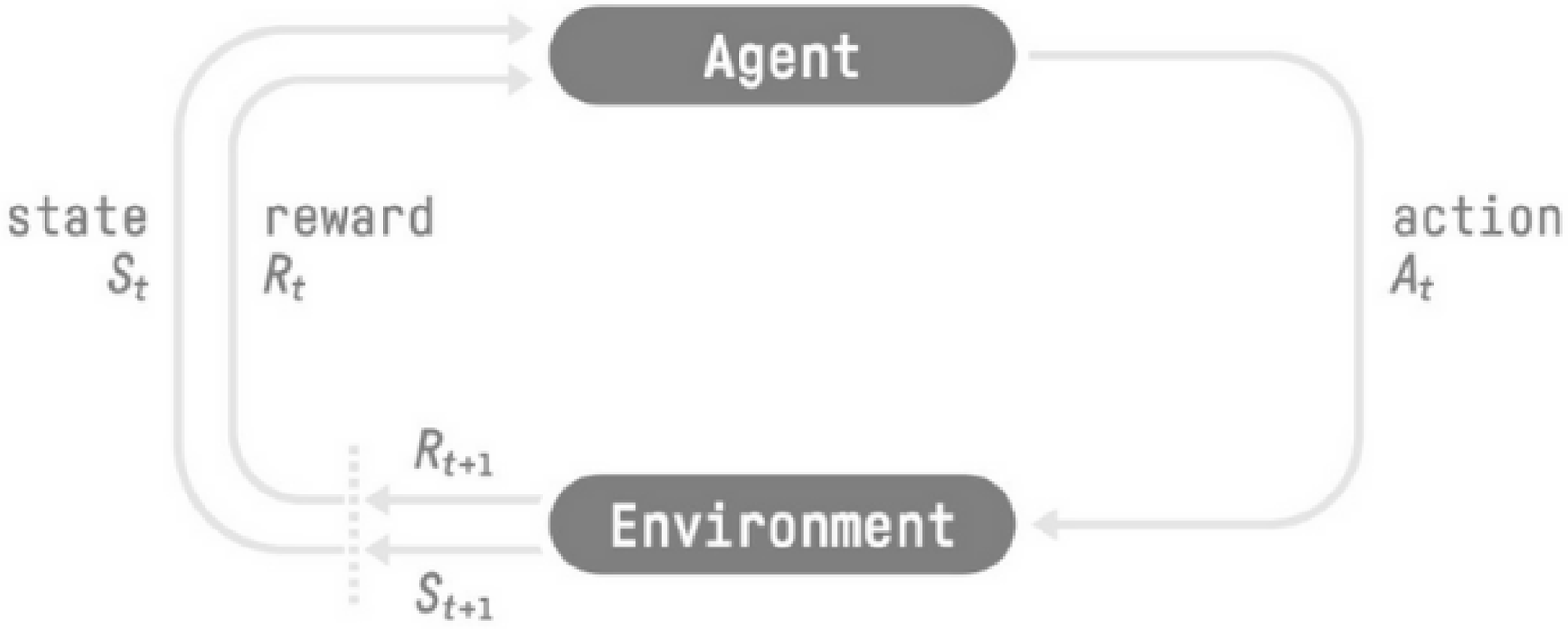


How did we learn about the world?

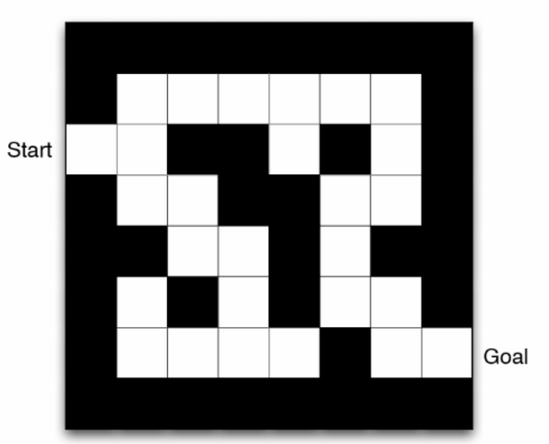
Reinforcement Learning

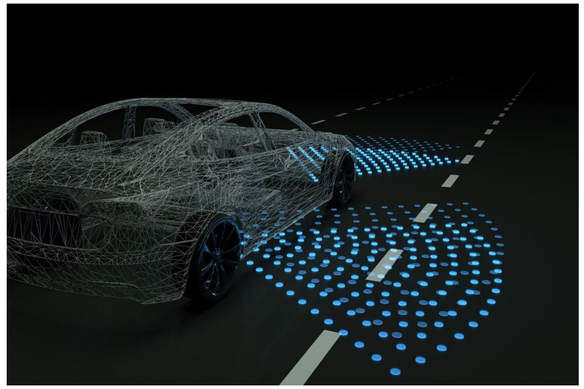
The AI discipline of learning through interaction and feedback signal from a given environment (Trial & Error)

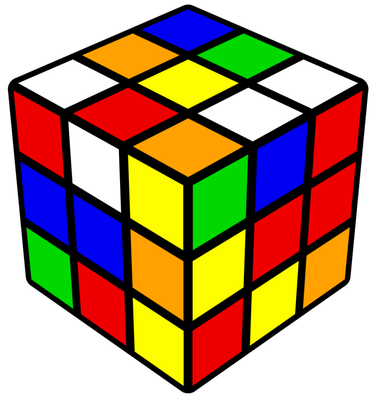
Reinforcement Learning Framework

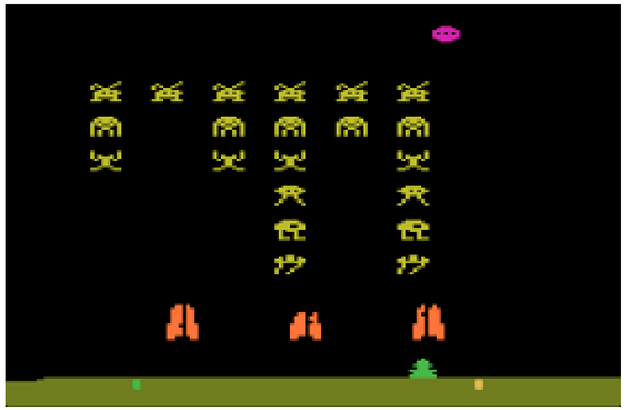


Reinforcement Learning Illustrated









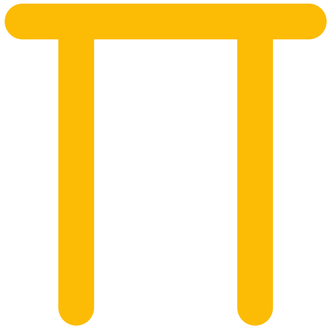
Reinforcement Learning
(Maybe)

The Reward Hypothesis (why RL works)

"All of what we mean by goals and purposes can be well thought of as maximization of the expected value of the cumulative sum of a received scalar signal (reward)."

Sutton, R. S. The reward hypothesis. 2004
http://incompleteideas.net/rlai.cs.ualberta.ca/RLAI/rewardhypothesis.html

Mathematics (AKA AI)



The building blocks of solving RL problems

Return
(Cumulative Reward)

$$R(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots$$

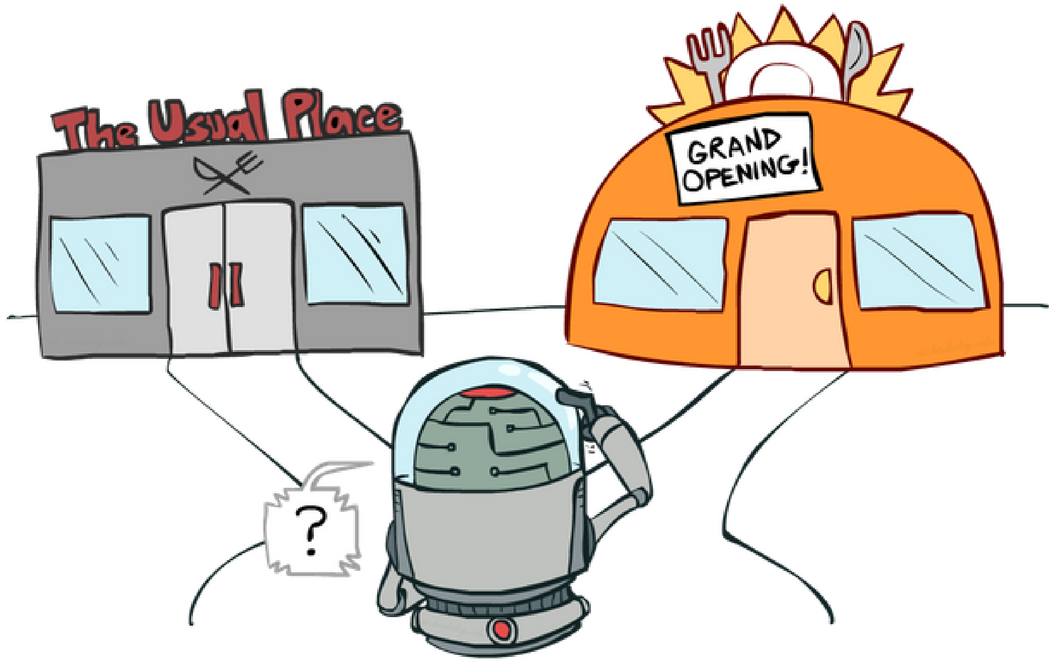
Return: cumulative reward

Gamma: discount rate

Trajectory (read Tau)
Sequence of states and actions

Goal: Maximize the expected return

Exploration VS Exploitation
(to keep in mind)



Solving RL

Value Based Methods

Estimating how good is it to be in a given state and taking a particular action

Policy Based Methods

Directly adjusting the agent's policy over the actions space

Value Based Method (Q-Values)

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\Big[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \Big| s_t = s, a_t = a\Big]$$

Explanation: Being at state s and taking action a, how much return do we expect

Temporal Difference Learning

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

New Q-value estimation

Former Q-value estimation

Learning Rate

Immediate Reward

Discounted Estimate optimal Q-value of next state

Former Q-value estimation

TD Target

TD Error

Explanation: Q-Values estimations get adjusted through experience by the immediate reward and bootstrapping on next trajectories

The Policy
(given final estimations)

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

Explanation: For each state s, we take the action a that maximizes the state-action value

Building Intuition
(Mouse wants Cheese)

Q-Learning Algorithm

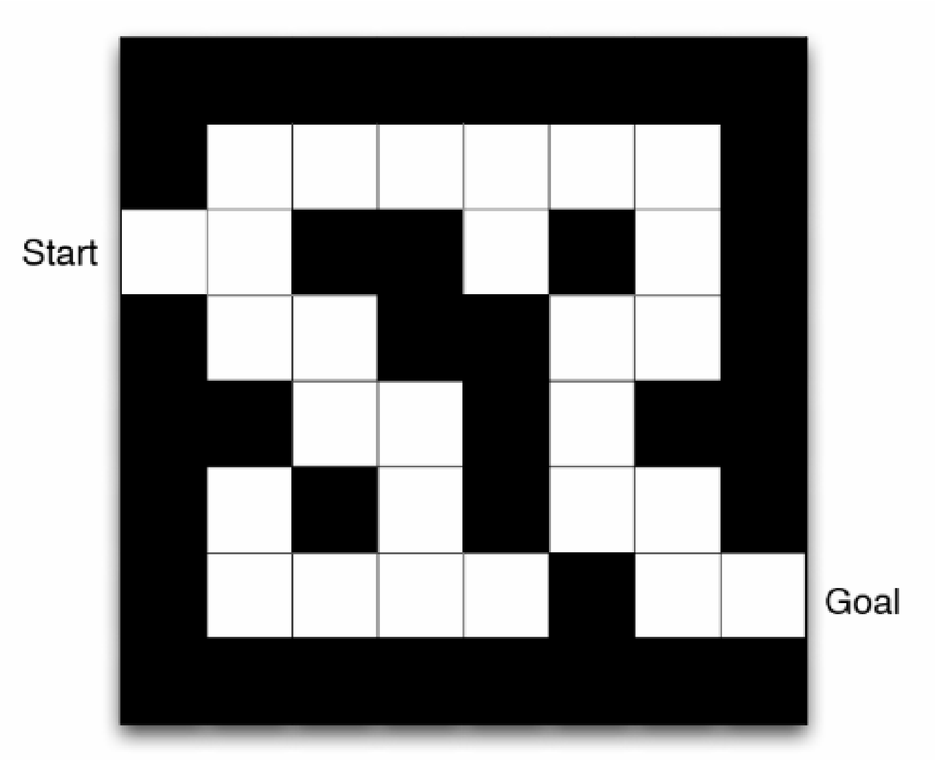
Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$
Initialize $Q(s, a)$, for all $s \in \mathcal{S}^+, a \in \mathcal{A}(s)$, arbitrarily except that $Q(\mathit{terminal}, \cdot) = 0$

Loop for each episode:
  Initialize S
  Loop for each step of episode:
    Choose A from S using policy derived from Q (e.g., ε-greedy)
    Take action A, observe R, S'
    $Q(S, A) \leftarrow Q(S, A) + \alpha\big[R + \gamma \max_a Q(S', a) - Q(S, A)\big]$
    $S \leftarrow S'$
  until S is terminal

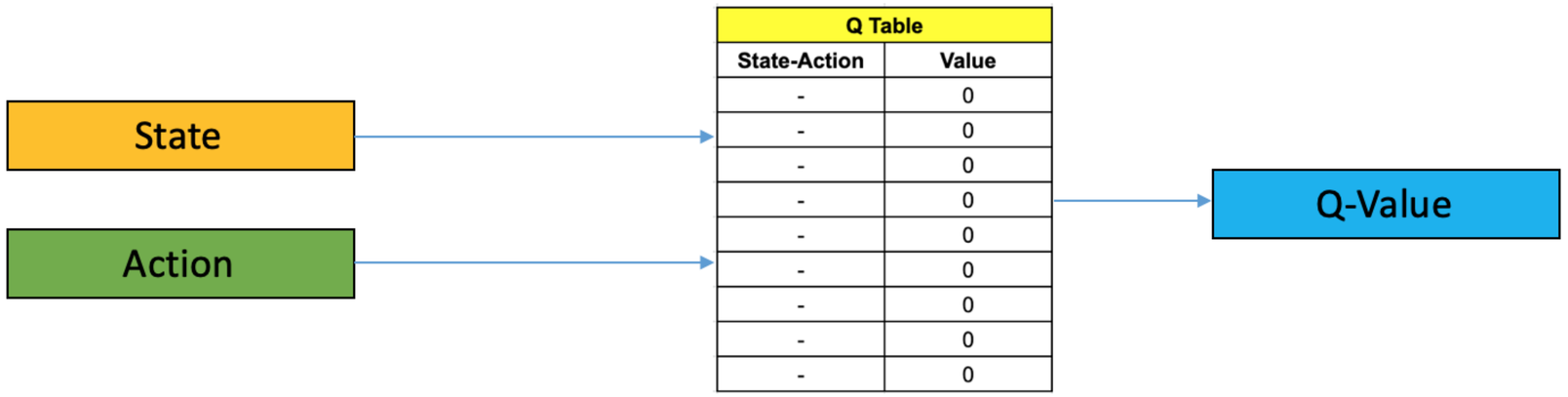
https://medium.com/analytics-vidhya/q-learning-expected-sarsa-and-comparison-of-td-learning-algorithms-e4612064de97

Applied Q-Learning
(Navigate the Maze)

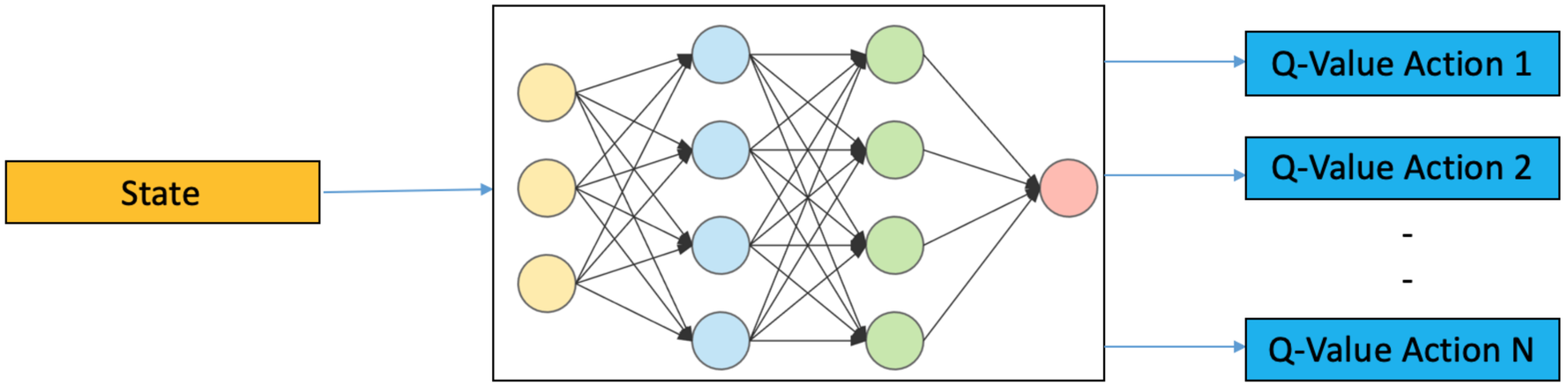


Q-Learning Limitations

- Handling high dimensional state/action spaces
- Function approximation generalization capabilities
- Familiarity and usability only in discrete action spaces



Q Learning

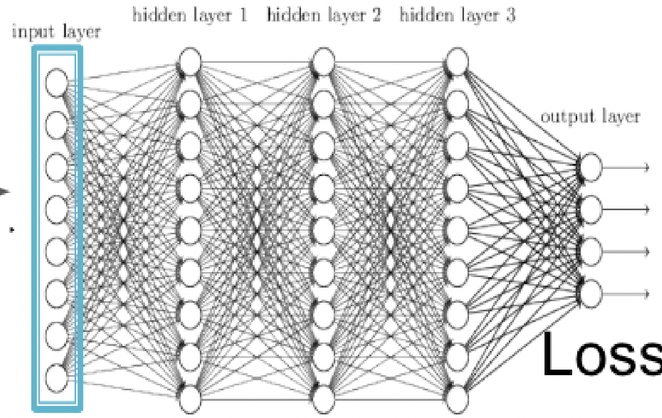


Deep Q Learning

Deep Q-Learning

Q-Network

State-S



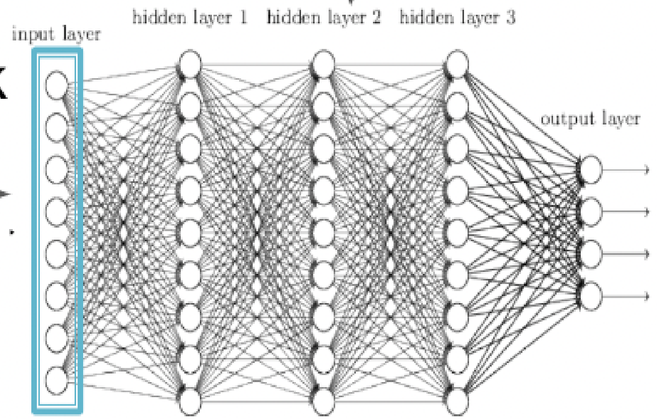
$$Q(s, a; \theta)$$

$$\text{Loss} = \left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2$$

θ updates θ⁻
every C timesteps

Target-Network

State-S`



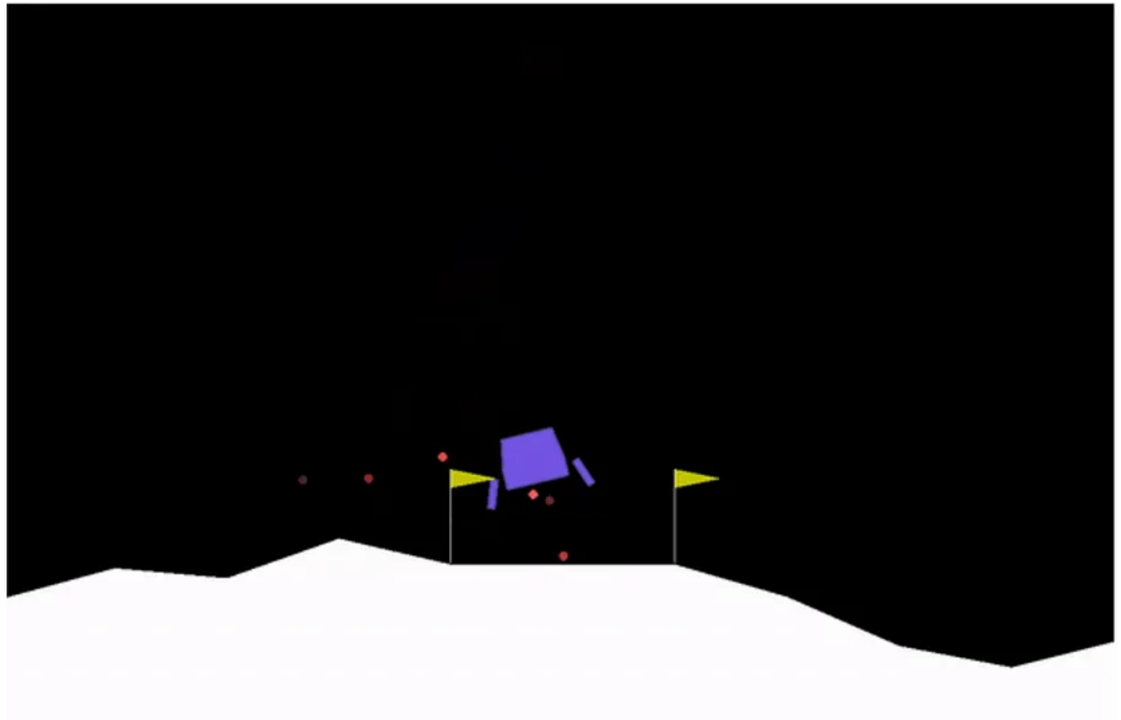
$$Q_T(s', a'; \theta^-)$$

Deep Q-Learning

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

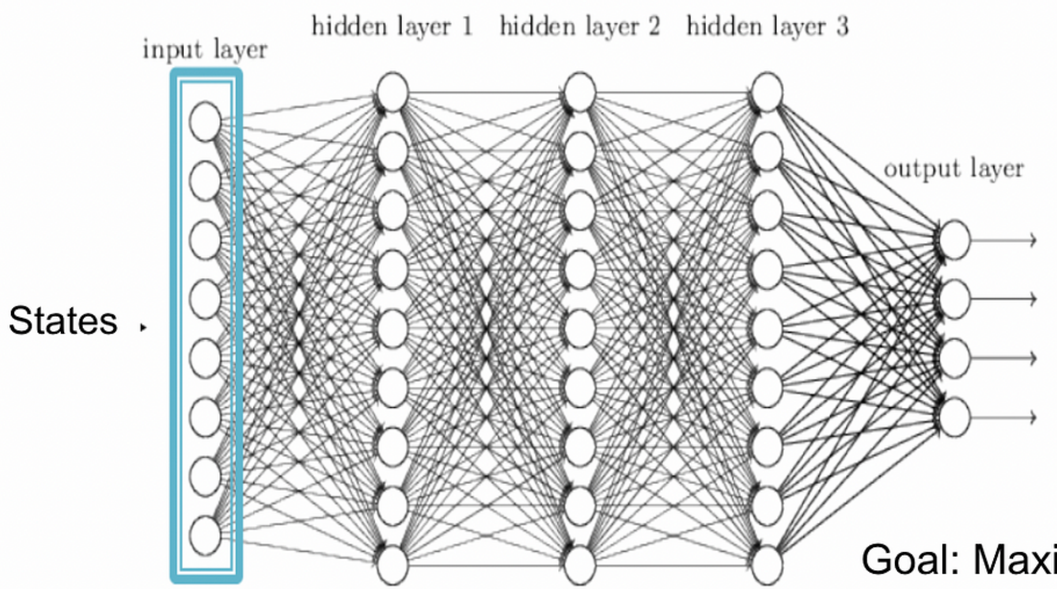
Applied Deep Q-Learning
(Lunar Lander)



Policy Based Method
(Policy Gradient)

If we consider the expected return as an objective function to maximize, we can apply gradient ascent on a differentiable policy function (a neural network as an example)

Policy Based Method (Policy Gradient)



π_θ

Goal: Maximize Expected Return J(θ)

Policy Gradient

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha\widehat{\nabla J(\boldsymbol{\theta}_t)}$$

Update policy parameter, θ using gradient ascent

Policy Based Method (Policy Gradient)

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[R(\tau)] \\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) && \text{Expand expectation} \\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) && \text{Bring gradient under integral} \\
&= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) && \text{Log-derivative trick} \\
&= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau|\theta)R(\tau)] && \text{Return to expectation form} \\
\therefore \nabla_\theta J(\pi_\theta) &= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] && \text{Expression for grad-log-prob}
\end{aligned}
$$

Policy Based Method (Policy Gradient)

$$
\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[R(\tau)] \\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) && \text{Expand expectation} \\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) && \text{Bring gradient under integral} \\
&= \int_\tau P(\tau|\theta)\nabla_\theta \log P(\tau|\theta)R(\tau) && \text{Log-derivative trick} \\
&= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}[\nabla_\theta \log P(\tau|\theta)R(\tau)] && \text{Return to expectation form} \\
\therefore \nabla_\theta J(\pi_\theta) &= \mathop{\mathrm{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t)R(\tau)\right] && \text{Expression for grad-log-prob}
\end{aligned}
$$

Policy gradient methods maximize the expected total reward by repeatedly estimating the gradient $g := \nabla_\theta \mathbb{E}\left[\sum_{t=0}^{\infty} r_t\right]$. There are several different related expressions for the policy gradient, which have the form

$$g = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right], \tag{1}$$

where Ψ_t may be one of the following:

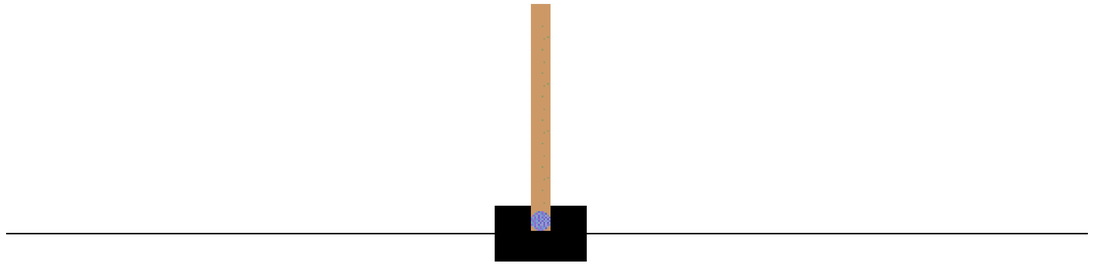
1. $\sum_{t=0}^{\infty} r_t$: total reward of the trajectory.
2. $\sum_{t'=t}^{\infty} r_{t'}$: reward following action a_t.
3. $\sum_{t'=t}^{\infty} r_{t'} - b(s_t)$: baselined version of previous formula.
4. $Q^\pi(s_t, a_t)$: state-action value function.
5. $A^\pi(s_t, a_t)$: advantage function.
6. $r_t + V^\pi(s_{t+1}) - V^\pi(s_t)$: TD residual.

The latter formulas use the definitions

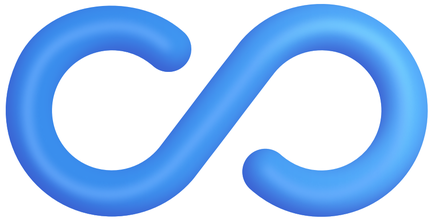
$$V^\pi(s_t) := \mathbb{E}_{\substack{s_{t+1:\infty},\\ a_{t:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \qquad Q^\pi(s_t, a_t) := \mathbb{E}_{\substack{s_{t+1:\infty},\\ a_{t+1:\infty}}}\left[\sum_{l=0}^{\infty} r_{t+l}\right] \tag{2}$$

$$A^\pi(s_t, a_t) := Q^\pi(s_t, a_t) - V^\pi(s_t), \quad \text{(Advantage function)}. \tag{3}$$

Applied Policy Gradient
(Cart Pole)



Human Learners!



Agents are learning, so should we...

For once, forever.

Thank you!