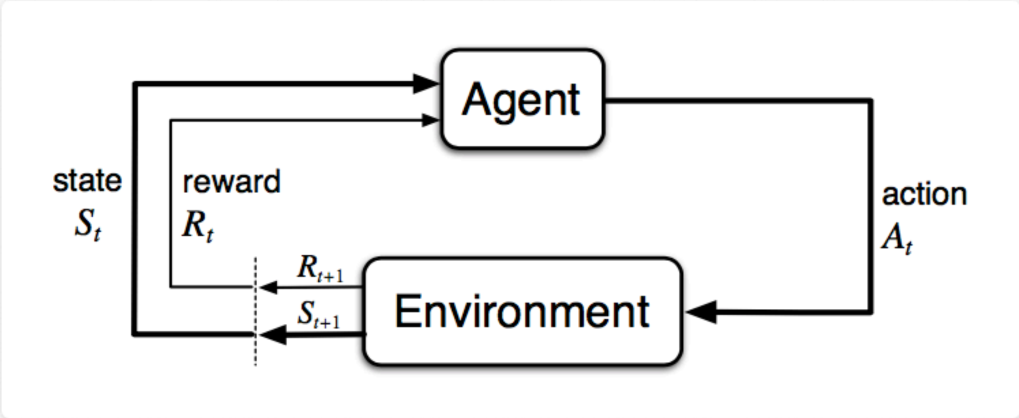
REINFORCEMENT LEARNING

Ahmed El-Khouly

CONTENTS

- RL refresher
- Value based methods
  - Q-Learning
  - Deep Q-Learning
  - DDQN
  - DDDQN
- Policy based methods
  - Policy gradient
- What’s next

RECAP



- ***Probability of future states only depends on present***

  $P(S_{t+1}|\, S_t) = P(S_{t+1}|\, S_1, S_2, S_3, \dots, S_t)$
- **Expected discounted reward** $G_t = R_{t+1} + \gamma\, G_{t+1}$
- ***RL functions***
  - $\pi(a|S_t) = p(A_{t=a}\,|S_t)\ \forall\, a\ \in A$
  - $V_\pi(s) = E_\pi[G_t \mid S_t = s]$
  - $Q_\pi(s, a) = E_\pi[G_t \mid S_t = s,\ A_t = a]$
- **Bellman Optimality equations**

$$Q_*(s, a) = E[R_{t+1} + \gamma max_{a'}\, Q_*(s', a') \mid S_{t+1} = s', A_t = a]$$

LEARNING ALGORITHMS – Q-LEARNING

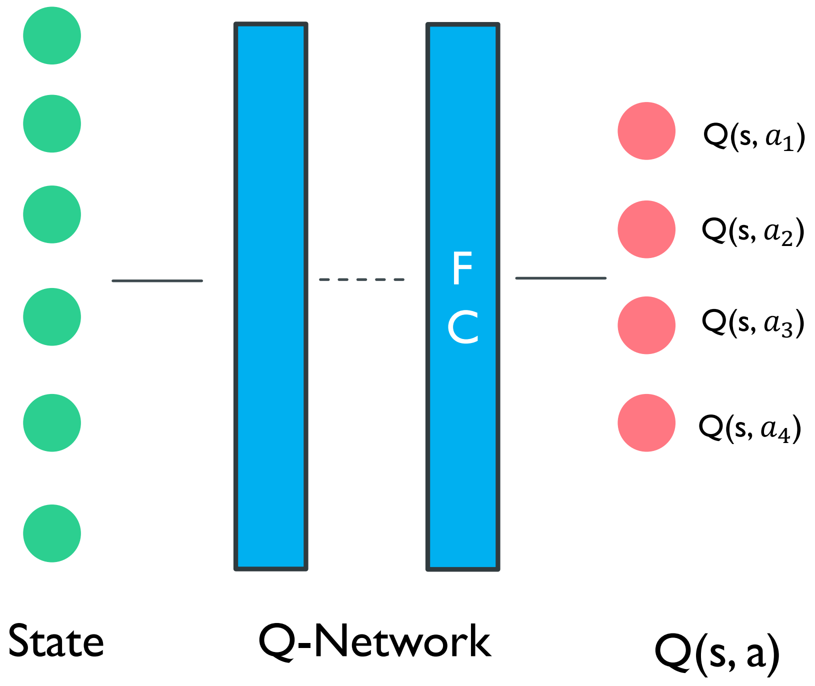
- **Value-based learning**
- **Temporal Difference-** can learn partial time-step subsequence
- **Continuous and episodic** tasks- does not need to wait until episodes end
- **Off-policy-** updates Q-value using greedy action on next state
- **Exploring starts convergence condition**
  - $\pi(a|s) > 0 \ \forall \, a \ \in A, \ \mathrm{s} \in S$
  - ϵ-$Greedy$ policy (soft policy)
- **Q-function update rule:** $q(s,a) \mathrel{+}= \ \alpha(\underbrace{\boxed{R + \ max_{a'} q(s',a') - q(s,a)}}_{\text{TD Error}})$

Note: *from bellman optimality equation* $q_*(s,a)$ = R+ $\gamma \ max_{a}{}' \ q(s',a')$

LEARNING ALGORITHMS – DEEP Q-NETWORK (DQN)

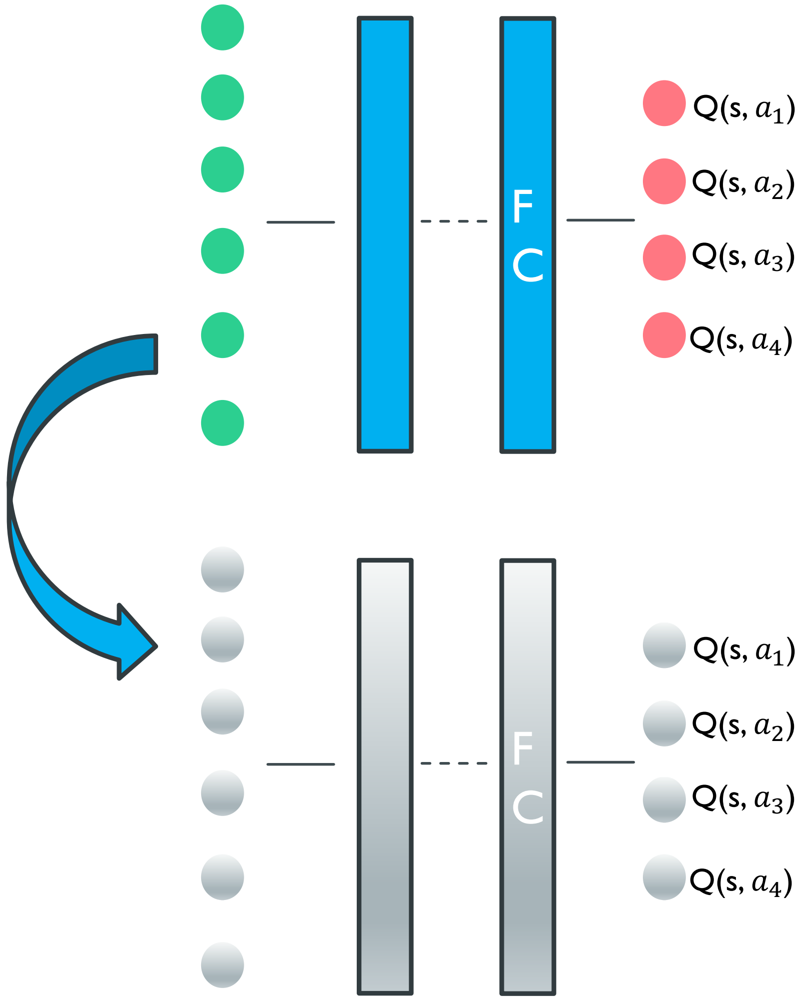
- Q-learning performance drops as the state/actions gets more sophisticated
  - Frozen Lake with 4x4 grid had 4x4x4 = 64 entries that need to be learnt
  - The method does not scale as the number of states increase
- Use Neural networks to approximate optimal Q function
  - **Input layer-** State
  - **Output-** predict the Q-value for every action
  - **Q-Target-** $\boldsymbol{Q_{target} = R + max_a q(s', a)}$
  - **GD Update rule-** $\boldsymbol{\Delta w = \alpha[(R + \gamma max_a q(s', a) - q(s, a)]\Delta q(s, a)}$

  Note: *from bellman optimality equation* $q_*(S,A) = R + \gamma\ max_a\ Q(s', a')$
- **Experience replay**
  - Remember old experiences
  - Reduce correlation between experiences

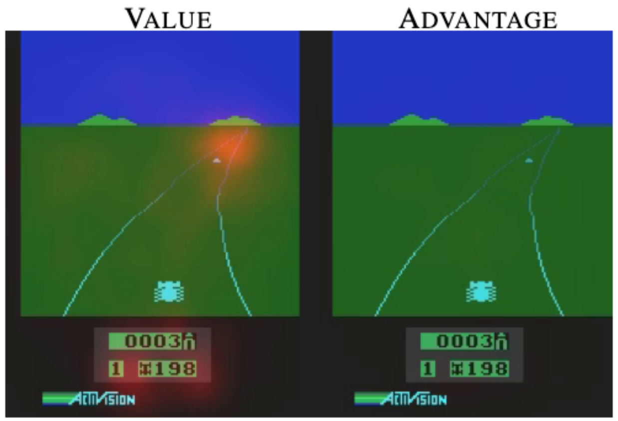


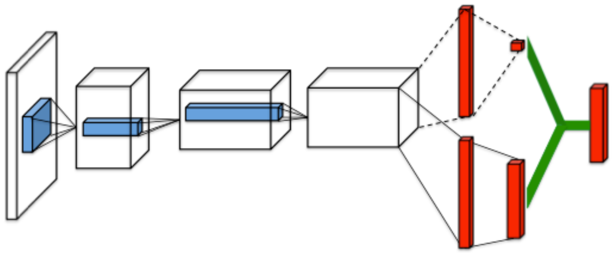
LEARNING ALGORITHMS – DOUBLE DEEP Q-NETWORK (DDQN)

- **Problem- moving targets**
  - Target $= R + \gamma max_a q(s', a)$ predicted using same weights
  - "network chasing its own tail" phenomenon slow training
- **Solution:** use two neural networks with identical architecture
  - Main DQN - learn approximating Q function
  - Target DQN – fixed weights network used to provide target estimates
  - Update target DQN weights from main DQN every τ time-setps
    - Soft update
    - Hard update



DUELING DOUBLE DEEP Q-NETWORK (DDDQN)





- In DQN, network must learn Q-value for every state-action pair
- Some states are valuable or detrimental regardless of action
- It is beneficial for network to learn value of states & actions separately
  - V(s) – is the value of certain state
  - A(s, a) – advantage of actions within a state
- Simple aggregation leads to identifiability issue during back propagation
- The two value streams are combined through aggregation layer:

$$Q(s,a) = \mathrm{V(s)} + A(s,a) - \max A(s,a)$$

$$Q(s,a) = \mathrm{V(s)} + A(s,a) - {}^{1}/_{|A|} \sum_a A(s,a)$$

POLICY BASED REINFORCEMENT

- Policy based reinforcement optimize policy function (π) directly
- Several state-of-art reinforcement methods are derivatives of Policy based learning:
  - Advantage Actor Critic(A2C)
  - Proximal Policy Optimization
- Several advancements based on policy gradient methods
  - Beating best players in Defense of the Ancients (DOTA 2) game with (PPO)
- In general, policy-based methods outperform value-based methods

ELEMENTS OF POLICY GRADIENT

- Policy function parametrized with modeled parameter Ø: $\boldsymbol{\pi_\theta}$
- Objective function: J(Ø) that measures how good the policy is

$J(\emptyset) = E_\pi(R)$

$= \sum_s d(s) \sum_a \pi(a|s) \sum_{s',r} P(s',r \mid s,a)\, r$

- Learning Algorithm: Gradient ascent
  - Update rule: $\emptyset = \emptyset + \nabla_\emptyset J(\emptyset)$ ← maximise rewards

$\nabla J(\emptyset) = \nabla_\emptyset \left(\sum_s d(s) \sum_a \pi(a|s) \sum_{s',r} P(s',r \mid s,a)\, r \right)$

d(s) depends on Ø: $as\ \emptyset \rightarrow policy \rightarrow actions \rightarrow d(s)$

POLICY GRADIENT – OBJECTIVE GRADIENT

- Policy gradient theorem allows $\nabla J(\emptyset)$ to be expressed as:

  $\nabla J(\emptyset) = \sum_s d(s) \sum_a \nabla_\emptyset \pi(a|s) Q_\pi(s,a)$

- Eliminate summation over all states

  $\nabla J(\emptyset) = \sum_s d(s) \nabla_\emptyset \sum_a \pi(a|s) Q_\pi(s,a)$

  $= E_d[\nabla_\emptyset \sum_a \pi(a|s) Q_\pi(s,a)]$

- Eliminate summation over all actions

$$\sum_a \nabla_\emptyset \pi(a|s) Q_\pi(s,a) = \sum_a \pi(a|s) \frac{\nabla_\emptyset \pi(a|s)}{\pi(a|s)} Q_\pi(s,a)$$

$$= E_\pi[\frac{\nabla_\emptyset \pi(a|s)}{\pi(a|s)} Q_\pi(s,a)]$$

$$= E_\pi[\nabla_\emptyset \log\pi(a|s) Q_\pi(s,a)]$$

Note: $\nabla \log(f(x)) = \frac{\nabla f(x)}{f(x)}$

$\Rightarrow$ $\nabla J(\emptyset) = E_d[E_\pi[\nabla_\emptyset \log\pi(a|s) Q_\pi(s,a)]]$

POLICY GRADIENT – STOCHASTIC SAMPLING

- Approximate expectations(E_d, E_π) using stochastic sampling:

  $\nabla J(\emptyset) = \nabla_\emptyset \, log\pi(\tau) \, E_\pi[R(\tau)]$ where $\tau: s_0, a_0, r_0, s_1, a_1, r_1$

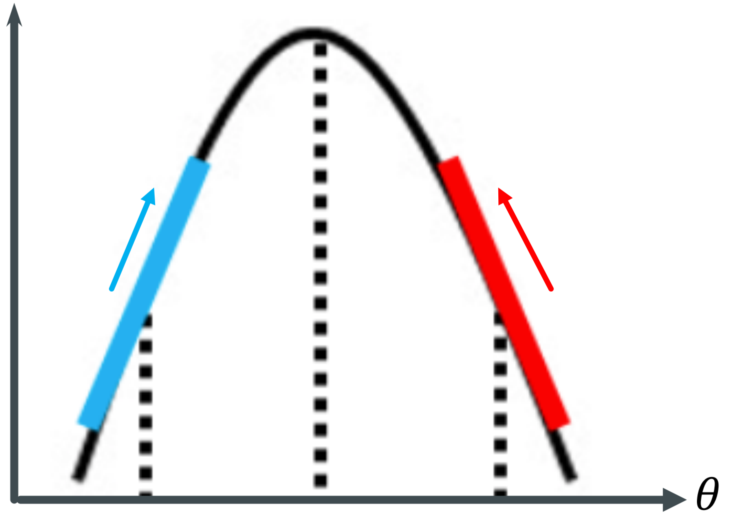
- Update rule for parameter θ: $\boxed{\theta = \theta + \alpha \nabla_\emptyset \, log\pi(\tau) \, R(\tau)}$
- Intuition

direction to change θ to increase $\pi(a|s)$

$$\theta = \theta + \alpha \frac{\nabla_\emptyset \, \pi(a|s)}{\pi(a|s)} Q_\pi(s,a)$$

weighted by expected rewards

divide by $\pi(a|s)$ to punish sub optimal actions with initial high probability



POLICY GRADIENT – DEEP LEARNING

- Define the Loss function

$L(\theta) = -\, log\pi(\tau)\, R(\tau)$

- Update rule for parameter θ

$\theta = \theta - \nabla_\theta L(\theta)$ … gradient descent

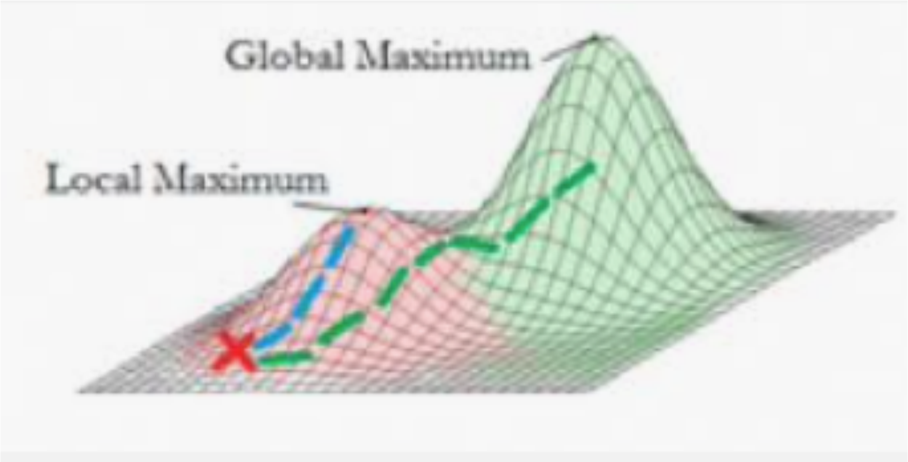
$= \theta - (-\nabla_\theta\, log\pi(\tau)\, R(\tau))$

$= \theta + \nabla_\theta\, log\pi(\tau)\, R(\tau)$

- Allow usage of deep learning library optimization algorithms.

PROS AND CONS OF POLICY GRADIENT

- Pros
  - Convergence
    - Policy gradient follow gradient of policy
    - Q-learning methods deploy exploration/exploitation tradeoff
  - Learn stochastic policies
    - solve perceptual aliasing
  - Support for large action space(continuous)
- Cons
  - Following the policy gradient, can converge on local optima



WHAT IS NEXT

- Advanced learning algorithms
  - Advantage Actor Critic (A2C)
  - Proximal Policy Optimization (PPO)
- Interesting complex applications
- Curiosity driven learning

THANK YOU