Computational & Statistical Challenges of DeepQ/Reinforcement Learning

Saptarashmi Bandyopadhyay
Bodhisatwa Chatterjee

CSE 597 – Large Scale Machine Learning
April 18, 2019

Presentation Outline

- Problem Statement and Motivation
- Introduction to Reinforcement Learning (Background Material)
- Alternative Approaches to Q-Learning (Value Function Approach)
- Convergence Analysis of Q-Learning (Asymptotic Rate)
- Convergence of Q-Learning in Polynomial Time (Learning Rate Trick)
- Challenges to Theoretical Analysis of Deep Q Networks (Two Major Challenges)
- Statistical Convergence Rate of Deep Q-Learning
- Sample Efficiency of Deep Q-Learning
- Over-Estimation Challenges in Deep Q-Learning (Double Q-Learning)
- Conclusion and Major Takeaways

Problem Statement and Motivation

- Problem: to investigate the computational and statistical challenges of Deep Q / Reinforcement Learning

- Application areas – game theory, operations research, information theory, Robotics and various areas of AI

Computational and Statistical Challenges

- The iterative update of value iteration algorithm for computation of optimal Q function in Reinforcement Learning is Not Scalable.
- The scalability of machine learning algorithms to both high dimensionality and large sample are major issues.

Computational and Statistical Challenges

- Scalable implementations of large-scale non-smooth optimization procedures
    - Dealing with non-convexity – use of deep neural networks as the function approximator
    - Investigation of Convergence rate as the major Computational challenge
    - Over-estimation challenge in Deep Q Learning

- Statistical Challenge – sample efficiency
    - Size of the dataset has to be large enough to achieve convergence in the expected reward value
    - Convergence is also dependent on the variance of the data

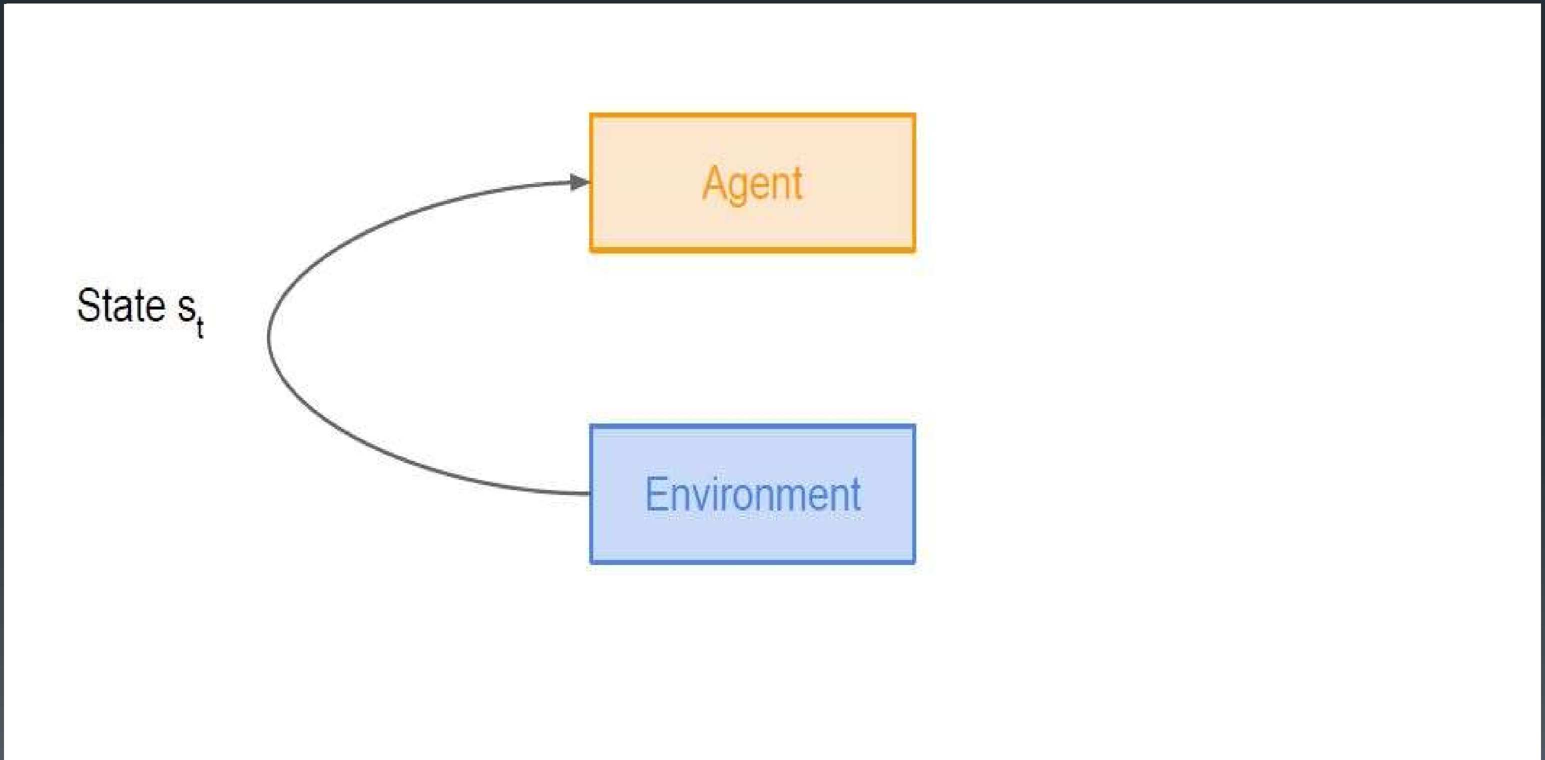
Introduction to Reinforcement Learning

- Problem involving an agent and an environment

Agent
Environment

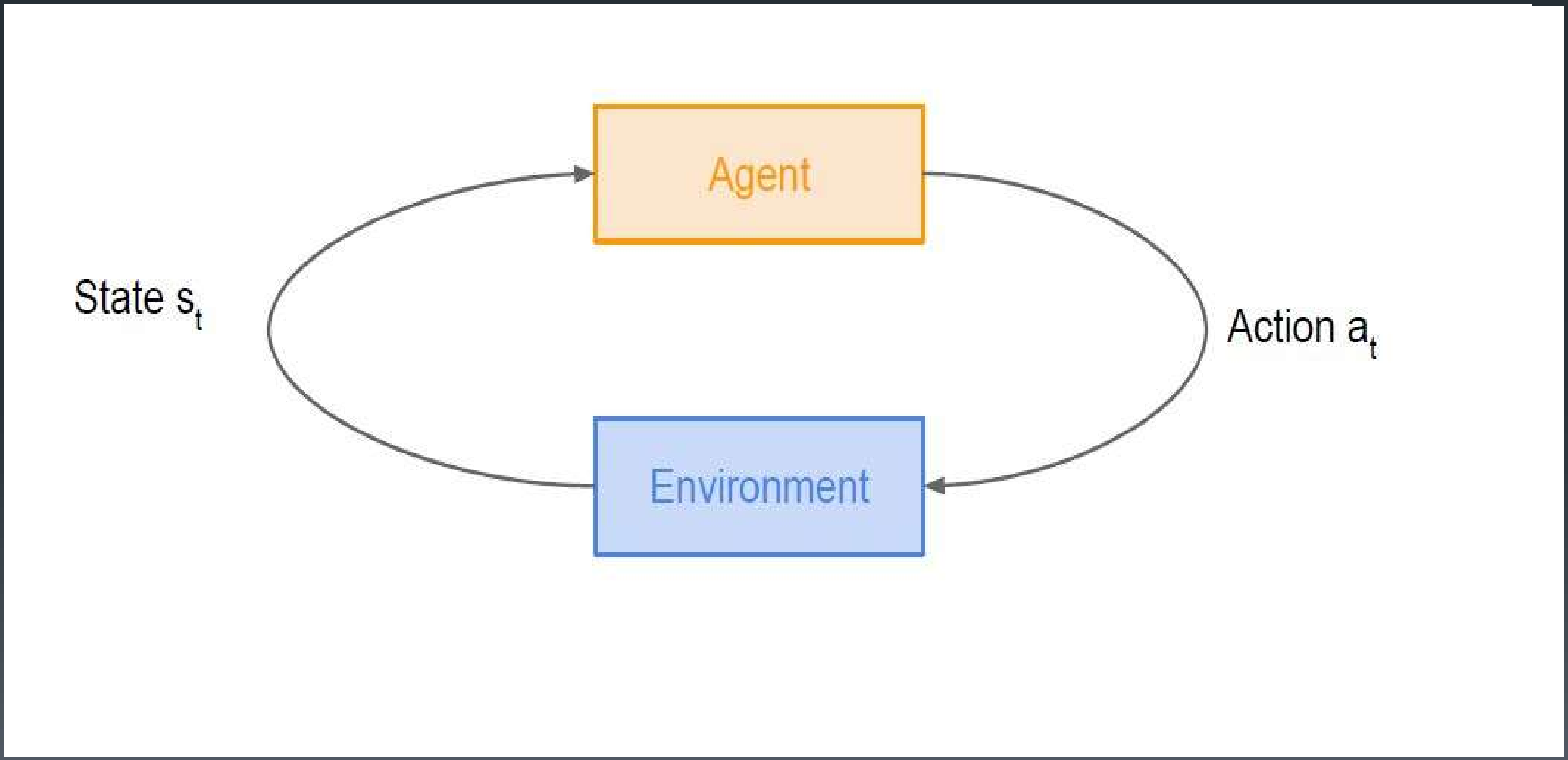
Introduction to Reinforcement Learning

- Environment gives current state information to agent



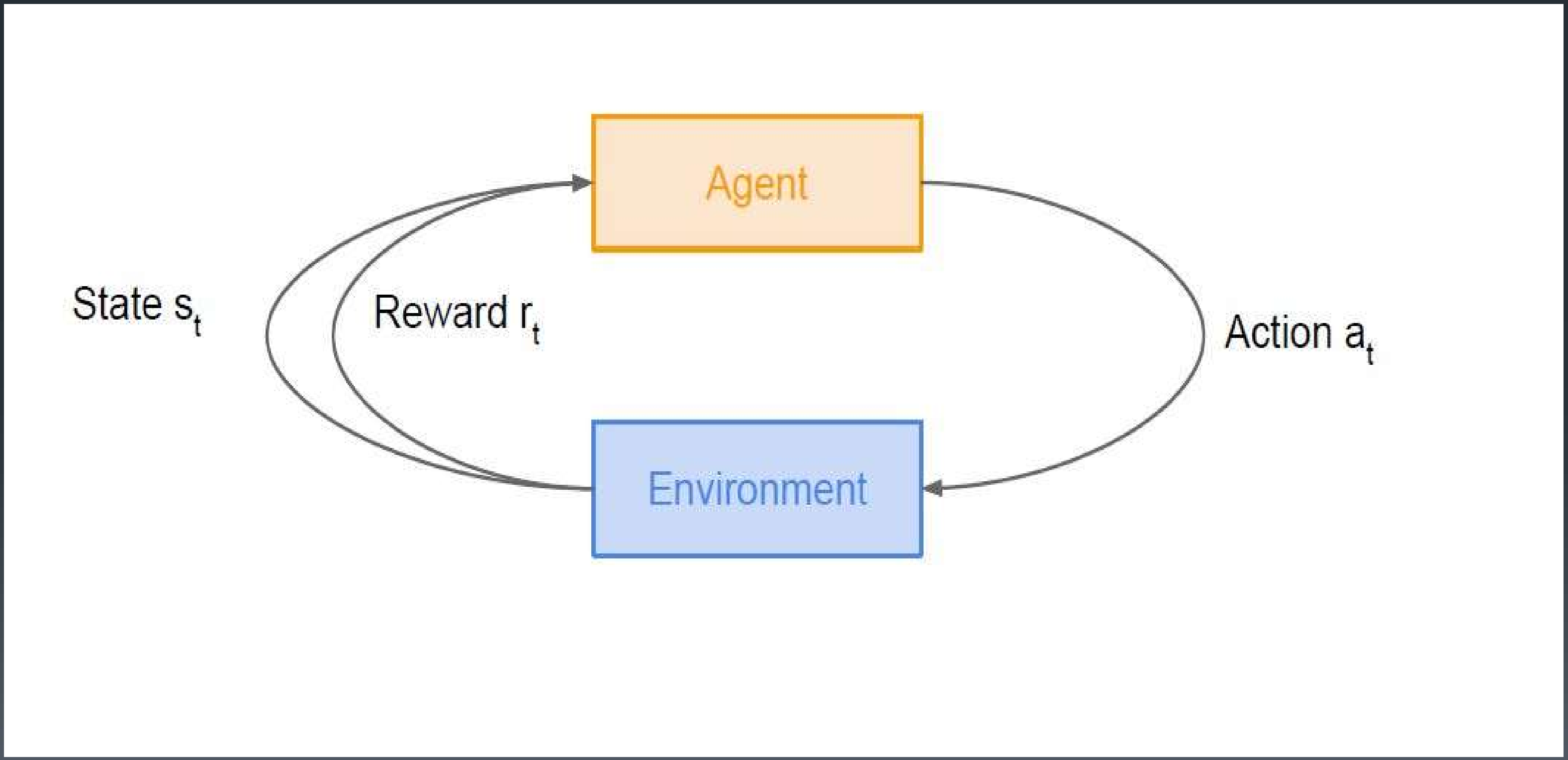
Introduction to Reinforcement Learning

- Agent takes an action on the environment



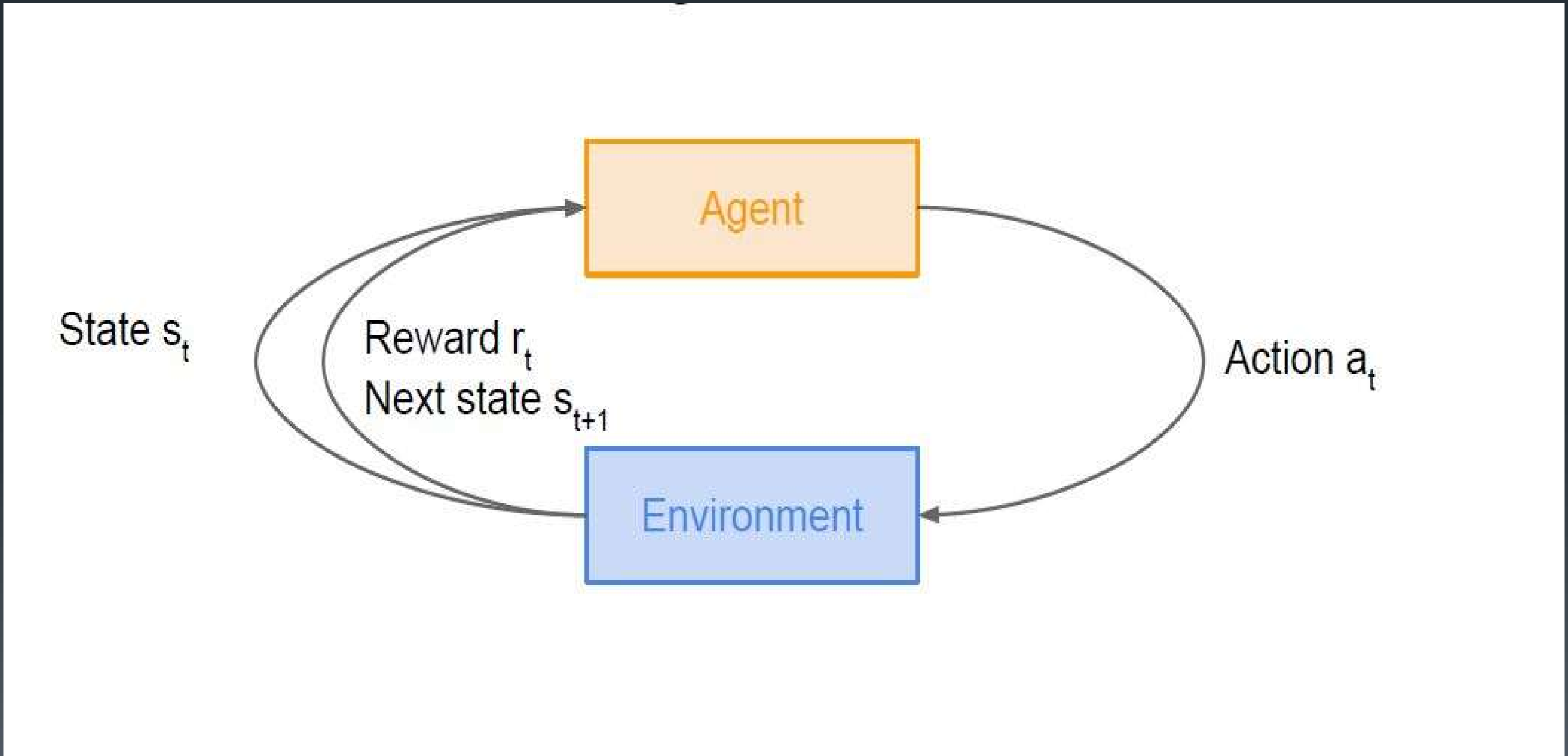
Introduction to Reinforcement Learning

- Agent receives a reward from the environment



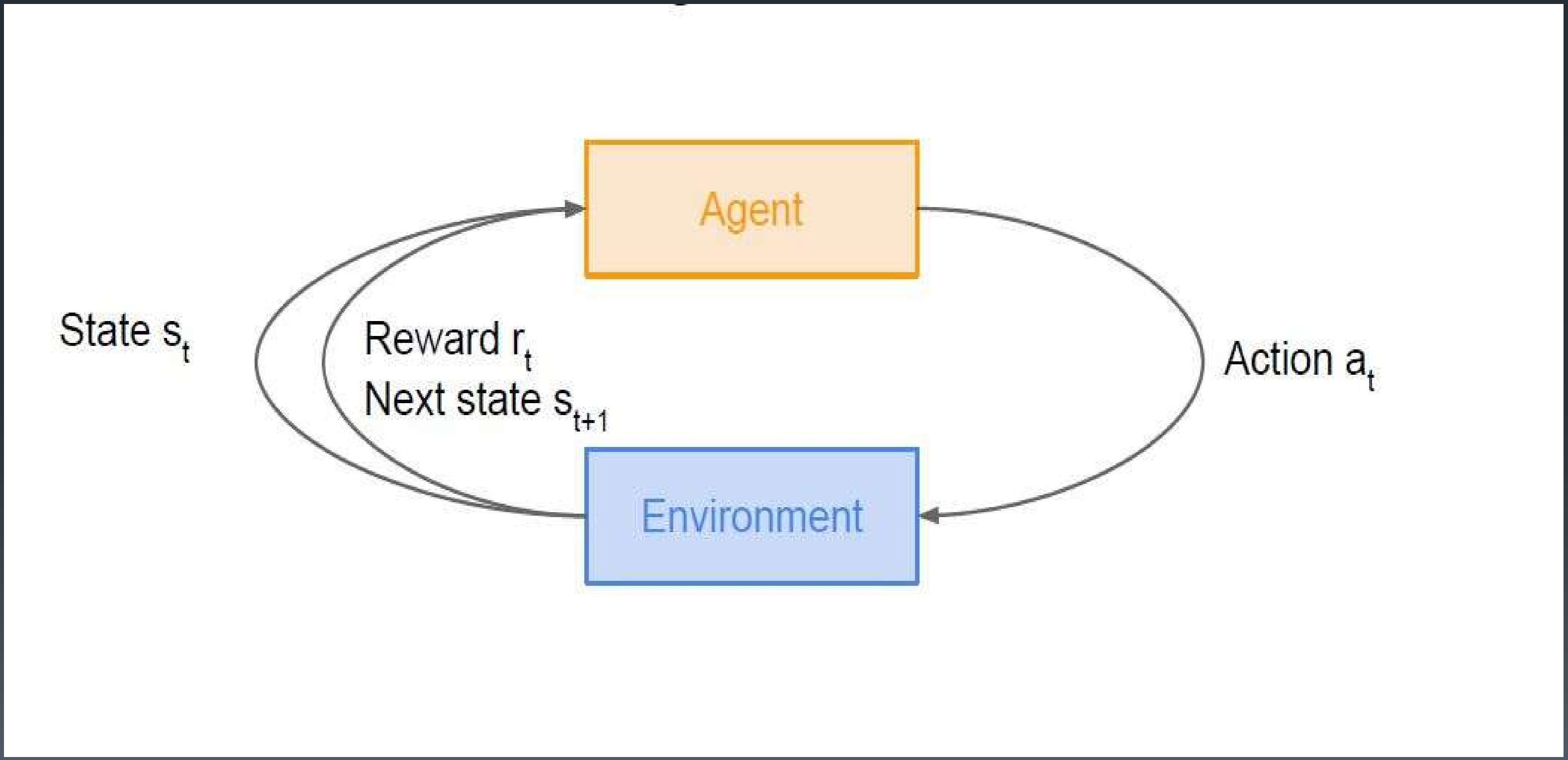
Introduction to Reinforcement Learning

- Agent receives a reward from the environment along with the next state



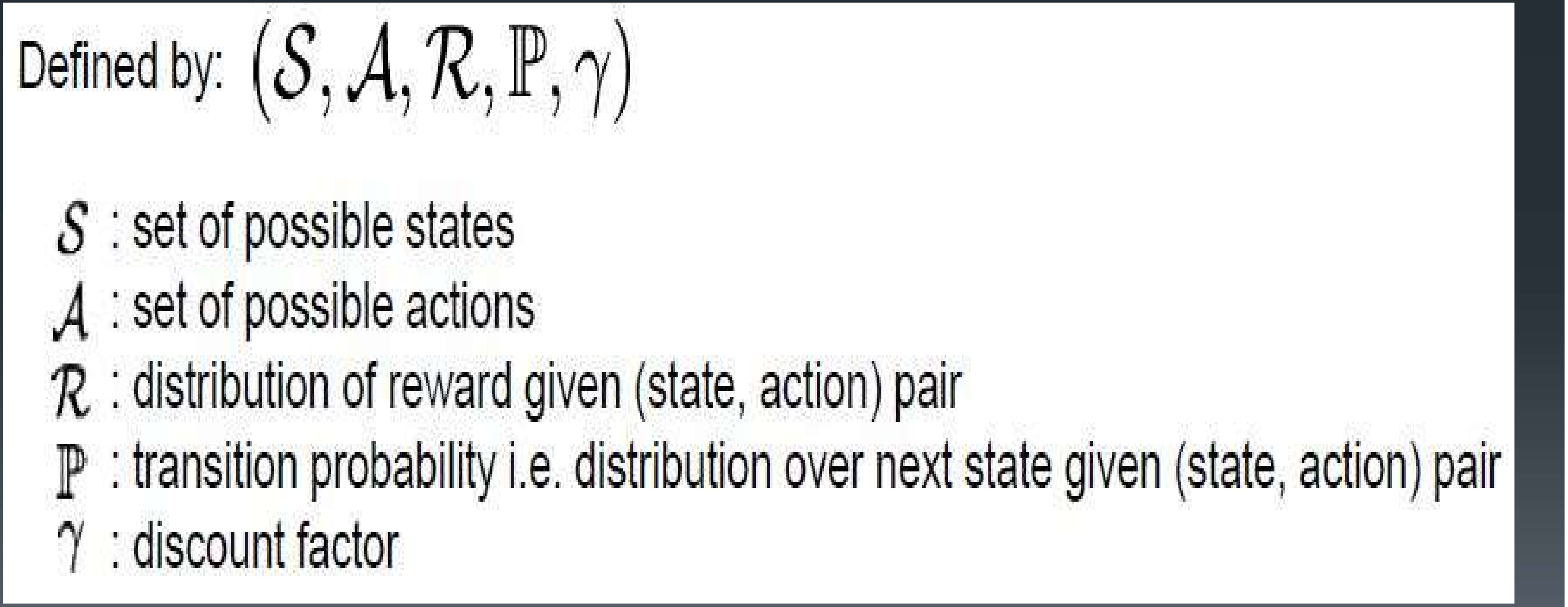
What is the objective here?

- The goal is to maximize the numerical reward signal.



How can we represent this mathematically?

- Markov Decision Process (MDP)
- Follows the Markovian Property – current state categorizes the entire state of world



Defined by: $(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)$

$\mathcal{S}$: set of possible states

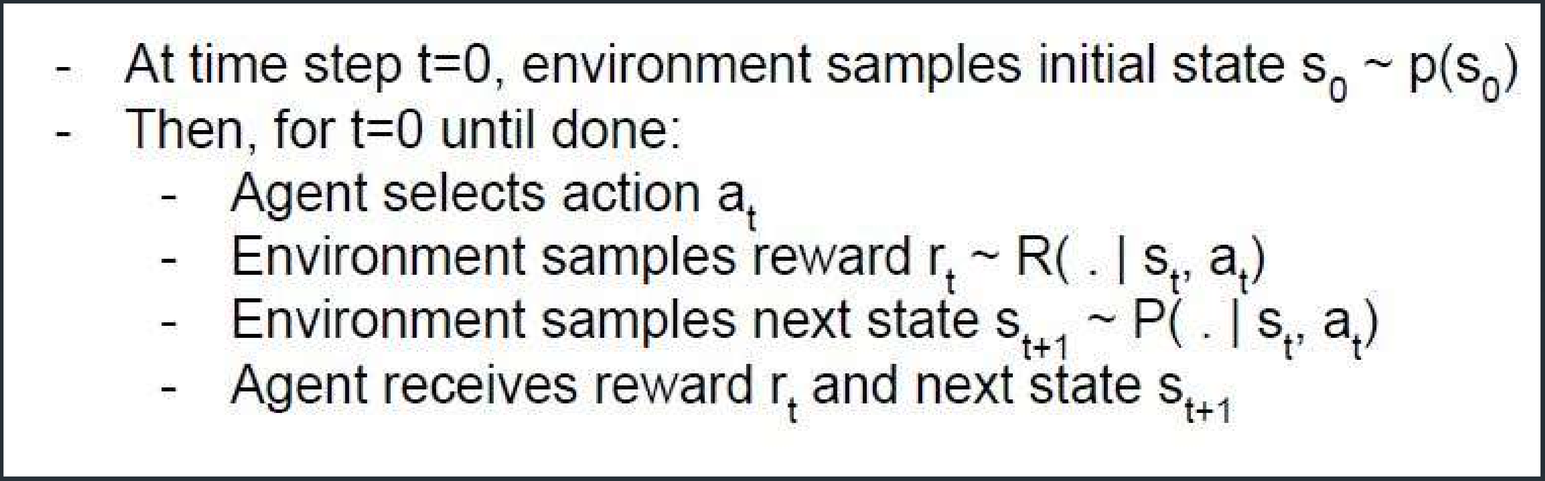
$\mathcal{A}$: set of possible actions

$\mathcal{R}$: distribution of reward given (state, action) pair

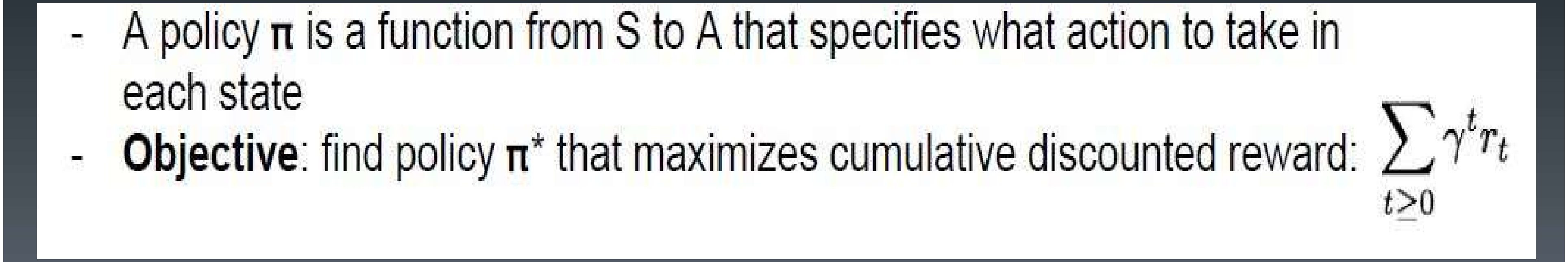
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair

γ : discount factor

Mathematical Formulation



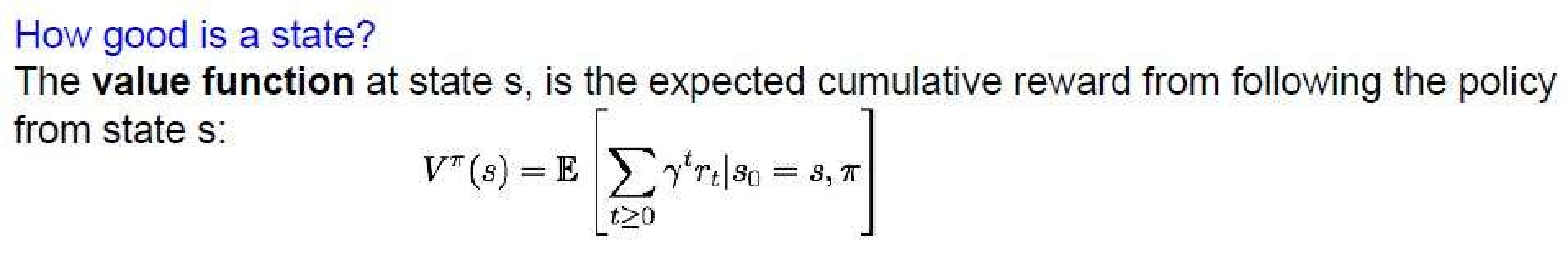
- At time step t=0, environment samples initial state $s_0 \sim p(s_0)$
- Then, for t=0 until done:
  - Agent selects action a_t
  - Environment samples reward $r_t \sim R(. \mid s_t, a_t)$
  - Environment samples next state $s_{t+1} \sim P(. \mid s_t, a_t)$
  - Agent receives reward r_t and next state s_{t+1}



- A policy **π** is a function from S to A that specifies what action to take in each state
- **Objective**: find policy **π*** that maximizes cumulative discounted reward: $\sum_{t \geq 0} \gamma^t r_t$

Two Function Metrics in RL

- Following a policy results in various trajectories
- Value Function

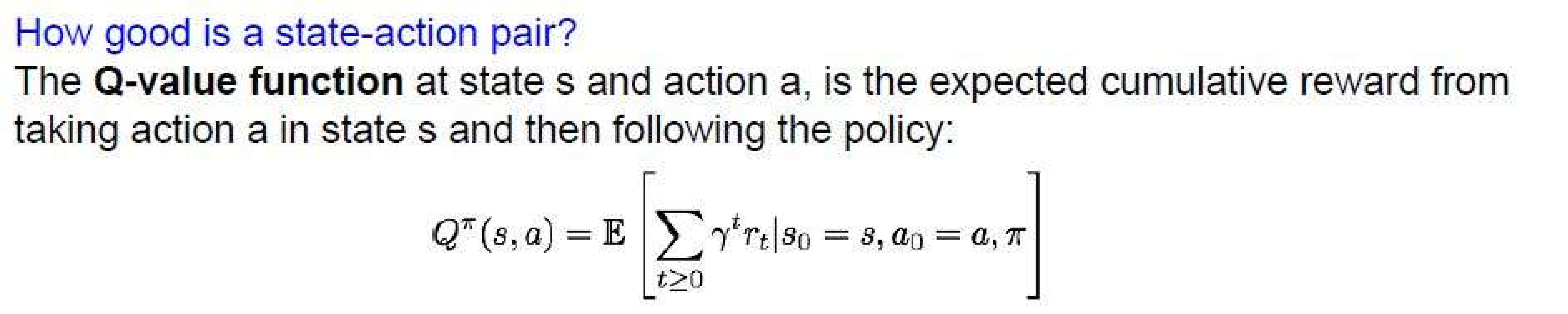


How good is a state?

The **value function** at state s, is the expected cumulative reward from following the policy from state s:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

- Q-value Function

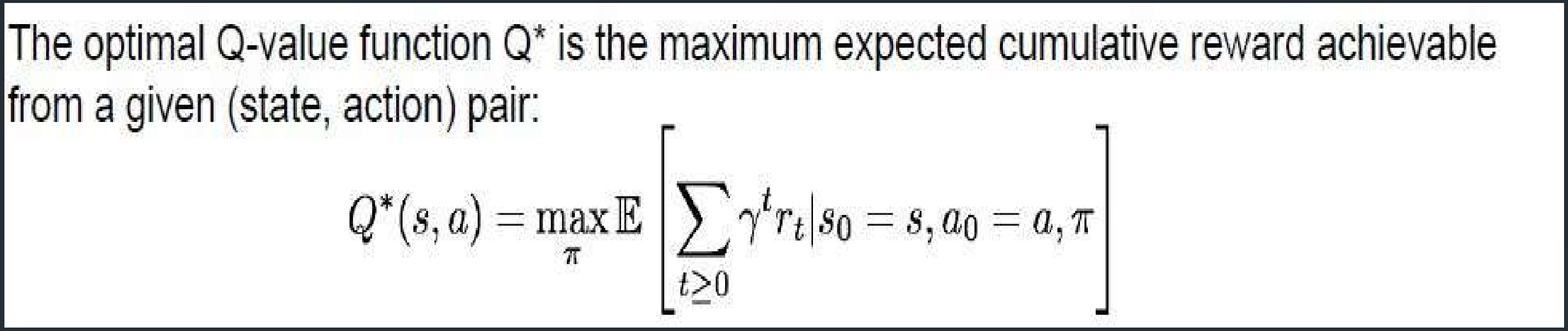


How good is a state-action pair?

The **Q-value function** at state s and action a, is the expected cumulative reward from taking action a in state s and then following the policy:

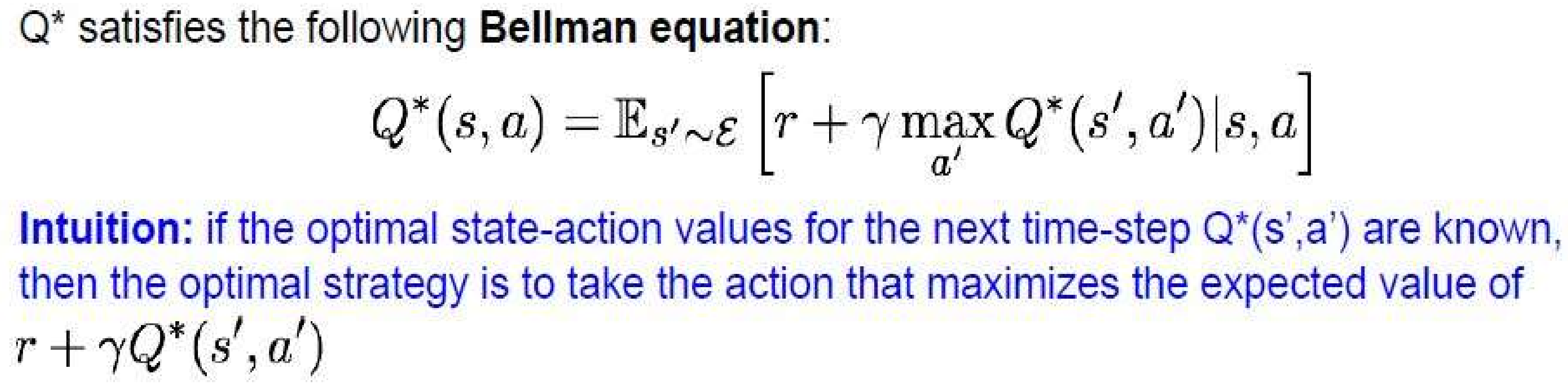
$$Q^{\pi}(s,a) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Solving for Optimal Policy and Bellman Equation



The optimal Q-value function Q* is the maximum expected cumulative reward achievable from a given (state, action) pair:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

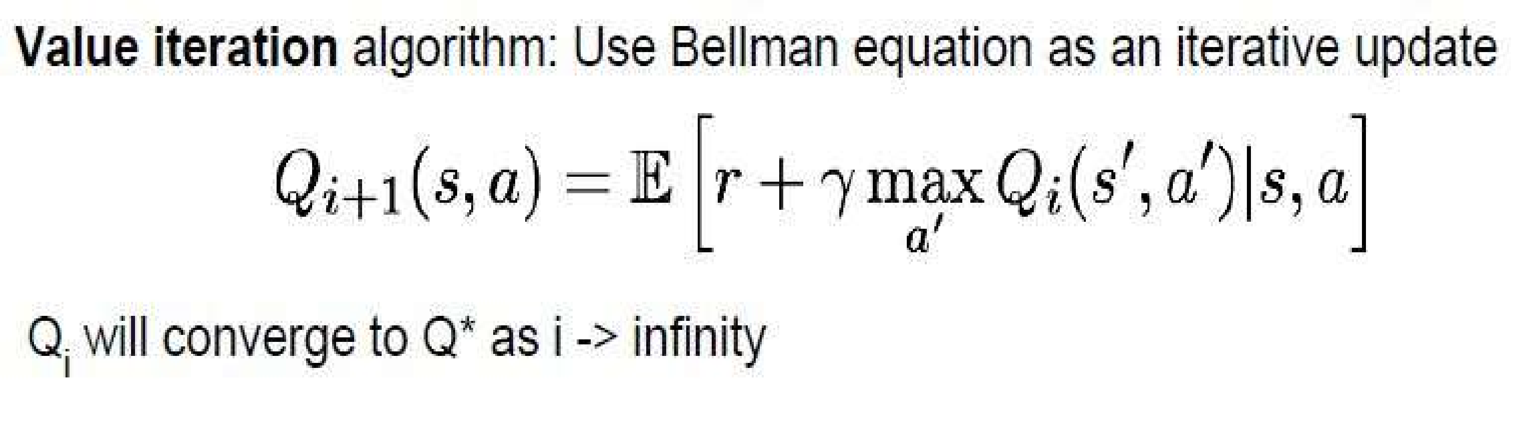


Q* satisfies the following **Bellman equation**:

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}}\left[r + \gamma \max_{a'} Q^*(s',a') | s, a\right]$$

Intuition: if the optimal state-action values for the next time-step Q*(s',a') are known, then the optimal strategy is to take the action that maximizes the expected value of $r + \gamma Q^*(s',a')$

Solving for Optimal Policy: Value-Iteration Algorithm



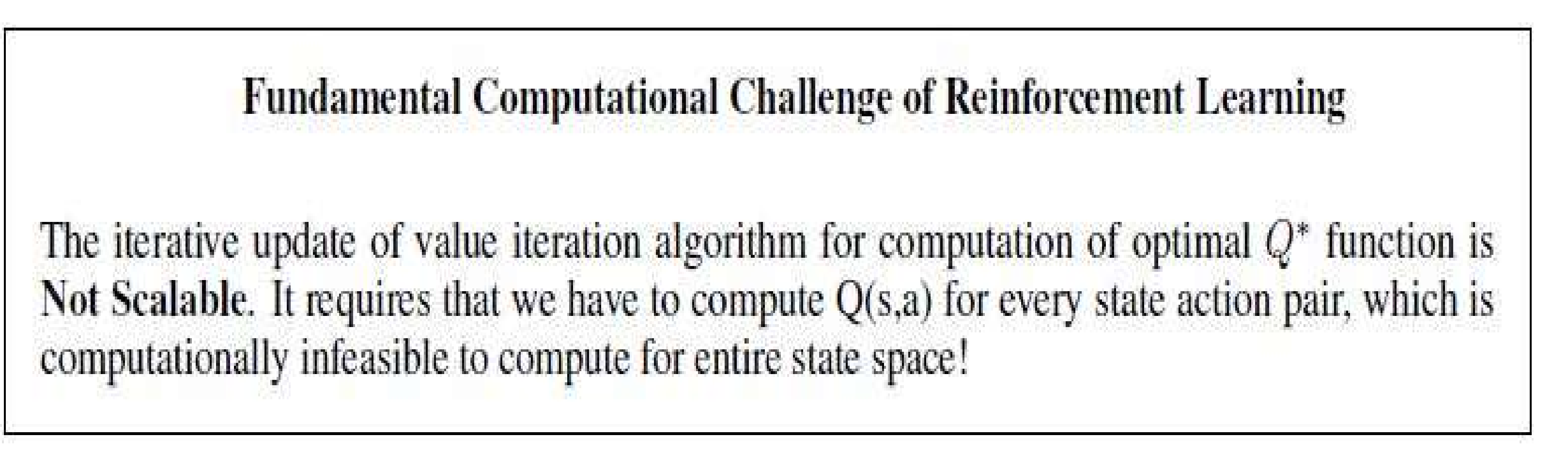
Value iteration algorithm: Use Bellman equation as an iterative update

$$Q_{i+1}(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q_i(s',a') | s, a \right]$$

Q_i will converge to Q^* as i -> infinity

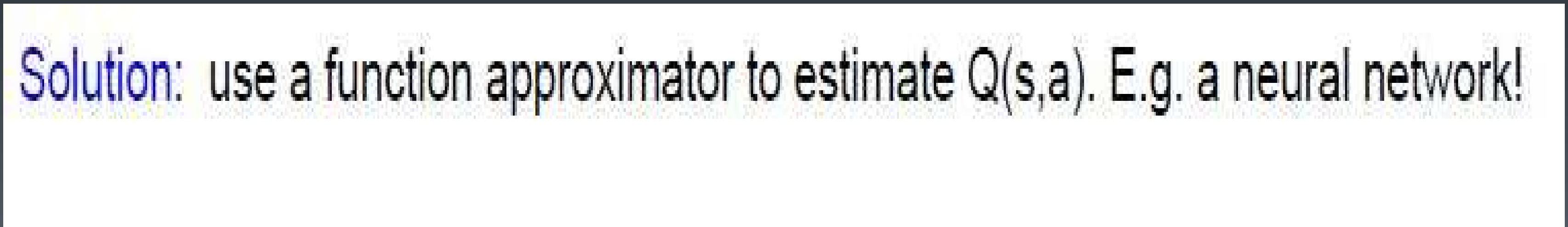
- Brings us to the fundamental computational challenge of Reinforcement Learning

Fundamental Computational Challenge of Reinforcement Learning



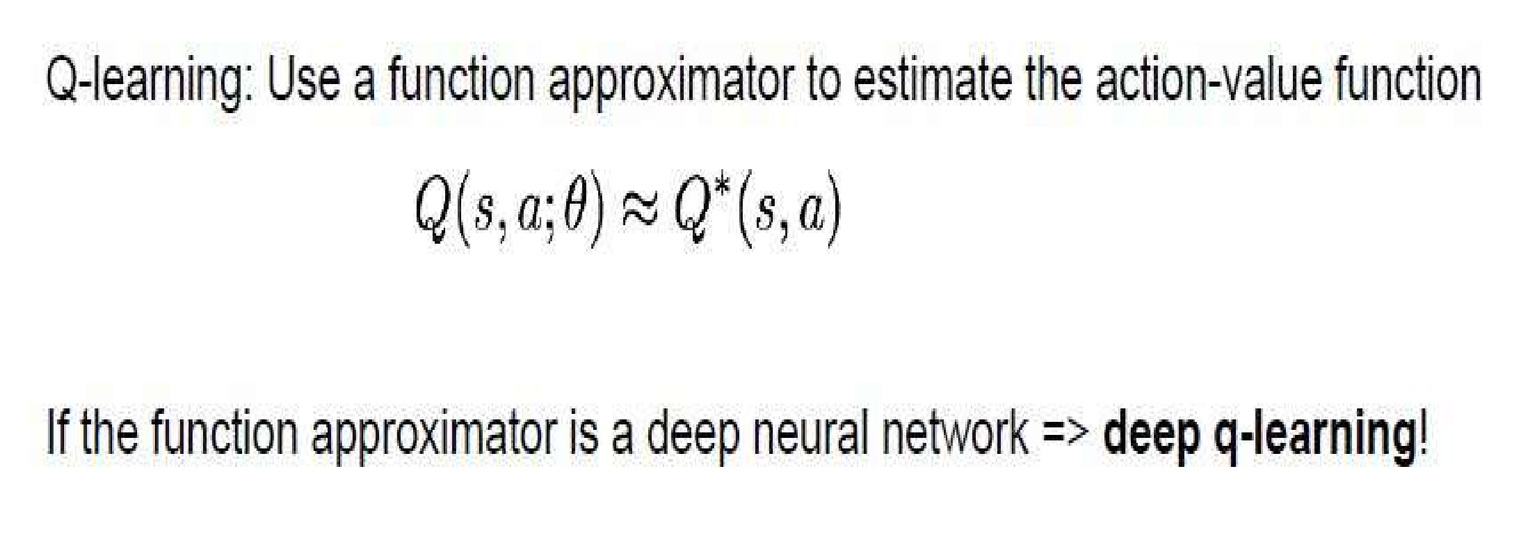
Fundamental Computational Challenge of Reinforcement Learning

The iterative update of value iteration algorithm for computation of optimal Q^* function is **Not Scalable**. It requires that we have to compute Q(s,a) for every state action pair, which is computationally infeasible to compute for entire state space!



Solution: use a function approximator to estimate Q(s,a). E.g. a neural network!

Q-Learning and Deep Q-Learning

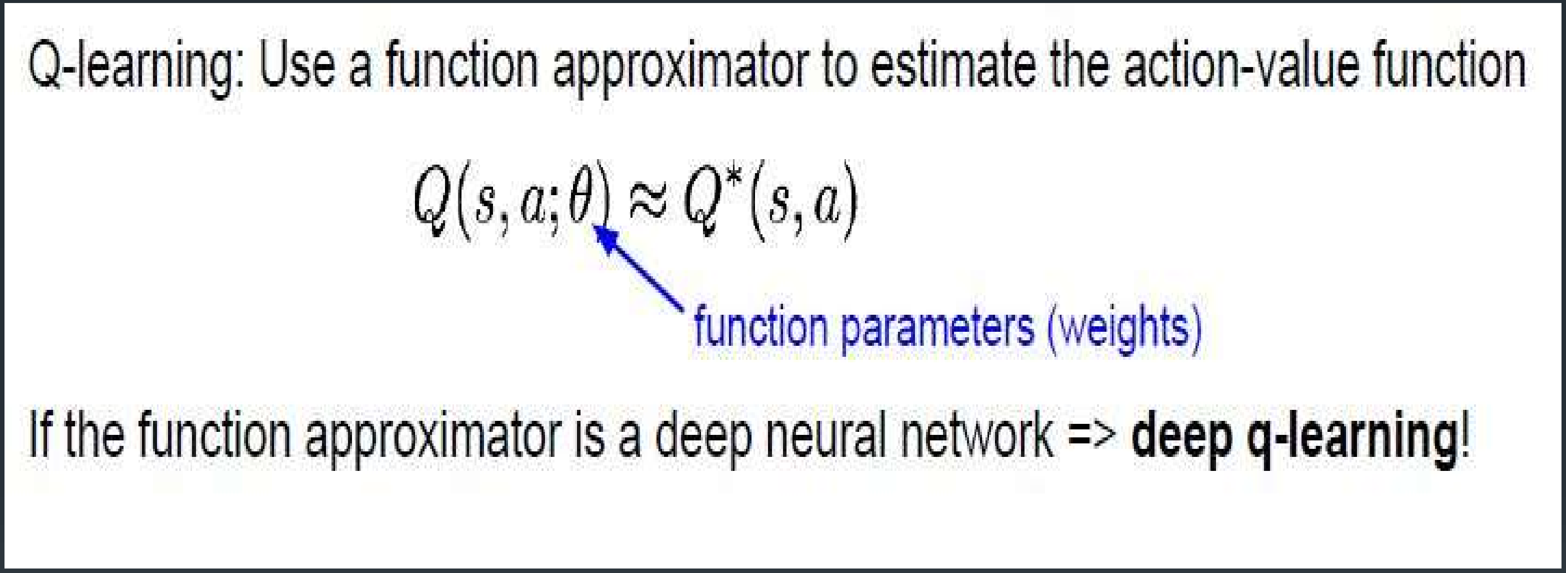


Q-learning: Use a function approximator to estimate the action-value function

$$Q(s,a;\theta) \approx Q^*(s,a)$$

If the function approximator is a deep neural network => **deep q-learning**!

Q-Learning and Deep Q-Learning



Q-learning: Use a function approximator to estimate the action-value function

$$Q(s,a;\theta) \approx Q^*(s,a)$$

function parameters (weights)

If the function approximator is a deep neural network => **deep q-learning**!

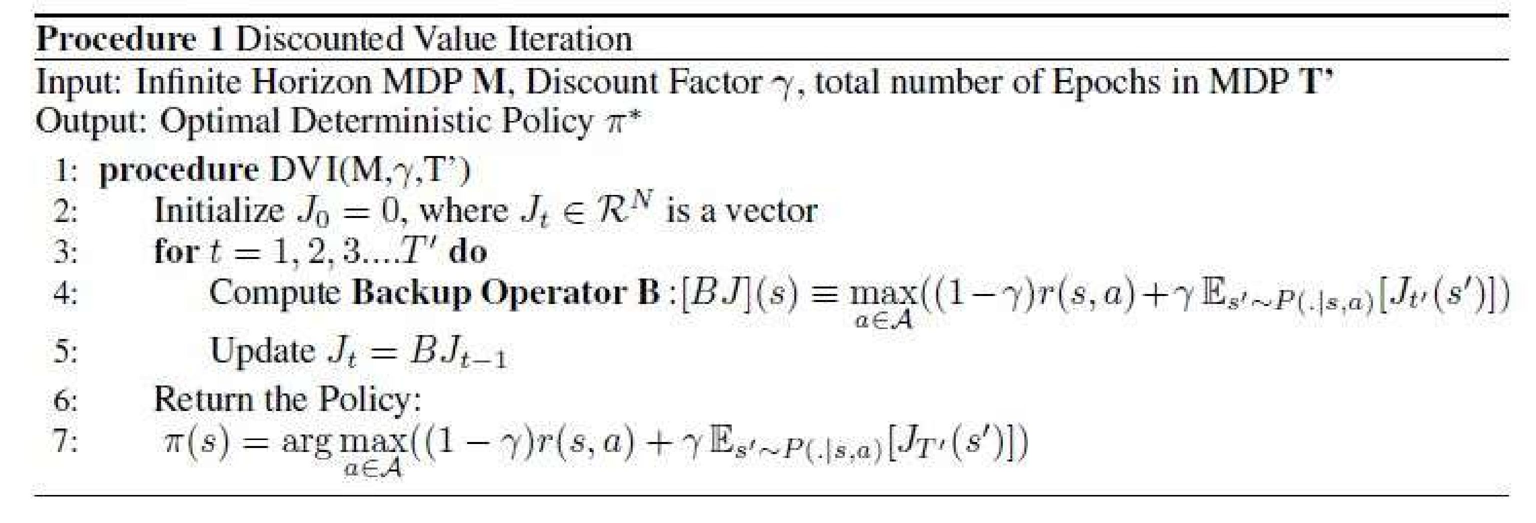
Alternative Approaches for Achieving Optimality Criteria

- In Reinforcement Learning, we care about the maximum cumulative reward
- Q-Learning is one of the ways in which we could achieve the optimality criteria of RL, sure!
- However, it is still worth to investigate how well we can do using the Value Function
- The optimal value function is defined by: (M is a MDP, π is the given policy):

$$V^{*}_{\gamma,M}(s) = \sup_{\pi \in \Pi} V_{\pi,\gamma,M}(s)$$

- This value function also has to satisfy the Bellman Equation

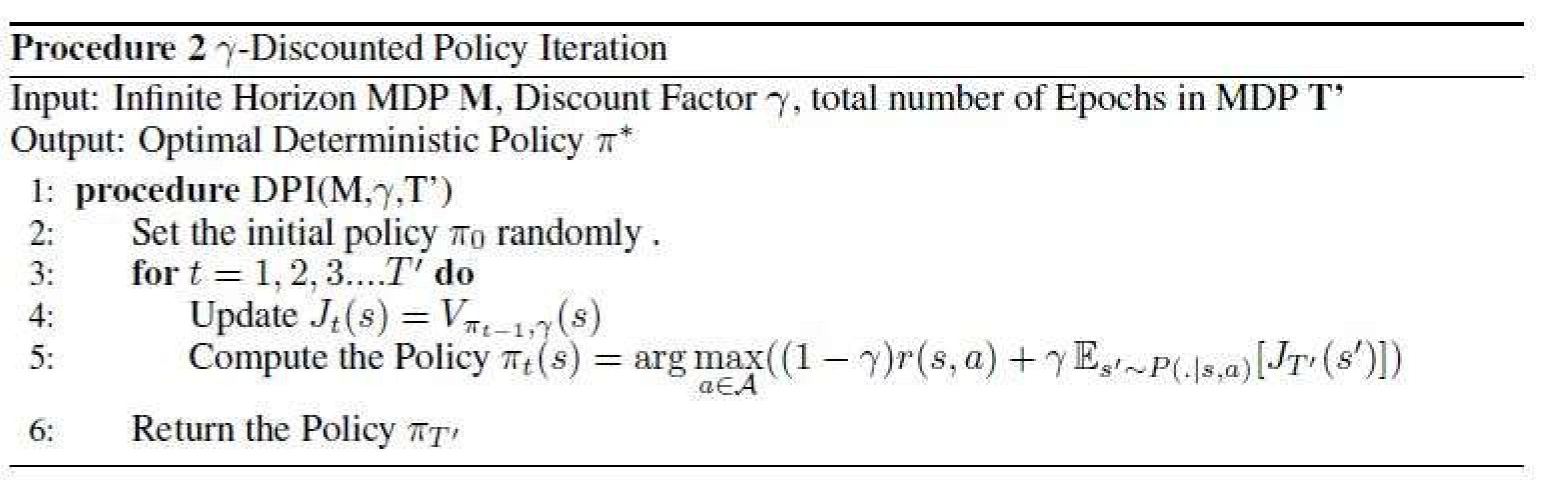
Value Iteration Algorithms (alternate version)



Procedure 1 Discounted Value Iteration

Input: Infinite Horizon MDP **M**, Discount Factor γ, total number of Epochs in MDP **T'**
Output: Optimal Deterministic Policy π^*

```
procedure DVI(M,$\gamma$,T')
    Initialize $J_0 = 0$, where $J_t \in \mathcal{R}^N$ is a vector
    for $t = 1, 2, 3....T'$ do
        Compute Backup Operator B : $[BJ](s) \equiv \max_{a \in \mathcal{A}}((1-\gamma)r(s,a) + \gamma\,\mathbb{E}_{s' \sim P(.|s,a)}[J_{t'}(s')])$
        Update $J_t = BJ_{t-1}$
    Return the Policy:
     $\pi(s) = \arg\max_{a \in \mathcal{A}}((1-\gamma)r(s,a) + \gamma\,\mathbb{E}_{s' \sim P(.|s,a)}[J_{T'}(s')])$
```



Procedure 2 γ-Discounted Policy Iteration

Input: Infinite Horizon MDP **M**, Discount Factor γ, total number of Epochs in MDP **T'**
Output: Optimal Deterministic Policy π^*

```
procedure DPI(M,$\gamma$,T')
    Set the initial policy $\pi_0$ randomly .
    for $t = 1, 2, 3....T'$ do
        Update $J_t(s) = V_{\pi_{t-1},\gamma}(s)$
        Compute the Policy $\pi_t(s) = \arg\max_{a \in \mathcal{A}}((1-\gamma)r(s,a) + \gamma\,\mathbb{E}_{s' \sim P(.|s,a)}[J_{T'}(s')])$
    Return the Policy $\pi_{T'}$
```

Convergence Analysis

- For Discounted Value Iteration Algorithm, we have:

$$V_{\pi,\gamma}(s) \geq V_\gamma^* - 2\gamma^{T'}$$

- For γ-Discounted Policy Iteration Algorithm, we have:

$$||V_{\pi_{T'},\gamma} - V_\gamma^*||_\infty \leq \gamma^{T'}$$

- Selection of number of epochs in MDP T’:

$$T' = \log_\gamma \epsilon = O(\frac{-\log \epsilon}{1-\gamma})$$

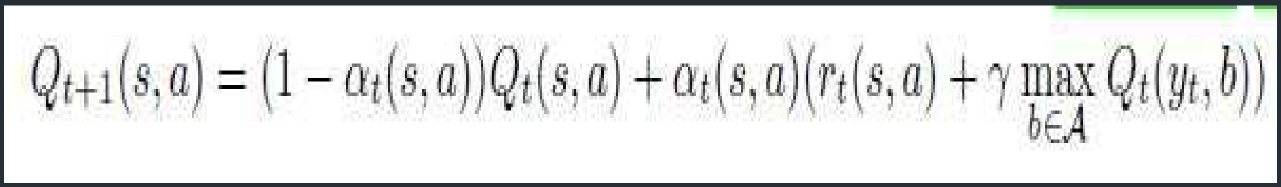
- Proof Idea:
- Apply contraction property between vectors
- Consider 1st and last epoch
- Rearrange the terms

Coming Back to Q-Learning

- In our project, we will not focus on achieving the optimality criteria using the Value Function.
- We will instead use the Q-Value Function, since $Q(s, a)$ gives a better idea about the reward with respect to state and action space as it considers both the parameters ‘s’ and ‘a’.
- Now, we will focus on the asymptotic convergence rate of Q-Learning.

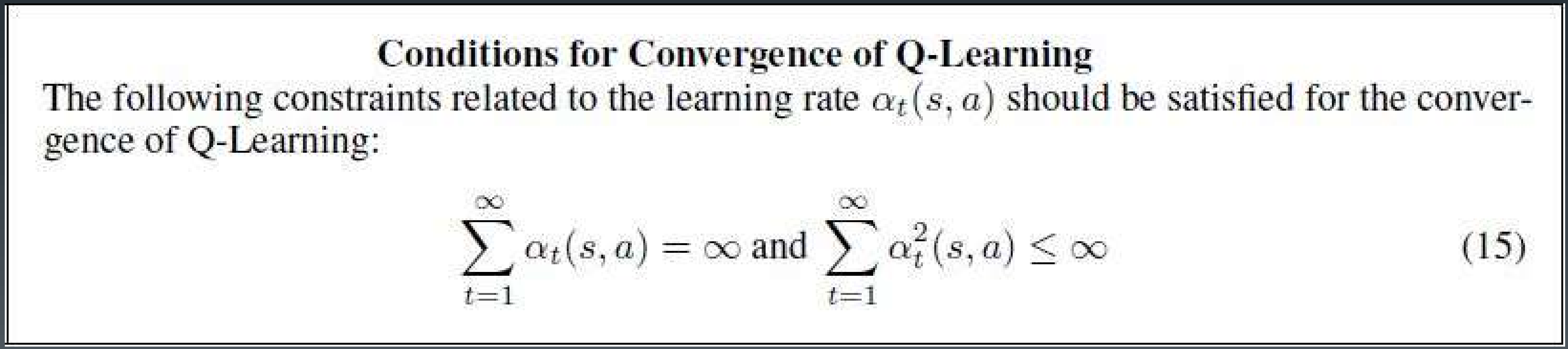
Asymptotic Convergence Rate of Q-Learning

- For the estimation of optimal Q-Value function, we need to incorporate the concept of learning rate in update of Q.



$$Q_{t+1}(s,a) = (1 - \alpha_t(s,a))Q_t(s,a) + \alpha_t(s,a)(r_t(s,a) + \gamma \max_{b \in A} Q_t(y_t,b))$$

- For the convergence of Q-Learning, we need to assume that:



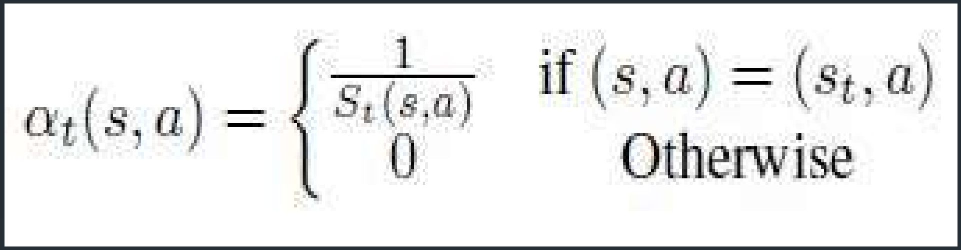
Conditions for Convergence of Q-Learning

The following constraints related to the learning rate $\alpha_t(s,a)$ should be satisfied for the convergence of Q-Learning:

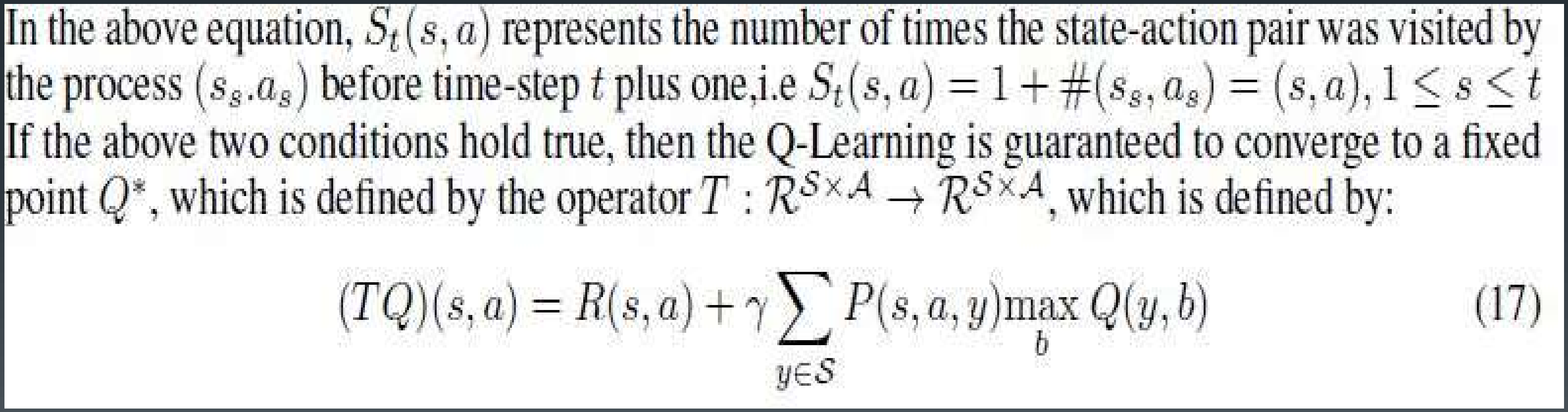
$$\sum_{t=1}^{\infty} \alpha_t(s,a) = \infty \text{ and } \sum_{t=1}^{\infty} \alpha_t^2(s,a) \leq \infty \qquad (15)$$

More Conditions for Convergence

- Learning Rate has to be of this specific form:



$$\alpha_t(s,a) = \begin{cases} \frac{1}{S_t(s,a)} & \text{if } (s,a) = (s_t,a) \\ 0 & \text{Otherwise} \end{cases}$$

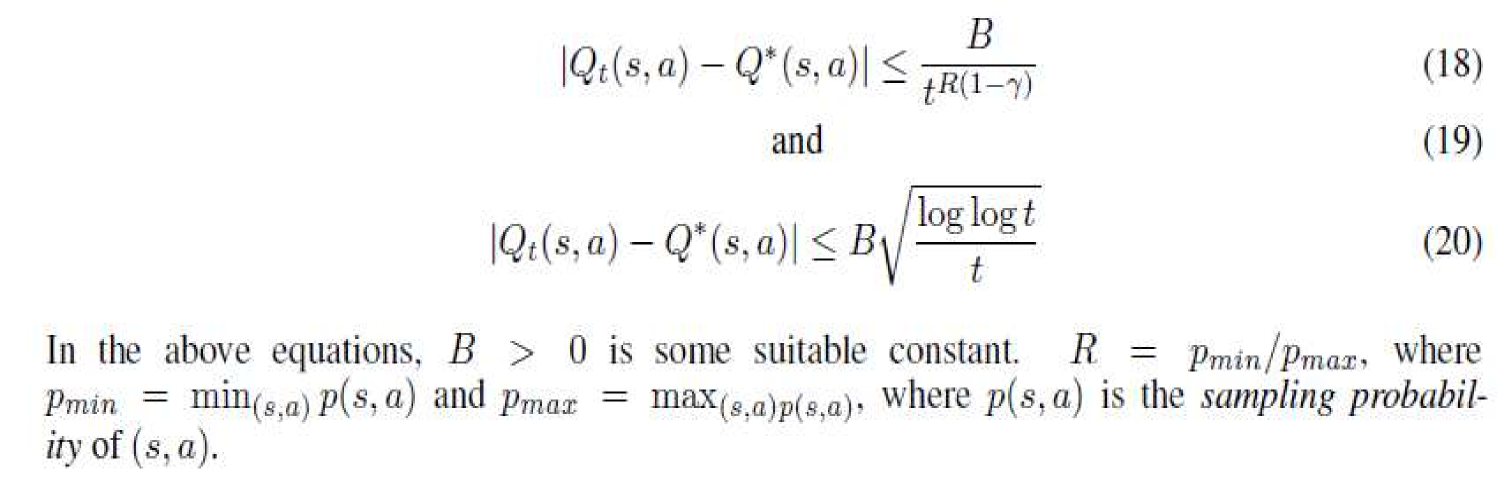


In the above equation, $S_t(s,a)$ represents the number of times the state-action pair was visited by the process $(s_s.a_s)$ before time-step t plus one,i.e $S_t(s,a) = 1 + \#(s_s,a_s) = (s,a), 1 \leq s \leq t$ If the above two conditions hold true, then the Q-Learning is guaranteed to converge to a fixed point Q^*, which is defined by the operator $T : \mathcal{R}^{\mathcal{S}\times\mathcal{A}} \rightarrow \mathcal{R}^{\mathcal{S}\times\mathcal{A}}$, which is defined by:

$$(TQ)(s,a) = R(s,a) + \gamma \sum_{y \in \mathcal{S}} P(s,a,y) \max_b Q(y,b) \qquad (17)$$

Convergence Rate of Q-Learning

- If all the prior conditions holds true, then we have the following asymptotic convergence rate with probability 1:



$$|Q_t(s,a) - Q^*(s,a)| \leq \frac{B}{t^{R(1-\gamma)}} \tag{18}$$

and (19)

$$|Q_t(s,a) - Q^*(s,a)| \leq B\sqrt{\frac{\log\log t}{t}} \tag{20}$$

In the above equations, $B > 0$ is some suitable constant. $R = p_{min}/p_{max}$, where $p_{min} = \min_{(s,a)} p(s,a)$ and $p_{max} = \max_{(s,a)p(s,a)}$, where $p(s,a)$ is the *sampling probability* of (s,a).

- Proof Idea:
- Define a simpler process $\hat{Q}_t$, which will converge to Q^*
- Take the difference between both the processes and show that it is converging to 0
- Use law of iterated logarithm

Learning Rate Trick for Convergence Analysis of Q Learning

- For time t, in case of polynomial learning rate 1/(t^ω), where ω ∈ (½,1), the convergence rate is polynomial in 1/(1-γ), where γ is the discount factor.
- In case of linear learning rate 1/t, the convergence rate is exponential in 1/(1-γ).

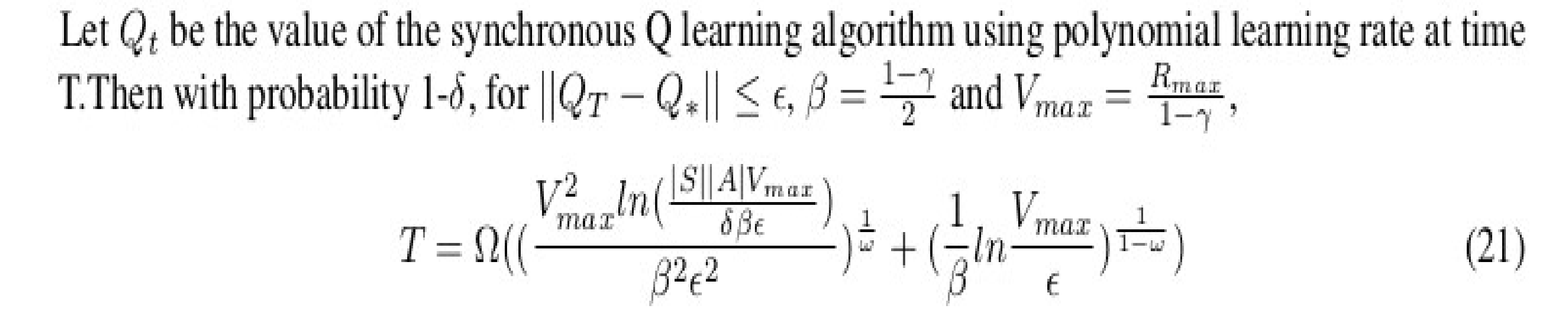
Intuition behind the Learning Rate Trick for Convergence Analysis of Q Learning

•The different behaviour might be explained by the asymptotic behaviour of $\Sigma_t \alpha_t$, one of the conditions that ensure that Q-learning converges from any initial value.

•In the case of a linear learning rate, we have that $\Sigma_t \alpha_t$ = $O(\ln(T))$, whereas using polynomial learning rate it behaves as $O(T$^(1−w)).

•Therefore, using polynomial learning rate each value can be reached by polynomial number of steps and using linear learning rate each value requires exponential number of steps.

Convergence Analysis of Synchronous Q Learning with Polynomial Learning Rate



Let Q_t be the value of the synchronous Q learning algorithm using polynomial learning rate at time T.Then with probability 1-δ, for $||Q_T - Q_*|| \leq \epsilon$, $\beta = \frac{1-\gamma}{2}$ and $V_{max} = \frac{R_{max}}{1-\gamma}$,

$$T = \Omega((\frac{V_{max}^2 ln(\frac{|S||A|V_{max}}{\delta \beta \epsilon})}{\beta^2 \epsilon^2})^{\frac{1}{\omega}} + (\frac{1}{\beta} ln \frac{V_{max}}{\epsilon})^{\frac{1}{1-\omega}}) \qquad (21)$$

✓Proof Intuition:assume that ω is a constant and consider first only its dependence on ϵ.

✓This gives us Ω((ln(1/*e)/e^2)^(1/ ω) +(ln(1/e))^(1/(1− ω)), which* is optimized when ω approaches one.

✓Considering the dependence only on β, recall that *V_{max} =R_{max}/(2* β*), therefore the complexity is* Ω *(1/*β*^(*4/ω*)* +*1/*β*^(*1/(1− ω*)))* *which is optimized for* ω *= 4/5.*

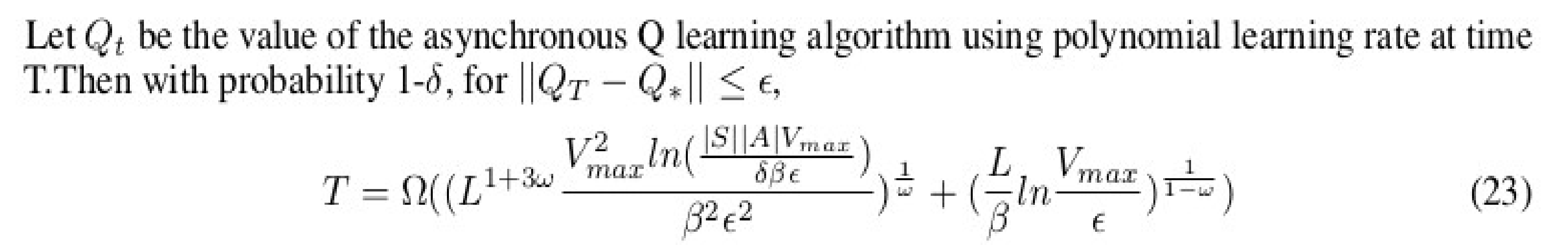
Convergence Analysis for Asynchronous Q Learning

- Difference between Asynchronous and synchronous Q Learning:
  - Asynchronous Learning updates the value for only one state action pair at a time-step
  - Synchronous Learning updates the value for all state action pairs in a single time-step.
- Concept of covering time has been introduced to find the convergence rate for asynchronous Q Learning.

Covering Time

- Each iteration of an algorithm having a covering time L conveys the meaning that, from any start state s, in L time-steps, all state action pairs are executed.
- For the theoretical convergence analysis of Q Learning, the following assumptions have been considered.
    - The condition of covering time occurs with probability ½.
    - With high probability the covering time is LlogT over T iterations.
    - The sequence of state-action pairs can be arbitrary.

Modification of the Convergence Rate with Covering Time for Asynchronous Q Learning



Let Q_t be the value of the asynchronous Q learning algorithm using polynomial learning rate at time T.Then with probability 1-δ, for $||Q_T - Q_*|| \leq \epsilon$,

$$T = \Omega((L^{1+3\omega}\frac{V_{max}^2 ln(\frac{|S||A|V_{max}}{\delta\beta\epsilon})}{\beta^2\epsilon^2})^{\frac{1}{\omega}} + (\frac{L}{\beta}ln\frac{V_{max}}{\epsilon})^{\frac{1}{1-\omega}}) \quad (23)$$

Proof Intuition: The difference with synchronous learning is the introduction of covering time. The dependence on the covering time, in the above theorem, is Ω(*L^(2+1/ω)+L^(1/(1− ω))), which is* optimized for *ω≈0.77*.

Convergence Analysis of Deep Q Learning

Challenges for Theoretical Analysis of Deep Q Learning (1)

- Approximation of the action value function can make the Deep Q Learning unstable. Correlated samples can lead to in-efficient learning.
- Experience replay is adopted to address the issue.
- A table (replay memory) continually stores the transitions (s_t,a_t,r_t,s_{t+1}) of the MDP
- At each iteration of Deep Q Learning, the transitions are sampled randomly from the replay memory to train the Q network, approximating the Q function.
- Stability is achieved by breaking the temporal dependency among the transitions.

Challenges for Theoretical Analysis of Deep Q Learning (2)

- A target network is used along with the Q network to ensure unbiased estimation of the Q function.
- The Q networks and the target network have to be synchronized after a set of iterations for Deep Q Learning.
- This makes the theoretical analysis of Deep Q Learning to be completed.

Stability in Deep Q Learning – Experience Replay

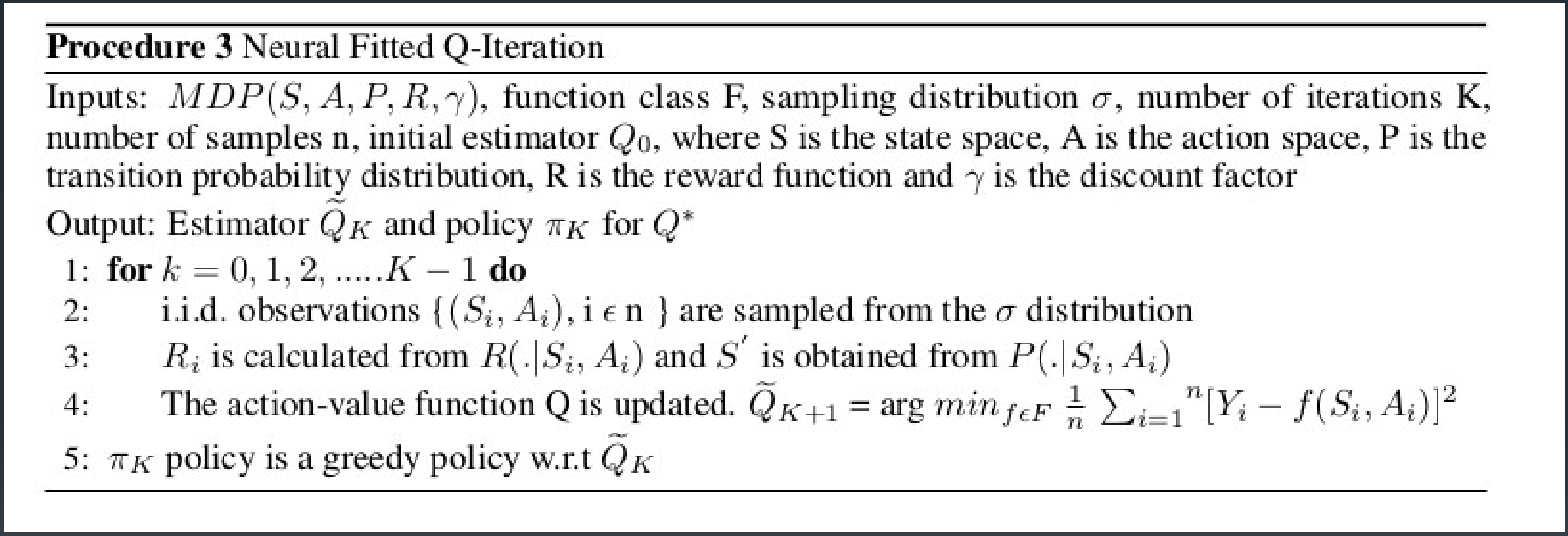
➢Reinforcement learning is unstable when general non-linear or even linear function approximators are directly implemented.

➢Experience replay is used with DQN to stabilize Deep Q Learning by removal of the temporal dependency of observations that are used while training the neural network.

➢For the purpose of theoretical investigation of DQN, experience replay is simplified with an independence assumption that independent and identically distributed (i.i.d) observations (S_i;A_i) iϵ[n] are sampled from a fixed distribution σ(SxA).

➢Deep neural network with rectified linear units (ReLU) and large batch size has been used as the function approximator.

Simplified Q Learning Algorithm (NFQI) for Convergence Analysis



Procedure 3 Neural Fitted Q-Iteration

Inputs: $MDP(S, A, P, R, \gamma)$, function class F, sampling distribution σ, number of iterations K, number of samples n, initial estimator Q_0, where S is the state space, A is the action space, P is the transition probability distribution, R is the reward function and γ is the discount factor

Output: Estimator $\widetilde{Q}_K$ and policy π_K for Q^*

```
for $k = 0, 1, 2, .....K - 1$ do
    i.i.d. observations $\{(S_i, A_i)$, i $\epsilon$ n $\}$ are sampled from the $\sigma$ distribution
    $R_i$ is calculated from $R(.|S_i, A_i)$ and $S^{'}$ is obtained from $P(.|S_i, A_i)$
    The action-value function Q is updated. $\widetilde{Q}_{K+1}$ = arg $min_{f \epsilon F}$ $\frac{1}{n} \sum_{i=1}{}^{n}[Y_i - f(S_i, A_i)]^2$
$\pi_K$ policy is a greedy policy w.r.t $\widetilde{Q}_K$
```

Intuition of Convergence Rate for Deep Q Learning

- The statistical rate of convergence for Deep Q Learning is the sum of an statistical error and an algorithmic error.

- The statistical error represents the bias and the variance due to approximation of the action value function with the neural network which gives an idea about the difficulty of the problem.

- When the number of iterations is sufficiently large, the algorithmic error is over-shadowed by the statistical error.

Sample Efficiency of Deep Q Learning

- In simple situation of i.i.d data X_i where $S_n=\sum_1^n X_i$ and μ is the expectation on X_i, Chebyshev's inequality can be applied.

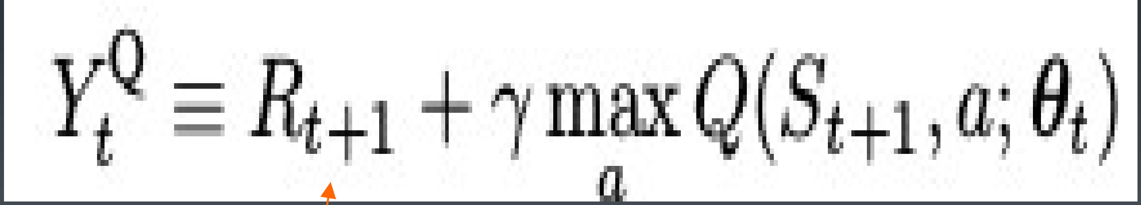
$$\forall \epsilon > 0, P(|\frac{S_n}{n} - \mu| > \epsilon) \leq \frac{\sigma}{n\epsilon^2}$$

- It shows that even in simple situations, lots of data are needed to achieve convergence in the expectation.
- The convergence rate depends on the variance of the dataset.
- In practice, the datasets are normally not independent and identically distributed.

Over-Estimation Challenge in Deep Q Learning

- Deep Q Learning suffers from the problem of over-estimation of the action value parameters under certain conditions

- Sometimes extremely high action values Q are learned due to a maximization step over estimated action values while updating the objective function, thus being biased to over-estimated action values.



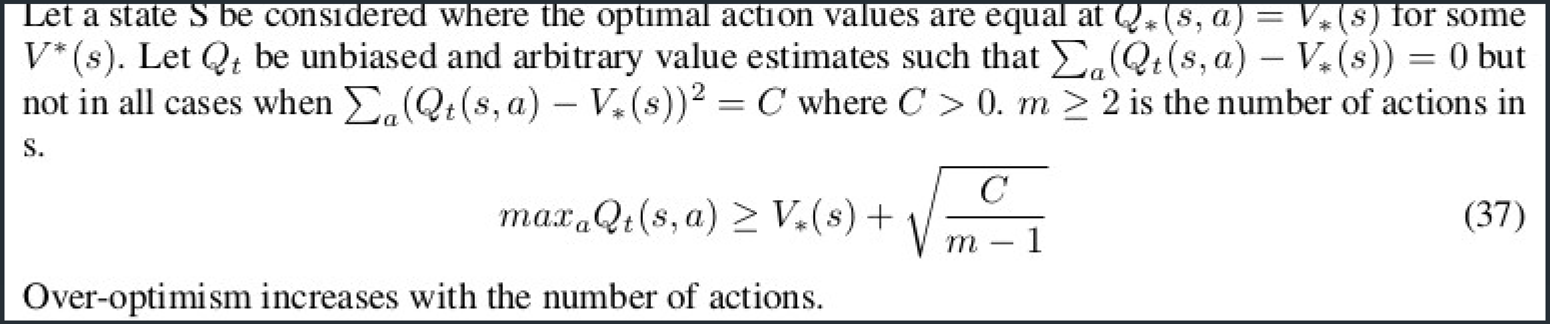
$$Y_t^{\mathrm{Q}} \equiv R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \theta_t)$$

Intermediate Reward

Causes and Effects of Over-Estimation

- Causes:
  - Inaccurate action values irrespective of the source of approximation error
  - Inflexible function approximation
  - Noise
- Effects:
  - If Q value is over-estimated, it will be difficult to achieve convergence with an optimal rate
  - Quality of the resulting policy is negatively impacted if
    - Over-estimations are not uniform
    - Over-estimations are not concentrated across the states

Lower Bound of Over-Estimation in Deep Q Learning



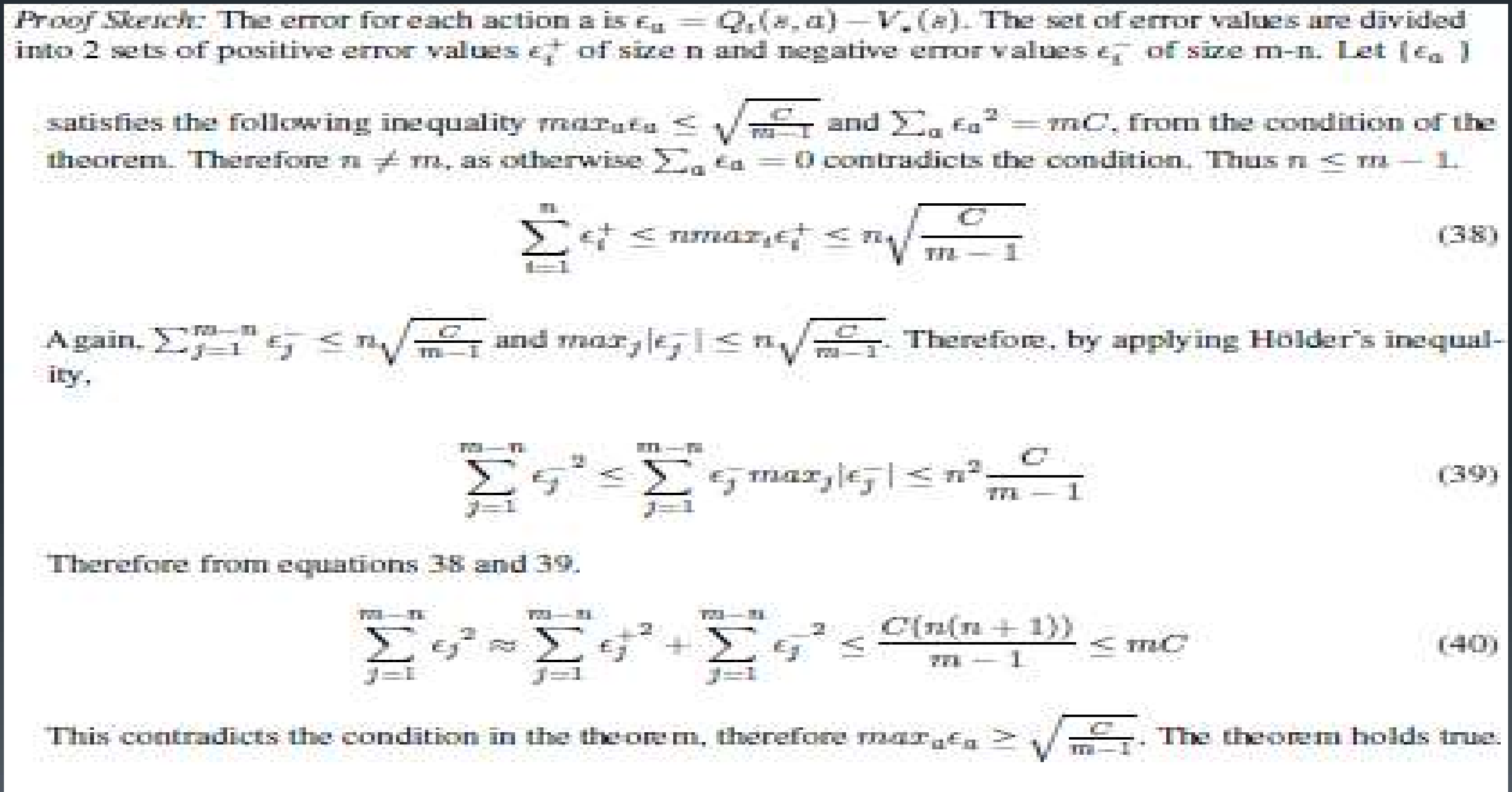
Let a state S be considered where the optimal action values are equal at $Q_*(s,a) = V_*(s)$ for some $V^*(s)$. Let Q_t be unbiased and arbitrary value estimates such that $\sum_a (Q_t(s,a) - V_*(s)) = 0$ but not in all cases when $\sum_a (Q_t(s,a) - V_*(s))^2 = C$ where $C > 0$. $m \geq 2$ is the number of actions in s.

$$max_a Q_t(s,a) \geq V_*(s) + \sqrt{\frac{C}{m-1}} \tag{37}$$

Over-optimism increases with the number of actions.

Proof Intuition: Proof by Contradiction

Proof Sketch



Proof Sketch: The error for each action a is $\epsilon_a = Q_t(s,a) - V_*(s)$. The set of error values are divided into 2 sets of positive error values ϵ_i^+ of size n and negative error values ϵ_i^- of size m-n. Let $\{\epsilon_a\}$

satisfies the following inequality $max_a \epsilon_a \leq \sqrt{\frac{C}{m-1}}$ and $\sum_a {\epsilon_a}^2 = mC$, from the condition of the theorem. Therefore $n \neq m$, as otherwise $\sum_a \epsilon_a = 0$ contradicts the condition. Thus $n \leq m-1$.

$$\sum_{i=1}^{n} \epsilon_i^+ \leq n max_i \epsilon_i^+ \leq n\sqrt{\frac{C}{m-1}} \tag{38}$$

Again, $\sum_{j=1}^{m-n} \epsilon_j^- \leq n\sqrt{\frac{C}{m-1}}$ and $max_j |\epsilon_j^-| \leq n\sqrt{\frac{C}{m-1}}$. Therefore, by applying Hölder's inequality,

$$\sum_{j=1}^{m-n} {\epsilon_j^-}^2 \leq \sum_{j=1}^{m-n} \epsilon_j^- max_j |\epsilon_j^-| \leq n^2 \frac{C}{m-1} \tag{39}$$

Therefore from equations 38 and 39.

$$\sum_{j=1}^{m-n} {\epsilon_j}^2 \approx \sum_{j=1}^{m-n} {\epsilon_j^+}^2 + \sum_{j=1}^{m-n} {\epsilon_j^-}^2 \leq \frac{C(n(n+1))}{m-1} \leq mC \tag{40}$$

This contradicts the condition in the theorem, therefore $max_a \epsilon_a \geq \sqrt{\frac{C}{m-1}}$. The theorem holds true.

Double Q Learning

✓Over-estimation is reduced by breaking up the maximization step to action selection and action evaluation.

✓2 Q functions Q_A and Q_B are calculated.

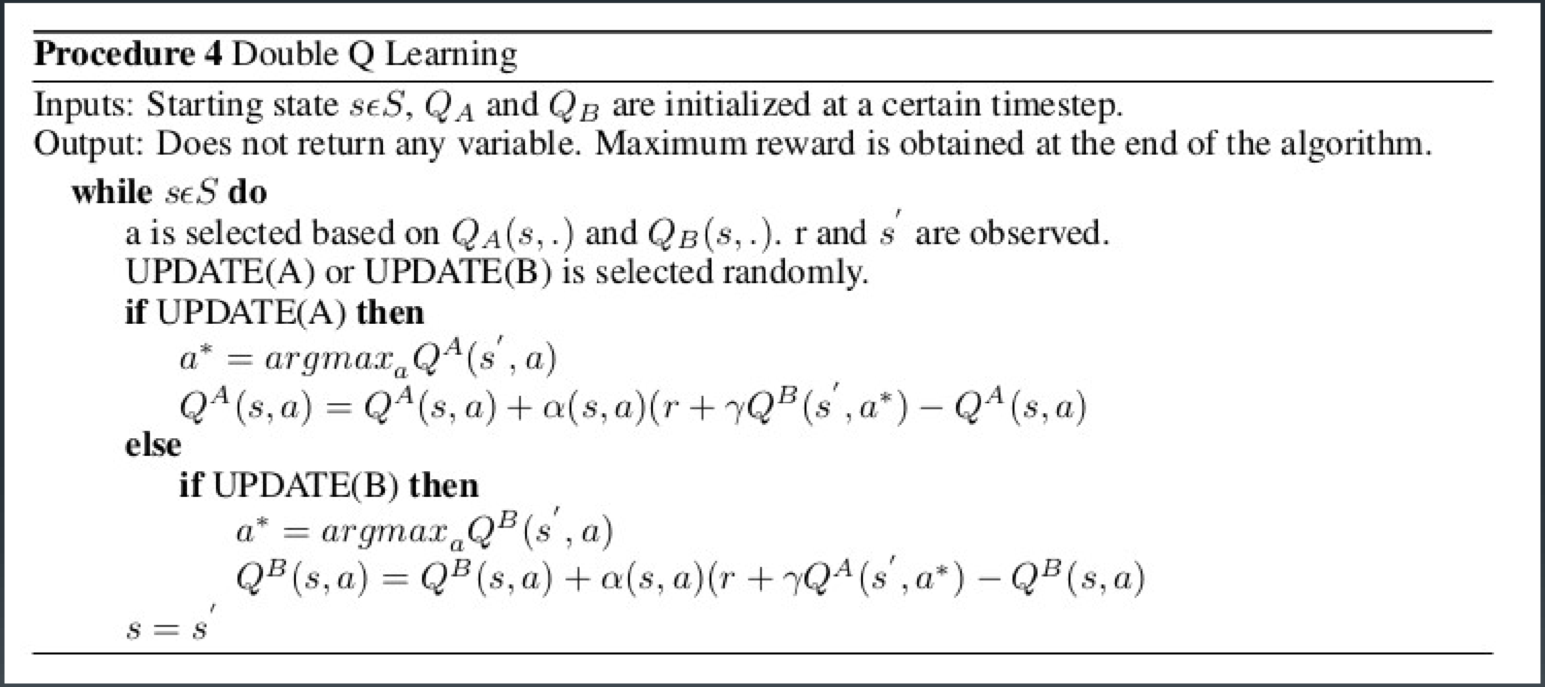
✓Q_A is updated with max(Q_B) instead of max(Q_A) over state s and vice-versa.

✓Q_B can be considered to be an unbiased estimate for Q_A and vice-versa as it is updated in the same problem with a different set of experience samples.

✓Double Q Learning is not less data-efficient than Deep Q Learning.

  ✓Both Q_A and Q_B are used to select an action.

Double Q Learning Algorithm



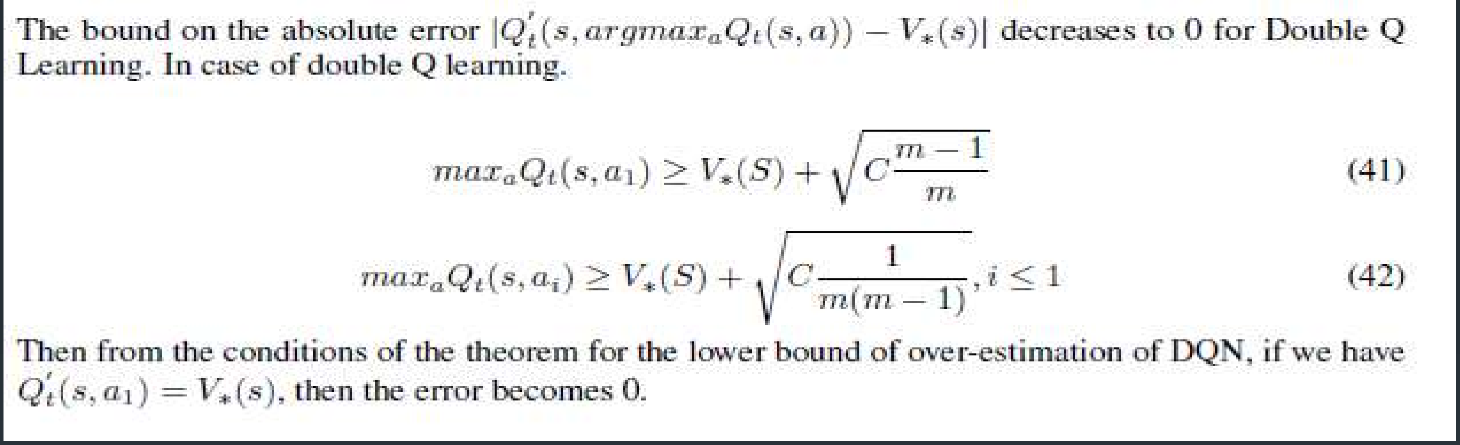
Procedure 4 Double Q Learning

Inputs: Starting state $s\epsilon S$, Q_A and Q_B are initialized at a certain timestep.

Output: Does not return any variable. Maximum reward is obtained at the end of the algorithm.

```
  while $s\epsilon S$ do
      a is selected based on $Q_A(s,.)$ and $Q_B(s,.)$. r and $s^{'}$ are observed.
      UPDATE(A) or UPDATE(B) is selected randomly.
      if UPDATE(A) then
          $a^* = argmax_a Q^A(s^{'}, a)$
          $Q^A(s,a) = Q^A(s,a) + \alpha(s,a)(r + \gamma Q^B(s^{'}, a^*) - Q^A(s,a)$
      else
          if UPDATE(B) then
              $a^* = argmax_a Q^B(s^{'}, a)$
              $Q^B(s,a) = Q^B(s,a) + \alpha(s,a)(r + \gamma Q^A(s^{'}, a^*) - Q^B(s,a)$
      $s = s^{'}$
```

Lower Bound of Over-Estimation in Double Q Learning



The bound on the absolute error $|Q'_t(s, argmax_a Q_t(s,a)) - V_*(s)|$ decreases to 0 for Double Q Learning. In case of double Q learning.

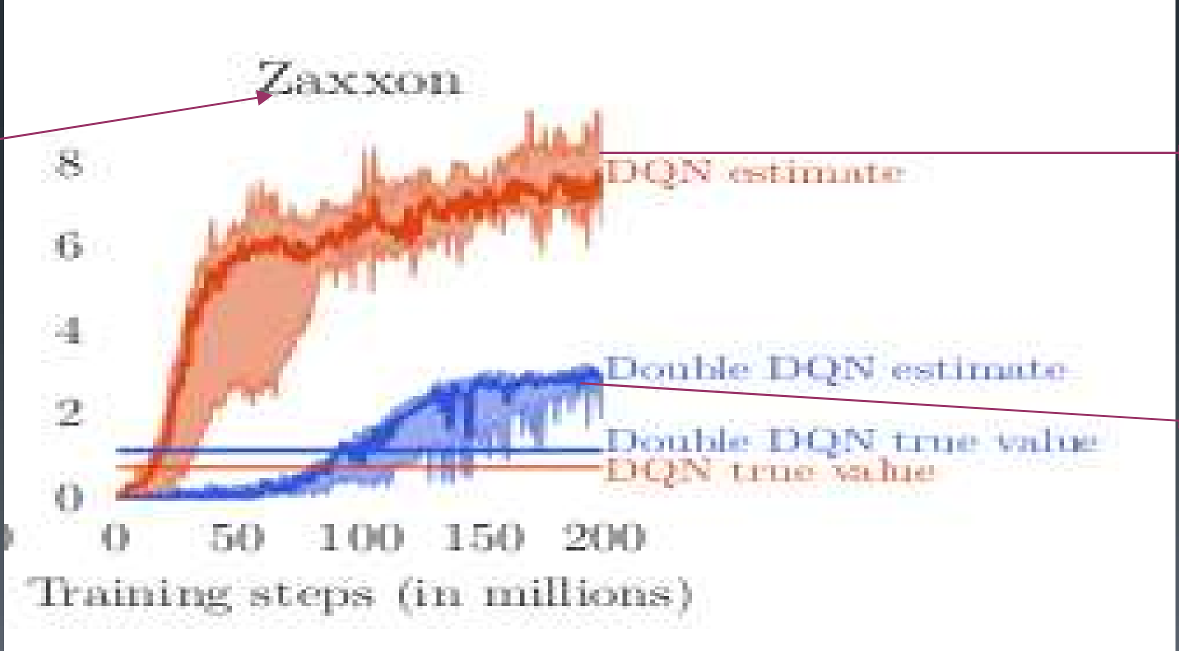
$$max_a Q_t(s,a_1) \geq V_*(S) + \sqrt{C\frac{m-1}{m}} \tag{41}$$

$$max_a Q_t(s,a_i) \geq V_*(S) + \sqrt{C\frac{1}{m(m-1)}}, i \leq 1 \tag{42}$$

Then from the conditions of the theorem for the lower bound of over-estimation of DQN, if we have $Q'_t(s,a_1) = V_*(s)$, then the error becomes 0.

Improvements with Double Q Learning

- Double Q-Learning reduces the over-optimism leading to more stable and reliable learning.
- Deep Q Network with Double Q-Learning (DDQN) is more robust than Deep Q-Learning and provides higher mean and median scores for the Atari games.



An Atari Game

Over-estimation of DQN

Reduced Over-estimation for DDQN

Conclusion

- The statistical convergence rate of Deep Q Learning is the sum of an algorithmic and statistical error, under the independent sampling assumption on experience replay and by using ReLU in deep neural networks, that captures the main features of DQN.

- The algorithmic error converges to 0 for infinite action space while the statistical error represent the bias and variance of the function approximation.

- The asymptotic convergence rate of Q Learning has also been studied in the project and it has been observed that appropriate selection of the learning rate leads to convergence of Q Learning in polynomial time.

Conclusion

- Over-estimation is another major computational and statistical challenge of Deep Q Learning. The lower bound of the over-estimation in Deep Q Learning has been investigated in the project.

- The mechanism of Double Q Learning has been used for large scale function approximation to solve over-estimation and ensure stable and reliable learning.

Future Work

Future work involves computation of the convergence rate to enable a comparative analysis of the Deep Q Learning and Deep Reinforcement Learning with Double Q Learning with a theoretical perspective.

References

1.F.-F. Li, “Cs231n: Convolutional neural networks for visual recognition,” in Stanford University, 2019, http://cs231n.stanford.edu/.

2.S. Machandranath Kakade, “On the sample complexity of reinforcement learning,” 01 2003.

3.C. Szepesvári, “The asymptotic convergence-rate of q-learning,” in Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10, ser. NIPS ’97. Cambridge, MA, USA: MIT Press, 1998, pp. 1064–1070. [Online]. Available: http://dl.acm.org/citation.cfm?id=302528.302898

4.E. Even-Dar and Y. Mansour, “Learning rates for q-learning,” J. Mach. Learn. Res., vol. 5, pp. 1–25, Dec. 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005332.1005333

5.V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” in NIPS Deep Learning Workshop, 2013.

6.11Z. Yang, Y. Xie, and Z. Wang, “A theoretical analysis of deep q-learning,” CoRR, vol. Abs/1901.00137, 2019. [Online]. Available: http://arxiv.org/abs/1901.00137

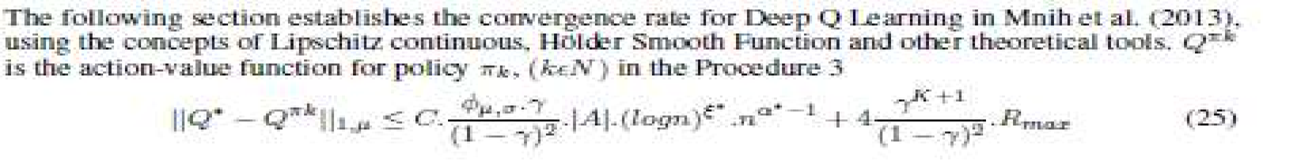
7.A. Rocke, “Theoritical limitations of dqn,” in Online Reinforcement Learning Blog, 2017, https://paulispace.com/inference/2017/08/29/dqn.html.

8.H. v. Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” in Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, ser. AAAI’16. AAAI Press, 2016, pp. 2094–2100. [Online]. Available: http://dl.acm.org/citation.cfm?id=3016100.3016191

9.H. V. Hasselt, “Double q-learning,” in Advances in Neural Information Processing Systems 23, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp.2613–2621. [Online]. Available: http://papers.nips.cc/paper/3964-double-q-learning.pdf

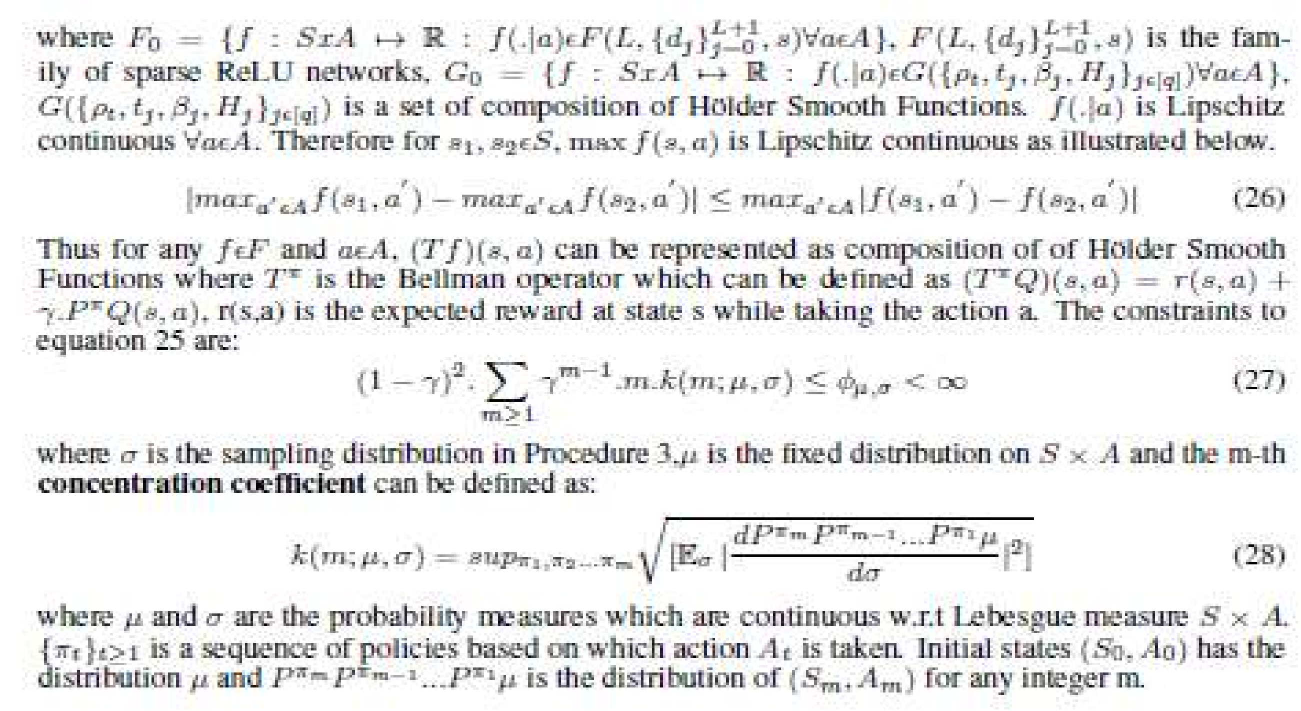
Thank You

Statistical Convergence Rate of Deep Q Learning



The following section establishes the convergence rate for Deep Q Learning in Mnih et al. (2013), using the concepts of Lipschitz continuous, Hölder Smooth Function and other theoretical tools. $Q^{\pi k}$ is the action-value function for policy π_k, $(k \epsilon N)$ in the Procedure 3

$$||Q^* - Q^{\pi k}||_{1,\mu} \leq C.\frac{\phi_{\mu,\sigma}.\gamma}{(1-\gamma)^2}.|A|.(logn)^{\xi^*}.n^{\alpha^*-1} + 4\frac{\gamma^{K+1}}{(1-\gamma)^2}.R_{max} \tag{25}$$



where $F_0 = \{f : SxA \mapsto \mathbb{R} : f(.|a) \epsilon F(L, \{d_j\}_{j=0}^{L+1}, s) \forall a \epsilon A\}$, $F(L, \{d_j\}_{j=0}^{L+1}, s)$ is the family of sparse ReLU networks, $G_0 = \{f : SxA \mapsto \mathbb{R} : f(.|a) \epsilon G(\{\rho_t, t_j, \beta_j, H_j\}_{j \epsilon [q]}) \forall a \epsilon A\}$, $G(\{\rho_t, t_j, \beta_j, H_j\}_{j \epsilon [q]})$ is a set of composition of Hölder Smooth Functions. $f(.|a)$ is Lipschitz continuous $\forall a \epsilon A$. Therefore for $s_1, s_2 \epsilon S$, $\max f(s,a)$ is Lipschitz continuous as illustrated below.

$$|max_{a' \epsilon A} f(s_1, a^{'}) - max_{a' \epsilon A} f(s_2, a^{'})| \leq max_{a' \epsilon A} |f(s_1, a^{'}) - f(s_2, a^{'})| \tag{26}$$

Thus for any $f \epsilon F$ and $a \epsilon A$, $(Tf)(s,a)$ can be represented as composition of of Hölder Smooth Functions where T^π is the Bellman operator which can be defined as $(T^\pi Q)(s,a) = r(s,a) + \gamma.P^\pi Q(s,a)$, r(s,a) is the expected reward at state s while taking the action a. The constraints to equation 25 are:

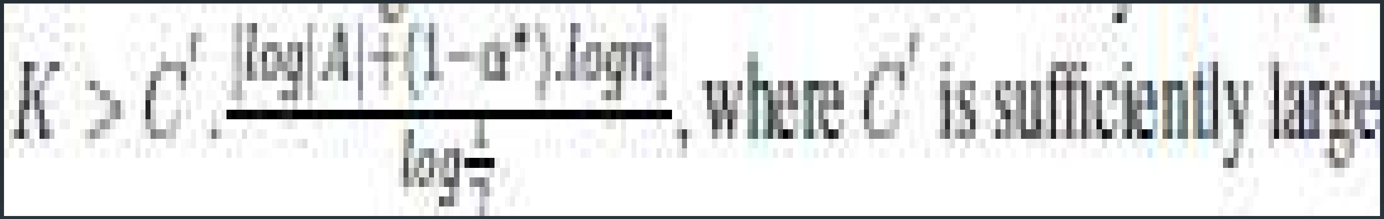
$$(1-\gamma)^2. \sum_{m \geq 1} \gamma^{m-1}.m.k(m; \mu, \sigma) \leq \phi_{\mu,\sigma} < \infty \tag{27}$$

where σ is the sampling distribution in Procedure 3,μ is the fixed distribution on $S \times A$ and the m-th **concentration coefficient** can be defined as:

$$k(m; \mu, \sigma) = sup_{\pi_1, \pi_2 ... \pi_m} \sqrt{[\mathbb{E}_\sigma \, |\frac{dP^{\pi_m} P^{\pi_{m-1}} ... P^{\pi_1} \mu}{d\sigma}|^2]} \tag{28}$$

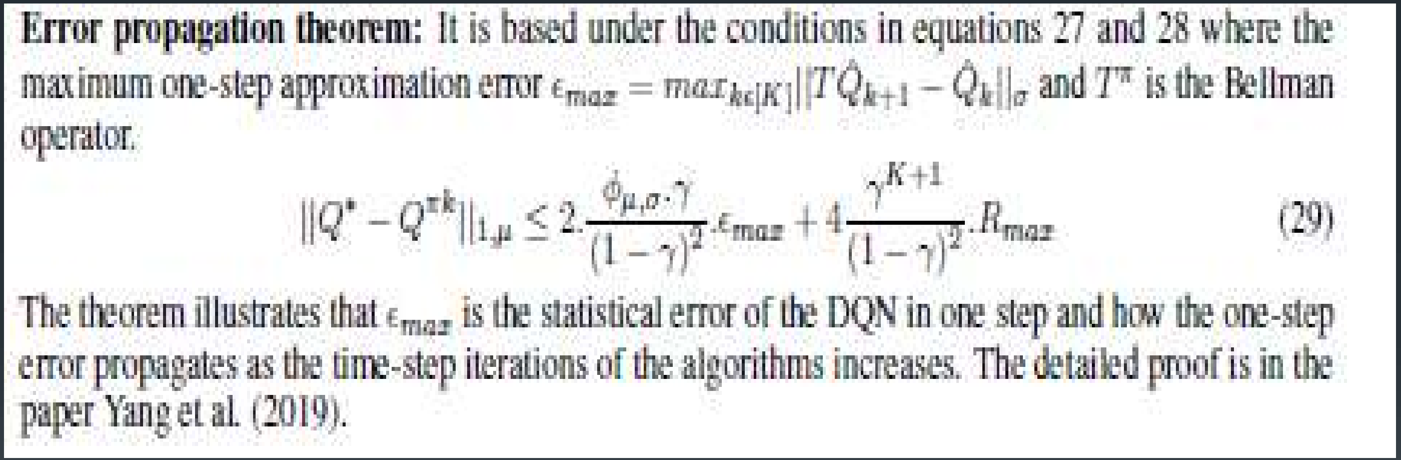
where μ and σ are the probability measures which are continuous w.r.t Lebesgue measure $S \times A$. $\{\pi_t\}_{t \geq 1}$ is a sequence of policies based on which action A_t is taken. Initial states (S_0, A_0) has the distribution μ and $P^{\pi_m} P^{\pi_{m-1}} ... P^{\pi_1} \mu$ is the distribution of (S_m, A_m) for any integer m.

Condition of Algorithmic Error being over-shadowed by Statistical Error



$$K > C' \cdot \frac{[\log|A| + (1-\alpha^*) \cdot \log n]}{\log\frac{1}{\gamma}}$$, where C' is sufficiently large

Error Propagation Theorem

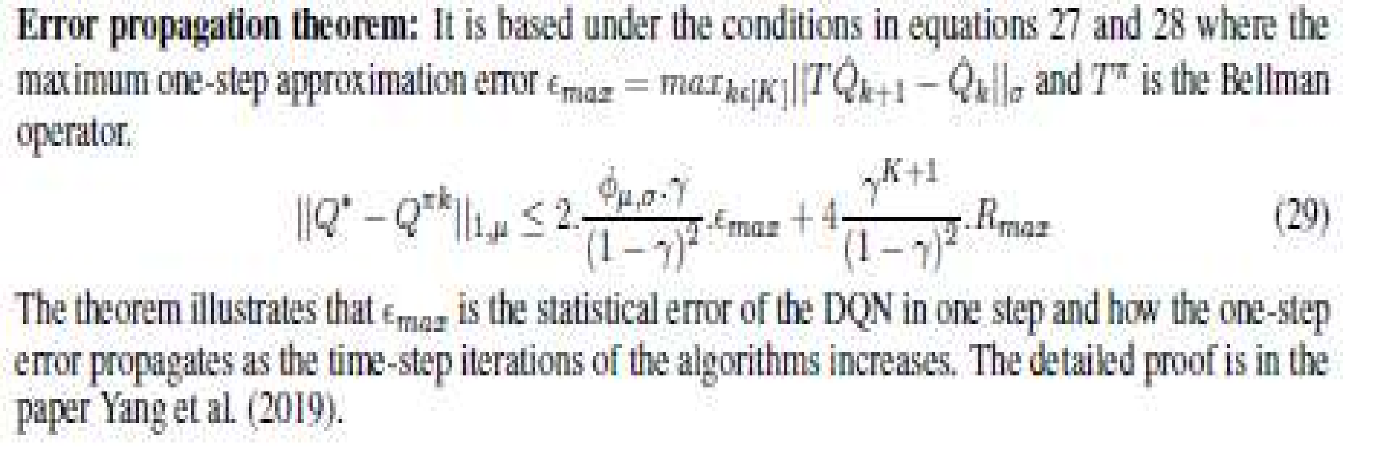


Error propagation theorem: It is based under the conditions in equations 27 and 28 where the maximum one-step approximation error $\epsilon_{max} = max_{k\epsilon[K]}||T\hat{Q}_{k+1} - \hat{Q}_k||_\sigma$ and T^π is the Bellman operator.

$$||Q^* - Q^{\pi k}||_{1,\mu} \leq 2.\frac{\dot{\phi}_{\mu,\sigma}.\gamma}{(1-\gamma)^2}.\epsilon_{max} + 4\frac{\gamma^{K+1}}{(1-\gamma)^2}.R_{max} \tag{29}$$

The theorem illustrates that ϵ_{max} is the statistical error of the DQN in one step and how the one-step error propagates as the time-step iterations of the algorithms increases. The detailed proof is in the paper Yang et al. (2019).

One-step Approximation Theorem

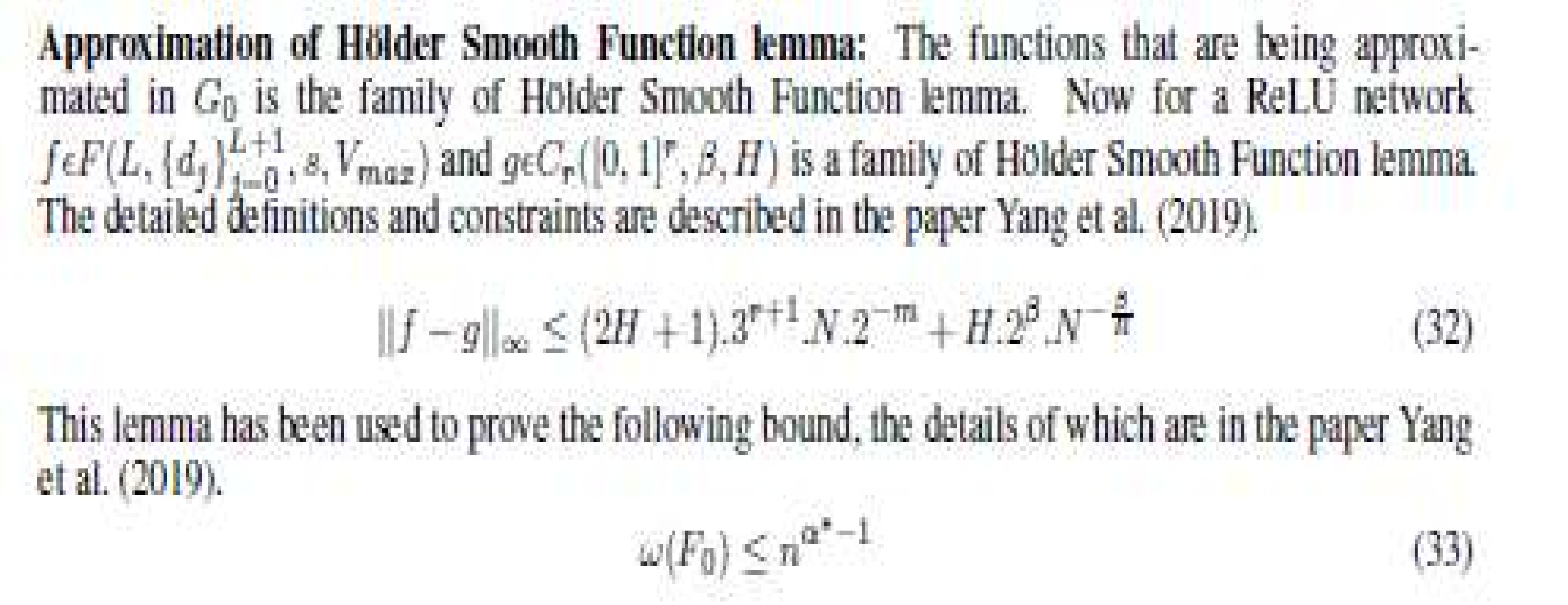


Error propagation theorem: It is based under the conditions in equations 27 and 28 where the maximum one-step approximation error $\epsilon_{max} = max_{k\epsilon[K]}||T\hat{Q}_{k+1} - \hat{Q}_k||_\sigma$ and T^π is the Bellman operator.

$$||Q^* - Q^{\pi k}||_{1,\mu} \leq 2.\frac{\phi_{\mu,\sigma}.\gamma}{(1-\gamma)^2}.\epsilon_{max} + 4\frac{\gamma^{K+1}}{(1-\gamma)^2}.R_{max} \qquad (29)$$

The theorem illustrates that ϵ_{max} is the statistical error of the DQN in one step and how the one-step error propagates as the time-step iterations of the algorithms increases. The detailed proof is in the paper Yang et al. (2019).

Approximation of Holder Smooth Function Lemma



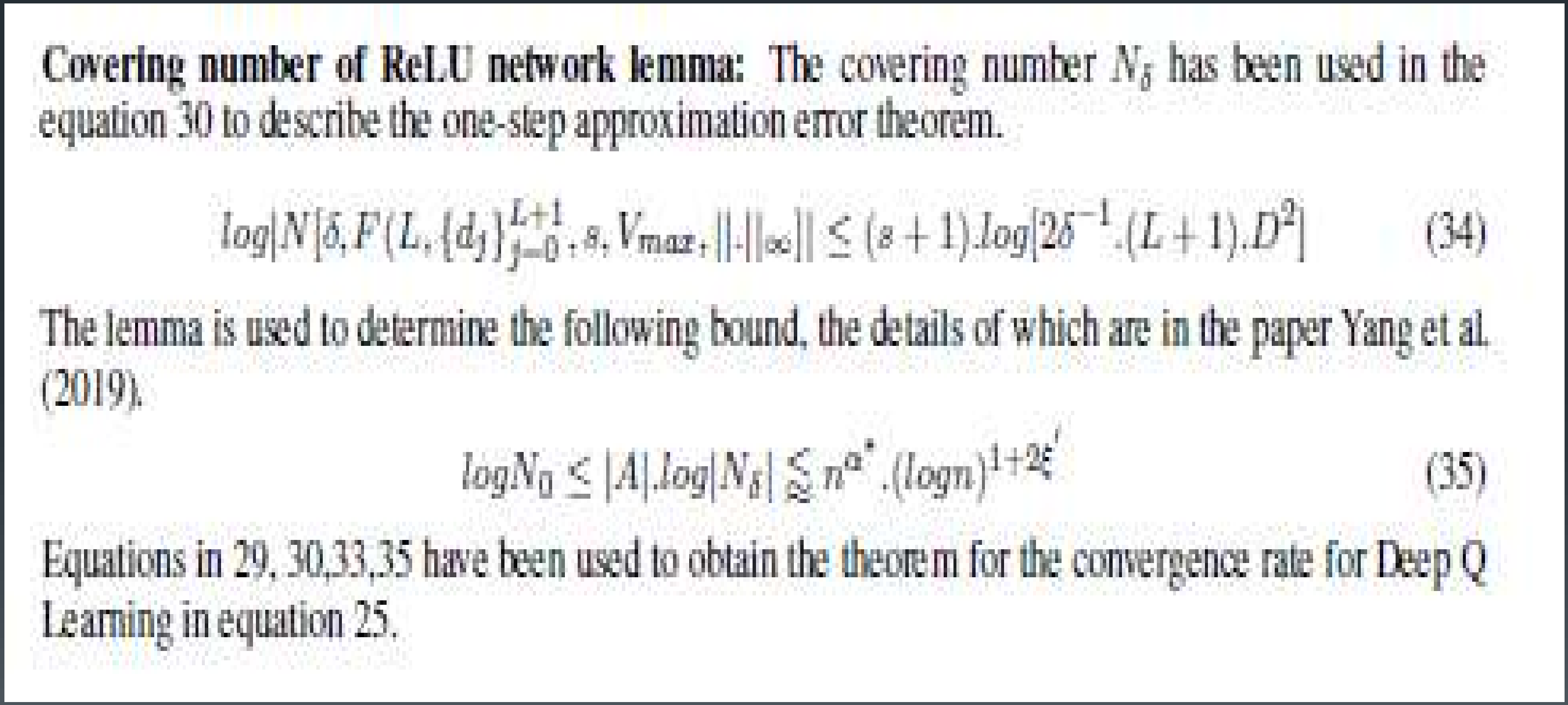
Approximation of Hölder Smooth Function lemma: The functions that are being approximated in G_0 is the family of Hölder Smooth Function lemma. Now for a ReLU network $f \epsilon F(L, \{d_j\}_{j=0}^{L+1}, s, V_{max})$ and $g \epsilon C_r([0,1]^r, \beta, H)$ is a family of Hölder Smooth Function lemma. The detailed definitions and constraints are described in the paper Yang et al. (2019).

$$||f - g||_\infty \leq (2H+1).3^{r+1}.N.2^{-m} + H.2^\beta.N^{-\frac{\beta}{r}} \tag{32}$$

This lemma has been used to prove the following bound, the details of which are in the paper Yang et al. (2019).

$$\omega(F_0) \leq n^{\alpha^*-1} \tag{33}$$

Covering Number of ReLU Network Lemma



Covering number of ReLU network lemma: The covering number N_δ has been used in the equation 30 to describe the one-step approximation error theorem.

$$log|N[\delta, F(L, \{d_j\}_{j=0}^{L+1}, s, V_{max}, ||.||_\infty]| \leq (s+1).log[2\delta^{-1}.(L+1).D^2] \qquad (34)$$

The lemma is used to determine the following bound, the details of which are in the paper Yang et al. (2019).

$$logN_0 \leq |A|.log|N_\delta| \lesssim n^{\alpha^*}.(logn)^{1+2\xi'} \qquad (35)$$

Equations in 29, 30,33,35 have been used to obtain the theorem for the convergence rate for Deep Q Learning in equation 25.