Reinforcement Learning: Policy Gradient methods

Instructor: Prof. Yasser Alginahi

Presented By:

Asif Shaik

Ajay Guru Madu

Lahiri Kanipakam

Date: 17th November, 2023.

University of Windsor

Agenda

- Overview of learning models
- Reinforcement Learning
- Classification of Reinforcement Learning
- Model based and Model-free Algorithm
- Markov Decision Process
- REINFORCE Algorithm
- Actor-Critic Method
- Deep Deterministic policy gradient(DDPG)
- DDPG Example
- Reference

Comparing Supervised, Unsupervised and Reinforcement Learning

- Supervised Learning: the goal is to generate formula based on input and output values.
- Unsupervised Learning: we find an association between input values and group them.
- Reinforcement Learning: an agent learn through delayed feedback by interacting with the environment.

VS
VS
Environment
State, Reward
Action
Agent
Supervised Learning
Unsupervised Learning
Reinforcement Learning

Task driven (Classification/ Regression)

Data driven (Clustering)

Algorithm learns to react to an environment

https://tinyurl.com/2s83ktye

What is Reinforcement Learning?

Reinforcement Learning (RL) is the science of decision making. It is about learning the optimal behavior in an environment to obtain maximum reward. This optimal behavior is learned through interactions with the environment and observations of how it responds, similar to children exploring the world around them and learning the actions that help them achieve a goal.

Reinforcement Learning Model

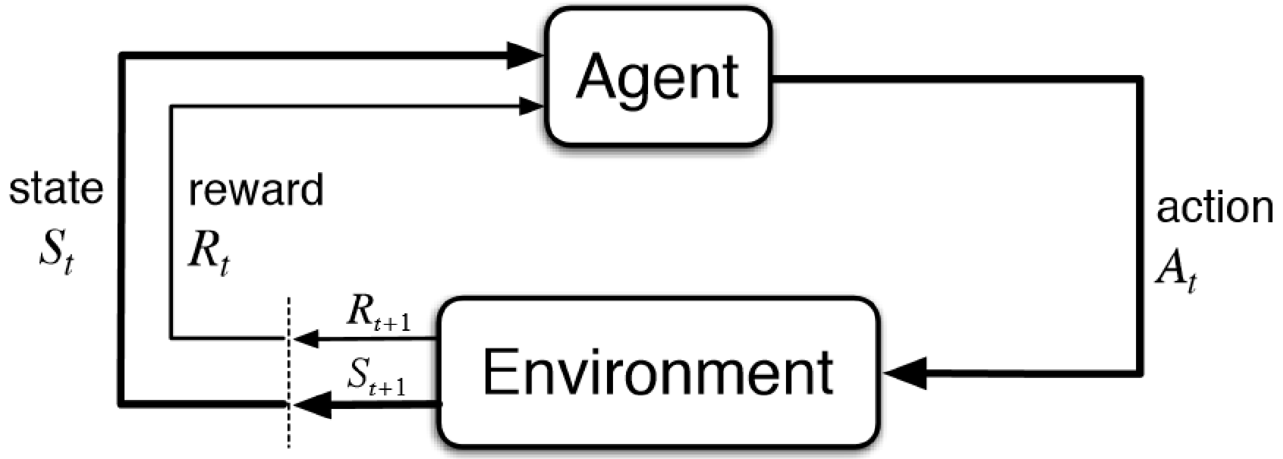


Figure: The agent environment interaction in a Markov decision process

https://tinyurl.com/2m6tc392

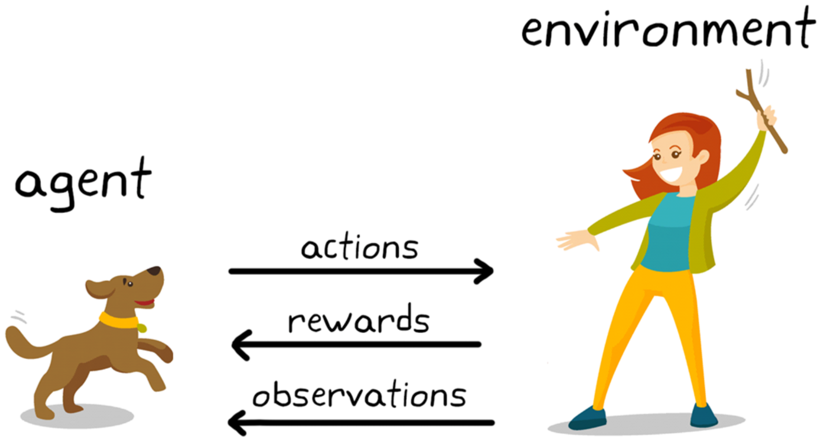
How Reinforcement Learning is different ?

- Delayed Reward
- Agent chooses training data
- Explore vs Exploit (Lifelong learning)

Reinforcement Learning Example

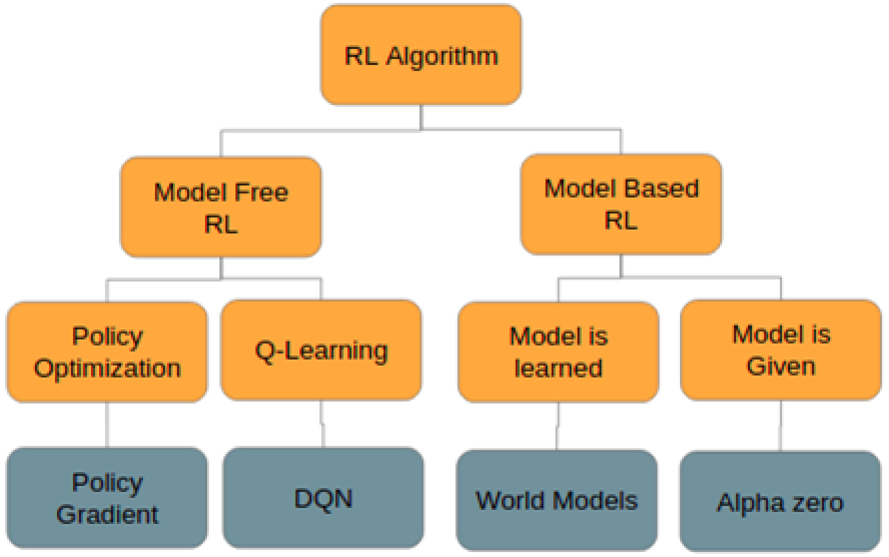
- In the realm of reinforcement learning, the goal is to train a dog (the agent) to perform tasks within a specified environment, encompassing both physical surroundings and the trainer.

- The ultimate objective is for the dog to develop an optimal policy, maximizing rewards like treats by consistently responding correctly to cues.



https://tinyurl.com/mr3ryfy5

Classification of Reinforcement Learning



https://tinyurl.com/282fpnr2

Model based and Model-free

Model-based, as it sounds, has an agent trying to understand its environment and creating a model for it based on its interactions with this environment. In such a system, preferences take priority over the consequences of the actions i.e. the greedy agent will always try to perform an action that will get the maximum reward irrespective of what that action may cause.

Model-free algorithms seek to learn the consequences of their actions through experience via algorithms such as Policy Gradient, Q-Learning, etc. In other words, such an algorithm will carry out an action multiple times and will adjust the policy (the strategy behind its actions) for optimal rewards, based on the outcomes.

Markov Decision Processes

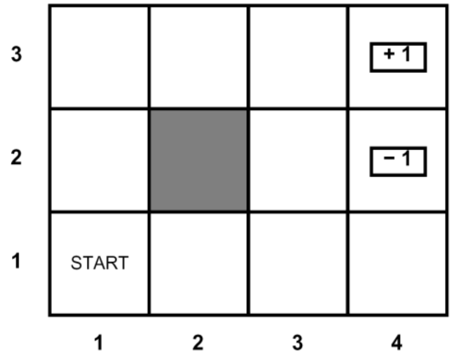
- A Markov decision process (MDP) refers to a stochastic decision-making process that uses a mathematical framework to model the decision-making of a dynamic system. It is used in scenarios where the results are either random or controlled by a decision maker, which makes sequential decisions over time. MDPs evaluate which actions the decision maker should take considering the current state and environment of the system.

Mathematical expression

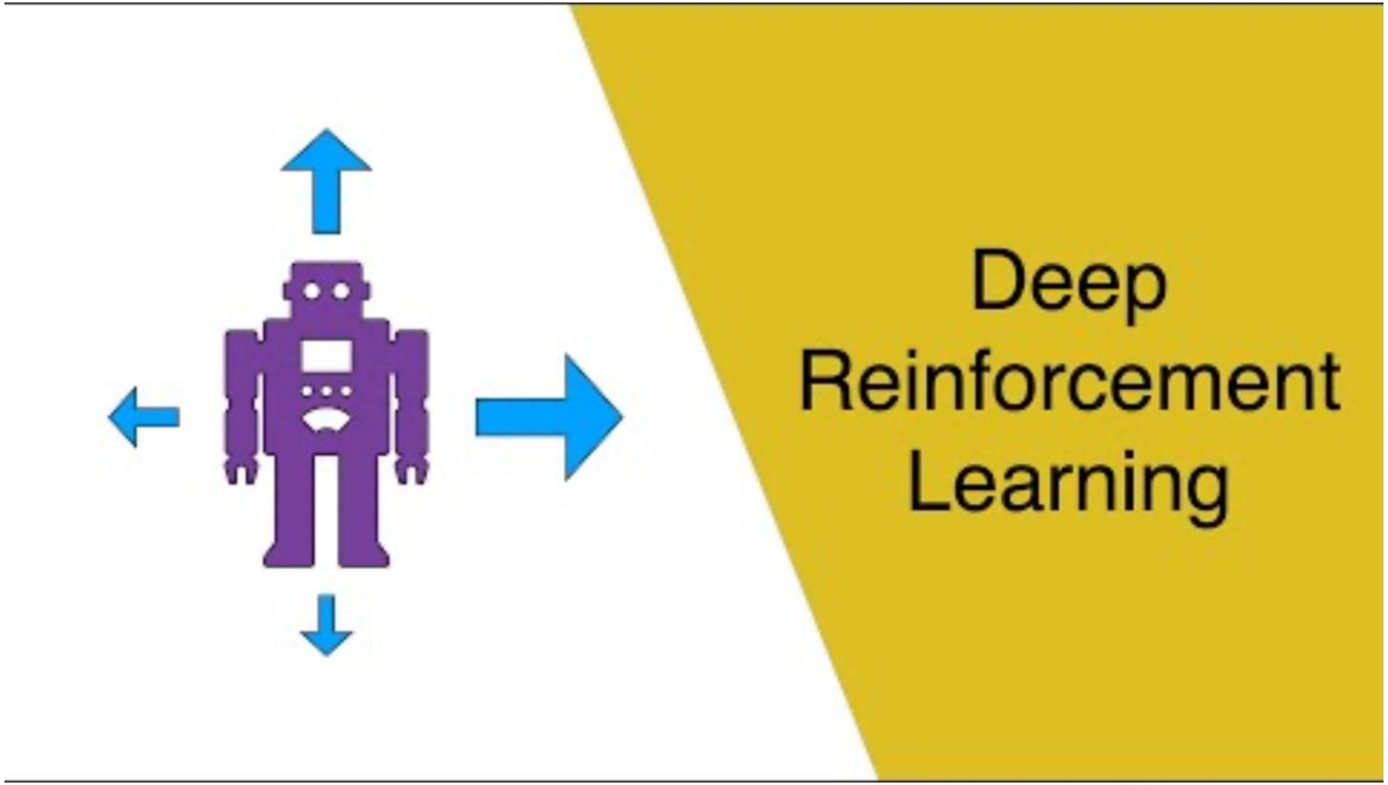
$$P(S_{t+1} = s' \mid S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots\ldots S_0 = s_0$$

An MDP is defined by:

- A set of states s ∈ S
- A set of actions a ∈ A
- A transition function T(s,a,s')
- Prob that a from s leads to s' i.e., P(s' | s,a). Also called the model
- A reward function R(s, a, s') Sometimes just R(s) or R(s')
- A start state (or distribution)
- Maybe a terminal state
- A policy π(a | s) defines the probability of taking action a in state s.

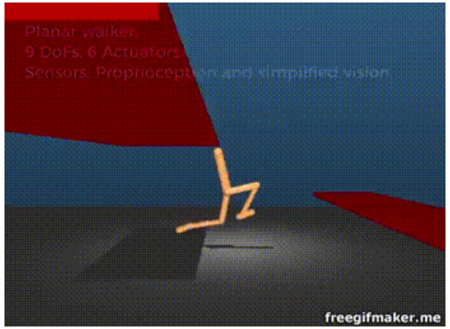


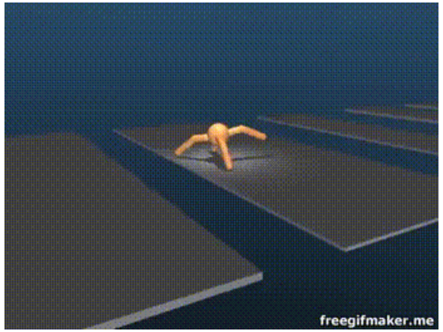
Markov Decision Explained



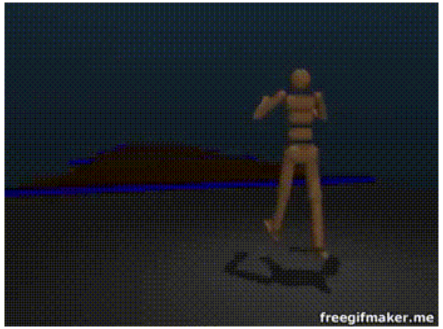
Robotics and Locomotion

State:
  Joint States/Velocities
  Accelerometer/Gyroscope
  Terrain
Actions: Apply Torque to Joints
Reward: Positive(if jumps) else Negative(Retrain)





https://youtu.be/hx_bgoTF7bs

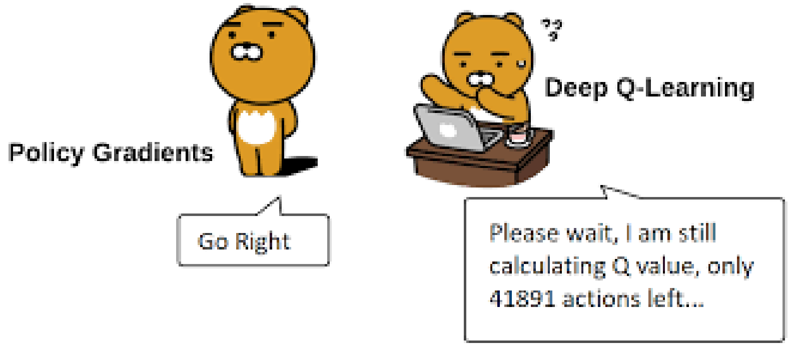


https://arxiv.org/pdf/1707.02286.pdf

University of Windsor

Why Policy Gradient?

- Q-Learning- >learn a value function
  $Q(s,a)$ = an estimate of the expected discounted reward of taking a from s
  Performance time: take the action that has the highest estimated value
- Policy Gradient-> Learn policy Directly
  $\pi(s)$ = Probability distribution over A_s
  Performance time: choose action according to distribution


Policy Gradients
Go Right
Deep Q-Learning
Please wait, I am still calculating Q value, only 41891 actions left...

https://tinyurl.com/z3acx6zc

REINFORCE Algorithm

Core Concept:

REINFORCE, also known as Monte-Carlo Policy Gradient, updates policy parameters θ based on an estimated return using episode samples.

Process:

1. *Initialization*: Start with random policy parameters θ.
2. *Trajectory Generation*: Generate a trajectory by running policy π_θ : S_1, A_1, R_2 , S_2 , A_2 ,..., S_T.
3. *Policy Update*:

   For each timestep *t*:

   - Calculate return G_t.

Mathematical Representation:

$$\nabla_{\theta} J(\theta) = E_{\pi} [G_t \nabla_{\theta} \ln\pi_{\theta} (A_t \mid S_t).$$

where G_t is the total discounted return from timestep t.

Example of REINFORCE algorithm

- Let's say we have a simplified scenario where a drone is learning to fly through a series of waypoints to reach a target. Each action the drone takes is either moving forward, ascending, or descending. The reward is given based on the drone's proximity to the next waypoint—closer proximity yields a higher reward.
- For this example, let's consider a single episode where the drone takes three actions to reach the target.

https://tinyurl.com/3be72bh8

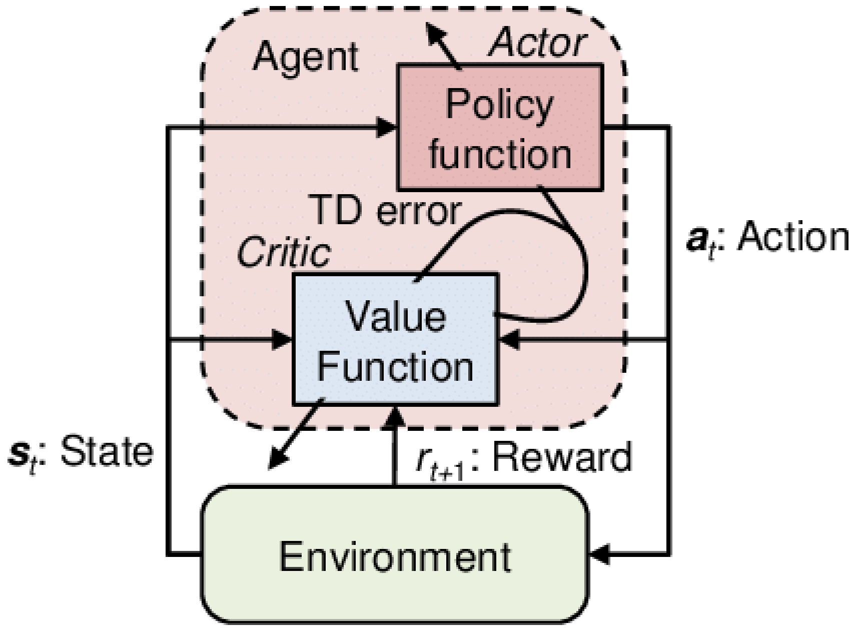
University of Windsor

Actor-Critic Method in Reinforcement Learning

- **Actor (Policy Model)**:
  - The actor makes decisions, mapping states to actions.
  - It is responsible for generating actions in an environment based on a policy.
- **Critic (Value Model)**:
  - The critic assess the actions taken by the actor.
  - It estimates the value function of the current policy.

https://tinyurl.com/y8fj264m

Actor-Critic Method in Reinforcement Learning



https://tinyurl.com/ydfsby2f

Actor: decides which action to take.

Critic : tells the actor how good its action was and how it should adjust.

Actor Critic Algorithm

```
Initialize policy parameter $\theta$, baseline b
for iteration=1, 2 .,... do
    Collect a set of trajectories by executing the current policy
    At each timestep in each trajectory, compute the return R = $\sum_{t'=t}^{T-1} \gamma^{t'-t} r_t$ , and
    the advantage estimate $\widehat{A_t}$ = $R_t$-$b(s_t)$( difference between return and baseline)
    Re-fit the baseline, by minimize $\left|\left|b(S_t) - R_t\right|\right|^2$
    $(squared\ difference\ between\ the\ baseline\ and\ predictions\ of\ return\ )$,
    summed over all trajectories and timesteps.
    Update the policy, using a policy gradient estimate $\hat{g}$, which is a sum of terms
    $\nabla_\theta\ log\pi(a_t \mid s_{t,\theta})A_t$
end for
```

Why Actor Critic and Not REINFORCE??

- As in the REINFORCE algorithm, we update the policy parameter through Monte Carlo updates (i.e. taking random samples). This introduces in inherent high variability in log probabilities (log of the policy distribution) and cumulative reward values, because each trajectories during training can deviate from each other at great degrees.

<u>Noisy gradients and high variance.</u>

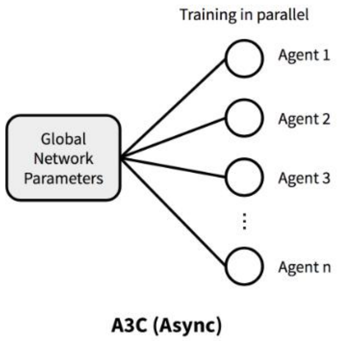
<u>Reinforce with Baseline:</u>

$$\nabla_\theta J(\theta) = E\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta (a_t \mid s_t)(G_t - b(s_t))\right]$$

Actor-Critic Methods

Asynchronous Advantage Actor-Critic (A3C) :

- Classic policy gradient method with a special focus on parallel training.
- Asynchronous nature leads to diverse experience and better policy exploration.
- More resource-intensive due to its parallel nature
- Less stable

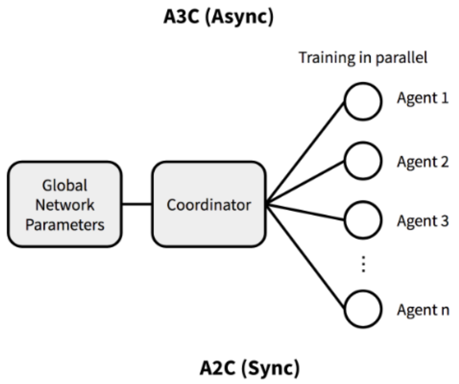


Architecture of A3C

Actor-Critic Methods (Contd.)

Advantage Actor-Critic (A2C):

- Synchronous: Waits for all agents to finish before updating the policy.
- More resource-efficient in terms of computational resources
- More stable due to synchronous updates and experience averaging.



Architecture of A3C vs A2C

Policy Gradient

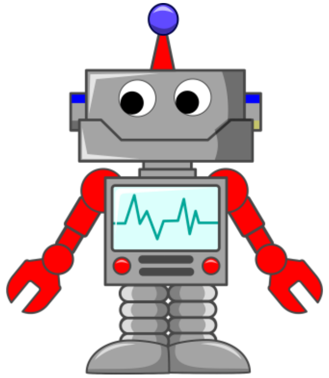
- Q- Learning can be performed only in discrete action spaces.
- Target Networks and Experience replay buffers are used.
- Deep Neural networks for efficient learning.
- Better convergence rate in the process.

Deep Deterministic policy gradient (DDPG)

- DDPG is a combination of powerful DRL techniques such as Deep Q learning and Actor- critic algorithms.
- DDPG is a DRL algorithm used to solve problems in continuous action spaces.
- It is a model free off policy algorithm.
- The agent interacts with environment to learn a policy that maximize the cumulative reward.

Applications

Robotics(here) Autonomous vehicles(here) Health care(here)



https://tinyurl.com/ypatea3c

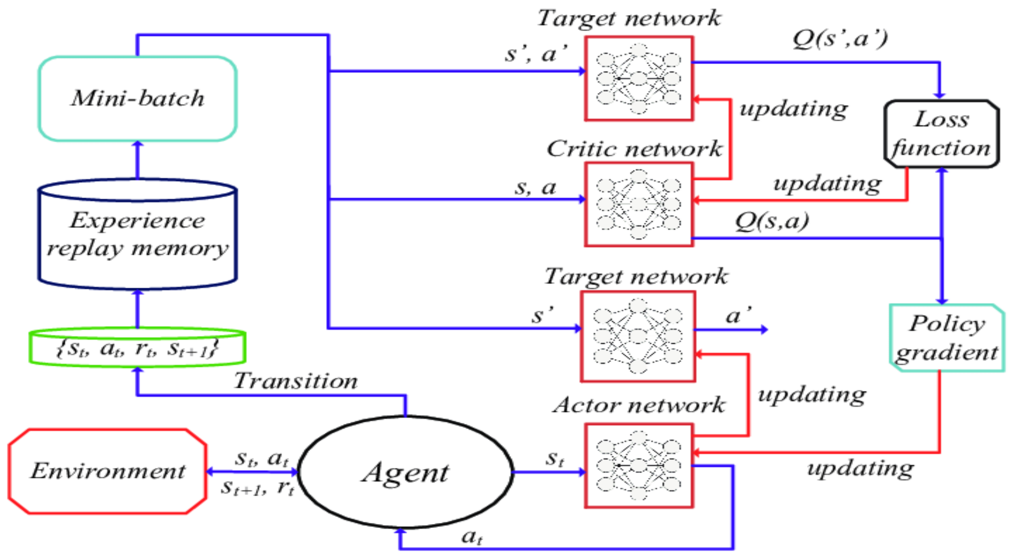
https://tinyurl.com/ytm37f6e

https://tinyurl.com/5ue3kwb9

DDPG Architecture

State: X, Y, Angular velocity
Action: Torque
Reward: 1(stands UP) else 0

https://tinyurl.com/bdhwb8jz

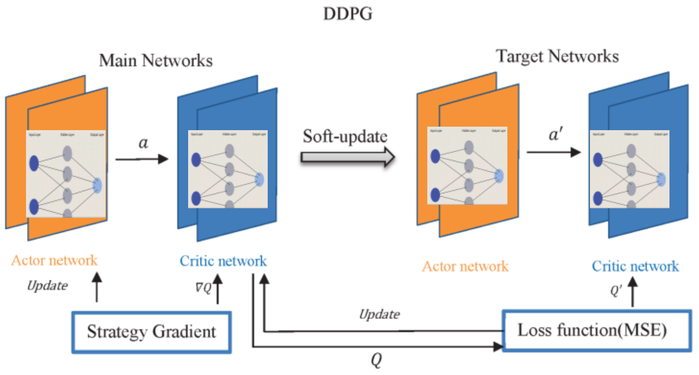


https://tinyurl.com/yck59ydp

Main and Target Networks

Target Networks:

- The old target network parameters and the new corresponding network parameters are weighte averaged and then assigned to the target networks.
- Soft update method (epsilon -greedy, soft-max)
- Stabilize the learning process.



https://tinyurl.com/bdsvfxce

University of Windsor

Main and Target Networks (Contd.)

Policy gradient:

- Updates the actor network parameters that gives higher expected rewards.

Loss function:

- Minimizes the loss by encouraging the policy to generate actions that leads to maximum rewards.
- Measures the error between target and actual values.

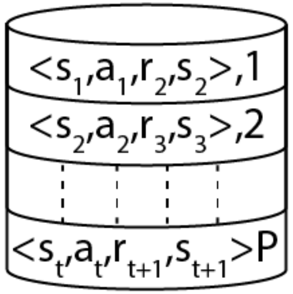
Terms

Replay buffer:

- Stores the cumulative transition (s, a, r, s’) of past experiences.
- The agent uses past experiences to learn and get better.
- Avoids choosing suboptimal policies.

Noise:

- Added to target actor networks.
- To make the target actions less deterministic and smooth out the Q value estimates.



https://tinyurl.com/4uh9cy8z

DDPG Algorithm

```
1.  Initialize the actor network and critic network.
2.  Initialize the target actor and target critic networks.
3.  Initialize the replay buffer.
4.  **Loop:**
5.      Noise added for action exploration.
6.      Agent interacts with environment from the initial state until terminal state is
    reached.
7.       Observe next state and reward.
8.       Store trajectories in a replay buffer for sampling
9.       Sample a mini batch of transitions
10.      Update Critic network parameters
11.      Update Policy network parameters
12.     Update Target networks.
13. **Repeat** steps (4) –(12)
14. Observe the results.
```

OpenAI Gym Documentation

- OpenAI is commonly used in reinforcement learning (RL) as a tool for developing and training RL models.
- The documentation includes following functions:

  Initialization: Initialize all the parameters that are being used in the code.

  Step: It takes the action as input, computes the state of environment and gives a tuple of values.

  Reset: Called when the done signal shows up and restarts another episode.

  Render: To visualize the results.

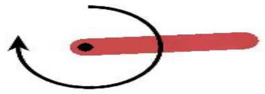
  Close

https://tinyurl.com/mum5jnc7

Example:

https://keras.io/examples/rl/ddpg_pendulum/

Results:



Conclusion:

- Deep Deterministic policy gradient(DDPG) is a combination of Reinforce, Actor- Critic and Deep Q Networks.
- A trade-off balance should be maintained between Exploration and Exploitation actions.

Reference

- Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. "Continuous Control with Deep Reinforcement Learning." *arXiv* **2015**, arXiv:1509.02971
- Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. "Deterministic policy gradient algorithms." In International Conference on Machine Learning, pp. 387–395, 2014.
- R. S. Sutton and A. G. "Barto, Reinforcement Learning: An Introduction, Second." The MIT Press, 2018 [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html
- Tsou, C. X. (2021, August 18). *Actor-critic: Implementing actor-critic methods*. Medium. https://medium.com/geekculture/actor-critic-implementing-actor-critic-methods-82efb998c273
- YouTube. (2020, November 4). *Everything you need to know about deep deterministic policy gradients (DDPG) | tensorflow 2 tutorial*. YouTube. https://www.youtube.com/watch?v=4jh32CvwKYw
- Karunakaran, D. (2020, June 8). *Reinforce-a policy-gradient based reinforcement learning algorithm*. Medium. https://medium.com/intro-to-artificial-intelligence/reinforce-a-policy-gradient-based-reinforcement-learning-algorithm-84bde440c816#:~:text=Policy%20Gradient%20algorithm,free%20reinforcement%20learning(RL).
- Source: D. Silver et al., "Deterministic policy gradient algorithms," in International Conference on Machine Learning, pp. 387-395, 2014.

THANK YOU