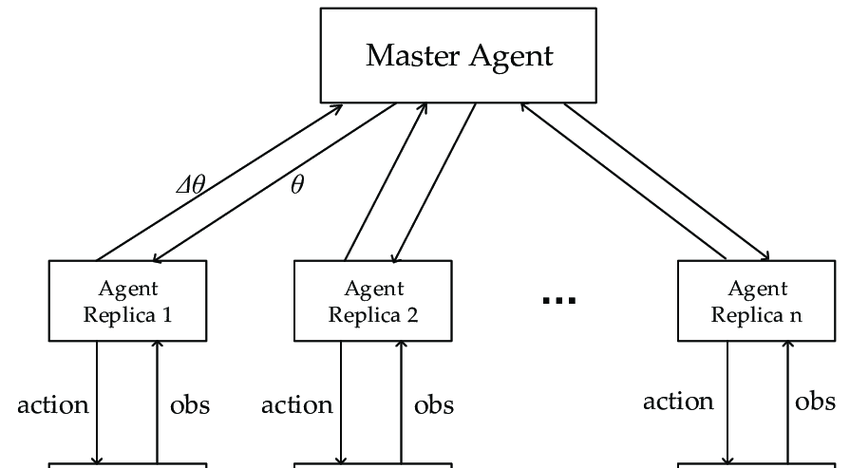
**Asynchronous Advantage Actor-Critic (A3C)**

****

The Asynchronous Advantage Actor Critic (A3C) algorithm is one of the newest algorithms to be developed under the field of Deep Reinforcement Learning Algorithms. This algorithm was developed by Google’s DeepMind which is the Artificial Intelligence division of Google.

It is known for its efficiency in training agents using parallel computation.

What is Deep Reinforcement Learning?

Reinforcement Learning is a branch of machine learning where an agent learns to make decisions by interacting with an environment, aiming to maximize cumulative reward over time. Deep Learning uses neural networks to automatically extract features and make decisions based on large, complex data inputs (like images or text). Deep Reinforcement Learning combines these two, using neural networks to approximate value functions, policies, and Q-functions that help an agent navigate complex environments.

Unlike other popular Deep Reinforcement Learning algorithms like Deep Q-Learning which uses a single agent and a single environment, This algorithm uses multiple agents with each agent having its own network parameters and a copy of the environment.

This setup mimics the real-life environment in which humans live as each human gains knowledge from the experiences of some other human thus allowing the whole “global network” to be better.

Actor-Critic: Unlike some simpler techniques which are based on either Value-Iteration methods or Policy-Gradient methods, the A3C algorithm combines the best parts of both the methods ie the algorithm predicts both the value function V(s) as well as the optimal policy function \pi (s) . The learning agent uses the value of the Value function (Critic) to update the optimal policy function (Actor). Note that here the policy function means the probabilistic distribution of the action space. To be exact, the learning agent determines the conditional probability P(a|s ;\theta ) ie the parameterized probability that the agent chooses the action a when in states.

### **Actor-Critic Framework**

A3C combines two networks: **actor** and **critic**.

**Actor**: This network takes the current state as input and outputs actions (or policy) to be taken.

**Critic**: This network evaluates the action taken by the actor by estimating the **value function**—the expected long-term reward from the current state.

The actor-critic framework allows the model to have a policy (actor) while also evaluating how good that policy is (critic).

The advantage function measures the benefit of taking an action compared to an average action in a given state. This helps guide the actor network toward better actions and is computed as: A(s,a)=Q(s,a)−V(s)A(s, a) = Q(s, a) - V(s)A(s,a)=Q(s,a)−V(s) where:

Q(s,a)Q(s, a)Q(s,a): Expected return after taking action aaa in state sss.

V(s)V(s)V(s): Expected return from state sss using the current policy.

This advantage function provides a baseline, reducing variance and speeding up learning.

### **Asynchronous Learning**

* A3C uses multiple parallel agents (or "workers") to explore the environment independently, each interacting with its own copy of the environment.
* Each agent updates the global model asynchronously, allowing for faster learning as different experiences are accumulated.
* This parallelism avoids the need for experience replay buffers (a typical component in other deep reinforcement learning algorithms), as multiple agents naturally provide diverse experiences.

Actor-Critic: Unlike some simpler techniques which are based on either Value-Iteration methods or Policy-Gradient methods, the A3C algorithm combines the best parts of both the methods ie the algorithm predicts both the value function V(s) as well as the optimal policy function \pi (s) .

The learning agent uses the value of the Value function (Critic) to update the optimal policy function (Actor). Note that here the policy function means the probabilistic distribution of the action space. To be exact, the learning agent determines the conditional probability P(a|s ;\theta ) ie the parameterized probability that the agent chooses the action a when in state s.

Advantage: Typically in the implementation of Policy Gradient, the value of Discounted Returns(\gamma r ) to tell the agent which of it’s actions were rewarding and which ones were penalized. By using the value of Advantage instead, the agent also learns how much better the rewards were than it’s expectation. This gives a new-found insight to the agent into the environment and thus the learning process is better. The advantage metric is given by the following expression:- Advantage: A = Q(s, a) – V(s)

### **Key Steps in A3C**

* **Step 1**: Each worker runs a copy of the policy and value networks and collects experiences from the environment.
* **Step 2**: Using the collected experiences, each worker computes gradients for both the actor and critic.
* **Step 3**: The gradients are then asynchronously applied to the global model.
* **Step 4**: Each worker synchronizes its model with the updated global model and continues interacting with the environment.

### **Benefits of A3C**

* **Efficiency**: Due to asynchronous parallel processing, A3C is faster and less prone to getting stuck in local optima.
* **Stability**: The advantage function reduces variance, making the learning process smoother.
* **Adaptability**: Works well in complex, high-dimensional environments and continuous action spaces.

Code:-

**Constants**:

* N\_GAMES: Total number of episodes to run the training.
* T\_MAX: The number of time steps after which gradients are updated.

https://docs.google.com/document/d/1Xq7tpz3tN2YHyVAwDZCAesDrbSR2DYM7VDVzYPJu7qU/edit?usp=sharing