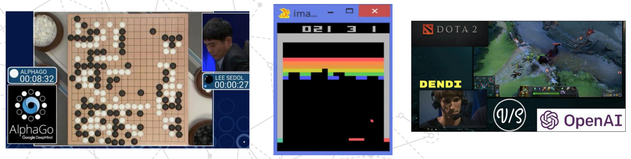
Asynchronous Double DQN Ensemble through Shared Experience Learning

Luckyson Khaidem, Ankit Anand and Alina Vereshchaka

University at Buffalo
The State University of New York

Overview

Deep reinforcement learning uses deep learning in reinforcement learning (RL) algorithms to solve problems that otherwise can't be solved using conventional RL methods. We propose a variation of deep RL algorithm wherein we use multiple instances of deep reinforcement learning agents that learn from each others experiences, that increase the learning speed. We have tested our model against 3 state-of-the-art deep RL algorithms (DQN, Double DQN, Actor-Critic) on two OpenAI environments (Cartpole and Acrobot).



Our goal

Develop a novel deep reinforcement learning framework that leverages the predictive performance of ensemble learning techniques. The proposed framework consists of multiple independent DDQN/DQN agents that learn from each other by sharing experiences.

Current problems:

- Current deep RL algorithms store the experiences of the agent in a database, a random sample of which is used every time to train the agent.
- This makes the learning process slow as the agent needs time to explore the environment and collect enough samples.

Our approach:

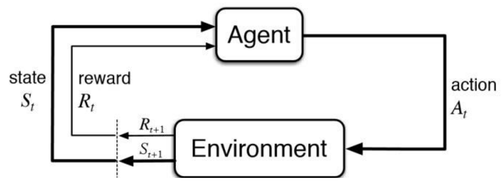
- We propose a deep RL framework where instead of one agent we spawn multiple instances of the environment and in each instance there is an agent learning to navigate in the environment independently of other agents.
- All the agents store their experiences in a shared database which allows them to learn from the experiences of other agents. This in turn should expedite the learning process.

Methodology

Markov decision process (MDP), defined by the tuple ⟨*s, a, o,* P, *r*⟩, where

- $s \in S$ denotes states, describing the possible configurations of all agents;
- $a \in A$ denotes actions, which can be discrete or continuous;
- P: S x A x S → R is the states transition probability distribution, where states evolve according to the stochastic dynamics $p(s_{t+1}|s_t,a_t)$;
- O is a set of observations for each agents;
- *r*: S → R is the reward function;

Q-Value – Q-value of an action in a particular state tells us how good that action is in that particular state.



$$Q^{new}(s_t,a_t) \leftarrow (1-\alpha)\cdot \underbrace{Q(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha}_{\text{learning rate}} \cdot \overbrace{\Big(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \overbrace{\max_a Q(s_{t+1},a)}^{a}\Big)}^{\text{learned value}}$$

Action a_t is sampled from the policy π_0 and the next state $s_{t+1}$is chosen based on the transition probability distribution $P(s_{t+1}|s_t, a_t)$.

Double Deep Q-network (DDQN)

DDQN is an improved version of DQN, that approximates the Q-values using **two neural networks** which are trained independently and then decides the action using epsilon greedy policy.

$$\min_{\Theta_A} \frac{1}{m}\sum_{i=1}^{i=m}((R_i+\gamma * Q_B(S_i', \arg\max_a Q_A(S_i',a,\Theta_A),\Theta_B)) - Q_A(S_i,a_i,\Theta_A))^2$$

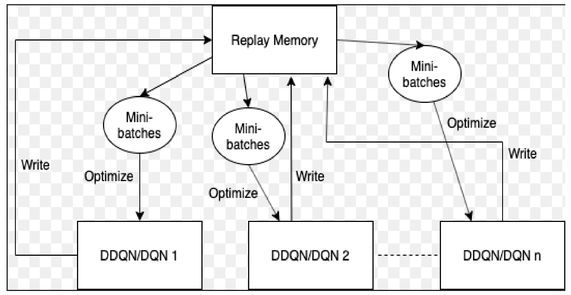
$$\min_{\Theta_B} \frac{1}{m}\sum_{i=1}^{i=m}((R_i+\gamma * Q_A(S_i', \arg\max_a Q_B(S_i',a,\Theta_B),\Theta_A)) - Q_A(S_i,a_i,\Theta_B))^2$$

Algorithm 1 : Double Q-learning (Hasselt et al., 2015)

Initialize primary network Q_θ, target network $Q_{\theta'}$, replay buffer $\mathcal{D}$, $\tau << 1$
for each iteration **do**
  for each environment step **do**
    Observe state s_t and select $a_t \sim \pi(a_t, s_t)$
    Execute a_t and observe next state s_{t+1} and reward $r_t = R(s_t, a_t)$
    Store (s_t, a_t, r_t, s_{t+1}) in replay buffer $\mathcal{D}$
  for each update step **do**
    sample $e_t = (s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}$
    Compute target Q value:
      $Q^*(s_t,a_t) \approx r_t + \gamma\, Q_\theta(s_{t+1}, argmax_{a'} Q_{\theta'}(s_{t+1},a'))$
    Perform gradient descent step on $(Q^*(s_t,a_t) - Q_\theta(s_t,a_t))^2$
    Update target network parameters:
      $\theta' \leftarrow \tau * \theta + (1-\tau) * \theta'$

Our Solution

We propose the deep RL framework that is built on top on DDQN. There is series of '*n*' agents each running independent from each other. The replay memory is the shared database of experiences. Each agent learns on a mini-batch from the replay memory.

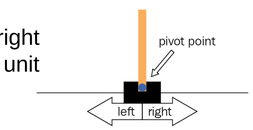


Results

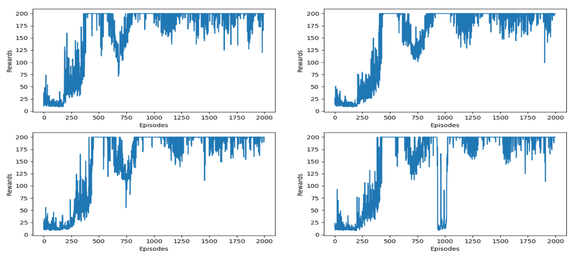
We test our framework on two OpenAI gym environments – (i) Cartpole (ii) Acrobot against three deep RL algorithms.

Case Study: Cartpole

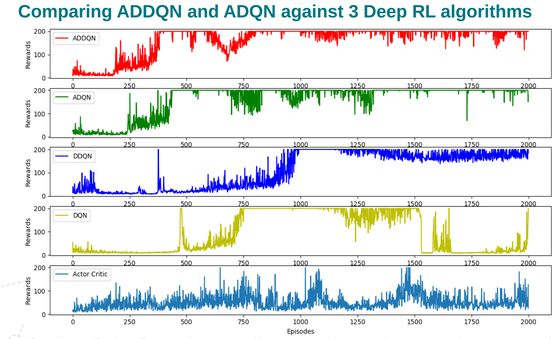
- A cart with a pole attached to it
- The cart is movable left and right along the x-axis by applying a unit amount of force in either sides
- Action space: [-1,1]



Performance among 4 agents



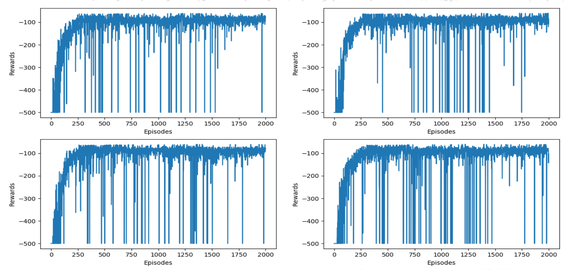
Comparing ADDQN and ADQN against 3 Deep RL algorithms



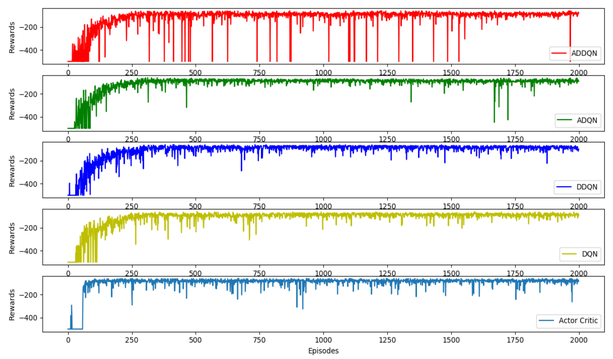
Case Study: Acrobot-v1

- Acrobat system has two robotic arms or links connected by two joints
- The goal is to swing the links such that the bottom of the lower robotic arm is raised upward beyond a particular height.
- The agent is rewarded -1 for every discrete time step elapsed.

Performance among 4 agents



Comparing ADDQN and ADQN against 3 Deep RL algorithms



Conclusion

As the result, we have showed that our proposed asynchronous structure with a shared replay memory between multiple agents expedites the learning process as they can explore the environment much faster. In particular, a fast learning rate is shown in Cartpole environment. As a future study, this work can be extended to much complex environments.

References

- Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning. In Thirtieth AAAI conference on artificial intelligence 2016 Mar 2.
- CSE 4/510 Reinforcement Learning Course lecture slides

Contacts: {luckyson, aanand8}@buffalo.edu