

Abstract

I present a comparison of computational efficiency between training an agent for playing Pong in the Atari environment in a sequential manner and in a parallelized manner. The agent is trained using a Deep Reinforcement Learning Advantage Actor Critic (A2C). The parallel version of the algorithm Asynchronous A2C (A3C) is analyzed using multiple CPU cores on a single machine instead of special hardware like GPU(Graphical Processing Units) and TPU(Tensor Processing Units).

Keywords

Advantage Actor Critic, Asynchronous, Synchronous

Introduction

Reinforcement Learning algorithms are a class of algorithms that have proven to be quite promising in the field of decision making. That makes games a suitable area for their application. Combine them with Deep Learning which is known to produce rich representations of input data (visual data in this case) and we get the ability to play games at a human or even better-than-human level (1,2). The game chosen here is Pong which is an atari game and is provided as an environment in OpenAI gym (3).

Reinforcement Learning

The setup consists of an agent A that interacts with the environment E . The agent interprets the environment by moving from one state s_t at time step t to another state s_{t+1} at time step $t + 1$. In each state it receives an observation in the form of an n -dimensional tensor which might contain partial or all information about the environment in that state. It chooses an action from an action space $A = (a_1, a_2, \dots, a_m)$ based on some computation and as a result lands in another state and receives a signal called the reward R_t quantifying how good or bad the action was. The agent keeps accumulating these rewards until it reaches the goal or the episode terminates. An episode runs for a predefined t_{max} number of time steps.

The action at each time step is taken by the agent in accordance with a policy π that is either learned as in Policy Gradient methods or predefined as in Value Function Based methods.

A policy is a function that given a state s , gives a probability distribution over A .

The value of a state s under a policy π is the expected return when starting in s and following π thereafter. Similarly the value of taking action a in state s under policy π is the expected return starting from s , taking the action a and thereafter following policy π .

Advantage Actor Critic

The agent here consists of 2 neural networks. One of them, the Actor, spits out a probability distribution over a set of actions A at each time step and learns the optimal policy, based on a loss function, more about which is described later. The other neural network, the Critic calculates the action-value at each time step telling how good or bad the action taken is and learns to give better estimates of the value, that is learns to be a better critic.

Advantage is a function that tells how good or bad the actor network performed relative to the critic's expectations.

Loss Function

Let us have a look at the components one by one and finally bring them together

1. We use the estimated value by the critic as a baseline to tell how good or bad the actor network is performing. For this we use the advantage function

$$\gamma^t R_T - V_\pi(t)$$

where $V_\pi(t)$ denotes the value estimated at time step t . γ is the discount factor used to place decaying importance, in this case, on later time steps so that earlier steps are reinforced. In Pong, if the opponent misses, it was probably because the shooter maneuvered the ball in a good way on receiving. Similarly earlier actions had a greater role to play in helping the agent becoming a better player.

2. The log probability of the sampled action is given by

$$\log \max P(a_i|s)$$

where $a_i \in A$ One of the reasons why we use log probabilities is that probabilities are confined in $[0,1]$ while log-probabilities are bounded in $(-\infty,0)$ and hence a finite number of bits can represent a wide range of values without overflowing or underflowing the computer's numerical precision.

3. The Actor loss is given by

$$\sum_t -\log \max P_t(a_i|s)(\gamma^t R_T - V_\pi(t))$$

4. The Critic loss is given by

$$\sum_t (\gamma^t R_T - V_\pi(t))^2$$

5. Finally the agent's loss is given by

$$actorLoss + c.CriticLoss$$

where c is a parameter used to control the relative learning of the actor and the critic

The implementation is done in a way such that the actor and the critic have some parameters in common. The critic does not update these shared parameters because it can lead to instability.

Parallelized Setup

The setup consists of multiple cores being used, each of which belongs to the same system. Each process gets its own copy of environment and agents and each agent learns independently and updates the parameters of the model asynchronously [BA3C reference] or synchronously [reference]. In the asynchronous approach, the different processes share the model's parameters and hence update the parameters independently, however these can lead to stale gradients[reference] as a process might be working on outdated parameters. In the synchronous setting, there is a central process which waits for all the processes to send their updated weights, averages them and then updates the actual model which is then shared to the other processes and the iterations continue.

Atari Pong

Pong is one of the Atari games.[reference]. Each state is available as a 3 dimensional tensor of dimensions 3 X 210 X 160 where 3 represents the RGB channels. Each element is an integer in $[0,255]$. There are six actions with three of them being redundant. FIRE equals NOOP, LEFT equals LEFTFIRE and RIGHT equals RIGHTFIRE. LEFT and RIGHT move the agent in the corresponding directions while NOOP keeps the agent in its previous position. The agent is displayed on the right and the enemy on the left. (Artificial Intelligence, Martin Atzmueller, Wouter Duivesteijn) If the agent misses the ball a reward of -1 is received, if the enemy misses the ball a reward of +1 is received. In all the other states a reward of 0 is received. The first to score 21 goals wins and that marks the end of the episode.

A reward is what the agent implicitly makes use of to learn while the points displayed on the screen are the scores of each of the players. The winner is the first player to score 21.

Implementation

The model is a 2-headed neural network. One head is the Actor and the other, the Critic. The first few layers of the network are shared. Only fully connected layers have been used. The input is flattened giving an input size of 100800. The entire frame is not needed for decision making as is shown in the figure. So the input is cropped and converted to grayscale before feeding into the network. That brings down the input size to 21190 since each frame is now 1 X 210 X 160.

The output of the Actor head consists of three nodes giving a probability distribution over the action space. The output of the Critic head has one node because it gives a scalar. The learning rate chosen is 10^{-3} and the optimizer used is Adam. [Appropriate reference] The Critic network does not update the shared parameters.