Autonomous Agents

Assault game - A3C agent

2016030010-Kosmas Pinitas

Technical University of Crete

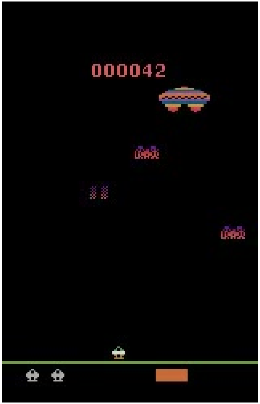
February 23, 2020

Outline

Background

Environment

- states: 4 grayscaled images (84 x 84)
- actions: 7 supported actions (6 permitted actions)
  - do nothing, shoot, move left, move right, shoot left, shoot right



Background
MDPs

A Markov Decision Process (MDP) is a set $(S, A, P_\alpha, R_\alpha)$ where:

- S is a finite set of states,
- A is a finite set of actions,
- P_α is the probability that action α in state s at time t will lead to state s' at time $t+1$,
- R_α is the immediate reward (or expected immediate reward) received after transitioning from state s to state s', due to action α

Background

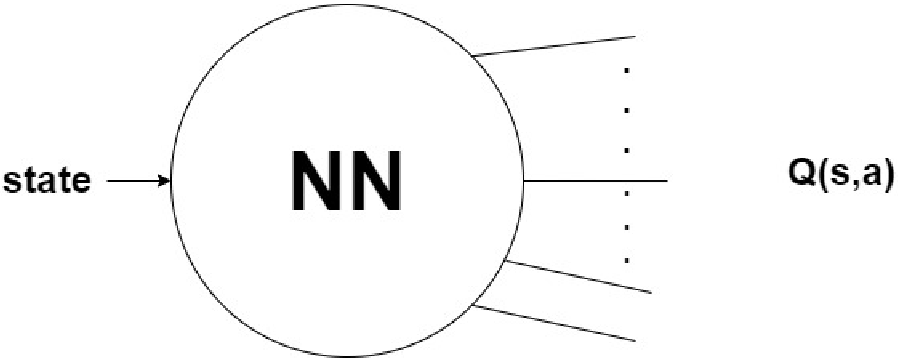
Q-Learning

The goal of Q-learning is to learn a policy, which tells an agent what action to take under what circumstances. It does not require a model of the environment, and it can handle problems with stochastic transitions and rewards.

$$Q^{new}(s_t, \alpha_t) = Q(s_t, \alpha_t) + a \cdot (r_t + \gamma \cdot max_{\alpha}\{Q(s_{t+1}, \alpha)\} - Q(s_t, \alpha_t))$$

- r_t is the reward received when moving from state s_t to state s_{t+1} ,
- a is the learning rate or step size and determines to what extent newly acquired information overrides old information,
- γ is the discount factor and determines the importance of future rewards.
- For problems with big dimensionality we use a neural network as Q approximator in order to reduce the complexity (Deep Q-Learning)

Background
Q-Learning (Cont.)

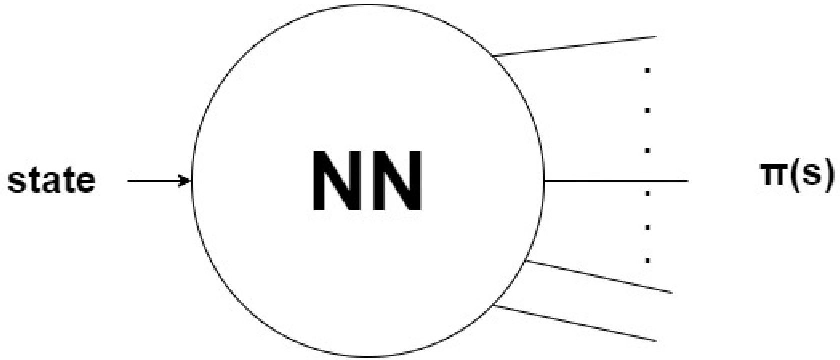


Background

Policy-gradients

- Direct approximation of policy function $\pi(s)$,
- $J(\pi) = E_{\rho^{s_0}}[V(s+0)]$ (Objective function)
- $\nabla_\theta\ J(\pi) = E_{s\sim\rho^\pi,\ a\sim\pi(s)}[A(s,a) \cdot \nabla_\theta\ log\ \pi(a\|s)]$ (Gradient)
  - $\nabla_\theta\ log\ \pi(a\|s)$ tells us a direction in which logged probability of taking action α in state s rises
  - $A(s,a)$ is a scalar value and tells us what's the advantage of taking this action.
  - If we combine the above terms , we will see that the likelihood of actions that are better than average is increased, and the likelihood of actions worse than average is decreased.

Background
Policy-gradients (Cont.)

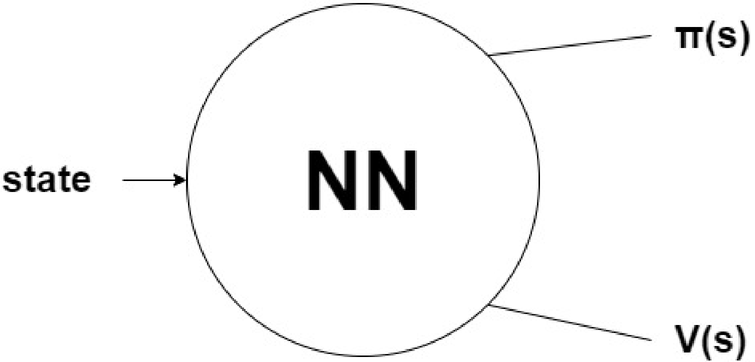


A3C

Definition

- **Asynchronous**
  - Multiple agents in parallel and each one has its own network parameters and a copy of the environment.
  - This agents learn only from their respective environments
  - As each agent gains more knowledge, it contributes to the total knowledge of the global network
- **Advantage**
  - $A(s, a) = Q(s, a) - V(s) = r + \gamma V(s') - V(s)$
  - Expresses how good it is to take an action α in a state s compared to average.
- **Actor-Critic**
  - Combines the best parts of Policy-Gradient and Value-Iteration methods.
  - Predicts both the value function $V(s)$ as well as the optimal policy function $\pi(s)$.
  - Agent uses the value of the Value function (Critic) to update the optimal policy function (Actor) (stochastic policy)

A3C
Actor-Critic Network



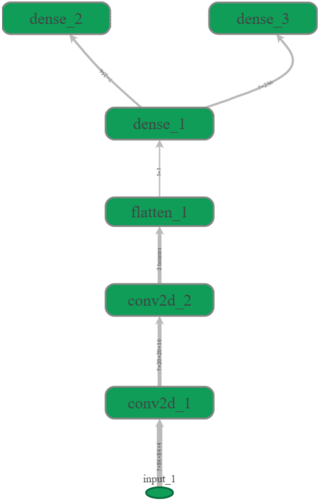
A3C

Advantages

- Faster and more robust than the standard Reinforcement Learning Algorithms.
- Performs better than the other Reinforcement learning techniques because of the diversification of knowledge.
- It can be used on discrete as well as continuous action spaces.

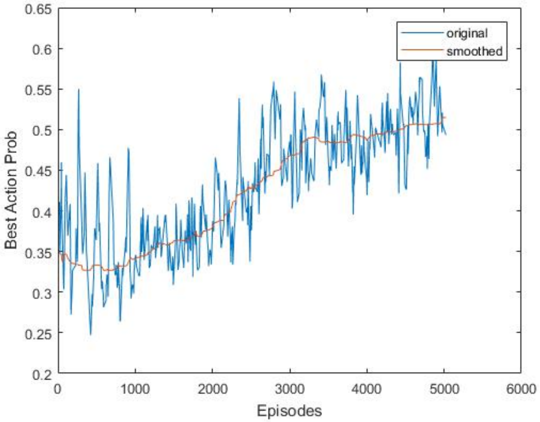
Model

Architecture



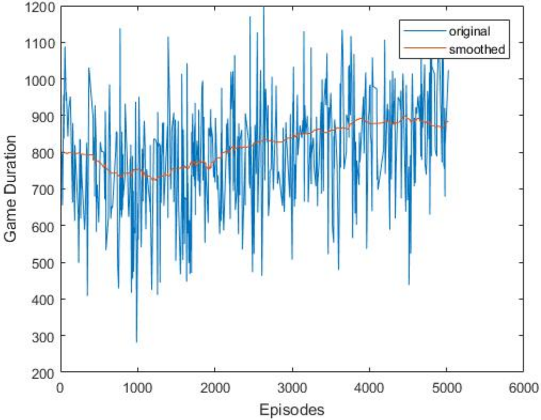
Model

Results



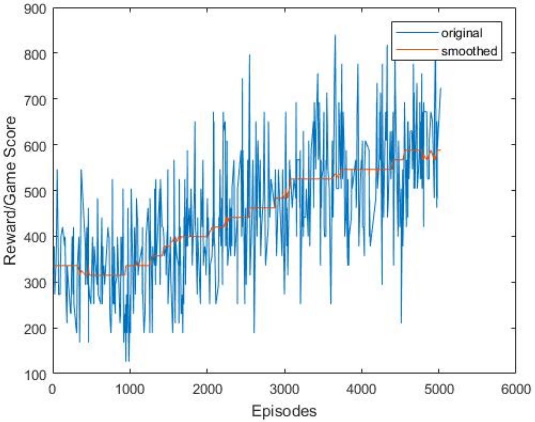
Model

Results (Cont.))



Model

Results (Cont.))



References

- environment: https://gym.openai.com/envs/Assault-ram-v0/
- MDP: https://en.wikipedia.org/wiki/Markov_decision_process
- Q-Learning: https://en.wikipedia.org/wiki/Q-learning
- Policy-Gradients: https://jaromiru.com/2017/02/16/lets-make-an-a3c-theory/
- A3C
  - https://jaromiru.com/2017/02/16/lets-make-an-a3c-theory/
  - https://www.geeksforgeeks.org/asynchronous-advantage-actor-critic-a3c-algorithm/