Technische Universität Kaiserslautern

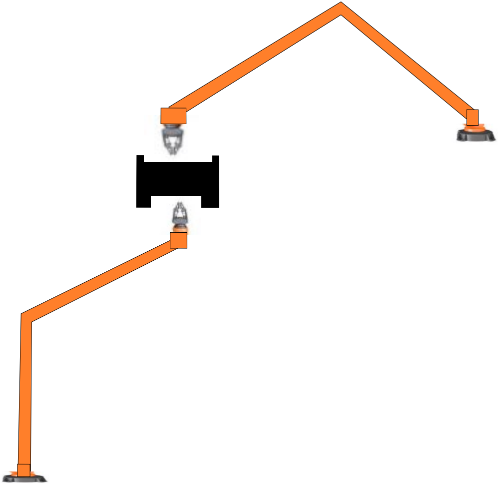
Reinforcement Learning for Multi-Agent Systems

Adalbert Mwombeni

Mechanical and Process Engineering
University of Kaiserslautern

20.04.2018

Motivation



- The black boxes are coming randomly in the area of two robots.
- The mission is completed when all two robots reach the black box.

Content

Multi-Agent Markov Decision Process

- Multi-agent Markov games can be defined by N agents with:
  - A set of observation $o_1, ..., o_N$
  - A set of actions $a_1, ..., a_N$
  - A set of states S
  - A state transition function $T : S \times a_1 \times a_2 \times ... \times a_N \rightarrow S$ which determines the Markov process.
- Each agent i interacts with environment by taking actions following its policy $\pi : o_i \times a_i \rightarrow [0, 1]$, transformed into the next state and gets a reward.
- The reward $r_i : S \times a_i \rightarrow R$ judges the policy's performance.
- Each agent tries to maximize the accumulated discounted return

$$R = \sum_{t=0}^{T} \gamma^t r_i^t,$$

where T is the expected time horizon and γ is the discount parameter.

Q-Learning and DQN

- Q learning is a traditional value iteration reinforcement learning method, which update a single Q value based on Bellman equation.

$$
\begin{aligned}
Q(s,a) &= \sum_{t=t_0}^{T} r_t \gamma^{t-t_0} | s_{t_0} = s, a_{t_0} = a \\
V(s) &= \sum_{t=t_0}^{T} r_t \gamma^{t-t_0} | s_{t_0} = s
\end{aligned}
\tag{1}
$$

- Tubular methods are often used to solve simple problems with finite states.
- The iteration formula can be written as:

$$
\begin{aligned}
Q(s,a) &= r + \gamma Q(s^{'}, a^{'}) \\
a^{'} &= arg \max_{a} Q(s^{'}, a)
\end{aligned}
\tag{2}
$$

Deep Q Network

- Deep neural network function approximation are used to estimate the action-value function.
- The temporal difference function (TD) loss function can be written as:

$$
\begin{aligned}
E_{s,a,r,s'} &= (Q(s,a|\theta) - y)^2 \\
y &= r + \gamma Q(s^{'}, a^{'}|\theta^{'}) \\
a^{'} &= arg \max_a Q(s^{'}, a|\theta^{'}),
\end{aligned} \tag{3}
$$

where θ represents the current Q function parameter.

Drawbacks of DQN

- While DQN solves problems with high dimensional observation spaces, it can only handle discrete and low dimensional action space.
- Many task of interest have continuous and high dimensional action spaces.
- DQN cannot be straightforwardly applied to continuous domains.
- Because DQN relies on a finding the action that maximizes the action-value function, which in the continuous valued case requires an iterative optimization process at every step.

Policy Gradient

- Policy gradient are widely used in reinforcement learning problems with continuous action spaces.
- The basic idea is to represent the policy by parametric probability distribution $\pi_\theta = \mathbb{P}[a|s;\theta]$ that stochastically selects action a in state s according to the parameter vector θ.
- Policy gradient algorithm typically proceeds by sampling this stochastic policy and adjusting the policy parameters in the direction of greater cumulative reward.

Policy Gradient

- The policy gradient maintains a parameterized function $\mu(s|\theta^{\mu})$ which specifies the current policy by deterministically mapping states to action.
- The function $Q(s, a)$ is learned using Bellman equation as in Q-learning.
- The update is done by applying the chain rule to the expected return from the start distribution J with respect to the policy parameter as follows:

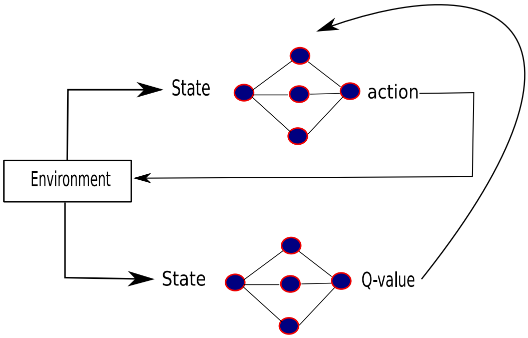
$$\begin{aligned} \nabla_{\theta^{\mu}} J &\approx \mathbb{E}_{s_t \sim \rho^{\beta}} [\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|s = s_t, a = \mu(s_t|\theta^{\mu})] \\ &= \mathbb{E}_{s_t \sim \rho^{\beta}} [\nabla_{a} Q(s, a|\theta^{Q})|s = s_t, a = \mu(s_t) \nabla_{\theta_{\mu}} \mu(s|\theta^{\mu})|s = s_t] \end{aligned} \tag{4}$$

ρ^{β} is visitation distribution, β is the behavior policy and ρ is the parameter.

Drawbacks of Policy Gradient

- The algorithm sees a lot of training examples of high rewards from good actions and negative rewards from bad actions.
- Algorithm increases the probability of good actions.
- Problem: the value function cannot be obtained until the current episode ends.

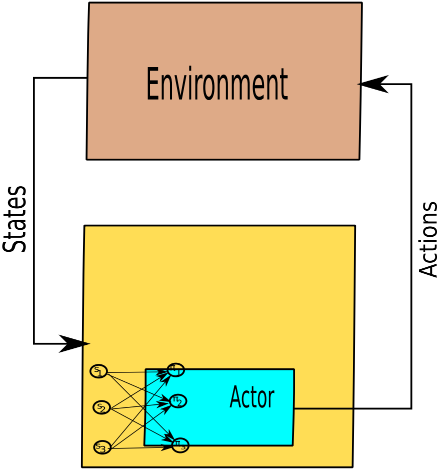
Idea for Improving Policy Gradient



- Split the model into two parts.
- One outputs the desired action in the continuous space.
- An other is taking action as input to produce Q-values

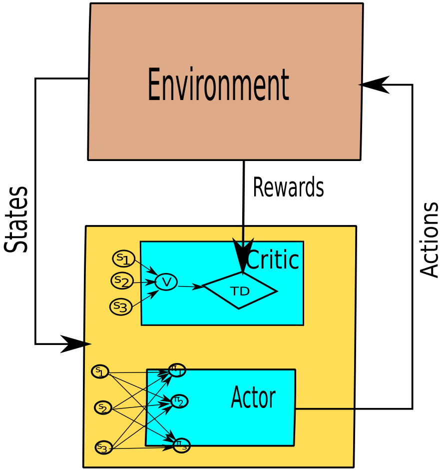
Actor Critic

- The actor critic model has two components: actor and critic.



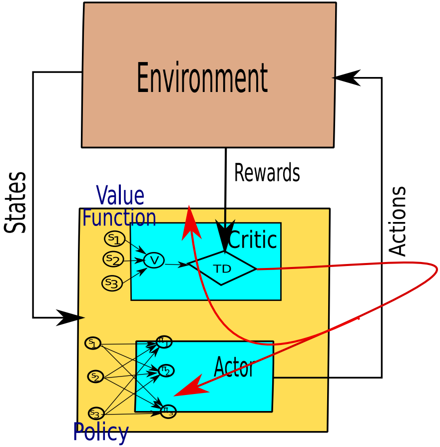
Actor Critic

- The actor takes the current environment state and determines the best action to take from there.

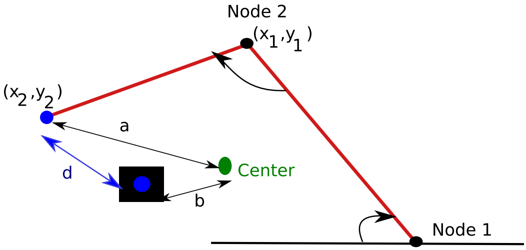


Actor Critic

- The critic plays evaluation role by taking environment state and action and returning a score that represents how good the action is for the state.



Environment



- $c = \{0, 1\}$: 1 means that the robot is grabbing the object.
- action = (node 1 angular velocity, node 2 angular velocity)
- state = $[a, b, c, x_1, y_1, x_2, y_2]$

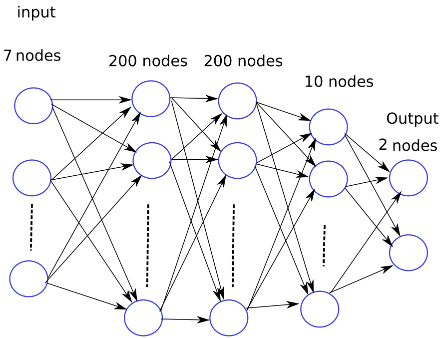
Reward

- The arm tries to get to the black box. The environment returns the distance for the arm to learn.
- $r = -\sqrt{(d)^2}$: The far away from black box the less reward.
- Touch black box: r+=1; stop at black box for a while then get r=+10.

Agent

- The agent needs two neural network, one for actor another one for critic.
- Input of actor network is the current state.
- The output is a real value representing action from a continuous action space.

Actor neural network



Agent

Critic neural network

- The critic must take both environment state and action as input and calculate evaluation.
- The output of critic is the estimated Q-value of the current state and action given by actor.
- The policy gradient theorem provides the update rule for the wights of the actor.
- The critic network is updated from the gradient obtained from the TD error signal.

Actor Critic

In order to store some experiences of the agent during the training, the experience replay is used. However each agent possesses its memory. Nevertheless,

- Agents share episodes and training.
- Agents do not share reward function.

Thank you for your attention