Continuous Control with Reinforcement Learning

Anders Hillmar Damm Andersen (s163926) & Jeppe Heini Mikkelsen (s152939)

DTU Automation and Control

Motivation

In this project design and implementation of a reinforcement learning algorithm for continuous control of a cart-pole is done, to showcase that reinforcement learning can be used for solving continuous control problems. Ultimately it is desired to make a comparison between RL-control design and standard control design such as Linear Quadratic Regulators (LQR). A popular framework for testing and developing physics-based simulation and continuous-time reinforcement learning algorithms is the *DeepMind Control Suite* (DMCS) driven by the *MuJoCo* physics engines, which is used in this project.

Reinforcement Learning

The goal of reinforcement learning is to train a network to perform some task without explicitly telling it what to do but by giving it a reward when it takes a correct action. The type of problems that RL can solve is processes which can be described as a Markov decision process (MDP). In this project a deterministic MDP is used, which is a tuple

$$\langle S, A, F_a(s_t, a_t), R_a(s_t, s_{t+1})\rangle \tag{1}$$

consisting of a set of states S in an environment E, a set of possible actions A that an agent can take, a state transition function $s_{t+1} = F_a(s_t, a_t)$ and a reward function $r_t = R_a(s_t, s_{t+1})$ which is the metric for how well the action was for the given state. The goal of reinforcement learning is to maximise the discounted cumulative reward

$$R_t = \sum_{i=t}^{t+T} \gamma^{(i-t)} r_i, \quad 0 \leq \gamma \leq 1. \tag{2}$$

RL methods

Method	Continuous control	Sample efficient
DQN	×	✓
Policy gradient	✓	×
Actor-critic	✓	✓

Actor-Critic Method

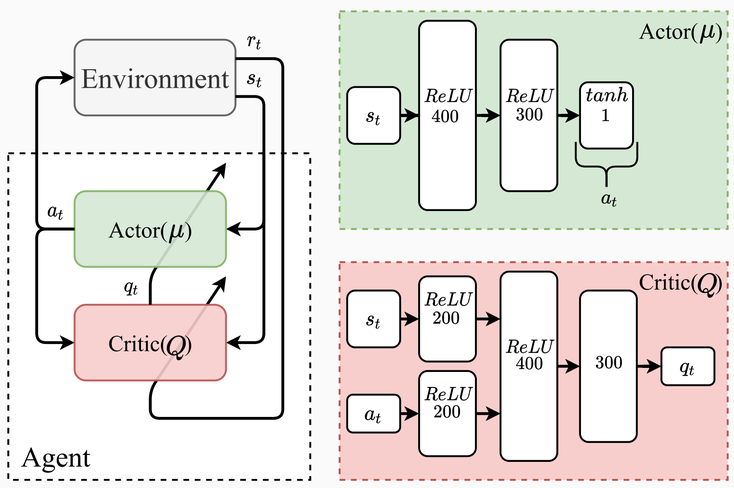

Fig. 1: *Actor-Critic arcitecture for DDPG algorithm*

$$a_t = \mu(s_t), \quad q_t = Q(s_t, a_t) \tag{3}$$

In actor-critic methods there are two networks, an actor and a critic. The actor chooses an action a_t, given a state s_t, to enact on the environment. The critic is used to evaluate the action by producing Q-values q_t. In Q-learning if the optimal state action-value function Q^* is known then in any given state the optimal action a_t^* is found

$$a_t^* = arg\max_{a_t} Q^*(s_t, a_t) \tag{4}$$

where Q^* satisfy the *Bellman optimality equation*:

$$Q^*(s_t, a_t) = r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \tag{5}$$

With Actor-Critic methods a function approximator to Q^* is learned: $\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) \approx Q(s_t, \mu(s_t))$.

Training the actor-critic network

The actor-critic network is implemented as two feedforward neural nets that are trained using stochastic gradient descent. The goal of the actor is to choose the action that maximises the cumulative reward and the goal of the critic is to approximate the Bellman equation.

$$\mathcal{L}_\mu = -\frac{1}{T}\sum_{t=0}^{T} Q(s_t, a_t), \quad \mathcal{L}_Q = \frac{1}{T}\sum_{t=0}^{T}[Q(s_t, a_t) - (r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}))]^2 \tag{6}$$

where $\mu'(s_t)$ and $Q'(s_t, a_t)$ are a delayed version of the actor and critic referred to as *target*-networks with the following update law

$$\mu' = \tau\mu + (1-\tau)\mu', \quad Q' = \tau Q + (1-\tau)Q', \quad 0 \leq \tau \leq 1. \tag{7}$$

Baseline

For this project the actor-critic method called Deep Deterministic Policy Gradients (DDPG) is used. The pseudocode of DDPG is presented in Algorithm 1.

Pseudocode

Algorithm 1: DDPG Algorithm

Randomly initialise critic $Q(s,a)$, target critic $Q'(s,a)$, actor $\mu(s,a)$ and target actor $\mu'(s,a)$
for *episode 1 to M* **do**
  Initialise environment E
  Initialise random process ε
  for t *= 1 to* T **do**
    Select action $a_t = \mu(s_t) + \varepsilon_t$
    Execute a_t on E
    if *done* **then**
      Terminate episode
    Store transition $\langle s_t, a_t, r_t, s_{t+1}\rangle$ in replay memory $\mathcal{M}$
    if *size of* $\mathcal{M} >$ *2000* **then**
      Sample random batch $\mathcal{B} = \langle s, a, r, s^*\rangle$ from $\mathcal{M}$
      Set $y = r + \gamma Q'(s^*, \mu'(s^*))$
      Update critic by minimising loss $\mathcal{L}_Q = [y - Q(s,a)]^2$
      Update actor by minimising loss $\mathcal{L}_\mu = -Q(s,a)$
      Update target networks using soft-update $Q' = \tau Q + (1-\tau)Q'$, $\mu' = \tau\mu + (1-\tau)\mu'$
  end
end

The network architecture is based on the actor-critic architecture seen in Fig. 1 with network layers of actor and critic both with 400 and 300 hidden units. The hyperparameters can be seen in the table below.

Hyperparameters

lr μ	lr Q	γ	batch size	memory size	τ	# episodes
0.0005	0.001	0.99	200	$1e^6$	0.005	200

Exploration & Exploitation

For RL to train it is necessary that initially the environment is explored such that the optimal policy can be found. Later in the training the policy is exploited more such that the policy converges to the optimal policy. For actor-critic networks two approaches can be used; *action noise* and *parameter space noise*.

$$\underbrace{a_t = \mu(s) + \varepsilon}_{\text{action space noise}}, \quad \underbrace{a_t = \mu(s|\tilde{\vartheta}_\mu), \quad \tilde{\vartheta}_\mu = \vartheta_\mu + \varepsilon}_{\text{parameter space noise}}, \quad \varepsilon \sim \mathcal{N}. \tag{8}$$

In action space noise, noise is added to the output of the actor network. In parameter space noise, noise is added to the parameters of the actor network. Both Ornstein-Uhlenbeck (OU) process noise and white Gaussian noise is investigated for action space noise. The parameter noise is also white Gaussian noise.

Results

Application and exploration of noise have been of main focus. Four different methods each based on averaging 8 tests for exploration & exploitation is seen on Fig. 2

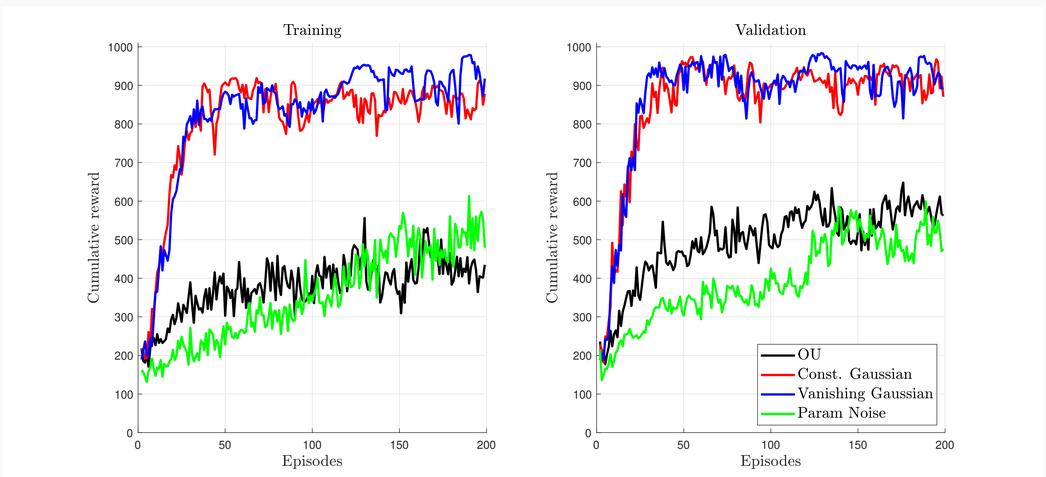

Fig. 2: *Cumulative reward for cart-pole with four noise sources. Max reward is 1000.*

Conclusion

In this project a sample efficient Reinforcement learning algorithm for continuous action spaces called Deep Deterministic Policy Gradients (DDPG) has successfully been implemented on the classical cart-pole problem. The focus has been on investigating different strategies for exploration and exploitation in which different stochastic processes for actions space have been compared to parameter space noise. The Ornstein-Uhlenbeck noise which was proposed by the original Authors of the DDPG and parameter space noise did not yield any satisfying results while simple Gaussian noise both constant and vanishing did. We have found that we are able to converge to a cumulative reward of approximately 900 out of a possible 1000 within 50,000-100,000 time steps. Future work will be to design Linear Quadratic Regulators from modern control theory to benchmark the DDPG.