Project: Smart three-sphere swimmer near a wall

2.2 Introduction to deep reinforcement learning

Luca Berti, L.G., C.P.

18 October 2021

State spaces and action spaces

In the previous presentation we had few questions concerning reinforcement learning.

- can the state space be discrete, continuous?
- can actions space be discrete, continuous?
- how does "learning" actually happens when we have these elements?
- what is the mathematical theory hiding behind all of this?

State spaces and action spaces

In the previous presentation we had few questions concerning reinforcement learning.

- can the state space be discrete, continuous?
- can actions space be discrete, continuous?
- how does "learning" actually happens when we have these elements?
- what is the mathematical theory hiding behind all of this?

Action and state spaces can be discrete or continuous!

State spaces and action spaces

In the previous presentation we had few questions concerning reinforcement learning.

- can the state space be discrete, continuous?
- can actions space be discrete, continuous?
- how does "learning" actually happens when we have these elements?
- what is the mathematical theory hiding behind all of this?

Action and state spaces can be discrete or continuous! However,

- dealing with large state spaces is now common practice/possible (RL with function approximation)
- dealing with large/continuous action spaces has been a topic of active research in the recent years

RL with function approximation

Function approximation → use parametrized functions that can approximate the value function (ex. Neural Networks, Radial Basis Functions)

Parameters will be obtained via training while learning (ex. loss function minimisation, back-propagation)

Using function approximation, the hope is to construct a model that can be generalised (impossible to sample the whole space!)

Three aspects are present when using NN with RL

- Storing results for experience replay
- Using of a companion ("target") network
- Using batch training

Example: Deep Q-Network

Approximate $Q(s, a) \approx Q(s, a; \theta)$, where θ are the weights of the NN BUT

- training of NN is done with independent data (but data from RL are correlated) Hence, store learning data and use them to train the network
- training of NN is done with identically distributed data (but the training data distribution is non-stationary as the agent learns) Hence, copy the weights, use them to compute the temporal difference and update this copy less frequently.

Example: Deep Q-Network

Require: Initialize replay memory D to capacity N
  Initialize action-value function Q with random weights θ
  Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
  for episode 1, M **do**
    Initialize sequence $s_1 = x_1$ and preprocessed sequence $\phi_1 = \phi(s_1)$
    for t = 1, T **do**
      With probability ϵ select a random action a_t
      otherwise select $a_t = \arg\max_a Q(\phi(s_t), a; \theta)$
      Execute action at in the emulator and observe reward r_t and image x_{t+1}
      Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
      Store experience $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
      Sample random minibatch of experiences $(\phi_t, a_t, r_t, \phi_{t+1})$ from D
      $y_j = r_j + \gamma \hat{Q}(\phi_{j+1}, a_j; \theta^{-})$
      Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the weights θ
      Every c steps reset $\hat{Q} = Q$
    end for
  end for

Applications in fluid mechanics

DRL has been used in fluid mechanics for different purposes:

- Fish swimming synchronisation [DQN]
- collective swimming [DQN with LSTM layers]
- control strategies for active flow control [PPO]

Paul Garnier, Jonathan Viquerat, Jean Rabault, Aurélien Larcher, Alexander Kuhnle, Elie Hachem, A review on deep reinforcement learning for fluid mechanics, Computers & Fluids, Volume 225, 2021.

LSTM cell = Long-Short Term Memory → particular type of recurrent NN cell

PPO = proximal policy optimisation

Our case: Three sphere swimmer next to a wall

Important (state?) variables:

- N: number of spheres
- Length of the arms ($N - 1$ variables)
- Orientation of the swimmer (1 variable)
- Distance from the wall (of all the spheres? just the centre of mass of the swimmer?)

Possible actions:

- Elongate one arm
- Shrink one arm

What about control of multiple arms at the same time?

Reward:

- Swimming at a constant height
- Reaching a target