RL - Reinforcement Learning

Exercises on Neural Fitted Q (NFQ)

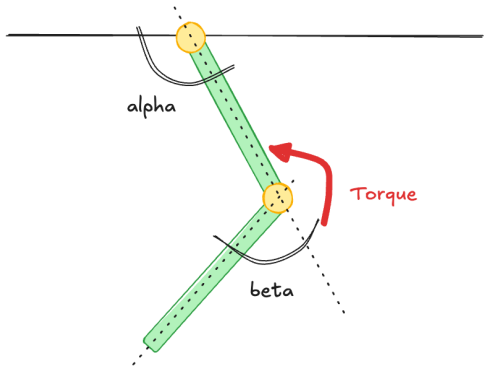
Prof. Riccardo Berta

2025.10.03

0.1 Acrobot

The Acrobot is a classic problem in control theory. The environment consists of two links connected linearly to form a chain, with one end of the chain fixed. The joint between the two links is actuated. The goal is to apply torques on the actuated joint to swing the free end of the linear chain above a given height while starting from the initial state of hanging downwards.

- The action space is discrete, deterministic, and represents the torque applied on the actuated joint between the two links: (0) apply -1 torque, (1) apply 0 torque to the actuated joint, (2) apply 1 torque to the actuated joint
- The observation space is a 6-dimensional space with the sine and cosine of the two joint angles and their angular velocities.



The goal is to have the free-end reach a designated target height in as few steps as possible, and as such all steps that do not reach the goal incur a reward of -1. Achieving the target height results in termination with a reward of 0. The reward threshold is -100.

1 - import the environment from Gymnasium and show it with a random policy

```
# YOUR CODE HERE

# You can get the environment from Gymnasium 'Acrobot-v1';
# In order to visually plot the environment you can import it
# using render_mode="rgb_array". Get also th state size
# and action_size.
```

2 - Show the policy in action by rendering the environment several times after different decisions from a random policy:

```
# YOUR CODE HERE

# You can use the same function already used for the cartpole environment to
# implement a random aget and to show the policy.
```

3 - Create a neural network as a function approximator for the Q-function.

```
# YOUR CODE HERE

# Add the code for a neural network using the PyTorch library and
# the code of an optimizer as in the cartpole example
;
```

4 - Add the code for the experience buffer

```
# YOUR CODE HERE

# Add the code for the experience buffer as in the cartpole example
```

5 - Add the code of the exploration strategy based on epsilon-greedy method:

```
# YOUR CODE HERE

# Add the code for an exploration strategy based on the epsilon-greedy algorithm
```

6 - Add the code to train the neural network using the collected experiences:

```
# YOUR CODE HERE

# Re-use the optimize method as in the cartpole example
```

7 - Write the code of the NFQ algorithm

```
# YOUR CODE HERE

# Re-use the optimize method as in the cartpole example
```

8 - Apply NFQ to the environment

```
# YOUR CODE HERE

# Apply NFQ to the acrobot environment

# define hyperparameters values
```

9 - Evaluate the learned policy

```
# YOUR CODE HERE

# Write a function to evaluate its performance
```

10 - Compare performace of random policy vs the one learned by the neural network

```
# YOUR CODE HERE

# Run the evaluation on the two policies and compare
```

11 - Show the policy behavior

```
# YOUR CODE HERE

# You can show the policy learned by the agent using the show_policy function
```