# Lab Report

Ruyi TANG, Guanyu CHEN

## Exercice 1: FeatureFilterWrapper

This wrapper is designed to filter out specific feature from the observation space. By removing certain features from the environment’s observation, the agent must learn to operate with partial information, which can help in analyzing the importance of certain features for decision-making.

### Key parameters:

* env: For the environment to be wrapped and from which we remove the designated features, we use:gym.make('CartPoleContinuous-v1')to create the environment.
* feature\_index: The index of the feature to be removed from the observation space. And in the exercice, we need to filter the features of **velocity (index=1)** and **angular velocity (index=3)**.

### Functions:

All the wrappers in this lab have three functions:\_\_init\_\_(), reset(), and step() but are designed differently for each wrapper.

* \_\_init\_\_(): In this function, we designate the index of features to be removed and adjust the observation space to delete the designated features.
* reset(): When the environment is reset, it returns the observation with the designated feature removed.
* step(): In this function we execute the action to remove the designated features and return the filtered observation.

The algorithm of feature filtering is to test whether the agent can learn and adapt without certain features in the environment.

## Exercice 2: ObsTimeExtensionWrapper

This wrapper extends the observation space by including both the current observation and the previous one. This effectively provides the agent with a memory of the last state, which can be useful in partially observable environments, where the current observation alone may not provide enough information to make an optimal decision.

### Key parameters:

* env: we continue to use the same environment to be wrapped

### Functions:

* \_\_init\_\_(): In this function, we get the shape of the observation space and extend it by concatenating the current observation with the previous one, meaning double the dimensional size of the observation space and we finally initialize to save the previous observations.
* reset(): Upon resetting, the previous observation is initialized to one, and we extend the the observation space with current observation..
* step(): In this function the previous observation is updated to be the current observation, and the observation returned to the agent includes both the previous and current observations.

The algorithm of observation extension is to provide additional temporal information (current + previous state) to the agent.

## Exercice 3: AcrionTimeExtensionWrapper

This wrapper extends the action space by allowing the agent to output a sequence of **M** actions at once, instead of a single action. The environment only executes the first action in the sequence during each step, but the wrapper allows for experiments where an agent can plan or provide action sequences in advance.

### Key parameters:

* env: we continue to use the same environment to be wrapped.
* M: The number of actions in the extended action sequence. This parameter determines how many actions are outputted by the agent at each step, although only the first action will be executed. And in the experiment, we first give results for cases when **M=2**, and in the Bonus part, we will show results when **M=3**.

### Functions:

* \_\_init\_\_(): In this function, we Store the original action space and Extend the action space size to M times the original action space.
* reset(): We call the reset method of the inner environment to initialize it and returns the initial observation.
* step(): In this function we Extract the first action from the extended action. Then we clip the action to make sure it's within the action space bounds.

The algorithm of action extension is for the agent to plan over an extended horizon by providing action sequences, but only the first action is used, which could be useful for exploring environments where action sequences are meaningful, or for agents that need to learn to refine long-term action planning.

## Exercice 4: