Reinforcement Learning WS 2025/26
TAs: Mikel Zhobro, Pavel Kolev
Due-date: **27.11.2025, 14:00** (upload to ILIAS as one file)
Filename: homework6-NAME1-NAME2-...zip
Homework Sheet 6
November 20, 2025

The goal of this homework is to get familiar with value approximation using neural networks. So we start using pytorch and the gymnasium framework.

1 Function approximation

(a) Show that tabular methods are a special case of linear function approximation. What would the feature vectors be?

(b) For polynomial features: Now consider a continous state space with $s \in \mathbb{R}^k$ (i.e. k numbers, $s_1, s_2, \ldots, s_k$, with each $s_i \in \mathbb{R}$). For this k-dimensional state space, each order-n polynomial-basis feature x_i can be written as

$$x_i(s) = \Pi_{j=1}^{k} s_j^{c_{i,j}} \tag{1}$$

where each $c_{i,j}$ is an integer in the set $\{0, 1, \ldots, n\}$ for an integer $n \geq 0$. These features make up the order-n polynomial basis for dimension k. Why does Eqn.1 define an $(n+1)^k$-dimensional feature space for dimension k?

2 Feature designing

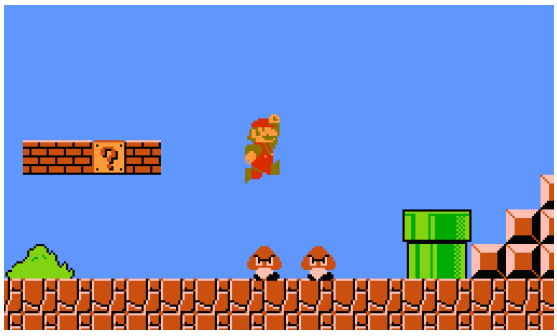


Figure 1: Super Mario Bros., ©NINTENDO 1985

(a) Imagine you want to implement a RL agent to play Super Mario (the original one). We want to learn a state value function linear in features $x(s)$. Please give an example feature vector and what each component encodes.

Hint: A feature vector is given by $x = (x_1, x_2, \dots,)$, where x_1, for example, encodes the x-coordinate of Mario, x_2 encodes the y-coordinate of Mario, etc. Please give a feature vector of at least 5 dimensions.

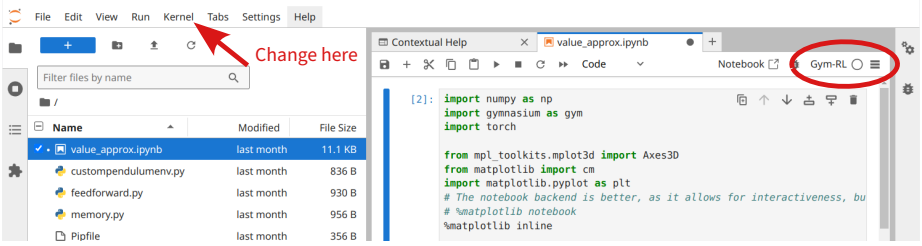
3 Value function fitting

Preparation

Download the code from ILIAS: `6-gym-valueapprox.zip`. Unzip and enter the containing folder. Then create a virtual environment. We strongly advice to use a virtual environment to keep you python installations isolated. However, you can reuse that env for the next homeworks. The simplest solution is to create a venv via (`python3 -m venv .venv`) in the folder, activate it (`. ./.venv/bin/activate`). Alternatively use `uv`. Then install all dependencies via `pip install -r requirements.txt`. You need to have `swig` installed on your system, e.g. via`apt install swig` on ubuntu/debian. *Hint:*[1] Alternatively, we also have a `poetry` file, or you can use `anaconda`.
The main code is in the notebook `value_approx.ipynb`. In order to make jupyter aware of the python kernel from the virtual pipenv environment: run

```
python -m ipykernel install --user --name=Gym-RL
```

Then you can fire up `jupyter-lab` and you should be ready to go. Make sure the right kernel is selected, shown on the top right:



[1] If you have problems with torch and cuda, you can fall back to the cpu version of torch. In the requirement file are already two lines to uncomment, for that case. You can also checkout: `https://stackoverflow.com/questions/51730880/where-do-i-get-a-cpu-only-version-of-pytorch`

Value function for the Pendulum

In this exercise, you will fit an approximated value function for a given policy. The code contains the following files:

policy.py This file implements a policy. It exposes a method *get_action* which you can use to query an action a for a given state s. (no changes required)

memory.py Implements a replay buffer with a fixed size that provides a method *add_transition* to add new transitions to the buffer and *sample* that, in our case, returns a tuple of the form $(s_t, a_t, r_{t+1}, s_{t+1}, d_{t+1})$ where s_t is the current and s_{t+1} the next state, a_t is the action and r_{t+1} the reward and d_{t+1} is the done signal (always false for this environment). (no changes required)

feedforward.py Implementation of a feed forward neural network. (No changes required)

value_plot.ipynb In this notebook, you will implement the value fitting with a function approximator (NN). The notebook provides already routines for data collection and value function visualization. (Changes required)

Tasks:

(a) Plot a trajectory of the system

(b) Use a neural network function approximator, to fit the value function. Use an appropriate target (TD update) for the value fitting with $\gamma = 0.95$. Monitor the learning progress.

(c) Plot the value function before and after learning

(d) Give an intuitive interpretation of the plotted value function.

(e) Compare the learning curve and the resulting value function when using $\gamma = 0.5$. What happens?