Region-based HER – Code and Results

Commands for reproducing results:

- First clone/fork gym from github.com/mbrucker07/gym (“master” branch)
- Then clone/fork EnergyBasedPrioritization from github.com/mbrucker07/EnergyBasedPrioritization
    - !!! Go to branch “new”!
- Be sure to have uninstalled any other baseline version except for the module within EnergyBased Prioritization!

```
# Push Energy adaptive

python baselines/her/experiment/train.py --env_name FetchPushNew-v1 --prioritization energy --n_epochs 50 --num_cpu 6 --
save_path ../energy_logs/pushobstacle_oneregionadaptive50_test1_energy --binding none


# Push Energy uniform (have to modify EnergyBasedPrioritization/her/experiment/config.py first!)

python baselines/her/experiment/train.py --env_name FetchPushNew-v1 --prioritization energy --n_epochs 50 --num_cpu 6 --
save_path ../energy_logs/pushobstacle_uniform50_test1_energy --binding none


# Push Energy final (have to modify EnergyBasedPrioritization/her/experiment/config.py first!)

python baselines/her/experiment/train.py --env_name FetchPushNew-v1 --prioritization energy --n_epochs 50 --num_cpu 6 --
save_path ../energy_logs/pushobstacle_final50_test1_energy --binding none


# Curling Energy adaptive

python baselines/her/experiment/train.py --env_name FetchCurling-v1 --prioritization energy --n_epochs 100 --num_cpu 6 --
save_path ../energy_logs/curling_oneregionadaptive100_test1_energy --binding none


# Curling Energy final (have to modify EnergyBasedPrioritization/her/experiment/config.py first!)

python baselines/her/experiment/train.py --env_name FetchCurling-v1 --prioritization energy --n_epochs 100 --num_cpu 6 --
save_path ../energy_logs/curling_final100_test1_energy --binding none


# Curling Energy uniform (have to modify EnergyBasedPrioritization/her/experiment/config.py first!)

python baselines/her/experiment/train.py --env_name FetchCurling-v1 --prioritization energy --n_epochs 100 --num_cpu 6 --
save_path ../energy_logs/curling_uniformt1-t3100_test1_energy --binding none
```

Code Explanation:

Files which have Region-Based HER components

- gym/envs/robotics/fetch/push_new.py: (and corresponding xml-file in assets)
    - Push-around-obstacle environment with defined regions
    - Adapt_dict with entries regions and probs defined in __init__ for communication with EBP
- EnergyBasedPrioritization/her/experiment/config.py (branch “new”!!)
    - Dict with train_modes: (probabilities and min_success_rates as described below)
    - For both FetchPushNew and FetchCurling
- EnergyBasedPrioritization/her/experiment/train.py (branch “new”!!)

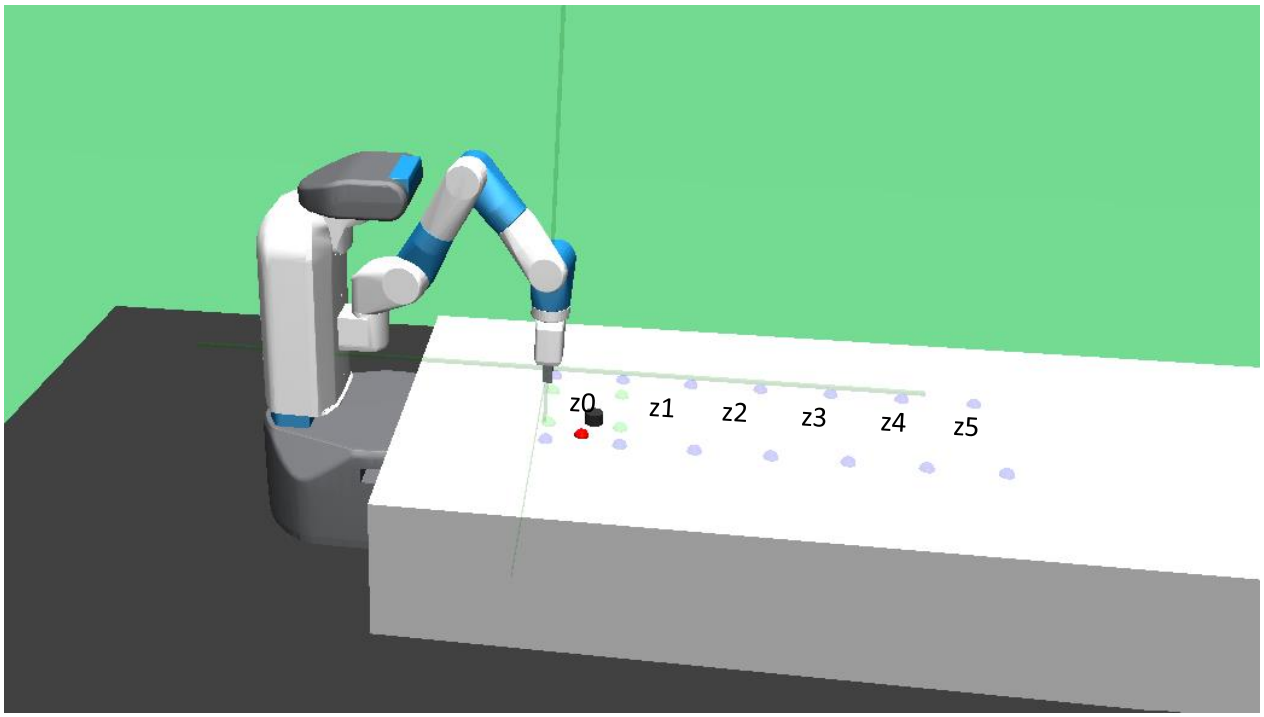
- In function launch: create one rollout workers and evaluator for each train_mode (probabilities + min_success_rate) specified in config.py
- In function train: after each epoch: check success_rate and switch to next train_mode rollout_worker if min_success_rate specified in config.py has been reached

Region-based HER: concrete implementation

- To apply region-based HER to environments, regions (z0, z1, z2, …) have to be defined (hand-crafted so far)
- We then define a sequence of training modes: For each training mode, we define
  - The training mode's **name**
  - The **probabilities** for sampling from each of the defined regions (when a region is chosen for sampling, we sample uniformly from that region).
  - The **min_success_rate** threshold, indicating when to advance to the next training mode

For examples, see the next section.

FetchCurling



- In the FetchCurling environment, six regions (z0, z1, z2, z3, z4, z5) were define as shown in the figure below
- The goal we are mainly interested in is to slide the puck to **region z3**

Adaptive Strategy

```
# in training order
"t0": [[1, 0, 0, 0, 0, 0], 0.6],
"t1": [[0, 1, 0, 0, 0, 0], 0.5],
"t2": [[0, 0, 1, 0, 0, 0], 0.4],
"t3": [[0, 0, 0, 1, 0, 0], 0.3],
"t4": [[0, 0, 0, 0, 1, 0], 0.2],
"t5": [[0, 0, 0, 0, 0, 1], 0.1],
```

- For the adaptive strategy, the sequence consists of 6 training modes (t0, t1, t2, t3, t4, t5).
    - The training starts with the first mode t0. In this mode, goals are sampled from z0 (corresponds to first entry of the array) with probability 1. Probability of sampling from the other regions in mode t0 is 0. Min_success_rate = 0.6, which means that we advance to the next training mode, as soon as a success_rate of 0.6 is reached.
    - When a success_rate of 0.6 is reached, we switch to training mode t1. In this mode, we sample from region z1 (corresponds to second entry of the array) with probability 1, and from the other regions with probability 0 until a minimum success rate of 0.5 FOR THIS TRAINING MODE is reached (evaluation is of course always based on the current training mode)
    - This goes through the entire sequence of training modes

Uniform Strategy

```
# in training order
"uniform": [[1/4, 1/4, 1/4, 1/4, 0, 0], 1],
```

- In the uniform strategy (for comparison purposes), there is only one training mode. We sample from regions z0, z1, z2, z3 with a probability of ¼ each.
- Evaluation, however, performed separately on the regions

Final Strategy

```
# in training order
"final_t3": [[0, 0, 0, 1, 0, 0], 1],
```

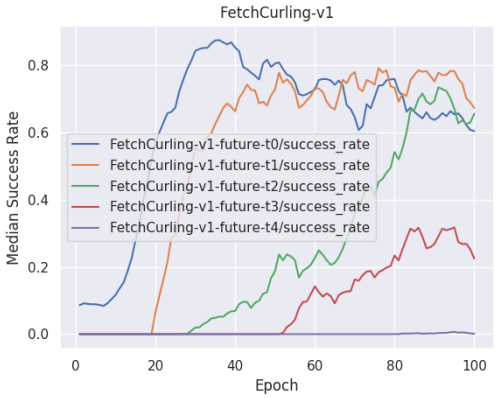
- In the final strategy (for comparison purposes), there is only one training mode. We sample from region z3 with a probability of 1
- This Strategy corresponds to normal HER+EBP

Results

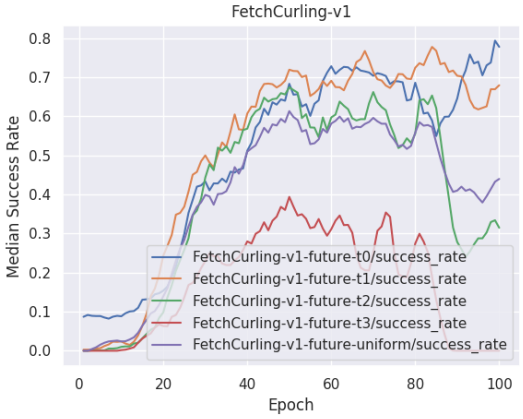
When looking at the figures below, one can see that adaptive Region-based HER does not outperform the uniform strategy (on t_3 which is the most relevant and furthest-away region, uniform performs better). The normal HER+EBP final strategy outperforms the other strategies significantly, which shows that region-based HER is not applicable to the FetchCurling environment.

The main reason for this might be that sliding a puck to a further-away region is not much different from sliding it to a close region. In the final strategy, the neural network only learns how to reach the further-away region and does not have to “memorize” how to reach closer regions. The policy is therefore more optimized towards far-away regions.

- Adaptive (red line corresponds to region of interest)

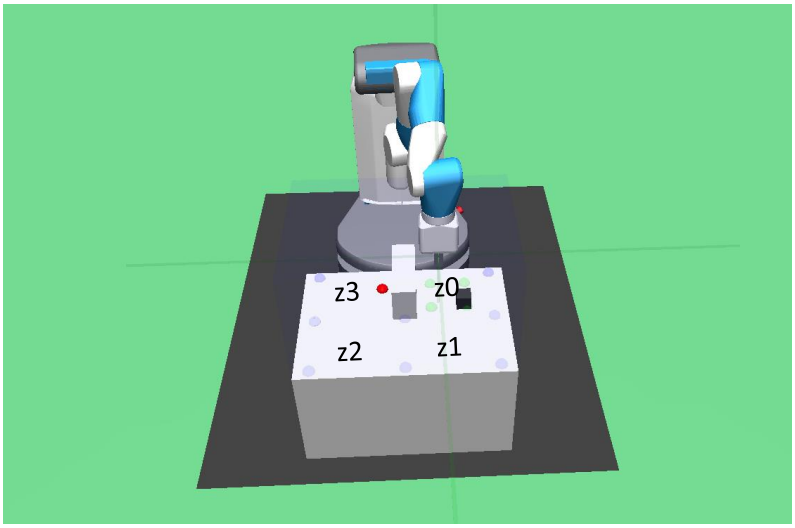


- Uniform (red line corresponds to region of interest)



- Final (orange line corresponds to region of interest)

FetchPushNew (with obstacle)



- In the FetchPushNew environment, four regions (z0, z1, z2, z3) were define as shown in the figure below
- the goal is to push the cube around the obstacle to a goal located somewhere in **region z3**

Adaptive Strategy:

```
# in training order
"t0": [[1, 0, 0, 0], 0.6],
"t1": [[0, 1, 0, 0], 0.6],
"t2": [[0, 0, 1, 0], 0.6],
"t3": [[0, 0, 0, 1], 0.6],
```

- strategy similar to adaptive in FetchCurling: we start in the closest region and advance to the goal region
- min_success_rate of 0.6 for each training mode

Uniform Strategy:

```
# in training order
"uniform": [[1/4, 1/4, 1/4, 1/4], 1],
```

- uniformly sample from all regions (for comparison purposes)

Final Strategy:

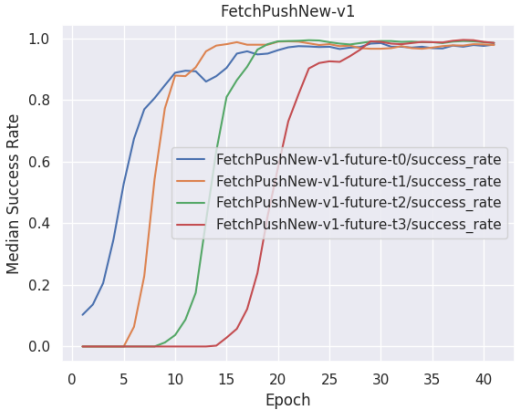
```
# in training order
"final_t3": [[0, 0, 0, 1], 1],
```

- only sample from final region of interest (for comparison purposes)
- corresponds to normal EBP + HER

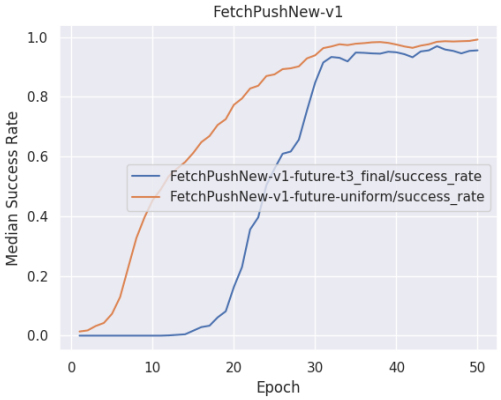
Results

When looking at the figures below, one can see that the adaptive strategy learns faster than the final and uniform strategy whilst achieving similar maximum success rates. (Note: we are only interested in reaching goals in t3, the other graphs are not really relevant to us). This indicates that region-based HER might be applicable to FetchPushNew. More precisely, when setting min_success_rate to lower values (e.g. 0.4) which means we switch faster from one training mode to another one, we might see even faster learning progress (to be evaluated).

- Adaptive (red line corresponds to region of interest)



- Uniform (blue line corresponds to region of interest)



- Final (blue line corresponds to region of interest)