# CS 330 Autumn 2020/2021 Homework 3

Goal Conditioned Reinforcement Learning and
Hindsight Experience Replay
Due Monday October 26th, 11:59 PM PT

SUNet ID:
Name:
Collaborators:

By turning in this assignment, I agree by the Stanford honor code and declare that all of this is my own work.

## Overview

In this assignment we will be looking at goal-conditioned learning and hindsight experience replay (HER). In particular, you will:

1. Adapt an existing model (a Deep Q-Network) to be goal-conditioned.
2. Run goal-conditioned DQN on two environments
3. Implement Hindsight Experience Replay (HER) [1,2] on top of a goal-conditioned DQN for each environment
4. Compare the performance with and without HER

**Submission:** To submit your homework, submit one PDF report to Gradescope containing written answers/plots to the questions below and the link to the colab notebook. The PDF should also include your name and any students you talked to or collaborated with.

This year, we are releasing assignments as Google Colab notebooks. The Colab notebook for this assignment can be found here:
https://colab.research.google.com/drive/1uYqzq4Ix91DD4QdElyrbC_If0TKDf_3t?usp=sharing
As noted in the notebook instructions, you will need to save a copy of the notebook to your Google Drive in order to make edits and complete your submission. **When you submit your PDF responses on Gradescope, please include a clearly-visible link to your Colab notebook so we may examine your code. Any written responses or plots to the questions below must appear in your PDF submission.**

**Code Overview:** The code consists of a few key sections. You should make modifications to areas where required.

For the BitFlip Environment: there exists three subsections

- `BitFlipEnv`: Bit flipping environment for problems 1-3. You should not modify this cell, though it may be helpful to look at it and understand the environment.
- `Buffer`: Buffers for storing experiences. You should not modify this file, though it may be useful to look it.
- `BitFlip Goal Condition RL and Training`: Main loop and helper functions for solving the bit flipping environment. You will add your solutions to problems 1 and 2 to this file and run it for problem 3.

For the Sawyer Environment: there exist a main section (but it reuses Buffer)

- `Sawyer Environment Goal-Conditioned RL`: Main loop and helper functions for solving the Sawyer arm environment. You will add your solutions to problem 4 to this file and run it for problem 5.

For plotting

- `Plotting`: Custom code to help you plot. Please see what variables are needed before running your experiments.

**Dependencies:** You should run everything under the `Setup` section that is provided in the Google Colab notebook. Due to a large number of issues of students installing mujoco on their local machine in the past, we imagine there will be obstacles in doing so and encourage you to use Colab. But if you want to run it locally, we expect code in Python 3.5+ with `Pillow, scipy, numpy, tensorflow, gym, mujoco, multiworld` installed.

## Environments

You will be running on two environments:

### Environment 1: Bit Flipping Environment

In the bit-flipping environment, the state is a binary vector with length `n`. The goal is to reach a known goal vector, which is also a binary vector with length `n`. At each step, we can flip a single value in the vector (changing a 0 to 1 or a 1 to 0). This environment can very easily be solved without reinforcement learning, but we will use a DQN to understand how adding HER can improve performance.

The bit flipping environment is an example of an environment with sparse rewards. At each step, we receive a reward of -1 when the goal and state vector do not match and a reward of 0 when they do. With a larger vector size, we receive fewer non-negative rewards. Adding HER helps us train in an environment with sparse rewards (more details later).

### Environment 2: 2D Sawyer Arm

The Sawyer Arm is a multi-jointed robotic arm for grasping and reaching (`https://robots.ieee.org/robots/sawyer/`). The arm operates in a 2D space, and the goal is to move the robot to a set of coordinates.

To run the Sawyer Arm environment, you will have to install several packages that is provided in the Colab notebook. We **do not recommend installing on a local machine** as students in the past have had major issues, but if you have any trouble with installation on a local machine, feel free to post questions on Piazza.

## Problem 1: Implementing Goal-Conditioned RL on Bit Flipping

The section `BitFlip Goal Condition RL and Training` contains a DQN that runs in a bit-flipping environment. Currently the DQN is not goal-conditioned and does not have HER implemented. The Q-function takes in only the state as the input, and does not consider the goal. To modify the Q-function to be goal-conditioned, concatenate the observation vector with the goal vector and pass the combined vector to the Q-function. You can think of the goal-conditioned implementation as an extended Markov decision process (MDP), where your state space contains both the original state and the goal.

a) Run the colab with the following arguments:

   `success_rate = flip_bits(num_bits=7, num_epochs=150, HER='None')`

   **Plot** the success rate and include it in your homework. This plot illustrates the performance without goal conditioning.

b) Modify the DQN so that it is goal-conditioned in `solve_environment`. You should not make modifications to `Buffer` or `BitFlipEnv`. You **do not** need to submit a plot for this. Hint: the bit flipping environment returns the state and goal vector when `reset()` is called.

## Problem 2: Adding HER to Bit Flipping

With HER, the model is trained on the actual (state, goal, reward) tuples along with (state, goal, reward) tuples where the goal has been relabeled. The goals are relabeled to be what state was *actually reached* and the rewards correspondingly relabeled. In other words, we pretend that the state we reached was always our goal. HER gives us more examples of actions that lead to positive rewards. The reward function for relabeled goals is the same as the environment reward function; for the bit flipping environment, the reward is -1 if the state and goal vector do not match and 0 if they do match.

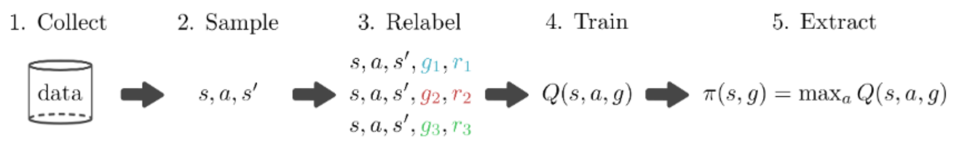

Figure 1: Illustration of HER. The goals and rewards are relabeled so that we have more positive rewards to learn from. In practice, these steps are performed iteratively, returning to step 1 after step 5.

There are three different variations of HER: `final`, `random`, and `future`. In each variation, the goal is relabeled differently:

- `final`: The final state of the episode is used as the goal
- `random`: A random state in the episode is used as the goal
- `future`: A random future state of the episode is used as the goal

More details of these three implementations are in the comments of `sawyer_main.py`.

Implement the three variations of HER in the function `update_replay_buffer()`. You may modify other parts of the code, but this is not necessary. You **do not** need to submit a plot for this.

## Problem 3: Bit Flipping Analysis

Once you have completed the previous problems, run the bit flipping environment both with and without HER:

Each run should generate a separate plot showing the success rate of the DQN. Include these six plots in your writeup.

a) Run code cells with the following commands:

```
success_rate = flip_bits(num_bits=7, num_epochs=250, HER='None')
success_rate = flip_bits(num_bits=7, num_epochs=250, HER='final')
```

b) Run code cells with the following commands:

```
success_rate = flip_bits(num_bits=15, num_epochs=500, HER='None')
success_rate = flip_bits(num_bits=15, num_epochs=500, HER='final')
```

c) Run code cells with the following commands:

```
success_rate = flip_bits(num_bits=25, num_epochs=1000, HER='None')
success_rate = flip_bits(num_bits=25, num_epochs=1000, HER='final')
```

d) Next you will compare the three versions of HER. Plot the performance of the three versions of HER along with running without HER for the following arguments:

```
--num_bits=15 --num_epochs=500
```

Include these **four plots** in your homework. Since two of the plots (HER final and HER none) are identical to part b), feel free to reuse your prior plots.

e) For the above, explain your findings and why you expect the methods to perform in the observed manner for varying numbers of bits and varying relabeling strategies. Your write-up should include around 2-3 sentences for each part.

## Problem 4: Implementing Goal-Conditioning and HER on Sawyer Reach

You will now implement goal-conditioning and HER on a second environment. In the section `Sawyer Environment Goal-Conditioned RL`, implement goal-conditioning and HER for the Sawyer arm environment. Your implementation will be similar to the modifications you made to `BitFlip Goal Condition RL and Training`.

**Hint:** The reward function for the Sawyer environment is the negative Euclidean distance between the goal and state vector.

## Problem 5: Sawyer Analysis

Compare the performance of the Sawyer arm with and without HER. Run the environment with the following arguments, and **include the plots** in your report. If you would like to render the Sawyer Arm and see it move in the environment, add the argument `render=True`.

```
success_rate = run_sawyer(num_epochs=150, HER = "None")
success_rate = run_sawyer(num_epochs=150, HER = "final")
```

Finally, **discuss your findings**, explaining what you see and why the results either differ or are the same.

## References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, Wojciech Zaremba. Hindsight Experience Replay. Neural Information Processing Systems (NeurIPS), 2017. https://arxiv.org/abs/1707.01495

[2] Leslie Kaelbling. Learning to Achieve Goals. International Joint Conferences on Artificial Intelligence (IJCAI), 1993. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.51.3077