Project

Project

You consider the chain environment made up of 5 discrete states and 2 discrete actions, where you get a reward on 0.2 on one end of the chain and 1 at the other end (see illustration below).

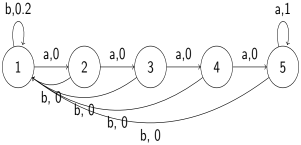

Figure: The chain environment ($\gamma = 0.9$). Initial state is state 1.

In part 1, you work in the tabular context:

- Solve using tabular Q-learning and ϵ-greedy. Provide the optimal Q-values, discuss the learning rate α and ϵ (3 points)
- Increase the size of the chain to 10 states while keeping the rewards at both end of the chain. Discuss the new results, in particular ϵ (2 points).

Project

In part 2, you will solve the chain problem using function approximators (5points) for $\gamma = 0.9$ and 10 states.

- Provide illustrations of the solutions of your optimal Q-values (2 points)
- Discuss the hyper-parameters and the convergence (3 points)

If you go for deep Q learning, Here are additional tips:

- We advise to start from an existing implementation (e.g. DeeR, doc and examples available).
- Normalize the state encoding, e.g. uniformly between $[-1, 1]$.
- Start the code as early as possible.

Deadline : 24th of December (try to aim for one week earlier!)

Example: run_toy_env_simple.py

If you start from
https://github.com/VinF/deer/blob/master/examples/toy_env,
You must modify Toy_env.py and run_toy_env_simple.py.

- You must code the MDP transition (and the reward) in the method act (you don't need to use *rng*)

```
def act(self, action):
    ...
```

- Your state is simply defined as one scalar (without history).

```
def inputDimensions(self):
    return [(1,)]
```

- Your have two actions

```
def nActions(self):
    return 2
```

Example: run_toy_env_simple.py

- You never have terminal states:

```
def inTerminalState(self):
    return False
```

- The function "observe" provides the encoded representation of the state

```
def observe(self):
    return np.array(self._last_ponctual_observation)
```