RLProblem

You will use the `kuimaze2.RLProblem` environment when learning the best strategy for an unknown MDP using reinforcement learning methods (*reinforcement learning*). It is used in the fourth compulsory task `11-RL` .

Public Interface

After creating an instance of the `RLProblem` class (see Usage [/wiki/courses/be5b33kui/semtasks/kuimaze/30_rlproblem#usage]), you can use the following methods:

- `env.get_states()` returns a list of all free states (instances of the `State` class), i.e., without walls.
- `env.get_action_space()` returns a list of all actions that make sense in the given environment. The result is not dependent on the current state (which is different from the `MDPProblem.get_actions(state)` method).
- `env.sample_action(action_probs)` randomly selects one of the possible actions. The `action_probs` parameter contains probabilities for individual actions (it is a dictionary where the key is the action and the value is its probability). If the distribution is not specified, it is chosen uniformly from all actions.
- `env.reset()` resets the environment to the initial state. You need to call this method before each new episode (even before the first one). The method **returns** the initial state (`State`), in which the agent is located at the beginning of an episode.
- `env.step(action)` performs the selected action in the environment. It returns a triplet:
  - `new_state` is the new state (`State`) that the agent got into after performing the action. If the environment is stochastic, the new state may be different from the one that would correspond to the selected action. If you execute `env.step(action)` from a terminal state, the `new_state` is `None` .
  - `reward` is the immediate reward (`float`) for the *state-action-new_state* transition.
  - `episode_finished` is `True` if the episode has already ended. Any further call to the `step()` method in this case will throw an exception. The environment needs to be reset before a new episode using the `reset()` method.
- `env.render()` updates the graphical display of the environment. (If you did not turn on the graphical display when creating the environment, it does nothing.) It allows you to “color” individual states and q-states, allows you to display texts in them, allows you to display the strategy, the current state and the selected action, or the path that the agent has gone through in the environment. You can find an explanation of the individual parameters of this method in

the help (`help(env.render)`). See also the use of `env.render()` in the helper method `RLAgent.render()` in the sample module `example_rl.py` .

Usage

The `RLProblem` environment is created the same way as `MDPProblem` , but the usage is different.

Environment import:

```
>>> from kuimaze2 import Map, RLProblem
```

Creating a map to initialize the environment:

```
>>> MAP = "SG"
>>> map = Map.from_string(MAP)
```

Creating a deterministic environment with graphical display:

```
>>> env1 = RLProblem(map, graphics=True)
```

Creating a non-deterministic environment (specifying the probabilities of where the agent will actually move):

```
>>> env2 = RLProblem(map, action_probs=dict(forward=0.8, left=0.1, right=0.1, backward=0.
```

List of all valid states in the environment:

```
>>> env2.get_states()
[State(r=0, c=0), State(r=0, c=1)]
```

List of all actions that can be performed in some state of the environment:

```
>>> env2.get_action_space()
[<Action.UP: 0>, <Action.RIGHT: 1>, <Action.DOWN: 2>, <Action.LEFT: 3>]
```

A randomly selected action may also be useful:

```
>>> env2.sample_action()  # The result can be any of the possible actions.
```

```
<Action.UP: 0>
```

The step method attempts to perform the selected action in the environment:

```
>>> env2.step(env2.sample_action())


Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "...\kuimaze2\rl.py", line 60, in step
    raise NeedsResetError(
kuimaze2.exceptions.NeedsResetError: RLProblem: Episode terminated. You must call reset()
```

As you can see, before the first use of the `step()` method, you need to `reset` the environment.

Calling the `reset()` method will return the initial state of the agent for the given episode:

```
>>> state = env2.reset()
>>> state
State(r=0, c=0)
```

Now we can call the `step()` method:

```
>>> action = env2.sample_action()
>>> action
<Action.DOWN: 2>
>>> new_state, reward, episode_finished = env2.step(action)
>>> new_state
State(r=0, c=0)
>>> reward
-0.04
>>> episode_finished
False
```

We tried to perform the `Action.DOWN` action, but we probably hit a wall, because the new state `new_state` is identical to the original one. We received an immediate reward of `-0.04` for performing the action and we see that the episode has not ended, we can continue.

So let's try to make random steps until the episode ends:

```
>>> while not episode_finished:
...     action = env2.sample_action()
...     new_state, reward, episode_finished = env2.step(action)
...     print(f"{state=} {action=} {reward=} {new_state=} {episode_finished=}")
...     state = new_state
...
state=State(r=0, c=0) action=<Action.DOWN: 2> reward=-0.04 new_state=State(r=0, c=0) epis
state=State(r=0, c=0) action=<Action.RIGHT: 1> reward=-0.04 new_state=State(r=0, c=1) epi
state=State(r=0, c=1) action=<Action.UP: 0> reward=1.0 new_state=None episode_finished=Tr
```

Another call to the `step()` method would again throw an exception. The episode ended, we want to start a new one, so we need to call `reset()` again.

courses/be5b33kui/semtasks/kuimaze/30_rlproblem.txt · Last modified: 2024/04/16 14:11 by xposik