3. Markov Decision Processes

Your task is to implement the *value iteration* and *policy iteration* methods to find the optimal strategy (policy) for the given MDP.

Specification

In the module `mdp_agent.py` , implement two classes:

- `ValueIterationAgent` which will find the optimal strategy using the *value iteration* method, and
- `PolicyIterationAgent` which will find the optimal strategy using the *policy iteration* method.

The interface of both classes is identical, both must implement the following methods:

method	input parameters	output parameters	explanation
`__init__`	`env: MDPProblem` , `gamma: float` , `epsilon: float`	*none*	Agent initialization.
`find_policy`	*none*	`Policy`	Returns the optimal strategy, i.e., a dictionary of pairs (state, action).

- The class will be initialized with the following **parameters**:
  - `env` is the environment, i.e., an object of type `kuimaze2.MDPProblem`
  - `gamma` is the so-called "discount factor" from the range `(0,1)`
  - `epsilon` is the maximum allowed error for the values of individual states (used in value iteration)
- The **output** of the `find_policy()` method must be a **policy** represented as a **dictionary**, where the key is always a state (instance of the class `kuimaze2.State`) and the value is the optimal action for that state (instance of the class `kuimaze2.Action`). The strategy must contain an action for all free states, including terminal ones. The specific action chosen for terminal states does not matter.
- **Timeout** for individual runs of value/policy iteration for a given problem instance is set to 30s. (But you should only need a fraction of this time.)

- The algorithms implemented in the classes `ValueIterationAgent` and `PolicyIterationAgent` must correspond to the assignment. For example, it is not allowed to simply call `ValueIteration.find_policy()` in `PolicyIterationAgent.find_policy()` or to implement the *value iteration* algorithm in it (or vice versa). In such a case, the entire task will be evaluated with 0 points!
- In the implementation of the algorithms, you can only use public methods of the ''MDPProblem'' class [/wiki/courses/be5b33kui/semtasks/kuimaze/20_mdpproblem]. If you feel that you need to use methods of other classes than `MDPProblem` , or that you need to use non-public variables and methods (whose name starts with `_`), discuss it with your instructor.

How to

1. We recommend creating a new working directory for the task. Set up [/wiki/courses/be5b33kui/semtasks/kuimaze/00_install] an updated version of the `kuimaze2` package in it.
2. Familiarize yourself with the MDPProblem [/wiki/courses/be5b33kui/semtasks/kuimaze/20_mdpproblem] environment.
3. In the `kuimaze2` package, you will also find the script `example_mdp.py` , which also shows how to work with the environment. It can be used as a starting code for the implementation of both classes.
4. It is quite possible that both classes will have some common parts. In such a case, we recommend (as indicated in `example_mdp.py`) to extract shared parts into a common ancestor of both classes:

```
class MDPAgent:
    # Parts common to both methods/agents
    ...

class ValueIterationAgent(MDPAgent):
    # Parts specific for value iteration
    ...

class PolicyIterationAgent(MDPAgent):
    # Parts specific for policy iteration
    ...
```

Submission

- The deadline for submitting the task can be found in BRUTE [https://cw.felk.cvut.cz/brute], task `08-MDPs` .
- Submit the module `mdp_agent.py` , or a ZIP archive with the module `mdp_agent.py` and other modules you created that your agent needs/imports. **These files must be in the root of the**

archive, the archive must not contain any directories! Do not include/submit any modules that you received from us!

Evaluation

Learn about evaluation and scoring [/wiki/courses/be5b33kui/semtasks/03_mdp/scoring] of the task.

courses/be5b33kui/semtasks/03_mdp/start.txt · Last modified: 2024/03/26 20:34 by xposik