Documentation

Le module `base` contient un ensemble de classes permettant de définir des MDP.

Classe `Environment`

Méthodes :

- `get_name()` : retourne le nom de l'environnement
- `get_nb_states()` : retourne le nombre d'états
- `get_nb_actions()` : retourne le nombre d'actions
- `reset()` :
  - réinitialise l'environnement
  - retourne une observation de l'état courant
- `step(action)` :
  - exécute l'action
  - retourne le tuple (observation, reward, done, info)
- `render()` : affiche l'environnement dans son état courant

La classe `MDP`

Extension de `Environment`

Méthodes d'observation :

- `is_final(state)` : teste si l'état est terminal
- `is_valid(state)` : teste si l'état est valide (s'il peut servir d'état courant)
- `p(state=None, action=None, next_state=None)` :
  - retourne les probabilités de transitions filtrées selon les critères d'entrées
- `r(state=None, action=None, next_state=None)` :
  - retourne les renforcements (rewards) filtrés selon les critères d'entrées
- `render_values(values, precision)` :

- affiche l'environnement en donnant la valeur de chaque état
- `precision` permet de paramétrer la précision de l'affichage des valeurs

- `render_policy(policy)` :
  - affiche l'environnement en donnant la meilleure action de la politique dans chaque état

Méthodes d'exécution d'un épisode :

- `observe_episode(policy, limit)` : exécute un épisode en affichant l'environnement à chaque étape
- `perform_episode(policy, limit)` : exécute un épisode sans faire d'affichage
  - retourne un tuple `(n, r)` avec `n` le nombre d'étape exécutée et `r` le total des renforcements reçus

Méthodes d'initialisation :

- `init_all_proba_to_hole(states=None)` :
  - initialise les probabilités de transitions depuis les états spécifiés (tous si None) de manière à ce qu'ils soient des puits (toute action est sans effet)

- `init_all_proba_to_fixed_value(value, states=None)` :
  - initialise les probabilités de transitions depuis les états spécifiés (tous si None) de manière à la valeur `value`

- `set_transition(state, action, next_state, proba, reward)` :
  - affecte une proba et un renforcement à une transition `(state, action, next_state)`

- `set_transition_proba(state, action, next_state, proba)` :
  - affecte une proba à une transition `(state, action, next_state)`

- `set_transition_reward(state, action, next_state, reward)` :
  - affecte un renforcement à une transition `(state, action, next_state)`

La classe `TabularMDP`

Extension de MDP représentant les probabilités et les renforcement en utilisant des ndarray de numpy

Les environnements de type "grille"

Etats

- Les n états sont numérotés de 0 à $n-1$ en partant de la case en haut à gauche puis en poursuivant de gauche à droite et de haut en bas
- L'état initial par défaut est l'état 0
  - Le paramètre `initial_state` permet de spécifier un autre état initial
  - Si `initial_state=None`, l'état initial est tiré aléatoirement parmi tous les états valides

Actions

- 4 actions de déplacement : Ouest(0), Sud(1), Est(2), Nord(3)

Environnements

- La classe `Maze` implémente l'environnement de l'exercice 1
- Les classes `FrozenLake44` et `FrozenLake88` sont deux environnements de type "lac gelé"
- La classe `FoorRooms_key`implémente l'environnement de l'exercice 4