Reinforcement Learning

APPLICATIONS L'outils GYM. d'OpenIA

Open IA organisation de recherche sur l'IA.

OPenIA fournit une boite à outils Gym pour former un agent d'apprentissage par renforcement.

Si on veut former un agent à conduire une voiture : il faut un environnement pour former l'agent. Cet environnement doit être simulé (puisqu'on procédé par essais erreurs, trop dangereux de le faire sur un environnement réel).

Il existe différentes boites à outils qui fournissent un environnement simulé pour la formation d'un agent RL.

L'une de ces boites à outils est Gym.

Reinforcement Learning

Gym fournit un ensemble d'environnements pour former des agents RL (taches de contrôles classiques aux environnements de jeu Atari).

On pourra donc former un agent RL à apprendre dans ces environnements simulés en utilisant des algorithmes RL.

Reinforcement Learning

Préliminaire ; Installation de Gym (à faire en TP).

Creer un environnement Gym

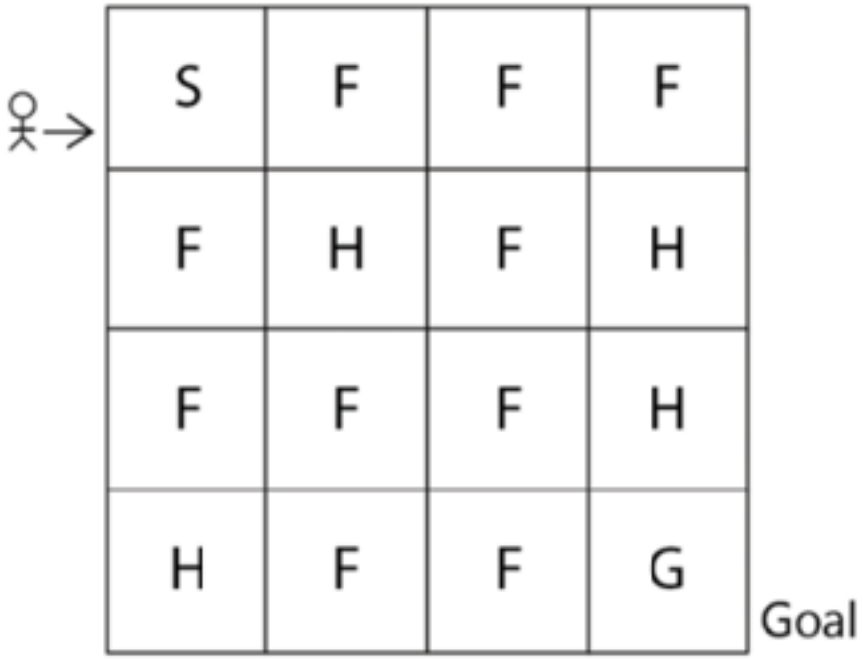
Exemple l'environnement Frozen Lake : but atteindre l'état but G à partir de l'état initial S

S état initial

G état but

F état Frozen

H état GoalPréliminaire ; Installation de Gym (à faire en TP).



Creer un environnement Gym

Exemple l'environnement Frozen Lake

Reinforcement Learning

Chaque case de la grille de l'environnement est appelée un. État.

Donc 16 stats de S à G et 4 actions possibles (up, down, left, right).

Objectif : atteindre G à partir de A sans passer par les etats H.

Rewards : +1 pour l'état G, 0 pour les autres états (par exemple mais pas le plus performant mais le but est de découvrir gym).

Donc pour avoir accès à gym sous python: import gym.

Il faudra ensuite créer l'environnement par :

env = gym.make(''FrozenLake-v0'')

on peut ensuite voir à quoi ressemble l'environnement par la fonction render

env.render()

```
>>> env= gym.make("FrozenLake-v0")
>>> env.render()

SFFF
FHFH
FFFH
HFFG
>>>
```

Reinforcement Learning

Nous avons vu qu'un environnement d'apprentissage automatique était modélisé par un MDP (Markov decision Process) impliquant:

- l'ensemble des états
- l'ensemble des actions
- . les probabilités de transition notées P(s' I (s,a) - la probabilité de passer d'un état s à un état s' en executant l'action a.
- les rewards notées R(s,a,s') : la récompense renvoyée à l'agent en passant de l'état s à l'etat s' en executant l'action a.

Avec Gym

Les états ;

```
>>> print (env.observation_space)
Discrete(16)
>>>
```

Discrete(16) veut dire que nous avons 16 états discrets dans l'espace d'états à partir d l'état S à l'état G.

Reinforcement Learning

Dans Gym, les états sont codées sous forme de nombre :

0 S	1 F	2 F	3 F
4 F	5 H	6 F	7 H
8 F	9 F	10 F	11 H
12 H	13 F	14 F	15 G

Les ACTIONS

L'ensemble des actions possible est donné par : env.action_space

```
>>> print(env.action_space)
Discrete(4)
>>>
```

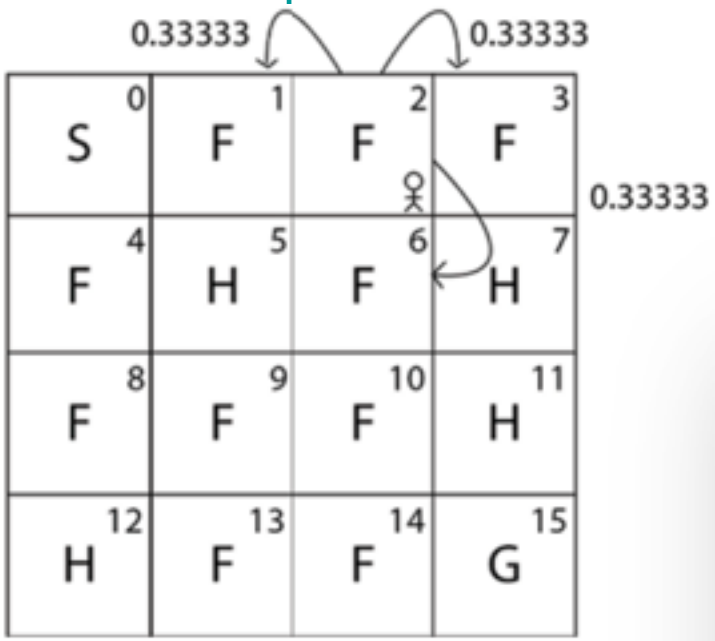
Reinforcement Learning

Dans Gym, les actions sont codées sous forme de nombre :

Number	Action
0	Left
1	Down
2	Right
3	Up

PROBABILITE de TRANSITION et REWARD FUNCTION

L'environnement FROZEN LAKE est stochastique



Reinforcement Learning

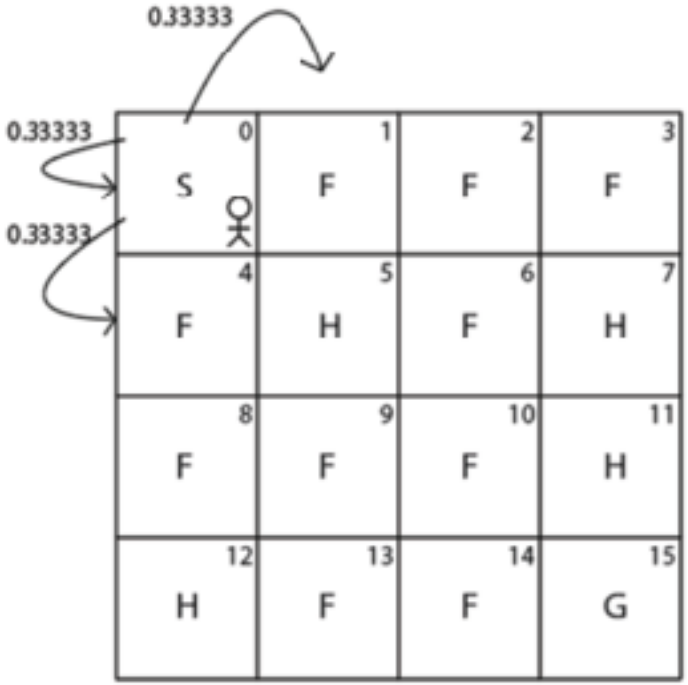
Pour obtenir les probabilités de transition et les rewards, il suffit de taper :

env.P[state][action].

Donc pour obtenir la probabilité de passer de l'état S aux autres états en executant l'action right : env.P[S][right].

Bien sur dans Gym le codage se fait à l'aide des nombres donc: env.P[0][2]

```
>>> print (env.env)
<FrozenLakeEnv<FrozenLake-v0>>
>>> print (env.env.P[0][2])
[(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0, 0.0, False)]
>>>
```



Reinforcement Learning

Attention env.env (????).

```
>>> print (env.env)
<FrozenLakeEnv<FrozenLake-v0>>
>>> print (env.env.P[0][2])
[(0.3333333333333333, 4, 0.0, False), (0.3333333333333333, 1, 0.0, False), (0.3333333333333333, 0, 0.0, False)]
>>>
```

En tapant env.P[state][action], on a comme résultat quelque chose de ce format: [(probabilité de transition, état suivant, reward, est-ce l'état but?)].
Donc résultat

Transition Probability	Next State	Reward	Is terminal state
0.33333	4(F)	0.0	False
0.33333	1(F)	0.0	False
0.33333	0(S)	0.0	False

Reinforcement Learning

Générer un épisode avec Gym.

Rappel : un épisode est l'interaction agent-environnement depuis l'état initial jusqu'à l'état terminal : l'agent interagit avec l'environnement en effectuant une action dans chaque état.

Nous allons générer un episode en prenant des actions aléatoires dans chaque état.

Il faut tout d'abord définir le nombre de pas : (timesteps)

num_timesteps = 20

Puis boucler sur le nombre de pas en

- générant une action aléatoire à executer avec :

    random_action = env.action_space.sample()

-executant l'action sélectionnée par :

    new_state,reward,done,info = env.step(random_action)

-sortir de la boucle quand l'état terminal est obtenu (tester done)

Reinforcement Learning

Exemple de résultat à obtenir en utiliser env.render et affichant le time Step.

Time Step 0 :

SFFF
FHFH
FFFH
HFFG

Time Step 1 :
(right)

SFFF
FHFH
FFFH
HFFG

Time Step 2 :
(right)

SFFF
FHFH
FFFH
HFFG

Time Step 3 :
(right)

SFFF
FHFH
FFFH
HFFG

Time Step 4 :
(down)

SFFF
FHFH
FFFH

Reinforcement Learning

De la meme falcon nous pouvons faire de meme en générant une série d'episodes au lieu d'un seul episode.

Utiliser la variable num_episodes = 10 par exemple.

Dans l'exemple étudié, on a juste pris des actions aléatoires dans chaque état sur tous les épisodes.

Evidemment nous avons besoin d'un algorithme d'apprentissage comme vu précédemment pour trouver la politique optimale (c-a-d la politique qui dit quelle action effectuée dans chaque état.