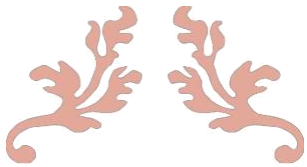

INTELLIGENCE ARTIFICIELLE DEVELOPPEMENTALE

TP - Agents

22 NOVEMBRE 2020

PIERRE-HENRI CHUPIN

SOMMAIRE

INTRODUCTION

Le but de ces TPs est d'apprendre à implémenter un agent selon les principes de ***l'Intelligence Artificielle Développementale***.

AGENT 1 – L'AGENT QUI N'AIMAIT PAS S'ENNUYER

OBJECTIF

Cet agent à pour objectif d'apprendre à prédire le résultat de ses actions en fonction de son environnement pour pouvoir choisir une action différente quand ses prédictions sont correctes depuis trop longtemps.

IMPLEMENTATION

Variables :

Pour l'implémentation de cet agent, nous avons eu besoin de rajouter trois nouvelles variables :

- '*compteurEnnui*' : il agit comme un compteur. C'est-à-dire qu'on l'incrémente tant qu'on n'est pas supérieur ou égale à 4, une fois qu'on est à 4, notre agent s'ennuie.
- *'memoire'* : c'est un dictionnaire qui permet d'avoir le lien entre une action et son outcome. Si une à un moment donné une action n'a pas l'outcome prévu par la mémoire, il est mis à jour. C'est qu'il faut donc modifier l'outcome.

Action :

Tout d'abord, la première chose à savoir c'est : est ce que notre agent s'ennuie ? Pour cela, nous utilisons notre compteurEnnui, s'il arrive à 4 : on inverse l'action.
Ensuite, on initialise la mémoire de l'action avec l'outcome.

Puis nous regardons l'anticipation de l'outcome, s'il n'est pas égal à l'outcome alors on donne à la mémoire la valeur de cet outcome à l'indice de l'action fait à ce moment. Dans le cas où l'anticipation de l'outcome est égale à l'outcome alors on incrémente de un notre 'ennuie'.
Finalement, on retourne l'action obtenue.

Anticipation :

On donne à l'anticipation la valeur de la mémoire pour l'action faite puis on la retourne.

Satisfaction :

Nous n'avons rien modifié dans cette fonction.

TRACE ENVIRONNEMENT 1

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/world.py
L'agent 1 fait tous le temps la même action mais au bout d'un cycle de 4 bonne anticipation il s'ennuie et change alors d'action à faire

 Action:  , Anticipation:  , Outcome:  , (Bonne Anticipation , Valeur Hedoniste, Ennuie)

 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
 Action: 1, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
```

TRACE ENVIRONNEMENT 2

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/world.py
L'agent 1 fait tous le temps la même action mais au bout d'un cycle de 4 bonne anticipation il s'ennuie et change alors d'action à faire

 Action:  , Anticipation:  , Outcome:  , (Bonne Anticipation , Valeur Hedoniste, Ennuie)

 Action: 0, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 1, Anticipation: 1, Outcome: 0, Satisfaction: (False, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
```

COMPORTEMENT OBTENU

On peut voir sur la première trace que l'agent commence par l'action 0, il l'anticipe correctement. Au bout de quatre anticipations correctes, on voit que l'agent s'ennuie et qu'il est donc forcé de changer d'action. Il passe donc à l'action 1 mais il n'anticipe pas la bonne action car pour lui cela semblait logique de continuer dans une logique d'action zéro. Il n'est donc pas satisfait d'où le False. On peut aussi dire qu'il n'était encore jamais tombé dans cette situation, il a juste continué dans cette lancée. Une fois qu'il retombe dans le cas de l'ennui on peut voir qu'il a appris de la dernière situation où il s'était trompé, et peut donc anticiper la bonne action.

Pour ce qui concerne la deuxième trace, on voit qu'il anticipe mal dès le début, mais ce n'est aucunement lié à l'ennui mais juste à l'environnement. C'est ce qui explique que plus tard, une fois que l'agent s'ennuie il se trompe dans son anticipation car il n'a pas encore fait face à cette situation. Mais on voit bien que quand il rencontre de nouveau cette situation il arrive à bien anticiper.

AGENT 2 – L'AGENT QUI PREFERAIT LES INTERACTIONS POSITIVES

OBJECTIF

Cet agent doit choisir préférentiellement les interactions qui ont une valeur hédoniste positive, sauf s'il ennuie, auquel cas il préfère faire une action différente même si elle conduit à une interaction de valeur négative.

IMPLEMENTATION

Variables :

Pour l'implémentation de cet agent, nous avons eu besoin de rajouter plusieurs nouvelles variables :

- *'maxEnnui'* : un entier qui correspond à la limite de l'ennui, il est fixé à 4.
- *'mémoire'* : ici on retrouve la même structure de données que pour l'agent 1.
- *'bonneAnticipationCompteur'* : comme son nom l'indique cet variable est un compteur qui nous permet de compter le nombre de bonne anticipation effectué par l'agent.
- *'actionPrefereCompteur'* : cette variable est un compteur qui nous permet de compter le nombre d'actions préférées que l'on a fait.
- *'actionPrefere'* : cette variable nous permet de retenir l'action préférée donc l'action qui rapporte le plus.
- *'outcomePrefere'* : cette variable nous permet de retenir l'outcome préféré.
- *'derniereAction'* : cette variable nous permet de garder en mémoire la dernière action effectué. Cela nous est utile pour la mise à jour de la mémoire.

Action :

Tout d'abord, nous devons regarder la table hédoniste et savoir si l'outcome actuel nous donne une meilleure valeur. Si c'est le cas, on retient l'action comme notre action préférée, l'outcome, comme notre outcome préféré et on réinitialise le nombre d'action préféré effectuer jusqu'à présent.
On prend comme action, notre action préférée.
Si l'agent s'ennuie, on change l'action et on remet les compteurs à zéro.
On met à jour la mémoire en fonction de l'outcome, comme on le faisait pour l'agent 1.
On incrémente nos compteurs avant de renvoyer l'action choisit.

Anticipation :

On donne à l'anticipation la valeur de la mémoire pour l'action faite puis on la retourne.

Satisfaction :

Nous n'avons rien modifié dans cette fonction.

TRACE ENVIRONNEMENT 1

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/wolrd2.py
L'agent 2 cherche à faire la meilleur action mais il peut s'ennuyer comme l'agent 1 et alors il change d'action juste un fois pour tester de nouveau horizon

 Action:  , Anticipation:  , Outcome:  , (Bonne Anticipation , Valeur Hedoniste, Ennuie)

 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
 Action: 1, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
```

TRACE ENVIRONNEMENT 2

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/wolrd2.py
L'agent 2 cherche à faire la meilleur action mais il peut s'ennuier comme l'agent 1 et alors il change d'action juste un fois pour tester de nouveau horizon

 Action:  , Anticipation:  , Outcome:  , (Bonne Anticipation , Valeur Hedoniste, Ennuie)

 Action: 0, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 1, Anticipation: 1, Outcome: 0, Satisfaction: (False, -1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
```

COMPORTEMENT OBTENU

Sur la première trace, on peut voir que dans un premier temps notre agent effectue la même action tout en anticipant comme il faut ses actions. Au moment où il s'ennuie, il change d'action comme pour l'agent 1, avec la même erreur sur l'anticipation étant donné que c'est la première fois qu'il rencontre ce cas. Il a changé d'action, continue dans la même lancé et répète cette action jusqu'à l'ennui. Au prochain ennui, il change d'action mais seulement une fois, cela s'explique par le fait que l'action 0 a une plus petite valeur que quand il fait l'action 1. Du coup, il rechange d'action le tour suivant, et l'anticipe comme il faut car l'agent savait que l'action 1 est meilleur que l'action 0 étant donné qu'il avait déjà expérimenté les deux actions.

C'est à ce niveau là qu'on repère une différence entre les deux traces. Le début est identique que pour le premier agent, mais arrivé au premier ennui, il enchaîne deux fois une mauvaise

anticipation. En effet, l'agent est forcé de changer d'action car il s'ennuie, comme c'est la première fois qu'il tombe dans ce cas, il se trompe sur son anticipation. Il pense se réadapter le tour d'après mais comme l'agent 2 favorise les actions positives, il rechange d'action, et se loupe encore une fois dans son anticipation. Mais à partir de ce stade, il a pu voir tous les différents cas possibles et peut à partir de maintenant anticiper sans erreur.

AGENT 3 – L'AGENT QUI EST CAPABLE D'APPRENDRE A ETRE CONTENT

OBJECTIF

Cet agent doit être capable d'apprendre à obtenir des interactions positives quand il est placé dans n'importe lequel des 3 environnements. Pour cela il doit choisir sa prochaine action en fonction du contexte dans lequel il se situe.

IMPLEMENTATION

Variables :

A noter que pour cet agent, on est reparti de l'agent 2, c'est-à-dire qu'on ne sait pas à l'avance qu'alterner les actions est ce qu'il faut privilégier.

Pour l'implémentation de cet agent, voici les différentes variables que nous avons utilisées :

- *'maxEnnui'* : un entier qui correspond à la limite de l'ennui, il est fixé à 4.
- *'mémoire'* : ici on retrouve la même structure de données que pour l'agent 1 et l'agent 2.
- *'boolEnnui'* : un booléen qui est à True si on s'ennuie, et à False sinon.
- *'valActionCourante'* : c'est un entier qui permet de se déplacer dans le tableau du meilleur cycle.
- *'dernieresActions'* : c'est un tableau qui garde en mémoire les dernières actions effectuées.
- *'meilleurCycle'* : c'est un tableau qui nous permet de garder en mémoire le meilleur cycle (action, outcome) à répéter.

Action :

Tout d'abord, nous devons nous occuper de mettre à jour le meilleur cycle. Si le tableau de meilleur cycle est vide on lui donne la paire d'action et d'outcome actuel. Ensuite on

récupère la valeur hédoniste de cette action et de l'outcome. Plusieurs actions sont possibles :

- Si la valeur hédoniste est plus grande que la valeur du meilleur cycle alors on change le meilleur cycle par l'action et l'outcome actuel.
- Si on a la même valeur et que le couple (action, outcome) n'est pas dans le meilleur cycle alors on l'ajoute au meilleur cycle.

On met à jour la mémoire comme pour les autres agents.

On décide de l'action à faire en fonction du meilleur cycle que l'on a obtenu jusqu'à présent.

On vérifie que notre agent ne s'ennuie pas, s'il s'ennuie on change l'action.

Avant de retourner l'action, il faut mettre à jour tous les compteurs ainsi que rajouter l'action qu'on va faire dans notre tableau last_actions pour pouvoir le garder en mémoire. Et on finit par renvoyer l'action.

Anticipation :

On donne à l'anticipation la valeur de la mémoire pour l'action faite puis on la retourne.

Satisfaction :

Nous n'avons rien modifié dans cette fonction.

TRACE ENVIRONNEMENT 1

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/world3.py
L'agent 3 fonctionne avec une recherche du meilleur cycle car si il faut pour avoir de bonne valeur alterner entre 2 action ici 0 et 1 alors il doit comprendre
 qu'il y a un cycle à faire l'ennuie à une limite de 4 ici

 Action:  , Anticipation:  , Outcome:  , (Bonne Anticipation , Valeur Hedoniste, Ennuie)

 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
 Action: 1, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
```

TRACE ENVIRONNEMENT 2

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/world3.py
L'agent 3 fonctionne avec une recherche du meilleur cycle car si il faut pour avoir de bonne valeur alterner entre 2 action ici 0 et 1 alors il doit comprendre
 qu'il y a un cycle à faire l'ennuie à une limite de 4 ici

 Action:  , Anticipation:  , Outcome:  , (Bonne Anticipation , Valeur Hedoniste, Ennuie)

 Action: 0, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 1, Anticipation: 1, Outcome: 0, Satisfaction: (False, -1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
```

TRACE ENVIRONNEMENT 3

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/world3.py
L'agent 3 fonctionne avec une recherche du meilleur cycle car si il faut pour avoir de bonne valeur alterner entre 2 action ici 0 et 1 alors il doit comprendre
 qu'il y a un cycle à faire l'ennuie à une limite de 4 ici

 Action:  , Anticipation:  , Outcome:  , (Bonne Anticipation , Valeur Hedoniste, Ennuie)

 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
 Action: 1, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 1, Anticipation: 1, Outcome: 0, Satisfaction: (False, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
 Action: 0, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 1, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
```

COMPORTEMENT OBTENU

Pour cet agent nous expliquerons que la troisième trace avec l'environnement 3 car les deux premières traces sont identiques à l'agent 2. Du coup, ici ce qui nous intéresse c'est la valeur hédoniste et pour avoir une valeur hédoniste de 1, il faut alterner l'action 1 et 0. Notre agent 3 fait donc comme pour l'agent au départ. Il effectue 4 fois l'action puis s'ennuie. Jusqu'à présent il avait pour valeur hédoniste -1, comme il s'ennuie il change d'action et on voit qu'il obtient 1 donc il les garde en mémoire. Ensuite, Il se dit qu'il va continuer d'obtenir 1 avec l'action 1, mais au bout d'un cycle il s'ennuie de nouveau. Il change donc de nouveau d'action. Cette action lui donne comme valeur hédoniste 1, il le rajoute dans son meilleur

cycle. A partir de ce point, on repère que l'agent a compris quel est le cycle qui lui permet d'obtenir toujours 1 en valeur hédoniste.

AGENT 4 – L'AGENT QUI S'ADAPTE AU CHANGEMENT D'ENVIRONNEMENT

OBJECTIF

Comparé à l'agent 3, l'agent 4 récupère la fréquence des cycles pour pouvoir supprimer les cycles, qui ne sont plus utilisés par l'agent, de nos meilleurs cycles. Cela permet donc de créer un agent qui est capable de s'adapter au changement d'environnement.

IMPLEMENTATION

Variables :

Tout comme pour l'agent 3 dont nous étions partis de l'agent 2, ici nous sommes repartis de l'agent 3.

Pour l'implémentation de cet agent, voici les différentes variables que nous avons utilisés :

- *'maxEnnui'* : un entier qui correspond à la limite de l'ennui, il est fixé à 4.
- *'mémoire'* : ici on retrouve la même structure de données que pour l'agent 1 et l'agent 2.
- *'meilleurCycle'* : c'est un tableau qui nous permet de garder en mémoire le meilleur cycle (action, outcome) à répéter.
- *'dernieresActions'* : c'est un tableau qui garde en mémoire les dernières actions effectuées.
- *'valActionCourante'* : c'est un entier qui permet de se déplacer dans le tableau du meilleur cycle.
- *'frequenceActionsOutcome'* : c'est un dictionnaire qui garde en mémoire le nombre de fois qu'on effectue un couple d'action et outcome. Il nous sert principalement pour pouvoir enlever les actions et outcome qui ne sont plus effectués de son cycle.

Action :

Tout d'abord, nous devons nous occuper de mettre à jour le meilleur cycle. Dans cette mise à jour nous devons vérifier si l'action et l'outcome actuel sont dans notre dictionnaire de fréquence, si oui on incrémente ce tuple, sinon on le rajoute. Ensuite, si le tableau de

meilleur cycle est vide on lui donne la paire d'action et d'outcome actuel. Puis on récupère la valeur hédoniste de cette action et de l'outcome. Plusieurs actions sont possibles :

- Si la valeur hédoniste est plus grande que la valeur du meilleur cycle alors on change le meilleur cycle par l'action et l'outcome actuel.
- Si on a la même valeur et que le couple (action, outcome) n'est pas dans le meilleur cycle alors on l'ajoute au meilleur cycle.

On met à jour la mémoire comme pour les autres agents.

On décide de l'action à faire en fonction du meilleur cycle que l'on a obtenu jusqu'à présent.

On vérifie que notre agent ne s'ennuie pas, s'il s'ennuie on change l'action.

Quand on change d'action, ce n'est plus comme pour les autres agents, on va aussi mettre à jour les fréquences de nos actions/outcome. Pour cela, on parcourt notre dictionnaire de fréquence, on garde la valeur actuelle, on remplace celle du dictionnaire à 0. Si la valeur qu'on a gardée est plus petite que 1 et que ce tuple d'action outcome est dans notre meilleur cycle, on le supprime de nos meilleurs cycles et de notre dictionnaire de fréquence. Avant de retourner l'action, il faut mettre à jour tous les compteurs ainsi que rajouter l'action qu'on va faire dans notre tableau dernieresActions pour pouvoir la garder en mémoire. Et on finit par renvoyer l'action.

Anticipation :

On donne à l'anticipation la valeur de la mémoire pour l'action faite puis on la retourne.

Satisfaction :

Nous n'avons rien modifié dans cette fonction.

TRACE

```
chupin@chupin-LIFEBOOK-S762:~/Bureau/TestROS-master$ /usr/local/bin/python3.8 /home/chupin/Bureau/TestROS-master/world4.py
i:0  Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
i:1  Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
i:2  Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
i:3  Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
i:4  Action: 1, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
i:5  Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:6  Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:7  Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:8  Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
i:9  Action: 0, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
changement d'Environment
i:10 Action: 1, Anticipation: 1, Outcome: 0, Satisfaction: (False, -1, False)
i:11 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
i:12 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
i:13 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
i:14 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, True)
i:15 Action: 0, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
i:16 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:17 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:18 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:19 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
i:20 Action: 1, Anticipation: 0, Outcome: 0, Satisfaction: (True, -1, False)
i:21 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:22 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:23 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:24 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
changement d'Environment
i:25 Action: 1, Anticipation: 0, Outcome: 1, Satisfaction: (False, 1, False)
i:26 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:27 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:28 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:29 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:30 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:31 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:32 Action: 0, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, False)
i:33 Action: 1, Anticipation: 1, Outcome: 1, Satisfaction: (True, 1, True)
i:34 Action: 1, Anticipation: 1, Outcome: 0, Satisfaction: (False, -1, False)
```

COMPORTEMENT OBTENU

Pour l'agent 4, on commence avec l'environnement 1 et au bout de 10 actions on passe sur l'environnement 2 qui lui-même passe à l'environnement 3 au bout de 15 actions. Ainsi on peut vérifier que notre agent s'adapte au changement d'environnement. La principale différence avec l'agent 3 c'est qu'ici notre agent supprime de sa mémoire les cycles qui ne sont plus effectués depuis un certain temps. On voit qu'au premier changement d'environnement, l'agent voit que le cycle action 1 outcome 1 n'est plus effectué du coup il l'enlève de sa mémoire. Il tente donc de retrouver un cycle lui donnant une valeur hédoniste de 1, une fois retrouvé il le place dans sa mémoire des meilleurs cycles. Il ne possède plus l'ancien meilleur cycle de l'ancien environnement mais seulement le nouveau meilleur cycle.

Quand il change de nouveau d'environnement pour passer du 2^{e} au 3^{e} , on voit qu'il arrive de nouveau à s'habituer.

CONCLUSION

Tout au long de ce TP, nous avons suivi le même fil conducteur. En effet, on est parti d'un agent simple qui n'avait pas beaucoup de fonctionnalités. Puis petit à petit nous l'avons peaufiné. Au début, on ne voulait pas qu'il s'ennuie, puis on voulait favoriser les actions positives, puis on a modifié son environnement pour voir s'il était capable d'effectuer un cycle d'action pour obtenir une action positive, et pour finir on voulait voir s'il était capable de s'adapter à des changements d'environnement. On a trouvé intéressant le fait de toujours partir de l'agent précédent pour l'améliorer en une version meilleure. C'était l'un de nos premiers TP, où on était vraiment dans la création d'un agent qui apprend en fonction des situations qu'il rencontre.

POUR ALLER PLUS LOIN

Nos agents fonctionnent correctement mais on rencontre un petit défaut qu'on pourrait améliorer. On repère ce défaut notamment pour l'agent 3 et 4. En effet, nous sommes partis du principe que tant qu'il ne connaît pas mieux, il continue de faire ce qu'il connaît jusqu'à l'ennui qui le fait changer d'action. Du coup pour l'agent 3, il faut attendre un que l'agent s'ennuie avant de pouvoir découvrir une nouvelle action et ainsi trouver un cycle nous permettant d'obtenir une valeur hédoniste positive. Dans le cas où nous avons que deux actions possibles cela n'est pas très dérangeant, mais si on augmente le nombre d'actions, il faudrait un temps considérable avant de trouver le bon cycle. Pour aller plus loin, on pourrait justement faire en sorte que notre agent teste les différentes actions possibles dès le début pour pouvoir ainsi commencer beaucoup plus tôt les meilleurs cycles qu'il aura trouvé. Ainsi notre agent pourra faire des actions positives beaucoup plus tôt !