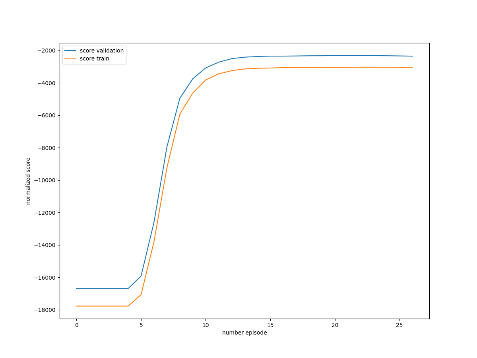
Analyse des hyper paramètre

Pour rappel, notre environnement comporte 365 \*24 = 8 760 *steps* et l’environnement avait un historique de taille 12 avec équinoxe activé (mais pas prédiction activé)

Les premiers paramètre ont été analysés sans coefficient de lissage de la *target* (le réseaux fixe pendant un certain temps)

L’objectif est d’atteindre ce type de courbe



J’ai fixé la graine à 3 pour faire un comparatif mais les comportements sont identiques sans graine fixé.

**Learning rate**

Plage sur lequel a été lancer

1 à 10\*\*(-9) (par puissance de 10)

|  |  |  |
| --- | --- | --- |
| 10\*\*(0) | 10\*\*(-1) | 10\*\*(-2) |
|  |  |  |
| 10\*\*(-3) | 10\*\*(-4) | 10\*\*(-5) |
|  |  |  |
| 10\*\*(-6) | 10\*\*(-7) | 10\*\*(-8) |
|  |  |  |
| 10\*\*(-9) |  |  |
|  |  |  |

Entre 10\*\*(-4) et 1 il y a un saut mais plus d’apprentissage après on a plus aucun apprentissage (ce qui ne faisais pas un beau plot et qui restait sur une borne inférieure à d’autre score

10\*\*(-5) optimal pour le *micro-grid*

10\*\*(-6) apprend de temps en temps mais n’est pas idéal.

10\*\*(-7) à 10\*\*(-9) n’apprend plus (Learning rate trop faible)

**Replay buffer**

Important pour la monotonie

|  |  |  |
| --- | --- | --- |
| 10\*\*6 | 10\*\*5 | 10\*\*(4) |
|  |  |  |
| 10\*\*(3) | 10\*\*(2) | 10\*\*(1) |
|  |  |  |
| 10\*\*(0) |  |  |
|  |  |  |

Trop grand (10\*\*(6)) il stock tous et le score d’apprentissage et très aléatoire (selon ce qu’il va être donné à l’apprentissage les données les plus récente risque de ne pas être utilisée.

Le choix pris est 10\*\*5 pour avoir assez de donnée sans être aveugle à l’action choisi.

Trop petit (10\*\*(4) et moins) il perd trop vite les données et l’entrainement a du mal à démarrer à apprendre et peux aussi facilement désapprendre.

**Taille du batch (lot de donné pris dans le replay buffer pour un pas d’apprentissage)**

Plus la taille du batch est élevé moins les apprentissages étaient aléatoires.

Je ne suis pas aller au-dessus de 256 en donnée lié à un manque de mémoire.

La taille choisie a été donc de 256.

2\*\*0 à 2\*\*6 : trop petit pour commencer un apprentissage correct à coup sûr (trop instable)

2\*\*8 est idéal, les fluctuations de début sont faibles et la courbe dévolution est moins abrute.

2\*\*9 surapprentissage sur le début car peu de donnée stocké (il faut reprendre des valeurs aléatoires dans le replay buffer jusqu’à l’avoir rempli ce qui fait qu’au départ il va apprendre sur la même valeur n fois).

|  |  |  |
| --- | --- | --- |
| 2\*\*0 | 2\*\*1 | 2\*\*2 |
| 2\*\*3 | 2\*\*4 | 2\*\*5 |
|  |  |  |
| 2\*\*6 | 2\*\*7 | 2\*\*8 |
|  |  |  |
| 2\*\*9 | 2\*\*10 |  |
|  |  |  |

**Décroissance de l’exploration**

Pour le choix de ce paramètre est de faire décroitre assez rapidement le taux d’exploration à 10%

Pour cela il fallait résoudre la formule :

Init\_val – X\*decay = min\_val

Ce qui donne X = (init\_val-min\_val)/decay

X représente le nombre de pas dans l’environnement. Or, il nous faut regarder le nombre d’épisode.

Ce qui nous donne la formule

Nb\_ep = X / 8 760 = (init\_val-min\_val)/decay/8 760 = 0,9/decay/8760

Avec un decay de 4\*10\*\*(-6) nous atteignons le min en ~26 épisodes (25,7)

Mais les valeurs entre 2\*10\*\*(-6) et celle choisi sont utilisable cela n’est qu’un choix arbitraire

**Discount factor**

Def : Discounting factor est le terme anglais employé pour désigner le coefficient d'actualisation. Il s'agit d'un coefficient dont l'objectif est de ramener la valeur d'un flux futur à sa valeur actuelle. Ainsi, il s'applique aussi bien à un coût ou un bénéfice futur, mais corrigé à sa valeur actuelle. (<https://www.boursedescredits.com/lexique-definition-discounting-factor-1616.php>)

Même si la valeur habituelle est proche de 1 nous avons abaisser cette valeur pour réduire la chute d’apprentissage si une mauvaise action a été prise. Pour cela nous avons fait vairé cette valeur par pas de 0.1.

L’idéale serais d’avoir une IA qui regarde l’impact de ses choix sur le long terme (1mois).

Pour cela, le discount factor dois être d’environ 0.1 (10% de l’impact du score de l’action choisi) pour le 30eme jour. Ce qui nous donne cette formule

df^(30\*24) = 0.1 => Df = (0.1)^(1/(30\*24)) ~= 0.996

Entre 0.996 et 0.6 l’IA n’arrive tout simplement pas à apprendre

0.4 comme 0.8 ont l’air d’être des solutions envisageables (courbe correct avec un bon score).

En revanche on peut remarquer que le réseau apprend à optimiser au court terme plutôt qu’au long terme (0.8 donne un regard de l’impact de la solution sur 10h)

Entre 0.1 à 0.3 à l’air de donner des résultats incorrects (apprend puis désapprend)

|  |  |  |
| --- | --- | --- |
| 0.1 | 0.2 | 0.3 |
|  |  |  |
| 0.4 | 0.5 | 0.6 |
|  |  |  |
| 0.7 | 0.8 | 0.9 |
|  |  |  |
| 0.99 | 0.996 |  |
|  |  |  |

**Tau, freeze et train freq**

**Train\_freq**

|  |  |  |
| --- | --- | --- |
| 2 | 5 | 8 |
|  |  |  |
| 11 | 14 | 17 à 100 |
|  |  |  |

Comme on peut le constater les valeurs faibles de fréquence d’apprentissage sont de meilleure qualité.

Les valeurs entre ces trois paramètres sont intimant liés, il me reste à tester des combinaisons

**Freeze\_freq**

Quel que soit la valeur de fréquence de freeze prise entre 1 et 1000 les courbes sont presque identiques.

Après 1000 les apprentissages sont moins performant.

10 000 et plus les résultats de départ et/ou d’arrivé deviennent de moins en moins bon.

|  |  |  |
| --- | --- | --- |
| 1 à 1000 | 1000 | 10 000 |
|  |  |  |
| 100 000 | 1 000 000 |  |
|  |  |  |

**Tau**

Les courbes d’évolutions ne changent quasi pas. En effet, à part 0.1, 0 .15, et 1.0, les autres courbes sont identiques.

|  |  |  |
| --- | --- | --- |
| 0.1 | 0.15 | 0.2 |
|  |  |  |
| 0.25 | 0.3 | 0.35 |
|  |  |  |
| 0.4 | 0.45 | 0.5 |
|  |  |  |
| 0.55 | 0.6 | 0.65 |
|  |  |  |
| 0.7 | 0.75 | 0.8 |
|  |  |  |
| 0.85 | 0.9 | 0.95 |
|  |  |  |
| 1. |  |  |
|  |  |  |

**Conclusion :**

Nos hyperparamètres sont :

Lr : 10\*\*(-5)

Df : 0.8

Train\_freq : 2

Freeze\_freq : 50 (valeur arbitraire qui dois être entre 1 et 1000)

Tau : 0.95

Decay : 4 \* 10\*\*(-6) (par calcul numérique)

Replay\_buff\_size : 10\*\*(5)

Size\_buff : 2\*\*(8) = 256