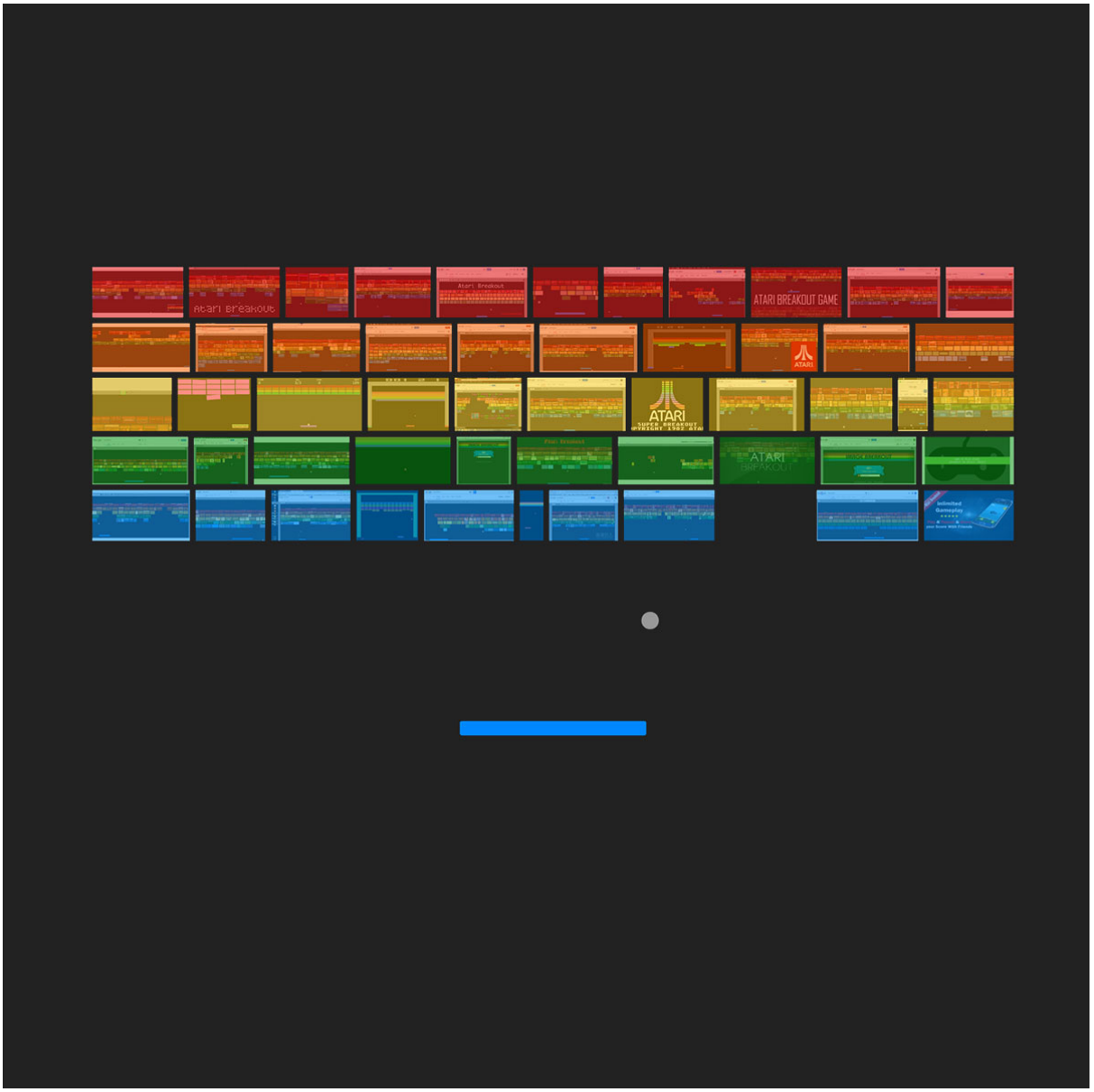
Rapport IA

Breakout - IMAGE - Policy Gradient

Comparison of strategies for epsilon

Semestre 8 - 2021

GIRARD Alexandre
POUEYTO Clément



Introduction

Nous avons choisi le jeu Breakout d'Atari (Breakout-v0), en utilisant les images de la partie afin d'améliorer le comportement de notre intelligence artificielle.
Nous utilisons la méthode Policy Gradient afin d'améliorer notre IA au fil de l'exécution des épisodes.
Le but de ce projet est de varier les différentes valeurs d'epsilon afin de comparer les résultats d'entraînement du programme.
En effet, la valeur d'epsilon permet lors du déroulement de l'algorithme de décider si le but de la prochaine action est d'explorer l'environnement ou bien d'utiliser les données récoltées afin de réaliser une action calculée.

Règles

Le but du jeu Breakout est de casser toutes les briques en renvoyant la balle avec la plateforme. Un épisode correspond à 5 vies. Lorsque la balle tombe, une vie est perdue.
Il y a 4 actions possibles :

1) Ne rien faire
2) Lancer la balle
3) Se déplacer à droite
4) Se déplacer à gauche

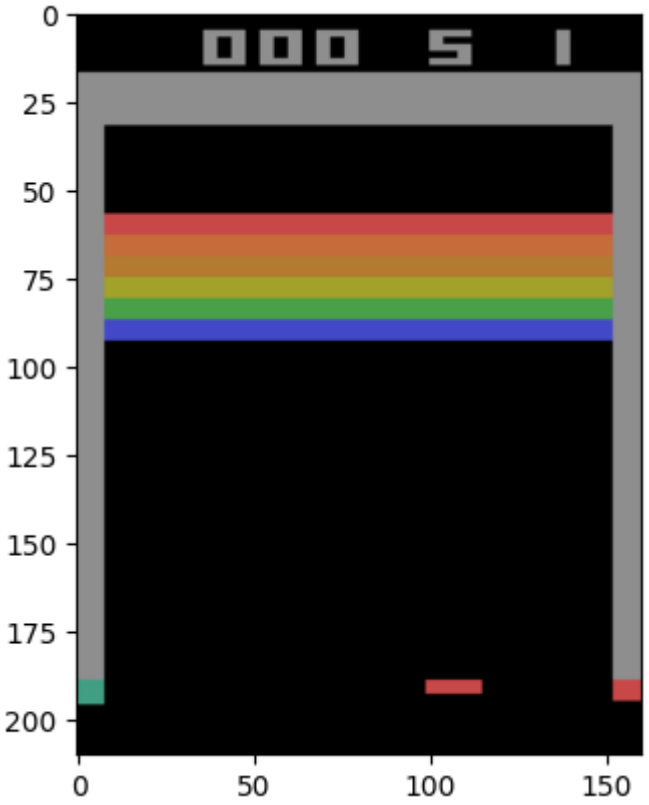
Dans notre cas, nous avons retiré la possibilité à l'I.A de lancer la balle. En effet, cela n'était pas vraiment intéressant pour le projet puisqu'il faudrait que l'I.A détecte lorsque la balle n'est plus à l'écran. Sans entraînement au préalable, le programme ne pourrait donc pas fonctionner. Ainsi, c'est le programme qui va relancer la balle lorsqu'une vie est perdue
Comme pour le jeu Pong, nous n'avons donc que 3 actions réalisables par le programme.

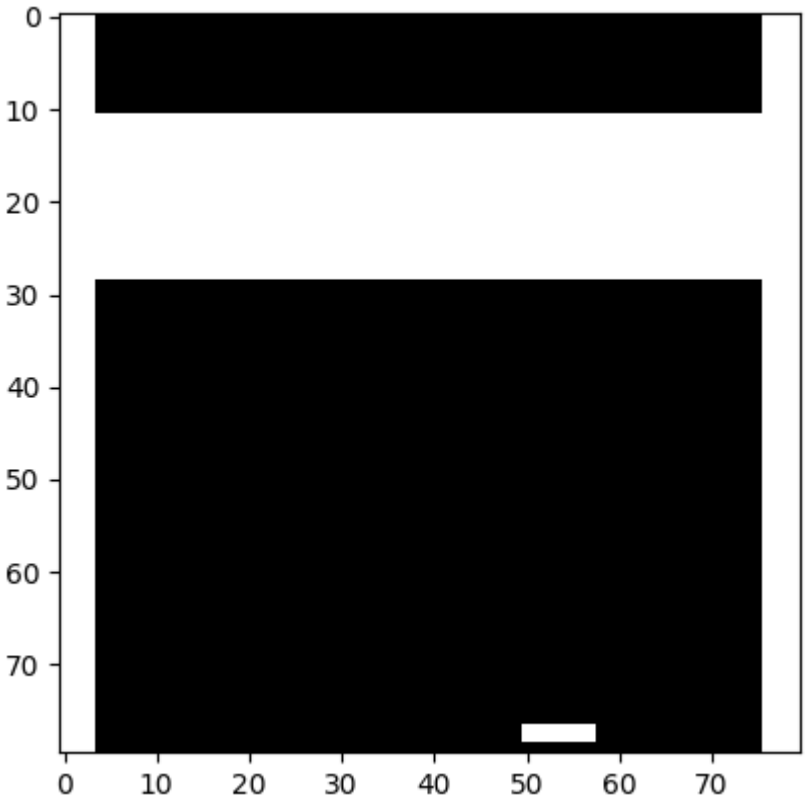
Dans notre modèle d'apprentissage, cela correspond à 3 entrées. (1, 3, 4)

Pour nous aider, nous avons utilisé la base du code fourni dans le TP5, que nous avons modifié pour le jeu du casse-brique et pour coller à notre sujet d'étude.

Images

Nous souhaitons faire jouer l'IA toute seule afin d'apprendre de ses erreurs.
Notre programme capture les images du jeu associé à un score pour les fournir au modèle d'entraînement.
De plus, nous utilisons 2 images successives afin de saisir le mouvement de la balle dans l'environnement.
Ces 2 images sont redimensionnées en 80x80, et transformées en noir et blanc puis soustraites entre elles.





Reward policy

Pour les essais nous avons fixé les valeurs suivantes:

- learning rate : 0.1
- gamma : 0.99
- fréquence de mise à jour des paramètres : 1

Nous avons testé 2 techniques de récompenses différentes afin d'observer le comportement de l'IA après plusieurs centaines d'épisodes.

1) Dans un premier temps nous avons essayé de récompenser l'IA lorsque celle-ci renvoyait la balle (+1) et de la pénaliser lorsqu'elle perdait une vie (-1). Le score de récompense se situait donc entre -5 et N le nombre de blocs.
   Après plusieurs essais de 100 épisodes, nous avons noté que le comportement était très similaire d'un entraînement à l'autre. L'IA se bloquait dans un côté car cela permettait de renvoyer la balle dès son lancement, mais elle ne la rattrapait que très rarement au 2e rebond car son "reward" était alors de 0 (+1 -1). Au fur et à mesure, elle n'était donc pas assez entraînée à renvoyer la balle par la suite.

De plus, en ne prenant que des actions aléatoires au début, il était très difficile d'obtenir un score de récompense positif.

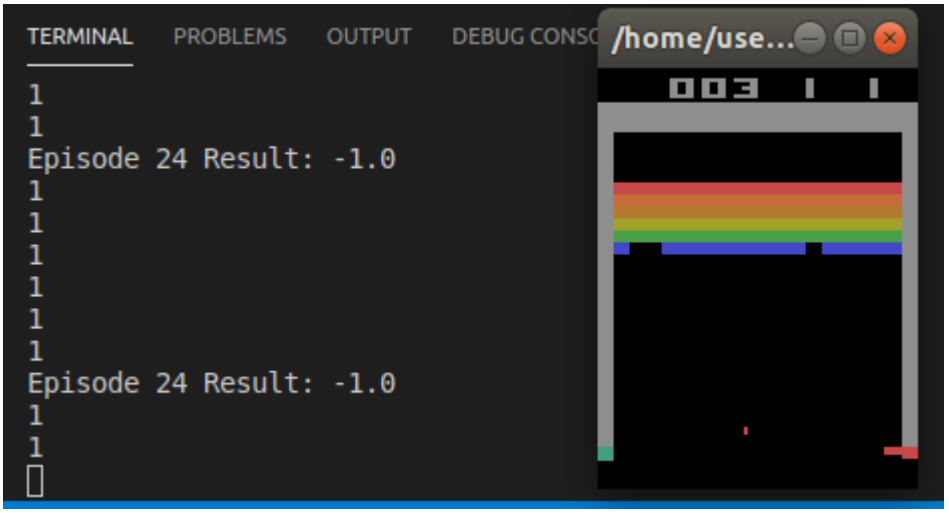
2) Dans un second temps, nous avons cherché à uniquement récompenser l'IA lorsqu'elle renvoie la balle. Ainsi, la récompense en cas de vie perdue n'avait pas d'influence sur le score final. Une défaite ne fait pas perdre de points, la seule différence se fait sur le nombre de balles renvoyées au cours de l'épisode.
   Le score de récompense se situe entre 0 et N le nombre de blocs

Il est difficile de trouver la meilleure méthode car nous avons été confrontés à des soucis de performance. Il était très compliqué de part le temps et les performances de l'ordinateur de dépasser les 1000 épisodes.

Lorsque l'IA doit sélectionner une action, elle obtient 3 probabilités pour 3 actions et en sélectionne une aléatoirement. C'est pourquoi il est important d'effectuer un grand nombre d'épisodes afin d'obtenir des valeurs plus sûres.

Ainsi pour un faible nombre d'épisodes le score moyen au bout de 500 épisodes semble être meilleur avec la première technique qui atteignait -2.31. La balle était donc renvoyée presque une fois par vie. Cependant il se pourrait qu'avec plus d'épisodes, la 2e technique pourrait s'avérer meilleure.

Par la suite, nous avons repris ces méthodes en utilisant la fonction argmax() de numpy afin de sélectionner une action en fonction du coefficient le plus haut parmi les 3 actions.
Comme on peut le voir sur l'image suivante :



L'IA converge encore plus rapidement au bout de quelques épisodes vers la même valeur,
1 ou 2. (droite ou gauche)

Epsilon-greedy

L'apprentissage de l'algorithme dépend de ses anciennes parties. C'est pourquoi, il n'est pas possible pour l'IA de décider sans données au préalable. Il faut donc lui permettre de jouer aléatoirement durant les premiers épisodes et ainsi lui permettre de découvrir l'environnement.
On fixe donc une valeur Epsilon de départ entre 0 et 1 symbolisant une probabilité de jouer complètement aléatoirement ou bien de jouer en utilisant un modèle de prédiction. A chaque épisode, on diminue la valeur d'Epsilon afin de diminuer cette probabilité.
Plus le nombre d'épisodes augmente, plus les actions réalisées seront liées au modèle de prédiction.

Dans un premier temps nous avons fixé la valeur de départ d'Epsilon à 1 puis à chaque épisode cette valeur était multipliée par un pourcentage fixé au préalable.
(~0.995)

Puis nous avons décidé de fixer le coefficient de soustraction en fonction d'epsilon, du nombre d'épisodes et de la valeur minimum d'Epsilon tel que :

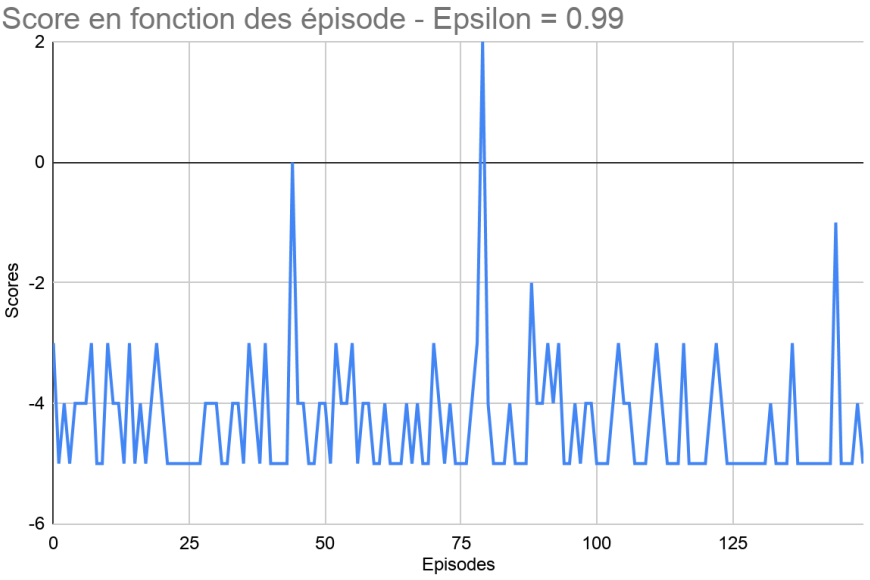
Epsilon_decay = (Epsilon - Epsilon_Min) / nombre d'épisodes

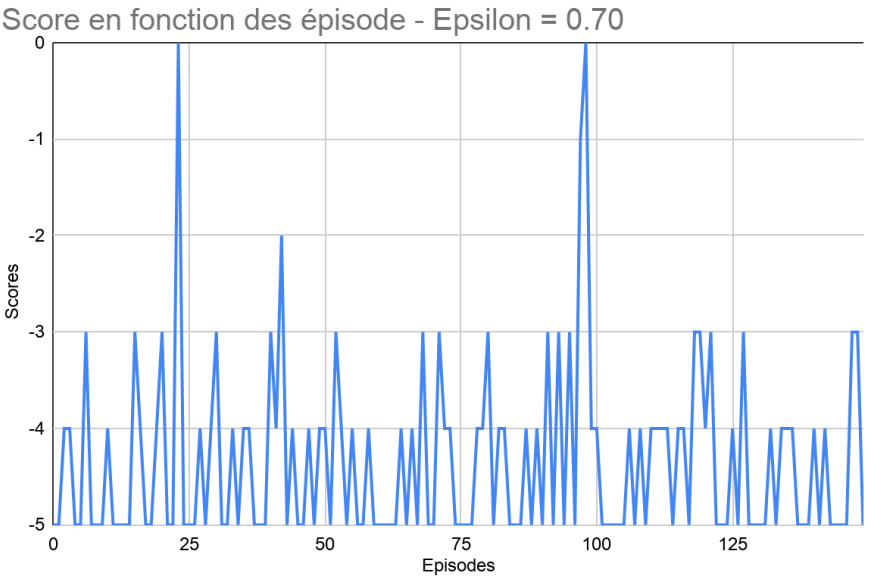
A chaque épisode on soustrait Epsilon_decay à la valeur d'Epsilon car cela permet d'avoir un retour sur l'efficacité de la stratégie même pour un faible nombre d'épisodes.

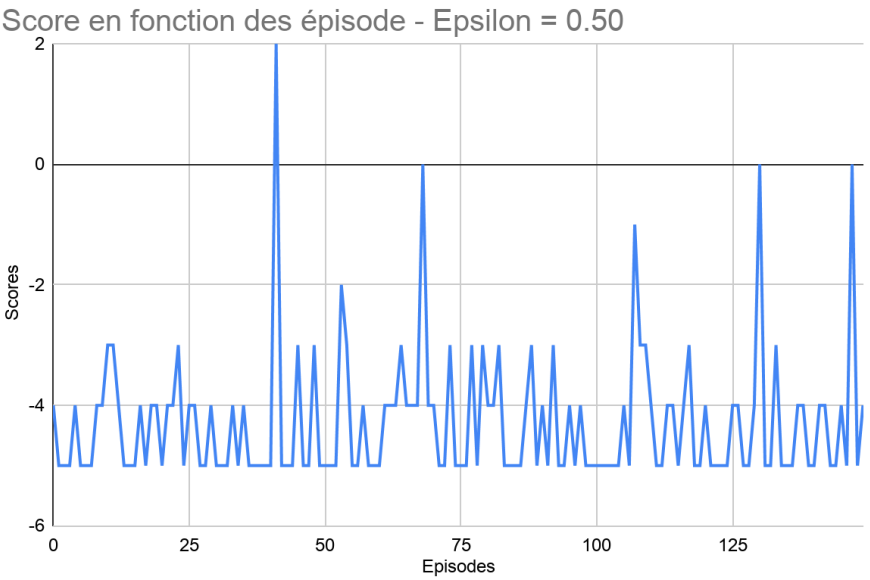
Enfin dans les deux cas, on fixe une valeur minimum d'Epsilon afin de garder une part aléatoire lorsqu'on utilise np.argmax() pour essayer de ne pas converger vers une seule valeur.

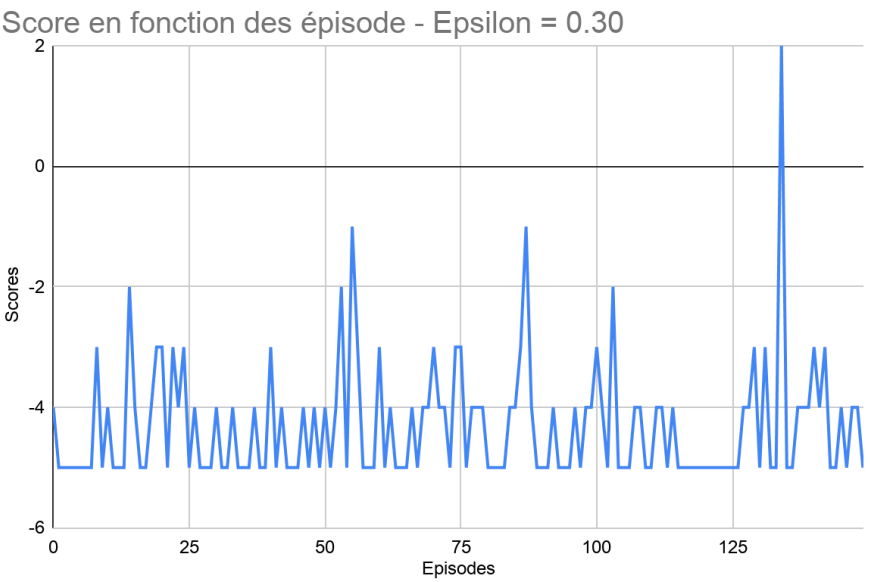
Tests avec différents epsilon

Nous avons entraîné l'intelligence artificielle en faisant varier le paramètre epsilon. Mais comme dit précédemment, par manque de performance, l'IA prendrait des jours à apprendre et atteindre un niveau convenable.
Voici les graphiques contenant les données d'entraînement, même si les résultats ne sont pas vraiment significatifs.






Score en fonction des épisode - Epsilon = 0.50
Scores
Episodes
2
0
-2
-4
-6
0
25
50
75
100
125


Score en fonction des épisode - Epsilon = 0.30
Scores
Episodes
2
0
-2
-4
-6
0
25
50
75
100
125