|  |
| --- |
| Sciences informatique 1ère année |
| Rapport : Projet d'année Intelligence Artificielle Quoridor |
| Université Libre de Bruxelles |
|  |
| **Loris Wintjens** |
| **11/04/2019** |

|  |
| --- |
| [Tapez le résumé du document ici. Il s'agit généralement d'une courte synthèse du document. Tapez le résumé du document ici. Il s'agit généralement d'une courte synthèse du document.] |

Table des matières

**Introduction3**

**Méthodes4**

Stratégies d'apprentissages4

Stratégies gloutonnes5

Fonctions d'activations5

**Résultats6**

**Discussion7**

**Conclusion8**

**1 Introduction**

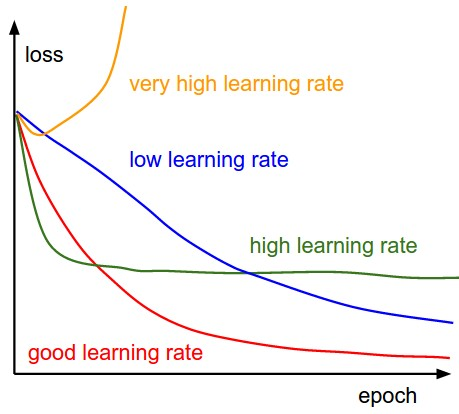
Dans ce rapport, nous allons décrire nos observations et conclusions faites suites au développement d'une Intelligence Artificielle en Python 3 pour le jeu de plateau *Quoridor*[[1]](#footnote-1). Le but de chaque joueur est d'arriver le premier de l'autre côté du plateau. Afin d'empêcher l'adversaire d'atteindre son but, chaque joueur dispose de murs qui peuvent être placés sur le plateau entre les cases, appelées tuiles. Cette Intelligence Artificielle développée, optimisée pour Quoridor est entrainée contre elle-même en utilisant des *méthodes d'apprentissage par renforcement*[[2]](#footnote-2) en combinaison avec des *réseaux de neurones*[[3]](#footnote-3).

**2 Méthodes**

Dans cette partie, nous allons décrire les différentes méthodes utilisées sur notre Intelligence Artificielle et expliciter les choix faits.

**2.1 Stratégies d'apprentissages**

Il est important de noter que dans les stratégies d'apprentissages présentées, le facteur d'apprentissage est un élément majeur et nous avons pris la décision de le mettre à 0,4. Il est important de trouver un juste équilibre entre exploration et exploitation, choisir un coup non-optimal mais qui pourrait nous faire découvrir de nouvelles stratégies par après ou choisir le coup optimal. Un facteur nul entraine simplement de l'exploitation et aucune exploration (l'IA exploite exclusivement des connaissances antérieures), tandis qu'un facteur de 1 force l'IA à ne prendre en compte que les informations les plus récentes (en ignorant les connaissances antérieurs pour explorer les possibilités).

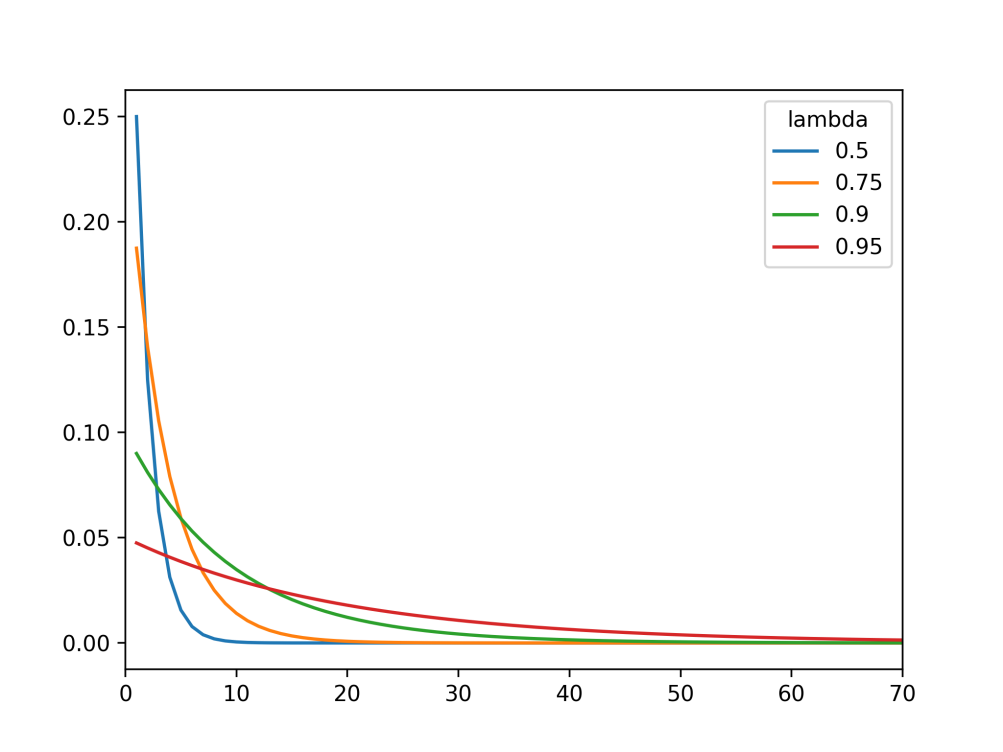


Effet des différents taux d'apprentissage sur la convergence (Img Credit: [cs231n](http://cs231n.github.io/neural-networks-3/))

**2.1.1 TD-Lambda**

Cette méthode d’apprentissage fait partie d’une famille de stratégies on-line appelées *Temporal Difference Learning[[4]](#footnote-4)* . Les stratégies on-lines permettent d’apprendre en même temps qu’on interagit avec l’environnement (dans le cas de Quoridor, qu’on se déplace sur le plateau de jeu). Le principe de cette stratégie est basée sur le bootstraping, nous utilisons l’approximation actuelle d'un état[[5]](#footnote-5) (qui pourrait être fausse) pour mettre à jour l’estimation d’un autre état. Ceci est similaire à TD(0) qui consiste à évaluer la probabilité d’un mouvement à partir de l’état après-coup correspondant. Seulement, nous ajoutons un concept d' "egilibity trace", consistant à intervenir dans les cas où 0 < λ ≤ 1, comme une trace en mémoire à court terme. Au lieu d'attendre ce qui va se passer pour effectuer une mise à jour, nous nous souviendrons de ce qui s'est passé et utiliserons les informations actuelles pour mettre à jour les approximations pour chaque état. Les raisons pour lesquelles nous avons opté pour TD-Lambda au lieu de Monte-Carlo et TD(0). M-C a beaucoup de variations étant donné qu'on utilise le résultat d'une partie pour mettre à jour les états. Ce qui implique qu'il y a une grosse partie d'aléatoire dans le choix d'un mouvement. Tandis que TD(0) utilise l'information que d'un mouvement pour mettre à jour un état.

Le paramètre lambda est définit comme étant la vitesse de décroissance de l'egibility trace. Nous avons développé notre algorithme avec un lambda égal à 0,9. Le graphique ci-dessous nous montre la valeur de (1−*λ*)*λn* pour des valeurs de n différentes.l'impact des différentes valeurs de lambda sur la valeur initiale de retour et comment cette valeur décroit avec le temps. Des valeurs plus importantes de lambda entraînent une décroissance plus lente (donc les eligibility traces sont plus durables et les informations du passé ont une importance non négligeable). C'est la raison pour laquelle nous avons choisi un lambda égal à 0,9.



Graphe montrant l'évolution de (1−*λ*)*λn* afin de d'imager l'impact des différentes valeurs de lambda (n évoluant avec le temps, représentant le nombre de parties)

**2.1.2 Q-Learning**

Q-Learning est une autre méthode d'apprentissage similaire à TD(0), mettant à jour les probabilités selon le choix de mouvement glouton, c'est à dire celui maximisant la probabilité. Il est à noter que cette méthode est similaire à TD(0) dans le cas d'un choix glouton. La mise à jour des estimations s'effectue selon cette formule :

ps′←(1−α)·ps′+α·ps\*

Mais il est important de noter que dans le cas d'un mouvement glouton, ps\*, s\* étant l'état-après-coup correspondant à un choix glouton, ps\* est donc égal à ps. Ce qui correspond à la mise à jour des estimations selon TD(0) qui se définit comme tel :

ps′←(1−α)·ps′+α·ps

1. https://www.gigamic.com/files/catalog/products/rules/quoridor-classic-fr.pdf [↑](#footnote-ref-1)
2. Barto, Andrew G and Sutton, Richard S. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, 1998. [↑](#footnote-ref-2)
3. John A. Hertz, Richard G. Palmer, Anders S. Krogh.. Introduction to the theory of neural computation .CRC Press Web, 2018 [↑](#footnote-ref-3)
4. Un état se définit comme étant un mouvement qui mène à un état [↑](#footnote-ref-4)
5. [↑](#footnote-ref-5)