Apprentissage par renforcement

Sergey Kirgizov

Université de Bourgogne

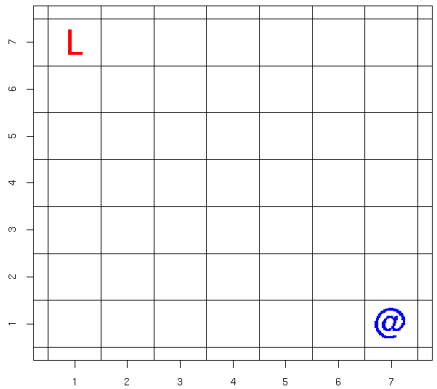
2016

Exemple motivant

Jeu du loup

Exemple motivant

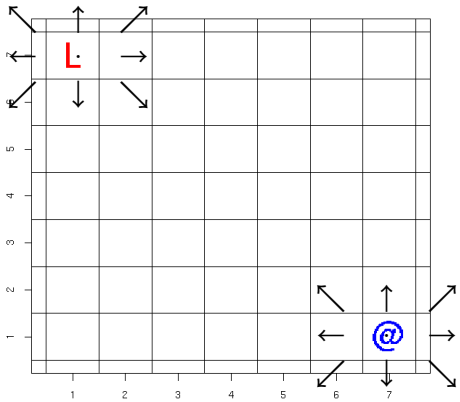
Jeu du loup



Pas bien pour le loup ↑

Exemple motivant

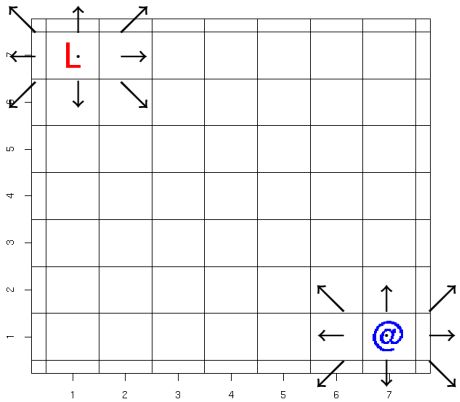
Jeu du loup

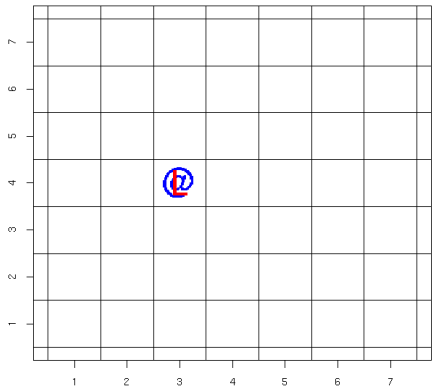


Pas bien pour le loup ↑

Exemple motivant

Jeu du loup





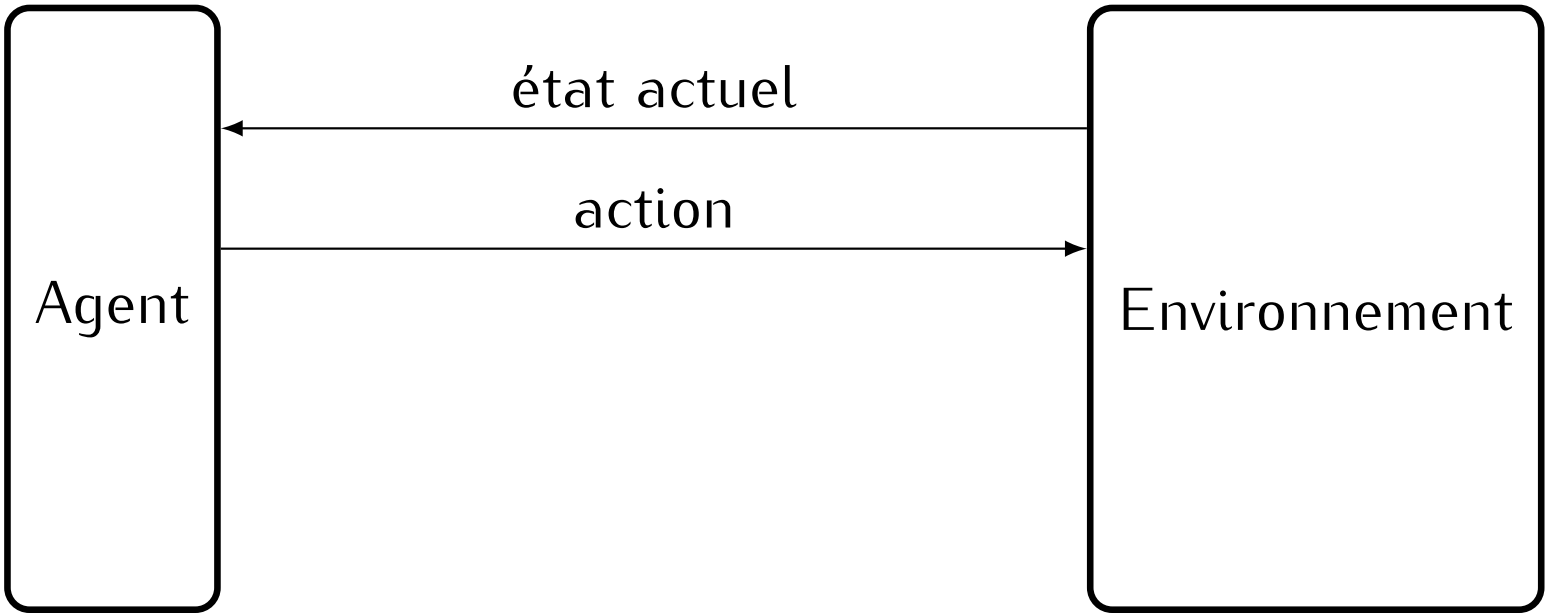
Pas bien pour le loup ↑

↑ Bien pour le loup

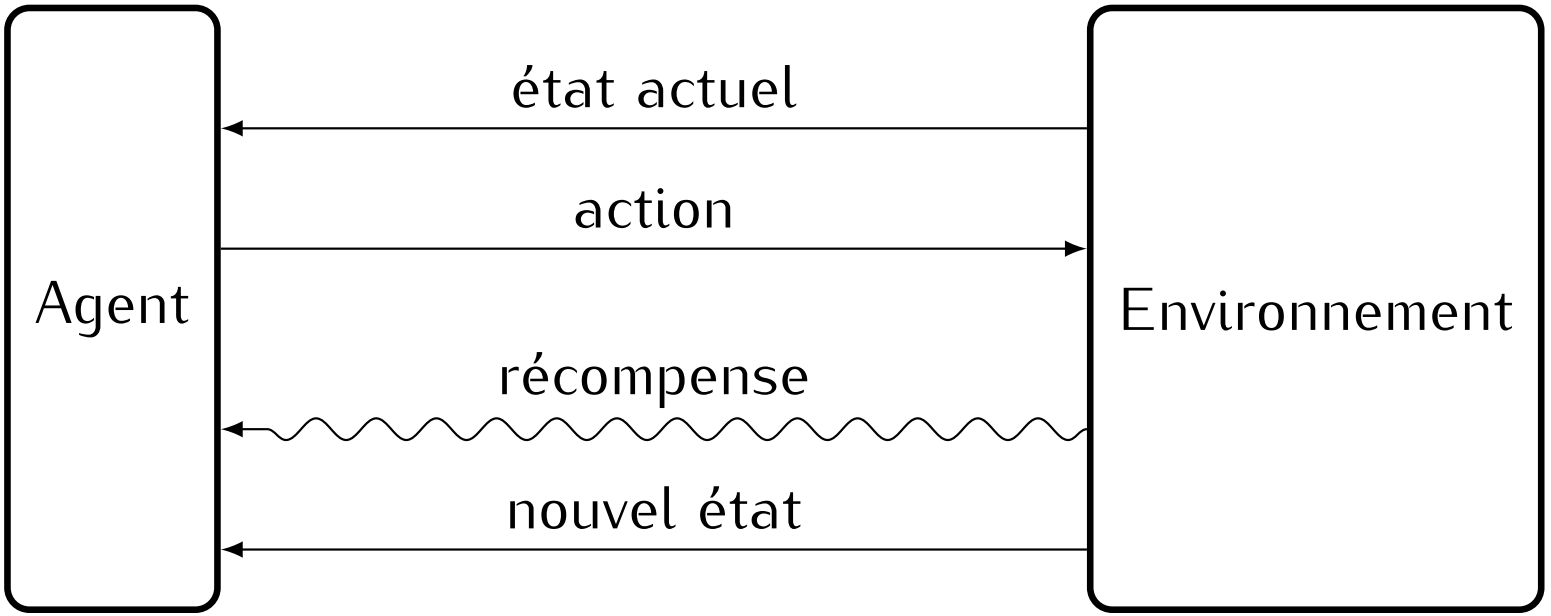
Idée de l'apprentissage par renforcement

Idée de l'apprentissage par renforcement

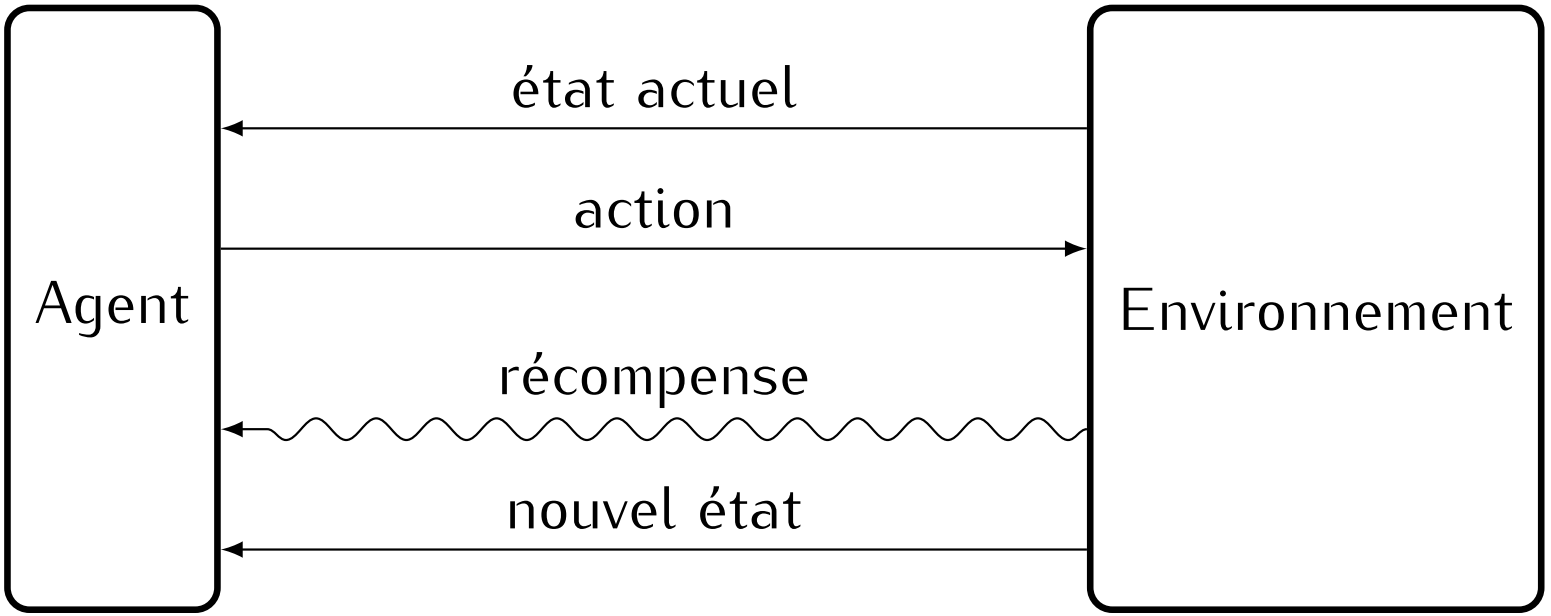
Idée de l'apprentissage par renforcement



Idée de l'apprentissage par renforcement



Idée de l'apprentissage par renforcement



But de l'agent : choisir les actions afin de maximiser les récompenses immediates et futures

Historique

Historique

Apprentissage par renforcement

1988 TD-learning, Richard Sutton

1989 Q-learning, Chris Watkins

Historique

"Toutefois, l'origine de l'apprentissage par renforcement est plus ancienne. Elle dérive de formalisations théoriques de méthodes de contrôle optimal, visant à mettre au point un contrôleur permettant de minimiser au cours du temps une mesure donnée du comportement d'un système dynamique. La version discrète et stochastique de ce problème est appelée un processus de décision markovien et fut introduite par Richard Ernest Bellman en 1957"

https://fr.wikipedia.org/wiki/Apprentissage_par_renforcement

Théories de psychologie animale → Intelligence artificielle

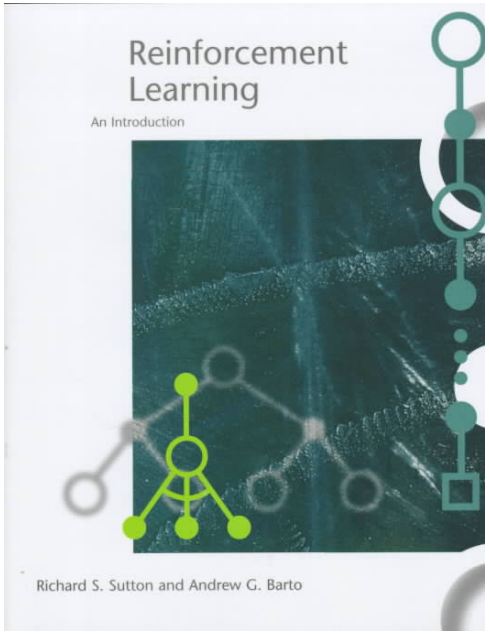
"D'autre part, la formalisation des problèmes d'apprentissage par renforcement s'est aussi beaucoup inspirée de théories de psychologie animale, comme celles analysant comment un animal peut apprendre par essais-erreurs à s'adapter à son environnement. Ces théories ont beaucoup inspiré le champ scientifique de l'intelligence artificielle et ont beaucoup contribué à l'émergence d'algorithmes d'apprentissage par renforcement au début des années 1980."

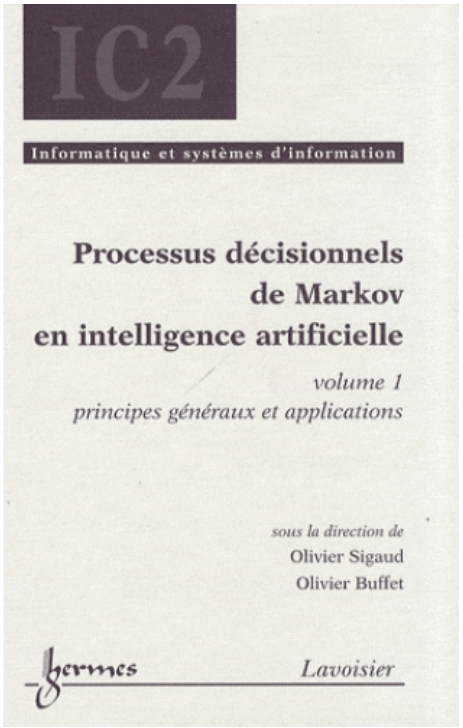
— Wikipedia

Théorie

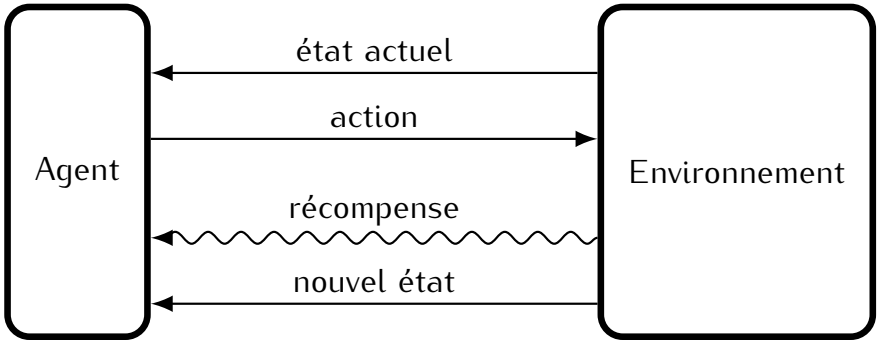
Livres

1. **Reinforcement Learning : An Introduction**
   Richard S. Sutton et Andrew G. Barto
2. **Processus décisionnels de Markov en intelligence artificielle**
   Olivier Sigaud et Olivier Buffet





Processus de décision markovien (MDP)



But de l'agent : choisir les actions afin de maximiser les récompenses immediates et futures

Definitions

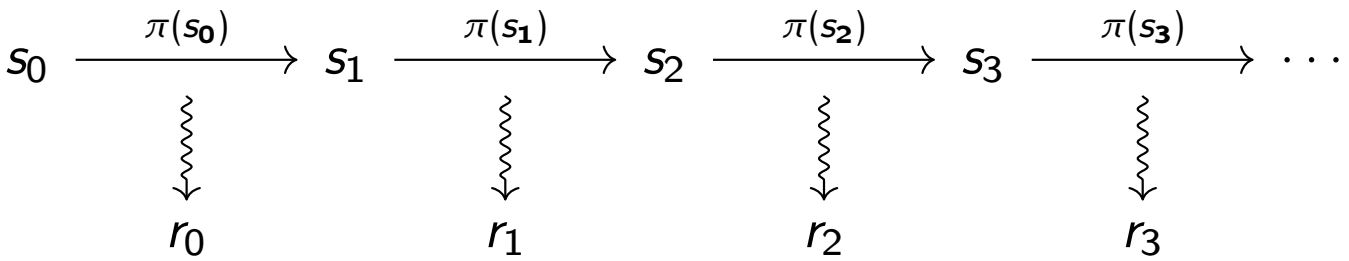
- S — ensemble d'états
- A — ensemble d'actions
- $\Pr[s'|s,a]$ — probabilité de se retrouver dans l'état s' en faisant l'action a dans l'état s (probabilité de transition)
- $R : S \times A \times S \to \mathbb{R}$ — fonction de récompense
- $\pi : S \to A$ — politique de l'agent

Processus de décision markovien (MDP)

Definitions

- S — ensemble d'états
- A — ensemble d'actions
- $\Pr[s'|s,a]$ — probabilité de se retrouver dans l'état s' en faisant l'action a dans l'état s (probabilité de transition)
- $R : S \times A \times S \to \mathbb{R}$ — fonction de récompense
- $\pi : S \to A$ — politique de l'agent

Processus



s_n — état à la n-ième iteration

r_n — n-ième récompense

Bonne politique maximise $\mathbb{E}\sum_{n=0}^{\infty}\gamma^n r_n$, $0 < \gamma < 1$

Q-learning

Bonne politique maximise l'espérance de $\sum_{n=0}^{\infty} \gamma^n r_n$

Comment trouver cette politique ?
Par exemple, par l'algorithme Q-learning.

Definition

- $Q : S \times A \to \mathbb{R}$ — fonction qualité de l'action
- $\pi(s) = \arg\max_{a \in A} Q(s, a)$

Algorithm

Initialisation : $\forall s \in S, a \in A \quad Q_0(s, a) \leftarrow 0$

Règle de mise à jour :

$$Q_{n+1}(s_n, a_n) \leftarrow (1 - \alpha_n) \cdot Q_n(s_n, a_n) + \alpha_n \cdot \left(r_n + \gamma \max_a Q_n(s_{n+1}, a) \right)$$

Convergence

Algorithm

$Q_{n+1}(s_n, a_n) \leftarrow Q_n(s_n, a_n)(1 - \alpha_n) + \alpha_n \left(r_n + \gamma \max_a Q_n(s_{n+1}, a)\right)$

Optimum

$\mathfrak{Q}$ — ensemble de fonctions de type $S \times A \to \mathbb{R}$.
$Q^* \in \mathfrak{Q}$ est une fonction t.q. $\forall Q \in \mathfrak{Q}, s \in S, a \in A$ on a $Q(s, a) \leq Q^*(s, a)$

On peut prouver l'existence de Q^* en utilisant le théorème de point fixe de Banach.

Convergence

Algorithm

$Q_{n+1}(s_n, a_n) \leftarrow Q_n(s_n, a_n)(1 - \alpha_n) + \alpha_n \left(r_n + \gamma \max_a Q_n(s_{n+1}, a)\right)$

Théorème

Q_n converge presque sûrement vers Q^* si :

1. A et S sont finis
2. les probabilites de transitions ne changent pas
3. l'agent essaie chaque action dans chaque état un nombre infini de fois
4. $\sum_{n=0} \alpha_n = \infty$ et $\sum_{n=0}^{\infty} \alpha_n^2 < \infty$
5. $\gamma < 1$

La preuve est basée sur les idées des algorithmes itératives d'optimisation stochastiques des années 1950 (algorithms de Robbins-Monro, Kiefer-Wolfowitz, etc)

Exploration de l'environnement

Il faut agir et apprendre

Alors,

- parfois l'agent utilise Q : $\pi(s) = \arg\max_{a \in A} Q(s, a)$
- parfois il agit au hasard : $\pi(s) = \text{rand}(a \in A)$

Remarque

Algorithm

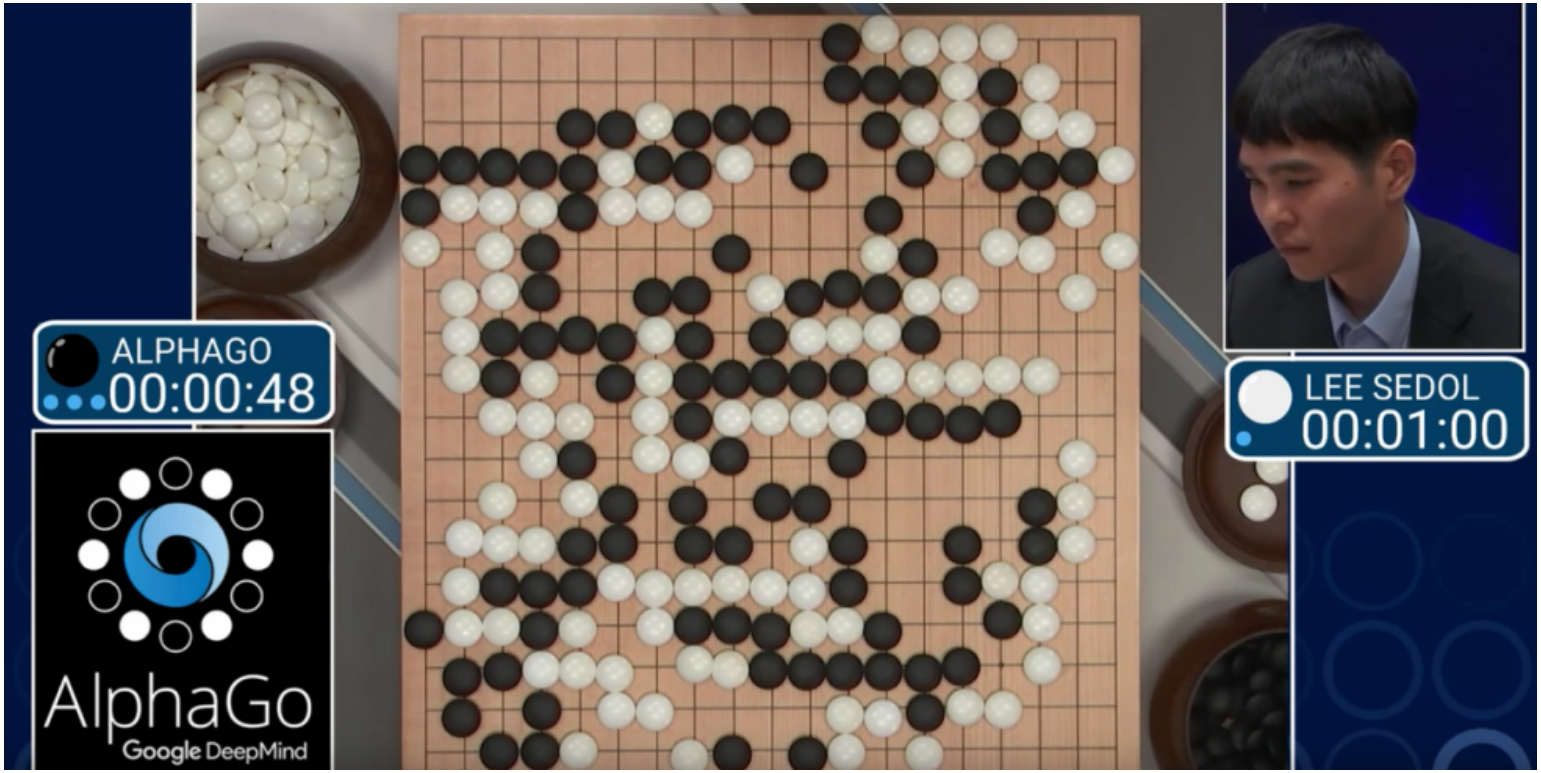
$Q_{n+1}(s_n, a_n) \leftarrow Q_n(s_n, a_n)(1 - \alpha_n) + \alpha_n \left(r_n + \gamma \max_a Q_n(s_{n+1}, a)\right)$

α_n est parfois appelée “la vitesse d’apprentissage”.

Dans la vie réelle les probabilites de transitions ne sont pas nécessairement fixes. Il faut apprendre toujours et donc α_n peut être constante.

Applications et recherche actuelle

AlphaGo



"This program was based on general-purpose AI methods [**including reinforcement learning**], using deep neural networks to mimic expert players, and further improving the program by learning from games played against itself."

— `https://deepmind.com/research/alphago`

Allocation des ressources dans le cloud

ICAS 2011 : The Seventh International Conference on Autonomic and Autonomous Systems

Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow

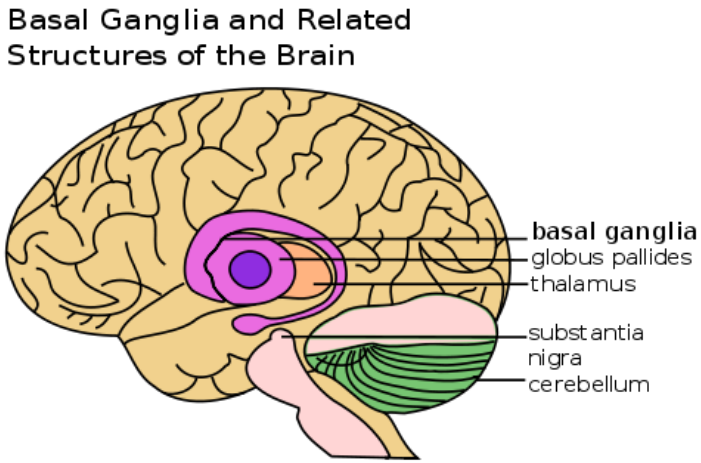
Xavier Dutreilh[†], Sergey Kirgizov[*], Olga Melekhova[*], Jacques Malenfant[*], Nicolas Rivierre[†] and Isis Truck[‡]
[*]Université Pierre et Marie Curie – Paris 6 CNRS, UMR 7606 LIP6, 4 place Jussieu, Paris, 75005, France
Email: {Olga.Melekhova, Jacques.Malenfant}@lip6.fr
[†]Orange Labs, 38-40 rue du Général Leclerc, Issy-les-Moulineaux, 92130, France
Email: {xavier.dutreilh, nicolas.rivierre}@orange-ftgroup.com
[‡]LIASD – EA 4383, Université Paris 8, 2 rue de la Liberté, Saint-Denis Cedex, 93526, France
Email: truck@ai.univ-paris8.fr

***Abstract*—Dynamic and appropriate resource dimensioning is a crucial issue in cloud computing. As applications go more and more 24/7, online policies must be sought to balance performance with the cost of allocated virtual machines. Most industrial approaches to date use *ad hoc* manual policies, such as threshold-based ones. Providing good thresholds proved to be tricky and hard to automatize to fit every application requirement. Research is being done to apply automatic decision-making approaches, such as reinforcement learning. Yet, they face a lot of problems to go to the field: having good policies in the early phases of learning, time for the learning to converge to an optimal policy and coping with changes in the application performance behavior over time. In this paper, we propose to deal with these problems using appropriate initialization for the early stages as well as convergence speedups applied throughout the learning phases and we present our first experimental results for these. We also introduce a performance model change detection on which we are currently working to complete the learning process management. Even though some of these proposals were known in the reinforcement learning field, the key contribution of this paper is to integrate them in a real cloud controller and to program them as an automated workflow.**

they don't require the *a priori* knowledge of the application performance model, but rather learn it as the application runs. Yet, RL faces a lot of problems to go to the field [3][4], such as: having good policies in the early phases of learning, time for the learning to converge to an optimal policy and coping with changes in the application performance behavior over time. In this paper, we propose to deal with these problems using appropriate initialization for the early stages, convergence speedups applied throughout the learning phases and performance model change detection. Even though some of these proposals were known in the RL field, the key contribution of this paper is to integrate them in a real cloud controller and to program them as an automated workflow.

We present our first results towards this automated learning management workflow. Section II introduces the resource allocation problem for cloud computing. Section III presents the formulation of the problem in the Q-learning framework, as we have modeled it for a private cloud deployed at Orange

Neurobiologie



"La collaboration entre neurobiologistes et chercheurs en intelligence artificielle a permis de découvrir qu'une partie du cerveau fonctionnait de façon très similaire aux algorithmes d'apprentissage par renforcement."

— Wikipedia

A Model of How the Basal Ganglia Generate and Use Neural Signals that Predict Reinforcement

James C. Houk, J. L. Adams, A. G. Barto

Recherche actuelle et future

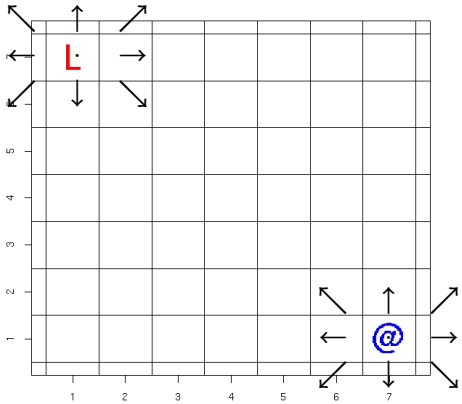
- Environnements sont plus grands
- Il faut augmenter la vitesse de convergence
- Combinaison avec d'autres techniques d'apprentissage par machine (les réseaux de neurones, etc)

En TP : Jeu du loup en R

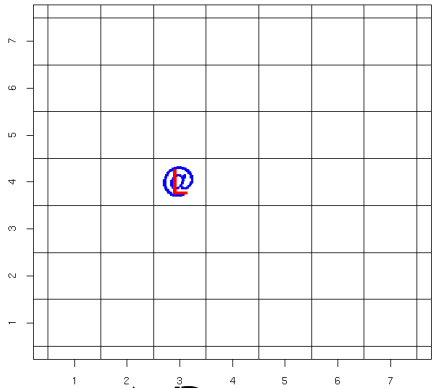
Jeu du loup

en R

Code source : `https://github.com/kerzol/jeu-du-loup`



Pas bien pour le loup ↑



↑ Bien pour le loup

En TP / Devoir à la maison

1. Télécharger le code source : `https://github.com/kerzol/jeu-du-loup`
2. Lire le code, comprendre le code
3. Trouvez et décrire les différences entre `q-loup.R` et `q-loup-1.R`
4. Pourquoi `q-loup-1.R` apprend mieux le comportement cyclique du chat ?
5. Visualisez l'evolution de valeurs de Q
6. Change `ALPHA = 0.6` dans le code par quelque chose qui respecte la condition $\sum_{n=0}^{\infty} \alpha_n < \infty$
7. Ajouter quelques murs dans l'environnement.

Merci