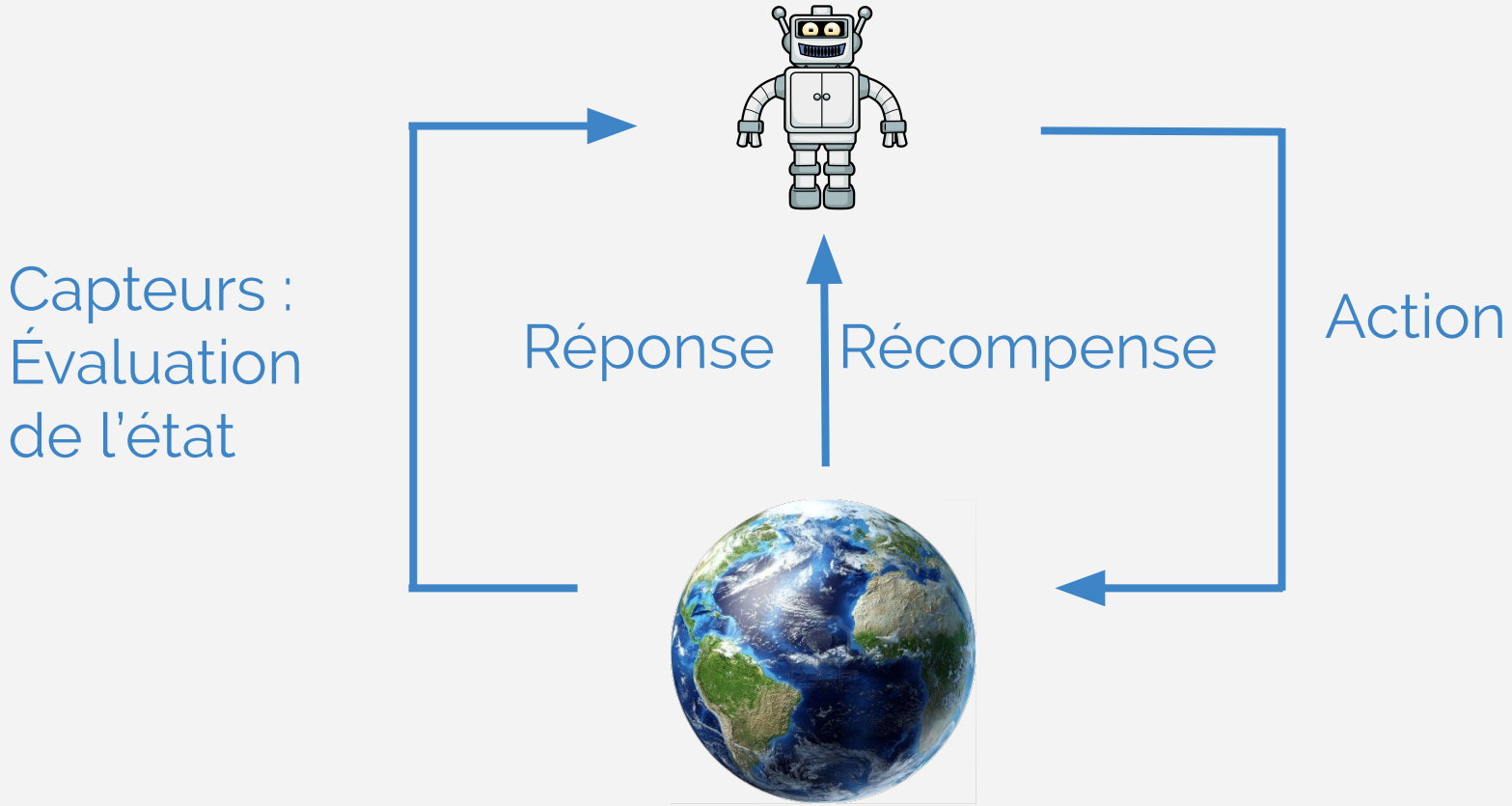
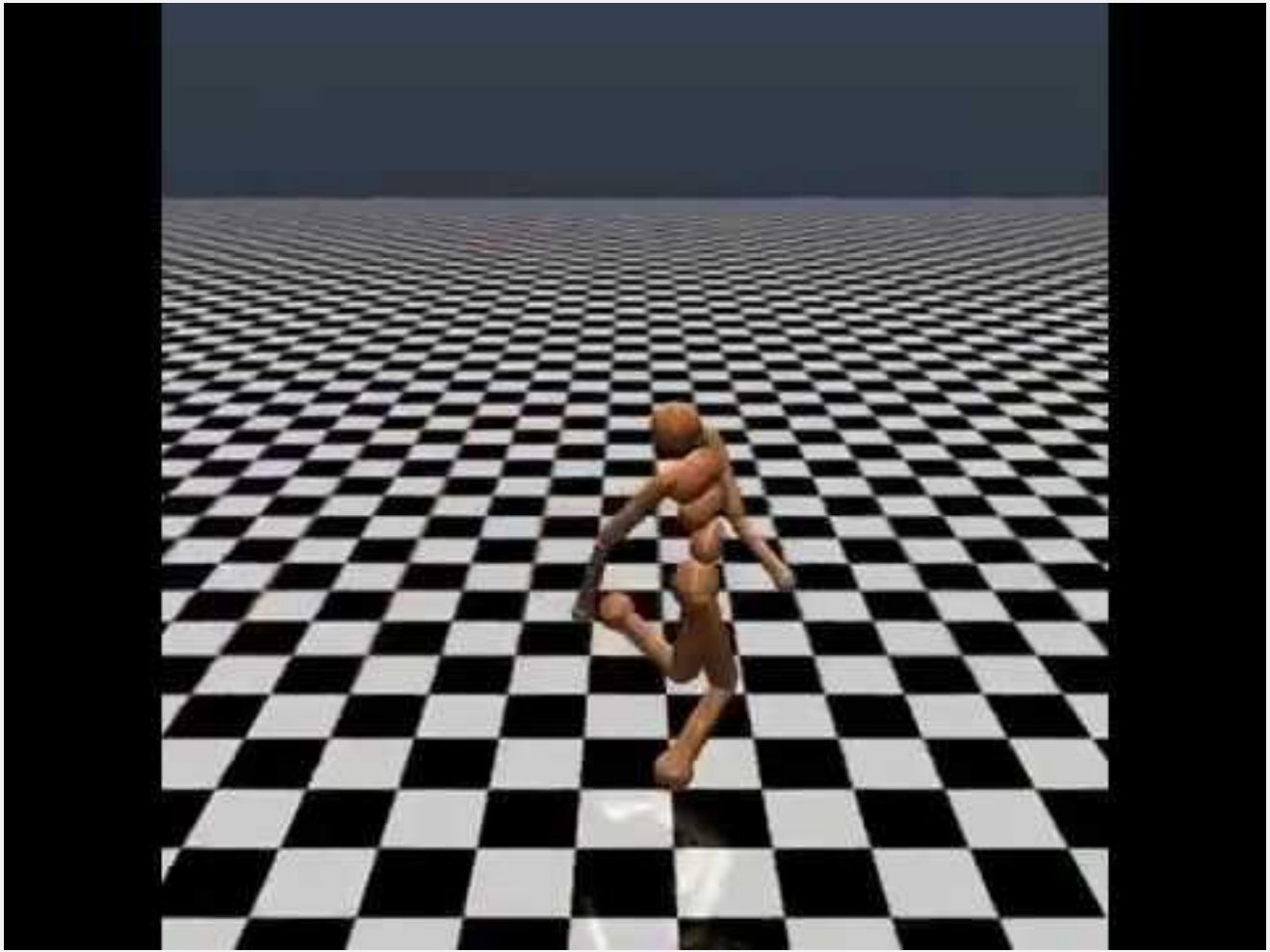
Apprentissage par renforcement appliqué

Claire Vernade
DeepMind

Apprentissage par renforcement




ima...
021 3 1



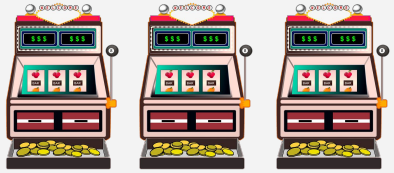
Apprentissage par renforcement ~~appliqué~~ théorique

Claire Vernade
DeepMind

Le monde des Bandits Manchots

Exploration vs. exploitation

L'objectif (simple)



L'apprentissage est séquentiel donc on discrétise le temps t=1,2,...,T

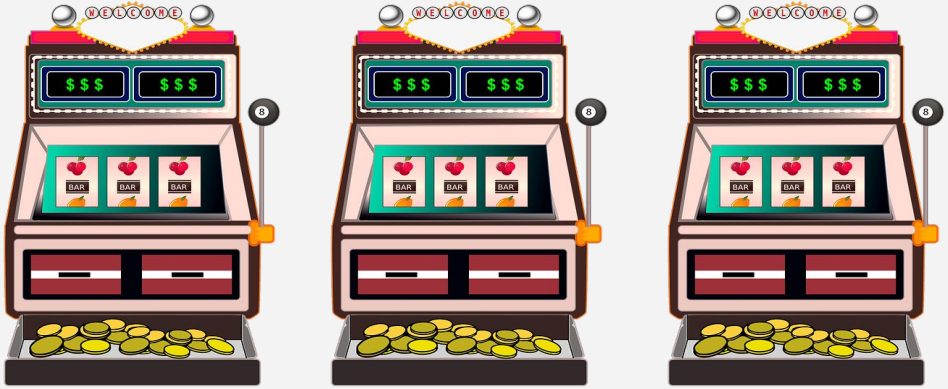
L'agent (c'est nous) choisit une action a chaque instant t : $A_t \in \{1, ..., K\}$

L'environnement envoie en retour une récompense : $r_t(A_t) \in [0, 1]$

L'objectif de l'agent est de minimiser son **Regret** apres T interactions :

$$R(T) = \sum_{t=1}^{T} r_t(a^*) - r_t(A_t)$$

Bandits manchots:
un modele du compromis exploration- exploitation



Deviations et régions de confiance

Soient X_1,...,X_n variables aleatoires independentes et identiquement distribuees telles que $\mathbb{E}[X_1] = \theta$ et $\mathrm{Var}(X) = \sigma^2$

On definit l'estimateur empirique : $\hat{\theta}(n) = \frac{\sum_{i=1}^{n} X_i}{n}$

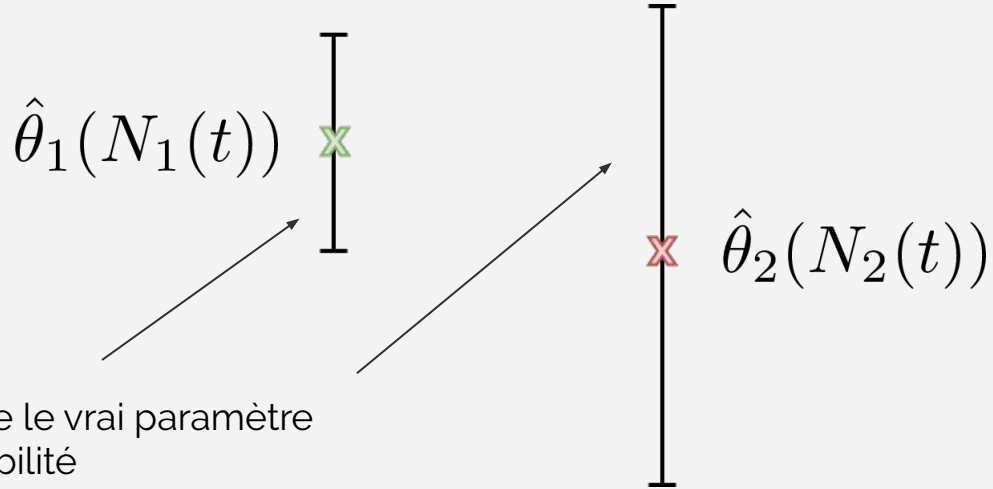
On sait que $\mathbb{E}[\hat{\theta}(n)] = \theta$ et que $\sqrt{n}(\hat{\theta}(n) - \theta) \to_{\mathcal{L}} \mathcal{N}(0, \sigma^2)$ (CLT, asymptotique)

Et plus precisement, d'apres le theoreme de Hoeffding,

$$\mathbb{P}\left(\theta > \hat{\theta} + \sigma\sqrt{\frac{2\log(1/\delta)}{n}}\right) < \delta$$

Upper Confidence Bound algorithm

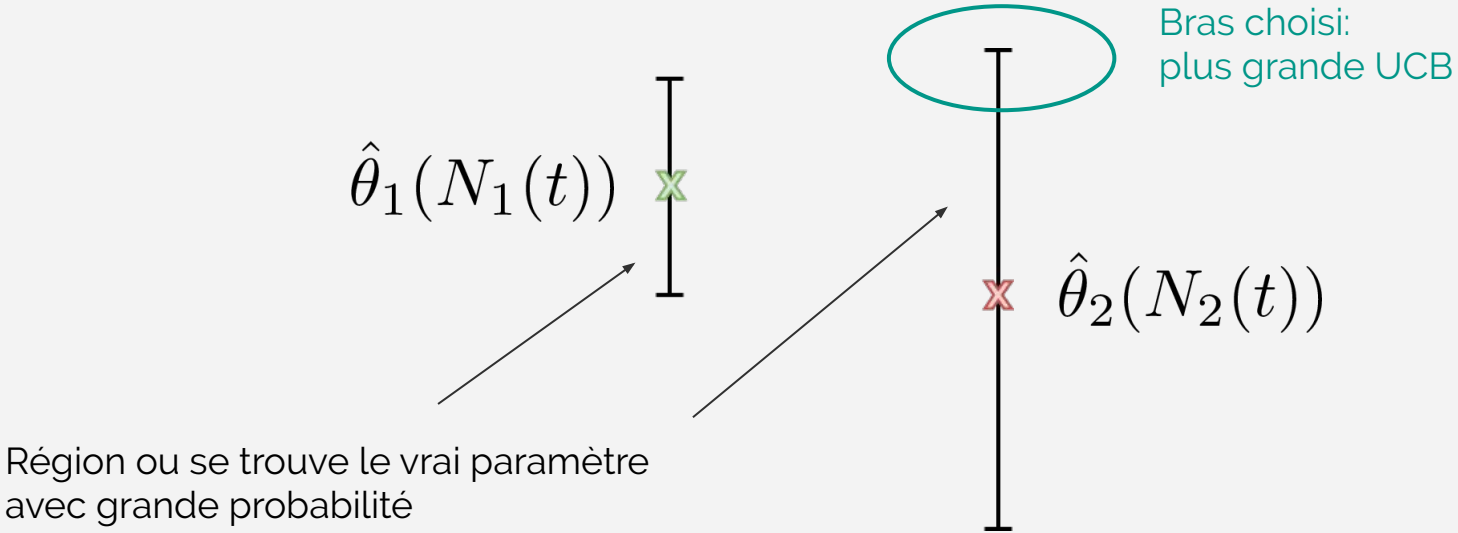
On utilise les régions de confiance de manière optimiste :



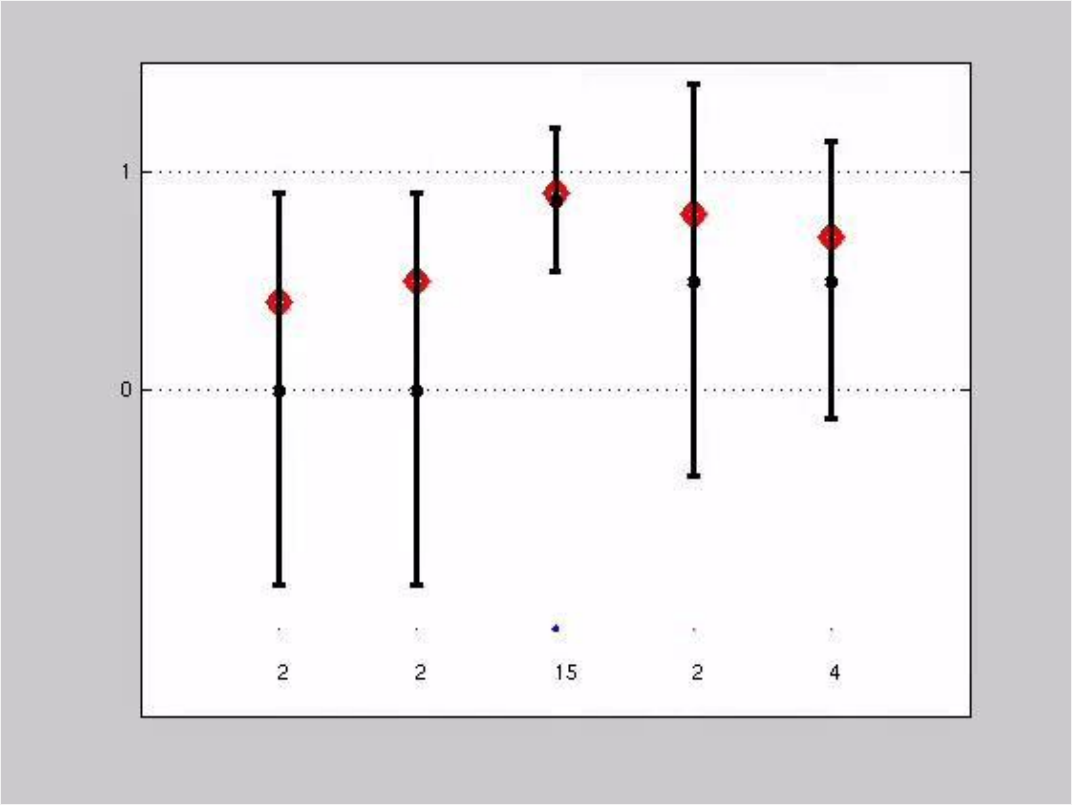
Région ou se trouve le vrai paramètre
avec grande probabilité

Upper Confidence Bound algorithm

On utilise les régions de confiance de manière optimiste :



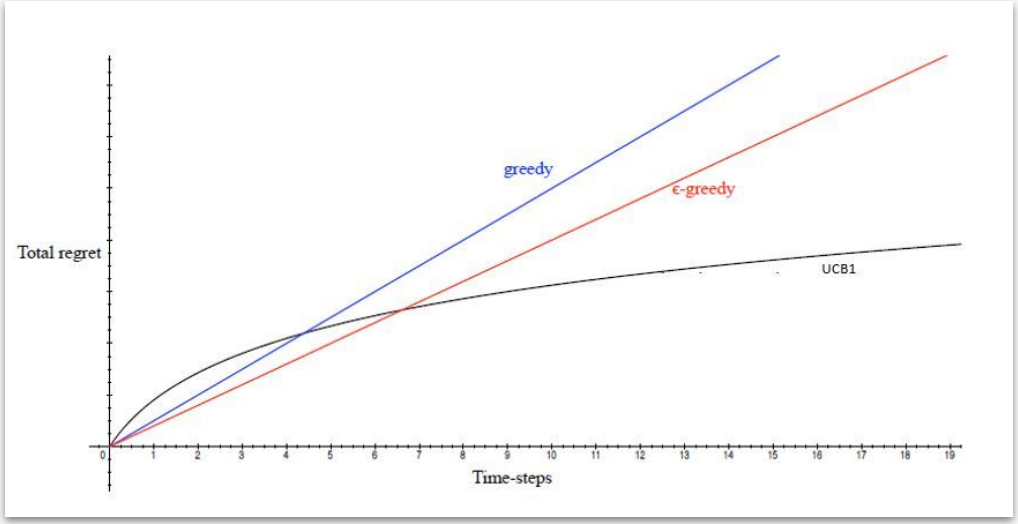
UCB à l'oeuvre



Performances

On peut proposer deux manières d'évaluer les performances de l'algorithme:

- Empiriquement : On réalise des simulations et on compare le regret moyen de UCB avec le regret d'une ou plusieurs baselines



Performances

On peut proposer deux manières d'évaluer les performances de l'algorithme:

- **Empiriquement** : On réalise des simulations et on compare le regret moyen de UCB avec le regret d'une ou plusieurs baselines
- **Théoriquement**: On analyse et on borne le regret R(T) de UCB en utilisant les propriétés de l'algorithme.

$$R(T) \leq \log(T) \sum_{a \neq a*} \frac{1}{2\Delta_a} + o(\log T)$$

Note bibliographique : Cette preuve a été faite et refaite mais on cite souvent
Auer, Peter. "Using confidence bounds for exploitation-exploration trade-offs." Journal of Machine Learning Research 3.Nov (2002): 397-422.

Conclusions

On a défini le **Regret** comme métrique à optimiser,

Et cela nous a mis face à un dilemme **exploration-exploitation**.

En faisant des hypothèses sur la distribution des récompenses conditionnellement aux actions, on a pu contrôler la validité de notre estimateur en utilisant une **inégalité de concentration**.

Cela nous a donne… **UCB** !

Question théorique: Peut-on faire “mieux” que UCB ? Quelles pourraient être les performances optimales ?

Et apres...

De nombreux problèmes de bandit ont été formulés et étudiés:

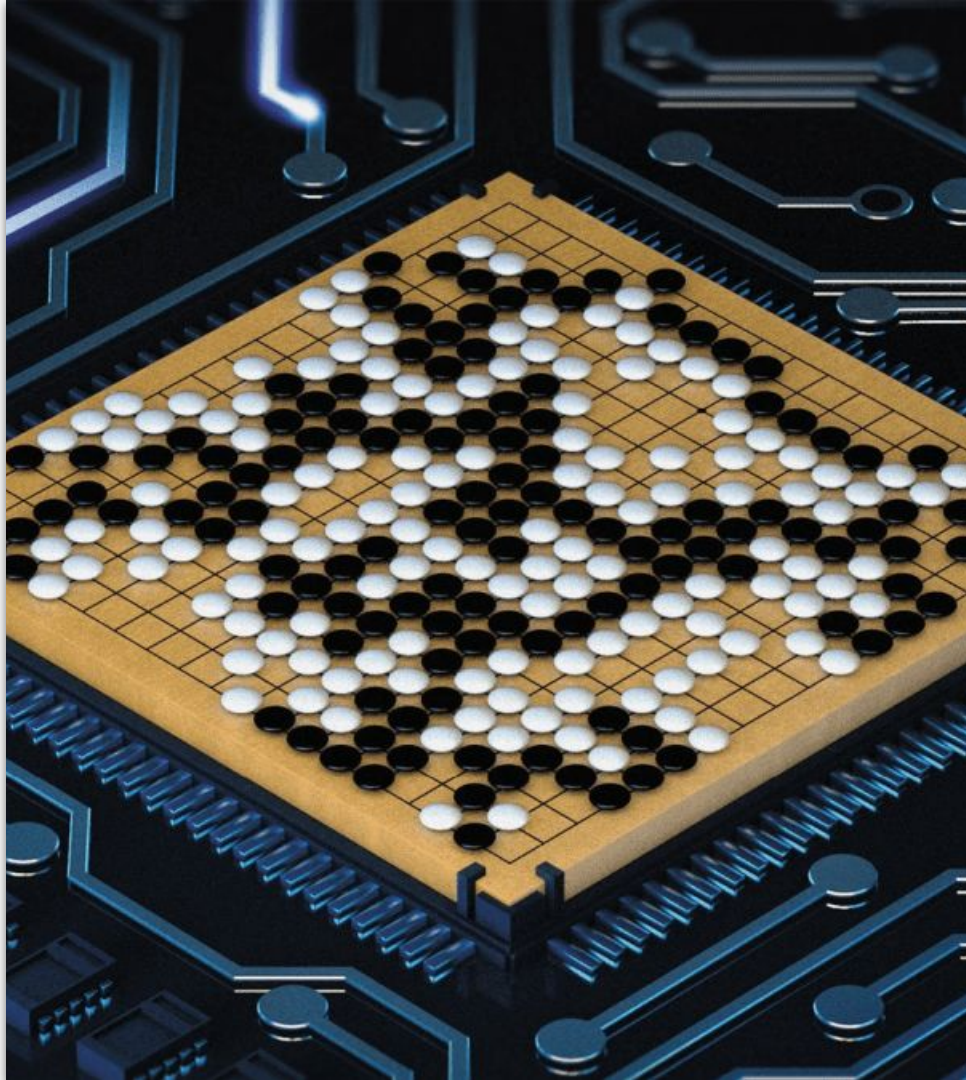
https://banditalgs.com/

Par exemple, les gens s'intéressent a des problèmes avec une infinité de bras, ou bien un espace structuré de bras, des récompenses non-stationnaires ou observees apres un delai, des problèmes avec plusieurs joueurs.

Tout ceci ne permet **pas directement** de faire de l'**Apprentissage par renforcement**.

Optimisme en apprentissage par renforcement

De la theorie a
(presque) la pratique



Markov Decision Processes

dits aussi Processus de decision markoviens

On souhaite modeliser une situation ou les actions de l'agents ont des consequences sur l'etat de l'environnement, comme dans un jeu ou pour le personnage de MuJoCo ou en general pour tout systeme dynamique.

Un MDP est defini par: Un espace d'etats, un espace d'actions,

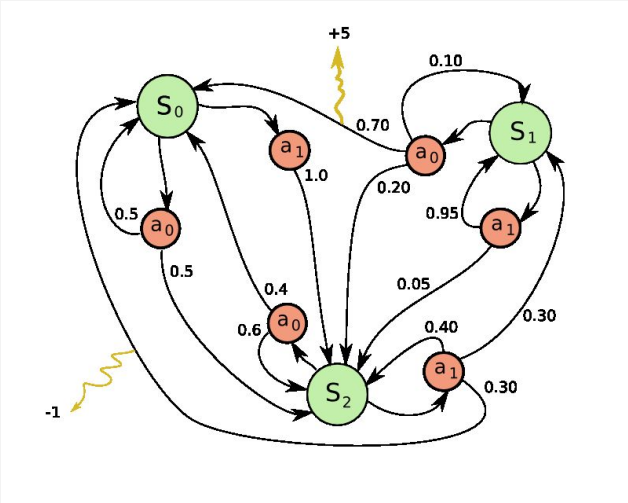
$$\mathcal{M} = (\mathcal{S},\ \mathcal{A},\ P,\ r)$$

des probabilites de transition et une fonction de recompense

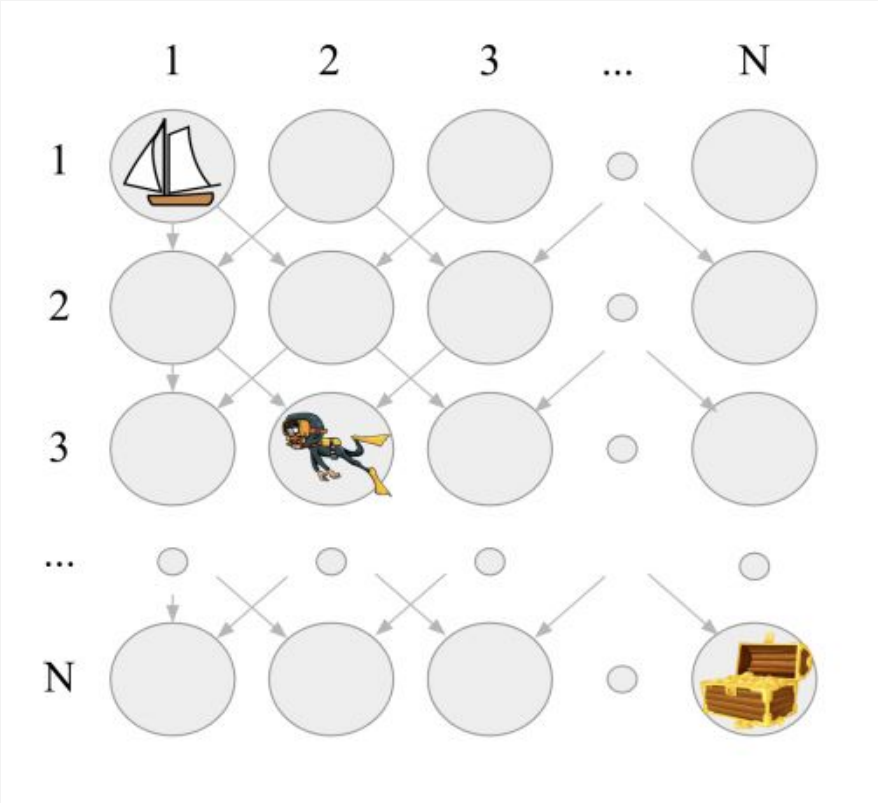
Exemple 1

La récompense +5 est obtenue en passant de S1 a S0 par l'action a0.

La "valeur" d'un etat est la moyenne des récompenses obtenues a partir de cet etat si on agit optimalement. La "valeur" d'une action dans un état est la moyenne des récompenses obtenues après avoir pris cette action puis agi optimalement.



Exemple 2: Deep Sea



UCRL : Optimisme dans un MDP

Upper Confidence Reinforcement Learning

On peut construire des estimateurs pour

- Les probabilités de transitions entre états étant donné les actions,
- La fonction de récompense pour chaque action dans chaque etat.

Pour chaque estimateur, on peut construire un intervalle de confiance.

Donc, a l'instant t, dans un état donné S, pour chaque action on peut calculer **la plus grande récompense possible dans le meilleur des mondes.**

Un problème d'optimisation linéaire sous contraintes

On sait que chaque etat a une "valeur" estimée maximale $U_t(s)$

On sait que étant donnée une action a, la probabilite de transition vers les etat voisins est $\hat{p}_t(a) \in \Delta_S$

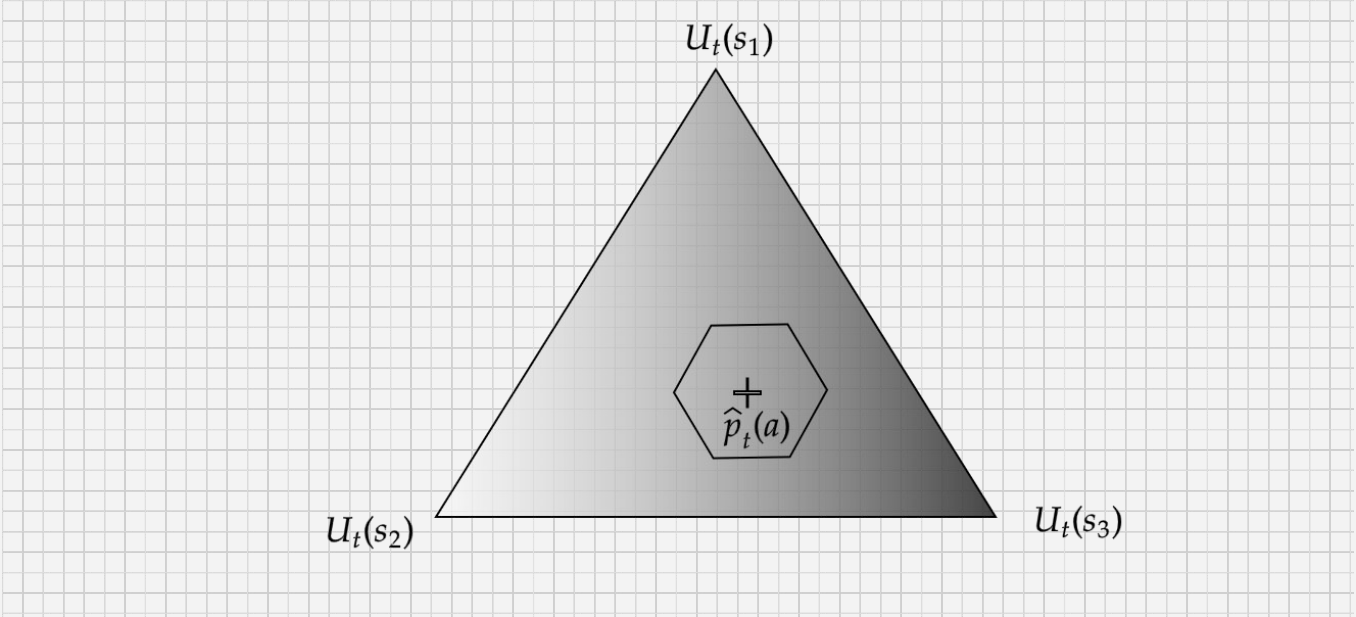
Donc en moyenne, la valeur "greedy" de l'action a est $\sum_{s \in S} \hat{p}_t(a, s) U_t(s)$

Et on cherche sa valeur "optimiste":

$$\rho_t^+(s,a) = \max_{q \in \Delta_S} \left\{ q^\top U_t : \|\hat{p}_t(a) - q\|_1 \leq \sqrt{\frac{C_{T,\delta}}{N_t(s,a)}} \right\}$$

UCRL en image: exemple avec 3 états

Pour chaque action a, on cherche le vecteur p dans la boule L1 qui maximise le produit scalaire avec U -- i.e les valeurs optimistes des etats voisins.



Conclusions -- sujets de recherche

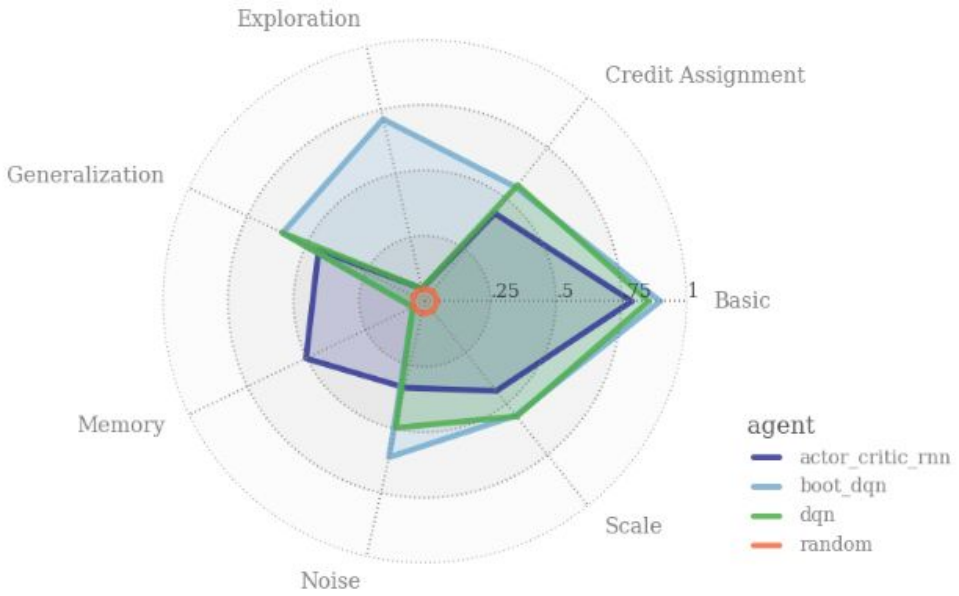
Un MDP modélise un problème dynamique: la stratégie optimale est une séquence d'actions optimales, fonction des états du système.

UCRL est un algorithme possible, pour lequel on peut prouver des garanties théoriques de type borne de regret.

Ici, on n'a pas modélise la fonction valeur, chaque état est indépendant. Mais on pourrait représenter les etats par des vecteurs, c'est l'idée de **RL avec approximation de la fonction valeur**

Il existe beaucoup d'autres algorithmes qui exploitent cette idée !

Behaviour Suite for Reinforcement Learning (`bsuite`)



Introduction

`bsuite` is a collection of carefully-designed experiments that investigate core capabilities of a reinforcement learning (RL) agent with two main objectives.

1. To collect clear, informative and scalable problems that capture key issues in the design of efficient and general learning algorithms.
2. To study agent behavior through their performance on these shared benchmarks.

This library automates evaluation and analysis of any agent on these benchmarks. It serves to facilitate reproducible, and accessible, research on the core issues in RL, and ultimately the design of superior learning algorithms.

Conclusions -- sujets de recherche

Les bandits et sujets liés en apprentissage en ligne: maximization d'influence dans des graphes, recherche Monte Carlo dans des arbres aleatoires, bandits contextuels...

Exploration en RL: Optimiser la découverte de l'espace en maximisant l'information nouvelle en chemin.

State representation et approximation de la fonction d'etat: Deep Q-Network et autres algorithmes reposant sur l'apprentissage de la représentation. Garanties theoriques manquantes.

Notes bibliographiques

CS 188 Berkeley : https://www.youtube.com/channel/UCB4_W1V-KfwpTLxH9jG1_iA

Introduction to RL by David Silver : https://www.youtube.com/playlist?list=PLqYmG7hTraZDM-OYHWgPebj2MfCFzFObQ

Algorithms to live by: The CS of Human Decisions https://g.co/kgs/iikQ6V

Librairie Bsuite : https://github.com/deepmind/bsuite

Bandit Algorithms (a.k.a. La nouvelle Bible des bandits) : https://banditalgs.com