Apprentissage par renforcement

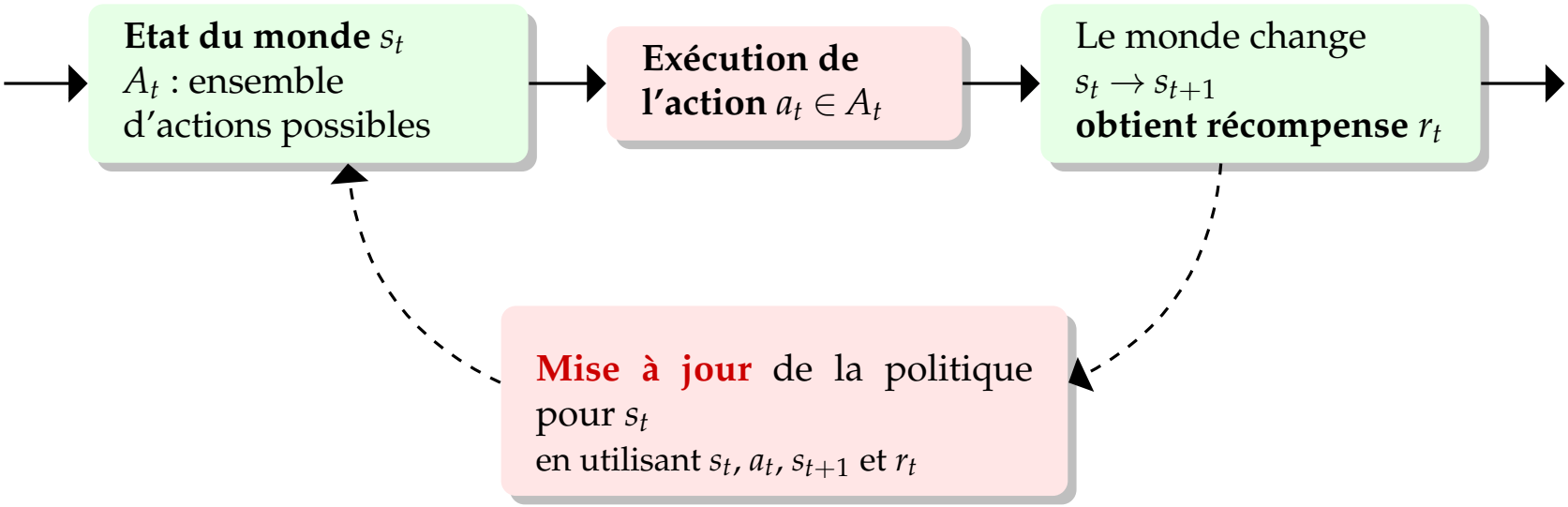
Stéphane Airiau

Université Paris Dauphine

Un type d'apprentissage : apprentissage par renforcement

- apprentissage **supervisé** :
  les données contiennent des observations dont la *réponse*
  - pour les arbres de décision une *classification*
  - pour les réseaux de neurones on nous donne la *valeur de la fonction objective*
- apprentissage **non-supervisé**
  On nous donne des données, on cherche à trouver des similitudes
  ex : clustering
- apprentissage **par renforcement** Un type d'apprentissage qui est basé sur l'**interaction avec son environnement**.
  Après chaque action exécutée dans un environnement, on reçoit un *signal* dont l'intensité indique si on agit bien ou non
  - si on reçoit un encouragement, on fait quelque chose de bien et on persevère
  - si on reçoit une punition, on a fait quelque chose de mal, et se doit de changer quelque chose
  - initialement, on ne sait pas forcément dicerner un encouragement d'une punition ! 0.7, est-ce ou non bien ?

Apprentissage par renforcement



- **But** : accumuler le plus de récompenses
- **Apprentissage par renforcement :** comment spécifier la fonction de mise à jour ?

Apprentissage par renforcement

⇨ dans ce cours, on va étudier une approche computationnelle pour apprendre de son interaction.

Le but de l'apprentissage par renforcement est de savoir quoi faire dans un contexte donné pour maximiser un signal récompense.

- l'apprenti doit découvrir par lui-même les actions qui donnent les plus hautes récompenses en les essayant !
- ces choix peuvent avoir des conséquences sur la récompense directement obtenue, mais aussi sur la situation suivante.

Récompenses

- **But** choisir une action qui maximise la récompense future
- les actions peuvent avoir des conséquences *à long terme*
- l'obtention de la récompense peut avoir un délais
  ⇨ il peut être bon de sacrifier une bonne récompense immédiate pour obtenir une meilleure récompense sur le long terme !

Un type d'apprentissage : apprentissage par renforcement

- la récompense n'est pas donnée de manière instantanée (délais)
  - ⇨ peut être il est difficile de savoir exactement ce qu'on a **bien** fait !)
- ⇨ en accumulant de l'expérience, on va apprendre à modifier le comportement pour optimiser a récompense
- le temps est donc important (les données ne sont pas i.i.d !)
- L'apprentissage par renforcement travaille dans un monde incertain
  - le résulat d'une action ne peut pas être entièrement prédit
  - ⇨ un agent doit surveiller son environnement et réagir à tout changement

Exemple d'application

On peut utiliser l'apprentissage par renforcement pour résoudre des problèmes ou l'objectif est d'optimiser une fonction cumulative.

- optimisation dans des usines, régulation du traffic aérien (simulations)
- ordonnancement (ex : quel taxi choisir pour une course)
- jouer à des jeux (ex. les vieux jeux Atari, mieux que les meilleurs humains)
  - en 1995 Tesauro (IBM Watson) à utilisé l'apprentissage par renforcement pour son programme TD-Gammon qui joue au backgammon au niveau des meilleurs mondiaux
  - avec les réseaux de neurones, l'apprentissage par renforcement est utilisé par AlphaGo pour jouer au jeu de Go et gagner contre le meilleur joueur humain
- faire marcher un robot humanoid
- publicité en ligne, campagne essais cliniques

Plan du cours

1. Introduction et bandits manchots.
2. Processus de décision Markovien (PDM) et modélisation.
3. Différents algorithmes pour résoudre un PDM :
   programmation dynamique, méthodes de Monte Carlo, méthodes de différences temporelles
4. Apprentissage par renforcement utilisant approximation
   (⇨ utiliser des méthodes d'apprentissage supervisé)
   Deep Q-learning, policy gradient

Evaluation du cours

(tentative)

- 20% CC – tester les algorithmes avec implémentation.
- 80% examen

Mise en bouche : le problème des k-bandits manchots

k bandits manchots $\Leftrightarrow$ k machines à sous (en anglais *k-armed bandit problem*)

Supposons qu'on ait à notre disposition un saut de 1000 jetons pour jouer aux machines à sous.

Supposons qu'il y a k machines à sous, et qu'elles ont des propriétés différentes sur les gains.

On suppose que chaque le fonctionnement machine est indépendente de chacunde des autres.

Quelle est notre politique pour utiliser nos 1000 jetons ?

Autres applications des bandits

- Essais cliniques :
  on a k molécules pour un symptôme et l'efficacité est incertaine.
  ⇨ étant donnée la réponse sur les précédents patients ayant le même symptôme, quel traitement est à proposer ?
- Publicité en ligne (€€€) k publicités peuvent être affichées
  Laquelle doit-on afficher pour un utilisateur, basé sur le comportement d'autres utilisateurs (similaires) ?
- dans des jeux (ex Go), on a k coups possibles, lequel choisir pour nous mener à la victoire ?

problème des k-bandits manchots

On note A_t la machine choisie pour jouer le t^{ième} jeton et R_t la récompense obtenue.

On suppose que chaque machine a utilise une distribution de probabilité de moyenne $q_\star(a) = \mathbb{E}[R_t \mid A_t = a]$

Si on connaissait $q_\star(a)$ pour chaque machine, le problème serait trivial

⇨ on passe notre temps à jouer $\operatorname{argmax}_a q_\star(a)$!

Mais on peut former une estimation.

Supposons qu'on ait une estimation pour chaque machine

- vous jouez la machine qui a la meilleure estimation
  ⇨ "action *gloutonne*"
  ⇨ on exploite sa connaissance actuelle.
- Vous ne jouez pas pas l'action gloutonne ⇨ exploration.
- Durant l'exploration, la récompense est moins bonne à court terme. Mais elle peut permettre de plus grands gains à long terme !

⇨ il faut trouver *un équilibre* entre l'exploitation et l'exploration.

- ce problème a été étudié en mathématique, et on peut construire des méthodes sophistiquées pour trouver le meilleur équilibre, mais cela nécessite des hypothèses précises sur les machines.

Une estimation simple

On va utiliser la moyenne des gains obtenus jusqu'au moment présent.

$$
\begin{aligned}
Q_t(a) &= \frac{\text{Somme des récompenses obtenues en jouant } a}{\text{nombre de fois où on a joué } a} \\
&= \frac{\sum_{i=1}^{t-1} R_i \mathbf{1}_{\mathbf{A_i}=\mathbf{a}}}{\sum_{i=1}^{t-1} \mathbf{1}_{\mathbf{A_i}=\mathbf{a}}}, \text{ où } \mathbf{1}_{\mathbf{cond}} \begin{cases} 1 \text{ si cond est vraie} \\ 0 \text{ sinon} \end{cases}
\end{aligned}
$$

en utilisant la loi des grands nombres, $Q_t(a)$ va converger vers $q_\star(a)$.

$$\lim_{t \to +\infty} Q_t(a) = q_\star(a)$$

Action gloutonne et exploration ϵ-gloutonne

Une action gloutonne est donc

$$\operatorname*{argmax}_a Q_t(a)$$

ϵ-glouton : une variante est d'être glouton presque tout le temps, et de temps en temps, d'explorer aléatoirement

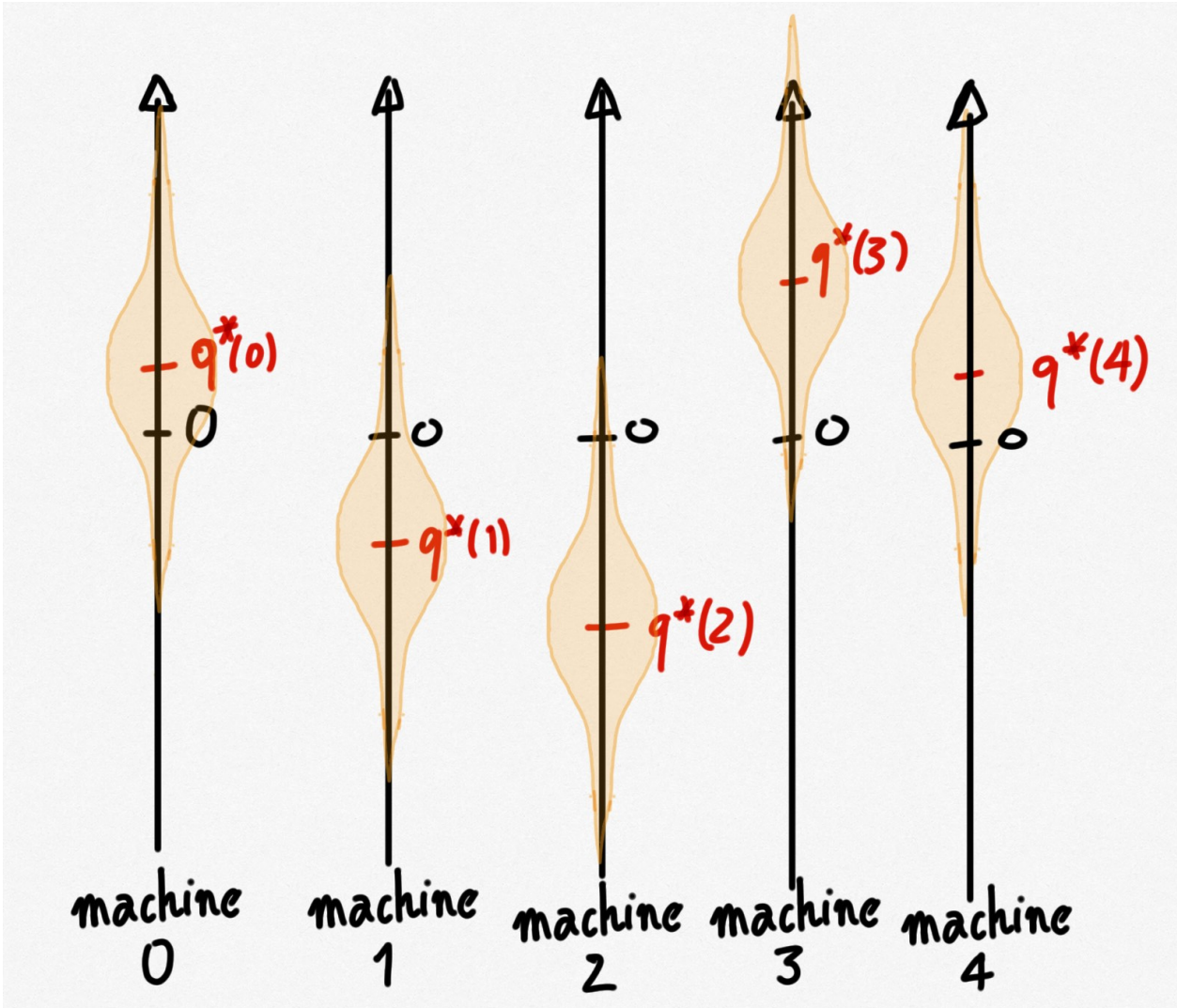
- avec une probabilité ϵ, on choisit une action au hasard (en utilisant une probabilité uniforme sur les machines)
- avec une probabilité $1 - \epsilon$, on choisit l'action gloutonne

Evaluation empirique

On utilise une simulation avec 10 machines à sous.

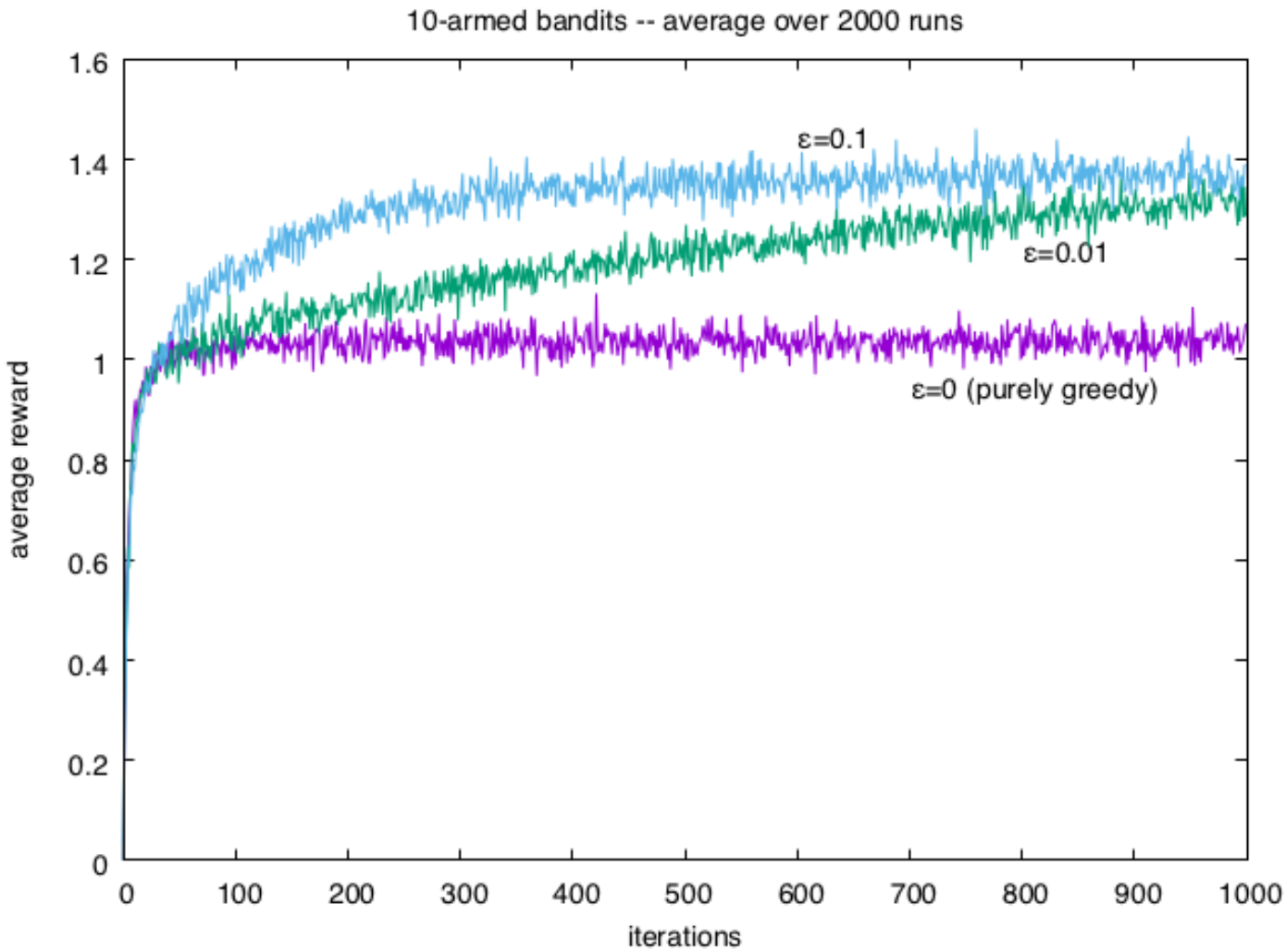
- Pour chaque machine a, on génère $q_\star(a)$ en tirant un échantillon d'une distribution normale de moyenne 0 et de variance 1 ($\mathcal{N}(0,1)$).
- à chaque fois qu'on joue à la machine a, on tire un échantillon en suivant une loi normale de moyenne $q_\star(a)$ et de variance 1.
- on utilise un algorithme en choisissant 1000 fois une machine.
  ⇨ un "run"
- pour obtenir des informations plus fiables, on répète 2000 "runs" (on tire au sort les valeurs $q_\star$, puis on choisit 1000 fois une machine)

Simulation



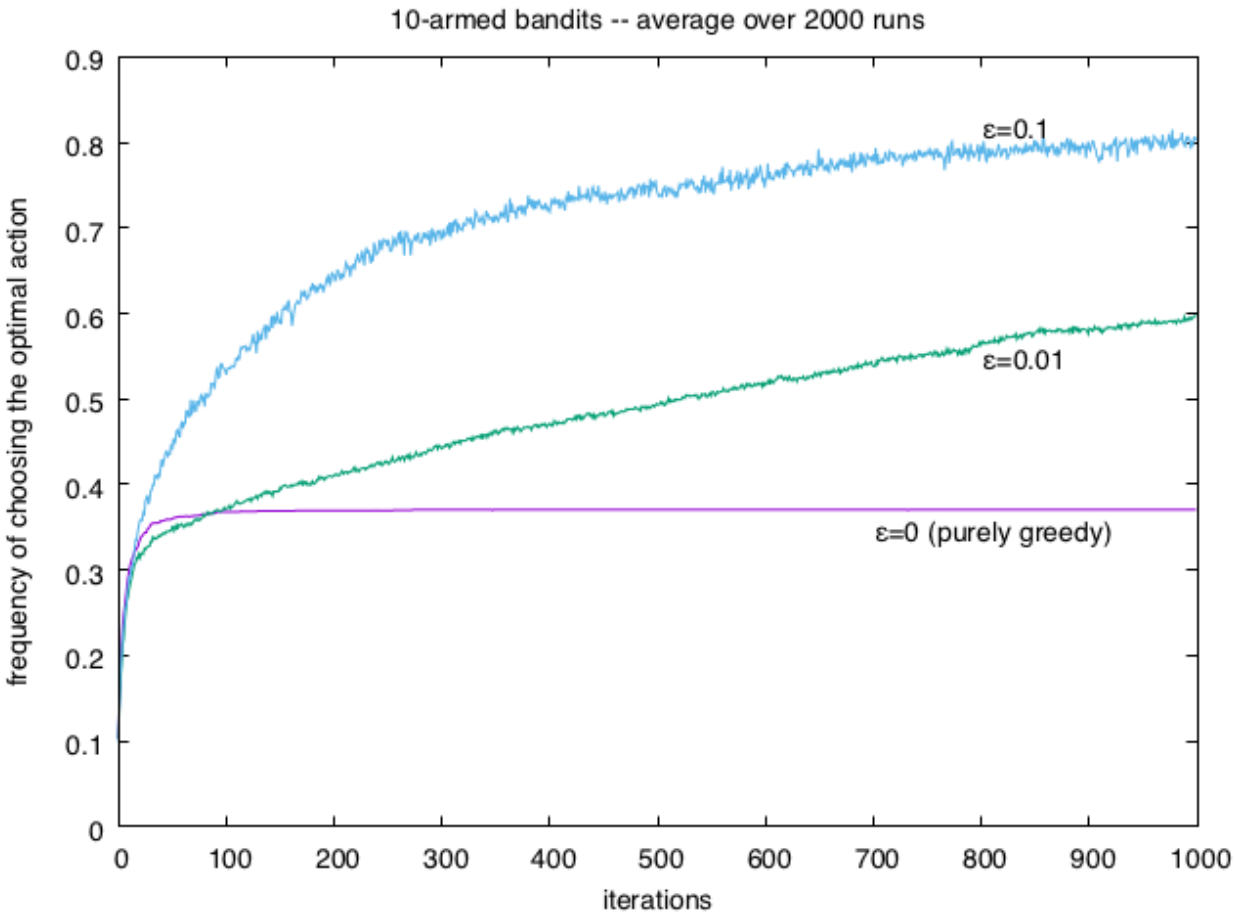
Exemple d'*une* expérience avec seulement 5 machines
Ici, la meilleure machine est 3, la moins bonne est 2.

Evaluation empirique : Glouton, ϵ-glouton



récompenses en moyenne

Evaluation empirique : Glouton, ϵ-glouton



fréquence du choix de la meilleure machine

Evaluation empirique : Glouton, ϵ-glouton

- la méthode glouton se "bloque" trop vite sur une machine non optimale.
- 0.1-glouton explore assez rapidement, mais va plafonner à la limite (en théorie, devrait atteindre 91% de bon choix)
- 0.01-glouton explore plus lentement au début, mais avec plus d'itérations, devrait atteindre un meilleur résultat.
- en regardant ce graphe, et si vous avez 1000 jetons, quelle stratégie utilisez vous ?

Implémentation efficace

On tient à calculer l'estimation de la valeur des machines de manière computationnellement efficace !

- on veut éviter de stocker toutes les données
- on veut éviter de faire une boucle pour calculer la moyenne

C'est trivial, mais c'est une préoccupation à avoir !

Implémentation efficace

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}\left(R_n + \sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)\frac{1}{n-1}\sum_{i=1}^{n-1} R_i\right) \\
&= \frac{1}{n}\left(R_n + (n-1)Q_n\right) \\
&= \frac{1}{n}\left(R_n + nQ_n - Q_n\right) \\
&= Q_n + \frac{1}{n}\left(R_n + -Q_n\right)
\end{aligned}
$$

Dans ce cours, on va beaucoup utiliser une règle de mise à jour

$$estimate_{new} \leftarrow estimate_{old} + \alpha \cdot (observation - estimate_{old})$$

Mise à jour

$$estimation_{new} \leftarrow estimation_{old} + \alpha \cdot (observation - estimation_{old})$$

Etant donnée la nouvelle observation

- s'il semble que notre estimation est surestimée
  ⇨ on baisse notre estimation
- s'il sembe que notre estimation est sousestimée
  ⇨ on augmente notre estimation
- $(observation - estimate_{old})$: estimation de l'erreur
- la réduction ou l'augmentation se fait proportionnellement à l'erreur, avec un coéfficient α (généralement petit).
- α peut dépendre du nombre de mises à jour effectuées
  dans notre exemple $\alpha = \frac{1}{n}$ où n est le nombre de fois qu'on a choisi une machine en particulier.

Et si le réglage des machines à sous varie au cours du temps ?

Le problème est **non-stationnaire** et dépend donc du temps.
Idée : on peut donner un poids plus *lourd* aux récompenses *récentes* par rapport aux récompenses obtenues il y a longtemps.
Par exemple :

$$Q_{n+1} = Q_n + \alpha\,[R_n - Q_n]$$

où le paramètre $\alpha \in]0,1]$.

$$\begin{aligned}
Q_{n+1} &= Q_n + \alpha\,[R_n - Q_n] \\
&= \alpha R_n + (1-\alpha) Q_n \\
&= \alpha R_n + (1-\alpha)\,[\alpha R_{n-1} + (1-\alpha) Q_{n-1}] \\
&= \alpha R_n + \alpha(1-\alpha) R_{n-1} + (1-\alpha)^2 Q_{n-1} \\
&= \alpha R_n + \alpha(1-\alpha) R_{n-1} + (1-\alpha)^2\,[\alpha R_{n-2} + (1-\alpha) Q_{n-2}] \\
&= \dots \\
&= (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i
\end{aligned}$$

Et si le réglage des machines à sous varie au cours du temps ?

$$Q_{n+1} = (1-\alpha)^n Q_1 + \sum_{i=1}^{n} \alpha(1-\alpha)^{n-i} R_i$$

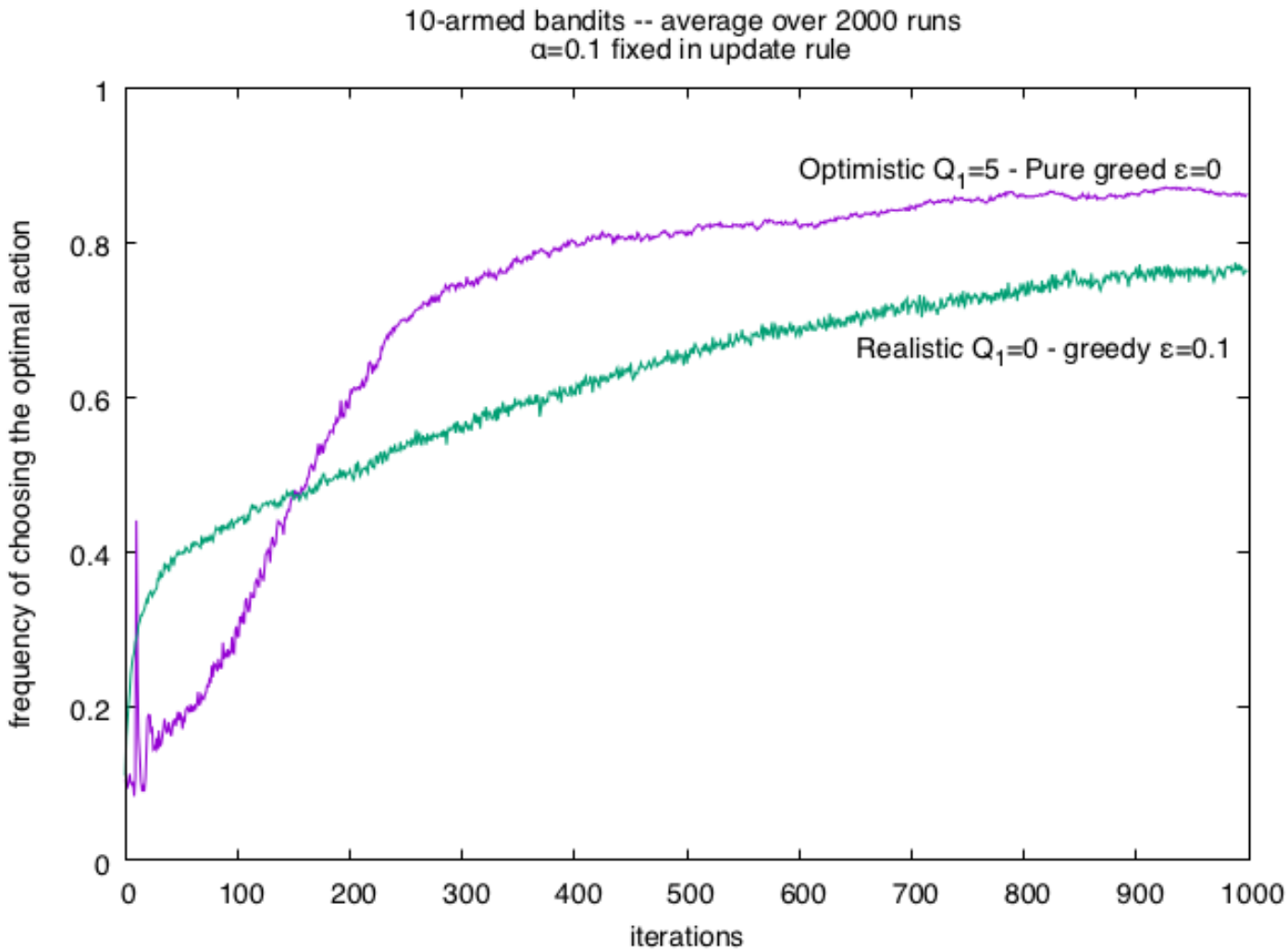
- c'est une moyenne pondérée
  on peut vérifier que la somme des poids est égale à 1
- $0 < \alpha \leqslant 1$ donc on a une chute exponentielle du poids en fonction de l'ancienneté.

Initialisation de l'estimation

Quelle valeur pour $Q_1(a)$?

- le choix de $Q_1(a)$ peut introduire un biais
- pas un problème en soi, et peut même être utile ⇨ on peut *forcer* l'exploration
- **idée** : initialisation optimiste des valeurs initiales.
- ⇨avec nos méthodes gloutonnes, on va explorer ces actions qui apparaissent prometteuses
  - si ce ne sont pas de bonnes actions ⇨ l'estimation va vite baisser
  - on va forcer que chaque action soit explorée un certains nombre de fois, ce qui va aider la convergence !

Evaluation empirique



C'est une ruse qui fonctionne généralement bien pour les problèmes stationnaires.

Utiliser un intervalle de confiance

- On a besoin d'explorer
- ϵ-glouton traite les actions non-gloutonne de la même manière.

Cependant :

- Certaines actions ont peut-être été utilisées plus souvent.
- On n'a peut être pas intérêt à tester plusieurs fois une action vraiment mauvaise.
- Il faut peut être tester plus vite des actions qui paraissent plus prometteuses.

⇨ confiance de l'estimation

⇨ Méthode des intervalles de confiance

Utiliser un intervalle de confiance

- idée : essayer de mesurer l'incertitude sur la valeur Q(a) qu'on estime
- but : on calcule une borne supérieure probable de la valeur Q(a) qu'on estime
- La borne supérieure est calculée par l'expression

$$Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

- plus la borne est élevée, plus a est prometteuse !

On va donc choisir l'action $\operatorname{argmax}_a \left[Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}} \right]$

⇨ On explore les actions les plus prometteuses
⇨ meilleure utilisation des échantillons.

Utiliser un intervalle de confiance

$$Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}$$

- $N(a)$ est le nombre de fois où l'action a a été choisie
- c est une constante qui détermine la mesure de confiance
- t est le nombre total d'itérations

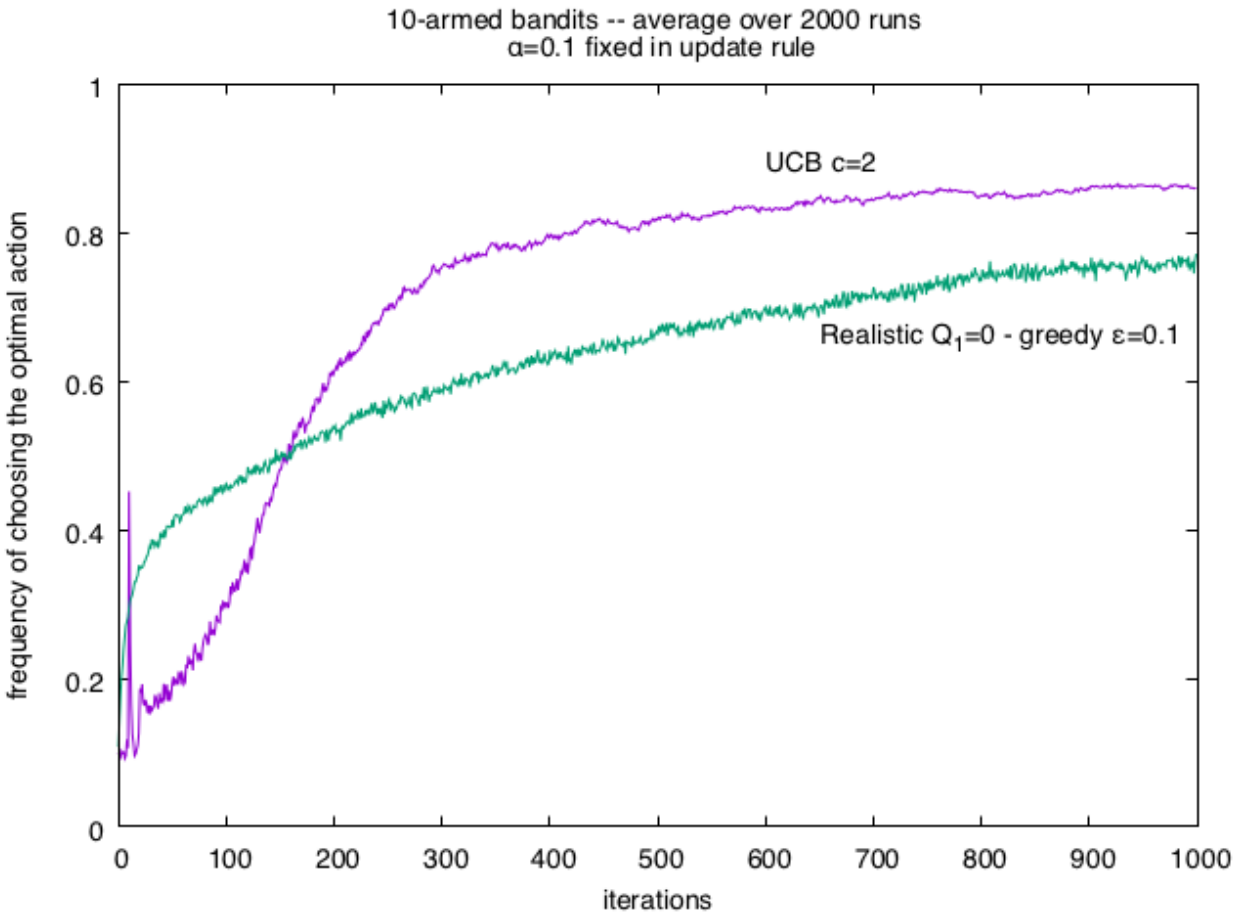
⇨ si a est choisie ⇨ l'incertitude baisse (donc la borne aussi)

⇨ si a n'est pas choisie, t croît, et l'incertitude croît.

- avec le log, l'accroissement devient de plus en plus petit.

Peter Auer, Nicolò Cesa-Bianchi, Paul Fischer. (2002). Finite-time Analysis of the Multiarmed Bandit Problem, *Machine Learning*, 47, 235—256

Evaluation empirique



fréquence du choix de la meilleure machine

Utiliser un intervalle de confiance

Marche avec des problèmes de bandits stationnaires.

Personne n'a encore réussi à adapter ce type d'idée pour les modèles plus compliqués d'apprentissage par renforcement que l'on va voir dans le cours.

certains d'entre vous on peut-être vu son utilisation dans les jeux à deux joueurs (Monte Carlo et UCB ont donné de bons résultats).

Montée de gradient stochastique

- jusqu'ici : on apprend une estimation, puis on utilise cette estimation pour choisir une action
- maintenant, on veut apprendre une fonction de préférences : plus on préfère une action, plus souvent on va la prendre.
  On note $H_t(a)$ la préférence pour l'action a au pas de temps t.

⇨ idée de distribution de Boltzmann

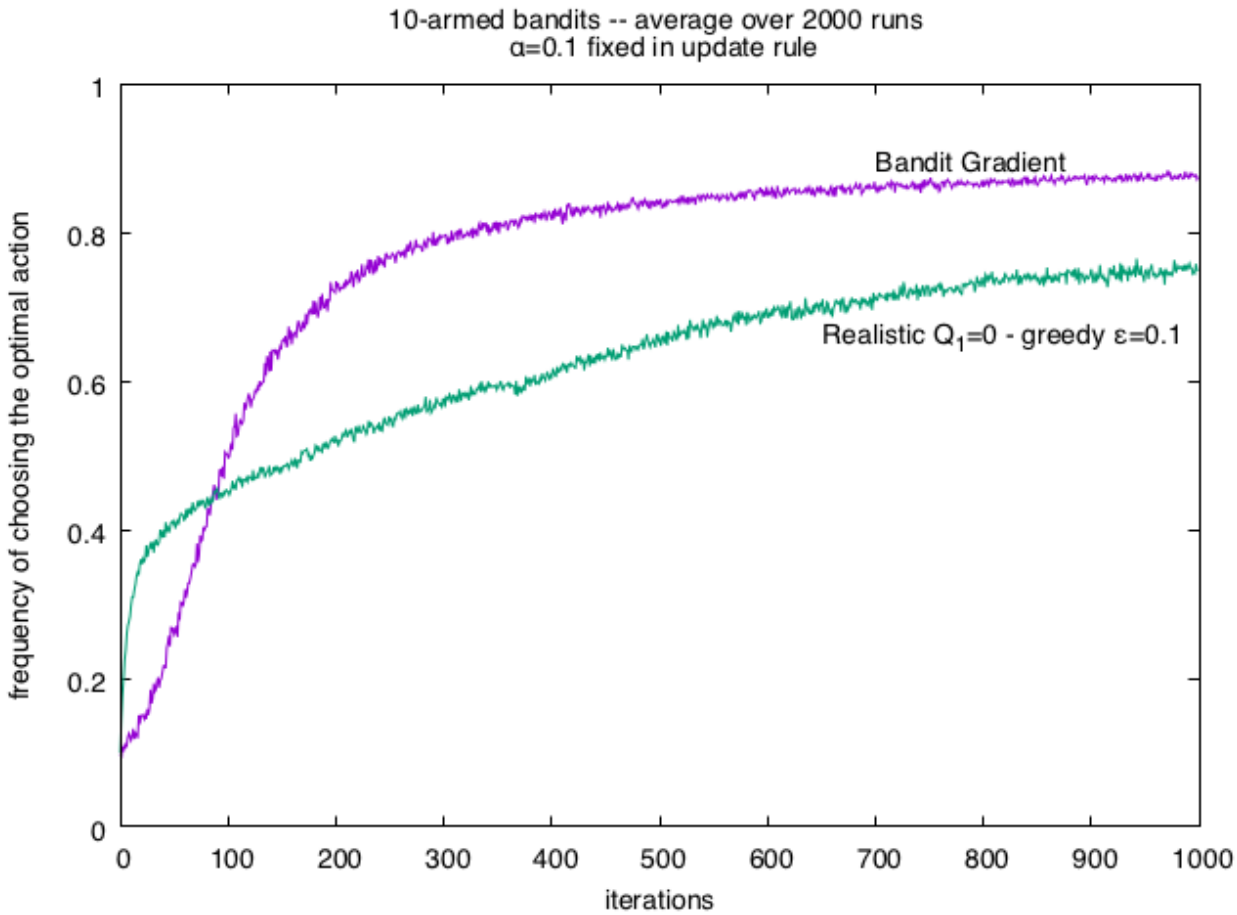
$$Pr\{A_t = a\} = \frac{e^{H_t(a)}}{\sum_{b=1}^{10} e^{H_t(b)}} = \pi_t(a)$$

- On peut utiliser l'idée de la montée de gradient stochastique.

$$H_{t+1}(a) = \begin{cases} H_t(a) + \alpha(R_t - \bar{R}_t)(1 - \pi_t(a)) & \text{if } a = A_t \\ H_t(a) - \alpha(R_t - \bar{R}_t)\pi_t(a) & \text{otherwise} \end{cases}$$

où $\bar{R}_t$ est la moyenne des récompenses jusqu'au pas de temps t, comme calculé avant.

Montée de gradient stochastique



Conclusion

- On a vu une tâche "dégénérée" d'apprentissage par renforcement : on doit choisir, grâce à notre expérience, un bandit manchot parmi n.
- On a vu certains algorithmes plus ou moins simples qui peuvent nous aider.
- Evidemment, avec plus d'information, on pourrait avoir une stratégie encore meilleure, voir optimale (ex : si on connait la distribution pour choisir $q^\star$, si on exploitait le fait qu'ensuite chaque machine suit une loi normale, etc...).
- Dans la suite du cours, on va généraliser ces idées : l'outil de base sera un processus décisionnel markovien (PDM) dans lequel on aura des états, et on passera d'un état à un autre en choisissant une action. L'état d'arrivée est stochastique et dépend de l'action et de l'état précédent.
- Dans le problème des bandits, on peut considérer que c'est un PDM avec un seul état.