Apprentissage par renforcement
Optimisation en univers incertain

Richard Lassaigne
IMJ/Logique mathématique
CNRS-Université Paris Diderot

• Ensemble de techniques et d'outils permettant à un **ordinateur** de réaliser une tâche **simple** qu'un humain pourrait faire facilement comme **reconnaître** des caractères écrits, des images,...

• Une des directions récentes : avoir à sa disposition un ensemble de **données** correctement **traitées** et créer une règle d'étiquetage automatique de données **nouvelles**

• L'apprentissage **humain** ne fonctionne pas de cette manière mais plutôt selon un système d'**essais-erreurs**
L'enfant procède, sans le savoir, par apprentissage **séquentiel**

• Ce concept d'apprentissage séquentiel, où les données sont traitées **à la volée**, a été modélisé depuis longtemps
La première **modélisation** est attribuée à Thompson, dans les années 30 pour traiter des **essais cliniques**

• Un modèle pour la conception **séquentielle** d'expériences
Problème de la répétition d'un **choix** parmi différentes actions
Chaque action donne lieu à une **récompense** donnée par une distribution de probabilités dépendant de l'action choisie

• Objectif : construire une stratégie permettant de **maximiser** la récompense totale **espérée** sur une période de temps donnée
Mesure de **performance** : différence entre le paiement cumulé obtenu à l'aide d'une machine **optimale** et le paiement effectif

• Difficulté : comment réaliser un compromis entre acquisition et utilisation de l'information (**exploration** ou **exploitation** ?)
Les stratégies classiques reposent sur l'idée de **renforcement**
Analyse délicate : trouver la proportion minimale de temps sur les machines **sous-optimales** par rapport à l'exploration

• Progrès récents de l'apprentissage par **renforcement**
Voir le tutoriel de D. Silver sur **Deep** Reinforcement Learning
Programmes AlphaGo et AlphaGo Zero (DeepMind)

• Apprentissage par **interaction** avec l'**environnement**
Amélioration du comportement d'un preneur de décision

• Le cadre général est similaire à celui de l'**optimisation**
en univers **incertain** (processus de décision markoviens)

• Objectifs différents : **prédiction** vs **optimisation**
Les fonctions d'**évaluation** des stratégies sont les mêmes

• Problème de la grande dimension (en pratique) :
pour des raisons de **complexité**, on utilise les mêmes
classes de méthodes (**approximation**, **Monte-Carlo**,...)

Reinforcement learning is learning what to do
–how to map situations to actions–so as to maximize
a numerical **reward** signal in an **unknown uncertain**
environment. The learner is not to told which actions to take,
as in most forms of machine learning, but must discover which
actions yield the most **reward** by trying them. In the most
interesting and challenging cases, actions may affect not only the
immediate reward but also the **next situation** and, through that,
all **subsequent rewards**. These two characteristics
–**trial-and-error** search and **delayed reward**–
are the most important distinguishing features of
reinforcement learning.

"An introduction to reinforcement learning"
Sutton and Barto (1998, 2017)

- Apprentissage **supervisé**
  Entraînement à partir d'ensembles d'exemples étiquetés
  Objectif : apprendre à identifier une **catégorie** de situations
  pour **régression** (numérique)
  ou **classification** (non numérique)
  Méthodes : régression, arbres de décision,
  apprentissage **profond** (réseaux de neurones convolutifs),
  régression logistique, arbres de classification, SVM

- Apprentissage **non supervisé**
  Objectif : découvrir des **structures** cachées
  dans des collections de données non étiquetées
  Analyse en **composantes principales**
  Méthodes de **regroupement** : K-Means, regroupement
  hiérarchique, regroupement pour régions denses

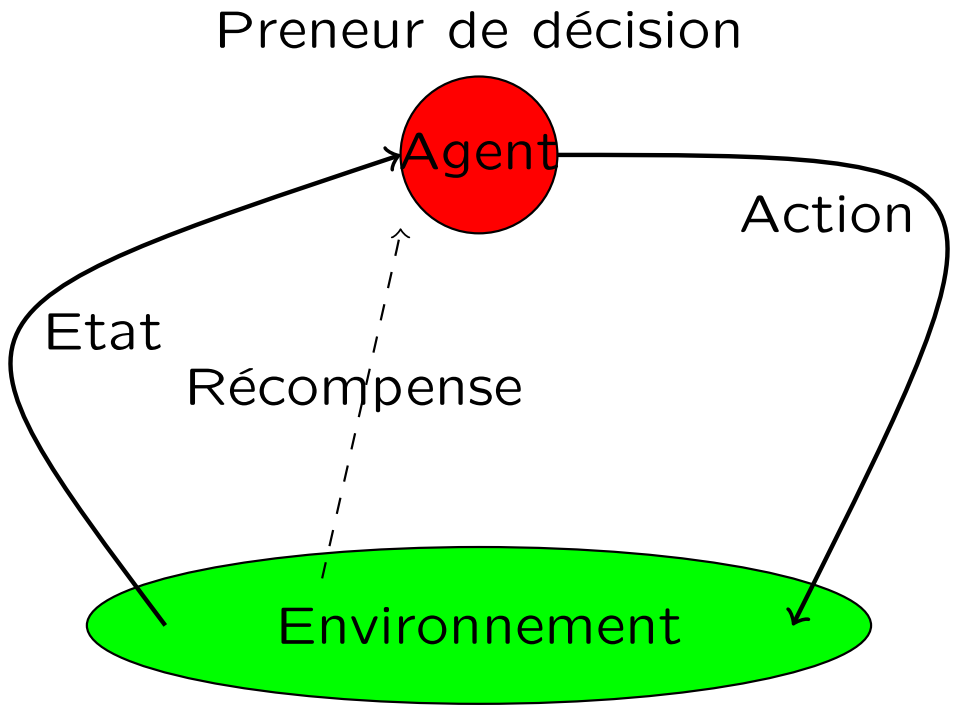
- Apprentissage par **renforcement**
  Objectifs et **challenges**

- Les problèmes de **bandits** à plusieurs bras
  Optimisme face à l'incertain. Algorithmes UCB et EXP3

- Optimisation dans les processus de décision **markoviens**
  Itérations sur la fonction **valeur**, sur la **stratégie**

- Les méthodes classiques d'**optimisation** sont impraticables
  car **polynomiales** dans la taille du **modèle**
  La phase de modélisation conduit à l'explosion **combinatoire**

- Méthodes d'**exploration** et d'**exploitation**
  Algorithmes UCT et AMS. Monte-Carlo **Tree Search**



- **Environnement** : peut être **stochastique**, **adversaire**, partiellement **inconnu**, partiellement **observable**
- **Information disponible** : le **renforcement** (par récompense)
- **Objectif** : maximiser la **somme espérée** des récompenses

  Problème :

  comment sacrifier de petites récompenses à **court terme**

  pour privilégier des récompenses plus grandes à **long terme** ?

Raisonnement dans l'incertain

Apprendre le modèle	Problèmes de **bandits** à plusieurs bras	Apprentissage par **renforcement**
Modèle stochastique **donné**	Théorie de la la **décision**	Processus de décision **markoviens**
	Les actions **ne changent pas** le monde	Les actions **changent** le monde

Les dynamiques des états et des récompenses sont inconnues

Deux approches :

- Sans modèle : **échantillonner** pour apprendre directement une bonne fonction d'**évaluation**
- Basée sur un modèle : apprendre d'abord un **modèle**, puis utiliser la **programmation dynamique** (MDP)

Problème de la grande dimension :

La **complexité** de calcul peut être prohibitive

Nécessité d'apprendre une **approximation** de la fonction d'évaluation ou de la stratégie

Problème d'exploration :

Comment explorer l'**espace des états** pour construire une bonne représentation de la fonction d'**évaluation** inconnue ?

Allocation séquentielle de ressources

Essais cliniques

- K **traitements** possibles (d'effet inconnu)
- Quel traitement **prescrire** à chaque patient en fonction des effets observés sur les patients précédents ?

Publicité en ligne

- K **publicités** éventuelles à afficher
- Quelle publicité **montrer** à chaque utilisateur en fonction des clics des utilisateurs précédents ?

Contexte

- Paramètres connus : K et n (nombre d'étapes)
- Ensemble de K bras dont les récompenses suivent des distributions ν_k **inconnues** à support dans $[0, 1]$
- A chaque étape t, on choisit un bras k_t et on reçoit une **récompense** r_t suivant la distribution ν_{k_t}
- **Objectif** : une **stratégie** de sélection des bras permettant de **maximiser** la somme **espérée** des récompenses

Le regret

- Une mesure de la **performance** d'une stratégie donnée
- Soit $\mu_k = \mathbb{E}[\nu_k]$ l'espérance associée au bras k
- Soit $\mu^* = max_k \ \mu_k$ la meilleure valeur espérée
- Le **regret** cumulé espéré est :

$$R_n = n\mu^* - \sum_{t=1}^{n} \mathbb{E}[\mu_{k_t}]$$

[P. Auer, N. Cesa-Bianchi, P. Fisher, 2002]

Principe d'optimisme face à l'incertain

- On attribue à chaque machine k un **indice** somme de :
  la **moyenne** des récompenses déjà reçues et
  d'une **borne supérieure** de la récompense espérée μ_k
  en **forte probabilité** (intervalle de confiance)
- On sélectionne la machine avec le plus **grand** indice

Algorithme :

- Sélectionner chaque machine une fois
- Pour $t = 1, 2, \ldots, n$ faire :
  Sélectionner la machine k qui **maximise**

$$\underbrace{\hat{\mu}_{k,s}}_{\text{exploitation}} + \underbrace{\sqrt{\frac{2log(t)}{s}}}_{\text{exploration}}$$

où $\hat{\mu}_{k,s}$ est la récompense **moyenne** obtenue avec le bras k
et $s = n_k$ est le nb de fois où la machine k a été sélectionnée

Formulation du regret cumulé espéré

$$R_n = (\sum_{k=1}^{K} \mathbb{E}[n_k])\mu^* - \mathbb{E}[\sum_{k=1}^{K} n_k \mu_k] = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k]$$

où $\Delta_k = \mu^* - \mu_k$ et n_k est le nombre de fois où le bras k à été sélectionné jusqu'à l'étape n

Hypothèse (H) : **Optimisme** dans l'incertain

A l'instant t, les **moyennes** *empiriques* des bras sont dans leurs intervalles de confiance respectifs

$$\mu_k - \sqrt{\frac{2log(t)}{s}} \leq \hat{\mu}_{k,s} \leq \mu_k + \sqrt{\frac{2log(t)}{s}}$$

Conséquence : si le bras k est choisi à l'étape t,

alors $\mu_k + 2\sqrt{\frac{2log(t)}{s}} \geq \mu^*$ c'est-à-dire $s \leq \frac{8log(t)}{\Delta_k^2}$

Inégalité de **Chernoff-Hoeffding** :

L'hypothèse (H) est satisfaite avec **probabilité** $\geq 1 - \frac{2}{t^4}$

Résultats :

- Chaque bras k **sous-optimal** est visité en moyenne, au plus

$$\mathbb{E}[n_k] \leq \frac{8log(t)}{\Delta_k^2} + C$$

où $\Delta_k = \mu^* - \mu_k$ et C est une constante < 4.3

- Le **regret** cumulé espéré est alors borné par

$$R_n = \sum_{k=1}^{K} \Delta_k \mathbb{E}[n_k] \leq 8\Big(\sum_{k:\Delta_k>0} \frac{log(n)}{\Delta_k} \Big) + C\Big(\sum_{k:\Delta_k>0} \Delta_k \Big)$$

- Le regret est **logarithmique** dans le nombre d'étapes :

$$R_n = O\Big(\sum_{k:\Delta_k>0} \frac{1}{\Delta_k} log(n) \Big)$$

- Il est possible d'obtenir une borne **indépendante** des distributions :

$$R_n = O(\sqrt{Knlog(n)})$$

Contexte : A chaque étape t

- l'**adversaire** affecte une récompense $r_{k,t} \in [0,1]$ à chaque bras
- le **joueur** choisit un bras k et reçoit la récompense r_{k_t}

L'objectif du joueur est de **maximiser** la somme des récompenses

Le regret

$$R_n = max_{k \in K}(\sum_{t=1}^{n} r_{k,t}) - \sum_{t=1}^{n} r_{k_t}$$

- Performance relativement à la **meilleure** stratégie fixée
- Peut-on espérer $lim_{n \to \infty} \sup_{recompenses} \mathbb{E}R_n/n = 0$?
- Si la stratégie du joueur est **déterministe**,
  alors l'espérance du regret peut être linéaire en n
- Une solution : les stratégies **probabilistes**

[P. Auer, N. Cesa-Bianchi, Y. Freund, R.E. Schapire, 2002]

Idée :

- A chaque étape, le joueur sélectionne un bras suivant une distribution mélange de la distribution **uniforme** et d'une distribution affectant à chaque action une probabilité multipliée par un facteur **exponentiel** dans la récompense cumulée espérée
- L'algorithme utilise la méthode des poids **multiplicatifs** pour mettre à jour les probabilités en tenant compte des **récompenses moyennes actuelles** de tous les bras

Analyse de l'algorithme

- Cette stratégie a tendance à donner des poids plus importants sur les décisions à récompenses plus fortes à **long terme**
- L'espérance du regret est **sous-linéaire**

$$\sup_{recompenses} \mathbb{E}R_n = O(\sqrt{nKlog(K)})$$

Quelques acteurs majeurs

Richard Bellman

Ronald Howard

Christos Papadimitriou

Michael Kearns

Modèles dynamiques :

- Processus **stochastiques** bien connus
- Critères de **performance** utilisés en
  recherche opérationnelle, économie,
  optimisation combinatoire, théorie du contrôle...

Applications (parmi beaucoup d'autres) :

- Modélisation de protocoles de communication
- Modélisation du contrôle dans les réseaux
- Optimisation dans les réseaux de capteurs sans fil
- Contrôle dans les systèmes embarqués

Modèle de décision :

- à chaque étape, un agent (ou un contrôleur) observe l'**état** du système et choisit une **action** (non déterminisme)
- il reçoit une **récompense** (ou subit un **coût**) et le système évolue vers un nouvel **état** en suivant une **distribution de probabilités** déterminée par le choix

Problème de décision markovien :

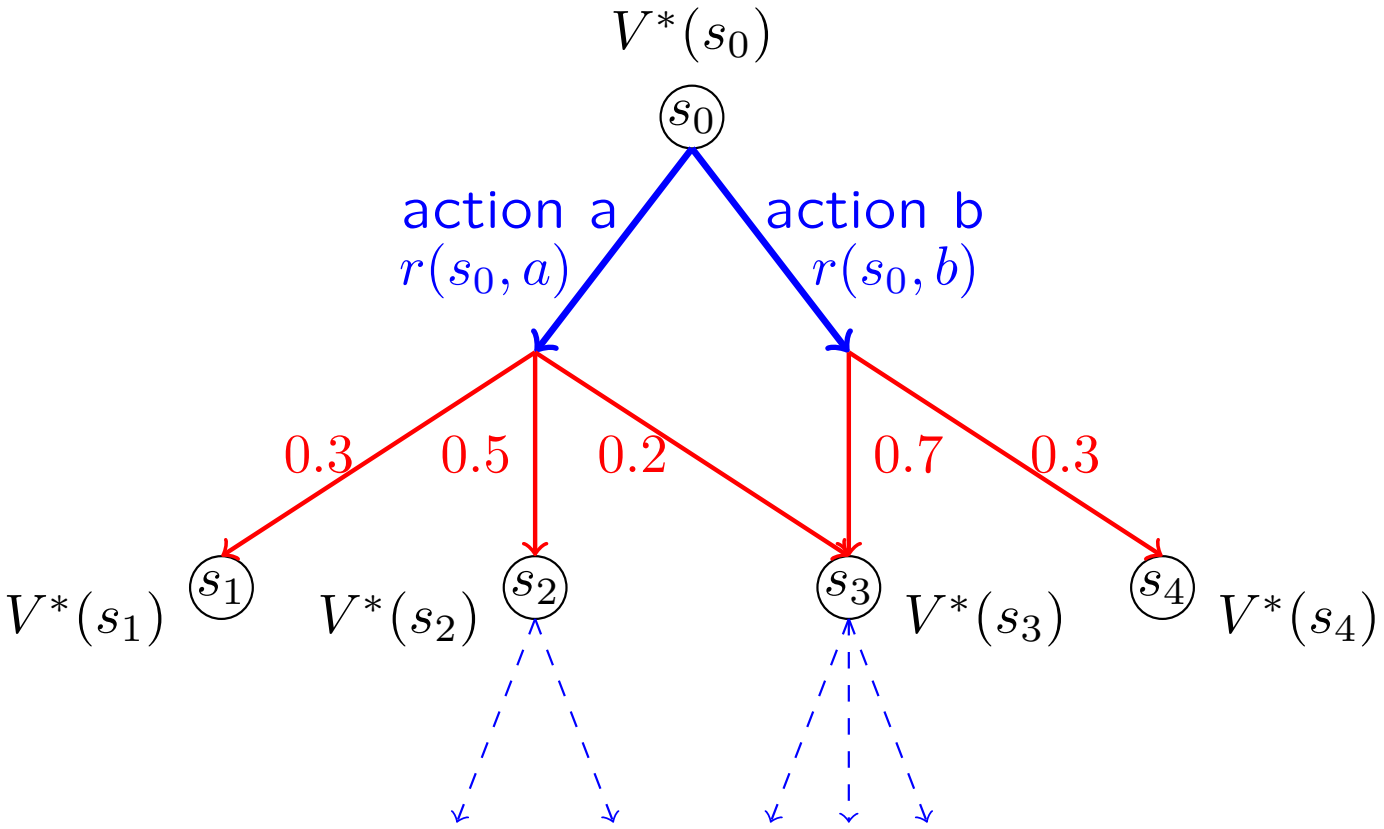
- **Evaluation** suivant un critère de performance
- Optimisation d'une fonction **valeur** associée à chaque état

Stratégie :

- Fonction associant une **action** à chaque **état**
- Détermine une suite de transitions dont la **valeur** est la **récompense** totale ou le **coût** total

Arbre des **trajectoires** à partir d'un état initial :

- Dans un état s, choix d'une **action** a
- Pour chaque couple (s, a), une **récompense** et une distribution de **probabilités** de transition sur les états



A chaque état s, est associée une valeur **optimale** $V^*(s)$?

$$\mathcal{M} = (S, s_0, A, P, R)$$

- S ensemble fini d'**états** (nb d'états $= n$)
- s_0 état initial (ou μ distribution de probabilités sur les états)
- A ensemble d'**actions** (nb d'actions $= m$)
- $P : S \times A \longrightarrow Distr(S)$ Relation de **transition**
  $Distr(S)$ ensemble des **distributions de probabilités** sur S
- $R : S \times A \longrightarrow \mathbb{R}^+$ Fonction de **récompense** (ou de **coût**)

Pour chaque couple état-action $(s, a) \in S \times A$:

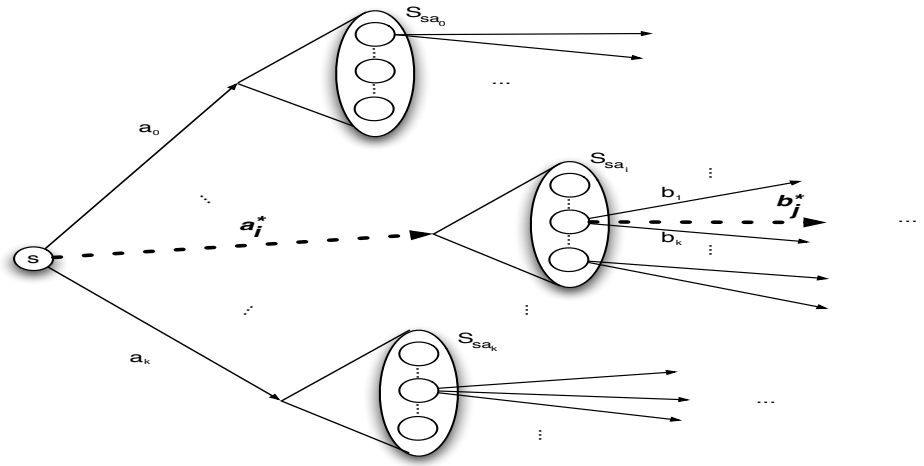
- $P_{s,a}(.)$ **distribution de probabilités** sur les états
- $R_{s,a}$ **récompense** (ou **coût**) associée à l'action a
  à partir de l'état s

Stratégie (stationnaire) :

- **déterministe** et sans mémoire $\pi : S \longrightarrow A$
- **probabiliste** et sans mémoire $\pi : S \longrightarrow Distr(A)$

Arbre des **trajectoires** :

- Chaque branche correspond à l'application d'une stratégie



- Si l'on se restreint à une stratégie π, on obtient la **chaîne de Markov** induite par la stratégie π

Problème de décision markovien :
trouver une stratégie **optimale** suivant un critère de **performance**

- Horizon H **fini** : récompense totale **espérée**

$$V^{\pi}(s) = \mathbb{E}_s^{\pi}(\sum_{i=1}^{H} R(s_i, \pi(s_i)))$$

  s_i étant l'état résultant à l'étape i suivant
  la chaîne de Markov induite par la stratégie π
- Horizon **infini** : récompense totale **espérée**
  avec taux de discount $0 < \gamma < 1$

$$V^{\pi}(s) = \mathbb{E}_s^{\pi}(\sum_{i=1}^{\infty} \gamma^{i-1} R(s_i, \pi(s_i)))$$

- Horizon **infini** : récompense **espérée moyenne**

$$V^{\pi}(s) = \lim_{T \to +\infty} \frac{1}{T} \mathbb{E}_s^{\pi}(\sum_{i=1}^{T} R(s_i, \pi(s_i)))$$

Problème de décision markovien : Horizon **infini**, avec **discount**

Fonction **auxiliaire** (Q-fonction) :

$$Q^{\pi}(s,a) = R_{s,a} + \mathbb{E}_s^{\pi}(\sum_{s' \in S} \gamma P_{s,a}(s') V^{\pi}(s'))$$

Fonction de récompense **optimale** (ou coût optimal) :
$V^*(s) = max_{\pi}\ V^{\pi}(s)$ et
$Q^*(s,a) = max_{\pi}\ Q^{\pi}(s,a)$ pour tout $(s,a) \in S \times A$
Stratégie **optimale** π^* : $\pi^*(s) = argmax_a\ Q^*(s,a)$ pour tout $s \in S$

Equation de **Bellman** :

$$V^*(s) = max_{a \in A}\ (R(s,a) + \gamma \sum_{s' \in S} P_{s,a}(s').V^*(s'))$$

Résolution :

- Méthodes itératives (**value** iteration, **policy** iteration)
- Programmation linéaire

Algorithme :

- Pour chaque $s \in S$, initialiser $V_0(s)$
- $h := 1$
- Tant que $h <$ nb maximum d'itérations
  pour chaque $(s, a) \in S \times A$,
  $Q_h(s, a) := R(s, a) + \gamma \sum_{s' \in S} P_{s,a}(s').V_{h-1}(s')$
  $V_h(s) := max_{a \in A} Q_h(s, a)$
  $h := h + 1$
- Pour chaque $s \in S$, $\pi^*(s) := argmax_{a \in A} Q_h(s, a)$
- Retourner π^*

Nombre maximum d'itérations :

- soit l'horizon **fini** H
- soit déterminé par un **critère d'arrêt** (horizon infini)
  $max_{s \in S} \mid V_h(s) - V_{h-1}(s) \mid \leq \varepsilon' = \varepsilon \frac{(1-\gamma)}{2\gamma}$

Ce critère garantit que la stratégie résultante π^* est ε-**optimale**

Le temps de calcul pour chaque itération est $O(mn^2)$

Algorithme :

- Soit π_0 une stratégie stationnaire déterministe
- Boucle :

  $\pi := \pi_0$

  Déterminer, pour chaque $s \in S$, $V^{\pi}(s)$ par résolution de l'équation de **Bellman** (à n inconnues)

  Pour chaque $s \in S$, s'il existe $a \in A$ t.q. $(R(s,a) + \gamma \sum_{s' \in S} P_{s,a}(s').V^{\pi}(s')) < V^{\pi}(s)$, alors $\pi_0(s) := a$, sinon $\pi_0(s) := \pi(s)$

  Répéter la boucle si $\pi \neq \pi_0$
- Retourner π

Remarque :

- Il existe au plus m^n stratégies distinctes
- Comme chaque stratégie améliore la précédente (Puterman, 1994) l'algorithme termine en au plus un nombre **exponentiel** d'étapes

Algorithme en 2 phases :

- Détermination de la fonction **valeur** (résolution d'un système d'équations linéaires) en $O(n^3)$ étapes
- Amélioration (éventuelle) de la **stratégie** en $O(mn^2)$ étapes

Complexité :

- Le nombre d'itérations est en $O(\frac{mn}{1-\gamma} log(\frac{n}{1-\gamma}))$ (Ye, 2010)
- Le nombre d'itérations est en $O(\frac{m}{1-\gamma} log(\frac{n}{1-\gamma}))$ (Hansen, Miltersen et Zwick, 2011)
- Borne exponentielle pour le problème sans discount, ou lorsque le taux de discount fait partie de l'entrée (Friedmann, 2009, Fearnley, 2010)

Sparse sampling : [M. Kearns, Y. Mansour, A.Y. Ng, 2001]

- Pour chaque action a, on engendre C **successeurs** de l'état s
- L'arbre des trajectoires est alors étendu jusqu'à un **horizon** H
- A une profondeur h, on estime **récursivement** la valeur $V_h^*(s)$

$$\hat{V}_h^*(s) = max_a \left(R(s,a) + \frac{\gamma}{C} \sum_{i=1}^{C} \hat{V}_{h-1}^*(succ_i(s,a)) \right)$$

- Les paramètres H et C sont choisis de manière à obtenir une stratégie **presque optimale**

Adaptative sampling : [H.S. Chang, M.C. Fu, S.I. Marcus, 2002]

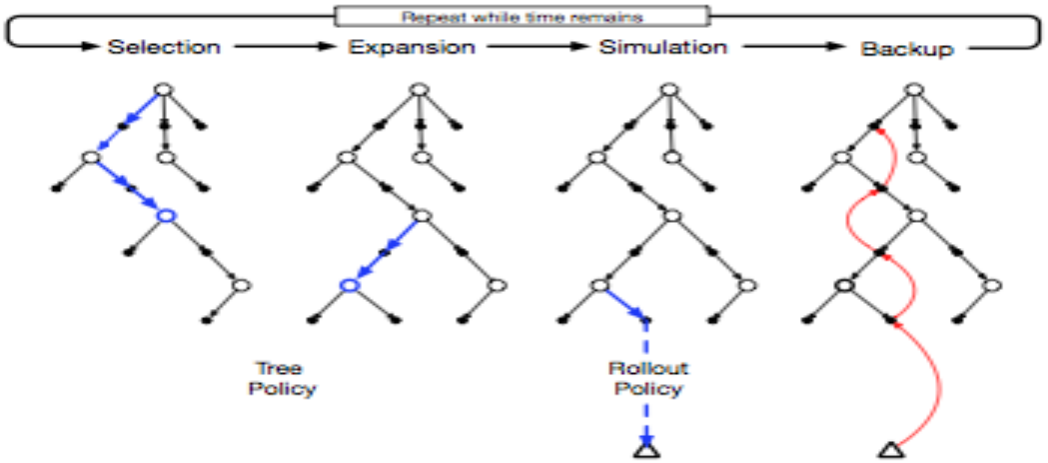
- Plutôt qu'échantillonner uniformément pour **chaque** action, l'algorithme UCB permet de **choisir** l'action à utiliser
- Les **meilleures** actions sont ainsi essayées plus souvent mais les autres actions incertaines sont aussi **explorées**

Remarques :

- On utilise un **générateur** probabiliste de trajectoires
- La complexité est **indépendante** du nombre d'états

Idée de base :

- A chaque noeud, est associée une statistique composée d'une récompense **moyenne** et d'un **compteur** du nb de visites



Algorithme : 4 étapes à chaque itération

- **Sélection** : descente dans l'arbre jusqu'à une feuille
- **Expansion** : ajout d'un successeur à la feuille choisie
- **Simulation** : stratégie aléatoire de choix d'actions et calcul d'une récompense moyenne
- **Propagation** arrière : la valeur obtenue est alors propagée depuis le nouveau noeud jusqu'à la racine

[L. Kocsis, C. Szepesvari, 2006]

Idée :

- Le **choix** d'un noeud successeur est traité comme un problème de **bandit** à plusieurs bras
- Appliquer la **stratégie UCB** sur les noeuds intérieurs
- La récompense d'un noeud successeur est la **moyenne** obtenue comme **approximation** par simulation **Monte-Carlo**

Stratégie UCB :

- $Q(s,a)$ est la récompense **moyenne** obtenue après avoir choisi l'**action** a dans l'état s
- $n_{s,a}$ = nb de fois où l'action a est choisie dans l'état s
- n_s = nb de fois où l'**état** s a été rencontré

$$\pi_{UCT}(s) = argmax_a\big(Q(s,a) + c\sqrt{\frac{2log(n_s)}{n_{s,a}}}\big)$$

où c est une constante d'**exploration**

Stratégies :

- Si le noeud **sélectionné** (UCB) a des successeurs non explorés, l'un des successeurs est choisi **aléatoirement**
- La stratégie de simulation est **uniformément** aléatoire

Contributions et limites :

- Performances expérimentales dans divers domaines
- La probabilité de sélectionner une action **sous-optimale** à la racine converge **asymptotiquement** vers 0
- La convergence de UCT peut être **très lente**
- Différentes améliorations ont été proposées
  Optimisation hiérarchique optimiste [Bubeck et al., 2011]

- Renouveau récent de l'apprentissage par **renforcement** :
  par exemple, pour l'allocation **séquentielle** de ressources
  dû également au succès impressionnants dans les **jeux**

- Importance de la modélisation par **bandits** manchots
  Le compromis entre **exploration** et d'**exploitation**
  Principe d'**optimisme** dans l'incertain

- Cadre plus général d'optimisation en univers **incertain** (MDP)
  L'objectif est l'**optimisation** plutôt que la **prédiction**

- Les méthodes classiques d'optimisation sont **impraticables**
  pour cause d'explosion **combinatoire** de la taille des modèles

- Les méthodes de **recherche** dans les **arbres** allient
  l'échantillonnage **adaptatif** aux simulations **Monte-Carlo**

- [ACF02] P. Auer, N. Cesa-Bianchi, and P. Fischer. *Finite time analysis of the multiarmed bandit problem*. Machine Learning, 47(2-3), p.235-256, 2002.
- [ACFS02] P. Auer, N. Cesa-Bianchi, Y. Freund, and R.E. Schapire. *The nonstochastic multi-armed bandit problem*. SIAM Journal on Computing, 32, p.48-77, 2002.
- [BPW12] C. Browne et al. *A Survey of Monte Carlo Tree Search Methods*. IEEE Transactions on Computational Intelligence and AI in Games, 4(1), 2012
- [H63] W. Hoeffding. *Probability inequalities for sums of bounded random variables*. Journal of the American Statistical Association, 58(301), p.13-30, 1963.
- [KS06] L. Kocsis and C. Szepesvari. *Bandit based Monte-Carlo planning*. Proc. of ECML06, p.282-293, 2006.
- [BS17] R.S. Sutton and A.G. Barto. *Reinforcement Learning : An Introduction.* Bradford Book. MIT Press, 1998, 2017.

- [B57] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957
- [CFM05] H.S. Chang, M.C. Fu, J. Hu and S.I. Marcus. *An Adaptive Sampling Algorithm for Solving Markov Decision Processes*. Operations Research, 53(1), p.126-139, 2005
- [HMZ11] T.D. Hansen, P.B. Miltersen and U. Zwick. *Strategy iteration is strongly polynomial*. Proc. Innovations in Computer Science, p.253-263, 2011
- [H60] R.A. Howard. *Dynamic programming and Markov process*. MIT Press, 1960
- [KMN02] M. Kearns, Y. Mansour and A.Y. Ng. *A sparse sampling algorithm for near-optimal planning in large Markov decision processes*. Machine Learning, 49(2-3), p. 193-208, 2002
- [P94] M. Puterman. *Markov Decision Processes*. John Wiley and Sons, 1994

Merci pour votre attention