# TME1 RLD

## Problèmes de bandits

Brient, Hein - Décembre 2021

## 1 Objectifs

L'objectif de ce TP est d'appliquer différentes algorithmes pour résoudre un problème de bandits multi-bras. Nous disposons d'un fichier de données pour la sélection de publicité en ligne contenant les contextes et les taux de clics sur les publicités de 10 annonceurs pour 5000 articles. Pour chaque visite, l'objectif est de choisir la publicité d'un des 10 annonceurs permettant d'engrenger le plus fort taux de clics.

Premièrement, nous implémentons trois baselines qui nous serviront de comparaisons pour les performances de nos algorithmes :
**(1)** Random : A chaque itération, on choisit n'importe quel annonceur.
**(2)** StaticBest : A chaque itération, on choisit l'annonceur avec le meilleur taux de clics cumulés.
**(3)** Optimale : A chaque iteration, on choisit l'annonceur qui a le meilleur taux de clics à cette itération.

Nous nous focalisons ensuite sur l'implémentation de trois algorithmes classiques : epsilon-greedy, UCB et UCB-lin afin de comparer leurs performances.

## 2 Algorithme Epsilon-greedy

Le principe de l'algorithme epsilon-greedy cherche à apporter une solution intuitive au compromis exploitation-exploration. A chaque itération, il y a deux cas de figure possibles : le premier consiste à sélectionner le bras de meilleure esperance empirique avec une probabilité $1 - \epsilon$ (exploitation) alors que le second consiste à sélectionner un bras aléatoire avec une probabilité $\epsilon$ (exploration). Les performances de cet algorithme sont très dépendantes du paramètre $\epsilon$, et il est parfois difficile de l'optimiser.

Nous implémentons l'algorithme en prenant soin de tester plusieurs valeurs de $\epsilon$, fixes pendant toute la durée de l'expérience, pour observer son impact sur les performances. Les courbes résultats sont présentées sur la figure 1. Les performances sont en effet très dépendantes du paramètre $\epsilon$. A titre indicatif, il est montré qu'un paramètre fixé à 1 correspond à une exploration tout au long de l'expérience et donne donc des performances semblables à la baseline Random. Par ailleurs, dans notre expérience, plus $\epsilon$ se rapproche de 0, meilleures sont les performances. Toutefois, il faut bien comprendre que dans la

plupart des expériences, sans exploration, une solution sous-optimale est très souvent atteinte. Une valeur "classique" de $\epsilon$ est 0.1, que nous garderons pour la suite.

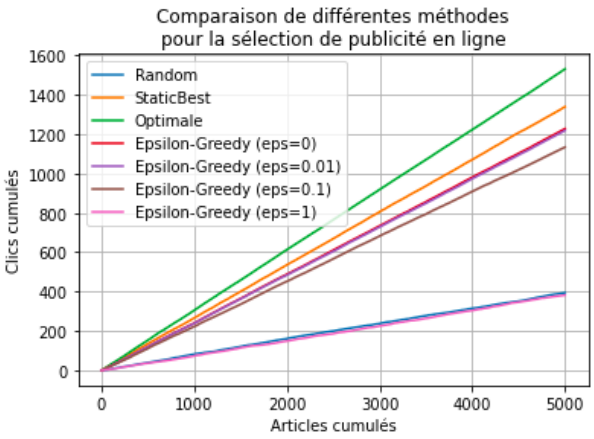


Figure 1 : influence de $\epsilon$

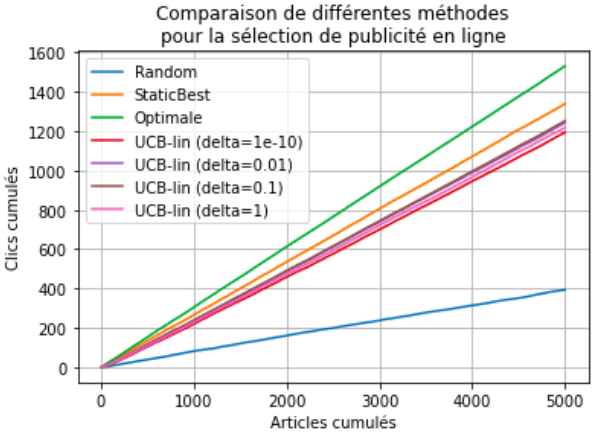


Figure 2 : influence de $\delta$

## 3 Algorithme UCB

L'algorithme UCB (Upper-Confidence Bound) implémente une stratégie dite "optimiste". En effet, à chaque itération et pour chaque bras, l'algorithme calcule un intervalle de confiance pour l'espérance de la récompense, et particulièrement sa borne supérieure, basés sur les observations de toutes les itérations précédentes. Ces bornes sont mises à jour à chaque itération et la politique consiste à choisir le bras de borne supérieure de confiance la plus haute. Cette borne peut être calculée pour chaque bras $i$ par la formule :

$$B_{t,s}(i) = \hat{\mu}_{i,s} + \sqrt{\frac{2log(t)}{s}}$$

Ici, $t$ représente le nombre d'itérations courant, $s$ le nombre de fois que le bras en question a été choisi parmi les $t$ itérations et $\hat{\mu}_{i,s}$ est l'espérance empirique de ce bras à l'itération $t$. Elle découle directement des inégalités de Chernhoff-Hoeffding avec lesquelles un intervalle de confiance de niveau $1-2t^{-4}$ a été fixé. La politique à l'itération $t$ est donc la suivante :

$$\pi_t = argmax_i(B_{t,s}(i))$$

## 4 Algorithme UCB-lin

L'algorithme UCB-lin a de plus que l'algorithme précédent la capacité de prendre individuellement en compte le contexte pour chaque bras. Dans le cas où nous disposons, comme ici, du contexte $x_{i,t}$ de chaque bras à chaque itération, nous pouvons rechercher des corrélations entre le contexte et le reward obtenu. Ces corrélations sont ainsi, en plus de l'intervalle de confiance de l'espérance de la récompense, des informations à prendre en compte pour prédire le meilleure bras à choisir en fonction du contexte de la nouvelle itération. Ainsi, selon le même principe que précedemment, un intervalle de confiance de

l'espérance conditionnelle de la récompense sachant le contexte est calculé et la borne supérieure est cette fois définie par la formule :

$$B_{t,s}(i) =< x_{i,t}, \hat{\theta}_i > +(1 + \sqrt{log(2/\delta)/2}) \times \sqrt{x_{i,t}^T (D_i^T D_i + I)^{-1} x_{i,t}}$$

Où un intervalle de confiance de niveau $1-\delta$ a été défini et $D_i$ repésente la matrice des contextes observés pour le bras $i$. Par ailleurs, les paramètres $\theta_i$ propres à chaque bras et linéairement associés aux contextes $x_{i,t}$, sont mis à jour par régression au fur et à mesure du processus :

$$\hat{\theta}_i = argmin_{\theta_i}(||D_i\theta_i - c_i||^2 + ||\theta_i||^2)$$

Nous montrons sur la figure 2 les courbes résultats de cet algorithme pour quelques différentes valeurs de $\delta$. Les performances sont très semblables mais un paramètre $\delta$ égal à 0.1 donne les meilleures performances.

## 5 Comparaison

Nous avons implémenté trois algorithmes classiques et avons déterminé, lorsque nécessaire, les hyper-paramètres semblant être les plus appropriés pour comparer de façon générale ces méthodes. Le graphique ci-dessous présente les courbes de performances de tous ces algorithmes.

Premièrement, on note une première distinction entre l'évolution des courbes des baselines et d'epsilon-greedy, qui s'apparentent à des droites et celles des algorithmes UCB, qui sont beaucoup plus convexes, traduisant un meilleur apprentissage des données. On peut d'ailleurs noter que lorque le nombre d'articles cumulés devient conséquent, la politique des algorithmes UCB se rapproche très fortement de la politique optimale au vu de l'évolution semblable des clics cumulés, ce qui n'est par exemple pas le cas pour l'algorithme epsilon-greedy. En conclusion, l'algorithme présentant les meilleures performances sans compromis est UCB-lin, qui parvient à tirer profit du contexte pour avoir une meilleure intuition et "mieux démarrer" que l'algorithme UCB. Toutefois, entre les deux autres algorithmes, le choix n'est pas aussi simple. On peut noter que pour un "faible" nombres d'itérations, un algorithme epsilon-greedy bien paramétré peut donner de très bonnes performances, mais il expose ses limites lorsque ce nombre augmente (pour un $\epsilon$ fixé), car l'exploration perd de son utilité avec les itérations et empêche de converger vers une politique optimale. Au contraire, selon ce graphique, l'algorithme UCB semble une méthode efficace lorsque le nombre d'itérations est plus grand, puisqu'il met beaucoup de temps à converger vers une politique suffisamment bonne, mais qui semble bien meilleure que celle suivie par epsilon-greedy.

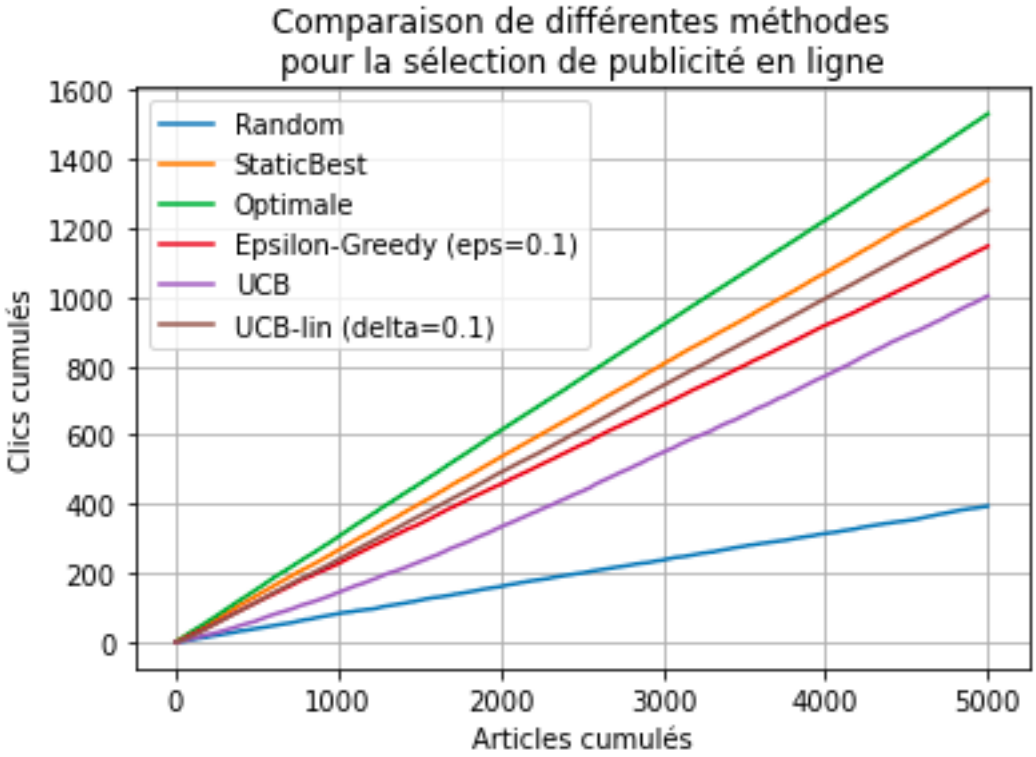


Figure 3 : comparaison finale