Rapport Projet 1 - Bandits

Aymeric - Hassan - Mohamed

October 2020

1 Introduction au problème des bandits [2]

Un problème des bandits est un jeu séquentiel impliquant un agent et un environnement. A chaque round t, l'agent va devoir choisir une action A_t parmi un ensemble $\mathcal{A}$ d'actions. En réponse à cette action, l'environnement lui donnera un reward $X_t \in \mathbb{R}$. Le but de l'agent est de maximiser $S_t = \sum_t X_t$ qui est son reward total sur les n rounds.

Pour maximier ses gains, l'agent va devoir apprendre de son environnement en adoptant une stratégie pour déterminer quelles actions faire pour maximiser S_t.

Définition 1.1. *Soit μ_a la moyenne du bras a, on définit le regret R_n comme suit*

$$R_n = n \max_{a \in \mathcal{A}} \mu_a - \mathbb{E}\left[\sum_{t=1}^{n} X_t\right] \tag{1}$$

Un algorithme pour maximiser ses gains est l'UCB pour Upper Confidence Bound. Il se base sur le principe de l'optimisme face à l'incertitude, c'est à dire qu'il suppose que l'environnement réagira de la meilleure manière possible. Dans notre cas, ce principe résulte dans le fait d'assigner à chaque action une UCB qui est une sur-estimation de la moyenne du reward donné par cette action.

Définition 1.2. *Soit X une variable aléatoire, elle est dite 1-sous-Gaussienne si*

$$\forall \gamma \in \mathbb{R}, M_X(\gamma) \leq e^{\gamma^2/2} \tag{2}$$

Où $M_X(\gamma)$ est la fonction génératrice de X de la forme $M_X(\gamma) = \mathbb{E}[e^{\gamma X}]$

Choisir des variables aléatoires de la forme précédente nous permet d'éviter la présence importante d'outliers qui pourraient compromettre notre principe de l'optimisme face à l'incertitude. Plus exactement, on peut citer le théorème suivant:

Theorème 1.1. *Soit $(X_t)_{t \leq n}$ une séquence indépendante de 1-sous-Gaussienne variables aléatoires, soit $\hat{\mu} = \frac{1}{n}\sum_t X_t$ la moyenne empirique, on a alors*

$$\forall \delta \in [0,1], \mathbb{P}\left(\mu \geq \hat{\mu} + \sqrt{\frac{2\log(1/\delta)}{n}}\right) \leq \delta \tag{3}$$

Du théorème précédent on tire la définition suivante qui nous sera nécéssaire pour l'algorithme que nous allons utiliser.

Définition 1.3. *Au round t, l'agent a tiré $T_i(t-1)$ exemples de l'action i. On définit alors*

$$UCB_i(t-1,\delta) = \begin{cases} \infty & si\ T_i(t-1) = 0 \\ \hat{\mu}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}} & sinon. \end{cases} \tag{4}$$

Une valeur recommandée pour $\delta \propto \frac{1}{n^2}$

2 Premier implémentation de UCB

Un premier algorithme UCB(δ) se base sur la quantité de l'équation 4, en effet, à chaque round, l'agent va choisir l'action A_t qui maximise la quantité $UCB_i(t-1,\delta)$

Prenons 6 actions, ou bras, que nous modéliserons par des distributions gaussiennes dont les moyennes sont aléatoires et les variances fixée à 1 pour respecter les hypothèse du théorème 1.1. Ces distributions sont illustrées par la figure 2

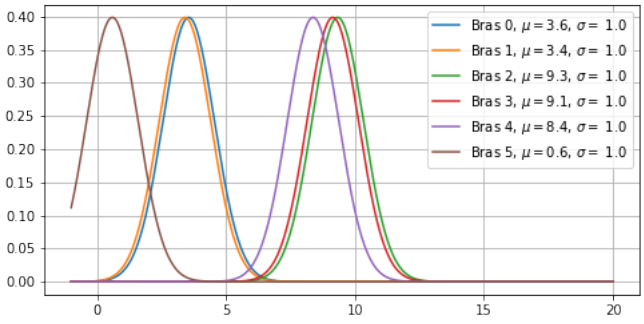

Figure 1: Bras

Un premier algorithme UCB(δ), décrit dans l'algorithme 1 nous permet, à partir des réalisations X_t, de déterminer quel est le bras le plus intéressant.

Algorithme 1 : UCB(δ)

Inputs k, δ and n;
Initialisation $X = 0_n$, $T = 0_k$, $\mu = 0_k$;
for $t \in [\![1, n]\!]$ **do** ;
    $\forall i \in [\![1, k]\!]$ compute $UCB_i(t-1,\delta)$;
    Choose action $A_t = argmax_i UCB_i(t-1,\delta)$;
    Observe reward X_t given by A_t ;
    Update $X^{[t]} = X_t$, $T^{[A_t]} += 1$, $\mu^{[A_t]} = \frac{1}{T^{[A_t]}} \sum_k 1_{A^{[k]}=A^{[t]}} X_k$;
Returns: $X = (X_t)_t$

On se propose de regarder le résultat de cet algorithme sur les bras choisis précédement. Nous traçons alors X_t, le regret R_t et les bras choisis A_t sur un premier graphe, et le regret en échelle logarithmique sur un second graphe (figure 2).

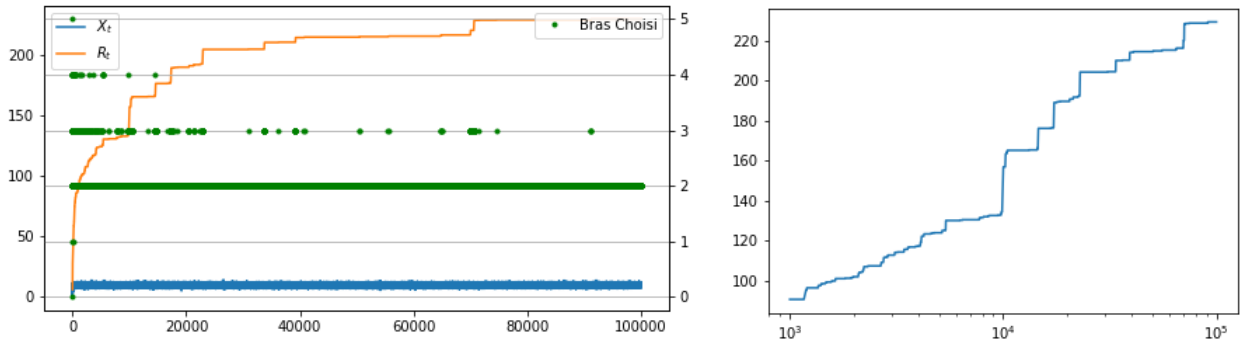

Figure 2: A gauche: Risque et Réalisations, à droite : Risque en échelle log

On constate qu'au bout d'un certain nombres de réalisations, l'algorithme converge vers le bras 2 qui est celui avec la moyenne la plus grande. Cette convergence à lieu à vitesse logarithmique comme le montre le graphique

3 Application à la recommandation de films

L'idée de cette approche avec les linear bandits est de choisir une action qui donnera le meilleur reward de la part de l'environnement. Dans le cadre de la recommandation de films, l'action a_t sera le film recommandé au round t et le reward X_t sera la note donnée par l'utilisateur à ce film.

On se propose d'utiliser l'algorithme LinUCB présenté dans le papier écrit par Lihong Li, Wei Chu, John Langford, and Robert E. Schapire [3].

On modélise notre dataset par une matrice $R \in \mathbb{R}^{u \times m}$, avec u le nombre d'utilisateurs et m le nombre de films. Ainsi, une ligne correspondra a un utilisateur et le coefficient de la colonne $k \in [1, m]$ de la ligne d'un utilisateur correspondra a la note qu'il a donné au film k.

Avec l'aide de la factorisation de matrice NMF, on obtient une décomposition de la forme

$$R = U \times M \qquad U \in \mathbb{R}^{u \times d},\ M \in \mathbb{R}^{d \times m} \tag{5}$$

avec d la dimension de la factorisation choisie.

Dans l'algorithme, on définit ainsi le terme $x_{t,a}$ par la colonne a de la matrice M, c'est à dire les informations concernant le film a, qui peuvent s'interpréter comme une sorte de moyenne si $d = 1$ de notre film a. De plus, r_a correspondra à la note que donnera l'utilisateur au film qu'on lui propose au round t.

Algorithme 2 : LinUCB

Inputs: $\alpha \in \mathbb{R}_+$, $n \in \mathbb{N}$;
for $t \in [\![1, n]\!]$ **do** ;
Initialisation $A_a = I_d$, $B_a = 0_{d\times 1}$;
for all $a \in A_t$ **do** ;
$\hat{\theta}_a = A_a^{-1} b_a$;
$p_{t,a} = \hat{\theta}_a^T x_{t,a} + \alpha\sqrt{x_{t,a}^T A_a^{-1} x_{t,a}}$;
Choose arm $a_t = argmax_{a\in A_t} p_{t,a}$;
$A_{a_t} = A_{a_t} + x_{t,a} x_{t,a}^T$;
$b_{a_t} = b_{a_t} + r_t x_{t,a_t}$;
Returns: $(a_t)_t$

On prend le dataset avec tous les films, on trie pour avoir ceux qui ont pas mal de ratings et les utilisateurs pertinents. On obtient ainsi la matrice R défini précédemment.

Nous allons regarder l'effet de l'algorithme 2 appliqué à un utilisateur (cet utilisateur est enlevé de la matrice lors de la NMF). On donne la liste des films que l'utilisateur a donné et les notes correspondantes dans le tableau 3

Id du film	12	24	52	58	63	65	69	71	80	101	128	135	161	192	197	201
Note	5	4	4	4	2	5	3	4	5	5	5	5	5	4	1	4

Figure 3: Notes de l'utilisateur considéré

L'algorithme nous renvoie les outputs de la figure 4. Celui-ci nous propose à tour de roles les films 12,52,65,80,101 et 135 pour lesquelles un seul a la note de 4 et les autres la note de 5 par cet utilisateur ce qui nous semble totalement satisfaisant.

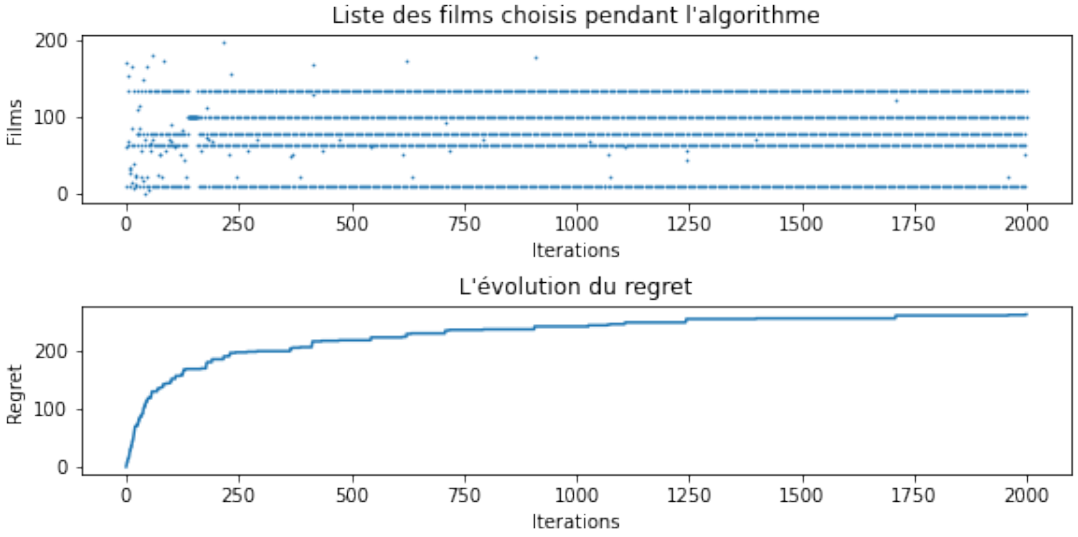

Figure 4: Résultats de l'algorithme LinUCB

Bien que le résultat soient satisfaisant dans la théorie, nous souhaitons quand même recommander des films différents aux utilisateurs et pas toujours le même film. C'est ce à quoi nous nous intéressons dans la partie suivante

4 Groupes d'utilisateurs avec K-Means et UCB

4.1 Idée de l'algorithme

Notre objectif est de pallier au problème du **cold-start**, c'est-à-dire quand un utilisateur est nouveau à la plateforme de recommandation. Pour ce faire, il faut réunir rapidement et efficacement des informations sur ce nouvel utilisateur en lui proposant d'abord de nouveaux films à regarder et observer la note qu'il leur a attribué.

→ Cependant, on veut maximiser sa satisfaction au cours de ce processus en lui proposant des films qu'il est plus susceptible d'aimer en premier.

L'algorithme va se dérouler en 4 temps :

- **Prédiction de ratings :** En utilisant la factorisation de matrice pour les non-cold users de la matrice notée R. On construit la matrice R_d en réglant l'hyper-paramètre de la factorisation de matrice d. La matrice R_d contient donc les notes prédites pour tous les films.
- **Clustering :** On cherche des clusters d'utilisateurs en se basant sur leurs notes de films de la matrice R_d. On utilise l'algorithme K-Means et du fine-tuning pour trouver la valeur adéquate.
  Les clusters vont représenter les bras dans notre algorithme UCB présenté en section 1.
- **Consensus :** Une fois les clusters obtenus, on calcule **la moyenne** des notes des utilisateurs pour chaque cluster. On obtient alors K séries de notes $(\theta_k)_{k=1,..K}$ correspondant aux K clusters pour tous les films de notre base de données where $\theta_k = (\theta_{k,1}, ..., \theta_{k,Nmovies})$.
- **Recommandation :** Comme dans l'algorithme UCB, on va au cours de n rounds demander le reward aux clusters pour pouvoir déterminer celui avec la meilleure note en moyenne.
  Pour simuler l'effet de 'sampling' des bras/clusters, on procède comme-suit :
  - On classe les θ_k par ordre décroissant de sorte à avoir le meilleurs films de chaque cluster en premier.
  - L'action de tirer un bras correspond dans notre cas à recommander à l'utilisateur le film qui est en début de la liste s'il n'a pas été déjà recommandé, si c'est le cas, on passe au suivant et ainsi de suite.
  - Le reward correspond à la note que donne l'utilisateur au film qu'on vient de lui proposer.

C'est de cette manière qu'on simule le tirage des bras et qu'on applique l'algorithme UCB afin de trouver le cluster/bras avec la meilleure note moyenne qui correspondrait donc au cluster de prédilection du-dit utilisateur.

4.2 Etape de clustering

Pour faire le clustering, on utilise l'algorithme K-means sur la matrice R_d obtenu par la factorisation de matrices afin de regrouper les utilisateurs.
On utilise la méthode du coude pour choisir un intervalle sur lequel on fait du fine-tuning de la valeur nb-clusters.

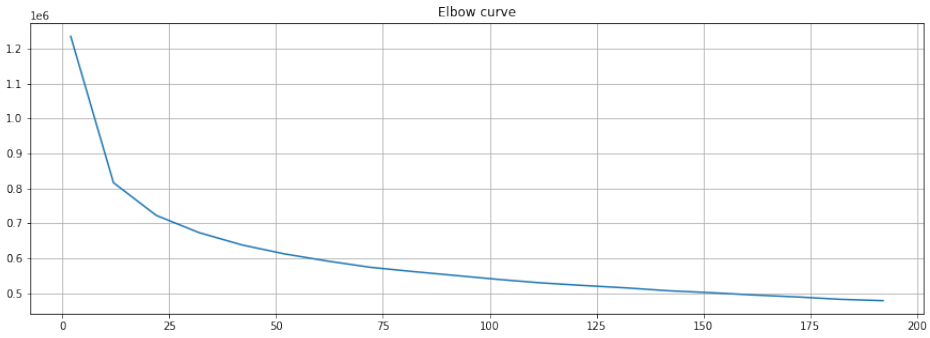

Figure 5: Coude du K-means

On constate que la baisse de distortion la plus grande s'effectue sur l'interval nb-clusters$<$ 50. C'est donc sur cet intervalle qu'on fait tourner l'algorithme de fine-tuning: on constatera dans la prochaine partie que la valeur nb-clusters $= 6$ donne les meilleurs résultats selon une certaine métrique d'évaluation. (Voir figure 6)

4.3 Procédé d'évaluation

Comme nous nous intéressons aux colds users, nous adoptons le protocole du Leave-One-Out.

A chaque tour, nous nous entraînons sur tous les utilisateurs, sauf un, qui sert de test utilisateur. Comme nous n'utilisons aucune information sur l'utilisateur du test, il s'agit d'un cold user.

4.3.1 Métrique choisie

La qualité de nos recommandations est mesurée en calculant la métrique du Normalized Discounted Cumulated Gain ($NDCG$) présentée dans le papier [1]

$$NDGC = \frac{1}{N} \sum_{i=1}^{N} \frac{DGC(u_i)}{IDGC(u_i)} \tag{6}$$

- $DCG(u)$ est le gain cumulé actualisé du classement prévu pour une cible utilisateur u.
- $IDCG(u)$ est l'Ideal DCG' qui correspond à la série optimale de recommandation possible pour u.
- N est le nombre d'utilisateurs dans l'ensemble de résultats.

Au vu des définitions précédentes, maximiser la DGC revient donc à essayer de recommander les films les mieux notés en premier afin de maximiser la satisfaction de l'utilisateur.

$$DGC(u) = r_{u,1} + \sum_{j=1}^{rounds} \frac{r_{u,j}}{log_2(j)} \tag{7}$$

4.3.2 Baselines pour comparer

Afin de mieux situer la performance de notre algorithme, nous le comparons aux deux approches suivantes :

- **Random selection(Dummy model) :** Dans cette méthode, l'algorithme va proposer des films au hasard à l'utilisateur au fur et à mesure des rounds.
- **Best average :** Dans cette méthode, l'algorithme va proposer les films par ordre décroissant de leur moyenne dans le dataset.

4.4 Résultats

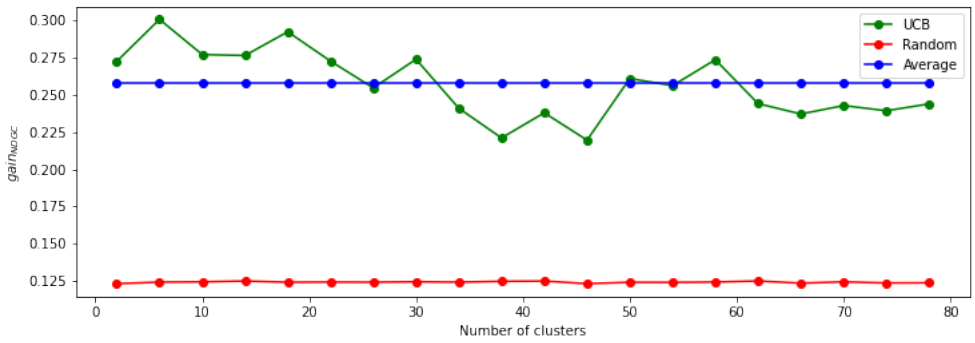

Figure 6: Gain en fonction du nombre de clusters

D'après la figure 6, nous constatons que pour la plupart des valeurs de K (le nombre de clusters), l'approche UCB est meilleure que les deux autres selon les métriques définies précédemment.

Par ailleurs, nous retrouver le pic de performance à $K = 6$ que nous avons avancé précédemment.

L'influence de d n'est pas très grande. Nous choisissons la valeur $d = 10$ qui donne le plus grand gain comme le montre la figure 7.

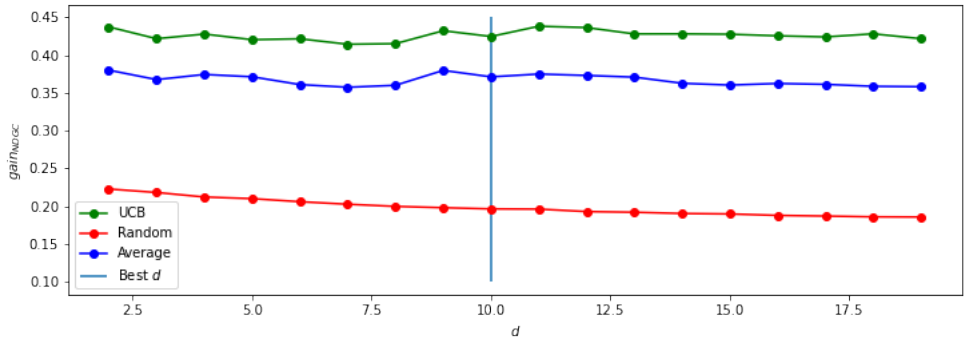

Figure 7: Gain en fonction de d

Recommendation Method	Random	Best over average	UCB algorithm
NDGC value	0.152	0.375	**0.443**

Figure 8: Résultats du gain NDGC par algorithmes

On regroupe dans le tableau suivant les résultats des algorithmes UCB/Best-average/Random :

On constate que l'algorithme UCB nous donne les meilleures performances vis-à-vis des métriques définies précédemment.

4.5 Présentation des résultats sur deux exemples

Une fois la méthodologie présentée, nous pouvons l'appliquer sur des nouveaux utilisateurs pour voir quels seront les films qui leur seront présentés.

Nous avons généré deux utilisateurs, u_0 qui sera un utilisateur définit comme une moyenne de 10 utilisateurs du cluster 0 et de même pour u_3 avec le cluster 3.

On fait tourner l'algorithme et la méthode précédente pour obtenir la moyenne de rewards par classe pour nos 2 utilisateurs.

Clusters	0	1	2	3	4	5
u_0	**2.62**	2.22	2.07	1.41	1.24	0.49
u_3	2.39	2.49	2.32	**3.72**	3.71	1.13

Figure 9: Reward des users

On constate d'après le tableau 10 que les clusters 0 et 3 ont respectivement les valeurs les plus élevées pour les utilisateurs u_0 et u_3.

$\rightarrow$ Ils sont bien associés aux bons clusters, on peut ainsi leur recommandé des films dans ces clusters.

Pour finir, les 7 premiers films recommandés pour les utilisateurs sont les suivants, on précise également le ratings, qui est le rating normalisé r_n qui est le rating du film divisé par la note moyenne que l'utilisateur donne aux films

u_0	id	172	41	12	63	167	35	15
	r_n	3.006	2.083	3.831	3.535	2.276	1.302	1.745
u_3	id	172	41	12	63	69	154	29
	r_n	1.876	2.116	4.504	4.328	3.804	4.017	3.997

Figure 10: Films proposés par l'algorithme

On constate que tous les films ont un rating normalisé supérieur à 1 ce qui montre que les films proposés semblent judicieux.

References

[1] Crícia Z. Felício, Klérisson V.R. Paixão, Celia A.Z. Barcelos, and Philippe Preux. A multi-armed bandit model selection for cold-start user recommendation. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*, UMAP '17, page 32–40, New York, NY, USA, 2017. Association for Computing Machinery.

[2] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[3] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *Proceedings of the 19th international conference on World wide web - WWW '10*, 2010.