Introduction au problème des bandits multi-bras

Tommy Mastromonaco

Août 2021

Introduction

Présentation du problème

Dans le problème des bandits multi-bras, on doit choisir entre des options distinctes dont la récompense aléatoire est incertaine. Le terme "bandit" désigne les machines à sous des casinos que l'on actionne en tirant le bras.

Figure – Exemple de machines à sous, ou "bandits"

Applications

Voici quelques applications du problème :

- **Affichage de publicités :** Sur un site web, un algorithme doit choisir entre différentes publicités à afficher pour inciter l'utilisateur à cliquer.
- **Serveurs informatiques :** Dans un *data center*, on doit sélectionner un serveur parmi d'autres pour accomplir une tâche dans le délai le plus court.
- **Essais cliniques :** Dans le cadre d'essais cliniques, on doit choisir un traitement expérimental à administrer à un sujet qui soit le plus efficace contre une certaine maladie.

Voici un petit jeu pour présenter le modèle que nous allons explorer.

Le modèle

Description mathématique

- Trois bras (numérotés de 1 à 3), et l'action du bras k donne une récompense de 1 avec probabilité θ_k, et 0 avec probabilité $1 - \theta_k$.
- Le choix du bras à l'instant t est $a_t \in \{1, 2, 3\}$, donc la récompense est $R_t \mid a_t, \theta \sim \mathrm{Ber}(\theta_{a_t})$, où $\theta = (\theta_1, \theta_2, \theta_3)$ est fixe.
- Si on joue à T reprises, l'objectif est de maximiser $\mathbb{E}_\pi \left[\sum_{t=1}^{T} R_t \mid \theta \right]$, où $\pi = (a_1, a_2, \ldots, a_T)$ est le vecteur des bras choisis sur les T périodes.
- Si le bras à la probabilité la plus élevée est $a^* = \mathrm{argmax}_{k \in \{1,2,3\}} \theta_k$, alors le vecteur d'actions optimal est évidemment $\pi^* = \{a^*\}^T$.

Bandit Beta-Bernoulli

- Toutefois, on ne connaît pas θ. Pour choisir intelligement les bras, nous devons estimer θ en accumulant de l'information sur les bras.

- Nous utilisons un modèle bayésien, donc avec un *a priori* sur θ, et qui est indépendant sur chaque bras.

- À la période initiale, la distribution *a priori* sur θ_k suit une loi beta de paramètres (α_k, β_k). On a donc les vecteurs $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ et $\beta = (\beta_1, \beta_2, \beta_3)$. La distribution sur le bras k est donc

$$f(\hat{\theta}_k) = \frac{\Gamma(\alpha_k + \beta_k)}{\Gamma(\alpha_k)\Gamma(\beta_k)} \hat{\theta}_k^{\alpha_k - 1} (1 - \hat{\theta}_k)^{\beta_k - 1}.$$

Avantages de ce modèle

- Le support de la loi beta est $[0, 1]$ et la distribution prend des formes bien différentes selon les paramètres.

- À l'instant t, les paramètres *a posteriori* sur le bras k sont

$$(\alpha'_k, \beta'_k) = \begin{cases} (\alpha_k, \beta_k), & \text{si } a_t \neq k \\ (\alpha_k, \beta_k) + (r_t, 1 - r_t), & \text{si } a_t = k. \end{cases}$$

- α_k et β_k représentent en quelque sorte le nombre de *succès* et d'*échecs*, respectivement.

- L'espérance de la beta est $\frac{\alpha_k}{\alpha_k + \beta_k}$, et la distribution se concentre autour de la moyenne lorsque $\alpha_k + \beta_k$ croît.

Algorithme gourmand

Présentation de l'algorithme

Nous avons besoin d'un algorithme qui estime θ et qui choisit un bras à chaque période. Voici ce que fait l'algorithme dit *gourmand* à un instant t :

- On utilise l'espérance des distributions actuelles comme estimation, c'est-à-dire que $\hat{\theta} = \frac{\alpha}{\alpha+\beta}$.
- Le bras choisi est $a_t = \text{argmax}_k\, \hat{\theta}_k$, soit celui dont l'estimation est maximale. Si plus d'un bras est maximal, on en tire un au hasard.
- On actionne ce bras et on observe la récompense r_t générée aléatoirement suivant une Bernoulli $p = \theta_{a_t}$.
- On met à jour les paramètres du bras : $(\alpha'_{a_t}, \beta'_{a_t}) = (\alpha_{a_t}, \beta_{a_t}) + (r_t, 1 - r_t)$. Les paramètres des autres bras ne changent pas.

On répète ensuite ces étapes à l'instant suivant, et ainsi de suite.

Proportion des choix

La figure suivante montre la fraction des bras choisis à chaque période sur 1000 simulations. L'*a priori* est $\alpha = \beta = (1, 1, 1)$.

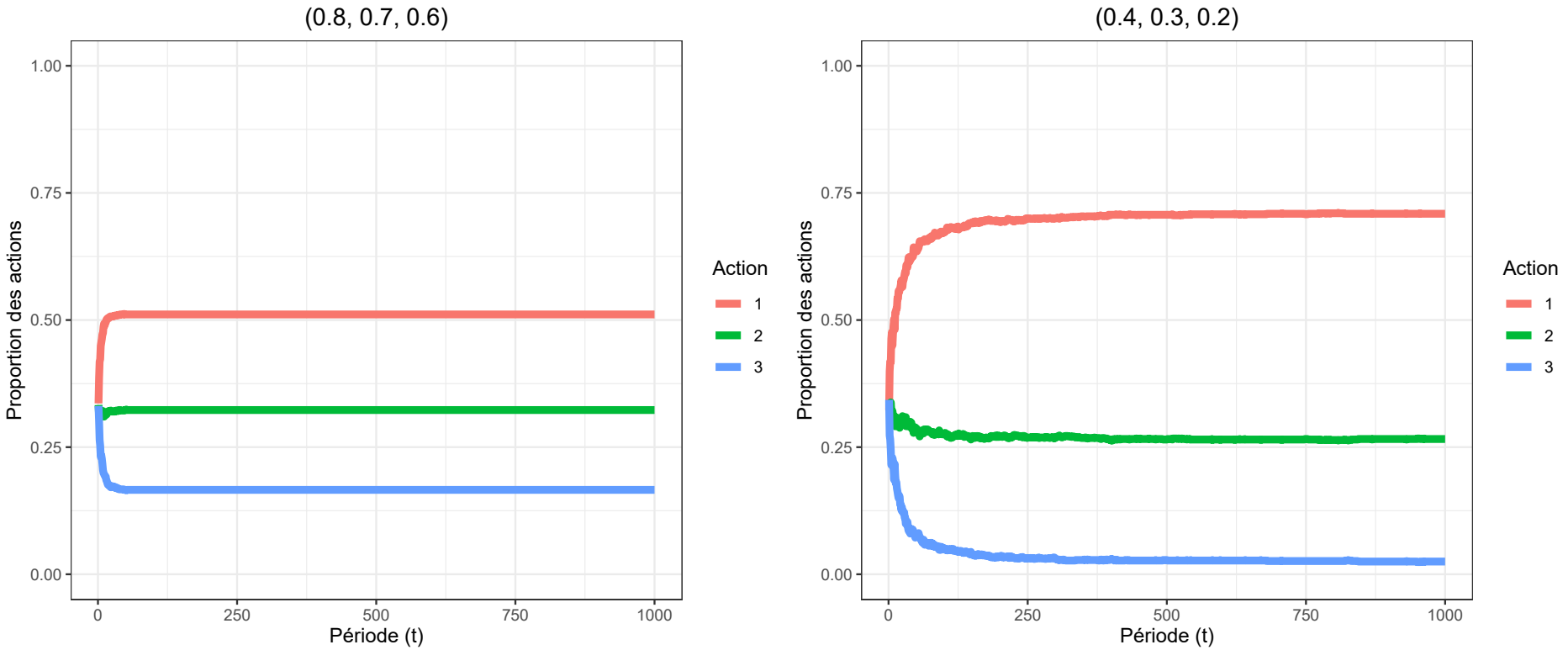

Figure – Proportion des bras choisis avec l'algorithme gourmand

Performance de l'algorithme

- Toutes les simulations de cette présentation auront comme *a priori* $\alpha = \beta = (1, 1, 1)$, soit une loi uniforme sur $[0, 1]$.

- L'algorithme gourmand utilise majoritairement le meilleur bras, mais pas suffisamment. La raison est qu'il **exploite** trop et qu'il n'**explore** pas assez. Il va toujours utiliser le même bras sans en explorer d'autres, même si ce bras n'est pas optimal.

- L'algorithme est sensible à la "chance" aux premières périodes. Si on est malchanceux en utilisant le meilleur bras, l'algorithme gourmand va en exploiter un autre.

Illustration

Dans cette figure, les bras rouge et bleu sont meilleurs. Toutefois, le bras bleu a une espérance supérieure (0,80 contre 0,75) ; l'algorithme risque d'exploiter longtemps ce bras sans essayer d'explorer le bras rouge.

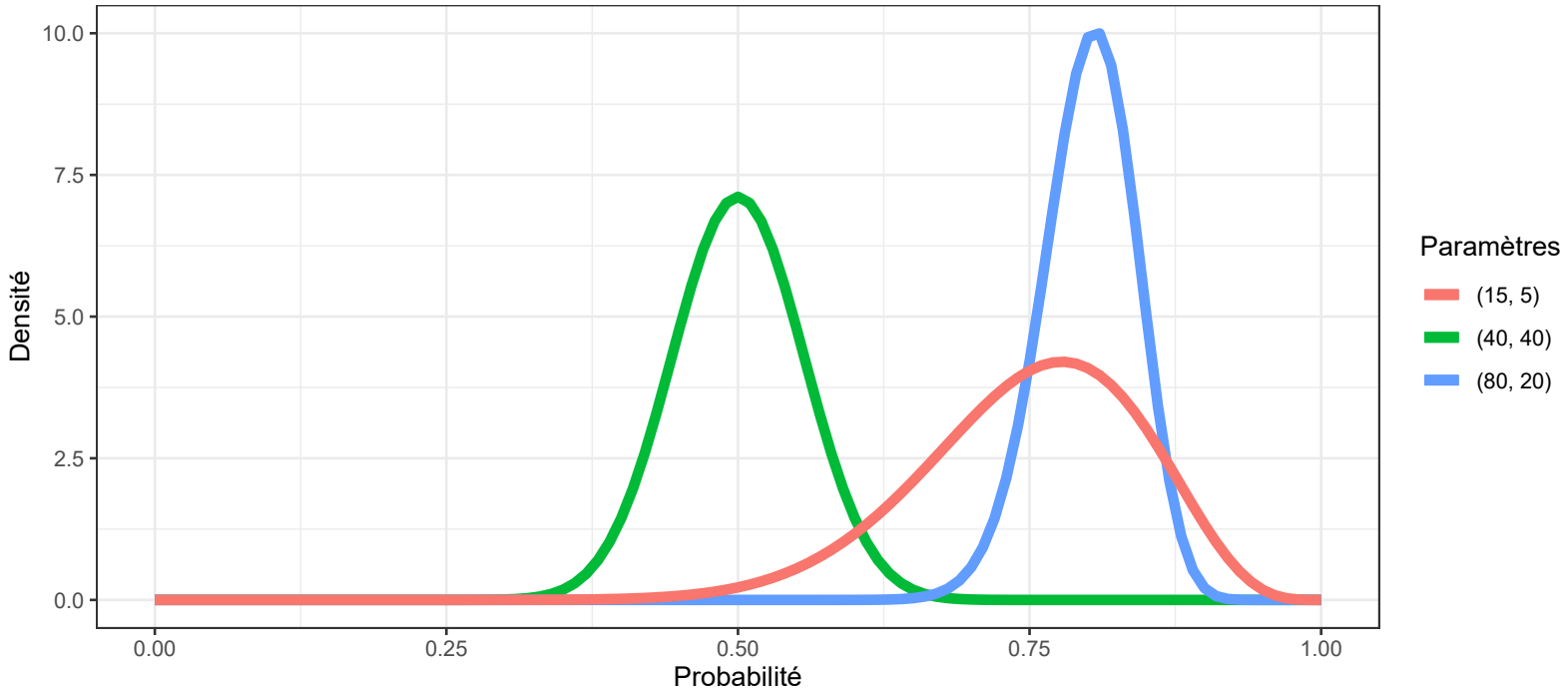

Figure – Distribution *a priori* de trois bras

Algorithme ϵ-gourmand

Présentation de l'algorithme

Pous ajouter une part d'**exploration** à l'algorithme gourmand, on peut incorporer une perturbation aléatoire :

- La manière doit on choisit le bras est tirée aléatoirement. On applique l'action gourmande avec probabilité $1 - \epsilon$, ou on choisit l'un des trois bras au hasard (uniformément) avec probabilité ϵ.
- On actionne ce bras et on observe la récompense r_t générée aléatoirement suivant une Bernoulli $p = \theta_{a_t}$.
- On met à jour les paramètres du bras : $(\alpha'_{a_t}, \beta'_{a_t}) = (\alpha_{a_t}, \beta_{a_t}) + (r_t, 1 - r_t)$. Les paramètres des autres bras ne changent pas.

On répète ensuite ces étapes à l'instant suivant, et ainsi de suite. $\epsilon = 0.1$ est une valeur typique pour ce paramètre.

Proportion des choix

La figure suivante montre la fraction des bras choisis à chaque période sur 10000 simulations. L'*a priori* est $\alpha = \beta = (1, 1, 1)$, et $\epsilon = 0.1$.

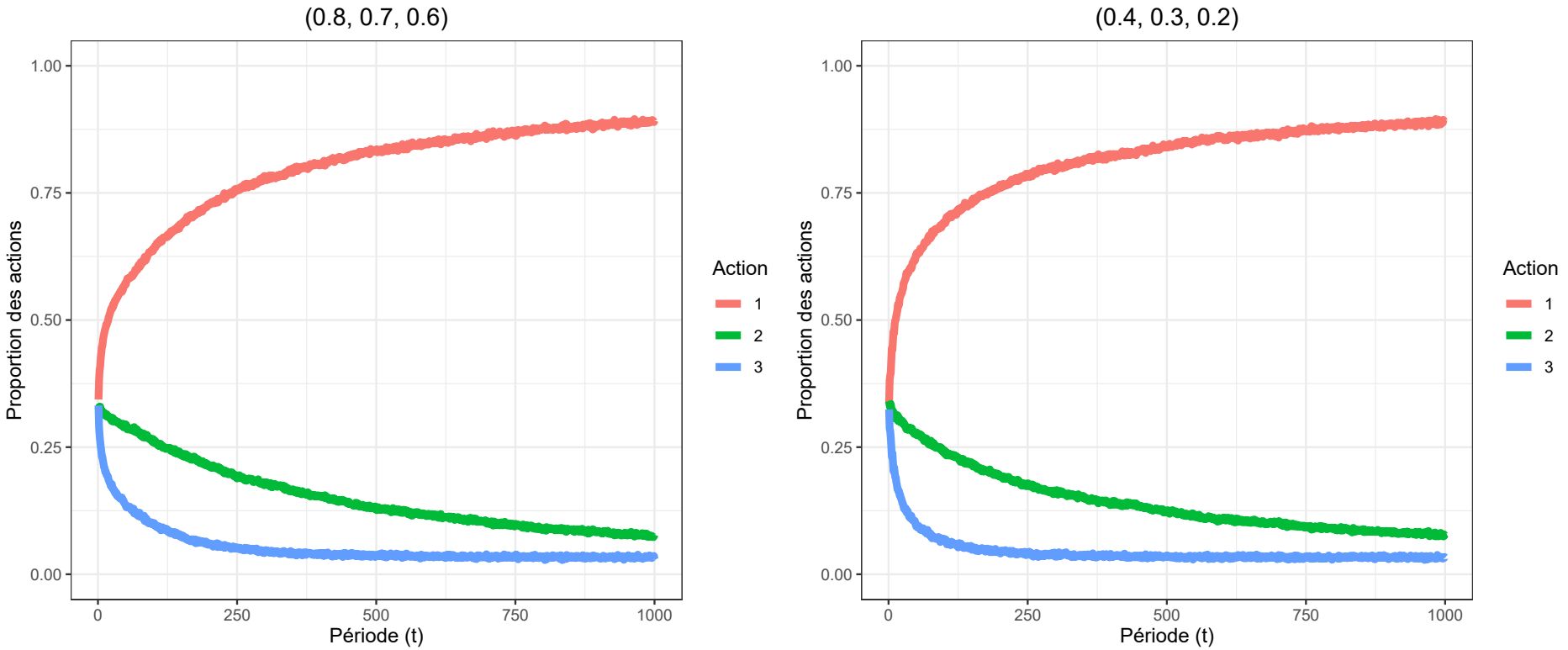

Figure – Proportion des bras choisis avec l'algorithme ϵ-gourmand

Illustration

L'algorithme n'explore pas intelligemment : il va parfois utiliser le bras vert alors qu'il est sans aucun doute inférieur. De plus, l'exploration devrait diminuer au fil du temps.

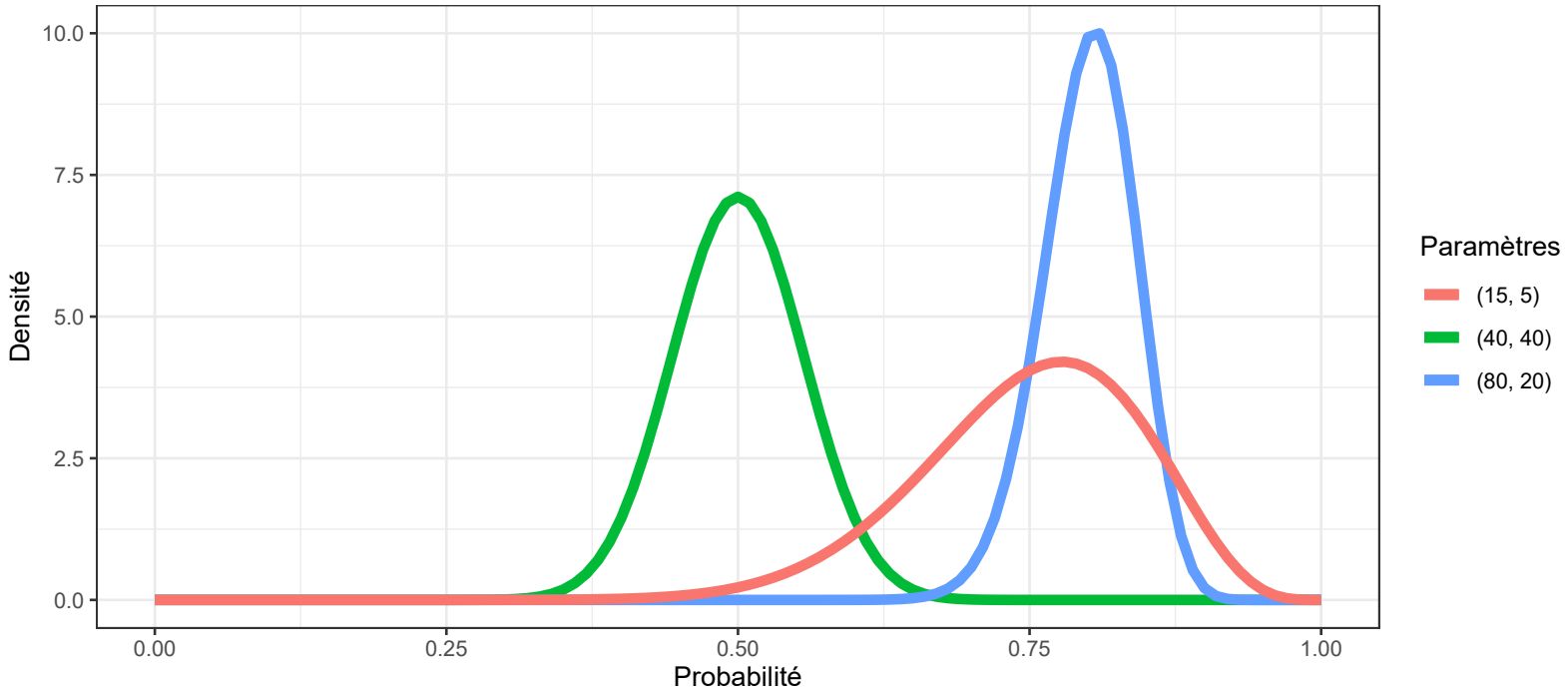

Figure – Distribution *a priori* de trois bras

Échantillonnage de Thompson

Présentation de l'algorithme

L'échantillonnage de Thompson jauge intelligemment la part d'exploration et d'exploitation en fonction de l'information acquise sur les bras :

- On estime la probabilité des bras en échantillonnant sur leur loi beta respsective, c'est-à-dire que $\hat{\theta}_k \sim \mathrm{Beta}(\alpha_k, \beta_k)$.
- Le bras choisi est $a_t = \mathrm{argmax}_k\, \hat{\theta}_k$, soit celui dont l'estimation est maximale.
- On actionne ce bras et on observe la récompense r_t générée aléatoirement suivant une Bernoulli $p = \theta_{a_t}$.
- On met à jour les paramètres du bras : $(\alpha'_{a_t}, \beta'_{a_t}) = (\alpha_{a_t}, \beta_{a_t}) + (r_t, 1 - r_t)$. Les paramètres des autres bras ne changent pas.

On répète ensuite ces étapes à l'instant suivant, et ainsi de suite.

Proportion des choix

La figure suivante montre la fraction des bras choisis à chaque période sur 10000 simulations. L'*a priori* est $\alpha = \beta = (1, 1, 1)$.

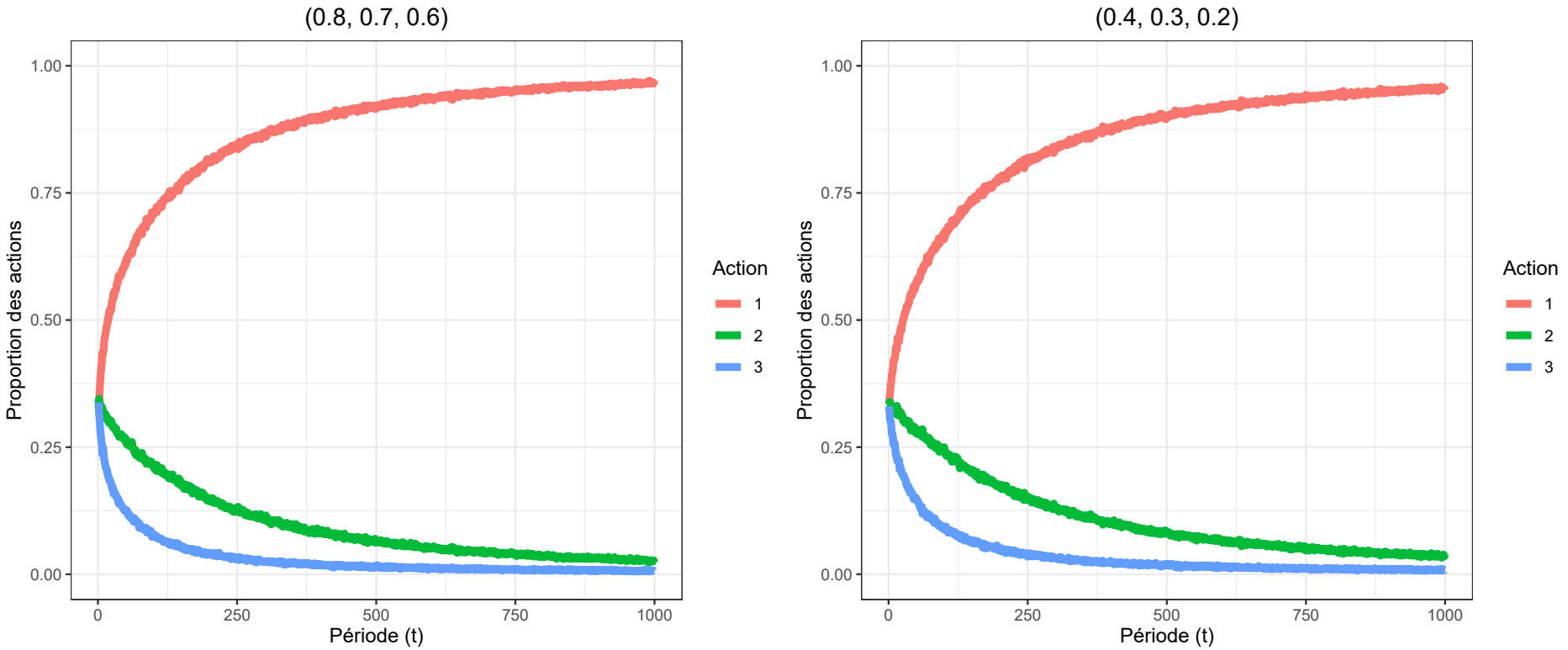

Figure – Proportion des bras choisis avec l'échantillonnage de Thompson

Regret

- Pour comparer efficacement la performance des algorithmes, on compare leur **regret** à chaque instant t.
- Le *regret par période* est la différence entre la récompense espérée du bras optimal et du bras utilisé à l'instant t, soit $\mathbb{E}_\pi \left[\theta^* - R_t \mid \theta\right]$, où $\theta^* = max_k\, \theta_k$.
- Le **regret** à l'instant T est alors la somme des regrets par période jusqu'à T, soit $\mathbb{E}_\pi \left[\sum_{t=1}^{T} \left(\theta^* - R_t\right) \mid \theta\right]$.

Regret des algorithmes

La figure suivante montre le regret moyen des algorithmes sur 1000 simulations. À chaque simulation, θ est déterminé au hasard pour observer la performance générale des algorithmes.

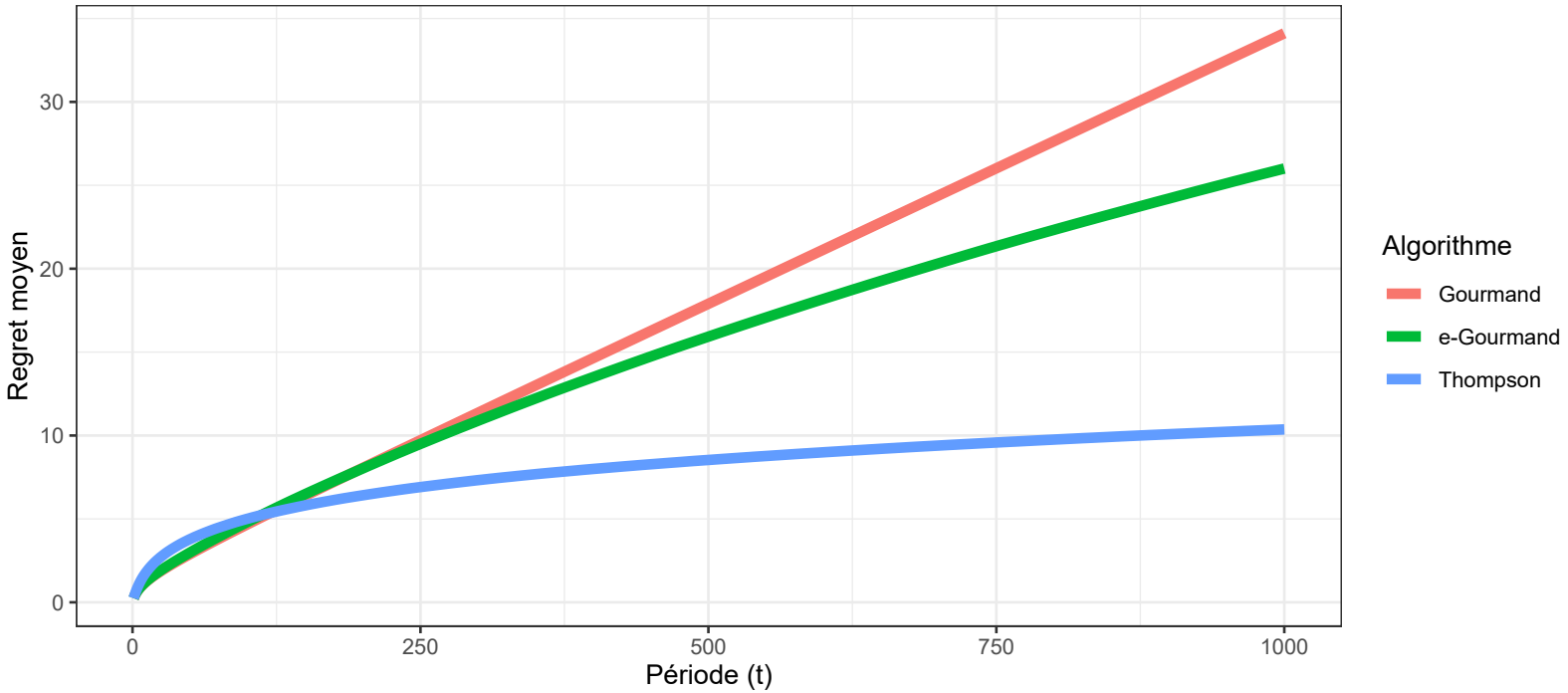

Figure – Regret moyen des trois algorithmes ($\epsilon = 0.01$)

Illustration

L'échantillonnage de Thompson explore intelligemment puisqu'il choisit le bras vert, bleu et rouge avec probabilité 0, 0.67 et 0.33. De plus, l'exploration décroît avec le temps.

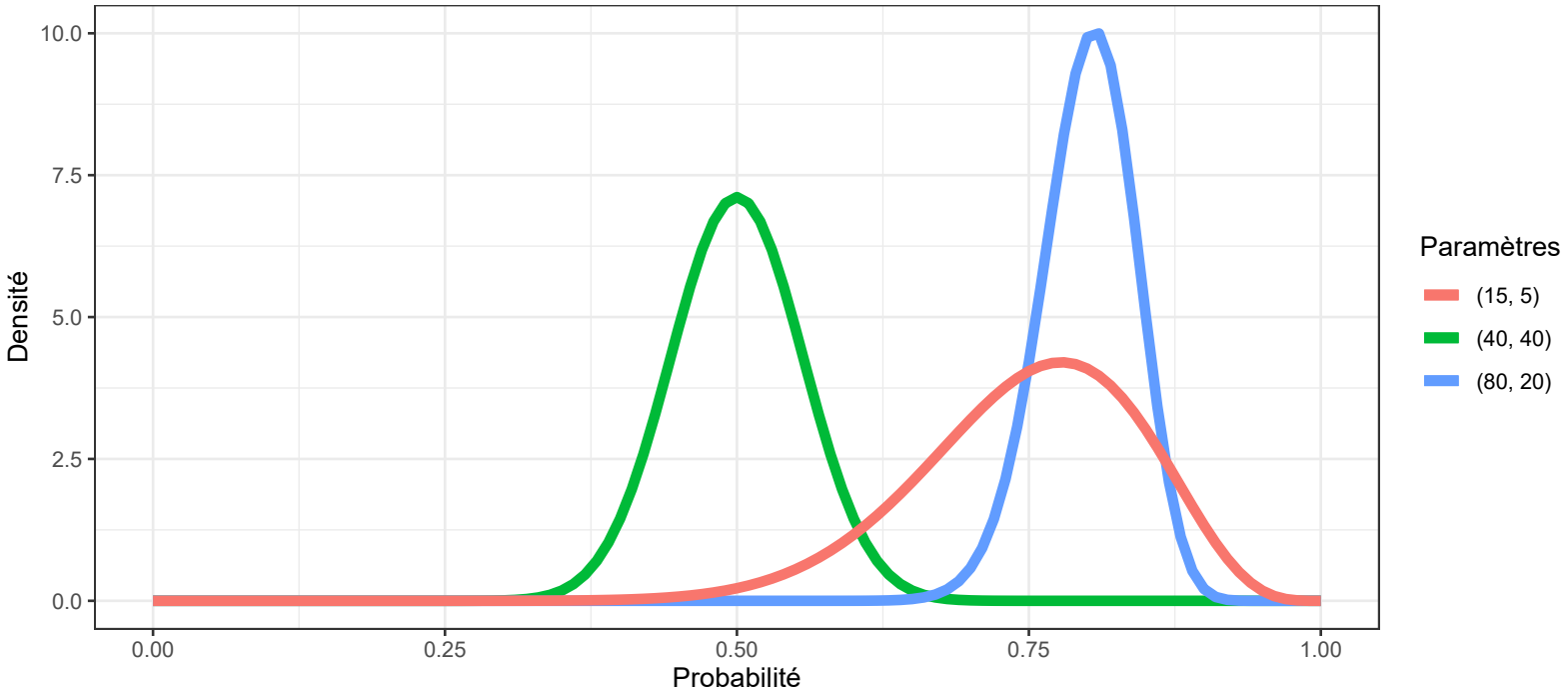

Figure – Distribution *a priori* de trois bras

Conclusion

Conclusion

- L'algorithme gourmand est simple, mais a tendance à exploiter un même bras, qu'il soit optimal ou non. Il n'explore pas suffisamment les autres bras.

- L'algorithme ϵ-gourmand conserve une simplicité tout en explorant davantage, ce qui le rend plus efficace que l'algorithme gourmand. Toutefois, son exploration reste constante et peu intelligente.

- L'échantillonnage de Thompson est le plus performant de ces trois algorithmes ; son exploration est intelligente et adaptée à l'information acquise sur les bras. Toutefois, le fait d'échantillonner le rend plus complexe, et donc plus long à exécuter.

- On obtient les mêmes résultats avec un nombre de bras K.

Références

- Russo, D. *et al.* (2017). *A Tutorial on Thompson Sampling*. https://arxiv.org/abs/1707.02038

- Slivkins, A. (2019). *Introduction to Multi-Armed Bandits*. https://arxiv.org/abs/1904.07272