INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Bureau d'étude :

Algorithmes de bandits

Auteurs :
Paul CHARNAY
Louis DELVAUX

Groupe :
3MIC B

Encadrant :
Valentin DEBARNOT

Année universitaire 2017-2018

Table des matières

Introduction

Nous nous sommes intéressés dans ce projet à la modélisation des algorithmes de bandits. Ces algorithmes permettent de faire des choix à partir d'observations passées pour pouvoir maximiser ou minimiser une quantité à laquelle on s'intéresse.

Nous avons tout d'abord considéré le cas des machines à sous, l'objectif étant de déterminer la meilleure stratégie à adopter pour obtenir un gain maximum. Nous avons ensuite appliqué nos stratégies à un problème médical. Dans ce problème, nous considérons différents traitements et le but est de déterminer le meilleur traitement après un certain nombre d'essais cliniques.

Les différentes fonctions que nous avons utilisées tout au long de ce projet ont été implémentées en langage Python et sont disponibles ici.

Partie 1

Machine à sous

1.1 Présentation du problème

Nous considérons un joueur qui rentre dans un casino pour jouer aux machines à sous. Il se présente devant K machines et à chaque fois qu'il joue sur une machine, il récupère une certaine somme. La différence entre la somme mise pour jouer sur la machine et la somme obtenue sur cette machine est appelée gain. Le gain est une valeur comprise entre $a < 0$ et $b > 0$. On modélise donc le gain comme une variable aléatoire à valeur dans $[\![a, b]\!]$.

Nous numérotons les machines de 1 à K. Lorsque l'on "tire le bras k", c'est-à-dire lorsque l'on actionne la machine k ($1 \leq k \leq K$), nous obtenons un certain gain que l'on représente par la variable aléatoire X_k et qui suit une loi notée ν_k.

Nous avons étudié plusieurs possibilités pour modéliser cette loi :

- Loi normale : peu réaliste car c'est une loi continue et non bornée, même si les valeurs extrêmes sont peu probables. Elle est cependant pratique à manipuler car beaucoup de résultats théoriques sont connus.
- Loi uniforme : peu réaliste car elle est continue et les valeurs extrêmes devraient être plus rares que les petites valeurs.
- Loi binomiale : intéressante car elle est discrète et bornée.
- Loi de Poisson : c'est une loi bornée à gauche, ce qui est réaliste car on ne peut pas perdre plus que la mise de départ. De plus, elle possède une faible probabilité de résultats très élevés et une forte probabilité de résultats plutôt faibles.

Nous avons finalement opté pour la loi de Poisson à laquelle nous avons soustrait une certaine constante afin de modéliser la somme du jeton utilisé pour lancer la machine.

1.2 Un outil de comparaison des performances : le regret

Afin de pouvoir comparer les performances des différents algorithmes que nous allons implémenter, on définit le regret cumulé. Ce regret, noté R_T, est la différence entre le gain moyen du meilleur bras que l'on pouvait jouer et le gain moyen du bras que l'on a joué. En notant μ_{k^*} le gain du meilleur bras, on le calcule de la façon suivante :

$$\begin{aligned} R_T &= \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left(\mu_{k^*} - X_{k(t)}\right) \\ &= \frac{1}{T}\sum_{t=1}^{T}\left(\mu_{k^*} - \mu_{k(t)}\right) \end{aligned}$$

Dans toute cette première partie, nos algorithmes retourneront un vecteur correspondant au regret en fonction du nombre de jetons joués, ce qui permettra de les comparer. Afin d'obtenir des courbes plus intéressantes à exploiter, nous moyennons toujours notre regret par le temps final T plutôt que par le temps courant t. Ainsi, le regret ne diminue jamais et on évite un effet d'oscillations et de pics sur les premières itérations. De plus, cela nous permet de déterminer directement le meilleur algorithme lorsque nous observons la valeur du regret à la dernière itération.

1.3 Génération de données

Nous avons donc décidé de modéliser nos gains par une loi de Poisson modifiée, c'est-à-dire que pour tout $1 \leq k \leq K$, on a $X_k \sim \nu_k = \mathcal{P}(\mu_k + J) - J$ avec J le coût d'un jeton. Pour le joueur, l'objectif est de déterminer la machine avec la moyenne μ_k la plus élevée, avec le nombre de tentatives le plus faible possible.

Pour nos tests numériques, nous avons choisi de considérer $K = 5$ machines, avec pour moyennes : $\mu = [\mu_1 = 0.2, \mu_2 = 1.2, \mu_3 = 1.5, \mu_4 = 0.7, \mu_5 = 1]$ et de fixer le coût des jetons à $J = 2$.

La fonction permettant de renvoyer le gain du joueur en fonction du numéro de la machine jouée est donnée ci-dessous. Elle renvoie une réalisation aléatoire x_k du gain pour le bras choisi, l'espérance du bras choisi $\mu[k]$ et également l'espérance du gain maximal utile au calcul du regret.

Algorithm 1 Gain

Require: $k \in [\![1; K]\!], J$
$\mu \leftarrow [\mu_1, .., \mu_K]$
return x_k, $\mu[k]$, $\max(\mu)$

1.4 Choix de l'estimateur de la moyenne

Afin de trouver quelle machine a la moyenne μ_k la plus élevée, il nous faut estimer celle-ci. Pour cela, nous allons déterminer l'estimateur du maximum de vraisemblance $\hat{\mu}_k$ de la moyenne de la machine k.

Considérons un échantillon de n réalisations $x_k^1, ..., x_k^n$ indépendantes et identiquement distribuées de la variables X_k. Comme X_k suit une loi de Poisson $\mathcal{P}(\theta)$, où $\theta = \mu_k$, sa densité de probabilité est : $f_X(x_k) = \mathbb{P}(X = x_k) = \frac{e^{-\theta}\theta^{x_k}}{x_k!}$.

La vraisemblance de cet échantillon vaut alors :

$$L(x_k^1, ..., x_k^n; \theta) = \prod_{i=1}^{n} f_X(x_k^i) = \prod_{i=1}^{n} \frac{e^{-\theta}\theta^{x_k^i}}{x_k^i!} = e^{-n\theta} \frac{\theta^{\sum_{i=1}^n x_k^i}}{\prod_{i=1}^n x_k^i!}$$

En passant à la logvraisemblance, on obtient :

$$\ell(x_k^1, ..., x_k^n; \theta) = \ln(L(x_k^1, ..., x_k^n; \theta)) = -n\theta + \ln(\theta) \sum_{i=1}^{n} x_k^i - \sum_{i=1}^{n} \ln(x_k^i!)$$

Pour trouver l'estimateur du maximum de vraisemblance, il faut maximiser la logvraisemblance :

- $\frac{\partial \ell(\underline{X},\theta)}{\partial \theta} = -n + \frac{1}{\theta}\sum_{i=1}^n x_k^i$ d'où $\frac{\partial \ell(\underline{X},\theta)}{\partial \theta} = 0 \iff \theta = \frac{1}{n}\sum_{i=1}^n x_k^i = \overline{x_k}$
- $\frac{\partial^2 \ell(\underline{X},\theta)}{\partial \theta^2} = -\frac{1}{\theta^2}\sum_{i=1}^n x_k^i \leq 0$ car $\forall i, X_k^i \sim \mathcal{P}(\theta)$ et donc $x_k^i \geq 0$

Ainsi, $\hat{\theta}_{MV} = \overline{x_k} = \frac{1}{n}\sum_{i=1}^n x_k^i$ est le maximum de vraisemblance de $\theta = \mu_k$.

Calculons le biais de notre estimateur :

$$\mathbb{E}\left[\hat{\theta}\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} x_k^i\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[x_k^i\right] = \frac{1}{n}\sum_{i=1}^{n} \theta = \theta$$

La moyenne empirique $\overline{x_k}$ est donc un estimateur sans biais de μ_k. Il est donc cohérent de la choisir pour estimer nos moyennes dans nos algorithmes.

1.5 Premier algorithme

Pour ce premier algorithme de bandit, nous utilisons une stratégie simple découpée en deux phases : l'exploration et l'exploitation. Si nous disposons au total de T jetons, nous décidons d'en utiliser T_1 durant la phase d'exploration pour déterminer laquelle des K machines est la plus intéressante à jouer. A l'issue de ces T_1 lancers, nous jouons tous les jetons restants dans la machine sur laquelle nous avons constaté la meilleure moyenne empirique. L'algorithme utilisé est le suivant :

Algorithm 2 Stratégie 1

Require: T, T_1, K
  $n = T_1/K$
  for k in range $[\![1, K]\!]$ **do**
    for i in range $[\![1, n]\!]$ **do**
      Tirer le bras k et noter le regret associé.
    end for
  end for
  $k = \text{argmax}(gain)$
  for i in range $[\![T_1 + 1, T]\!]$ **do**
    Tirer le bras k et noter le regret associé.
  end for
  return regrets

Après avoir expérimenté différentes valeurs, nous nous rendons compte que le T_1 optimal est environ 180 dans notre cas. Nous avons alors une courbe de regret qui se décompose en deux parties : pendant la phase d'exploration, elle est composée de K segments de droite de pente respective $\mu_i - \mu^*$. Ensuite le regret augmente de manière linéaire avec une pente qui dépend de la qualité du choix de machine qui a été fait à la fin de l'exploration. On se rend compte que si le T_1 choisi est trop petit, nous choisissons souvent la mauvaise machine et le regret augmente plus vite durant l'exploitation, alors que si T_1 est trop élevé l'exploration générera un regret important.

Pour réduire l'importance du choix de T_1, nous souhaitons pouvoir contrôler plus finement la sortie de la phase d'exploration. Nous allons donc maintenant nous servir d'intervalles de confiance de la moyenne du gain de chaque machine pour décider si nous passons en phase d'exploitation ou pas.

1.6 Amélioration avec les intervalles de confiance

Le principe général de cette nouvelle version est le suivant :

- On commence par jouer quelques jetons dans chaque machine pour ne pas réaliser des intervalles de confiance sur trop peu de valeurs. Dans notre cas, ce nombre est compris entre 5 et 10. C'est beaucoup plus avantageux que les 180 jetons utilisés dans la première stratégie. On calcule alors un intervalle de confiance du gain pour chaque machine.
- On continue à jouer chaque machine jusqu'à ce que la borne supérieure de son intervalle de confiance soit inférieure aux bornes inférieures de tous les autres intervalles, on arrête alors de la jouer car on considère qu'elle ne pourra pas être la plus avantageuse. On vérifie aussi si un des intervalles est supérieur aux autres, c'est-à-dire si sa borne inférieure est supérieure à toutes les autres bornes supérieures. On arrête alors l'exploration et on joue uniquement cette machine.

Cette stratégie produit des regrets beaucoup plus faibles que la précédente car l'exploration dure moins longtemps et car le choix fait au terme de cette phase est généralement de meilleure qualité.

Les paramètres à ajouter sont le coefficient α pour les intervalles de confiance et n le nombre de jetons à jouer en exploration initiale. Contrairement aux intervalles de confiance réalisés habituellement, il n'est pas souhaitable de choisir un α très petit, car on a alors des intervalles de confiance plus larges qui font durer l'exploration trop longtemps. Dans notre cas, les meilleurs résultats sont atteints pour $\alpha = 0.3$ et $n = 8$.

Cependant, l'implémentation de cet algorithme est difficile avec la loi de Poisson que nous avons choisie car il n'y a pas de forme simple de l'intervalle de confiance et l'approximation gaussienne n'est pas applicable sur le petit nombre de données que nous avons en sortie de la première phase d'exploration ($n = 8$). Nous réaliserons et testerons donc cet algorithme directement dans la deuxième partie.

Algorithm 3 Stratégie 2

Require: $T, K, alpha, n$
  $J = 0$ {J est le nombre de jetons joués}
  while $J \leq n * K$ **do**
    for i in range $[\![1, n]\!]$ **do**
      Tirer le bras k et noter le gain obtenu et le regret associé.
      $J = J + 1$
    end for
  end while
  Calculer l'intervalle de confiance pour chaque machine.
  $trouve = False$
  while $J \leq T$ and not $trouve$ **do**
    for k in range $[\![1, K]\!]$ **do**
      if intervalle k pas inférieur aux autres **then**
        if intervalle k supérieur aux autres **then**
          $meilleur = k$
          $trouve = True$
        else
          $J = J + 1$
          Jouer le bras k, noter le regret, mettre à jour l'intervalle de confiance.
        end if
      end if
    end for
    if Plus qu'un seul intervalle i à tester **then**
      $meilleur = i$
      $trouve = True$
    end if
  end while
  for i in range $[\![J + 1, T]\!]$ **do**
    Jouer la machine d'indice $meilleur$, noter le regret associé.
  end for
  return regrets

1.7 Algorithme ϵ-greedy

Contrairement aux algorithmes précédents, nous allons dans celui-ci joindre les phases d'exploration et d'exploitation. Cela va permettre de corriger les erreurs qui peuvent subvenir lorsque la phase d'exploration est trop petite.

Pour l'algorithme ϵ-greedy, nous choisissons une valeur de $\epsilon \in]0, 1[$ assez petite de telle sorte que :

- on a une probabilité $1 - \epsilon$ de choisir le bras qui possède la meilleure moyenne empirique
- on a une probabilité ϵ de choisir uniformément un bras quelconque

Pour pouvoir déterminer quel bras possède la meilleure moyenne empirique pour le premier lancer, nous avons choisi une phase d'initialisation qui consiste à jouer une fois chaque bras. L'algorithme est le suivant :

Algorithm 4 ϵ-greedy

Require: T, ϵ, K
  for k in range $[\![1, K]\!]$ **do**
    {Initialisation} Tirer le bras k et noter le regret associé.
  end for
  for t in range $[\![K, T]\!]$ **do**
    if RANDOM(0,1) $<$ 1 - ϵ **then**
      Tirer le bras avec la meilleure moyenne empirique. {Si 2 bras ont la meilleure moyenne, choisir le bras le plus joué}
    else
      Tirer un bras aléatoirement avec une probabilité uniforme.
    end if
    Noter le regret et mettre à jour la moyenne empirique du bras joué.
  end for
  return regrets

Dans cet algorithme, nous pouvons voir que nous avons besoin de recalculer la moyenne empirique $\overline{x_k}$ du bras k que l'on vient de jouer. Or, en notant n le nombre de fois où ce bras a été joué et $\overline{x_n}$ la moyenne empirique de ce bras après n essais, la moyenne empirique après $n + 1$ essais est :

$$\begin{aligned}\overline{x_{n+1}} = \frac{1}{n+1}\sum_{i=1}^{n+1} x_i &= \frac{1}{n+1}x_{n+1} + \frac{1}{n+1}\sum_{i=1}^{n} x_i \\ &= \frac{1}{n+1}x_{n+1} + \frac{n}{n+1}\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) = \frac{1}{n+1}x_{n+1} + \frac{n}{n+1}\overline{x_n}\end{aligned}$$

Ainsi, cette formule récursive nous permet de mettre à jour la moyenne empirique du bras joué plus rapidement qu'avec l'utilisation de la formule classique de la moyenne. Elle apporte un gain de temps et plus d'efficacité à notre algorithme.

De plus, dans le cas où deux bras possèdent la même moyenne empirique, nous avons choisi de sélectionner celui qui a été le plus joué parmi les deux car dans ce cas, la moyenne empirique du bras choisi est plus précise et nous possédons ainsi plus de certitude sur la vraie valeur de la moyenne de ce bras. Il est aussi plus susceptible d'être le meilleur bras.

1.8 Algorithme ϵ-greedy dépendant du temps

Il est possible d'améliorer l'algorithme ϵ-greedy en faisant décroître ϵ en fonction du temps. Ainsi, en divisant ϵ par une fonction du temps $f(t)$ dans l'algorithme, nous obtenons globalement des résultats plus satisfaisants. En comparant les algorithmes pour $f(t) = 1/t$, $f(t) = 1/t^2$ et $f(t) = 1/\log(t)$, nous avons pu observer les plus faibles regrets pour $1/t^2$.

1.9 Algorithme UCB

L'algorithme UCB (Upper Confidence Bound) permet de calculer une borne supérieure de confiance pour chaque bras et de jouer le bras avec la plus haute borne. Cet algorithme doit être utilisé pour des gains à valeurs bornées. Dans notre cas, la loi de Poisson n'est pas bornée à droite mais étant donné la faible probabilité d'obtenir des résultats très élevés, nous l'avons considérée en tant que telle, avec des bornes ayant pour valeurs $a = 0$ et $b = 15$. A chaque fois qu'un des bras est joué, chaque borne supérieure doit être recalculée. Pour un bras, elle vaut :

$$B_k = \hat{\mu_k} + (b - a)\sqrt{\frac{2log(1/0.95)}{s(k,t)}}$$

où $\hat{\mu_k}$ est la moyenne empirique du bras k que l'on calcule avec la formule récursive vue précédemment et $s(k,t)$ est le nombre de fois où on a joué le bras k.

Comme pour l'algorithme ϵ-greedy, il est nécessaire d'effectuer une phase d'initialisation pour obtenir une première valeur des bornes supérieures. Nous allons donc jouer une fois chaque bras pour permettre ces calculs.

Algorithm 5 UCB

Require: T, K, a, b
  for k in range $[\![1, K]\!]$ **do**
    {Initialisation} Tirer le bras k et noter le regret associé.
  end for
  for t in range $[\![K+1, T]\!]$ **do**
    for k in range $[\![1, K]\!]$ **do**
      Mettre à jour la borne supérieure de chaque bras.
    end for
    Tirer le bras avec la plus grande borne supérieure et noter le regret associé.
  end for
  return regrets

1.10 Algorithme Exp3

L'algorithme Exp3 est un algorithme de bandit adversarial, c'est-à-dire qu'il consiste à supposer que les gains sur chaque bras, en plus d'être aléatoires, dépendent de ce que le joueur a déjà joué. Il considère donc que les machines à sous sont autorisées à changer de moyenne à chaque instant. Dans notre cas, cela reviendrait à changer la valeur moyenne μ_k du gain d'une machine, après avoir été jouée.

Du fait que l'on ait considéré des moyennes fixes pour nos machines, l'algorithme adversarial Exp3 n'est pas utilisé dans le bon cadre. Nous ne nous attendons donc pas à trouver de très bon résultats avec cet algorithme mais il est tout de même intéressant à implémenter vu qu'il ne possède pas d'argument spécifique et n'a pas besoin d'initialisation.

Algorithm 6 Exp3

Require: T, K
  $\gamma = \sqrt{\frac{Klog(K)}{T}}$; $\eta = \sqrt{\frac{2log(K)}{TK}}$
  for t in range $[\![1, T]\!]$ **do**
    Tirer un bras selon la loi p_t {Uniforme à t=1}
    Noter le regret associé
    Calculer le gain estimé k : $\tilde{X}_{k,t} = \frac{X_{k,t}}{p_{k,t}}$ pour le bras k tiré
    Mettre à jour la somme des gains estimés $\tilde{G}_{k,t} = \sum_{s=1}^{t} \tilde{X}_{k,s}$ pour le bras k tiré
    Mettre à jour la loi $p_{k,t+1} = (1-\gamma)\frac{exp(\eta \tilde{G}_{k,t})}{\sum_{k=1}^{K} exp(\eta \tilde{G}_{k,t})} + \frac{\gamma}{K}$ pour chaque bras $1 \leq k \leq K$
  end for
  return regrets

1.11 Comparaison des stratégies

Pour comparer nos algorithmes, nous traçons leurs regrets cumulés moyennés sur $N = 1000$ expériences. Ainsi, nos courbes représentent le profil général du regret de nos stratégies et ne dépendent donc plus de paramètres aléatoires. En effet, pour de faibles valeurs de N, l'algorithme le plus efficace peut être différent suivant les gains obtenus. Par exemple, pour $N = 1$ expérience, suivant la machine choisie après la phase d'exploration, la Stratégie 1 peut être la plus efficace ou la pire. Avec $N = 1000$, nous nous assurons donc de la performance de nos algorithmes.

Nous obtenons les courbes suivantes :

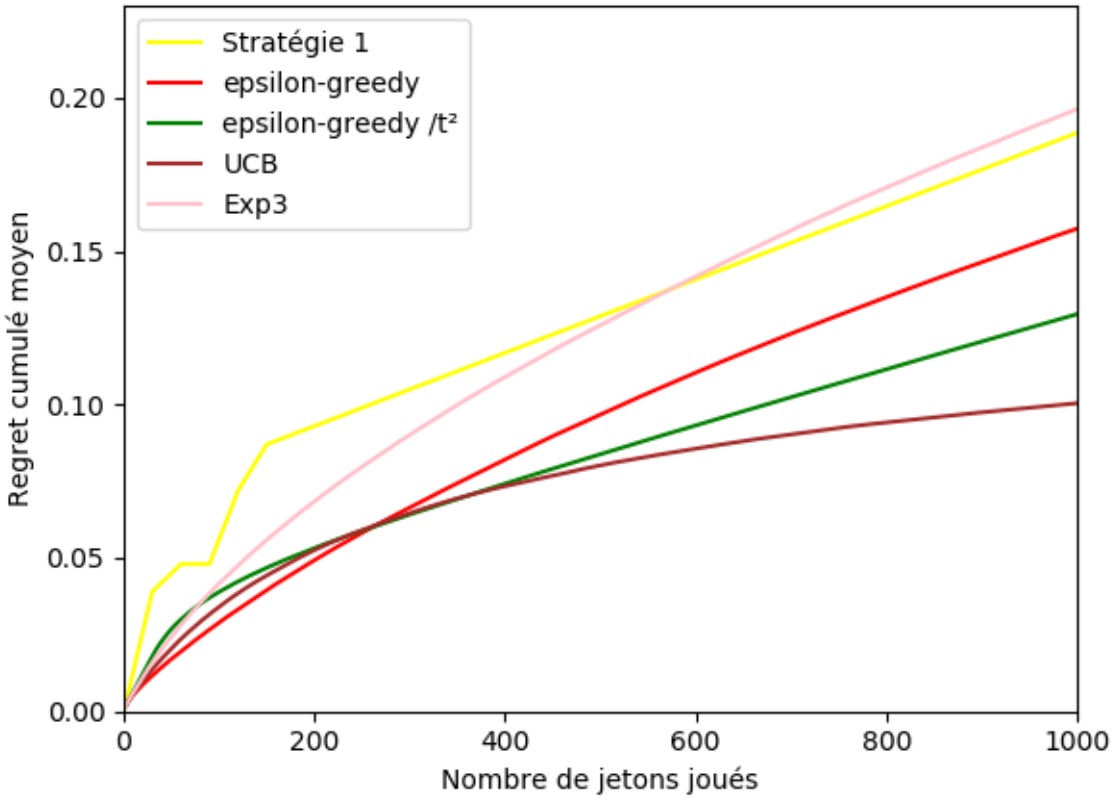


FIGURE 1.1 – Regret moyen des différentes stratégies sur N=1000 expériences

Pour la Stratégie 1, nous observons bien les deux phases : l'exploration sur les différentes machines et l'exploitation sur la machine choisie qui donne un regret augmentant linéairement.
L'algorithme ϵ-greedy donne un regret également assez linéaire mais est tout de même plus

performant que la Stratégie 1. De plus, nous pouvons remarquer que pour un faible nombre de jetons joués, il est le plus performant.
L'algorithme ϵ-greedy dépendant du temps (de t^2 dans notre cas) a un plus grand regret que ϵ-greedy pour un faible nombre de jetons joués mais devient plus efficace au cours du temps (après environ 270 jetons joués).
La stratégie la plus performante pour 1000 jetons joués est obtenue avec l'algorithme UCB. Comme attendu, l'algorithme Exp3 est le moins performant pour 1000 jetons joués. Cependant, il reste plus performant que l'algorithme Stratégie 1 jusqu'à environ 570 jetons joués. Cela est dû au fort regret engendré par la phase d'exploration de la Stratégie 1.

Partie 2

Essais cliniques

2.1 Présentation du problème

Les algorithmes de bandits que nous avons développés dans la première partie peuvent être utilisés pour d'autres applications que les gains de casino. Par exemple, la première application de ces algorithmes concerne le choix de traitements médicamenteux. Le but est alors de savoir parmi plusieurs médicaments lequel est le plus efficace pour vaincre une maladie en faisant le moins de tests possible. Nous allons nous placer dans le cadre d'un laboratoire qui a mis au point plusieurs traitements et qui réalise des tests pour savoir lequel commercialiser.

2.2 Modélisation du problème

Dans ce nouveau problème, nous modélisons l'issue des tirages par des lois de Bernoulli, qui prennent la valeur 1 avec une probabilité $p \in]0,1[$ fixée pour chaque médicament si le traitement guérit le patient et 0 sinon. Nous cherchons à maximiser le nombre de 1, c'est-à-dire à trouver le traitement qui a la plus grande probabilité de guérir les patients. En effet, ce choix semble être le plus logique si l'on considère que notre laboratoire possède une éthique et cherche à soigner ses patients plutôt qu'à les garder malades pour maximiser son profit...

2.3 Choix de l'estimateur de p

Comme dans la première partie, nous devons estimer notre paramètre d'intérêt, ici p_k, pour chacun des médicaments. Déterminons alors le maximum de vraisemblance $\hat{p}_k$ de la

probabilité de guérison des patients par le médicament k.

Considérons un échantillon de n réalisations $x_k^1, ...x_k^n$ indépendantes et identiquement distribuées de la variables X_k. Comme X_k suit une loi de Bernoulli $\mathcal{B}(p_k)$, sa densité de probabilité est : $f_X(x_k) = \mathbb{P}(X = x_k) = p_k^{x_k}(1-p_k)^{1-x_k}$.

La vraisemblance de cet échantillon vaut alors :

$$\begin{aligned} L(x_k^1, ..., x_k^n; p_k) &= \prod_{i=1}^{n} f_X(x_k^i) = \prod_{i=1}^{n} p_k^{x_k^i}(1-p_k)^{1-x_k^i} \\ &= p_k^{\sum_{i=1}^n x_k^i}(1-p_k)^{\sum_{i=1}^n 1-x_k^i} = p_k^{\sum_{i=1}^n x_k^i}(1-p_k)^{n-\sum_{i=1}^n x_k^i} \end{aligned}$$

En passant à la logvraisemblance, nous obtenons :

$$\ell(x_k^1, ..., x_k^n; p_k) = \ln(L(x_k^1, ..., x_k^n; p_k)) = \ln(p_k)\left(\sum_{i=1}^{n} x_k^i\right) + \ln(1-p_k)\left(n - \sum_{i=1}^{n} x_k^i\right)$$

Pour trouver l'estimateur du maximum de vraisemblance, il faut maximiser la logvraisemblance :

- $\frac{\partial \ell(\underline{X}, p_k)}{\partial p_k} = \frac{\sum_{i=1}^n x_k^i}{p_k} - \frac{n-\sum_{i=1}^n x_k^i}{1-p_k}$ d'où $\frac{\partial \ell(\underline{X}, p_k)}{\partial p_k} = 0 \iff p_k = \frac{1}{n}\sum_{i=1}^n x_k^i = \overline{x_k}$
- $\frac{\partial^2 \ell(\underline{X}, p_k)}{\partial p_k^2} = -\frac{\sum_{i=1}^n x_k^i}{p_k^2} - \frac{n-\sum_{i=1}^n x_k^i}{(1-p_k)^2} \leq 0$ car $\forall i, x_k^i = 0$ ou 1 et donc $0 \leq \sum_{i=1}^n x_k^i \leq n$

Ainsi, $\hat{p}_k = \overline{x_k} = \frac{1}{n}\sum_{i=1}^n x_k^i$ est le maximum de vraisemblance de notre paramètre p_k.

Calculons le biais de notre estimateur :

$$\mathbb{E}\left[\hat{p}_k\right] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} x_k^i\right] = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}\left[x_k^i\right] = \frac{1}{n}\sum_{i=1}^{n} p_k = p_k$$

La moyenne empirique $\overline{x_k}$ est donc un estimateur sans biais de p_k. Il est donc cohérent de la choisir pour estimer nos probabilités de guérison de nos médicaments.

2.4 Adaptation de nos algorithmes

Comme l'estimateur de notre paramètre d'intérêt est le même que dans la première partie, nous pouvons alors facilement adapter nos algorithmes en remplaçant la loi de Poisson par une loi de Bernoulli, les machines à sous par les médicaments et les jetons par les essais sur les patients.

Nous modifions également nos algorithmes pour renvoyer le nombre de fois où chaque traitement a été choisi en plus de renvoyer le regret. Afin de savoir quel traitement est le plus efficace, il suffira de regarder lequel est le plus souvent choisi par l'algorithme.

Comme annoncé dans la première partie, nous avons également implémenté en plus l'algorithme Stratégie 2, la version améliorée de Stratégie 1.

De plus, pour nos tests dans cette partie, nous avons choisi de considérer les probabilités de guérison suivantes pour nos médicaments : $p = [p_1 = 0.2, p_2 = 0.75, p_3 = 0.6, p_4 = 0.8, p_5 = 0.4]$. Ainsi, nous observerons également la capacité de nos algorithmes à choisir entre des médicaments ayant des probabilités de guérison fortes et surtout proches.

2.5 Comparaison de nos stratégies

Pour les mêmes raisons que celles citées dans la première partie, nous calculons nos regrets moyens pour $N = 1000$ expériences. Les courbes de regret obtenues sont les suivantes :

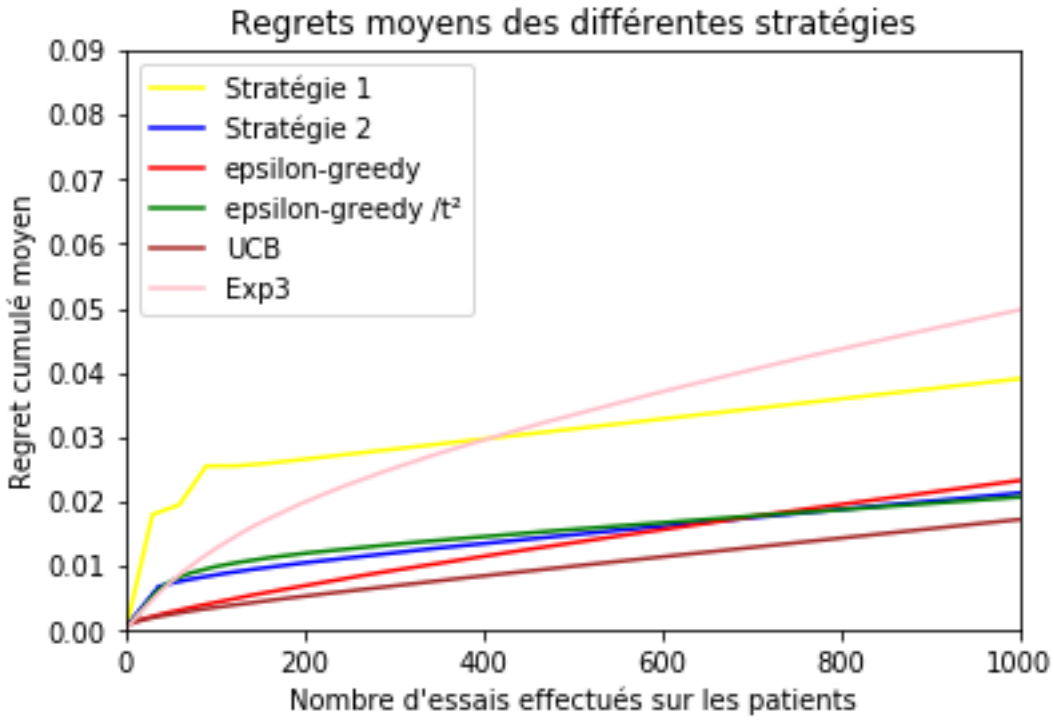
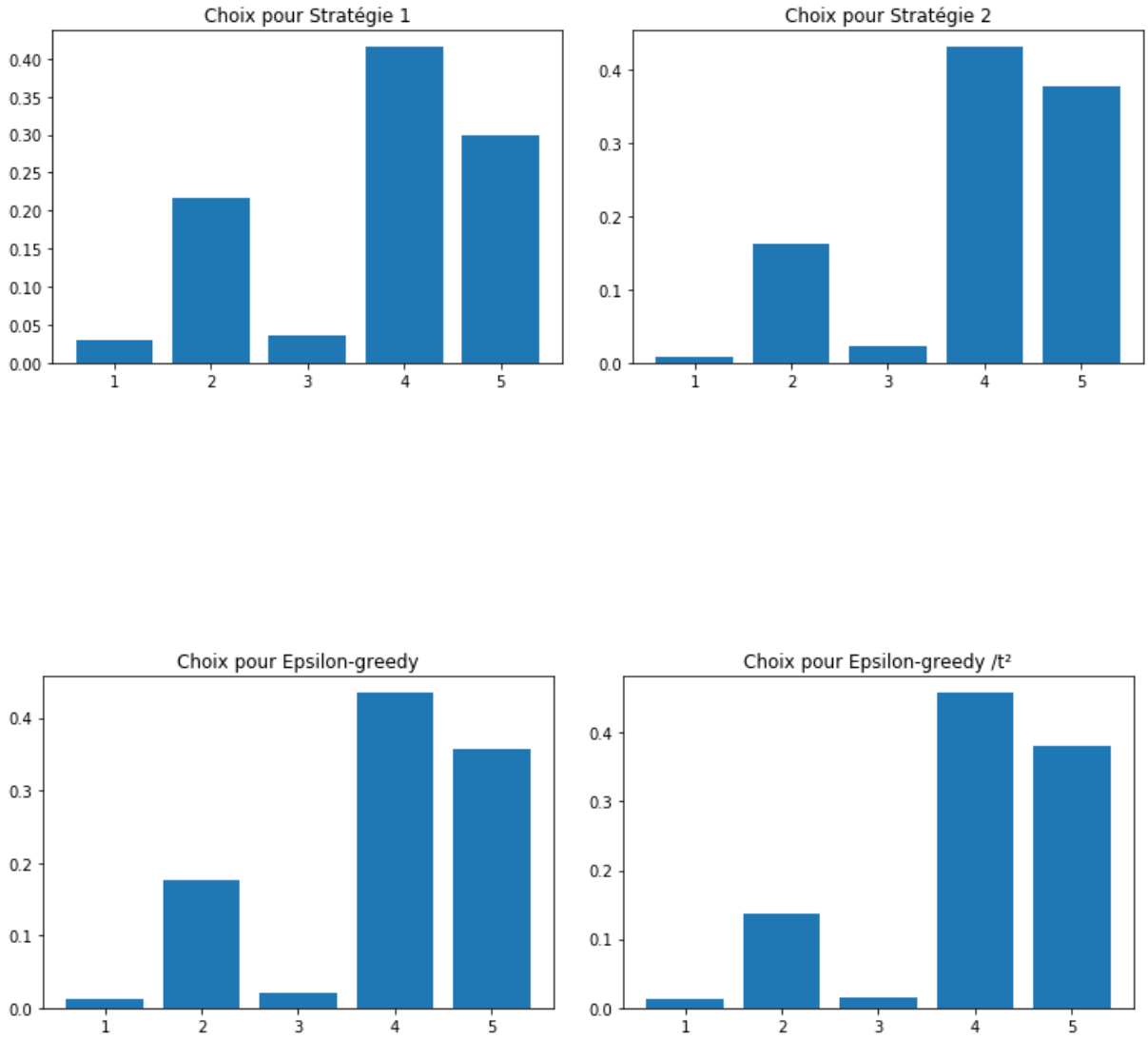


FIGURE 2.1 – Regret moyen des différentes stratégies sur N=1000 expériences

L'algorithme Exp3 n'étant toujours pas dans le bon cadre, son regret reste très important dans ce cas. L'algorithme Stratégie 1 reste également peu performant ici car la phase

d'exploration génère toujours beaucoup de regret. L'algorithme ϵ-greedy garde la même forme de courbe mais celui dépendant de t^2 devient plus performant. En effet, le regret augmente peu lorsque l'algorithme choisit un médicament ayant une probabilité proche du meilleur. Ainsi, son regret tend à se stabiliser. L'algorithme UCB devient moins performant que l'algorithme ϵ-greedy dépendant de t^2. En effet, il possède un regret qui augmente plus linéairement même s'il apparaît plus efficace pour un faible nombre d'essais sur les patients. Enfin, l'algorithme Stratégie 2 est l'algorithme qui génère le moins de regret après 1000 essais sur les patients. Son exploration étant rapide et efficace, le regret de cette phase est plutôt faible et le médicament qui sera joué par la suite sera soit le meilleur, soit un médicament avec une probabilité proche de celle du meilleur.



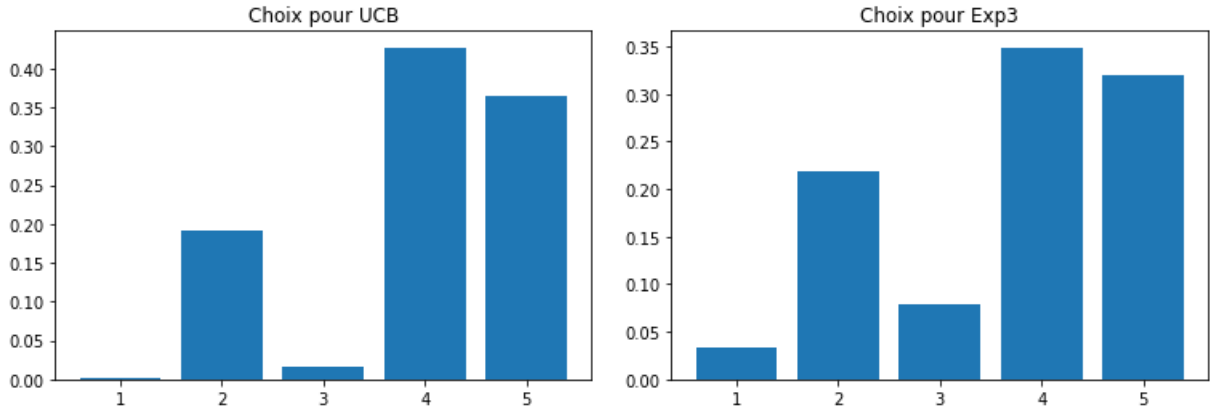


FIGURE 2.2 – Choix moyens pour les différentes stratégies

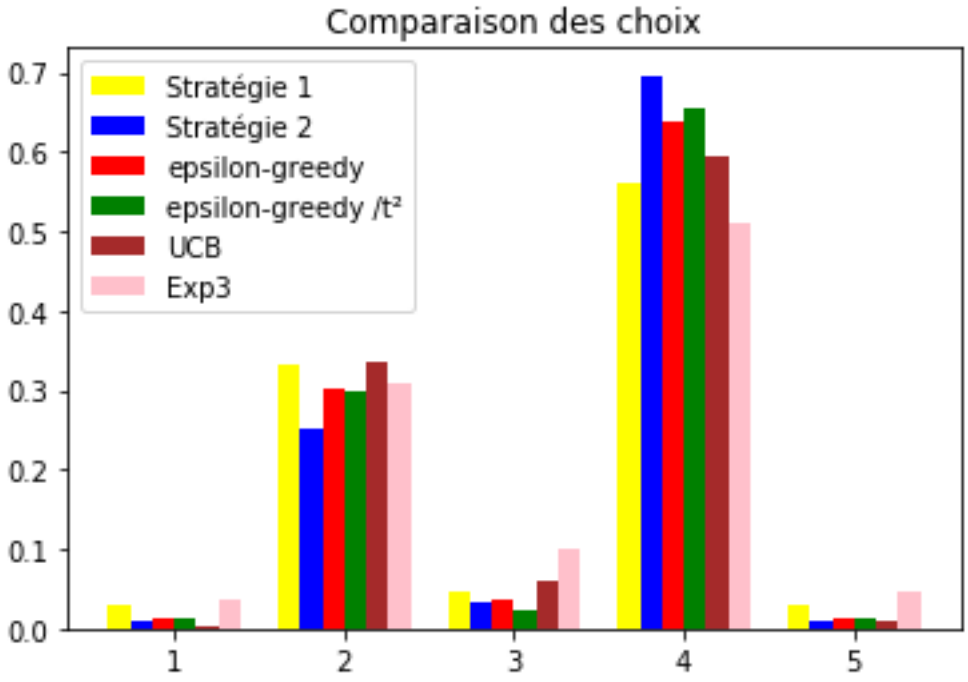


FIGURE 2.3 – Comparaison des choix

Nous pouvons analyser à partir de ces graphiques les forces et les faiblesses de chaque algorithme. Nous voyons que le médicament 1, qui est de loin le moins efficace, n'est presque jamais choisi par UCB, ce qui est une de ses qualités. Le pire algorithme pour ce critère est Exp3. Stratégie 2 réalise aussi de très bonnes performances et les autres algorithmes sont corrects.

Nous pouvons à présent comparer les différents algorithmes entre eux. On voit que globalement, tous les algorithmes parviennent à déterminer que le médicament 4 est le plus

efficace, même si la Stratégie 1 donne des résultats proches pour les médicaments 2 et 4. Ce graphe confirme également que UCB est un algorithme que nous pourrions qualifier de "sûr", car il ne choisit quasiment jamais les deux médicaments les moins efficaces. C'est une caractéristique très intéressante pour une application médicale.

Un autre point intéressant est la capacité de nos algorithmes à discerner le meilleur médicament si deux d'entre eux ont des moyennes théoriques très proches. Pour tester cela, nous modifions la moyenne du médicament 5 et nous la fixons à $p_5 = 0.79$, soit un centième de moins que la meilleure probabilité théorique. Nous obtenons les graphes suivants :

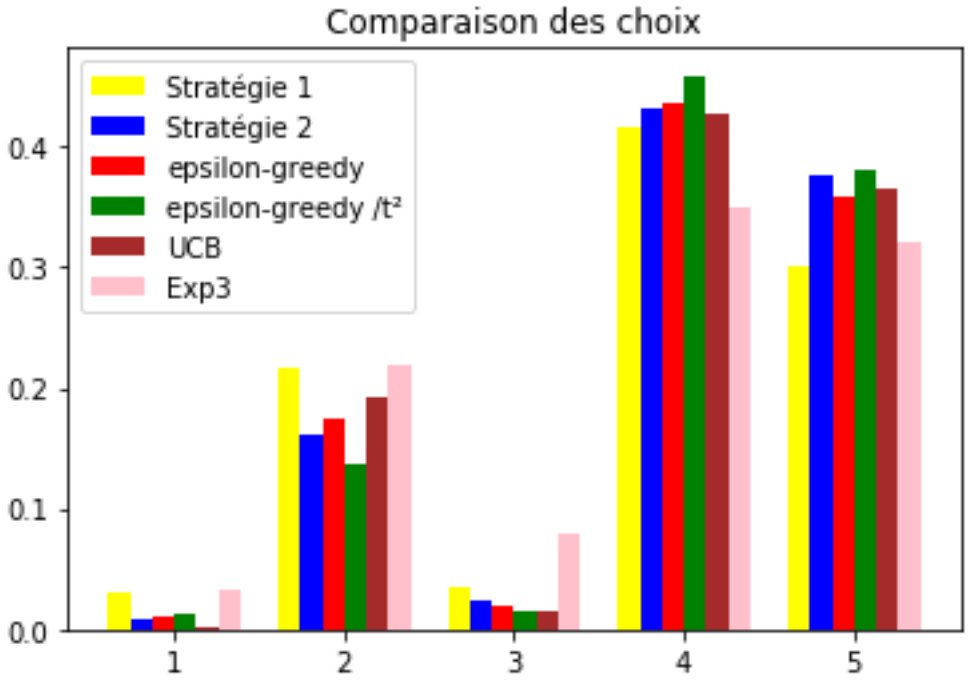


FIGURE 2.4 – Comparaison des choix pour des probabilités proches ($p_4 = 0,8$ et $p_5 = 0,79$)

Nous voyons que tous les algorithmes parviennent encore à trouver que le médicament 4 est plus efficace que le 5. Epsilon-greedy $/t^2$ est le plus efficace pour distinguer deux médicaments d'efficacités très proches, alors que Exp3 n'y parvient quasiment pas.

Conclusion

Nous avons découvert les algorithmes de bandits avec le cas des machines à sous. Cela nous a permis de comprendre et d'implémenter ces algorithmes. Nous avons alors pu observer le fonctionnement de chacun et déterminer leurs performances.

Nous avons ensuite considéré un cas avec plus d'intérêt, celui du choix de traitements médicamenteux. Nous avons ainsi observé quel algorithme est le plus efficace pour déterminer le traitement à adopter face à une maladie. De plus, l'utilité de nos algorithmes a été confirmée par la précision de ceux-ci face au choix à faire entre deux médicaments très efficaces. En effet, dans ce cas, tous nos algorithmes choisissent en moyenne plus souvent le meilleur médicament.

Ces algorithmes de bandits peuvent être utilisés dans de nombreux domaines. En effet, après avoir trouvé la bonne loi pour modéliser un problème, il est assez simple d'adapter les différentes stratégies pour ensuite obtenir les résultats relatifs à la situation. Les algorithmes de bandits sont donc très utiles dans tous les cas nécessitant de faire des choix séquentiels.