Travail Pratique 1 (IFT-4201/IFT-7201)

Présenté à Audrey Durand

Équipe 10 : Adam Cohen, Maxime Genest, Vincent Massé

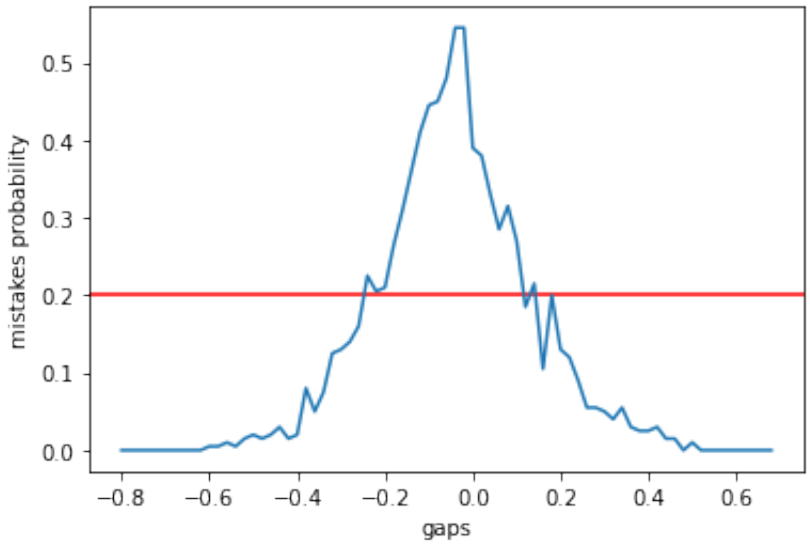
Note : Pour chacun des numéros 1,2,3,4, un notebook identifié explicitement permet de générer les figures.

1 Algorithme de base

1.1 Experience sur l'algorithme ETC

L'algorithme ETC (explore then commit) n'est pas uniformément efficace, c'est-à-dire qu'il souffrira d'un regret linéaire pour les instances plus difficiles. Il va avoir tendance à converger vers des actions sous-optimales due à leurs exploration limitée.
Pour appuyer cela, nous avons réalisé plusieurs expériences avec à chaque fois une instance de two armed bandits. À chaque pas de temps, notre gap variera et nous enregistrerons les performances, c'est-à-dire la probabilité de choisir la mauvaise action.
Nous pouvons voir grâce à la figure 1, que plus le gap va être petit, plus la probabilité d'erreur va être élevée. On peut aussi en déduire que tout gap supérieur à 0.2 positifs ou négatif échouera avec une probabilité d'au moins 20%.
Pour cette expérience, nous avons utilisé des instances de bandit $\mu_1 = 0.1$ et $\mu_2 = 0.9$ en incrémentant μ_1 de 0.01 et en décrémentant μ_2 de 0.01 à chaque itération. Pour chaque itération, l'instance est jouée $N = 200$ et $T = 2m + 1$ pour connaître quel sera le bras choisi comme étant optimal.
En observant le graphique, on peut facilement conclure que les instances de bandits ayant un gap de moins de 0.2 ont une probabilité d'échoué de plus de 20%. Cela implique que pour notre expérience, l'instance avec $\mu_1 = 0.49$ et $\mu_2 = 0.51$ donne une instance avec une grande probabilité d'échec qui est grandement supérieur à 20%, comme nous pouvons le constater sur la figure 1.

FIGURE 1 – Étude de la probabilité d'échoué selon le gap

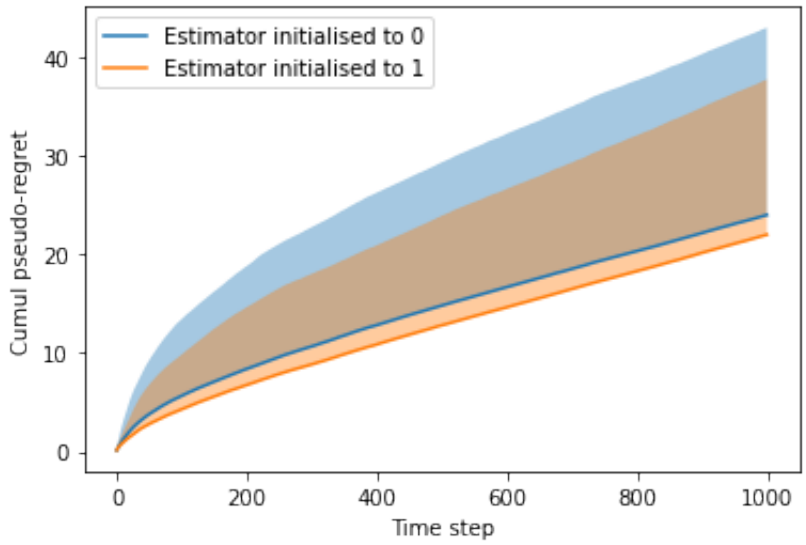


1.2 Optimisation de l'algorithme epsilon-greedy

Pour rendre l'algorithme ϵ-greedy optimiste, nous avons procédé à une expérience comparative sur des bandits à distribution normale.
Dans un premier temps, nous avons généré $N = 500$ instances de two armed bandits avec bras de moyenne et écart-type entre 0 et 1.

Pour l'expérience 1, nous avons initialisé les estimateurs empiriques à 0 et pour l'expérience 2, nous les avons initialisés à 1.
Nous avons ensuite affiché les pseudos regret cumulatif de ses 500 instances dans un graphique. Pour que l'effet soit bien perceptible et non biaisé par l'aléatoire, il est important de prendre un grand nombre d'instances de bandits.
Nous pouvons constater à première vu que le pseudo regret cumulatif moyen de l'expérience 2 est légèrement plus petit. En initialisant les estimateurs empiriques à 1, nous augmentons les chances que les premiers tirages ne biaisent pas le résultat final.

FIGURE 2 – Comparaison de l'algorithme ϵ-greedy optimiste et non optimiste



2 KL UCB

2.1 Pseudo-code

On considère un bandit à K bras sur un horizon de temps de T

- Pour chaque $t \leq K$:
  - $N[t] = N[t] + 1$
  - $S[t] = S[t] + Play(t)$
- Pour chaque $K < t \leq T$:
  - $k_t = argmax(2)$
  - $N[k_t] = N[k_t] + 1$
  - $S[k_t] = S[k_t] + Play(k_t)$

$$d(p,q) = p\ \log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q} \tag{1}$$

$$max\left\{q \in \Theta : N[a]\ d\left(\frac{S[a]}{N[a]},\ q\right) \leq \log(t) + c\ \log(\log(t))\right\} \tag{2}$$

Pour le pseudocode qui précède l'équation 1 est la divergence de Kullback-Leibler et l'équation 2 représente la borne maximale sur l'intervalle de confiance (UCB). N[i] représente le nombre de fois où l'action i à été jouée et S[i] cumule les rewards de l'action i.

2.2 Implementation bernoulli

Pour implementer cet algorithme, on doit utiliser un optimiseur numérique, car il n'existe pas de forme fermée lorsque les récompenses suivent une distrubution de Bernoulli. De plus, l'implémentation inclu un bris d'égalité. Ainsi si 2 bras ont le même UCB et qu'ils représentent les bras avec le plus grand UCB alors l'algorithme choisira le bras ayant le plus petit nombre de jeu. Avec cette approche, on favorise l'exploration en cas d'égalité.

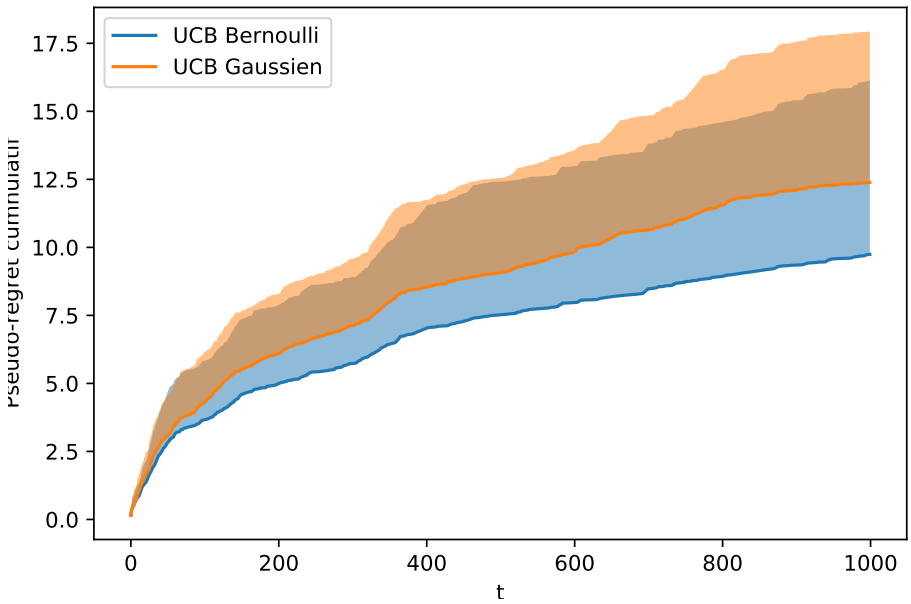
2.3 Implementation UCB Gaussienne

Pour l'implémentation de l'algorithme UCB gaussien il existe une forme fermée tel que vu dans le cours. Cette forme a donc été utilisée dans l'implémentation, car elle rend l'exécution du code beaucoup plus rapide, car il n'a pas d'optimisation à faire pour déterminer l'UCB.
Pour calculer la borne UCB gaussien, l'écart-type doit être connu, mais ce n'est pas le cas pour la configuration de bandit utilisée. L'écart-type maximale d'une distribution de Bernoulli est donc utilisée (0.5).

2.4 Comparaison des modèles

On peut observer que les deux modèles performent bien sur des bandits Bernoulli. Les deux modèles ont un regret cumulatif sous linéaire, mais le modèle utilisant la borne KL UCB pour bandit Bernoulli performe mieux. Cela peut être dut au fait qu'avec le modèle qui utilise la borne KL UCB pour des bandits gaussien utilise une borne avec une variance de 0.25. Cette variance est utilisée, car elle représente la variance maximale dans un bandit Bernoulli ainsi, elle permet de conserver des garanties. Par contre l'utilisation de cette borne cause une sur-exploration. On peut observer l'effet de la sur exploration dans la figure 3. On peut aussi observer que la variance semble un peu plus élevée pour le KL UCB optimisé. Cela peut être du au fait que relativement peu d'instances ont été générées ($N = 200$).

FIGURE 3 – Comparaison entre UCB Bernoulli et UCB Gaussien



3 Algorithme Thompson Sampling (TS)

3.1 Rappel sur la loi de gamma

X est une variable aléatoire suivant une loi gamma ($X \sim \text{Gamma}(\alpha, \beta)$) si sa fonction de densité est

$$f(x; \alpha, \beta) = \begin{cases} \dfrac{\beta^\alpha x^{\alpha-1} \mathrm{e}^{-\beta x}}{\Gamma(\alpha)} & \text{si } x > 0 \\ 0 & \text{sinon} \end{cases} \tag{3}$$

où $\alpha > 0$ et $\beta > 0$ sont des paramètres.
Avec cette paramétrisation, l'espérance et la variance de cette loi sont

$$\mu = \frac{\alpha}{\beta} \tag{4}$$

$$\sigma^2 = \frac{\alpha}{\beta^2} \tag{5}$$

La loi gamma est asymétrique à droite (asymétrie de $2/\sqrt{\alpha}$).
On observe que le système d'équations (4) et (5) est équivalent à

$$\alpha = \frac{\mu^2}{\sigma^2} \tag{6}$$

$$\beta = \frac{\mu}{\sigma^2} \tag{7}$$

Cela est utile notamment si l'on souhaite obtenir les paramètres d'une loi gamma étant donnée sa moyenne et sa variance.

3.2 Rappel sur la loi de Poisson

La loi de Poisson est une loi de probabilité discrète dont le support est $\mathbb{N} = \{0, 1, 2, 3, ...\}$.
On dit que X suit une loi de Poisson ($X \sim \text{Pois}(\lambda)$) si sa fonction de probabilité (fonction de masse) est

$$f(x; \lambda) = \begin{cases} \frac{\mathrm{e}^{-\lambda} \lambda^x}{x\,!} & \text{si } x \in \mathbb{N} \\ 0 & \text{sinon} \end{cases}$$

où λ est un paramètre. L'espérance est $\mu = \lambda$ et la variance $\sigma^2 = \lambda$.

3.3 Algorithme Thompson Sampling pour des bandits à distribution Poisson

Pour des bandits à distributions de Poisson, c'est-à-dire dans le cas où chaque bras génère un reward selon une loi de Poisson de paramètre λ inconnu, le conjugué à utilisé est la loi gamma [1].

Plus précisément, après n observations $x_1, x_2, ..., x_n$ tirées d'une loi de Poisson inconnue, la distribution posterior modélisant le paramètre λ inconnu sera

$$\text{Gamma}\left(\alpha_0 + \sum_{i=1}^{n} x_i,\ \beta_0 + n\right) \tag{8}$$

1. Voir la page web citée au cours : https ://en.wikipedia.org/wiki/Conjugate_prior

où α_0 et β_0 sont les paramètres de la distribution de gamma prior, ils correspondent donc aux paramètres choisis permettant d'initier la distribution.

Cela mène au pseudo-code suivant pour un algorithme Thompson Sampling joué sur un horizon T sur des bandits à K bras à distribution de Poisson.

3.3.1 Pseudo-code

Entrées de l'algorithme : bandit, α_0, β_0, T

- Pour $k = 1, 2, ..., K$ initialiser $\alpha_k = \alpha_0$ et $\beta_k = \beta_0$
- Pour chaque $1 \leq t \leq T$:
  - pour $k = 1, 2, ..., K$
    - échantillonner $\theta_k \leftarrow \text{Gamma}(\alpha_k, \beta_k)$
  - $k_t \leftarrow \underset{k}{\text{argmax}}(\theta_k)$
  - $r_t \leftarrow \text{play}(k_t)$
  - $\alpha_{k_t} \leftarrow \alpha_{k_t} + r_t$
  - $\beta_{k_t} \leftarrow \beta_{k_t} + 1$

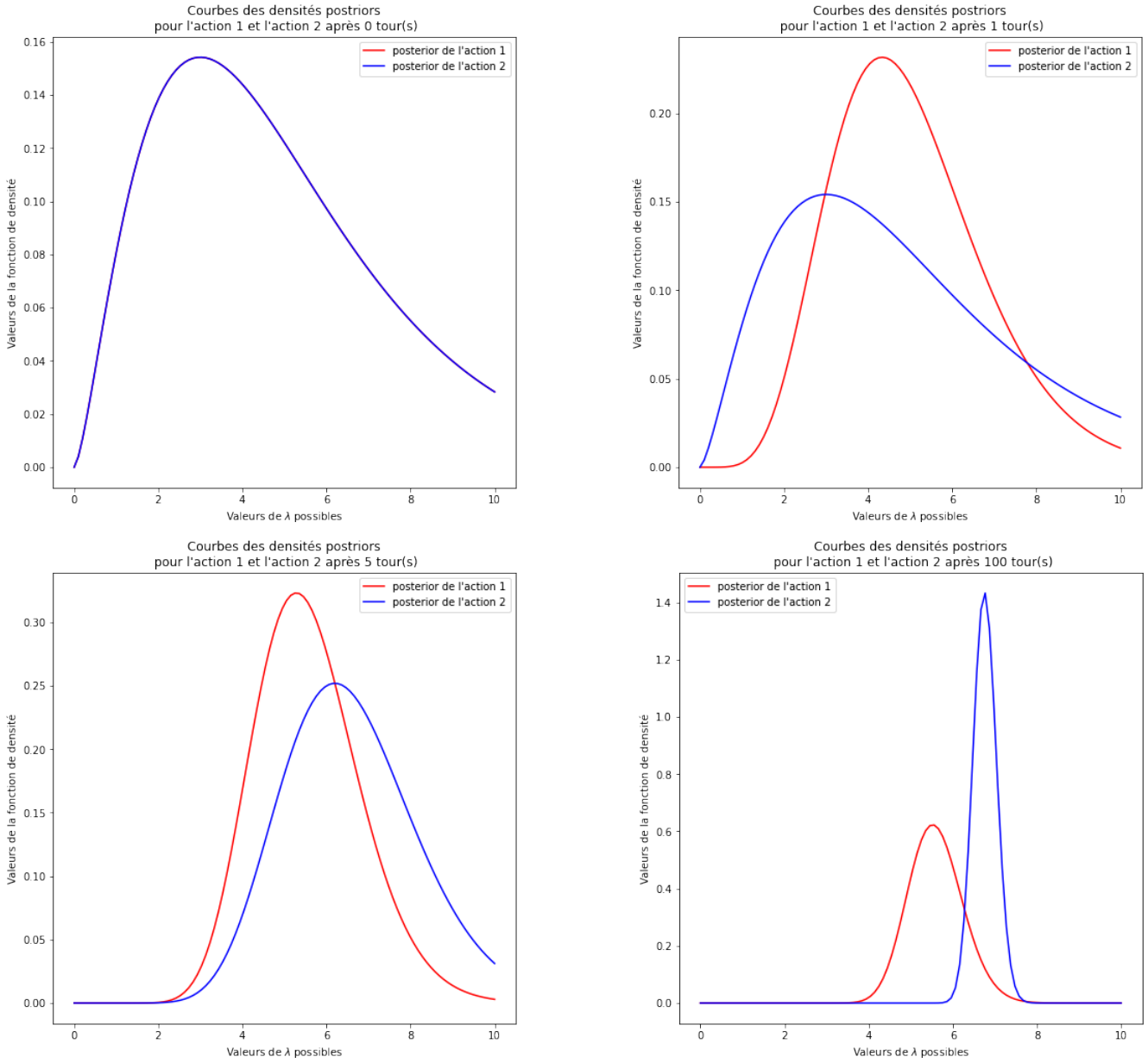
Remarque : même si ce n'est pas toujours mentionné explicitement dans la suite, les différents bras d'un bandit auront toujours la même paramétrisation de leur distribution prior.

3.4 Exploration du comportement de l'alogorihme (validation de la fonctionnalité)

Pour explorer le comportement de l'algorithme et vérifier qu'il fonctionne bien, on propose une expérience. Cette dernière consistera à lancer l'algorithme sur un horizon $T = 100$ sur une instance de bandit à deux bras à distribution de Poisson de moyennes respectives $\lambda_1 = 6$, $\lambda_2 = 7$ et avec une paramétrisation prior $\alpha_0 = 5/2$, $\beta_0 = 1/2$ identiques pour les deux bras. Pour tout $t = 0, 1, ..., 100$, on affiche les 2 distributions posteriors. On pourra ainsi voir l'évolution des posteriors selon les pas de temps. Notons qu'ici, on choisit une loi gamma initiale avec $\mu_0 = 5$ et $\sigma_0^2 = 10$. Nous discuterons plus loin du choix de la paramétrisation initiale. La figure **??** montre le graphiques des densités posteriors après 0 tour (priors), 1 tour, 5 tours, 100 tours. Dans le notebook, l'ensemble des 101 figures sont générées.

Initialement, on voit que les deux priors sont superposées. Après un tour, on voit qu'une seule distribution est modifiée (celle de l'action jouée, ici l'action 1). On voit par la suite que les distributions posteriors se déplacent et se concentrent, notamment celle de l'action optimale 2 qui se concentre de plus en plus autour de la moyenne associée à son bras ($\lambda_2 = 7$) et celle de l'action minimale se concentrant plus à gauche. Le comportement des posteriors est attendu et rassurant. À long terme, à chaque tour, il est de plus en plus probable que l'action jouée soit l'action optimale.

FIGURE 4 – Évolution des posteriors



3.5 Expérimentation sur l'algorithme

Dans cette section, on explore le comportement de l'algorithme dans diverses situations tout en s'efforçant de discuter de la sélection des priors. En effet, la loi de gamma n'ayant aucune paramétrisation la faisant corresponde à une loi uniforme, il faut faire un choix de priors qui n'est pas trivial. Pour ce faire, on supposera ici que l'on ne connaît pas les valeurs de λ associée à chaque bras, mais que l'on connaît à tout de moins un intervalle sur lequel ils se trouvent. Dépendant des choix des priors, ces derniers sont des distributions qui peuvent couvrir bien ou non l'intervalle possibles des λ, on suspecte que cela aura un effet sur le pseudo-regret cumulatif.

Pour établir une prior, il faut établir un couple α_0, β_0. Nous établirons plutôt un moyenne (μ_0) et une variance (σ_0^2) pour cette distribution prior et obtiendrons α_0 et β_0 associées avec les équations (6) et (7).

Remarque sur les simulations

Dans les simulations suivantes, nous avons opté pour un $N = 200$, puisque cela offre un bon compromis entre un N grand offrant un temps de calcul trop grand et un N trop petit donnant lieu à des résultats biaisés par certaines instances atypique. Un $N = 200$ facilite aussi la correction puisque les images peuvent être reproduites sans perdre trop de temps. Conséquemment, nous ne commenterons pas les écarts entre certaines courbes rapprochées, puisque dans d'autres simulations avec $N = 1000$, on observe que ces écarts s'atténuent ou s'inversent. Nos commentaires se feront donc sur les observations significatives dans le cas présenté $N = 200$, ces dernières restaient les mêmes ou très similaires dans le cas $N = 1000$.

3.5.1 Effet de la variance de la loi prior

Expérience sur des variances trop petites

On génère $N = 200$ instances de bandits de Poisson à deux bras.
Pour chaque instance, les moyennes λ_1, λ_2 des bras sont tirées aléatoirement d'une loi uniforme continue sur l'intervalle $[0, 10]$.

Pour chaque couple $(\mu_0, \sigma_0^2,)$ suivant :

$$(5, 10),\ (5, 5),\ (5, 1),\ (5, 1/2),\ (5, 1/4)$$

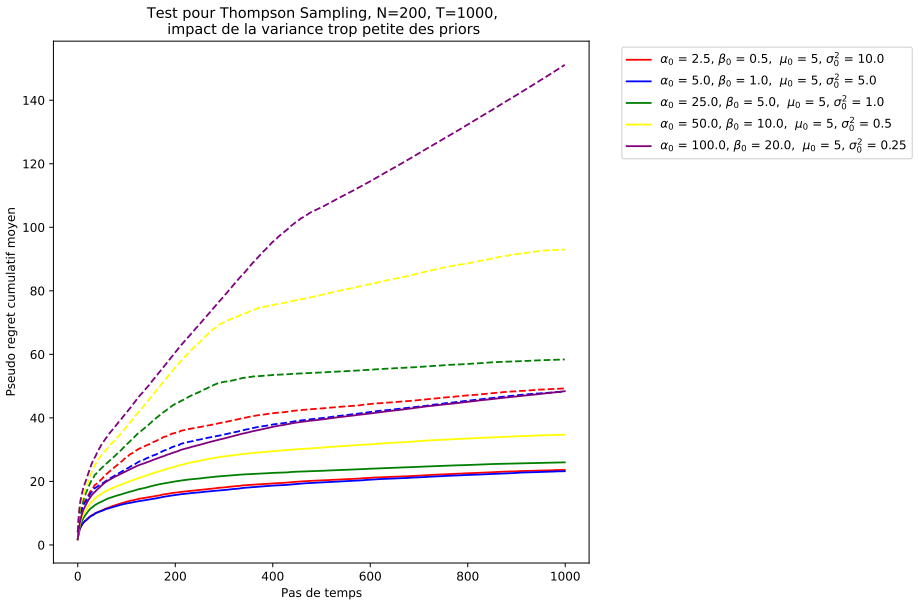
On trouve α_0 et β_0 associée à l'aide des équations (6) et (7).

On fait jouer Thompson Sampling sur les $N = 200$ instances sur un horizon $T = 1000$.

On trace les pseudos-regrets cumulatifs moyens avec une déviation standard au dessus (en pointillés).

Notons qu'ici, dans les différents couples (μ_0, σ_0^2) explorés, la moyenne est toujours 5 (le centre de l'intervalle [0,10]) et la variance varie (partant de 10 correspondant à la grandeur de l'intervalle et se rapetissant).

FIGURE 5 – Comparaisons de la performance selon des priors de variances plus petites



On remarque que si la distribution prior est choisie avec une variance trop petite, la distribution prior ne couvre pas suffisamment bien l'intervalle $[0, 10]$ des valeurs possibles de λ et cela semble se traduire par une augmentation du pseudo-regret cumulatif moyen. On remarque qu'il semble y avoir un certaine tolérance sur des variations de σ_0^2 qui ne changent pas significativement la courbe du pseudo-regret cumulatif moyen.

Expérience sur des variances trop grandes

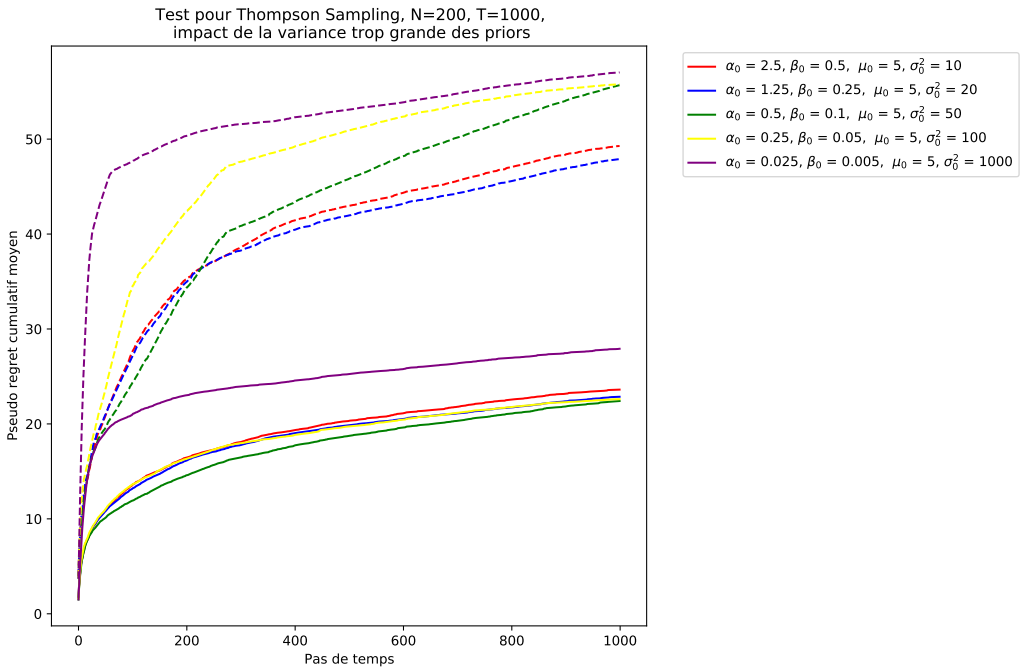
Même méthode que l'expérience précédente, mais sur les couples (μ_0, σ_0^2) suivants.

$$(5, 10),\ (5, 20),\ (5, 50),\ (5, 100),\ (5, 1000)$$

La moyenne des priors est toujours le centre de l'intervalle $[0, 10]$, la variance varie (partant de 10 correspondant à la grandeur de l'intervalle et se grandissant).

La figure suivante montre le résultat obtenu.

FIGURE 6 – Comparaisons de la performance selon des priors de variances plus grandes



On peut voir qu'une variance légèrement trog grande n'aura pas beaucoup d'effet sur le regret cumulatif moyen, par contre, une variance excessivement trop grande peut en avoir un considérable. Dans ce cas ($\sigma_0^2 = 1000$), la distribution prior sur-couvre grandement l'intervalle [0,10] des valeurs possibles de λ. Conséquemment, possiblement que les distributions posteriors ne se concentrent pas suffisamment rapidement pour pouvoir bien cerner les moyennes λ_1 et λ_2 des distributions de Poisson des bras.

3.5.2 Effet de la moyenne de la loi prior avec variance «bonne»

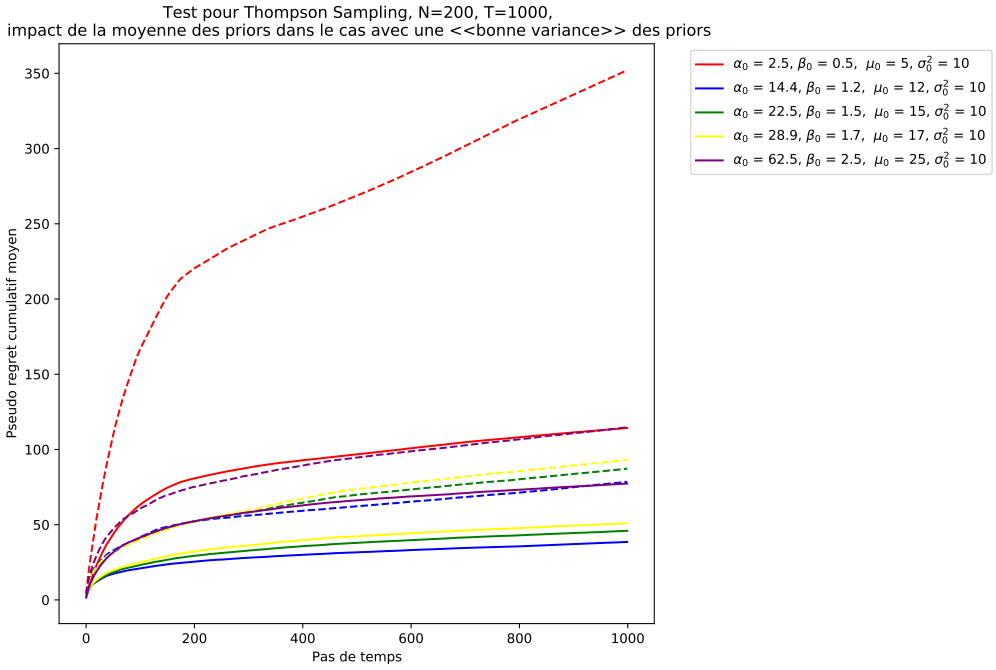
Pour bien pouvoir faire varier les moyennes de la prior choisi, on va ici considérer une situation analogue aux deux premières expériences, mais où l'intervalle sur lequel sont choisis les moyennes λ_1, λ_2 est $[10, 20]$ au lieu de $[0, 10]$. On effectue la même expérience que si haut, sur les couples (μ_0, σ_0^2) qui suivent.

$$(5, 10),\ (12, 10),\ (15, 10),\ (17, 10),\ (25, 10)$$

La variance de la prior est fixe égale à la longueur de l'intervalle des valeurs possibles de λ.

On obtient la figure suivante

FIGURE 7 – Comparaisons de la performance selon des priors de variances égale à la longueur de l'intervalle et de moyennes variables.



On remarque que si la moyenne du prior est fortement décalée du centre de l'intervalle $[10, 20]$, cela aura un effet sur la performance de l'algorithme, notamment le cas où $\mu_0 = 5$. Lorsque la moyenne μ_0 reste dans l'intervalle $[10, 20]$, il semble que le fait que la variance soit bonne (distribution prior qui couvre bien l'intervalle) fait en sorte que l'algorithme reste bon même si μ_0 n'est pas centré.

3.5.3 Effet de la moyenne de la loi prior avec variance plus petite

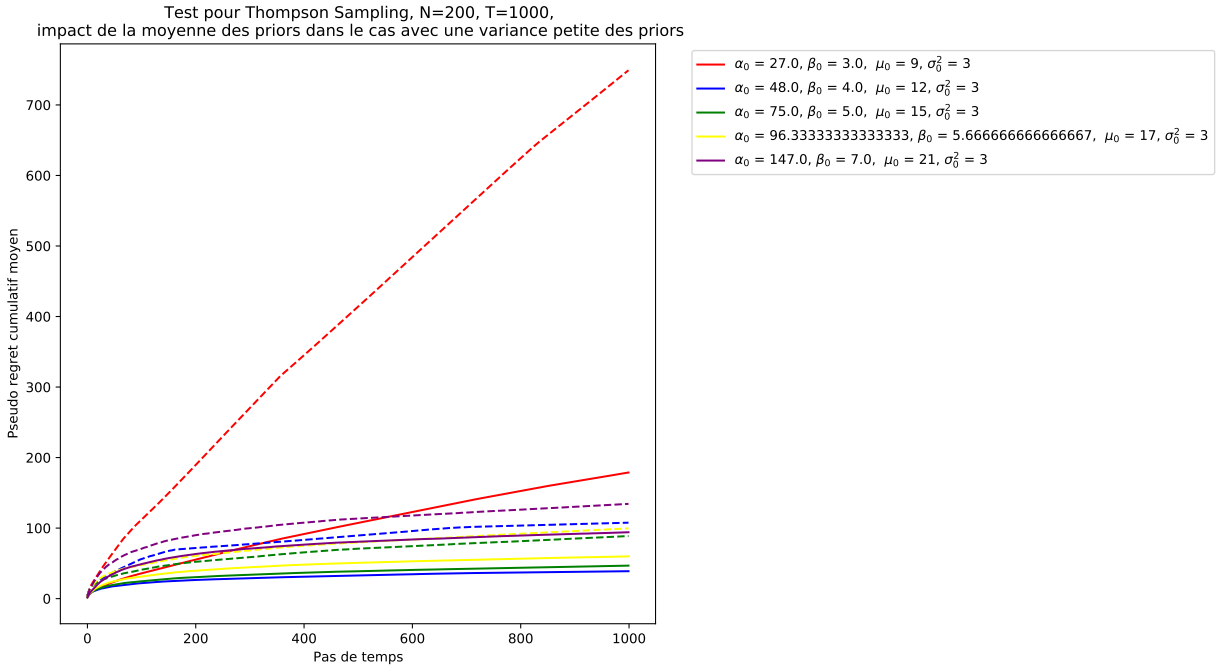
Même expérience que précédemment, mais avec les couples (μ_0, σ_0^2) qui suivent.

$$(9, 3),\ (12, 3),\ (15, 3),\ (17, 3),\ (21, 3)$$

Cette fois, la variance est plus petite (3).

On obtient la figure suivante

FIGURE 8 – Comparaisons de la performance selon des priors de variances égalent à 3 et moyennes variables.



On voit qu'ici, avec une variance des priors plus petite, un même décalage de la moyenne μ_0 par rapport au centre de l'intervalle semble être plus coûteux que dans l'expérience précédente. D'ailleurs, le cas $\mu_0 = 9$ semble en être un bon exemple car il atteint une regret cumulatif moyen final plus élevé que le cas $\mu_0 = 5$ dans le cas où la variance était de 10 (voir expérience précédente). Conséquemment, si l'information que l'on a à priori sur les λ (intervalle de possibilité) n'est pas bonne pour fixer les priors, lancer un tel algorithme avec une variance trop petite peut avoir un impact sur la performance, notamment si par malchance on choisit un prior avec μ_0 décalé du centre du réel intervalle des valeurs possibles de λ.

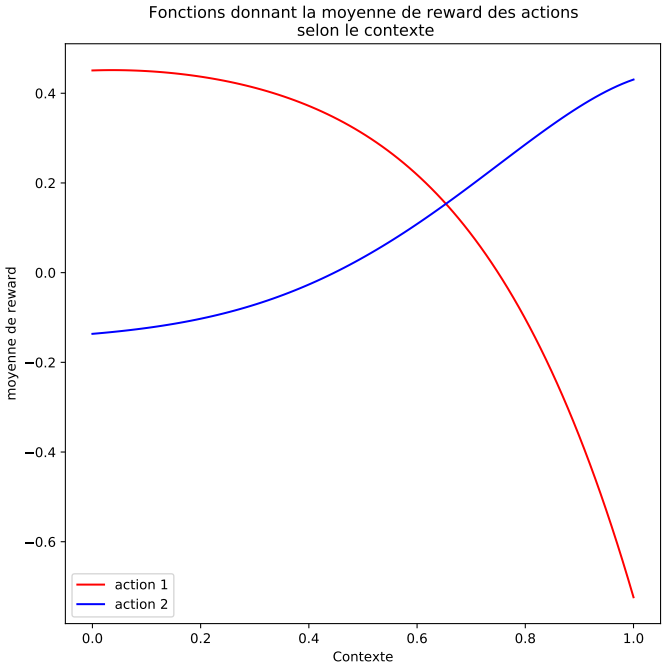
En conclusion, à la lumière des simulations effectuées, les priors semblent devoir être choisis pour couvrir raisonnablement bien l'intervalle des valeurs possibles de λ. On pourrait suggérer de prendre la distribution prior telle que μ_0 est le centre de l'intervalle et σ_0^2 la longueur de l'intervalle. Selon nos simulations (notamment avec $N = 1000$ non-présenté ici), cela semble être un choix raisonnable.

4 Bandits contextuels et structurés

4.1 Contexte

On considère le cas d'un bandit contextuel à M actions avec un espace de contexte S. Dans ce «setting», à chaque temps t, l'agent observe un contexte s_t à partir duquel il doit sélectionner une actions $m_t \in 1, 2, \ldots, M$ et observer un reward $r_t = f_{m_t}(s_t) + \eta_t$ où $\eta_t \sim \mathcal{N}(0, \sigma^2)$. La figure suivante montre une situation hypothétique avec un contexte $S = [0, 1]$ et $M = 2$ actions.

FIGURE 9 – Illustration d'un contexte de bandit contextuel



On désire définir l'algorithme Kernel Thompson Sampling dans cette situation. En s'inspirant de *Durand, A., Maillard, O. A., et Pineau, J. (2018)*, on arrive au pseudo-code suivant.

4.2 Pseudo-code

- Pour $1 \leq t \leq T$
  - Pour chaque bras $m = 1, 2, \ldots, M$
    - Calculer les moyennes posteriors $\widehat{f}_{t-1,m} = (f_{\lambda_t, t-1, m}(s))_{s \in \mathbb{S}}$
    - Calculer la covariance posteriors $\widehat{\Sigma}_{t-1,m} = \dfrac{\sigma^2_{+,t-1}}{\lambda_t} \left(k_{\lambda_t, t-1}(s, s')\right)_{(s,s') \in \mathbb{S}}$
    - échantillonner $\widetilde{f}_{t,m} = \mathcal{N}\left(\widehat{f}_{t-1,m}\,;\, v_t^2\, \widehat{\Sigma}_{t-1,m}\right)$
  - observer un contexte $s_t \in \mathbb{S}$
  - $m_t = \underset{m}{\operatorname{argmax}} \widetilde{f}_{t,m}(s)$
  - Jouer l'action m_t
  - Observer un reward : $y_t = f_{m_t}(s_t) + \eta_t$

4.2.1 Détails de l'implémentation

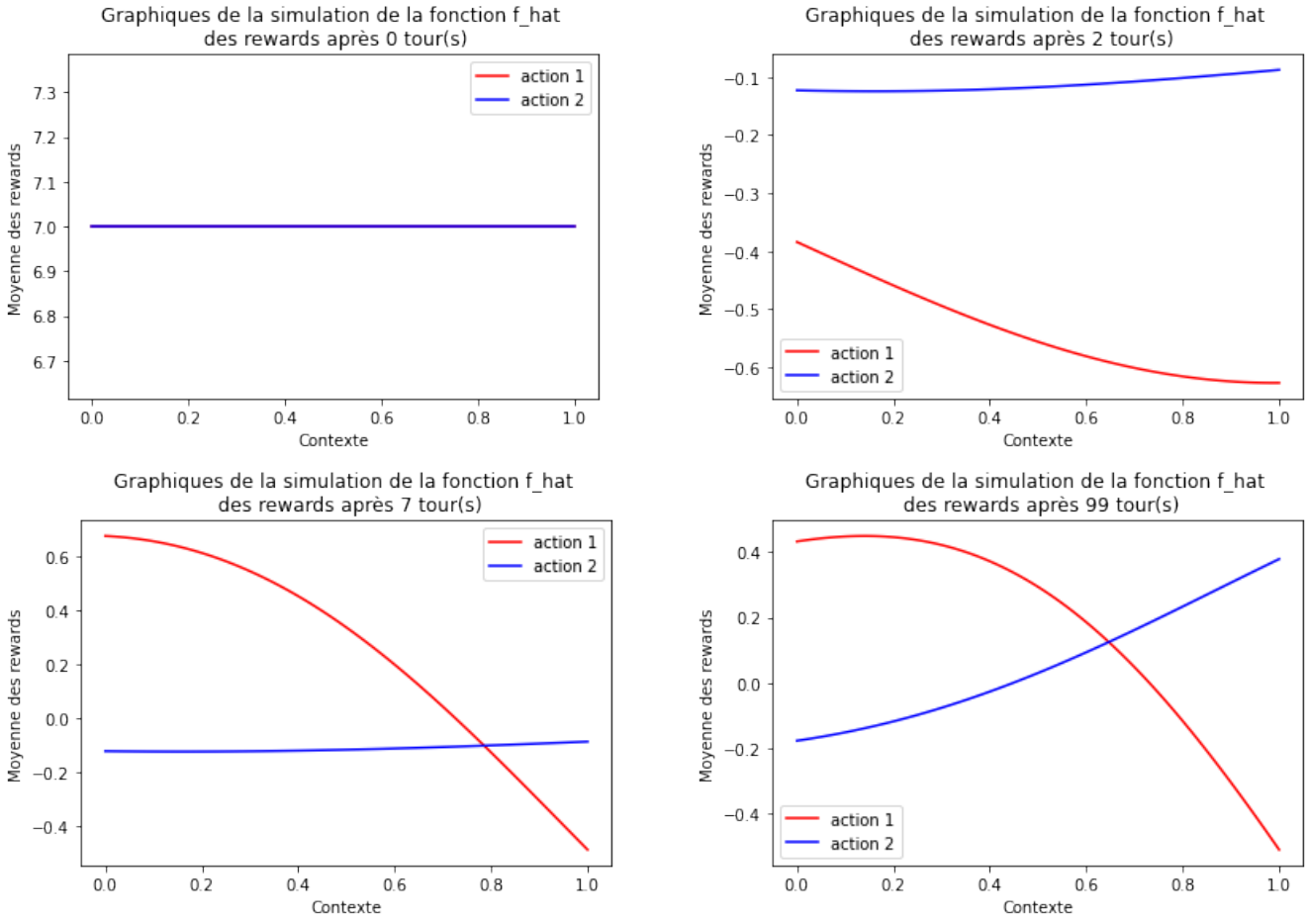
Pour simplifier l'algorithme, on lui passera simplement σ comme valeur de $\sigma^2_{+,t-1}$ et on prendra $\lambda = \sigma^2$ étant donner que les fonctions de rewards seront des fonctions polynomiales dont les coefficients sont les éléments d'un vecteur θ de norme 1. **Aussi, notons que le pseudo-code précédent conserve la forme de l'article, alors que pour l'implémentation, on initialisera $\widehat{f}$ et $\widehat{\Sigma}$ pour chaque bras avant le premier tirage de $\widetilde{f}$, ensuite, à la fin de chaque itération, $\widehat{f}$ et $\widehat{\Sigma}$ seront calculés avec l'aide d'un «Gaussian Process Regressor»**. Nous avons fait le choix d'initialiser nos $\widehat{f}$ à 7 pour que la fonction de départ soit toujours suppérieur à nos deux courbes d'actions possibles. De plus nous avons fait le choix d'initialiser la matrice $\widehat{\Sigma}$ comme la matrice 0. Aussi, l'espace de contexte $\mathcal{S}$ s'est vue discrétisée en $\mathbb{S}$ pour faciliter l'implémentation. De plus, dans le Gaussian Process Regressor, nous avons

utilisé un noyau RBF avec une bandwidth de 1 dans l'expérience ci-dessous. Finalement, comme écrit dans l'article de référence au sujet des performances en pratiques, nous avons choisi un v_t^2 de 1.

4.3 Expérimentation

On a joué l'algorithme sur un horizon $T = 100$ sur une instance des deux bandits dont le graphe des rewards moyen est présenté à la figure 9. Les figures suivantes montrent les fonction $\widehat{f}$ pour chacune des deux actions selon le nombre de tours joués.

FIGURE 10 – Évolution des posteriors

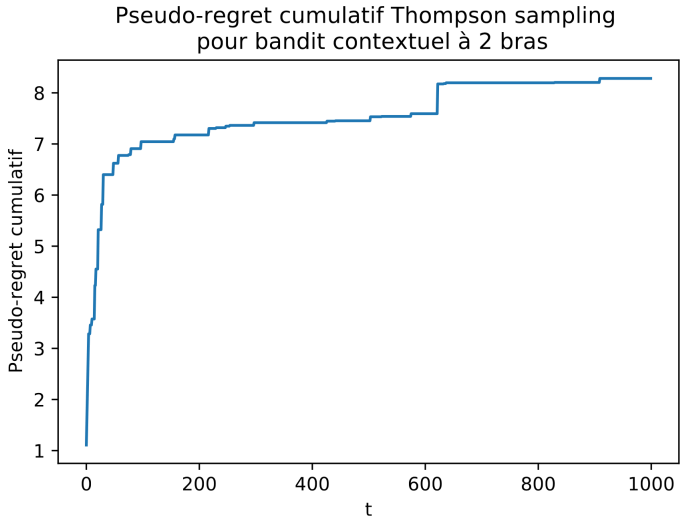


On peut voir que l'algorithme estime de mieux en mieux les fonctions des moyennes des rewards à mesure qu'il joue. On voit qu'à la fin, le graphe estimé est très près des vrais graphiques représentant les actions montrés à la figure 9.

Pour une action donnée, on observe que pour les contextes qui la favorisent, l'estimation de la courbe est meilleur que l'estimation de la courbe pour les contextes qui défavorisent cette action. Cela est du au fait que pour les contextes qui défavorisent une action, l'action est très rarement jouée, donc il est difficile pour l'estimateur de connaître en détails la géométrie la courbe de l'action dans ces contextes.

Voici le pseudo regret cumulatif pour la même expérience

FIGURE 11 – Performance de Kernel Thompson Sampling



On voit un comportement typique pour ce type d'algorithme, c'est-à-dire que celui-ci cumule du regret passablement souvent au début à mesure qu'il cumule de l'information pour ensuite être bien en selle pour choisir presque tout le temps l'action optimale. On peut voir dans le graphique certaines brisures suivies de plateaux même pour de grands t, ces brisures sont potentiellement associées à des moments où le contexte s donné à l'agent était problématique, étant près de l'abscisse d'intersection entre les deux courbes d'action, ce qui peut produire un mauvais choix de l'action optimale.

Nous avons aussi expérimenté pour connaître l'influence de la taille du noyau à l'intérieur du GaussianProcessRegressor et nous avons obtenu une meilleur performance avec une taille de 2 pour 5 instances différentes pour T = 500 comme le montre la figure suivante. Il importe de nuancé le résultat ici car seulement 5 instances ont été utilisés (le temps de calcul de l'algorithme implanté étant passablement long). Avec plus de temps, plusieurs autres expériences auraient pu être présentées pour tenter de bien cerner l'effet de divers paramètres.

FIGURE 12 – Évolution pseudo-regret cumulatif