NNT : xxxxxxx

Thèse de doctorat de l'Université Paris-Saclay préparée Université Paris-Sud

Ecole doctorale n°580
Sciences et technologies de l'information et de la communication
Spécialité de doctorat : Informatique

par

M. Robin Allesiardo

Bandits Manchots sur Flux de Données Non Stationnaires

Thèse présentée et soutenue à Orsay, le 19 octobre 2016.

Composition du Jury :

M.	Ludovic Denoyer	Professeur LIP6 - UPMC	(Examinateur)
M.	Odalric-Ambrym Maillard	Chargé de Recherche LRI, INRIA	(Examinateur)
M.	Jérémie Mary	Maître de Conférences HDR CRISTAL, INRIA	(Rapporteur)
M.	Éric Moulines	Professeur École Polytechnique	(Examinateur)
M.	Liva Ralaivola	Professeur LIF, CNRS	(Rapporteur)
Mme.	Michèle Sébag	Directrice de recherche LRI, INRIA	(Directrice de thèse)

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Bandits Manchots sur Flux de Données Non Stationnaires

Mots clés : Apprentissage automatique, apprentissage en ligne, non-stationnarité, bandits manchots.

Résumé :

Le problème des bandits manchots est un cadre théorique permettant d'étudier le compromis entre exploration et exploitation lorsque l'information observée est partielle. Dans celui-ci, un joueur dispose d'un ensemble de K bras (ou actions), chacun associé à une distribution de récompenses $D(\mu_k)$ de moyenne $\mu_k \in [0,1]$ et de support $[0,1]$. A chaque tour $t \in [1,T]$, il choisit un bras k_t et observe la récompense y_{k_t} tirée depuis $D(\mu_{k_t})$. La difficulté du problème vient du fait que le joueur observe uniquement la récompense associée au bras joué; il ne connaît pas celle qui aurait pu être obtenue en jouant un autre bras. À chaque choix, il est ainsi confronté au dilemme entre l'exploration et l'exploitation; explorer lui permet d'affiner sa connaissance des distributions associées aux bras explorés tandis qu'exploiter lui permet d'accumuler davantage de récompenses en jouant le meilleur bras empirique (sous réserve que le meilleur bras empirique soit effectivement le meilleur bras). Dans la première partie de la thèse nous aborderons le problème des bandits manchots lorsque les distributions générant les récompenses sont non-stationnaires. Nous étudierons dans un premier temps le cas où même si les distributions varient au cours du temps, le meilleur bras ne change pas. Nous étudierons ensuite le cas où le meilleur bras peut aussi changer au cours du temps. La seconde partie est consacrée aux algorithmes de bandits contextuels où les récompenses dépendent de l'état de l'environnement. Nous étudierons l'utilisation des réseaux de neurones et des forêts d'arbres dans le cas des bandits contextuels puis les différentes approches à base de méta-bandits permettant de sélectionner en ligne l'expert le plus performant durant son apprentissage.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Title : Multi-armed Bandits on non Stationary Data Streams

Keywords : Machine learning, online learning, non-stationarity, multi-armed bandits.

Abstract :

The multi-armed bandit is a framework allowing the study of the trade-off between exploration and exploitation under partial feedback. At each turn $t \in [1, T]$ of the game, a player has to choose an arm k_t in a set of K and receives a reward y_{k_t} drawn from a reward distribution $D(\mu_{k_t})$ of mean μ_{k_t} and support $[0, 1]$. This is a challeging problem as the player only knows the reward associated with the played arm and does not know what would be the reward if she had played another arm. Before each play, she is confronted to the dilemma between exploration and exploitation; exploring allows to increase the confidence of the reward estimators and exploiting allows to increase the cumulative reward by playing the empirical best arm (under the assumption that the empirical best arm is indeed the actual best arm). In the first part of the thesis, we will tackle the multi-armed bandit problem when reward distributions are non-stationary. Firstly, we will study the case where, even if reward distributions change during the game, the best arm stays the same. Secondly, we will study the case where the best arm changes during the game. The second part of the thesis tacles the contextual bandit problem where means of reward distributions are now dependent of the environment's current state. We will study the use of neural networks and random forests in the case of contextual bandits. We will then propose meta-bandit based approaches for selecting online the most performant expert during its learning

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Remerciements

Merci.

Table des matières

Introduction

Ce chapitre introduit dans un premier temps l'apprentissage automatique dans la section 1. Nous présentons les hypothèses permettant de construire des modèles prédictifs dans ce cadre. Nous relaxons ensuite successivement les hypothèses de stationnarité puis de complétude de l'information pour introduire l'apprentissage en ligne dans la section 1.2 puis l'apprentissage par renforcement dans la section 1.3. Finalement, nous détaillerons le plan de la thèse ainsi que les contributions dans la section 2.

1 Apprentissage Automatique

L'apprentissage automatique (*machine learning* en anglais) est une discipline dédiée aux méthodes permettant à une machine d'apprendre grâce aux expériences passées. Ces méthodes permettent d'extraire des régularités dans des données et de créer des modèles accomplissant des tâches difficiles à spécifier (de par leur complexité ou à cause du manque d'expertise dans le domaine) mais dont il existe des exemples de comportement attendus.

Voici quelques exemples de tâches d'apprentissage :

- **La classification supervisée** : la machine doit prédire une classe associée à une donnée. Par exemple, dans une tâche de reconnaissance de chiffres manuscrits, la donnée est une image (une matrice de pixel) et la classe désigne le chiffre représenté dans l'image.
- **La régression** est similaire à la classification supervisée mais la prédiction porte sur une variable continue plutôt qu'une classe, par exemple, une probabilité.
- **La classification non supervisée** : contrairement à l'exemple précédent, l'apprentissage n'est pas guidé par la classe. Dans le cas d'une tâche de partitionnement de données (*clustering* en anglais), la machine créera des groupes de données de manière à minimiser la distance intra-groupe (mettre les données similaires ensemble) et à maximiser la distance intergroupes (mettre les données hétérogènes dans des groupes différents). Ceci peut notamment être utilisé pour dégager des populations ayant les mêmes comportements dans une base de clients lors d'études marketing.

- **L'estimation de densité** est la création d'une fonction permettant d'associer la valeur d'une variable à sa probabilité. Par exemple, pour une variable catégorielle, un estimateur de densité pourrait-être la fréquence de chaque valeur dans un échantillon. Cette méthode peut par exemple être utilisée pour prédire la langue d'un texte en fonction de la fréquence de chaque mot.

1.1 Apprentissage hors-ligne

Lorsque l'apprentissage hors-ligne est supervisé, trois éléments sont nécessaires afin d'effectuer une tâche d'apprentissage :

- un jeu de données, dont chacun des éléments est un couple composé d'une « entrée de la machine »(habituellement appelé exemple d'apprentissage) et de la « sortie de la machine »(la classe) associée permettant de superviser l'apprentissage. Ces éléments composent l'ensemble d'apprentissage.
- Un modèle, composé de différents paramètres et permettant d'effectuer la tâche de prédiction.
- Un algorithme d'apprentissage, qui règle les paramètres du modèle afin de minimiser l'espérance d'une fonction de perte.

Une fois le modèle d'apprentissage finalisé, il pourra est déployé et servir à prédire une classe, alors que celle-ci est inconnue, à partir d'une nouvelle donnée.

Une des limites de l'apprentissage hors-ligne : la non-stationnarité

L'une des principales hypothèses permettant la création d'algorithmes d'apprentissage est de supposer que les données sont collectées de manière indépendante et identiquement distribuée. En apprentissage hors-ligne, l'hypothèse de stationnarité est utilisée pour permettre aux algorithmes d'apprendre sur des données précédemment collectées et d'obtenir des garanties théoriques sur l'erreur de généralisation (l'erreur en prédiction sur des données inconnues jusqu'à présent), par exemple via la théorie de Vapnik-Chervonekis [1] ou la théorie PAC-Bayes [2]. Ainsi, un modèle entrainé sur ces données pourra être utilisé pour effectuer des prédictions sur de nouvelles données où l'information à prédire est manquante. Intuitivement, l'indépendance entre deux évènements A et B signifie que la probabilité d'observer un évènement reste identique, que le second évènement se soit produit ou non. Formellement, A et B sont indépendants si et seulement si $\mathbf{P}(A \cap B) = \mathbf{P}(A)\mathbf{P}(B)$. L'identité de la distribution suppose que les évènements sont générés par la même distribution de probabilité. Nous qualifions de stationnaire tout processus dont les réalisations sont générées de manière indépendante et identiquement distribuée. Cette hypothèse est vérifiée sur certains problèmes, comme la reconnaissance de chiffres manuscrits ou de visages. À l'échelle de la vie des modèles, on peut considérer qu'un changement de distribution est très peu

probable. La démocratisation des algorithmes prédictifs dans les systèmes de décisions sur internet ou dans les objets connectés remettent cependant en question cette hypothèse. Les prédictions doivent être personnalisées suivant les utilisateurs et leurs comportements peuvent changer, rendant obsolètes les données plus anciennes.

1.2 Apprentissage en ligne

La principale différence entre les apprentissages hors ligne et en ligne est la manière dont sont acquises les données. En ligne, celles-ci arrivent une à une sous forme de flux et ne sont pas disponibles en totalité au début de l'apprentissage. Les approches en ligne (voir Algorithme 1) permettent de construire un modèle à partir d'un flux de données et les algorithmes en ligne sont capables de fournir un modèle prédictif à chaque instant, quel que soit le nombre de données observées. Ce contact avec le flux peut leur permettre de minimiser l'impact des changements de stationnarité en adaptant leur modèle à la nouvelle distribution.

Lors de la conception d'algorithmes en ligne, plusieurs contraintes doivent être prises en compte :

- **L'espace mémoire limité :** Les flux de données peuvent être de grande taille et de ce fait considérés comme infinis. Il devient alors important d'assurer un stockage performant de l'information nécessitant peu d'espace mémoire. Une stratégie communément utilisée est de mettre à jour le modèle d'apprentissage chaque fois qu'un nouvel élément du flux apparaît. Les algorithmes en ligne utilisent ces éléments un à un pour maintenir des estimateurs (des moyennes par exemple) ou bien procéder à une itération de descente de gradient (qui est naturellement un algorithme en ligne). Après cette mise à jour du modèle, il n'est pas nécessaire de garder la donnée en mémoire.
- **Le temps de traitement :** Il est nécessaire que le temps de traitement global soit au maximum linéaire suivant la taille du flux. Ainsi, avec une architecture logicielle et matérielle adaptée (notamment via la parallélisation des algorithmes) il est possible de traiter les données du flux en temps réel. Si la complexité algorithmique était sur-linéaire vis-à-vis de la taille du flux, qu'importe la puissance de calcul à disposition, il existera toujours un temps à partir duquel le processus de classification sera saturé et ne sera plus en mesure de traiter les données en temps réel.
- **Les garanties théoriques :** Les algorithmes d'apprentissage en ligne peuvent être utilisés afin de prendre des décisions en temps réel sur des données jusqu'alors inconnues. Il est donc important pour les algorithmes d'apprentissage en ligne de posséder des garanties théoriques en pire cas. Les bornes en pire cas permettent de contrôler les risques lors de l'utilisation de l'algorithme sur un problème inconnu. Ces garanties portent souvent sur le coût à payer afin d'obtenir un modèle aux performances similaires à celles d'un algorithme hors-ligne apprenant sur la totalité du flux de données. Lorsque ce coût est sous-linéaire par rapport à la taille du flux (le

gain de l'algorithme est lui linéaire lorsque le modèle a atteint son niveau de performance final) alors celui-ci s'évanouit lorsque la taille du flux tend vers l'infini.

Algorithme 1 : CLASSIFICATION EN LIGNE À INFORMATION COMPLÈTE

$t = 0$ `// L'itération courante`
T `// L'horizon de l'algorithme`
pour $t = 1; t \leq T; t++$ **faire**
  Recevoir une donnée
  Effectuer la prédiction à partir de la donnée reçue `// Facultatif`
  Recevoir la valeur qui aurait due être prédite
  Mettre à jour le modèle en prenant en compte cette nouvelle information
  Libérer la mémoire occupée par la donnée
fin

Le problème de l'information partielle

Similairement à l'hypothèse de stationnarité, l'information est souvent supposée complète et les algorithmes d'apprentissage disposent de l'information à prédire pour chaque instance des données. L'information à prédire est soit déjà présente à l'intérieur des données ou alors est ajoutée via un processus d'étiquetage. Dans le cas applicatif de la reconnaissance de chiffres manuscrits, les images de chiffres ont été étiquetées manuellement avant l'apprentissage. Si l'étiquetage manuel parait raisonnable pour ce problème, il peut être remis en question lorsque les données proviennent d'un flux. Prenons l'exemple de la recommandation de contenus sur des sites internet d'actualités. Les profils des visiteurs sont très hétérogènes, en grande quantité et les contenus à recommander peuvent changer très rapidement. Ici, l'étiquetage ne peut en aucun cas être manuel et même son automatisation pose problème. Après avoir recommandé un contenu à l'utilisateur, il n'est pas possible de connaitre quelle aurait été sa réaction en présence d'un autre contenu. Le système peut uniquement savoir si l'utilisateur a consommé le contenu présenté, on parle alors d'information partielle. Récolter un ensemble de données dans de telles conditions pour effectuer un apprentissage peut être couteux. Comment définir la proportion de chaque contenu à présenter aux visiteurs ? Combien de fois faudra-t-il les présenter ? En observant directement les profils utilisateurs et en choisissant le contenu à présenter, un algorithme d'apprentissage par renforcement peut explorer l'espace des contenus de manière efficace.

1.3 Apprentissage par renforcement et Bandits manchots

L'apprentissage par renforcement trouve ses racines dans la Loi de l'effet [3] ; des actions possibles face à une même situation, celles qui sont récompensées seront reproduites plus fréquemment que celles punies, cette fréquence étant proportionnelle à l'intensité des récompenses ou des punitions. Selon Sutton et Barto [4], cette

définition inclut deux aspects principaux, la sélection (choisir la meilleure action) et l'association (associer une action à une situation). En s'inspirant des observations biologiques, de nombreuses architectures décrivant les interactions entre le modèle d'apprentissage, le contexte, les différentes actions et la récompense ont été proposées [5] [6]. La plupart de ces méthodes cherchent, en estimant la probabilité d'obtenir une récompense à partir d'un état, une succession d'états menant à un état final récompensé. Le problème des bandits manchots est un cas particulier de l'apprentissage par renforcement né du besoin de comparer l'efficacité de plusieurs traitements dans le cadre des tests pharmaceutiques [7]. Il a ensuite été formalisé par Lai et Robbins [8]. Un ensemble de bras (ou actions) est à la disposition d'un joueur. Celui-ci dispose d'un nombre fini de tours de jeu durant lesquels il doit choisir un bras avant de recevoir une récompense. Son but est de maximiser le cumul de ses récompenses à la fin du jeu. Cela introduit un compromis entre l'exploration et l'exploitation. L'exploration consiste à jouer les bras dans le but d'affiner l'estimation de leurs gains potentiels et l'exploitation consiste à jouer les bras pouvant apporter les plus hauts gains. Le problème des bandits manchots dits stochastiques suit la formulation de Lai et Robbins et suppose que chaque bras est associé à une distribution-stationnaire générant les récompenses. Le problème des bandits manchots offre un cadre théorique à l'étude des algorithmes en ligne travaillant avec une information partielle et est très adapté aux applications réelles dans lesquelles les effets des décisions non prises ne sont pas connus. Nous aborderons donc le problème de la non-stationnarité en utilisant les bandits manchots comme base théorique.

2 Contributions

La première partie de la thèse (voir Partie I) traite du problème des bandits manchots lorsque les distributions générant les récompenses sont non-stationnaires.

Le chapitre 1 formalise le problème des bandits manchots et dresse un état de l'art des principaux algorithmes de bandits. La section 1.2.1 présente les algorithmes fonctionnant en régime stationnaire par parties. Cette stationnarité par partie permet de diviser le jeu en segments temporels de tailles variables durant lesquels les distributions générant les récompenses sont stationnaires. Les garanties théoriques des algorithmes développés dans ce cadre dépendent du nombre de changement de stationnarité. Dans la section 1.2.2, nous présentons les algorithmes de bandits adverses. Dans ce formalisme, les séquences de récompenses obtenues par chaque bras sont fixées avant le début du jeu par un adversaire. Pour finir, la section 1.2.3 présente des algorithmes développés pour fonctionner sur des régimes hybrides, mixant régime stationnaire et régime adverse.

Le chapitre 2 développe le cas où les distributions stochastiques générant les récompenses peuvent changer à chaque tour de jeu. Cette forme de non-stationnarité est plus forte que celles présentées dans le chapitre précédent. En effet, celle-ci

est un cas pathologique de la stationnarité par partie où chaque segment temporel contient uniquement un tour de jeu (impliquant ainsi des bornes linéaires en fonction du temps pour les algorithmes dont les bornes dépendent du nombre de segments). Le régime adverse est aussi contenu dans le régime stochastique non-stationnaire, où la séquence de récompenses choisie par l'adversaire au début du jeu peut-être obtenue en considérant une séquence de distributions ayant chacune comme moyenne la valeur de chaque récompense de la séquence définie par l'adversaire ainsi que des variances nulles. Dans ce chapitre, nous posons des hypothèses permettant d'effectuer une tâche d'identification du meilleur bras en régime stochastique non-stationnaire, avec l'algorithme SER3 (voir section 2.3). Nous proposons ensuite deux algorithmes différents pour le cas plus général où le meilleur bras peut changer au cours du jeu. L'algorithme SER4 (voir section 2.3) utilise SER3 comme sous-routine pour identifier le meilleur bras, tout en ayant une probabilité à chaque tour de réinitialiser cette sous-routine pour vérifier si le meilleur bras a changé. L'algorithme EXP3.R propose une approche similaire, utilisant l'algorithme adverse EXP3 comme sous-routine. Contrairement à SER4, les réinitialisations de EXP3 sont ici contrôlées par un test de détection de changement de bras et ne sont pas effectuées indépendamment des données. Le chapitre contient les analyses théoriques de ces algorithmes ainsi que plusieurs simulations numériques illustrant leurs comportements sur différents problèmes synthétiques.

La seconde partie de la thèse (voir Partie II) aborde le problème des bandits manchots contextuels. Dans les bandits contextuels, le joueur est capable d'observer un ou plusieurs vecteurs de contextes à chaque tour. Les récompenses obtenues par chaque bras peuvent être dépendantes de ce contexte. Le chapitre 3 présente le formalisme des bandits manchots contextuels ainsi que les algorithmes de l'état de l'art, suivit de quatre chapitres de contributions.

Le chapitre 4 propose une heuristique permettant l'utilisation de réseaux de neurones pour les bandits contextuels. Nous proposons une approche basée sur des méta-bandits pour sélectionner en ligne le meilleur paramétrage parmi un ensemble de paramétrages différents proposés par le joueur. Les séquences de récompenses dépendent des prédictions des réseaux puisque ceux-ci influencent l'exploration de l'espace des bras. Optimiser un réseau de neurones est un problème non-convexe, empêchant l'analyse théorique de l'algorithme. De plus, les algorithmes pouvant être utilisés comme méta-bandit supposent que l'évolution des récompenses n'est pas dépendante des choix du joueur. Cette hypothèse n'est pas vérifiée lorsque ces algorithmes sont utilisés pour sélectionner en ligne le meilleur expert durant son apprentissage et les analyses théoriques de ces algorithmes ne sont pas valables dans ce cas-là. Nous appelons ce problème, sélection d'experts apprenants et le développons dans le chapitre 6. Finalement, nous validons expérimentalement l'approche proposée.

Le chapitre 5 adapte les forêts aléatoires au cadre bandit. Tout d'abord, nous proposons un algorithme d'élimination permettant de sélectionner dans chaque nœud d'un arbre, de manière gloutonne, la variable de coupure engendrant le plus de récompense au prochain niveau. Lorsqu'un chemin dans l'arbre est assez pro-

fond, l'algorithme sélectionne le bras qui sera joué dans cette feuille en utilisant un algorithme d'identification du meilleur bras. Les garanties théoriques obtenues montrent la quasi-optimalité de l'approche. Les garanties théoriques de cet algorithme sont ensuite utilisées dans le chapitre 6 dans une analyse complète de la sélection d'experts apprenants.

Le chapitre 6 pose le problème de la sélection d'experts apprenants dégagé dans le chapitre 4 et propose plusieurs méthodologies permettant leur sélection en ligne. Les trois méthodes proposées correspondent chacune à un niveau de connaissance sur les garanties théoriques de l'algorithme. L'algorithme LTEE permet de sélectionner le meilleur expert lorsqu'une borne sur le nombre d'observations nécessaires à sa convergence vers le niveau performance souhaité est connue. L'algorithme LEE est utilisable lorsqu'une borne sur la différence cumulée entre les performances moyennes de l'algorithme durant son apprentissage et ses performances cumulées après convergence est connue. Finalement, nous analysons l'utilisation d'EXP3, déjà utilisé dans ce cadre dans le chapitre 4, pour le cas où la différence des performances moyennes cumulées existe, mais est inconnue du joueur.

3 Publications

Ces travaux de thèse ont donné lieu aux publications scientifiques suivantes :

Conférences internationales à comité de lecture

- R Allesiardo, R Féraud, D Bouneffouf. A neural networks committee for the contextual bandit problem. International Conference on Neural Information Processing (ICONIP), 2014.
- R Allesiardo, R Féraud. EXP3 with drift detection for the switching bandit problem. IEEE International Conference on Data Science and Advanced Analytics (DSAA), 2015.
- R Féraud, R Allesiardo, T Urvoy, F Clérot. Random Forest for the Contextual Bandit Problem. Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTAT), 2016.

Conférences nationales à comité de lecture

- R Allesiardo, R Féraud, D Bouneffouf. Prise de décision contextuelle en bande organisée : Quand les bandits font un brainstorming. CAP'14 3 (16), 2014.
- R Allesiardo, R Féraud.Un Algorithme pour le Probleme des Bandits Manchots avec Stationnarité par Parties. CAP'15, 2015.

Prépublications

- R Allesiardo, R Féraud. Selection of Learning Experts.
- R Allesiardo, R Féraud, O-A Maillard. Random Shuffling and Resets for the Non-stationary Stochastic Bandit Problem.

Première partie

Le Problème des Bandits manchots

Chapitre 1

Formalisme général et état de l'art

Contents

1.1 Bandits manchots stationnaires

1.1.1 Formalisation

Un problème de bandits manchots stochastiques est composé d'un ensemble $[K]$ comprenant K bras, chacun associé à une distribution de récompense $D(\mu_k)$ de moyenne $\mu_k \in [0,1]$ et de support $[0,1]$. À chaque tour $t \in [1,T]$, un joueur choisit un bras k_t et observe la récompense y_{k_t} tirée depuis $D(\mu_{k_t})$. Le bras optimal k^* est celui permettant d'obtenir la plus haute récompense en moyenne à chaque tour :

$$k^* = \arg\max_{1 \leq k \leq K} \mu_k \,.$$

La récompense moyenne de k^* est :

$$\mu^* = \max_{1 \leq k \leq K} \mu_k \,.$$

La performance des algorithmes de bandits manchots peut-être mesurée de deux manières différentes, par la complexité d'échantillonnage ou grâce à la notion de regret.

Complexité d'échantillonnage

En apprentissage (ϵ, δ)-PAC, la complexité d'échantillonnage (sample complexity en anglais) [9] est le nombre d'observations nécessaires à un algorithme pour estimer une fonction à ϵ près avec une probabilité d'au moins $1 - \delta$. Dans le formalisme des bandits manchots, cela revient à identifier, avec une probabilité d'au moins $1-\delta$, un bras k ayant une récompense moyenne satisfaisant $\mu_k \geq \mu^* - \epsilon$.

Nous présentons les algorithmes dont le critère de performance est la complexité d'échantillonnage en sous-section 1.1.2.

Regret cumulé

Le regret cumulé est la différence entre le gain de la politique du joueur lorsque l'horizon T est atteint et celui qui aurait pu être obtenu par une politique, dite optimale, connaissant les paramètres cachés du problème. Dans le cas présent, la politique optimale a la connaissance des récompenses moyennes de chaque bras et joue k^*, le bras ayant la plus haute récompense moyenne.

Le gain de la politique optimale sur l'ensemble du jeu est :

$$G^* = \sum_{t=1}^{T} y_{k^*}(t) \, .$$

L'espérance du gain de la politique optimale est :

$$\mathbb{E}[G^*] = T\mu^* \, .$$

Le regret cumulé de l'algorithme ayant joué la séquence de bras $k_1, ..., k_T$ est :

$$R(T) = G^* - \sum_{t=1}^{T} y_{k_t}(t) \, ,$$

Le pseudo-regret cumulé de l'algorithme ayant joué la séquence de bras $k_1, ..., k_T$ est :

$$\bar{R}(T) = \mathbb{E}\left[\sum_{t=1}^{T} \Delta_{k_t}\right] \, ,$$

où $\Delta_{k_t} = \mu^* - \mu_{k_t}$.

Le pseudo-regret cumulé peut aussi être écrit de la manière suivante :

$$\bar{R}(T) = \sum_{k=1}^{K} \mathbb{E}\left[T_k\right] \Delta_k \,,$$

où T_k est le nombre de tirages du bras k au cours du jeu. Dans les sous-sections suivantes t_k est défini comme le temps local au bras k, c'est-à-dire, le nombre de fois où le bras k a été joué jusqu'à présent.

Il est possible de transposer une borne supérieure sur la complexité d'échantillonnage en une borne sur le pseudo-regret en multipliant le nombre de fois où un bras sous-optimal k a été joué par son regret instantané Δ_k.

Les algorithmes cherchant à minimiser le regret cumulé sont présentés en sous-section 1.1.3.

1.1.2 Algorithmes identifiants le meilleur bras

Dans les problèmes d'identification du meilleur bras, le joueur cherche à retourner le meilleur bras possible après une période d'exploration. Nous reprenons l'exemple proposé dans [10] : une entreprise possède K variantes d'un produit et doit identifier la meilleure avant de la commercialiser. Durant une phase de test, les différentes variantes sont présentées à des clients potentiels. Chaque client essaye un produit et lui attribue une note (la récompense). L'intérêt de cette campagne est d'identifier le meilleur produit afin de le lancer sur le marché. Les scores obtenus durant la campagne de test (la récompense cumulée) ne sont pas importants.

Les algorithmes d'identification du meilleur bras laissent apparaitre deux principales contraintes.

- **La contrainte de confiance.** Lorsque la confiance sur le meilleur bras est recherchée, la probabilité de retourner le meilleur bras est définie avant le début du jeu. Dans le cas des bandits manchots stochastiques stationnaires, cela revient à chercher des algorithmes retournant, avec une probabilité $1-\delta$, un bras ayant au minimum une récompense moyenne de $\mu^* - \epsilon$ en un minimum de tours de jeu. C'est notamment le cas de l'algorithme Successive Elimination, présenté dans cette sous-section.
- **La contrainte de budget.** Lorsque le budget est contraint, le nombre maximal de tours de jeu est défini et connu du joueur qui doit alors maximiser la probabilité de trouver le meilleur bras en dépensant librement la totalité du budget (voir [11] et les algorithmes d'exploration unifiée, présentés dans la suite de cette sous-section, par exemple).

Borne inférieure sur la complexité d'échantillonage

La borne inférieure sur la complexité d'échantillonnage du problème d'identification du meilleur bras est montrée dans le théorème 1 [12]. Cette borne est limitée au cas où les récompenses sont tirées depuis des distributions de Bernoulli.

Théorème 1. *Il existe des constantes positives c_1, c_2, ϵ_0 et δ_0, telles que pour tout $K \geq 2$, $\epsilon \in [0, \epsilon_0]$, $\delta \in (0, \delta_0)$, et pour toute politique (ϵ, δ)-correcte, il existe $\mu^k \in [0,1]^K$ tel que :*

$$\mathbb{E}[L] \geq c_1 \frac{K}{\epsilon^2} \log(\frac{c_2}{\delta}) \,,$$

où L est le nombre de tirages ayant été nécessaires pour trouver la dite politique.

Algorithmes d'éliminations

Les algorithmes d'éliminations [13, 14] jouent tous les bras de manière séquentielle, testent un critère pour déterminer les bras sous-optimaux, les éliminer de l'ensemble des bras joués et finalement retourner le bras optimal.

Une manière naïve est d'utiliser une approche « explorer puis exploiter » en définissant la taille de la période d'exploration en fonction de la probabilité de succès et de la précision désirée. Pour que le bras ayant la plus haute moyenne empirique soit ϵ-optimal avec une probabilité $1 - \delta$, en se basant sur l'inégalité d'Hoeffding [15], il est nécessaire de jouer $\frac{4}{\epsilon^2} \log(\frac{2K}{\delta})$ fois chaque bras. Bien que naïve, cette approche obtient une complexité d'échantillonnage en $O(\frac{K}{\epsilon^2} \log \frac{K}{\delta})$. Ce résultat est basé sur l'inégalité d'Hoeffding qui fait le lien entre le nombre d'observations, l'écart à la moyenne et la probabilité avec laquelle cet intervalle de confiance est valide.

Cependant, cette approche n'est pas optimale et il est possible de mettre en place une stratégie d'éliminations successives. L'algorithme se décompose alors en deux parties. Dans la première, chaque action disponible est jouée une fois. Dans la seconde partie, un intervalle de confiance ϵ_{t_k} est calculé à partir du nombre d'observations utilisées dans le calcul de la moyenne empirique de la récompense de chaque bras. L'intervalle de confiance ϵ_{t_k} est défini de la manière suivante :

$$\epsilon_{t_k} = \sqrt{\frac{\log(cK{t_k}^2/\delta)}{t_k}} \tag{1.1}$$

où $c > 4$ est une constante.

Le bras ayant la plus haute moyenne empirique $\hat{\mu}_{\max}(t)$ est nommé $k_{\max}(t)$. Si il existe une action k telle que $\hat{\mu}_{\max}(t) - \hat{\mu}_k(t) > 2\epsilon_{t_k}$ alors l'action k est éliminée.

Algorithme 2 : SUCCESSIVE ELIMINATION

$t = 0$, $S = [K]$ et $c > 4$
pour $t = 1; t \leq T; t++$ **faire**
  // `Phase d'exploration`
  Jouer l'action $RR(S)$
  // $RR(S)$ `une fonction retournant le prochain élément dans` S `en suivant un round-robin` [1]
  // `Phase d'élimination`
  $\hat{\mu}_{\max}(t) = \max_{k \in S} \hat{\mu}_k(t)$
  Retirer de S toutes les actions $k \in S$ telles que :
  $\hat{\mu}_{\max}(t) - \hat{\mu}_k(t) \geq 2\sqrt{\log(ct_k^2 K/\delta)/t_k}$.
fin

Théorème 2. *Pour tout $K > 1$, supposons que $\Delta_k > 0$ pour chaque $k \neq k_{\max}$. L'algorithme* SUCCESSIVE ELIMINATION *est un algorithme $(0, \delta)$-PAC et, avec une probabilité d'au moins $1 - \delta$, la complexité d'échantillonnage est bornée supérieurement par :*

$$O\left(\sum_{k \neq k_{\max}} \frac{\log(\frac{K}{\delta \Delta_k})}{\Delta_k^2}\right).$$

Il est possible de transformer cette borne en borne sur le pseudo-regret dépendant de T en multipliant le nombre de fois où chaque bras sous optimal est joué par son regret moyen instantané Δ_k et en fixant $\delta = 1/T$. Nous obtenons ainsi le corolaire suivant :

Corollaire 1. *Pour tout $K > 1$, supposons que $\Delta_k > 0$ pour chaque $k \neq k_{\max}$. Avec une probabilité d'au moins $1 - \frac{1}{T}$, le pseudo-regret de* SUCCESSIVE ELIMINATION *est borné supérieurement :*

$$O\left(\sum_{k \neq k_{\max}} \frac{\log(\frac{KT}{\Delta_k})}{\Delta_k}\right).$$

L'algorithme SUCCESSIVE ELIMINATION joue un rôle important dans les contributions de la thèse et sert de base à plusieurs des algorithmes proposés. La variante de SUCCESSIVE ELIMINATION, nommée MEDIAN ELIMINATION [14] élimine les bras dont la récompense moyenne empirique est inférieure à la médiane sur l'ensemble des bras. Contrairement à SUCCESSIVE ELIMINATION, MEDIAN ELIMINATION est optimal. Il souffre cependant de performances médiocres dans ses applications pratiques.

1. Un round-robin parcours les éléments d'un ensemble ordonné un par un. Après avoir parcouru le dernier élément, il recommence son parcours à partir du premier.

Algorithme d'exploration unifiée

Les algorithmes d'exploration unifiée UGAPEB et UGAPEC [10] utilisent la même stratégie de sélection de bras (à un facteur multiplicatif près dans l'intervalle de confiance) pour les cas à budget contraint et confiance contrainte.

Algorithme 3 : UGAPEB

Données : ϵ, m, T, a
Jouer chaque bras une fois et mettre à jour les moyennes empiriques $\hat{\mu}_k(t)$;
pour $t = K; t \leq T; t++$ **faire**
  SelectionDuBras(t);
fin
Retourner $\arg\min_{J(t)} B_{J(t)}(t)$;

Algorithme 4 : UGAPEC

Données : ϵ, m, δ, c
Jouer chaque bras une fois et mettre à jour les moyennes empiriques $\hat{\mu}_k(t)$;
$t = K + 1$ **répéter**
  SelectionDuBras(t) ; $t++$;
jusqu'à $B_{J(t)} \geq \epsilon$;
Retourner $J(t)$;

Algorithme 5 : SELECTIONDUBRAS

Données : $t > 0$
Identifier l'ensemble des m bras $J(t) \in \arg\min_{k\in[K]}^{1,\dots,m} B_k(t)$;
Jouer le bras $k_t = \arg\max_{k\in\{l_t,u_t\}} \beta_k(t-1)$;
Mettre à jour $\hat{\mu}_k(t)$;

Nous définissons l'index $B_k(t)$:

$$B_k(t) = \max_{i\neq k}^{m} U_i(t) - L_k(t)$$

où $U_k(t) = \hat{\mu}_k(t-1) + \beta_k(t-1)$ est l'intervalle de confiance supérieur du bras k et $L_k = \hat{\mu}_k(t-1) - \beta_k(t-1)$ son intervalle de confiance inférieur. Les extremums des différents types d'intervalle de confiance sont $u_t = \arg\max_{j\notin J(t)} B_k(t)$ et $l_t = \arg\min_{j\in J(t)} B_k(t)$.

Les largeurs des intervalles de confiance sont différentes suivant la cible :

$$\text{UGAPEB} : \beta_k(t-1) = b\sqrt{\frac{a}{t_k}}, \quad \text{UGAPEC} : \beta_k(t-1) = b\sqrt{\frac{c\log\frac{4K(t-1)^3}{\delta}}{t_k}}.$$

La complexité du problème d'identification du meilleur bras est :

$$H_\epsilon = \sum_{i=1}^{K} \frac{b^2}{\max(\frac{\Delta_i+\epsilon}{2}, \epsilon)^2}.$$

Théorème 3. Budget fixé. *Pour tout $K > 2$, si* UGAPEB *est exécuté avec comme paramètre $0 < a < \frac{T-K}{4H_\epsilon}$ alors tout bras k retourné par l'algorithme vérifie :*

$$P(\mu_{k^*} - \mu_k > \epsilon) \leq 2KTexp(-2a).$$

Théorème 4. Confiance fixé. *Pour tout $K > 2$, si* UGapEc *est exécuté avec comme paramètre $c = 0.5$ alors sa complexité d'échantillonnage est bornée supérieurement par :*

$$O\left(K + H_\epsilon \log \frac{H_\epsilon}{\delta}\right).$$

1.1.3 Algorithmes minimisant le regret

Dans cette section, nous présentons plusieurs autres algorithmes de bandits stochastiques optimisant le regret cumulé.

Bornes inférieures sur le pseudo-regret cumulé

Un résultat similaire concernant la borne inférieure sur le regret cumulé a été démontré par Lai et Robbins [8]. Celui-ci montre que tout algorithme convergeant vers le bras optimal joue asymptotiquement chacun des bras sous-optimaux $k \neq k^*$ au moins $\log(T)C(D(\mu^*), D(\mu_k))$ fois. La valeur de $C(D(\mu^*), D(\mu_k))$ est inversement proportionnelle) la divergence de Kullback-Leibler entre la distribution $D(\mu^*)$ et la distribution $D(\mu_k)$. En effet, plus les distributions sont proches, plus il est difficile de séparer le bras optimal des bras sous-optimaux.

Théorème 5. *[16] Pour une stratégie satisfaisant $\mathbb{E}[T_k] = o(T^a)$ pour toutes récompenses tirées depuis des distributions de Bernoulli, tout bras k avec $\Delta_k > 0$ et tout $a > 0$, on a :*

$$\underline{\lim}_{T\to\infty} \bar{R}(T) \geq \sum_{k\in[K]\setminus k^*} \frac{\log(T)\Delta_k}{kl(\mu_k, \mu^*)}.$$

Afin de comparer cette borne avec celle de la complexité d'échantillonnage, nous pouvons utiliser l'encadrement suivant :

$$2(\mu_k - \mu^*)^2 \leq \mathrm{kl}(\mu_k, \mu^*) \leq \frac{(\mu_k - \mu^*)^2}{\mu_k(1-\mu^*)}.$$

$$\sum_{k\in[K]\setminus k^*} \frac{\log(T)(\mu_k - \mu^*\mu_k)}{\Delta_k} \leq \sum_{k\in[K]\setminus k^*} \frac{\log(T)\Delta_k}{\mathrm{kl}(\mu_k, \mu^*)} \leq \sum_{k\in[K]\setminus k^*} \frac{\log(T)}{\Delta_k}.$$

Cet encadrement nous permet de remarquer qu'un algorithme dont la complexité d'échantillonnage est optimale est aussi optimal vis-à-vis du pseudo-regret (à un facteur d'au plus $\mu_k - \mu^*\mu_k$ et aux constantes c_1 et c_2 près).

γ-GREEDY

Le principe des approches γ-GREEDY [2] est de jouer un bras au hasard avec une proportion $\gamma \in [0, 1]$ et le reste du temps, de jouer le bras ayant la plus haute estimation.

Les phases d'exploration peuvent être distribuées de trois façons différentes :

- L'approche **explorer puis exploiter** explore durant γT itérations avant de jouer uniquement le bras avec la plus haute moyenne empirique. Le test A/B, où deux bras sont joués uniformément avant d'éliminer la moins bonne, est un exemple d'approche explorer puis exploiter.
- Les phases d'exploration peuvent être aussi tirées avec une probabilité constante γ. Le problème de cette approche est de souffrir d'un regret linéaire lorsque l'horizon est infini.
- Il est aussi possible de faire décroitre γ au cours du temps afin d'obtenir un regret sous-linéaire. La vitesse à laquelle décroit γ doit cependant être contrôlée afin de garantir la convergence de l'algorithme.

L'algorithme γ_t-GREEDY [17] est un exemple de stratégie γ-GREEDY possédant un γ décroissant permettant d'obtenir des garanties logarithmiques sur le regret instantané. γ_t est défini de la manière suivante :

$$\gamma_t = \min\{1, \frac{cK}{d^2 t}\}, \tag{1.2}$$

où $c > 0$ est une constante.

Théorème 6. *Pour tout $K > 1$ si γ_t-GREEDY est exécuté avec comme paramètre $0 < d \leq \min_{k:\mu_k<\mu^*} \Delta_k$ alors la probabilité qu'à n'importe quel tour $t \geq cK/\epsilon$ γ_t-GREEDY choisisse un bras sous-optimal est au plus :*

$$\frac{c}{\epsilon^2 t} + 2\Big(\frac{c}{\epsilon^2}\log\frac{(t-1)\epsilon^2 e^{1/2}}{cK}\Big)\Big(\frac{cK}{(t-1)\epsilon^2 e^{1/2}}\Big)^{c/(5\epsilon^2)} + \frac{4e}{\epsilon^2}\Big(\frac{cK}{(t-1)\epsilon^2 e^{1/2}}\Big)^{c/2}$$

avec $\Delta_k = \mu^* - \mu_k$.

Intervalles de confiance supérieur

L'algorithme UCB [17] (Upper-Confidence Bound) est basé sur l'optimisme face à l'incertitude. Il utilise la borne de Chernoff-Hoeffding afin de calculer un intervalle de confiance sur la récompense moyenne de chaque bras. Une estimation optimiste (ou index) de la récompense est calculée pour chaque bras en ajoutant la moyenne empirique de ses récompenses à son intervalle de confiance. Celui avec le plus haut index est joué. Contrairement aux algorithmes d'élimination ayant une

2. Cette approche est habituellement appelée ϵ-GREDDY. Pour éviter les conflits avec le ϵ correspondant aux approximations, γ est utilisé pour quantifier l'exploration uniforme.

exploration uniforme des actions non éliminées, l'exploration d'UCB est contrôlée dynamiquement par l'intervalle de confiance et dépend à la fois des moyennes empiriques, du nombre d'observations ainsi que du nombre d'itérations depuis le début du jeu. La valeur de l'intervalle de confiance est :

$$\sqrt{\frac{2\log t}{t_k}}, \tag{1.3}$$

où t_k est le nombre de fois où le bras k a été joué jusqu'à présent.

Algorithme 6 : UCB

pour $t = 1; t \leq T; t++$ **faire**
Si un bras k n'a pas encore été joué, jouer celui-ci, sinon
Jouer le bras k qui maximise :

$$\hat{\mu}_k(t) + \sqrt{\frac{2\log(t)}{t_k}}$$

fin

Théorème 7. *Pour tout $K > 1$ l'espérance du pseudo-regret d'UCB vérifie :*

$$\mathbb{E}[\bar{R}(T)] \leq \big[8 \sum_{k \neq k^*} \big(\frac{\log T}{\Delta_k}\big)\big] + \big(1 + \frac{\pi^2}{3}\big)\big(\sum_{j=1}^{K} \Delta_j\big)$$

avec $\Delta_k = \mu^* - \mu_k$.

Cette borne montre l'optimalité théorique de l'algorithme UCB pour le problème des bandits manchots stochastiques. Il est possible d'accélérer la convergence de l'algorithme UCB en multipliant l'intervalle de confiance d'UCB par un terme dépendant de la variance observée des bras. Bien que dénuée de borne, cette version d'UCB est souvent plus performante que la version originale. D'autres versions d'UCB ont été proposées, utilisant notamment la divergence de Kullback-Leibler avec KL-UCB [18] ou des hypothèses Bayésiennes avec Bayses-UCB [19].

Échantillonnage des observations

Le Thompson Sampling [7] est l'une des plus vieux algorithmes pour le problème des bandits manchots stochastiques. C'est un algorithme randomisé basé sur des idées Bayésiennes. Une distribution a priori est utilisée pour modéliser la récompense associée à chaque bras. À chaque tour, un tirage est effectué dans chaque distribution et le bras ayant obtenu le tirage le plus élevé est joué. La récompense obtenue est alors utilisée pour mettre à jour la distribution du bras joué. Lors des premiers tours de jeu, les distributions seront proches de l'uniforme avant de se concentrer autour de la moyenne empirique de chaque bras. Ce mécanisme permet de réduire l'exploration au cours du temps au profit de l'exploitation.

L'espérance du regret de cet algorithme a été analysé dans le cas des lois Betas et Gaussiennes, obtenant une borne supérieure en $O(\sqrt{KT\log T})$ [20, 21]. Cette borne est presque optimale si l'a priori utilisé est une loi Beta et optimale dans le cas d'une loi Gaussienne.

L'algorithme BESA (Best Empirical Sample Average) [22] suit une idée similaire basée sur le sous-échantillonnage. Afin d'obtenir une comparaison équitable entre les différents bras, les observations de chaque bras sont sous-échantillonnées de manière à ce que les moyennes soient calculées avec le même nombre d'observations. Le bras obtenant la plus haute moyenne empirique après sous-échantillonnage est joué. L'analyse de cet algorithme montre que la borne supérieure de son regret cumulé est logarithmique. Contrairement au Thompson Sampling, BESA est non paramétrique et ainsi plus robuste, ne dépendant pas de la capacité de la distribution a priori à modéliser la vraie distribution générant les récompenses. Bien qu'offrant souvent des performances supérieures aux autres algorithmes, BESA souffre d'une plus grande complexité algorithmique due aux multiples échantillonnages.

1.2 Bandits manchots non-stationnaires

Dans un cadre applicatif, la stationnarité des données peut souvent être remise en cause, la distribution de probabilité générant les données pouvant changer au cours du temps. Cette section présente les modélisations de la non-stationnarité existantes dans le cadre des bandits manchots ainsi que les algorithmes associés

1.2.1 La stationnarité par parties

Un problème de bandit avec stationnarité par parties [23] est composé d'un ensemble $[K]$ contenant K bras, où $1 \leq k \leq K$ est l'indice de chaque bras. Chaque pas de temps $1 \leq t \leq T$ est associé au vecteur de récompense $\mathbf{y}(t) = (y_1(t), ..., y_K(t)) \in \{0,1\}^K$. Chaque récompense est tirée depuis une distribution de support $[0,1]$ et de moyenne $\mu^k(t)$.
Les ruptures. Sur la majorité du jeu, les moyennes des récompenses ne changent pas, i.e. $\forall k, \mu^k(t) = \mu^k(t+1)$. Cependant, ces moyennes peuvent changer sur $M-1$ points de ruptures choisies à l'avance par un adversaire, i.e. $\exists k, \mu^k(t) \neq \mu^k(t+1)$.
La politique optimale. Le jeu est divisé en $N \leq M$ segments. S est l'indice du segment contenant les pas de temps $[T_S, T_{S+1}[$. Un segment S commence quand $\arg\max_k \mu^k(T_S) \neq \arg\max_k \mu^k(T_S - 1)$. Le bras optimal sur le segment S est noté k_S^*, avec :

$$k_S^* = \arg\max_k \mu^k(T_S)$$

L'intérêt de cette modélisation est de pouvoir utiliser les outils ayant des contraintes de stationnarité entre les différents points de rupture.

Borne inférieure

Contrairement aux bandits stochastiques stationnaires, lorsque le processus générant les récompenses est non-stationnaire, le meilleur bras peut changer au cours du jeu et une exploration constante est nécessaire. Cette exploration supplémentaire empêche l'algorithme de converger totalement vers un unique bras et d'obtenir un regret cumulé logarithmique. La bonne inférieure dans le cas de distributions non-stationnaires est en effet de $\Omega(\sqrt{T})$ [23]. La démonstration porte sur le cas des bandits stochastiques avec deux changements de distributions entrainant un changement de meilleur bras. Celle-ci est basée sur l'intuition qu'un algorithme explorant peu nécessitera beaucoup d'itérations avant de détecter les changements de distribution ayant rendu sous-optimal un bras précédemment optimal.

Modélisation par chaîne de Markov

Supposer qu'un processus est stationnaire par parties revient à considérer que celui-ci est contrôlé par une chaîne de Markov. Une chaîne de Markov est un graphe orienté où chaque nœud représente un état et les arêtes les probabilités de transition entre états. La propriété caractéristique des chaînes de Markov est l'indépendance entre le futur et le passé : la probabilité d'apparition d'un évènement dépend uniquement de l'état courant.

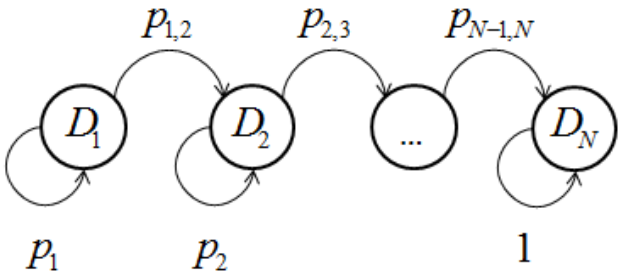


FIGURE 1.1 – Une chaîne de Markov représentant les changements de distribution dans un processus stationnaire par parties.

La figure 1.1 est un exemple de chaîne de Markov représentant les changements de distribution d'un processus stationnaire par parties. Chaque réalisation du processus est tirée depuis la distribution D_n où $n \leq N$ est l'état du système. Après chaque tirage depuis D_n, le système reste dans le même état avec la probabilité p_n et change d'état avec une probabilité $p_{n,n+1}$.

Cette modélisation est approfondie avec le modèle des « restless bandits » [24, 25, 26]. Dans celui-ci, les probabilités de transition peuvent être différentes suivant si un bras a été joué ou non. La politique optimale des algorithmes de « restless bandits » maximise la récompense obtenue en tirant parti des probabilités de transition. On peut imaginer un « Restless bandits » à deux bras et deux états. Dans les deux états le même bras est optimal, mais sa récompense moyenne est

supérieure dans le second état. Si la seule manière de passer dans le second état est de jouer un bras sous-optimal du premier état alors la politique optimale serait de jouer ce bras sous-optimal en début de partie pour changer d'état et pouvoir jouer le bras optimal ayant maintenant une récompense moyenne plus élevée. Cette modélisation est très contraignante, les algorithmes existant (notamment dans [26] atteignant un regret cumulé de l'ordre de $O\left((\log T)^{1/3}T^{2/3}\right)$.

Les autres formalismes présentés dans ce manuscrit considèrent que les actions du joueur n'ont pas d'impact sur l'évolution de la non-stationnarité. Transposé dans un cadre de « restless bandits », cela signifie que les probabilités de transition, sachant si un bras a été joué ou non, sont identiques.

D-UCB et UCB à fenêtre glissante

Les algorithmes D-UCB [27] et UCB à fenêtre glissante (SW-UCB) [23] sont des variantes d'UCB pour le problème des bandits avec stationnarité par parties. Dans D-UCB, les observations utilisées pour calculer la moyenne sont pondérées par un facteur α diminuant au cours du temps. La moyenne est ainsi biaisée en faveur des observations plus récentes. La somme des pondérations des observations associées à un bras est utilisée dans l'intervalle de confiance, lui permettant de grandir au cours du temps pour que les bras considérés comme sous-optimaux soient à nouveau explorés. L'analyse de D-UCB montre une borne supérieure du regret cumulé en $O(\sqrt{MT\log T})$. Dans SW-UCB, un historique de taille H est utilisé pour stocker les informations de chaque bras. Lorsqu'une nouvelle observation est ajoutée à un historique alors que celui-ci a déjà atteint sa taille maximale la plus ancienne est retirée. L'algorithme obtient aussi une borne sur le pseudo-regret du même ordre que D-UCB, à savoir $O(\sqrt{MT\log T})$. Bien que cette borne soit éloignée de la borne inférieure d'un facteur $\sqrt{\log T}$, aucun algorithme de bandits s'adaptant aux changements de bras ne possède de borne plus serrée.

Algorithme 7 : D-UCB ET SW-UCB

Données : $c > 0$
pour $t = 1; t \leq T; t++$ **faire**
  Si un bras k n'a pas encore été joué, jouer celui-ci, sinon
  Jouer le bras k qui maximise :

$$\breve{\mu}_k(t) + c\sqrt{\frac{\log \sum_{i=1}^{K} \breve{t}_i}{\breve{t}_k}}$$

fin

Dans le cas de D-UCB,

$$\breve{\mu}_k(t) = \frac{1}{\breve{t}_k}\sum_{i=1}^{t} \alpha^{t-i} y_{k_t} [\![k = k_t]\!] \tag{1.4}$$

et

$$\breve{t}_k = \sum_{i=1}^{t} \alpha^{t-i} [\![k = k_t]\!] \,. \tag{1.5}$$

Dans le cas de SW-UCB,

$$\breve{\mu}_k(t) = \frac{1}{\breve{t}_k} \sum_{i=\max\{0,t-H+1\}}^{t} y_{k_t} [\![k = k_t]\!] \tag{1.6}$$

$$\breve{t}_k = \sum_{i=\max\{0,t-H+1\}}^{t} [\![k = k_t]\!] \,. \tag{1.7}$$

Meta-Eve

Alors que D-UCB et SW-UCB utilisent une approche passive basée sur l'oubli régulier du passé, Meta-Eve [28] utilise une approche active basée sur un détecteur de rupture. L'algorithme fait l'hypothèse que lorsqu'un changement de stationnarité se produit, cela induit un changement dans la moyenne du bras optimal. En suivant ce pragmatisme, un algorithme de bandit stochastique peut être utilisé et les changements de meilleur bras gérés en détectant les variations de moyenne du meilleur bras empirique puis en réinitialisant l'algorithme. Dans son implémentation, Meta-Eve utilise l'algorithme UCB pour le choix des bras et un test de Page-Hinckley [29] pour détecter les changements de moyenne. Afin d'être plus robuste aux faux positifs du test de détection, un second UCB est utilisé comme Meta-Bandit durant les itérations suivant une réinitialisation avec comme bras, l'UCB utilisé au moment du test et l'UCB nouvellement initialisé. L'algorithme peut cependant difficilement détecter les changements lorsqu'un bras sous-optimal devient optimal, ceux-ci étant peu joués par l'algorithme. Bien qu'en pratique cet algorithme offre de bonnes performances lorsque ses hypothèses sont respectées, il ne dispose pas de garanties théoriques.

1.2.2 Bandit manchots avec adversaire

L'une des formes de non-stationnarités parmi les plus fortes est l'hypothèse adverse. Celle-ci suppose que les éléments de la séquence générée par le processus sont définis par un adversaire. Celui-ci peut-être inconscient, dans ce cas, on considère que les éléments sont générés à l'avance et ne changent pas quelle que soit la nature des interactions avec le système. Lorsque d'adversaire est conscient, celui-ci peut définir les éléments de la séquence tout en ayant conscience des interactions. Toutes les formes de non-stationnarité peuvent être, à posteriori, représentées par l'adversité, rendant les algorithmes basés sur celle-ci très robustes. Suivant le formalisme de [30, 31], nous considérons que la séquence de récompenses $Y(1), ..., Y(T)$ avec $Y(t) = (y_1(t), ..., y_K(t))$ est définie à l'avance par un adversaire inconscient.

Borne inférieure

Le théorème suivant donne une borne inférieure sur le regret vis-à-vis du bras ayant la récompense cumulée la plus haute dans le cas où les récompenses sont choisies par un adversaire inconscient. Remarquons que cette borne inférieure s'applique aussi lorsque les séquences sont tirées depuis des distributions stochastiques non-stationnaires. En effet, une séquence de récompenses adverses est équivalente à une séquence de récompenses tirées depuis des distributions de moyenne 1 ou 0 et de variance nulle.

Le gain de la politique optimale est :

$$G^* = \max_{k \in [K]} \sum_{t=1}^{T} y_k(t) \,. \tag{1.8}$$

Le regret faible est l'espérance du regret cumulé lorsque celui-ci prend en compte la randomisation des choix de l'algorithme :

$$\mathbb{E}\left[R\right] = \sum_{t=1}^{T} \left(y_k(t) - \sum_{k=1}^{K} \mathbf{P}_t(k_t = k) y_{k_t}(t) \right) . \tag{1.9}$$

Théorème 8. *[31] Pour tout $K \geq 2$ et tout horizon T, il existe une distribution sur les séquences de récompenses telle que l'espérance du regret faible (où l'espérance est prise vis-à-vis de la randomisation des récompenses et de la randomisation interne de l'algorithme) est au moins :*

$$\Omega \left(\frac{1}{20} \min\{\sqrt{KT}, T\} \right) .$$

EXP3

EXP3 [30, 31] est un algorithme de bandits manchots avec adversaire. De manière similaire aux approches γ-GREEDY, l'algorithme possède une exploration constante contrôlée par le paramètre $\gamma \in [0, 1]$. Les performances de chaque bras sont estimées en maintenant un estimateur débiaisé de la récompense cumulée du bras :

$$\mathbb{E}\left[\sum_{t=1}^{T} \frac{y_k(t) \mathbf{1}[i = k_t]}{\mathbf{P}_t(k)} \right] = \sum_{t=1}^{T} y_k(t) \,, \tag{1.10}$$

où $\mathbb{E}[.]$ est l'espérance conditionnelle à la probabilité $\mathbf{P}_t(k)$ de choisir le bras k au tour t. $\mathbf{P}_t(k) = \mathbf{P}(k|k_1, ..., k_{t-1})$.

L'exploitation est effectuée en utilisant une distribution de Gibbs prenant comme paramètres les estimateurs débiaisés de chaque bras. L'aléa introduit par le mélange des distributions d'exploration et d'exploitation est l'une des grandes

forces de l'algorithme face à l'adversité. En effet, si un algorithme est déterministe et connu de l'adversaire, il est possible de construire une séquence de récompense lui infligeant un regret maximal. La randomisation des choix d'EXP3 lui permet de se prémunir contre ce type de scénarios.

Algorithme 8 : EXP3

Données : $\gamma \in [0,1]$

La probabilité de choisir le bras k au tour t est :

$$\mathbf{P}_t(k) = (1-\gamma)\frac{w_t^k}{\sum_{i=1}^K w_t^i} + \frac{\gamma}{K}\,, \tag{1.11}$$

avec pour tout k, $w_0^k = 1$ et :

$$w_{t+1}^i = w_t^i \exp\left(\frac{\gamma \mathbf{1}[i = k_t] y_{k_t}(t)}{\mathbf{P}_t(i) K}\right) \tag{1.12}$$

Le regret de l'algorithme est calculé face à une politique jouant un seul bras durant la totalité du jeu. Sous ces conditions, le gain de la politique optimale est :

$$G^* = \max_{k\in[K]} \sum_{t=1}^{T} y_k(t)\,. \tag{1.13}$$

Cette politique est cependant très faible si les bras possèdent des récompenses cumulées similaires mais concentrées dans des périodes de temps distinctes.

Théorème 9. *Le regret des choix de bras d'EXP3 vis-à-vis de la politique optimale vérifie :*

$$G_{max} - \mathbb{E}[G_{EXP3}] \leq (e-1)\gamma G_{max} + \frac{K \log K}{\gamma}$$

Théorème 10. *En supposant que $g \geq G_{max}$ et en exécutant EXP3 avec le paramètre*

$$\gamma = min\{1, \sqrt{\frac{K \log K}{(e-1)g}}\}$$

Alors

$$G_{max} - \mathbb{E}[G_{EXP3}] \leq 2,63\sqrt{gK \log K}$$

Récemment, une variante d'EXP3 a été proposée, basée sur une exploration implicite [32] utilisant uniquement la distribution de Gibbs. Cette exploration implicite est obtenue en remplaçant l'estimation de la récompense cumulée par une estimation de la perte cumulée. L'exploration constante ayant disparu, le terme γ est maintenant utilisé pour biaiser l'algorithme et empêcher les récompenses d'être divisées par une probabilité trop faible. Dans ce cas, les poids sont calculés de la

manière suivante :

$$w_t^i = \exp\left(\sum_{j=1}^{t} \frac{\mathbf{1}[i = k_j]\left(1 - y_{k_j}(j)\right)}{\mathbf{P}_t(i) + \gamma}\right) \tag{1.14}$$

Plus un bras est joué, plus sa perte cumulée augmente si les récompenses sont différentes de 1, provocant ainsi une augmentation de la probabilité de jouer les autres bras.

EXP3.S

Contrairement à EXP3, dont le regret est comparé à une politique jouant le bras ayant la plus haute récompense cumulée sur l'ensemble du jeu, EXP3.S [31] se compare à une politique arbitraire pouvant jouer n'importe quelle séquence de bras. Chaque séquence est caractérisée par sa difficulté N, i.e., le nombre minimal de sous-séquences où le même bras est joué. L'algorithme EXP3.S est obtenu en rajoutant un paramètre de régularisation à l'équation de mise à jour des poids. Cette régularisation permet de diminuer l'importance des récompenses plus anciennes, permettant à l'algorithme de converger vers un bras différent si les récompenses du meilleur bras empirique courant baissent au profit d'un autre.

Algorithme 9 : EXP3.S

Données : $\gamma \in [0, 1]$ et $\alpha > 0$

La probabilité de choisir le bras k au tour t est :

$$\mathbf{P}_t(k) = (1 - \gamma)\frac{w_t^k}{\sum_{i=1}^{K} w_t^i} + \frac{\gamma}{K}\,, \tag{1.15}$$

avec pour tout k, $w_0^k = 1$ et :

$$w_{t+1}^i = w_t^i \exp\left(\frac{\gamma \mathbf{1}[i = k_t] y_t^{k_t}}{\mathbf{P}_t(i) K}\right) + \frac{e\alpha}{K}\sum_{k=1}^{K} w_t^k\,, \tag{1.16}$$

L'espérance du regret cumulé d'EXP3.S est bornée par $O(\sqrt{KT(N\log(KT) + e)})$.

1.2.3 Stochastique et adverse

L'algorithme SAO [33] est motivé par le fait d'avoir un algorithme pouvant être utilisé dans le cas adverse tout en maintenant des garanties logarithmiques dans le cas stochastique. L'algorithme est décomposé en trois phases : une phase d'exploration, une phase d'exploitation et une phase adverse. Lors de la première phase, les bras sont joués uniformément. Les bras sont ensuite éliminés jusqu'à rentrer dans la phase d'exploitation. Durant ces deux phases, les bras éliminés

sont toujours joués avec une faible probabilité (suffisamment petite pour garder un regret logarithmique) et différentes conditions garantissant la stationnarité des bras sont vérifiées à chaque nouveau tirage. Si l'une de ces conditions échoue, l'algorithme passe dans la phase adverse en utilisant EXP3 comme sous-routine. Dans le cas stochastique, le regret cumulé de SAO est borné par $O(\frac{\log^3 T}{\Delta})$ et par $O(\sqrt{T}\log^2 T)$ dans le cas adverse.

L'algorithme EXP3++ [34] est une variante d'EXP3 capable d'être utilisée sur des régimes adverses, stochastiques et pseudo-adverses. L'algorithme obtient une borne sur l'espérance du regret en $O(\sqrt{T})$ face à un adversaire. Dans le régime avec des *récompenses contaminées*, les récompenses sont majoritairement tirées depuis des distributions stationnaires à l'exception de certaines choisies à l'avance par un adversaire. Lorsque ces récompenses sont *modérément contaminées*, c'est-à-dire que l'adversaire n'a pas baissé l'écart entre les bras de plus d'un facteur 1/2, alors EXP++ obtient des garanties en $O(\frac{\log^2 T}{\Delta}) + \tilde{O}(\frac{1}{\Delta^3})$ pour le pseudo-regret cumulé. Cette borne logarithmique peut aussi être obtenue avec des récompenses adverses lorsqu'il existe un pas de temps à partir duquel, pour tout t, la somme entre 1 et t des récompenses du bras optimal est toujours supérieure à celle des autres bras. Cependant, EXP++ nécessite un paramétrage spécifique à chaque régime pour que ces garanties s'appliquent.

1.3 Variantes du problème

De nombreuses variantes du problème des bandits manchots existent. Nous en présentons brièvement quelque unes où la non-stationnarité intervient sur la disponibilité des bras. La publicité en ligne est un exemple pratique où la disponibilité des bras peut-être variable au cours du temps. De nouveaux contrats peuvent être souscrits, faisant apparaitre de nouveaux bras. Les contraintes contractuelles peuvent nécessiter un nombre d'affichage minimum et maximum, forçant l'algorithme à jouer des bras sous-optimaux ou entrainants la disparition d'un bras dont le nombre d'affichage contractuel a été atteint.

- **Bandits dormants.** Le problème des bandits dormants [35] considère le cas où à chaque tour, les bras peuvent « dormir », le joueur ne pouvant ainsi choisir le bras à jouer que parmi un sous-ensemble.
- **Bandits mortels.** Les bandits mortels [36] sont un cas particulier des bandits dormants, les bras ne sont disponibles que pendant un nombre de tours limités puis disparaissent définitivement du jeu.
- **Jeux à gratter.** Les jeux à gratter [37] sont une variante des bandits mortels où la disparition des bras ne dépend pas du nombre de tours. Ici, les bras disposent d'un nombre de tirage limité. Lorsque ce nombre de tirages est atteint, les bras ne sont plus disponibles.

Chapitre 2

Bandits Manchots Stochastiques Non-stationnaires

Contents

2.1 Introduction

Dans la littérature, les bandits manchots stochastiques (voir section 1) utilisent des distributions stationnaires. Lorsque les récompenses sont stationnaires par parties (voir section 1.2.1) les bornes supérieures sur le regret possèdent une dépendance en racine vis-à-vis du nombre de changements de distribution. Jusqu'à récemment, l'alternative à cette stationnarité était la formulation adverse (voir section 1.2.2). Seldin et Slivkins ont cependant proposé diverses formulations du problème mêlant récompenses stationnaires et non-stationnaires (voir section 1.2.3), via l'introduction d'une adversité restreinte, tout en obtenant des garanties logarithmiques sur le regret grâce à un paramétrage spécifique d'EXP3 lorsque le meilleur bras ne change pas. Dans ce chapitre, nous étendons le problème aux bandits manchots stochastiques non-stationnaires. Nous proposerons dans un

premier temps une version randomisée de l'algorithme d'éliminations successives permettant de trouver le meilleur bras en un temps logarithmique. Nous détaillerons ensuite une variante de cet algorithme puis de l'algorithme EXP3 pour le cas où le meilleur bras change au cours du temps.

2.2 Formalisation

Soit $[K] = 1, ..., K$ un ensemble de bras. Après avoir joué le bras k_t au tour $t \leq T$, le joueur obtient une récompense $y_{k_t}(t) \in [0, 1]$ tirée depuis une distribution de moyenne $\mu_{k_t}(t) \in [0, 1]$.

La notion de regret. Soit $k^*(t)$ le bras jouée par la politique optimale au temps t. Le pseudo-regret cumulé est défini comme la différence entre les récompenses moyennes des bras joués par la politique optimale et celles du joueur :

$$\mathbb{E}\left[\sum_{t=1}^{T} \mu_{k^*(t)}(t) - \mu_{k_t}(t)\right] . \tag{2.1}$$

L'écart instantané. Habituellement, l'écart est mesuré sur une séquence stationnaire. La non-stationnarité des distributions stochastiques nécessite l'introduction de l'écart instantané qui est la différence des récompenses moyennes au pas de temps t. L'écart instantané entre les bras k et k' au temps t est :

$$\Delta_{k,k'}(t) = \mu_k(i) - \mu_{k'}(i) . \tag{2.2}$$

2.3 Politique optimale à bras unique

Dans cette section, nous étudions le cas où le bras optimal est unique même si les distributions générant les récompenses sont non-stationnaires. **Le bras optimal** est :

$$k^* = \arg\max_{k \in [K]} \sum_{t=1}^{T} \mu_k(t) . \tag{2.3}$$

2.3.1 Éliminations Successives avec Round Robin Randomisé

Dans cette section, nous présentons l'algorithme Éliminations Successives avec Round-Robin Randomisé (SER3) (voir algorithme 10). L'algorithme est basé sur le mélange aléatoire de l'ensemble des bras utilisés lors des tours de round-robin. SER3 obtient les mêmes garanties en terme de complexité d'échantillonnage

et de pseudo-regret cumulé que SUCCESSIVE ELIMINATION mais sur une plus grande classe de problèmes, délimitée par une hypothèse faible restreignant la non-stationnarité de la séquence de récompenses moyennes.

Le mécanisme d'élimination

La mécanique d'élimination dans les bandits manchots a été proposée par Even-Dar (voir section 1.1.2). Des estimations des récompenses moyennes sont construites en jouant séquentiellement chaque bras. Après τ_{min} tours de round-robin, l'algorithme commence à comparer les performances de chaque bras afin d'éliminer les bras sous-optimaux. Pour cela, une borne inférieure sur la récompense moyenne du meilleur bras empirique est calculée ainsi que des bornes supérieures sur les récompenses moyennes des autres bras. Si la borne inférieure du meilleur bras empire est supérieure à la borne supérieure associée à un autre bras, alors ce bras est éliminé. Ce processus est répété jusqu'à l'élimination de tous les bras à l'exception d'un, le bras optimal.

L'Inégalité d'Hoeffding

SUCCESSIVE ELIMINATION suppose que les récompenses sont tirées depuis des distributions stochastiques identiques au cours du temps. Cependant, l'inégalité d'Hoeffding ne nécessite pas cette contrainte de stationnarité :

Inégalité d'Hoeffding [15]. Si $X_1, X_2, ..., X_\tau$ sont des variables aléatoires indépendantes et que $0 \leq X_i \leq 1$ pour tout $(i = 1, 2, ..., \tau)$, alors pour $\epsilon_\tau > 0$

$$P\left(|\sum_{i=1}^{t} \frac{X_i}{\tau} - \mathbb{E}[\sum_{i=1}^{t} \frac{X_i}{\tau}]| \geq \epsilon_\tau\right) \leq 2\exp\left(-2\epsilon_\tau^2 \tau\right) .$$

Cette inégalité peut être utilisée pour calculer des intervalles de confiances sur des moyennes empiriques composées de récompenses tirées depuis des distributions non-dentiques.

La randomisation des Round-Robin

Nous illustrons les besoins de randomisation de l'algorithme avec un exemple pouvant mettre en échec un algorithme déterministe (voir figure 2.1).

$\mu_k(t)$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
$k=1$	0.6	1	0.6	1	0.6	1
$k=2$	0.4	0.8	0.4	0.8	0.4	0.8

FIGURE 2.1 – Une séquence de récompenses moyennes pouvant mettre en échec un algorithme de bandits déterministes.

Étant donné que $\mu_1(t)$ est plus grand que $\mu_2(t)$ à chaque pas de temps t, le meilleur bras est $k=1$ sans ambiguïté. Cependant, un algorithme déterministe jouant séquentiellement le premier bras puis le second observera uniquement des récompenses tirée depuis une distribution de moyenne $0,6$ pour $k=1$ et de moyenne $0,8$ pour $k=2$. Après suffisamment de temps, un algorithme d'élimination éliminera le premier. SER3 évite ce comportement en mélangeant l'ensemble des bras après chaque tour de round-robin. Cette randomisation est particulièrement nécessaire lorsque les récompenses moyennes sont choisies par un adversaire. En effet, dans une analyse en pire des cas, l'adversaire pourra user du déterminisme de l'algorithme du joueur pour l'empêcher de trouver le bras optimal.

Identification du meilleur bras

Éffectuer une tâche d'identification du meilleur bras suppose l'existence d'un critère permettant d'identifier ce bras à chaque instant, sans ambiguïté. Il est nécessaire de poser une hypothèse assurant qu'il est possible d'identifier k^* tout au long du jeu. Dans un premier temps, nous définissons quelques notations. Une exécution de SER3 est une succession de tours de round-robin. L'ensemble $[\tau] = \{(t_1, |S_1|), ..., (t_\tau, |S_\tau|)\}$ est une réalisation de SER3 et t_i est le pas de temps où le i^{eme} tour de round-robin de taille $|S_i|$ commence. Étant donné que les bras sont seulement éliminés, $|S_i| \geq |S_{i+1}|$. Nous notons $\mathbb{T}(\tau)$ l'ensemble contenant toutes les réalisations possibles de τ tours de round-robin. Nous définissons maintenant l'hypothèse 1 qui assure que le meilleur bras peut être identifié à chaque pas de temps.

Hypothèse 1. *Pour tout* $k \in [K] - \{k^*\}$ *et tout* $[\tau] \in \mathbb{T}(\tau)$ *avec* $\tau \geq \tau_{\min}$*, nous avons :*

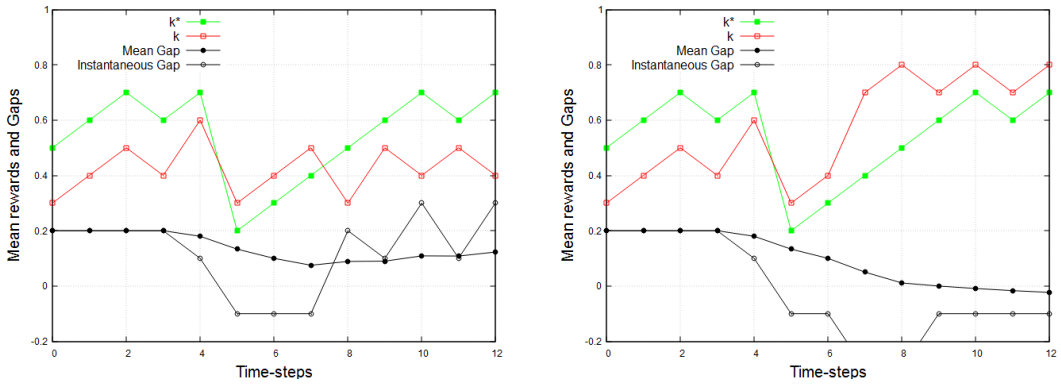
$$\Delta_k^*([\tau]) = \frac{1}{\tau} \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_i|-1} \frac{\Delta_{k^*,k}(j)}{|S_i|} > 0\,. \tag{2.4}$$

Cette hypothèse est assez faible et permet de tolérer une grande quantité de bruit lorsque τ est grand. Étant donné que le bras optimal doit se distinguer des autres, les écarts instantanés sont d'avantage contraints en début de jeu. Cette contrainte est similaire à celle utilisée par Seldin et Slivkins (voir section 1.2.3) pour atteindre un regret logarithmique sur des récompenses *modérément contaminées*. Un autre parallèle peut-être fait entre le régime *adverse avec écart* (voir section 1.2.3), $\tau_{\min}$ représentant le temps accordé au bras optimal pour générer assez de récompenses pour se distinguer des bras sous-optimaux.

La figure 2.2(a) illustre l'hypothèse 1. Sur cet exemple, la moyenne du bras optimal k^* est plus faible que celle du second bras sur les pas de temps $t \in \{5, 6, 7\}$. Même si l'écart instantané est négatif durant ces pas de temps, l'écart moyen $\Delta^*([\tau])$ reste positif. Le paramètre $\tau_{\min}$ protège l'algorithme d'un bruit pouvant être présent à l'initialisation de l'algorithme. Afin de faciliter la lecture des résultats analytiques, nous supposons que $\tau_{\min} = \log \frac{K}{\delta}$.

L'hypothèse 1 peut-être considérée comme un test de cohérence, assurant que chercher à trouver un unique meilleur bras est sensé. Le cas où l'hypothèse 1 n'est pas vérifiée (voir figure 2.2(b)) est considéré comme un problème avec changement de meilleur bras et est étudié en section 2.4. Le chapitre 6 présente une formalisation du problème de sélection d'experts apprenants comme un problème de bandit adverse avec budget où l'écart moyen cumulé peut-être négatif à certains pas de temps ainsi qu'une modification de SUCCESSIVE ELIMINATION utilisable dans ce cas là .

Algorithme



(a) L'hypothèse 1 est vérifiée. L'écart moyen reste positif

(b) L'hypothèse 1 n'est pas vérifiée. Cette séquence implique un changement de meilleur bras.

FIGURE 2.2 – Séquences de récompenses illustrant la notion d'écart moyen

Analyse

Nous montrons tout d'abord les garanties en complexité d'échantillonnage de l'algorithme puis nous les transposons en garanties sur le regret cumulé. Nous rappelons la définition de la complexité d'échantillonnage.

La complexité d'échantillonnage est le nombre d'observations nécessaires à un algorithme pour trouver un bras ϵ-optimal avec une forte probabilité.

Nous considérons un bras k comme ϵ-optimal, si il existe un pas de temps pour

Algorithme 10 : Éliminations Successives avec Round-Robin Randomisé (SER3)

Données : $\delta \in (0, 0.5]$, $\epsilon \in [0, 1)$
Résultat : un ϵ-approximation du meilleur bras
$S_1 = [K]$, $\forall k$, $\hat{\mu}_k(0) = 0$, $t = 1$, $\tau = 1$;
tant que $|S_\tau| > 1$ **faire**
  Mélanger S_τ;
  pour chaque $k \in S_t$ **faire**
    Jouer k;
    $\hat{\mu}_k(\tau) = \frac{\tau-1}{\tau}\hat{\mu}_k(\tau-1) + \frac{y_{k_t}(t)}{\tau}$;
    $t = t + 1$;
  si $\tau \geq \tau_{\min}$ **alors**
    $k_{\max} = \arg\max_{m \in S} \hat{\mu}_k(t)$;
    Retirer de $S_{\tau+1}$ tout k tel que :

$$\hat{\mu}_{\max}(\tau) - \hat{\mu}_k(\tau) + \epsilon \geq 2\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}$$

  $\tau = \tau + 1$;

lequel $\Delta_k^*([\tau]) \leq \epsilon$. Les résultats de cette section sont donnés pour $\epsilon = 0$, ainsi seul le bras optimal est accepté comme solution.

Théorème 11. *Pour $K \geq 2$, $\delta \in (0, 0.5]$, et $\tau_{\min} = \log\frac{K}{\delta}$, la complexité d'échantillonnage de* SER3 *est bornée supérieurement par :*

$$O\left(\frac{K}{\Delta^2}\log(\frac{K}{\delta\Delta})\right)$$

où $\Delta = \min_{[\tau],k} \frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{t=t_i}^{t_i+|S_i|-1}\frac{\Delta_{k^*,k}(t)}{|S_i|}$.

La preuve du Théorème 11 est détaillée en sous-section 2.6.

Corollaire 2. *Pour $K \geq 2$, $\delta = 1/T$, et $\tau_{\min} = \log(KT)$, l'espérance du pseudo-regret cumulé de* SER3 *est bornée supérieurement par :*

$$\min\left(O\left(\frac{K-1}{\Delta}\log(\frac{KT}{\Delta})\right), O\left(\sqrt{TK\log\frac{T}{K}}\right)\right)$$

La preuve du Corollaire 2 est détaillée en sous-section 2.6.

L'utilisation d'un round-robin randomisé permet à notre algorithme SER3 d'obtenir les mêmes garanties que lorsque l'exploration est contrôlée par un round-robin déterministe sur des bras ayant des récompenses stationnaires. En effet pour tout t et tout $[\tau]$ nous avons :

$$\frac{1}{\tau}\sum_{i=1}^{\tau}\sum_{t=t_i}^{t_i+|S_i|-1}\frac{\Delta_{k^*,k}(t)}{|S_i|} = \Delta_{k^*,k}(t) = \Delta_{k^*,k}(t+1)\,. \tag{2.5}$$

L'algorithme non randomisé SUCCESSIVE ELIMINATION ne possède pas de garanties lorsque les récompenses ne sont pas stationnaires. L'adversaire peut 'utiliser le déterminisme du round-robin pour construire une séquence de récompenses menant à l'élimination du bras optimal tout en respectant l'hypothèse 1.

Ce résultat peut sembler contredire la borne inférieure en $\Omega(\sqrt{T})$ [23] sur le problème des bandits non-stationnaires. Ce coût en racine est du à l'exploration constante nécessaire à la détection des changements du meilleur bras. Les garanties logarithmiques sont obtenues en se comparant à une politique optimale jouant un seul bras et en contraignant l'écart moyen à être positif.

2.3.2 Simulations numériques

Nous comparons dans cette sous-section les performances de SER3 à celles d'algorithmes de l'état de l'art sur deux problèmes synthétiques. Le nombre de bras est $K = 20$ et l'horizon est $T = 10^7$. L'écart instantané $\Delta = 0.05$ entre le bras optimal et les autres est constant, i.e. si la récompense moyenne d'un bras sous-optimal à l'instant t est $\mu(t)$ alors la récompense moyenne du bras optimal est $\mu^*(t) = \mu(t) + \Delta$. Durant ces expériences, la probabilité d'échec de SUCCESSIVE ELIMINATION (SE) et de SER3 est fixée à $\delta = 0.05$, $\epsilon = 0.7$ et l'exploration d'EXP3 est $\gamma = 0.05$. Les courbes sont moyennées sur 50 exécutions des algorithmes.

Problème 1 : Moyennes sinusoïdales. L'index du bras optimal k^* est tiré au début du jeu et ne change pas durant les T itérations. La moyenne des bras sous-optimaux est $\mu(t) = \cos(2\pi t/K)/5 + 0.5$.

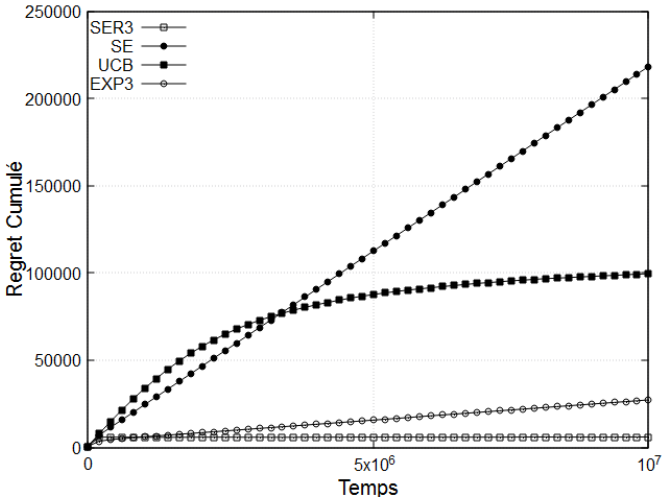


FIGURE 2.3 – Les regrets cumulés de SER3, SE, UCB et EXP3 sur le Problème 1.

Ce problème est construit de manière à avoir une périodicité de K et des variations de moyenne supérieures à Δ. Ainsi, un algorithme choisissant le bras à jouer suivant un round-robin n'observera que les récompenses tirées depuis des moyennes basses si les itérations où k^* est joué se situent dans un creux de la sinusoïde et réciproquement, des récompenses tirées depuis des moyennes hautes si les itérations où les bras sous optimaux sont joués se situent dans la partie haute de la sinusoïde. De cette manière, SE devrait éliminer le bras optimal durant une grande partie des exécutions de l'algorithme.

L'algorithme SE se comporte comme prévu, la périodicité des récompenses moyennes ainsi du round-robin lui cachant l'optimalité de k^* et entrainant son élimination. Bien que non adapté à la non-stationnarité, la politique déterministe d'UCB converge vers le bras optimal après avoir souffert d'un regret élevé. Les estimateurs non biaisés d'EXP3 lui permettent de converger rapidement vers le meilleur bras. L'algorithme souffre cependant du regret associé à son exploration constante jusqu'à la fin du jeu. Finalement, SER3 obtient un regret cumulé faible, éliminant avec succès les bras sous-optimaux au début du jeu. Ceci illustre l'apport, à coût nul, de la randomisation des tours de round-robin dans les algorithmes d'élimination.

Problème 2 : Moyennes décroissantes avec écart positif. Le bras optimal k^* ne change pas durant le jeu. La moyenne des récompenses des bras sous-optimaux est $\mu(t) = 0.95 - \min(0.45, 10^{-7}t)$.

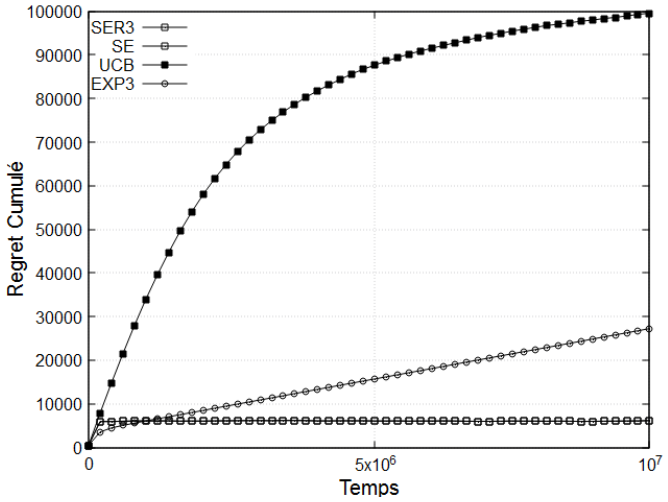


FIGURE 2.4 – Les regrets cumulés de SER3, SE, UCB et EXP3 sur le Problème 2.

Dans ce problème, les récompenses moyennes baissent lentement au fur et à mesure des tours de jeux. Un bras sous-optimal longuement joué au début du jeu aura ainsi une récompense moyenne empirique supérieure au bras optimal si celui-ci est joué tardivement.

Similairement au problème 1, SER3 obtient un faible regret cumulé en éliminant rapidement les bras sous-optimaux. La non-stationnarité n'est pas suffisamment forte pour dérouter SE et lui permet de se comporter de la même manière que SER3 sur ce problème. UCB est une fois de plus mis en échec par la non-stationnarité des récompenses et commence à converger tardivement vers le bras optimal. Le regret cumulé d'EXP3 est une fois de plus faible au début du jeu, mais continue d'augmenter linéairement au cours du temps.

2.4 Politique optimale à bras multiples

Similairement à SER3, l'algorithme adverse EXP3 [31] est construit de manière à trouver le bras ayant la plus haute récompense cumulée sur la totalité du jeu. L'exemple présenté en figure 2.5 montre la limite de cette approche. La moyenne des récompenses sur la totalité du jeu est identique pour les deux bras, $\left(\frac{\mu'_1+\mu'_2}{2}\right)$. Jouer le bras k_1 durant la première moitié du jeu puis ensuite le bras k_2 est cependant une meilleure stratégie.

La **politique optimale** est une séquence arbitraire de bras

$$\{(k_1^*, 1), ..., (k_N^*, T_N)\}, \tag{2.6}$$

avec $k_n^* \neq k_{n+1}^*$ et $\Delta_{k_n^*,k}(t) > 0$ pour tout $k \in [K] - \{k_n^*\}$ et tout $t \in [T_n, T_{n+1})$. La politique optimale commence à jouer le bras k_n^* à T_n.

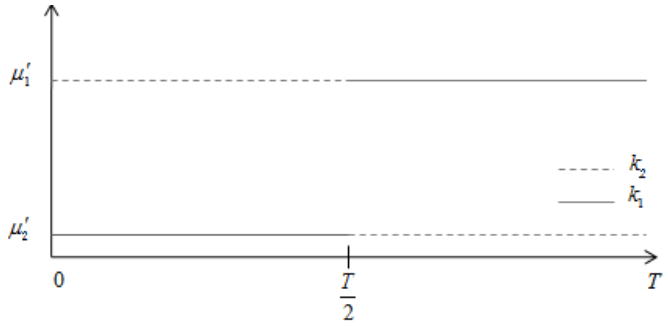


FIGURE 2.5 – Un exemple de problème où les bras changent de moyenne au cours du temps.

Aller plus loin que la stationnarité par partie

Les travaux précédents incluant des distributions stochastiques non-stationnaires, notamment D-UCB [27] et UCB à fenêtre glissante (SW-UCB) [23], les supposaient stationnaires par parties (voir 1.2.1). Les cas où les changements de distributions entrainent un changement de meilleur bras ne sont pas différenciés des cas où le meilleur bras ne change pas. Les garanties théoriques de D-UCB et SW-UCB mettent en avant cette dépendance avec un regret en $O(\sqrt{MT \log T})$, où M est le nombre de changements de distributions (voir figure 2.6). Dans le pire des cas, si la distribution change à chaque itération ($M = T$), le regret devient linéaire en T, même si le meilleur bras ne change pas.

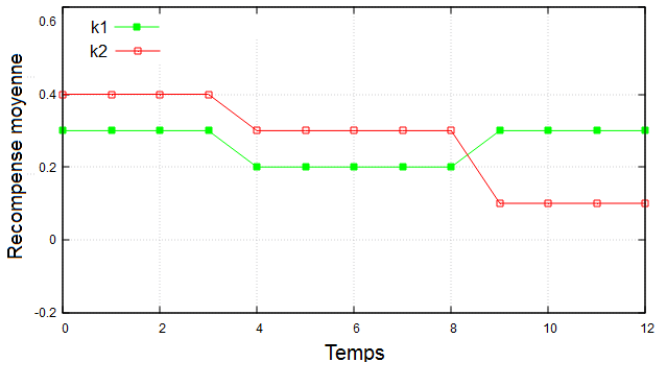


FIGURE 2.6 – Ce jeu comporte trois changements de moyennes, mais la politique optimale ne changerait de bras qu'une seule fois. Ici, $M = 3$ et $N = 2$.

2.4.1 Éliminations Successives Randomisées avec Réinitialisations

La complexité d'échantillonnage pour le problème d'identification du meilleur bras avec changements

Nous proposons une nouvelle définition de la complexité d'échantillonnage prenant en compte les changements de meilleurs bras au cours du jeu.

Reprise de l'échantillonnage et le coup de l'erreur. Lorsque le meilleur bras ne change pas, le processus d'identification est divisé en deux étapes. Dans la première, l'algorithme collecte des observations puis lorsqu'il est capable d'effectuer une prédiction avec le niveau de certitude requis, il retourne le bras prédit. Étant donné que le meilleur bras ne change pas, il n'y a plus aucune raison de récolter de nouvelles observations. Dans cette sous-section, le meilleur bras peut changer au cours du temps et son identification devient un processus plus dynamique. Au pas de temps t, l'algorithme peut ainsi se trouver dans deux états. Dans le premier état, $s(t) = 1$, l'algorithme échantillonne un bras afin de collecter d'avantage d'information. Dans le second état $s(t) = 0$, l'algorithme effectue une prédiction et peut n'obtenir aucun retour (le bras k_t n'est pas joué, mais uniquement prédit). Après avoir effectué une prédiction, l'algorithme peut reprendre le processus d'échantillonnage pour vérifier si le meilleur bras a changé. Deux coûts sont inclus dans notre définition, le coût d'échantillonnage et le coût de l'erreur de prédiction. Après un changement, l'algorithme souffrira de ce coût tant qu'il n'aura pas identifié le nouveau meilleur bras.

Définition 1. *La complexité d'échantillonnage d'un algorithme A effectuant une tâche d'identification du meilleur bras avec changements pour la segmentation $\{T_n\}_{n=1..N}$ de $[1:T]$, avec $T_1 = 1 < T_2 < \cdots < T_N < T$, est :*

$$\sum_{n=1}^{N} \sum_{t=T_n}^{T_{n+1}-1} \left(s(t) + (1 - s(t))[\![k_t \neq k_n^*]\!]\right), \tag{2.7}$$

où k_t est le bras identifié comme optimal par A au temps t, k_n^* est le bras optimal sur le segment n, $T_{n+1} = T + 1$ et $s(t)$ est une variable binaire prenant comme valeur 1 si le pas de temps t a été utilisé dans le processus d'échantillonnage, 0 si le pas de temps t a été utilisé pour effectuer une prédiction.

Algorithme

L'algorithme SER3 peut être facilement adapté au cas où le meilleur bras change au cours du temps. Pour cela nous introduisons la possibilité que l'algorithme se réinitialise avec une probabilité $\varphi \in [0, 1]$, lui permettant de converger vers un nouveau bras si celui-ci a changé. Nous appelons cet algorithme ÉLIMINATIONS SUCCESSIVES AVEC ROUND-ROBIN RANDOMISÉS ET RÉINITIALISATIONS (SER4).

Algorithme 11 : Éliminations Successives Randomisé avec Réinitialisations (SER4)

Données : $\delta \in (0, 0.5]$, $\epsilon \in (0, 1], \varphi \in [0, 1]$
$S_1 = [K]$, $\forall k$, $\hat{\mu}_k(0) = 0$, $t = 1$, $\tau = 1$;
tant que $t \leq T$ **faire**
  Mélanger $|S_\tau|$;
  pour chaque $k \in S_t$ **faire**
    Jouer k; `// Si le joueur cherche à minimiser la complexité d'échantillonnage, alors il ne joue pas de bras lorsque` $|S_\tau| = 1$
    $\hat{\mu}_k(\tau) = \frac{\tau-1}{\tau}\hat{\mu}_k(\tau-1) + \frac{y_{k_t}(t)}{\tau}$;
    $t = t + 1$;
    si *random()*$\leq \varphi$ **alors**
      $S_t = [K]$;
      $\forall k$, $\hat{\mu}_k(t) = 0$;
      $\tau = 0$;
      Sortir des deux boucles courantes ; `// On retourne à la phase de mélange`
  $k_{\max} = \arg\max_{m \in S} \hat{\mu}_k(t)$;
  si $\tau \geq \tau_{\min}$ **alors**
    $k_{\max} = \arg\max_{m \in S} \hat{\mu}_k(t)$;
    Retirer de $S_{\tau+1}$ tout k tel que :

$$\hat{\mu}_{\max}(\tau) - \hat{\mu}_k(\tau) + \epsilon \geq 2\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}$$

  $\tau = \tau + 1$;

Analyse

Théorème 12. *Pour $K \geq 2$, $\delta = 1/T$, $\tau_{\min} = \log\frac{K}{\delta}$ et $\varphi \in (0, 1]$, l'espérance de la complexité d'échantillonnage vis-à-vis de la randomisation des réinitialisations de* SER4 *est bornée supérieurement par :*

$$O\left(\frac{\varphi K}{\delta\Delta^2}\log(\frac{K}{\delta\Delta}) + \frac{N}{\varphi}\right)$$

avec une probabilité d'au moins $1 - \delta$.

La preuve du Théorème 12 est détaillée en sous-section **??**.

Nous optimisons φ pour minimiser la complexité d'échantillonnage.

Corollaire 3. *Pour $K \geq 2$, $\delta = 1/T$, $\tau_{\min} = \log\frac{K}{\delta}$, $\Delta \geq \frac{1}{KT}$ et $\varphi = \sqrt{\frac{N}{TK\log(TK)}}$,*

l'espérance de la complexité d'échantillonnage de SER4 *vis-à-vis de la randomisation des réinitialisations est bornée supérieurement par :*

$$O\left(\frac{\sqrt{NTK\log(TK)}}{\Delta^2}\right).$$

Ce résultat peut aussi être transposé en borne sur l'espérance du regret cumulé. Dans ce cas, l'algorithme continue à jouer le bras considéré comme optimal après avoir éliminé tous les autres jusqu'à ce qu'une réinitialisation se produise.

Corollaire 4. (Borne dépendante de la distribution) *Pour* $K \geq 2$, $\delta = 1/T$, $\tau_{\min} = \log(KT)$, $\Delta \geq \frac{1}{KT}$ *et* $\varphi = \sqrt{\frac{N}{T}}$, *l'espérance du pseudo-regret cumulé de* SER4 *vis-à-vis de la randomisation des réinitialisations est bornée supérieurement par :*

$$O\left(\frac{\sqrt{NTK\log(KT)}}{\Delta}\right).$$

(Borne indépendante de la distribution) *Pour* $K \geq 2$, $\delta = 1/T$, $\tau_{\min} = \log(KT)$, $\Delta \geq \frac{1}{KT}$ *et* $\varphi = \varphi = \frac{\sqrt{N}}{T^{2/3}}$, *l'espérance du pseudo-regret cumulé de* SER4 *vis-à-vis de la randomisation des réinitialisations est bornée supérieurement par :*

$$\mathbb{E}[\bar{R}(T)] = O\left(T^{2/3}\sqrt{NK\log\frac{T}{K}}\right). \tag{2.8}$$

Les preuves des Corollaires 3 et 2.6 sont détaillées en sous-section **??**.

2.4.2 EXP3 avec Réinitialisations

Dans la section précédente, les changements de bras étaient combattus grâce aux réinitialisations randomisées. Bien que cette approche puisse sembler brutale, elle a permis d'introduire la notion de complexité d'échantillonnage sur des problèmes où le meilleur bras peut changer et possède des garanties sur le regret cumulé rivalisant avec l'état de l'art. Nous présentons maintenant l'algorithme EXP3.R, basé lui aussi sur des réinitialisations, mais maintenant contrôlées par un détecteur de ruptures.

Algorithme

EXP3 (voir section 1.2.2) minimise le regret face au meilleur bras en utilisant un estimateur non biaisé des récompenses cumulées au temps t pour calculer les probabilités de choisir chaque bras. Même si cette stratégie peut être vu comme

optimale dans un cas adverse, dans beaucoup d'applications pratiques, la non-stationnarité à l'intérieur d'une période de temps est faible et les changements de bras sont uniquement remarquables sur le long terme. Si un bras est performant sur une période de grande taille, mais devient très mauvais ensuite, l'algorithme EXP3 peut nécessiter un nombre d'itérations égal à la taille de la première période avant de changer le bras majoritairement joué. Couplé avec un détecteur de changement, l'algorithme EXP3 a plusieurs avantages. Premièrement, dans un environnement non-stationnaire, une exploration constante est nécessaire pour détecter les changements. Cette exploration est donnée naturellement par l'algorithme. Deuxièmement, le nombre de changements de moyennes est supérieur au nombre de changements de meilleurs bras ($M \geq N$) (voir Figure 2.6). Cela signifie que le nombre de réinitialisations requises par EXP3 est plus petit que celui nécessaire à un algorithme stochastique comme UCB. Troisièmement, EXP3 est robuste face aux non détections ou aux non-stationnarités locales.

Le détecteur de rupture (voir Algorithme 12) utilise des intervalles de confiance pour estimer les espérances de récompenses sur la période de temps précédente. La distribution de probabilité des bras dans EXP3 est un mélange entre une distribution de Gibbs et une distribution uniforme. Nous appelons γ-observation, une observation sélectionnée via la distribution uniforme. Les paramètres γ, H et δ définissent un nombre minimal de γ-observations nécessaires à l'utilisation d'un détecteur d'une précision ϵ avec une probabilité d'au moins $1-\delta$. Ces paramètres seront fixés plus tard dans l'analyse (voir Corollaire 5) et la validité du test est prouvée dans le Lemme 1. Nous notons $\hat{\mu}^k(I)$ la moyenne empirique des récompense obtenues par le bras k sur l'intervalle I en utilisant uniquement les γ-observations et $\Gamma_{\min}$(I) le plus petit nombre de γ-observations parmi tous les bras sur l'intervalle I. Le détecteur est appelé uniquement quand $\Gamma_{\min}(I) \geq \frac{\gamma H}{K}$.

Celui-ci lève une détection quand le bras $k_{\max}$, qui est celui possédant la plus haute moyenne empirique $\hat{\mu}^k(I-1)$ sur l'intervalle $I-1$ est éliminée par une autre sur l'intervalle courant.

Algorithme 12 : DetectionDeDerive(I)

Données : Intervalle courant $I > 1$

$k_{\max} = \arg\max_k \hat{\mu}^k(I-1)$;

$\epsilon = \sqrt{\frac{K \log(\frac{1}{\delta})}{2\gamma H}}$;

retourner $\left[\!\left[\exists k, \quad \hat{\mu}^k(I) - \hat{\mu}^{k_{\max}}(I) \geq 2\epsilon\right]\!\right]$;

L'algorithme EXP3.R est obtenu en combinant EXP3 et le détecteur de dérive. Dans un premier temps, une instance d' EXP3 est initialisée et utilisée pour sélectionner les bras. Des γ-observations sont alors collectées jusqu'à ce que $\Gamma_{\min}(I) \geq \frac{\gamma H}{K}$. Lorsque ce compte est atteint, le test de détection est exécuté. Si, dans l'intervalle correspondant, la différence entre la moyenne empirique d'un bras et celle du meilleur bras est supérieure à 2ϵ alors une détection est levée. Dans ce cas, les poids de l'algorithme EXP3 sont réinitialisés. Commence alors un nouvel

intervalle de collecte en préparation du prochain test. Ces étapes sont répétées jusqu'à la fin du jeu (voir Algorithme 13).

Algorithme 13 : EXP3 avec Réinitialisation

Données : Réels δ, γ et entier H
$I = 1$;
pour ***chaque*** $t = 1, ..., T$ **faire**
  Appeler EXP3 sur le pas de temps t;
  si $\Gamma_{min}(I) \geq \frac{\gamma H}{K}$ **alors**
    si $I > 1$ *et* $DetectionDeDerive(I)$ **alors**
      Réinitialiser EXP3;
    $I = I + 1$;

Avec une précision ϵ, seules les différences supérieures à 4ϵ sont détectées avec une forte probabilité.

Analyse

Dans cette section, nous analysons le détecteur de rupture et bornons l'espérance du regret de l'algorithme EXP3.R.

Hypothèse 2. *Durant chacune des M périodes stationnaires, la différence entre la récompense moyenne du bras optimal et celle de n'importe quel autre bras est au moins de 4ϵ avec*

$$\epsilon = \sqrt{\frac{K \log(\frac{1}{\delta})}{4\gamma H}}\,. \tag{2.9}$$

Le lemme 1 garantit que quand l'hypothèse 1 est valide et que l'intervalle I est inclus dans l'intervalle S alors, avec une forte probabilité, une détection sera levée si et seulement si le bras optimal k_S^* élimine un bras sous-optimal.

Lemme 1. *Quand l'hypothèse 1 est valide et que $I \subseteq S$ alors, avec une probabilité d'au moins $1 - 2\delta$, pour tout $k \neq k_S^*$:*

$$\hat{\mu}^{k_S^*}(I) - \hat{\mu}^k(I) \geq 2\sqrt{\frac{K \log(\frac{1}{\delta})}{2\gamma H}} \Leftrightarrow \mu^{k_S^*}(I) \geq \mu^k(I)\,. \tag{2.10}$$

La preuve du Lemme 1 est détaillée en sous-section 2.6.

Le Théorème 13 borne l'espérance du regret cumulé d'EXP3.R.

Théorème 13. *Pour tout* $K > 0$, $0 < \gamma \leq 1$, $0 \leq \delta < \frac{1}{2}$, $H \geq K$ *et* $N \geq 1$, *quand l'hypothèse 1 est valide, l'espérance du regret d'*EXP3.R *vérifie*

$$G^* - \boldsymbol{E}[G_{EXP3.R}] \leq (e-1)\gamma T + \frac{\left(N - 1 + \frac{K\delta T}{H} + K\delta\right) K \log(K)}{\gamma} + (N-1)HK\left(\frac{1}{1-2\delta} + 1\right). \quad (2.11)$$

La preuve du Théorème 13 est détaillée en sous-section 2.6.

Dans le Corollaire 5 nous optimisons les paramètres de la borne obtenue dans le théorème 13.

Corollaire 5. *Pour tout* $K \geq 1$, $T \geq 10$, $N \geq 1$ *et* $C \geq 1$, *quand l'hypothèse 1 est vérifiée, l'espérance du regret d'un* EXP3.R *exécuté avec comme paramètres*

$$\delta = \sqrt{\frac{\log T}{KT}}, \gamma = \sqrt{\frac{K \log K \log T}{T}} \text{ et } H = C\sqrt{T \log T} \quad (2.12)$$

est

$$G^* - \boldsymbol{E}[G_{EXP3.R}] \leq (e-1)\sqrt{TK \log K \log T} + N\sqrt{TK \log K} + (C+1)K\sqrt{T \log K} + 3NCK\sqrt{T \log T}. \quad (2.13)$$

Suivant C, la précision ϵ est :

$$\epsilon = \sqrt{\frac{1}{2C}}\sqrt{\frac{\log\sqrt{\frac{KT}{\log T}}}{\log T}}\sqrt{\frac{K}{\log K}} \leq \sqrt{\frac{1}{2C}}\sqrt{\frac{K}{\log K}}. \quad (2.14)$$

La preuve du Corollaire 5 est détaillée en sous-section 2.6.

2.4.3 Simulations numériques

Nous comparons dans cette sous-section les performances de SER4 et EXP3.R à celles d'algorithmes de l'état de l'art sur deux problèmes synthétiques. Le nombre de bras est $K = 20$ et l'horizon est $T = 10^7$. L'écart instantané Δ entre le bras optimal et les autres est constant, i.e. si la récompense moyenne d'un bras sous-optimal à l'instant t est $\mu(t)$ alors la récompense moyenne du bras optimal est $\mu^*(t) = \mu(t) + \Delta$. Durant ces expériences, la probabilité d'échec de SER4 est

fixée à $\delta = 0.05$, $\epsilon = 0.7$ et sa probabilité de réinitialisation est $\varphi = 5 \times 10^{-6}$. L'exploration d'EXP3.R et d'EXP3.S est $\gamma = 0.01$. L'historique considéré par EXP3.R est $H = 4 \cdot 10^5$ et le paramètre de régularisation d'EXP3.S est $\alpha = 10^{-6}$. La taille de la fenêtre de SW-UCB est 10^5. Les courbes sont moyennées sur 50 exécutions des algorithmes.

Problème 3 : Moyennes décroissantes avec changement de bras. À chaque tour, le bras optimal change avec une probabilité de 10^{-6}. En espérance, il y a 10 changements de bras par exécution. La récompense moyenne des bras sous-optimaux est $\mu(t) = 0.95 - \min(0.45, 10^{-7}(t[\text{mod } 10^6])$ et $\Delta = 0.05$.

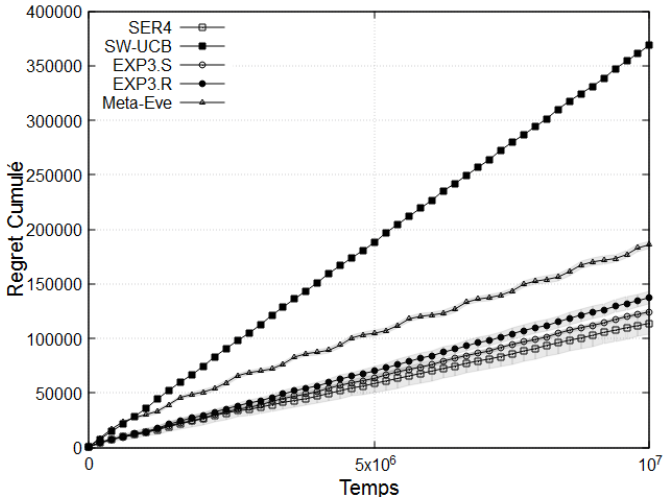


FIGURE 2.7 – Les regrets cumulés de SER4, SW-UCB, EXP3.S, EXP3.R et META-EVE sur le Problème 3.

Malgré l'ajout d'une fenêtre à UCB, SW-UCB souffre de la non-stationnarité permanente et des changements de bras, obtenant un regret cumulé final élevé. Du fait de la dérive des moyennes, META-EVE se réinitialise très fréquemment. De manière surprenante, ces multiples réinitialisations permettent à UCB de lutter à la fois contre les changements de bras et contre le biais dans les moyennes du aux observations passés et au temps écoulé. Le Problème 3 vérifie les hypothèses de SER.4, EXP3.R et EXP3.S, expliquant leurs bonnes performances face à SW-UCB et META-EVE. Les regrets de EXP3.S et EXP3.R sont faibles, mais EXP3.R est pénalisé par la grande taille d'historique nécessaire à la détection d'un changement de bras avec un écart de Δ. Bien que SER4 obtienne le regret cumulé le plus faible, la randomisation de ses réinitialisations implique une plus grande variance. En effet, les réinitialisations peuvent se produire rapidement après un changement ou au contraire être tardives. Á l'inverse, l'utilisation d'une fenêtre ou d'un paramètre de régularisation apporte aux algorithmes un comportement plus cohérent.

Problème 4 : Changement de bras avec certaines moyennes constantes. Á chaque tour, le bras optimal change avec une probabilité de 10^{-6}.

En espérance, il y a 10 changements de bras par exécution. La moyenne du bras optimal est $\mu^*(t) = 0.8$ lorsque $k^*[\text{mod } 3] \neq 2$. Ce problème comporte des bras dits *constants* dont les moyennes ne changent pas. Un bras k est *constants* lorsque $k[\text{mod } 3] = 2$ et a pour récompense moyenne $\mu_k(t) = 0.7$. La propriété de *constance* prévaut sur celle d'optimalité. Pour tous les autres bras k tel que $k \neq k^*$ et $k[\text{mod } 3] \neq 2$, la récompense moyenne est $\mu(t) = 0.2$.

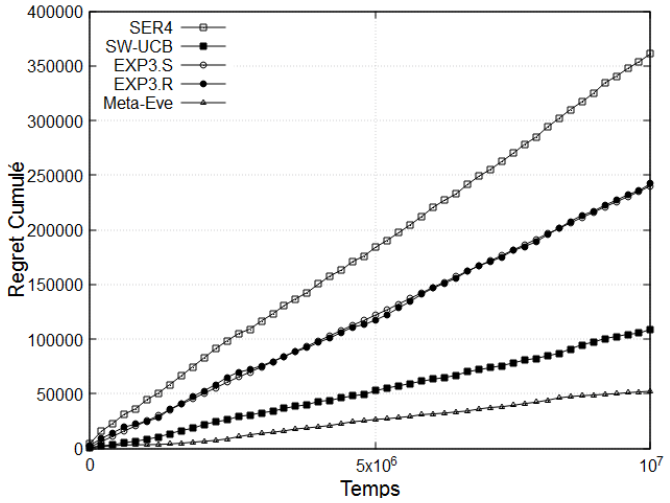


FIGURE 2.8 – Les regrets cumulés de SER4, SW-UCB, EXP3.S, EXP3.R et META-EVE sur le Problème 4.

Le problème 4 illustre le principal inconvénient de SER4, les réinitialisations ne dépendent pas des données et obligent l'algorithme à recommencer son optimisation du début. Même si SER4 converge plus vite vers le bras optimal, le coût d'exploration des bras sous-optimaux conjugué à la mécanique d'élimination lui inflige un regret plus élevé que les autres algorithmes. En effet, les autres algorithmes maintiennent leurs estimateurs tout au long du jeu et si la récompense du bras optimal chute brusquement, la moyenne empirique commencera à baisser jusqu'à ce qu'elle devienne inférieure à celle des bras constants. Ces bras constants permettent aux autres algorithmes de capitaliser des récompenses en attendant que leurs mécaniques d'oubli leur permettent de converger vers le nouveau bras. Les algorithmes supposant l'existence de stationnarité par partie obtiennent les meilleurs résultats sur ce problème, démontrant ainsi leur efficacité lorsque leurs hypothèses sont respectées.

Le problème 3 illustrait l'un des pires cas : les écarts de tous les bras sous-optimaux sont identiques. Dans ce cas, maintenir un historique est un handicap lors d'un changement de meilleur bras et seule compte la vitesse de convergence de l'algorithme.

2.5 Récapitulatif

Dans ce chapitre, nous avons généralisé les bandits adverses et stationnaires par parties en générant les séquences de récompenses à partir de distributions pouvant changer à chaque tour de jeu. En utilisant une hypothèse assurant l'unicité du bras optimal sur toutes les fenêtres temporelles $\{0, ..., t\}$ avec $0 \leq t \leq T$, l'algorithme SER3 peut effectuer la tâche de sélection du meilleur bras avec une complexité d'échantillonnage en $O\left(K\Delta^{-2}\log(K\Delta^{-1}\delta^{-1})\right)$. Nous avons ensuite proposé deux algorithmes basés sous les réinitialisations de sous-routines de bandits (SER4, utilisant SER3 et EXP3.R, utilisant EXP3) pour permettre de poursuivre des politiques optimales changeant de meilleur bras au cours du jeu. Nous adaptons la définition de la complexité d'échantillonnage au cas où le meilleur bras peut changer N fois au cours du jeu et analysons SER4 dans ce cadre. L'algorithme SER4 effectue une réinitialisation à chaque tour selon une probabilité fixée en paramètre. Bien que le nombre de réinitialisations soit élevé (de l'ordre de $\sqrt{T\log(\delta\Delta)^2}$) sa complexité d'échantillonnage logarithmique lui permet d'obtenir une complexité d'échantillonnage finale dont l'espérance est en $O\left(\Delta^{-2}\sqrt{NTK\log(TK)}\right)$. Le regret en $O\left(\sqrt{KT\log K}\right)$ d'EXP3, plus élevé que celui de SER3, est compensé par le détecteur de rupture présent dans EXP3.R qui permet d'effectuer les réinitialisations uniquement quand le meilleur bras change. L'inclusion de ce détecteur de rupture nécessite cependant un paramètre d'exploration plus élevé dont la valeur dépend de la précision requise par le test. Les analyses de SER4 et EXP3.R montrent des bornes supérieures sur l'espérance du regret cumulé de l'ordre de $O\left(CNK\sqrt{T\log T}\right)$ (où C est une fonction décroissante suivant Δ) et $O\left(\Delta^{-1}\sqrt{NTK\log(TK)}\right)$. L'avantage de nos méthodes par rapport à d'autres algorithmes supposant l'existence de stationnarité par parties (comme SW-UCB ou DISCOUNTED-UCB) est d'obtenir une dépendance en N (le nombre de changements de meilleurs bras) et non en M (le nombre de changements de distributions), qui peut-être égal à T dans notre formalisme.

2.6 Preuves

2.6.1 Preuve du Théorème 11

Démonstration. La preuve est composée de trois étapes principales :

1 - Expliciter les conditions menant à l'élimination d'un bras.

2 - Montrer que le bras optimal ne sera pas éliminé.

3 - Montrer que les bras sous-optimaux seront éliminés après avoir été joués au maximum τ^* fois, permettant ainsi d'obtenir une borne supérieure sur la complexité d'échantillonnage.

1 - La condition d'élimination.

En utilisant l'inégalité d'Hoeffding, pour toutes séquences de tours de round-robin

de taille τ et tout bras k, nous avons :

$$P\left(|\hat{\mu}_k - \mathbb{E}[\hat{\mu}_k]| \geq \epsilon_\tau\right) \leq 2\exp\left(-2\epsilon_\tau^2 \tau\right) .$$

où $\mathbb{E}$ désigne l'espérance vis-à-vis de la randomisation des tours de round-robin et des récompenses. En définissant

$$\epsilon_t = \sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}, \text{nous avons :}$$

$$P\left(|\hat{\mu}_k - \mathbb{E}[\hat{\mu}_k(\tau)]| \geq \epsilon_t\right) \leq 2\exp\left(-2\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}^2 \tau^2\right) = \frac{\delta}{2K\tau^2} . \quad (2.15)$$

Nous rappelons que $\mathbb{T}(\tau)$ est l'ensemble contenant toutes les séquences de tours de round-robin de taille τ pouvant être réalisées par l'algorithme. Chaque bras k est joué une fois durant chaque tour de round-robin.

En répétant l'inégalité 2.15 sur chaque élément de $\mathbb{T}(\tau)$, en utilisant l'inégalité de Boole et le fait que t $\sum_{\tau=1}^{\infty} 1/\tau^2 = \pi^2/6$, l'inégalité suivante est vraie simultanément pour tout τ avec une probabilité d'au moins $1 - \frac{\delta\pi^2}{12K}$:

$$\hat{\mu}_k(\tau) - \epsilon_\tau \leq \mathbb{E}[\hat{\mu}_k] \leq \hat{\mu}_k(\tau) + \epsilon_\tau . \quad (2.16)$$

Nous rappelons que S_i est l'ensemble contenant les bras non éliminés par l'algorithme au début du i^{eme} tour de round-robin. Par construction, un bras k' reste dans l'ensemble des bras aussi longtemps que pour chaque $k \in S_\tau - \{k'\}$ nous avons :

$$\hat{\mu}_k(\tau) - \epsilon_\tau < \hat{\mu}_{k'}(\tau) + \epsilon_\tau \text{ et } \tau \geq \tau_{\min} \quad (2.17)$$

En combinant (2.16) et l'inégalité de gauche de (2.17), nous obtenons l'évènement Ω, qui se produit avec une forte probabilité :

$$\mathbb{E}[\hat{\mu}_k(\tau)] - 2\epsilon_\tau < \mathbb{E}[\hat{\mu}_{k'}(\tau)] + 2\epsilon_\tau . \quad (2.18)$$

Le pas de temps où le τ^{eme} tour de round-robin débute est noté t_τ ($t_\tau = 1 + \sum_{i=1}^{\tau-1} |S_i|$). La récompense moyenne empirique $\hat{\mu}_k(\tau)$ de chaque bras k après le τ^{eme} tour de round-robin est :

$$\hat{\mu}_k(\tau) = \sum_{r \in \mathbb{T}(\tau)} \frac{[\![r = [\tau]]\!]}{\tau} \sum_{j=1}^{t_\tau + |S_\tau| - 1} y_k(j) [\![k = k_j]\!] .$$

En décompensant la seconde somme en tours de round-robin puis en prenant l'espérance vis-à-vis de la distribution de récompenses D_y, nous avons :

$$\mathbb{E}_{D_y}[\hat{\mu}_k(\tau)] = \sum_{r\in\mathbb{T}(\tau)} \frac{[\![r=[\tau]]\!]}{\tau} \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \mu_k(j)[\![k=k_j]\!]\,. \tag{2.19}$$

En prenant maintenant l'espérance de l'équation (2.19) vis-à-vis de la randomisation des tours round-robin, nous avons :

$$\mathbb{E}[\hat{\mu}_k(\tau)] = \sum_{r\in\mathbb{T}(\tau)} \frac{[\![r=[\tau]]\!]}{\tau} \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_k(j)}{|S_i|}\,. \tag{2.20}$$

Maintenant, lorsque l'évènement Ω se produit, c'est-à-dire que l'inéquation (2.18)est vraie pour k et k', en utilisant (2.20) nous déduisons que

$$\sum_{r\in\mathbb{T}(\tau)} \frac{[\![r=[\tau]]\!]}{\tau} \left(\sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_k(j)}{|S_i|} - \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_{k'}(j)}{|S_i|} \right) < 4\epsilon_\tau\,. \tag{2.21}$$

Nous rappelons la notion d'écart moyen :

$$\Delta_{k,k'}([\tau]) = \sum_{r\in\mathbb{T}(\tau)} \frac{[\![r=[\tau]]\!]}{\tau} \left(\sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_k(j)}{|S_i|} - \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_{k'}(j)}{|S_i|} \right)\,.$$

En remplaçant la valeur de ϵ_t dans (2.21), nous avons :

$$\Delta_{k,k'}([\tau]) < 4\sqrt{\frac{1}{2\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}\,,$$

$$\Delta_{k,k'}([\tau])^2 < \frac{8}{\tau}\log\left(\frac{4K\tau^2}{\delta}\right)\,. \tag{2.22}$$

Un bras sera éliminé si (2.22) devient fausse et si $\tau \geq \tau_{\min}$.

2 - La non-élimination du bras optimal :
Pour $k' = k^*$ et $k \neq k^*$, par hypothèse ($\Delta_{k,k^*}([\tau])$ est négatif après $\tau_{\min}$), (2.22) est toujours vraie lorsque $\tau \geq \tau_{\min}$, impliquant que le bras optimal ne sera jamais éliminé avec une forte probabilité pour tout τ.

3 - L'élimination des bras sous-optimaux :
Si le bras k' est toujours dans l'ensemble, alors il sera éliminé si l'inégalité (2.22) n'est pas vérifiée et si $\tau^* \geq \tau_{\min}$.

Considérons maintenant le bras optimal k^* et le bras sous-optimal $k \neq k^*$, et définissons la quantité

$$\Delta_k([\tau]) = \sum_{r \in \mathbb{T}(\tau)} \frac{[\![r = [\tau]]\!]}{\tau} \left(\sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_{k^*}(j)}{|S_i|} - \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_k(j)}{|S_i|} \right).$$

Nous introduisons aussi la valeur critique

$$\tau^* = \frac{64}{\Delta_k([\tau])^2} \log\left(\frac{4K}{\delta \Delta_k([\tau])}\right).$$

Remarquons que $\tau^* \geq \tau_{\min}$, satisfaisant l'une des deux conditions d'élimination.

Nous notons

$$C(t) = \frac{8}{\tau} \log\left(\frac{4K\tau^2}{\delta}\right).$$

Pour $\tau = \tau^*$, nous dérivons la borne suivante :

$$\begin{aligned}
C(\tau^*) &= \frac{\Delta_k([\tau])^2}{8 \log \frac{4K}{\delta \Delta_k([\tau])}} \left(\log \frac{4K}{\delta} + 2 \log \frac{64K^2}{\Delta_k([\tau])^2} + 2 \log\log \frac{4K}{\delta \Delta_k([\tau])} \right), \\
&= \frac{\Delta_k([\tau])^2}{8 \log \frac{4K}{\delta \Delta_k([\tau])}} \left(\log \frac{4K}{\delta} - 4 \log \Delta_k([\tau]) + 12 \log 2 + 2 \log\log \frac{4K}{\delta \Delta_k([\tau])} \right), \\
&\leq \frac{\Delta_k([\tau])^2}{8 \log \frac{4K}{\delta \Delta_k([\tau])}} \left(4 \log \frac{4K}{\delta \Delta_k([\tau])} + 12 \log 2 + 2 \log\log \frac{4K}{\delta \Delta_k([\tau])} \right).
\end{aligned}$$

Nous remarquons que pour $X > 13$ nous avons :

$$12 \log 2 + 2 \log\log X < 4 \log X.$$

Ainsi, sous réserve que $K \geq 2$, $\delta \in (0, 0.5]$ et $\Delta_k([\tau]) > 0$, nous avons $\frac{4K}{\delta \Delta_k([\tau])} > 13$ et

$$\begin{aligned}
C(\tau^*) &\leq \frac{\Delta_k([\tau])^2}{8 \log \frac{4K}{\delta \Delta_k([\tau])}} \left(8 \log \frac{4K}{\delta \Delta_k([\tau])} \right) \\
&\leq \Delta_k([\tau])^2. \qquad (2.23)
\end{aligned}$$

Comme $C(\tau^*)$ est strictement décroissante suivant t, (2.23) est vraie pour tout $\tau > \tau^*$, invalidant (2.22) et impliquant l'élimination du bras sous-optimal k avec une probabilité d'au moins $1 - \delta/K$.

Nous concluons la preuve en sommant sur l'ensemble des bras, en utilisant l'inégalité de Boole et en bornant inférieurement tout les $\Delta_k([\tau])$ par

$$\Delta = \min_{[\tau] \in \mathbb{T}(\tau), k} \sum_{r \in \mathbb{T}(\tau)} \frac{[\![r = [\tau]]\!]}{\tau} \left(\sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_k(j)}{|S_i|} - \sum_{i=1}^{\tau} \sum_{j=t_i}^{t_i+|S_\tau|-1} \frac{\mu_{k'}(j)}{|S_i|} \right). \qquad (2.24)$$

□

2.6.2 Preuve du Corollaire 2

Démonstration. Nous prouvons dans un premier temps la borne supérieure dépendante des distributions.

Soit τ_k le dernier tour de round-robin avant l'élimination du bras k.

Le pseudo-regret cumulé de l'algorithme est :

$$\bar{R}(T) = \sum_{k \neq k^*} \sum_{i=1}^{\tau_k} \sum_{t=t_i}^{t_i+|S_i|-1} \Delta_{k^*,k}(t) [\![k = k_t]\!] \,. \tag{2.25}$$

En prenant l'espérance de la variable aléatoire k_t vis-à-vis de la randomisation des round-robin (notée $\mathbb{E}_{k_t}$), nous avons :

$$\begin{aligned} \mathbb{E}[\bar{R}(T)] &= \mathbb{E}\Big[\sum_{k \neq k^*} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \mathbb{E}_{k_t}[\Delta_{k^*,k}(t) [\![k = k_t]\!]] \Big] \\ &= \mathbb{E}\Big[\sum_{k \neq k^*} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \frac{\Delta_{k^*,k}(t)}{|S_i|} \Big] \,. \end{aligned}$$

$$\mathbb{E}[\bar{R}(T)] = \mathbb{E}\Big[\sum_{k \neq k^*} \tau \underbrace{\frac{1}{\tau} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \frac{\Delta_{k^*,k}(t)}{|S_i|}}_{\Delta_k^*} \Big] = \mathbb{E}\Big[\sum_{k \neq k^*} \tau \Delta_k^* \Big] \,. \tag{2.26}$$

L'avant-dernière étape de la preuve du théorème 11 nous permet de borner supérieurement τ_k avec la valeur τ^* sur un évènement de forte probabilité $1 - \delta$, tandis-que le regret cumulé est contrôlé par la borne supérieure triviale T sur l'évènement complémentaire de probabilité au plus δ, menant à :

$$\mathbb{E}[\bar{R}(T)] \leq \sum_{k \neq k^*} \frac{64}{\Delta_k^2} \log\left(\frac{4K}{\delta \Delta_k}\right) \Delta_k + \delta T \,. \tag{2.27}$$

Nous concluons la preuve de la borne distribution dépendante en fixant $\delta = 1/T$:

$$\mathbb{E}[\bar{R}(T)] = O\left(\frac{K-1}{\Delta} \log(\frac{KT}{\Delta}) \right) \,, \tag{2.28}$$

avec $\Delta = \min_{[\tau],k} \frac{1}{\tau} \sum_{i=1}^{\tau} \sum_{t=t_i}^{t_i+|S_i|-1} \frac{\Delta_{k^*,k}(t)}{|S_i|}$.

Nous bornons maintenant supérieurement le regret cumulé dans le pire des cas pour obtenir une borne indépendante de la distribution. Nous construisons pour cela une séquence de récompenses évitant l'élimination des bras sous-optimaux

avec une forte probabilité tout en maximisant le regret instantané à la fin des T tours de jeu. Conformément à (2.18), un bras n'est pas éliminé tant que :

$$\mathbb{E}[\hat{\mu}_k(\tau)] - \mathbb{E}[\hat{\mu}_{k'}(\tau)] < 4\epsilon_\tau \,. \tag{2.29}$$

En injectant (2.29) dans (2.26) et en remplaçant ϵ_τ par sa valeur $\sqrt{\frac{2}{\tau}\log\left(\frac{4K\tau^2}{\delta}\right)}$ nous obtenons :

$$\mathbb{E}[\bar{R}(T)] < \sum_{k\neq k^*} \tau 4\sqrt{\frac{2}{\tau}\log\left(\frac{4K\tau^2}{\delta}\right)} + \delta T \,. \tag{2.30}$$

Par construction, les bras sous-optimaux ne sont pas éliminés et $\tau = \frac{T}{K}$. En fixant $\delta = \frac{1}{T}$ nous obtenons la borne supérieure indépendante de la distribution :

$$\mathbb{E}[\bar{R}(T)] < (K-1)\frac{T}{K}4\sqrt{\frac{K}{T}\log(\frac{4T^3}{K})} + 1 \,, \tag{2.31}$$

$$\mathbb{E}[\bar{R}(T)] = O\left(\sqrt{TK\log\frac{T}{K}}\right) \,. \tag{2.32}$$

□

2.6.3 Preuve du Théorème 12

Démonstration. Les quantités suivantes sont considérées :

- L'espérance du nombre de fois où les estimateurs sont réinitialisés est $N_{\text{reinit}} = \varphi T$.
- La complexité d'échantillonnage nécessaire à l'obtention de le meilleur bras entre chaque réinitialisation est $S_{\text{SER3}} = O\left(\frac{K}{\Delta^2}\log(\frac{K}{\delta\Delta})\right)$.
- Le temps écoulé avant une réinitialisation suit une loi binomiale négative de paramètres $r = 1$ et $p = 1-\varphi$. Son espérance est bornée supérieurement par $1/\varphi$.
- Le nombre de changements de bras est $N - 1$.

La complexité d'échantillonnage de SER4 est le nombre total d'itérations passées à jouer chaque bras plus le temps écoulé entre chaque changement de meilleurs bras et la réinitialisation de l'algorithme.

L'espérance de la complexité d'échantillonnage vis-à-vis de la randomisation des réinitialisations est :

$$O\left(\frac{\varphi TK}{\Delta^2}\log\left(\frac{K}{\delta\Delta}\right) + \frac{N}{\varphi}\right) \,. \tag{2.33}$$

Le premier terme est l'espérance du nombre total de pas de temps où l'algorithme échantillonnera les bras à la première initialisation de l'algorithme puis après chaque réinitialisation. Le second terme est l'espérance du nombre de pas de temps perdus par l'algorithme avant une réinitialisation après les $N-1$ changements de bras. Nous concluons la preuve en fixant $T = \frac{1}{\delta}$. □

2.6.4 Preuves des Corollaires 3 et 2.6

Démonstration. Convertir le théorème 3 en une borne supérieure sur le regret cumulé dépendante de la distribution est direct, en fixant $\delta = \frac{1}{T}$, puis en remplaçant la borne sur complexité d'échantillonnage par celle sur le regret cumulé obtenue dans le Corollaire 2.

$$\mathbb{E}[\bar{R}(T)] = O\left(\frac{\varphi TK}{\Delta}\log\left(\frac{KT}{\Delta}\right) + \frac{N}{\varphi}\right). \tag{2.34}$$

En fixant $\varphi = \sqrt{\frac{N}{TK\log(KT)}}$ et en supposant $\Delta \geq \frac{1}{KT}$ nous obtenons la borne suivante :

$$\mathbb{E}[\bar{R}(T)] = O\left(\frac{\sqrt{NTK\log(KT)}}{\Delta}\right), \tag{2.35}$$

concluant la preuve de la première partie du Corollaire.

Nous montrons maintenant la borne supérieure indépendante de la distribution. Nous introduisons tout d'abord quelque notations, N_{reinit} est le nombre de réinitialisations de l'algorithme, τ_i^{reinit} est le nombre de tours de round-robin entre la i^{eme} et la $i+1^{\text{eme}}$ réinitialisation et L_n est le nombre de pas de temps avant qu'une réinitialisation se produise après le n^{eme} changement de meilleur bras.

Quand les pas de temps où les réinitialisations se produisent sont fixés, l'espérance du regret cumulé est :

$$\mathbb{E}[\bar{R}(T)] < \mathbb{E}\left[\sum_{i=1}^{N_{\text{reinit}}+1}(K-1)\tau_i^{\text{reinit}}4\sqrt{\frac{2}{\tau_i^{\text{reinit}}}\log(\frac{4(\tau_i^{\text{reinit}})^2}{\delta})} + \sum_{n=1}^{N}L_n + \delta T\right], \tag{2.36}$$

$$\mathbb{E}[\bar{R}(T)] < \mathbb{E}\left[\sum_{i=1}^{N_{\text{reinit}}+1}\underbrace{(K-1)4\sqrt{2\tau_i^{\text{reinit}}\log(\frac{4(\tau_i^{\text{reinit}})^2}{\delta})}}_{f(\tau_i^{\text{reinit}})}\right] + \mathbb{E}\left[\sum_{n=1}^{N}L_n\right] + \delta T\,. \tag{2.37}$$

Nous remarquons que $\{\tau_i^{\text{reinit}}\}_i$ est une séquence i.i.d. de variables aléatoires et que N_{reinit} est un temps d'arrêt aléatoire vis-à-vis de cette séquence. De plus, f est

une fonction concave. Nous pouvons donc appliquer l'inéquation de Wald suivie de l'inégalité de Jensen et déduire que :

$$\begin{aligned}\mathbb{E}[\sum_{i=1}^{N_{\text{reinit}}+1} f(\tau_i^{\text{reinit}})] &\leq \mathbb{E}[N_{\text{reinit}}+1]\mathbb{E}[f(\tau_1^{\text{reinit}})] \\ &\leq \mathbb{E}[N_{\text{reinit}}+1] f(\mathbb{E}[\tau_1^{\text{reinit}}])\,.\end{aligned}$$

Nous bornons supérieurement $\log(\frac{4(\tau_i^{\text{reinit}})^2}{\delta})$ par $\log(\frac{4T^2}{\delta K^2})$ et fixons $\delta = \frac{1}{T}$. Étant donné que $\mathbb{E}[N_{\text{reinit}}] = \varphi T$, $\mathbb{E}[\tau_1^{\text{reinit}}] = \frac{1}{\varphi K}$ et que $\mathbb{E}[L_n] \leq \frac{1}{\varphi}$, nous avons :

$$\mathbb{E}[\bar{R}(T)] < 4(\varphi T+1)\sqrt{\frac{2}{\varphi}K\log\left(\frac{4T^3}{K^2}\right)} + \frac{N}{\varphi} + 1\,. \tag{2.38}$$

L'équation précédente fait apparaitre un compromis en φ et nous fixons $\varphi = \frac{\sqrt{N}}{T^{2/3}}$.

Finalement, nous avons montré que :

$$\mathbb{E}[\bar{R}(T)] = O\left(T^{2/3}\sqrt{NK\log\frac{T}{K}}\right)\,. \tag{2.39}$$

□

2.6.5 Preuve du Lemme 1

Démonstration. Nous justifions notre test de détection en considérant les γ-observations comme des tirages sans remplacement dans une urne. Plus précisément, quand toutes les observations nécessaires sont collectées, la procédure de détection est lancée. Durant l'intervalle de collecte, les récompenses sont tirées depuis $1 \leq m \leq M$ différentes distributions de moyenne $\mu_0^k(I), ..., \mu_m^k(I)$. Nous appelons t_i le pas de temps où la réponse moyenne commence à être $\mu_i^k(I)$ et t_{m+1} le pas de temps où le détecteur est appelé. A posteriori, chaque $x_k(t)$ a une probabilité $(t_{i+1}-t_i)/(t_{m+1}-t_0)$ d'avoir été tiré depuis la distribution de moyenne $\mu_i^k(I)$. L'espérance moyenne de l'urne contenant les récompenses correspondant au bras k est :

$$\mu^k(I) = \sum_{i=1}^{m} \frac{t_{i+1}-t_i}{t_{m+1}-t_0}\mu_i^k(I)\,. \tag{2.40}$$

A chaque pas de temps, par hypothèse, la récompense moyenne du meilleur bras est supérieure à celle de n'importe quel autre bras d'au moins 4ϵ. Par conséquent, la différence entre la récompense moyenne de l'urne du bras optimal k^* et celle d'un autre bras k est au moins de 4ϵ si le meilleur bras ne change pas au cours de l'intervalle.

$$\mu^k(I) \leq \sum_{i=1}^{m} \frac{t_{i+1}-t_i}{t_{m+1}-t_0}(\mu_i^{k_S^*} - 4\epsilon) \leq \mu^{k_S^*}(I) - 4\epsilon\,. \tag{2.41}$$

Les arguments suivants montrent l'équivalence entre une détection et l'optimalité de k_S^* avec une forte probabilité.

En utilisant l'inégalité de Serfling [38], nous avons :

$$P(\hat{\mu}^{k_S^*}(I) + \epsilon \geq \mu^{k_S^*}(I)) \leq e^{\frac{-2n\epsilon^2}{1-\frac{n-1}{U}}} \leq e^{-2n\epsilon^2} = \delta \tag{2.42}$$

et

$$P(\hat{\mu}^k(I) - \epsilon \leq \mu^k(I)) \leq \delta\,, \tag{2.43}$$

où $n = \frac{\gamma H}{K}$ est le nombre d'observations et U la taille de l'urne. Nous avons :

$$\mu^{k_S^*}(I) - \mu^k(I) \geq 4\epsilon \tag{2.44}$$

et avec une probabilité d'au moins $1 - 2\delta$,

$$\hat{\mu}^{k_S^*}(I) + \epsilon \geq \mu^{k_S^*}(I) \tag{2.45}$$

et

$$\hat{\mu}^k(I) - \epsilon \leq \mu^k(I) \tag{2.46}$$

En sommant (2.47), (2.45) et (2.46) nous obtenons :

$$\hat{\mu}^{k_S^*}(I) - \hat{\mu}^k(I) \geq 2\epsilon\,, \tag{2.47}$$

assurant avec une forte probabilité un test positif si $\hat{\mu}^{k_{\max}}$ n'est pas le bras optimal.

Réciproquement, nous avons aussi :

$$\hat{\mu}^k(I) - \hat{\mu}^{k_S^*}(I) \leq -2\epsilon\,. \tag{2.48}$$

assurant avec une forte probabilité un test négatif si $\hat{\mu}^{k_{\max}}$ est le bras optimal.

□

2.6.6 Preuve du Théorème 13

Démonstration. Premièrement, nous obtenons la structure principale de la borne. Dans ce qui suit, $L(T)$ décrit l'espérance du nombre d'intervalles après un changement de meilleur bras avant sa détection et $F(T)$ est l'espérance du nombre de fausses détections jusqu'alors. En utilisant les même arguments que [39], nous déduisons la forme de la borne depuis la borne classique d'EXP3. Il y a $N-1$ changements de meilleur bras, ainsi l'espérance du nombre de réinitialisations à l'horizon T est borné supérieurement par $N - 1 + F(T)$. Le regret d'EXP3 sur ces périodes est $(e-1)\gamma T + \frac{K \log K}{\gamma}$ [31]. Alors que notre politique optimale joue le bras ayant la plus haute récompense moyenne, la politique d'EXP3 joue le bras ayant la plus haute récompense cumulée sur la période $[T_S, T_{S+1})$. Comme

$$\sum_{t=T_S}^{T_{S+1}-1} x_{k_S^*}(t) \leq \max_k \sum_{t=T_S}^{T_{S+1}-1} x_k(t)\,, \tag{2.49}$$

le gain de notre politique optimale sur une période est borné supérieurement par le gain de la politique optimale d'EXP3. En sommant sur chaque période, nous obtenons $(e-1)\gamma T + \frac{(N-1+F(T))K\log K}{\gamma}$.
Le regret comprend aussi le délai entre le changement de meilleur bras et sa détection. Pour évaluer l'espérance de la taille de l'intervalle séparant chaque appel du test, nous considérons un algorithme fictif collectant uniquement les observations d'un bras avant de passer au suivant jusqu'à avoir collecté toutes les observations. Les γ-observations sont tirées avec une probabilité $\frac{\gamma}{K}$ et $\frac{\gamma H}{K}$ observations sont nécessaires par bras. L'espérance du nombre d'échecs avant de réussir à collecter $\frac{\gamma H}{K}$ γ-observations suit une loi binomiale négative d'espérance

$$\frac{\gamma H}{K}(1-\frac{\gamma}{K})\frac{K}{\gamma} = H - \frac{\gamma H}{K}\,. \tag{2.50}$$

L'espérance du nombre de tirages à la fin de la collecte est le nombre de succès plus le nombre espéré d'échecs :

$$\frac{\gamma H}{K} + H - \frac{\gamma H}{K} = H\,. \tag{2.51}$$

En sommant sur tous les bras, nous obtenons une espérance totale de HK. Étant donné que notre algorithme peut collecter les observations de n'importe quel bras à n'importe quel moment, sur une même séquence de tirages, notre algorithme aura fini sa collecte avant l'algorithme précédemment décrit. Par conséquent, l'espérance du temps entre chaque appel du test est bornée supérieurement par HK et inférieurement par H, l'espérance du temps de collecte pour un bras. Il y a $N-1$ changements de meilleur bras et la détection se produit en moyenne au plus $\lceil L(T)\rceil HK$ tour après le changement. Pour finir, il y a aussi au plus $N-1$ intervalles où les changements de meilleur bras se produisent. Durant ces intervalles, nous n'avons pas de garanties sur le comportement du test. Dans le pire des cas, le test ne détecte pas de changement et le regret instantané est de 1.

$$\begin{aligned} G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq\ & (e-1)\gamma T \\ & + \frac{(N-1+F(T))K\log K}{\gamma} + (N-1)HK(\lceil L(T)\rceil + 1)\,. \end{aligned} \tag{2.52}$$

Nous bornons maintenant $F(T)$ et $L(T)$. Les intervalles de confiance sont valides avec une probabilité d'au moins $1-\delta$ et ils sont utilisés K fois à chaque test. En utilisant l'inégalité de Boole, nous déduisons que $F(T) \leq K\delta(\frac{T}{H}+1)$. $L(T)$ est la première occurrence de l'évènement DÉTECTION après un changement de meilleur bras. Quand un tel changement se produit, le lemme 1 garantit que la détection se produira avec une probabilité d'au moins $1-2\delta$. Nous avons $L(T) \leq \frac{1}{1-2\delta}$.

$$G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq (e-1)\gamma T + \frac{\left(N - 1 + \frac{K\delta T}{H} + K\delta\right) K \log K}{\gamma} + (N-1)HK\left(\frac{1}{1-2\delta} + 1\right). \quad (2.53)$$

□

2.6.7 Preuve du Corollaire 5

Démonstration. Nous fixons les paramètres $\delta = \sqrt{\frac{\log T}{KT}}$ et $H = C\sqrt{T \log T}$ dans le théorème 13

$$G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq (e-1)\gamma T + \frac{(N - 1 + (C+1)\sqrt{K})K \log K}{\gamma} + 3(N-1)CK\sqrt{T \log T}. \quad (2.54)$$

Avec $\gamma = \sqrt{\frac{K \log K \log T}{T}}$ nous obtenons :

$$G^* - \mathbf{E}[G_{\text{EXP3.R}}] \leq (e-1)\sqrt{TK \log K \log T} + N\sqrt{TK \log K} + (C+1)K\sqrt{T \log K} + 3NCK\sqrt{T \log T}. \quad (2.55)$$

□

Deuxième partie

Le problème des bandits contextuels

Chapitre 3

Formalisme et état de l'art

Contents

3.1 Introduction

Le problème des bandits contextuels est une généralisation du problème des bandits manchots présenté dans la partie I. Il possède un grand nombre d'utilisations pratiques, notamment en optimisation publicitaire, en recommandation de produits, en maintient de la qualité de service dans les boîtiers décodeurs ou en optimisation du trafic mobile sur les réseaux sans fils. Dans ce formalisme, les récompenses obtenues par le joueur dépendent à la fois du bras joué et d'un vecteur de contexte. Deux principales formulations existent. La première a été initialement introduite par Langford [40] et est une généralisation du problème de classification en ligne. À chaque tour, un contexte est produit par l'environnement et présenté au joueur. Après l'observation du contexte, le joueur doit sélectionner le bras pouvant lui apporter la récompense la plus élevée. Contrairement à la classification en ligne à information totale, le joueur n'observe que la récompense associée au bras joué et ne pas connaît le bras optimal associé à ce contexte. Maintenir un estimateur des récompenses de chacun des bras n'est plus suffisant et il est maintenant nécessaire pour le joueur de découvrir la relation entre les contextes et les

récompenses obtenues pour chacun des bras. La seconde formulation est utilisée par Li et al. [41]. Dans celle-ci, un contexte est associé à chacun des bras, permettant de représenter leurs similarités. En effet, il est raisonnable de supposer que deux bras similaires obtiendront des récompenses similaires. Cette représentation du problème est utile afin de résoudre des problèmes de bandit ayant un nombre de bras très élevé. Il devient alors possible d'estimer la récompense moyenne d'un bras, sans avoir à le jouer, uniquement à partir de son contexte grâce aux capacités de généralisation du modèle utilisé.

Dans ce chapitre ainsi que dans les contributions de cette partie de la thèse, nous nous concentrerons principalement sur le formalisme utilisant des contextes produits par l'environnement.

3.2 Contextes produits par l'environnement

Soit une séquence $((x_1, \mathbf{y}_1), ..., (x_T, \mathbf{y}_T))$ avec $x_t \in X$ un contexte, $k \in [K] = \{1, ..., K\}$ un des K bras pouvant être choisis et $\mathbf{y}_t \in Y$ un vecteur de récompense $(y_{t,1}, ..., y_{t,K})$ avec $y_{t,k} \in [0, 1]$ la récompense pour le bras k. Les séquences peuvent être tirées selon un processus stochastique ou définies à l'avance par un adversaire. Le problème est répété sur T tours : à chaque tour $t < T$ le contexte x_t est annoncé. Le joueur, qui cherche à maximiser la somme de ses récompenses, choisit un bras k_t. La récompense y_{t,k_t} du bras choisi par le joueur, et uniquement celle-ci, est dévoilée.

Algorithme 14 : BANDIT CONTEXTUELS AVEC CONTEXTES PRODUITS PAR L'ENVIRONNEMENT

pour $t = 1; t \leq T; t++$ **faire**
  Le contexte x_t est dévoilé au joueur
  Le joueur joue le bras k_t
  L'environnement dévoile la récompense y_{t,k_t}
  Le joueur modifie sa politique de choix de bras
fin

Soit $\Pi : X \to [K]$ un ensemble de politiques.Soit $D_{x,y}$ une distribution sur les contextes et les récompenses. La politique optimale est :

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmax}} \, \mathbb{E}_{x,y}[y_{t,\pi(x)}] \, .$$

Complexité d'échantillonnage dans le cas stationnaire La complexité d'échantillonnage est le nombre d'observations nécessaires à un algorithme pour trouver une politique $\pi \in \Pi$ telle que, avec une probabilité $1 - \delta$, $\mathbb{E}_{x,y}[y_{t,\pi(x)}] \geq \mathbb{E}_{x,y}[y_{t,\pi^*(x)}] - \epsilon$.

Regret cumulé dans le cas stationnaire Soit $\pi_t \in \Pi$ une politique trouvée

par l'algorithme A au tour t. Le pseudo-regret instantané est :

$$R_t(A) = \mathbb{E}_{x,y}[y_{t,\pi^*(x_t)} - y_{t,\pi_t(x_t)}] \,.$$

Le pseudo-regret cumulé est :

$$R(A) = \sum_{t=1}^{T} R_t(A) \,.$$

La non-stationnarité dans les bandits contextuels Dans le cas contextuel, la non-stationnarité peut prendre différentes formes, suivant si elle porte sur :

- La distribution $D_{y|x}$ des récompenses, où elle est appelée **concept-drift/shift**. Ce type de non-stationnarité se produit lorsque seule la distribution des récompenses change. Un exemple trivial est le cas du bandit manchot non-stationnaire, le contexte est vide et ne change pas. En pratique, on peut considérer que ce type de non-stationnarité se produit uniquement lorsque peu de variables sont observées et que la non-stationnarité impacte ces variables non observée.
- La distribution D_x des exemples, aussi appelée **covariate-shift** [42]. Ce type de non-stationnarité peut par exemple être observé lorsque l'environnement représenté par certaines variables de contexte change (un capteur qui est déplacé), lors du lancement d'une offre ciblant de nouveaux clients ou bien lors de la fusion de deux bases de données. Localement à un bras, le **covariate-shift** peut aussi être causé par les choix de l'algorithme lui-même. Par exemple, lorsqu'une approche un contre tous est utilisée. Dans ce cas-là, un modèle prédictif est associé à chacun des bras et le bras ayant la plus haute prédiction est joué. La distribution des contextes pour lequel ce bras est joué peut changer au cours du temps si les modèles sont mis à jour au cours d'une tâche d'apprentissage, même si les distributions globalement observées par l'algorithme ne varient pas. Si aucun mécanisme compensant ce **covariate-shift** n'est mis en place, alors l'apprentissage du modèle est biaisé.
- La non-stationnarité peut aussi être un mélange de **concept-drift/shift** et de **covariate-shift** et modifier les distributions D_x et $D_{y|x}$.

3.3 Algorithmes construisant des modèles prédictifs

Une première approche aux bandits contextuels consiste en la création de modèles prédictifs. Les algorithmes utilisant des modèles prédictifs cherchent à capturer la dépendance entre les contextes, les bras et les récompenses. L'algorithme Epoch-Greedy (voir sous-section 3.3.1) permet de convertir tout algorithme de

classification en ligne en algorithme de bandits contextuels. Les algorithmes BANDITRON (voir sous-section 3.3.2) ou LINUCB (voir sous-section 3.3.3) adaptent quant à eux des modèles linéaires aux problème des bandits contextuels.

3.3.1 Epoch-Greedy

Epoch-Greedy [40] est un algorithme générique semblable aux algorithmes γ-GREEDY. Durant le jeu, il alterne les phases d'exploration uniforme et l'exploitation. Les exemples d'apprentissage sont collectés durant l'exploration uniforme et un modèle prédictif est appris hors ligne. Il est ensuite déployé et utilisé pour choisir les bras jusqu'à la prochaine phase d'exploration. Le principal défi motivant ces phases d'exploration et d'exploitation successives est de pouvoir avoir un algorithme ne nécessitant pas la connaissance de l'horizon T. Si l'horizon T était connu, il suffirait de faire une seule grande phase d'exploration au début du jeu pour pouvoir ensuite utiliser un modèle appris avec un maximum d'exemple durant la phase d'exploitation. Afin d'obtenir des garanties, l'analyse suppose l'existence d'une borne supérieure S sur la complexité d'échantillonnage du modèle prédictif utilisé. Si cette borne S existe, alors sans connaissance à priori sur la distribution des données, il est possible de définir des tailles de périodes d'exploitation permettant d'obtenir une borne sur l'espérance du regret cumulé de l'ordre de $O(T^{2/3}S^{1/3})$ dans le pire des cas.

Algorithme 15 : EPOCH-GREEDY

Données : $\{\Gamma_1, , ..., \Gamma_L\}$ avec $\forall l$, $\Gamma_l < T$
$t = 1$, $l = 1$
EnsembleApprentissage $= \{\emptyset\}$;
répéter
  `// Phase d'exploration`
  Observer x_t;
  Jouer un bras $k_t \in [K]$ tiré depuis une distribution uniforme et recevoir y_{t,k_t};
  Ajouter (x_t, y_{t,k_t}) à l'ensemble d'apprentissage;
  Optimiser π_i sur l'ensemble d'apprentissage;
  `// Phase d'exploitation`
  pour $t = t + 1; t < \Gamma_l; t++$ **faire**
    Observer x_t;
    Jouer le bras $\pi_t(x_t)$;
  fin
  $l = l + 1$;
jusqu'à *Arrêt de l'algorithme*;

3.3.2 Banditron

Le Banditron [43] est une variante du Perceptron [44] adaptée au problème des bandits contextuels. K perceptrons apprennent chacun un hyperplan séparant la classe $k \in [K]$ des autres classes et classifient un contexte à l'aide d'une approche un contre tous. Sous contrainte de séparabilité, le nombre d'erreurs de classification de Banditron est borné supérieurement par $O(\sqrt{TKM})$ (en supposant que $T \geq d\sqrt{d/K}$). Banditron s'adapte très vite aux changements de non-stationnarité, mais sa règle de mise à jour, dérivée de l'algorithme de Rosenblatt offre des performances médiocres en présence de bruit. De plus, sa règle de mise en jour implique des récompenses binaires.

Algorithme 16 : BANDITRON

Données : $\gamma \in (0, 0.5)$
$\forall k \in [K]$, $W_t^k = 0^M$
pour $t = 0; t \leq T; t++$ **faire**
  Observer x_t;
  $k_t^{\max} = \arg\max_{k \in [K]} W_t^k x_t$;
  $\forall k \in [K]$, $P_t(k) = (1-\gamma)[\![k = k_t^{\max}]\!] + \frac{\gamma}{K}$;
  Jouer le bras k_t tiré depuis P et recevoir y_{t,k_t};
  $\forall k \in [K]$ et $\forall i \in [0, K]$, $W_{t+1}^k = W_t^k - x_t \left(\frac{y_{t,k_t}[\![k=k_t]\!]}{P_t(k)} - [\![k = k_t^{\max}]\!] \right)$;
fin

3.3.3 LinUCB et CTS

Bandits avec des contextes associés aux bras Une seconde formulation du bandit contextuel est introduite avec l'algorithme LinUCB [**?**]. Ici, K contextes sont disponibles à chaque tour et le joueur doit choisir l'un des contextes.

Soit une séquence $((x_1^1, ..., x_1^K, \mathbf{y}_1), ..., (x_T^1, ..., x_T^K, \mathbf{y}_T))$ avec $x_t^k \in X$ un contexte de taille d associé au bras $k \in [K] = \{1, ..., K\}$ et $\mathbf{y}_t \in Y$ un vecteur de récompense $(y_{t,1}, ..., y_{t,K})$ avec $y_{t,k} \in [0, 1]$ la récompense pour le bras k. Les séquences peuvent être tirées selon un processus stochastique ou définies à l'avance par un adversaire. Le problème est répété sur T tours : à chaque tour $t < T$ les contextes $x_t^0, ..., x_t^K$ sont annoncés. Le joueur choisit un bras k_t. La récompense y_{t,k_t} du bras choisi par le joueur, et uniquement celle-ci, est dévoilée.

Regret cumulé dans le cas stationnaire Soit $\Pi : X \to [0, 1]$ un ensemble de politiques et $\pi_t \in \Pi$ une politique trouvée par l'algorithme A au tour t. Le bras joué par la politique π en présence des contextes $\{x^0, ..., x^K\}$ est

$$k(\pi_t, x^0, ..., x^K) = \underset{k \in [K]}{\operatorname{argmax}} \pi_t(x^k) \,.$$

Dans le cas de LinUCB et CTS, les politiques sont des modèles linéaires de paramètre $\theta \in \mathbb{R}^d$. Le bras joué par la politique de paramètre θ est $k(\theta, x^0, ..., x^K) = \underset{k \in [K]}{\operatorname{argmax}}\, \theta x^k$.

L'hypothèse optimale est $\pi^* = \underset{h \in H}{\operatorname{argmax}}\, \mathbb{E}[y_{t,k(\pi,x^0,...,x^K)}]$. Pour faciliter les notations, nous définissons $k_t = k(\pi_t, x_t^0, ..., x_t^K)$ et $\pi_t^* = k(\pi^*, x_t^0, ..., x_t^K)$. Le regret instantané est :

$$R_t(A) = \mathbb{E}[y_{t,\pi_t^*} - y_{t,\pi_t}]\,.$$

Le regret cumulé est :

$$R(A) = \sum_{t=1}^{T} R_t(A)\,.$$

Algorithmes Deux versions de LinUCB ont été présentées. La version initialement proposée par Li, LinUCB avec modèles linéaires disjoints [41], un seul contexte est disponible à chaque tour et l'algorithme effectue une régression linéaire par bras pour estimer les récompenses et avec un intervalle de confiance similaire à celui d'UCB, choisit le bras possédant la plus haute estimation. La seconde version de l'algorithme [45] résout la variante du problème des bandits contextuels où il n'y a qu'une seule distribution de récompense et K contextes différents par tour.

Algorithme 17 : LINUCB

Données : $\alpha \in \mathbb{R}^+$
$A = I_d$; `// La matrice identité de taille` d `par` d
$b = 0_d$; `// Le vecteur nul de taille` d
pour $t = 1; t \leq T; t++$ **faire**
  $\theta_t = A^{-1}b$;
  Observer $x_t^1, ..., x_t^K$;
  pour $k = 1; k \leq K; k++$ **faire**
    $\hat{\mu}_{t,k} = \theta_t^\top x_t^k + \alpha\sqrt{{x_t^k}^\top A^{-1} x_t^k}$;
  fin
  Jouer $k_t = \arg\max_k \hat{\mu}_{t,k}$;
  Recevoir $y_{t,k}$;
  $A = A + x_t^{k_t} {x_t^{k_t}}^\top$;
  $b = b + x_t^{k_t} y_{t,k}$;
fin

Une variante de l'algorithme, SupLinUCB [45], a été introduite à cause de difficultés techniques dans l'analyse de LinUCB et son regret est borné par $O\left(\sqrt{TM\log^3(KT\log(T)/\delta)}\right)$, où d est la dimension du contexte. Dans [46], une méthode utilisant une nouvelle inégalité de concentration basée sur des techniques

de martingales a été proposée pour définir la valeur adéquate de α, permettant ainsi d'obtenir une borne en $O(M\sqrt{T\log((1+T)/\delta)})$.

Le Thompson Sampling contextuel (CTS) [47] résout un système linéaire de la même manière que LinUCB pour obtenir les paramètres de la distribution qui servira au tirage du bras. L'algorithme obtient une borne sur le regret de $O(M^2\sqrt{T^{1/\log T}}\log T)$.

LinUCB et CTS nécessitent l'inversion d'une matrice de covariance, un processus couteux qui possède une complexité de $O(M^3)$. Dans ce cas particulier, il existe un algorithme itératif nécessitant la matrice inverse à l'instant $t-1$ dont la complexité est $O(M^2N)$ où N est le nombre de composantes modifiées du nouveau vecteur. Dans le cas de vecteurs creux, la complexité peut-être quadratique, elle redevient cependant cubique si les vecteurs sont pleins. Cette complexité limite l'utilisation de ces algorithmes pour le traitement de problèmes en grande dimension.

3.4 Algorithmes de sélection de politiques

Une autre approche au problème des bandits contextuels est la sélection de politiques. L'algorithme considère un ensemble de politiques Π et cherche la meilleure dans cet ensemble. Les bornes sur le regret de ces algorithmes possèdent une dépendance vis-à-vis du nombre de politiques en $O(\sqrt{\log|\Pi|})$, leur permettant d'en comparer un nombre élevé.

Ces algorithmes possèdent cependant plusieurs faiblesses entravant leur utilisation pratique :

- Leurs complexités algorithmiques sont sur-linéaires, allant à l'encontre des contraintes de temps de réponse des algorithmes en ligne.
- La complexité en mémoire est linéaire en $|\Pi|$ si chaque politique doit être instanciée.
- Si l'algorithme considère une classe de politiques, il faut construire une stratégie discrétisant l'espace des politiques afin de pouvoir les dénombrer.

3.4.1 RandomizedUCB

Les algorithmes classiques de bandits cherchent à estimer une distribution par bras, leur regret grandissant linéairement en fonction du nombre de bras. Afin d'éviter ce regret linéaire en K, RandomizedUCB [48] utilise un oracle résolvant un problème d'optimisation convexe fournissant une distribution de probabilité sur les politiques. Son regret en $O\left(\sqrt{TK\log\frac{|\Pi|}{\delta}}\right)$ est presque optimal mais sa complexité algorithmique est polynomiale, nécessitant entre $\tilde{O}(T^5)$ et $\tilde{O}(T^6)$ appels à l'oracle

(qui est lui-même un algorithme d'apprentissage hors-ligne ayant une complexité linéaire en fonction du nombre d'observations).

3.4.2 ILOVETOCONBANDITS

L'algorithme ILOVETOCONBANDITS [49] reprend les principes de RandomizedUCB mais réduit le nombre d'appel à l'oracle en l'utilisant seulement lors de pas de temps spécifiques. Cette approche permet de réduire le nombre d'appels à l'oracle à $\tilde{O}\left(\sqrt{TK/\log\frac{|\Pi|}{\delta}}\right)$ avec une probabilité $1-\delta$. Cependant, sa complexité algorithmique reste toujours de l'ordre de $\tilde{O}((KT)^{3/2})$.

3.4.3 EXP4

EXP4 [31] est une extension d'EXP3 pour le cas où un ensemble d'experts est disponible. À chaque nouvelle itération, les experts fournissent chacun un vecteur de probabilité sur les bras. EXP4 ne fait pas d'hypothèse de stationnarité sur les contextes, les récompenses ou les politiques. Nous détaillons maintenant le formalisme utilisé par EXP4. Un expert m est une séquence infinie $\xi^m(1), \xi^m(2), ... \in [0,1]^K$ de vecteurs de probabilités, où la k^{eme} composante $\xi^m_k(t)$ de $\xi^m(t)$ représente la probabilité recommandée de jouer le bras k au temps t. Cette formulation étant adverse, la séquence des récompenses $y_k(t) \in [0,1]$ est définie avant le début du jeu par un adversaire inconscient. L'expert optimal jusqu'à l'horizon T est :

$$m^* = \arg\max_m \sum_{t=1}^{T} \xi^m(t)\mathbf{y}(t)\,.$$

EXP4 obtient un regret faible en $O(\sqrt{TK\log|\Pi|})$ vis-à-vis du meilleur expert de l'ensemble. Il souffre cependant d'une complexité algorithmique élevée, celle-ci étant linéaire en $K|\Pi|$.

Chapitre 4

Comités de Réseaux de Neurones pour les Bandits Contextuels Non Stationnaires

Contents

4.1 Introduction

Les algorithmes de bandits contextuels présentés en section II présentent certaines faiblesses pour une utilisation pratique dans un contexte non-stationnaire.

La stationnarité. Hormis EXP4, tous les algorithmes présentés supposent que la distribution $D_{y|x}$ est stationnaire. Banditron peut cependant s'adapter aux changements de stationnarité grâce à sa formule de mise à jour similaire à une descente de gradient.

La linéarité. BANDITRON, LINUCB et CTS sont des classificateurs linéaires, si des non linéarités sont présentes dans les données, ces algorithmes auront de mauvaises performances.

La complexité algorithmique. LINUCB et CTS nécessitent l'inversion d'une matrice lors de la mise à jour de leurs indicateurs, cette opération a une complexité quadratique vis-à-vis du nombre de variables. Les algorithmes comme RANDOMIZEDUCB ou ILOVETOCONBANDIT ont des complexités algorithmiques sur-linéaires en T, les rendant difficilement utilisable lorsque l'horizon de temps grandit. De la même manière, EXP4 nécessite un temps de calcul à chaque itération linéaire en $K|\Pi|$, limitant le nombre d'experts de l'ensemble.

Dans ce chapitre, nous présentons tout d'abord un algorithme estimant les distributions des récompenses à partir des contextes à l'aide de réseaux de neurones. Une méthode γ-GREEDY est utilisée (voir section 1.1.3) pour contrôler l'exploration. Afin de pouvoir suivre naturellement la non-stationnarité, nous choisissons de modéliser la dépendance entre les contextes et les récompenses par des réseaux de neurones pouvant être appris par descente de gradient. Dans un premier temps, nous décrivons la manière dont les réseaux de neurones sont architecturés ainsi que l'algorithme de descente de gradient. Nous décrirons ensuite leur utilisation dans le cadre du problème de bandits contextuels.

4.2 Formalisation

Nous reprenons le formalisme présenté en sous-section 3.2. Soit une séquence $((x_1, \mathbf{y}_1), ..., (x_T, \mathbf{y}_T))$ avec $x_t \in X$ un contexte, $k \in [K] = \{1, ..., K\}$ un des K bras pouvant être choisis et $\mathbf{y}_t \in Y$ un vecteur de récompense $(y_{t,1}, ..., y_{t,K})$ avec $y_{t,k} \in [0, 1]$ la récompense pour le bras k. Soit $D_{x,y}(t)$ la distribution générant les contextes et les récompenses. La distribution $D_{x,y}(t)$ peut évoluer au cours du temps. Le problème est répété sur T tours : à chaque tour $t < T$ le contexte x_t est annoncé. Le joueur, qui cherche à maximiser la somme de ses récompenses, choisit un bras k_t. La récompense y_{t,k_t} du bras choisi par le joueur, et uniquement celle-ci, est dévoilée.

4.3 Neural Bandit

4.3.1 Réseaux de neurones multi-couches

Architecture

Un réseau de neurones est l'association, sous forme de graphe orienté, de plusieurs neurones formels. Le neurone formel le plus connu est le Perceptron [44]. Un

réseau de neurones multi-couches est composé de couches successives de neurones. Les neurones de chaque couche ne sont pas reliés entre eux et prennent en entrées les sorties de la couche précédente. Les neurones de la première couche recopient en sortie la valeur d'une variable de l'exemple présenté au réseau et les sorties des neurones de la dernière couche contiennent la prédiction. Les couches intermédiaires sont appelées couches cachées. Si la fonction d'activation des neurones est non-linéaire alors la succession de couches permet de construire un modèle non-linéaire pouvant faire office d'approximateurs universels [50].

Les réseaux de neurones bénéficient actuellement d'un regain d'intérêt grâce aux architectures profondes pouvant être composées de plusieurs millions de connections [51]. Ces architectures permettent de résoudre des problèmes auparavant trop complexes pour la capacité d'apprentissage des autres modèles notamment en traitement d'image, sur des jeux de données comportant plusieurs millions d'images associées à des milliers de labels [52] ou sur des jeux d'arcade [53] où l'apprentissage s'effectue à partir de la sortie vidéo du jeu. En mars 2016, AlphaGo [54] a battu Lee Sedol (9^{eme} dan professionnel, le rang maximum), sans handicap, à l'aide de réseaux de neurones profonds et d'algorithmes de Recherche Arborescente Monte-Carlo (MCTS).

4.3.2 Algorithme

Nous décrivons dans cette sous-section l'algorithme NeuralBandit (voir Algorithme 18). Étant dans un cas à information partielle, seule la récompense obtenue en jouant le bras k_t est disponible. Pour apprendre le meilleur bras à jouer, une première approche consisterait à effectuer une première phase d'exploration où chaque bras serait joué le même nombre de fois. Ainsi, l'estimateur obtenu ne serait pas biaisé sur les bras les plus joués. Cependant, une telle approche aurait des performances catastrophiques en cas de non-stationnarité des données. Notre approche consiste à garder un facteur d'exploration γ constant au cours du temps, permettant de poursuivre la mise à jour des modèles en cas de non-stationnarité des données. La probabilité de jouer le bras k au tour t en sachant que $\hat{k}_t$ est le bras possédant la plus haute prédiction de récompense est :

$$\mathbf{P}_t(k) = (1-\gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$$

Soit K le nombre de bras, C le nombre de neurones composant les couches cachées des réseaux et $\mathbf{N}_t^k : X \to Y$ la fonction associant un contexte x_t à la sortie du réseau de neurones correspondant au bras k au tour t. Le nombre de connexions pour chaque réseau est noté N avec $N = dim(X)C + C$. Pour faciliter les notations, nous plaçons l'ensemble des connexions dans la matrice W_t de taille $K \times N$, ainsi chaque ligne de la matrice W_t contient les poids d'un réseau.

Algorithme de rétro-propagation du gradient

L'algorithme de rétro-propagation du gradient [55] permet de calculer le gradient de l'erreur pour chaque neurone, de la dernière couche vers la première en minimisant une fonction de coût. Les algorithmes de descente de gradient peuvent être utilisés lorsque la fonction d'activation est dérivable. Ceux-ci sont naturellement utilisables en ligne et offrent de bonnes performance en généralisation [56].

Nous notons Δ_t la matrice de taille $K \times N$ contenant les modifications de chaque poids entre le tour t et le tour $t+1$. L'équation de mise à jour des poids est

$$W_{t+1} = W_t + \Delta_t$$

Soit λ le pas d'apprentissage, $\hat{x}_t^{n,k}$ la valeur de l'entrée associée à la connexion n du réseau k et $\delta_t^{n,k}$ le gradient de la fonction d'erreur au tour t pour le neurone possédant la connexion n du réseau k. $\Delta_t^{n,k}$ est la valeur correspondant aux coordonnées (n,k) de la matrice Δ_t. Lorsque la récompense d'un bras est connue, on peut calculer :

$$\Delta_t^{n,k} = \lambda \hat{x}_t^{n,k} \delta_t^{n,k}$$

Nous proposons une modification de la règle d'apprentissage permettant de tenir compte de cette probabilité de tirage pour débiaiser l'apprentissage :

$$\tilde{\Delta}_t^{n,k} = \frac{\lambda \hat{x}_t^{n,k} \delta_t^{n,k} \mathbf{1}[\hat{k}_t = k]}{\mathbf{P}_t(k)}$$

Proposition 1. *L'espérance de $\tilde{\Delta}_t^{n,k}$ est égale à $\Delta_t^{n,k}$.*

Démonstration.

$$\begin{aligned} \mathbf{E}[\tilde{\Delta}_t^{n,k}] &= \sum_{k=1}^{K} \mathbf{P}_t(k) \left(\frac{\lambda \hat{x}_t^{n,k} \delta_t^{n,k} \mathbf{1}[\hat{k}_t = k]}{\mathbf{P}_t(k)}\right) \\ &= \lambda \hat{x}_t^{n,k} \delta_t^{n,k} \\ &= \Delta_t^{n,k} \end{aligned}$$

□

La nouvelle équation de mise à jour des poids est :

$$W_{t+1} = W_t + \tilde{\Delta}_t \tag{4.1}$$

Algorithme 18 : NEURALBANDIT

Données : $\gamma \in [0, 0.5]$ et $\lambda \in]0, 1]$
Initialisation de $W_1 \in]-0.5, 0.5[^{N \times K}$;
pour $t = 1, 2, ..., T$ **faire**
  Le contexte x_t est dévoilé;
  $\hat{k}_t = \arg\max k \in [K] \mathbf{N}_t^k(x_t)$;
  $\forall k \in [K]$ on a $\mathbf{P}_t(k) = (1-\gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$;
  On tire $\tilde{k}_t$ suivant $\mathbf{P}_t$;
  On prédit $\tilde{k}_t$ et on reçoit la récompense $y_{t,\tilde{k}_t}$;
  On définit $\tilde{\Delta}_t$ tel que $\tilde{\Delta}_t^{n,k} = \frac{\lambda \hat{x}_t^{n,k} \delta_t^{n,k} \mathbf{1}[k_t = k]}{\mathbf{P}_t(k)}$;
  $W_{t+1} = W_t + \tilde{\Delta}_t$;

La descente de gradient contre la non-stationnarité

La descente de gradient permet de minimiser la minimisation d'une fonction de coût en faisant varier les poids associés aux connexions du réseau. Cette optimisation utilise comme point de départ une matrice de poids tirée aléatoirement lors de l'initialisation de l'algorithme. Considérons une exécution de l'algorithme sur un problème dont les contextes sont tirés depuis D_x^1 et les récompense depuis $D_{y|x}^1$ de l'initialisation de l'algorithme jusqu'au pas de temps t. Supposons maintenant qu'après $t+1$, ces distributions deviennent D_x^2 et $D_{y|x}^2$. Il n'est pas déraisonnable de considérer que continuer la descente de gradient à partir du modèle déjà appris est similaire à débuter l'apprentissage d'un nouveau modèle dont les poids auraient été initialisés aux mêmes valeurs que les poids précédents.

Cette approche comporte cependant quelques limites :

- Afin de faciliter l'apprentissage, le support de la distribution initialisant les poids est souvent borné. Par exemple, dans l'algorithme proposé, les poids sont initialisés avec des valeurs entre $]-0.5, 0.5[$.
- Au cours de l'apprentissage, des poids d'une grande amplitude peuvent saturer la fonction d'activation des neurones. Dans ce cas, le gradient sera quasi-nul et chacune des itérations de la descente de gradient ne modifiera que très peu les poids.

Dans le cas où un tel comportement serait observé, ajouter un paramètre de régularisation à l'équation de mise à jour pourrait permettre de relancer la descente de gradient.

Récapitulatif

Procédons à un récapitulatif du fonctionnement de notre algorithme :

- Chaque bras k est associé à un réseau.

- Lorsqu'un contexte x_t est présenté à l'algorithme le score $\mathbf{N}_t^k(x_t)$ de chaque bras est calculé.
- Une exploration des bras peut avoir lieu avec une probabilité γ.
- S'il y a exploration, le bras à jouer est tiré uniformément dans l'ensemble des bras, sinon le bras obtenant le plus haut score est joué.
- Après avoir reçu la récompense, les poids du réseau correspondant au bras joué sont mis à jour.

L'algorithme proposé, NEURALBANDIT, peut s'adapter à des non-stationnarités en continuant à apprendre au cours du temps, tout en obtenant en moyenne, en régime stationnaire, le même résultat que si nous avions appris les poids dans une première phase d'exploration, où chaque bras serait joué le même nombre de fois.

4.4 Sélection d'architectures

Malgré leur prédisposition à l'apprentissage en ligne, les réseaux de neurones sont surtout utilisés hors-ligne à cause de leur difficulté de paramétrage. Les performances des réseaux de neurones dépendent de plusieurs paramètres, comme le pas d'apprentissage, le nombre de couches cachées, leur taille ou les valeurs d'initialisation des poids. Un modèle est le prédicteur initialisé avec ces paramètres. Lors des problèmes d'apprentissage hors-ligne, la sélection de modèle est réalisée en utilisant un ensemble de validation [57]. Dans le cas de l'apprentissage en ligne, les modèles sont entrainés en parallèle sur le flux de données. Pour sélectionner les meilleures architectures, nous proposons d'utiliser l'algorithme de bandits adverse Exp3 [30, 31].

4.4.1 NEURALBANDIT.A

L'idée est d'instancier un algorithme de bandits non-stochastiques qui cherchera, parmi les modèles générés à partir des paramètres proposés par l'utilisateur, le plus performant (voir Algorithme 19). L'algorithme prend en paramètre une liste contenant M paramétrages et un paramètre d'exploration de modèle γ_{model}. Pour chaque élément de la liste, une instance de NEURALBANDIT est initialisée et représente un bras d'un algorithme EXP3. C'est celui-ci qui définira quel modèle choisira le bras à jouer à chaque tour. La distribution de probabilités utilisée pour tirer l'expert qui choisira le bras à jouer au temps t est notée $\mathbf{P}_{\text{model}}(t)$. Après avoir obtenu la récompense, les réseaux de neurones correspondant au bras joué seront mis à jour pour chaque modèle, ainsi que l'EXP3. Nous appelons *sélection d'experts apprenants*, la tâche de sélection en ligne de modèle en cours d'apprentissage.

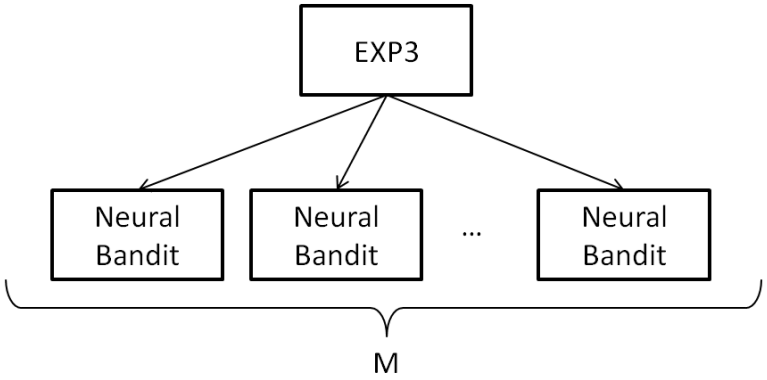


FIGURE 4.1 – **NeuralBandit.A.** Chacune des M instances de NeuralBandit correspond à un ensemble de paramètres. L'algorithme de bandit EXP3 sélectionne l'une de ces instances et celle-ci choisit ensuite le bras à jouer.

Cette utilisation d'EXP3 n'est pas conforme aux hypothèses utilisées dans son analyse, les récompenses n'étant pas choisies par un adversaire au début du jeu mais dépendantes des choix du méta-bandit. En effet, on peut imaginer une instance de NeuralBandit entourée d'un grand nombre d'instances d'un algorithme déterministe choisissant toujours un bras identique k'. Si le méta-bandit sélectionne l'instance de NeuralBandit à chaque tour de jeu, alors cette instance aura un contrôle total sur les bras joués et ainsi pourra explorer l'espace des contextes et des bras à sa convenance. Au contraire, si le méta-bandit sélectionne les modèles de manière uniforme alors, k' sera joué durant une grande partie des tours de jeux, inhibant l'apprentissage de l'instance de NeuralBandit. Le chapitre 6 est consacré à l'analyse formelle de la sélection d'experts apprenants et EXP3 y est analysé dans ce nouveau cadre.

Algorithme 19 : NeuralBandit.A

Données : $\gamma_{\text{model}} \in [0, 0.5]$ et un ensemble de M paramétrage de modèles
début
  Initialisation des M NeuralBandit
  Initialisation du vecteur de poids w_0 d'EXP3 avec $\forall m \in [M]\ w_0^m = 1$
  pour $t = 1, 2, ..., T$ **faire**
    Le contexte x_t est dévoilé
    m_t est tiré selon $\mathbf{P}_{\text{model}}(t)$ (1.15)
    Le modèle m_t choisit le bras $\tilde{k}_t$
    $\tilde{k}_t$ est prédit et la récompense $y_{t,\tilde{k}_t}$ est reçue
    Mise à jour des réseaux correspondant au bras $\tilde{k}_t$ pour chaque modèle avec (4.1)
    Mise à jour des vecteurs de poids des EXP3 avec (1.14)

4.4.2 NEURALBANDIT.B

L'algorithme NEURALBANDIT.A cherche l'instance de NEURALBANDIT dont le paramétrage, parmi ceux proposés, permet d'obtenir la plus haute récompense cumulée. Chacune des instances est considéré comme une boite noire par le méta-bandit. On peut cependant supposer que pour certains bras les meilleurs paramétrages pourraient être différents. L'algorithme NEURALBANDIT.B permet aux bras d'être associés à des modèles différents. Au lieu de chercher à sélectionner la meilleure instance d'un algorithme de bandit contextuel, le méta-bandit utilisé dans NEURALBANDIT.B cherche à identifier les modèles de régression offrant les meilleures performances lorsqu'ils sont combinés à l'intérieur de $\arg\max_{k\in[K]} s_t^k$.

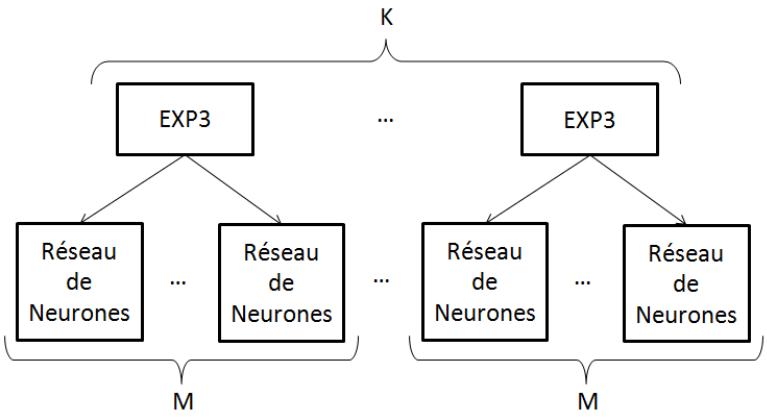


FIGURE 4.2 – **NeuralBandit.B.** Pour chacun des K bras, un réseau de neurone est instancié pour chacun des M paramétrages. Une instance d'EXP3 est initialisée pour chaque ensemble de M paramétrages. Chacune a pour tâche la sélection du réseau qui prédira la récompense du bras associé. Après la sélection des réseaux, le bras ayant la plus haute prédiction est joué avec une probabilité $(1-\gamma)$. Avec une probabilité γ le bras joué est choisi aléatoirement via une distribution uniforme.

S'il est possible d'énumérer toutes les combinaisons possibles d'architectures afin de réduire ce problème en un problème résoluble par NEURALBANDIT.A, avec M architecture différentes par bras, cela reviendrait à considérer M^K configurations différentes. Afin de simplifier le problème, nous choisissons de considérer un problème d'optimisation local à chacun des K bras. Chaque méta-bandit devra ainsi sélectionner l'architecture à utiliser pour prédire la probabilité de gain du bras associé sans avoir conscience des choix et des performances observées par les autres méta-bandits.

NEURALBANDIT.B possède une plus grande capacité d'expression que NEURALBANDIT.A en permettant d'associer des modèles différents à chaque bras. NEURALBANDIT.A. Sous cette formalisation, le problème de sélection de la meilleure architecture est cependant non convexe et les optimisations locales pourraient amener l'algorithme à converger vers un minimum local.

Algorithme 20 : NeuralBandit.B

Données : $\gamma \in [0, 0.5]$, $\gamma_{\text{model}} \in [0, 0.5]$ et un ensemble de M modèles
début
  Initialisation de K réseaux de neurones pour chaque modèle m
  Initialisation de K EXP3
  pour $t = 1, 2, ..., T$ **faire**
    Le contexte x_t est dévoilé
    pour $k = 1, 2, ..., K$ **faire**
      m_t^k est tiré suivant $\mathbf{P}_{\text{model}^k(t)}$ (1.15)
      Le bras k reçoit le score $s_t^k = \mathbf{N}_t^{m_t^k, k}(x_t)$
    $\hat{k}_t = \arg\max_{k \in [K]} s_t^k$
    $\forall k \in [K]$ on a $\mathbf{P}_t(k) = (1 - \gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$
    $\tilde{k}_t$ est tiré suivant $\mathbf{P}_t$
    $\tilde{k}_t$ est prédit et la récompense $y_{t,\tilde{k}_t}$ est reçue
    Mise à jour des réseaux correspondant au bras $\tilde{k}_t$ pour chaque modèle avec (4.1)
    Mise à jour des vecteurs de poids de chaque l'EXP3 avec (1.14)

4.5 Simulations numériques

L'évaluation des algorithmes de bandits contextuels est une tâche compliquée du fait de l'interactivité de la tâche. L'utilisation en conditions réelles d'un nouvel algorithme à des fins de tests n'est souvent pas envisageable et ne pourra être possible que lorsque celui-ci aura été testé et éprouvé. Il est donc nécessaire de pouvoir procéder à une évaluation hors-ligne de l'algorithme à partir de données précédemment collectées.

Nous avons choisi d'utiliser des jeux de données de classification supervisée en fournissant une récompense de 1 à l'algorithme s'il a prédit la bonne classe et 0 s'il a prédit la mauvaise classe. Les jeux de données doivent cependant être de grande taille et la nature du problème d'apprentissage peut-être très éloignée de ceux habituellement rencontrés dans le cadre des problèmes de bandits. Nous utilisons trois jeux de données provenant de l'*UCI Machine Learning Repository* [58]. Les pré-traitements des jeux de données consistent en :

- Transformer les variables continues en variables catégorielles en les divisant leur plage de valeur suivant leurs quintiles.
- Binariser les variables catégorielles.

Le flux de contextes est obtenu en bouclant sur les jeux de données après les avoir mélangées. Le point de départ dans le jeu de données est tiré aléatoirement au début de chaque exécution d'un algorithme. Un bruit de 0.05 par variable est ajouté aux contextes observés, i.e. une variable à 0 peut passer à 1 avec une

probabilité de 0.05 et inversement. Les regrets cumulés sont calculés face à un oracle prédisant la bonne classe à chaque itération (nombres d'itérations - gain cumulé)

4.5.1 Description des jeux de données

- **Forest Cover Type.** Le jeu de donnée Forest Cover Type associe des types de végétations à des données cartographiques. Il est initialement composé de 54 variables entières ou catégorielles qui, après pré-traitement, sont transformées en 94 variables binaires. Il contient 581,012 contextes associés à 7 classes.
- **Adult.** Le jeu de donnée Adult associe des catégories professionnelles à des données de recensements. Il contient 14 variables entières ou catégorielles, qui après pré-traitement, sont transformées en 82 variables binaires. Il contient 48,842 contextes associés à 14 classes.
- **Census1990.** Le jeu de donnée Census1990 est composé de données de recensements. Il contient 68 variables catégorielles qui, après pré-traitement, sont transformées en 255 variables binaires. Il contient 2,458,286 contextes associés à 18 classes (la variable *Yearsearch*).

4.5.2 Étude de sensibilité et sélection d'architectures

Dans cette sous-section, nous présentons les différents paramétrages utilisés par NEURALBANDIT et ses variantes et étudions l'influence des différentes tailles de couches cachées.

- **NeuralBandit** Le paramètre d'exploration est $\gamma = 0.05$. Les tailles des couches cachées varient suivant les exécutions et sont explicitées dans les résultats.
- **NeuralBandit.A et NeuralBandit.B.** Les réseaux de neurones sont initialisés avec les combinaisons de paramètres suivantes :
  - La fonction d'activation utilisée, une sigmoïde.
  - Les tailles de couches cachées $C = \{1, 5, 5, 10, 10, 25, 25, 50, 50, 100\}$.
  - Les pas d'apprentissage $\lambda \in \{0.1, 0.01\}$.

NeuralBandit.A obtient un regret cumulé similaire à celui de la meilleure architecture, montrant l'efficacité de cette méthode de sélection. **NeuralBandit.B** obtient cependant des performances supérieures, certainement due à la plus grande expressivité conférée par les multiples méta-bandits (voir Figures 4.3 et 4.4).

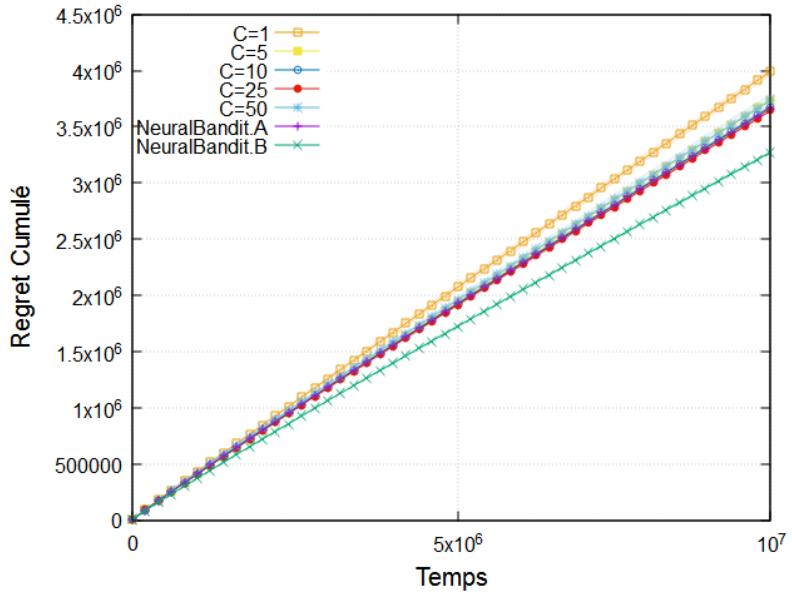


FIGURE 4.3 – **Forest Cover Type.** Cette figure présente les regrets cumulés de NEURALBANDIT pour différents paramétrages C de la couche cachée.

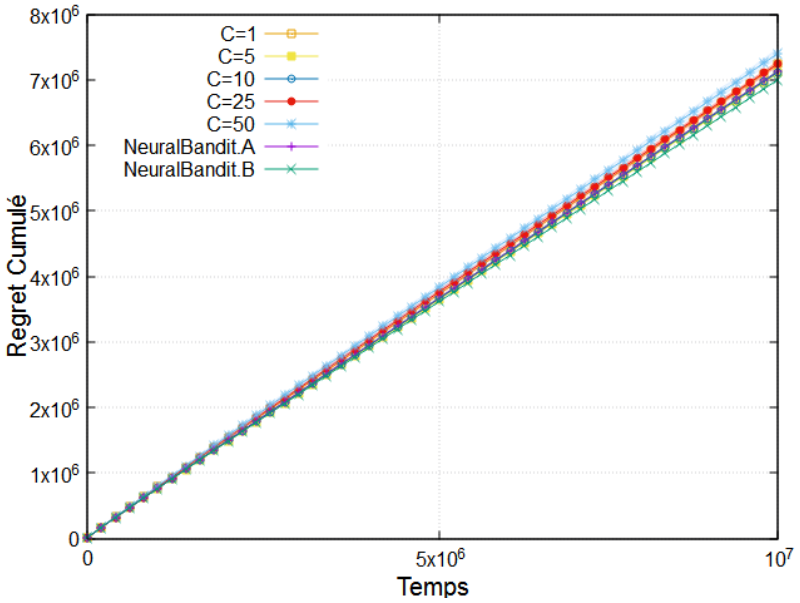


FIGURE 4.4 – **Adult.** Cette figure présente les regrets cumulés de NEURALBANDIT pour différents paramétrages C de la couche cachée.

4.5.3 Étude de sensibilité aux changements de stationnarité

Afin d'observer le comportement de NEURALBANDIT face aux changements de stationnarité, nous introduisons une rotation des classes toutes les 2×10^6 d'itérations ($1 \rightarrow 2, ..., K \rightarrow 1$). À titre de comparaison, nous affichons les même

indicateurs pour LINUCB dont le paramétrage est décrit dans la sous-section suivante.

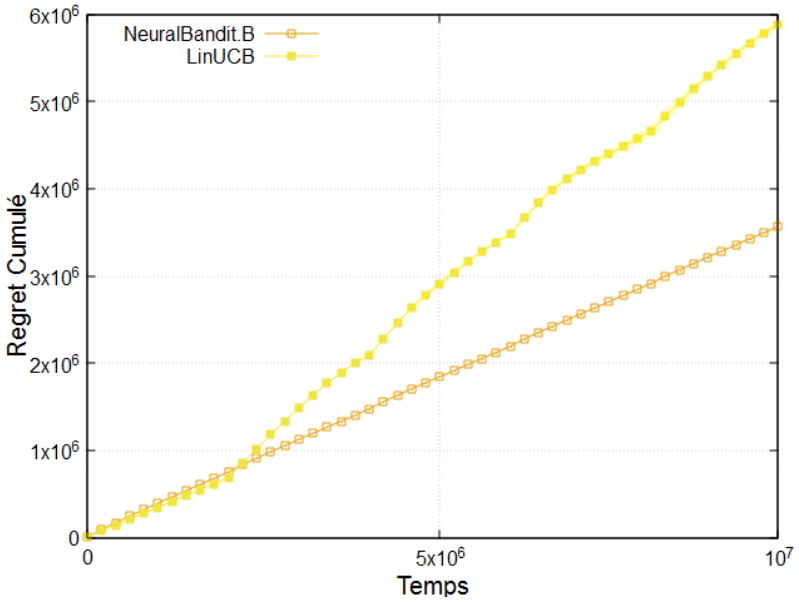


FIGURE 4.5 – **Forest Cover Type.** Cette figure présente les regrets cumulés de NEURALBANDIT.B en présence de changements de stationnarité.

Le modèle réagit très rapidement aux changements de stationnarité. La convergence du modèle vers la nouvelle distribution peut-être observée en figure 4.6.

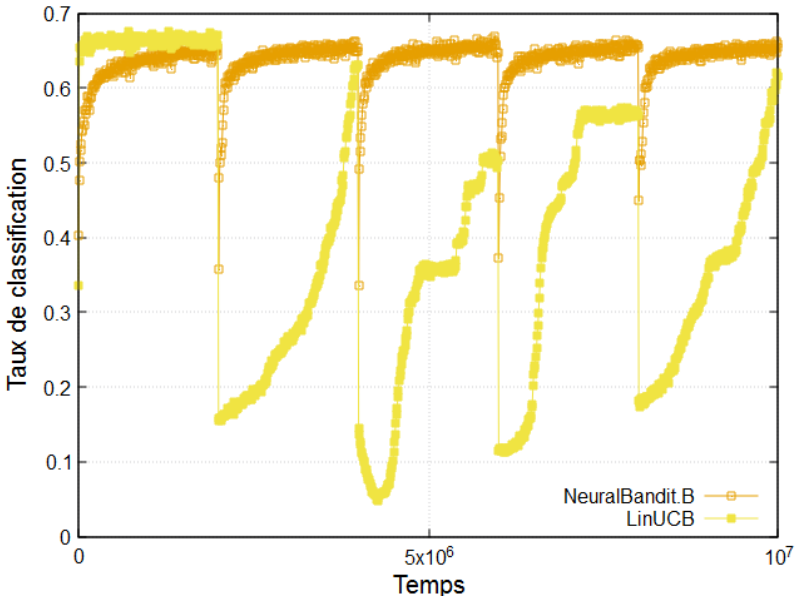


FIGURE 4.6 – **Forest Cover Type.** Le taux de classification instantané de NEURALBANDIT.B en présence de changements de stationnarité. Chaque point représente le taux de classification moyen sur une fenêtre de 10000 exemples.

LinUCB résout quant à lui le système linéaire lui permettant d'estimer le paramètre θ en utilisant toutes les observations recueillies depuis le début du jeu, ce qui biaise ses estimateurs vers l'ancienne distribution après un changement de stationnarité.

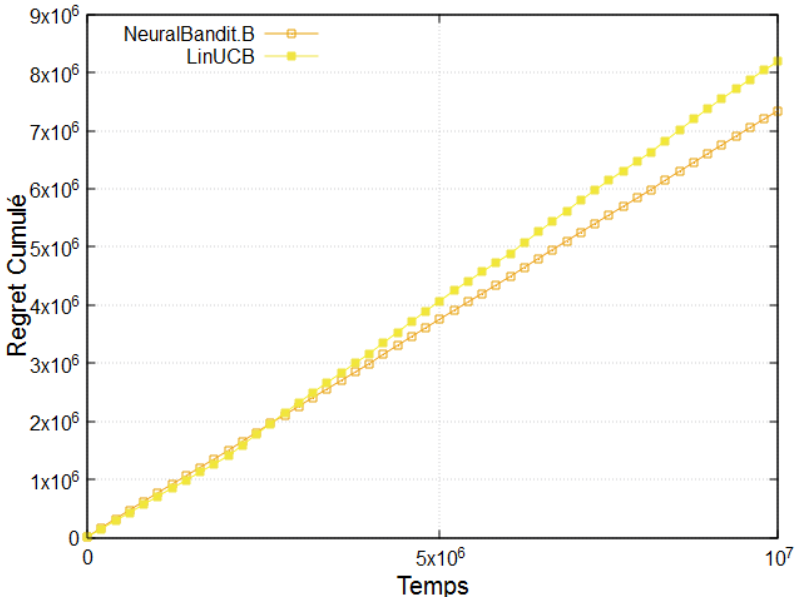


Figure 4.7 – **Adult.** Cette figure présente les regrets cumulés de NeuralBandit.B en présence de changements de stationnarité.

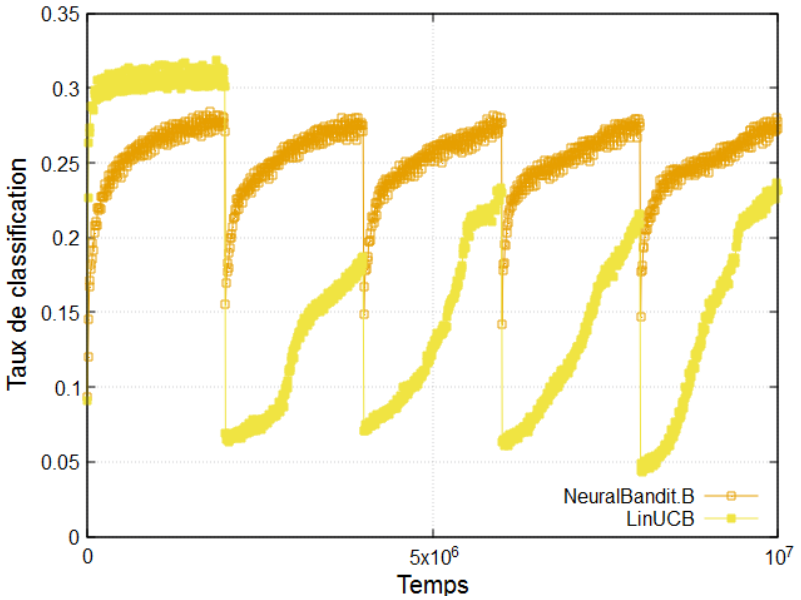


Figure 4.8 – **Adult.** Le taux de classification instantané de NeuralBandit.B en présence de changements de stationnarité. Chaque point représente le taux de classification moyen sur une fenêtre de 10000 exemples.

Ces comportements se répètent sur les autres jeux de données, comme observé

en figure 4.7 et 4.8.

4.5.4 Comparaison avec BANDITRON et LINUCB

Paramétrages des algorithmes

- **Banditron.** Le paramètre d'exploration est $\gamma = 0.05$.
- **LinUCB.** Tel que présenté dans [41], LINUCB est uniquement composé d'un seul modèle linéaire permettant de choisir le meilleur contexte parmi plusieurs. Dans ces simulations, un seul contexte est présenté à chaque tour, avec une récompense différente par bras. Afin de permettre à LINUCB d'être utilisé dans ce contexte, nous utilisons la version de LINUCB avec modèles linéaires disjoints [41].

  L'inversion de matrice étant couteuse, celle-ci n'est effectuée que toutes les 10000 itérations. Même avec ce réglage, une exécution complète de l'algorithme sur le jeu de données Census nécessite environs une journée de calculs.

Résultats

Un récapitulatif des expériences de cette sous-section est présenté dans la Figure 4.1. En observant les courbes de regrets, on peut observer que les algorithmes se séparent en deux groupes. Les algorithmes LINUCB et NEURALBANDIT ont des performances similaires, leur plus grand écart en classification étant de 4%. D'un autre côté, BANDITRON accuse de faibles performances, ses taux de classification étant inférieurs aux autres de 7.7% à 15%. Ces faibles performances peuvent être expliquées par la règle de mise à jour des PERCEPTRON internes à BANDITRONS, règle supposant la séparabilité linéaire des contextes en fonction de leur classe. En effet, bien que reposant lui aussi sur des modèles linéaires, LINUCB obtient de bien meilleures performances.

La Table 4.2 montre les complexités algorithmiques des différents algorithmes. Bien qu'offrant de bonnes performances lorsqu'une dépendance linéaire entre les contextes et les récompenses existe, LINUCB possède une complexité algorithmique ayant une dépendance cubique vis-à-vis du nombre de variables de contextes. Cette dépendance rend inutilisable LINUCB sur des problèmes en grande dimension. Les algorithmes de la famille NEURALBANDIT possèdent quand à eux uniquement des dépendances linéaires. Bien que le nombre de connections ou d'architectures puissent être élevés, ces algorithmes peuvent-être facilement parallélisés.

TABLE 4.1 – Tableau récapitulatif des expériences. Les taux de classification sont calculés sur les 100000 derniers contextes. Le temps d'exécution moyen a été évalué sur un ordinateur personnel utilisant un processeur Intel Xeon 2, 80GHz et 6Go de RAM.

Algorithme	Regret	Taux de classification	Temps d'exécutions
Forest Cover Type, classe cible : Cover Type (7 classes)			
BANDITRON	$4.88\ 10^6\ \pm 10^5$	51.3%	10 min
LINUCB	$3.39\ 10^6\ \pm 10^3$	66.4%	360 min
NEURALBANDIT.B	$3.33\ 10^6\ \pm 2.10^4$	68.1%	150 min
Adult, classe cible : occupation (14 classes)			
BANDITRON	$7.89\ 10^6\ \pm 3.10^4$	21.9%	20 min
LINUCB	$6.91\ 10^6\ \pm 4.10^4$	31.6%	400 min
NEURALBANDIT.B	$7.13\ 10^6\ \pm 10^5$	29.6%	140 min
Census1990, classe cible : Yearsch (18 classes)			
BANDITRON	$6.97\ 10^6\ \pm 2.10^5$	30.7%	26 min
LINUCB	$5.5\ 10^6\ \pm 5.10^4$	45.7%	1080 min
NEURALBANDIT.B	$6.03\ 10^6\ \pm 10^5$	41.7%	300 min

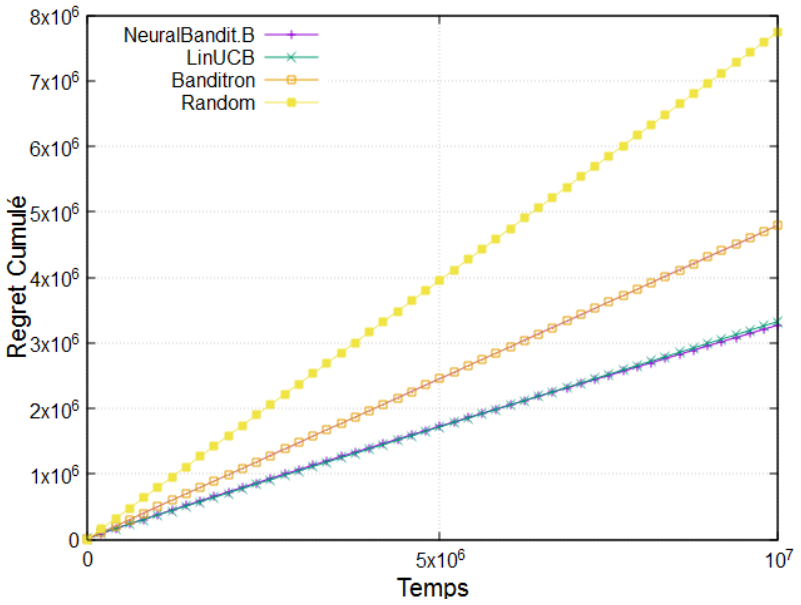


FIGURE 4.9 – **Forest Cover Type.** Les regrets cumulés de NEURALBANDIT2, FORÊT DE BANDITS, LINUCB et BANDITRON.

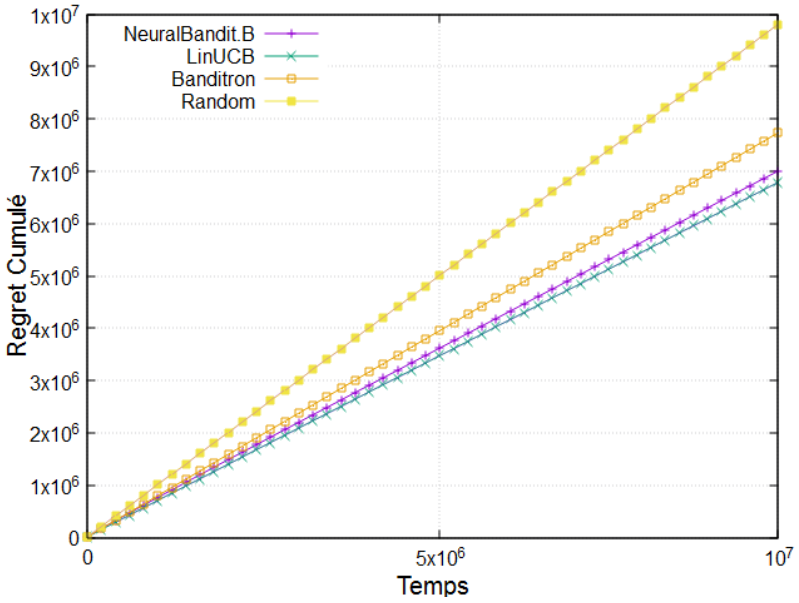


FIGURE 4.10 – **Adult.** Les regrets cumulés de NEURALBANDIT2, FORÊT DE BANDITS, LINUCB et BANDITRON.

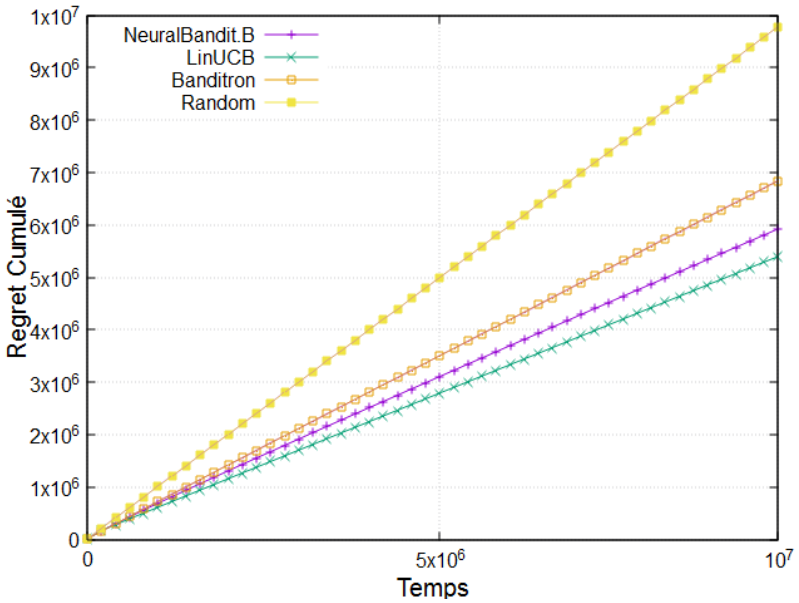


FIGURE 4.11 – **Census1990.** Les regrets cumulés de NEURALBANDIT2, FORÊT DE BANDITS, LINUCB et BANDITRON.

TABLE 4.2 – Les complexité algorithmiques d'une itération (choix du bras à jouer puis mise à jour du modèle) des algorithmes de bandit contextuels. K est le nombre de bras,$dim(X)$ le nombre de variables de contexte, M le nombre d'architectures de NEURALBANDIT.A OU B et N le nombre de connections composant les réseaux de neurones utilisés par NEURALBANDIT.A OU B

Algorithme	Complexité algorithmique instantanée
BANDITRON	$O(Kdim(X))$
LINUCB	$O(Kdim(X)^3)$
NEURALBANDIT.A OU B	$O(MKdim(X) + MKN)$

4.6 Récapitulatif

Dans ce chapitre, nous avons présenté une méthodologie permettant l'utilisation de réseaux de neurones (NEURALBANDIT) pour traiter des problèmes de bandits contextuels. La non-stationnarité est gérée par l'algorithme de manière passive, la descente de gradient se poursuivant vers un nouveau minimum local si le paysage de la fonction de coût change. Nous avons ensuite proposé deux méthodes différentes (NEURALBANDIT.A et NEURALBANDIT.B) permettant de remplacer la validation croisée lorsque les algorithmes sont déployés en ligne.

Chapitre 5

Forêt de bandits pour les bandits contextuels

Contents

5.1 Introduction

Dans ce chapitre, nous proposons un algorithme de bandits contextuels basé sur les forêts d'arbres, composées d'arbres de décisions, permettant de modéliser des dépendances non-linéaires entre les contextes et les récompenses. Dans ce chapitre, les distributions D_x et $D_{y|x}$ sont stationnaires et n'évoluent pas au cours du temps. Nous montrons des garanties en termes de complexité d'échantillonnage similaires à celles obtenues par les algorithmes non-contextuels (à un facteur 2^D près, où D est la profondeur des arbres) en réduisant la construction de la forêt à de multiples problèmes de bandits non-contextuels ou de sélections de politiques uni-variées (voir exemple en sous-section 5.3.1).

Les arbres de décisions divisent de manière récursive l'espace des données à l'aide de tests sur les différentes variables. Chaque nœud de l'arbre effectue l'une

de ces coupures et les feuilles prédisent la classe. Trouver l'arbre de décision minimisant l'erreur sur l'ensemble d'apprentissage est en général un problème NP-complet. Pour contourner cette complexité, les arbres sont construits de manière gloutonne, en essayant de maximiser le bénéfice de la coupure niveau par niveau.

L'algorithme VFDT [59] permet de construire des arbres en ligne et utilise pour cela l'inégalité d'Hoeffding [15]. Cette borne permet de calculer un intervalle de confiance sur les sommes de variables aléatoires indépendantes. Lorsque cet intervalle devient suffisamment petit pour pouvoir décider quel est le meilleur critère avec une faible probabilité d'erreur alors la coupure est effectuée. La borne d'Hoeffding n'est cependant valable que pour les sommes de variables aléatoires et ne peut donc pas être utilisée dans le calcul de bornes sur le gain d'information, utilisé dans ID3 ou sur l'index de Gini, utilisé dans CART. Dans ce cas, les bornes peuvent être calculées en utilisant l'inégalité de McDiarmid [60]. Cet algorithme a ensuite été adapté pour prendre en compte la non-stationnarité dans les données avec l'algorithme CVFDT [61].

Un procédé similaire aux arbres de décision pour les bandits contextuels à été proposé avec l'algorithme ABSE (*Adaptively Binned Successive Elimination* en anglais) [62]. Dans celui-ci, plusieurs boites (*bin* en anglais) partitionnant différemment l'espace sont à la disposition du joueur. Le joueur peut alors choisir la boite permettant d'obtenir la plus haute récompense de manière gloutonne. De la même manière que les arbres de décision, ce procédé peut ensuite être répété récursivement.

Dans les sections suivantes, nous proposons un processus similaire à celui de l'algorithme VFDT, mais utilisant un critère basé sur la récompense moyenne obtenue à ce niveau de coupure et adaptée à l'information partielle du formalisme des bandits contextuels. La méthode utilisée pour construire les arbres est, de plus, étendue aux forêts d'arbres.

5.2 Formalisation

Nous reprenons le formalisme présenté en sous-section 3.2. Soit une séquence $((x_1, \mathbf{y}_1), ..., (x_T, \mathbf{y}_T))$ avec $x_t \in X$ un contexte, $k \in [K] = \{1, ..., K\}$ un des K bras pouvant être choisis et $\mathbf{y}_t \in Y$ un vecteur de récompense $(y_{t,1}, ..., y_{t,K})$ avec $y_{t,k} \in [0, 1]$ la récompense pour le bras k. L'ensemble des variables des contextes est noté V et contient C variables binaires. Soit $D_{x,y}$ la distribution générant les contextes et les récompenses. Le problème est répété sur T tours : à chaque tour $t < T$ le contexte x_t est annoncé. Le joueur, qui cherche à maximiser la somme de ses récompenses, choisit un bras k_t. La récompense y_{t,k_t} du bras choisi par le joueur, et uniquement celle-ci, est dévoilée.

5.3 L'arbre de décision de profondeur 1

Dans cette section nous décrivons l'arbre de décision minimal de profondeur 1 nécessaire à la construction d'un arbre de profondeur D. Celui-ci est composé d'un nœud de décision, comprenant l'index de la variable de coupure et du bras optimal à jouer pour chacune des valeurs de cette variable.

Nous définissons quelques notations. Nous notons $1 \leq i \leq C$ l'indice de la i^{eme} variable. La récompense moyenne du bras k sachant que la variable i prend la valeur $v \in \{0, 1\}$ est notée $\mu_k^i|v = \mathbb{E}_{D_y}[y_k[\![x_i = v]\!]]$.

5.3.1 Un exemple jouet

Nous illustrons le gain de la séparation de l'espace via les valeurs d'une variable par un exemple jouet. Soit k_1 et k_2 deux bras. Soit x_{i_1} et x_{i_2}, deux variables aléatoires binaires décrivant le contexte. Dans cet exemple, nous supposons que les variables contextuelles sont indépendantes les unes par rapport aux autres.

Les probabilités des contextes et des récompenses sont résumées dans la Table 5.1. Par exemple, $\mu_{k_2}^{i_1}|v$ est l'espérance conditionnelle de la récompense du bras k_2 sachant que $x_{i_1} = v$, et $P(x_{i_1} = v)$ est la probabilité que la variable contextuelle x_{i_1} prenne la valeur v .

TABLE 5.1 – La récompense moyenne des bras k_1 et k_2, connaissant chaque variable de contexte ainsi que la probabilité d'observer chaque valeur de variable.

	v_0	v_1
$\mu_{k_1}^{i_1}\|v$	0	1
$\mu_{k_2}^{i_1}\|v$	3/5	1/6
$P(x_{i_1} = v)$	5/8	3/8
$\mu_{k_1}^{i_2}\|v$	1/4	3/4
$\mu_{k_2}^{i_2}\|v$	9/24	5/8
$P(x_{i_2} = v)$	3/4	1/4

Nous comparons deux stratégies différentes. Le Joueur 1 n'observe pas les contextes et prend ses décisions uniquement sur la base des récompenses moyennes de chaque bras alors qu'un second joueur, Joueur 2 utilise la valeur de la variable x_{i_1} pour l'aider dans ses choix. La Table 5.1 indique que la meilleure stratégie pour le Joueur 1 serait de toujours jouer le bras k_2, avec une récompense moyenne de $\mu_{k_2} = 7/16$. Le Joueur 2 peut quand à lui adapter sa stratégie suivant le contexte : le meilleur choix sera de jouer k_2 quand $x_{i_1} = v_0$ et k_1 quand $x_{i_1} = v_1$. Sa récom-

pense moyenne sera de :

$$\begin{aligned}\mu^{i_1} &= P(x_{i_1} = v_0) \cdot \mu^{i_1}_{k_2}|v_0 + P(x_{i_1} = v_1) \cdot \mu^{i_1}_{k_1}|v_1 \\ &= \mu^{i_1}_{k_2,v_0} + \mu^{i_1}_{k_1,v_1} = 3/4 \ ,\end{aligned}$$

où $\mu^{i_1}_{k_2,v_0}$ et $\mu^{i_1}_{k_1,v_1}$ représentent respectivement, la récompense moyenne du bras k_2 quand $x_{i_1} = v_0$ et la récompense moyenne du bras k_1 quand $x_{i_1} = v_1$.

Qu'importe la récompense moyenne associée à chaque valeur de la variable, un joueur utilisant le contexte aura toujours l'avantage sur un joueur ne l'utilisant pas. En effet, nous avons :

$$\mu^i = \max_k \mu^i_{k,v_0} + \max_k \mu^i_{k,v_1} \geq \max_k \mu_k$$

Maintenant, si un troisième joueur utilise la variable contextuelle x_{i_2}, l'espérance de ses récompenses sera :

$$\mu^{i_2} = \mu^{i_2}_{k_2,v_0} + \mu^{i_2}_{k_1,v_1} = 15/32$$

Le Joueur 2 garde l'espérance de récompense la plus haute et x_{i_1} est la meilleure variable contextuelle pour décider quel bras jouer entre k_1 et k_2.

5.3.2 Sélection de la meilleure variable

Nous présentons l'algorithme SÉLECTION DE VARIABLES (voir Algorithme 21) dans cette sous-section. Cet algorithme est exécuté localement dans chaque nœud non-terminal afin de sélectionner la meilleure variable de coupure. L'ensemble [K] contenant la totalité des bras est exploré suivant un round-robin (les appels successifs de la fonction $RR([K])$) retournant chaque élément de $[K]$ suivant un round-robin). Nous détaillons maintenant le processus d'élimination des variables.

Soit V l'ensemble des variables de contexte. Soit $\mu^i_{k,v} = P(x_i = v)\mu^i_k|v$.

$$\mu^i = \left(\max_k \mu^i_{k,0} + \max_k \mu^i_{k,1}\right) \ .$$

L'arbre de décision de profondeur 1 optimal est celui maximisant la récompense moyenne μ^*.

$$\mu^* = \max_i \mu^i \ .$$

La variable $i^* = \arg\max_i \mu^i$ est la variable de coupure optimale.

L'algorithme VE (voir Algorithme 22) élimine une variable i', empiriquement sous-optimale, si il existe i tel que :

$$\hat{\mu}^i - \hat{\mu}^{i'} + \epsilon \geq 4\sqrt{\frac{1}{2t_k} \log \frac{4KCt_k^2}{\delta}} \ . \tag{5.1}$$

Algorithme 21 : Sélection de variables

début
- $\forall k\, t_k = 0$, $S = V$, $\forall (i,k,v)\, \mu^i_{k,v} = 0$, $\forall i \hat{\mu}^i = 0$
- **répéter**
  - Observer le vecteur de contexte $\mathbf{x_t}$
  - Jouer le bras $k = RR([K])$
  - Observer la récompense $y_k(t)$
  - $t_k = t_k + 1$
  - $S = \mathrm{VE}(t_k, k, \mathbf{x_t}, y_k(t), S, [K])$
  - $t = t + 1$
- **jusqu'à** $|S| = 1$

Algorithme 22 : VE

Données : $t \in \mathbb{N}^+$, $K \in [K]$, $x_t \in \{0,1\}^C$, S, $[K]$

début
- **pour chaque** $i \in S$ **faire**
  - **pour chaque** $v \in \{0,1\}$ **faire**
    - $\hat{\mu}^i_{k,v} = \frac{y_k}{t} [\![v = x_i]\!] + \frac{t-1}{t} \hat{\mu}^i_{k,v}$
  - $\hat{\mu}^i = \sum_{v \in \{0,1\}} \max_k \hat{\mu}^i_{k,v}$
- **si** $k = K$ **alors**
  - Retirer de S les variables sous-optimales suivant l'équation (5.1) ou (5.8)
- **retourner** S

Nous présentons maintenant deux lemmes bornant la complexité d'échantillonnage de la tâche de sélection de variables. Le Lemme 3 est une borne inférieure et le Lemme 2 borne supérieurement la complexité d'échantillonnage de notre algorithme. Il est intéressant de remarquer que la complexité d'échantillonnage a une faible dépendance vis-à-vis du nombre C de variables. La dépendance logarithmique en C vient de la technique de preuve utilisée, qui est basé sur l'hypothèse d'indépendance entre les échecs des intervalles de confiance correspondant à chaque variable. Dans le cas où les validités des intervalles de confiances associés aux variables sont dépendantes, alors cette dépendance en C peut totalement disparaitre (par exemple dans le cas pathologique d'un vecteur de contexte contenant une seule variable dupliquée C fois).

Lemme 2. *Pour $K \geq 2$, $C \geq 2$ et $\epsilon = 0$, la complexité d'échantillonnage de* Sélection de variable *nécessaire à la sélection de la variable optimale avec une probabilité d'au moins $1 - \delta$ est bornée supérieurement par :*

$$O\left(\frac{K}{\Delta^2} \log \frac{KC}{\delta \Delta}\right), \tag{5.2}$$

avec $\Delta = \min_{i \in V} \mu^* - \mu^i$.

La preuve du Lemme 2 est disponible en sous-section 5.7.

Lemme 3. *Il existe une distribution $D_{x,y}$ telle que tout algorithme cherchant la variable optimale a une complexité d'échantillonnage d'au moins :*

$$\Omega\left(\frac{K}{\Delta_1^2}\log\frac{1}{\delta}\right). \tag{5.3}$$

La preuve du Lemme 3 est disponible en sous-section 5.7.

5.3.3 Sélection des meilleures actions

Pour terminer la création de l'arbre de profondeur 1, nous devons maintenant identifier le meilleur bras pour chaque valeur de la variable. N'importe quel algorithme de bandits manchots (UCB [17] par exemple) pourrait être utilisé pour contrôler le compromis entre l'exploration et l'exploitation dans les feuilles de l'arbre. Afin de pouvoir continuer l'analyse en complexité d'échantillonnage, nous choisissons d'utiliser un algorithme d'éliminations successives [14].

Le bras k est éliminé du nœud correspondant à la valeur v de la variable i^* quand il existe $k' \neq k$ tel que :

$$\hat{\mu}_{k'} - \hat{\mu}_k + \epsilon \geq 4\sqrt{\frac{1}{2t_k}\log\frac{4KCt_k^2}{\delta}}. \tag{5.4}$$

Algorithme 23 : Sélection d'action

Données : $\delta \in (0,1]$, $\epsilon \in [0,1)$
Résultat : une ϵ-approximation du meilleur bras
$\forall k\, t_k = 0$, $S = [K]$;
répéter
  Observer le vecteur de contexte $\mathbf{x_t}$;
  Jouer le bras $k = RR(S)$;
  Observer la récompense $y_k(t)$;
  $t_k = t_k + 1$;
  $S = \mathrm{AE}(t_k, k, \mathbf{x_t}, S)$;
  $t = t + 1$;
jusqu'à $|S| = 1$;

Lemme 4. *Pour $K \geq 2$, $C \geq 2$ et $\epsilon = 0$, la complexité d'échantillonnage de* Sélection d'action *nécessaire à la sélection de la variable optimale avec une probabilité d'au moins $1-\delta$ est bornée supérieurement par :*

$$O\left(\frac{K}{\Delta^2}\log\frac{KC}{\delta\Delta}\right), \tag{5.5}$$

avec $\Delta = \min_{k\in[K]} \mu^* - \mu_k$.

Algorithme 24 : AE

Données : $t \in \mathbb{N}^+$, $K \in [K]$, $x_t \in \{0,1\}^C$, S, $[K]$
$S_1 = S$, $\forall m$, $\hat{\mu}_k(0) = 0$, $t = 1$;
Jouer le bras $k = RR(S)$;
$\hat{\mu}_k(t) = \frac{t-1}{t}\hat{\mu}_k(t-1) + \frac{y_k(t)}{t}$;
si $k =$ DERNIER(S_t) ; `// La fonction` DERNIER `revoie le dernier élément de l'ensemble` S_t
alors
  Retirer de S_t tout les bras k suivant l'équation (5.4) ou (5.9);
retourner S;

Démonstration. Ce lemme est une reformulation des garanties en complexité d'échantillonnage de l'algorithme SUCCESSIVE ELIMINATION [14]. □

5.3.4 Construction de l'arbre de décision de profondeur 1

L'algorithme SÉLECTION DE VARIABLE est exécuté de manière à trouver la meilleure variable. Pour chaque feuille, correspondant à chaque modalité de la variable de coupure, l'algorithme SÉLECTION D'ACTION est exécuté. Nous notons k^* le meilleur bras, après coupure sur la variable i^*

Théorème 14. *Pour $K \geq 2$, $C \geq 2$ et $\epsilon = 0$, la complexité d'échantillonnage de* SÉLECTION DE VARIABLE *nécessaire à la sélection de la variable optimale avec une probabilité d'au moins $1 - 3\delta$ est bornée supérieurement par :*

$$O\left(\frac{K}{\Delta_0^2}\log\frac{KC}{\delta\Delta_0} + \frac{K}{\Delta_1^2}\log\frac{K}{\delta\Delta_1}\right) \tag{5.6}$$

avec $\Delta_0 = \min_{i\in V} \mu^* - \mu^i$ *et* $\Delta_1 = \min_{k\in[K],v\in\{0,1\}} \mu_{k^*}^{i^*} - \mu_k^{i^*}$.

Démonstration. Ce résultat est une application directe des Lemmes 2 et 4 ainsi que de l'inégalité de Boole. □

5.4 Forêt de bandits

L'arbre de décision de profondeur 1 construit dans la section précédente est un classifieur univarié. Combiner davantage de variables dans un arbre permet d'augmenter la capacité du classifieur, mais à l'inconvénient d'être un problème ayant une très forte combinatoire. L'approche usuelle utilisée pour combattre cette combinatoire est de construire les arbres de manière gloutonne. La première variable est sélectionnée et l'espace des contextes est divisé suivant les modalités de cette variable. La tâche de sélection de variable est répétée de manière récursive dans les

sous-espaces obtenus sur les variables n'ayant pas encore été utilisées dans les niveaux supérieurs jusqu'à obtenir la granularité désirée. L'arbre glouton est ensuite finalisé en instanciant des tâches de sélection d'actions au niveau le plus fin.

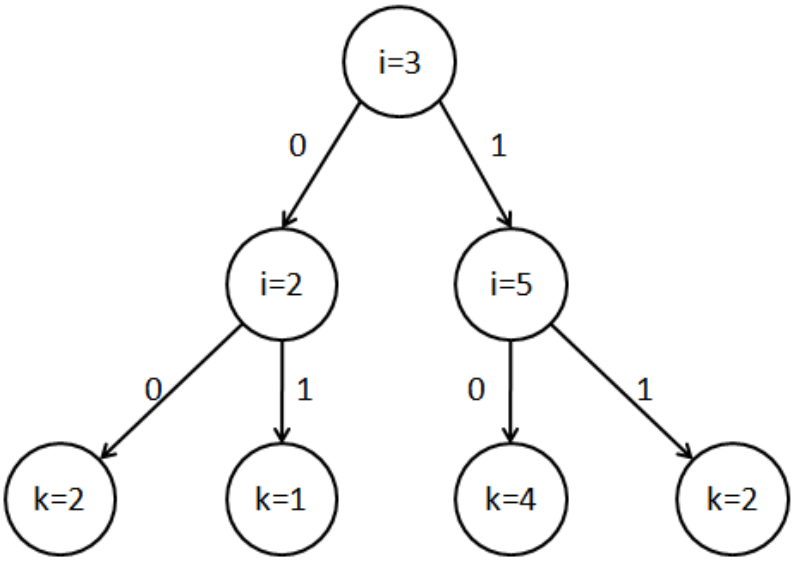


FIGURE 5.1 – Un exemple d'arbre de profondeur $D = 2$. La variable de coupure de chaque nœud est désignée par son index i. Les arcs entre les nœuds indiquent le chemin à suivre suivant la valeur de la variable binaire i. Chaque chemin se termine par une feuille désignant le bras k à jouer.

Les arbres de décision peuvent être agrégés en forêts aléatoires [63]. Il existe différentes variantes permettant leur construction [64]. Ici, nous choisissons de retirer aléatoirement une partie des variables disponibles à chaque nœud pour introduire d'avantage de diversité dans les arbres. Le bras à jouer sera sélectionné via un vote à la majorité si tous les chemins correspondant au contexte observé se terminent par une feuille. Dans le cas contraire, le bras joué sera tiré depuis une distribution uniforme.

Soit $\Theta = \{\theta_1, ..., \theta_L\}$ un ensemble variables aléatoires indépendantes et identiquement distribuées avec $\theta_l \in [0, 1]$ pour tout $l \in \{1, ..., L\}$. $c_\theta = \{i, v\}^d$ avec $i \in V$ et $v \in \{0, 1\}$ est un chemin de profondeur d. Le chemin sélectionné au temps t par l'arbre θ est noté $c_\theta(x_t)$. Lorsque $c_\theta(x_t)$ est de profondeur $d \leq D$, Soit $f(\theta) : [0, 1] \times \{0, 1\}^{2^D} \times C \to \{0, 1\}$ une fonction définissant le paramétrage de l'arbre θ.

Les paramètres générés sont :

- la profondeur maximale D_θ,
- les différents $V_{\theta,x,d}$, correspondant à un sous-ensemble de l'ensemble des variables disponibles au nœud de profondeur d de l'arbre θ correspondant au contexte x.

L'arbre glouton paramétré par θ et obtenu avec la connaissance de la distribution $D_{x,y}$ est appelé arbre glouton θ-optimal. Nous appelons forêt aléatoire optimale de taille L, la forêt composée de L arbres gloutons θ-optimaux.

Le bras choisi par l'arbre glouton θ-optimal en présence du contexte x_t est notée $k^*_{\theta,t}$. Le bras choisi par la forêt optimale de taille L est issu d'un vote à la majorité des arbres composant la foret :

$$k^*_t = \arg\max_k \sum_{l=1}^{L} [\![k^*_{\theta_l,t} = k]\!] \,. \tag{5.7}$$

Algorithme 25 : FORÊT DE BANDITS

$t = 1$;
$\forall\theta$ CreationNoeud$(\theta, \{\varnothing\})$; `// Initialisation de la racine de chaque arbre`
répéter
  Observer le vecteur de contexte $\mathbf{x_t}$;
  cheminsTerminés $= 1$;
  pour chaque θ **faire**
    $c_\theta = c_\theta(x_t)$;
    si $d_\theta \neq D_\theta$ *ou* $|S^{var}_{c_\theta}| \neq 1$ *ou* $|S^{bras}_{c_\theta}| \neq 1$ *;* `// Le chemin correspondant à` $\mathbf{x_t}$ `n'est pas terminés`
    alors
      cheminsTerminés $= 0$;
  si *cheminsTerminés* **alors**
    $k = \arg\max_k \sum_{l=1}^{L} [\![k_{\theta_l,t} = k]\!]$; `// Vote à la majorité`
  sinon
    $k = RR([K])$; `// Exploration à l'aide d'un round-robin`
  Jouer le bras k et observer la récompense $y_k(t)$;
  pour chaque θ **faire**
    $t_{c_\theta,k} = t_{c_\theta,k} + 1$; `// On incrémente le temps local au nœud`
    si $|c_\theta| = D_\theta$ **alors**
      $S^{\text{bras}}_{c_\theta} = \text{AE}(t_{c_\theta,k},\, k,\, x_t,\, y_k,\, S^{\text{bras}}_{c_\theta})$;
    sinon
      $S^{\text{var}}_{c_\theta} = \text{VE}(t_{c_\theta,k},\, k,\, x_t,\, y_k,\, S^{\text{var}}_{c_\theta},\, [K])$;
      si $|S^{var}_{c_\theta}| = 1$ **alors**
        CreationNoeud$(\theta, c_\theta + \{S^{\text{var}}_{c_\theta}, 0\})$;
        CreationNoeud$(\theta, c_\theta + \{S^{\text{var}}_{c_\theta}, 1\})$;
  $t = t + 1$;
jusqu'à $t = T$;

Algorithme 26 : CreationNoeud

Données : L'arbre de paramètre θ] et le chemin c
si $|c| = D_\theta$ **alors**
  Créer la feuille correspondant à c et y associer une nouvelle instance de AE;
sinon
  Créer le nœud correspondant à c et y associer une nouvelle instance de VE;

Nous résumons le fonctionnement de l'algorithme. Une FORET DE BANDITS est composée de L arbres de paramètres $\theta_1, ..., \theta_L$ construits en parallèle. Á l'initialisation de l'algorithme, une instance de VE est instanciée à la racine de chaque arbre. Quand un contexte $\mathbf{x_t}$ est reçu :

- Pour chaque arbre θ, la feuille ou le nœud correspondant au chemin $c_\theta(\mathbf{x_t})$ est sélectionné.
- Si tous les chemins mènent à des feuilles ayant chacune éliminées tous les bras sauf un, alors le bras à jouer est choisit avec un vote à la majorité des différents arbres.
- Dans le cas contraire, le bras à jouer est choisi à l'aide d'un round-robin.
- Lorsque qu'une variable a été choisie dans un nœud , alors si la profondeur maximale n'a pas encore été atteinte alors un nouveau nœud contenant une instance de l'algorithme d'élimination de variables est créé pour chaque valeur de la variable sélectionnée. Si la profondeur maximale a été atteinte, une feuille contenant une instance de l'algorithme d'élimination d'action est créée pour chaque valeur de la variable.

L'intervalle de confiance utilisé par les algorithmes d'élimination possède, sous la racine carrée, des quantités influant sur la probabilité de succès de l'algorithme. Les forêts possédant de multiples nœuds et la probabilité globale de l'algorithme étant contrôlée par une inégalité de Boole, il est nécessaire d'inclure de nouveaux termes dans l'intervalle de confiance pour tenir compte des différents algorithmes d'élimination correspondant à chaque chemin. Nous indiquons qu'une quantité est locale à un chemin la suffixant par $|c$.

L'algorithme VE élimine une variable i' d'un nœud de l'arbre de paramètre θ ,s'il existe i tel que :

$$\hat{\mu}^{i'}|c - \hat{\mu}^{i}|c + \epsilon \geq 4\sqrt{\frac{1}{2t_{c,k}} \log \frac{4 \times 2^D K D_\theta L C t_{c,k}^2}{\delta}}\,. \tag{5.8}$$

Le bras k est éliminé de la feuille correspondant à la valeur v de la variable i^* dans l'arbre de paramètre θ quand il existe $k' \neq k$ tel que :

$$\hat{\mu}_{k'}|c - \hat{\mu}_{k}|c + \epsilon \geq 4\sqrt{\frac{1}{2t_{c,k}} \log \frac{4 \times 2^D K L t_{c,k}^2}{\delta}}\,. \tag{5.9}$$

Théorème 15. *Pour $K \geq 2$, $C \geq 2$ et $\epsilon = 0$, la complexité d'échantillonnage de* FORÊT DE BANDITS *nécessaire à la création de la forêt optimale avec une probabilité d'au moins $1 - \delta$ est bornée supérieurement par :*

$$O\left(\frac{2^D K}{\Delta_0^2} \log \frac{KDLC}{\delta \Delta_0} + \frac{2^D K}{\Delta_1^2} \log \frac{KL}{\delta \Delta_1}\right) \tag{5.10}$$

avec $D = \max_\theta D_\theta$, $\Delta_0 = \min_{i \in V,\, c_\theta} \mu^* | c_\theta - \mu^i | c_\theta$ *et* $\Delta_1 = \min_{k \in [K],\, v \in \{0,1\},\, , c_\theta} \mu^{i^*}_{k^*|v} | v,\, c_\theta - \mu^{i^*}_k | v,\, c_\theta$.

Démonstration. Ce résultat est une application directe des Lemmes 2 et 4 ainsi que de l'inégalité de Boole après avoir remplacé les intervalles de confiances des équations (5.1) et (5.4) par ceux des équations (5.8) et (5.9). Plus précisément, le Lemme 2 utilisant l'équation (5.1) □

Théorème 16. *Il existe une distribution $D_{x,y}$ telle que tout algorithme cherchant la* FORÊT DE BANDITS *de taille L a une complexité d'échantillonnage d'au moins :*

$$\Omega\left(2^D [\frac{1}{\Delta_1^2} + \frac{1}{\Delta_2^2}] K \log \frac{1}{\delta}\right) . \tag{5.11}$$

Démonstration. La partie correspondant à la sélection d'action est une application de la borne inférieure donnée par le théorème 1. La partie correspondant à la sélection de variable est une application de la borne inférieure donnée par le lemme 3. Le processus doit-être répété une fois par nœud et une fois par feuille (soit 2^D fois au total). □

5.5 Simulations numériques

Dans cette section, nous complétons les simulations de la section 4.5 en y ajoutant les résultats obtenues par l'algorithme de FORÊTS DE BANDITS(voir Figure 5.2).

FORÊT DE BANDITS

Le nombre d'arbre dans les forêts est fixé à 200. Sont choisis aléatoirement : la profondeur de chaque arbre (entre 10 et 18), la valeur d'epsilon à chaque nœud (entre $0,4$ et $0,8$) ainsi que le sous-ensemble (80% des variables restantes dans le nœud) des variables disponibles.

Lors de l'implémentation, deux modifications de l'algorithme ont été effectuées. Les phases de round-robin sont remplacées par un tirage uniforme depuis l'union des actions restantes dans chaque chemin associé au contexte observé. Les récompenses obtenues sont débiaisées en utilisant le SCORE DE PROPENSION INVERSE (voir [65]).

Les résultats montrent que Forêt de Bandits est un algorithme performant. Ses performances sont de l'ordre de celles de NeuralBandit et de LinUCB. Bien qu'ayant un regret légèrement plus élevé que LinUCB sur le jeu de donnée *Adult*, Forêt de Bandits obtient un taux de classification final plus élevé et un regret cumulé asymptotiquement plus faible.

Table 5.2 – Tableau récapitulatif des expériences. Les taux de classification sont calculés sur les 100000 derniers contextes. Le temps d'exécution moyen a été évalué sur un ordinateur personnel utilisant un processeur Intel Xeon 2, 80GHz et 6Go de RAM.

Algorithme	Regret	Taux de classification	Temps d'exécutions
Forest Cover Type, classe cible : Cover Type (7 classes)			
Banditron	$4.88\ 10^6 \pm 10^5$	51.3%	10 min
LinUCB	$3.39\ 10^6 \pm 10^3$	66.4%	360 min
NeuralBandit.B	$3.33\ 10^6 \pm 2.10^4$	68.1%	150 min
Forêt de Bandits	$3.52\ 10^6 \pm 5.10^4$	65.8%	500 min
Adult, classe cible : occupation (14 classes)			
Banditron	$7.89\ 10^6 \pm 3.10^4$	21.9%	20 min
LinUCB	$6.91\ 10^6 \pm 4.10^4$	31.6%	400 min
NeuralBandit.B	$7.13\ 10^6 \pm 10^5$	29.6%	140 min
Forêt de Bandits	$6.93\ 10^6 \pm 7.10^4$	31.7%	400 min
Census1990, classe cible : Yearsch (18 classes)			
Banditron	$6.97\ 10^6 \pm 2.10^5$	30.7%	26 min
LinUCB	$5.5\ 10^6 \pm 5.10^4$	45.7%	1080 min
NeuralBandit.B	$6.03\ 10^6 \pm 10^5$	41.7%	300 min
Forêt de Bandits	$5.83\ 10^6 \pm 5.10^4$	43.2%	1000 min

Table 5.3 – Les complexité algorithmiques d'une itération (choix du bras à jouer puis mise à jour du modèle) des algorithmes de bandit contextuels. K est le nombre de bras,$dim(X)$ le nombre de variables de contexte, M le nombre d'architectures de NeuralBandit.A ou B, N le nombre de connections composant les réseaux de neurones utilisés par NeuralBandit.A ou B et L est le nombre d'arbres composant la Forêt de bandits.

Algorithme	Complexité algorithmique instantanée
Banditron	$O(Kdim(X))$
LinUCB	$O(Kdim(X)^3)$
NeuralBandit.A ou B	$O(MKdim(X) + MKN)$
Forêt de Bandits	$O(LKdim(X))$

La complexité algorithmique de Forêt de bandits est ajouté à la Table 4.2. Tout comme Neural Bandit, les Forêts de bandits peuvent être facilement

parallélisées en mettant à jour et en calculant la sortie de chaque arbre en parallèle. De cette manière, un facteur d'accélération d'au maximum L peut être atteint.

5.6 Récapitulatif

Dans ce chapitre, nous avons présenté une variante de l'algorithme SUCCESSIVE ELIMINATION adaptée à la sélection de variable dans les nœuds d'arbres de décisions pour le problème des bandits contextuels. Nous avons ensuite utilisé cet algorithme comme brique élémentaire dans la construction de forêts de bandits. L'algorithme de FORÊT DE BANDITS est capable de modéliser des dépendances non-linéaires entre les contextes et les récompense et a l'avantage, contrairement à NEURALBANDIT de posséder des garanties théoriques fortes. En effet, la complexité d'échantillonnage de l'algorithme est optimale à un facteur logarithmique près. Les dépendances vis-à-vis du nombre de variables contextuelles sont logarithmiques, permettant de traiter des contextes de grande taille. L'algorithme est, de plus, adapté aux applications temps réel, sa complexité algorithmique ayant une dépendance linéaire en la profondeur D et les traitements de la tâche de sélection de variables étant parallélisables.

5.7 Preuves

5.7.1 Preuve du Lemme 2

Démonstration. En utilisant l'inégalité d'Hoeffding, au temps t nous avons :

$$P(|\hat{\mu}^i_{k,v} - \mu^i_{k,v}| \geq \epsilon_{t_k} \leq 2\exp\left(-2\epsilon^2 t_k\right) = \frac{\delta}{4KCt_k^2}\,, \tag{5.12}$$

avec $\epsilon_{t_k} = \sqrt{\frac{1}{2t_i}\log\frac{4KCt_i^2}{\delta}}$.

En utilisant l'inégalité d'Hoeffding à chaque pas de temps t_k, en appliquant l'inégalité de Boole puis le fait que $\sum 1/t_k^2 = \pi^2/6$. Pour la variable i, l'inégalité suivante est vérifiée lors d'un évènement $\Omega_{i,v}$ de probabilité au moins $1 - \frac{\delta\pi^2}{24KC}$

$$\hat{\mu}^i_{k,v} - \epsilon_{t_k} \leq \mu^i_{k,v} \leq \hat{\mu}^i_{k,v} + \epsilon_{t_k}\,. \tag{5.13}$$

Lorsque l'évènement $\Omega_{i,v}$ est réalisé, pour l'action $k' = \arg\max_k \hat{\mu}^i_{k,v}$ et $k^* = \arg\max_k \mu^i_{k,v}$, nous avons :

$$\begin{aligned} &\hat{\mu}^i_{k',v} - \epsilon_{t_k} \leq \mu^i_{k',v} \leq \mu^i_{k^*,v} \leq \mu^i_{k^*,v} + \epsilon_{t_k} \leq \mu^i_{k',v} + \epsilon_{t_k} \\ &\Rightarrow \hat{\mu}^i_{k',v} - \epsilon_{t_k} \leq \mu^i_{k^*,v} \leq \mu^i_{k',v} + \epsilon_{t_k}\,. \end{aligned} \tag{5.14}$$

Lors de l'évènement $\Omega_i = \cup_{v\in\{0,1\}}\Omega_{i,v}$, $\Omega_{i,v}$ est réalisé pour toute les valeurs v de la variable i et :

$$\begin{aligned}&\sum_{v\in\{0,1\}}(\hat{\mu}^i_{k',v} - \epsilon_{t_k}) \leq \sum_{v\in\{0,1\}}\mu^i_{k^*,v} \leq \sum_{v\in\{0,1\}}\mu^i_{k',v} + \epsilon_{t_k}\\ &\Leftrightarrow \hat{\mu}^i - \epsilon_{t_k} \leq \mu^i \leq \mu^i + \epsilon_{t_k}\,.\end{aligned} \tag{5.15}$$

Lors de l'évènement $\Omega_{i'}$, avec $i' = \arg\max_i \hat{\mu}^i$, nous avons :

$$\hat{\mu}^{i'} - 2\epsilon_{t_k} \leq \mu^i \leq \mu^*\,. \tag{5.16}$$

Par conséquent, la variable i ne peut pas être la meilleure lorsque :

$$\hat{\mu}^i + 2\epsilon_{t_k} \leq \hat{\mu}^{i'} - 2\epsilon_{t_k}\,. \tag{5.17}$$

Nous notons $\Omega = \cup_{i\in V}\Omega_i$. La probabilité de faire une erreur sur la sélection de la prochaine variable est ainsi inférieure à :

$$\sum_{i\in V} P(\bar{\Omega}_i) \leq \sum_{i\in V}\sum_{v\in\{0,1\}} \frac{K\delta\pi^2}{24KC} \leq \sum_{i\in V}\frac{\delta}{C} \leq \delta\,. \tag{5.18}$$

Nous devons maintenant borner supérieurement le nombre de tirages nécessaires à l'élimination d'un bras sous-optimal. Lorsque l'évènement Ω est réalisé, la meilleure variable n'a pas été éliminée et la variable sous-optimale i est éliminée quand :

$$\hat{\mu}^* - \hat{\mu}^i \geq 4\epsilon_{t_k}\,. \tag{5.19}$$

L'écart entre la variable i et la variable optimale est :

$$\Delta_i = \mu^* - \mu^i\,. \tag{5.20}$$

Supposons que t_k soit assez grand pour que :

$$\Delta_i \geq 4\epsilon_{t_k}\,. \tag{5.21}$$

Si Ω est réalisé, nous avons :

$$\hat{\mu}^i - 2\epsilon_{t_k} \leq \mu^i \leq \hat{\mu}^i + 2\epsilon_{t_k} \tag{5.22}$$

En injectant l'inégalité précédente dans l'inégalité (5.21), nous obtenons :

$$(\hat{\mu}^* + 2\epsilon_{t_k}) - (\hat{\mu}^i + 2\epsilon_{t_k}) \geq 4\epsilon_{t_k}\,. \tag{5.23}$$

Nous avons donc :

$$\hat{\mu}^* - \hat{\mu}^i \geq 4\epsilon_{t_k} \,. \tag{5.24}$$

La condition $\Delta_i \geq 4\epsilon_{t_k}$ implique donc l'élimination de la variable i. En plaçant ϵ_{t_k} par sa valeur, nous avons :

$$\Delta_i \geq \frac{2}{t_i} \log \frac{4KCt_i^2}{\delta} \,. \tag{5.25}$$

Étant donné que :

$$\Delta_i \geq \frac{8}{t_i} \log \frac{4KCt_i^2}{\delta} \Rightarrow \Delta_i \geq \frac{2}{t_i} \log \frac{4KCt_i^2}{\delta} \,, \tag{5.26}$$

il suffit de trouver une valeur de t_k telle que $\Delta_i \geq \frac{4}{t_i} \log \frac{4KCt_i^2}{\delta}$ pour satisfaire la condition $\Delta_i \geq 4\epsilon_{t_k}$.

Nous pouvons maintenant introduire la valeur critique $t_k^* = \frac{64}{\Delta_i^2} \log \frac{4KC}{\delta\Delta_i}$.

Nous injectons t_k^* dans $\frac{8}{t_i} \log \frac{4KCt_i^2}{\delta}$:

$$\begin{aligned}
&\frac{\Delta_i^2}{8 \log \frac{8KC}{\delta\Delta_i}} \left(\log \frac{4KC}{\delta} + 2 \log \frac{64}{\Delta_i^2} + 2 \log \log \frac{4KC}{\delta\Delta} \right) = \\
&\frac{\Delta_i^2}{8 \log \frac{8KC}{\delta\Delta_i}} \left(\log \frac{4KC}{\delta} + 4 \log \Delta_i + 12 \log 2 + 2 \log \log \frac{4KC}{\delta\Delta} \right) \leq \\
&\frac{\Delta_i^2}{8 \log \frac{8KC}{\delta\Delta_i}} \left(\log \frac{4KC}{\delta} + 12 \log 2 + 2 \log \log \frac{4KC}{\delta\Delta} \right) \,.
\end{aligned} \tag{5.27}$$

Pour $x \geq 13$, nous avons :

$$12 \log 2 + 2 \log \log x \leq 4 \log x \,. \tag{5.28}$$

Pour $4KC \geq 13$ nous avons :

$$\begin{aligned}
&\frac{\Delta_i^2}{8 \log \frac{4KC}{\delta\Delta_i}} \left(\log \frac{4KC}{\delta} + 12 \log 2 + 2 \log \log \frac{4KC}{\delta\Delta} \right) \leq \\
&\frac{\Delta_i^2}{8 \log \frac{4KC}{\delta\Delta_i}} 8 \log \frac{4KC}{\delta\Delta_i} = \Delta_i^2 \,.
\end{aligned} \tag{5.29}$$

Ainsi, étant donné que $4\epsilon_{t_k}$ est une fonction décroissante en fonction de t_k, la condition $\Delta_i \geq 4\epsilon_{t_k}$ est vérifiée à partir de $t_k = t_k^*$, entrainant l'élimination de la variable i et bornant supérieurement le nombre de tirages avant l'élimination de la variable i.

Nous notons $\Delta = \min_{i \in V} \Delta_i$. Les actions étant jouées suivant un round-robin, nous avons $t = Kt_k$.

Nous pouvons donc conclure que toutes les variables sous-optimales seront éliminées après $\frac{64K}{\Delta^2}\log\frac{4KC}{\delta}$ itérations avec une probabilité d'au moins $1-\delta$.

□

5.7.2 Preuve du Lemme 3

Démonstration. Soit $y^i_{k,v}$ une variable aléatoire bornée correspondant à la récompense de l'action k quand la valeur v de la variable i est observée . Soit y^i une variable aléatoire telle que :

$$y^i = \max_k y^i_{k,v}$$

Nous avons :

$$\mathbb{E}_{D_{x,y}}[y^i] = \mu^i$$

Chaque y^i est mis à jour à chaque pas de temps t_k quand chaque action a été jouée une fois. Soit Θ la somme des variables aléatoires binaires $\theta_1, ..., \theta_{t_k}, ..., \theta_{t^*_k}$ telles que $\theta_{t_k} = \mathbb{1}_{y^i(t_k)\geq y^j(t_k)}$. Soit p_{ij} la probabilité que l'utilisation de la variable i engendre plus de récompenses que l'utilisation de la variable j. Nous avons :

$$p_{ij} = \frac{1}{2} - \Delta_{ij} \text{ , où } \Delta_{ij} = \mu^i - \mu^j.$$

L'inégalité de Slud (voir [66]) affirme que si $p \leq 1/2$ et $t^*_k \leq x \leq t^*_k.(1-p)$, nous avons :

$$P(\Theta \geq x) \geq P\left(Z \geq \frac{x - t^*_k.p}{\sqrt{t^*_k.p(1-p)}}\right) \text{ ,} \tag{5.30}$$

où Z est une variable aléatoire $\mathcal{N}(0,1)$ suivant une loi normale.

Pour choisir la meilleure variable entre i et j, nous devons trouver le temps e t^*_k où $P(\Theta \geq t^*_k/2) \geq \delta$. Pour expliciter le nombre d'observations t^*_k nécessaires à l'estimation de Δ_{ij}, nous adaptons les arguments développés dans [67]. En utilisant l'inéquation de Slud (voir équation 6.16), nous avons :

$$P(\Theta \geq t^*_k/2) \geq P\left(Z \geq \frac{t^*_k.\Delta_{ij}}{\sqrt{t^*_k.p_{ij}(1-p_{ij})}}\right) \text{ ,} \tag{5.31}$$

Ensuite, nous utilisons la borne inférieure de la fonction d'erreur (voir [68]) :

$$P(Z \geq z) \geq 1 - \sqrt{1 - \exp\left(-\frac{z^2}{2}\right)}$$

Ainsi, nous avons :

$$
\begin{aligned}
P(\Theta \geq t_k^*/2) &\geq 1 - \sqrt{1 - \exp\left(-\frac{t_k^*.\Delta_{ij}^2}{2p_{ij}(1-p_{ij})}\right)} \\
&\geq 1 - \sqrt{1 - \exp\left(-\frac{t_k^*.\Delta_{ij}^2}{p_{ij}}\right)} \\
&\geq \frac{1}{2}\exp\left(-\frac{t_k^*.\Delta_{ij}^2}{p_{ij}}\right)
\end{aligned}
$$

Comme $p_{ij} = 1/2 - \Delta_{ij}$, nous avons :

$$
\log\delta = \log\frac{1}{2} - \frac{t_k^*.\Delta_{ij}^2}{1/2 - \Delta_{ij}} \geq \log\frac{1}{2} - 2t_k^*.\Delta_{ij}^2
$$

Ains i :

$$
t_k^* = \Omega\left(\frac{1}{\Delta_{ij}^2}\log\frac{1}{\delta}\right)
$$

Nous utilisons maintenant le fait que comme toutes les valeurs de toutes les variables sont observées quand une action est jouée : les $C(C-1)/2$ estimations de biais sont résolues en parallèle. Dans le pire des cas, $\min_{ij}\Delta_{ij} = \min_j \Delta_{i^*j} = \Delta$. Ainsi tout algorithme nécessite une complexité d'échantillonnage d'au moins t^*, où :

$$
t^* = K.t_k^* = \Omega\left(\frac{K}{\Delta^2}\log\frac{1}{\delta}\right)
$$

□

Chapitre 6

Sélection d'experts apprenants

Contents

6.1 Introduction

Nous avons présenté deux approches adaptées aux bandits contextuels dans le chapitre II). La première approche apprend directement les dépendances entre les contextes, les bras et les récompenses. Pour cela, des modèles linéaires [43, 41], des FORÊTS DE BANDITS (voir chapitre 5) ou bien des réseaux de neurones peuvent-être utilisés (voir chapitre 4). La seconde est basée sur les algorithmes de sélection de politiques ou d'experts comme RandomizedUcb [48] ou EXP4 [31]. Ces algorithmes délèguent la partie contextuelle du problème à un ensemble fini d'experts et cherchent à trouver celui dont les choix de bras offrent les plus hautes

performances. Ces deux approches ont cependant des faiblesses. Les algorithmes construisant une politique à partir données possèdent souvent des garanties théoriques et sont performants en pratique, mais sont sujet à l'erreur d'approximation inhérente à l'espace des politiques explorées par l'algorithme. Leurs performances peuvent cependant dépendre fortement du paramétrage de l'algorithme utilisé. Les algorithmes de sélection de politiques possèdent aussi des garanties théoriques, mais dépendent en pratique de la présence d'une politique performante dans leur ensemble. Leur plus grand inconvénient est leur complexité algorithmique en $O(\text{poly}(T))$, rendant leur utilisation compliquée lorsque l'horizon de temps T est grande.

Ce chapitre présente une approche hybride, la sélection d'experts apprenants. Un expert apprenant est composé des éléments caractérisant une instance de bandits contextuels (paramètres, graine aléatoire, ...). Lorsque cette approche est utilisée, un tour de jeu est composé des étapes suivantes :

- Le joueur choisit un expert.
- L'expert choisit le bras à jouer.
- Une récompense est obtenue.
- Tout les experts modifient leurs politiques en prenant en compte le dernier bras joué.
- Le joueur modifie sa politique de sélection des experts.

Étant donné que les experts apprennent, leurs performances évoluent au cours du temps (voir Figure 6.1). Il est nécessaire pour le joueur de prendre en compte cette non-stationnarité pour lui permettre de trouver l'expert ayant les meilleures performances à la fin de son apprentissage. Nous verrons qu'en utilisant des connaissances a priori sur la structure de cette non-stationnarité, il est possible d'obtenir un regret logarithmique par rapport au meilleur expert ayant terminé sur apprentissage.

Cette approche à plusieurs avantages, l'espace des politiques de chaque expert est exploré par des algorithmes efficaces ayant de fortes garanties théoriques (construction de forêts d'arbres gloutons avec les forêts de bandits, en résolvant un système linéaire pour linUCB). Ces algorithmes souffrent cependant d'une erreur de biais due aux politiques considérées. Le but de l'algorithme de sélection d'experts apprenants est de réduire ce biais en sélectionnant l'expert le plus performant parmi l'ensemble considéré.

6.2 Formalisation

Soit $[K] = \{1, ..., K\}$ un ensemble de bras. Soit $x_t \in X$ un vecteur de contexte décrivant l'environnement au tour t. Soit $y_k(t) \in [0, 1]$ la récompense obtenue

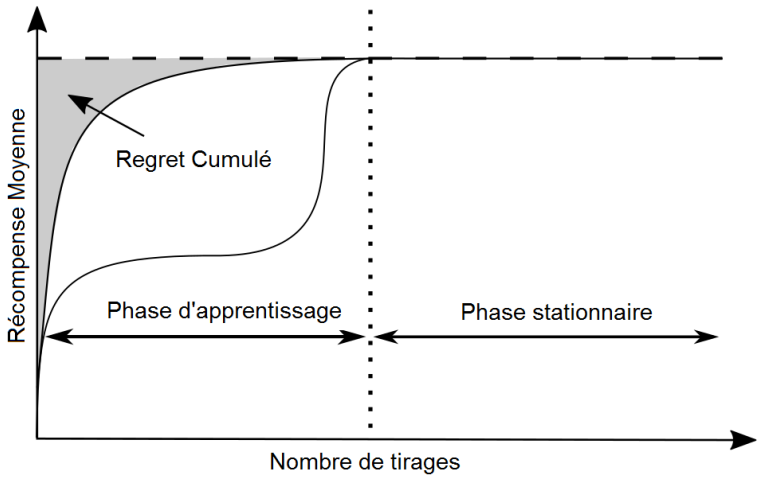


FIGURE 6.1 – Deux exemples de courbes d'apprentissage.

par le bras k au tour t. Soit $D_{x,y}$ la distribution jointe sur (x, y). Soit $\pi : X \to [K]$ une politique. Soit $[M] = \{1, ..., M\}$ un ensemble d'experts. L'ensemble des politiques atteignables par l'expert m au cours de son apprentissage est Π_m et $\Pi = \bigcup m = 1^M \Pi_m$ est l'ensemble des politiques. La politique utilisée par l'expert m au tour t est noté $\pi_{m,t}$ et sa récompense moyenne est définie par :

$$\mu_m(t) = \mathbb{E}_{D_{x,y}}[y_{\pi_{m,t}(x)}].$$

La politique optimale de l'ensemble Π_m est :

$$\pi_m^* = \arg\max_{\pi \in \Pi_m} \mathbb{E}_{D_{x,y}}[y_{\pi(x)}].$$

La récompense moyenne de la politique optimale de l'ensemble Π_m est μ_m. La politique optimale est :

$$\pi^* = \arg\max_m \mu_m \,.$$

Nous définissons le pseudo-regret de la tâche de sélection d'experts apprenants comme :

$$\bar{R}(T) = \sum_{t=1}^{T} \mathbb{E}_{D_{x,y}} \left[y_{\pi^*(x_t)}(t) - y_{\pi_{m_t,t}(x_t)}(t) \right],$$

où m_t est l'expert joué au tour t.

L'expert dont l'ensemble des politiques contient π^* est noté m^*.

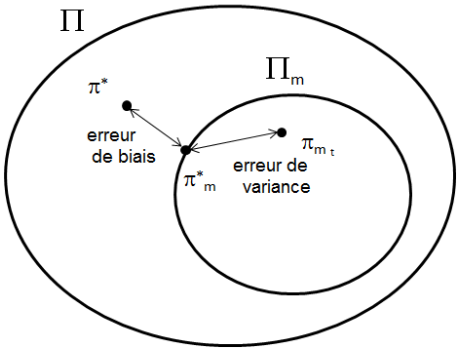


FIGURE 6.2 – L'algorithme d'apprentissage réduit l'erreur de variance dans l'ensemble Π_m. L'algorithme de sélection d'experts apprenants réduit l'erreur de biais.

Le pseudo-regret de l'expert m peut-être décomposée en terme de biais et de variance (voir Figure 6.2) :

$$\begin{aligned}\bar{R}_m(T) &= \sum_{t=0}^{T} (\mu_{m^*} - \mu_m(t)) \\ &= \sum_{t=0}^{T} (\mu_{m^*} - \mu_m) + \sum_{t=0}^{T} (\mu_m - \mu_m(t)) \,.\end{aligned}$$

Il est important de remarquer que m^* est considéré comme un expert sans erreur de biais. L'espérance du pseudo-regret de l'algorithme de sélection d'experts apprenants est définie comme :

$$\bar{R}(T) = \sum_{t=0}^{T} (\mu_{m^*} - \mu_{m_t}) + \sum_{t=0}^{T} (\mu_{m_t} - \mu_{m_t}(t)) \,.$$

6.3 Réduction en problèmes de bandits stationnaires

Une première solution au problème de sélection d'experts apprenant est de supprimer la non-stationnarité au travers d'une approche Apprendre puis Explorer et Exploiter (LTEE) :

— M problèmes de bandits contextuels sont initialisés et résolus durant la phase d'apprentissage,

— un algorithme de bandits stationnaire est ensuite utilisé pour sélectionner le meilleur expert.

Les algorithmes de bandits contextuels apprennent les dépendances entre contextes, bras et récompenses et optimisent l'erreur due à la variance. La tâche de sélection d'experts apprenants réduit ensuite l'erreur due au biais des politiques.

6.3.1 Apprendre puis Explorer et Exploiter

Phase d'apprentissage

Lors de la phase d'apprentissage, les bras sont joués de manière séquentielle afin d'obtenir un apprentissage non biaisé des experts. La taille de cette période d'apprentissage est contrôlée par la complexité d'échantillonnage des experts. Dans le cas où ceux-ci sont capables d'apprendre en parallèle, l'apprentissage est stoppé lorsque le nombre d'observations correspond à la complexité d'échantillonnage maximale parmi les experts. Lorsque les experts nécessitent un apprentissage séquentiel, la somme des complexités d'échantillonnage est utilisée à la place. Les algorithmes Forêts de bandits ou les bandits linéaires [69] possèdent des complexités d'échantillonnage connues.

Définition. *La complexité d'échantillonnage n_m de l'expert m est le nombre d'observations tirées depuis $D_{x,y}$ nécessaires pour trouver π^*_m avec une probabilité d'au moins $1-\delta$.*

Proposition 2. *Après avoir obtenue n observations depuis $D_{x,y}$, la récompense moyenne de chaque expert m est μ^*_m avec une probabilité $1-M\delta$, où $n = \max_m n_m$.*

Démonstration. La preuve est une application directe de la définition de la complexité d'échantillonnage et de l'inégalité de Boole. □

Proposition 2 est utilisée pour définir la taille de la phase d'apprentissage. À la fin de celle-ci, le compromis entre l'exploration et l'exploitation des experts peut-être contrôlé par tout algorithme de bandit tel que UCB [17] ou Successive Elimination [14].

Phase d'exploration et d'exploitation

À la fin de la phase d'exploration, les experts ont terminé leur apprentissage et leurs performances sont stationnaires. Afin de trouver le meilleur expert, nous utilisons maintenant l'algorithme Successive Elimination [14].

Successive Elimination joue les experts de manière séquentielle. La moyenne empirique de chaque expert commence à être maintenue à partir du tour n :

$$\hat{\mu}_m(t) = \frac{1}{t-n} \sum_{i=n+1}^{t} [\![m_i = m]\!] y_{\pi_{m_i}(x_i)}(i).$$

Lorsque l'écart entre la moyenne empirique de l'expert m et du meilleur expert empirique est assez grand, alors l'expert m est sous-optimal avec une forte probabilité et n'est plus joué par l'algorithme.

Algorithme 27 : LTEE

Données : $\delta \in (0,1]$, $\epsilon \in [0,1)$
Résultat : une ϵ-approximation du meilleur expert
pour $t = 1, ..., n$ **faire**
  Jouer séquentiellement un bras $k_t \in [K]$;
  Mettre à jour chaque experts $m \in [M]$ avec le vecteur $(x_t, k_t, y_{k_t}(t))$;
$S_t = [M]$, $r = 0$;
tant que $t \leq T$ **faire**
  pour chaque $m \in S_t$ **faire**
    Jouer le bras $k_t = \pi_m(x_t)$;
    Mettre à jour $\hat{\mu}_m(t+1)$;
    $t = t + 1$ [1];
  $r = r + 1$;
  $m_{\max} = \arg\max_{m \in S} \hat{\mu}_m(t)$;
  Retirer de S_t tout les experts m satisfaisant :

$$\hat{\mu}_{\max}(t) - \hat{\mu}_m(t) + \epsilon \geq 2\sqrt{\log(4r^2M/\delta)/2r}\,;$$

Théorème 17. *Pour $M > 0$, $\delta > 0$, et $\epsilon = 0$, la complexité d'échantillonnage de* LTEE *est borné supérieurement par :*

$$O\left(\frac{M}{\Delta^2}\log\frac{M}{\delta\Delta} + n\right),$$

où Δ est la différence de récompense moyenne entre les deux meilleurs experts.

Démonstration. La complexité d'échantillonnage deLTEE est bornée supérieurement par celle de SUCCESSIVE ELIMINATION (voir théorème 2) et n, la complexité d'échantillonnage nécessaire à l'apprentissage de tous les experts (voir propriété 2). □

Théorème 18. *Il existe une distribution $D_{x,y}$ telle que tout algorithme apprenant M experts et cherchant ensuite le meilleur a une complexité d'échantillonnage d'au moins :*

$$\Omega\left(\frac{M}{\Delta^2}\log\frac{1}{\delta} + \mathcal{N}\right),$$

où $1 - \delta$ est la probabilité de trouver le meilleur expert de l'ensemble et $\mathcal{N}$ une borne inférieure sur la complexité d'échantillonnage de l'ensemble d'experts.

Démonstration. La borne inférieure est obtenue en sommant la borne inférieure du problème d'identification du meilleur bras (voir théorème 1 dans [12]) et la borne

1. Afin de faciliter la lecture de l'algorithme, $t \leq T$ est testé seulement au début du tour de round-robin. Afin d'assurer l'arrêt de l'algorithme au tour T, il est nécessaire de tester cette condition une fois de plus à cette marque.

inférieure $\mathcal{N}$ de l'apprentissage des experts. Dans le pire des cas, tous les experts terminent leur apprentissage au même moment et les observations provenant de la phase d'apprentissage ne peuvent pas être utilisées pour évaluer les performances des experts. □

Lorsque la complexité d'échantillonnage des experts est optimale, i.e. quand $n = \mathcal{N}$, l'algorithme LTEE est optimal à un facteur logarithmique près.

6.3.2 Application aux Forêts de bandits

Nous illustrons l'algorithme LTEE avec une application aux Forêts de bandits. L'ensemble d'experts est composé de MForêts de bandits initialisées avec différents paramètres : le nombre L_m d'arbres, la profondeur D_m, le nombre de variables contextuelles C disponibles à chaque coupure. Après la phase d'apprentissage, la meilleure forêt est cherchée à l'aide de Successive Elimination.

Théorème 19. *Pour $M > 0$, $K \geq 2$, $C \geq 2$, $\delta > 0$, et $\epsilon = 0$, la complexité d'échantillonnage de* LTEE *utilisant un ensemble de*Forêts de bandits *est borné supérieurement par :*

$$O\left(\frac{M}{\Delta^2}\log\frac{M}{\delta\Delta} + 2^D\left[\frac{KD}{\Delta_1^2}\log\frac{KDLC}{\delta\Delta_1} + \frac{KD}{\Delta_2^2}\log\frac{KL}{\delta\Delta_2}\right]\right),$$

où Δ est la différence de récompenses moyennes entre les deux meilleurs experts, et où Δ_1 et Δ_2 sont les minimums des différences de récompenses moyennes entre les deux meilleures variables dans un nœud non terminal et les deux meilleurs bras dans une feuille. $L = \max_m L_m$ et $D = \max_m D_m$.

Démonstration. Le théorème 15 est utilisé pour définir n. □

Théorème 20. *Il existe une distribution $D_{x,y}$ telle que tout algorithme apprenant M*Forêts de bandits *et cherchant ensuite la meilleure à une complexité d'échantillonnage d'au moins*

$$\Omega\left(\frac{M}{\Delta^2}\log\frac{1}{\delta\Delta} + 2^D\left[\frac{1}{\Delta_1^2} + \frac{K}{\Delta_2^2}\right]\log\frac{1}{\delta}\right),$$

où $1-\delta$ est la probabilité de construire la foret optimale et de la trouver parmi l'ensemble d'experts.

Démonstration. Le théorème 16 donne la valeur de $\mathcal{N}$, c'est à dire la borne inférieure sur la complexité d'échantillonnage nécessaire à la construction des Forêts de bandits. □

Dans le cas de la sélection de Forêts de bandits, l'algorithme LTEE est optimal à des facteurs logarithmiques près.

6.4 Bandits adverses avec budget

Dans la section précédente, les experts étaient appris au début du jeu puis sélectionnés à l'aide d'un algorithme de bandits lorsque leurs performances devenaient stationnaires. L'algorithme LTEE obtient de fortes garanties théoriques et peut être très efficace en pratique lorsque les experts possèdent des complexités d'échantillonnage similaires. Cependant, lorsque les complexités algorithmiques diffèrent, LTEE dépense un grand nombre d'itérations à jouer les bras séquentiellement pour apprendre les derniers experts.

Dans cette section, nous présentons l'algorithme LEE (*Learn, Explore and Exploit* en anglais), qui apprend les experts en parallèle, tout en estimant leurs récompenses et en éliminant les moins performants. L'apprentissage, l'exploration et l'exploitation en parallèle sont modélisés par une nouvelle formalisation que nous appelons *bandits adverses avec budget*. Nous analysons EXP3 dans ce formalisme et nous expliquons ensuite la méthodologie utilisée par LEE. Dans ce cadre, un nouvel algorithme appelé ÉLIMINATION SUCCESSIVES RANDOMISÉES AVEC BUDGET est décrit puis analysé. Cet algorithme permet d'optimiser le compromis exploration/exploitation sur des distributions de récompenses possédant un support de distribution plus grand que 1, permettant ainsi d'apprentissage en parallèle des experts depuis une distribution non uniforme.

6.4.1 Formalisation

Soit $S = 1, ..., M$ un ensemble d'experts. La récompense $y_m(t) \in [0, 1]$ obtenue par le joueur après avoir sélectionné l'expert m est tirée depuis la distribution $D_m(t)$ de moyenne $\mu_m(t) \in [0, \mu_m]$ avec $\mu_m \in [0, 1]$. Les différentes valeurs $\mu_m(t)$ sont choisies arbitrairement par un adversaire inconscient au début du jeu. L'adversaire est contraint par un budget B de manière à ce que $\forall m$, $\sum_{t=1}^{T} \mu_m - \mu_m(t) \leq B$.

Le problème de sélection d'experts apprenants est abstrait en considérant les étapes où $\mu_m(t) \neq \mu_m$ comme choisies par l'adversaire. Cette contrainte de budget est la principale innovation comparée aux *récompenses contaminées* introduites par Seldin dans [34]. On remarque que contrairement à l'algorithme proposé par Seldin, notre formalisation permet de modéliser différents types de courbes d'apprentissage : le budget peut être dépensé entièrement durant une courte période de temps ou au contraire dispersé au cours du jeu (voir Figure 6.1).

6.4.2 EXP3 pour les bandits adverses avec budget

Lorsque le montant du budget B n'est pas connu mais que l'horizon de temps l'est, EXP3 peut-être utilisé sur ce problème. L'espérance du regret cumulé de l'algorithme est bornée supérieurement dans le théorème 21.

Théorème 21. *Pour tout $M > 0$, l'espérance du regret de l'algorithme* EXP3

appliqué sur un problème de sélection d'experts apprenants et utilisant comme paramètre $\gamma = \sqrt{\frac{M\log(M)}{(e-1)T}}$ *est borné par :*

$$\mathbb{E}\left[R(T)\right] \leq 2\sqrt{(e-1)MT\log(M)} + B\,, \tag{6.1}$$

où $\mathbb{E}$ *désigne l'espérance vis-à-vis de la distribution jointe des experts* $(m_1, ..., m_T)$ *et des récompenses* $(y_1, ..., y_T)$*.*

La preuve du théorème 21 est détaillée en sous section 6.7.

6.4.3 Apprentissage Non Biaisé des Experts

Un estimateur non biaisé $r_{k_t}(t)$ de $y_{k_t}(t)$ est utilisé afin de pouvoir mettre à jour en parallèle chaque expert m en utilisant le n-uplet $(x_t, k_t, r_{k_t}(t))$:

$$r_{k_t}(t) = y_{k_t}(t)/P_t(k_t)$$

où k_t est le bras choisi au pas de temps t par une politique randomisée avec une probabilité $P_t(k_t)$.

L'espérance de l'estimateur vis-à-vis de la randomisation des bras est :

$$\mathbb{E}[r_{k_t}] = P_t(k_t).y_{k_t}/P_t(k_t) = y_{k_t}.$$

Cette approche est appelée *Score de Propension Inverse* [65] (*Inverse Propensity Scoring* en anglais) et peut être utilisée pour évaluer la performance des experts lorsque la probabilité de jouer chaque bras n'est jamais nulle (voir théorème 1 dans in [70]).

Le Lemme suivant garantie que la probabilité de tirage des bras non éliminés de l'ensemble des experts ne peut être égale à zéro lorsque LEE est utilisé.

Lemme 5. *Lorsque l'algorithme* LEE *utilise des experts randomisés tirant un bras depuis l'ensemble* $A_{m,t}$ *suivant une distribution uniforme, la probabilité de tirer tout bras non éliminé* k *est :*

$$P_t(k) \geq \frac{1}{MK}$$

Démonstration. Soit S_t l'ensemble des experts restants au tour t et $A_{m,t}$ l'ensemble des bras restants pour l'expert m. Soit A_t l'union des sous-ensembles de bras, $A_t = \bigcup_m A_{m,t}$. Dans le pire des cas, il existe un bras $k \in A_t$ qui est joué uniquement par un expert. Chaque expert à une probabilité $1/|S_t|$ d'être tirés. La probabilité de tirer le bras k est :

$$\begin{aligned} P_t(k) =& \sum_m P_t(k|\pi_m).P_t(\pi_m) \\ =& \frac{1}{|S_t|}.\sum_m \frac{1}{|A_{m,t}|} \\ =& \frac{1}{MK}.\sum_m \frac{K}{|A_{m,t}|} \geq \frac{1}{MK} \end{aligned}$$

□

6.4.4 Éliminations Successives Randomisées avec Budget

Dans cette section, nous présentons un algorithme d'identification du meilleur expert capable d'être utilisé lorsque les distributions de récompenses ont un support supérieur à 1 et que les récompenses sont contaminées par un adversaire avec un budget B. Cet algorithme est utilisé comme brique élémentaire pour permettre auxFORÊTS DE BANDITS d'apprendre en parallèle avec des récompenses obtenues via un tirage non uniforme des bras.

Il y a trois différences significatives avec l'algorithme SUCCESSIVE ELIMINATION : le choix des experts (et réciproquement des bras) est randomisé, les récompense sont débiaisées par la probabilité de jouer l'expert et la contamination des récompenses est prise en compte lors de l'élimination des experts. Lorsque les récompenses sont débiaisées, le problème devient équivalent à un problème à information complète où l'expert joué a une récompense entre 0 et $|S_t|.y_m(t)$ et où les experts non joués ont une récompense nulle.

Théorème 22. *Pour $M > 0$, et $\delta > 0$, et $\epsilon = 0$, la complexité d'échantillonnage de l'algorithme* ÉLIMINATIONS SUCCESSIVES RANDOMISÉES AVEC BUDGET *est bornée supérieurement par :*

$$O\left(\frac{M^2}{\Delta^2}\left(\log(\frac{M}{\delta\Delta}) + B\right)\right).$$

où Δ est la différence entre les récompenses moyennes des deux meilleurs experts à la fin de leur apprentissage.

Corollaire 6. *Pour $M > 0$, $\delta > 0$, et $\epsilon = 0$, l'espérance du pseudo-regret de l'algorithme* ÉLIMINATIONS SUCCESSIVES RANDOMISÉES AVEC BUDGET *est bornée supérieurement par :*

$$O\left(\frac{M^2}{\Delta}\left(\log(\frac{MT}{\Delta}) + B\right)\right).$$

Algorithme 28 : Éliminations Successives Randomisées avec Budget

Données : $\delta \in (0,1]$, $\epsilon \in [0,1)$, $B \geq 0$
Résultat : une ϵ-approximation du meilleur expert
$S_1 = S$, $\forall m$, $\hat{\mu}_m(0) = 0$, $Z(0) = |S|^2$, $t = 1$;
tant que $|S_t| > 1$ **faire**
  $Z(t) = Z(t-1) + |S_t|^2$;
  Tirer un expert $m_t \sim 1/|S_t|$;
  pour chaque $m \in S_t$ **faire**
    $\hat{\mu}_m(t) = \frac{t-1}{t}\hat{\mu}_m(t-1) + \frac{|S_t|.y_m(t)}{t}[\![m = m_t]\!]$;
  $m_{\max} = \arg\max_{m \in S_t} \hat{\mu}_m(t)$;
  Retirer de S_t tout les experts m tels que :

$$\hat{\mu}_{\max}(t) - \hat{\mu}_m(t) + \epsilon \geq B/t + 2\sqrt{\frac{Z(t)}{2t^2}\log\left(\frac{4Mt^2}{\delta}\right)}$$

Les preuves du théorème 22 et du corollaire 6 sont détaillées dans les sous-sections 6.7 et 6.7.

Cette borne supérieure, incluant un facteur M^2 peut sembler haute mais est inévitable lorsque les récompenses sont débiaisées par la probabilité de jouer l'expert. Nous fournissons une borne inférieure de la complexité d'échantillonnage du problème d'identification du meilleur expert lorsque l'information est complète, que le support de la distribution des récompenses est $[0, M]$ au lieu de $[0, 1]$ et que $B = 0$.

Théorème 23. *Pour $B = 0$ et $\epsilon = 0$, il existe une distribution D_Y telle que tout algorithme observant à chaque tour les récompenses $0 \leq Y_m(t) \leq M$ de tout les experts et cherchant le meilleur à une complexité d'échantillonnage d'au moins :*

$$\Omega\left(\frac{M^2}{\Delta^2}\log\frac{1}{\delta}\right),$$

où Δ est la différence de récompense moyenne entre les deux meilleurs experts.

La preuve du théorème 23 est fournie en sous-section 6.7. Cette borne inférieure montre l'optimalité de l'algorithme Éliminations Successives Randomisées avec Budget lorsque le support des distributions générant les récompenses est $[0, M]$ pour $B = 0$.

Algorithme 29 : APPRENDRE, EXPLORER ET EXPLOITER (LEE)

Données : $\delta \in (0,1]$, $\epsilon \in [0,1)$, $B \geq 0$
Résultat : une ϵ-approximation du meilleur expert
$S_1 = S$, $\forall m$, $\hat{\mu}_m(0) = 0$, $Z(0) = |S|^2$;
pour tous les $t = 1, ..., T$ **faire**
  $Z(t) = Z(t-1) + |S_t|^2$;
  Tirer un expert $m_t \sim 1/|S_t|$;
  Tirer un bras $k_t \sim A_{m,t}$;
  $P_{k_t} = 0$;
  pour chaque $m \in S_t$ **faire**
    $P_{k_t} = P_{k_t} + \frac{1}{|A_{m,t}|.|S_t|}$;
  pour chaque $m \in S_t$ **faire**
    $\hat{\mu}_m(t) = \frac{t-1}{t}\hat{\mu}_m(t-1) + \frac{|S_t|.y_m(t)}{t}[\![m = m_t]\!]$;
    Mise à jour de l'expert m;
  $m_{\max} = \arg\max_{m \in S_t} \hat{\mu}_m(t)$;
  Retirer de S_t tous les experts m tels que :

$$\hat{\mu}_{\max}(t) - \hat{\mu}_m(t) + \epsilon \geq B/t + 2\sqrt{\frac{Z(t)}{2t^2}\log\left(\frac{4Mt^2}{\delta}\right)}$$

6.4.5 Apprendre, Explorer et Exploiter pour les FORÊTS DE BANDITS

En utilisant la même méthodologie que ÉLIMINATIONS SUCCESSIVES RANDOMISÉES AVEC BUDGET nous pouvons modifier FORÊT DE BANDITS pour lui permettre d'apprendre avec des récompenses provenant de bras tirés depuis des distributions non uniformes. Nous appelons l'algorithme obtenu FORETS DE BANDITS RANDOMISÉES. Le théorème 24 borne le pseudo-regret pour le cas où l'ensemble contient uniquement des FORETS DE BANDITS RANDOMISÉES initialisées avec des paramétrages différents.

Théorème 24. *Pour $M > 0$, $K \geq 2$, $C \geq 2$, $\delta > 0$, et $\epsilon = 0$, la complexité d'échantillonnage de* LEE *utilisant un ensemble de* FORETS DE BANDITS RANDOMISÉES *est borné supérieurement par :*

$$O\left(\max(t_1^*, t_2^*)\right),$$

$$où\ t_1^* = \frac{M^2}{\Delta^2}\left(\log(\frac{M}{\delta\Delta}) + B\right),$$

$$et\ t_2^* = 2^D\left[\frac{K^2M^2D}{\Delta_1^2}\log\frac{KMDLC}{\delta\Delta_1} + \frac{K^2M^2D}{\Delta_2^2}\log\frac{KL}{\delta\Delta_2}\right],$$

où Δ est la différence de récompenses moyennes entre les deux meilleurs experts et où Δ_1 et Δ_2 sont les différences minimales entre les récompenses moyennes de chacune des deux meilleures variables dans un nœud non terminal et des deux meilleurs bras dans une feuille, $L = \max_m L_m$, $D = \max_m D_m$.

La preuve du théorème 24 est détaillée en sous-section 6.7. La borne supérieure obtenue sur la complexité d'échantillonnage nécessaire à la sélection de la meilleure FORETS DE BANDITS RANDOMISÉES ne semble pas être aussi serrée que celle de LTEE (voir théorème 19). En effet, le premier terme t_1^* contient un facteur additionnel M et le terme de budget B, correspondant au pseudo-regret cumulé du meilleur expert, ne peut pas être considéré comme négligeable. De plus, le second terme t_2^*, correspondant à la complexité d'échantillonnage de l'apprentissage d'une instance de FORETS DE BANDITS RANDOMISÉES, souffre d'un facteur additionnel M^2K, due au tirage des experts.

6.4.6 Discussion

L'analyse en pire des cas de LEE n'est pas très flatteuse et une autre manière d'aborder le problème aurait pu être envisagée : retirer toute randomisation et choisir les experts de manière déterministe tout en les faisant apprendre uniquement lorsqu'ils sont à l'origine du choix d'action. De cette manière, les dépendances en M^2, K^2 et B seraient respectivement devenues des dépendances en M, K et MB. Cette approche a cependant certains désavantages, le budget n'est pas partagé entre les experts et chacun doit effectuer sa propre période d'apprentissage. Certains arguments moraux peuvent cependant montrer que la dépendance quadratique en M^2 peut être évitée. Lors de la période critique d'apprentissage, l'analyse en pire des cas utilise comme hypothèse le fait qu'un bras nécessaire à l'apprentissage a une probabilité $\frac{1}{MK}$ d'être joué. En pratique, notamment dans le cas des arbres, nous pouvons considérer que cette probabilité est d'avantage de l'ordre de $\frac{1}{K}$ lorsque les algorithmes sont en période d'exploration. En adoptant ce point de vu et en paramétrant le critère d'élimination de manière optimiste, la dépendances en M^2 disparait. LEE est comparé à l'alternative dont nous venons de discuter dans la section suivante.

L'algorithme LTEE nécessite de connaitre une borne supérieure sur la complexité d'échantillonnage S des experts, LEE une borne supérieure sur le budget B et EXP3 ne nécessite aucune connaissance à priori. Chaque niveau de connaissance inclue le suivant étant donné que $S \geq B$. Cette relaxation des connaissances induit cependant un regret cumulé théorique plus élevé.

6.5 Simulations numériques

Dans cette section, nous utilisons les jeux de données présentés dans la section 4.5 pour illustrer les méthodes de sélection d'experts apprenant sur des ensembles

d'ARBRES DE BANDITS. Il est important de remarquer que dans l'approche *Apprendre puis Explorer Et Exploiter*, l'apprentissage des experts n'est pas totalement arrêté après la période d'apprentissage, un expert m continue d'apprendre durant les itérations où celui-ci est à l'origine du choix d'action.

L'ensemble d'experts. L'ensemble est composé de 24 arbres de profondeur 1, chacun ne pouvant observer que 20% des variables de contextes et d'un arbre de profondeur 12.

Les résultats expérimentaux de cette section sont résumés dans la table 6.1.

TABLE 6.1 – Tableau récapitulatif des expériences. Les taux de classification sont calculés sur les 100000 derniers contextes.

Algorithme	Regret	Taux de classification
Forest Cover Type, classe cible : Cover Type (7 classes)		
LTEE	$3.56\ 10^6 \pm 2.10^4$	65.4%
LEE	$3.68\ 10^6 \pm 2.10^5$	65.4%
EXP3.S	$4.21\ 10^6 \pm 6.10^4$	62.9%
ARBRE OPTIMAL	$3.55\ 10^6 \pm 4.10^4$	65.4%
Adult, classe cible : occupation (14 classes)		
LTEE	$7.01\ 10^6 \pm 2.10^4$	29.6%
LEE	$7.19\ 10^6 \pm 5.10^5$	29.6%
EXP3.S	$7.30\ 10^6 \pm 5.10^4$	28.5%
ARBRE OPTIMAL	$7.00\ 10^6 \pm 7.10^4$	29.6%

Les Figures 6.3 et 6.7 illustrent les différences de performances pouvant être observées avec des ARBRES DE BANDITS initialisés avec des profondeurs maximales différentes.

La Figure 6.4 montre les regrets cumulés obtenues par l'approche *Apprendre puis Explorer Et Exploite* sur le jeu de données **Forest Cover Type**. Lorsque la période d'apprentissage S est égale à 0, l'arbre de profondeur 12 est éliminé avant qu'il ait le temps de converger. L'augmentation de la période d'apprentissage à $S = 50000$ réduit ce phénomène, l'arbre de profondeur 12 étant parfois sélectionné. Malgré une baisse du regret cumulé moyen, la variance reste forte. A partir de $S = 100000$ itérations dédiées à l'apprentissage, la sélection d'experts s'effectue correctement et le meilleur arbre est sélectionné de manière consistante. La sélection de l'expert optimal après la fin de son apprentissage est rapide, les écarts de performances avec les experts sous-optimaux étant élevés, et n'a que peut d'impact sur le regret cumulé. Comme indiqué par l'analyse, lorsque le paramètre S est sur-estimé, le regret cumulé augmente linéairement avec S.

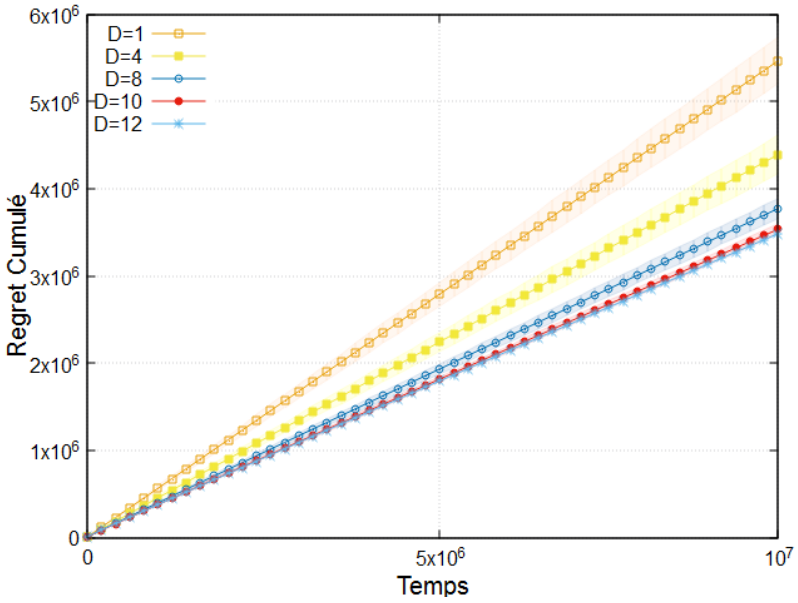


FIGURE 6.3 – **Forest Cover Type.** Le regret cumulé d'ARBRES DE BANDITS initialisés avec les profondeurs maximales différentes.

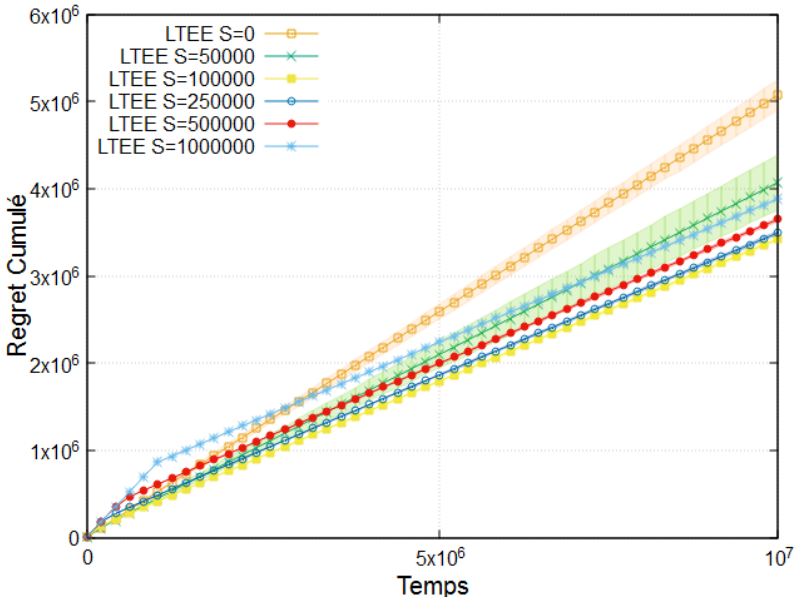


FIGURE 6.4 – **Forest Cover Type.** Le regret cumulé de la sélection d'experts apprenants utilisant des stratégies *Apprendre puis Explorer Et Exploiter* est utilisée.

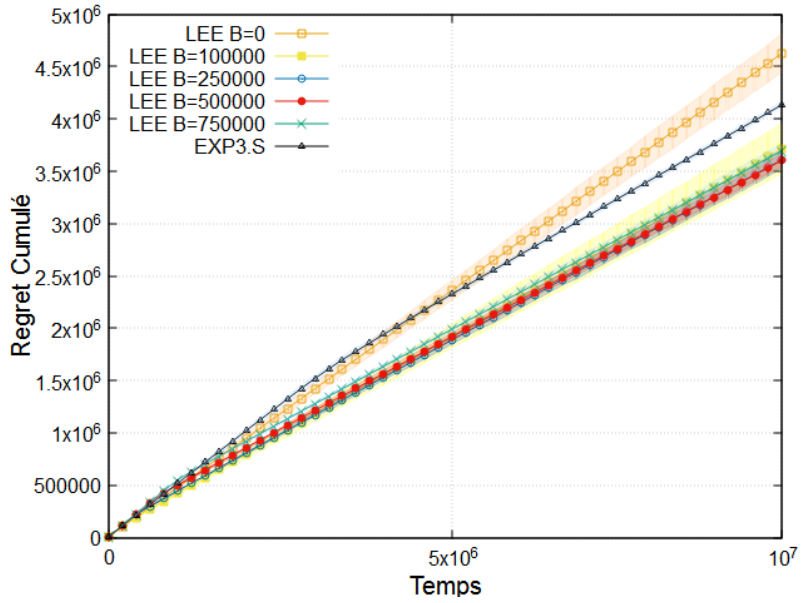


FIGURE 6.5 – **Forest Cover Type.** Le regret cumulé de la sélection d'experts apprenants utilisant des stratégies Apprendre Et Explorer Et Exploiter est utilisée.

La Figure 6.5 montre quant à elle les performances de LEE. Avec le bon paramétrage, les regrets cumulés sont légèrement supérieurs à ceux de l'arbre optimal. EXP3.S, bien que tardivement, sélectionne le meilleur expert à la fin du premier tiers de l'exécution de l'algorithme.

La Figure 6.6 montre les performances de l'approche alternative à LEE, présentée dans la discussion de la section précédente. Lorsque cette alternative est utilisée, les experts n'apprennent pas en parallèle et possèdent un budget qui leur est propre. Les experts apprennent ainsi plus lentement et leur sélection est plus longue. En utilisant cette approche alternative, le regret cumulé de LEE avec son meilleur paramétrage n'est que marginalement meilleur que celui d'EXP3.S (dont les experts n'apprennent aussi que lorsqu'ils choisissent l'action à jouer). Contrairement à LEE, EXP3.S ne nécessite pas de connaissance du budget.

Les performances des algorithmes de sélection d'experts apprenants sur le jeu de données **Adult** sont présentées en Figures 6.7, 6.8 et 6.9. L'ordonnancement des courbes suivant les algorithmes ainsi que la sensibilité aux paramètres de complexité d'échantillonnage et de budget sont identiques.

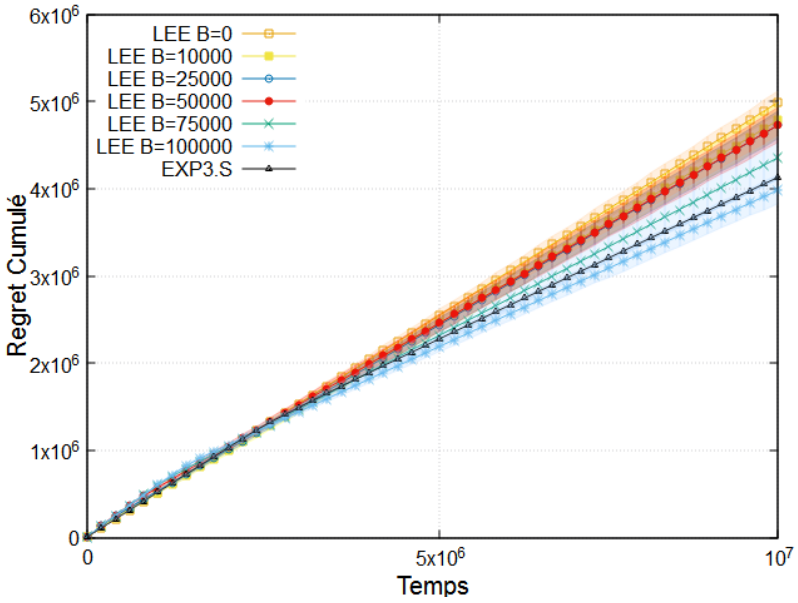


FIGURE 6.6 – **Forest Cover Type.** Le regret cumulé de la sélection d'experts apprenants utilisant une approche alternative à la stratégies Apprendre Et Explorer Et Exploiter est utilisée. Ici, le budget n'est pas partagé et propre à chaque expert.

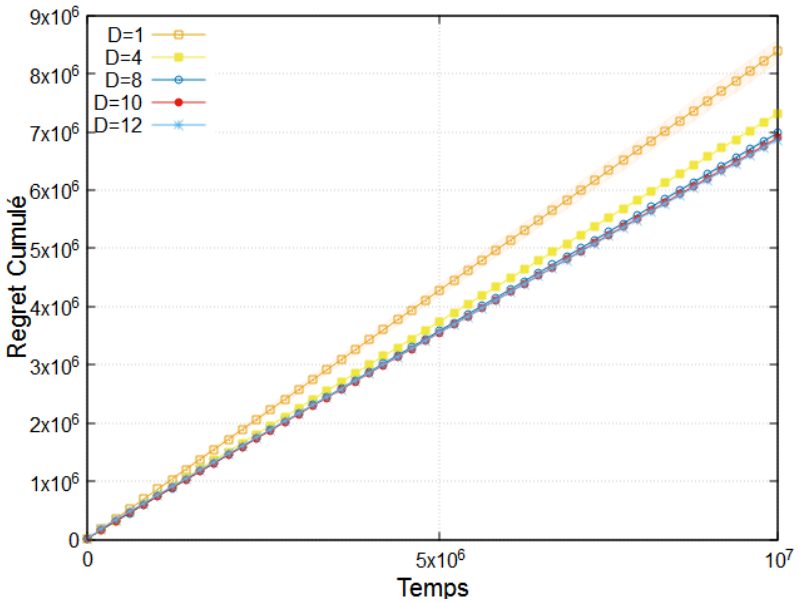


FIGURE 6.7 – **Adult.** Le regret cumulé d'ARBRES DE BANDITS initialisés avec les profondeurs maximales différentes.

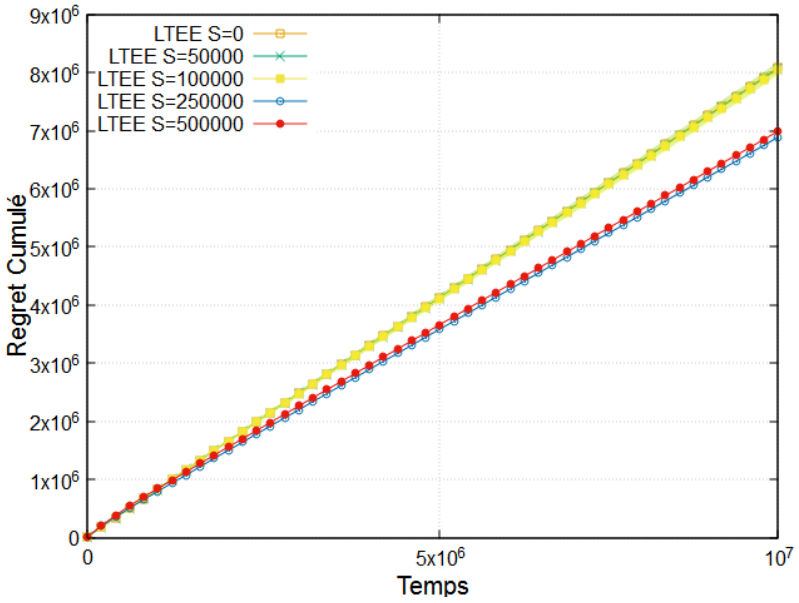


FIGURE 6.8 – **Adult.** Le regret cumulé de la sélection d'experts apprenants utilisant des stratégies *Apprendre puis Explorer Et Exploiter* est utilisée.

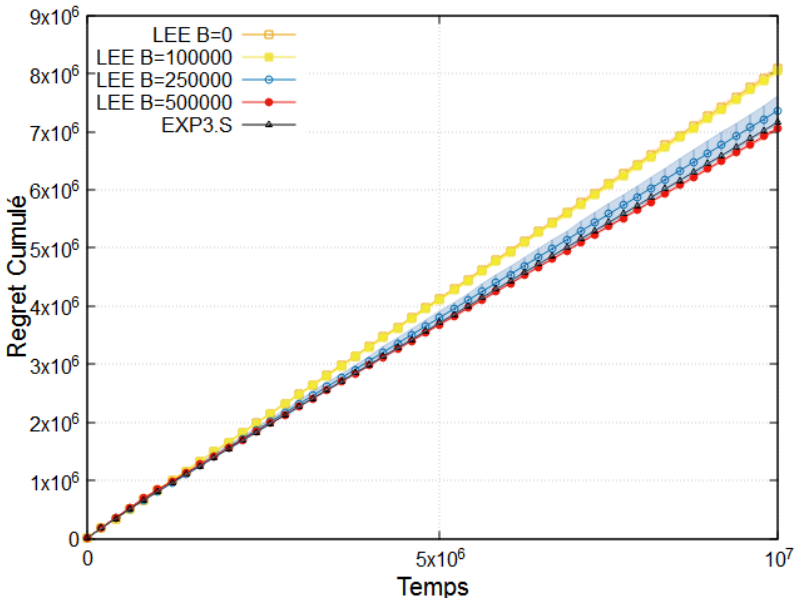


FIGURE 6.9 – **Adult.** Le regret cumulé de la sélection d'experts apprenants utilisant des stratégies Apprendre Et Explorer Et Exploiter est utilisée.

6.6 Récapitulatif

Dans ce chapitre nous avons formulé la sélection d'experts apprenants comme un problème de bandits non-stationnaires avec budget. Nous avons proposé trois

méthodes permettant de résoudre le problème suivant les informations dont dispose le joueur. Lorsqu'une borne supérieure sur la complexité d'échantillonnage est connue, l'algorithme LTEE peut être utilisé. Celui-ci consiste à tout d'abord entrainer les experts, utilisant les bornes sur la complexité d'échantillonnage pour arrêter l'apprentissage. Lorsque celui-ci est terminé, les performances des experts sont théoriquement stationnaires et le meilleur peut-être maintenant être identifié avec SUCCESSIVE ELIMINATION. L'algorithme LEE suppose quant à lui la connaissance d'une borne sur le pseudo-regret des experts. Incorporer cette connaissance dans l'intervalle de confiance utilisé pour l'élimination des experts permet d'entrainer les experts tout en les sélectionnant. Ce problème est beaucoup plus compliqué que celui résolu par LTEE, les récompenses devant être débiaisées pour l'apprentissage des experts. Nous avons de plus analysé l'usage d'EXP3 en tant que méta-bandit pour le cas où aucune garantie théorique sur les experts n'est connu du joueur (nous supposons cependant qu'une borne supérieure sur le pseudo-regret cumulé existe). Les analyses de LEE et LTEE sont finalement poursuivies dans le cas où les experts sont des FORÊTS DE BANDITS.

6.7 Preuves

6.7.1 Preuve du théorème 21

Démonstration. La preuve originale d' EXP3 est suivie jusqu'à l'équation :

$$\sum_{t=1}^{T} y_{m_t}(t) \geq (1-\gamma) \sum_{t=1}^{T} \hat{y}_j(t) - \frac{M \log(M)}{\gamma} - (e-2)\frac{\gamma}{M} \sum^{M} \sum_{t=1}^{T} \hat{y}_i(t)\,.$$

Comme $y_m(t) \leq 1$ nous avons :

$$\mathbb{E}\left[\sum_{t=1}^{T} \hat{y}_m(t)\right] \leq T\,.$$

Pour tout m nous avons :

$$\sum_{t=1}^{T} \hat{y}_m(t) = \sum_{t=1}^{T} \frac{y_m(t)}{p_{m_t}(t)} [\![m = m_t]\!]\,.$$

Nous prenons maintenant l'espérance vis-à-vis de la distribution des récompenses :

$$\mathbb{E}\left[\sum_{t=1}^{T} \hat{y}_m(t)\right] = \sum_{t=1}^{T} \frac{\mu_m^* - \mu_m^* + \mu_m(t)}{p_{m_t}(t)} [\![m = m_t]\!]\,.$$

$$\mathbb{E}\left[\sum_{t=1}^{T}\hat{y}_m(t)\right] = \sum_{t=1}^{T}\left(\frac{\mu_m^*}{p_{m_t}(t)} - \frac{\mu_m^* - \mu_m(t)}{p_{m_t}(t)}\right)[\![m = m_t]\!]\,.$$

Nous majorons le coté droit de l'équation précédente en utilisant l'hypothèse de budget et prenons ensuite l'espérance vis-à-vis de la distribution des bras :

$$\mathbb{E}\left[\sum_{t=1}^{T}\hat{y}_m(t)\right] \geq \sum_{t=1}^{T}\mu_m^*(t) - \frac{MB}{\gamma}\,.$$

Nous fixons $j = m^*$ et injections l'équation précédente dans la première équation :

$$\mathbb{E}\left[\sum_{t=1}^{T}\mu_{m_t}(t)\right] \geq (1-\gamma)\left(\sum_{t=1}^{T}\mu_m^*(t) - \frac{MB}{\gamma}\right) - \frac{M\log(M)}{\gamma} - (e-2)\gamma T\,,$$

$$\sum_{t=1}^{T}\mu_m^*(t) - E\left[\sum_{t=1}^{T}\mu_{m_t}(t)\right] \leq (e-1)\gamma T + \frac{M\log(M)}{\gamma} + \frac{MB}{\gamma}\,.$$

$$\mathbb{E}\left[\sum_{t=1}^{T}\mu_m^*(t) - \sum_{t=1}^{T}\mu_{m_t}(t)\right] \leq (e-1)\gamma T + \frac{M\log(M)}{\gamma} + \frac{MB}{\gamma}\,.$$

En fixant $\gamma = \sqrt{\frac{M\log(M)}{(e-1)T}}$, nous obtenons :

$$\mathbb{E}\left[R(T)\right] \leq 2\sqrt{(e-1)MT\log(M)} + B\,.$$

□

6.7.2 Preuve du théorème 22

Démonstration. En utilisant l'inégalité d'Hoeffding au temps t, pour tout m nous avons :

$$P\left(|\hat{\mu}_m - \mathbb{E}[\hat{\mu}_m]| \geq \epsilon_t\right) \leq 2\exp\left(-\frac{2\epsilon_t^2 t^2}{\sum_{i=1}^{t}|S_t|^2}\right)\,.$$

où $\mathbb{E}$ l'espérance vis-à-vis de la distribution jointe $D_{y,m}$ où $p_m(t) = 1/|S_t$.

Nous bornons supérieurement $\sum_{i=1}^{t}|S_t|^2$ par tM^2.

En fixant

$$\epsilon_t = \sqrt{\frac{M^2}{2t}\log\left(\frac{4Mt^2}{\delta}\right)}\,,\ \text{nous avons :}$$

$$P\left(|\hat{\mu}_m - \mathbb{E}[\hat{\mu}_m]| \geq \epsilon_t\right) \leq 2\exp\left(\frac{-2\sqrt{\frac{M^2}{2t}\log\left(\frac{4Mt^2}{\delta}\right)}^2 t}{M^2}\right) = \frac{\delta}{2Mt^2}\,.$$

En utilisant l'inégalité d'Hoeffding sur chaque pas de temps t, en utilisant l'inégalité de Boole et le fait que $\sum 1/t^2 = \pi^2/6$, alors l'inégalité suivante est vérifiée avec une probabilité d'au moins $1 - \frac{\delta\pi^2}{12M}$:

$$\hat{\mu}_m - \epsilon_t \leq \mathbb{E}[\hat{\mu}_m] \leq \hat{\mu}_m + \epsilon_t\,. \tag{6.2}$$

Un expert m' reste dans l'ensemble S aussi longtemps que pour chaque $m \in S - \{m'\}$:

$$\hat{\mu}_m - \epsilon_t < \hat{\mu}_{m'} + \frac{B}{t} + \epsilon_t\,. \tag{6.3}$$

En utilisant (6.2) dans (6.3) :

$$\mathbb{E}[\hat{\mu}_m] - 2\epsilon_t < \mathbb{E}[\hat{\mu}_{m'}] + \frac{B}{t} + 2\epsilon_t\,. \tag{6.4}$$

Pour chaque m nous avons :

$$\hat{\mu}_m = \frac{1}{t}\sum_{i=0}^{t}\frac{y_m(i)}{p_m(i)}[\![m = m_i]\!]\,,\ \text{où}\ p_m(i) = \frac{1}{|S_t|}.$$

En prenant l'espérance vis-à-vis de la distribution des récompenses D_y nous avons :

$$\mathbb{E}_{D_y}[\hat{\mu}_m] = \frac{1}{t}\sum_{i=0}^{t}\frac{\mu_m - \mu_m + \mu_m(i)}{p_m(i)}[\![m = m_i]\!]\,,$$

$$\mathbb{E}_{D_y}[\hat{\mu}_m] = \frac{1}{t}\sum_{i=0}^{t}\left(\frac{\mu_m}{p_m(i)} - \frac{\mu_m - \mu_m(i)}{p_m(i)}\right)[\![m = m_i]\!]\,. \tag{6.5}$$

En prenant l'espérance vis-à-vis de la distribution des experts $\langle m_1, ..., m_T\rangle$ des deux cotés de l'équation (6.5) avec $p_m(t) = 1/|S_t|$ et en utilisant $B \geq 0$, nous avons :

$$\mu_m - \frac{B}{t} \leq \mathbb{E}[\hat{\mu}_m] \leq \mu_m\,. \tag{6.6}$$

Si (6.4) est vérifiée pour m et m' , alors en utilisant (6.6) :

$$\mu_m - 2\epsilon_t - \frac{B}{t} \leq \mu_{m'} + 2\epsilon_t + \frac{B}{t}\,.$$

$$\Delta_{m,m'} < 4\epsilon_t + \frac{2B}{t}\,. \tag{6.7}$$

En remplaçant ϵ_t dans (6.7) :

$$\Delta_{m,m'} < 4\sqrt{\frac{M^2}{2t}\log\left(\frac{4Mt^2}{\delta}\right)} + \frac{2B}{t}\,,$$

$$\Delta^2_{m,m'} < \frac{8M^2}{t}\log(\frac{4Mt^2}{\delta}) + \frac{4B^2}{t^2}\,. \tag{6.8}$$

Pour $m' = m^*$ et $m \neq m^*$, l'équation (6.8) est toujours vraie, impliquant que l'expert optimal restera dans l'ensemble avec une forte probabilité pour tout t.

Nous supposons $t \geq B$.

$$\Delta^2_{m,m'} < \frac{8M^2}{t}\log(\frac{4Mt^2}{\delta}) + \frac{4B}{t}\,. \tag{6.9}$$

L'expert m' a ou a été éliminé si l'inégalité (6.9) est fausse.

Soit $\Delta = \min_{i \neq m^*} \Delta_{m^*,i}$ et

$$t_1^* = \frac{64^2}{\Delta^2} M^2 \log\left(\frac{16M}{\delta\Delta}\right)\,.$$

Nous notons

$$C_1(t) = \frac{8M^2}{t}\log(\frac{4Mt^2}{\delta})\,.$$

Pour $t = t_1^*$,

$$C_1(t_1^*) = \frac{8\Delta^2}{64^2 \log\frac{16M}{\delta\Delta}}\left(\log\frac{4M}{\delta} + 4\log\frac{64M}{\Delta} + 2\log\log\frac{16M}{\delta\Delta}\right)\,,$$

$$C_1(t_1^*) = \frac{8\Delta^2}{64^2 \log\frac{16M}{\delta\Delta}}\left(\log\frac{4M}{\delta} - 4\log\Delta + 4\log M + 24\log 2 + 2\log\log\frac{16M}{\delta\Delta}\right)\,,$$

$$C_1(t_1^*) \leq \frac{8\Delta^2}{64^2 \log\frac{16M}{\delta\Delta}}\left(8\log\frac{16M}{\delta\Delta} + 24\log 2 + 2\log\log\frac{16M}{\delta\Delta}\right)\,.$$

Pour $X > 8$ nous avons

$$24 \log 2 + 2 \log \log X < 8 \log X \,.$$

Ainsi, nous avons

$$C_1(t_1^*) \leq \frac{8\Delta^2}{64^2 \log \frac{16M}{\delta\Delta}} \left(16 \log \frac{16M}{\delta\Delta} \right) ,$$

$$C_1(t_1^*) \leq \frac{\Delta^2}{512} \,. \tag{6.10}$$

Comme $C_1(t_1^*)$ est strictement décroissante en fonction de t alors (6.10) est vraie pour tout $t > t_1^*$.

Quand $t > t_1^*$, il existe $C_2(t)$ tel que :

$$\Delta^2 = C_1(t) + C_2(t) \,.$$

Pour invalider l'équation (6.9) nous avons besoin de trouver une valeur de $t_2^* > t_1^*$ pour laquelle

$$t \geq \frac{4B}{C_2(t)} \,.$$

Comme $C_2(t) = \Delta^2 - C_1(t)$ nous avons $C_2(t) \geq \Delta^2 - \frac{\Delta^2}{512}$,

$$t \geq \frac{2048B}{511\Delta^2} \,. \tag{6.11}$$

Pour $t = t_2^*$ avec

$$t_2^* = \frac{64^2}{\Delta^2} M^2 \log \left(\frac{16M}{\delta\Delta} \right) + \frac{5B}{\Delta^2} \,.$$

(6.11) est toujours vraie, invalidant (6.9) et impliquant l'élimination de tout les experts sous-optimaux avec une probabilité d'au moins $1 - \delta$, concluant la preuve.

□

6.7.3 Preuve du corollaire 6

Démonstration. Soit t^* le nombre de pas de temps nécessaires pour trouver le meilleur expert avec une forte probabilité.

L'espérance du regret-cumulé au temps T est :

$$\mathbb{E}_{D_{x,y}}[R(T)] = \mathbb{E}_{D_{x,y}}\left[\sum_{t=1}^{t^*} r(t) + \sum_{t=t^*+1}^{T} r(t)\right].$$

Ensuite, pour chaque pas de temps de la phase d'exploration (i.e. $t \leq t^*$), nous bornons l'espérance du pseudo-regret instantané $\mathbb{E}_{D_{x,y}}[r(t)]$ par $\Delta = \min_{m \neq m^*}(\mu_{m^*} - \mu_m)$, nous obtenons :

$$\begin{aligned}\mathbb{E}_{D_{x,y}}[R(T)] &\leq t^*.\Delta + (T - t^*)\mathbb{E}_{D_{x,y}}[r(t)] \\ &\leq t^*.\Delta + (T - t^*)\left[1.\mathbb{P}(k_t \neq k_t^*) + 0.\mathbb{P}(k_t = k_t^*)\right].\end{aligned}$$

En utilisant l'inégalité de Boole, nous avons :

$$\begin{aligned}\mathbb{E}_{D_{x,y}}[R(T)] &\leq t^*.\Delta + (T - t^*)\left[\mathbb{P}(i \neq i^*) + \mathbb{P}(k \neq k^*)\right] \\ &\leq t^*.\Delta + 2(T - t^*)\delta\,.\end{aligned}$$

La complexité d'échantillonnage t^* est donnée par le théorème 22. Si nous fixons $\delta = \frac{1}{T}$, nous obtenons :

$$\mathbb{E}_{D_{x,y}}[R(T)] \leq t^*.\Delta + 2\frac{T - t^*}{T} \leq O\left(\frac{M^2}{\Delta}\left(\log(\frac{MT}{\Delta}) + B\right)\right).$$

□

6.7.4 Preuve du théorème 23

Démonstration. Soit $0 \leq Y_i(t) \leq M$ une variable aléatoire bornée correspondant aux récompenses de l'expert i. En divisant $Y_i(t)$ par M nous avons :

$$0 \leq \frac{Y_i(t)}{M} \leq 1 \tag{6.12}$$

et

$$\mathbb{E}\left[\sum_{t=1}^{t^*} \frac{Y_i(t)}{M}\right] = \frac{\mu_i}{M}\,. \tag{6.13}$$

Soit Θ la somme des variables aléatoires binaires $\theta_1, ..., \theta_t, ...\theta_{t^*}$ où

$$\theta_t = \left[\!\left[\frac{Y_i(t)}{M} \geq \frac{Y_j(t)}{M}\right]\!\right]. \tag{6.14}$$

Soit $p_{i,j}$ la probabilité que l'utilisation de l'expert i mène à d'avantage de récompense que l'utilisation de lu bras j. Nous avons

$$p_{i,j} = \frac{1}{2} - \Delta_{i,j}, \text{ où } \Delta_{i,j} = \frac{\mu_i - \mu_j}{M}\,. \tag{6.15}$$

L'inégalité de Slud (voir [66]) pour $p \leq 1/2$ et $t_k^* \leq x \leq t_k^*.(1-p)$ est :

$$P(\Theta \geq x) \geq P\left(Z \geq \frac{x - t_k^*.p}{\sqrt{t_k^*.p(1-p)}}\right) , \tag{6.16}$$

où Z est une variable aléatoire normale $\mathcal{N}(0,1)$.

Pour choisir le meilleur expert entre i et j, nous devons trouver le temps t^* où $P(\Theta \geq t^*/2) \geq \delta$.

Afin de fixer le nombre d'observations t^* nécessaire à l'estimation de Δ_{ij}, nous rappelons et développons les arguments de [67].

En utilisant l'inégalité de Slud (voir inéquation 6.16), nous avons :

$$P(\Theta \geq t_k^*/2) \geq P\left(Z \geq \frac{t^*.\Delta_{ij}}{\sqrt{t^*.p_{ij}(1-p_{ij})}}\right) , \tag{6.17}$$

Ensuite, nous utilisons la borne inférieure de la fonction d'erreur (voir cite C55) :

$$P(Z \geq z) \geq 1 - \sqrt{1 - \exp\left(-\frac{z^2}{2}\right)}$$

Ainsi, nous avons :

$$\begin{aligned} P(\Theta \geq t^*/2) &\geq 1 - \sqrt{1 - \exp\left(-\frac{t^*.\Delta_{ij}^2}{2p_{ij}(1-p_{ij})}\right)} \\ &\geq 1 - \sqrt{1 - \exp\left(-\frac{t^*.\Delta_{ij}^2}{p_{ij}}\right)} \\ &\geq \frac{1}{2}\exp\left(-\frac{t^*.\Delta_{ij}^2}{p_{ij}}\right) \end{aligned}$$

Comme $p_{ij} = 1/2 - \Delta_{ij}$, nous avons :

$$\log\delta = \log\frac{1}{2} - \frac{t^*.\Delta_{ij}^2}{1/2 - \Delta_{ij}} \geq \log\frac{1}{2} - 2t^*.\Delta_{ij}^2$$

Nous en déduisons :

$$t^* = \Omega\left(\frac{1}{\Delta_{ij}^2}\log\frac{1}{\delta}\right)$$

Dans le pire des cas, μ_m est le même pour tout les experts sous-optimaux $m \neq m^*$ et $\min_{ij} \Delta_{ij} = \min_j \Delta_{i^*j} = \Delta'$.

En utilisant le fait que θ_t est connu à chaque itération pour toutes les bras et que $\Delta' = \frac{\Delta}{\mu_{m^*} - \mu_m}$ alors la complexité d'échantillonnage est bornée inférieurement par :

$$\Omega\left(\frac{M^2}{(\mu_{m^*} - \mu_m)^2} \log \frac{1}{\delta}\right) . \tag{6.18}$$

□

6.7.5 Preuve du théorème 24

Démonstration. LEE joue le bras optimal lorsque l'expert optimal a terminé son apprentissage et qu'il a été sélectionné par l'algorithme. L'apprentissage et la sélection du meilleur expert sont effectués en parallèle. Ainsi, la complexité d'échantillonnage nécessaire pour obtenir l'expert optimal est le maximum de t_1^*, la complexité d'échantillonnage nécessaire à la sélection du meilleur expert et t_2^*, la complexité d'échantillonnage nécessaire à sa construction. En effet, en cas d'écart de performances élevé entre les experts, l'expert optimal peut-être sélectionné avant qu'il ait terminé son apprentissage.

L'algorithme FORÊT DE BANDITS RANDOMISÉES diffère de FORÊT DE BANDITS par une tirage randomisé des bras, la pondération des récompenses par la probabilité de tirer le bras joué et un intervalle de confiance différent.

La preuve de FORÊT DE BANDITS RANDOMISÉES suit les même étapes que la preuve du théorème 15 et contrairement à l'algorithme initial, le support de la distribution de récompense est maintenant $[0, KM]$ au lieu de $[0, 1]$.

□

Chapitre 7

Conclusion et perspectives

Dans nous avons abordé l'apprentissage en ligne en nous appuyant sur le formalisme des bandits manchots et bandit contextuels. Nous avons dans un premiers temps étudié les impacts de la non-stationnarité des récompenses en l'absence de contexte.

Dans la Partie I, nous avons tout d'abord dégagé une hypothèse permettant d'effectuer la tache d'identification du meilleur bras lorsque les distributions générant les récompenses sont non-stationnaires. Cette formulation généralise le problème des bandits stochastique stationnaires et ouvre les algorithmes d'élimination d'action à une nouvelle classe de problème. L'algorithme SER3 obtient des bornes supérieures sur la complexité d'échantillonnage et le pseudo-regret cumulé quasi-optimales. Nous avons ensuite étudié le cas où le meilleur bras peut changer au cours du jeu, invalidant l'hypothèse utilisée par SER3. Deux algorithmes différents SER4 et EXP3.R sont proposés pour résoudre ce problème. Ils sont tout deux basés sur les réinitialisations successives de leurs sous-routines (SER3 et EXP3) afin de faciliter les changements de meilleurs bras empiriques. Les réinitialisations de SER4 sont randomisées et indépendantes des données, compensant les nombreuses phases de sélection de la meilleure action entre les réinitialisations par la complexité d'échantillonnage logarithmique de SER3. Les réinitialisations de EXP3.R sont quant à elles contrôlées par un détecteur de rupture, ne réinitialisant l'algorithme que lorsqu'un changement de meilleur bras est détecté. Nous avons adapté la définition de la complexité d'échantillonnages au cas où le meilleur bras peut changer au cours du jeu et analysé SER4 dans ce cadre. Étant le premier algorithme à utiliser cette mesure, quantifier la qualité de la borne est difficile. Pour cela, nous dérivons de ce résultat une borne supérieure sur l'espérance du pseudo-regret cumulé de SER4. Ces garanties théoriques sont comparables à celles de EXP3.R et des autres algorithmes de l'état de l'art. L'avantage de nos méthodes par rapport à d'autres modifications d'algorithmes stochastiques (comme SW-UCB ou Discounted-UCB) est d'obtenir une dépendance en N (le nombre de changements de meilleurs bras) et non en M (le nombre de changements de distributions), qui peut-être égal à T dans notre formalisme.

La Partie II se consacre aux bandits contextuels. Dans le chapitre 4 nous avons proposé une heuristique pour l'utilisation de réseaux de neurones dans les bandits contextuels, permettant d'apprendre des motifs non-linéaires. Les expériences montrent une grande variance dans les performances suivant les différents paramétrages. Afin de contrôler en ligne la sélection de la meilleure architecture, nous avons utilisé des méta-bandits non-stationnaires et pu observer une forte augmentation des performances. Le chapitre 5 nous a permit d'adapter l'algorithme des forêts aléatoires au formalisme des bandits contextuels. La complexité d'échantillonnage de l'algorithme de construction de forêts de bandits est proche de l'optimal à un facteur logarithmique près. Celle-ci possède une dépendance logarithmique vis-à-vis du nombre de variables contextuelles, permettant de prendre en compte des contextes de grande taille avec un faible impact sur le regret. Sa complexité algorithmique est faible, mais grandissant exponentiellement avec la profondeur de l'arbre. L'algorithme FORET DE BANDITS offre ainsi une alternative convaincante à NEURALBANDIT pour l'obtention de modèles non-linéaires pour les bandits contextuels tout en ayant l'avantage de posséder des garanties théoriques fortes.

Le chapitre 6 pose le problème de la sélection d'experts apprenants comme un méta-problème de bandit non-stationnaires (les bras sont des instances de bandits contextuels dont les performances évoluent au cours de leur apprentissage) et propose plusieurs méthodologies permettant leur sélection en ligne. Trois cas sont traités, dépendant du niveau de connaissance sur les garanties théoriques des algorithmes dont dispose le joueur ; le joueur connait une borne supérieure sur la complexité d'échantillonnage des experts ; le joueur connait une borne supérieure sur le pseudo-regret cumulé des experts ; une borne supérieure sur le pseudo-regret cumulé des experts existe mais est inconnue du joueur. L'usage de ces méthodes est illustré en poursuivant leur analyse lorsque les experts sont des FORÊTS DE BANDITS.

Ces travaux peuvent être poursuivis dans plusieurs directions. La première est l'adaptation d'autre algorithmes d'apprentissage pour leur permettre d'être utilisable lorsque l'information est partielle, par exemple, un algorithme de BOOSTING [71] utilisant des ARBRES BANDITS de faible profondeur. La seconde est l'adaptation des méthodes utilisées par SER4 et EXP3.R aux algorithmes contextuels pour leur permettre d'être utilisables lorsque les distributions générant les contextes ou les récompenses sont non-stationnaires. L'adaptation de SER4 aux bandits contextuels est directe, n'importe quel algorithme de bandits pouvant être utilisé à la place de SER3. Celle d'EXP3.R est quand à elle plus complexe et nécessiterait l'élaboration d'un détecteur de rupture contextuel. Une dernière piste de recherche prometteuse dans le cadre contextuel concernerait la création de détecteurs de ruptures pouvant être utilisés à partir des observations récoltées par un algorithme de bandit contextuel ayant déjà terminé son apprentissage, et cela, sans nécessiter d'exploration explicite. Sous certaines hypothèses, la randomisation des contextes peut en effet induire une exploration explicite de l'espace des contextes et des récompenses.

Bibliographie

[1] Vladimir N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[2] David A. McAllester. Pac-bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, COLT '99, pages 164–170, New York, NY, USA, 1999. ACM.

[3] Edward Lee Thorndike. *Animal Intelligence*. 1911.

[4] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[5] Richard S. Sutton. Learning to predict by the methods of temporal differences. In *MACHINE LEARNING*, pages 9–44. Kluwer Academic Publishers, 1988.

[6] Christopher J. C. H. Watkins. *Learning from Delayed Rewards*. PhD thesis, King's College, Cambridge, UK, 1989.

[7] W.R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25 :285–294, 1933.

[8] T. L. Lai and Herbert Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1) :4–22, 1985.

[9] Leslie G. Valiant. A theory of the learnable. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 436–445, 1984.

[10] Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. Best arm identification : A unified approach to fixed budget and fixed confidence. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *NIPS*, pages 3221–3229, 2012.

[11] Sébastien Bubeck, Rémi Munos, and Gilles Stoltz. Pure exploration in multi-armed bandits problems. In *Proceedings of the 20th International Conference on Algorithmic Learning Theory*, ALT'09, pages 23–37, Berlin, Heidelberg, 2009. Springer-Verlag.

[12] Shie Mannor, John N. Tsitsiklis, Kristin Bennett, and Nicolò Cesa-bianchi. The sample complexity of exploration in the multi-armed bandit problem. *Journal of Machine Learning Research*, 5 :2004, 2004.

[13] Eyal Even-dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *In Fifteenth Annual Conference on Computational Learning Theory (COLT*, pages 255–270, 2002.

[14] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7 :1079–1105, 2006.

[15] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301) :13–30, March 1963.

[16] S. Bubeck and N. Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. In *Foundations and Trends in Machine Learning*, 2012.

[17] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3) :235–256, 2002.

[18] Odalric-Ambrym Maillard, Rémi Munos, and Gilles Stoltz. Finite-time analysis of multi-armed bandits problems with kullback-leibler divergences. In *Proceedings of the 24th annual Conference On Learning Theory*, COLT 2011, 2011.

[19] Emilie Kaufmann, Aurélien Garivier, and Telecom Paristech. On bayesian upper confidence bounds for bandit problems. In *In AISTATS*, 2012.

[20] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling : An asymptotically optimal finite-time analysis. In NaderH. Bshouty, Gilles Stoltz, Nicolas Vayatis, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, volume 7568 of *Lecture Notes in Computer Science*, pages 199–213. Springer Berlin Heidelberg, 2012.

[21] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Annual Conference on Learning Theory (COLT)*, June 2012.

[22] Akram Baransi, Odalric-Ambrym Maillard, and Shie Mannor. Sub-sampling for multi-armed bandits. In Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, editors, *Machine Learning and Knowledge Discovery in Databases*, volume 8724 of *Lecture Notes in Computer Science*, pages 115–131. Springer Berlin Heidelberg, 2014.

[23] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for non-stationary bandit problems. In *Algorithmic Learning Theory*, pages 174–188, Oct. 2011.

[24] P. Whittle. Restless bandits : Activity allocation in a changing world. *Journal of Applied Probability*, 1988.

[25] Sudipto Guha, Kamesh Munagala, and Peng Shi. Approximation algorithms for restless bandit problems. *J. ACM*, 2010.

[26] Sarah Filippi. *Optimistic strategies in Reinforcement Learning*. Theses, Ecole nationale supérieure des telecommunications - ENST, November 2010.

[27] L. Kocsis and C. Szepesvàri. Discounted ucb. In *2nd PASCAL Challenges Workshop, Venice, Italy, April*, 2006.

[28] C. Hartland, N. Baskiotis, S. Gelly, O. Teytaud, and M. Sebag. Multi-armed bandit, dynamic environments and meta-bandits. In *Online Trading of Exploration and Exploitation Workshop, NIPS*, Whistler, Canada, December 2006.

[29] E. S. Page. Continuous inspection schemes. *Biometrika*, 41(1/2) :100–115, 1954.

[30] Peter Auer and Nicolò Cesa-Bianchi. On-line learning with malicious noise and the closure algorithm. *Ann. Math. Artif. Intell.*, 23(1-2) :83–99, 1998.

[31] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1) :48–77, 2002.

[32] Gergely Neu. Explore no more : improved high-probability regret bounds for non-stochastic bandits. *NIPS*, 2015.

[33] Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds : stochastic and adversarial bandits. *CoRR*, abs/1202.4473, 2012.

[34] Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *31th Intl. Conf. on Machine Learning (ICML)*, 2014.

[35] Robert D. Kleinberg, Alexandru Niculescu-Mizil, and Yogeshwer Sharma. Regret bounds for sleeping experts and bandits. In *COLT*, pages 425–436, 2008.

[36] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *NIPS*, pages 273–280, 2008.

[37] Raphaël Feraud and Tanguy Urvoy. A stochastic bandit algorithm for scratch games. In *ACML*, pages 129–143, 2012.

[38] R.J. Serfling. Probability inequalities for the sum in sampling without replacement. In *The Annals of Statistics, Vol 2, No.1, pages = 39–48, year = 1974,*.

[39] Jia Yuan Yu and Shie Mannor. Piecewise-stationary bandit problems with side observations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1177–1184, New York, NY, USA, 2009. ACM.

[40] John Langford and Tong Zhang. The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*, 2007.

[41] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. *CoRR*, 2010.

[42] Joaquin Quionero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D. Lawrence. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

[43] Sham M. Kakade, Shai Shalev-Shwartz, and Ambuj Tewari. Efficient bandit algorithms for online multiclass prediction. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 440–447, New York, NY, USA, 2008. ACM.

[44] Frank Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6) :386–408, 1958.

[45] Wei Chu, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, pages 208–214, 2011.

[46] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 2312–2320, 2011.

[47] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. *CoRR*, 2012.

[48] Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. *CoRR*, abs/1106.2369, 2011.

[49] Alekh Agarwal, Daniel Hsu, Satyen Kale, John Langford, Lihong Li, and Robert E. Schapire. Taming the monster : A fast and simple algorithm for contextual bandits. In *The 31st International Conference on Machine Learning (ICML 2014)*. JMLR : Workshop and Conference Proceedings, June 2014.

[50] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5) :359–366, July 1989.

[51] J. Schmidhuber. Deep learning in neural networks : An overview. *Neural Networks*, 61 :85–117, 2015. Published online 2014 ; based on TR arXiv :1404.7828 [cs.NE].

[52] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet : A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[53] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, 2013.

[54] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529 :484–503, 2016.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing : Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[56] Léon Bottou and Yann LeCun. On-line learning for very large datasets. *Applied Stochastic Models in Business and Industry*, 21(2) :137–151, 2005.

[57] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4 :40–79, 2010. Published in Statistics Surveys (2010) 4, 40-79.

[58] M. Lichman. UCI machine learning repository, 2013.

[59] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 71–80, New York, NY, USA, 2000. ACM.

[60] Leszek Rutkowski, Lena Pietruczuk, Piotr Duda, and Maciej Jaworski. Decision trees for mining data streams based on the mcdiarmid's bound. *IEEE Trans. Knowl. Data Eng.*, 25(6) :1272–1279, 2013.

[61] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 97–106, New York, NY, USA, 2001. ACM.

[62] Vianney Perchet and Philippe Rigollet. The multi-armed bandit problem with covariates. *Ann. Statist.*, 41(2) :693–721, 04 2013.

[63] Leo Breiman. Random forests. *Mach. Learn.*, 45(1) :5–32, October 2001.

[64] Antanas Verikas, Adas Gelzinis, and Marija Bacauskiene. Mining data with random forests : A survey and results of new tests. *Pattern Recognition*, 44(2) :330–349, 2011.

[65] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260) :663–685, 1952.

[66] E. V. SSlud. Distribution inequalities for the binomial law. *Ann. Probab.*, 1977.

[67] url = https ://ece.uwaterloo.ca year = "2010" N Mousavi., title = How tight is Chernoff bound ?

[68] John T. Chu. On bounds for the normal integral. *Biometrika*, 42, 1955.

[69] M. Soare, A. Lazaric, and R. Munos. Best-arm identification in linear bandits. In *NIPS*, 2014.

[70] J. Langford and A. Strehl. Exploration scavenging. In *ICML*, 2008.

[71] Shang-Tse Chen, Hsuan-Tien Lin, and Chi-Jen Lu. An online boosting algorithm with theoretical justifications. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

Liste des Algorithmes

Table des figures