NEURALRANDUCB : RANDOMISATION DE NEURALUCB

RAPPORT DE PROJET

Alexandre Bouras
111 149 065

Louis Dubois
111 045 885

17 décembre 2021

1 Introduction

L'environnement traditionnel de bandit (*multi-armed bandit*) [1] suppose que la récompense ne dépend pas du contexte, celui-ci étant fixe. Cependant, dans de nombreuses applications concrètes, le problème, et la récompense associée à une action donnée, dépend directement du contexte. En effet, une telle formulation du problème de bandits peut s'appliquer à la recommandation d'articles de nouvelles, de publicités ou même de traitements [2]. Cet environnement de bandits a été largement étudié dans la littérature, particulièrement en supposant une fonction de récompense linéaire en fonction des *features* du contexte [3] [4] [5]. Cependant, l'hypothèse de la linéarité de la fonction de récompense ne tient pas toujours en pratique. En effet, il peut être intéressant de considérer la situation où celle-ci est non linéaire en fonction des *features* de contexte. Plusieurs méthodes ont été étudiées, notamment les bandits non-paramétriques [6], l'utilisation de processus gaussiens [7] ou de méthodes à noyaux [8] ainsi que les *neural bandits* [9] [10] [11] [12]. Ces derniers algorithmes amènent la non-linéarité avec l'utilisation d'un réseau de neurones artificiels pour approximer la fonction de récompense; ceci est d'intérêt puisque les réseaux de neurones sont des approximateurs de fonction universels [13]. Ainsi, voyant qu'il est possible d'utiliser des réseaux de neurones en appliquant des stratégies d'exploration traditionnelles, dont UCB avec NeuralUCB [11], nous investiguons la randomisation de cet algorithme en échantillonnant une variable aléatoire Z à partir d'une distribution donnée, tel que présenté pour l'algorithme RandUCB [14]. Dans ce travail, nous abordons donc le problème de l'utilisation des réseaux de neurones dans l'environnement des bandits contextuels lorsque la fonction de récompense est non-linéaire en fonction des *features* de contexte. De plus, nous examinons l'ajout d'un paramètre de bonus d'exploration qui est échantillonné à partir d'une distribution afin de créer une version randomisée de NeuralUCB, soit l'algorithme NeuralRandUCB. Tout d'abord, nous voulons observer la performance de notre algorithme en fonction des différentes distributions à partir de laquelle est échantillonné la variable aléatoire. Nous effectuons cette évaluation dans un environnement synthétique, avec une fonction de récompense polynomiale. Ensuite, nous voulons déterminer si l'ajout de la randomisation permet d'obtenir de meilleures performances que NeuralUCB et NeuralTS dans notre environnement synthétique, ainsi que sur certaines des tâches de classification présentées dans [12]. Par ailleurs, nous voulons évaluer la robustesse de notre algorithme au délai des récompenses au sein des mêmes environnements. Cette dernière propriété est importante, car dans plusieurs applications, la rétroaction de l'environnement n'est souvent pas immédiate, mais arrive plutôt en *batchs* sur une certaine période. On s'attend donc à ce que les algorithmes randomisés soient plus robustes à un tel délai [15]. Finalement, nous effectuons une expérience préliminaire en utilisant NeuralUCB, NeuralTS et NeuralRandUCB dans un environnement de prise de décision plus appliqué, soit la recommandation d'articles de nouvelles.

2 Bandits contextuels à actions disjointes

Le problème que nous considérons est celui des bandits contextuels à K actions disjointes où le nombre de pas de temps T est connu à l'avance. Ce problème se décrit par l'observation, à chaque pas de temps $t \in T$, du contexte par l'agent. Le contexte se compose de K vecteurs de *features* $\mathbf{x}_{t,a} \in \mathbb{R}^d$, avec $a \in K$ et où chaque action est liée à un contexte. Par la suite, l'agent choisit une action a_t et reçoit une récompense r_{t,a_t}. Le but de l'agent est de minimiser le pseudo-regret (ou regret, pour simplifier) cumulatif : $R_T = \mathbb{E}\big[\sum_{t=1}^{T}(r_{t,a_t^*} - r_{t,a_t})\big]$, où a_t^* est l'action ayant la récompense espérée la plus élevée au pas de temps t. De plus, la formulation traditionnelle de ce problème suppose également que la récompense a la forme suivante : $r_{t,a_t} = f(\mathbf{x}_{t,a_t}) + \eta_t$, où f est une fonction inconnue satisfaisant $0 \leq f(\mathbf{x}) \leq 1$ pour tout vecteur de contexte $\mathbf{x}$ et η_t est un bruit sous-Gaussien satisfaisant $\mathbb{E}\big[\eta_t|\mathbf{x}_{1:t}, \eta_{1:t-1}\big] = 0$ [5].

3 NeuralRandUCB

Algorithm 1: NeuralRandUCB

Input : Nombre de pas de temps T, réseau de neurones de largeur m et de profondeur L.

Initialiser $\mathbf{U}_0 \leftarrow \mathbf{I}$

Initialiser $\theta_0 \leftarrow (vec(\mathbf{W}_1; ...; vec(\mathbf{W}_L)) \in \mathbb{R}^p$, où pour chaque $1 \leq l \leq L-1$, $\mathbf{W}_p = (\mathbf{W}, \mathbf{0}, \mathbf{0}, \mathbf{W})$

for $t \leftarrow 1$ **to** T **do**

  $Z_t \sim$ distribution discrète

  for $a \leftarrow 1$ **to** K **do**

    Observation du contexte $\mathrm{x}_{t,a}$

    $U_{t,a} \leftarrow f(\mathbf{x}_{t,a}; \theta_{t-1}) + Z_t \sqrt{\mathbf{g}(\mathbf{x}_{t,a}; \theta_{t-1})^T \mathbf{V}_{t-1}^{-1} \mathbf{g}(\mathbf{x}_{t,a}; \theta_{t-1})/m}$

  end for

  Jouer l'action a_t, où $a_t \leftarrow \mathrm{argmax}_a U_{t,a}$

  Observer la récompense r_{t,a_t}

  $\theta_t \leftarrow$ sortie du réseau de neurones pour résoudre $\min_\theta L(\theta) = \sum_{i=1}^{t} [f(\mathbf{x}_{i,a_i}; \theta) - r_{i,a_i}]^2 + m\lambda ||\theta - \theta_\mathbf{0}||_2^2/2$

  $\mathbf{V}_t \leftarrow \mathbf{V}_{t-1} + \mathbf{g}(\mathbf{x}_{t,a_t}; \theta_t)\mathbf{g}(\mathbf{x}_{t,a_t}; \theta_t)^T/m$

end for

L'algorithme que nous proposons est fortement inspiré de NeuralUCB, dont l'idée principale est l'utilisation d'un réseau de neurones $f(\mathbf{x}; \theta)$ afin d'effectuer l'estimation de la récompense du contexte $\mathbf{x}$ et du gradient $\mathbf{g}(\mathbf{x}; \theta)$ afin d'effectuer le calcul de la borne supérieure [11]. Le réseau de neurones est défini de la façon suivante : $f_1 = \mathbf{W}_1\mathbf{x}$, $f_l = \mathbf{W}_l$ ReLU(f_{l-1}) (où $2 \leq l \leq L$), $f(\mathbf{x}; \theta) = \sqrt{m} f_L$, avec ReLU$(x) := \max\{x, 0\}$ et $\theta = (vec(\mathbf{W}_1; ...; vec(\mathbf{W}_P)) \in \mathbb{R}^p$ est l'ensemble des paramètres du réseau de neurones, $p = dm + m^2(L-2) + m$. Une fois l'action optimale choisie et la récompense observée, on utilise le réseau de neurones afin de mettre-à-jour les paramètres θ, qui représentent la solution du problème de minimisation de la fonction de perte $L(\theta)$. Notre algorithme apporte la randomisation avec l'ajout de la variable aléatoire Z, où $Z_1, ..., Z_T$ sont des échantillons i.i.d. échantillonnés à partir d'une distribution discrète en M points équidistants dans l'intervalle $[L, U]$ et où $\alpha_1 = L, ..., \alpha_M = U$ [14]. Lorsque nous avons $L = 0$ nous ne considérons alors que des valeurs positives de Z, conservant ainsi le principe *OFU* [5] [14]. Si $L = U$, alors nous retrouvons NeuralUCB, dont la borne de confiance est paramétrée par une constante lors des expériences présentées dans [11]. Il est également possible de découpler l'échantillonnage de Z_t, ce qui implique d'avoir une variable aléatoire pour chacune des K actions à chacun des pas de temps. On obtient alors un niveau de confiance différent par action. On observe qu'en effectuant un découplement avec une valeur de $L < 0$, on se rapproche du comportement du Thompson Sampling [14]. La variable Z permet donc d'ajuster le compromis exploration/exploitation à chaque pas de temps [14].

4 Expériences et résultats

4.1 Méthodologie expérimentale

Pour implémenter l'aglorithme NeuralRandUCB, nous avons utilisé et adapté le code[1] [2] fournit par [12], qui contient une implémentation de NeuralUCB ainsi que NeuralTS, et [14] pour l'ajout de la randomisation. Dans ce travail, nous avons utilisé la même architecture de réseau de neurones que les auteurs de NeuralTS [12], soit un réseau pleinement connecté d'une couche cachée de 100 neurones. Cette architecture a été utilisée à la fois pour NeuralRandUCB et les *baselines* NeuralUCB et NeuralTS. Pour optimiser le réseau de neurones, nous avons utilisé l'algorithme Adam [16] avec un taux d'apprentissage de 10^{-2} plutôt que l'algorithme SGD comme les auteurs de [12], car nous avons observé des meilleurs résultats avec Adam. Pour comparer les performances de NeuralRandUCB avec d'autres algorithmes, nous avons considéré les *baselines* suivantes : LinUCB ($\lambda = 1, \nu = 1$), LinTS ($\lambda = 1, \nu = 10^{-2}$), KernelUCB ($\lambda = 1, \nu = 1$), KernelTS ($\lambda = 1, \nu = 10^{-2}$), NeuralUCB ($\lambda = 1, \nu = 10^{-1}$) et NeuralTS ($\lambda = 1, \nu = 10^{-2}$). Les valeurs des hyperparamètres λ et ν considérées pour trouver les meilleures paires sont celles présentées dans [12]. Pour les algorithmes linéaires et à noyau, la sélection a été effectuée avec une recherche en grille. Pour les algorithmes neuronaux, nous avons plutôt procédé à tâtons, en raison du temps important pour réaliser les entraînements. Pour tester l'approche proposée nous avons considéré des instances de bandits contextuels à actions disjointes avec récompenses décrites par des fonctions polynomiales d'ordre 5 auxquelles un bruit d'écart-type de 0.1 a été ajouté. Plus précisément, nous avons considérés 4 actions disjointes ainsi qu'un espace de contexte à 5 dimensions. Nous avons aussi testé les

[1] https://github.com/ZeroWeight/NeuralTS

[2] https://github.com/vaswanis/randucb

algorithmes sur les jeux de données de classification *Mushroom* et *Shuttle* [17], des tâches utilisées pour l'évaluation de NeuralTS [12], à partir desquels des données ont été sélectionnées au-hasard. Pour chaque expérience réalisée, nous avons effectué plusieurs répétitions avec des instances différentes (générées avec différents *seeds*), les mêmes pour chaque algorithme, afin de comparer les performances de manière équitable et d'assurer la reproductibilité des résultats.

4.2 Expérience 1: Performance de NeuralRandUCB selon différentes configurations

Cette expérience consiste à évaluer la performace de NeuralRandUCB selon plusieurs configurations de la distribution de la variable Z, et constater s'il est possible de retrouver le comportement de NeuralTS avec une configuration particulière. Nous avons considéré les trois configurations suivantes : 1. Exploration couplée, optimiste, distribution uniforme discrétisée en 20 points; 2. Exploration découplée, réaliste, distribution normale ($\sigma = 10^{-1}$) discrétisée en 20 points; 3. Exploration couplée, réaliste, distribution normale ($\sigma = 1/16$) discrétisée en 20 points. La figure 1 présente le regret cumulatif moyen obtenu sur l'environnement de bandits synthétique, pour 2000 pas de temps et en faisant 8 répétitions pour chaque algorithme. On remarque qu'il est possible de retrouver le comportement de NeuralTS ($\nu = 10^{-2}$) avec la configuration 1 de NeuralRandUCB. La configuration 3 permet d'obtenir les meilleurs résultats, c'est donc cette configuration qui est retenue pour réaliser les autres expériences de ce travail.

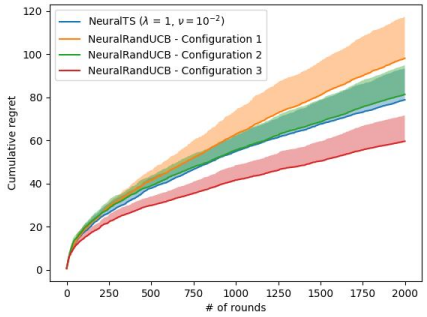

Figure 1: Performance de NeuralRandUCB sur l'environnement synthétique avec diverses configurations.

4.3 Expérience 2: Performance de NeuralRandUCB comparativement à plusieurs algorithmes

Afin de bien évaluer les performances de NeuralRandUCB, nous le comparons aux *baselines* mentionnées dans la section 4.1 sur les trois environnements de bandits considérés dans ce travail. Pour chacun des algorithmes, 5 répétions ont été effectuées avec 6000 pas de temps, à chaque fois avec une instance différente (les mêmes pour chaque algorithme). La figure 2 présente le regret cumulatif moyen avec un écart-type ajouté. On remarque que NeuralRandUCB obtient le plus faible regret cumulatif, suivi de près par NeuralUCB et NeuralTS. Des résultats similaires, présentés en annexe, ont également été obtenus sur les jeux de données *Mushroom* et *Shuttle*.

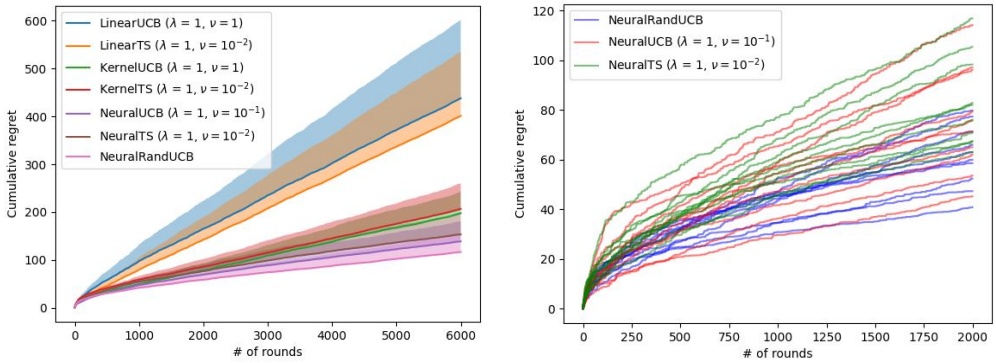

Figure 2: Performance de NeuralRandUCB et des *baselines* sur l'environnement synthétique de bandits (à gauche). Courbes de regrets obtenues pour NeuralRandUCB, NeuralUCB et NeuralTS pour chacune des 10 répétions réalisées sur l'environnement de bandits synthétique (à droite).

Pour vérifier plus rigoureusement les performances de NeuralRandUCB par rapport à NeuralUCB et NeuralTS, on réalise une analyse statistique simple, en regardant individuellement chacune des répétitions effectuées sur l'environnement synthétique. On s'intéresse ensuite au nombre de fois que NeuralRandUCB performe mieux que les deux autres algorithmes, ainsi qu'à sa pire et meilleure performance. Concrètement, on réalise 10 répétions sur des instances différentes en considérant un horizon $T = 2000$, ce qui est présenté à la figure 2. Pour chacune des répétitions, NeuralRandUCB obtient un regret cumulatif plus faible que NeuralUCB et NeuralTS.

4.4 Expérience 3: Robustesse de NeuralRandUCB au délai dans les récompenses

Cette expérience vise à simuler un scénario pratique, dans lequel les signaux de récompenses sont retardés. On vérifie la robustesse des meilleurs algorithmes de l'Expérience 2, soit NeuralRandUCB, NeuralUCB et NeuralTS, lorsqu'il y a un délai dans les récompenses, en réalisant des mises à jour en *batches*. Comme pour l'Expérience 2, on réalise 5 répétitions sur des instances différentes de bandits synthétiques avec un horizon $T = 6000$. D'après la figure 3, il ne semble pas y avoir de réel avantage à utiliser de la randomisation, les trois algorithmes ayant environ la même robustesse au délai. Cependant, ceci pourrait être expliqué par le fait que la configuration retenue de NeuralRandUCB n'effectue pas d'échantillonnage découplé de la variable aléatoire Z, afin d'obtenir une borne de confiance distincte pour chacune des actions. Ceci semble aider à la robustesse, puisque NeuralTS, qui calcule une borne pour chaque action, obtient, de façon générale, une meilleure performance.

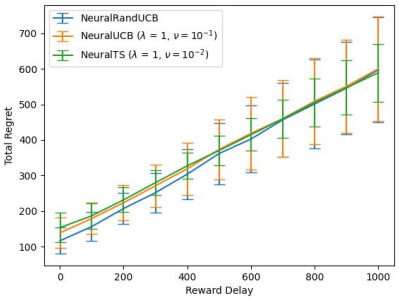

Figure 3: Robustesse au délai des récompenses pour NeuralRandUCB, NeuralUCB et NeuralTS dans l'environnement synthétique.

4.5 Expérience 4: Recommandation d'articles de nouvelles avec NeuralRandUCB

En utilisant un jeu de données de *Yahoo*[3] (1.6 millions d'évènements et 45 articles au total), on réalise une expérience préliminaire afin d'évaluer la performance des algorithmes de bandits contextuels neuronaux sur la tâche de recommandation d'articles de nouvelles. Pour réaliser l'expérience, on utilise la même méthodologie que [18], et on compare les trois *neural bandits* à l'implémentation de LinUCB, en utilisant la meilleure valeur d'hyperparamètre qu'ils rapportent ($\alpha = 0.3$)[4]. Le tableau 1, présente les valeurs de taux de clics obtenues pour chaque algorithme. Bien que les *neural bandits* donnent des résultats satisfaisants, la performance de LinUCB demeure supérieure. Une explication possible serait que la paramétrisation et l'architecture du réseau utilisées ne soient pas optimales pour cette tâche différente des autres étudiées dans ce travail. Une autre explication peut être que la configuration retenue de NeuralRandUCB n'effectue pas d'échantillonnage découplé de la variable aléatoire Z, diminuant ainsi sa robustesse au délai des récompenses. Cette propriété pourrait s'avérer importante pour une tâche telle que la recommandation d'articles de nouvelles.

Algorithme	CTR (%)
LinUCB	**6.28**
NeuralRandUCB	5.22
NeuralUCB	5.28
NeuralTS	5.91

Table 1: Valeurs de taux clics (CTR) obtenues sur le *Yahoo! Front Page Today Module User Click Log Dataset*

5 Retour sur les objectifs et travaux futurs

Nous considérons que nous avons atteint les objectifs que nous nous étions fixés. En effet, nous avons réussi à effectuer une randomisation de l'algorithme NeuralUCB. De plus, nous réussissons à retrouver certains des avantages de cette randomisation, soit la capacité d'adapter la distribution d'échantillonnage de la variable aléatoire Z au problème considéré et la robustesse au délai des récompenses, quoi que des expériences supplémentaires sont nécessaires pour valider qu'un échantillonnage découplé améliore la robustesse. Finalement, nous avons effectué une évaluation préliminaire de la performance de NeuralUCB, NeuralTS et NeuralRandUCB dans une application plus concrète de prise de décision, soit la recommandation d'articles de nouvelles. Les résultats de cette expérience sont mitigés et exigent une exploration plus approfondie que nous réservons pour nos travaux futurs. Nous avons donc démontré l'intérêt d'utiliser les réseaux de neurones dans une approche n'utilisant que des actions disjointes dans des environnements de bandits contextuels, que ce soit avec l'algorithme NeuralTS, NeuralUCB ou notre version randomisée, NeuralRandUCB. Cet environnement est d'une application limitée pour des cas réels, où le nombre d'actions possible à chaque pas de temps peut être très grand ou quasi infini. Ainsi, nous voulons également explorer l'exploitation de la structure entre les actions de notre algorithme NeuralRandUCB.

[3] R6A - Yahoo! Front Page User Click Log Dataset : https://webscope.sandbox.yahoo.com/catalog.php?datatype=rdid=49
[4] https://github.com/antonismand/Personalized-News-Recommendation

References

[1] T. L. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in applied mathematics*, vol. 6, no. 1, pp. 4–22, 1985.

[2] M. Dudik, D. Hsu, S. Kale, N. Karampatziakis, J. Langford, L. Reyzin, and T. Zhang, "Efficient optimal learning for contextual bandits," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, UAI'11, (Arlington, Virginia, USA), p. 169–178, AUAI Press, 2011.

[3] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th international conference on World wide web*, pp. 661–670, 2010.

[4] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudík, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 208–214, PMLR, 11–13 Apr 2011.

[5] Y. Abbasi-yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.

[6] S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári, "Parametric bandits: The generalized linear case," in *Advances in Neural Information Processing Systems* (J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, eds.), vol. 23, Curran Associates, Inc., 2010.

[7] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (Madison, WI, USA), p. 1015–1022, Omnipress, 2010.

[8] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini, "Finite-time analysis of kernelised contextual bandits," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, UAI'13, (Arlington, Virginia, USA), p. 654–663, AUAI Press, 2013.

[9] C. Riquelme, G. Tucker, and J. Snoek, "Deep bayesian bandits showdown," in *International conference on learning representations*, 2018.

[10] T. Zahavy and S. Mannor, "Deep neural linear bandits: Overcoming catastrophic forgetting through likelihood matching," *arXiv preprint arXiv:1901.08612*, 2019.

[11] D. Zhou, L. Li, and Q. Gu, "Neural contextual bandits with UCB-based exploration," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 11492–11502, PMLR, 13–18 Jul 2020.

[12] W. ZHANG, D. Zhou, L. Li, and Q. Gu, "Neural thompson sampling," in *International Conference on Learning Representations*, 2021.

[13] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[14] S. Vaswani, A. Mehrabian, A. Durand, and B. Kveton, "Old dog learns new tricks: Randomized ucb for bandit problems," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics* (S. Chiappa and R. Calandra, eds.), vol. 108 of *Proceedings of Machine Learning Research*, pp. 1988–1998, PMLR, 26–28 Aug 2020.

[15] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Advances in Neural Information Processing Systems* (J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Q. Weinberger, eds.), vol. 24, Curran Associates, Inc., 2011.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[17] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[18] L. Li, W. Chu, J. Langford, and X. Wang, "Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 297–306, 2011.

A Résultats supplémentaires

A.1 Performances de NeuralRandUCB et des baselines sur les jeux de données *Mushroom* et *Shuttle*.

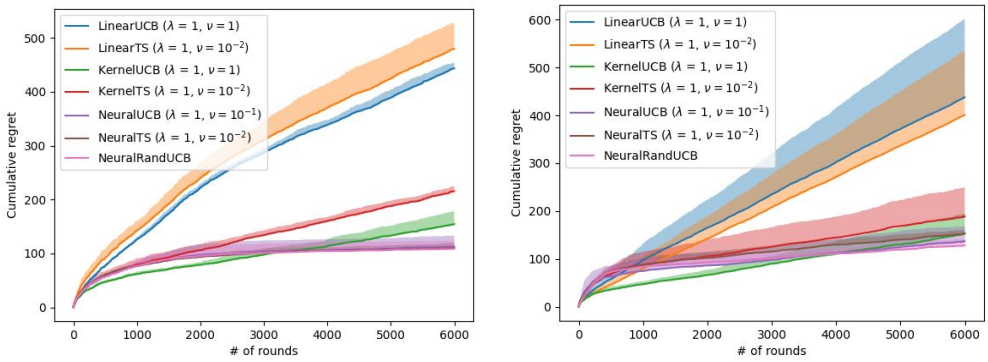

Figure 4: Performance des algorithmes sur *Mushroom* (à gauche) et *Shuttle* (à droite). On observe que NeuralRandUCB est l'algorithme qui performe le mieux pour *Shuttle* et parmi les meilleurs pour *Mushroom*.

A.2 Robustesse de NeuralRandUCB, NeuralUCB et NeuralTS au délai dans les récompenses sur les jeux de données *Mushroom* et *Shuttle*.

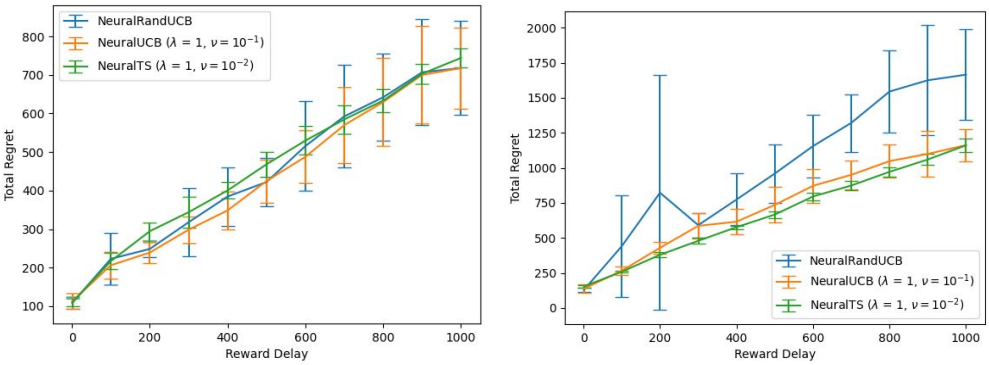

Figure 5: Performance des algorithmes sur *Mushroom* (à gauche) et *Shuttle* (à droite). On observe que NeuralRandUCB performe relativement bien pour le jeu de données *Mushroom*, mais à la pire performance pour le jeu de données *Shuttle*. Ceci peut s'expliquer par le fait que la configuration de NeuralRandUCB retenue n'effectue pas d'échantillonnage découplé de Z pour chacune des actions. En effet, il est possible que d'avoir une borne de confiance distincte pour chacune des actions soit important pour obtenir des performances robustes, tel qu'on le remarque pour NeuralTS.