A Neural Networks Committee for the Contextual Bandit Problem

Robin Allesiardo[1,2], Raphaël Féraud[2], and Djallel Bouneffouf[2]

[1] TAO - INRIA, CNRS, University of Paris-Sud, 91405 Orsay (France)
[2] Orange Labs, 2 av. Pierre Marzin, 22300 Lannion (France)

Abstract. This paper presents a new contextual bandit algorithm, NeuralBandit, which does not need hypothesis on stationarity of contexts and rewards. Several neural networks are trained to modelize the value of rewards knowing the context. Two variants, based on multi-experts approach, are proposed to choose online the parameters of multi-layer perceptrons. The proposed algorithms are successfully tested on a large dataset with and without stationarity of rewards.

1 Introduction

In online decision problems such as online advertising or marketing optimization, a decision algorithm must select amoung several actions. Each of these options is associated with side information (profile or context) and the reward feedback is limited to the chosen option. For example, in online advertising, a visitor queries a web page; a request with the context (web page address, cookies, customer profile, etc.) is send to the server; the server sends an ad which is displayed on the page. If the visitor clicks on the ad the server receives a reward. The server must trade-off between the explorations of new ads and the exploitation of known ads. Moreover, in an actual applications, both rewards and data distributions can change with time. For instance, the display of a new ad can change the probability of clicks of all ads, the content of a web page can change over time. Robustness to non-stationarity is thus strongly recommended.

2 Previous work

The multi-armed bandit problem is a model of exploration and exploitation where one player gets to pick within a finite set of decisions the one which maximizes the cumulated reward. This problem has been extensively studied. Optimal solutions have been provided using a stochastic formulation [1, 2], a Bayesian formulation [3, 4], or using an adversarial formulation [5, 6]. Variants of the initial problem were introduced due to practical constraints (appearance of a new advertisment after the beginning of learning [7, 8], fixed number of contractual page views [9, 10]). However these approaches do not take into account the context while the arm's performance may be correlated therewith.

A naive solution to the contextual bandit problem is to allocate one bandit problem for each context. A tree structured bandit such as X-armed bandits [11] or a UCT variant [12] can be used to explore and exploit a tree structure of contextual variables to find the leaves which provide the highest rewards [13]. The combinatorial aspect of these approaches limits their use to small context size. Seldin et al [14] modelize the contexts by state sets, which are associated with bandit problems. Without prior knowledge of the contexts, it is necessary to use a state per context, which is equivalent to the naive approach. Dudík et al [15] propose an algorithm of policies elimination. The performance of this algorithm depends on the presence of a good policy in the set. The epoch-greedy algorithm [16] alternates exploration then exploitation. During exploration, the arms to play are randomly drawn to collect an unbiased training set. Then, this set is used to train a classifier which will be used for exploitation during the next cycle. The nature of the classifier remains to be defined for concrete use. In LINUCB [17, 18] and in Contextual Thompson Sampling (CTS) [19], the authors assume a linear dependency between the expected reward of an action and its context and model the representation space using a set of linear predictors. Banditron [20] uses a perceptron per action to recognize rewarded contexts. Furthermore, these algorithms assume that the data and the rewards are drawn from stationary distribution, which limits their practical use. The EXP4 algorithm [6] selects the best arm from N experts advices (probability vectors). Exploring the link between the rewards of each arm and the context is delegated to experts. Unlike the previous algorithms, the rewards are assumed to be chosen in advance by an adversary. Thus, this algorithm can be applied to non-stationnary data.

At first we will formalize the contextual bandit problem and will propose a first algorithm: NeuralBandit1. Inspired by Banditron [20] it estimates the probabilities of rewards by using neural networks in order to be free of the hypothesis of linear separability of the data. Neural networks are universal approximators [21]. They are used in reinforcement learning [22, 23] and can estimate accurately the probabilities of rewards within actual and complex problems. In addition, the stochastic gradient achieves good performances in terms of convergence to the point of best generalization [24] and has the advantage of learning online. In seeking to reach a local minimum, the stochastic gradient can deal with non-stationarity. This will result in a change of the cost function landscape over the time. If this landscape changes at a reasonable speed, the algorithm will continue the descent to a new local minimum. The main issue raised by the use of multilayer perceptrons remains the online setting of various parameters such as the number of hidden neurons, the value of the learning step or the initalization of the weight seed. We propose two advanced versions of the algorithm NeuralBandit1 to adjust these settings using adversarial bandits that seek, among several models initialized with different parameters, the best one. We conclude by comparing these different approaches to the state-of-the-art on stationnary and on non-stationnary data.

3 Our algorithm: NeuralBandit1

Definition 1 (Contextual bandit). *Let $x_t \in X$ be a contextual vector and $(y_{t,1}, ..., y_{t,K})$ a vector of rewards associated with the arm $k \in [K] = \{1, ..., K\}$ and $((x_1, \boldsymbol{y}_1), ..., (x_T, \boldsymbol{y}_T))$ the sequence of contexts and rewards. The sequence can be drawn from a stochastic process or chosen in advance by an adversary. At each round $t < T$, the context x_t is announced. The player, who aims to maximize his cumulated rewards, chooses an arm k_t. The reward $y_{t,k}$ of the played arm, and only this one, is revealed.*

Definition 2 (Cumulated regret). *Let $H : X \rightarrow [K]$ be a set of hypothesis , $h_t \in H$ a hypothesis computed by the algorithm A at the round t and $h_t^* = \underset{h_t \in H}{\operatorname{argmax}}\, y_{t,h_t(x_t)}$ the optimal hypothesis at the same round. The cumulated regret is:*

$$R(A) = \sum_{t=1}^{T} y_{t,h_t^*(x_t)} - y_{t,h_t(x_t)}$$

The purpose of a contextual bandit algorithm is to minimize the cumulative regret.

Each action k is associated with a neural network with one hidden layer which learns the probability of reward for an action knowing its context. We choose this modelization rather than one neural network with as many outputs as actions to be able to add or remove actions easier.

Let K be the number of actions, C the number of neurons of each hidden layers and $\mathbf{N}_t^k : X \rightarrow Y$ the function associating a context x_t to the output of the neural network corresponding to the action k at the round t. N denotes the number of connections of each network with $N = dim(X)C + C$. To simplify the notations, we place the set of connections in the matrix W_t of size $K \times N$. Thus, each row of the matrix W_t contains the weight of a network. Δ_t is the matrix of size $K \times N$ containing the update of each weight between rounds t and $t+1$. The update equation is:

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

The backpropagation algorithm [25] allows calculating the gradient of the error for each neuron from the last to the first layer by minimizing a cost function. Here, we use the quadratic error function and a sigmoid activation function.

Let λ be the learning step, $\hat{x}_t^{n,k}$ the input associated with the connection n in the network k, $\delta_t^{n,k}$ the gradient of the cost function at round t for the neuron having as input the connection n in the network k and $\Delta_t^{n,k}$ the value corresponding to the index (n, k) of the matrix Δ_t. When the reward of an arm is known, we can compute:

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

In the case of partial information, only the reward of the arm k_t is available. To learn the best action to play, an approach consists of a first exploration phase, where each action is played the same number of times in order to train a model, and then an exploitation phase where the obtained model is used. Thus, the estimator would not be biased on the most played action. However, this approach would have abysmal performances in case of non-stationary data. We choose to use an exploration factor γ, constant over time, allowing continuing the update of the model in the case of non-stationary data. The probability of playing the action k at round t knowing that $\hat{k}_t$ is the arm with the highest reward prediction is:

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

We propose a new update rule taking into account the exploration factor:

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

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

Theorem 1. *The expected value of* $\tilde{\Delta}_t^{n,k}$ *is* $\Delta_t^{n,k}$.

The proof is straightforward:

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

□

The proposed algorithm, NeuralBandit1, can adapt to non-stationarity by continuing to learn over time, while achieving the same expected result (in the case of stationary data) as a model trained in a first phase of exploration.

Algorithm 1: NeuralBandit1

```
Data: $\gamma \in [0, 0.5]$ et $\lambda \in ]0, 1]$
begin
    Initialize $W_1 \in ] - 0.5, 0.5[^{N \times K}$
    for $t = 1, 2, ..., T$ do
        Context $x_t$ is revealed
        $\hat{k}_t = \arg\max_{k \in [K]} \mathbf{N}_t^k(x_t)$
        $\forall k \in [K]$ on a $\mathbf{P}_t(k) = (1-\gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$
        $\tilde{k}_t$ is drawn from $\mathbf{P}_t$
        $\tilde{k}_t$ is predicted and $y_{t,\tilde{k}_t}$ is revealed
        Compute $\tilde{\Delta}_t$ such as $\tilde{\Delta}_t^{n,k} = \frac{\lambda \hat{x}_t^{n,k} \delta_t^{n,k} \mathbf{1}[k_t = k]}{\mathbf{P}_t(k)}$
        $W_{t+1} = W_t + \tilde{\Delta}_t$
```

4 Models selection with adversarial bandit

Performances of neural networks are influenced by several parameters such as the learning step, the number of hidden layers, their size, and the initalization of weights. The multi-layer perceptron corresponding to a set of parameters is called model. Using batch learning, the models selection is done with a validation set. Using online learning, we propose to train the models in parallel and to use the adversarial bandit algorithm EXP3 [5, 6] to choose the best model. The choice of an adversarial bandit algorithm is justified by the fact that the performance of each model changes overtime due to the learning itself or due to the non-stationarity of the data.

Exp3 Let $\gamma_{\text{model}} \in [0,1]$ be an exploration parameter, $w_t = (w_t^1, ..., w_t^M)$ a weight vector, where each of its coordinate is initialized to 1, and M the number of models. Let m be the model chosen at time t, and $y_{t,m}$ be the obtained reward. The probability to choose m at round t is:

$$\mathbf{P}_{\text{model}\,t}(m) = (1 - \gamma_{\text{model}}) \frac{w_t^m}{\sum_{i=1}^{M} w_t^i} + \frac{\gamma_{\text{model}}}{M} \tag{2}$$

The weight update equation is:

$$w_{t+1}^i = w_t^i \text{exp} \left(\frac{\gamma_{\text{model}} \mathbf{1}[i = m] y_{t,m}}{\mathbf{P}_{\text{model}\,t}(i) M} \right) \tag{3}$$

NeuralBandit2 (see Algorithm 2) If we consider that a model is an arm, a run of this model can be considered as a sequence of rewards. The algorithm takes as inputs a list of M models and a model exploration parameter γ_{model}. For each element of the list, one NeuralBandit1 instance is initialized. Each instance corresponds to an arm. EXP3 algorithm is used to choose the arms over time. After receiving a reward, each neural network corresponding to the played arm is updated, and the weights of EXP3 are updated.

NeuralBandit3 (see Algorithm 3) The use of the algorithm NeuralBandit2 corresponds to the assumption that there is a model NeuralBandit1 which is the best for all actions. The algorithm NeuralBandit3 lifts this limitation by associating one EXP3 per action.

NeuralBandit3 has greater capacity of expression than NeuralBandit2 as each action can be associated with different models. However if the best model in NeuralBandit3 exists in NeuralBandit2, then NeuralBandit2 should find this model faster than NeuralBandit3 because it has only one instance of EXP3 with less possibility to update.

5 Experiments

The Forest Cover Type dataset from the UCI Machine Learning Repository is used. It contains 581.000 instances and it is shuffled. We have recoded each continuous variable using equal frequencies into five binary variables and we have recoded each categorical variable into binary variables. We have obtained 94 binary variables for the context, and we have used the 7 target classes as the set of actions. In order to simulate a datastream, the dataset is played in loop. At each round, if the algorithm chooses the right class the reward is 1 or else 0. The cumulated regret is computed from the rewards of an offline algorithm fitting the data with 93% of classification. The plot of the curves (Figure 1) are produced by averaging 10 runs of each algorithms with $\gamma = 0.005$, $\gamma_{model} = 0.1$. Each run began at a random position in the dataset. The parameters of each model (NeuralBandit1) are the combination of different sizes of hidden layer $(1, 5, 25, 50, 100)$ and different values of λ $(0.01, 0.1, 1)$.

Algorithm 2: NeuralBandit2

```
Data: $\gamma_{\text{model}} \in [0, 0.5]$ and a list of $M$ models parameters
begin
    Initialize $M$ NeuralBandit1 ;
    Initialize the EXP3 weight vector $w_0$ with $\forall m \in [M]$ $w_0^m = 1$;
    for $t = 1, 2, ..., T$ do
        Context $x_t$ is revealed;
        $m_t$ is drawn from $\mathbf{P}_{\text{model}\,t}$ (2);
        The model $m_t$ choose action $\tilde{k}_t$;
        $\tilde{k}_t$ is predicted and $y_{t,\tilde{k}_t}$ is revealed;
        Update of each network corresponding to the action $\tilde{k}_t$ for each model with (1);
        Update of the EXP3 weight vector $w_t$ with (3);
```

Algorithm 3: NeuralBandit3

```
Data: $\gamma \in [0, 0.5]$, $\gamma_{\text{model}} \in [0, 0.5]$and a list of $M$ model parameters
begin
    Initialize $K$ neural networks per model $m$ ;
    Initialize $K$ instance of EXP3;
    for $t = 1, 2, ..., T$ do
        Context $x_t$ is revealed;
        for $k = 1, 2, ..., K$ do
            $m_t^k$ is drawn from $\mathbf{P}_{\text{model}_t^k}$ (2);
            Action $k$ is scored $s_t^k = \mathbf{N}_t^{m_t^k,k}(x_t)$;
        $\hat{k}_t = \arg\max_{k \in [K]} s_t^k$;
        $\forall k \in [K]$ on a $\mathbf{P}_t(k) = (1-\gamma)\mathbf{1}[k = \hat{k}_t] + \frac{\gamma}{K}$;
        $\tilde{k}_t$ is drawn from $\mathbf{P}_t$;
        $\tilde{k}_t$ is predicted and $y_{t,\tilde{k}_t}$ is revealed;
        Update of each network corresponding to the action $\tilde{k}_t$ for each model with (1);
        Update of each EXP3 weight vectors with (3);
```

On stationary data (left part of Figure 1) Banditron achieves a high cumulated regret (57% of classification computed on the 100.000 last predictions) and is outperformed by all other contextual bandit algorithms on this dataset. The cumulated regret and classification rate of LinUCB (72%), CTS (73%) and NeuralBandit3 (73%) are similar and their curves tend to be the same. NeuralBandit2 has the fastest convergence rate and achieves a smaller cumulated regret with 76% of classification.

On non-stationary data (right part of Figure 1) Non-stationarity is simulated by swapping classes with a circular cycle ($1 \rightarrow 2$, $2 \rightarrow 3$,...,$7 \rightarrow 1$) every 500.000 iterations. At each concept drift, curves increase then stabilize. On stationary data LinUCB and CTS achieve a lowest regret than Banditron but can't deal with non-stationarity thus are outperformed by it after the first drift. Banditron converges again near instantanly.

Models selection algorithms need between 75.000 and 350.000 in the worst case to stabilize at each drift. NeuralBandit2 and NeuralBandit3 are more complex models than Banditron but achieve better performances on this non-stationary datastream. NeuralBandit3 seem to be more robust to nonstationarity than NeuralBandit2 on this dataset. This can be explained by the fact that sometime, one neural network can stay on a bad local minimum. If this append in NeuralBandit2, the entire model is penalized while in NeuralBandit3 the algorithm can still use all the other networks.

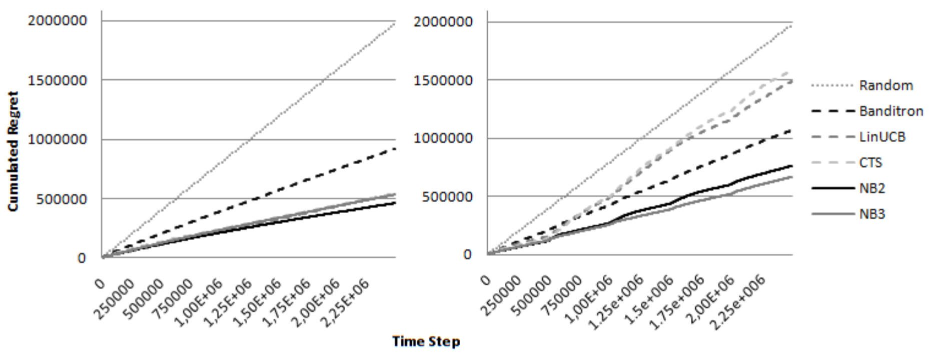

Fig. 1: The cumulated regret of different contextual bandit algorithms over time on Forest Cover Type. The left part is on a stationary datastream and the right part on a non-stationary datastream.

6 Conclusion

We introduced a new contextual bandit algorithm NeuralBandit1. Two variants with models selection NeuralBandit2 and NeuralBandit3 used an adversarial bandit algorithm to find the best parameters of neural networks. We confronted them to stationary and non-stationary datastream. They achieve a smaller cumulated regret than Banditron, LinUCB and CTS. This approach is successful and has the advantage of being trivially parallelizable. Models differentiation show a significant gain on the Forest Covert Type dataset with nonstationarity. We also showed empirically that our two models selection algorithms are robust to concept drift. These experimental results suggest that neural networks are serious candidates for addressing the issue of contextual bandit. However, they are freed from the constraint of linearity at the expense of the bound on the regret, the cost function being not convex.

References

1. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics **6**(1) (1985) 4–22
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2-3) (2002) 235–256
3. Thompson, W.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika **25** (1933) 285–294

4. Kaufmann, E., Korda, N., Munos, R.: Thompson Sampling: An Asymptotically Optimal Finite Time Analysis. In: Algorithmic Learning Theory, Proc. of the 23rd International Conference (ALT). Volume LNCS 7568., Lyon, France, Springer (2012) 199–213
5. Auer, P., Cesa-Bianchi, N.: On-line learning with malicious noise and the closure algorithm. Ann. Math. Artif. Intell. **23**(1-2) (1998) 83–99
6. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multiarmed bandit problem. SIAM J. Comput. **32**(1) (2002) 48–77
7. Kleinberg, R.D., Niculescu-Mizil, A., Sharma, Y.: Regret bounds for sleeping experts and bandits. In: COLT. (2008) 425–436
8. Chakrabarti, D., Kumar, R., Radlinski, F., Upfal, E.: Mortal multi-armed bandits. In: NIPS. (2008) 273–280
9. Feraud, R., Urvoy, T.: A stochastic bandit algorithm for scratch games. In Hoi, S.C.H., Buntine, W.L., eds.: ACML. Volume 25 of JMLR Proceedings., JMLR.org (2012) 129–143
10. Feraud, R., Urvoy, T.: Exploration and exploitation of scratch games. Machine Learning **92**(2-3) (2013) 377–401
11. Bubeck, S., Munos, R., Stoltz, G., Szepesvári, C.: Online optimization in x-armed bandits. In Koller, D., Schuurmans, D., Bengio, Y., Bottou, L., eds.: NIPS, Curran Associates, Inc. (2008) 201–208
12. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In Fürnkranz, J., Scheffer, T., Spiliopoulou, M., eds.: ECML. Volume 4212 of Lecture Notes in Computer Science., Springer (2006) 282–293
13. Gaudel, R., Sebag, M.: Feature selection as a one-player game. In Fürnkranz, J., Joachims, T., eds.: ICML, Omnipress (2010) 359–366
14. Seldin, Y., Auer, P., Laviolette, F., Shawe-Taylor, J., Ortner, R.: Pac-bayesian analysis of contextual bandits. In Shawe-Taylor, J., Zemel, R.S., Bartlett, P.L., Pereira, F.C.N., Weinberger, K.Q., eds.: NIPS. (2011) 1683–1691
15. Dudík, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., Zhang, T.: Efficient optimal learning for contextual bandits. CoRR (2011)
16. Langford, J., Zhang, T.: The epoch-greedy algorithm for multi-armed bandits with side information. In Platt, J.C., Koller, D., Singer, Y., Roweis, S.T., eds.: NIPS, Curran Associates, Inc. (2007)
17. Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation. CoRR (2010)
18. Chu, W., Li, L., Reyzin, L., Schapire, R.E.: Contextual bandits with linear payoff functions. In Gordon, G.J., Dunson, D.B., Dudk, M., eds.: AISTATS. Volume 15 of JMLR Proceedings., JMLR.org (2011) 208–214
19. Agrawal, S., Goyal, N.: Thompson sampling for contextual bandits with linear payoffs. CoRR (2012)
20. Kakade, S.M., Shalev-Shwartz, S., Tewari, A.: Efficient bandit algorithms for online multiclass prediction. In: Proceedings of the 25th International Conference on Machine Learning. ICML '08, New York, NY, USA, ACM (2008) 440–447
21. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Netw. **2**(5) (July 1989) 359–366
22. Tesauro, G.: Programming backgammon using self-teaching neural nets. Artificial Intelligence **134** (2002) 181–199
23. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. CoRR (2013)
24. Bottou, L., LeCun, Y.: On-line learning for very large datasets. Applied Stochastic Models in Business and Industry **21**(2) (2005) 137–151
25. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. MIT Press, Cambridge, MA, USA (1986) 318–362