Recurrent neural-linear posterior sampling for non-stationary bandits

Paulo Rauber
IDSIA, USI, SUPSI
Lugano, Switzerland
paulo@idsia.ch

Aditya Ramesh
USI
Lugano, Switzerland
aditya.ramesh@usi.ch

Jürgen Schmidhuber
IDSIA, USI, SUPSI, NNAISENSE
Lugano, Switzerland
juergen@idsia.ch

Abstract

An agent in a non-stationary multi-armed bandit problem should balance between exploration and the exploitation of (periodical or structural) patterns present in its previous experiences. Current successful agents require a handcrafted context that allows treating a non-stationary problem as a contextual multi-armed bandit problem. Unfortunately, even a carefully designed context may introduce spurious relationships or lack a convenient representation of crucial information. In order to address these issues, we propose an approach that learns to represent the relevant context for a decision based solely on the raw history of interactions between the agent and the environment. This approach relies on a combination of Bayesian recurrent neural networks and posterior sampling. Preliminary experiments show that our recurrent approach outperforms its feedforward counterpart in simple non-stationary problems when the handcrafted context is less than ideal.

1 Introduction

In a multi-armed bandit problem, an agent chooses an action (arm) based on its previous interactions with an environment (multi-armed bandit). In response, the environment transitions into a new hidden state and provides a reward. The goal of the agent is to maximize cumulative reward through a finite number of interactions with the environment, which requires balancing exploration and exploitation.

A multi-armed bandit problem is non-stationary if every policy that always chooses the same action should be considered poor, even if this action is the so-called best fixed action in hindsight [1]. Many interesting problems are naturally non-stationary. For instance, consider the problem of product recommendation. The acceptance rate for a recommendation may depend both on the time of the year (periodicity) and the results of previous recommendations (structure).

There are two main classes of agents that deal with non-stationary problems. *Passive* agents bias their decisions based on recent interactions, while *active* agents attempt to detect when a significant change occurs [2, 3]. Unfortunately, agents in both classes are incapable of exploiting periodicity and structure. An often successful alternative is to transform a non-stationary problem into a contextual multi-armed bandit problem, where the agent receives a context that informs its decision during each interaction [1]. In the product recommendation example, such context could encode the time of the year and the results of previous recommendations, which allows generalization from experience.

Riquelme et al. [4] recently showed that a simple Bayesian approach to represent uncertainty about feedforward neural network models achieves remarkable success in contextual multi-armed bandit problems when combined with posterior sampling (Thompson sampling) [5]. In this setting, posterior sampling relies on a posterior over models of the reward given the context and each prospective action. A single model is drawn from this posterior before each interaction, and the best action according to this model is chosen. Intuitively, increased certainty leads to decreased exploration.

4th workshop on Bayesian Deep Learning (NeurIPS 2019), Vancouver, Canada.

The feedforward neural-linear approach of Riquelme et al. [4] requires a handcrafted context when applied to non-stationary problems. Unfortunately, even a carefully designed context may introduce spurious relationships or lack a convenient representation of crucial information. In order to address these issues, we propose a recurrent neural-linear approach that learns to represent the relevant context based solely on the history of interactions. Preliminary experimental results indicate that the recurrent approach outperforms the feedforward approach in simple non-stationary problems when the handcrafted context is less than ideal.

2 Neural-linear posterior sampling

Notation. We denote random variables by upper case letters and assignments to these variables by corresponding lower case letters. A multi-armed bandit problem is a special case of the following partially observable reinforcement learning problem. The agent interacts with the environment (multi-armed bandit) during a single episode. At a given time step t, the environment is in a hidden state S_t, and the agent uses a policy to choose an action (arm) A_{t+1} given the history H_t, which encodes the actions $A_{1:t}$ and the rewards $R_{1:t}$. In response to this choice, the environment transitions into a hidden state S_{t+1} and outputs a reward R_{t+1}. This process is represented by the graphical model in Figure 1. Note that this formulation includes environments where an optimal policy is not necessarily greedy at every time step, although such environments are out of our scope.

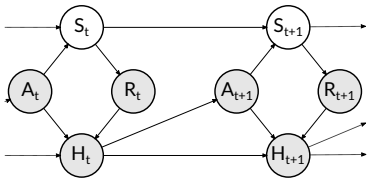

Figure 1: Multi-armed bandit problem.

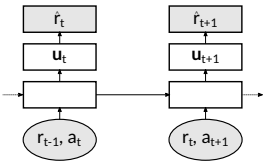

Figure 2: Recurrent neural-linear network.

We consider two approaches to maximize the reward at a given time step $t + 1$. They are both based on building a model of the distribution over R_{t+1} given the history h_t and each possible action.

Feedforward neural-linear posterior sampling. For a given time step t, consider the supervised dataset $\mathcal{D}_t = \{(\boldsymbol{\phi}(h_{t'-1}, a_{t'}), r_{t'})\}_{t'=1}^{t}$, where $\boldsymbol{\phi}$ is a function that encodes the information that allows predicting the reward $r_{t'}$ from the history $h_{t'-1}$ and the action $a_{t'}$. For example, $\boldsymbol{\phi}$ may encode (a periodic function of) the current time step, statistics regarding each arm, and the last n actions and rewards. Given a choice of parametric model and prior, a posterior distribution may be obtained from this dataset. All that is needed to complete posterior sampling is to draw a model from this posterior and choose the arm a_{t+1} with maximum expected reward given $\boldsymbol{\phi}(h_t, a_{t+1})$.

In the feedforward neural-linear approach, the model is based on a feedforward neural network. Parameter uncertainty is represented by a simple method that is highly successful in contextual bandit problems [4]. The first step of this method consists of fitting a feedforward neural network to the dataset $\mathcal{D}_t$ by minimizing the usual (regularized) mean squared error. The last layer of this network has a linear unit (with no bias). After the network is fit, let $\mathbf{z}_{t'}$ denote the output of its penultimate layer (last hidden layer) when given $\boldsymbol{\phi}(h_{t'-1}, a_{t'})$ as input, and consider the dataset $\mathcal{Z}_t = \{(\mathbf{z}_{t'}, r_{t'})\}_{t'=1}^{t}$. The second step of the method consists of applying Bayesian linear regression to $\mathcal{Z}_t$. In this work, we assume that the reward $R_{t'}$ given a history $h_{t'-1}$ and an action $a_{t'}$ is Gaussian with mean $\mathbf{z}_{t'} \cdot \mathbf{w}^*$ and known variance σ^2. We also assume a Gaussian prior over $\mathbf{W}$ with mean zero and covariance matrix $\tau^2\mathbf{I}$. Given these choices, the corresponding (Gaussian) posterior over $\mathbf{W}$ given $\mathcal{Z}_t$ can be obtained analytically [6, Section 3.3], which makes posterior sampling very efficient.

Recurrent neural-linear posterior sampling. In this approach, the model is based on a recurrent neural network (Figure 2). At a given time step t, this network receives the reward r_{t-1} and the action a_t as inputs, and outputs an estimate of the reward r_t. Such estimates attempt to minimize the usual (regularized) mean squared error. Suppose that the last layer of this network has a linear unit (with no bias), and let $\mathbf{u}_{t'}$ denote the output of its penultimate layer (last hidden layer) when given the rewards $r_{1:t'-1}$ and the actions $a_{1:t'}$ as inputs. Analogously to the previous approach, Bayesian linear regression may be applied to the dataset $\mathcal{U}_t = \{(\mathbf{u}_{t'}, r_{t'})\}_{t'=1}^{t}$ in order to allow efficient posterior

sampling. Most importantly, note that the recurrent neural-linear approach eliminates the need for a handcrafted feature map $\boldsymbol{\phi}$.

3 Experiments

Multi-armed bandits. We performed preliminary experiments in two unstructured and two structured problems, which were chosen for their simplicity and flexibility. In the *flipping Bernoulli* problem, the reward for each arm k is Bernoulli, with a mean that changes from p_k to $1 - p_k$ every h time steps. Similarly, in the *flipping Gaussian* problem, the mean of the Gaussian reward for each arm k changes from μ_k to $-\mu_k$ every h time steps, while the corresponding variance s^2 is fixed across arms. In the *circular Markov chain* problem, the reward for every arm is Gaussian with mean μ and variance s^2, except for a single arm whose reward is Gaussian with mean $\mu^* > \mu$. After this arm is chosen, it trades place with the next arm in a predefined cyclic order. In the *graph distance* problem, each arm is a node in a random graph. At a given time step, there is a single reference arm k. The reward given by each arm j is Gaussian with mean $\exp(-d_{j,k})$ and fixed variance s^2, where $d_{j,k}$ is the (graph) distance between j and k. After the reference arm k is chosen, a random arm becomes the new reference arm. The problem settings used in our experiments are detailed in Appendix A.

Policies. We compared at least five policies for each multi-armed bandit problem. The *Random* policy chooses arms at random. The *Best (R)NN* policy uses the best (feedforward or recurrent) neural-linear posterior sampling hyperparameters found for a given problem, while the *Default (R)NN* policy uses hyperparameters that work well across problems. Appendix B details our implementation and hyperparameter search. [1]

Results. The *regret* at a given time step is the difference between the expected cumulative reward of an oracle that knows which arm has maximum expected reward at every time step and the cumulative reward obtained by an agent. Experimental results are represented by regret plots in Appendix C.

Analysis. Figures 3 and 5 show the results for the flipping Bernoulli problem and for the flipping Gaussian problem when $h = 8$. In both cases, the feature map $\boldsymbol{\phi}$ encodes periodic functions of the current time step for a fixed set of periods Λ that includes the period $2h$ underlying the changes in the environments. The feedforward approaches clearly outperform the recurrent approaches in the flipping Bernoulli problem, while both approaches obtain comparable performance in the flipping Gaussian problem. However, when neither the period $2h$ nor one of its multiples is in the set of periods Λ, the recurrent approaches outperform the feedforward approaches, as shown by Figures 4 and 6 for $h = 10$. The performance of the feedforward approaches shown in Figure 6 does not make it obvious that important information is being ignored, which is a general issue with handcrafted contexts. Most interestingly, Figures 7 and 8 show that the presence of periodic functions of the current time step in the handcrafted context significantly compromises the performance of the feedforward approaches in both the circular Markov chain problem and the graph distance problem, which illustrates the risk of encoding irrelevant information. Finally, Figure 9 shows that the recurrent approaches achieve similar performance to a Beta-Bernoulli posterior sampling baseline in a *stationary* flipping Bernoulli problem ($h = \infty$), which is evidence of their robustness to model misspecification.

4 Conclusion

We introduced a recurrent neural-linear posterior sampling approach that may be applied to solve arbitrary non-stationary multi-armed bandit problems. This approach learns to represent the relevant context for a decision based solely on the history of interactions between the agent and the environment, which avoids carefully handcrafted contexts required by previous approaches. Preliminary experiments show that our recurrent approach outperforms its feedforward counterpart in simple non-stationary problems when the handcrafted context introduces spurious relationships or lacks a convenient representation of crucial information. We hypothesize that our approach may be particularly useful for a problem that is both non-stationary *and* contextual, since handcrafting features may become even more challenging. We hope to find an appropriate problem to test this hypothesis in future work. We are also interested in comparing the neural-linear approach to other Bayesian neural

[1] An open-source implementation is available on `https://github.com/paulorauber/rnlps`.

network approaches. Finally, our approach may also benefit from specially designed recurrent neural network architectures, which were found to be very useful in other applications.

Acknowledgments

We would like to thank Sjoerd van Steenkiste, Francesco Faccio, Raoul Malm, and Imanol Schlag for their valuable feedback. This research was supported by the Swiss Natural Science Foundation grant (200021_165675/1).

References

[1] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press (draft), 2019.

[2] Fang Liu, Joohyun Lee, and Ness Shroff. A change-detection based framework for piecewise-stationary multi-armed bandit problem. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[3] Yang Cao, Zheng Wen, Branislav Kveton, and Yao Xie. Nearly optimal adaptive procedure with change detection for piecewise-stationary bandit. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 418–427, 2019.

[4] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep Bayesian bandits showdown: An empirical comparison of Bayesian deep networks for Thompson sampling. In *International Conference on Learning Representations*, 2018.

[5] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[6] C. M. Bishop. *Pattern Recognition and Machine Learning*. Information science and statistics. Springer, 2013. ISBN 9788132209065.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, 2014.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

[9] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

A Multi-armed bandit problems

In our experiments, the following settings were used for each problem.

Flipping Bernoulli: 8 arms, initial means in $\{0.1, 0.2, \ldots, 0.9\} \setminus \{0.5\}$.

Flipping Gaussian: 8 arms, initial means in $\{0.1, 0.2, \ldots, 0.9\} \setminus \{0.5\}$, variance $s^2 = 0.1^2$.

Circular Markov Chain: 8 arms, common mean $\mu = 0$, best mean $\mu^* = 1$, variance $s^2 = 0.05^2$.

Graph distance: 8 arms, variance $s^2 = 0.05^2$, edge probability 0.5.

B Policies

In our experiments, a trial comprises 2048 time steps. The results for a given policy aggregate ten independent trials, and show bootstrapped confidence intervals of 95%. This appendix details the neural-linear posterior sampling policies and the hyperparameter search protocol.

B.1 Feedforward neural-linear posterior sampling

Inputs. For a given time step t, the feature map $\boldsymbol{\phi}$ encodes the action a_t together with the last n pairs of actions and rewards $\{(a_{t-k}, r_{t-k})\}_{k=1}^{n}$, where n is the so-called order. Actions are one-hot encoded. The feature map $\boldsymbol{\phi}$ also encodes different periodic functions of the current time step t. Concretely, for each λ in a given set of periods Λ, the pair $(\sin \alpha_\lambda, \cos \alpha_\lambda)$ is encoded by $\boldsymbol{\phi}$, where $\alpha_\lambda = 2\pi t \lambda^{-1}$. In words, for each $\lambda \in \Lambda$, the feature map $\boldsymbol{\phi}$ encodes the sine and the cosine of the angle α_λ of a (counterclockwise) rotating particle that completes one revolution every λ time steps. Depending on the experiment, the set of periods Λ is either $\{2, 4, 8, 16, 32\}$ or empty.

Network architecture. The network has three hidden layers. The first hidden layer has L_1 linear units. The second and third hidden layers have L_2 and L_3 hyperbolic tangent units, respectively. The last layer has one linear unit (with no bias).

Training. The network is trained to minimize the mean-squared error with an L2 regularization penalty of 0.001. A sequence of e training steps is performed every q time steps using Adam [7] with a learning rate η. Note that each training step requires computing the gradient of the loss on the entire dataset. The linear regression posterior is recomputed using the entire dataset at every time step. One forward pass is required to evaluate each available action.

Hyperparameter search An independent grid search is required to find the *Best NN* policy for each multi-armed bandit problem. This policy achieves maximum cumulative reward averaged over five independent trials. The hyperparameter grid is described below. The underlined hyperparameters correspond to the *Default NN* policy, which (somewhat subjectively) works well across problems.

- Order n: $\underline{1}$, 4.
- Numbers of units (L_1, L_2, L_3): $(16, 16, 16)$, $\underline{(32, 32, 32)}$, $(64, 64, 64)$.
- Learning rate η: 0.001, $\underline{0.01}$, 0.1.
- Number of epochs e by training step: 16, $\underline{64}$.
- Interval q between training steps: $\underline{32}$, 128.
- Assumed variance σ^2 of the reward distribution: 0.1, $\underline{0.3}$.
- Variance τ^2 of the prior distribution over each weight: 0.5, $\underline{1}$.

B.2 Recurrent neural-linear posterior sampling.

Inputs. At a given time step t, the input to the recurrent neural network is the reward r_{t-1} and the action a_t. This action is one-hot encoded.

Network architecture. The network has three hidden layers. The first hidden layer has L_1 linear units. The second hidden layer has L_2 long short-term memory units [8, 9]. The third hidden layer has L_3 hyperbolic tangent units. The last layer has one linear unit (with no bias).

Training. The network is trained to minimize the mean-squared error with an L2 regularization penalty of 0.001. A sequence of e training steps is performed every q time steps using Adam [7] with a learning rate η. Note that each training step requires computing the gradient of the loss on the entire sequence. The linear regression posterior is recomputed using the entire dataset at every time step. One forward pass is required to evaluate each available action. Note that this does not require forward passing the entire sequence for each action.

Hyperparameter search An independent grid search is required to find the *Best RNN* policy for each multi-armed bandit problem. This policy achieves maximum cumulative reward averaged over five independent trials. The hyperparameter grid is described below. The underlined hyperparameters correspond to the *Default RNN*, which (somewhat subjectively) works well across problems.

- Numbers of units (L_1, L_2, L_3): $\underline{(16, 16, 16)}$, $(32, 32, 32)$.
- Learning rate η: 0.001, $\underline{0.01}$, 0.1.
- Number of epochs e by training step: $\underline{16}$, 64.
- Interval q between training steps: $\underline{32}$, 128.
- Assumed variance σ^2 of the reward distribution: 0.1, $\underline{0.3}$.
- Variance τ^2 of the prior distribution over each weight: 0.5, $\underline{1}$.

C Experimental results

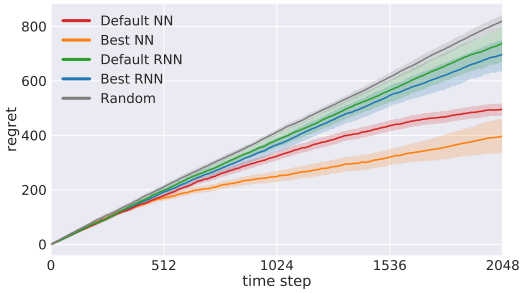

Figure 3: Flipping Bernoulli ($h = 8$).

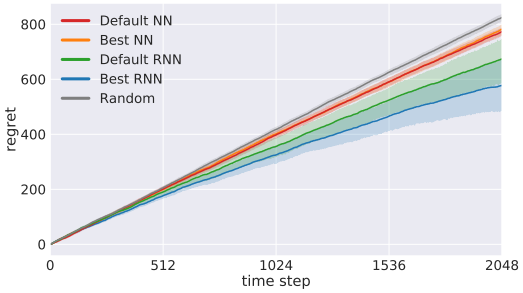

Figure 4: Flipping Bernoulli ($h = 10$).

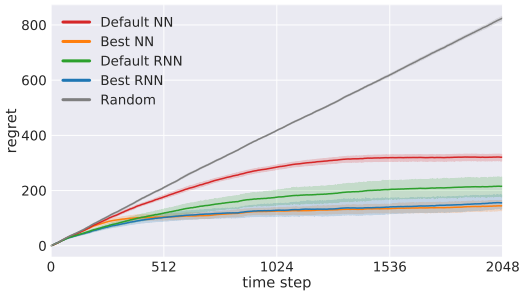

Figure 5: Flipping Gaussian ($h = 8$).

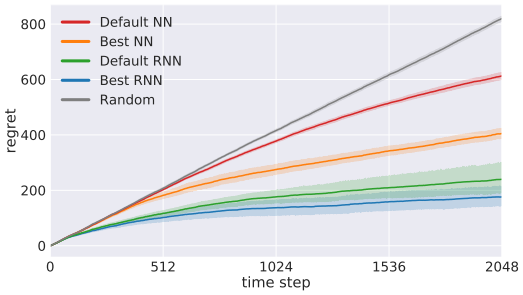

Figure 6: Flipping Gaussian ($h = 10$).

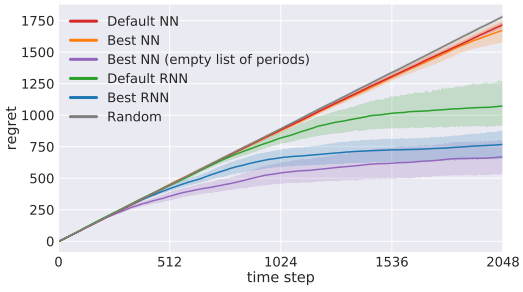

Figure 7: Circular Markov Chain.

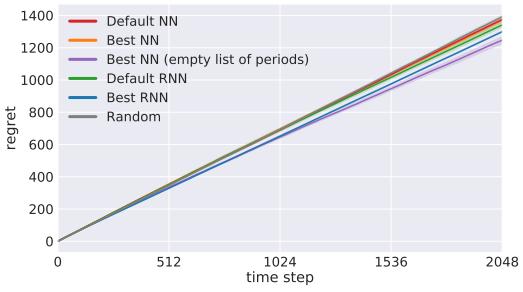

Figure 8: Graph distance.

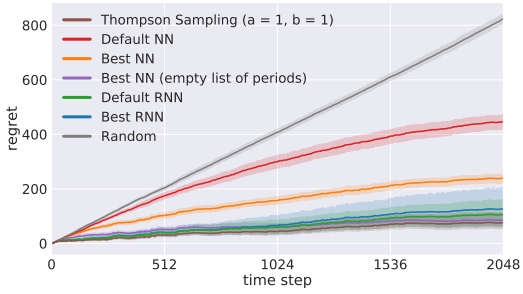

Figure 9: Flipping Bernoulli ($h = \infty$).