Recurrent Neural-Linear Posterior Sampling for Non-Stationary Contextual Bandits

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

Paulo Rauber *
QMUL
London, United Kingdom
p.rauber@qmul.ac.uk

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

Abstract

An agent in a non-stationary contextual bandit problem should balance between exploration and the exploitation of (periodic or structured) patterns present in its previous experiences. Handcrafting an appropriate historical context is an attractive alternative to transform a non-stationary problem into a stationary problem that can be solved efficiently. However, even a carefully designed historical 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 features extracted by recurrent neural networks with a contextual linear bandit algorithm based on posterior sampling. Our experiments on a diverse selection of contextual and non-contextual non-stationary problems show that our recurrent approach consistently outperforms its feedforward counterpart, which requires handcrafted historical contexts, while being more widely applicable than conventional non-stationary bandit algorithms.

1 Introduction

In a broad formulation of a contextual bandit problem, an agent chooses an arm (action) based on a context (observation) and previous interactions with an environment. In response, the environment transitions into a new hidden state and provides a reward and a new context. 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.

Many practical problems can be seen as contextual bandit problems [1]. For example, consider the problem of product recommendation: a context may encode information about an individual, an arm may represent a recommendation, and a reward may signal whether a recommendation succeeded.

If the expected reward is an (unknown) fixed linear function of a (known) vector that represents the preceding arm and context, independently of the remaining history of interactions between the agent and the environment, then several contextual linear bandit algorithms provide strong performance guarantees relative to the best fixed policy that maps contexts to arms [2–5].

However, in a non-stationary contextual bandit problem, the state of the environment changes in such a way that the performance of any fixed policy that maps contexts to arms is unsatisfactory [6]. In the product recommendation example, the success rate of a recommendation may depend both on the time of the year and the results of previous recommendations. Therefore, the presence of this information in the contexts determines whether the problem is non-stationary.

*Equal contribution.

Preprint. Under review.

Handcrafting an appropriate context is an attractive alternative to transform a non-stationary problem into a stationary problem that can be solved efficiently [6]. Unfortunately, an inappropriate context may introduce spurious relationships or lack a convenient representation of crucial information.

Another alternative is to employ non-stationary bandit algorithms, which can be divided into two main families. *Passive* algorithms bias their decisions based on recent interactions with the environment, while *active* algorithms attempt to detect when a significant change occurs [7–9]. Unfortunately, algorithms from both families are incapable of exploiting periodicity and structure (the effect of actions on the rewards of future actions), which may be important even when no planning is required.

In order to address these issues, we propose an approach based on a recurrent neural network that receives the raw history of interactions between the agent and the environment. This network is trained to predict the reward for each pair of arm and context. The features extracted by the network are combined with a contextual linear bandit algorithm based on posterior sampling [4], which potentially allows an agent to achieve high performance in a non-stationary contextual problem without carefully handcrafted historical contexts. Besides its advantages in contextual problems, our approach is also radically different from previous approaches that are able to exploit periodic or structured patterns in non-contextual non-stationary bandit problems.

Our approach is partially motivated by the work of Riquelme et al. [10], whose comprehensive experiments have shown that the combination of features extracted by a (feedforward) neural network with a contextual linear bandit algorithm based on posterior sampling achieves remarkable success in (stationary) contextual bandit problems. Our approach can also be seen as a model-based counterpart to recent model-free meta-learning approaches based on recurrent neural networks that have been applied to non-contextual stationary bandit problems [11, 12].

We evaluate our approach using a diverse selection of contextual and non-contextual non-stationary bandit problems. The results of this evaluation show that our recurrent approach consistently outperforms its feedforward counterpart, which requires handcrafted historical contexts, while being more widely applicable than conventional non-stationary bandit algorithms.

2 Preliminaries

We denote random variables by upper case letters and assignments to these variables by corresponding lower case letters. We omit the subscript that typically relates a probability function to random variables when there is no risk of ambiguity. For example, we may use $p(x)$ to denote $p_X(x)$ in the same context where we use $p(y)$ to denote $p_Y(y)$.

A contextual bandit problem can be seen as a special case of the following partially observable reinforcement learning problem. An agent interacts with an environment (multi-armed bandit) during a single episode that lasts T time steps. At a given time step $t \in \{1, \ldots, T\}$, the environment is in a hidden state S_t, and the agent uses a policy π to choose an action (arm) A_{t+1} given the history H_t, which encodes the previous rewards $R_{1:t}$, observations (contexts) $X_{1:t}$, and actions $A_{2:t}$. In response to this action, the environment transitions into a hidden state S_{t+1}, and outputs a reward R_{t+1} and an observation X_{t+1}. This process can be represented by the directed graphical model in Figure 1.

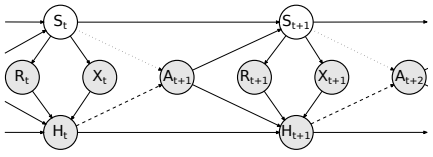

Figure 1: Directed graphical models that represent the interaction between the agent or oracle and the environment. Dashed or dotted edges belong respectively to either the agent or the oracle.

In contrast to an agent, an *oracle* uses a policy π^* to choose an action a_{t+1} that maximizes the immediate expected reward $\mathbb{E}\left[R_{t+1} \mid s_t, a_{t+1}\right]$ given the hidden state s_t, for all t. Note that such oracle makes greedy decisions. Although non-greedy agents may achieve higher expected cumulative reward in fully fledged reinforcement learning environments, such environments are out of our scope.

The regret of a policy π is given by $\sum_{t=1}^{T} \mathbb{E}_{\pi^*}[R_t] - \mathbb{E}_{\pi}[R_t]$, where π^* is an oracle policy, and the subscript on an expectation denotes the policy used for choosing actions. We are generally interested in policies that have low regret across a family of environments.

3 Posterior sampling for contextual linear bandits

This section presents a decision-making algorithm for contextual linear bandits that is at the core of our proposed approach. Agrawal and Goyal [4] were the first to provide strong theoretical guarantees for this algorithm under standard technical assumptions in the adversarial setting.

Suppose that the expected reward for time step t given the history h_{t-1}, the action a_t, and an (unknown) weight vector $\mathbf{w}$ is given by $\mathbb{E}[R_t \mid h_{t-1}, a_t, \mathbf{w}] = \mathbf{w} \cdot \boldsymbol{\phi}(x_{t-1}, a_t)$, where the feature map $\boldsymbol{\phi}$ is a (known) function responsible for encoding any given pair of observation and action into a feature vector. In other words, suppose that the expected reward for a given time step is an unknown linear function of a known feature vector that represents the previous observation and the chosen action, independently of the rest of the history.

In this setting, posterior sampling starts by representing knowledge about $\mathbf{W}$ in a prior distribution. At a given time step t, the algorithm consists of four simple steps: (1) drawing a single parameter vector $\mathbf{w}_t$ from the prior over $\mathbf{W}$; (2) choosing an action a_t that maximizes $\mathbf{w}_t \cdot \boldsymbol{\phi}(x_{t-1}, a_t)$, (3) observing the reward r_t; (4) computing the posterior over $\mathbf{W}$ to be used as a prior for step $t+1$. Intuitively, at a given time step, an action is drawn according to the probability that it is optimal.

In order to derive an efficient algorithm, suppose that the prior density for $\mathbf{w}$ is given by $p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \mathbf{w}_0, \mathbf{V}_0)$, for some hyperparameters $\mathbf{w}_0$ and $\mathbf{V}_0$.

Furthermore, consider the dataset $\mathcal{D} = \{(\boldsymbol{\phi}(x_{t'-1}, a_{t'}), r_{t'})\}_{t'=2}^{t}$, and suppose that the conditional likelihood of the parameter vector $\mathbf{w}$ is given by $p(\mathcal{D} \mid \mathbf{w}) = \mathcal{N}(\mathbf{r} \mid \boldsymbol{\Phi}\mathbf{w}, \sigma^2\mathbf{I})$, where $\mathbf{r} = (r_2, \ldots, r_t)$ is the reward vector, $\boldsymbol{\Phi}$ is the design matrix where each row corresponds to a feature vector in $\mathcal{D}$, $\mathbf{I}$ is the appropriate identity matrix, and $\sigma^2 > 0$ is a hyperparameter.

In that case, the posterior density for $\mathbf{w}$ is given by $p(\mathbf{w} \mid \mathcal{D}) \propto_{\mathbf{w}} \mathcal{N}(\mathbf{r} \mid \boldsymbol{\Phi}\mathbf{w}, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{w} \mid \mathbf{w}_0, \mathbf{V}_0)$. Because the random vectors $\mathbf{W}$ and $\mathbf{R}$ are related by a linear Gaussian system [13], the desired posterior density is given by $p(\mathbf{w} \mid \mathcal{D}) = \mathcal{N}(\mathbf{w} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, where

$$\boldsymbol{\Sigma}^{-1} = \mathbf{V}_0^{-1} + \frac{1}{\sigma^2}\boldsymbol{\Phi}^T\boldsymbol{\Phi}, \qquad \boldsymbol{\mu} = \frac{1}{\sigma^2}\boldsymbol{\Sigma}\boldsymbol{\Phi}^T\mathbf{r} + \boldsymbol{\Sigma}\mathbf{V}_0^{-1}\mathbf{w}_0. \tag{1}$$

At a given time step, it is straightforward to draw a parameter vector from this multivariate Gaussian posterior density function (Step 1), choose the best corresponding action (Step 2), observe the outcome (Step 3), and update the dataset and the posterior (Step 4), which completes the algorithm.

Crucially, the assumptions of a multivariate Gaussian prior and a multivariate Gaussian likelihood are only used to derive an efficient algorithm. The conditions under which this algorithm achieves its theoretical guarantees are very permissive and somewhat unrelated [4]. This is important because the dataset $\mathcal{D}$ is generally not composed of independent and identically distributed sample elements.

4 Feedforward neural-linear feature vectors

This section presents the process of extracting feedforward neural-linear feature vectors that ideally allow predicting the expected reward from any pair of history and action. A comprehensive benchmark has shown that the combination of these feature vectors with posterior sampling for contextual linear bandits often outperforms other posterior sampling approaches for contextual bandits [10].

Consider the dataset $\mathcal{D} = \{(\boldsymbol{\psi}(h_{t'-1}, a_{t'}), r_{t'})\}_{t'=2}^{t}$, where the feature map $\boldsymbol{\psi}$ is a function responsible for encoding the information that ideally allows predicting the reward $r_{t'}$ from the history $h_{t'-1}$ and the action $a_{t'}$ into a feature vector, for all t'. In contrast to the previous section, we do not assume that the expected reward is a linear function of the corresponding feature vector.

In a stationary contextual problem, $\boldsymbol{\psi}(h_{t'-1}, a_{t'})$ may encode just the observation $x_{t'-1}$ and the action $a_{t'}$ in order to enable predicting $r_{t'}$. In a non-stationary contextual problem, $\boldsymbol{\psi}(h_{t'-1}, a_{t'})$ may encode a (periodic function of) the current time step t'; statistics regarding actions; statistics

regarding observations; the last n rewards, observations, and actions; and arbitrary combinations of similar information. As will become clear, the need to handcraft an appropriate feature map $\boldsymbol{\psi}$ for a specific non-stationary problem is a potential weakness, since $\boldsymbol{\psi}$ may introduce spurious relationships or dismiss crucial information.

Extracting feedforward neural-linear feature vectors requires fitting a feedforward neural network to the dataset $\mathcal{D}$, which may be accomplished by searching for parameters that minimize a cost function using typical methods. Note that such methods assume that the dataset $\mathcal{D}$ is composed of independent and identically distributed sample elements, which is generally not the case, as in the previous section.

The feedforward neural-linear feature vector $\mathbf{z}_{t'}$ is the output of the penultimate layer (last hidden layer) of the (fitted) neural network when given $\boldsymbol{\psi}(h_{t'-1}, a_{t'})$ as input (Fig. 2). The only restriction on the network architecture is that the last layer should have a single linear unit (with no bias).

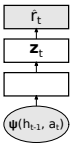

Figure 2: Feedforward neural-linear network.

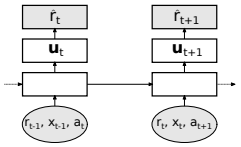

Figure 3: Recurrent neural-linear network.

By construction, if the parameters of the neural network achieve low cost on the training dataset $\mathcal{D}$, then it should be possible to approximate the reward $r_{t'}$ as a linear function of the feedforward neural-linear feature vector $\mathbf{z}_{t'}$, for any $t' \leq t$. Under the strong assumption that this is also true for $t' > t$, feedforward neural-linear feature vectors can be combined with posterior sampling for contextual linear bandits to provide a complete algorithm for contextual bandits. Despite its lack of general theoretical guarantees, this algorithm excels experimentally [10].

5 Recurrent neural-linear feature vectors

This section introduces the novel process of extracting recurrent neural-linear feature vectors that ideally allow predicting the expected reward from any pair of history and action. In contrast to the feedforward approach, this process eliminates the need for a handcrafted feature map $\boldsymbol{\psi}$.

Consider the dataset $\mathcal{D} = \{(\tau_{t'}, r_{t'})\}_{t'=2}^{t}$, where $\tau_{t'} = (r_1, x_1, a_2, \ldots, r_{t'-1}, x_{t'-1}, a_{t'})$ represents the interaction between the agent and the environment up to time step t'.

Extracting recurrent neural-linear feature vectors requires fitting a recurrent neural network to the dataset $\mathcal{D}$, which may be accomplished by searching for parameters that minimize a cost function using typical methods. At a given time step t', this network receives as input the reward $r_{t'-1}$, the observation $x_{t'-1}$, the action $a_{t'}$, and attempts to predict the reward $r_{t'}$ (Fig. 3).

The recurrent neural-linear feature vector $\mathbf{u}_{t'}$ is the output of the penultimate layer of the (fitted) recurrent neural network when given $\tau_{t'}$ as input. As in the previous section, the only restriction placed on the network architecture is that the last layer should have a single linear unit (with no bias).

As in the feedforward approach, if the parameters of the recurrent neural network achieve low cost on the training dataset $\mathcal{D}$, then it should be possible to approximate the reward $r_{t'}$ as a linear function of the recurrent neural-linear feature vector $\mathbf{u}_{t'}$, for any $t' \leq t$. Under the strong assumption that this is also true for $t' > t$, recurrent neural-linear feature vectors can be combined with posterior sampling for contextual linear bandits to provide a complete algorithm for contextual bandits.

Most importantly, the recurrent neural-linear approach eliminates the need for a handcrafted feature map $\boldsymbol{\psi}$. Besides its potential advantages in contextual problems, this approach is also radically different from previous approaches that are able to exploit periodic or structured patterns in non-contextual problems.

6 Experiments

This section reports results of an empirical comparison between our recurrent neural-linear approach, its feedforward counterpart, and conventional non-stationary bandit algorithms.

6.1 Bandit problems

We performed experiments on a diverse selection of contextual and non-contextual non-stationary bandit problems, which is outlined below and detailed in Appendix A.1. Because there are no standard benchmarks for non-stationary bandit algorithms, this selection combines original problems with problems borrowed from previous work. These problems may be partitioned into four categories according to their underlying non-stationarity: abrupt periodic, smooth periodic, structured (where an action may affect the rewards of future actions), or unknown (derived from a real dataset).

Non-contextual bandit problems. In the (abrupt periodic) *flipping Gaussian* and *flipping Bernoulli* problems, the mean reward of each arm switches abruptly every fixed number of time steps. In the (smooth periodic) *sinusoidal Bernoulli* problem, the mean reward of each arm is a sinusoidal function of the current time step. In the (structured) *circular Markov chain* problem, the best arm trades place with the next arm in a pre-defined cyclical order after it is found.

Contextual bandit problems. In the (abrupt periodic) *flipping digits* problem, each context corresponds to an image of a digit, and the best arm for each digit switches every fixed number of time steps. In the (unknown) *wall-following robot* problem, each context encodes readings of sensors from a real robot, and the best arm depends on the underlying movement pattern of the robot. The remaining two problems are non-stationary contextual linear bandit problems. In the (abrupt periodic) *flipping vector* problem, the expected reward measures the alignment between an action-dependent vector and a vector that switches direction every fixed number of time steps. In the (smooth periodic) *rotating vector* problem, the expected reward measures the alignment between an action-dependent vector and a vector rotating about the origin.

6.2 Implementation

This section details the implementation of the feedforward and recurrent neural-linear posterior sampling approaches.[2] Section 6.3 details the grid search for hyperparameters.

Feedforward and recurrent neural-linear networks are trained to minimize the mean squared error with an L2 regularization penalty $\lambda = 0.001$. Every network weight is initially drawn from a standard Gaussian distribution, and redrawn if far from the mean by two standard deviations, and every network bias is initially zero. A sequence of e training steps is performed every q time steps (interactions with the environment) using Adam [14] with a learning rate η. Each training step requires computing the gradient of the loss on the entire dataset or sequence. The linear regression posterior is recomputed using the entire dataset at every time step. The prior hyperparameters are $\mathbf{w}_0 = \mathbf{0}$ and $\mathbf{V}_0 = \tau^2\mathbf{I}$, where $\tau^2 > 0$ is another hyperparameter. One forward pass is required to evaluate each available action. In the recurrent case, note that this does not require forward passing the entire sequence for each action.

Feedforward neural-linear posterior sampling. The feature map $\boldsymbol{\psi}$ encodes the last observation x_{t-1} and the action a_t together with the last n triplets of observations, actions, and rewards $\{(x_{t-k-1}, a_{t-k}, r_{t-k})\}_{k=1}^{n}$, where n is the so-called order. All actions are one-hot encoded. The feature map $\boldsymbol{\psi}$ also encodes the current time step t, which is the sole input to a sinusoidal layer with D units. Each sinusoidal unit i computes $\sin(a_i t + b_i)$, where a_i and b_i are network parameters. The output of this sinusoidal layer is concatenated with the remaining inputs from the feature map, comprising the input to the remaining network. This network has three additional 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). For the two non-stationary contextual linear bandit problems (see App. A.1), the feature map $\boldsymbol{\psi}$ encodes the action-dependent vector $\mathbf{x}_{t-1,a_t}$ instead of any observation x_{t-1}, while the corresponding action a_t is not encoded.

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

Recurrent neural-linear posterior sampling. At a given time step t, the input to the recurrent neural network is the reward r_{t-1}, the observation x_{t-1}, and the action a_t. This action is one-hot encoded. 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 [15, 16]. The third hidden layer has L_3 hyperbolic tangent units. The last layer has one linear unit (with no bias). For the two non-stationary contextual linear bandit problems, at a given time step t, the input to the recurrent neural network is the reward r_{t-1} and the action-dependent vector $\mathbf{x}_{t-1,a_t}$, while the corresponding action a_t is not an input.

6.3 Evaluation

We present results of at least five policies for each bandit problem. The *random* policy chooses arms at random. The *best (R)NN* policy employs (feedforward or recurrent) neural-linear posterior sampling with hyperparameters that achieve maximum cumulative reward averaged over five independent trials according to an independent grid search for each problem. In contrast, the *default (R)NN* policy employs (feedforward or recurrent) neural-linear posterior sampling with hyperparameters that perform well across either the contextual or the non-contextual problems (including two variations of the rotating vector problem). Concretely, such *default hyperparameters* achieve maximum *normalized score* averaged across either the contextual or the non-contextual problems. The normalized score of a hyperparameter setting ξ on a problem is given by $(m_\xi - m_-)/(m_+ - m_-)$, where m_ξ is the average cumulative reward of the setting ξ over five independent trials, m_- is the average cumulative reward of the random policy, and m_+ is the average cumulative reward of the corresponding best (R)NN policy. Appendix A.2 contains a complete description of the hyperparameter grid and the resulting default hyperparameters for each neural-linear approach.

For some of the bandit problems, we also present results of policies based on more conventional algorithms. For non-contextual bandit problems, we present the results of policies based on discounted UCB (D-UCB) and sliding-window UCB (SW-UCB) [17]. The hyperparameters for each of these algorithms were selected based on the same protocol used to select hyperparameters for the best (R)NN policy. The corresponding hyperparameter grids are described in Appendix A.2. For the non-stationary contextual *linear* bandit problems, we present the results of policies based on discounted linear UCB (D-LinUCB) [18] and sliding-window linear UCB (SW-LinUCB) [19]. The hyperparameters for each of these algorithms were selected optimally based on the total number of time steps and the variation budget of each problem [18], requiring additional knowledge in comparison with the neural-linear approaches.

6.4 Analysis

Appendix A.3.2 and Appendix A.3.3 present a regret curve for each combination of problem and policy. Each of these curves aggregates the empirical regret across ten independent trials (not considered for hyperparameter search), and shows bootstrapped confidence intervals of 95%. The average empirical regret at the end of these trials is summarized in Table 1.

Table 1: Average final regret across ten trials for each combination of problem and policy.

Bandit problem	Best NN	Best RNN	Default NN	Default RNN	D-(L)UCB	SW-(L)UCB
Flipping Gaussian	719.35	**254.59**	643.38	357.58	1381.3	1327.27
Flipping Bernoulli	1267.9	1251.9	**1151**	1308.5	1199.9	1220.6
Sinusoidal Bernoulli	1344.94	**643.94**	1003.24	**643.94**	935.24	1154.64
C. Markov chain	2151.97	**895.23**	2151.97	2001.57	3005.1	3154.96
Flipping digits	3372.38	**3014.88**	3334.48	3314.28	-	-
Wall-following robot	2791.47	2383.47	2790.07	**2348.27**	-	-
Flipping vector	1082.86	**1052.22**	1082.86	**1052.22**	1073.92	1084.6
R. vector ($f = 32^{-1}$)	1102.34	**473.72**	1333.47	482.05	3345.18	1028.9
R. vector ($f = 2048^{-1}$)	1103.81	910.81	1121.37	1118.18	642.28	**31.52**

Appendix A.3.4 and Appendix A.3.5 present a hyperparameter sensitivity curve for each combination of problem and neural-linear approach. A hyperparameter sensitivity curve displays the average cumulative reward achieved by each hyperparameter setting (sorted from highest to lowest along the horizontal axis). Such curves are useful to assess robustness regarding hyperparameter choices.

The remainder of this section highlights the most notable aspects of these results.

Non-contextual bandit problems. In the flipping Gaussian problem (Fig. 4), the recurrent policies outperform the other policies by a large margin, and their regret grows very slowly by the end of the trials. The conventional non-stationary policies perform very poorly, which illustrates the importance of the fact that the recurrent approach is able to *predict* rather than *react* in order to exploit periodicity. The hyperparameter sensitivity plot shows that the recurrent approach is also much more robust to hyperparameter choices than the feedforward approach (App. A.3.4).

In the flipping Bernoulli problem (App. A.3.2), the default NN outperforms the other non-random policies by an insignificant margin. This problem is much more difficult than the flipping Gaussian problem, as evidenced by the regret that grows quickly for every policy by the end of the trials. The hyperparameter sensitivity plot shows that the feedforward approach is arguably more robust to hyperparameter choices (App. A.3.4). Although the best hyperparameters for the recurrent approach outperform the best hyperparameters for the feedforward approach during hyperparameter search, the (longer and more numerous) definitive trials lead to the opposite conclusion. In Appendix A.3.2, we also present regret curves for the default policies in a stationary variant of this problem ($h \to \infty$), which are comparable to the curve for conventional posterior sampling for Bernoulli bandits [20].

In the sinusoidal Bernoulli problem (Fig. 5), the recurrent policies outperform every other policy by a significant margin, and their regret grows slowly by the end of the trials. D-UCB outperforms the default NN policy, which in turn outperforms the best NN policy. The hyperparameter sensitivy plot also heavily favors the recurrent approach (App. A.3.4).

In the circular Markov chain problem (Appendix A.3.2), the best RNN policy outperforms the remaining policies by a significant margin, and its regret grows slowly by the end of the trials. However, the default RNN policy exhibits an atypical large variance in regret. Because the hyperparameter sensitivity plot does not suggest a lack of robustness for the recurrent approach (App. A.3.4), we decided to investigate the cause of this variance, and noticed that the default RNN achieves worst than random performance across three of the ten trials. This suggests that the recurrent approach may benefit from a more careful initialization of recurrent neural network parameters. Unsurprisingly, the conventional non-stationary policies are not able to exploit the structure of this problem.

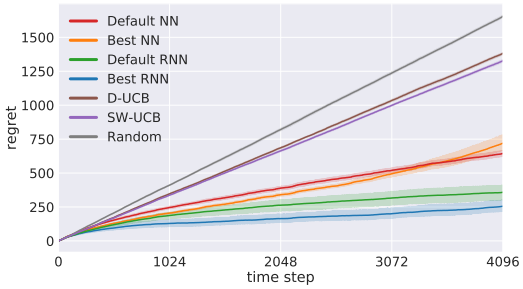

Figure 4: Flipping Gaussian.

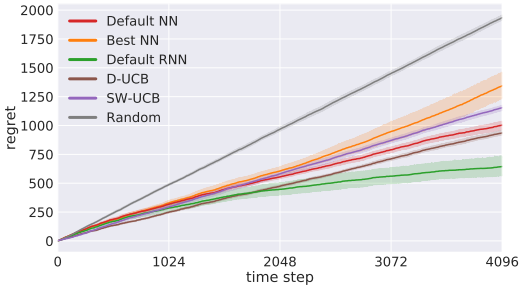

Figure 5: Sinusoidal Bernoulli.

Contextual bandit problems. In the flipping digits problem (App. A.3.3), the best RNN policy outperforms the other non-random policies, which achieve equivalent performance. This is a difficult problem, as evidenced by the regret that grows quickly for every policy by the end of the trials.

In the wall-following robot problem (App. A.3.3), the recurrent policies outperform the feedforward policies by a very small margin, and their regret grows slowly by the end of the dataset.

In the flipping vector problem (App. A.3.3), the combination of non-stationarity with high-dimensional observations proves too challenging for all policies.

In the low-frequency rotating vector problem ($f = 1/2048$, 2 rotations per trial, Fig. 6), the conventional non-stationary policies outperform every other policy. This is not surprising, since the corresponding algorithms have access to additional knowledge, and were designed specially for similar problems. More interestingly, in the high-frequency rotating vector problem ($f = 1/32$, 120 rotations per trial, Fig. 7), the recurrent policies significantly outperform every other policy, which once again illustrates the importance of prediction over reaction in order to exploit periodicity. The

success of conventional non-stationary policies is highly dependent on the so-called variation budget [18], which explains their poor performance in environments that change quickly.

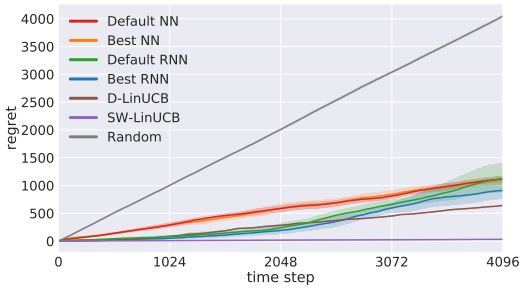

Figure 6: Rotating vector ($f = 2048^{-1}$).

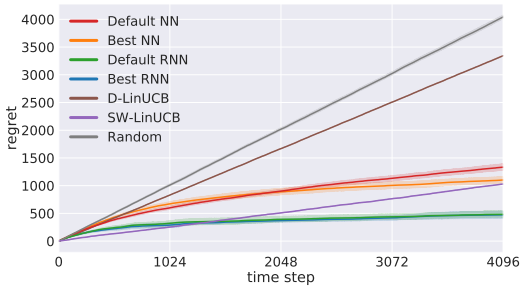

Figure 7: Rotating vector ($f = 32^{-1}$).

The hyperparameter sensitivity plots for contextual problems show that the recurrent approach is consistently more robust to hyperparameter choices than the feedforward approach (Appendix A.3.5).

7 Conclusion

We introduced an approach to non-stationary contextual bandit problems that learns to represent the relevant context for a decision based solely on the raw history of interactions between the agent and the environment. Prior to our work, solving such problems required carefully handcrafting a historical context, which could introduce spurious relationships or omit a convenient representation of crucial information; or employing conventional non-stationary bandit algorithms, whose assumptions had to coincide with the (typically unknown) underlying changes to the environment. Notably, our approach is also radically different from previous approaches that are able to exploit periodic or structured patterns in non-contextual non-stationary bandit problems.

The success of our approach relies on the strong assumption that the expected reward for any action at a given time step can be predicted as a (fixed but unknown) linear function of features extracted by a recurrent neural network that was trained to predict previous rewards. Consequently, it is difficult to provide theoretical performance guarantees comparable to those provided by conventional bandit algorithms, which is the most significant drawback of our approach. Nevertheless, our experiments on a diverse selection of contextual and non-contextual non-stationary problems show that our approach achieves satisfactory performance on problems that can be solved by conventional non-stationary bandit algorithms, while also being applicable when such algorithms fail completely. Our approach also consistently outperforms its feedforward counterpart, which requires handcrafting a historical context, even when its hyperparameters are fixed across very dissimilar environments. These findings make our approach particularly appealing when there is limited knowledge about a problem.

Another potential weakness of our approach is the computational cost of backpropagation through time, which is required to train the recurrent neural network. Fortunately, this issue may be mitigated by reducing the frequency of network training steps (as we have done), or by employing truncated backpropagation through time. These alternatives may compromise the quality of the learned contexts.

Because there are no standard benchmarks for non-stationary contextual bandit algorithms, we employed our own selection of problems, some of which were borrowed from previous work. Future work could focus on finding, creating, and adapting problems to further evaluate our approach. We are particularly interested in realistic applications and adversarial (adaptive) environments.

There are many possibilities for future work besides integrating our approach into real applications: combining alternative Bayesian recurrent neural network approaches with posterior sampling; combining recurrent neural-linear features with other contextual linear bandit algorithms; designing specialized recurrent neural network architectures; improving recurrent neural network parameter initialization; inferring the variance of the reward distribution; providing theoretical guarantees for restricted classes of problems; and comparing our approach with additional non-stationary contextual bandit algorithms.

Acknowledgments and Disclosure of Funding

We would like to thank Sjoerd van Steenkiste, Claire Vernade, 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] Djallel Bouneffouf and Irina Rish. A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*, 2019.

[2] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[3] 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.

[4] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135, 2013.

[5] Marc Abeille, Alessandro Lazaric, et al. Linear thompson sampling revisited. *Electronic Journal of Statistics*, 11(2):5165–5197, 2017.

[6] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2019.

[7] 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.

[8] 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.

[9] Yoan Russac, Olivier Cappé, and Aurélien Garivier. Algorithms for non-stationary generalized linear bandits. *arXiv preprint arXiv:2003.10113*, 2020.

[10] 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.

[11] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[12] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

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

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

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

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

[17] Aurélien Garivier and Eric Moulines. On upper-confidence bound policies for switching bandit problems. In *International Conference on Algorithmic Learning Theory*, pages 174–188. Springer, 2011.

[18] Yoan Russac, Claire Vernade, and Olivier Cappé. Weighted Linear Bandits for Non-Stationary Environments. In *NeurIPS 2019 - 33rd Conference on Neural Information Processing Systems*, Vancouver, Canada, December 2019.

[19] Wang Chi Cheung, David Simchi-Levi, and Ruihao Zhu. Learning to optimize under non-stationarity. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1079–1087. PMLR, 16–18 Apr 2019. URL `http://proceedings.mlr.press/v89/cheung19b.html`.

[20] 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.

[21] Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database, 2010. Available: http://yann.lecun.com/exdb/mnist.

[22] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. Available: http://archive.ics.uci.edu/ml.

[23] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela. Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *2009 6th Latin American Robotics Symposium (LARS 2009)*, pages 1–6, 2009.

A Experiments

A.1 Bandit problems

We performed experiments on four non-contextual bandit problems, which are described below.

Flipping Gaussian. 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. We chose $K = 8$ arms, $h = 10$, initial means in $\{0.1, 0.2, \ldots, 0.9\} \setminus \{0.5\}$, variance $s^2 = 0.1^2$.

Flipping Bernoulli. The mean of the Bernoulli reward for each arm k changes from p_k to $1 - p_k$ every h time steps. We chose $K = 8$ arms, $h = 10$, initial means in $\{0.1, 0.2, \ldots, 0.9\} \setminus \{0.5\}$.

Sinusoidal Bernoulli. The mean of the Bernoulli reward for each arm is a sinusoidal function of the current time step. The frequency of each function is the same across arms, but the phase is different. Concretely, the mean of the Bernoulli reward for each arm k at time step t is given by $p_k = 1/2 + \sin[2\pi f t + 2\pi(k-1)/K]/2$, where $K = 5$ is the number of arms and $f = 1/32$ is the frequency. This environment enables comparing our proposed approach with more conventional algorithms such as discounted UCB and sliding-window UCB, which were previously compared in a similar (albeit much simpler) environment [17].

Circular Markov chain. The mean of the Gaussian reward for every arm is μ, with corresponding variance s^2, except for a single arm whose Gaussian reward has a mean $\mu^* > \mu$. After this arm is chosen, it trades place with the next arm in a predefined cyclical order. Note that an action may affect the reward of other actions in the future. We chose $K = 8$ arms, common mean $\mu = 0$, best mean $\mu^* = 1$, and variance $s^2 = 0.05^2$.

We also performed experiments on four contextual bandit problems, which are described below.

Flipping digits. Each of the ten arms is labeled with a different digit. Each observation corresponds to an image of a digit from a subset of the MNIST dataset [21]. Initially, the mean of the Gaussian reward for every arm is μ, with corresponding variance s^2, except for the arm that is labeled with the digit depicted in last observation, whose mean is $\mu^* > \mu$. Every h time steps, the arm labeled with digit k becomes labeled with digit $9 - k$. We chose $h = 64$, common mean $\mu = 0$, best mean $\mu^* = 1$, and variance $s^2 = 0.05^2$.

Wall-following robot. This problem is derived from a sequential classification dataset [22, 23]. The observation x_{t-1} for time step $t - 1$ encodes readings of 24 sensors from a mobile robot. Each of the arms corresponds to one of four recognized movement patterns (forward, right turn, sharp right turn, left turn). The mean of the Gaussian reward for every arm is μ, with corresponding variance s^2, except for the arm that corresponds to the current movement pattern. Identifying a movement pattern may require combining observations across time steps. We chose common mean $\mu = 0$, best mean $\mu^* = 1$, and variance $s^2 = 0.05^2$.

The last two problems described below are non-stationary contextual linear bandit problems. These problems require a slightly modified implementation, which is detailed in Section 6.2.

Flipping vector. The observation x_{t-1} for time step $t - 1$ encodes a vector $\mathbf{x}_{t-1,k}$ for each action k. Each of these vectors is drawn from a finite set of unit vectors in $\mathbb{R}^d$, without replacement within a time step. The probability density function for the reward at time step t given the history h_{t-1}, the action a_t, and a parameter vector $\mathbf{w}_t$ is given by $p(r_t \mid h_{t-1}, a_t, \mathbf{w}_t) = \mathcal{N}(r_t \mid \mathbf{w}_t \cdot \mathbf{x}_{t-1,a_t}, s^2)$, where $s^2 > 0$ is a variance. The parameter vector $\mathbf{w}_2$ is a randomly chosen unit vector. Every h time steps, the parameter vector changes from $\mathbf{w}_t$ to $-\mathbf{w}_t$. In simple terms, the expected reward measures the alignment between an action-dependent vector and a vector that changes direction every h time steps. We chose $K = 25$ arms, $h = 64$, dimension $d = 50$, and variance $s^2 = 0.05^2$.

Rotating vector. The observation x_{t-1} for time step $t - 1$ encodes a vector $\mathbf{x}_{t-1,k}$ for each action k. Each of these vectors is drawn from a finite set of unit vectors in $\mathbb{R}^2$, without replacement within a time step. The probability density function for the reward at time step t given the history h_{t-1}, the action a_t, and a parameter vector $\mathbf{w}_t$ is given by $p(r_t \mid h_{t-1}, a_t, \mathbf{w}_t) = \mathcal{N}(r_t \mid \mathbf{w}_t \cdot \mathbf{x}_{t-1,a_t}, s^2)$, where $s^2 > 0$ is a variance. The parameter vector $\mathbf{w}_t$ is given by $\mathbf{w}_t = (\cos(2\pi f t), \sin(2\pi f t))$, where f is a frequency. In simple terms, the expected reward measures the alignment between an action-dependent vector and a vector rotating about the origin. We chose $K = 25$ arms, variance $s^2 = 0.05^2$, and different frequencies depending on the experiment. This problem is similar to a problem employed by Russac et al. [18], which enables comparing our proposed approach with more conventional algorithms such as discounted linear UCB [18] and sliding-window linear UCB [19].

A.2 Hyperparameter search

Preliminary experiments were employed to choose suitable hyperparameter ranges for the neural-linear approaches (Table 2). Note that more hyperparameter settings are considered for the feedforward approach (576) than for the recurrent approach (96), which is potentially advantageous for the feedforward approach.

Table 2: Hyperparameter grid and default hyperparameters for neural-linear approaches.

		Non-contextual problems	
Hyperparameter	Candidates	NN	RNN
Learning rate η	$\{0.001, 0.01, 0.1\}$	0.1	0.01
Number of epochs e by training step	$\{16, 64\}$	16	16
Interval q between training steps	$\{32, 128\}$	32	32
Assumed variance σ^2 of the reward	$\{0.1, 0.3\}$	0.1	0.1
Variance τ^2 of the prior distribution	$\{0.5, 1\}$	1	0.5
Units per layer	$\{(16, 16, 16), (32, 32, 32)\}$	(32,32,32)	(32,32,32)
Order n	$\{1, 4\}$	1	-
Number of sinusoidal units D	$\{1, 2, 4\}$	1	-

		Contextual problems	
Hyperparameter	Candidates	NN	RNN
Learning rate η	$\{0.001, 0.01, 0.1\}$	0.01	0.001
Number of epochs e by training step	$\{16, 64\}$	64	64
Interval q between training steps	$\{32, 128\}$	32	32
Assumed variance σ^2 of the reward	$\{0.1, 0.3\}$	0.1	0.3
Variance τ^2 of the prior distribution	$\{0.5, 1\}$	1	0.5
Units per layer	$\{(32, 32, 32), (64, 64, 64)\}$	(32,32,32)	(32,32,32)
Order n	$\{1, 4\}$	1	-
Number of sinusoidal units D	$\{2, 4, 8\}$	2	-

The definitive ten independent trials for each combination of non-contextual bandit problem and policy have double the length of the hyperparameter search trials in order to enable a more conclusive regret analysis. Note that it is quite difficult to establish appropriate trial lengths *before* hyperparameter search.

For discounted UCB, the hyperparameter grid contains candidates for the discount factor $\gamma \in \{0.8, 0.85, 0.9, 0.925, 0.95, 0.97, 0.98, 0.99, 0.995, 0.999\}$. For sliding-window UCB, the hyperparameter grid contains candidates for the window length $\tau \in \{5, 10, 25, 50, 75, 100, 150, 200, 250, 300\}$. For both algorithms, $\xi = 0.5$.

A.3 Results

A.3.1 Average final regret results

Table 3: Average final regret across ten trials for each combination of problem and policy.

Bandit problem	Best NN	Best RNN	Default NN	Default RNN	D-(L)UCB	SW-(L)UCB
Flipping Gaussian	719.35	**254.59**	643.38	357.58	1381.3	1327.27
Flipping Bernoulli	1267.9	1251.9	**1151**	1308.5	1199.9	1220.6
Sinusoidal Bernoulli	1344.94	**643.94**	1003.24	**643.94**	935.24	1154.64
C. Markov chain	2151.97	**895.23**	2151.97	2001.57	3005.1	3154.96
Flipping digits	3372.38	**3014.88**	3334.48	3314.28	-	-
Wall-following robot	2791.47	2383.47	2790.07	**2348.27**	-	-
Flipping vector	1082.86	**1052.22**	1082.86	**1052.22**	1073.92	1084.6
R. vector ($f = 32^{-1}$)	1102.34	**473.72**	1333.47	482.05	3345.18	1028.9
R. vector ($f = 2048^{-1}$)	1103.81	910.81	1121.37	1118.18	642.28	**31.52**

A.3.2 Regret curves: non-contextual problems

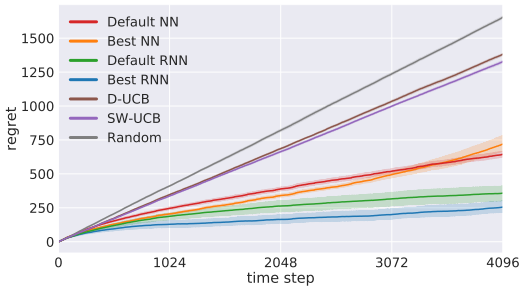

Figure 8: Flipping Gaussian.

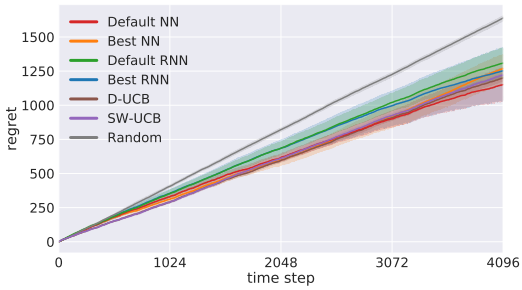

Figure 9: Flipping Bernoulli.

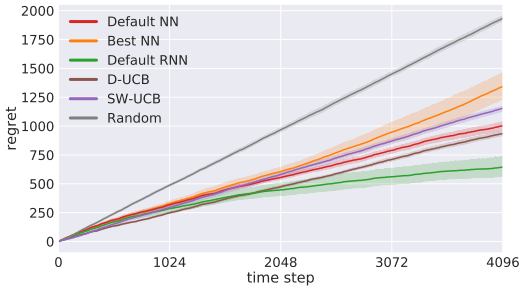

Figure 10: Sinusoidal Bernoulli.

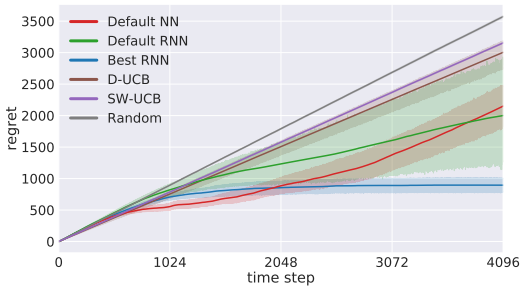

Figure 11: Circular Markov chain.

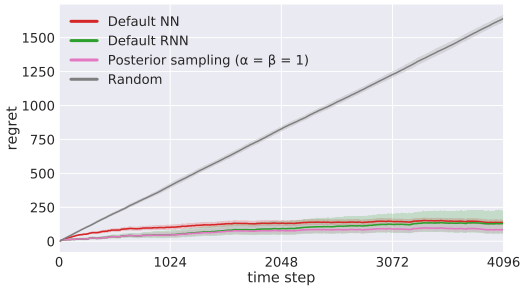

Figure 12: Stationary Bernoulli.

A.3.3 Regret curves: contextual problems

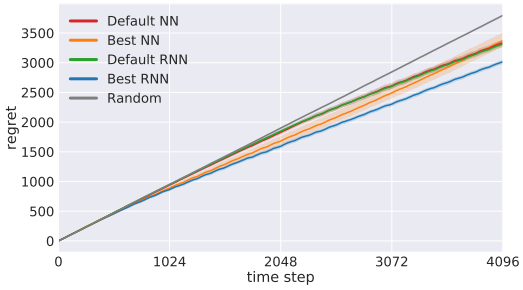

Figure 13: Flipping digits.

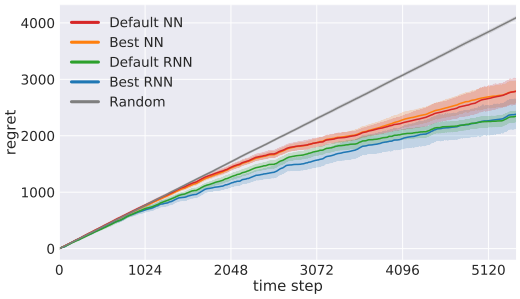

Figure 14: Wall-following robot.

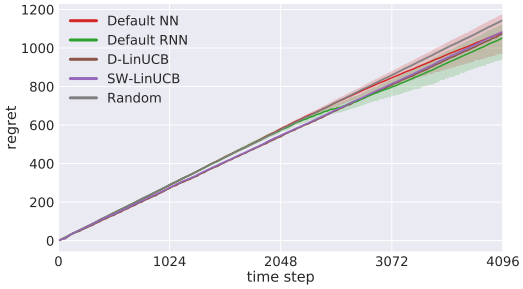

Figure 15: Flipping vector.

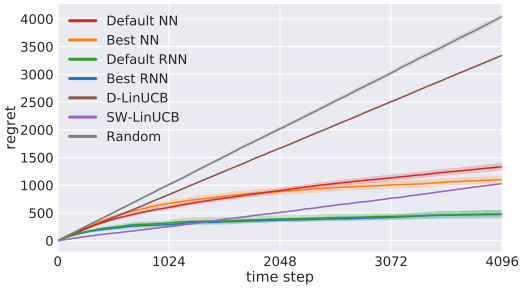

Figure 16: Rotating vector ($f = 32^{-1}$).

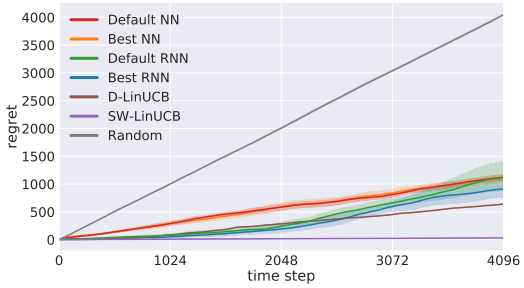

Figure 17: Rotating vector ($f = 2048^{-1}$).

A.3.4 Hyperparameter sensitivity plots: non-contextual problems

Figure 18: Flipping Gaussian.

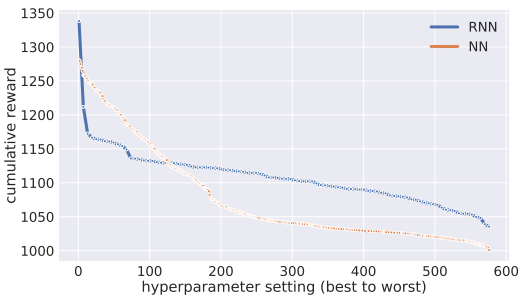

Figure 19: Flipping Bernoulli.

Figure 20: Sinusoidal Bernoulli.

Figure 21: Circular Markov chain.

A.3.5 Hyperparameter sensitivity plots: contextual problems

Figure 22: Flipping digits.

Figure 23: Wall-following robot.

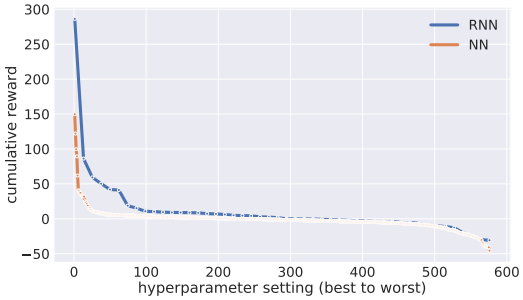

Figure 24: Flipping vector.

Figure 25: Rotating vector ($f = 32^{-1}$).

Figure 26: Rotating vector ($f = 2048^{-1}$).