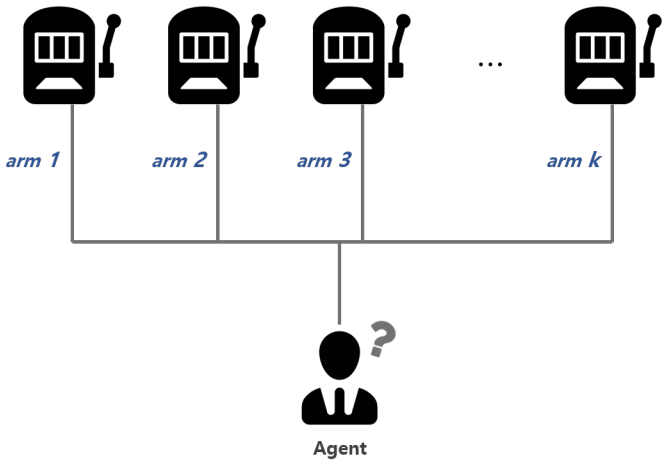
Neural Thompson Sampling

Nikolaos Tsikouras

September 6, 2022

1 Introduction

The multi–armed bandit problem has been a problem studied for several years throughout history and because of its use in various disciplines has enjoyed an extensive literature, [35]. The main goal of the problem is to select between competing choices in a way that maximizes the expected reward in the long run. The complexity occurs since each choice's properties is not fully known but can be estimated as we get more information.



The multi–armed bandit problem belongs in the Reinforcement Learning area of Machine Learning where an agent learns to choose one of k actions (pull an arm), in order to maximize cumulative reward by rewarding desired behaviors and/or punishing undesired ones. In this setting, the agent is required to solve the exploration–exploitation trade-off, where they have to choose between getting something close to what they expect ("exploitation") and choosing something they are not as certain but for which they could possibly learn more ("exploration"), [38].

There are several ways one could balance this exploration–exploitation tradeoff. The simplest approach is *greedy* algorithms, where an action is chosen to simply maximize the next reward, [35]. Another common approach is the *UCB1* algorithm, which uses the Chernoff-Hoeffing inequality to aid in choosing an action, [21]. The *Thompson Sampling* (TS) algorithm is a Bayesian approach, which is also very common because of its simplicity and strong mathematical guarantees, [35].

In this report we are going to present an extension to the classical Thompson Sampling, the *Neural Thompson Sampling* (NeuralTS), [42]. This algorithm is used in the contextual bandit problem, which we are going to provide a detailed explanation in a later section. The novelty from its predecessors is that to estimate the reward the authors used a fully connected neural network.

NeuralTS is not the first algorithm to use neural networks models to apply neural networks to in the multi–armed bandit context, [11, 28]. However, it is the first neural network based Thompson Sampling algorithm with near–optimal regret bounds, which is what caught our attention and motivated us for this report. Additionally, it manages to solve some of the disadvantages its predecessors had, while also having good empirical performance.

The rest of this report is organized as follows. In Section 2 we give a brief background of multi–armed bandits and give a description of contextual bandits. In Section 3 we give a detailed explanation of Thompson Sampling and provide the advantages and limitations of it. In Section 4 we introduce the Neural Thompson Sampling algorithm and provide details about the regret analysis and the regret bound, as well as some experiments the authors used to test the algorithm's performance. Finally, in Section 5 we provide some numerical experiments and give some empirical results on how the number of gradient descent steps affect performance. Additionally, we introduce Adaptive Neural Thompson Sampling, an extension to Neural Thompson Sampling, and we empirically test that it has similar performance to it, while enjoying a smaller computational cost.

2 Multi–Armed Bandits

As we mentioned in the introduction the high–level view of the multi–armed bandit problem is one in which an automated agent is tasked to gain new knowledge by exploring its environment and to exploit its current, attested knowledge, [21]. It is a very simple theoretical formulation to tackle the exploration–exploitation tradeoff. We will provide a brief overview in this section, for a more comprehensive description the reader is referred to [10].

2.1 Mathematical Formulation

A (stochastic) bandit problem consists of a set of a collection of distributions $\nu = (P_a : a \in \mathcal{A})$, where $\mathcal{A}$ is the set of available actions, [24]. In general, the distributions are not known to the agent. At every round $t \in \{1, \dots, T\}$ (usually T is finite), the agent follows some strategy and selects an action (pulls an arm) $a_t \in \mathcal{A}$, which is fed to the environment, and afterwards the environment samples a reward $X_{t,a_t} \sim \nu_{a_t}$ and returns it to the agent.

We can think of the agent playing many slot machines each with a single arm to pull and each arm having an underlying reward distribution. The agent's goal is to maximise their reward by pulling the arms over many rounds. It is easily understood that the agent has a two–fold goal. First, they have to figure out which distribution (arm) has the highest expected value and secondly they have to maximise their rewards over a number of rounds. We note that the interaction between the agent and environment induces a probability measure on the sequence of outcomes $A_1, X_1, A_2, X_2, \dots, A_n, X_T$, [24].

2.2 Regret

The main objective of an agent is to identify a strategy (or policy) for sequentially selecting arms which would maximise the expected cumulative reward over T rounds. To be confident about how good a policy is, we require some kind of a quantitative performance measure.

There exist several performance measures for any of the different policies the agent follows. One of the most popular ones is the *regret* of suboptimal play, [12], where we compare the performance of a chosen policy to that of a (optimal) policy where an agent picks the best arm every round. More

specifically, given $K \geq 2$ arms and corresponding (unknown) rewards $X_{i,1}, X_{i,2}, \ldots, i = 1, \ldots, K$, we are interested in a policy that at time t pulls arm a_t and receives reward $X_{a_t,t}$. For any fixed number of rounds T the regret of such a policy π is defined as, [12],

$$Reg_\pi(T) = \max_{i=1,\ldots,K} \sum_{t=1}^{T} X_{i,t} - \sum_{t=1}^{T} X_{a_t,t}, \tag{1}$$

However, generally, both the rewards and agent's choices might be stochastic. This gives rise to two new notions of averaged regret: the *expected regret*, [12],

$$\mathbb{E}Reg_\pi(T) = \mathbb{E}\left[\max_{i=1,\ldots,K} \sum_{t=1}^{T} X_{i,t} - \sum_{t=1}^{T} X_{a_t,t}\right], \tag{2}$$

and the *pseudo–regret*, [12],

$$\overline{Reg}_\pi(T) = \max_{i=1,\ldots,K} \mathbb{E}\left[\sum_{t=1}^{T} X_{i,t} - \sum_{t=1}^{T} X_{a_t,t}\right], \tag{3}$$

where the expectation in Equations 2, 3 is taken with respect to the probability measure on outcomes induced by the interaction of the actions and rewards we mentioned above, [24]. We note that pseudo–regret is a weaker concept to regret, as the comparison is to the optimal action in expectation, [12]. Additionally, for completeness we state that $\overline{Reg}_\pi \leq \mathbb{E}Reg_\pi$. The goal is to minimise regret which is equivalent to maximising the expected reward.

A more intuitive form of the pseudo–regret (which from now on we will be referring to as regret) is, for any fixed number of rounds T,

$$Reg_\pi(T) = T\mu^* - \sum_{t=1}^{T} \mathbb{E}_\pi(\mu_{a_t}), \tag{4}$$

where $\mu_k = \mathbb{E}(X_{k,t}), \forall k \in \{1, \ldots, K\}$, $\mu^* = \max_{k \in 1,\ldots,K} \mu_k$ and the expectation in Equation 4 is again taken as previously.

This formulation gives an intuitive meaning on why one might use this quantity as a measure of performance and where the term "regret" comes from. To put it simply it is the total amount of maximum expected reward minus the total amount of the expected reward with respect to the chosen policy π. Therefore this quantity is the regret of an agent to the loss due to the fact that the globally optimal policy is not followed all times.

The following Lemma gives us some more intuition on why regret might be a "valid" performance measure, [24].

Lemma 1. *Let ν be a stochastic bandit environment and T being a fixed number of rounds. Then,*

- *$Reg(T) \geq 0$ for every strategy (or policy) π.*
- *Policy π choosing $a_t \in \arg\max_a \mu_a, \forall t$ gives $Reg(T) = 0$.*
- *If $Reg(T) = 0$ for policy π, then $\mathbb{P}(\mu_{a_t} = \mu^*) = 1, \forall t$.*

The third point from Lemma 1 states that to achieve zero regret the agent should know the underlying distributions (so as to know the optimal arm) of the bandit. In practice however, we can understand that this is not feasible, thus achieving zero regret is not possible, therefore in a real setting, regret will grow as a function of T.

2.3 Understanding the Optimal Policy

In general for any bandit problem, playing a suboptimal arm depends on the number of rounds of the horizon T. Intuitively, if we have 1 round left and we are certain "enough" that a specific arm would return the largest reward it would make sense to pull that arm, however if we have 1000 rounds, say, left then if might be sensible to explore rather than exploit. To put this in a more formal way we define the suboptimality gap for arm a, $\Delta_a = \mu^* - \mu_a$. The larger the time horizon T, the more we ought to be exploring. The smaller Δ_a, the more evidence we would require to reject arm a. The authors in [23], found out that $O(\Delta_a \ln(T))$, has the right balance between exploration and exploitation. We will further explore this in the next subsection.

Additionally, let $T_a(T) = \sum_{s=1}^{T} \mathbb{1}\{a_s = a\}$ be the amount of times the agent selected action a after T rounds. We now give the Lemma which provides understanding of the optimal policy, [24],

Lemma 2. *For any policy π and a stochastic bandit with $\mathcal{A}$ the set of available actions and horizon T, the regret of policy π satisfies,*

$$Reg_\pi(T) = \sum_{a \in \mathcal{A}} \Delta_a \mathbb{E}_\pi[T_a(T)]. \tag{5}$$

Lemma 2 gives a decomposition of the regret in terms of the loss created by using any of the arms. It gives us an idea on how to create an optimal strategy, that is, to keep regret low, one should try to minimise the weighted sum of expected action counts, where the weights are the suboptimality gaps corresponding to each action. It tells us that we ought to use the arm with a greater suboptimality gap proportionally fewer times in the long run.

2.4 Lower Bounds on the Regret

Often it is hard to explicitly calculate the regret of a bandit algorithm. However, many times one can estimate it through simulation, but even better, one could obtain theoretical guarantees for the performance of an algorithm. Furthermore, there exist cases where it is possible to derive lower bounds on the regret that should hold for any bandit algorithm in that setting, [33]. It is also possible to derive an upper bound on the regret of a particular algorithm, [33], and if it coincides with the aforementioned lower bound, then this algorithm is considered to be optimal in the specific setting. There are two main types of regret that are of use in this problem, the *problem dependent* and the *problem independent* regret, [33].

The problem dependent regret of a bandit depends on the specific elements of the problem setting we are working with, for example the reward means μ_a. Under mild assumptions on the distributions of the rewards the authors in [23], showed that the regret (Equation 5) of any algorithm for the multi–armed bandit satisfies,

$$\lim_{T \to \infty} \inf \frac{Reg_\pi(T)}{\log T} \geq \sum_{j=1; j \neq j^*} \frac{\Delta_j}{KL(\nu_j, \nu_{j^*})},$$

where $KL(\nu_j, \nu_{j^*})$ is the Kullback–Leibler divergence between the reward distribution of the optimal arm and the reward distribution of arm j. This shows that the regret for such a problem should grow at least logarithmically in the number of rounds. Hence, in this specific setting there exists no algorithm with a smaller problem dependent rate of regret, and we should therefore construct algorithms with regret of this order.

Often it is not interesting to consider the regret of a bandit algorithm for a specific problem setting. In a real setting, the expected reward μ_a, for each arm is unknown and our goal might be to have some regret bounds that hold regardless of the reward distributions or the problem setting we are facing. This is where the *problem independent* or *worst case regret* might be useful, [33]. The author in [8] gave the following lower bounds on the problem independent regret of any bandit algorithm,

$$Reg_\pi(T) \geq \frac{1}{20}\min\{\sqrt{KT}, T\}$$

Thus for an algorithm to be optimal in a more general setting, we should aim to construct algorithms with worst case regret which increase based on $O(\sqrt{T})$. We note here that, since this is referring to a more general setting, and it is the worst case, it is natural that this is always larger than the problem dependent regret bound.

2.5 Contextual Bandits

There are several different versions of the multi–armed bandit problem, one of which is the contextual multi–armed bandit, which we are going to be interested in this report. In this problem, at every round $t = 1, \ldots, T$, the agent is again required to pull one out of K arms. The difference here is that before pulling any arm they receive a d–dimensional feature vector, b_i for each arm i, which we call *context*. The context, together with the arm pulled and the observed reward up to time $t-1$, are gathered in a vector which we call history $\mathcal{H}_{t-1}$, [6]. More specifically, $\mathcal{H}_{t-1} = \{a(\tau), r_{a(\tau)}(\tau), b_i(\tau), i = 1, \ldots, K, \tau = 1, \ldots, t-1\}$, where $a(\tau)$ is the arm pulled at round τ. The agent uses the history, which contains knowledge about the reward distribution and the current context, to make an informed choice of which arm to pull in every round. Over time, the agent's goal is to collect sufficient information about the interaction of rewards and context, so as to assist in pulling the optimal arm.

To put it more mathematically, in the contextual multi–armed bandit problem, there is a relationship between the context and the reward which we assume can be modelled by some function h, i.e. for any $1 \leq t \leq T$ and $1 \leq i \leq K$, we have,

$$r_{t,i} = h(b_{t,i}) + \epsilon_{t,i},$$

where a widely adapted assumption on the noise $\epsilon_{t,i}$ is that it follows an R–sub–Gaussian, [6], with constant $R > 0$. As we shall see in the next subsections, h does not have a limitation on what it could be, it can be parametric as well as non–parametric, and each form gives rise to a different problem as well as different limitations.

Contextual bandits are very relevant as the author in [41], notes that in the majority of sequential decision making problems, there probably exists additional information which can be of use for decision making. An example would be in a clinical trial where we are interested in two drugs, and we might have the participants demographic or genetic information available, [38]. If this is true the objective could be to map user features in one of the available actions, that being either of the two drugs in this example.

Contextual bandits saw interest largely due to web personalization problems, [38]. They are widely used in problems such as learning to optimally schedule internet banner advertisement, [2], providing an approach to personalized news article recommendation, [26] or even in robotic control, [30]. Furthermore, the authors in [38] believe that many of the methods already developed for personalizing ads will start to be used on the emergence field of mobile health.

It is therefore understood why this field has enjoyed a growing literature, with new algorithms emerging frequently. One of the most effective, simple and widely used techniques is Thompson Sampling, [39], which we are going to introduce with more details in a subsequent section. Two influential papers which have been the basis of a lot of subsequent results in contextual bandits are [1, 16]. The former provided a UCB algorithm for the linear stochastic bandit, while the latter provided a UCB algorithm for the generalized linear bandit. Below we will briefly introduce the linear and generalized linear bandit setting.

2.5.1 Contextual Bandits with Linear Payoffs

It is often the case, where with suitable context, the reward function of a contextual bandit can be approximated accurately with a linear function. In this setting, we assume that there are K arms and for every round $t = 1, \dots, T$ we have a context vector $b_i(t) \in \mathbb{R}^d$, which is given for every arm i. Given $b_i(t)$, the reward is then generated from an unknown distribution with mean $b_i(t)^\intercal \mu$, where $\mu \in \mathbb{R}^d$ is a fixed but unknown parameter, [6]. This setting is expressive enough to provide good results in real–world problems without making the problem intractable, [15].

An application of a linear multi–armed bandit is the "optimal" Internet advertisement problem. The authors in [3] argue that given context such as keywords searched by the user, age, sex, the domain, ad genres, etc, the probability of clicking on a certain advertisement could be roughly approximated by a linear function of combinations of the aforementioned attributes.

2.6 Contextual Bandits with Generalized Linear Payoffs

Most of the times however the reward function would not be able to be approximated by a linear function, which is natural in the real world. For this reason, one should use a richer model to the linear one, for example the generalized linear. Following the same notation as before, we say that a multi–armed bandit has generalized linear payoffs if the expectation of the reward at round t is conditionally independent of past rewards and is given by $g(b_i(t)^\intercal \mu)$, where $g(\cdot)$ is a real–valued, possibly non–linear function, commonly referred to as the link or inverse function, [16].

It is clear that the generalized linear payoff model allows for much more general reward function approximations. For instance when working with a binary reward model, this would lead to the *logistic regression model*, where a suitable choice for the link function is $\mu(x) = \exp(x)/(1+\exp(x))$. Additionally, if one is working with with integer valued reward functions, this would lead to the *Poisson regression model*, where a suitable link function is $\mu(x) = \exp(x)$.

Similar to before, an application of a generalized linear payoff multi–armed bandit would be to predict the conversion rate of a customer of an Internet advertisement, [36].

3 Thompson Sampling

As mentioned already Thompson Sampling is a simple and effective algorithm which is widely used for tackling multi–armed bandit problems. The initial idea is traced back to 1933, [39], but it started becoming popular by this empirical paper, [13]. The main idea of this algorithm is to compute the posterior distribution of each arm for the present context, draw a sample from every distribution and choose arm with the largest sampled value, and continue updating the posteriors until the final round.

Although this report is focused mainly on an extension to Thompson Sampling, we will briefly give a description to the two other main families of algorithms for completeness.

First, the ϵ–greedy algorithm, which is widely used due to its simplicity in implementation. At every round t the algorithm selects the arm with the highest empirical mean, i.e. the "greedy" arm, with probability $1-\epsilon$. Otherwise, the algorithm will select a random arm with probability ϵ, [21].

The next family of algorithms is the Upper Confidence Bounds (UCB) one. The intuition behind this class of algorithms, is to use an optimistic estimate of the expected reward for each arm given any information accessible, [33]. Subsequently, pulling the arm with the largest aforementioned estimate, will lead to a policy where the arms selected will have either high reward or high uncertainty and thus should be explored more.

3.1 Thompson Sampling Algorithm

We now give the main structure of Thompson Sampling. We are interested in the setting where at each round we have a context b (optionally) and a set of actions $\mathcal{A}$. Next, we choose an action $a \in \mathcal{A}$ and receive a reward r. The goal is to find a policy that sequentially selects actions to maximize cumulative reward over T rounds.

Thompson Sampling is a Bayesian algorithm. The likelihood function $P(r|a,b,\theta)$, depending on some parameters θ is made out of the set of past observations D, which is the triplets (b_i, a_i, r_i), [13]. If $P(\theta)$ is the prior distribution on the parameters θ, the Bayes rule gives the posterior distribution $P(\theta|D) \propto \prod P(r_i|a_i,b_i,\theta)P(\theta)$. The general TS algorithm is given in Algorithm 1, [13].

In the contextual bandit case, which is the one we are interested in, the reward the agent receives is a stochastic function of the action, the context and the unknown true parameter, say θ^*. The aim, ideally, is to maximize the expected reward $\mathbb{E}(r|a,b,\theta^*)$. In practice θ^* is unknown, therefore we have to integrate it out. Since we are interested in balancing exploration and exploitation, the probability matching heuristic instructs us to select action a according to its probability being optimal, [13]. Action a is thus picked with probability,

$$\int \mathbb{1}\left[\mathbb{E}(r|a,b,\theta) = \max_{a'} \mathbb{E}(r|a',b,\theta)\right] P(\theta|D)d\theta. \tag{6}$$

The integral in Equation 6 does not need to be evaluated exactly, it suffices to draw a random parameter θ at each round as explained in Algorithm 1, [13]. It is straightforward to adapt the algorithm to the required case.

Algorithm 1 Thompson Sampling

Require: Prior distribution $P(\theta)$ for the K arms

$D = \emptyset$
for $t = 1, \ldots, T$
  Receive context b_t
  Draw θ^t from $P(\theta|D)$
  Select action $a_t = \arg\max_a \mathbb{E}_r(r|x_t, a, \theta^t)$
  Get reward r_t
  $D = D \cup (x_t, a_t, r_t)$
end for
Return D

It has been shown empirically that TS achieves regret analogous to the optimal regret lower bound,

[13], for the basic stochastic multi–armed bandit problem. Although UCB algorithms are simple to implement and have good theoretical regret bounds, [26], it has been shown that TS holds better empirical performance in many simulated and real-world settings without being very complicated, [31]. Additionally, the authors in [13] showed that TS is more robust to delayed feedback than other methods especially when the delay is long. Also, it is noteworthy to mention that the authors in [20] provide empirical evidence which shows that TS also outperforms the "best optimal" versions of UCB and ϵ–greedy in several applications.

3.2 Theoretical Guarantees for Thompson Sampling

We have mentioned before that TS enjoys some strong theoretical guarantees on the regret bounds. It would be very hard, and probably without much use, to state every result for every different setting. The literature is big and is still growing, however we are going to present the result for the most basic contextual and non–contextual setting which shifted researchers' attention to it and made it more popular in the literature. For the non–contextual case we follow the authors in [5] and provide two different formulations of TS which allow for near–optimal problem–independent regret bound.

3.2.1 Thompson Sampling with a Beta and Gaussian prior

Suppose we are interested in the typical Bernoulli bandit problem, where the reward is either 0 or 1 and the likelihood for the reward 1, i.e. the probability of success, at each round is μ_i. A natural prior distribution to use it the Beta distribution since it is a conjugate to the Bernoulli distribution.

Then, in the N–armed stochastic bandit setting, this version of TS, has problem independent regret bounded by $O(\sqrt{NT \ln T})$, where T is the number of rounds, [5].

Suppose now, that $k_i(t)$ denotes the number of times arm i has been pulled until time $t-1$ and $a_i(t)$ denote the arm played at time t. Let $r_i(t)$ denote the reward at time t of arm i, and define $\mu_i(t) = \frac{\sum_{w=1:i(w)=i}^{t-1} r_i(t)}{k_i(t)+1}$. We also assume that the likelihood of the reward is $N(\mu_i, 1)$ and that the prior is given by $N\left(\mu_i(t+1), \frac{1}{k_i(t+1)+1}\right)$

Then, in the N–armed stochastic bandit setting, this version of TS, has problem independent regret bounded by $O(\sqrt{NT \ln N})$, where $T \geq N$ is again the number of rounds, [5].

Before the work of the authors in [5] there was no strong theoretical regret analysis for Thompson Sampling. However, it already had created significant interest after empirical studies showed great performance. This paper, to the best of our knowledge, solidified Thompson Sampling as an approach and has been the basis of a lot of subsequent results in Thompson Sampling in several different settings. One of those, and the one we are interested in, is the contextual bandit setting.

3.2.2 Thompson Sampling in the Contextual Bandit Setting

As mentioned already, a simple but widely studied version of contextual bandits is the multi– armed bandit with a linear payoff function. Thus, it was natural for the first theoretical guarantees for the contextual version of Thompson Sampling to be given for this setting.

To give those guarantees the authors in [6] provided a generalization of the Thompson Sampling algorithm. They used a Gaussian likelihood and prior to design their TS algorithm. More specifically the likelihood for the reward at round t, $r_i(t)$, given context $b_i(t)$ and fixed parameter μ to is a $N(b_i(t)^\intercal \mu, v^2)$, where $v = R\sqrt{\frac{24}{\epsilon} d \ln\left(\frac{1}{\delta}\right)}$, with $\epsilon \in (0,1)$, R is a constant to ensure that the quantity $r_i(t) - b_i(t)^\intercal \mu$ is

R–sub–Gaussian, [6], and δ is a constant indicating probability in the regret analysis. It is not in scope of this report to get into details for these constants. Define,

$$B(t) = I_d + \sum_{\tau=1}^{t-1} b_{a(\tau)}(\tau) b_{a(\tau)}(\tau)^\intercal$$
$$\hat{\mu}(t) = B(t)^{-1} \left(\sum_{\tau=1}^{t-1} b_{a(\tau)}(\tau) r_{a(\tau)}(\tau) \right),$$

then the prior for μ at round t is given by $\mathcal{N}(\hat{\mu}(t), v^2 B(t)^{-1})$, which using Bayes' theorem gives the posterior at round $t+1$ to be, $\mathcal{N}(\hat{\mu}(t+1), v^2 B(t+1)^{-1})$. Following this procedure, one could easily adapt Algorithm 1 to this setting.

Then, in the stochastic contextual bandit with linear reward function, with probability $1-\delta$, this version of Thompson Sampling has regret after T rounds bounded by $O\left(\frac{d^2}{\epsilon}\sqrt{T^{1+\epsilon}(\ln(Td)\ln\frac{1}{\delta})}\right)$, for any $\epsilon \in (0,1), \delta \in (0,1)$, [6].

Although this bound was not better than the best available regret bounds at the time, it provided researchers new tools for regret analysis of Thompson Sampling and led researchers into further improvements and extensions.

4 Literature Review on Neural Network Methods

As we have already mentioned several times in this report, every contextual multi–armed bandit setting is associated with a function which approximates the reward function given some context b. We have already seen some common ones such as linear models [7], generalized linear models, [16], but of course there exist various others reward models proposed in the literature (i.e. kernel–based models, [40]). Recently in the reinforcement learning community, scientists have increasingly started to make use of deep neural networks in sequential decision making, [34, 43]. Neural networks are a powerful and most importantly flexible approximators, and they have been proven to work extremely well in many situations, [9, 18]. Scientists have seen that neural networks allow for a more powerful approximation of the underlying reward distribution, [17], some examples follow.

The authors in [34] proposed NeuralLinear which uses a neural network and in each round chooses the best arm based on a Bayesian linear regression on top of the last network layer, and is a good attempt in bringing strong representational power coupled with uncertainty estimates. The authors in [22] proposed DeepFPL, which uses perturbed data to train the neural network and uses the neural network output to choose the best arm every round. The authors in [43] proposed NeuralUCB, which uses the representational power of neural networks to construct an upper confidence bound (UCB).

The issues with the aforementioned algorithms are that,

- Despite the empirical successes, there are no strong regret mathematical guarantees for the first two.
- They are computationally expensive as they require updating the parameters of the neural networks in each round.
- They are not very robust to the batched multi–armed problem, which is quite common in practice, [25, 32].

There have been some extensions which solve some of these issues, see for example BatchedNeuralUCB, [17]. The main focus of this report however, is a neural network–based Thompson Sampling, called Neural Thompson Sampling, which we introduce in the next subsection.

4.1 Neural Thompson Sampling

Assuming the total number of rounds T is known, we consider K contextual vectors $\{\boldsymbol{x}_{t,k} \in \mathbb{R}^d | k \in K\}$ (note we changed the notation from b to x, to follow the widely used notation for neural networks). The agent pulls arm a_t and receives reward r_{t,a_t}. The goal as it has always been throughout this report is to minimize the regret.

To estimate this unknown reward r_{t,a_t}, the authors in [42] propose to use a fully connected neural network $f(\boldsymbol{x};\boldsymbol{\theta})$ with depth $L \geq 2$, where $\boldsymbol{x}$ is the context. This neural network is defined recursively as follows,

$$
\begin{aligned}
f_1 &= \boldsymbol{W}_1 \boldsymbol{x} \\
f_l &= \boldsymbol{W}_l \mathrm{ReLU}(f_{l-1}), 2 \leq l \leq L \\
f(\boldsymbol{x};\boldsymbol{\theta}) &= \sqrt{m} f_L,
\end{aligned}
$$

where $\mathrm{ReLU}(x) := \max\{x, 0\}$, m is the width of the neural network, $\boldsymbol{W}_1 \in \mathbb{R}^{m \times d}, \boldsymbol{W}_l \in \mathbb{R}^{m \times m}, 2 \leq l < L, W_L \in \mathbb{R}^{1 \times m}$ are the weights, and $\boldsymbol{\theta} = (\mathrm{vec}(\boldsymbol{W}_1); \ldots; \mathrm{vec}(\boldsymbol{W}_L) \in \mathbb{R}^p$ is the collection of parameters of the neural network, $p = dm + m^2(L-2) + m$. We also define $\boldsymbol{g}(\boldsymbol{x};\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta})$ to be the gradient of $f(\boldsymbol{x};\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

We provide a pseudocode of the Neural Thompson Sampling in Algorithm 2. This algorithm carries a Gaussian distribution for each arm's reward distribution. To select an arm, it samples a reward from the posterior distribution (i.e. the distribution in line 6) from each arm, and picks the one with the maximum value. After choosing an arm, and receiving a reward, it updates the parameters of the neural network via gradient descent. The mean of the posterior distribution is set to be the output of the neural network, the parameters of which are the solution to the problem:

$$
\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) = \sum_{i=1}^{t} [f(\boldsymbol{x}_{i,a_i};\boldsymbol{\theta}) - r_{i,a_i}]^2/2 + m\lambda ||\boldsymbol{\theta} - \boldsymbol{\theta}_0||_2^2/2. \tag{7}
$$

Equation 7 is a l_2–regularized square loss minimization problem, where the regularization term is centered around a randomly initialized parameter $\boldsymbol{\theta}_0$. There are a couple of ways one could solve this minimization problem (Adam, Adagrad, RMSProp), the authors in the original paper chose gradient descent.

There are a couple of things we have to mention here. Due to its nature, namely sampling from the posterior distribution of the scalar reward, instead of the network parameters like some of each adversaries, it is simpler and more efficient as the amount of parameters can be quite large in practice, [42].

Additionally, the authors mention that one could use stochastic gradient descent to solve the minimization problem in Equation 7 and have similar theoretical guarantees to when using vanilla gradient descent.

Algorithm 2 Neural Thompson Sampling (NeuralTS)

Require: Number of rounds T, exploration variance ν, network width m, regularization parameter λ.

Set $\boldsymbol{U}_0 = \lambda \boldsymbol{I}$.

Initialize $\boldsymbol{\theta}_0 = (\text{vec}(\boldsymbol{W}_1); \dots; \text{vec}(\boldsymbol{W}_L)) \in \mathbb{R}^p$, where $\forall 1 \leq l \leq L-1, \boldsymbol{W}_l = (\boldsymbol{W}, \boldsymbol{0}; \boldsymbol{0}, \boldsymbol{W})$, each entry of $\boldsymbol{W}$ is generated independently from $N(0, 4/m)$ and $\boldsymbol{W}_L = (\boldsymbol{w}^\intercal, -\boldsymbol{w}^\intercal)$, where each entry of w is generated independently from $N(0, 2/m)$.

for $t = 1, \dots, T$

  for $k = 1, \dots, K$

    $\sigma^2_{t,k} = \lambda \boldsymbol{g}^\intercal(\boldsymbol{x}_{t,k}; \boldsymbol{\theta}_{t-1}) \boldsymbol{U}_{t-1}^{-1} \boldsymbol{g}^\intercal(\boldsymbol{x}_{t,k}; \boldsymbol{\theta}_{t-1})/m$.

    Sample estimated reward $\tilde{r}_{t,k} \sim \mathcal{N}(f(\boldsymbol{x}_{t,k}; \boldsymbol{\theta}_{t-1}), \nu^2 \sigma^2_{t,k})$.

  end for

  Pull arm a_t and receive reward r_{t,a_t}, where $a_t = \arg\max_a \tilde{r}_{t,a}$.

  Set $\boldsymbol{\theta}_t$ to be the output of gradient descent for solving Equation 7.

  $\boldsymbol{U}_t = \boldsymbol{U}_{t-1} + \boldsymbol{g}(\boldsymbol{x}_{t,a_t}; \boldsymbol{\theta}_t) \boldsymbol{g}(\boldsymbol{x}_{t,a_t}; \boldsymbol{\theta}_t)^\intercal / m$.

end for

4.2 Regret Bound for Neural Thompson Sampling

To give the bound for NeuralTS we have to define the neural tangent kernel and the effective dimension.

Definition 4.1. *[19] Define,*

$$\begin{aligned}
\tilde{H}^{(1)}_{i,j} &= \Sigma^{(1)}_{i,j} = \langle \boldsymbol{x}^i, \boldsymbol{x}^j \rangle, \boldsymbol{A} i, j^{(l)} = \begin{pmatrix} \Sigma^l_{i,i} & \Sigma^l_{i,j} \\ \Sigma^l_{i,j} & \Sigma^l_{j,j} \end{pmatrix}, \\
\Sigma^{(l+1)}_{i,j} &= 2\mathbb{E}_{(u,v)\sim N(0, \boldsymbol{A}^{(l)}_{i,j})} \max\{u, 0\} \max\{v, 0\}, \\
\tilde{\boldsymbol{H}}^{(l+1)}_{i,j} &= 2\tilde{\boldsymbol{H}}^{(l)}_{i,j} \mathbb{E}_{(u,v)\sim N(0, \boldsymbol{A}^{(l)}_{i,j})} \mathbb{1}(u \geq 0) \mathbb{1}(v \geq 0) + \Sigma^{(l+1)}_{i,j}.
\end{aligned}$$

Then $\boldsymbol{H} = (\tilde{\boldsymbol{H}}^{(L)} + \Sigma^{(L)})/2$ is called the neural tangent kernel (NTK) matrix on the context set.

This NTK technique is a well–known method which connects the deep neural network theory with kernel methods. It allows one to describe the complexity of the neural network. More specifically, this is done by the effective dimension.

Definition 4.2. *[42] The effective definition $\tilde{d}$ of matrix $\boldsymbol{H}$ with regularization parameter λ is defined as,*

$$\tilde{d} = \frac{\log \det(\boldsymbol{I} + \boldsymbol{H}/\lambda)}{\log(1 + TK/\lambda)}$$

The authors in [42] state that the NeuralTS algorithm enjoys a $\tilde{O}(\tilde{d}\sqrt{T})$ regret, where $\tilde{d}$ is the effective dimension, T is the number of rounds, and $\tilde{O}(\cdot)$ hides the log factor in $\tilde{O}(\cdot)$. This is a result analogous to the previous bounds, if we consider the simpler linear setting, where the effective dimension is just the feature dimension, [6, 14].

4.3 Discussion on Neural Thompson Sampling

We have mentioned already that Thompson Sampling has enjoyed a large literature and a wide use in real applications as it has been shown to be extremely effective for bandit problems both in practice,

[13], and in theory, [4]. It is particularly interesting to combine it with deep neural networks as neural networks have the advantage of being able to arbitrarily approximate any continuous function.

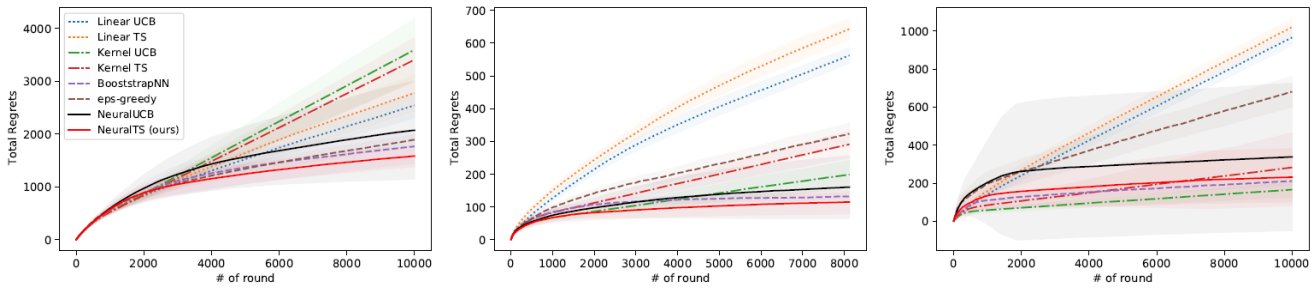
Before Neural Thompson Sampling, there wasn't any Thompson Sampling algorithm which used neural networks but also had theoretical strong guarantees. Apart from it having a near–optimal regret bound, it has empirically shown to have strong performance.

To check the performance of their method they provided an empirical evaluation of the algorithm on classification tasks against other benchmark algorithms. We now provide a detailed explanation on how to modify a classification problem into a contextual multi–armed bandit problem. This follows from [26] which builds a *context feature vector* for each arm. For every input feature $\boldsymbol{x} \in \mathbb{R}^d$ of a classification problem with k classes, we build the context feature vector with dimension kd as: $\boldsymbol{x}_1 = (\boldsymbol{x}; \mathbf{0}; \ldots; \mathbf{0}), \ldots, \boldsymbol{x}_k = (\mathbf{0}; \mathbf{0}; \ldots; \boldsymbol{x})$. Then, by following Algorithm 2, we generate an estimated reward for each arm, and pull the arm with the highest reward. If the algorithm selects the correct class, i.e. pulls the correct corresponding arm, then it receives a reward of 1, otherwise it receives a reward of 0. The performance measure chosen here is the cumulative regret over time horizon T, and the experiments are averaged over 8 times with reshuffled data.

The authors were interested in seeing how their algorithm performed against other, in terms of regret, and also checked if their algorithm would be consistent with theory and be more robust to delayed reward, to the best UCB method from the first experiment.

4.3.1 Performance Against Other Methods

The authors in [42] tested their algorithms against other commonly used algorithms. They concluded that it is among the best algorithms in 6 datasets and significantly better than every other algorithm in 2. The difference could be explained by the fact that neural networks have a strong representational power and linear methods could struggle, because of the nonlinearity of rewards in the data. The results on three of the datasets are shown in Figure 1. Note that for the MNIST and Shuttle dataset the time horizon is 10000, except the Mushroom dataset as it only contains 8124 data, [42].



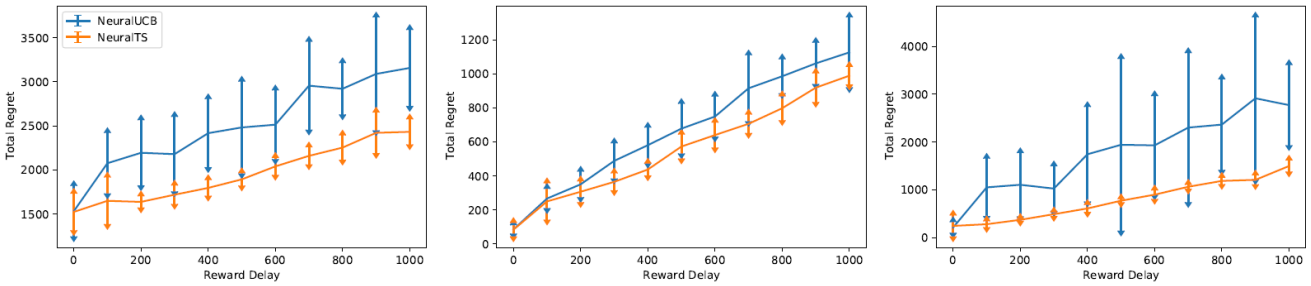
(a) MNIST (b) Mushroom (c) Shuttle

Figure 1: Comparing the Neural Thompson Sampling algorithm against other benchmark multi–armed bandit algorithms three datasets. The cumulative regret for each experiment is averaged over 8 runs, with the shaded areas indicating the standard errors,[42].

4.3.2 Robustness to Reward Delay

The authors in [42] also tested how robust were the the two most competitive algorithms, NeuralUCB and Neural Thompson Sampling to reward delay, which is common in practice, [13]. More specifically, the reward after pulling an arm is not given immediately, but they arrive in batches and then the algorithms update their respective models. The setting is the same as the first experiment, with the only difference

being that they vary the batch size from 0 to 1000. The results on the same three datasets are shown in Figure 3. Unsurprisingly, the Neural Thompson Sampling algorithm deteriorates more smoothly than Neural UCB, as the theory indicates that Thompson Sampling algorithms are more robust to reward delay than UCB ones. This difference could be explained due to the nature of the Thompson Sampling framework, as it has an inbuilt randomized exploration nature which allows more exploration between batches. This indicates that Neural Thompson Sampling could see wider use with better results in practice, although that would require more experimenting.



(a) MNIST (b) Mushroom (c) Shuttle

Figure 2: Comparing Neural Thompson Sampling and the best performing algorithm from the first experiment, Neural UCB on three datasets, when the rewards arrive with delay (0 to 1000). The cumulative regret for each experiment is averaged over 8 runs with the arrows indicating the standard errors, [42].

5 Numerical Experiments

Despite the fact that Neural Thompson Sampling has strong theoretical guarantees and empirical performance, it has a big disadvantage, which is a high computational cost. Step 9 of Algorithm 2 states that for every round, we have to run a gradient descent algorithm, and the theory states that to achieve this strong regret bound, one has to run it for a "large" amount of steps. In practice the authors in [42], used $n = 100$, but an interesting question arises; what would happen if we used $n = 1$ in the gradient descent for each round. Another challenge is to try and find a way to reduce the computational cost, without sacrificing performance.

5.1 Reducing the Number of Gradient Descent Steps

The $\tilde{O}(d\sqrt{T})$ regret bound of Neural TS is only achievable for a large amount of steps. This understandably brings a big computational cost even for "small" multi–armed bandit problems. The question we will try to empirically answer is if, and how, the performance is affected by using less steps, than $n = 100$, which is what the authors in the original paper used.

The dataset we are going to use in these subsections is the mushroom dataset. Following the authors in [42] we shuffle all datasets, normalize the features (so that their l_2 norm is unity), use one–hiden–layer neural network with 100 neurons, and 0.001 as the learning rate of the gradient descent. For all the experiment we also let $\lambda = 10^{-5}$.

We try the following number of steps $n \in \{1, 5, 10, 25, 50, 100\}$ and set the exploration variance parameter as $\nu \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. Each experiment is averaged over 5 times. The best case for each number of steps is shown in Figure 3.

Unsurprisingly, using $n = 100$ steps in gradient descent returns the best result, but it has a run time of 1183 seconds. However, using half of those steps, $n = 500$, returns almost identical results with less than half the run time, 549 seconds, but it has a higher standard error (45, compared to 25). Then, as the numbers of steps decrease to $n = 1$, although the run time is faster than previously, 43 seconds, the regret grows more rapidly. This could be explained due to the fact that using a smaller amount of gradient descent steps, the neural network fails to adapt its weights fast enough to approximate the reward function correctly. However, using $n = 50, 100$, it is clear that after around round 1500 the neural network can accurately predict the best reward which is why the regret ed.

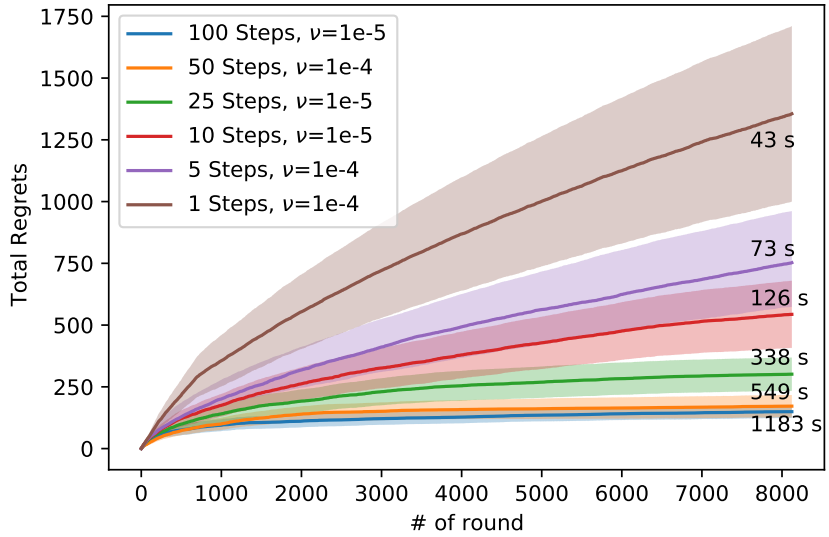


Figure 3: Comparing Neural Thompson Sampling algorithm for different amount of gradient descent steps $n = 1, 5, 10, 25, 50, 100$ in the **mushroom dataset**. This figure shows only the best try for each amount of steps, where we allowed the exploration variance parameter to vary as $\nu \in \{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$. The cumulative regret for each experiment is averaged over 5 runs with the shaded area indicating the standard errors. The run time for each case is also indicated in the plot.

Certainly, this is just an indication as we are only testing one example here. For a fuller study we should repeat every classification example from the authors' original paper to get a bigger picture and arrive in more concrete conclusions. However, our intuition tells us that we should expect similar results in most examples.

5.2 Adaptive Neural Thompson Sampling

In this subsection we are going to propose an extension to Neural Thompson Sampling, which we call Adaptive Neural Thompson Sampling, which we have empirically seen reduces the computational cost but does not sacrifice performance. The idea behind this algorithm is that ideally we would like the rate of the regret to be decreasing at each round, but usually after a certain number of rounds the regret function has a stable increase rate, which could mean one of two things. First, that the neural network can already approximate the reward function at a good level, therefore the reward arrives at a and increases marginally every round, thus training it more is not worth the computational cost. Another reason could be that the neural network fails to approximate the reward function better (perhaps due to the reward function being hard to approximate, or that the neural network is stuck at a local minima). Further, this means training the neural network using 100 steps in the gradient descent for each round might not

be as beneficial since the weights in the neural network will start to not shift by a large amount, as we can assume that the neural network has started to reach the model fitting and performance "capacity". Therefore, one should decrease the number of gradient descent steps.

More specifically we are going to use the backward difference formula for the cumulative regrets, but instead of looking only one round behind, we are looking 100 rounds behind, that is,

$$f'(x_j) \approx \frac{f(x_j) - f(x_{j-100})}{h},$$

where $h = 1$ in this case. We allow the algorithm to run for 20% of the time horizon T, to allow the neural network to train without the concern of having a "lucky" initialization. Then we evaluate the state of the algorithm every 100 steps and if for any point i we start to see the neural network consistently making the same number of mistakes, i.e. for any $i \in \{0.2 \times T, 0.2 \times T + 100, 0.2 \times T + 200, \dots, \}, |f'(x_i) - f'(x_{i-1})| \leq 3$, for 5 consecutive times, that is for $i, i+1, i+2, \dots, i+5$, this would indicate that the neural network has started to reach its fitting capacity and therefore we decrease the number of gradient descent steps to 5. The parameters we are mentioning here need to be formally chosen, as discussed later. Currently they are picked based on intuition and preliminary testing, but perhaps we can achieve even better computational saving by formally choosing them.

To test our algorithm we use the best parameters from the Neural TS original paper for both algorithms in the mushroom and covertype datasets, and we run the experiment 10 times and average the cumulative regret over the rounds. The results are shown in Figures 4, 5. It is clear that both algorithms have similar performance but our proposed algorithm is over 50% faster for the mushroom experiment, and almost 70% faster in the covertype dataset. These time differences can be explained due to the fact that it takes different amounts of time for the neural network to reach its capacity, as some reward functions could be approximated easier.

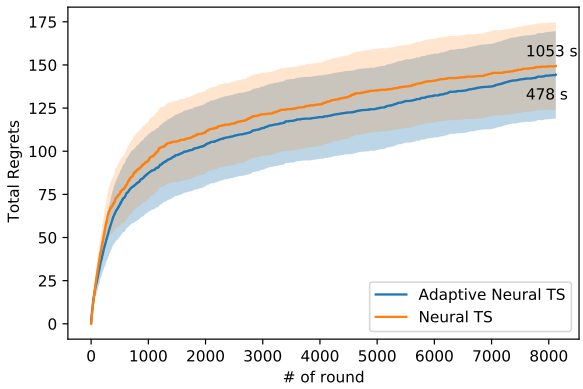


Figure 4: Comparing Neural Thompson Sampling and Adaptive Neural Thompson Sampling algorithms in the **mushroom dataset**. The cumulative regret for each experiment is averaged over 10 runs with the shaded area indicating the standard errors. The run time for each case is also indicated in the plot.

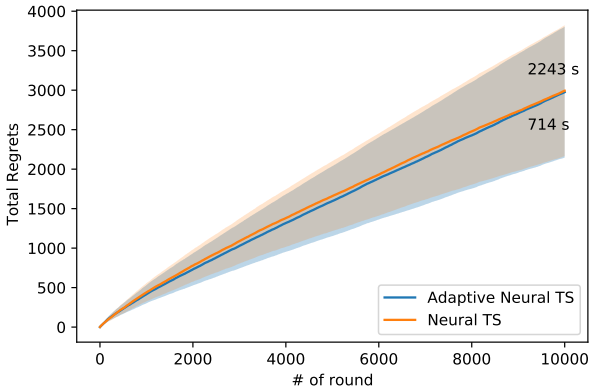


Figure 5: Comparing Neural Thompson Sampling and Adaptive Neural Thompson Sampling algorithms in the **covertype dataset**. The cumulative regret for each experiment is averaged over 10 runs with the shaded area indicating the standard errors. The run time for each case is also indicated in the plot.

6 Future Work

We now discuss future work that we could have done, provided we had more time and a few possible extensions to this problem.

6.1 Improve Batched Neural Thompson Sampling

We saw in Section 4.3.2 a batched version of Neural Thompson Sampling. Although this version of the algorithm is more robust to delay and deteriorates more smoothly than the best performing algorithm from Section 4.3.1, Neural UCB, it would be a good idea to find ways to further improve it. In our experiments we tried varying the Thompson Sampling variance according to batch size, but we did not find any significant improvement. By trying various inflation/deflation rates, one could possible achieve some sort of improvement, but this would require further experimentation. Additionally, one could try increasing the gradient descent steps at each round. This could make a difference, both computationally and performance wise, as the version of Neural Thompson Sampling with delay, is training the model less times and receives more data per iteration which suggests that it would require more gradient descent steps per iteration.

6.2 Formal Selection of Parameters

Although our method seems to be working empirically, there is still work needed to be done in order for it to be publishable, but in our opinion it is certainly a method worth considering.

First the heuristic $|f'(x_i) - f'(x_{i-1})| \leq 3$, for 5 consecutive times, needs to be explained formally as the parameters used were numbers that had preliminary good results and intuitively made sense. Possible ways to choose these parameters is, to use cross validation or grid search to pick the parameters that return the lowest regret, and we could search the literature for papers which considered a similar problem to ours.

6.3 Theoretical Guarantees

For our method to become formal there needs to be some mathematical proofs supporting it. Although in the experiments we tested it shows empirically to also have near–optimal regret like Neural TS, it certainly requires more experimentation, and it needs to be formally proven. This could potentially give more insights on our method, which in turn could lead to further speed up in computational time. Additionally, a common pattern we have noticed in this multi–armed bandit field is that although some methods (e.g. Thompson Sampling) were empirically shown to have good performance, they required formal theoretical guarantees for them to be widely accepted and be used in practice.

6.4 Using a Deeper Neural Network

Reward functions in practice can be extremely difficult to approximate. Using a one–hidden–layer neural network, although faster to train, could lead to inability of the Neural TS algorithm to correctly approximate the reward function. There exist many results in the literature stating that small approximation error can be achieved if the network size is sufficiently large, some references include [27, 29, 37].

An interesting set of question therefore arises: if and how is the performance of Neural TS or Adaptive Neural TS going to be improved by using a deeper neural network. However, we have to keep in mind, that to train a deeper neural network effectively, we might require more gradient descent steps and/or

more training time, which could make the computational cost prohibiting, especially for problems that require fast decision making.

It would also be worth exploring how different optimizing algorithms would make a difference in performance and speed of convergence. For instance, Adam or RMSProp are two popular algorithms that are widely used in practice and could be used in this problem.

7 Conclusion

In this report we presented the general multi–armed bandit problem as well as an extension which is quite common in practical applications, the contextual multi–armed bandit problem. We presented two of the most common contextual bandit cases, the linear and generalized and gave a brief introduction to one of the most widely used and effective algorithms tackling the contextual bandit problem and more generally many multi–armed bandit problems, Thompson Sampling. Then, the main focus of this report was an extension to Thompson Sampling, which used a fully–connected neural network to approximate the reward function, called Neural Thompson Sampling. We presented some of the results the authors in the original paper gave, and tried to empirically answer the question: how is the performance of the algorithm affected if we reduce the number of gradient descent steps. Finally, we proposed a novel extension to this algorithm, called Adaptive Neural Thompson Sampling, and we showed empirically that it has similar performance to Neural Thompson Sampling, while being faster than its adversary.

References

[1] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. "Improved algorithms for linear stochastic bandits". In: *Advances in neural information processing systems* 24 (2011).

[2] Naoki Abe. "Learning to optimally schedule internet banner advertisements". In: *Proc. of 16th Int. Conf. on Machine Learning*. 1999, pp. 12–21.

[3] Naoki Abe, Alan W Biermann, and Philip M Long. "Reinforcement learning with immediate rewards and linear hypotheses". In: *Algorithmica* 37.4 (2003), pp. 263–293.

[4] Shipra Agrawal and Navin Goyal. "Analysis of thompson sampling for the multi-armed bandit problem". In: *Conference on learning theory*. JMLR Workshop and Conference Proceedings. 2012, pp. 39–1.

[5] Shipra Agrawal and Navin Goyal. "Further optimal regret bounds for thompson sampling". In: *Artificial intelligence and statistics*. PMLR. 2013, pp. 99–107.

[6] Shipra Agrawal and Navin Goyal. "Thompson sampling for contextual bandits with linear payoffs". In: *International conference on machine learning*. PMLR. 2013, pp. 127–135.

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

[8] Peter Auer et al. "The nonstochastic multiarmed bandit problem". In: *SIAM journal on computing* 32.1 (2002), pp. 48–77.

[9] Andrew R Barron. "Neural net approximation". In: *Proc. 7th Yale workshop on adaptive and learning systems*. Vol. 1. 1992, pp. 69–72.

[10] Dirk Bergemann and Juuso Valimaki. "Bandit problems". In: (2006).

[11] Charles Blundell et al. "Weight uncertainty in neural network". In: *International conference on machine learning*. PMLR. 2015, pp. 1613–1622.

[12] Sébastien Bubeck and Nicolo Cesa-Bianchi. "Regret analysis of stochastic and nonstochastic multi-armed bandit problems". In: *arXiv preprint arXiv:1204.5721* (2012).

[13] Olivier Chapelle and Lihong Li. "An empirical evaluation of thompson sampling". In: *Advances in neural information processing systems* 24 (2011).

[14] Sayak Ray Chowdhury and Aditya Gopalan. "On kernelized multi-armed bandits". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 844–853.

[15] Wei Chu et al. "Contextual bandits with linear payoff functions". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 208–214.

[16] Sarah Filippi et al. "Parametric bandits: The generalized linear case". In: *Advances in Neural Information Processing Systems* 23 (2010).

[17] Quanquan Gu et al. "Batched neural bandits". In: *arXiv preprint arXiv:2102.13028* (2021).

[18] Kurt Hornik. "Some new results on neural network approximation". In: *Neural networks* 6.8 (1993), pp. 1069–1072.

[19] Arthur Jacot, Franck Gabriel, and Clément Hongler. "Neural tangent kernel: Convergence and generalization in neural networks". In: *Advances in neural information processing systems* 31 (2018).

[20] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. "Thompson sampling: An asymptotically optimal finite-time analysis". In: *International conference on algorithmic learning theory*. Springer. 2012, pp. 199–213.

[21] Volodymyr Kuleshov and Doina Precup. "Algorithms for multi-armed bandit problems". In: *arXiv preprint arXiv:1402.6028* (2014).

[22] Branislav Kveton et al. "Randomized exploration in generalized linear bandits". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2020, pp. 2066–2076.

[23] Tze Leung Lai, Herbert Robbins, et al. "Asymptotically efficient adaptive allocation rules". In: *Advances in applied mathematics* 6.1 (1985), pp. 4–22.

[24] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[25] Hoang Le, Cameron Voloshin, and Yisong Yue. "Batch policy learning under constraints". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3703–3712.

[26] Lihong Li et al. "A contextual-bandit approach to personalized news article recommendation". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 661–670.

[27] Shiyu Liang and Rayadurgam Srikant. "Why deep neural networks for function approximation?" In: *arXiv preprint arXiv:1610.04161* (2016).

[28] Xiuyuan Lu and Benjamin Van Roy. "Ensemble sampling". In: *Advances in neural information processing systems* 30 (2017).

[29] Guido F Montufar et al. "On the number of linear regions of deep neural networks". In: *Advances in neural information processing systems* 27 (2014).

[30] Sergey Muravyov and Andrey Filchenkov. “A Cloud-based Network of 3D Objects for Robust Grasp Planning”. In: *2020 International Conference on Control, Robotics and Intelligent System*. 2020, pp. 99–105.

[31] Min-hwan Oh and Garud Iyengar. “Thompson sampling for multinomial logit contextual bandits”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[32] Vianney Perchet et al. “Batched bandit problems”. In: *The Annals of Statistics* 44.2 (2016), pp. 660–681.

[33] Ciara Pike-Burke. *Sequential decision problems in online education*. Lancaster University (United Kingdom), 2019.

[34] Carlos Riquelme, George Tucker, and Jasper Snoek. “Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling”. In: *arXiv preprint arXiv:1802.09127* (2018).

[35] Daniel J Russo et al. “A tutorial on thompson sampling”. In: *Foundations and Trends® in Machine Learning* 11.1 (2018), pp. 1–96.

[36] Eric M Schwartz, Eric T Bradlow, and Peter S Fader. “Customer acquisition via display advertising using multi-armed bandit experiments”. In: *Marketing Science* 36.4 (2017), pp. 500–522.

[37] Matus Telgarsky. “Benefits of depth in neural networks”. In: *Conference on learning theory*. PMLR. 2016, pp. 1517–1539.

[38] Ambuj Tewari and Susan A Murphy. “From ads to interventions: Contextual bandits in mobile health”. In: *Mobile Health*. Springer, 2017, pp. 495–517.

[39] William R Thompson. “On the likelihood that one unknown probability exceeds another in view of the evidence of two samples”. In: *Biometrika* 25.3-4 (1933), pp. 285–294.

[40] Michal Valko et al. “Finite-time analysis of kernelised contextual bandits”. In: *arXiv preprint arXiv:1309.6869* (2013).

[41] Michael Woodroofe. “A one-armed bandit problem with a concomitant variable”. In: *Journal of the American Statistical Association* 74.368 (1979), pp. 799–806.

[42] Weitong Zhang et al. “Neural thompson sampling”. In: *arXiv preprint arXiv:2010.00827* (2020).

[43] Dongruo Zhou, Lihong Li, and Quanquan Gu. “Neural contextual bandits with ucb-based exploration”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 11492–11502.