Thompson Sampling Based on Deep Representation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Thompson Sampling (TS) is an effective way to deal with the exploration–exploitation dilemma in the (contextual) multi-armed bandit problem. In real-world applications, when the correlation between the context and the reward is sophisticated, neural networks are often utilized owing to their representation capacity. In this paper, we study the problem of combining neural networks and TS. We propose to maintain a posterior distribution over the reward mean based on the prediction and the deep representation of the neural network for the context. Our proposed algorithm, TSDR (Thompson Sampling based on Deep Representation), introduces no additional computational overhead for training, but sampling from a univariate Gaussian distribution during serving for every context. Theoretically, we prove that TSDR achieves a $\mathcal{O}(\sqrt{T})$ regret bound, which matches the state-of-the-art regret bound of neural network-based TS but with a much lower computational complexity for each iteration. Experimental results on public datasets for traditional classification and recommendation problems validate the effectiveness and efficiency. Furthermore, it is of great practical value that TSDR has been deployed in the advertising platform (a recommender system) of a massive online retailer, archiving improvement of revenue and clicks by 5% and 3% in online A/B testing respectively.

1 Introduction

The *multi-armed bandit* problem characterizes the inherent exploration–exploitation trade-off in some sequential decision problems — balancing between exploiting acquired knowledge for immediate performance and exploring more knowledge for future performance [1–4]. A widely used version is the *multi-armed contextual bandit* problem. At each round, given contexts for all arms along with historical contexts and corresponding rewards, the agent has to decide which arm to pull in order to maximize cumulative rewards. Take the recommender system as an example, which needs to select an item from a candidate item set, and presents it to the user. The candidate items are the arms, the contexts involve user and item features, and the rewards are derived from users' feedback.

Thompson Sampling (TS, also known as *posterior sampling* or *probability matching*), dating to Thompson [5], addresses the exploration–exploitation trade-off in a computationally efficient manner and shows great empirical performance in practice [6–9]. The basic idea is to construct a prior distribution over the underlying parameters of the reward distribution, and at each round, TS plays an arm according to the posterior probability of being the optimal one among all arms. Simple and computationally efficient Bayesian inference for the posterior distribution (updating of posterior parameters) often relies on conjugacy properties [10].

In real-world applications, the correlation between contexts and rewards is usually sophisticated; it is a natural idea to resort to deep neural networks to model this correlation owing to their representation capacity. For example, deep learning based CTR (click-through rate) models in recommender systems have achieved great success in recent years due to their capacity to learn the correlation between user

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

behaviors and item characteristics [11–13]. However, tackling the exploration–exploitation trade-off via TS in these situations is much more complicated. The key challenge lies in the fact that TS has to maintain a posterior distribution over the parameters of the reward distribution, whereas the exact Bayesian inference is generally computationally intractable in the neural network case [14]. Therefore, a series of work has been dedicated to reasonably efficient and accurate methods which could approximately sample from posterior distributions such as variational methods and stochastic mini-batch Markov Chain Monte Carlo [15–18].

In contrast to the methods mentioned above, in this paper, we propose a "simple" TS algorithm addressing the exploration–exploitation trade-off with neural networks. Given a context for an arm, we propose to utilize the network's prediction and deep representation (the output before the last layer) to model the conditional posterior distribution over the reward mean. Specifically, it is assumed that a scalar parameter (could also be the reward mean itself) of the reward mean has a probability distribution to be Gaussian, of which the mean is the network's prediction and the standard deviation is proportional to the norm of the deep representation. We call this simple algorithm as *TSDR*, the abbreviation for *Thompson Sampling based on Deep Representation*.

It is noteworthy that TSDR does not explicitly make Bayesian inference; the posterior parameter is implicitly updated along with the neural network updating. In other words, TSDR does not introduce any additional computational overhead for the whole training process, while only one sample from a univariate Gaussian is required when serving for each context.

We illustrate the effectiveness of our proposed TSDR from two perspectives. From the theoretical side, under classical conditions, we prove that TSDR enjoys a regret bound of $\mathcal{O}(\sqrt{T})$ where T is the total round number, based on the neural tangent kernel theory [18, 19], which matches the state-of-the-art theoretical regret result. From the empirical side, we conduct experiments on various public datasets ranging from traditional classification datasets to recommendation datasets, which are used to simulate the online decision environment. Our experimental results also demonstrate the advantage of the proposed TSDR algorithm.

Notably, attributed to its simplicity and efficiency, we successfully deploy TSDR for industrial production in an advertising platform of a massive online retailer. In a carefully designed online A/B testing system — data served by one model is only used to train itself but not used to train other models. TSDR achieves an improvement of revenue and clicks by 5% and 3% respectively.

The remainder of this paper is organized as follows. In Section 2, we formally introduce the preliminaries on the contextual bandit problem and Thompson Sampling. Our proposed algorithm TSDR and its intuitions are presented in Section 3. Theoretical analysis on regret bounds is described in Section 4. We then demonstrate the experimental results on both public datasets and online A/B testing in Section 5. Section 6 discusses related literature, and Section 7 concludes this paper finally.

2 Preliminaries

We consider the multi-armed contextual bandit problem. At round t, context vectors $\{\boldsymbol{x}_{t,k}\}_{k=1}^{K}$ for the K arms are observed with $\boldsymbol{x}_{t,k} \in \mathbb{X}$ for all $t \in [T]$ and $k \in [K]$. After selecting one of the arms $a_t \in [K]$, the corresponding reward $r_{t,a_t} \in \mathbb{R}$, a random variable, is obtained. The goal is to minimize the (cumulative pseudo) regret:

$$\mathrm{Regret}_T = \mathbb{E}\left[\sum_{t=1}^{T}(r_{t,a_t^*} - r_{t,a_t})\right], \tag{1}$$

where a_t^* is the optimal arm at round t that has the maximum expected reward, i.e.,

$$a_t^* = \arg\max_{a \in [K]} \mathbb{E}\left[r_{t,a}\right], \tag{2}$$

and the expectation in (1) is taken over randomness of r_{t,a_t} and r_{t,a_t^*} for all $t \in [T]$.

Thompson Sampling (TS) provides a natural way to select arms for the multi-armed contextual bandit problem. It is assumed that the reward $r \in \mathbb{R}$ is randomly generated according to a conditional probability measure conditioning on the context $\boldsymbol{x} \in \mathbb{X}$, i.e., $r \sim q_\theta(\cdot; \boldsymbol{x})$, where θ is an unknown parameter for reward's distribution q. As we will see later, we only need to compute $\mathbb{E}_{q_\theta}[r|\boldsymbol{x}]$ for $\boldsymbol{x} \in \mathbb{X}$ in some cases. TS maintains a distribution on θ, denoted as $p_t(\cdot)$ for every $t \in [T]$. At each

Algorithm 1: Thompson Sampling for multi-armed contextual bandit

Input: prior distribution p_1 over unknown θ, conditional probability measure q for reward

for $t = 1, \ldots, T$ **do**
  Receive context vectors $\{\boldsymbol{x}_{t,k}\}_{k=1}^{K}$;
  Sample $\hat{\theta}_t \sim p_t$;
  $a_t \leftarrow \arg\max_{a \in [K]} \mathbb{E}_{q_{\hat{\theta}_t}}[r|\boldsymbol{x}_{t,a}]$;
  Apply a_t and observe r_{t,a_t};
  Update posterior distribution via Bayesian inference $p_{t+1} \leftarrow \mathbb{P}_{p_t,q}(\theta \in \cdot \mid \boldsymbol{x}_{t,a_t}, r_{t,a_t})$;
end

round t, TS samples $\hat{\theta}_t$ from $p_t(\cdot)$, and then a_t is selected as the optimal arm that has the maximum expected reward according to $q_{\hat{\theta}_t}(\cdot; \boldsymbol{x}_{t,k})$, i.e.,

$$a_t = \arg\max_{a \in [K]} \mathbb{E}_{q_{\hat{\theta}_t}}[r|\boldsymbol{x}_{t,a}]\,. \tag{3}$$

After observing the returned reward r_{t,a_t}, updating $p_t(\cdot)$ with the labeled instance $(\boldsymbol{x}_{t,a_t}, r_{t,a_t})$ by Bayesian inference leads to $p_{t+1}(\cdot)$, often relying on conjugacy properties. It is noteworthy that in practice, exact Bayesian inference is not always tractable / necessary, whereas approximate sampling from the posterior distribution could be used instead. Algorithm 1 presents Thompson Sampling for the multi-armed contextual bandit problem.

3 Proposed Algorithm

At each round t, let $f(\cdot; \boldsymbol{w}_t) : \mathbb{X} \to \mathbb{R}$ be a neural network parameterized by $\boldsymbol{w}_t$, where $\boldsymbol{w}_t$ is the collection vector of all parameters for the neural network ordered by the layers, and its last layer is a linear function about the output of the penultimate layer. For every $\boldsymbol{x} \in \mathbb{X}$, we have

$$f(\boldsymbol{x}; \boldsymbol{w}_t) = \boldsymbol{w}_{t,\text{last}}^{\top} \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_{t,\text{repr}}), \tag{4}$$

where $\boldsymbol{w}_{t,\text{last}} \in \mathbb{R}^d$ denotes the parameters (weights) for the linear function of the last layer and $\boldsymbol{f}_{\text{repr}}(\cdot; \boldsymbol{w}_{t,\text{repr}}) : \mathbb{X} \to \mathbb{R}^d$ denotes the deep representation (feature mapping) function. $\boldsymbol{w}_{t,\text{repr}}$ consists of all weights in $\boldsymbol{w}_t$ except $\boldsymbol{w}_{t,\text{last}}$, i.e., $\boldsymbol{w}_t = [\boldsymbol{w}_{t,\text{repr}}; \boldsymbol{w}_{t,\text{last}}]$.

We consider a special case where θ could be decomposed as parameters corresponding to every $\boldsymbol{x} \in \mathbb{X}$, i.e., $\theta = \{\theta_{\boldsymbol{x}}\}_{\boldsymbol{x} \in \mathbb{X}}$, and $\mathbb{E}_{q_\theta}[r|\boldsymbol{x}]$ only depends on $\theta_{\boldsymbol{x}}$. To ease our notation, we denote

$$r(\theta_{\boldsymbol{x}}) = \mathbb{E}_{q_\theta}[r|\boldsymbol{x}], \tag{5}$$

e.g., $r(\theta_{\boldsymbol{x}}) = \theta_{\boldsymbol{x}}$ and $r(\theta_{\boldsymbol{x}}) = 1/(1 + \exp(-\theta_{\boldsymbol{x}}))$ are two simple cases. Therefore, we only need to maintain a distribution $p_{t,\boldsymbol{x}}$ over $\theta_{\boldsymbol{x}}$ for every $\boldsymbol{x} \in \mathbb{X}$ at each round.

We are ready to propose our algorithm framework TSDR, which is the abbreviation of Thompson Sampling based on Deep Representation. We show TSDR in Algorithm 2. TSDR models $p_{t,\boldsymbol{x}}$ as a Gaussian distribution of which the mean is $f(\boldsymbol{x}; \boldsymbol{w}_t)$, and the variance is $\nu \|\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_t)\|^2_{\boldsymbol{M}_t^{-1}}$ where $\boldsymbol{M}_t$ is a positive definite matrix which we will discuss later in detail,

$$\|\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_t)\|^2_{\boldsymbol{M}_t^{-1}} = \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_t)^{\top} \boldsymbol{M}_t^{-1} \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_t), \tag{6}$$

and $\nu \in \mathbb{R}^+$ is a hyper-parameter controlling the strength of exploration. At round t, TSDR samples

$$\hat{\theta}_{t,\boldsymbol{x}_{t,k}} \sim \mathcal{N}(f(\boldsymbol{x}_{t,k}; \boldsymbol{w}_t)), \nu \|\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,\text{repr}})\|^2_{\boldsymbol{M}_t^{-1}}), \tag{7}$$

for all $a \in [K]$ and selects the action a_t as

$$a_t = \arg\max_{a \in [K]} r(\hat{\theta}_{t,\boldsymbol{x}_{t,a}}). \tag{8}$$

After observing the returned reward r_{t,a_t}, we update the neural network parameter $\boldsymbol{w}_t$ with the collected labeled data so far, $\{(\boldsymbol{x}_{i,a_i}, r_{i,a_i})\}_{i=1}^{t}$.

Algorithm 2: TSDR (Thompson Sampling based on deep representation) framework

Input: Initialization parameter of neural network $\boldsymbol{w}_1$, hyper-parameter for exploration strength ν, positive definite matrix $\boldsymbol{M}_1$

for $t = 1, \ldots, T$ **do**
  Receive context vectors $\{\boldsymbol{x}_{t,k}\}_{k=1}^{K}$;
  for $k = 1, \ldots, K$ **do**
    Sample $\hat{\theta}_{t,\boldsymbol{x}_{t,k}} \sim \mathcal{N}(f(\boldsymbol{x}_{t,k}; \boldsymbol{w}_t), \nu \| \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,\text{repr}}) \|^2_{\boldsymbol{M}_t^{-1}})$;
  end
  $a_t \leftarrow \arg\max_{a \in [K]} r(\hat{\theta}_{t,\boldsymbol{x}_{t,a}})$;
  Apply a_t and observe r_{t,a_t};
  Update $\boldsymbol{w}_{t+1}$ based on $\boldsymbol{w}_t$ and $\{(\boldsymbol{x}_{i,a_i}, r_{i,a_i})\}_{i=1}^{t}$;
  Update $\boldsymbol{M}_{t+1}$;
end

Note that TSDR does not explicitly restrict the way updating the neural network. In practice, we could update $\boldsymbol{w}_t$ at arbitrary intervals rather than at each round, and use only the most recent data rather than all observed instances as we will show in the experiment section.

One possible simple example of TSDR could be as follows: $\boldsymbol{w}_1$ is randomly initialized; $\boldsymbol{w}_{t+1}$ is derived by retraining the neural network with $\{(\boldsymbol{x}_{i,a_i}, r_{i,a_i})\}_{i=1}^{t}$; $\boldsymbol{M}_t = t\boldsymbol{I}$ where $\boldsymbol{I}$ is the identity matrix. This example will be implemented in the experiment section.

We would like to discuss the twofold intuition behind our framework. The first part of the intuition is about the possible contribution for a context to update the neural network. Let $(\boldsymbol{x}, r)$ be a labeled instance, and $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is the loss function used to train the neural network. Then, we have

$$\nabla_{\boldsymbol{w}_{\text{last}}} \ell(f(\boldsymbol{x}; \boldsymbol{w}), r) = \frac{\partial \ell(f(\boldsymbol{x}; \boldsymbol{w}), r)}{\partial f(\boldsymbol{x}; \boldsymbol{w})} \nabla_{\boldsymbol{w}_{\text{last}}} f(\boldsymbol{x}; \boldsymbol{w}) = \frac{\partial \ell(f(\boldsymbol{x}; \boldsymbol{w}), r)}{\partial f(\boldsymbol{x}; \boldsymbol{w})} f_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_{\text{repr}}). \tag{9}$$

Consider an extreme case with $f_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_{\text{repr}}) = \boldsymbol{0}$, where incorporating $\boldsymbol{x}$ as a training point will not lead to updating the parameters of the last layer whatever the ground-truth reward r is. It is our motivation to give a small variance for small $f_{\text{repr}}(\boldsymbol{x}; \boldsymbol{w}_{\text{repr}})$ since it hardly leads to informative updates for the neural network.

The second part of our intuition is about uncertainty — the deep representation characterizes uncertainty of the network's prediction to a certain extent. Consider a toy case. The training set $\mathbb{S}$ contains two disjoint subsets $\mathbb{S}_1$ and $\mathbb{S}_2$: $\mathbb{S}_1 \bigcup \mathbb{S}_2 = \mathbb{S}$, $\mathbb{S}_1 \bigcap \mathbb{S}_2 = \emptyset$; $|\mathbb{S}_1| = n_1$, and $\forall (\boldsymbol{x}, y) \in \mathbb{S}_1$, $\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}) = \boldsymbol{f}_{\text{repr},1}$, $y = y_1$; $|\mathbb{S}_2| = n_2$, and $\forall (\boldsymbol{x}, y) \in \mathbb{S}_2$, $\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}) = \boldsymbol{f}_{\text{repr},2}$, $y = y_2$. We further let $f_1 = \boldsymbol{w}_{\text{last}}^\top \boldsymbol{f}_{\text{repr},1}$ and $f_2 = \boldsymbol{w}_{\text{last}}^\top \boldsymbol{f}_{\text{repr},2}$. The neural network is optimized such that it satisfies the first-order optimality condition, i.e.,

$$n_1 \frac{\partial \ell(f_1, y_1)}{\partial f_1} \boldsymbol{f}_{\text{repr},1} + n_2 \frac{\partial \ell(f_2, y_2)}{\partial f_2} \boldsymbol{f}_{\text{repr},2} = \boldsymbol{0}. \tag{10}$$

Then, we have

$$n_1 \left| \frac{\partial \ell(f_1, y_1)}{\partial f_1} \right| \| \boldsymbol{f}_{\text{repr},1} \| = n_2 \left| \frac{\partial \ell(f_2, y_2)}{\partial f_2} \right| \| \boldsymbol{f}_{\text{repr},2} \|. \tag{11}$$

That is, in this toy case, if $\left| \frac{\partial \ell(f_1, y_1)}{\partial f_1} \right| \approx \left| \frac{\partial \ell(f_2, y_2)}{\partial f_2} \right|$, the deep representation which shows up frequently (intuitively with small uncertainty) will have a small norm.

4 Theoretical Analysis

We next provide the regret bound for TSDR. For theoretical analysis, we make several constraints about the architecture of the neural network: f is a L-layered fully-connected neural network with ReLU as the activation function, of which the width is m, and the conditional reward mean is characterized as $r(\theta_{\boldsymbol{x}}) = \theta_{\boldsymbol{x}}$. We show this instance of TSDR, named Theoretical TSDR (TSDR-T) in Algorithm 3.

Algorithm 3: TSDR-T (Theoretical TSDR)

Input: Initialization parameter of neural network $\boldsymbol{w}_1$, hyper-parameter for exploration strength ν, network width m, regularization parameter $\lambda > 0$, positive definite matrix $\boldsymbol{M}_1 = \lambda \boldsymbol{I}$

for $t = 1, \ldots, T$ **do**
  Receive context vectors $\{\boldsymbol{x}_{t,k}\}_{k=1}^{K}$;
  for $k = 1, \ldots, K$ **do**
    Sample $\hat{\theta}_{t,\boldsymbol{x}_{t,k}} \sim \mathcal{N}(f(\boldsymbol{x}_{t,k}; \boldsymbol{w}_t)), \nu \|\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,\text{repr}})\|^2_{\boldsymbol{M}_t^{-1}})$;
  end
  $a_t \leftarrow \arg\max_{a \in [K]} \hat{\theta}_{t,\boldsymbol{x}_{t,a}}$;
  Apply a_t and observe r_{t,a_t};
  $\boldsymbol{w}_{t+1} \leftarrow \arg\min_{\boldsymbol{w}} \sum_{i=1}^{t} (f(\boldsymbol{x}_{t,a_t}) - r_{t,a_t})^2/2 + m\lambda \|\boldsymbol{w} - \boldsymbol{w}_1\|^2/2$;
  $\boldsymbol{M}_{t+1} \leftarrow \boldsymbol{M}_t + \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,a_t}; \boldsymbol{w}_{t+1}) \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,a_t}; \boldsymbol{w}_{t+1})^\top$;
end

We make a mild assumption that the difference between the deep representation and the gradient for all parameters of the neural network is bounded with respect to the width of the neural network.

Assumption 1 *For all $t \in [T]$ and $k \in [K]$, we have*

$$\|\tilde{\boldsymbol{f}}_{repr}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,repr}) - \nabla_{\boldsymbol{w}} f(\boldsymbol{x}_{t,k}; \boldsymbol{w}_t)\| = o(\sqrt{m}), \tag{12}$$

where $\tilde{\boldsymbol{f}}_{repr}$ is expanded from $\boldsymbol{f}_{repr}$ by aligning its elements with $\boldsymbol{w}_{t,last}$ in $\boldsymbol{w}_t$ and adding 0 in other positions, i.e., $\tilde{\boldsymbol{f}}_{repr}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,repr}) = [\boldsymbol{0}; \boldsymbol{f}_{repr}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,repr})]$.

We are ready to present the main theorem.

Theorem 1 *Under classical conditions and Assumption 1, if the network width m satisfies $m \geq \text{poly}(\lambda, T, K, L, \log(1/\delta))$, then with probability $1 - \delta$, TSDR-T enjoys*

$$\text{Regret}_T = \mathbb{E}\left[\sum_{t=1}^{T} (r_{t,a_t^*} - r_{t,a_t})\right] = \mathcal{O}(\sqrt{T}). \tag{13}$$

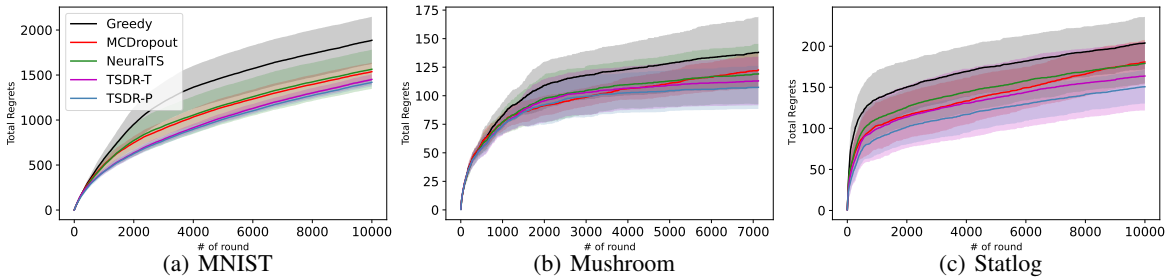
Due to space limitation, we postpone the details of conditions and proof in Appendix A. Theorem 1 shows that if the neural network has sufficiently large width, then TSDR-T has the $\mathcal{O}(\sqrt{T})$ regret bound. This result matches the state-of-the-art theoretical analysis of the recently proposed algorithm NeuralTS [18].

Note that $\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,\text{repr}})$ is the gradient with respect to the last layer, i.e., $\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,\text{repr}}) = \nabla_{\boldsymbol{w}_{t,\text{last}}} f(\boldsymbol{x}_{t,k}; \boldsymbol{w}_t)$. From this perspective, TSDR-T has a natural extension, Generalized Theoretical TSDR (TSDR-GT) as shown in Appendix A, by introducing gradients of other layers. Specifically, TSDR-GT replaces $\boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,\text{repr}})$ in Algorithm 3 with $\nabla_{\boldsymbol{w}_{t,\text{subset}}} f(\boldsymbol{x}_{t,k}; \boldsymbol{w}_{t,k})$ where $\boldsymbol{w}_{t,\text{subset}}$ is a subset of $\boldsymbol{w}_t$. Then, under an assumption similar with Assumption 1, TSDR-GT also achieves the $\mathcal{O}(\sqrt{T})$ regret bound. Interestingly, NeuralTS [18] could also be seen as a special case of TSDR-GT when $\boldsymbol{w}_{t,\text{subset}} = \boldsymbol{w}_t$.

A significant advantage of TSDR over TSDR-GT (including NeuralTS) is that TSDR does not need the backward propagation procedure to compute the gradient during serving; in other words, TSDR has better per-iteration computational complexity. We will show the empirical advantage of TSDR in Section 5.

5 Experiments

In this section, we present the results in three kinds of experiment settings: (i) a simulated contextual bandit problem transformed from traditional classification datasets, (ii) an unbiased simulation of a recommender system, and (iii) an online A/B test on an advertising platform. The first two kinds of experiments are conducted on NVIDIA GeForce RTX 2080 Ti.



(a) MNIST (b) Mushroom (c) Statlog

Figure 1: Results on classification datasets

Several typical neural network-based baselines are included in our experiments. Concretely, we compare our proposed TSDR with the Greedy algorithm and two representative TS-based algorithms: MCDropout (Monte-Carlo Dropout) [20] and NeuralTS (Neural Thompson Sampling) [18]. The Greedy algorithm applies a strategy with pure exploitation, which directly takes the network's prediction as the expectation of the parameter for the reward's distribution instead of sampling as TS. MCDropout takes the dropout method to approximate the posterior distribution, which is widely used in practical applications due to its simplicity and good performance. NeuralTS, which is theoretically guaranteed, uses the neural tangent features of the neural network to guide exploration. More details of these baseline algorithms are in Appendix B. Note that the comparison with NeuralTS is mainly to verify our theoretical conclusion, and the purpose of involving Greedy and MCDropout is to show the efficiency and effectiveness of our proposed TSDR.

5.1 Classification Datasets

We evaluate our algorithms on a range of multi-armed contextual bandit problems, which are transformed from traditional classification datasets. The simulation from classification datasets is a classic experimental setting for theoretical algorithms, since the regret could be exactly computed [15, 18, 21]. Therefore, we choose this idealized experimental setting to verify our theoretical results.

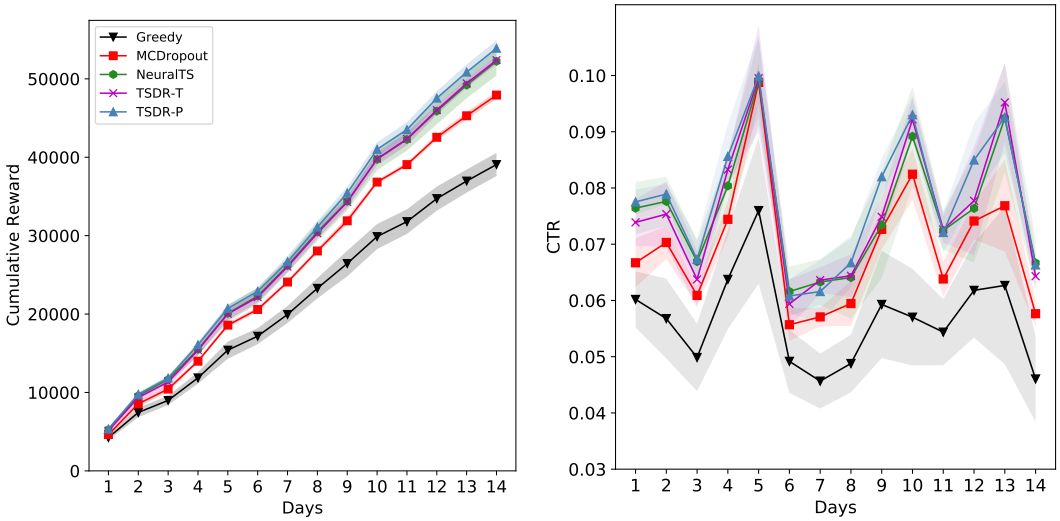
In particular, we experiment on the MNIST dataset [22] and datasets from the UCI Machine Learning Repository [23], including Mushroom, Statlog, Covertype, Adult, and Magic Telescope. Following the transformation setting in [15], we adapts the disjoint model [6] to build contextual features for each arm. For each input feature $\boldsymbol{x} \in \mathbb{R}^d$ of a k-class classification problem, the corresponding k contextual vectors are $\boldsymbol{x}_1 = (\boldsymbol{x}; \mathbf{0}; \ldots; \mathbf{0}), \boldsymbol{x}_2 = (\mathbf{0}; \boldsymbol{x}; \ldots; \mathbf{0}), \ldots, \boldsymbol{x}_k = (\mathbf{0}; \mathbf{0}; \ldots; \boldsymbol{x}) \in \mathbb{R}^{kd}$. If the arm with the correct class is selected, a reward 1 will be received, and 0 otherwise. The cumulative regret is used as the metric for comparison.

In order to verify the regret bound of our proposed TSDR-T, we compare it with NeuralTS, which enjoys similar theoretical results as discussed in Section 4. Considering the difficulty of matrix inversion, we make the same approximation for TSDR-T as NeuralTS, that is, replace $\boldsymbol{M}^{-1}$ with the inverse of the diagonal elements of $\boldsymbol{M}$ when computing the variance. Other components of TSDR-T are kept the same as those described in algorithm 3.

In addition, we realize another instance of TSDR with $\boldsymbol{M}_t = t\boldsymbol{I}$, named TSDR-P (Practical TSDR), whose computational complexity is the almost same with the Greedy algorithm, since it does need to maintain a complicated $\boldsymbol{M}$.

All algorithms share the same neural network structure which is a two-layer full connected neural network with ReLU as the activation function. We set the network width $m = 100$, and use the SGD optimizer to update the network weights for 100 iterations at each round with a learning rate 0.01. For all algorithms involved, we randomly select 1,000 samples from the dataset to tune hyperparameters and another 10,000 samples to test these algorithms, except for Mushroom which contains only 8,124 points. All algorithms are evaluated with 8 independent repeated runs, reshuffling the test data. We also repeat 8 runs when tuning hyperparameters, and choose the one with the optimal average result.

Due to space limit, we defer the results on Covertype, Adult and Magic Telescope to Appendix B. Experimental results on MNIST, Mushroom and Statlog are shown in Figure 1, from which we can



(a) Cumulative Reward (b) Daily CTRs

Figure 2: Results on Yahoo!R6B dataset

find that the performance of these algorithms are consistent across all datasets. First, TSDR-T is slightly better than NeuralTS and the growth rates of the cumulative regret are similar, which supports our theoretical result that TSDR-T enjoys a $\mathcal{O}(\sqrt{T})$ regret bound. Second, TSDR-P and TSDR-T perform similarly, which questions whether maintaining the matrix $\boldsymbol{M}$ containing historical context information is necessary. One possible answer is that the information has already been learned in the deep representation during neural network training. Third, our proposed algorithms, TSDR-T and TSDR-P, outperform all the other algorithms across these datasets.

5.2 Recommendation Datasets

In this section, we compare the above algorithms in a more practical manner. We simulate an online recommender system based on an unbiased dataset, Yahoo!front page news dataset (Yahoo!R6B) dataset [24]. It is motivated by the fact that from the above experimental results on classification datasets, we also find that the greedy algorithm is sometimes empirically competitive against NeuralTS, which reflects the limitations of only using classification datasets to evaluate Thompson Sampling-based algorithms.

Yahoo!R6B contains around 28 million lines of log data in chronological order, within 15 days of October 2011. Each line in Yahoo!R6B dataset contains four types of information: a user feature, a displayed item, a binary label for the user's feedback (clicked or not clicked) and a candidate set of items, where the displayed item was randomly sampled from the candidate set. The random impression makes the dataset unbiased and enables us to evaluate Thompson Sampling-based algorithms in the simulated online recommender system [24, 25].

In our simulation, we initialize the neural network with the first 80,000 log entries and conduct the contextual bandit experiment with the remaining data. For every subsequent log entry, we use the comparing algorithm to select an item from its corresponding candidate set. If the item selected happens to be the one recorded as randomly displayed in the log entry, we take this impression as "real", and collect the corresponding item, the user feature, and the user's feedback as a new sample in the training set. Meanwhile, we remove the earliest sample in the training set to keep the size of the training set to be 80,000. If the user's feedback is positive, the cumulative reward will be added by 1, otherwise added by 0. If the selected item is not the recorded one, the log entry is omitted. After walking through every 80,000 log entries, the neural network is retrained with the updated training set, consistent with real scenes where we only use data in a certain previous period of time to update neural networks. The simulation process is summarized in Algorithm 4.

Algorithm 4: Online Recommender System Simulator

Input: randomized impression dataset $\mathcal{D}$, initialization dataset $\mathcal{D}_{\text{init}}$, selecting strategy $s : \mathbb{X} \times \mathbb{A} \to \mathbb{R}$ and corresponding neural network f, size of partition $N > 0$

$\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{init}}$;
Train f with $\mathcal{D}_{\text{train}}$;
$r_{\text{all}} \leftarrow 0$ // $r_{\texttt{all}}$:`cumulative reward (number of clicks)`;
for *each N-size log entries partition $\mathcal{D}_{part} \subseteq \mathcal{D}$* **do**
    Initialize $\mathcal{D}_{\text{simulator}} \leftarrow \emptyset$;
    `// x:user feature; a:displayed item; r:binary label; A:candidate set`
    for $(x, a, r, A) \in \mathcal{D}_{part}$ **do**
        if $\arg\max_{a' \in A} s(x, a') == a$ **then**
            $\mathcal{D}_{\text{simulator}} \leftarrow \mathcal{D}_{\text{simulator}} \cup \{(x, a, r)\}$;
            $r_{\text{all}} \leftarrow r_{\text{all}} + r$;
        end
    end
    Remove the oldest $|\mathcal{D}_{\text{simulator}}|$ samples from $\mathcal{D}_{\text{train}}$;
    $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{simulator}}$;
    Train f with $\mathcal{D}_{\text{train}}$;
end

Output: r_{all}

In summary, the differences between the recommendation setting and the classification setting are as follows. First, we update neural networks in a batched way with candidate batch size $N = 80,000$ instead of updating neural networks immediately and using all samples. Second, we set $r(\theta_{\boldsymbol{x}}) = 1/(1 + \exp(-\theta_{\boldsymbol{x}}))$ since the click-through rate (CTR, the probability of the event that the user clicks the item) should be bounded in $[0, 1]$ (see Section 3). Considering the above differences, we make some adjustments for our TSDR algorithms. For TSDR-T we take

$$\boldsymbol{M}_t = \lambda \boldsymbol{I} + \sum_{i=0}^{N} \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{i_0+i,a_i}; \boldsymbol{w}_t) \boldsymbol{f}_{\text{repr}}(\boldsymbol{x}_{i_0+i,a_i}; \boldsymbol{w}_t)^\top \tag{14}$$

where i_0 is the beginning index of the previous batch. In this setting, $\boldsymbol{M}$ and $\boldsymbol{w}$ are updated at the end of each batch. For TSDR-P we take $\boldsymbol{M}_t = \boldsymbol{I}$.

All algorithms involved in comparison share the same network architecture. Concretely, the user feature in each log entry is a 136-dimensional multi-hot vector, and the item feature is an item ID. We first use a 6-dimensional embedding layer with sum pooling to convert the user features to a 6-dimensional dense vector; we also embed the item ID to another 6-dimensional vector. Then we concatenate them to get a 12-dimensional vector, followed by three fully-connected layers (16-8-1) with ReLU as the activation function to predict the logit value of the reward mean. In this case, the reward mean is CTR.

The algorithm hyperparameters are chosen on log entries of the first day by grid search, and log entries of the last 14 days are used for evaluation. We train neural networks in a mini-batch way with mini-batch size 64, and use the cross entropy loss along with the Adam optimizer to update the network weights with learning rate 0.01. We take the cumulative reward and the average daily CTR on the last 14 day as metrics. All experiments are repeated 8 times.

Figure 2 shows the cumulative reward and the daily CTRs obtained by each algorithm. First, our proposed TSDR-P achieves the best performance on both cumulative reward and daily CTRs. Second, we notice that all these bandit algorithms are significantly better than the greedy algorithm, suggesting the simulation could be a more suitable benchmark. Third, it is demonstrated that the performance of TSDR-T is very close to that of NeuralTS, which supports our theoretical finding that gradient information of all network layers is not essential for exploration.

In addition, we compare the run time of each algorithm and record the results in Table 1. We can observe that the efficiency of our proposed algorithms is close to Greedy. Specifically, TSDR-P only requires an additional sampling from a univariate Gaussian distribution during serving. In contrast, due to the extra calculation of the gradient of all layers for each context, NeuralTS is the slowest, whose run time is about 7 times that of our proposed algorithms.

Table 1: Run time of 10,000 samples on Yahoo!R6B over 10 trials

	Greedy	MCDropout	NeuralTS	TSDR-T	TSDR-P
run time / s	6.0 ± 0.0	15.3 ± 0.3	42.7 ± 0.2	7.4 ± 0.0	6.9 ± 0.0

5.3 Industrial Scenario

Remember that for applying TSDR-P to existing tasks, we only need to change the forward process of the neural network during serving, i.e., adding a routine for sampling from a univariate Gaussian distribution. Owing to its simplicity, we successfully deploy TSDR-P in industrial production — an advertising platform of a massive online retailer. Specifically, the CTR prediction model is armed with TSDR-P to be a TS bandit algorithm to maximize cumulative reward, i.e., revenue and number of clicks in this case. To validate performance of TSDR-P in production, we carefully designed an online A/B testing system. In this system, network traffic is divided into two streams, for which model A and model B serve respectively; more importantly, the data served by model A is only used to retrain model A itself, and so does model B. This mechanism assures that information explored by one model will not be leaked to the other. Our online A/B test lasting for two weeks shows that TSDR-P achieves improvement of revenue and clicks by 5% and 3% respectively compared with the greedy CTR model.

6 Related Work

Thompson Sampling (TS) could date back to 1930s [5, 26], considering the Bernoulli bandit problem with two arms. TS attracts a surge of interests after its strong empirical performance has been demonstrated [7, 27]. Soon after, theoretical results follow for multi-armed bandit problems [28, 29] and linear bandit problems [30, 31].

It is much more complicated for TS to maintain a posterior distribution when modeling the relationship between contexts and rewards with a neural network. Recently, significant effort has been dedicated to approximating Bayesian inference for neural networks, ranging from variational methods to stochastic minibatch Markov Chain Monte Carlo [15–18]. For example, Riquelme et al. [15] proposed to perform a Bayesian linear regression on top of the representation of a neural network, and then make decisions accordingly via Thompson Sampling based on uncertainty estimation. Although it is a natural attempt to combine the representation ability of neural networks with the uncertainty estimation of the Bayesian regression, there is still no theoretical guarantee on the regret bound for such a heuristic algorithm.

By utilizing the neural tangent kernel (NTK) theory [19] and the technique of analyzing Thompson Sampling under linear assumptions [30], Zhang et al. [18] designed the NeuralTS (Neural Thompson Sampling) algorithm and derived a $\mathcal{O}(\sqrt{T})$ regret bound which requires the width of neural network is sufficiently large and is the first regret bound for Thompson Sampling based on neural networks. However, the complexity of the NeuralTS is too high to be applied to practical problems. Different from above works, our proposed algorithm not only has the same regret bound under classical assumptions, but also achieves the state-of-the-art performance with the same complexity as the greedy algorithm.

7 Conclusion

In this paper, we propose a neural network-based Thompson Sampling method, named TSDR, modeling the posterior distribution over the reward mean based on the deep representation. We mainly discuss two instances of TSDR. TSDR-T achieves the $\mathcal{O}(\sqrt{T})$ regret bound under classical conditions, including maintaining a positive definite matrix. TSDR-P is simpler and more efficient, which shows great success empirically, and has been deployed in production. An open question is whether TSDR-P also enjoys the $\mathcal{O}(\sqrt{T})$ regret bound; in other words, whether it is necessary to explicitly model historical context information in the positive definite matrix.

References

[1] Sébastien Bubeck and Nicolo Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Found. Trends Mach. Learn.*, 5(1):1–122, 2012.

[2] Aleksandrs Slivkins. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*, 2019.

[3] Djallel Bouneffouf and Irina Rish. A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*, 2019.

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

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

[6] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *International Conference on World Wide Web (WWW)*, pages 661–670, 2010.

[7] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 24, pages 2249–2257, 2011.

[8] Jaya Kawale, Hung Bui, Branislav Kveton, Long Tran Thanh, and Sanjay Chawla. Efficient thompson sampling for online matrix-factorization recommendation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 28, pages 1297–1305, 2015.

[9] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. A tutorial on thompson sampling. *Found. Trends Mach. Learn.*, 11(1):1–96, 2018.

[10] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on bayesian neural networks–a tutorial for deep learning users. *arXiv preprint arXiv:2007.06823*, 2020.

[11] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *ACM Conference on Recommender Systems (RecSys)*, pages 191–198, 2016.

[12] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 1059–1068, 2018.

[13] Hu Liu, Jing Lu, Xiwei Zhao, Sulong Xu, Hao Peng, Yutong Liu, Zehua Zhang, Jian Li, Junsheng Jin, Yongjun Bao, and Weipeng Yan. Kalman filtering attention for user behavior modeling in ctr prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[14] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. In *Case Studies in Applied Bayesian Data Science*, pages 45–87. Springer, 2020.

[15] 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 (ICLR)*, 2018.

[16] Dalin Guo, Sofia Ira Ktena, Pranay Kumar Myana, Ferenc Huszar, Wenzhe Shi, Alykhan Tejani, Michael Kneier, and Sourav Das. Deep bayesian bandits: Exploring in online personalized recommendations. In *ACM Conference on Recommender Systems (RecSys)*, pages 456–461, 2020.

[17] Branislav Kveton, Manzil Zaheer, Csaba Szepesvári, Lihong Li, Mohammad Ghavamzadeh, and Craig Boutilier. Randomized exploration in generalized linear bandits. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 2066–2076, 2019.

[18] Weitong Zhang, Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural thompson sampling. In *International Conference on Learning Representations (ICLR)*, 2021.

[19] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 8571–8580, 2018.

[20] Nicolas Brosse, Carlos Riquelme, Alice Martin, Sylvain Gelly, and Éric Moulines. On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. *arXiv preprint arXiv:2001.08049*, 2020.

[21] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with UCB-based exploration. In *International Conference on Machine Learning (ICML)*, volume 1, pages 11492–11502, 2020.

[22] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] Dheeru Dua and Casey Graff. Uci machine learning repository, 2017.

[24] Lihong Li, Wei Chu, John Langford, and Xuanhui Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *International Conference on Web Search and Web Data Mining (WSDM)*, pages 297–306, 2011.

[25] Chao Du, Yifan Zeng, Zhifeng Gao, Shuo Yuan, Lining Gao, Ziyan Li, Xiaoqiang Zhu, Jian Xu, and Kun Gai. Exploration in online advertising systems with deep uncertainty-aware learning. *arXiv preprint arXiv:2012.02298*, 2020.

[26] William R. Thompson. On the theory of apportionment. *American Journal of Mathematics*, 57(2):450, 1935.

[27] Steven L. Scott. A modern bayesian look at the multi-armed bandit. *Applied Stochastic Models in Business and Industry*, 26(6):639–658, 2010.

[28] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory (COLT)*, 2012.

[29] Shipra Agrawal and Navin Goyal. Further optimal regret bounds for thompson sampling. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 99–107, 2013.

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

[31] Marc Abeille and Alessandro Lazaric. Linear thompson sampling revisited. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 176–184, 2017.

[32] Pan Xu, Zheng Wen, Handong Zhao, and Quanquan Gu. Neural contextual bandits with deep representation and shallow exploration. *arXiv preprint arXiv:2012.01780*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]