A Parameter Updates for Learning to Interact Algorithms

Table 1 presents the parameter updates for the various learning to interact algorithms discussed in the main paper. The second, third, and fourth columns correspond to lines 0, 4, and 9 of Algorithms 1-2 of the main paper.

B Sensitivity Analysis

Next, we present a set of experiments to determine the sensitivity of collaborative learning to interact algorithm Low Rank TS to different choices of hyper-parameters. Specifically, we study how sensitive the algorithm is to (a) the initial parameter setting of the covariance matrices of the latent factors, (b) the noise present in the Gaussian rewards, (c) the rank of the latent factors. For the first two we consider synthetic data which exhibit the CF principle; for (c) we use the real datasets described in Table 1 in the main paper. In what follows, we describe our synthetic data setup, and then we discuss our sensitivity results.

Synthetic Data. For Gaussian rewards, we generated our data via the PMF model [8], i.e., every entry $r_{ij}$ of the reward matrix $R$ is $r_{ij} = \mathbf{u}_i^T \mathbf{v}_j + \epsilon_{ij}$. For the Bernoulli rewards, we generated our data based on the logistic matrix factorization model: every entry $r_{ij} \sim \text{Bernoulli}(\text{sigmoid}(\mathbf{u}_i^T \mathbf{v}_j^*))$. For both cases we created different data sets, varying both the number of users $M$ and the number of items $N$ in the range of $\{10, 100, 1000\}$. For every $M, N$ combination, we created five random data sets with different random seeds and report results averaged over them.

Effect of $\sigma_u, \sigma_v$ for Bernoulli. To determine the effect of the initialization of the covariance matrices $S_u, S_v$, on Low Rank TS, we vary $\sigma_u^2, \sigma_v^2$ in the range of $\{0.001, 0.01, 0.1, 1, 10, 100\}$. We present our results for Bernoulli rewards for the synthetic data set of $M = 100, N = 1000$ in Figure 1(a). We can see that typically, small values of these hyper-parameters lead to lower cumulative regret. We observed a similar trend for the real data sets for both types of rewards.

Effect of noise $\sigma_{ij}$ for Gaussian. Next, we study how the noise level in the user feedback under Gaussian rewards affects Gauss Low Rank TS compared to the Gauss Low Rank Greedy baseline. We vary the noise standard deviation $\sigma_{ij}$ to $\{0.5, 1.5\}$ and report the results in Figure 1(b) for the synthetic data set of 1,000 users interacting with 1,000 items. Similar trends hold for the real data sets too. We can see that when the noise level is small, Low Rank Greedy and Low Rank TS have similar performance, and the former tends to be better. However, when the noise is larger, TS is more robust and outperforms Greedy. We expect real-world user feedback to be noisy, making Thompson Sampling a possibly better choice.

Effect of Rank. To study the effect of the rank $d$ of the latent factors $U, V$ on the performance of Low Rank TS, we vary $d$ in the range of $\{2, 5, 10, 20\}$. Based on Figure 1(c)-(d), we can see that for Movielens 100K with Bernoulli rewards, rank 2 results in the lowest cumulative regret, while for the data set of Netflix with Gaussian rewards, rank 20 is the best, as the number of observed ratings is an order of magnitude larger. Although in the main paper we present results with rank set to 2, this brief sensitivity experiment shows that a tuned value of the rank would result in even smaller regret.

C Underlying Parametric Assumptions for User Rewards

Here we provide some more details in the derivation of the learning to interact algorithms under the different parametric assumptions described in Section 3.1 of the main paper, and reported in Table 1 here.

C.1 Independent Bandits Independent Gaussian Bandits. For a given user $i$, when $r_{ij} \sim \text{Gaussian} (\theta_{ij}, \sigma_{ij}^2)$, with $\sigma_{ij}^2$ known and fixed and unknown mean $\theta_{ij}$ for every arm $j$, we can model the prior distribution of the reward mean $\theta_{ij}$ with the Gaussian distribution $\theta_{ij} \sim \mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$, as it is the conjugate prior to the Gaussian distribution [3]. After incorporating the
LinTS assumes that the posterior distribution of \( d_j \) have linear structure: 

\[ a_j = n_j + 1 | r_j = 1 \]

\[ b_j = n_j + 1 | r_j = 0 \]

This algorithm is referred to as **Independent Bernoulli TS** [2].

### C.2 Contextual Linear Bandits

**LinTS** [5].

- **Bernoulli TS** [2]:
  
  \[ a_j = b_j = 1 \]

- **BetaTS**: 
  
  \[ \text{Beta}(\pi_j | a_j, b_j, r_j) \]

\[ a_j = a_j + 1 | r_j = 1 \]

\[ b_j = b_j + 1 | r_j = 0 \]

- **LinTS** [5]:
  
  \[ A_i = \beta_{d_{j,i}} b_i = 0_{d \times 1} \]

\[ \theta_i = 0_{d \times 1} \]

\[ \beta_i \sim \text{Beta}(\pi_j | a_j, b_j, r_j) \]

\[ A_i = A_i + x_j^T \]

\[ b_i = b_i + r_j x_j \cdot \theta_i = A_i^{-1} b_i \]

- **LogTS** [6]:
  
  \[ A_i = \beta_{d_{j,i}} b_i = 0_{d \times 1} \]

\[ \theta_i = 0_{d \times 1} \]

\[ \beta_i \sim \text{Beta}(\pi_j | a_j, b_j, r_j) \]

\[ A_i = A_i + p_j (1 - p_j) x_j x_j^T \]

\[ \theta_i = \beta_i - n_j x_j - r_j x_j \]

- **CluLinTS** [7]:
  
  \[ A_i = \beta_{d_{j,i}} b_i = 0_{d \times 1} \]

\[ \theta_i = 0_{d \times 1, \# \text{clusters}} \]

\[ \beta_i \sim \text{Beta}(\pi_j | a_j, b_j, r_j) \]

\[ A_i = A_i + x_j^T \]

\[ b_i = b_i + r_j x_j \cdot \theta_i = A_i^{-1} b_i \]

- **CluLogTS** [7]:
  
  \[ A_i = \beta_{d_{j,i}} b_i = 0_{d \times 1, \# \text{clusters}} \]

\[ \theta_i = 0_{d \times 1, \# \text{clusters}} \]

\[ \beta_i \sim \text{Beta}(\pi_j | a_j, b_j, r_j) \]

\[ A_i = A_i + x_j^T \]

\[ b_i = b_i + r_j x_j \cdot \theta_i = A_i^{-1} b_i \]

---

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Prior/ Parameter Init.</th>
<th>Posterior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent</td>
<td>( \mathcal{N}(\theta_j</td>
<td>\mu_j, \sigma_j^2) )</td>
</tr>
<tr>
<td>Gauss TS</td>
<td>( \mathcal{N}(\theta_j</td>
<td>\mu_j, \sigma_j^2) )</td>
</tr>
</tbody>
</table>

**Table 1**: Summary of parametric assumptions in learning to interact recommendation algorithms. After incorporating the feedback \( r_j \) of user \( i \) on shown item \( j \), the new posterior becomes the previous posterior. Time indices are omitted to avoid clutter. The \( \hat{\cdot} \) symbol denotes sampled values.

---

**reward of the item shown \( j \) for the incoming user \( i \), using Bayes rule we can update the posterior, which will also be a Gaussian due to conjugacy, with mean and variance specified in Table 1. We refer to this as **Independent Gauss TS**.**

**Independent Bernoulli Bandits.** For a user \( i \), when \( r_{ij} \) is Bernoulli(\( \pi_{ij} \)) it is standard to model the prior distribution of the reward of every arm \( j \) with the Beta distribution \( \pi_{ij} \sim \text{Beta}(a_{ij}, b_{ij}) \), as it is the conjugate prior to the Bernoulli distribution. Due to conjugacy, the posterior will be a Beta distribution \( \pi_{ij} \sim \text{Beta}(a_{ij}, b_{ij}) \) with new parameters \( a_{ij}', b_{ij}' \) based on number of observed successes \( 1 \) and failures \( 0 \). This algorithm is referred to as **Independent Bernoulli TS**, and details for each Bernoulli bandit can be found in [2].**

---

**C.3 Contextual Clustering Bandits**

**CluLinTS** [7]. In [7], users are allowed to move to other clusters, based on context.

---

**LogTS** In [6], Li et al. extended contextual linear bandits for the case when the rewards come from a generalized linear model (e.g. Bernoulli, Poisson, etc.) [1]. Here, we consider only when rewards come from a Bernoulli distribution, where the reward assumption is that \( \mathbf{E}[r_{ij}] = \sigma(\theta_i^T x_j) = 1/(1 + \exp(-\theta_i^T x_j)) \). LogTS uses online logistic regression to update the underlying parameters. Given that the posterior distribution is not a Gaussian anymore due to the non-linearity introduced, moment matching is employed to approximate it with a Gaussian \( \approx \mathcal{N}(\theta_j | \hat{\theta}_j, A_j^{-1}) \) [1]. Also, as there is no closed form update for \( \theta_i \), an iterative algorithm is needed.
(a) Synthetic Data $M = 100$, $N = 1000$
(b) Synthetic Data $M = N = 1000$
(c) ML100K, Bernoulli
(d) Netflix, Gaussian

Figure 1: (a) Effect of variance $s = \sigma_u, \sigma_v$ on Bernoulli Low Rank TS for Bernoulli synthetic data with 100 users, 1000 items. Smaller values of $s$ lead to lower $R_T$. (b) Effect of noise $\sigma_{ij} = \{0.5, 1.5\}$ on Gauss Low Rank TS vs. Low Rank Greedy for Gaussian synthetic data with 1000 users, 1000 items. For higher noise, Low Rank TS outperforms Low Rank Greedy. (c)-(d) Effect of the latent factor rank $d$, varying in $\{2, 5, 10, 20\}$ on Low Rank TS.

(a) Movielens 100K
(b) Movielens 1M
(c) Netflix
(d) Yahoo

Figure 2: Comparison of one-item recommendation methods for Gaussian Rewards. Low Rank (LR) TS is among the top performing methods for Movielens 100K and Movielens 1M, and outperforms all for larger scale data sets: Netflix and Yahoo.

(a) Movielens 100K
(b) Movielens 1M
(c) Netflix
(d) Yahoo

Figure 3: Comparison of one-item recommendation methods for Bernoulli Rewards. Low Rank (LR) TS and Greedy Low Rank are the best performing methods.

CluLogTS. (new method) Here, we devise a new contextual clustered TS algorithm for Bernoulli rewards, referred to as CluLogTS, to better capture binary data, such as click/no-click. CluLogTS assumes that the reward of item $j$ for user $i$ assigned to cluster $c$ follows: $E[r_{ij}] = 1/(1+\exp(-\theta^*^\top x_j))$. The parameters of the reward distribution of every user $i$ are learned via LogTS. Initially, users are randomly assigned to clusters, and cluster parameters are initialized. At every round, an item $j$ is shown to the incoming user $i$ based on the sampled parameters of the corresponding cluster $c$ the user is currently assigned to. After incorporating the observed feedback $r_{ij}$, user $i$’s LogTS parameters get updated, the user is allowed to move to the closest cluster based on the proximity of the user-cluster parameters, and the clusters’ parameters get updated based on the parameters of the users belonging to the cluster.

C.4 CF low-rank Bandits ICF. ICF [9] was the first work to use CF for recommendation bandits.
However, the authors formulated the problem as a contextual linear bandit, where they used as contextual features the latent features of the items as pre-learned from training PMF [8] on some user-item observations. In their formulation, \( v_j \)'s are given as input to the algorithm and only \( u_i \)'s are learned online. They devised both Gauss ICF and Bernoulli ICF.

**Gauss Low Rank TS**, variants of which have been considered in [4], assumes that the mean reward of the arms follows the low-rank assumption, i.e., \( \mu_{ij} = u^T_i v_j \). In particular, **Gauss Low Rank TS**, using as the underlying model PMF [8], makes the following parametric assumptions for the underlying parameters of the reward distribution at time step \( t \):

\[
\forall i: u^t_i \sim \mathcal{N}(\hat{u}^t_i, S_{u^t_i}) \quad \forall j: v^t_j \sim \mathcal{N}(\hat{v}^t_j, S_{v^t_j}) \\
\forall ij: r^t_{ij} \sim \mathcal{N}(u^T_i v^t_j, \sigma^2_{ij})
\]

where \( \hat{u}_i, \hat{v}_j, S_{u_i}, S_{v_j} \) is defined in Table 1.

**C.5 Cascade Ranking Bandits** The parametric assumptions of the cascade bandits are the same as the ones from the corresponding one-item recommenders.

**D Full Real Data Experiments**

Figures 2-5 contain the experiments on all four real datasets, for Bernoulli and Gaussian reward models, for one-item and cascade list recommendation setup.

**References**