1 Datasets description

1.1 Non-contextual

Random Synthetic. We create synthetic datasets to represent our set of actions $\mathcal{A}$. Each dataset will be composed of a set number (200 and 2,000) of vectors (actions) of $d \in \{100, 1000\}$ dimensions and where each value is sampled from a Gaussian distribution, $\mathcal{N}(0, \sigma^2)$. The resulting vectors are normalized, i.e., $\forall a \in \mathcal{A}, \|a\|_2 = 1$. Finally, we generate a vector of size d representing θ_*, and we ensure the linear form of the gain obtained at a specific time t: $r_t = \langle \theta_*, a_t \rangle + \eta$. We set $\eta = 0.1$, in line with experimental evaluation in the literature.

MovieLens 25M.[1] The following dataset is composed of movie ratings over 162,541 users and 62,423 movies. We first filter the users such that we keep only the users that rated between 1,500 and 3,000 movies. To build the vector representation of a movie, we extract the following features: (1) year of release, (2) genres, (3) statistical measures (mean, median, std, min, max) of the movie's ratings, and (4) the users that rated the movie (one-hot encoding). For each user in the dataset, we construct an itemset of movies rated by them. The resulting movie vectors are of dimension $d = 813$. For each round of the bandit algorithms, we proceed as follows: we select a user randomly, recommend a movie from the itemset and observe the reward as the real rating the user assigned the movie, plus a small noise $\eta \sim \mathcal{N}(0, \sigma^2)$.

MNIST.[2] We follow the experiment setup described in Kuzborskij et al. (2019), by adapting the classification problem to the bandit case. Given a dataset of C classes, we first select a target class. Then, at each iteration, the environment randomly chooses one sample from each class ($C = 10$ in this case, corresponding to the digits in the dataset), and then creates a set of C samples. The learner then selects one sample from the set and observes the reward, which is 1 if the selected sample belongs to the target class and 0 otherwise. The digits are represented as vectors of dimension $d = 784$ and we have a total of 60,000 samples.

Steam.[3] The dataset originally contains reviews from 7,495,040 users of 37,032 games. As in the Movielens dataset, we first filter the users such that we keep only users that recommended (positive or negative recommendations) more than 200 games. This reduces the number of users to 1,347. Then, we construct the game vector features using: (1) the release date, (2) OS support (Windows, Mac, Linux), (3) general appreciation of the game (Very positive, ..., mixed, ..., very negative), (4) ratio of positive ratings, (5) number of reviews, (6) prices (original price, discount if any, final price), (7) one-hot encoding of tags, (8) one-hot encoding of the users that played the game (only out of the 1,347 users we kept). This results in game vectors of dimension $d = 1{,}807$. We estimate the parameter θ_* corresponding to a user using linear regression between the vectors of games played by this user and their corresponding recommendations (positive/negative). We then construct the itemset of a user using the 2,000 most reviewed games that the user has not yet played. At each run, we select a random user, the algorithm recommends then a game a_t from the corresponding itemset and observes a reward which is computed as $r_t = \langle \theta_*, a_t \rangle + \eta$.

Amazon books.[4] This dataset consists of reviews from 2,878,115 users for 7,797,481 books. We filter the users such that we keep only users that reviewed more than 350 books. This reduces the number of users to 648. The book vectors are then constructed by concatenating one-hot encoding of users and book features (categories, price, number of reviews, minimum/maximum/median/mean of ratings.). The parameter θ_* corresponding to a user is estimated using linear regression between books vectors reviewed by them and their corresponding ratings. Similar to Steam, we then construct the itemset of a user using 1,000 most reviewed

[1] https://grouplens.org/datasets/movielens/
[2] https://pjreddie.com/projects/mnist-in-csv/
[3] https://kaggle.com/datasets/antonkozyriev/game-recommendations-on-steam
[4] https://amazon-reviews-2023.github.io/data_processing/index.html

books that the user have not yet reviewed. At each run, we select a random user, the algorithm recommends then a book a_t from the corresponding itemset and observes a noisy linear reward of similar form to Steam.

Yahoo.[5] This dataset contain reviews of 271 articles from 29,849,370. Users are filtered such that only users that interacted with more than 50 articles and clicked on more than 5 articles are kept, reducing the number of users to 1,550. The article vectors consist of one-hot encoding of the user(s) having interacted with the article and the article features directly taken from the dataset. The parameter θ_* corresponding to a user is estimated using linear regression between article vectors which the users interacted with and their corresponding interactions. The itemset of a user consists of 200 articles with which the user has not interacted. At each run, we select a random user, the algorithm recommends then an article a_t from the corresponding itemset and observes a noisy linear reward between the article vector and the user vector.

2 Contextual

MovieLens 25M. We filter users such that we keep only 20 users. To build the vector representation of a movie, we extract the same features as in the non-contextual case. User vectors are context vectors and are constructed using one hot encoding. At each iteration, we proceed as follows: we select a user randomly (context vector), and construct a contextual itemset of 3000 most rated movies (concatenation of selected context vector and items). Then, the reward observed by an algorithm when recommending a movie is 1 if the user has interacted with the corresponding movie in the original dataset and 0 otherwise.

Yahoo.

We filter users such that we keep only 10 users. Article vectors are built as in the non-contextual case. User vectors are context vectors and are available in the dataset (6-dimensional vectors). We then proceed as in Movielens dataset (select a user, construct contextual itemset, observe a reward equal to 0 or 1). In this dataset, we select 200 most reviewed articles.

Steam. We filter users such that we keep only 10 users. Games vectors are built as in the non-contextual case. Users vectors are context vectors and are constructed using one hot encoding. We then proceed as in Movielens and Yahoo datasets (select a user, construct contextual itemset, and observe a reward equal to 0 or 1). In this dataset, we select the 2000 most rated games.

In general, as real-world datasets are characterized by a large number of users and items, we focused on the most relevant users – those who have interacted with a sufficient number of items – allowing a good application of the algorithm with a low number of iterations constraint. Also, these thresholds were set differently for each dataset, based on the number of already available features and to ensure reasonable execution times across various algorithms.

[5] https://webscope.sandbox.yahoo.com/catalog.php?datatype=r&did=49