Dynamic Clustering of Contextual Multi-Armed Bandits

Trong T. Nguyen
School of Information Systems
Singapore Management University
ttnguyen.2014@smu.edu.sg

Hady W. Lauw
School of Information Systems
Singapore Management University
hadywlauw@smu.edu.sg

ABSTRACT

With the prevalence of the Web and social media, users increasingly express their preferences online. In learning these preferences, recommender systems need to balance the trade-off between *exploitation*, by providing users with more of the "same", and *exploration*, by providing users with something "new" so as to expand the systems' knowledge. Multi-armed bandit (MAB) is a framework to balance this trade-off. Most of the previous work in MAB either models a single bandit for the whole population, or one bandit for each user. We propose an algorithm to divide the population of users into multiple clusters, and to customize the bandits to each cluster. This clustering is dynamic, i.e., users can switch from one cluster to another, as their preferences change. We evaluate the proposed algorithm on two real-life datasets.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.8 [**Database Applications**]: Data Mining

Keywords

multi-armed bandit; clustering; exploitation and exploration

1. INTRODUCTION

With the rapid growth of the Web and the social media, users have to navigate a huge number of options in their daily lives. To help users in making these choices, content providers rely on recommender systems [1, 10] that learn user preferences based on their historical activities. In a rapidly changing environment [2], where new items appear all the time, relying on historical data alone (*exploitation*) may not work as well. Instead, what is needed is another paradigm that can continually explore the space of user preferences (*exploration*) as new items appear, or as users change their preferences. The exploitation vs. exploration trade-off refers to balancing the short term interest of making the next recommendation as accurate as possible, with the long term interest in learning about the users as much as possible (perhaps at the cost of lower accuracy in the short term).

One such paradigm is the multi-armed bandit [14]. A bandit (a recommender system) has multiple arms (items to recommend). Pulling an arm (recommending an item) generates some amount of reward (such as user liking the recommendation). This reward is not known in advance. Because the bandit has multiple chances, the main objective is to maximize the accumulated rewards (or to minimize the regret of not pulling the best arms) over time. For this, the bandit should not just pull the arms that produce the highest rewards in the past, but also explore other arms that could potentially generate even higher rewards in the future.

Multi-armed bandits have been shown to work well in various Web recommendation scenarios, such as advertisements [2], news articles [11], and comments [12]. In many cases, it is advantageous to contextualize the bandit, such that the reward of an arm also depends on the "context" of a recommendation, e.g., the content of a Web page (see Section 4).

In this paper, we study the research question of whether there is an appropriate number of bandits to serve a population of users. Most current approaches fall into two extremes. One option is to build a single bandit for all users, which has the advantage of scale, in learning from the observed rewards of many users. However, a global recommendation may not be appropriate for all users. Another option is to personalize it completely, by building one bandit for every user, which is fully customized to every individual, but may suffer from the sparsity of learning instances.

To address the above disadvantages of the two extremes, we advocate a *dynamic clustering* approach. In this approach, the population of users are partitioned into K clusters. The bandits of individual users in the same cluster can "collaborate" in estimating the expected reward of an arm for any one user in the same cluster. That way, we can keep a bandit customized to an individual user, and yet allow users to benefit from the collective set of learning instances in their cluster. Moreover, as a user changes preferences, or as we learn the user's preference better, the user may switch from one cluster to another more "suitable" cluster.

Contributions. *First*, we propose a clustering-based contextual bandit algorithm called DynUCB in Section 2, building on the contextual bandit LinUCB [11]. Its novelty arises from dynamic clustering, which we relate to existing work in Section 4. *Second*, in Section 3, we verify the efficacy of this approach through experiments on two real-life datasets, studying the appropriate number of clusters for each dataset, and comparing against the baselines.

CIKM'14, November 3–7, 2014, Shanghai, China.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2662063.

2. DYNAMIC CLUSTERING OF BANDITS

We first review the framework of contextual bandit, before introducing our proposed algorithm, which we call DynUCB.

Contextual Bandit. A contextual bandit algorithm proceeds in discrete iterations. At any iteration t, the bandit observes a particular user u_t. There are also a set of available arms $\mathcal{A}_t$ that the bandit may choose to pull for this user. For each arm $a \in \mathcal{A}_t$, the bandit observes its context, in the form of a d-dimensional feature vector $\mathbf{x}_{t,a} \in \mathbb{R}^d$. Based on the reward experience in previous iterations, the bandit may choose to pull an arm a_t. Upon pulling a_t, the bandit observes a reward r_{t,a_t}. There is no reward observation for $a \neq a_t$. The bandit therefore needs to learn from the observations of $\langle \mathbf{x}_{t,a_t}, a_t, r_{t,a_t} \rangle$ for ongoing iterations t's to improve its strategy for choosing arms in future iterations.

After T iterations, the bandit would observe a cumulative reward of $\sum_{t=1}^{T} r_{t,a_t}$. The objective is to design an intelligent way to choose arms so as to maximize this cumulative reward over time. Equivalently, we can express the objective in terms of minimizing total regret, where regret is defined as the difference between the observed reward r_{t,a_t} and the reward of the "optimal" arm in each iteration.

A popular framework for contextual bandit is LinUCB [11], which estimates the expected reward of each arm a as a linear regression on the context vector $\mathbf{w}^{\mathrm{T}}\mathbf{x}_{t,a}$, where $\mathbf{w} \in \mathbb{R}^d$ is the regression coefficient to be learned. However, maximizing the expected reward alone may result in a long term regret from not discovering a better arm through exploration. Therefore, it also considers the confidence bound $\alpha\sqrt{\mathbf{x}_{t,a}^{\mathrm{T}}\mathbf{M}^{-1}\mathbf{x}_{t,a}}$. Here, α is a parameter for the importance of exploration. It is expressed as $\alpha = 1 + \sqrt{\ln(2/\delta)/2}$, where $1-\delta$ is the confidence interval. We set $\delta = 0.05$ for 0.95 confidence interval. $\mathbf{M}^{-1} \in \mathbb{R}^{d\times d}$ is the update weights, which can be interpreted as the covariance of the coefficient $\mathbf{w}$. The arm a_t selected is the one maximizing the upper confidence bound: $a_t = \arg\max_{a\in\mathcal{A}_t}\left(\mathbf{w}^{\mathrm{T}}\mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}^{\mathrm{T}}\mathbf{M}^{-1}\mathbf{x}_{t,a}}\right)$.

Clustering of Contextual Bandits. To build a recommender system that serves N users, one option is to build a SINgle instance of LinUCB for all users, which we call LinUCB-SIN. Another option is to train a bandit for every INDividual user, which we call LinUCB-IND. The former benefits more from the wealth of training instances, while the latter benefits from a more customized bandit. However, we hypothesize that a large population of users are neither as monolithic as in LinUCB-SIN, nor as heavily splintered as in LinUCB-IND. Rather, there may be several communities or clusters in the population, where users within the same cluster may share preferences. By grouping together like-minded users, we can benefit from having a larger number of training instances, while still customizing the bandits.

We therefore propose an algorithm, called DynUCB, as described in Algorithm 1. Since the appropriate number of clusters may vary in different domains and populations, the algorithm takes as its input the desired number of clusters K. Initially, we start out with K random clusters, denoted C_k for $k = 1, \ldots, K$, and refine the clustering over iterations.

In a way, DynUCB still maintains N bandits for N users. For each user u, its coefficient $\mathbf{w}_u$ is learned from its own bandit parameters $\mathbf{b}_u$ and $\mathbf{M}_u$ (initialized to $\mathbf{0}$ and identity matrix I respectively). However, unlike LinUCB-IND's N independent bandits, in DynUCB the bandits in the same cluster C_k "collaborate" with one another. For instance, at iteration t, when generating a recommendation for $u_t \in C_k$, the estimation of expected reward for each arm $a \in \mathcal{A}_t$, i.e., $\left(\bar{\mathbf{w}}_k^{\mathrm{T}}\mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}{}^{\mathrm{T}}\bar{\mathbf{M}}_k^{-1}\mathbf{x}_{t,a}\log(t+1)}\right)$, is based on cluster-level coefficient $\bar{\mathbf{w}}_k$, learned from cluster-level parameters $\bar{\mathbf{b}}_k$ and $\bar{\mathbf{M}}_k$ derived from the bandit parameters $\mathbf{b}_u$ and $\mathbf{M}_u$ of each user $u \in C_k$. The confidence bound is a simplified version of the theoretical confidence bound shown in [8].

Algorithm 1: DynUCB

```
Input: The number of clusters $K$.
Output: At iteration $t$, recommended arm $a_t$ for $u_t$.

Set $\mathbf{b}_u = \mathbf{0} \in \mathbb{R}^d$, $\mathbf{M}_u = I \in \mathbb{R}^{d\times d}$ for all users $u = 1, \ldots, N$.
Randomly assign users to $K$ clusters $\{C_k\}_{k=1}^{K}$.
Compute the coefficient $\bar{\mathbf{w}}_k$ for each cluster $C_k$:
    $\bar{\mathbf{M}}_k = I + \sum_{u'\in C_k}(\mathbf{M}_{u'} - I)$
    $\bar{\mathbf{b}}_k = \sum_{u'\in C_k}\mathbf{b}_{u'}$
    $\bar{\mathbf{w}}_k = \bar{\mathbf{M}}_k^{-1}\bar{\mathbf{b}}_k$
for iteration $t = 1, \ldots, T$ do
    Select a user $u_t$, and its current cluster $C_k \ni u_t$.
    Observe the contexts of arms $\{\mathbf{x}_{t,a}\}, \forall a \in \mathcal{A}_t$.
    Find the arm $a_t$ with the highest UCB, i.e.,
    $a_t =$
    $\arg\max_{a\in\mathcal{A}_t}\left(\bar{\mathbf{w}}_k^{\mathrm{T}}\mathbf{x}_{t,a} + \alpha\sqrt{\mathbf{x}_{t,a}{}^{\mathrm{T}}\bar{\mathbf{M}}_k^{-1}\mathbf{x}_{t,a}\log(t+1)}\right)$
    Observe the reward $r_{t,a_t}$ from recommending $a_t$.
    Let $\tilde{\mathbf{x}}_t = \mathbf{x}_{t,a_t}$.
    Update the user $u_t$'s parameters:
        $\mathbf{M}_{u_t} = \mathbf{M}_{u_t} + \tilde{\mathbf{x}}_t\tilde{\mathbf{x}}_t^{\mathrm{T}}$
        $\mathbf{b}_{u_t} = \mathbf{b}_{u_t} + r_{t,a_t}\tilde{\mathbf{x}}_t$
        $\mathbf{w}_{u_t} = \mathbf{M}_{u_t}^{-1}\mathbf{b}_{u_t}$
    Re-assign the user $u_t$ to the closest cluster $C_{k'}$:
        $k' = \arg\min_{k'=1,\ldots,K}||\mathbf{w}_{u_t} - \bar{\mathbf{w}}_{k'}||$
        If $k' \neq k$, move $u_t$ from $C_k$ to $C'_k$.
        Re-compute coefficients $\bar{\mathbf{w}}_k$ and $\bar{\mathbf{w}}_{k'}$ as above.
```

Consequently, each user benefits from the reward experiences of other users in the same cluster. The observed reward r_{t,a_t} from recommending the arm a_t to u_t is then used to update u_t's own coefficient $\mathbf{w}_{u_t}$, which reflects u_t's reward experience over iterations. Due to the clustering hypothesis, u_t benefits more from belonging to the "right" cluster of like-minded users that complement one another. Therefore, at each iteration, we re-assign u_t to the cluster $C_{k'}$ whose coefficient $\bar{\mathbf{w}}_{k'}$ is closest to $\mathbf{w}_{u_t}$, a practice reminiscent of the K-means clustering algorithm but conducted within the contextual bandit framework. This dynamic re-assignment of clusters is a key feature of DynUCB, allowing it to be adaptive to changing contexts and user preferences over time.

3. EXPERIMENTS

The objective of experiments is to investigate the effectiveness of our proposed method DynUCB. First, we describe the two real-life datasets for experiments. Then, we investigate the effects of different number of clusters, before presenting a comparison against state-of-the-art baselines.

3.1 Experimental Setup

Datasets. We use two publicly-available[1] datasets that have previously been used for contextual bandits evaluation [6]. The first dataset is on the social bookmarking site **Deli-**

[1]`http://grouplens.org/datasets/hetrec-2011`

	Delicious	LastFM
No. of unique users	1867	1892
No. of unique tags	11619	9643
No. of unique items	69226	17632
No. of unique <user, item> pairs	104220	71064

Table 1: Dataset Sizes

cious, where a set of users assign tags to a set of bookmarked URLs. The second dataset is on the online radio **LastFM**, where a set of users assign tags to a set of music artists. We follow similar processing steps as in [6]. The statistics for these datasets after processing are shown in Table 1.

The task of interest is to recommend a new item to a user, where an item refers to a bookmark URL for Delicious, and a music artist for LastFM. Importantly, for both datasets, the tags are used to generate the contexts for items, as follows. First, we treat each item as a "document" consisting of tags (and their frequencies) assigned by all users. Then, we compute a TFIDF vector for each item from the "document" representations. We further reduce this vector into a 25-dimensional context vector using PCA [9].

Metric. The prediction task is as follows. For every round t of the bandit algorithm, for the user u_t, we pick one of her items i randomly. We then present the context vector of the item i, together with 24 other randomly generated context vectors, to a bandit algorithm. If it makes the correct recommendation, i.e., it picks the item i out of the 25 options, the reward is 1. Otherwise, the reward is $-\frac{1}{24}$. Random guesses are expected to have a cumulative reward of 0. A better algorithm is expected to have a higher positive cumulative reward over iterations. We consider $T = 50000$ iterations, which is considered large. For all algorithms, we average the cumulative rewards across ten different runs.

3.2 Number of Clusters

Here, we study the relationship between the number of clusters in DynUCB with the cumulative rewards.

Delicious. First, we consider the case of Delicious. Figure 1(a) shows the cumulative rewards of DynUCB after 50000 iterations, for different number of clusters K's. As we increase K from 1 to 256, there is a trend whereby the rewards at first increase, reaching the peak at around $K = 16$, and then eventually begin to decrease. This trend helps to validate that the clustering hypothesis indeed applies to the Delicious dataset. We hypothesize that being a social bookmarking website, Delicious may support a number of user communities, e.g., technology, music, sports. By clustering the bandits, we can customize the bandits to cater to different communities, while still benefiting from the collection of training instances from users of that community. Having too few or too many clusters may be counter-productive, as we begin to clump unrelated users, or to split related users.

It is also interesting to look into the distribution of cluster sizes. For $K = 16$, we get one large cluster containing 58% of the users, and the other 15 small clusters are roughly even-sized, containing between 2% to 5% of the users. These numbers are based on one specific run, but we observe virtually similar distributions across all the runs. This suggests the presence of one main group, and several smaller communities that benefit from having more customized bandits.

LastFM. Figure 1(b) shows a very different picture for the LastFM dataset. It shows that cumulative reward of DynUCB after 50000 iterations is highest for $K = 1$, and goes downhill for larger K's. This result is very revelatory, suggesting that there are populations, such as in LastFM, where most of the general users pretty much agree in their preferences. This potentially arises from the phenomenon where there are a few artists that practically everyone listens to, unlike social bookmarking (Delicious) where different users may have different bookmark preferences.

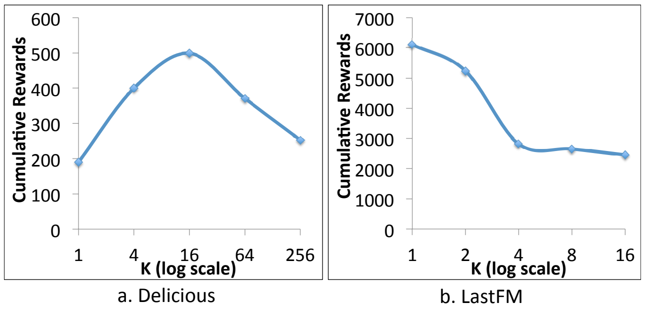

Figure 1: Vary Number of Clusters K

3.3 Comparison against Baselines

We now compare DynUCB (with the optimal number of clusters found in the previous section, i.e., $K = 16$ for Delicious, and $K = 1$ for LastFM) against baselines.

Baselines. Since we propose a dynamic clustering approach, we compare to two types of baselines. The first are the non-clustering baselines LinUCB-IND and LinUCB-SIN. The second is a clustering baseline CLUB [8], which is hierarchical and does not model dynamic movements between clusters (see Section 4). Because CLUB[2] assumes an input graph, we use a complete graph of users so as not to restrict the clustering that it could discover. We have also tried randomly-generated input graphs as described in [8], but find the results to be worse than a complete graph. We tune its parameter α_2 in the range 0 to 1, and use the best parameter at 5000 iterations ($\alpha_2 = 0.55$ for Delicious, $\alpha_2 = 0.9$ for LastFM) to obtain the rewards for 50000 iterations.

Delicious. For Delicious, Figure 2 shows the cumulative rewards over iterations up to $T = 50000$. Evidently, DynUCB at $K = 16$ has higher cumulative rewards than the baselines over the long run. In the short run (for $t < 20000$), LinUCB-SIN tends to have a higher cumulative reward, because it benefits from "faster" learning from the large number of training instances of all users. Since DynUCB partitions the users into different clusters, and begins with random clusters, it learns more slowly in the early stages as it figures out the clustering. In the long run, DynUCB more than catches up, benefiting from more customized bandits in each cluster. LinUCB-IND performs at a similar level, if slightly lower than LinUCB-SIN. Unexpectedly, the clustering baseline CLUB does not perform well. Upon further investigation, we observe that 90% of users belong to one cluster, while the other users are splintered into 6 clusters of 2 users each, and 175 independent users. It is unclear if this is an artefact of parameter tuning or the algorithm itself.

LastFM. Figure 3 shows the comparison for LastFM. As previously mentioned, LastFM is not conducive for cluster-

[2]We implement CLUB as there is no publicly available implementation at the point of writing.

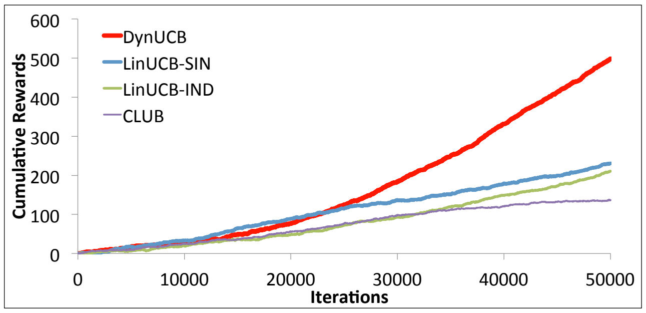

Figure 2: Delicious: Rewards over Iterations

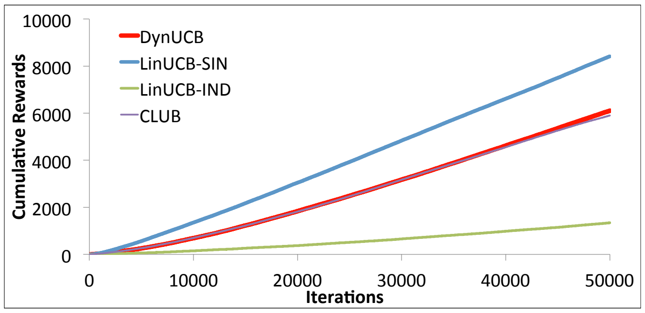

Figure 3: LastFM: Rewards over Iterations

ing, because of the bias for the most popular items. As expected, LinUCB-SIN has the highest reward, for the same reason why DynUCB is optimal for $K = 1$. DynUCB is second, followed by CLUB with a very similar performance. CLUB keeps virtually all the users (∼96%) within a single cluster. LinUCB-IND is the worst, because of the lack of training instances for the independent bandits. Notably, even for a non-conducive dataset, DynUCB does not degenerate completely (unlike LinUCB-IND), and still manages to get a reasonable performance. We interpret this as the need to fit the right algorithm for the right dataset, based on how well the underlying hypothesis holds for the dataset.

4. RELATED WORK

The principle behind bandits is to balance the trade-off between exploration and exploitation. For instance, ϵ-greedy [14] picks a random arm (exploration) with probability ϵ, and picks the arm with the highest expected reward (exploitation) with probability $(1 - \epsilon)$. Instead of a "random" exploration, the Upper Confidence Bound or UCB approach [3, 4] estimates not just the expected reward, but also the confidence interval, of every arm. It then picks the arm with the highest sum of reward and confidence interval, which is the upper confidence bound. Thompson Sampling [7] picks an arm that has the largest success probability.

Contextual bandits make bandit algorithms more adaptive to the changing "contexts". This context is usually expressed as a feature vector. Similar contexts would have correlated rewards. For instance, LinUCB [11], used as a foundation for our method, models the expected reward through a linear regression on context vectors. LogUCB [12] models it through logistic regression.

One related bandit clustering work is CLUB [8]. It models a cluster as a connected component in a graph of users. From the input graph, it slowly removes edges over iterations, splintering the graph into multiple clusters. CLUB and our method seek a partitioning of the user population. There are a couple of crucial differences between the two. Firstly, CLUB does hierarchical clustering, whereas we pursue a flat clustering. Secondly, and more importantly, our clustering is dynamic, allowing users to move between clusters, whereas CLUB only models the splintering of clusters, but not the movement of users across clusters. Other clustering works are based on standard bandits [13, 5]. Previously [11], contextual bandit (i.e., LinUCB) has been shown to outperform standard bandit (i.e., UCB). Social bandits [6] correlate bandits of different users based on a social network graph (which we do not consider).

5. CONCLUSION

We investigate the problem of dynamic clustering of contextual bandits, and propose an algorithm DynUCB. In the case where the clustering hypothesis applies (Delicious), DynUCB achieves a significant gain in rewards when compared to non-clustering bandit baselines. This result points to a promising direction of customizing bandits to specific segments of users who may have distinct preferences. As future work, we plan to investigate the clustering hypothesis further, to analysize the confidence bound the algorithm, and to consider extensions such as overlapping clusters.

Acknowledgments

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

6. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 2005.
[2] D. Agarwal, B.-C. Chen, and P. Elango. Explore/exploit schemes for web content optimization. In *ICDM*, 2009.
[3] P. Auer. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3, 2003.
[4] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3), 2002.
[5] L. Bui, R. Johari, and S. Mannor. Clustered bandits. In *arXiv:1206.4169v1 [cs.LG]*, 2012.
[6] N. Cesa-Bianchi, C. Gentile, and G. Zappella. A gang of bandits. In *NIPS*, 2013.
[7] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. In *NIPS*, 2011.
[8] C. Gentile, S. Li, and G. Zappella. Online clustering of bandits. In *ICML*, 2014.
[9] I. Jolliffe. *Principal component analysis.* Wiley Online Library, 2005.
[10] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.
[11] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *WWW*, 2010.
[12] D. K. Mahajan, R. Rastogi, C. Tiwari, and A. Mitra. LogUCB: an explore-exploit algorithm for comments recommendation. In *CIKM*, 2012.
[13] O.-A. Maillard and S. Mannor. Latent bandits. In *ICML*, 2014.
[14] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning.* MIT Press, 1998.