Cluster-Based Bandits: Fast Cold-Start for Recommender System New Users

Sulthana Shams, Daron Anderson, Douglas Leith
Trinity College Dublin, Ireland

ABSTRACT

How to quickly and reliably learn the preferences of new users remains a key challenge in the design of recommender systems. In this paper we introduce a new type of online learning algorithm, cluster-based bandits, to address this challenge. This exploits the fact that users can often be grouped into clusters based on the similarity of their preferences, and this allows accelerated learning of new user preferences since the task becomes one of identifying which cluster a user belongs to and typically there are far fewer clusters than there are items to be rated. Clustering by itself is not enough however. Intra-cluster variability between users can be thought of as adding noise to user ratings. Deterministic methods such as decision-trees perform poorly in the presence of such noise. We identify so-called distinguisher items that are particularly informative for deciding which cluster a new user belongs to despite the rating noise. Using these items the cluster-based bandit algorithm is able to efficiently adapt to user responses and rapidly learn the correct cluster to assign to a new user.

CCS CONCEPTS

• **Theory of computation → Online learning algorithms**.

KEYWORDS

cold-start, recommender systems, online learning, bandit algorithms

ACM Reference Format:
Sulthana Shams, Daron Anderson, Douglas Leith. 2021. Cluster-Based Bandits: Fast Cold-Start for Recommender System New Users. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3404835.3463033

1 INTRODUCTION

In this paper we revisit the cold-start task for new users of a recommender system, which remains a core challenge. When a new user joins the system it initially has no knowledge of the preferences of the user and so would like to quickly learn these[1]. The recommender system therefore initially starts in an "exploration" phase where the first few items that it asks the new user to rate are chosen with the aim of discovering the user's preferences. For brevity we focus on the simplest setup where a user explicitly rates items presented to them, e.g. on a 1-5 scale or binary like/dislike feedback, and the aim of the recommender system is to predict other items that the user may like.

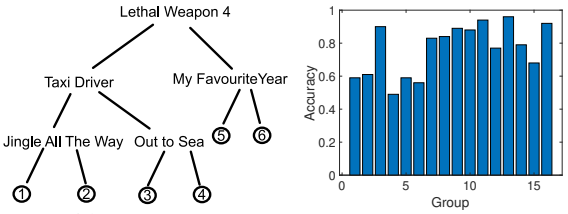

Figure 1: (a) Illustrating a movie recommender decision-tree (adapted from [10]), (b) Decision-tree accuracy for Netflix data (16 groups).

One common approach to this new user cold-start task is to take ratings already collected from a population of users, use these to cluster users into groups and then train a decision-tree to learn a mapping from item ratings to the user group, see for example Figure 1(a). When a new user joins the system this decision-tree is used to decide which items the user is initially asked to rate and in this way the group to which the user belongs is initially estimated. Once the group is estimated, the system recommends items liked by members of that group e.g. using matrix factorisation or another collaborative filtering approach.

However, typically users clustered in the same group do not give identical ratings to an item. Rather there is a spread of ratings, and this intra-cluster variability between users can be thought of as adding noise to the ratings. Unfortunately, decision trees can easily make mistakes in the face of such noise. For example, Figure 1(b) shows the measured decision-tree accuracy for Netflix data clustered into 16 groups (see later for more details). It can be seen that the accuracy is as low as 50-60% for a number of groups.

Multi-arm bandits (MABs), and more generally online convex learning, has been the subject of much interest in recent years. However, naive application of standard bandit algorithms to the cold-start task leads to poor performance. If we think of each recommender system item as an arm of a MAB then we run into the difficulty that (i) there are many arms and so learning is slow and (ii) repeated pulls of the same arm tend to be highly correlated[2].

This work was supported by SFI grant 16/IA/4610.

SIGIR '21, July 11–15, 2021, Virtual Event, Canada.

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463033

[1] The system may have some general context regarding the user, e.g. the country/city they are located in, in which case learning is conditioned on this but the fundamental cold start task otherwise remains unchanged.

[2] Measurement studies indicate that when people are repeatedly asked to rate the same item on a scale of 1-5 then if they rate 1 or 5 they tend to consistently stick with that rating although when they rate 3 or 4 they may change their rating back and forth

One remedy is to associate an arm with each group rather than each item. For each group the available items are sorted in descending order of their predicted rating by users in that group. Pulling the arm for a group then corresponds to asking the user to rate the next item from this sorted list, i.e. the unrated item predicted to have the highest rating for members of the group. While this greatly reduces the number of arms in the MAB, as we will see later the learning rate remains very slow. This is because items rated highly by members of one group tend to also be rated highly by members of at least some of the other groups, and so the user ratings for these items do not serve to strongly distinguish between groups and so allow rapid learning.

In this paper we propose a novel cluster-based bandit algorithm that achieves fast learning in cold-start, e.g. for the standard Netflix dataset < 10 items need to be rated in order to reliably distinguish between 16 user groups and < 12 items to reliably distinguish between 32 groups. We show that the group of a user is identified with significantly higher accuracy than with a decision-tree without incurring higher regret i.e the learning performance is fundamentally superior to that of a decision-tree.

2 RELATED WORK

For a recent survey of solutions to the cold-start see [4, 5]. Passive approaches include recommending popular items, use of item-based recommendation (once user starts rating items an item-based approach is used to recommend similar items), transfer learning from another recommender system previously used by a user, and asking new users to rate a fixed list of items. Examples of early work on active learning include IGCN (information gain through clustered neighbors) which uses a decision tree with user clusters as leaves [8] and the ternary decision-tree approach of [7]. More recently, [1] uses representative items i.e. after completing the ratings matrix R, k columns of R are selected, the ratings of the other items are represented as a linear combination of these and during cold start a new user is asked to rate these representative items. This approach is extended to use a decision-tree approach by [9]. In [10] a matrix factorization approach is proposed whereby a decision-tree is trained to map from item ratings to the latent feature vector for a user. Use of multi-arm bandits for cold-start has also received attention. In [6] after completing the ratings matrix R its rows are clustered and the average ratings vector for each cluster is used as a representative user. During cold start an MAB is used to select the average ratings vector to use and the user is asked to rate next highest item in the vector – this is akin to the cluster-bandit algorithm with no exploration phase. In [2] a MAB is used to select between recommender strategies, typically recommending popular items initially for a new user and later switching to a kNN or matrix factorisation model.

3 CLUSTER-BASED BANDIT ALGORITHM

We have a set G of user groups. Each user belongs to one group $g \in G$. We also have a set of items V. Given a new user our task is to quickly learn which group they belong to by asking the user to rate items in V, using the fact that the distribution of item ratings varies depending on the user's group.

3.1 Group Indicator Vector

What we would like to measure is an indicator vector I i.e. a vector for which as we collect more user ratings the element $I(g)$ tends towards 1 when a new user belongs to group g and all other elements $I(h), h \neq g$ tend towards 0. We can obtain such a vector as follows. Start by defining

$$R_n(g,h) = \sum_{i=1}^{n} \alpha_i^n(g,h) \frac{r(v_i) - \mu(h,v_i)}{\mu(g,v_i) - \mu(h,v_i)}$$

where $v_i, i = 1, \ldots, n$ is the sequence of items that the new user has rated, $r(v_i)$ is the new user's rating of item v_i, $\mu(g,v_i)$ is the mean rating of item v_i by users in group g and $\alpha_i^n(g,h)$ is a weighting such that $\sum_{i=1}^{n} \alpha_i^n(g,h) = 1$. An estimate of $\mu(g,v_i)$ is known, e.g. from ratings by existing users of the recommender system. Note that calculation of $R_n(g,h)$ only requires asking a user to rate each item once, although if the recommender system allows it then it is of course possible for sequence $v_i, i = 1, \ldots, n$ to contain duplicate elements i.e. for the same item to be rated multiple times.

To streamline notation suppose the new user is in group 1, we can always relabel the groups so that this holds[3]. Then $R_n(1,h) = \sum_{i=1}^{n} \alpha_i^n(1,h) \left(1 + \frac{r(v_i)-\mu(1,v)}{\mu(1,v_i)-\mu(h,v_i)}\right)$ for $h \neq 1$. Intuitively, the deviations $r(v_i) - \mu(1,v), i = 1, \ldots, n$ tend to fluctuate around 0, as otherwise there would be a consistent offset between the user's ratings and the group ratings in which case the user would better be assigned to a different group. Therefore $R_n(1,h) \to 1$ as $n \to \infty$ for $h \neq 1$. By the same argument, $R_n(g,1) \to 0$ as $n \to \infty$ for $g \neq 1$. Hence,

$$I(g) = \min_{h \in G \setminus \{g\}} R_n(g,h)$$

is an indicator vector of the desired form i.e. $I(1) \to 1$ and $I(g) \to 0, g \neq 1$ as $n \to \infty$. We then estimate the group $\hat{g}$ that the new user belongs to using

$$\hat{g} \in \arg\max_{g \in G} I(g)$$

The key to fast learning is that $I(g)$ converges quickly to a $(0,1)$ vector.

3.2 Fast Convergence

Intuitively, an item v helps to distinguish whether a user belongs to group g rather than group h when (i) the mean rating of v by users in group g is very different from that of users in group h i.e. $(\mu(g,v) - \mu(h,v))^2$ is large, and (ii) when the ratings tend to be consistent/reliable i.e. the variance $\sigma^2(g,v)$ is small. That is, we expect that

$$\Gamma_{g,h}(v) = \frac{(\mu(g,v) - \mu(h,v))^2}{\sigma^2(g,v)}$$

is a measure of the ability of item v to distinguish group g from group h i.e. the larger $\Gamma_{g,h}(v)$ the better item v is at distinguishing group g from group h. For $R_n(g,h)$ to converge quickly we then

between 3 and 4. That is, lumping ratings of 3-4 together in a single bucket these previous studies indicate that a user's rating of an item tends to be consistent.

[3] Of course we need to make sure that the learning algorithm we develop does not depend upon this relabeling.

want to choose the sequence of items $v_i, i = 1, \ldots, n$ that the new user is asked to rate so that $\Sigma_n(g,h) = \sum_{i=1}^{n} \Gamma_{g,h}(v_i)$ is large.

Another way to arrive at the same conclusion is to assume that for users belonging to group g the rating $r(v)$ of item v is i.i.d. subgaussian with mean $\mu(g,v)$ and variance $\sigma^2(g,v)$. Note that any bounded random variable is subgaussian. With this assumption standard concentration inequalities tell us that $Prob(|R_n(1,h) - 1| > \epsilon) < 2e^{-\frac{\epsilon^2}{2}\sum_{i=1}^{n}\Gamma_{1,h}(v_i)}$ and $Prob(|R_n(g,1) - 0| > \epsilon) < 2e^{-\frac{\epsilon^2}{2}\sum_{i=1}^{n}\Gamma_{g,1}(v_i)}$ when we select $\alpha_i^n(g,h) = \frac{\Gamma_{g,h}(v_i)}{\sum_{i=1}^{n}\Gamma_{g,h}(v_i)}$. That is, for $R_n(g,h)$ to converge quickly we want $\sum_{i=1}^{n}\Gamma_{g,h}(v_i)$ to be large.

3.3 Exploration vs Exploitation

The foregoing discussion suggests that for fast learning we want to initially ask the user to rate those items v_i for which $\Gamma_{g,h}(v_i)$ is largest. However, these may not be items that receive high ratings, and so there is a cost to this accelerated learning. We therefore want to limit the duration of the initial exploration phase and quickly switch to an exploitation phase where we recommend items that are predicted to be rated highly by the new user i.e. items for which $\mu(\hat{g},v)$ is large where $\hat{g}$ is the estimated group of the new user. Note that learning can still occur during the exploitation phase provided the items v_i rated have non-zero $\Gamma_{g,h}(v_i)$, but as we will see the learning rate can be much slower than during the exploration phase.

In addition to deciding when to switch from the initial exploration phase (selecting high $\Gamma_{g,h}(v)$ items) to the subsequent exploitation phase (selecting high $\mu(\hat{g},v)$ items), within each phase we need to balance between confirming the new estimate and exploring the other possibilities. It may happen that a new users rating for an item is unusually high or low for their group and so we need to ask them to rate multiple items in order to correct for mistakes and gain confidence in our estimate $\hat{g}$ of the group to which they belong. To do this we employ a form of upper confidence bound (UCB) strategy. Namely, at step n we select the next item v to present to the user to be the unrated item for which learning rate $\Gamma_{\hat{g},h}(v)$ is largest and h is the group for which $\Sigma_n(\hat{g},h)$ is lowest. Selecting h in this way means that we will tend to explore all pairs $(\hat{g},h)$ of groups in such a way that we gain a similar amount of information, as measured by $\Sigma_n(\hat{g},h)$, about each pair.

3.4 Algorithm

Pseudo-code for the exploration phase of the resulting bandit algorithm is given in Algorithm 1. Upon exiting its exploration phase the cluster-based bandit algorithm enters its exploitation phase: the algorithm is the same except that calls to Explore($\hat{g}$) are replaced by calls to Exploit($\hat{g}$), which selects the unrated item for which $\mu(\hat{g},v)$ is largest. Algorithm 1 has three design parameters B, C, D. We use $B = 5$, $C = 0.5$, $D = 3$.

Algorithm 1: Cluster-based Bandit Algorithm

Input: Groups G, mean ratings $\mu(g,v)$ for item v and variances $\sigma(g,v)^2$; parameters $B, C, D \in \mathbb{R}_+$.

Select initial $\hat{g} \in G$, e.g. randomly, and Explore($\hat{g}$)
for $n = 1, 2, 3. \ldots$ **do**
  Define the **candidates** as $M_n = \{g \in G : |\min_h |R_n(g,h)| - 1| \leq C)\}$
  if $M_n \neq \emptyset$ **then**
    $\hat{g} \in \arg\max_{g \in G} I(g)$
    Explore($\hat{g}$)
    if $\Sigma_n(\hat{g}) \geq B$ *or* $(|M_n| = 1$ *and* $n > D \log_2 |G|)$ **then**
      Estimate new user belongs to group $\hat{g}$
      Exit exploration phase for $\hat{g}$
  else
    $(\hat{g},h) \in \text{argmin}_{g,h}\Sigma_n(g,h)$
    Explore($\hat{g}$)

Algorithm 2: Explore($\hat{g}$)

$V_n = \{v_1, \ldots, v_{n-1}\}$ (set of items already rated by user)
$h \in \text{argmin}_h\, \Sigma_n(\hat{g},h) = \sum_{i=1}^{n}\Gamma_{\hat{g},h}(v_i)$
Ask user to rate item $v_n \in \arg\max_{v \notin V_n}\Gamma_{\hat{g},h}(v)$

4 PERFORMANCE EVALUATION

4.1 Evaluation Setup

Datasets. We evaluate the performance of the cluster-based bandit algorithm for cold start on the standard Netflix dataset (480,189 people 17,770movies, 104M ratings from 1–5), the Jester dataset (73,421 people rating 100 jokes, 4.1M ratings from -10–10) and the Goodreads10K dataset (53,424 people rating10,000 books, 5.9M ratings from 1-5).

Clustering Users. We use training data to cluster users into groups and estimate the mean $\mu(g,v)$ and variance $\sigma(g,v)^2$ of the ratings by each group g for item v. We use the BLC matrix-factorization clustering algorithm [3] for this, although other clustering algorithms might also be used. We vary the number of groups/clusters from 4 to 32 and report results for each.

Baseline Algorithms. We compare the performance of the cluster-based bandit algorithm against an optimised CART decision tree. This is a strong baseline, with good performance for cold-start active learning. In addition, as a second baseline we use the cluster-based bandit algorithm but without the initial exploration phase that selects high $\Gamma_{g,h}(v)$ items. This allows the added value of the exploration phase to be measured.

Modelling New Users. We generate the item ratings of a new user from group g by making a single draw from the multivariate Gaussian distribution with mean $\mu(g,v)$ and variance $\sigma(g,v)^2$ for each item equal to that estimated from the training data. This has the advantage that we can easily generate large numbers of new users in a clean, reproducible manner. In addition, we also evaluated performance when drawing ratings from the empirical distribution of ratings for a group (so relaxing the Gaussian assumption) and also by generating user ratings via a water-filling approach i.e. split the data into training and test data, pick a user from the test data and use their ratings, when we need a rating for an item that the user has not rated, pick a second user from the same group who has rated the item and merge the pair of user ratings. We found the performance of these setups to be very similar to simply drawing a new user from a Gaussian distribution.

Performance Metrics. We report the accuracy with which the group of a new user is estimated i.e. the fraction of times the correct group is estimated and also the regret i.e. $\sum_{i=1}^{n} r(v_i) - r(v_i^*)$

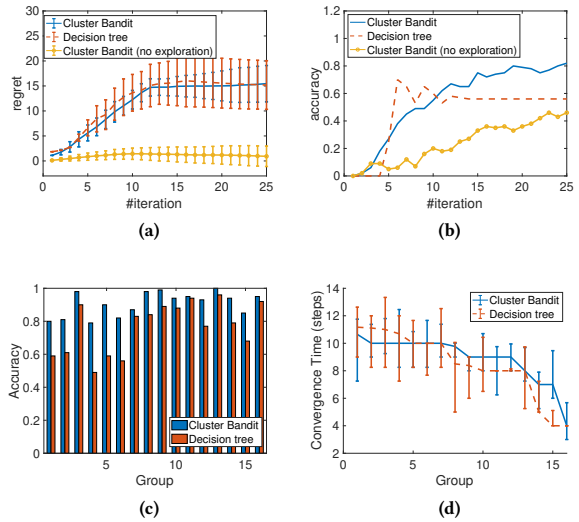

Figure 2: Performance measurements for Netflix dataset (16 groups). Solid lines indicate mean, error bars one std deviation (estimated over 1000 new users in each group).

where v_i^* is the unrated item with highest rating by the new user (so v_1^* is the highest rated item, v_2^* the next highest and so on). We measure the convergence time as the number of items rated before the regret reaches 80% of its final value over a test run. Statistics are calculated over 1000 new users per group.

4.2 Results

Figure 2 shows typical (i.e. representative of the full data) detailed performance measurements for the Netflix dataset with 16 groups. It can be seen from Figure 2(a) that the regret of the cluster-based bandit and decision-tree algorithms are similar while in Figure 2(b) it can be seen that the cluster-based bandit algorithm achieves a significantly higher accuracy than the decision-tree, and from Figure 2(c) this holds across all 16 groups. Figure 2(d) shows the convergence time across all groups. This is similar for both algorithms, consistent with Figure 2(a). The fact that both algorithms have similar regret and convergence time yet the bandit algorithm achieves higher accuracy indicates that the cluster-based bandit algorithm uses a more efficient learning approach.

It can also be seen from Figure 2 that the learning performance of the cluster-based bandit without the initial exploration phase is rather poor: the accuracy is typically substantially worse than for the decision-tree and the convergence time is slow. That is, it is inefficient to use only on the most highly-rated items during cold start, as might be expected.

Table 1 presents summary accuracy and convergence time measurements for the Netflix dataset and also for the Jester and Books datasets. The values shown are averages over the groups (with 1000 new users per group, so e.g. with 16 groups the value shown is the average over 1000×16 users). It can be seen that the cluster-based bandit algorithm consistently achieves substantially higher accuracy than a decision-tree and the cluster-based bandit algorithm without exploration phase. Importantly, this improved accuracy does not come at the cost of slower convergence time. That is, the performance of the cluster-based bandit dominates that of the other algorithms and this data indicates that it should always be used in preference to them if higher accuracy is desired.

Dataset/Algo	Accuracy				Convergence Time			
	#Groups				#Groups			
	4	8	16	32	4	8	16	32
Netflix/DT	0.93	0.88	0.75	0.54	6.09	7.68	9.30	11.52
Netflix/CB$^-$	0.83	0.79	0.65	0.53	-	-	-	-
Netflix/CB	**0.99**	**0.96**	**0.91**	**0.75**	5.15	7.21	9.85	12.03
Jester/DT	0.71	0.55	0.46	0.34	6.22	7.90	8.66	10.15
Jester/CB$^-$	0.84	0.75	0.60	0.52	-	-	-	-
Jester/CB	**0.91**	**0.83**	**0.73**	**0.68**	4.61	6.46	8.92	10.48
Books/DT	0.88	0.69	0.62	0.41	7.80	9.01	9.82	7.37
Books//CB$^-$	0.82	0.72	0.6	0.55	-	-	-	-
Books//CB	**0.96**	**0.87**	**0.84**	**0.67**	6.05	7.02	9.20	7.32

Table 1: Mean accuracy and convergence time for Netflix, Jester and Books datasets vs #groups. Legend: DT=decision-tree, CB=cluster-based bandit algorithm, CB$^-$=cluster-based bandit algorithm without exploration phase. Dash – indicates CB$^-$ failed to converge within 25 items/steps.

5 CONCLUSIONS

In this paper we revisit the cold-start task for new users of a recommender system and propose a novel cluster-based bandit algorithm that achieves fast learning in cold-start, e.g. for the standard Netflix dataset < 10 items need to be rated in order to reliably distinguish between 16 user groups and < 12 items to reliably distinguish between 32 groups. We show that the group of a user is identified with significantly higher accuracy than with a decision-tree without incurring higher regret or longer learning time i.e the learning performance is fundamentally superior to that of a decision-tree.

REFERENCES

[1] Xavier Amatriain, Neal Lathia, Josep M. Pujol, Haewoon Kwak, and Nuria Oliver. 2009. The Wisdom of the Few: A Collaborative Filtering Approach Based on Expert Opinions from the Web. In *Proc SIGIR* (Boston, MA, USA). 532–539. https://doi.org/10.1145/1571941.1572033

[2] Rocio Canamares, Marcos Redondo, and Pablo Castells. 2019. Multi-Armed Recommender System Bandit Ensembles. In *Proc RecSys*. https://doi.org/10.1145/3298689.3346984

[3] Alessandro Checco, Giuseppe Bianchi, and Douglas J. Leith. 2017. BLC: Private Matrix Factorization Recommenders via Automatic Group Learning. *ACM Trans. Priv. Secur.* 20, 2, Article 4 (May 2017), 25 pages. https://doi.org/10.1145/3041760

[4] Mehdi Elahi, Matthias Braunhofer, Tural Gurbanov, and Francesco Ricci. 2018. User Preference Elicitation, Rating Sparsity and Cold Start. *Collaborative Recommendations* (2018), 253–294. https://doi.org/10.1142/9789813275355_0008

[5] Mehdi Elahi, Francesco Ricci, and Neil Rubens. 2016. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review* 20 (2016), 29–50. https://doi.org/10.1016/j.cosrev.2016.05.002

[6] Crícia Z. Felício, Klérisson V.R. Paixão, Celia A.Z. Barcelos, and Philippe Preux. 2017. A Multi-Armed Bandit Model Selection for Cold-Start User Recommendation. In *Proc UMAP* (Bratislava, Slovakia). 32–40. https://doi.org/10.1145/3079628.3079681

[7] Nadav Golbandi, Yehuda Koren, and Ronny Lempel. 2011. Adaptive Bootstrapping of Recommender Systems Using Decision Trees. In *Proc WSDM* (Hong Kong, China). 595–604. https://doi.org/10.1145/1935826.1935910

[8] Al Mamunur Rashid, George Karypis, and John Riedl. 2008. Learning Preferences of New Users in Recommender Systems: An Information Theoretic Approach. *SIGKDD Explor. Newsl.* 10, 2 (Dec. 2008), 90–100. https://doi.org/10.1145/1540276.1540302

[9] Lei Shi, Wayne Xin Zhao, and Yi-Dong Shen. 2017. Local Representative-Based Matrix Factorization for Cold-Start Recommendation. *ACM Trans. Inf. Syst.* 36, 2, Article 22 (Aug. 2017), 28 pages. https://doi.org/10.1145/3108148

[10] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. 2011. Functional Matrix Factorizations for Cold-Start Recommendation. In *Proc SIGIR*. 315–324. https://doi.org/10.1145/2009916.2009961