EE392A - Bayesian Clustering of Contextual Bandits

Shashank Gupta, 160643
IIT Kanpur
gshasha@iitk.ac.in

Suryateja B.V., 160729
IIT Kanpur
suryab@iitk.ac.in

ABSTRACT

Advisors: Prof. Ketan Rajawat (EE), Prof. Piyush Rai (CSE)

Modelling sequential interactions between entities is crucial in various domains such as - social networks, e-commerce, education, and epidemiologyv(disease netwoks). For instance, wireless network selection problem could be seen as a set of sequential interactions between various users and available networks/items (WiFi, LTE, UMTS, LAN). While conventional recommender systems offer a simple solution to these problems, they do not model the temporal aspects that real-world instances demand. Also, in the real world, there are lots of items that we could choose from and potential users to cater to, often in the millions. Hence, it becomes prohibitive to use either conventional or sequential recommender systems to get the task done. Thus, we turn towards clustering of items and users in the bandit setting and leverage the correlations in the item set and user set to reduce the complexity of our model and speed up inference. Bandits, by definition, take up an online learning procedure suitable for modelling sequential interactions. We model clustering of items in a generative fashion by extending the works of DYN-UCB[11] and Online Interactive Collaborative Filtering for MABs [14], and show promising results.

KEYWORDS

contextual bandits, user clustering, item clustering, online recommendation, Thompson sampling, particle filtering

ACM Reference Format:
Shashank Gupta, 160643 and Suryateja B.V., 160729. 2020. EE392A - Bayesian Clustering of Contextual Bandits. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 RELATED WORK

While there is a fairly large volume of work associated to conventional recommender systems, interest in sequential recommender systems has grown only over the past decade. We cover various approaches to sequential prediction relevant to our work in the following paragraphs. For an extensive survey into this area, please refer to [16] or [15].

Multi-arm Bandits: Bandit algorithms [2, 8, 9, 12], centered on explore-exploit trade off, are well suited to the dynamic environment that we are interested in due to their intrinsic online learning nature. [5] is one of the first papers to propose an efficient Thompson sampling-based procedure for online Bayesian Matrix Factorization. [17] uses a time-varying reward mapping function, where the reward weight vectors are modeled using state space equations. Processing large feature vectors is a time-consuming process, so [1] uses bandits to learn a masking over feature vectors. In a similar vein of improving efficiency, many recent works on bandits focus on clustering of users and items. [3] proposes a multi-task learning setting to pool information of items together while doing reward regression. Other works [4, 7, 18] cluster users and items into classes and make the reward vector same for all members in a class - for instance, CoFiBa [7] co-clusters users and items dynamically, but the algorithm is quite involved and assumes that the content universe is known in advance. [4] improves upon CoFiBa by proposing a simple algorithm with context-dependent clustering, and avoiding the graph formulations of CoFiBa.

2 OUR WORK

We present two methods to utilize the correlations among dependent arms in the multi arm bandit setting.

2.1 Dynamic Clustering of Arms

One shortcoming of [11] is that the number of clusters must be set in advance as a hyperparameter. We extend this work with a non-parametric clustering algorithm which automatically infers the most suitable number of clusters (K) based on the data. We incorporate the DP-means algorithm proposed in [6] to the DYN-UCB algorithm. We provide a brief overview of DP-means below.

2.1.1 DP-means: This is a Bayesian non-parametrics based algorithm for hard K-means clustering. The algorithm is inspired by the observation that the EM algorithm for mixture of Gaussians approaches k-means in the limit of variance tending to zero i.e. when the covariance matrices corresponding to the clusters in a GMM are assumed to be σI and $\sigma \rightarrow 0$, the EM steps resemble the k-means steps. This form of argumentation is also known as variance asymptotics, and when applied to the Gibbs sampler for a standard Dirichlet process mixture model, it yields the following non-parametric hard clustering algorithm. The essential modification in the working of algorithm is as follows:

(1) At a given point of time, suppose there are K clusters. Compute the distance of the point x_i from K cluster means $\mu_{1...K}$.
(2) If the smallest of the distances (say d_{ik}) is less than hyperparameter λ, put the point in the cluster k. Else, put the point in a new cluster $K+1$ and recompute the cluster means.

As we can see, this method is very similar to the k-means procedure except for the $lambda$ hyperparameter and assigning to a new

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

cluster. Intuitively, *lambda* is the distance from a potentially new cluster. In fact, when we write a simple objective function for the algorithm, it turns out to be the usual k-means objective with an additional penalty on number of clusters K (like the Akaike Information Criterion). The objective is to minimize over all possible assignment of points to clusters:

$$\sum_{c=1}^{K} \sum_{x \in l_c} \|x - \mu_c\|^2 + \lambda K$$

2.1.2 DYN-UCB. : Here, users are clustered based on their past activity. Given, a user u_t at time t, the algorithm predicts the optimal arm a_t (item) to pick based on the cluster properties of the user. The algorithm is an extension of the Lin-UCB algorithm which considers a linear relationship between reward and item context vector. To enable exploration, the algorithm uses the Upper Confidence Bound (UCB) method. Lin-UCB method considers N bandits for N users and learns a weight vector for each user. Dyn-UCB method somewhat does the same, but the weight vectors of each user are guided by a collaborative step involving clusters. We present the algorithm below.

(1) Initialize user covariance weights (M) and biases (b). Randomly assign the users to K clusters. Compute the weight vector (ie, how the cluster interacts with item context vector to produce reward) as follows:

$$\hat{M}_k = I + \sum_{u \in C_k} (M_u - I)$$

$$\hat{b}_k = \sum_{u \in C_k} b_u$$

$$\hat{w}_k = \hat{M}_k^{-1} \hat{b}_k$$

(2) Draw a user u_t at timestep t. Observe the context for all the arms $x_{t,a}$. Find the arm with the highest UCB given by,

$$\hat{w}_k^T x_{t,a} + \alpha \sqrt{x_{t,a}^T \hat{M}_k^{-1} x_{t,a} \log(t+1)}$$

Note that the term in the square root is essentially the variance of reward ($\hat{w}_k^T x_{t,a}$). Let the arm with maximum UCB be a_t, and the context vector is $x_t = x_{t,a_t}$

(3) Observe the reward r_t by recommending a_t. This is available in the dataset.

(4) The user updates are given below.

$$M_{u_t} = M_{u_t} + x_t x_t^T$$

$$b_{u_t} = b_{u_t} + r_t x_t$$

$$w_{u_t} = M_{u_t}^{-1} b_{u_t}$$

(5) Reassign the user u_t to its closest cluster based on updated weights. Update the cluster M and b parameters.

As is clear, the algorithm takes number of clusters K as an input. We alleviate this dependence by incorporating the DP-means algorithm.

2.2 Particle Learning for Clustering

2.2.1 Online Interactive Collaborative Filtering. This model has been proposed by [14]. Here, we model the user as a document and the items that a user picks as words, making it intuitive to apply the Latent Dirichlet Allocation (LDA) setting, which is a heirarchical mixture of multinomials. Each user is modelled by a sample from a Dirichlet distribution, which indicated mixing proportions over a set of topics of size K. The exact topic is sampled from a Multinomial distribution parametrised by the user vector. The topic vector defines a distribution over the items, indicating how likely a particular item (word, in LDA terminology) is to be chosen by the user (or present in the document). The arms are modelled as samples from a Gaussian. The generative model is presented below. We refer the reader to [14] for a detailed analysis of the generative model.

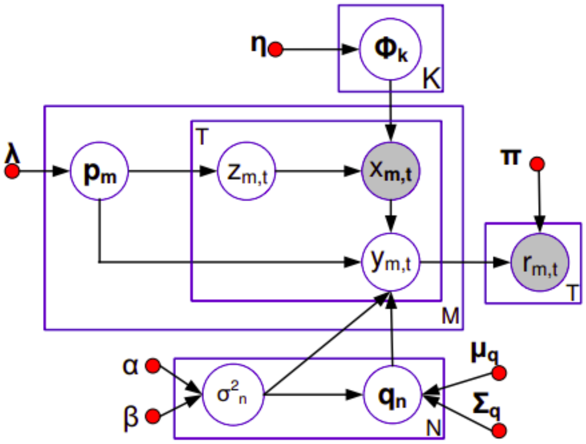

Figure 1: Plate diagram of LDA + Bayesian Linear Regression Model, from [14]

(1) Draw $p_m \sim \text{Dir}(\lambda) \in R^K$ for the user m picked at time t. This represents the user's preference over K latent topics of items.

(2) Draw $\phi_k \sim \text{Dir}(\eta) \in R^N$ for the latent aspect k. This is the item distribution for topic k.

(3) Draw $z_m \sim \text{Mult}(p_m)$ which represents the topic that user m picks.

(4) Draw $x_m \sim \text{Mult}(\phi_{z_m})$ which represents the item that user m picks based on the latent topic chosen.

(5) For each item n, sample a Gaussian mean item vector given by,

$$\sigma_n^2 \sim \text{IG}(\alpha, \beta)$$

$$q_n \sim \text{N}(\mu_q, \sigma_n^2 \Sigma_q)$$

where $\alpha, \beta, \mu_q, \Sigma_q^2$ are hyperparameters.

(6) Finally, the reward is given by, $r_{m,t} \sim N(p_m^T q_n, \sigma_n^2)$.

Inference for this model is done via a particle filtering-based Thompson sampling procedure. Particle sampling uses a set of weighted samples known as particles to estimate the posterior density in an online setting. Each particle is a container with the states of the hyperparamters and latent factors of the model at any given point in time. A set of diverse particles thus provides a much better representation of the posterior than a point estimate when the posterior is not available in closed form. As we are sampling from the posterior, the procedure naturally falls into the probability matching or Thompson sampling family of algorithms.

In order to proceed with the algorithm, we need the following:

- The likelihood of the observed data given the parameters at each time instant. This determines the weight of each particle at any given round. Basically, if a particle is able to better explain the observations, it has a higher chance of being carried over to the next round.
- The posterior over the latent states. In the ICTR model, this is effectively the posterior over the latent factor z_m, or the choice of the topic for each user. As z_m is sampled from a Multinomial with a Dirichlet prior, the conditional posterior is also a Multinomial distribution by conjugacy.
- Appropriate sufficient statistics from the data for the update of the hyperparameters.

For reference, the sufficient statistics for parameters are updated as follows:

$$\begin{aligned}
\Sigma_q' &= (\Sigma_q^{-1} + p_m p_m^T)^{-1} \\
\mu_q' &= \Sigma_q'(\Sigma_q^{-1}\mu_q + p_m r_{m,t}) \\
\alpha' &= \alpha + \frac{1}{2} \\
\beta' &= \beta + \frac{1}{2}(\mu_q^T \Sigma_q^{-1} \mu_q + r_{m,t}^T r_{m,t} - (\mu_q^T \Sigma_q^{-1} \mu_q)') \\
\lambda_k' &= I(z_{m,t} = k) r_{m,t} + \lambda_k \\
\eta_n' &= I(x_{m,t} = n) r_{m,t} + \eta_n
\end{aligned}$$

and the subsequent sampling follows the procedure as presented in the generative process.

2.2.2 ICTR with Gaussian Mixture Models. While the previous model provides a rich framework for modelling dependencies, there are a few key limitations of the same. Firstly, the LDA assumption is quite strong and inflexible. Secondly, the intuition behind the reward computation seems somewhat flawed. The reward is estimated to be the dot product between the user vector and item vector, hence intuitively one would expect higher rewards when the user and item have similar representations. However, the user vector is part of the D-simplex (as a sample from a Dirichlet) whereas the item vector could be from anywhere in $\mathbb{R}^n$, which not only limits the extent of similarity possible, but also implies that the dot product is like an weighing over the dimensions of the feature representation of the item, which has no explicit semantic significance. Thirdly, there is no clustering over the items, which means that certain steps become highly computationally expensive, such as sampling 'number of items'-dimensional vectors for the topic representations.

In order to remedy this, we propose that both the users and items may be sampled from a more flexible Gaussian Mixture Model. For simplicity, we will assume that the item vectors (or context) is known beforehand, and thus model only the user vectors. One may extend the model to the item vectors in an analogous fashion if necessary. The plate model for the generative story is in Figure 2. The generative story itself is as below, once a user has been chosen randomly from the set of users:

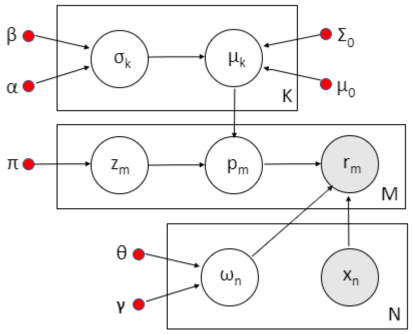

Figure 2: Plate diagram of ICTR-based GMM clustering

(1) Draw $z_m \sim \text{Multinomial}(\pi) \in R^K$ for the user m picked at time t. This represents the index of the cluster to which this user belongs.
(2) Draw $\sigma_k \sim \text{Inverse-Gamma}(\alpha, \beta)$ and $\mu_k \sim \mathcal{N}(\mu_0, \sigma_k \Sigma_n$, for the cluster center to which the user belongs. This is also taken as the user feature vector p_m.
(3) For each item n, sample a reward variance as $\omega_n^2 \sim \text{IG}(\gamma, \theta)$ (shape-scale parametrisation) and compute the reward as $r_m \sim \mathcal{N}(p_m^T x_n, \omega_n^2)$.

The arm with the maximum estimated reward is chosen, and the real reward is received.

Assuming there are B particles composed of the latent factors and hyperparameters, the weights ρ_i for resampling are given by

$$\rho_i \propto \sum_{k=1}^{K} \pi_k \mathcal{N}(r_m | \mu_k^T x_n, \omega_n^2) \mathcal{N}(\mu_k | \mu_0, \sigma_k^2 \Sigma_0)$$

i.e. the likelihood of the reward after marginalising over the latent factor z_m. Note that $\sum_i \rho_i = 1$.

The posterior over the latent factor z_m may be given by $z_m \sim Multinomial(\lambda)$, where

$$\lambda_k = \frac{\pi_k \mathcal{N}(r_m | \mu_k^T x_n, \omega_n^2)}{\sum_k \pi_k \mathcal{N}(r_m | \mu_k^T x_n, \omega_n^2)}$$

i.e. the proportions by which the cluster means match the reward.

The updates for the hyperparameters are similar to that of ICTR, so we can reuse their equations as appropriate.

$$\begin{aligned}
\Sigma_0' &= (\Sigma_0^{-1} + x_n x_n^T)^{-1} \\
\mu_0' &= \Sigma_0'(\Sigma_0^{-1}\mu_0 + x_n r_m) \\
\alpha' &= \alpha + \frac{1}{2} \\
\beta' &= \beta + \frac{1}{2}(\mu_0^T \Sigma_0^{-1} \mu_0 + r_m^2 - (\mu_0^T \Sigma_0^{-1} \mu_0)') \\
\gamma' &= \gamma + \frac{1}{2} \\
\theta' &= \theta + \frac{1}{2}(r_m - p_m^T x_n)^2 \\
\pi_k' &= \frac{K(\pi_k + I(z_m = k) r_m)}{K + r_m}
\end{aligned}$$

The other random variables $\sigma_k^2, \mu_k, \omega_n$ are resampled as per the generative story with the new parameters.

The overall algorithm is as follows:

```
(1) Randomly initialise B particles containing the parameters
(2) For t = 1, ..., T
    (a) User $m$ arrives for recommendation
    (b) Estimate the reward for each item as per the generative story for each particle, and take the final estimated reward as the average of these. Choose the arm with the highest average reward.
    (c) Receive real reward for the chosen arm.
    (d) Compute weights of each particle $\rho_i$ using the real reward.
    (e) Resample the particles according to the weights $\rho_i$.
    (f) For each particle, resample $z_m$ as per the posterior distribution, update the hyperparameters using the sufficient statistics, and resample the other parameters like cluster centres and variances.
```

3 EXPERIMENTS

3.1 Dataset and Setup

We use the publicly available **Delicious** dataset [1] and follow [11] in preprocessing the data for our experiments. The dataset consists of 1867 users, 69226 bookmarks (items), 69226 tags assigned by users to the bookmarks. The task is to predict if a user would bookmark a given website (item) or not. To prepare the item context vector, we use the tags for items and do a TFIDF-style feature representation. We then carry out a PCA to bring the context vector down to 25 dimensions. For training, since we can't present all the 69226 items at once to a user, we select one item that user did pick and 24 items randomly that user didn't pick (Note that even in real-world instances of choosing items, users get to see only 20 items on a page). The algorithm gets a payoff of +1 if it picks the right item for the user of the 25 avaialable items, and a payoff of $\frac{-1}{24}$ if it picks the wrong item.

3.2 DP-Means + Dyn-UCB

We found the choice of λ to be very important. When we choose a poor value of λ, all the users fall into the same cluster. To calculate λ, we followed the farthest-first heuristic. We take a small subset of items as T and take a global mean. We iteratively add the items that are farthest to this mean for K times (K is the approximate number of clusters we expect). The distance of the next farthest item (ie, $(K+1)^{th}$ iteration) is taken as λ. Code for this experiment can be found here [2]

3.3 Particle Learning for Clustering

In Table 1, we show how the total reward for the two algorithms changes as we vary the number of particles. Code for this experiment can be found here [3]

It is seen that for low number of particles, ICTR-1 fares well, whereas for high number of particles, ICTR-2 with GMM over user features works better.

Particles	ICTR-1	ICTR-2
2	56	42
4	67	56
8	74	68
10	81	**92**

Table 1: Total Reward obtained after 1500 iterations for the two algorithm variants

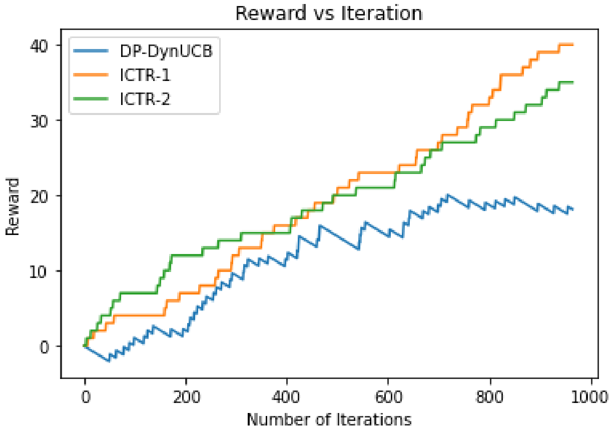

Figure 3: Total Reward vs Number of Iterations

The graph in Figure 2 shows how the total reward varies with number of iterations for the algorithms that we've tried. We ran ICTR-1 and ICTR-2 with 2 particles. From the graph, we see that both the variants of ICTR perform far better than DP-DynUCB. ICTR-1 is better than ICTR-2 since low number of particles are used. With a greater number of particles, we can expect ICTR-2 to fare better.

4 CHALLENGES AND FUTURE WORK

In the case of Dyn-UCB with DP-means, we found that the hyperparameter λ had to be tuned quite carefully. Both variants of the ICTR algorithm were remarkably slow, considering that they are both heavily sampling-based algorithms. This made it difficult to train them for even for a full epoch.

Keeping the above in mind, a few extensions of this work could be:

- Methods like parallel sampling [10] and variational inference (see section 5 of [13]) may be used to speed up inference. This also allow more thorough experimentation to see how ICTR-2 performs in the long run.
- Theoretical bounds for regret would establish the validity of the algorithms, especially considering how much fine-tuning is necessary for all the algorithms discussed. We found that proving this in the general case was quite non-trivial. Interested readers may find [3] instructive.

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

[2] http://tiny.cc/n2k0qz

[3] http://tiny.cc/u3k0qz

- Our experiments involve the case where item representations are fixed in advance. Further experimentation and work on the case where both are learned would be useful, although we expect that the performance would degrade in an online setting due to higher variability.
- Using richer, time-dependent embeddings for the users and items could enhance performance remarkably, as has been observed in various domains such as vision and natural language understanding.

REFERENCES

[1] Djallel Bouneffouf, Irina Rish, Guillermo A Cecchi, and Raphaël Féraud. 2017. Context attentive bandits: Contextual bandit with restricted context. *arXiv preprint arXiv:1705.03821* (2017).

[2] L Elisa Celis, Sayash Kapoor, Farnood Salehi, and Nisheeth Vishnoi. 2019. Controlling polarization in personalization: An algorithmic framework. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM, 160–169.

[3] Aniket Anand Deshmukh, Urun Dogan, and Clay Scott. 2017. Multi-task learning for contextual bandits. In *Advances in neural information processing systems*. 4848–4856.

[4] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Giovanni Zappella, and Evans Etrue. 2017. On context-dependent clustering of bandits. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1253–1262.

[5] Jaya Kawale, Hung H Bui, Branislav Kveton, Long Tran-Thanh, and Sanjay Chawla. 2015. Efficient Thompson Sampling for Online Matrix-Factorization Recommendation. In *Advances in neural information processing systems*. 1297–1305.

[6] Brian Kulis and Michael I Jordan. 2011. Revisiting k-means: New algorithms via Bayesian nonparametrics. *arXiv preprint arXiv:1111.0352* (2011).

[7] Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. 2016. Collaborative filtering bandits. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 539–548.

[8] Jérémie Mary, Romaric Gaudel, and Philippe Preux. 2015. Bandits and recommender systems. In *International Workshop on Machine Learning, Optimization and Big Data*. Springer, 325–336.

[9] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. 2018. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM Conference on Recommender Systems*. ACM, 31–39.

[10] Willie Neiswanger, Chong Wang, and Eric Xing. 2013. Asymptotically exact, embarrassingly parallel MCMC. *arXiv preprint arXiv:1311.4780* (2013).

[11] Trong T Nguyen and Hady W Lauw. 2014. Dynamic clustering of contextual multi-armed bandits. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 1959–1962.

[12] Carlos Riquelme, George Tucker, and Jasper Snoek. 2018. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127* (2018).

[13] Daniel Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen. 2017. A tutorial on thompson sampling. *arXiv preprint arXiv:1707.02038* (2017).

[14] Qing Wang, Chunqiu Zeng, Wubai Zhou, Tao Li, S Sitharama Iyengar, Larisa Shwartz, and Genady Ya Grabarnik. 2018. Online interactive collaborative filtering using multi-armed bandit with dependent arms. *IEEE Transactions on Knowledge and Data Engineering* 31, 8 (2018), 1569–1580.

[15] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).

[16] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet Orgun. 2019. Sequential Recommender Systems: Challenges, Progress and Prospects. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*. International Joint Conferences on Artificial Intelligence Organization, 6332–6338. https://doi.org/10.24963/ijcai.2019/883

[17] Chunqiu Zeng, Qing Wang, Shekoofeh Mokhtari, and Tao Li. 2016. Online context-aware recommendation with time varying multi-armed bandit. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2025–2034.

[18] Li Zhou and Emma Brunskill. 2016. Latent contextual bandits and their application to personalized recommendations for new users. *arXiv preprint arXiv:1604.06743* (2016).