A Scalable Approach for Periodical Personalized Recommendations

Zhen Qin Ish Rishabh John Carnahan

Ticketmaster
Hollywood, CA 91604, USA
{Zhen.Qin, Ish.Rishabh, John.Carnahan}@Ticketmaster.com

ABSTRACT

We develop a highly scalable and effective contextual bandit approach towards periodical personalized recommendations. The online bootstrapping-based technique provides a principled way for UCB-type exploitation-exploration algorithms, while being able to handle arbitrary sized datasets, well suited to learn the ever evolving user preference drift from streaming data, and essentially parameter-free. We further introduce techniques to handle arbitrary sized feature spaces using feature hashing, leverage existing state-of-art machine learning via learning reduction, and increase cache hits by managing bootstrapped models in memory effectively. The resulted model trains on millions of examples and billions of features within minutes on a single personal computer. It shows persistent performance in both offline and online evaluation. We observe around 10% click through rate (CTR) and conversion lift over a collaborative filtering approach in real-world A/B testing across more than 40 million users on the major Ticketmaster email recommendation product.

CCS Concepts

•**Information systems** → *Information systems applications; Data mining; Personalization;*

Keywords

Recommender Systems; Personalization; Online Learning; Contextual Bandits; Scalability; Learning Reductions

1. INTRODUCTION

We are interested in pushing periodical personalized recommendations to users, commonly seen for many e-commerce companies today. In many cases, users are not motivated to visit websites or launch apps to see online recommendations. Periodical "pushing" of relevant products such as weekly recommendation emails, sms, and notifications, remind users of the products for making purchases and further exploration of online content.

RecSys '16, September 15-19, 2016, Boston , MA, USA

DOI: http://dx.doi.org/10.1145/2959100.2959139

There are several key differences between periodical recommendations and the more extensively researched online recommendations. First, collaborative filtering based methods, even its incremental version, tend to produce similar results across time. Though it might be fine for a user to see two sets of the same recommendations across two website visits, pushing two very similar weekly emails can easily bore the user. Second, though contextual bandits that provide novel content and mitigate the cold-start problem via exploitation-exploration has been applied to online recommendations recently, existing algorithms are either computation or parameter intensive. More importantly, they typically employ sampling-based exploration strategies that depend on online user feedback for fast policy improvement. However, for periodical recommendation, we only get feedback as fast as we send out recommendations - such as once a week. Probabilistic exploration could be risky in this scenario. Third, providing refreshing periodic recommendation requires training or updating the model as frequent as sending out the recommendations. Thus a highly scalable approach that takes up-to-date user feedback into consideration, preferably in incremental mode, is necessary.

In this work, we develop an efficient and effective contextual bandit approach towards periodical personalized recommendation that addresses the above-mentioned concerns. The online bootstrapping based technique provides a principled way for upper confidence bound (UCB)-type exploration that is more stable than randomized methods, and it is essentially parameter-free (only the number of bootstrapped models needs to be specified). The online learning framework can naturally feed in user feedback in streaming mode. We further introduce feature hashing, learning reduction, and memory management for bootstrapped policies. The resulted framework can handle arbitrary sized examples and features efficiently, training on billions of features within minutes on a personal computer. We show consistent significant improvement over a productized collaborative filtering approach and other contextual bandits approaches in both offline testing and real-world A/B testing across more than 40 million users on the major Ticketmaster email recommendation product.

2. RELATED WORK

Collaborative filtering (CF) is widely used and researched for recommendation systems. Traditional CF approaches tend to have cold-start problem or generate similar results across time, neither is ideal for periodical recommendations. Context-aware [8] and serendipity-aware (including recom-

mendation diversification) recommendation systems [4] are hot recent research topics.

Contextual bandits is a promising approach to address both problems and it's popular especially for online recommendations [7]. As its name indicates, it considers the context (such as item and user properties) under which recommendations are made, and employs exploitation-exploration, where the exploitation part conducts context-aware prediction based on a learned model, and the exploration part learns about possibly drifting data distributions (e.g. introducing new items or user preference drift). There are two major exploration strategies: randomization-based and UCB-type. Randomization-based exploration samples policy or action to take (such as what to recommend) randomly (e.g. ϵ-greedy) or from some distribution (e.g. Thompson sampling [2]). They are ideal for online recommendation with fast user feedback loop, but are risky for periodical recommendations. UCB-type strategy derives some intervals of prediction and uses the upper bound for exploration [7]. These approaches are more stable, but existing methods are computation intensive. Also, there are usually several parameters to tune for both approaches (such as choosing a prior distribution for Thompson sampling, see [12]). Our model provides a principled UCB-type exploration strategy while being highly efficient and parameter-lite, suitable for large-scale real-world applications.

We apply feature hashing and learning reduction, which have been studied in the machine learning community recently [13] [1]. We show that these techniques dramatically improve efficiency while not sacrificing effectiveness in the personalized recommendation context.

3. ONLINE BOOTSTRAPPING BASED CONTEXTUAL BANDIT

We introduce how to perform stable and efficient contextual bandit using online bootstrapping.

3.1 Online Bootstrap

Bootstrapping is a classical way of generating prediction intervals. Traditional batch bootstrapping takes a dataset of N examples and generates B new datasets, each of size N, by sampling with replacement (for every example in a new dataset, each original example has a probability of $\frac{1}{N}$ to be selected). Then B models are trained on these B datasets and used for prediction. These B predictions naturally provide a prediction interval, and its upper bound could be used for UCB-type bandit algorithms.

It's clear that batch bootstrapping for large N is impractical, and impossible for streaming data with unknown N. However, it's noted [9] [10] that each example i will appear Z_i times in the bootstrapped sample and Z_i is a random variable. More precisely, Z_i is distributed as a $Binom(N, \frac{1}{N})$, because during re-sampling the i-th example will have N chances to be picked, each with probability $\frac{1}{N}$. Importantly, this $Binom(N, \frac{1}{N})$ converges to a Poisson with rate 1 even for modest N (see Fig. 1).

These are interesting observations for real-world big data problems. Since we can just sample a weight from Poisson with rate 1 for each data *independently*, regardless of N, this naturally leads to an online bootstrapping framework that can handle arbitrary sized dataset in streaming mode.

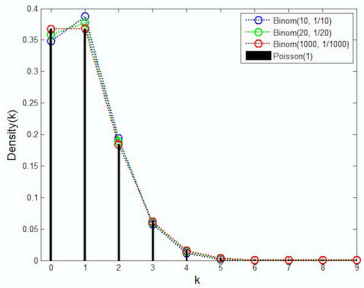

Figure 1: Binomial converges to Poisson for moderate sample sizes.

3.2 Learning and Prediction

The resulted learning and prediction algorithms are simple, see Fig. 2. Note only the number of bootstrapped models (B) needs to be specified.

```
Input: example E, number of bootstrapped model B

    Training:                  Prediction:
    for b = 1 to B             for b = 1 to B
    do                         do
      Z ~ Poisson(1)             p_b = predict(E, b)
      learn(E, Z, b)           done
    done                       return max(p)
```

Figure 2: Online bootstrapping for one example.

Note in both training and testing, each example is parsed online only once for all bootstrapping rounds. Only the importance weight is drawn repetitively. This largely mitigates example parsing (including I/O) overhead.

3.3 Learning Reductions

A closer look at Fig. 2 indicates that the proposed contextual bandit approach is built on some "learn" function. This leads to the idea of "learning reductions": we can take advantage of any existing effective and efficient learning algorithm or package that is (1) online and (2) weight-aware, as a black box. These criterions have been largely addressed by machine learning researchers recently. We use a logistic regression model (as we will focus on predicting the probability of clicks/purchases) described in [11], but the proposed framework is of more general use.

3.4 Feature Hashing

In real-world applications, the dimension of the context (feature) space can be quite large, and the space also evolves online. For example, it is common to use the item ID as a feature, thus learning from streaming data with new items leads to potentially unbounded number of features.

We apply feature hashing [13] to address this concern. The idea is to use a hash function to project the original feature space to a d-dimensional one (Fig. 3). We hash all features into the same space controlled by memory budget. Collisions are bounded to occur, but it is usually not a major concern in practice, as analyzed in [3] [13] and validated by our experiments

3.5 Memory Management

After introducing the hashing trick, size of each bootstrapped model is bounded, allowing for more efficient mem-

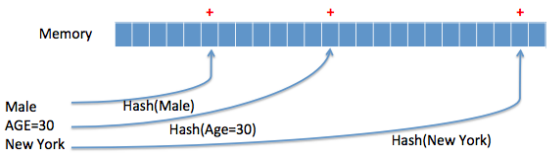

Figure 3: Illustration of hashing features to memory. In practice, features such as age are discretized into bins before hashing.

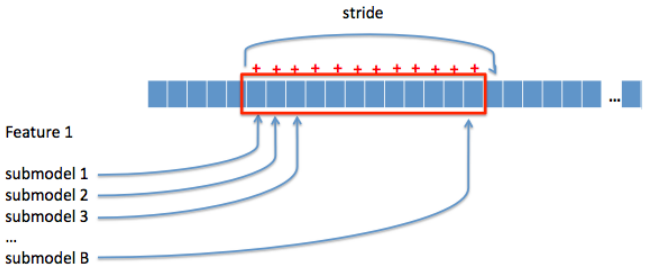

Figure 4: Illustration of memory management of one feature for different submodels. Each feature possesses a "stride" in memory.

ory management: we can safely pre-allocate available memory to hold all bootstrapping submodels.

As it is typical for online learning methods to conduct per-feature weight updates, weights for the same feature of different bootstrapping submodels are close to each other in memory (Fig. 4). These alleviate localize memory accesses and take advantage of caching.

3.6 Summary

We propose to use online bootstrapping as a heuristic for UCB-type contextual bandits. By applying feature hashing and learning reduction techniques, our algorithm is essentially parameter-free, handles arbitrary sized dataset, and takes advantage of existing models/implementations.

4. EXPERIMENTS

We first evaluate the efficiency of the proposed model, the running time of which grows sublinearly with the number of bootstrapping models. Then we move forward to both offline evaluation and real-world A/B testing using large-scale data. All of our experiments are run on a single personal computer (2.2 GHz, 16GB). Extending our framework to multiple machines is straightforward, but does not appear neccessary at our data scale due to its efficiency and streaming nature. We will open-source our implementation.

4.1 Dataset

We use a publicly available RCV1 dataset [6], a popular archive of categorized newswire stories for text categorization. It contains 781,265 examples and 80 features (using bag of words representation) per example on average. This dataset is mainly for running time evaluation.

We also use a real-world dataset that is gathered from Ticketmaster's weekly email recommendation product. This product serves over 40 million users in the US and sends out an email containing a list of around 20 recommended items (e.g. live events) to each subscribed user once a week. We log which items have been sent out to and clicked/purchased by each user for model training and offline evaluation.

Our contexts contain user's info (e.g. gender, age, race, geo, click/purchase history, etc.), item's info (e.g. category, price, rating, description, etc.). We also generate conjunction features that are the cartesian product of user info and item info. In our case, each user-item pair generates a context of around 100 dimensions. The actual context space is much larger because most features are categorical (e.g. gender=male and gender=female are different features).

4.2 Comparing methods

We compare with some popular approaches:

- TS(q_0): Thompson sampling randomly draws model coefficients from a posterior distribution and selects the item with largest predicted reward. The priori distribution is $N(0, q_0^{-1}I)$ [2].
- CF: Our productized collaborative filtering method. It is a variant of item-item based CF approach using implicit feedback [5].
- OBS: Online bootstrapping based contextual bandit, method proposed in this paper.
- LR: Contextual logistic regression, same as OBS with $B = 1$. This approach does not permit exploration.
- ϵ-greedy(ϵ): Randomly selects an item with probability ϵ and selects the item with the largest predicted reward (based on LR) with probability $1 - \epsilon$.
- Random: randomly selects an item, same as ϵ-greedy(1.0).

4.3 Offline Evaluation

4.3.1 *Running Time*

Though we are training B bootstrapping models in parallel, running time is expected to be faster than growing linearly with B, since (1) each example is parsed only once, (2) efficient memory management and (3) setting up overhead is only once. We evaluate on the RCV1 dataset and compare with estimated batch bootstrapping running time as $t \times B$ where t is time for $B = 1$ (single model). We believe this estimation is optimistic, as it is not even clear how to do batch resampling on large datasets.

We show results in Fig. 5. It shows sub-linear running time growth and no significant difference with different number of bootstrapping rounds. These can be explained as a lot of computation power is spent on example parsing (including disk I/O) and environment setting-up, while our approach can avoid repetitive example parsing and setting up environments for each submodel. Thus our strategy is particularly helpful for a dataset with many examples.

Note the absolute running time (in tens of seconds) on this over half-million sized dataset results from reducing online bootstrapping to an existing highly efficient implementation of [11]. We also observed better prediction performance by ensembles of predictions from bootstrapped models (i.e. bagging), but we omit details due to space constraints.

4.3.2 *Weekly Recommendation Dataset*

We conduct offline evaluation using our weekly email recommendation data. We evaluate precision@N on one week's data, using previous eight weeks' data for training. Training takes about 10 minutes for this million-sized dataset. We test on several weeks' data and saw consistent behavior so we only show one week's result in this paper. Also for

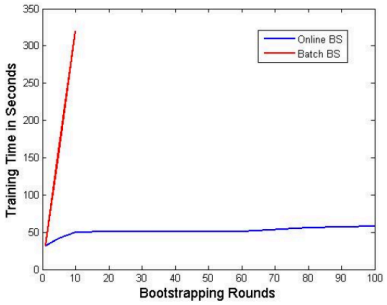

Figure 5: Speed comparison. Lower bound is calculated (some big numbers are not shown) for batch bootstrapping as pure training time, excluding time to generate samples with repetition.

models with randomization, the variance is neglectable so we only report the mean. For all comparing models with tunable parameters, we try different configurations and report the best result. We use $B = 5$ but saw robust results with a wide selection of B. The results are shown in Tbl. 1

Prec. @N	ϵ-greedy (0.01)	Rand	CF	TS (0.1)	LR	OBS
N=1	0.294	0.051	0.258	0.231	0.290	**0.312**
N=5	0.692	0.252	0.641	0.623	0.691	**0.709**
N=10	0.876	0.501	0.806	0.791	0.875	**0.885**

Table 1: Offline evaluation on the recommendation dataset.

We are particularly interested in Precision@1, as it is what an user sees immediately after opening the email. The proposed approach performs the best and improves upon CF by comparatively 20%. It should not be surprising that LR outperforms CF: LR still falls into our online learning framework that considers context and recent user feedback. Unguided exploration strategy such as ϵ-greedy only marginally improves upon LR in its best case, indicating the need for a guided exploration principle such as the one proposed in the paper.

We also tested OBS without using last 4 weeks' feedback, resulting in 0.265 for Precision@1, only marginally improving upon CF. This shows the necessity of a streaming framework that considers the evolving user preference drift.

4.4 Real-world A/B Testing

We have conducted real-world A/B testing on all Ticketmaster (over 40 million) subscribed users, comparing the proposed approach (OBS) with the existing CF method. We do not test other methods due to the possible risks and engineering overhead. To avoid leaking business-sensitive information, we report relative CTR and conversion lift in Tbl. 2. The differences passed z-test at 99% confidence level.

CTR Delivered	CTR Opened	Conversion Delivered	Conversion Opened
+9.56%	+8.51%	+10.70%	+9.64%

Table 2: Performance of OBS over CF in real-world A/B testing.

We attribute the better results on emails delivered over emails opened to the email previews being more relevant. The preview shows some info about top-ranked items and more relevant top results encourage more opens.

4.5 Model Transfer

We also show the model's generalization capability on a different product. The live events we deal with have timeliness, so we tested on a "last minute" email product that only recommend live events available in the near future. We directly use the model trained in the previous section and compare with a CF-based approach. The test is only rolled out among 1 million users so there is more variance. Results in Tbl. 3 show that, directly applying the learned model results in promising lifts, regardless of the data distribution drift between the two products.

CTR Delivered	CTR Opened	Conversion Delivered	Conversion Opened
+5.01%	+5.95%	+2.89%	+3.84%

Table 3: Performance of OBS over CF in another product.

5. CONLUSION

We provide a new contextual bandits strategy for personalized recommendation. By using online learning, feature hashing and learning reductions, the model can handle unbounded streaming data and take advantage of existing state-of-art. Its efficiency and effectiveness have been validated in both large-scale offline and online evaluations.

6. REFERENCES

[1] N. Ailon and M. Mohri. An efficient reduction of ranking to classification. *COLT*, 2008.
[2] O. Chapelle and L. Li. An empirical evaluation of thompson sampling. *NIPS*, 2011.
[3] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *ACM TIST*, 2015.
[4] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. *RecSys*, 2010.
[5] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. *ICDM*, 2008.
[6] D. D. Lewis, Y. Yang, T. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *IMLR*, 2004.
[7] L. Li, W. Chu, J. Langford, and R. E. Schapire. A contextual-bandit approach to personalized news article recommendation. *WWW*, 2010.
[8] A. Q. Macedo, L. B. Marinho, and R. L. T. Santos. Context-aware event recommendation in event-based social networks. *RecSys*, 2015.
[9] N. C. Oza and S. Russell. Online bagging and boosting. *AIStats*, 2001.
[10] Z. Qin, V. Petricek, N. Karampatziakis, L. Li, and J. Langford. Efficient online bootstrapping for large scale learning. *arXiv preprint arXiv:1312.5021*, 2013.
[11] S. Ross, P. Mineiro, and J. Langford. Normalized online learning. *UAI*, 2013.
[12] L. Tang, Y. Jiang, L. Li, C. Zeng, and T. Li. Personalized recommendation via parameter-free contextual bandits. *SIGIR*, 2015.
[13] K. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. Smola. Feature hashing for large scale multitask learning. *ICML*, 2009.