Sketching Dynamic User-Item Interactions for Online Item Recommendation

Takuya Kitazawa
Graduate School of Information Science and Technology
The University of Tokyo, Japan
k.takuti@gmail.com

ABSTRACT

Online recommendation techniques such as incremental matrix factorization have been widely studied to capture users' preferences from real-world dynamic user-item interactions. These recommenders generally take a model-based approach which relies on stochastic gradient descent optimization, but it sometimes shows difficulties on hyperparameter optimization and scalability. In order to work around this difficulty, the author considers a completely different approach based on matrix sketching, which is more mathematically tractable and flexible than the existing techniques. More specifically, this paper proposes a new online item recommender inspired by a matrix-sketching-based streaming anomaly detection framework. We discuss its fruitful properties with particular emphasis on the similarities between item recommendation and anomaly detection. Experimental results demonstrate robustness and flexibility of the proposed framework.

Keywords

Matrix sketching; online learning; implicit feedback

1. INTRODUCTION

On the real-world applications such as online ad and e-commerce, user-item interactions (e.g. click, purchase, rate) are valuable to capture users' preferences, and recommender systems need to tackle an item ranking problem based on such users' implicit feedback. Most importantly, recent studies showed that real-world item recommenders commonly suffer from cold-start persistency, and online updating of a recommendation engine is an effective way to overcome the problem [3]. However, the state-of-the-art techniques are not always practical for the following reasons.

First, most online item recommenders take a model-based approach which can be optimized by stochastic gradient descent, and they usually face the lack of effective ways to find proper hyperparameters in an incremental fashion. To give an example of state-of-the-art incremental matrix factorization (iMF) [6] and incremental factorization machines (iFMs) [3], even if we narrowly find the optima in an initial batch training step, the best values may show poor compatibility with further unforeseen events. Hence, streaming algorithms with less, simpler hyperparameters are preferable in reality.

Moreover, scores computed by the classical item recommenders are unnatural. Since modern recommenders stress "post-processing" such as generating explanation and incorporating into a hybrid recommender, the scores have to be easily interpretable. However, binary prediction $\hat{y}$ obtained from the existing techniques can be less than 0 or greater than 1, and incremental MF and FMs compute item scores by an inelegant heuristic function $|1 - \hat{y}|$.

As the example above shows, the current state-of-the-arts are in some sense inflexible. Therefore, this paper turns to a different direction and seeks a more mathematically tractable approach to propose a robust, flexible recommender that can fully take advantage of intuitive concepts and theoretical guarantees. To the end, we apply an efficient matrix sketching technique to a stream of user-item interactions, and recommend items based on the orthogonal projection. We also discuss how item recommenders resemble anomaly detectors, because our algorithm relies on a streaming anomaly detection framework. The proposed framework is a whole new paradigm of looking at item recommendation.

2. PROBLEM DESCRIPTION

Our goal is to develop a recommender system which works efficiently and effectively in a realistic streaming environment as depicted in Fig. 1.

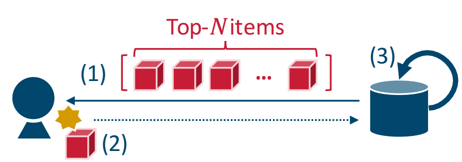

Figure 1: Online item recommender: (1) recommend a top-N list to a user, (2) interact with an item (e.g. click, buy, rate), and (3) update parameters based on the interaction.

More specifically, satisfying the following two properties is a minimum requirement for online item recommenders:

1. **Online-updating.** Algorithms have to be incremental.
2. **Implicit feedback.** Explicit rating is not considered.

In addition, this paper requires the proposed item recommender to hold the following properties:

CHIIR '17, March 07-11, 2017, Oslo, Norway

DOI: http://dx.doi.org/10.1145/3020165.3022152

3. **Feature-expressiveness.** Due to users' rare activity and dynamically changing user/item properties, modern recommenders have to use auxiliary information (e.g. time, demographics) as feature representations.
4. **Usability.** Hyperparameters and item scores should be simple; easy-to-choose integer hyperparameters and easily understandable scores are desired to improve feasibility of a recommender.
5. **Parallelizability.** Modern applications usually stand on distributed systems, so parallelizing and merging recommendation models are necessary. Furthermore, since a purely online algorithm may not be needed in reality, a recommender needs to flexibly support both of purely online and mini-batch update based on merging.

As the author showed in Sec. 1, the state-of-the-arts [1, 3, 6] are infeasible due to the lack of the above properties.

3. ONLINE ITEM RECOMMENDATION USING MATRIX SKETCHING

This section describes the proposed online item recommendation framework. Sec. 3.1 first states the key idea that anomaly detection and item recommendation can be discussed interchangeably, and Sec. 3.2 introduces a state-of-the-art online anomaly detection technique. Finally, the details of the proposal are clarified in Sec. 3.3.

3.1 Similarities between Anomaly Detection and Online Item Recommendation

This paper argues that **anomaly detectors can be seen as an online item recommender**, because both of them deal with a very similar problem. That is, anomaly detectors and item recommenders respectively find dissimilar and similar samples from captured patterns of the past events. Additionally, since both problems tackle dynamic data, their research fields share common challenges. In spite of the obvious similarity, this view has never been explicitly pointed out in the literature to the best of the author's knowledge.

It should be noted that anomaly detectors are generally categorized into *outlier* and *change-point* detectors. Both types of anomalies are strongly related to online item recommendation. In terms of outliers, finding suddenly-observed unforeseen patterns is a similar challenge to making reasonable recommendation for new users and items. At the same time, change-points can be interpreted as a trend in data streams; working around with the shifting (i.e., dynamic) patterns is a crucial challenge for online item recommenders.

Importantly, it is clear that all properties of online item recommenders described in Section 2 are also necessary for anomaly detectors as follows:

1. **Online-updating.** Since anomaly detection directly affects to system reliability, anomalies should be detected as fast as possible by using up-to-date models.
2. **Implicit feedback.** In a context of anomaly detection, a term "feedback" indicates single data point in a system, and the task is binary prediction to infer whether a data point is anomaly or not. This setting is exactly same as implicit-feedback-based recommendation which makes prediction for a unary target value.
3. **Feature-expressiveness.** Anomalies are usually detected by a combination of various factors monitored in a target system. For example, log files provide many different features such as timestamp and the number of requests.
4. **Usability.** Complex hyperparameters and confusing scores are not preferred because explanation of detectors' output is important in practice to make decision.
5. **Parallelizability.** Modern huge back-end systems are highly distributed, and data points may come from multiple data streams. Thus, the anomaly detectors should be parallelizable.

Therefore, the author connects the field of anomaly detection with online item recommendation.

3.2 Streaming Anomaly Detection using Matrix Sketching

Sketching is a technique to efficiently approximate particular properties of massive data, with strong theoretical guarantees. Recently, Liberty [4] proposed a matrix sketching technique, frequent directions (FD), and it had been proved that a huge matrix $A \in \mathbb{R}^{m \times n}$ ($m \ll n$) can be approximated by a tiny sketched matrix $B \in \mathbb{R}^{m \times \ell}$ ($\ell < m$) as $AA^{\mathrm{T}} \approx BB^{\mathrm{T}}$. In particular, matrix A and B satisfy the following property:

$$AA^{\mathrm{T}} - BB^{\mathrm{T}} \succeq 0 \text{ and } ||AA^{\mathrm{T}} - BB^{\mathrm{T}}|| \leq 2||A||_F^2/\ell,$$

where $\succeq 0$ indicates that a matrix is positive semidefinite. Thanks to the above property, orthonormal bases of A can be efficiently approximated by singular value decomposition (SVD) of the much smaller matrix B.

Huang and Kasiviswanathan [2] took advantage of FD's fast, iterative sketching algorithm and proposed a streaming anomaly detection framework. A key idea of their approach is to sketch a series of "normal" vectors to a tiny matrix B. By the orthogonal projection of a newly observed unit vector $\mathbf{x}$ onto the orthonormal bases U_ℓ obtained from $\mathrm{SVD}_\ell(B)$, an anomaly score can be defined as $a = ||(\mathbb{I} - U_\ell U_\ell^{\mathrm{T}})\mathbf{x}||$; larger scores indicate that the vector is far from a "normal" vector space.

Note that, they actually extended discussion about FD to randomized matrix sketching for computational efficiency, but we now focus on the original FD-based matrix sketching for simplicity. It can be easily extended to the randomized variant in the future.

3.3 Sketching User-Item Interactions

This paper proposes a novel online item recommender inspired by the matrix-sketching-based anomaly detector. In short, for a set of users U and items I, recommendation and update are done as Alg. 1 and Alg. 2, respectively. Here, m is the number of dimensions of a feature vector which represents an observed user-item interaction, and sketch size ℓ is a small integer. B is initialized by $O \in \mathbb{R}^{m \times \ell}$. Meanwhile, Fig. 2 illustrates the whole procedure.

First, Alg. 1 follows a concept of anomaly scores introduced in [2]. When we have the orthonormal bases U_ℓ derived from the past user-item interactions, the length of the orthogonal projection of a vector $\mathbf{x}'_{ui}$ onto U_ℓ (i.e. item score) represents how much the vector is similar to the past events; items which take smaller scores are more likely to be recommended.

Next, Alg. 2 updates a sketched matrix B as described in [4]. Since ℓ is a small integer, $\mathrm{SVD}_\ell(B)$ can be efficiently computed, and the procedure works well as an online algorithm. Tracking U_ℓ is also important to compute scores in the recommendation step.

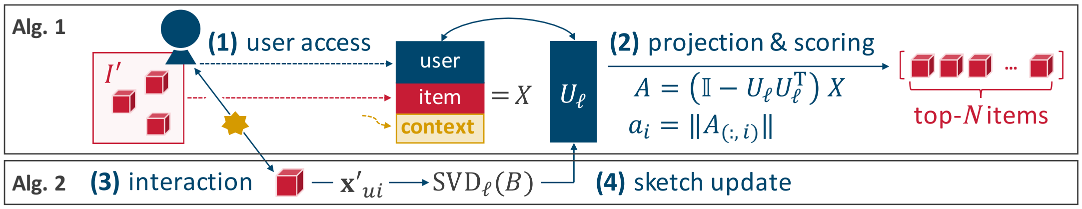

Figure 2: Overview of our new online item recommendation framework. In Alg. 1, (1) a recommender first creates a matrix X for the user-item pairs, and (2) top-N recommendation is launched based on the orthogonal projection. If (3) a user interacts with an item, Alg. 2 constructs a vector $\mathbf{x}'_{ui}$ and (4) updates the orthonormal bases U_ℓ by matrix sketching.

Algorithm 1 Recommend to a user $u \in U$ ((1) in Fig. 1)

Input: $U_\ell \in \mathbb{R}^{m \times \ell}$, Set of target items $I' \subseteq I$, Recommendation size N

Create feature vectors $\mathbf{x}_{ui} \in \mathbb{R}^m$ for every $i \in I'$
Normalize $\mathbf{x}'_{ui} \leftarrow \mathbf{x}_{ui} / \|\mathbf{x}_{ui}\|$
Create a matrix $X \leftarrow \left[\mathbf{x}'_{u1}, \cdots, \mathbf{x}'_{ui}, \cdots, \mathbf{x}'_{u|I'|}\right]$
$A \leftarrow \left(\mathbb{I} - U_\ell U_\ell^{\mathrm{T}}\right) X \in \mathbb{R}^{m \times |I'|}$
Compute scores $a_i \leftarrow \|A_{(:,i)}\|$ for each column of A
Sort items $i \in I'$ by a_i in an **ascending** order
return Top-N items

Algorithm 2 Update for an interaction between $u \in U$ and $i \in I$ ((3) in Fig. 1)

Input: $B \in \mathbb{R}^{m \times \ell}$

Create $\mathbf{x}_{ui} \in \mathbb{R}^m$ for an observed (u, i) interaction
Normalize $\mathbf{x}'_{ui} \leftarrow \mathbf{x}_{ui} / \|\mathbf{x}_{ui}\|$
$j \leftarrow$ index of the leftmost all-zero column in B
$B_{(:,j)} \leftarrow \mathbf{x}'_{ui}$
$[U_\ell, \Sigma_\ell, V_\ell] \leftarrow \mathrm{SVD}_\ell(B)$
$\hat{\Sigma}_\ell \leftarrow \mathrm{diag}\left(\sqrt{\sigma_1^2 - \sigma_\ell^2}, \sqrt{\sigma_2^2 - \sigma_\ell^2}, \ldots, \sqrt{\sigma_{\ell-1}^2 - \sigma_\ell^2}, 0\right)$
$B \leftarrow U_\ell \hat{\Sigma}_\ell$
return B, U_ℓ

Matrix sketching holds attractive properties [2, 4], and hence our online item recommender satisfies **usability**. Clearly, item scores are always in a [0.0, 1.0] range as shown in Alg. 1. In addition, we only have one simple hyperparameter ℓ thanks to discussions about error bounds on FD; setting a very tiny integer to ℓ is sufficient. For instance, Huang and Kasiviswanathan [2] simply used $\ell = \sqrt{m}$ as a reasonable choice.

In terms of **parallelizability**, Liberty [4] showed that, for distributed sketch matrices B_1 and B_2, they are successfully mergeable by just sketching $B' = [B_1, B_2] \in \mathbb{R}^{m \times 2\ell}$ to $B \in \mathbb{R}^{m \times \ell}$. This fact also suggests that FD works well both on purely online and mini-batch fashion; more than one vectors can be sketched at once.

It should be noticed that a user-item interaction is represented by a feature vector $\mathbf{x}_{ui}$ in our framework. The simplest representation is concatenating quantitative variables and one-hot-encoded categorical variables just like FMs [5], and pairwise feature representation as demonstrated in [1] is also available. In either case, the vectors can represent not only user (item) features but also contextual variables which describe a set of circumstances.

4. EVALUATION

4.1 Experimental Setup

We consider three types of time-stamped datasets, Click, ML100k and ML1M. Click is a synthetic dataset which contains randomly generated users' impressions for five ad variants, following to a rule-based procedure introduced in [1]. ML100k is binalized MovieLens 100k data; only 5-starred rating events were extracted like [6]. ML1M is also binalized version of the MovieLens 1M data which behaves similarly to ML100k, but larger than that. Each time-stamped dataset was separated into three parts: 20% for batch training, 10% for batch validation and 70% for incremental evaluation. Details of this evaluation procedure are explained in [3, 6].

Similarly to [3], $\mathbf{x}_{ui}$ was flexibly designed by concatenating as much as possible user, item and contextual features instead of the user-item 2-dimensional data. As a result, $m = 55$ for Click, and $m = 73$ for ML100k and ML1M. We also examine the following baselines:

- **Popularity** is a non-personalized recommender which always recommends the most popular items as a reference.
- **iMF**[1] [6] is an incremental variant of MF. This method only uses the user-item 2-dimensional data.
- **iFMs**[2] [3] are an incremental version of FMs. One-hot-encoded user/item IDs are incorporated into $\mathbf{x}_{ui}$.

4.2 Results and Discussions

The author first examined whether our framework **Sketch** outperforms state-of-the-art personalized recommenders, iMF and iFMs. We set $\ell = 1$ for Click and $\ell = \sqrt{m} = 8$ for ML100k and ML1M, and compared the results on recall@N and mean percentile rank (MPR) as shown in Table 1. On the Click data, Sketch showed the best accuracy both on recall@1 and MPR. In case of the MovieLens datasets, while recall@10 was worse than iFMs on the smaller 100k data, Sketch became much better than iFMs on the larger 1M data. This results indicate that complex hyperparameters of iFMs were rapidly unfitted as a result of online update, and Sketch demonstrated more robust results as we expected. It should be noted that, even though the accuracy of iMF was consistently poor, its running time was much faster than the feature-based recommenders. Here is a trade-off between accuracy and efficiency.

Remarkably, the popularity-based recommender was the best on the MovieLens data. Thus, using item scores as a part of personalized/non-personalized hybrid recommenders

[1] $\lambda = 0.01$, $\eta = 0.002$, $k = 40$ for ML100k and ML1M, and $\lambda = 0.01$, $\eta = 0.0003$, $k = 2$ for Click.

[2] $\lambda_0 = 2.0$, $\lambda_{\mathbf{w}} = 8.0$, $\lambda_{V_k} = 16.0$, $\eta = 0.004$, $k = 40$ for ML100k and ML1M, and $\lambda_0 = \lambda_{\mathbf{w}} = \lambda_{V_k} = 0.01$, $\eta = 0.00006$, $k = 2$ for Click

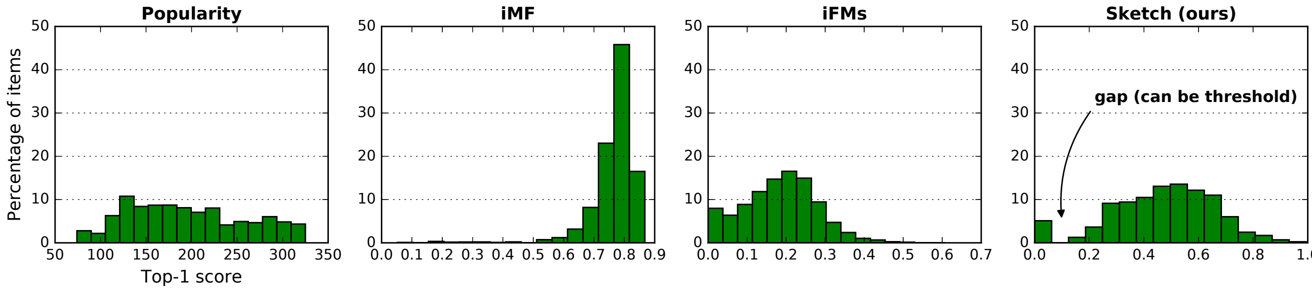

Figure 3: Distributions of top-1 item scores for each method on ML100k. Top-N largest-scored items are recommended in Popularity, and the other methods recommend top-N smallest-scored items.

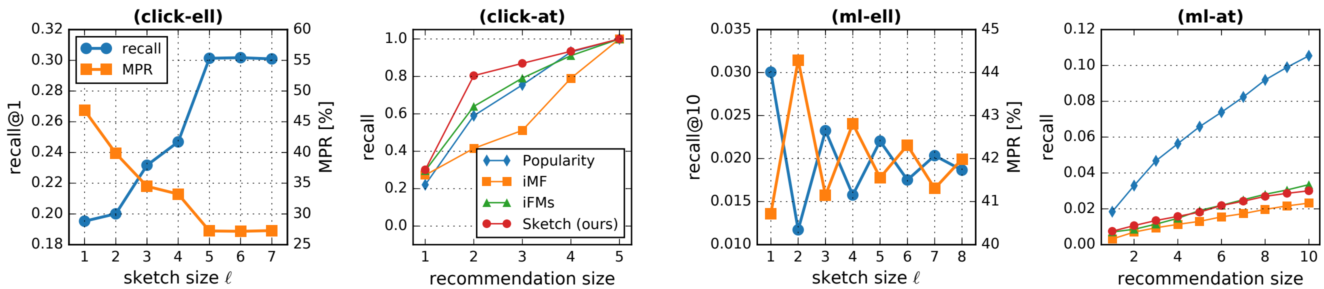

Figure 4: Relationship between accuracy and hyperparameters. Higher recall@N and lower MPR are better on the y-axis. **(click-ell)**: Different sketch size from 1 to $\sqrt{m} = 7$ on Click. **(click-at)**: Different recommendation size on Click. **(ml-ell)**: Different sketch size from 1 to $\sqrt{m} = 8$ on ML100k. **(ml-at)**: Different recommendation size on ML100k.

Table 1: Overall accuracy and running time. The best personalized recommender's accuracy is written in **bold**.

Method	recall@N	MPR [%]	Avg. time [sec.]	
			recommend	update
Click (@1)				
Popular	0.22089	37.62505	–	–
iMF	0.27251	50.27011	0.00002	0.00003
iFMs	0.29612	34.07363	0.00321	0.00026
Sketch	**0.30092**	**27.29092**	0.00190	0.00066
ML100k (@10)				
Popular	0.10538	17.10795	–	–
iMF	0.02318	40.91159	0.00017	0.00003
iFMs	**0.03349**	**27.51833**	0.02388	0.00142
Sketch	0.03005	40.72182	0.00650	0.00039
ML1M (@10)				
Popular	0.08454	12.14989	–	–
iMF	0.01249	42.18032	0.00061	0.00003
iFMs	0.01379	**27.84364**	0.30973	0.00605
Sketch	**0.02451**	37.67139	0.01626	0.00044

might be a requirement for getting more reasonable results. In that sense, we are now interested in distribution of item scores as illustrated in Fig. 3. Unlike the other methods, Sketch showed a gap between 0.0 and 0.2. We can infer that the items placed where score < 0.1 are particularly close to a space spanned by U_ℓ, and thresholding the scores leads a hybrid recommender. On the other hand, unnatural scores of iMF and iFMs are not suggestive.

Finally, by using Click and ML100k, Fig. 4 demonstrates accuracy of the various ℓ and cut-off @N to discuss limitations of the proposal. One thing we can observe is that larger ℓ does not always lead higher accuracy as shown in **(ml-ell)**. Even though attractive properties of the sketching technique help us to find promising ℓ, extra effort might be necessary to achieve the best accuracy.

5. CONCLUSION

This paper has proposed a matrix-sketching-based online item recommender. The matrix sketching technique has made the proposed framework highly usable and robust. In the future, **parallelizability** need to be evaluated in a realistic parallel environment, and scalability and efficiency should be improved from a practical point of view.

6. REFERENCES

[1] M. Aharon et al. OFF-Set: One-pass Factorization of Feature Sets for Online Recommendation in Persistent Cold Start Settings. In *Proc. of RecSys 2013*, pages 375–378, Oct. 2013.

[2] H. Huang and S. P. Kasiviswanathan. Streaming Anomaly Detection Using Randomized Matrix Sketching. *PVLDB*, 9(3):192–203, Nov. 2015.

[3] T. Kitazawa. Incremental Factorization Machines for Persistently Cold-Starting Online Item Recommendation. `arXiv:1607.02858 [cs.LG]`, July 2016.

[4] E. Liberty. Simple and Deterministic Matrix Sketching. In *Proc. of KDD 2013*, pages 581–588, Aug. 2013.

[5] S. Rendle. Factorization Machines with libFM. *ACM TIST*, 3(3), May 2012.

[6] J. Vinagre et al. Fast Incremental Matrix Factorization for Recommendation with Positive-only Feedback. In *Proc. of UMAP 2014*, pages 459–470, July 2014.