Incremental Factorization Machines for Persistently Cold-starting Online Item Recommendation

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

ABSTRACT

Real-world item recommenders commonly suffer from a persistent cold-start problem which is caused by dynamically changing users and items. In order to overcome the problem, several context-aware recommendation techniques have been recently proposed. In terms of both feasibility and performance, factorization machine (FM) is one of the most promising methods as generalization of the conventional matrix factorization techniques. However, since online algorithms are suitable for dynamic data, the static FMs are still inadequate. Thus, this paper proposes incremental FMs (iFMs), a general online factorization framework, and specially extends iFMs into an online item recommender. The proposed framework can be a promising baseline for further development of the production recommender systems. Evaluation is done empirically both on synthetic and real-world unstable datasets.

CCS Concepts

•**Information systems** → **Recommender systems;** *Data streams;* •**Computing methodologies** → *Factorization methods;*

Keywords

Factorization machines; Online learning; Item recommendation; Persistent cold-start

1. INTRODUCTION

In the real-world applications such as e-commerce and online ad, a user's activity is not frequent, and item properties change dynamically over time. In a context of item recommendation, such scenario is referred to as *persistent cold-start*. For instance, Booking.com [3] shows an example of users' rare activity and mixed personas (***user-side** persistent cold-start*), and Rakuten GORA [9] demonstrates price fluctuation and short life-span of packaged items (***item-side** persistent cold-start*).

Most importantly, classical recommendation techniques have some drawbacks under the persistent cold-start setting. In fact, matrix factorization (MF) [5] is one of the most typical and promising techniques, but MF only holds latent vectors for every user/item IDs; there is no way to make meaningful recommendation under an unforeseen condition. By contrast, context-aware recommender systems have been recently studied in order to profile more essential users' preferences with auxiliary features. In particular, factorization machines (FMs) [7] are alternative effective factorization models which enable us to make context-aware recommendation with flexible feature representation. Since the persistent cold-start problem commonly occurs in real applications, high feasibility of FMs is attractive compared to more specific methods developed by industrial researchers. However, the captured context by FMs is still static, and thus FM is incomplete in terms of robustness against persistently cold-starting data.

In order to adjust the model parameters according to variation of context, this paper extends FMs into online algorithms. It should be noticed that the persistent cold-start problem is closely related to *concept drift*, a phenomenon of "the relation between the input data and the target variable changes over time" [4] in data streams. On the user-side, since users' interests may be changed, systems must recommend different items even for the same user. Meanwhile, due to the instability of item trends and properties, one item can be preferred by totally different users at a different point in time. Past studies proved that online algorithms are effective to the unpredictable phenomena [4], so the author assumes that incremental update of FMs yields better accuracy compared to the static recommenders. In practice, online recommender systems behave as illustrated 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 concretely, the author generalizes prior work in incremental MF (iMF) [10], an online extension of a MF-based item recommender, to incremental FMs (iFMs). Here, the framework specially equips the following properties:

- **Positive-only feedback**: model update is fast thanks to an one-pass online learning scheme.
- **Incremental adaptive regularization**: regularization parameters are automatically adjusted on-the-fly.

2. COMPONENTS

This section introduces the different factorization techniques which compose the proposed framework.

2.1 Incremental Matrix Factorization

Vinagre et al. [10] proposed an efficient iMF algorithm for item recommendation, which is achieved by solving MF with a unique target value $y = 1$ over the stochastic gradient descent (SGD) optimization. Moreover, they evaluated the method in a *test-then-learn* scheme as outlined in Alg. 1. For each pair of a user $u \in U$ and an item $i \in I$, evaluation is first launched before updating the parameters.

Algorithm 1 Outline of the *test-then-learn* procedure

Input: data stream or finite set of positive events S_+, size of recommendation list N, window size T

Step 0: initialize parameters
Step 1: batch training by using $S_0 \subset S_+$
for $(u, i) \in S_+ \setminus S_0$ **do**
  Step 2: recommend and evaluate
  $L := \{i^* \mid i^*$ is in top-N recommended items for $u\}$
  recall@$N = 1$ if $i \in L$, otherwise 0
  recall@$N/T :=$ avg. recall@N for latest T samples
  Step 3: update

More specifically, iMF incrementally factorizes a binary matrix $R \in \mathbb{R}^{|U| \times |I|}$ into $P \in \mathbb{R}^{|U| \times k}$ and $Q \in \mathbb{R}^{|I| \times k}$ as:

- **Step 1:** Learn P and Q as the standard MF
- **Step 2:** Score items by Q $\mathbf{p}_u \in \mathbb{R}^{|I|}$, and recommend N closest items to 1
- **Step 3:** $\mathbf{p}_u \leftarrow \mathbf{p}_u + 2\eta \left(\left(1 - \mathbf{p}_u^{\mathrm{T}} \mathbf{q}_i\right) \mathbf{q}_i - \lambda \mathbf{p}_u \right)$
  $\mathbf{q}_i \leftarrow \mathbf{q}_i + 2\eta \left(\left(1 - \mathbf{p}_u^{\mathrm{T}} \mathbf{q}_i\right) \mathbf{p}_u - \lambda \mathbf{q}_i \right)$

where $\mathbf{p}_u, \mathbf{q}_i \in \mathbb{R}^k$ are respectively a user, item latent vector. The vectors are updated with a regularization parameter λ and a learning rate η.

2.2 Factorization Machines

FMs [7] have been recently developed as a general predictor. For an input vector $\mathbf{x} \in \mathbb{R}^d$, let us first imagine a linear model parameterized by $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^d$. In addition, by incorporating interactions of the d input variables, the linear model is extended to FMs as:

$$\hat{y}(\mathbf{x}) := \underbrace{w_0}_{\textbf{global bias}} + \underbrace{\mathbf{w}^{\mathrm{T}}\mathbf{x}}_{\textbf{linear}} + \sum_{i=1}^{d} \sum_{j=i}^{d} \underbrace{\mathbf{v}_i^{\mathrm{T}} \mathbf{v}_j}_{\textbf{interaction}} x_i x_j,$$

where $V \in \mathbb{R}^{d \times k}$ is a rank-k matrix which has $\mathbf{v}_1, \cdots, \mathbf{v}_d \in \mathbb{R}^k$. Arbitrary feature representation $\mathbf{x}$ (e.g. concatenation of one-hot vectors for several categorical variables) work well with FMs, and MF is actually a subset of the predictor.

3. INCREMENTAL FACTORIZATION MACHINES

At the beginning, the general iFM is proposed in Sec. 3.1 and 3.2. Next, Sec. 3.3 optimizes it for positive-only-feedback-based online item recommendation.

3.1 General Incremental Predictor

This paper focuses on running FMs in an incremental fashion. Generally, learning FM requires a set of parameters $\Theta = \{w_0, \mathbf{w}, V\}$ and a loss function $\ell(\hat{y}(\mathbf{x} \mid \Theta), y)$, and the parameters can be optimized by SGD. Specifically, for a set of samples S, the parameters of FM are updated as Alg. 2. For simplicity, $\ell(\hat{y}(\mathbf{x} \mid \Theta), y)$ is written as ℓ.

Algorithm 2 SGD update for the model parameters of FM

Input: S, learning rate η, regularization parameters $\lambda_0, \lambda_{\mathbf{w}}, \lambda_{V_1}, \ldots, \lambda_{V_k}$

repeat
  for $(\mathbf{x}, y) \in S$ **do**
    Θ-update
    $w_0 \leftarrow w_0 - \eta \left(\frac{\partial}{\partial w_0} \ell + 2\lambda_0 w_0\right)$
    for $i \in \{1, \ldots, d\} \wedge x_i \neq 0$ **do**
      $w_i \leftarrow w_i - \eta \left(\frac{\partial}{\partial w_i} \ell + 2\lambda_{\mathbf{w}} w_i\right)$
      for $f \in \{1, \ldots, k\}$ **do**
        $v_{i,f} \leftarrow v_{i,f} - \eta \left(\frac{\partial}{\partial v_{i,f}} \ell + 2\lambda_{V_f} v_{i,f}\right)$
until $w_0, \mathbf{w}, V$ are successfully learnt

Notice that **Step 3** of iMF in Sec. 2.1 is SGD update for single sample, so the basic idea of this paper is that we replace the step with **Θ-update** in Alg. 2. As a result, iFM which can be evaluated in the *test-then-learn* scheme is derived without loss of generality.

3.2 Incremental Adaptive Regularization

Rendle [8] proposed an adaptive regularization scheme for FMs. As shown in Alg. 3, the technique adjusts the regularization parameters by using a sample $(\mathbf{x}', y')$ in a validation set S', an extra set of samples which is different from S.

Algorithm 3 Update $\lambda_0, \lambda_{\mathbf{w}}, \lambda_{V_1}, \ldots, \lambda_{V_k}$

λ-update using $(\mathbf{x}', y')$ sampled from S'
$\lambda_0 \leftarrow \max(0,\ \lambda_0 - \eta \frac{\partial}{\partial \lambda_0} \ell')$
$\lambda_{\mathbf{w}} \leftarrow \max(0,\ \lambda_{\mathbf{w}} - \eta \frac{\partial}{\partial \lambda_{\mathbf{w}}} \ell')$
for $f \in \{1, \ldots, k\}$ **do**
  $\lambda_f \leftarrow \max(0,\ \lambda_f - \eta \frac{\partial}{\partial \lambda_f} \ell')$

Normally, Alg. 3 is launched after **Θ-update** in Alg. 2. However, when we consider incremental adaptive regularization, there is a difficulty that the validation set S' will be gradually outdated in a streaming environment. A key idea to conquer the problem is that a newly observed sample $(\mathbf{x}, y)$ is handled as a *pseudo* validation sample, so the regularization parameters are adjusted before updating Θ as demonstrated in Fig. 2.

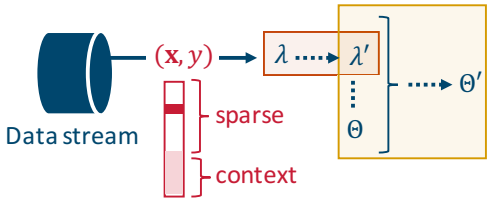

Figure 2: Incremental adaptive regularization.

3.3 Context-aware Online Item Recommendation with Positive-only Feedback

This section considers a particular combination of an output y and a loss function ℓ to utilize iFMs for online item recommendation. As the author explained in Sec. 2.1, iMF actually solves MF with $y = 1$. Similarly to the approach, let us again consider the unique output for a set of positive events S_+. As a consequence, for a sample $(\mathbf{x}, 1) \in S_+$, our loss function is defined as: $\ell(\hat{y}(\mathbf{x} \mid \Theta), 1) = (\hat{y}(\mathbf{x} \mid \Theta) - 1)^2$.

Eventually, for arbitrary design of a feature vector $\mathbf{x}$, the proposed iFM-based item recommender which is feasible in a streaming environment can be described in the *test-then-learn* framework as:

- **Step 1:** Learn w_0, $\mathbf{w}$ and V as the standard FMs
- **Step 2:** Predict $\hat{y}(\mathbf{x} \mid \Theta)$ for every items, and recommend N closest items to 1
- **Step 3:** λ**-update** $\rightarrow$ Θ**-update** with $(\mathbf{x}, 1)$

Importantly, since the number of users and items on real-world online applications is not constant, our systems must incorporate new users and items into a current model somehow. For example, iMF handles a new user (item) as an additional row of P (Q) (i.e. $\mathbf{p}_{|U|+1}$ for a new user, $\mathbf{q}_{|I|+1}$ for a new item obtained from Gaussian). Hence, iFMs also take the simple approach that a zero and random vector are respectively inserted into $\mathbf{w}$ and V as the initial parameters of new features. Fig. 3 depicts detection and insertion of new features in a stream of input vectors. It is notable that, beyond new users and items, adding new contextual variables is also possible in the middle of data streams.

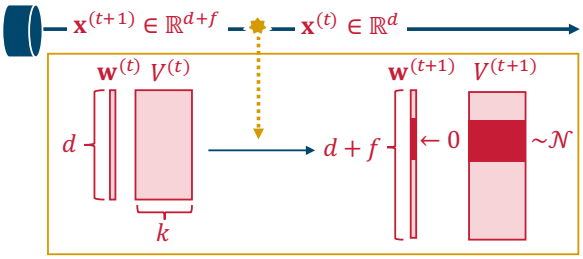

Figure 3: Incorporating new features into a model. When dimension of $\mathbf{x}$ is increased by new users, items and/or contexts, initial values fill the corresponding parameters.

In terms of computational complexity, iFMs compute the interaction term $\sum_{i=1}^{d}\sum_{j=i}^{d}\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j x_i x_j$ in $\mathcal{O}\left(kN_z(\mathbf{x})\right)$ by letting the number of nonzero elements in $\mathbf{x}$ be $N_z(\mathbf{x})$. In fact, this complexity is efficient enough due to sparsity of $\mathbf{x}$, but running time will be relatively long compared to the single vector updating of iMF. Therefore, there is trade-off between running time and context-awareness in practice.

4. EXPERIMENTS

4.1 Evaluation Method

In the experiments, the *test-then-learn* procedure described in Sec. 2.1 is launched for time-stamped unstable datasets. As shown in Fig. 4, the samples are separated similarly to what Matuszyk et al. [6] did. At the beginning, a batch train/test step is executed for the first 20% training and following 10% validation samples. Next, the 10% samples are just used for one-pass model updating, and the incremental evaluation step is finally performed for the remaining 70%.

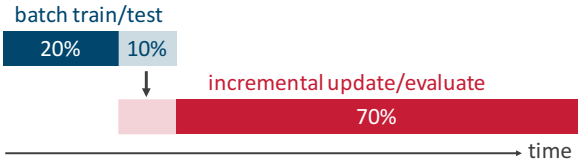

Figure 4: Splitting time-stamped samples.

Additionally, the 70% samples are also evaluated by mean percentile rank (MPR) in order to assess an ordered list of items obtained from **Step 2**. The metric is calculated based on percentile rank of the correct items in the ordered lists; that is, MPR = 0% indicates the best result that a recommender always gives the highest rank to a correct item, and MPR = 100% is opposite. In contrast to recall@N/T which evaluates top-N items, MPR can measure users' overall satisfaction from all items.

Following methods were employed as the competitors:

- **static MF**: the traditional MF,
- **iMF**: a fast extension of MF introduced in Sec. 2.1,
- **static FMs**: the parameters are not updated in the incremental stage similarly to **static MF**.

The author implemented all of the methods in Python 3.5.1, and the code was run in a typical personal computer with the 2.7 GHz Intel® Core™ i7 CPU and 4GB RAM.

Datasets used in the experiments were binarized version of MovieLens 100k (ML100k)[1] and synthetic click data; the former is a real-world example of user-side volatility, and the latter shows item-side instability. Table 1 summarizes statistics of the data. Each pair of a method and a dataset was tested five times with different initial parameters.

Table 1: Statistics of the datasets.

Dataset	Users	Items	Positive events		
			20%	10%	70%
ML100k	928	1172	4240	2120	14841
Synthetic	3570†	5	714	357	2499

† Random demographics were generated instead of user ID itself, so # of users and clicks are same.

4.2 Results and Discussions

ML100k. Since we focus on item recommendation, ML100k was binarized by extracting 5-starred rating events. An input vector of iFMs was designed as:

$$\mathbf{x} = (\underbrace{\text{user ID}}_{1/|U|}, \underbrace{\text{demographics}}_{3/23}, \underbrace{\text{movie ID}}_{1/|I|},$$
$$\underbrace{\text{genre}}_{18/18}, \underbrace{\text{last rated genre}}_{18/18}, \underbrace{\text{day}}_{1/7}, \underbrace{\text{last rated day}}_{1/7}).$$

"demographics" includes user's occupation (1/21), sex (1/1) and age (1/1), and "day" is day of week in the timestamps. Note that the numbers below the underbraces indicate {*max. # of nonzero dimensions*}/{*# of total dimensions*}.

Most users on ML100k report positive events just for the initial rating activity, and they do not rate any more. Even if some users continuously rate movies, intuition tells us that their interests change over time. Thus, ML100k is a real-world example of user-side persistent cold-start.

Fig. 5 shows the best recall behaviors on ML100k obtained from the five *test-then-learn* trials. ML100k is rich in both user and item features, and we also considered time-related contexts. Consequently, static FMs and iFMs respectively outperformed static MF and iMF, especially in terms of MPR. In addition, it is clear that online recommenders (iMF and iFMs) worked effectively compared to the static counterparts.

[1] http://grouplens.org/datasets/movielens/

Table 2: Hyperparameters and average experimental results of 5 trials. The best accuracy is written in **bold**.

Dataset	Method	Hyperparameters	Running time [sec.]		recall@N/T	MPR [%]
			recommend	update	*mean* ($\pm$ *std*)	*mean* ($\pm$ *std*)
ML100k (@10/3000)	static MF	$k = 40$, $\eta = 0.002$	0.00016	—	0.021 ($\pm$ 0.004)	49.68 ($\pm$ 0.320)
	iMF	$\lambda = 0.01$	0.00015	0.00003	0.026 ($\pm$ 0.006)	47.32 ($\pm$ 0.359)
	static FMs	$k = 40$, $\eta = 0.004$	0.02427	—	0.023 ($\pm$ 0.008)	36.07 ($\pm$ 0.127)
	iFMs	$\lambda_0 = 2.0$, $\lambda_{\mathbf{w}} = 8.0$, $\lambda_{V_k} = 16.0$	0.02449	0.00150	**0.035** ($\pm$ 0.008)	**32.55** ($\pm$ 0.023)
Synthetic (@1/500)	static MF	$k = 2$, $\eta = 0.0003$	0.00002	—	0.271 ($\pm$ 0.248)	54.59 ($\pm$ 4.386)
	iMF	$\lambda = 0.01$	0.00002	0.00003	**0.316** ($\pm$ 0.213)	49.24 ($\pm$ 2.254)
	static FMs	$k = 2$, $\eta = 0.00006$	0.00325	—	0.271 ($\pm$ 0.248)	37.83 ($\pm$ 3.385)
	iFMs	$\lambda_0 = \lambda_{\mathbf{w}} = \lambda_{V_k} = 0.01$	0.00315	0.00026	**0.316** ($\pm$ 0.208)	**34.26** ($\pm$ 1.429)

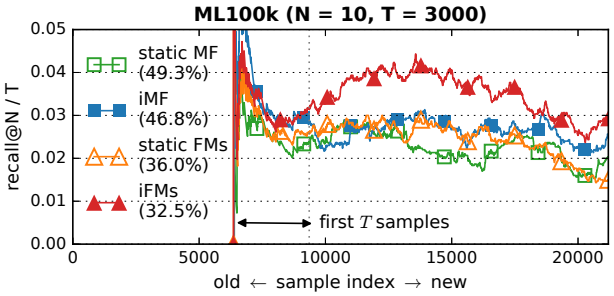

Figure 5: Recall behavior on ML100k. Higher recall is better on the y-axis. MPR is written in the legend area.

Synthetic. Synthetic click data was generated as an example of item-side persistent cold-start, based on a rule-based procedure demonstrated in [1]. In particular, our generator first produced 0.5 million impressions of five ad variants, and additional half million impressions were also generated after updating a rule for the most popular ad. As a result, from the one million impressions, 3,570 synthetic clicks were observed as positive events with erratic trend. Here, $\mathbf{x}$ for the synthetic data was:

$$\mathbf{x} = (\underbrace{\text{age}}_{1/1}, \underbrace{\text{sex}}_{1/1}, \underbrace{\text{geo (state)}}_{1/50}, \underbrace{\text{ad ID}}_{1/|I|}, \underbrace{\text{category}}_{1/3}).$$

The best result on the synthetic data is illustrated in Fig. 6. All methods were easily fit to the batch training samples due to the simplicity of data, but the difference can be observed after the most popular ad was changed. While recall of the static methods declined significantly, iMF and iFMs evidently adapted to the variation as expected. Moreover, MPR and recall of iFMs were even better than iMF. It should be noted that, even though static MF and FMs demonstrated the similar recall behavior, FMs showed reasonably lower MPR compared to MF.

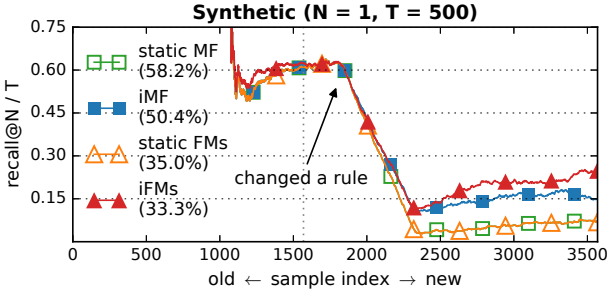

Figure 6: Recall behavior on synthetic click data.

Finally, Table 2 summarizes the results of 5 trials. The positive effect of context-awareness and online model updating was validated in terms of accuracy. On the other hand, running time of FMs was more than 100 times slower than MF. iFMs updated the parameters in a millisecond range, and recommendation for a user was done at least 30 milliseconds. iFM thus seems to satisfy practical time requirements for now, but higher-dimensional and denser input vectors may lead worse results in the future. As the author mentioned in Sec. 3.3, the fact proved trade-off between context-awareness and efficiency.

Overall, the proposed online item recommender based on iFMs worked well as generalization of iMF. In case that an organization develops production recommender systems, our highly feasible framework can be an easy-to-implement baseline. Furthermore, availability of many third-party libraries (e.g. [2, 7]) is an important advantage of FMs.

5. CONCLUSION

This paper has proposed an iFM-based context-aware online item recommender. Experimental results have demonstrated not only feasibility and effectiveness of the technique but also a new challenge in computational efficiency.

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] I. Bayer. fastfm: A library for factorization machines. *arXiv:1505.00641 [cs.LG]*, 2015.

[3] L. Bernardi et al. The continuous cold start problem in e-commerce recommender systems. In *Proc. of CBRecSys 2015*, pages 30–33, Sep. 2015.

[4] J. Gama et al. A survey on concept drift adaptation. *ACM CSUR*, 46(4), Mar. 2014.

[5] Y. Koren et al. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug. 2009.

[6] P. Matuszyk et al. Forgetting methods for incremental matrix factorization in recommender systems. In *Proc. of SAC 2015*, pages 947–953, Apr. 2015.

[7] S. Rendle. Factorization machines with libfm. *ACM TIST*, 3(3), May 2012.

[8] S. Rendle. Learning recommender systems with adaptive regularization. In *Proc. of WSDM 2012*, pages 133–142, Feb. 2012.

[9] R. Swezey and Y. Chung. Recommending short-lived dynamic packages for golf booking services. In *Proc. of CIKM 2015*, pages 1779–1782, Oct. 2015.

[10] J. Vinagre et al. Fast incremental matrix factorization for recommendation with positive-only feedback. In *Proc. of UMAP 2014*, pages 459–470, July 2014.