1C2-5

Incremental Factorization Machines for Item Recommendation in Data Streams

Takuya Kitazawa

Graduate School of Information Science and Technology, The University of Tokyo, Japan

When thousands of users frequently interact with items on real-world applications, how can we implement real-time recommender systems which update models incrementally? This study focuses on effective item recommendation in such streaming environments and examines factorization machines (FMs) in an incremental fashion. The author integrates incremental matrix factorization (iMF) and FMs, and implements incremental FMs for the real-time item recommendation. The proposed recommender is (1) a general incremental predictor: incremental FMs can achieve similar recommendation accuracy to the iMF-based conventional recommender, a specific incremental predictor for user-item matrices, (2) robust: context-aware incremental FMs with optimal hyperparameters show robust accuracy even in a streaming, drifting concepts environment, and (3) fast: recommendation and model updating for a user is done in less than 0.1 second in typical personal computers. These strengths are evaluated by an experiment using the time-stamped movie rating dataset.

1. Introduction

In this paper, to implement a highly flexible item recommender in a streaming fashion, the author proposes incremental factorization machines (iFMs). Additionally, the author also considers a relationship between iFMs and concept drift, a well-known issue that characteristics of data change over time in data streams.

While a lot of real-world applications such as e-commerce shows high-frequency interactions between users and items, conventional recommendation techniques like matrix factorization (MF) [1] of a user-item matrix are strongly optimized for batch processing. So, several incremental algorithms also have been studied, and an efficient incremental MF algorithm with positive-only feedbacks (iMF) [2] is one of the latest proposals.

On the other hand, recently, factorization machines (FMs) [3] have been proposed as an alternative effective factorization model. FMs are general predictors which can incorporate various variables into a model as contextual information. The model can be seen generalized MF, but, to the best of our knowledge, no one has studied to use FMs in data streams as shown in Table 1.

Table 1: Comparison of the factorization algorithms

	MF [1]	iMF [2]	FMs [3]	**iFMs**
incremental		✓		✓
context-aware			✓	✓

Contributions of this paper are listed as follows:

1. *General incremental predictor*: iFMs without contextual information achieve similar recommendation accuracy to the conventional iMF-based recommender.
2. *Robust*: Context-aware iFMs with optimal hyperparameters show robust accuracy even in streaming, drifting concepts environments.
3. *Fast*: For a user, iFMs recommend and update parameters in less than 0.1 second in typical personal computers.

Contact: k.takuti@gmail.com

2. Item Recommendation in Data Streams

This section formulates our recommendation problem and introduces iMF, one of the most promising previous work to solve the problem.

2.1 Problem Formulation

When a user u visits an application, our systems need to recommend top-N items under a scoring procedure. Moreover, after observation of an event by a user (e.g. click, buy, rate), the recommender incrementally updates a model based on the observed event between the user u and an item i. This recommend-then-update procedure is illustrated in Fig. 1.

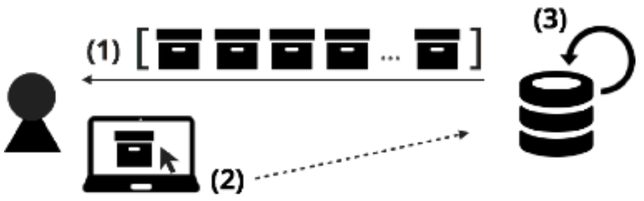

Figure 1: **Item recommendation in data streams**: (1) a recommender provides a top-N list to a user, (2) the user interacts with an item on an application, and (3) our model is updated based on the interaction.

Therefore, the problem is to develop an algorithm which can deal the above situation. Since such recommender systems always make a top-N list by using recently updated model, recommendation accuracy at the time can be maximized, and the systems flexibly adapt to unforeseen events. This paper focuses on rating, among other types of events, and try to predict high-rated items.

2.2 Incremental Matrix Factorization

MF [1] is one of the most promising solutions to make the top-N recommendation based on user-item matrices. When we have a matrix $R \in \mathbb{R}^{n \times m}$, MF decomposes it into two factorized matrices $P \in \mathbb{R}^{n \times k}$ and $Q \in \mathbb{R}^{m \times k}$. Hence, R is approximated by PQ^{T}.

Let U, I be a set of user, item indices respectively, and $R \in \mathbb{R}^{|U| \times |I|}$ be a user-item rating matrix. What MF actually does is to solve the following minimization problem with a regularization parameter λ, for a set of observed user-item ratings $S = \{(u,i) \in U \times I\}$, a user factorized matrix $P \in \mathbb{R}^{|U| \times k}$, and an item factorized matrix $Q \in \mathbb{R}^{|I| \times k}$.

$$\min_{P,Q} \sum_{(u,i) \in S} \left(r_{u,i} - \mathbf{p}_u^{\mathrm{T}} \mathbf{q}_i \right)^2 + \lambda \left(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2 \right). \quad (1)$$

Here, $r_{u,i}$ indicates a (u,i) element in R, and $\mathbf{p}_u, \mathbf{q}_i \in \mathbb{R}^k$ are respectively a factorized user, item vector. In a batch fashion, an optimal solution can be found by using optimization techniques such as stochastic gradient descent (SGD). Ultimately, we can predict unobserved ratings by computing PQ^{T} and make recommendation based on the predicted ratings.

While conventional MF has a great impact on modern recommender systems, the model is not suitable for incremental model updating and real-time prediction. So, Vinagre et al. [2] have proposed iMF, an incremental MF algorithm with positive-only feedbacks; a recommender incrementally updates a model P and Q when a positive event (e.g. rating with a maximum score) is observed. Thanks to the incrementally updating logic, a real-time prediction and evaluation algorithm in a streaming environment had been proposed [4]. Here, Alg. 1 demonstrates a *recommend-evaluate-update* procedure based on iMF. The algorithm computes the simple moving averages of recalls for the latest w positive samples, and enables us to track evolution of recommendation accuracy.

Algorithm 1 iMF: Recommendation and evaluation in a streaming environment

```
Input: data stream or finite set of positive-only events
    $S$, number of factorized features $k$,
       size of a recommendation list $N$,
       window length for the simple moving average $w$,
       regularization parameter $\lambda$, learning rate $\eta$
1: Generate $P \in \mathbb{R}^{|U| \times k}$, $Q \in \mathbb{R}^{|I| \times k}$ from Gaussian
2: for $(u,i) \in S$ do
       Step 1: Recommend and evaluate
3:     Score items: $Q\ \mathbf{p}_u \in \mathbb{R}^{|I|}$
4:     Search an item $i$ from the top-$N$ scored items
5:     Compute an average recall for the latest $w$ samples
       Step 2: Update (iMF)
6:     $\mathbf{p}_u \leftarrow \mathbf{p}_u + 2\eta\ \left( (1 - \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i)\ \mathbf{q}_i - \lambda\ \mathbf{p}_u \right)$
7:     $\mathbf{q}_i \leftarrow \mathbf{q}_i + 2\eta\ \left( (1 - \mathbf{p}_u^{\mathrm{T}}\mathbf{q}_i)\ \mathbf{p}_u - \lambda\ \mathbf{q}_i \right)$
```

3. Incremental Factorization Machines

3.1 Factorization Machines

Beyond numerous discussions about MF, recently FMs [3] have been developed as general predictors based on the "factorization" idea. In contrast to MF, FMs are formulated by one simple equation which is very similar to the polynomial regression, and the model can be applied all of the regression, classification and ranking problems.

First of all, for an input vector $\mathbf{x} \in \mathbb{R}^d$, let us imagine a linear model parameterized by $w_0 \in \mathbb{R}$, $\mathbf{w} \in \mathbb{R}^d$ as $\hat{y}^{\mathrm{LR}}(\mathbf{x}) := w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x}$. Next, by incorporating interactions of the d input variables, we extend the linear model into the following second-order polynomial model.

$$\hat{y}^{\mathrm{PR}}(\mathbf{x}) := w_0 + \mathbf{w}^{\mathrm{T}}\mathbf{x} + \sum_{i=1}^{d} \sum_{j=i}^{d} w_{i,j} x_i x_j. \quad (2)$$

Note that $w_{i,j}$ is an element in a symmetric matrix $W \in \mathbb{R}^{d \times d}$, and it indicates a weight of $x_i x_j$, an interaction between the i-th and j-th variable.

FMs assume that W can be approximated by a low-rank matrix; w_{ij} in Eq. (2) is approximated by using a rank-k matrix $V \in \mathbb{R}^{d \times k}$, and the weights are replaced with inner products of k dimensional vectors as $w_{i,j} \approx \mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j$ for $\mathbf{v}_1, \cdots, \mathbf{v}_d \in \mathbb{R}^k$. Finally, the model is formulated as follows:

$$\hat{y}(\mathbf{x}) := \underbrace{w_0}_{\textbf{global bias}} + \underbrace{\mathbf{w}^{\mathrm{T}}\mathbf{x}}_{\textbf{linear}} + \sum_{i=1}^{d} \sum_{j=i}^{d} \underbrace{\mathbf{v}_i^{\mathrm{T}}\mathbf{v}_j}_{\textbf{interaction}} x_i x_j. \quad (3)$$

Rendle [3] demonstrated that Eq. (3) can work as a generalized factorization model; that is, a wide variety of prediction model can be designed depending on a feature vector and a loss function. For instance, let $\mathbf{x} \in \mathbb{R}^{|U|+|I|}$ be a sparse input vector which has 1 only on x_u and x_i as:

$$(\underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{\text{user } (1/|U|)}, \underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{\text{item } (1/|I|)}) \in \mathbb{R}^{|U|+|I|}. \quad (4)$$

The input vector makes Eq. (3) simpler:

$$\hat{y}(\mathbf{x}) = w_0 + w_u + w_i + \mathbf{v}_u^{\mathrm{T}}\mathbf{v}_i, \quad (5)$$

and this formulation is clearly equivalent to biased MF; $\mathbf{v}_u$ and $\mathbf{v}_i$ respectively correspond to $\mathbf{p}_u$ and $\mathbf{q}_i$ in Eq. (1). Hence, if we choose the squared loss as a loss function, incremental biased MF (biased-iMF) can also be implemented in a similar way to Alg. 1.

3.2 Proposed Algorithm

This section focuses on FMs running in an incremental fashion. Generally, learning FMs requires a set of parameters $\Theta = \{w_0, \mathbf{w}, V\}$ and a loss function $\ell(\hat{y}(\mathbf{x} \mid \Theta), y)$ as the formulation in the previous section shows, and the parameters can be optimized by SGD. More specifically, for one sample $(\mathbf{x}, y) \in S$, the parameters are updated as Alg. 2. Here, $\hat{y}(\mathbf{x} \mid \Theta)$ is written as $\hat{y}$ for simplicity.

Algorithm 2 FMs: SGD parameter updating

```
Input: $(\mathbf{x}, y)$, $k$, $\eta$,
       regularization parameters $\lambda_0, \lambda_{\mathbf{w}}, \lambda_{V_1}, \ldots, \lambda_{V_k}$
1: $w_0 \leftarrow w_0 - \eta \left( \frac{\partial}{\partial w_0} \ell(\hat{y}, y) + 2\lambda_0 w_0 \right)$
2: for $i \in \{1, \ldots, d\} \wedge x_i \neq 0$ do
3:     $w_i \leftarrow w_i - \eta \left( \frac{\partial}{\partial w_i} \ell(\hat{y}, y) + 2\lambda_{\mathbf{w}} w_i \right)$
4:     for $f \in \{1, \ldots, k\}$ do
5:         $v_{i,f} \leftarrow v_{i,f} - \eta \left( \frac{\partial}{\partial v_{i,f}} \ell(\hat{y}, y) + 2\lambda_{V_f} v_{i,f} \right)$
```

Notice that **Step 2** (iMF) in Alg. 1 is SGD updating for single sample, so, if we replace the step with Alg. 2, an incremental version of FMs is easily derived.

Let the number of nonzero elements in $\mathbf{x}$ be $N_z(\mathbf{x})$. Importantly, computational complexity of the interaction term $\sum_{i=1}^{d} \sum_{j=i}^{d} \mathbf{v}_i^{\mathrm{T}} \mathbf{v}_j x_i x_j$ is $\mathcal{O}\left(k N_z(\mathbf{x})\right)$. Consequently, if $\mathbf{x}$ is sparse like Eq. (4), the term can be efficiently computed.

3.3 Context-aware Model against Concept Drift

We can incorporate arbitrary contextual variables into iFMs in the same way as FMs, so a dataset can be enriched on-the-fly and adapted to unforeseen concept drift. To give a motivating example, Fig. 2 shows an experimental result of iMF (Alg. 1), which is similar to the result written in the original iMF paper [2]. To avoid going into details, x-axis and y-axis are simplified as *time* and *accuracy* respectively; the details are explained in Section 4.

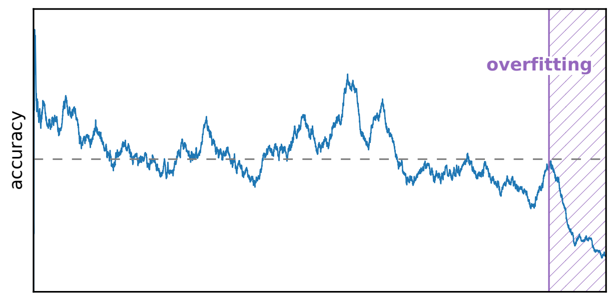

Figure 2: **Time evolution of accuracy**: computed at **Step 1** in Alg. 1. The center dashed line indicates an overall average of the accuracy.

Fig. 2 clearly illustrates that the accuracy around the tail of the graph (the purple diagonal area) is declined significantly, and the reason is probably concept drift. On the one hand, users interacted with relatively new items, and concepts (i.e. true items what the recommender should recommend) are changed. However, at the same time, a learnt model on our recommender is fitted in the interactions before drifting concepts. As a consequence, overfitting is observed as shown in Fig. 2.

Since drifting concepts require a recommender to predict unknown items for unknown users, contextual variables are greatly helpful. For example, a category and a price of an item assist to find similar items, and users' demographics are good supplementary information. These techniques can be easily implemented on iFMs, by just extending the feature vector described in Eq. (4).

4. Experiment on a Time-Stamped Dataset

4.1 The MovieLens Dataset for Incremental Recommender Systems

For experimental evaluation of the incremental recommender systems, Vinagre et al. [4] first extracted positive-only samples from time-stamped datasets. Next, they separated the extracted samples into two parts; the initial 20% samples are used for pre-training to avoid cold-start, and evaluation is done by using the remaining 80% samples.

In this paper, the author chooses the MovieLens 1M dataset, a well-known movie rating dataset, to examine our iFMs algorithm, because iMF had also been evaluated on the data [2], and concept drift is clearly observed on the dataset as shown in Fig. 2. Basic properties of the dataset is summarized in Table 2 and Fig. 4.

Table 2: Properties of the MovieLens 1M dataset

Event type	rating	
Value range	1–5 (5 is positive)	
# positive samples	pre-train (20%)	45,262
	evaluation (80%)	181,048

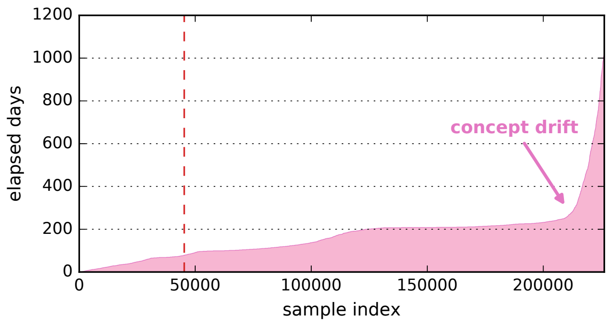

Figure 4: **Elapsed days**: from the first positive sample. The red dashed line marks the 20% pre-training samples.

According to Fig. 4, most positive samples are in less than 300 days from the first sample; unknown items and users are not likely to be observed, and the incremental algorithms probably work well. By contrast, concept drift may occur around the end of the dataset because the elapsed days are rapidly increased.

4.2 Experimental Condition

The author designed a context-aware input vector $\mathbf{x}$ as:

$$\Big(\underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{\text{user } (1/|U|)}, \underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{\text{item } (1/|I|)}, \underbrace{10}_{\text{elapsed days } (1)}, \underbrace{0, \cdots, 1, 0, 1, \cdots, 0}_{\text{genres } (n/18)}, \underbrace{1}_{\text{sex } (1)}, \underbrace{3}_{\text{age group } (1)}, \underbrace{0, \cdots, 0, 1, 0, \cdots, 0}_{\text{occupation } (1/21)} \Big) \in \mathbb{R}^{|U|+|I|+42}, \quad (6)$$

by extending Eq. (4). A variable "elapsed days" is from Fig. 4, and the other variables are directly obtained from the MovieLens dataset.

As the competitors,

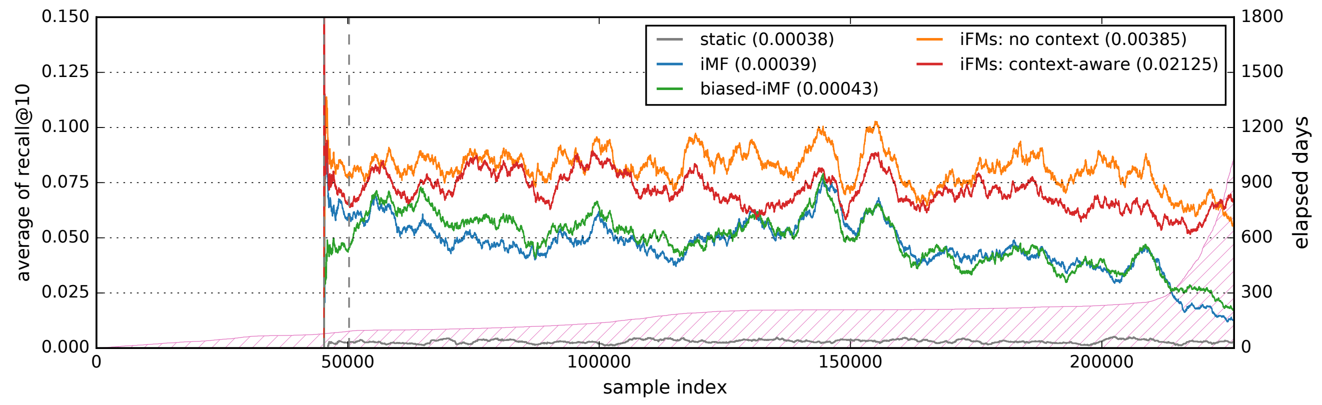

Figure 3: **Average recall@10 behavior**: The simple moving averages ($w = 5000$) of the recommendation accuracy for the MovieLens 1M dataset. The gray dashed line near $x = 50000$ indicates the first 5,000 evaluation samples, and the background pink area illustrates elapsed days which are identical with Fig. 4. Pre-training is done in the initial 20%. Values written in the legend area are average running times of the *recommend-evaluate-update* procedure for one sample.

① **static**: Alg. 1 without **Step 2**,
② **iMF**: Alg. 1,
③ **biased-iMF**: **Step 2** in Alg. 1 considers the bias terms in Eq. (5)

are employed. Additionally, iFMs (replaced **Step 2** in Alg. 1 with Alg. 2) are tested with the two different feature vectors:

④ **iFMs: no context**: Eq. (4),
⑤ **iFMs: context-aware**: Eq. (6).

Hyperparameters are set as Table 3. Here, η is a learning rate, and k indicates a factorized "rank" of each variable.

Table 3: Hyperparameters of the examined methods

	regularization parameters	η	k
①	$\lambda = 0.01$	0.030	4
②			
③			
④	$\lambda_0 = \lambda_{\mathbf{w}} = 0.01,$	0.003	
⑤	$\lambda_{V_1} = \cdots = \lambda_{V_k} = 30.0$		

In addition to Table 3, λ_0 and $\lambda_{\mathbf{w}}$ are automatically updated in an adaptive regularization scheme [5].

The author implemented all of the methods in Python 2.7.11, and the code was run in a typical personal computer with the 2.7 GHz Intel® Core$^{\mathrm{TM}}$ i7 CPU and 4GB RAM.

4.3 Results and Discussions

Time evolution of the average recalls resulting from the *recommend-evaluate-update* procedure is illustrated in Fig. 3. Recall@10 is 1 if and only if a true observed item i is in a top-10 list of items for a user u; higher plots indicate better, accurate recommendation. In terms of running time, all of the incremental algorithms recommended items and updated parameters in less than 0.1 second per sample.

Unsurprisingly, whereas the static baseline (grey) showed poor accuracy, iMF (blue) and biased-iMF (green) generated more accurate recommendations thanks to their fast, incremental algorithm. However, as the author mentioned in Section 3.3, incremental models without contextual information are easily fallen into overfitting. So, their recall@10 averages for the late samples were declined as Fig. 2.

On iFMs, the feature vector without contextual variables (orange) showed the best overall accuracy, but the late decline caused by concept drift is still observed. On the other hand, the context-aware input vector (red) showed robust recommendation even under drifting concepts condition, with similar overall accuracy to the orange line. These results suggest that context-aware iFMs work as fast, robust incremental recommender systems.

5. Conclusion

This paper has proposed and examined iFMs, general incremental predictors, for an item recommendation task in data streams. As a result of the experiment, iFMs have shown robust recommendation against concept drift, whereas the conventional factorization models fallen into overfitting. Meanwhile, the factorization models still require extra work to find optimal hyperparameters. So, in the future, adaptive optimization like [5] must be studied more, especially in a streaming fashion.

References

[1] Y. Koren, et al., "Matrix Factorization Techniques for Recommender Systems," *Computer*, 42(8):3037, Aug. 2009.

[2] J. Vinagre, et al., "Fast Incremental Matrix Factorization for Recommendation with Positive-only Feedback," *Proc. of UMAP 2014*, July 2014.

[3] S. Rendle, "Factorization Machines with libFM," *ACM Trans. Intell. Syst. Technol.*, 3(3), May 2012.

[4] J. Vinagre, et al., "Evaluation of Recommender Systems in Streaming Environments," *Proc. of REDD 2014*, Oct. 2014.

[5] S. Rendle, "Learning Recommender Systems with Adaptive Regularization," *Proc. of WSDM 2012*, Feb. 2012.