Incremental Graph of Sequential Interactions for Online Recommendation with Implicit Feedback

Murilo F. L. Schmitt
Federal University of Paraná
Curitiba, Paraná, Brazil
muriloschmitt@gmail.com

Eduardo J. Spinosa
Federal University of Paraná
Curitiba, Paraná, Brazil
spinosa@inf.ufpr.br

ABSTRACT

Recommender systems aim to recommend items to users based on their interests. Traditional models usually adopt batch processing. Considering that user feedback is generated continuously, it becomes desirable to design models that are capable of learning as data arrives. In this work, we propose an incremental graph of sequential user interactions using implicit feedback from a data stream, with the assumption that user behavior can be extracted from such sequence of interactions as time passes. The model was evaluated by recommending items with different strategies, and such strategies were compared with an incremental matrix factorization algorithm, using a prequential approach. Results highlight the potential of the proposed method, which obtained superior accuracy than the baseline with generally better update and recommendation times.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Data stream mining**; • **Theory of computation** → **Online algorithms**.

KEYWORDS

Recommender Systems; Data Streams; Incremental Learning; Implicit Feedback

Reference Format:
Murilo F. L. Schmitt and Eduardo J. Spinosa. 2020. Incremental Graph of Sequential Interactions for Online Recommendation with Implicit Feedback. In *3rd Workshop on Online Recommender Systems and User Modeling (ORSUM 2020), in conjunction with the 14th ACM Conference on Recommender Systems, September 25th, 2020, Virtual Event, Brazil.*

1 INTRODUCTION

Given the massive amount of data available in online services of various sorts, ways of filtering information to users are necessary in order to improve their interaction with the system. Recommender systems are designed to filter such data, guiding users through the item collection by presenting items based on their preferences.

Collaborative filtering (CF) is an effective technique to solve this problem, in which the prediction of unknown user-item preferences are inferred based on past user behavior. User feedback can be explicit, e.g., a user assigns a specific rating to an item, or implicit, which indirectly captures user behavior, for instance, through browsing history [9]. Traditional CF approaches such as neighborhood methods (K-nearest neighbors) and latent factor models (matrix factorization) usually adopt batch data processing.

While such approaches are generally effective in terms of predictive capability, there is the assumption that training data is always available for updating the model, and usually temporal sequence is disregarded. Considering that in many real scenarios intrinsically time-dependent data is generated continuously at unprecedented rates, it becomes impractical to update these models as new data is generated.

In that sense, learning and updating a model with one (or few) example(s) is preferable, and even required in real-world applications. To that end, incremental algorithms can be used for recommendation by treating feedback as a data stream, i.e., incorporating feedback into the model as data arrives and discarding examples after they are processed.

In this paper, we propose to incorporate implicit user feedback into a graph in incremental fashion with the assumption that user behavior can be extracted from the sequence of user interactions as time progresses, capturing short-term and long-term interests. To that end, edges are continuously included in the graph and their weights are updated according to the sequential user interactions, such that for each incoming user feedback in a data stream, a directed edge connects the last item interacted by the user to the current interaction, and the frequency in which this sequential interaction occurs is reinforced in the weight of the edge. We evaluated the proposed model by recommending items with different strategies and compare the results with an incremental matrix factorization method [23] using a prequential protocol [21], obtaining superior accuracy and generally better update and recommendation times.

The remainder of this paper is organized as follows: Section 2 presents related work. Section 3 presents the proposed approach. Section 4 presents experiments and results. Conclusions and future work are presented in Section 5.

2 RELATED WORK

This section describes the related work, categorized as follows:

Time-dependent CF. These approaches treat feedback as a chronological sequence, using time for modeling user preferences, while implicitly capturing temporal dynamics [19]. Approaches such as matrix and tensor factorization models have been studied [8, 26]. In Das et al. [4], an approach for news recommendation using pLSA, MinHash clustering and covisitation counts was proposed. The covisitation is implemented as a graph, such that nodes represent items and edges represent covisitation of items. Baluja et al. [2] proposed personalized video recommendation based on covisitation graphs. Assuming that recent data better reflects the interests of users, techniques to increase importance of recent feedback were proposed, such as decay functions [5, 6, 10] and sliding-windows [13, 15, 22]. Data pre-processing approaches to capture user interest drifts were studied by Cao et al. [3]. For comprehensive reviews considering algorithms and methodologies related to this topic, we refer the reader to the works of Vinagre et al. [25] and Quadrana et al. [17].

Incremental CF. Recently, studies with incremental CF for implicit feedback also known as one-class CF [16], have been developed. Vinagre et al. [21] proposed an incremental version of the Stochastic Gradient Descent method [9] (ISGD), which updates the model based solely on the current observation in a data stream. The paper also proposes a prequential evaluation methodology that allows the continuous monitoring of the systems' predictive capacity. Considering that in Vinagre et al. [21] all user feedback is treated as positive, a follow-up study [24] proposed a recency-based scheme to perform negative preference imputation into ISGD (RAISGD). In Anyosa et al. [1], an incremental co-factorization algorithm (CORAISGD) was proposed and compared to RAISGD on music domain datasets, obtaining superior results. Ramalho et al. [18] presented a robust comparison between incremental matrix and tensor factorization.

Graph-based methods. Regarding graph-based methods for time-dependent recommendation, the following papers are relevant to the scope of the present work. Xiang et al. [26] incorporate short-term and long-term user preferences into a bipartite graph, Session-based Temporal Graph (STG), where nodes represents users, items and sessions, and edges balance the influence of short-term and long-term preferences. User and item nodes are connected based on past user interactions, representing long-term preferences. Item and user-session nodes are connected based on user interactions in a time window, representing short-term interests. To address cold-start issues, Trevisiol et al. [20] proposed the use of two graphs, named BrowseGraph and ReferrerGraph to make news recommendation to new users. BrowseGraph [11, 12] is built according to users' browsing behavior, where nodes represent web-pages and edges connect nodes based on users' transitions between pages. ReferrerGraph is a subgraph of the BrowseGraph induced by user sessions with the same referrer domain. To predict the next page to a newcoming user, the neighbors of both graphs where considered as candidates, and four strategies to select the next page where used: random, content-based, most popular and edge-weight-based, with the edge-based approach obtaining the best results. For location recommendations, Zhang et al. [27] proposed to incorporate sequential patterns from users' check-in behaviors in a location-location transition graph in incremental manner, where nodes represent locations, edges represent transitions between locations, and edge weights are based on transitions count. The proposed method, LORE, integrates sequential influences with social and geographical information to make recommendations.

Our work is influenced by Vinagre et al. [23] and Trevisiol et al. [20]. Vinagre et al. [23] highlights the importance of updating a model incrementally by proposing an online evaluation protocol and also an algorithm capable of updating the model based solely on the current observation in a data stream (ISGD). The work of Trevisiol et al. [20] demonstrates the potential of the BrowseGraph approach for cold start issues, which is directly related to recommendation in an online manner. To that end, we compare our approach with ISGD using the prequential evaluation described in Vinagre et al. [21, 23].

3 PROPOSED APPROACH

This section presents the proposed approach, which treats the item recommendation problem under a data stream framework. That is, intrinsically time-dependent data (user feedback) is generated continuously at unprecedented rate and unpredictable order. In that sense, it is desirable to incrementally update the model, while being able to include new concepts and adapt old ones as new data arrives.

Considering the intrinsic relation between data and time, such that user preferences adjust over time, we assume that the sequence of user interaction can be important in defining user behavior. With such definition, potentially relevant item recommendations can be made to users based on past user behavior. As an example, the release of a film can lead a user to watch the director's past work before watching the film. Another example is the birth of a child. As time progresses, the family can direct their purchases towards products intended for the children as they grow. In that sense, it is reasonable to assume that the sequence of interactions can be useful in modeling short-term and long-term user interest.

Consequently, the premise of this work is to learn user behavior from implicit feedback as time progresses, representing the interactions between users and items in a graph in incremental manner, allowing the inclusion of new incoming users and items continuously. Then, information extracted from past interactions represented in the graph can be used to generate future recommendations.

3.1 Incremental Graph of Sequential Interactions

In order to continuously capture sequential interactions between users and items, we create a directed graph, where nodes represent items and edges represent user interactions, such that the edge direction indicates the order in which items where visited, i.e., sequential interaction. Thus, an edge from item i to item j exists if a user interacted with item i, and the next interaction was with item j. Therefore, for each new user interaction, the feedback is included into the graph by an edge that connects the last interacted item to the item of the new interaction.

To distinguish the relevance of edges, each edge has an associated weight, where the weights are inversely proportional to the frequency in which a transition is made by users. In other words, the higher the frequency of a sequential interaction of two items, the lower the edge weight between the two items, implicitly measuring the relevance of the edge for future recommendations. Notation used throughout this work is summarized in Table 1.

Figure 1 shows an example of the graph online maintenance, illustrating two possible scenarios based on sequential interaction. Consider the graph presented in Figure 1a and a user u, whose last interaction was with item 5. At some time t, u interacts with item 0. Since there is no edge connecting item 5 to item 0, the edge from item 5 to item 0 is inserted, as illustrated by the dotted edge in Figure 1b. Now consider that after t, u interacts with item 1. Since

Table 1: Table of notation.

Notation	Description
U	Set of users, $U = \{u_1, u_2, ..., u_m\}$
I	Set of items, $I = \{i_1, i_2, ..., i_n\}$
V	Set of nodes, V
E	Set of edges, E
$w(e)$	Weight of edge e
S_u	Ordered list of interactions by user u
li_u	Last item interacted by user u

an edge from item 0 to item 1 exists, its weight must be updated, as denoted in Figure 1c.

We create a weighted directed graph $G = (V, E, w)$, where $V = \{v_1, v_2, ..., v_n\} \subseteq I$ denotes the set of nodes and $E \subseteq V \times V$ denotes the set of edges. Each edge e has an associated weight $w(e) \in R_+$. We define $S_u = \{(v_1, t_1), (v_2, t_2), ..., (v_n, t_n)\}$ as a list of items interacted by user $u \in U$ ordered according to time t. The last interacted item by u is defined as $li_u \in I$, i.e., the last element of S_u. The graph is updated in an incremental manner considering the current observed interaction.

User feedback is modeled as a data stream, where each observation is defined as $< u, i, t >$, indicating that user u interacted with item i at time t, i.e., implicit feedback. When updating the graph considering the current observation, it is desirable that the model is able to include feedback from new incoming users and items, while also updating older concepts. In that sense, there are four possible scenarios, as presented in Algorithm 1:

(1) User and item are unknown by the system ($u \notin U$ and $i \notin I$). In this case, user and item are included in the system, a node for i is added to V, (i, t) is included in S_u and $li_u \leftarrow i$;
(2) User is unknown and item is known by the system ($u \notin U$ and $i \in I$). In this case, user is included in the system, (i, t) is included in S_u and $li_u \leftarrow i$. Given that i is the first interaction of u and i is known, there is no change in the graph;
(3) User is known and item is unknown by the system ($u \in U$ and $i \notin I$). In this case, item is included in the system and a node for i is added into V. In this scenario, u has already interacted with at least one item before. Since this is the first interaction by any user with i, an edge (li_u, i) is included in E, where $w((li_u, i)) = 1$. Lastly, (i, t) is included in S_u and $li_u \leftarrow i$;
(4) User and item are known by the system ($u \in U$ and $i \in I$). In this case, the current sequential interaction is li_u to i. If such interaction has happened before, i.e., $(li_u, i) \in E$, then $w(e)$ is updated according to Equation (1):

$$w((li_u, i)) = w((li_u, i)) \cdot \rho \quad (1)$$

where $\rho \in (0, 1)$ is a parameter that controls the impact of the interaction in the edge weight. If this is the first time such interaction happens, an edge (li_u, i) is included in E, where $w((li_u, i)) = 1$. Lastly, (i, t) is included in S_u and $li_u \leftarrow i$.

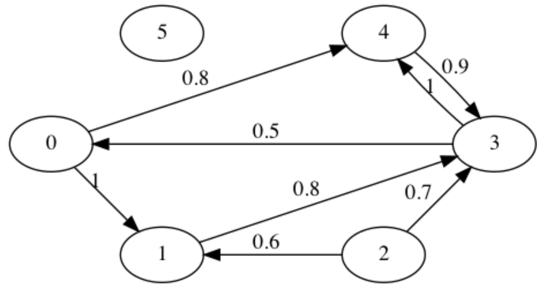
```
D = $< u, i, t >$: data stream;
$li_u$ = last item interacted by user $u$;
**for** $< u, i, t > \in D$ **do**
    **if** $u \in U$ **then**
        **if** $i \in I$ **then**
            **if** $(li_u, i) \in E$ **then**
                $w((li_u, i)) \leftarrow w((li_u, i)) \cdot \rho$;
            **else**
                $E \leftarrow E \cup \{(li_u, i)\}$;
                $w((li_u, i)) \leftarrow 1$;
            **end**
        **else**
            $I \leftarrow I \cup \{i\}$;
            $V \leftarrow I \cup \{i\}$;
            $E \leftarrow E \cup \{(li_u, i)\}$;
        **end**
    **else**
        $U \leftarrow U \cup \{u\}$;
        **if** $i \notin I$ **then**
            $I \leftarrow I \cup \{i\}$;
            $V \leftarrow I \cup \{i\}$;
        **end**
    **end**
    $S_u \leftarrow S_u \cup \{(i, t)\}$;
    $li_u \leftarrow i$;
**end**
```

Algorithm 1: Online graph update

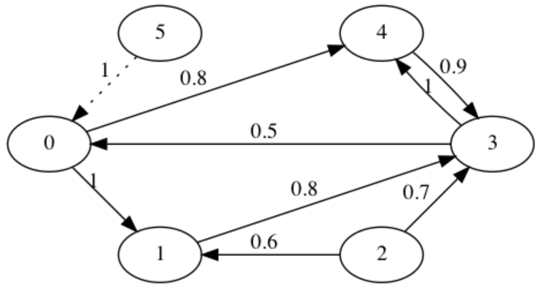
3.2 Recommendation methods

To evaluate the information that is inserted in the graph over time, we tested a few approaches to generate recommendations for users based on items in the list of interactions S. As baseline for comparisons we use the in-degree centrality, that can be seen as a popularity measure. The in-degree centrality captures the number of predecessors that a node has, and is calculated as: $indegree(v) = \frac{|P(v)|}{|V|-1}$, where $P(v)$ is the set of predecessors of node v. A recommendation is generated by calculating the in-degree centrality of all items and then recommending the k items with the highest values. This recommendation method does not distinguish users and simply recommends items with high centrality. We refer to this algorithm as **in-degree**.

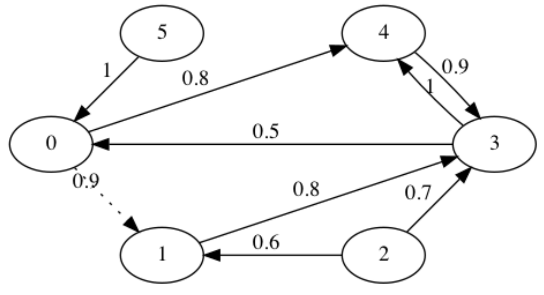
To assess the influence of the most recent interaction on user interest, two strategies based only on the last item interacted by the user u, i.e., li_u were evaluated. In these approaches, candidate items are filtered out as items that are successors to li_u. The first strategy considers the in-degree centrality. To generate recommendations to a user u, we calculate the in-degree centrality for all successors of li_u. The k items with highest values are then recommended to u. We refer to this algorithm as **in-degree_li$_\mathbf{u}$**. The second strategy, **edge_weight_li$_\mathbf{u}$**, considers the weight of edges that connects li_u to its successors as a measure of item relevance. Considering the manner in which the graph is constructed, this approach values the amount of times a sequential interaction happens. Candidate items j are ordered according to the weight of the edge that connects



(a) Example of a graph before updates based on user interactions.



(b) Insertion of edge from item 5 to item 0 based on a sequential interaction.



(c) Weight update of edge from item 0 to item 1 based on a sequential interaction.

Figure 1: An example of graph update based on a few user interactions. Considering the graph in (a) and a user u whose last interaction was with item 5, (b) presents a scenario where u interacts with item 0 by inserting the edge connecting 5 to 0. In (c), u interacts with item 1 after interacting with 0, and the weight of the edge connecting 0 to 1 is updated.

li_u to j, and the k items with the lowest weight are recommended. A similar approach has been shown to be effective in addressing cold-start issues in Trevisiol et al. [20].

To model long-term interest, recommendations to a user u are generated based on the entire list of interactions S_u. Two approaches that filter candidate items as successors to nodes in S_u were tested. The first approach measures the influence of sequential item interaction, where the relevance of candidate items are measured according to the weight of edges, similar to edge_weight_li_u. The value $a(j)$ that represents a candidate item j is the lowest weight between all the edges that connects a node in S_u to j, i.e., $a(j) = min(w((v,j))), \forall v \in S_u, w((v,j)) < 1$. The k items with lowest $a(j)$ are then recommended to u. We refer to this algorithm as **edge_weight_$\mathbf{S_u}$**. The second approach, **path_count_$\mathbf{S_u}$**, considers an item j to be relevant to u based on the number of short paths between items in S_u and j, i.e., an item j is relevant to u if j is successor to several items in S_u. To recommend k items to u, for each candidate item j we associate a value $c(j)$ that counts the amount of predecessors of j in S_u, i.e., $c(j) = \sum_{(v,j)\in E} 1, \forall v \in S_u, w((v,j)) < 1$. The k candidate items with highest $c(j)$ are recommended. We discard information from edges where $w(e) = 1$ since they indicate that the sequential interaction occurred only once.

Assuming that user preferences change over time, we have adapted strategies edge_weight_S_u and path_count_S_u to generate recommendations based on the r most recent user interactions. The resulting approaches, **edge_weight_r** and **path_count_r**, filter candidate items j for a user u as successors to the last r items in the ordered list of interactions S_u. Considering rS_u as the last r items in S_u, edge_weight_r associates for each candidate item j a value $a_r(j) = min(w((v,j))), \forall v \in rS_u, w((v,j)) < 1$ and recommends the k items with the lowest $a_r(j)$. Algorithm path_count_r associates for each candidate item j a value $c_r(j) = \sum_{(v,j)\in E} 1, \forall v \in rS_u, w((v,j)) < 1$ and recommends the k items with highest $c_r(j)$. The impact of parameter r is evaluated through experiments reported in Section 4.4.

4 EXPERIMENTS

In this section we report the experiments performed to evaluate the recommendations generated by the proposed approach, describe the evaluation methodology and discuss the obtained results. We compare the results with ISGD and present an analysis based on the results.

4.1 Evaluation

In order to evaluate the proposed approach on a data stream, a suitable evaluation methodology is required. In that sense, we use the prequential evaluation protocol proposed by Vinagre et al. [21]. For each incoming event $< u, i, t >$, the model is first tested and then updated based on the following steps:

(1) If u is a known user, use the current model to recommend N items to u, otherwise go to step 3;

(2) Score the recommendation list given the observed item i;
(3) Update the model with the observed event;
(4) Proceed to the next observation;

We measure accuracy through the HitRate@N metric at cutoffs of $N \in \{1, 5, 10\}$. HitRate@N returns 1 if item i is within the N first recommended items, and 0 otherwise.

4.2 Datasets

Two datasets from the movie domain were used, as summarized in Table 2. The MovieLens-1M dataset[1] contains around 1.000.000 timestamped ratings in a 1 to 5 scale. The Netflix dataset[2] contains around 100.000.000 timestamped ratings in a 1 to 5 scale. To simulate continuous implicit feedback, we discarded ratings below 5 and sorted events chronologically, where events are defined as $< user, item, time >$. For the Netflix dataset, we dropped users and items with less than 10 interactions, and then selected ratings from 10.000 randomly selected users.

Table 2: Dataset description.

Dataset	Events	Users	Items	Sparsity
MovieLens-1M	226.310	6.014	3.232	98.84%
Netflix	666.178	10.000	5.309	98.75%

4.3 Methodology

We compare the accuracy of the recommendation methods described in Section 3.2 with ISGD [23]. The initial models were built on the first 20% of each dataset, while the remaining 80% were used for incremental evaluation and learning, simulating a data stream. For ISGD, recommendations are generated by estimating the rating of all candidate items, sorting such estimations and selecting N items with the highest values.

Besides HitRate@N to evaluate accuracy, we also measure the average time to update the model and to generate recommendations. As stated in Section 3.1, edge weights are updated based on Equation 1, that updates the weight of an edge based on parameter ρ. In the subsequent experiments, we set $\rho = 0.9$. Applying the same value of ρ for every update does not distinguishing the importance of an interaction and simply decreases the weight based on the number of sequential interactions. All experiments were implemented in Python 2.7, with the NetworkX library [7] for graph manipulation, and executed on an Intel Core i7-4770 of 3.4 GHz with 16 GB RAM running Ubuntu 16.04.

4.4 Results

As discussed in Section 3.2, algorithms edge_weight_r and path_count_r generate recommendations to a user u based on the r most recent interactions in S_u. To that extent, we conducted experiments to evaluate the impact of r in the accuracy of both algorithms with metric HitRate@10. The results of these experiments for both datasets are presented in Figure 2.

[1]https://grouplens.org/datasets/movielens/1m/
[2]https://netflixprize.com/

For the MovieLens-1M dataset in Figure 2a, starting from $r = 20$, we can see that accuracy increases for algorithm edge_weight_r as r decreases until its peak at $r = 3$. For all values of r, edge_weight_r obtained accuracy above 0.11%. We emphasize that $r = 1$ corresponds to algorithm edge_weight_li_u. For the path_count_r algorithm, we can see that accuracy tends to grow as we decrease r until its peak at $r = 5$, and then decreases considerably after $r = 3$. The decrease occurs because it becomes more difficult to distinguish candidate items with less edges to them.

For the Netflix dataset in Figure 2b, starting from $r = 30$, the accuracy for algorithm edge_weight_r slightly increases as r decreases until reaching its peak at $r = 7$. After $r = 7$ the accuracy remains steady until reaching $r = 3$ and then drops considerably. For algorithm path_count_r, accuracy reaches its peak at $r = 13$ and then decreases as r decreases.

Overall, edge_weight_r is relatively stable to r, obtaining similar accuracy for different values of r, while path_count_r is less stable, given that it needs more feedback to distinguish candidate items. We note that the recommendation time is associated with r, since the algorithms iterate the successors to the last r nodes in S_u. Thus, recommendation time can be reduced by lowering r. In that sense, considering that lower values of r can obtain reasonable accuracy while generating faster recommendations, it is interesting to consider the most recent interactions when modeling user behavior. For overall results presented next, for the MovieLens-1M dataset we set $r = 3$ to edge_weight_r and $r = 5$ to path_count_r. For the Netflix dataset we set $r = 7$ and $r = 13$ respectively.

Table 3 presents overall results for all algorithms. Accuracy is measured through HitRate@N with $N \in \{1, 5, 10\}$ and time is measure through average update and recommendation times, with the best results highlighted in bold.

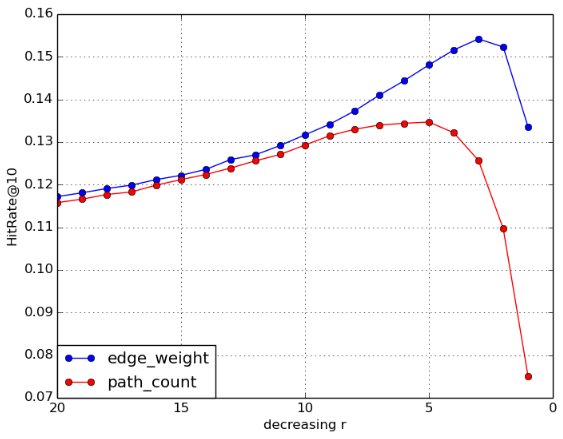
Observing the results presented in Table 3, we can see that edge_weight_r has better accuracy compared to all other methods for both datasets, also being the second fastest method. ISGD is outperformed in accuracy by all graph-based methods, and the two in-degree methods obtained the worst accuracy among the graph-based methods.

Although algorithms edge_weight_S_u and path_count_S_u obtained superior results to the baselines, they have a high recommendation time, since recommendation time is proportional to the size of S_u, which can make these algorithms impractical in some scenarios. However, time is substantially decreased with improved accuracy with its counterparts that only consider the most r recent interactions, i.e., edge_weight_r and path_count_r.

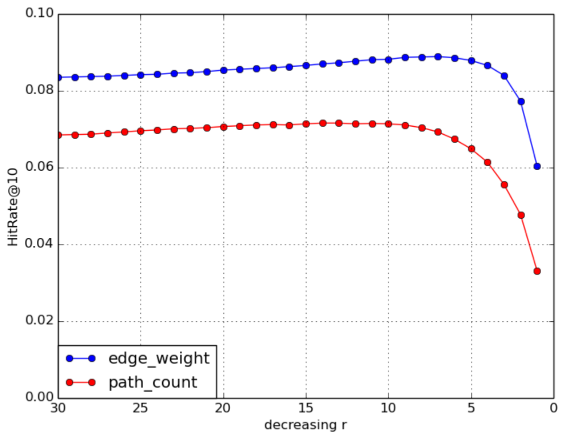
Comparing methods in-degree_li_u and edge_weight_li_u, we can see that edge_weight_li_u outperforms in-degree_li_u both in accuracy and recommendation time, with edge_weight_li_u presenting competitive results for MovieLens-1M, also being the fastest algorithm overall.

For both datasets, the three strategies based on edge_weight are among the four best algorithms, together with path_count_r. These results suggest the potential of including information from sequential interactions into the recommendations.

In terms of update time, both ISGD and graph-based methods achieve competitive results, with graph-update being faster since the update consists in inserting a new edge or updating an edge



(a) MovieLens-1M



(b) Netflix

Figure 2: Impact of parameter r in the accuracy of algorithms edge_weight_r and path_count_r.

Table 3: Results for all algorithms. Accuracy is measure by HitRate@N with $N \in \{1, 5, 10\}$, and time is reported by average update and recommendation times in milliseconds, with the best results highlighted in bold.

Dataset	Algorithm	HitRate@N			Time (ms)	
		1	5	10	Update	Rec.
MovieLens-1M	ISGD	0.007	0.030	0.055	0.06	3.43
	in-degree	0.012	0.046	0.080	**0.01**	1.87
	in-degree_li_u	0.012	0.051	0.093	**0.01**	1.24
	edge_weight_li_u	0.032	0.093	0.134	**0.01**	**0.10**
	edge_weight_S_u	0.017	0.064	0.105	**0.01**	5.00
	path_count_S_u	0.014	0.058	0.102	**0.01**	4.77
	edge_weight_r	**0.033**	**0.101**	**0.154**	**0.01**	0.23
	path_count_r	0.021	0.081	0.135	**0.01**	0.32
Netflix	ISGD	0.002	0.011	0.021	0.06	5.68
	in-degree	0.007	0.026	0.048	**0.02**	4.21
	in-degree_li_u	0.007	0.027	0.047	**0.02**	3.13
	edge_weight_li_u	0.017	0.043	0.061	**0.02**	**0.20**
	edge_weight_S_u	0.018	0.052	0.078	**0.02**	24.96
	path_count_S_u	0.009	0.035	0.061	**0.02**	23.97
	edge_weight_r	**0.021**	**0.059**	**0.089**	**0.02**	1.28
	path_count_r	0.010	0.042	0.072	**0.02**	2.13

weight based on the current interaction. Considering recommendation time, all algorithms but those that generate recommendations based on S_u present acceptable time, since recommendation time is proportional to the number of items. We note that ISGD is more efficient in terms of space complexity since it grows linearly to the number of users and items, while the space complexity for the graph with an adjacency list is $O(|V| + |E|)$.

In Figure 3 we present the accuracy of all algorithms over time with a moving average of the HitRate@10 metric for both datasets with a window of size $n = 5000$ to further evaluate the learning behavior of all algorithms through time. The evolution reinforces that edge_weight_r is superior than other algorithms throughout most of the time for both datasets. For the MovieLens-1M, all algorithms present similar behavior, with similar peaks and a decrease at the end. For both datasets, graph-based methods outperforms ISGD over time.

5 CONCLUSION AND FUTURE WORK

In this work, we proposed an incremental graph of sequential user interactions using implicit feedback, with the assumption that user behavior can be inferred from such sequence of interactions through time. We evaluated the model by recommending items with different strategies on two movie domain datasets and compared results with an incremental matrix factorization algorithm, ISGD, using

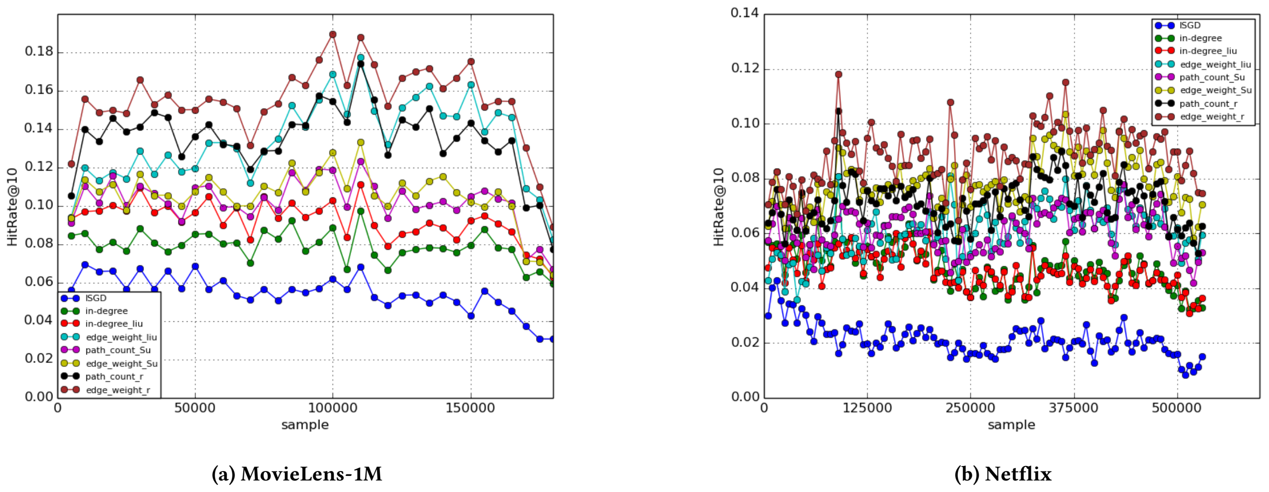


Figure 3: Evolution of HitRate@10 as events arrive for both datasets with window size $n = 5000$.

prequential evaluation. In terms of accuracy, the graph-based methods outperformed ISGD, generally with better recommendation time. The best results were achieved by considering the weight of the edges that connect the r most recent user interactions with the candidate items.

A limitation of the proposed approach is that it requires sufficient data from several users in order to distinguish items based on the sequential interactions. In future work we aim to explore ways to overcome this limitation. Another aspect is how to update the edges according to user sessions. In that sense, future work also include evaluation of dynamic values for ρ, for instance, based on the time elapsed between the interactions of u with li_u and i, considering the similarity between li_u and i and based on the number of interactions made by u. We also intend to include loss of edge relevancy through time, explore different ways of generating recommendations based on the graph, such as including information from both edge_weight and path_count into recommendations and make comparisons with other incremental algorithms, such as item-based K-nearest neighbors [14], RAISGD [24] and CORAISGD [1].

ACKNOWLEDGMENTS

The authors would like to thank CAPES for the financial support.

REFERENCES

[1] Susan C Anyosa, João Vinagre, and Alípio M Jorge. 2018. Incremental matrix co-factorization for recommender systems with implicit feedback. In *Companion Proceedings of the The Web Conference 2018*. 1413–1418.

[2] Shumeet Baluja, Rohan Seth, Dharshi Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. 2008. Video suggestion and discovery for youtube: taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*. 895–904.

[3] Huanhuan Cao, Enhong Chen, Jie Yang, and Hui Xiong. 2009. Enhancing recommender systems under volatile userinterest drifts. In *Proceedings of the 18th ACM conference on Information and knowledge management*. 1257–1266.

[4] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. 2007. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*. 271–280.

[5] Yi Ding and Xue Li. 2005. Time weight collaborative filtering. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 485–492.

[6] Yi Ding, Xue Li, and Maria E Orlowska. 2006. Recency-based collaborative filtering. In *Proceedings of the 17th Australasian Database Conference-Volume 49*. 99–107.

[7] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference*, Gaël Varoquaux, Travis Vaught, and Jarrod Millman (Eds.). Pasadena, CA USA, 11 – 15.

[8] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 447–456.

[9] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[10] Nathan N Liu, Min Zhao, Evan Xiang, and Qiang Yang. 2010. Online evolutionary collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*. 95–102.

[11] Yuting Liu, Bin Gao, Tie-Yan Liu, Ying Zhang, Zhiming Ma, Shuyuan He, and Hang Li. 2008. BrowseRank: letting web users vote for page importance. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 451–458.

[12] Yiqun Liu, Min Zhang, Shaoping Ma, and Liyun Ru. 2009. User Browsing Graph: Structure, Evolution and Application.. In *WSDM (Late Breaking-Results)*.

[13] Pawel Matuszyk, João Vinagre, Myra Spiliopoulou, Alípio Mário Jorge, and João Gama. 2015. Forgetting methods for incremental matrix factorization in recommender systems. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 947–953.

[14] Catarina Miranda and Alípio Mário Jorge. 2009. Item-based and user-based incremental collaborative filtering for web recommendations. In *Portuguese Conference on Artificial Intelligence*. Springer, 673–684.

[15] Olfa Nasraoui, Jeff Cerwinske, Carlos Rojas, and Fabio Gonzalez. 2007. Performance of recommendation systems in dynamic streaming environments. In *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 569–574.

[16] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. 2008. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*. IEEE, 502–511.

[17] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[18] Miguel Sozinho Ramalho, Joao Vinagre, Alípio Mário Jorge, and Rafaela Bastos. 2019. Incremental multi-dimensional recommender systems: co-factorization vs tensors. In *2nd Workshop on Online Recommender Systems and User Modeling*. 21–35.

[19] Yue Shi, Martha Larson, and Alan Hanjalic. 2014. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)* 47, 1 (2014), 3.

[20] Michele Trevisiol, Luca Maria Aiello, Rossano Schifanella, and Alejandro Jaimes. 2014. Cold-start news recommendation with domain-dependent browse graph. In *Proceedings of the 8th ACM Conference on Recommender systems*. 81–88.

[21] João Vinagre, Alípio Jorge, and João Gama. 2014. Evaluation of recommender systems in streaming environments. In *Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design in conjunction with the 8th ACM Conference on Recommender Systems (RecSys 2014)*.

[22] João Vinagre and Alípio Mário Jorge. 2012. Forgetting mechanisms for scalable collaborative filtering. *Journal of the Brazilian Computer Society* 18, 4 (2012), 271–282.

[23] João Vinagre, Alípio Mário Jorge, and João Gama. 2014. Fast incremental matrix factorization for recommendation with positive-only feedback. In *International Conference on User Modeling, Adaptation, and Personalization*. Springer, 459–470.

[24] João Vinagre, Alípio Mário Jorge, and João Gama. 2015. Collaborative filtering with recency-based negative feedback. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 963–965.

[25] João Vinagre, Alípio Mário Jorge, and João Gama. 2015. An overview on the exploitation of time in collaborative filtering. *Wiley interdisciplinary reviews: Data mining and knowledge discovery* 5, 5 (2015), 195–215.

[26] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 723–732.

[27] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. 2014. Lore: Exploiting sequential influence for location recommendations. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 103–112.