Fast incremental matrix factorization for recommendation with positive-only feedback

João Vinagre[1,2], Alípio Mário Jorge[1,2], and João Gama[1,3]

[1] LIAAD - INESC TEC, Porto, Portugal
[2] Faculdade de Ciências, Universidade do Porto, Portugal
[3] Faculdade de Economia, Universidade do Porto, Portugal
joao.m.silva@inescporto.pt, amjorge@fc.up.pt, jgama@fep.up.pt

Abstract. Traditional Collaborative Filtering algorithms for recommendation are designed for stationary data. Likewise, conventional evaluation methodologies are only applicable in offline experiments, where data and models are static. However, in real world systems, user feedback is continuously being generated, at unpredictable rates. One way to deal with this data stream is to perform online model updates as new data points become available. This requires algorithms able to process data at least as fast as it is generated. One other issue is how to evaluate algorithms in such a streaming data environment. In this paper we introduce a simple but fast incremental Matrix Factorization algorithm for positive-only feedback. We also contribute with a prequential evaluation protocol for recommender systems, suitable for streaming data environments. Using this evaluation methodology, we compare our algorithm with other state-of-the-art proposals. Our experiments reveal that despite its simplicity, our algorithm has competitive accuracy, while being significantly faster.

1 Introduction

The purpose of recommender systems is to aid users in the – usually overwhelming – choice of items from a large item collection. Collaborative filtering (CF) is a popular technique to infer unknown user preferences from a set of known user preferences [1]. This set can be conceptually laid out in a user-item matrix, where each cell represents the known preference of a user for an item. CF algorithms analyse this matrix and try to "guess" missing values – typically the vast majority. Recommendations are then produced by extracting the best "guesses" and making them available to the user.

The ultimate task of any recommender system is straightforward: recommend items to users. Depending on the type of feedback users provide, this problem can be formulated in two ways: rating prediction and tem prediction. In a rating prediction problem the main task is to predict missing values in a *numeric* user-item ratings matrix. This is a natural formulation when quantified preferences – ratings – are available, and the problem is most naturally seen as a regression task. However, some systems employ positive-only user preferences.

These systems are quite common – e.g. "like" buttons, personal playlists, shopping history, news reading, website logs. In these cases, numeric ratings are not available. The user-item matrix can be seen as a boolean value matrix, where $true$ values indicate a positive user-item interaction, and $false$ values indicate the absence of such interaction. In systems with positive-only feedback, the task is to predict unknown $true$ values in the ratings matrix – i.e. item prediction –, which can be seen as a classification task. This distinction has implications not only in the algorithms' mechanics but also in evaluation protocols and metrics.

Our work is focused on the item prediction problem, for two main reasons. First, while relatively few online systems have an implemented numeric rating feature, while most systems have some kind of positive-only feedback – e.g. web page visiting logs, customer buying history, click streams, music listening, event participation. Algorithms for rating prediction are not directly applicable in these cases. Second, while the majority of published research focuses on algorithms for rating prediction, some issues related with the specific properties of positive-only feedback remain unexplored. These properties encompass the absence of negative feedback, the inherent subjectivity of implicit ratings and the requirement for specific evaluation measures [2,3].

1.1 Incremental learning

If we look at ratings data in a real world system, it is reasonable to approach it as a data stream: ratings are continuously being generated, and we have no control over the data rate or the ordering of the arrival of new ratings [4]. Ideally, algorithms that learn from data streams maintain predictive models incrementally, requiring a single pass through data.

Research in incremental CF is not abundant in the literature. Incremental neighborhood-based CF is presented in [5] and further studied in our previous work [6,7]. One first solution for incremental matrix factorization is presented in [8], where the authors use the Fold-In method [9] to incrementally update the factor matrices. In [10] an incremental method to update user factors is proposed, by using a simplified version of the batch process. An incremental algorithm for ranking that uses a selective sampling strategy is proposed in [11]. Two incremental methods using Stochastic Gradient Descent (SGD) are evaluated in [12]. Our work differs from the aforementioned for the following. First, we are solving item prediction problems and second, our proposed evaluation methodology is substantially different (see Sec. 5), since we use a prequential approach to measure the algorithms' evolving accuracy.

Starting with a simple iterative SGD algorithm, we adapt its mechanics to work incrementally using only the currently available observation. We show experimental results that indicate that the predictive ability of our proposed incremental algorithm is competitive with state of the art proposals. We also describe our novel experimental setup specifically designed for streaming user feedback.

The remainder of this paper is structured as follows. In Sec. 2 we introduce the basics of modern MF algorithms for CF. Sec. 3 describes an incremental

version of the batch algorithm. Some evaluation issues are discussed in Sec. 4. Results are presented and discussed in Sec. 5. Finally we conclude in Sec. 6.

2 Matrix factorization for CF

Over the last decade several Matrix Factorization (MF) algorithms for CF have been proposed, and were greatly popularized by the Netflix Prize competition [13]. So far, MF methods have proven to be generally superior to other alternatives, in terms of both predictive ability and run-time complexity.

Matrix Factorization for CF is inspired by Latent Semantic Indexing [14], a popular technique to index large collections of text documents, used in the field of information retrieval. LSI performs the Singular Value Decomposition (SVD) of large document-term matrices. In a CF problem, the same technique can be used in the user-item rating matrix, uncovering a latent feature space that is common to both users and items. As an alternative, optimization methods [15,16,17,10] have been proposed to decompose (very) sparse ratings matrices. Supposing we have a ratings matrix R, the algorithms decompose R in two factor matrices A and B that, similarly to a classic SVD, cover a common latent feature space. Matrix A spans the user space, while B spans the item space. Given this formulation, a predicted rating by user u to item i is given by the dot product $\hat{R}_{ui} = A_u.B_i^T$.

Training is performed by minimizing the L_2-regularized squared error for known values in R and the corresponding predicted ratings:

$$\min_{A.,B.} \sum_{(u,i)\in D} (R_{ui} - A_u.B_i^T)^2 + \lambda(||A_u||^2 + ||B_i||^2) \tag{1}$$

In the above equation, D is the set of user-item pairs for which ratings are known and λ is a parameter that controls the amount of regularization. The regularization term $\lambda(||A_u||^2 + ||B_i||^2)$ is used to avoid overfitting. This term penalizes parameters with high magnitudes, that typically lead to overly complex models with low generalization power. The most successful methods to solve this optimization problem are Alternating Least Squares (ALS) [15] and Stochastic Gradient Descent (SGD) [16]. It has been shown [16,17] that SGD based optimization generally performs better than ALS when using sparse data, both in terms of model accuracy and run time performance.

Given a training dataset consisting of tuples in the form $< user, item, rating >$, SGD performs several passes through the dataset – iterations – until some stopping criteria is met – typically a convergence bound and/or a maximum number of iterations. At each iteration, SGD sweeps over all known ratings R_{ui} and updates the corresponding rows A_u and B_i^T, correcting them in the inverse direction of the gradient of the error, by a factor of $\eta \leq 1$ – known as step size or learn rate. For each known rating, the corresponding error is calculated as $err_{ui} = R_{ui} - \hat{R}_{ui}$, and the following update operations are performed:

$$\begin{aligned} A_u &\leftarrow A_u + \eta(err_{ui} B_i - \lambda A_u) \\ B_i &\leftarrow B_i + \eta(err_{ui} A_u - \lambda B_i) \end{aligned} \tag{2}$$

One obvious advantage of SGD is that complexity grows linearly with the number of known ratings in the training set, actually taking advantage of the high sparsity of R.

Algorithm 1 BSGD - Batch SGD

Data: $D = < u, i, r >$
input : $feat$, $iters$, λ, η
output: A, B

init:
  for $u \in$ `Users`(D) **do**
    $A_u \leftarrow$ `Vector`(`size` : $feat$)
    $A_u \sim \mathcal{N}(0, 0.1)$
  for $i \in$ `Items`(D) **do**
    $B_i \leftarrow$ `Vector`(`size` : $feat$)
    $B_i \sim \mathcal{N}(0, 0.1)$

for $count \leftarrow 1$ **to** $iters$ **do**
  $D \leftarrow$ `Shuffle`(D)
  for $< u, i, r > \in D$ **do**
    $err_{ui} \leftarrow r - A_u.B_i^T$
    $A_u \leftarrow A_u + \eta(err_{ui} B_i - \lambda A_u)$
    $B_i \leftarrow B_i + \eta(err_{ui} A_u - \lambda B_i)$

This method – Algorithm 1 – has first been informally proposed in [16] and many extensions have been proposed ever since [17,18,10,19].

3 Incremental Matrix Factorization for item prediction

As stated in Sec. 1 we are focusing on item prediction problems. In item prediction, the ratings matrix R contains either $true$ values – for positively rated items – or $false$ values – for unrated items. One important consideration is that a $false$ value in R may have two distinct meanings: the user either dislikes (negative preference) or did not interact with that item (unknown preference). In our current work, we assume that $false$ values are missing values – as opposed to negative ratings. In practice, this assumption has two main consequences. First, the sparsity of R is maintained because only positive ratings are used for training. Second, all $false$ values for each user are valid recommendation candidates. Another possible approach is to use some criterion to discriminate between negative and missing ratings within the $false$ values. This technique has been shown

to improve accuracy in some cases [3], however it typically requires batch data pre-processing, which is not viable in a data stream environment.

3.1 Incremental SGD

The optimization process of Alg. 1 consists of a batch procedure, requiring several passes – iterations – through the dataset to train a model. While this may be an acceptable overhead in a stationary environment, it is not acceptable for streaming data. As the number of observations increases and is potentially unbounded, repeatedly revisiting all available data eventually becomes too expensive to be performed online.

We propose Alg. 2, designed to work as an incremental process, that updates factor matrices A and B based solely on the current observation. This algorithm, despite its formal similarity with Alg. 1, has two fundamental differences. First, the learning process requires a single pass over the available data. Note that in Alg. 2, at each observation $< u, i >$, the adjustments to factor matrices A and B are made in a single step. One other possible approach is to perform several iterations over each new observation, with potential accuracy improvements, at the cost of the additional time required to re-iterate. Second, no data shuffling – or any other pre-processing – is performed. Given that we are dealing with positive-only feedback we approach the boolean matrix R by assuming the numerical value 1 for *true* values. Accordingly, we measure the error as $err_{ui} = 1 - \hat{R}_{ui}$, and update the rows in A and B^T using the update operations in (2). We refer to this algorithm as ISGD.

Algorithm 2 ISGD - Incremental SGD

Data: $D = \{< u, i >\}$, a finite set or a data stream
input : $feat$, λ, η
output: A, B

for $< u, i > \in D$ **do**
  if $u \notin \texttt{Rows}(A)$ **then**
    $A_u \leftarrow \texttt{Vector}(\texttt{size} : feat)$
    $A_u \sim \mathcal{N}(0, 0.1)$
  if $i \notin \texttt{Rows}(B^T)$ **then**
    $B_i^T \leftarrow \texttt{Vector}(\texttt{size} : feat)$
    $B_i^T \sim \mathcal{N}(0, 0.1)$
  $err_{ui} \leftarrow 1 - A_u.B_i^T$
  $A_u \leftarrow A_u + \eta(err_{ui}B_i - \lambda A_u)$
  $B_i \leftarrow B_i + \eta(err_{ui}A_u - \lambda B_i)$

Since we are mainly interested in predicting good recommendations, we order candidate items i for each user u using the function $f_{ui} = |1 - \hat{R}_{ui}|$, where $\hat{R}_{ui}$ is the non-boolean predicted score. In plain text, we order candidate items by descending proximity to value 1.

4 Evaluation issues

Classic evaluation methodologies for recommender systems begin by splitting the ratings dataset in two subsets – training set and testing set – randomly choosing data elements from the initial dataset. The training set is initially fed to the recommender algorithm to build a predictive model. To evaluate the accuracy of the model, different protocols have been used. Generally, these protocols group the test set by user and "hide" user-item interactions randomly chosen from each group. These hidden interactions form a holdout set. Rating prediction algorithms are usually evaluated by measuring the difference between predicted ratings and hidden ratings. Item recommendation algorithms are evaluated by matching recommended items with hidden items.

The main objective of these protocols is to simulate user behavior in lab conditions. Regarding this, some limitations need to be pointed out:

- Dataset ordering: randomly selecting data for training and test, as well as random hidden set selection, shuffles the natural sequence of data. Algorithms designed to deal with naturally ordered data cannot be rigorously evaluated if data are shuffled;
- Time awareness: shuffling data potentially breaks the logic of time-aware algorithms. For example, by using future ratings to predict past ratings;
- Online updates: incremental CF algorithms perform online updates of their models as new data points become available. This means that neither models or training and test data are static. Models are continuously being readjusted with new data;
- Session grouping: most natural datasets, given their unpredictable ordering, require some pre-processing to group ratings either by user or user session in order to use offline protocols. As data increases in size, it eventually may become too expensive to group data points;
- Recommendation bias: in production systems, user behavior is – expectedly – influenced by recommendations themselves. It is reasonable to assume, for instance, that recommended items will be more likely to be interacted with than if they were not recommended. Simulating this offline usually requires complicated user behavior modeling which can be expensive and prone to systematic error.

These limitations, along with other known issues (see [20]), weaken the assumption that user behavior can be accurately modeled or reproduced in offline experiments. Nevertheless, offline evaluation still provides a useful tool. In [21] some clues are provided on how to solve some of these problems.

5 Evaluation and discussion

Given the problems listed in the previous Section, we propose a prequential approach [22], especially suited for the evaluation of algorithms that deal with data streams. The following steps are performed for each observed event $< u, i >$, representing a positive interaction between user u and item i:

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 event in the dataset;

This protocol provides several benefits:

- It allows continuous online monitoring of the system's predictive ability;
- Online statistics can be integrated in algorithms' logic – e.g. automatic parameter adjustment, drift detection, triggering batch retraining;
- In ensembles, relative weights of individual algorithms can be adjusted;
- The protocol is applicable to both item prediction and rating prediction.

In an offline experimental setting, an overall average of individual scores can be computed at the end – because datasets are in fact finite. For a recommender running in a real-world setting, this process allows us to follow the evolution of the scores throughout the experiment by keeping a statistic of the score. Thereby it is possible to depict how the algorithm's performance evolves over time. In Sec. 5.2 we present both the overall average score and complement it with plots of the evolving score using a simple moving average.

We compare the overall accuracy of four algorithms, using the datasets described in Sec. 5.1. To avoid cold-start issues – which are not the subject of our research – we perform an initial batch training of the algorithms using the first 20% data points in each dataset. The remaining data is naturally used for incremental training. Five algorithms are tested: our incremental SGD algorithm (ISGD), Bayesian Personlized Ranking MF (BPRMF) and its weighted variant WBPRMF [23] and the incremental version of the classic user-based nearest-neighbors algorithm (UKNN), described in [6].

Parameters for all algorithms are independently tuned for each dataset. Because cross-validation is not applicable with streaming data, we use the same evaluation protocol adopted for the presented experiments, but evaluating only on the initial 20% of data of each dataset.

5.1 Datasets

We use four distinct datasets, described in Table 1. All datasets consist of a chronologically ordered set of pairs in the form $< user, item >$. Music-listen and Lastfm-600k consist of music listening events obtained from two distinct sources, where each tuple corresponds to a music track being played by a user. We removed unique occurrences of $< user, item >$ pairs, since these probably do not reflect a positive interaction. Music-playlist consists of a timestamped log of music track additions to personal playlists. MovieLens-1M is well known dataset[4] consisting of timestamped movie ratings in a 1 to 5 rating scale. To use this dataset in an item prediction setting, since we intend to retain only

[4] http://www.grouplens.org, 2003

positive feedback, movie ratings below the maximum rating 5 are excluded. Lastfm-600k consists of the first 8 months of activity observed in the Last.fm[5] dataset originally used in [24]. Both Music-listen and Music-playlist are extracted from the Palco Principal[6] website, a social network dedicated to non-mainstream music enthusiasts and artists.

Dataset	Events	Users	Items	Time frame	Sparsity
Music-listen	335.731	4.768	15.323	12 months	99,90%
Lastfm-600k	493.063	164	65.013	8 months	99,11%
Music-playlist	111.942	10.392	26.117	45 months	99,96%
MovieLens-1M	226.310	6.014	3.232	34 months	98,84%

Table 1. Dataset description

One particular consideration about Music-listen and Lastfm-600k is that they contain repeated events – in music listening, users listen to their favorite music tracks more than once. Most systems do not recommend items to a user that already knows them, except in very specific applications, such as automatic playlist generation. For this reason, we skip the evaluation of these points. However, we do consider them to update the models, in order to better reflect a real world scenario.

5.2 Results and discussion

Table 2 lists overall results, including the average incremental update times. We express accuracy using recall@N at cut-offs $N \in \{1, 5, 10\}$. Recall yields 1 if item i is within the N first recommended items, and 0 otherwise.

From the observation of results, we begin by pointing out that ISGD is from 2 to over 25 times faster – depending on the dataset – than the second faster algorithm, while having comparable accuracy. With Music-playlist and even more with Lastfm-600k, ISGD shows to be superior in both accuracy and processing time. With Music-listen, ISGD's accuracy only falls below the classic UKNN, which is well over 2.000 times slower. The worst relative accuracy obtained by ISGD is with Movielens-1M, performing under all other algorithms. In terms of speed, however, ISGD still performs many times faster than all three alternatives.

One relevant observation is that the relative accuracy between algorithms is highly dependent on the dataset. With Music-listen and Movielens-1M, the neighborhood algorithm is the most accurate, whereas with Lastfm-600k and Music-playlist ISGD performs best. Exclusively regarding MF algorithms, ISGD outperforms the two BPRMF variants with all datasets except Movielens-1M.

[5] http://last.fm
[6] http://www.palcoprincipal.com

Dataset	Algorithm	Recall@1	Recall@5	Recall@10	Update time
Music-listen	BPRMF	0,003	0,016	0,028	0,846 ms
	WBPRMF	0,012	0,037	0,056	1,187 ms
	ISGD	0,017	0,044	0,061	**0,118 ms**
	UserKNN	**0,038**	**0,101**	**0,139**	328,917 ms
Lastfm-600k	BPRMF	<0,001	0,001	0,003	28,061 ms
	WBPRMF	<0,001	0,002	0,003	29,194 ms
	ISGD	**0,012**	**0,027**	**0,034**	**1,106** ms
	UserKNN	0,001	0,004	0,006	290,133 ms
Music-playlist	BPRMF	<0,001	0,009	0,020	1,889 ms
	WBPRMF	0,011	0,038	0,057	2,156 ms
	ISGD	**0,060**	**0,136**	**0,171**	**0,949 ms**
	UserKNN	0,033	0,095	0,132	190,250 ms
Movielens-1M	BPRMF	0,012	0,045	0,080	0,173 ms
	WBPRMF	0,013	0,050	0,084	0,229 ms
	ISGD	0,007	0,028	0,050	**0,016 ms**
	UserKNN	**0,018**	**0,066**	**0,110**	84,927 ms

Table 2. Overall results. Best performing algorithms are highlighted in bold for each dataset. Update times are the average value of the update time for all data points.

Overall results are possible to obtain in offline experiments, given that datasets are finite. However, in an online environment – like production systems – these results can only be interpreted as a snapshot of the algorithms' performance within a predefined time frame. One valuable feature of our adopted evaluation protocol is that it allows the monitoring of the learning process as it evolves over time. To do that, we need to maintain statistics of the outcome of the predictions. We study how the algorithms' accuracy evolves over time by depicting in Fig. 1 a moving average of the recall@10 metric.

The plotted evolution of the algorithms with each dataset generally confirms overall results, however more information becomes available. For instance, although the overall averages of ISGD and UKNN are relatively close with the Music-playlist dataset, Fig. 1 c) shows that these algorithms behave quite differently, starting with a very similar accuracy level and then diverging substantially, starting near the 40.000^{th} data point. With this dataset, all algorithms except ISGD deteriorate over time. With Lastfm-600k and Music-playlist – Figs. 1 b) and c) –, ISGD clearly outperforms the other algorithms. For these datasets besides obtaining better aggregated scores, ISGD seems to take a clear advantage of the incremental learning process, yielding increasingly better results. With Music-listen – Fig. 1 a) – the user-based neighborhood algorithm achieves considerably better scores than all others and ISGD is the second best performing. With Movielens-1M – Fig. 1 d) –, all algorithms share a similar evolving pattern.

We also conducted statistical significance tests between ISGD and the other algorithms for every dataset, using the signed McNemar test over sliding windows [22] of the same size as the ones used for the moving averages used in Fig. 1,

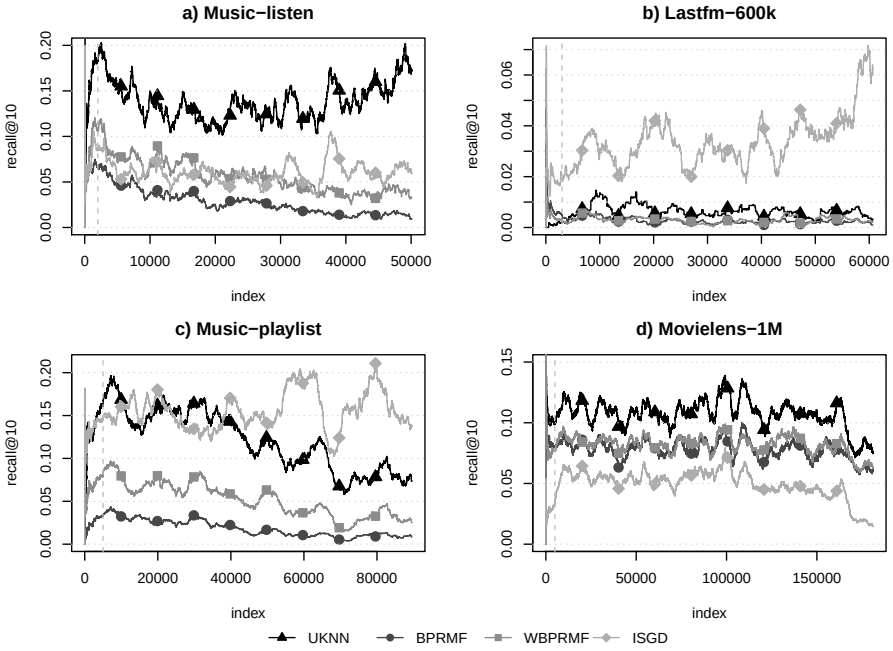

Fig. 1. Evolution of recall@10 with four datasets. The plotted lines correspond to a moving average of the recall@10 obtained for each prediction. The window size n of the moving average is a) $n = 2000$, b) $n = 3000$, c) $n = 5000$ and d) $n = 5000$. The first n points are delimited by the vertical dashed line and are plotted using the accumulated average. Plots a) and b) do not include repeated events in the datasets.

with a significance level of 1%. For the sake of space, we omit the details about these tests since we found that the diferences visible in Fig. 1 are statistically significant for all tested cases except for the following two:

1. ISGD vs WBPRMF with Music-listen – Fig. 1 a). Here significant difference is not detected for over 55% of the experiment. In less than 12% of the datapoints – and only in the first 30.000 –, ISGD is significantly worse than WBPRMF. In more than 28% of the experiment – mainly after the 30.000^{th} data point – ISGD is better than WBPRMF.
2. ISGD vs UKNN with Music-playlist – Fig. 1 c). In the first half of the dataset, ISGD is alternately significantly better, significantly worse or with no significant difference from UKNN. In the second half of Music-playlist, ISGD is significantly better than UKNN.

The online monitoring of the learning process allows a more detailed evaluation of the algorithms' performance. Figure 1 reveals phenomena that would otherwise be hidden in a typical batch evaluation. We consider that this finer grained evaluation process provides a deeper insight into the learning processes of predictive models.

The variability in the results suggests that some characteristics of the datasets – for instance, sparsity, length, user-item ratios or user/item frequency distributions – are somehow determinant in the relative performance of different algorithms. However we were not yet able to correlate meta-data characteristics with the algorithms' ability to produce good models.

6 Conclusion and future work

In this paper, we propose a fast matrix factorization algorithm that is able to deal with a stream of positive-only user feedback. To effectively evaluate CF algorithms in a streaming environment, we also propose a prequential evaluation framework that monitors algorithm accuracy as it continuously learns from a data stream. We use this protocol to compare our algorithm to other incremental algorithms for positive-only feedback. Results suggest that our algorithm, while being faster, has competitive accuracy. We further notice that our evaluation method allows for a finer grained assessment of algorithms, by being able to continuously monitor the outcome of the learning process.

The analysis of results motivates future work towards a better understanding of the effects of dataset properties, such as sparseness, user-item ratios or frequency distributions, in the predictive ability of different algorithms. We are also researching the convergence between the predictive abilities of incremental and batch variants of matrix factorization algorithms.

7 Acknowledgements

Project "NORTE-07-0124-FEDER-000059" is financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the European Regional Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundação para a Ciência e a Tecnologia (FCT). The first author's work is funded by the FCT grant SFRH/BD/77573/2011. The authors wish to thank Ubbin Labs, Lda. for kindly providing data from Palco Principal.

References

1. Goldberg, D., Nichols, D.A., Oki, B.M., Terry, D.B.: Using collaborative filtering to weave an information tapestry. Commun. ACM **35**(12) (1992) 61–70
2. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. [25] 263–272
3. Pan, R., Zhou, Y., Cao, B., Liu, N.N., Lukose, R.M., Scholz, M., Yang, Q.: One-class collaborative filtering. [25] 502–511
4. Domingos, P., Hulten, G.: Catching up with the data: Research issues in mining data streams. In: DMKD '01: Workshop on Research Issues in Data Mining and Knowledge Discovery. (2001)

5. Papagelis, M., Rousidis, I., Plexousakis, D., Theoharopoulos, E.: Incremental collaborative filtering for highly-scalable recommendation algorithms. In Hacid, M.S., Murray, N.V., Ras, Z.W., Tsumoto, S., eds.: ISMIS. Volume 3488 of Lecture Notes in Computer Science., Springer (2005) 553–561
6. Miranda, C., Jorge, A.M.: Incremental collaborative filtering for binary ratings. In: Web Intelligence, IEEE (2008) 389–392
7. Vinagre, J., Jorge, A.M.: Forgetting mechanisms for scalable collaborative filtering. J. Braz. Comp. Soc. **18**(4) (2012) 271–282
8. Sarwar, B.M., Karypis, G., Konstan, J., Riedl, J.: Incremental SVD-based algorithms for highly scalable recommender systems. In: Fifth International Conference on Computer and Information Technology. (2002) 27–28
9. Berry, M., Dumais, S., O'Brien, G.: Using linear algebra for intelligent information retrieval. SIAM review (1995) 573–595
10. Takács, G., Pilászy, I., Németh, B., Tikk, D.: Scalable collaborative filtering approaches for large recommender systems. Journal of Machine Learning Research **10** (2009) 623–656
11. Diaz-Aviles, E., Drumond, L., Schmidt-Thieme, L., Nejdl, W.: Real-time top-n recommendation in social streams. In Cunningham, P., Hurley, N.J., Guy, I., Anand, S.S., eds.: RecSys, ACM (2012) 59–66
12. Ling, G., Yang, H., King, I., Lyu, M.R.: Online learning for collaborative filtering. In: IJCNN, IEEE (2012) 1–8
13. Bennett, J., Lanning, S., Netflix, N.: The netflix prize. In: In KDD Cup and Workshop in conjunction with KDD. (2007)
14. Deerwester, S.C., Dumais, S.T., Landauer, T.K., Furnas, G.W., Harshman, R.A.: Indexing by latent semantic analysis. JASIS **41**(6) (1990) 391–407
15. Bell, R.M., Koren, Y.: Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: ICDM, IEEE Computer Society (2007) 43–52
16. Funk, S.: http://sifter.org/˜simon/journal/20061211.html (2006)
17. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD Cup and Workshop. Volume 2007. (2007) 5–8
18. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Li, Y., Liu, B., Sarawagi, S., eds.: KDD, ACM (2008) 426–434
19. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In Platt, J.C., Koller, D., Singer, Y., Roweis, S.T., eds.: NIPS, MIT Press (2007)
20. Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M.: Controlled experiments on the web: survey and practical guide. Data Min. Knowl. Discov. **18**(1) (2009) 140–181
21. Shani, G., Gunawardana, A.: Evaluating recommendation systems. In Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., eds.: Recommender Systems Handbook. Springer (2011) 257–297
22. Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In IV, J.F.E., Fogelman-Soulié, F., Flach, P.A., Zaki, M.J., eds.: KDD, ACM (2009) 329–338
23. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In Bilmes, J., Ng, A.Y., eds.: UAI, AUAI Press (2009) 452–461
24. Celma, Ò.: Music Recommendation and Discovery - The Long Tail, Long Fail, and Long Play in the Digital Music Space. Springer (2010)
25. Proceedings of the 8th IEEE Intl. Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy. In: ICDM, IEEE Computer Society (2008)