Personalised Dynamic Viewer Profiling for Streamed Data

Bruno Veloso[1], Benedita Malheiro[2], Juan Carlos Burguillo[3], Jeremy Foss[4], and João Gama[5]

[1] LIAAD - INESC TEC, Porto, Portugal
[2] ISEP - Polytechnic Institute of Porto and CRAS - INESC TEC, Porto, Portugal
[3] EET - University of Vigo, Spain
[4] CEBE - Birmingham City University, United Kingdom
[5] LIAAD - INESC TEC and FEP - Universidade do Porto, Porto, Portugal

Abstract Nowadays, not only the number of multimedia resources available is increasing exponentially, but also the crowd-sourced feedback volunteered by viewers generates huge volumes of ratings, likes, shares and posts/reviews. Since the data size involved surpasses human filtering and searching capabilities, there is the need to create and maintain the profiles of viewers and resources to develop recommendation systems to match viewers with resources. In this paper, we propose a personalised viewer profiling technique which creates individual viewer models dynamically. This technique is based on a novel incremental learning algorithm designed for stream data. The results show that our approach outperforms previous approaches, reducing substantially the prediction errors and, thus, increasing the accuracy of the recommendations.

Keywords: On-line Viewer Profiling, Data Stream Mining, Personalisation

1 Introduction

The number of multimedia sources, resources and crowd-sourced feedback data – ratings, likes, shares and posts/reviews – available makes real time processing impossible for humans as well as standard systems. This problem requires the development of dedicated tools, involving viewer profiling and recommendation, to provide viewers with resource suggestions matching their preferences.

The viewer generated data, which corresponds to explicit viewer preferences and intrinsic behaviours, can be used for defining the viewer profiles. In particular, dynamic viewer profiling, *i.e.* the ability to build and update profiles based on the continuous stream of viewer interactions (likes, posts, ratings, watched items, *etc.*), can be addressed as stream mining [4]. Our approach involved four phases: (*i*) data set selection; (*ii*) data set modelling; (*iii*) model update; and (*iv*) model validation. From the several multimedia data sets available on-line, we chose MovieLens 100k (ML 100k) and MovieLens 1M (ML 1M). The idea was to test our approach first with the smaller ML 100k data set and, later, to

use the larger ML 1M data set. The ML 100k data set was prepared and partitioned for the application of Stream Mining techniques. The initial model and the individual hyper-parameters were created using Singular Value Decomposition (SVD) and Stochastic Gradient Descent (SGD), respectively. The model is updated every time a viewer rates a resource. In particular, we update the model of the viewer, using Stochastic Gradient Descent, *i.e.*, update the viewer latent matrix. This incremental viewer profiling methodology is validated by calculating, for the active viewer, the Root Mean Square Error (RMSE), the Recall@10 and Target Recall@10 (TRecall@10) [10] between the recommendations generated by the model and the viewer ratings. Specifically, we apply our algorithm to two scenarios: rating feedback, where a numeric model is built based on all user ratings; and positive feedback, where a binary classification model is created based only on five star user ratings.

In this paper, we propose a viewer-profiling technique using an incremental matrix factorization algorithm designed for stream data. Our proposal creates an initial off-line model by determining for each viewer the optimal learning rate and over-fitting parameters and, then, for each on-line viewer event, updates the model by using SGD to recalculate the viewer latent matrix. This methodology, when compared with the pre-existing approaches referred in Section 2, leads to predictions with increased accuracy.

In terms of organisation, this document contains five sections. Section 2, which is dedicated to stream profiling. Section 3 describes our approach, including the model and the algorithm. Section 4 describes the experiments and discusses the results obtained. Finally, Section 5 draws the conclusions and suggests future developments.

2 Stream Profiling

The following subsections present literature review regarding the creation of off-line and on-line models for data streams, *i.e.*, with and without on-line updating.

2.1 Off-line

Billsus and Pazzani (1998) propose the use of the SVD factorization to create a model and make predictions. The authors convert ratings into booleans and use the resulting binary data to construct the initial model. The approach is evaluated with Precision@N, Recall@N and F-measure@N [2]. This off-line approach, if applied to an on-line data streaming scenario, results in the deterioration of the quality of the predictions with time.

Sarwar, Karypis, Konstan and Riedl (2000) propose the use of SVD factorization to create a reduced model to improve the filter performance. The authors use ratings to construct the initial model and the Mean Absolute Error (MAE) and F-measure [8] evaluation metrics. This proposal provides an off-line methodology to make recommendations and, consequently, in an on-line scenario suffers from the degradation of the accuracy of the recommendations with time.

Barragans, Montenegro, Burguillo, López, Fonte and Peleteiro (2010) proposes a hybrid recommender filter which uses the SVD factorization technique to create a model and dimensionality reduction to improve the filter performance. The authors use ratings to create the initial model and adopt, in terms of evaluation, the MAE [1]. This off-line methodology is inadequate for data streaming.

2.2 On-line

Xiang and Yang (2009) propose a factorized model with four stages: time bias, user bias shifting, item bias shifting and user preference shifting. First, they build the model using SVD matrix factorization and apply the four bias effects (user, item, time and user preference) to optimise the factorized model. Then, they use SGD to update the model and learn over time, determining a global learning rate and over-fitting parameter for all users. Finally, they evaluate the results with RMSE [13]. While this approach updates the global model, our approach updates the individual model.

Gower (2014) explores different recommender models on-line, using the SVD. The author uses learning algorithms and enhances the model with time-based biases [6]. In particular, this approach adopts two bias effects (user and item) to optimise the factorized model and calculates a global learning rate and over-fitting parameter (for all users). Our approach, alternatively, determines individual learning rates and over-fitting parameters.

Vinagre, Jorge and Gama (2014) introduce an incremental matrix factorization algorithm for positive-only feedback and propose a new evaluation methodology called prequential protocol. The matrix factorization and the learning techniques are SVD and SGD, respectively. The prequential protocol verifies, every time a new rating event occurs, if the rated item would have been recommended to that viewer and, if yes, counts as a hit [11]. In 2015, these authors included a rating-and-recency-based scheme to perform negative preference imputation [12]. This scheme creates and maintains a global item queue of size n where the top rated items, *i.e.*, items rated with the maximum rating, are kept at the head and all other items slide to the tail till, eventually, are removed from the queue. Every time a new item is rated, it is inserted in the queue depending on its rating: at the top if it was rated with the maximum rating or immediately after the top rated items, otherwise. When compared with our proposal, this on-line approach uses a global learning rate and over-fitting parameter as well as a global rating-and-recency-based item queue.

Takács, Pilászy, Németh and Tikk (2009) propose several matrix factorization approaches as well as a neighbour selection methodology for matrix factorization models. The authors use bias effects and weights to optimise the factorized model and adopt a global learning rate and over-fitting parameter for all users.

3 Individual Model Update

This paper proposes an individual adaptive algorithm for viewer profiling organised in three main steps: (*i*) creation of the initial off-line model; (*ii*) optimisation of the individual hyper-parameters (learning rate and over-fitting); and (*iii*) online model update with stream learning. Algorithm 1. describes the creation of the initial off-line model, which includes the SVD factorization and singular value dimensionality reduction.

Algorithm 1 Creation of the Initial Off-line Model

1: $R \leftarrow Batch$
2: $R = USV^T$
3: $p_u = U_k\sqrt{S_k}^T$
4: $q_i = \sqrt{S_k}V_k^T$
5: $\hat{r}_{u,i} = q_i^* p_u$

Algorithm 1 builds the initial rating matrix R (line 1) with the viewer ratings, where $r_{u,i}$ represents the rating given by a user u to item i; applies the SVD matrix factorization technique (line 2), where U and V^T are the left-singular and right-singular vectors of R and S is the singular value matrix; creates the latent viewer and item matrices based on the reduced model with size k (lines 3-4) where p_u is the latent user vector and q_i is the latent item vector; and, finally, generates the predictions ($\hat{r}_{u,i}$) using the latent matrices (line 5).

Algorithm 2 implements the individual hyper-parameter optimisation process, which determines the optimal learning rate (γ) and over-fitting (λ) for each viewer in terms of RMSE. These individual parameters are then used to update the user latent matrix.

Algorithm 2 Optimisation of the Individual Hyper-Parameters

1: **for** $\lambda = 0 \rightarrow 1$ **do**
2: **for** $\gamma = 0 \rightarrow 1$ **do**
3: **for** $r_{u,i} \leftarrow CrossValidation$ **do**
4: $R \leftarrow addNewEvent(r_{u,i})$
5: $\hat{r}_{u,i} = q_i^* p_u$
6: $rmse = CalcRMSE(\hat{r}_{u,i}, r_{u,i})$
7: **if** $rmse \leq rmse[user]$ **then**
8: $\gamma[user] \leftarrow \gamma$
9: $\lambda[user] \leftarrow \lambda$
10: $e_{u,i} = r_{u,i} - \hat{r}_{u,i}$
11: $p_u \leftarrow p_u + \gamma(e_{u,i}q_i - \lambda p_u)$

Algorithm 2 determines, for each rating in the cross-validation data, the optimal individual learning rate and the over fitting parameters (lines 1-2), adds

the rating to initial rating matrix (line 4), calculates the RMSE between the predictions and the real ratings (line 5-6). If the RMSE is smaller than the previously calculated RMSE, updates the individual hyper-parameters (lines 7-9) and, finally, updates the viewer latent matrix (the viewer row) using the individual hyper-parameters and the calculated rating error (line 10-11).

Finally, Algorithm 3 illustrates the on-line model update with stream learning. This approach uses the initial off-line cross-validation data, the streamed data and the individual hyper-parameters to update the model.

Algorithm 3 Stream Learning

1: **for** $r_{u,i} \leftarrow DataStream$ **do**
2: $R \leftarrow addNewRating(r_{u,i})$
3: $e_{u,i} = r_{u,i} - \hat{r}_{u,i}$
4: $p_u \leftarrow p_u + \gamma[user](e_{u,i}q_i - \lambda[user]p_u)$
5: $\hat{r}_{u,i} = q_i^* p_u$
6: **return** $\hat{r}_{u,i}$

Algorithm 3 represents the stream learning used with cross-validation data (off-line) to determine the individual hyper-parameters as well as with stream data (on-line). For each new rating, it adds the rating to the rating matrix (line 2), calculates the error between the prediction and the real rating (line 3), updates the model using the individual hyper-parameters and the calculated rating error (line 4) and, finally, generates new predictions using the updated viewer latent matrix (line 5).

4 Experiments and Results

The following subsections present the data set, the evaluation metrics together with the protocol, the experiments and the results obtained.

4.1 Data Set

Our proposal was evaluated with MovieLens 100k (ML 100k) and MovieLens 1M (ML 1M). ML100k data set was chosen due to the lower data sparsity 93.7 % and the size of the data set. It contains information about 943 users and 1682 movies, including 100 000 user ratings together with timestamps. ML 1M data set has higher data sparsity 95.5 % and size. It contains information about 6040 users and 3952 movies, including 1 000 000 user ratings together with timestamps.

4.2 Evaluation Metrics

Regarding the rating feedback, we calculate the incremental error prediction measure (RMSE), which is calculated after each new viewer rating event [9].

Instead of the standard recall metric used by [3,11], which considers a subset of the top rated items, we defined yet another recall based metric – Target Recall (TRecall) – which contemplates a subset of items centered around the target rating. TRecall evaluates the accuracy of the predictions when compared with the actual viewer ratings. For each new viewer rating event, we determine the TRecall@N. First, we predict the ratings of all items unseen by the viewer, including the newly rated item, then we select 1000 unrated items plus the new rated item and sort them in descending order. Finally, if the newly rated item belongs to the list of the top N viewer predicted items centred around the actual viewer rating, we count a hit.

Regarding the positive feedback, we use the Recall proposed by Cremonesi[3], and for each new viewer rating event, we determine the Recall@N. First, we predict the ratings of all items unseen by the viewer, including the new rated item, then we select 1000 unrated items plus the new rated item and sort them in descending order. Finally, if the newly rated item belongs to the list of the top N viewer predicted items, we count a hit. In particular, we calculate the *Recall*@10.

4.3 Evaluation Protocol

The evaluation protocol defines the data ordering, partitions and distribution. First, the data was ordered temporally and, then, partitioned. The "Batch Train", which corresponds to the first 20 % of the ratings made by each user, is used to build the initial model, whereas the remaining 80 % data or "Batch Test" is used for cross-validation. The hyper-parameter optimisation is performed based on the last 10 % of the "Batch Train". The "Stream Data", which corresponds to the remaining 80 % of the data set, is used for model updating. Each one of these ratings triggers the generation and immediate evaluation of the predictions. In particular, when a viewer rates a movie, the algorithm uses the new rating to update the predictions for that user. Finally, the algorithm updates the latent viewer model.

The adopted evaluation method was inspired by the prequential evaluation proposed by Gama et al. (2009) [5]. The predictions are evaluated using RMSE and TRecall@N with rating feedback and Recall@N with positive feedback.

4.4 Experiments

We implemented three main approaches: (*i*) the static approach [2,8,1], which creates the SVD model off-line; (*ii*) the global adaptive approach [13,12,7], which builds the initial SVD model off-line and applies SGD for incremental on-line update; and (*iii*) our individual adaptive approach, which generates the initial SVD model off-line, and uses personalised learning and over-fitting parameters with the on-line SGD incremental update. The three approaches were applied to the rating and positive feedback scenarios.

Rating Feedback Figure 1 presents the $TRecall@10$ accuracy of the three algorithms for the ML 100k. The individual adaptive approach improves dramatically the accuracy when compared with the two other algorithms. The plots represent the $TRecall@10$ after each data event, where higher results are better.

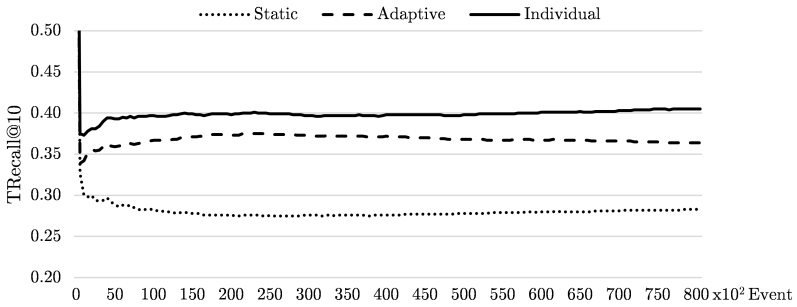

Figure 1. Rating Feedback with ML 100k – Evolution of $TRecall@10$.

Figure 2 presents the $TRecall@10$ accuracy of the three algorithms with ML 1M. The individual adaptive approach improves the accuracy when compared with the static algorithms, however the adaptive approach presents a small advantage when compared with our approach.

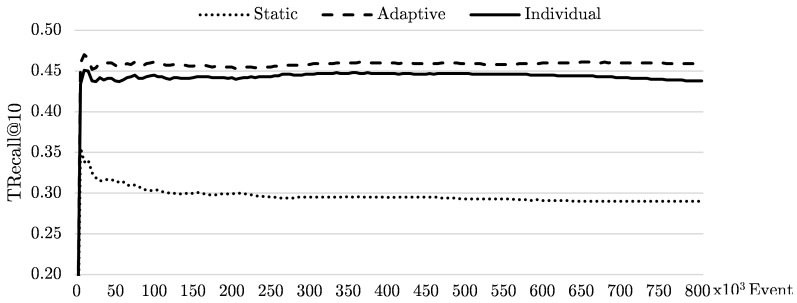

Figure 2. Rating Feedback with ML 1M – Evolution of $TRecall@10$.

Table 1 presents and compares the results of the static, adaptive and individual approaches in terms of $RMSE$ and $TRecall@10$. The individual approach outperforms the static approach in all cases and the adaptive approach in the case of the ML 100k data set. In the case of ML 1M, the individual approach

Table 1. Rating Feedback Results

Data set	Eval. Metric	Static	Adaptive	Individual	Δ_{IS} (%)	Δ_{IA} (%)
ML 100k	$RMSE$	0.228	0.190	<u>0.186</u>	−18	−2
	$TRecall$@10	0.283	0.364	<u>0.405</u>	+43	+11
ML 1M	$RMSE$	0.270	<u>0.200</u>	0.205	−24	+3
	$TRecall$@10	0.290	<u>0.459</u>	0.438	+51	−5

IS – Individual versus Static; IA – Individual versus Adaptive.

produces predictions with errors 3 % higher and a TRecall@10 5 % lower than the adaptive approach.

Positive Feedback Figure 3 presents the *Recall*@10 accuracy of the three algorithms. The individual adaptive approach improves the accuracy when compared with the two other algorithms. The plots represent the *Recall*@10 average after each data event, where higher results are better.

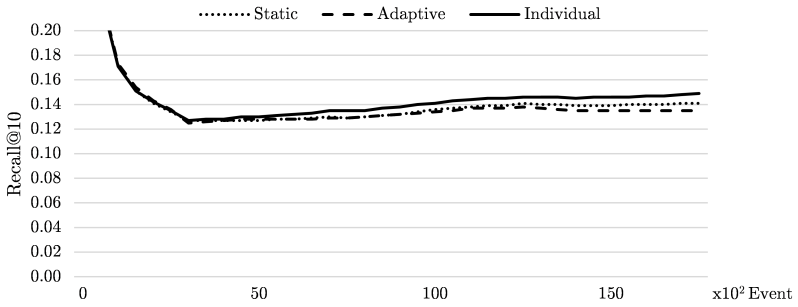

Figure 3. Positive Feedback with ML 100k – Evolution of *Recall*@10.

Figure 4 presents the *Recall*@10 accuracy of the three algorithms. The individual adaptive approach improves dramatically the accuracy when compared with the two other algorithms.

Table 2 displays and compares the results of the static, adaptive and individual approaches in terms of *Recall*@10. These results show the supremacy of

Table 2. Positive Feedback Results

Data set	Eval. Metric	Static	Adaptive	Individual	Δ_{IS} (%)	Δ_{IA} (%)
ML100k	*Recall*@10	0.141	0.135	<u>0.149</u>	+6	+10
ML1M	*Recall*@10	0.051	0.082	<u>0.112</u>	+120	+37

IS – Individual versus Static; IA – Individual versus Adaptive.

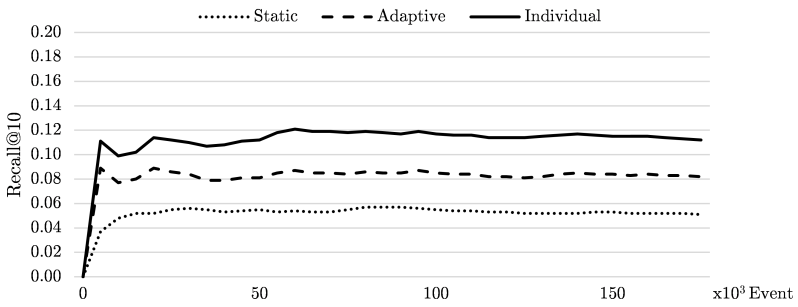

Figure 4. Positive Feedback with ML 1M – Evolution of *Recall*@10.

the individual approach when compared with the previous approaches.

5 Conclusions

This paper describes an individual learning algorithm for updating viewer models. This algorithm refines existing stream mining techniques in order to update individual viewer models based on viewer generated events. The goal of this research was to improve viewer profiling by taking into account the events generated on-line by each viewer. The individual algorithm uses these events to learn the preferences of each viewer, which are subject to external influences as well as personal evolution of interests, as they occur in time.

Our approach uses individual learning rates along the update process to optimise the individual viewer profiles, while keeping the prediction model isolated from the on-line viewer event streams. The results show that our algorithm outperforms the off-line matrix factorization algorithm proposed by Barragáns et al. (2010). Regarding the on-line incremental matrix factorization algorithm of Vinagre et al. (2014), our algorithm shows improved results in the positive feedback scenario and in the rating feedback scenario with the ML 100k; whereas in the case of the rating feedback scenario with the ML 1M, our individual model updating displays a 3 % increase in prediction errors and a 5 % decrease in $TRecall$@10.

As future work and concerning the incremental learning algorithm, we plan to explore forgetting strategies so that old and less relevant events slowly fade into oblivion, making the viewer profile more realistic and accurate. Additionally, we believe that we can improve the results for the larger data set using dynamic personalised learning parameters.

Acknowledgements

This research was carried out in the framework of the project TEC4Growth – RL SMILES –Smart, mobile, Intelligent and Large Scale Sensing and analytics NORTE-01-0145-FEDER-000020 which is financed by the north Portugal

regional operational program (NORTE 2020), under the Portugal 2020 partnership agreement, and through the European regional development fund.

References

1. A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290 – 4311, 2010.
2. D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *Proceedings of the 15th International Conference on Machine Learning*, volume 98 of *ICML '98*, pages 46–54. Morgan Kaufmann Publishers Inc., 1998.
3. P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, pages 39–46. ACM, 2010.
4. J. Gama. *Knowledge Discovery from Data Streams*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, 2010.
5. J. Gama, R. Sebastião, and P. P. Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 329–338. ACM, 2009.
6. S. Gower. Netflix prize and svd. 2014.
7. P. Matuszyk, J. Vinagre, M. Spiliopoulou, A. M. Jorge, and J. Gama. Forgetting methods for incremental matrix factorization in recommender systems. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 947–953, New York, NY, USA, 2015. ACM.
8. B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.
9. G. Takács, I. Pilászy, B. Németh, and D. Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623–656, June 2009.
10. B. Veloso, B. Malheiro, J. C. Burguillo, and J. Foss. Personalised fading for stream data. In *Proceedings of the Symposium on Applied Computing*, SAC '17, pages 870–872, New York, NY, USA, 2017. ACM.
11. J. Vinagre, A. M. Jorge, and J. Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In V. Dimitrova, T. Kuflik, D. Chin, F. Ricci, P. Dolog, and G.-J. Houben, editors, *Proceedings of the 22nd International Conference User Modeling, Adaptation, and Personalization*, UMAP 2014, pages 459–470. Springer International Publishing, Cham, 2014.
12. J. Vinagre, A. M. Jorge, and J. Gama. Collaborative filtering with recency-based negative feedback. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, pages 963–965, New York, NY, USA, 2015. ACM.
13. L. Xiang and Q. Yang. Time-dependent models in collaborative filtering based recommender system. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, WI-IAT '09, pages 450–457, Washington, DC, USA, 2009.