Stream Recommendation using Individual Hyper-Parameters

Bruno Veloso*
REMIT – Research on Economics, Management and Information Technologies, Portucalense University
Porto, Portugal
INESC TEC
Porto, Portugal
bruno.m.veloso@inesctec.pt

Benedita Malheiro
ISEP/IPP – School of Engineering, Polytechnic of Porto
Porto, Portugal
INESC TEC
Porto, Portugal
mbm@isep.ipp.pt

Jeremy D. Foss
DMT Lab – Birmingham City University
Birmingham, United Kingdom
Jeremy.Foss@bcu.ac.uk

ABSTRACT

Nowadays, with the widely usage of on-line stream video platforms, the number of media resources available and the volume of crowd-sourced feedback volunteered by viewers is increasing exponentially. In this scenario, the adoption of recommendation systems allows platforms to match viewers with resources. However, due to the sheer size of the data and the pace of the arriving data, there is the need to adopt stream mining algorithms to build and maintain models of the viewer preferences as well as to make timely personalised recommendations. In this paper, we propose the adoption of optimal individual hyper-parameters to build more accurate dynamic viewer models. First, we use a grid search algorithm to identify the optimal individual hyper-parameters (IHP) and, then, use these hyper-parameters to update incrementally the user model. This technique is based on an 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.

CCS CONCEPTS

• **Information systems** → **Data streams**; **Collaborative filtering**; • **Computing methodologies** → **Online learning settings**.

KEYWORDS

Stream Content Personalisation, Collaborative Filtering, Hyper-Parameter Optimisation

1 INTRODUCTION

The number of multimedia sources, resources and volume of feedback data – ratings, likes, shares and posts/reviews – available on-line are a challenge to standard recommendation algorithms and makes real time processing almost impossible. This problem requires the development of dedicated tools, involving profiling and recommendation, to provide viewers with suggestions matching their preferences in near real time. The user generated data, which corresponds to explicit preferences and intrinsic behaviours, can be used for extracting patterns and define viewer profiles. Dynamic viewer profiling, *i.e.*, the ability to build and maintain profiles based on the continuous stream of viewer interactions (likes, posts, ratings, watched items, *etc.*), can be addressed as stream mining Gama [6].

From the several data sets available on-line, we chose MovieLens 100k (ML 100k), MovieLens 1M (ML 1M), CiaoDVD, Jester, FilmTrust and EachMovie. The data sets were prepared and partitioned for the application of stream mining techniques. The initial model and the updating process is achieved using the Stochastic Gradient Descent (SGD) with the individual hyper-parameters. The model is updated every time a viewer rates a resource, *i.e.*, updates the viewer latent matrix using the SGD. This incremental methodology is validated by calculating, for the active viewer, the Root Mean Square Error (RMSE), the Recall@10 and TRecall@10 between the predictions generated by the model and the actual viewer ratings.

In this paper, we propose the adoption of incremental matrix factorisation algorithm together with individual hyper-parameters. We determine the individual optimal learning rate and regularisation parameters using: (*i*) off-line holdout training and validation to find and evaluate the hyper-parameters; and (*ii*) on-line predictive sequential data stream evaluation protocol to verify the performance of the proposed data stream processing approach. This methodology, when compared with the pre-existing approaches referred in Section 2, leads to predictions with increased accuracy. This work extends the work of [24] with an enriched related work discussion and evaluation, including new data sets, statistical analyses and prequential evaluation.

In terms of organisation, this document contains five sections. Section 2 is dedicated to the optimisation algorithms and on-line recommendation. Section 3 describes our approach, including the grid search and the recommendation algorithm. Section 4 describes the experiments and discusses the results obtained. Finally, Section 5 draws the conclusions and suggests future developments.

2 RELATED WORK

Nowadays, recommendation algorithms are widely used in different domains to suggest items or products, taking in consideration the behaviour and preferences of the active user. There are several different techniques to generate personalised recommendations, namely, content-based and collaborative filtering algorithms. We focus on collaborative filters, more specifically, on incremental model-based algorithms. These algorithms rely on hyper-parameters, which must be tuned according to the data used.

*All authors contributed equally to this research.

DataTV 2019, June 05, 2019, Manchester, UK

Regarding hyper-parameter optimisation algorithms, there are multiple solutions available in the literature. These solutions can be grouped in five categories: (*i*) the gradient descent algorithms, which detect the down-hills of the objective function to find local optima Vorontsov et al. [27] and Shamir and Zhang [19]; (*ii*) the particle swarm algorithms, which identify a set of candidate solutions called particles Poli et al. [15] and Kennedy [11]; (*iii*) the grid search algorithms, which perform a sequential search with progressive increments of the target hyper-parameters Coope and Price [4] and LaValle et al. [12]; (*iv*) random search algorithms, which try with random values until they satisfy a convergence criteria Price [16] and Bergstra and Bengio [2]; and (*v*) direct search algorithms, which apply heuristics to converge to an optimal solution Storn and Price [21] and Audet and Dennis Jr [1].

Vorontsov et al. [27] proposed and applied the gradient descent algorithm, an approach identical to the stochastic parallel gradient descent, to minimise image perturbations. The algorithm was designed to work in parallel to be applied to the multiple control channels provided by camera. Shamir and Zhang [19] presented an extensive study on the SGD convergence properties by applying averaging schemes on the SGD iterations to obtain optimal performances. The authors found that the polynomial-decay averaging improves the results with the same complexity in computational terms as the standard average.

Poli et al. [15] and Kennedy [11] performed an extended overview of particle swarm optimisation algorithms, including the adoption of nearest neighbours' velocity and craziness, cornfield vectors, the elimination of ancillary variables and distance acceleration.

Coope and Price [4] used grid search to follow a given direction of the gradient descent to reduce the search space. This suggestion tries to minimise the number of search iterations required to minimise the objective function, focusing the grid on locations where the gradient descent is higher. LaValle et al. [12] adopted probabilistic road maps which share similarities with grid search algorithms. The authors proposed low discrepancy and low dispersion samples to evaluate the objective function.

Price [16] proposed a controlled random search method to optimise parameters. The constrains are applied at variable level, specifying for each case the upper and lower bounds. This solution is more effective and less computational expensive than the typical random search. Bergstra and Bengio [2] presented random search as a substitute of common optimisation algorithms such as manual or grid search. Grid search algorithms perform an exponential number of irrelevant search trials if they have insufficient resolution to identify a local minimum. The manual search requires experts with confidence and experience to tune the parameters.

Storn and Price [21] described a differential evolution direct search method. This method is composed of three operations: (*i*) mutation, where a vector is generated with different indexes; (*ii*) crossover, which is designed to increase the diversity; and (*iii*) selection of the generated and target vectors. Audet and Dennis Jr [1] proposed a mesh adaptive direct search, which is an extension of the pattern search algorithm. The main difference is that the proposed algorithm is not constrained to a finite number of directions. The proposed algorithm outperforms the pattern algorithm when the number of function evaluations is limited to a fixed number.

In terms of incremental collaborative filtering, we found several proposals in the literature, including contributions designed to minimise problems afflicting collaborative recommendation. It is the case of: (*i*) the addition of new entities, *e.g.*, using folding-in techniques Sarwar et al. [17]; (*ii*) the minimisation of the cold-start effect, *e.g.*, using linear regression Sedhain et al. [18]; (*iii*) the slow decrease of importance of items with time, *e.g.*, using rating or latent factor fading Vinagre et al. [26]; (*iv*) the identification of outliers Gouvert et al. [8]; (*v*) the exploration of additional information, *e.g.*, using extension and sketching techniques of the latent matrices Song et al. [20]; and (*vi*) the imputation of missing data, *e.g.*, based on the popularity of the items He et al. [10] or on explicit user comments Manotumruksa et al. [13] and Zhang et al. [30].

Sarwar et al. [17] adopted the folding-in technique to add new entities to the singular-value decomposition (SVD) model. This technique is less computational expensive than to recompute the SVD every time a new entity arrives. However, the authors do not present a mechanism to update the latent matrices.

Xiang and Yang [29] proposed a factorised model with four stages: time bias and user bias shifting, item bias shifting and user preference shifting. First, they build the model, using SVD matrix factorisation, and apply the four bias effects (user, item, time and user preference) to optimise the factorised 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. Takács et al. [22] proposed several matrix factorisation approaches as well as neighbour selection methodologies for matrix factorisation models. The authors use bias effects and weights to optimise the factorised model and adopt a global learning rate and over-fitting parameter for all users. Gower [9] explored different on-line recommender models using SVD. The author applied learning algorithms to enhance the model with time-based biases. This approach adopts two bias effects (user and item) to optimise the factorised model and calculates a global learning rate and over-fitting parameter (for all users). Vinagre et al. [25] introduced an incremental matrix factorisation algorithm for positive-only feedback and proposed a new evaluation methodology called prequential protocol. The matrix factorisation 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 affirmative, counts as a hit. In 2015, the same authors included a rating-and-recency-based scheme to perform negative preference imputation [26]. 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 they are, eventually, 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.

Song et al. [20] presented an alternative updating method to be applied to factorised matrices. This method has three steps: (*i*) extension of the matrix with auxiliary features; (*ii*) application of a sketching technique to compress the feature matrix; and (*iii*) update of the latent elements based on the difference between the new and old models or between the new and old prediction errors. He et al. [10] exploited popularity to reduce the matrix sparsity and

initialise missing item data. The authors presented the element-wise Alternating Least Squares technique to build a matrix factorisation model with this variable-weighted missing item data. Additionally, the authors presented a refined version of the objective function which classifies as true negative all the initialised items.

Manotumruksa et al. [13] present a solution combining matrix factorisation with word embedding to enhance the final predictions. The user textual reviews are processed to extract semantic information and are put into latent matrices. These semantic latent matrices are combined with the matrix factorisation model to improve the recommendations. Sedhain et al. [18] propose a model which minimises the cold-start effect. The system called LoCo implements: (*i*) a multivariate linear regression to learn the weights that the social components have on the user preferences; (*ii*) a low-rank parameterisation of the linear model weights, to reduce the multitude of social parameters; and (*iii*) a randomised SVD to project the metadata and make predictions. Gouvert et al. [8] describe a solution to deal with outliers. The proposal extends the Poisson matrix factorisation, taking into consideration the model exposure, by multiplying the model by a hyper-parameter. The results show improved performance with dispersed data sets. Zhang et al. [30] describe a hybrid matrix factorisation recommender system. It extracts the user implicit emotional feedback from user reviews, using sentiment analysis, and, then, uses this information to revise the previously rated service. Additionally, the authors extract and incorporate user and service features into a conventional matrix factorisation algorithm to increase the accuracy of the predictions.

Our approach, alternatively, determines optimal individual learning rates and regularisation parameters with the help of grid search. The grid search optimisation algorithm is: (*i*) easy to implement and parallelise; and (*ii*) very reliable in low-dimension search space according to Bergstra and Bengio [2]. Regarding the recommendation algorithm, we extend the work proposed by Takács et al. [22] with the use of individual hyper parameters to maximise the accuracy of the model. The original algorithm implements matrix factorisation to generate the user and item representative latent matrices together with SGD to update the latent factors. This technique has two model parameters: the learning rate to control the speed of the learning process, and the regulator parameter to avoid over-fitting. Finally, the predictions are computed through the dot product between the corresponding latent vectors of each latent matrix. The specific version adopted in this work corresponds to the biased regularised incremental simultaneous matrix factorisation [22]. Our extension explores the usage of individual parameters for each user based on the assumption that each user has a different learning behaviour.

3 PROPOSED METHOD

Our proposal performs a grid search algorithm to select the best individual hyper-parameters. We implemented the biased regularised incremental simultaneous matrix factorisation to update sequentially the model. The search fulfils two conditions: (*i*) the global output metrics with the found hyper-parameters improve the baseline results; and (*ii*) the optimal individual hyper-parameters are selected in terms of RMSE. Algorithm 1 implements the individual hyper-parameter (IHP) optimisation process, using the initial 50 % of the data (train partition). It determines the optimal learning rate (η) and regularisation parameter (λ) for each user in terms of RMSE. These individual parameters are then used to update the user latent matrix. For each new stream rating (line 3), it adds the new rating to the rating matrix (line 4), where $r_{u,i}$ represents the rating given by a user u to item i, and calculates the individual and global RMSE between the prediction ($\hat{r}_{u,i}$) and the real rating (lines 5-10). Next, it updates the viewer latent matrix (the viewer row) (line 11), using the hyper-parameters found by the grid search and the calculated rating error (line 6), where $\vec{p_u}$ is the latent user vector and $\vec{q_i}$ is the latent item vector. Finally, if both global and individual RMSE have decreased (lines 12 and 13), it updates the user's hyper-parameters (lines 14-15). Line 16 increments the total number of events.

Algorithm 1 Optimisation of the Individual Hyper-Parameters

1: **for** $\lambda = 0 \rightarrow 1$ with increments of 0.01 **do**
2: **for** $\eta = 0 \rightarrow 1$ with increments of 0.01 **do**
3: **for** $r_{u,i} \leftarrow Train$ **do**
4: $r \leftarrow addNewEvent(r_{u,i})$
5: $\hat{r}_{u,i} = \vec{q_i} \cdot \vec{p_u}$
6: $e_{u,i} = (r_{u,i} - \hat{r}_{u,i})^2$
7: $e = e + e_{u,i}$
8: $rmse = \sqrt{\frac{e}{n}}$
9: $e_u = e_u + e_{u,i}$
10: $rmse_u = \sqrt{\frac{e_u}{n}}$
11: $\vec{p_u} \leftarrow \vec{p_u} + \eta(e_{u,i}\vec{q_i} - \lambda\vec{p_u})$
12: **if** $rmse \leq rmse_{old}$ **then**
13: **if** $rmse \leq rmse_{u,old}$ **then**
14: $\eta_u \leftarrow \eta$
15: $\lambda_u \leftarrow \lambda$
16: $n = n + 1$

Algorithm 2 illustrates the on-line model update with stream learning, using the remaining 50 % of the data. For each stream rating (line 1), it adds the rating to the rating matrix R (line 2), calculates the RMSE between the predicted and the real rating (lines 3-6). Finally, it updates the user latent matrix (the user row), using the optimal individual hyper-parameters and the calculated rating error (line 7). The variables represent the real rating ($r_{u,i}$), the prediction ($\hat{r}_{u,i}$), the latent user vector ($\vec{p_u}$), the latent item vector ($\vec{q_i}$), the user learning rate (η_u), the user regularisation parameter (λ_u), the accumulated error (e), the prediction error for the user u and item i ($e_{u,i}$), the root mean square error ($rmse$) and the total number of events (n).

Algorithm 2 Stream learning algorithm

1: **for** $r_{u,i} \leftarrow Stream$ **do**
2: $R \leftarrow addNewRating(r_{u,i})$
3: $\hat{r}_{u,i} = \vec{q_i} \cdot \vec{p_u}$
4: $e_{u,i} = (r_{u,i} - \hat{r}_{u,i})^2$
5: $e = e + e_{u,i}$
6: $rmse = \sqrt{\frac{e}{n}}$
7: $\vec{p_u} \leftarrow \vec{p_u} + \eta_u(e_{u,i}\vec{q_i} - \lambda_u\vec{p_u})$

4 EXPERIMENTS AND RESULTS

The following subsections present the data sets, the evaluation metrics together with the protocol, the experiments, the results obtained and a final discussion. The experiments were performed with an Intel Xeon CPU E5-2680 2.40 GHz Central Processing Unit (CPU), 32 GiB DDR3 Random Access Memory (RAM) and 1 TiB of hard drive platform running the Ubuntu 16.04.

4.1 Data Sets

Our proposal was evaluated with six different data sets: (*i*) MovieLens 100k (ML 100k) contains information about 943 users and 1682 movies, including 100 000 user ratings together with time stamps and has a data sparsity of 93.70 %; (*ii*) MovieLens 1M (ML 1M) has higher data sparsity 95.5 % and holds information about 6040 users and 3952 movies, including 1 000 000 user ratings together with time stamps; (*iii*) CiaoDVD has a data sparsity of 99.97 % and contains information about 17 615 users and 16 121 movies, including 72 665 user ratings together with time stamps; (*iv*) FilmTrust has a data sparsity of 98.86 % and holds information about 1508 users and 2071 movies, including 35 497 user ratings together with time stamps; (*v*) Jester has a data sparsity of 80.83 % and contains information about 59 132 users and 150 items, including 1 700 000 user ratings; (*vi*) EachMovie data set has a data sparsity of 97.63 % and holds information about 72 916 users and 1628 movies, including 2 811 983 user ratings together with time stamps.

Our selection of the data sets was based, first, on the type of the data, *i.e.*, we chose media-related data sets, and, next on the data sizes, *i.e.*, with diverse sizes. Different data sizes allow us to verify if the grid search algorithm finds hyper-parameter values which, in fact, improve the recommendation results.

4.2 Evaluation Metrics

Regarding the evaluation metrics, we calculate the incremental error prediction measure (RMSE), which is calculated after each new rating event Takács et al. [22]. In terms of accuracy metrics, we adopt the standard recall metric used by Cremonesi et al. [5], which considers a subset of the top-rated items and additionally we adopt another recall-based metric called – Target Recall (TRecall) – used by Veloso et al. [23] which contemplates a subset of items centred around the target rating. For each new rating event, we determine the Recall@N and the TRecall@N. First, we predict the ratings of all items unseen by the viewer, including the newly rated item, and 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 on the actual viewer rating, we count a hit for the TRecall and if the item belongs to the top N items we count a hit for the Recall.

4.3 Evaluation Protocol

The evaluation protocol defines the data ordering, partitions and distribution. In this work we adopt the holdout and the prequential protocol. The data was ordered temporally and, then, partitioned according to the selected protocol.

In the holdout protocol the "Train" subset is used to build the initial model and to optimise the hyper-parameters, whereas the remaining data or "Test" subset is used for the validation. Since in a real streaming environment it is impossible to anticipate the number events of each user, the training process is initiated with a predefined number of events and generic hyper-parameters. To guarantee that there are sufficient events to train each individual hyper-parameter, we ordered the data temporally and ensure that the train phase holds 50 % of the ratings, leaving the remaining 50 % of the ratings for validation. We adopt the holdout protocol, which is a simple version of the cross validation, to determine the general performance of the model. The main drawback of holdout when compared with cross-validation is that it presents higher variance Blum et al. [3].

In the case of the prequential protocol proposed by Gama et al. [7], all the data it is used to build, maintain and assess the algorithm performance. The hyper-parameters used in this step are the same used with holdout. Each one of data ratings triggers the generation and immediate evaluation of the predictions as well as the model updating in both evaluation protocols. We choose the prequential protocol because it is the unique approach that measures the learning process. This method uses the predictive sequential error estimated over a sliding window to verify the model reaction to non-stationary environments.

4.4 Significance Tests

To detect the statistical differences between the proposed and the baseline approaches we applied two different significance tests: (*i*) the Wilcoxon test Wilcoxon [28] to verify if the mean ranks of two samples differ; and (*ii*) the McNemar test McNemar [14] to assess if a statistically significant change occurs on a dichotomous trait at two time points on the same population. We define a 5 % of significance level for all tests. The goal of the Wilcoxon and McNemar tests is to reject the null-hypothesis, *i.e.*, that both approaches have the same performance. For a significance level (p) of 0.05, the value of McNemar test (M) is 3.84 and the value of the Wilcoxon test (W) is 137.

4.5 Experiments

To assess the proposed solution, the algorithm executed thirty times with each data set to compute the mean and the standard deviation of the performance metrics. The baseline algorithm corresponds to the biased regularised incremental simultaneous matrix factorisation algorithm proposed by Takács et al. [22] with global hyper-parameters. Table 2 presents the results of the holdout evaluation protocol with the best results of each data set highlighted in bold. In the case of ML 100k, the RMSE decreases 2.0 % and the Recall and TRecall increases 79.8 % and 30.0 %, respectively. The results for the ML 1M show that the RMSE decreases 49.2 %, whereas the Recall and TRecall increase 128.4 % and 126.8 %, respectively. For the Jester data set, the RMSE increases 3.5 % and Recall and TRecall improve 3.0 % and 2.5 %. In the case of FilmTrust, the RMSE increases 22.8 % and the Recall and TRecall decreases 73.6 % and 62.7 %, respectively. For the EachMovie data set, the RMSE decreases 34.3 % and the Recall and TRecall improve 214.5 % and 109.6 %. Finally, for the CiaoDVD, the RMSE inflates 3.7 % and Recall and TRecall improve 71.3 % and 35.4 %.

Table 1: Data set characteristics

Data set	Users	Items	Ratings	Ratings per User			Time Period
				Avg	Max	Min	
EachMovie	72 916	1628	2 811 983	45	1455	1	09/01/95 – 15/09/97
ML 100k	943	1682	100 000	106	737	20	19/09/97 – 22/04/98
ML 1M	6040	3952	1 000 000	165	2314	20	26/04/00 – 28/02/03
Jester	59 132	150	1 700 000	29	140	1	01/11/06 – 30/04/09
FilmTrust	1508	2071	35 497	23	244	1	01/01/12 – 31/12/12
CiaoDVD	17 615	16 121	72 665	4	1106	1	01/11/13 – 31/12/13

Table 2: Holdout evaluation results with 30 runs

Data set	Evaluation metric	Baseline	IHP
ML 100k	RMSE	$2.03 \times 10^{-1} \pm 6.55 \times 10^{-4}$	$\mathbf{1.99 \times 10^{-1} \pm 2.34 \times 10^{-4}}$
	Recall@10	$5.49 \times 10^{-3} \pm 2.38 \times 10^{-4}$	$\mathbf{9.87 \times 10^{-3} \pm 6.31 \times 10^{-5}}$
	TRecall@10	$7.09 \times 10^{-3} \pm 2.60 \times 10^{-4}$	$\mathbf{9.22 \times 10^{-3} \pm 2.08 \times 10^{-4}}$
	Recommendation Time	$7.38 \times 10^{-4} \pm 1.01 \times 10^{-4}$	$\mathbf{7.77 \times 10^{-5} \pm 8.37 \times 10^{-5}}$
ML 1M	RMSE	$3.84 \times 10^{-1} \pm 1.49 \times 10^{-3}$	$\mathbf{1.95 \times 10^{-1} \pm 8.06 \times 10^{-5}}$
	Recall@10	$5.78 \times 10^{-3} \pm 1.69 \times 10^{-3}$	$\mathbf{1.32 \times 10^{-2} \pm 3.31 \times 10^{-3}}$
	TRecall@10	$5.51 \times 10^{-3} \pm 1.30 \times 10^{-3}$	$\mathbf{1.25 \times 10^{-2} \pm 2.21 \times 10^{-3}}$
	Recommendation Time	$6.34 \times 10^{-4} \pm 8.30 \times 10^{-5}$	$\mathbf{1.29 \times 10^{-4} \pm 1.33 \times 10^{-4}}$
EachMovie	RMSE	$4.34 \times 10^{-1} \pm 4.52 \times 10^{-3}$	$\mathbf{2.85 \times 10^{-1} \pm 1.20 \times 10^{-3}}$
	Recall@10	$3.72 \times 10^{-3} \pm 2.11 \times 10^{-3}$	$\mathbf{1.17 \times 10^{-2} \pm 2.16 \times 10^{-3}}$
	TRecall@10	$6.49 \times 10^{-3} \pm 1.10 \times 10^{-3}$	$\mathbf{1.36 \times 10^{-2} \pm 1.17 \times 10^{-3}}$
	Recommendation Time	$4.71 \times 10^{-4} \pm 5.70 \times 10^{-5}$	$\mathbf{9.77 \times 10^{-5} \pm 9.98 \times 10^{-5}}$
Jester	RMSE	$\mathbf{2.00 \times 10^{-1} \pm 2.57 \times 10^{-3}}$	$2.07 \times 10^{-1} \pm 5.29 \times 10^{-4}$
	Recall@10	$1.66 \times 10^{-1} \pm 1.06 \times 10^{-2}$	$\mathbf{1.71 \times 10^{-1} \pm 5.40 \times 10^{-3}}$
	TRecall@10	$1.98 \times 10^{-1} \pm 1.88 \times 10^{-2}$	$\mathbf{2.03 \times 10^{-1} \pm 2.02 \times 10^{-2}}$
	Recommendation Time	$6.27 \times 10^{-5} \pm 7.20 \times 10^{-6}$	$\mathbf{6.77 \times 10^{-6} \pm 7.65 \times 10^{-6}}$
FilmTrust	RMSE	$\mathbf{1.62 \times 10^{-1} \pm 9.32 \times 10^{-4}}$	$1.99 \times 10^{-1} \pm 8.54 \times 10^{-3}$
	Recall@10	$\mathbf{2.01 \times 10^{-2} \pm 4.67 \times 10^{-4}}$	$5.30 \times 10^{-3} \pm 9.80 \times 10^{-5}$
	TRecall@10	$\mathbf{1.57 \times 10^{-2} \pm 1.18 \times 10^{-3}}$	$5.86 \times 10^{-3} \pm 1.91 \times 10^{-4}$
	Recommendation Time	$1.00 \times 10^{-3} \pm 2.55 \times 10^{-4}$	$\mathbf{9.44 \times 10^{-5} \pm 9.07 \times 10^{-5}}$
CiaoDVD	RMSE	$\mathbf{1.89 \times 10^{-1} \pm 7.64 \times 10^{-3}}$	$1.96 \times 10^{-1} \pm 5.34 \times 10^{-3}$
	Recall@10	$7.88 \times 10^{-3} \pm 3.32 \times 10^{-4}$	$\mathbf{1.35 \times 10^{-2} \pm 9.22 \times 10^{-5}}$
	TRecall@10	$9.16 \times 10^{-3} \pm 5.09 \times 10^{-4}$	$\mathbf{1.24 \times 10^{-2} \pm 3.78 \times 10^{-4}}$
	Recommendation Time	$3.28 \times 10^{-3} \pm 3.87 \times 10^{-4}$	$\mathbf{3.37 \times 10^{-4} \pm 3.48 \times 10^{-4}}$

The holdout evaluation shows that IHP improves the Recall-based results of five data sets (the exception is FilmTrust) and decreases the prediction errors of three data sets (EachMovie, ML 1M and ML 100k), including two of the largest data sets. The statistical results for the Wilcoxon and McNemar tests, displayed in Table 3, reject the null hypothesis for all data sets.

Figure 1 presents the prequential results for all data sets with a sliding window of 1000 events. The plots represent the logarithmic ratio between the predictive sequential error of the baseline (with global hyper-parameters) and the IHP algorithms. Positive values mean that IHP is more accurate than the baseline algorithm. We can observe with all data sets that IHP decreases significantly the prediction errors in the streaming scenario. Additionally, the prequential evaluation plots display the behaviour of the learning process over time with non-monotonic data streams. EachMovie, the larger data set with more than 2.8 million events and a data sparsity of 97.63 %, shows a continuous decrease in performance, indicating unstable user behaviour over time. This behaviour exposes a concept drift problem, which is difficult to detect in recommendation systems. Jester, the second larger data set data set with 1.7 million events and a sparsity of 80.83 %, shows no concept drift signs, suggesting that its lower sparsity contributes to a stabler performance.

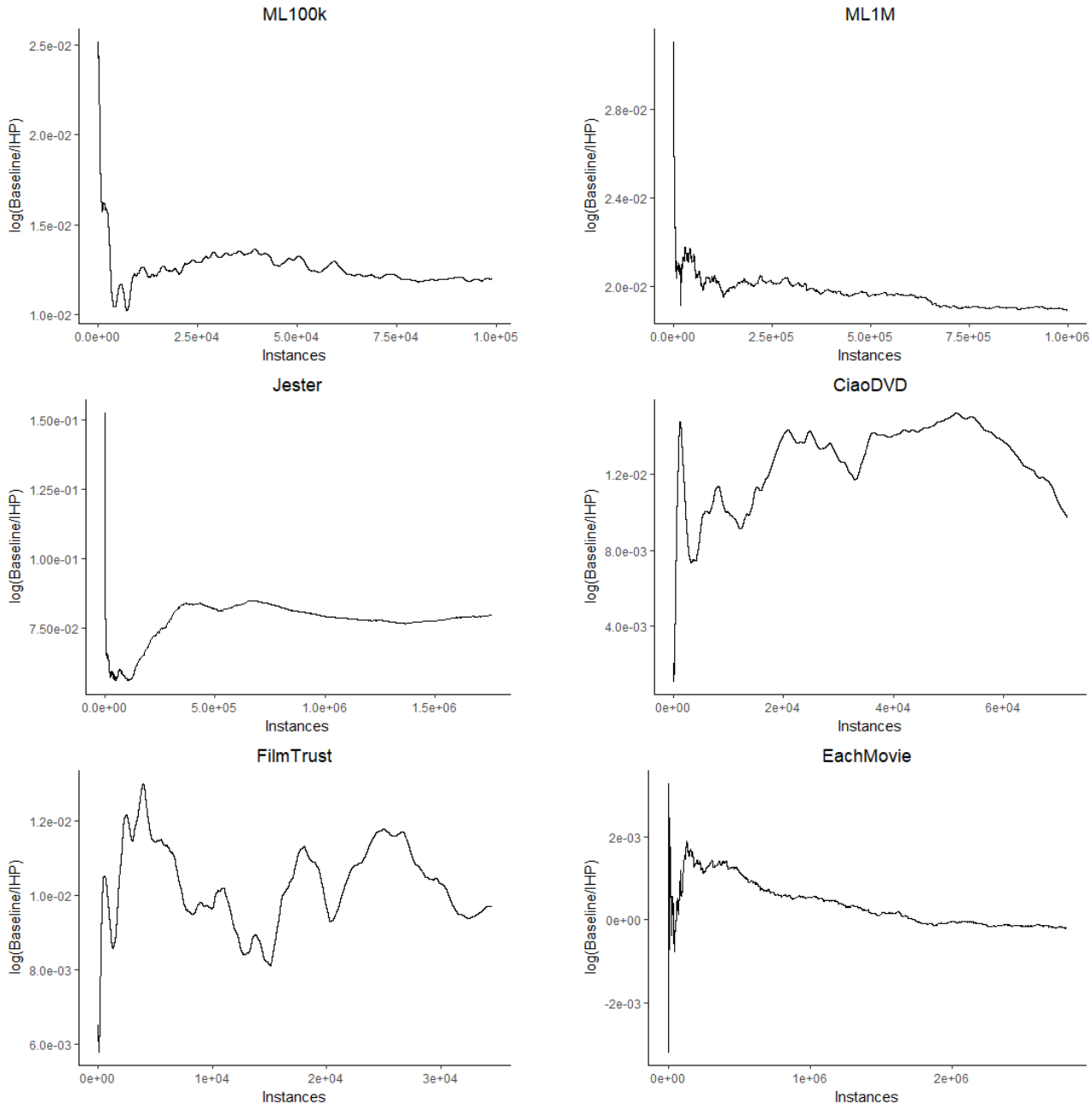

Figure 1: Prequential evaluation with a sliding window of 1000 events

4.6 Discussion

The proposed algorithm is designed to optimise the individual hyper-parameters of model-based collaborative filters by minimising the RMSE. However, to find the best individual hyper-parameters, it requires a significant number of rating events per user. This limitation is observable in the holdout evaluation (Table 3) where the three data sets with lower number of average events per user (Jester, FilmTrust and CiaoDVD in Table 1 perform worst with individual hyper-parameters than with the baseline approach. Specifically, the Jester, FilmTrust and CiaoDVD data sets have less than 30 average events per user and the minimal number of events per user is 1. Since we use 50 % of the user data to select the individual hyper-parameters, this means that, in the case of these data sets, the average number of events available to perform this task is less than 15. In terms of Recall, our algorithm surpasses the baseline with all data sets except with the smallest data set (FilmTrust), showing considerable immunity to the problem of low number of

Table 3: Statistical results

Data set	Test	Value	p-value (#)
ML 100k	McNemar (M)	9.6	1.91×10^{-3}
	Wilcoxon (W)	45.0	3.05×10^{-5}
ML 1M	McNemar (M)	28.0	1.19×10^{-7}
	Wilcoxon (W)	0.0	1.86×10^{-9}
EachMovie	McNemar (M)	14.7	1.26×10^{-4}
	Wilcoxon (W)	14.0	2.05×10^{-7}
Jester	McNemar (M)	28.0	1.19×10^{-7}
	Wilcoxon (W)	0.0	1.86×10^{-9}
FilmTrust	McNemar (M)	28.0	1.19×10^{-7}
	Wilcoxon (W)	0.0	1.86×10^{-9}
CiaoDVD	McNemar (M)	28.0	1.19×10^{-7}
	Wilcoxon (W)	0.0	1.86×10^{-9}

training events. The holdout experiments were repeated 30 times to compute the average and standard deviation of the metrics used.

The prequential evaluation protocol assesses the behaviour of the algorithm in streaming conditions. Figure 1 presents the difference between the baseline and IHP algorithms. The results show that IHP reduces the predictive errors with all data sets. This is due to the implemented search mechanism which, firstly, tries to optimise the global predictive error and, then, minimise the individual prediction errors. Finally, we made a set of statistical tests to verify if our algorithm is statistically different from the baseline algorithm. According to Table 3, we can reject the null hypothesis.

We can state, based on the obtained results, that the proposed algorithm represents the user behaviour more accurately, modelling users with distinct behaviours with different individual learning rate and regularisation parameters. Moreover, it requires an average of 23 events per user for the convergence of the two individual hyper-parameters and to outperform the baseline results with both evaluation protocols (holdout and prequential). Concept drift is expected to occur with large (size), long (time) and sparse data sets and, naturally, with data streams. To overcome this problem, it is necessary to readjust the individual hyper-parameters. However, concept drift detection is not a trivial task for recommendation systems since it is very difficult to detect significant data variations when user feedback can be highly variable, imitating the behaviour of outliers. A possible solution is to recalculate regularly the individual hyper-parameters. Regarding the state of the art algorithms, our algorithm is, as far as we know, the first to adopt independent user hyper-parameters to improve the final recommendations.

5 CONCLUSIONS

This paper describes an individual learning algorithm for updating user models. This algorithm refines existing stream mining techniques by building and updating individual user models based on the user stream of events. The goal of this research is to improve real time user profiling by considering the continuous stream of on-line user events. The algorithm uses these events to learn the preferences of each user, which are subject to external influences as well as to personal evolution of interests, as they occur in time.

Our approach applies the individual learning rate and regularisation parameters to build and update the individual user profiles, while keeping the prediction model isolated from the user event streams. Regarding the on-line incremental matrix factorisation algorithm of Vinagre et al. [25], our algorithm displays for all data sets improved recall results; in terms of RMSE, our approach has better results with the data sets with an average of at least 45 events per user.

As future work, concerning the incremental learning algorithm, we plan to explore forgetting strategies so that old and less relevant events slowly fade into oblivion, making the user profile more realistic and accurate. Further exploration it is necessary to adjust dynamically the individual hyper-parameters throughout time. The decision to recompute the individual hyper-parameters can be based on the detection of concept drifts since our results indicate that larger data sets display concept drift effects. Finally, to minimise the number of computational resources necessary, older inactive users can be deactivated.

ACKNOWLEDGMENTS

This work was partially financed by National Funds through the Portuguese funding agency, FCT – Fundação para a Ciência e Tecnologia, within project UID/EEA/50014/2019.

REFERENCES

[1] Charles Audet and John E Dennis Jr. 2006. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on optimization* 17, 1 (2006), 188–217.
[2] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.
[3] Avrim Blum, Adam Kalai, and John Langford. 1999. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *COLT*, Vol. 99. ACM, New York, NY, USA, 203–208.
[4] Ian D Coope and Christopher John Price. 2001. On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization* 11, 4 (2001), 859–869.
[5] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, New York, NY, USA, 39–46.
[6] Joao Gama. 2010. *Knowledge discovery from data streams*. Chapman and Hall/CRC, Boca Raton, FL, USA.
[7] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. 2009. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, New York, NY, USA, 329–338.
[8] Olivier Gouvert, Thomas Oberlin, and Cédric Févotte. 2018. Negative Binomial Matrix Factorization for Recommender Systems. *arXiv preprint arXiv:1801.01708* (2018).
[9] Stephen Gower. 2014. Netflix prize and SVD.
[10] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, New York, NY, USA, 549–558.
[11] James Kennedy. 2010. Particle swarm optimization. *Encyclopedia of machine learning* (2010), 760–766.
[12] Steven M LaValle, Michael S Branicky, and Stephen R Lindemann. 2004. On the relationship between classical grid search and probabilistic roadmaps. *The International Journal of Robotics Research* 23, 7-8 (2004), 673–692.
[13] Jarana Manotumruksa, Craig Macdonald, and Iadh Ounis. 2017. Matrix factorisation with word embeddings for rating prediction on location-based social networks. In *European Conference on Information Retrieval*. Springer, Cham, Switzerland, 647–654.
[14] Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12, 2 (1947), 153–157.

[15] Riccardo Poli, James Kennedy, and Tim Blackwell. 2007. Particle swarm optimization. *Swarm intelligence* 1, 1 (2007), 33–57.
[16] WL Price. 1983. Global optimization by controlled random search. *Journal of Optimization Theory and Applications* 40, 3 (1983), 333–348.
[17] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. In *Fifth international conference on computer and information science*, Vol. 27. Citeseer, 28.
[18] Suvash Sedhain, Aditya Menon, Scott Sanner, Lexing Xie, and Darius Braziunas. 2017. Low-Rank Linear Cold-Start Recommendation from Social Data. AAAI, Palo Alto, CA, USA.
[19] Ohad Shamir and Tong Zhang. 2013. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*. 71–79.
[20] Qiang Song, Jian Cheng, and Hanqing Lu. 2015. Incremental matrix factorization via feature space re-learning for recommender system. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, New York, NY, USA, 277–280.
[21] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.
[22] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2009. Scalable collaborative filtering approaches for large recommender systems. *Journal of machine learning research* 10, Mar (2009), 623–656.
[23] Bruno Veloso, Benedita Malheiro, Juan Carlos Burguillo, and Jeremy Foss. 2017. Personalised fading for stream data. In *Proceedings of the Symposium on Applied Computing*. ACM, New York, NY, USA, 870–872.
[24] Bruno Veloso, Benedita Malheiro, Juan Carlos Burguillo, Jeremy Foss, and João Gama. 2018. Personalised Dynamic Viewer Profiling for Streamed Data. In *World Conference on Information Systems and Technologies*. Springer, Cham, Switzerland, 501–510.
[25] 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, Cham, Switzerland, 459–470.
[26] 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*. ACM, New York, NY, USA, 963–965.
[27] MA Vorontsov, GW Carhart, and JC Ricklin. 1997. Adaptive phase-distortion correction based on parallel gradient-descent optimization. *Optics letters* 22, 12 (1997), 907–909.
[28] Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics bulletin* 1, 6 (1945), 80–83.
[29] Liang Xiang and Qing Yang. 2009. 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*, Vol. 1. IEEE Computer Society, 450–457.
[30] Yin Zhang, Min Chen, Dijiang Huang, Di Wu, and Yong Li. 2017. iDoctor: Personalized and professionalized medical recommendations based on hybrid matrix factorization. *Future Generation Computer Systems* 66 (2017), 30–35.