Selecting Collaborative Filtering algorithms using Metalearning

Tiago Cunha[1], Carlos Soares[1], and André C.P.L.F. de Carvalho[2]

[1] INESC-TEC/Faculdade de Engenharia da Universidade do Porto, Porto, Portugal
{tiagodscunha,csoares}@fe.up.pt
[2] ICMC - Universidade de São Paulo, São Paulo, Brasil
andre@icmc.usp.br

Abstract. Recommender Systems are an important tool in e-business, for both companies and customers. Several algorithms are available to developers, however, there is little guidance concerning which is the best algorithm for a specific recommendation problem. In this study, a metalearning approach is proposed to address this issue. It consists of relating the characteristics of problems (metafeatures) to the performance of recommendation algorithms. We propose a set of metafeatures based on the application of systematic procedure to develop metafeatures and by extending and generalizing the state of the art metafeatures for recommender systems. The approach is tested on a set of Matrix Factorization algorithms and a collection of real-world Collaborative Filtering datasets. The performance of these algorithms in these datasets is evaluated using several standard metrics. The algorithm selection problem is formulated as classification tasks, where the target attribute is the best Matrix Factorization algorithm, according to each metric. The results show that the approach is viable and that the metafeatures used contain information that is useful to predict the best algorithm for a dataset.

Keywords: Recommender System, Collaborative Filtering, Model Selection, Metalearning

1 Introduction

The digital economy enabled an important source of revenue for companies, by increasing the number of customers and markets available. However, e-commerce websites usually have an overwhelming amount of products in their catalog, which can easily result in the loss of purchase interest. This problem, known as information overload, has been reduced with the use of Recommender Systems (RSs), which recommend potentially interesting items [1]. Specifically in Collaborative Filtering (CF) algorithms, which is the focus of this work, these systems gather data from customers, products and relationships established between elements from these two groups (e.g. a customer visualizes the page of a product or buys that product) to extract patterns. These patterns can be used to recommend possibly interesting items in future sessions.

There are several recommendation methodologies, each one with a large variety of algorithms [1]. This makes it difficult to select the best algorithm for a new problem. The most common strategy is trial and error. However, it has a high computational cost. In fact, when the data size is large, it becomes virtually impossible to pursue this approach. The Metalearning (MtL) approach, which has proved successful in other Data Mining tasks, can provide a good solution to this problem. Besides, it allows the extraction of knowledge able to explain why a suggested algorithm is better suited for a specific dataset.

MtL studies how machine learning (ML) can be employed to understand the learning process and, improve the use of machine learning in future applications [6]. In MtL, learning occurs at two levels: base-level and meta-level [2]. At the base-level, base-learners (in this work, they are the CF algorithms) accumulate experience on a specific learning task (i.e., a single dataset). At the meta-level, meta-learners accumulate experience on the behavior of multiple base-learners on multiple learning tasks (i.e., multiple datasets). This experience is represented as a metamodel, which can be used to suggest the best base-learner for a specific dataset.

One of the main challenges in MtL is to define informative metafeatures, i.e. characteristics that effectively describe the area of competence of each algorithm [2]. In this study, the focus is on rating-based CF datasets and the metafeatures proposed here are based on three different perspectives on their distribution: in terms of user, item and global. These distributions are aggregated using simple, standard summary statistical functions [18]. These metafeatures are expected to contain some useful information about the (relative) performance of the algorithms. The experimental approach used in this work can be summarized as:

1. base-level experimental work to estimate the performance of the selected CF algorithms on the selected datasets;
2. extraction of metafeatures from the datasets;
3. meta-level learning to relate the metafeatures with the base-level algorithm performance;
4. extraction and presentation of metaknowledge extracted.

This work extends existing studies [7, 14, 28] by 1) proposing an approach with algorithm-independent metafeatures and 2) by performing the experimental work on a significantly larger number of datasets and base-level algorithms. The goal is to generalize the knowledge extracted from this process, rather than focus on specific application niches, unlike the related work studies.

This document is organized as follows: Section 2 presents the main aspects of CF and MtL, with emphasis on related work of model selection for RSs. Section 3 holds the explanation of the metafeature process to extract CF data characteristics. Section 4 describes the experimental procedure at the base and meta-levels, while Section 5 contains the results from both evaluation experiments. It also shows the knowledge extracted from the experiments performed. Section 6 presents the main conclusions and points out possible future works.

2 Related Work

2.1 Collaborative Filtering

RSs are inspired by human social behavior, where it is common to take into account the tastes and opinions of acquaintances when making decisions [1]. In this work, the application scope is limited to CF. Extensive surveys discussing other recommendation strategies can be found elsewhere [1, 26].

CF recommendations are based on the premise that a user must like the items favored by a similar user. Thus, it uses the feedback from each individual user to recommended items to similar users [26]. There are two types of recommendation tasks in CF. In rating prediction, the goal is to train models to accurately estimate the ratings users would give to items. Alternatively, item recommendation aims to recommend ordered lists of items, according to the preference of the users. These are fundamentally different problems and CF algorithms have been designed for each task. In this study, we will address both tasks.

Data Traditionally, the data used in CF approaches are numerical (implicit or explicit) feedback from the user, related with user preferences concerning some of the items [1]. Explicit feedback, also known as user ratings, is a numerical value, within a pre-defined scale, proportional to how the user likes the item. Probably, the most well known scale ranges from 1 to 5, based on the metaphor of 1 to 5 stars. Implicit feedback, on the other hand, derives a numerical value from the user interactions with the items on the website (e.g. clickstream data, click-through data from the search engine, the time users spends on the pages). Collecting user feedback through explicit and implicit methods present advantages and disadvantages: implicit methods are considered unobtrusive, but explicitly acquired data are more accurate in expressing the true preferences.

The data structure used in CF is known as the rating matrix R. It is described as $R = U \times I$, representing a set of users U, where $u \in \{1...N\}$ and a set of items I, where $i \in \{1...M\}$. Each element of this matrix is the numerical feedback provided by a user u relative to an item i, represented by r_{ui}.

Algorithms CF algorithms can be divided into two major classes: memory-based and model-based [1, 26, 13]. Memory-based algorithms apply heuristics on a rating matrix to compute recommendations, whereas model-based algorithms induce a model from this matrix and use it to recommend items. Memory-based algorithms are usually based on Nearest Neighbor (NN) approaches, while model-based algorithms are mostly based on Matrix Factorization (MF). For reasons explained below (Section 4.1), this work focuses solely on MF algorithms.

MF is one of the most efficient and robust algorithms for CF [12]. It assumes that the original rating matrix values can be approximated by the multiplication of at least two matrices with latent features that capture the underlying data patterns. The computation is iterative and optimizes a performance measure, usually RMSE. In the simplest formulation of MF, the rating matrix R is approximated by the product of two matrices: $R \approx PQ$, where P is an $N \times K$

matrix and Q is a $K \times M$ matrix. P is the user feature matrix, Q is the item feature matrix and K is the number of features in the given factorization.

There are three characteristics to be analyzed in each algorithm: the factorization process, the learning strategy and user/item bias. The factorization process is usually the one explained previously. However, there are algorithms using other approaches, such as Singular Value Decomposition (SVD).

The most commonly used learning strategies are Alternating Least Squares (ALS) and Stochastic Gradient Descent (SGD). These strategies are used in an iterative fashion. In each iteration, a specific formula is optimized until a threshold value is reached. ALS alternates between two steps: the P-step, which fixes Q and recomputes P, and the Q-step, where P is fixed and Q is recomputed. The re-computation on the P-step employs a regression model for each user, whose input is the vector q_i and the output is the original user rating vector. In the Q-step, the input is the q_u vector and the output is the item rating vector. For SGD, the original rating r_{ui} is compared with the predicted value, giving an error measure: $e_{ui} = r_{ui} - q_i^T q_u$. Afterwards, user and item factors are modified to minimize this error and a new iteration starts.

Next, the user/item bias is introduced in MF as a regularization measure. This bias (either for users, items or both), tries to compensate the specific user/item difference against the average values of either users/items. The purpose is to take into account the fact that users have different rating habits. Note that the user/item bias is different from the model bias: while the first is used to compensate the specific user/item difference against the average values in the CF problem, the second refers to the ML model preference for choosing one hypothesis explaining the data over other (equally acceptable) hypothesis.

There are multiple frameworks with implementations of MF algorithms available (e.g. Apache Spark[3], Recommenderlab[4], Prediction.io[5]). In this work, we focus on the *MyMediaLite* framework of MF algorithms [4].

Rating Prediction (**MF**) is the most basic algorithm for this task. It uses a standard factorization strategy, SGD, to perform the learning step and introduces no user/item bias. Another algorithm, *BiasedMatrixFactorization* (**BMF**) uses explicit user/item bias, but still learns through SGD and still uses the standard factorization approach [20]. SGD is also used as the learning technique in *LatentFeatureLogLinearModel* (**LFLLM**) [16]. However, it is inspired on logistic regression, instead of the standard MF. Besides, it has no user/item bias, since the authors state that the algorithm is insensitive to it. *SVDPlusPlus* (**SVD++**) is a MF strategy that extends the basic SVD strategy to include the items rated by the users in the optimization formula [11]. It is a combination of neighborhood algorithms with MF, which also includes user/item bias. Three asymmetric algorithms, which are variations of SVD, are also used. The asymmetric changes refer to the fact that the user (or item) factors are modeled by which items were

[3] http://spark.apache.org/
[4] https://cran.r-project.org/web/packages/recommenderlab/index.html
[5] https://prediction.io/

rated by the users (or by which users rated the items). The algorithms focus on asymmetric changes on *item* (**SIAFM**), *user* (**SUAFM**) and *both user and item* (**SCAFM**) [17]. These algorithms assume that by modeling the problem in an asymmetric fashion, the prediction formula in SVD can be linearly combined with these factors to obtain more accurate results. All **these** algorithms have user/item bias and the learning stage is conducted with SGD. A MF-based algorithm was adopted as baseline: *UserItemBaseline* (**UIB**) [12]. It uses the average rating value plus a regularized user/item bias for prediction. The optimization problem is solved with ALS. Three average-based algorithms were also included: *GlobalAverage* (**GA**), *ItemAverage* (**IA**) and *UserAverage* (**UA**). These algorithms make the predictions based on the average rating value of all ratings of all users, all ratings of an item and all ratings of an user, respectively.

Item Recommendation A different set of MF algorithms can be used to recommend rankings of items. **BPRMF** optimizes a criterion based on Bayesian logic [19]. It reduces the ranking problem to a pairwise classification task, optimizing the Area under the Curve (AUC) metric. It uses SGD as the learning strategy and no user/item bias. *MultiCoreBPRMF* (**MBPRMF**) is a parallel implementation of the previous algorithm. The algorithm *WeightedBPRMF* (**WBPRMF**) is a variation of *BPRMF* that includes a sampling mechanism that promotes low scored items and use/item bias. *SoftMarginRankingMF* (**SMRMF**) is another variation of *BPRMF*, but it replaces the optimization formula in SGD by a soft margin ranking loss inspired by SVM classifiers [24]. Another MF algorithm used is (**WRMF**) [10]. This algorithm uses ALS as the learning technique and introduces user/item bias to regularize the process. The only baseline algorithm available in this scope is *MostPopular* (**MP**). Here, items are ranked by how often they have been seen in the past.

Evaluation Due to the experimental nature of this work, the CF algorithms are evaluated using an offline approach. This evaluation involves a data split strategy (usually k-fold cross-validation, although others can be used) and the application of suitable metrics, depending on the application scope. In the case of Rating Prediction, the metrics are error based and evaluate the rating accuracy. Examples of these metrics are the Mean Average Error (MAE), the normalized version of MAE (NMAE) and the Root Mean Squared Error (RMSE) [9]. The evaluation for the Item Recommendation task is based on predicted rankings, using metrics like Mean Average Precision (MAP), Normalized Discount Cumulative Gain (NDCG), Mean Reciprocal Rank (MRR) and AUC [9].

2.2 Metalearning

MtL looks for an hypothesis or function associating the characteristics of a dataset and the behavior of learning techniques, when applied to this dataset. Its use helps understand algorithm behavior on different conformations of data [22].

There are two model induction levels in this methodology: the base-level and the meta-level. In the problem investigated in this paper, the base-level refers to the application of CF algorithms on CF datasets, while the meta-level studies the effect of the characteristics of CF datasets on the performance of CF algorithms. The MtL process addresses the algorithm selection problem in two phases: training and prediction. In the training phase, datasets are characterized by a set of measurable characteristics and CF algorithms have their performance evaluated on these datasets. Next, a learning algorithm is trained on the metadata to induce a metamodel able to associate the characteristics of the dataset with the best base-level algorithm to analyze it. In the second phase, this metamodel is used to predict the best algorithm for a given dataset [21].

Metafeatures are dataset descriptors that are expected to correlate well with the performance of the models learned by different techniques [2]. The literature describes two main types of meta-features: (1) Statistical and/or information-theoretical measures and (2) Landmarkers. This study adopts the first type of meta-features for CF. More information on metafeatures can be found elsewhere [22].

The metatarget determines the type of prediction to be made by the MtL model for a dataset. Common metatargets are (1) the algorithm with the best performance on the dataset (2) a non-ordered subset of algorithms that performed well on the dataset, (3) a ranking of algorithms according to their performance on the dataset and (4) the performance of a set of techniques for the dataset [2]. This study will follow the first approach, namely addressing MtL as a classification task.

2.3 Model selection for Recommender Systems

This section presents related work on model selection for RSs using MtL. Firstly, it is important to notice that, despite sharing the same nature, the problems have different goals: to predict the performance of CF algorithms at user level [7], to predict the performance of CF algorithms at dataset level [14] and to predict the best algorithm for group-oriented recommendations [28]. The studies diverge between using public [7, 14] and private datasets [28], although none has the appropriate number of datasets required: the maximum found is 4. This is important since the generalization of the metalearning process requires a large and diverse collection of datasets. The base level algorithms are mostly based on NN, which despite being an important technique, have several drawbacks with larger datasets and are somehow outdated. The main exception is on the group-aware recommendations, since the algorithms are simply heuristics. The metafeatures used are of several types:

1. rating distribution analysis: the number of ratings per user, the average rating per user, the standard rating deviation per user [7], the ratings entropy, the ratings Gini index and ratings sparsity [14];
2. neighbourhood analysis: the number of neighbors, the average similarity to the top closest 30 neighbors, the clustering coefficient of a group of users,

the average Jaccard coefficient per user [7], group size, social contact level and dissimilarity level [28];
3. general user analysis: the user influence [7], experience level and activity level [28];
4. general item analysis: the item popularity, the item preference, the user influence and the average item entropy [7].

The techniques used in the meta-level are divided into 2 types: regression [7, 14] and classification [28]. While the regression is evaluated with MAE measure, the classification problem uses error and rankings measures: RMSE and MRR.

3 Metafeatures for Recommender Systems Problems

One of the most important factors in the success of a metalearning approach is the definition of a set of metafeatures that contain information about the (relative) performance of the base-level algorithms [2]. Given that there is little work on MtL for recommender systems and that the nature of the data in these problems is quite different from traditional MtL problems (e.g. classification or regression), there is not much work we can build upon. The set of metafeatures proposed here is based on 1) the application of systematic procedure to develop metafeatures [18] and 2) extend and generalize the state of the art metafeatures for recommender systems [7, 14, 28].

The framework requires three main elements: the object that the metafeatures characterize, the function that analyzes the object and provides the result as a data distribution, and the post-processing functions that are applied on these distributions to extract their characteristics.

In the proposed approach, the objects can be of three types: dataset, row and column. As previously seen, row and column refer to user and item, respectively. On the dataset we analyze only the original rating distribution. However, for each row and column, we use three distinct functions: count the number of elements, mean value and sum of values. The post-processing functions used provide the following values: maximum, minimum, mean, standard deviation, median, mode, entropy, Gini index, skewness and kurtosis. The notation used to represent metafeatures follows the format: *object.function.post function.*

For each rating matrix $R = U \times I$, the set of meta-features, M, is extracted in two steps: (1) application of a function f to the ratings r_{ui} in each row ($f(U)$), column ($f(I)$) and the entire dataset ($f(R)$) to obtain three different ratings distributions and (2) post-process the outcome of each function f (in the shape of distribution) with the so-called post-functions pf by extracting statistics that can be used as meta-features. Therefore, the set of meta-features is described as $M = pf[f(U)] \cup pf[f(I)] \cup pf[f(R)]$. Four simple statistics were also included and presented in Table 1.

These combinations enable the exploration of the rating distribution analysis metafeatures commonly used in selection of CF algorithms and, more importantly, extend them in a systematic way.

4 Experimental setup

4.1 Base-level

The robustness of experimental results in MtL depends on the number of datasets available as each dataset represents a meta-example [2]. In most MtL studies, however, only a few dozen datasets are available. This is also true for CF tasks, as there are not many public datasets. Furthermore, very often these datasets are very large, which makes it hard to use them for MtL experiments, as it implies running all the base-level algorithms on the datasets. Thus, we selected 32 datasets for this study. Table 1 lists these datasets, providing their names, reference and a few simple statistics with approximate values for readability. To the best of our knowledge, this is the largest experimental study in terms of number of CF rating based datasets.

These datasets present different numbers of users, items and ratings. As expected, in most cases the sparsity is greater than 0.9 [26]. To ensure that the values of the metafeatures and the performance measures are comparable across datasets, it is necessary to normalize the rating scales. We decided to normalize all ratings to the scale $[1, 5]$, since it is the most common.

The performance of all selected CF algorithms on each dataset was estimated, using, as explained earlier, the *MyMediaLite* framework (Section 2.1). Some algorithms, namely the NN algorithms, were not able to obtain results on the largest datasets. Therefore, we decided to limit the algorithms to those able to process all the available datasets: the MF algorithms and the baselines. However, no tuning of these parameters was performed, as this is common practice in MtL.

We evaluated the algorithms using seven commonly employed recommendation metrics (Section 2.1). Each of those metrics evaluates the recommendation problem accordingly to a specific perspective. Thus, the best CF algorithm for a given dataset may vary for different evaluation metrics. Thus, we generate a different meta-target variable for each evaluation metric, yielding seven different classification meta-level tasks. The evaluation process uses 10-fold cross-validation and there is no parameter tuning at this stage. We decided to use the default parameters, since this is the usual approach in MtL experiments.

4.2 Meta level

The meta-features defined in this work were implemented using the *recommenderlab* package,[6] which is based on the *Matrix* package.[7] These packages provide a flexible interface for CF data through a sparse matrix data structure. The implementations from these packages not only allow the application of functions to each row, column and entire dataset, but also worked efficiently.

Since all meta-features are somehow related to the original ratings distribution, it is necessary to ensure that the correlated features are removed. Therefore, a Correlation Feature Selection strategy (CFS) was applied to them, using

[6] https://cran.r-project.org/web/packages/recommenderlab/index.html
[7] https://cran.r-project.org/web/packages/Matrix/index.html

Table 1: Datasets used in the base-level experiments

dataset	#users	#items	#ratings	sparsity	ratings scale	ref.
amazon-apps	1.3M	61k	2.6M	0.999	[1,5]	[15]
amazon-automotive	851k	320k	1.3M	0.999	[1,5]	[15]
amazon-baby	531k	64k	915k	0.999	[1,5]	[15]
amazon-beauty	1.2M	249k	2M	0.999	[1,5]	[15]
amazon-cd	1.5M	486k	3.7M	0.999	[1,5]	[15]
amazon-digital-music	478k	266k	836k	0.999	[1,5]	[15]
amazon-food	768k	166k	1.2M	0.999	[1,5]	[15]
amazon-games	826k	50k	1.3M	0.999	[1,5]	[15]
amazon-garden	714k	105k	993k	0.999	[1,5]	[15]
amazon-home	2.5M	410k	4.2M	0.999	[1,5]	[15]
amazon-instant-video	426k	24k	584k	0.999	[1,5]	[15]
amazon-instruments	339k	83k	500k	0.999	[1,5]	[15]
amazon-movies	73k	4k	111k	0.999	[1,5]	[15]
amazon-office	909k	130k	1.2M	0.999	[1,5]	[15]
amazon-pet-supplies	741k	103k	1.2M	0.999	[1,5]	[15]
amazon-phones	2.2M	320k	3.4M	0.999	[1,5]	[15]
amazon-sports	1.9M	479k	3.3M	0.999	[1,5]	[15]
amazon-tools	1.2M	260k	1.9M	0.999	[1,5]	[15]
amazon-toys	1.3M	328k	2.3M	0.999	[1,5]	[15]
flixter	148k	49k	8.2M	0.998	[0,5]	[27]
jester1	25k	100	1.8M	0.275	[-10,10]	[5]
jester2	24k	100	1.7M	0.273	[-10,10]	[5]
jester3	25k	100	617k	0.753	[-10,10]	[5]
movielens100k	1k	2k	100k	0.937	[0,5]	[8]
movielens10m	70k	11k	10M	0.987	[0,5]	[8]
movielens1m	6k	4k	1M	0.955	[0,5]	[8]
movielens20m	138k	27k	20M	0.995	[0,5]	[8]
movielens_latest	229k	27k	21M	0.997	[0,5]	[8]
movietweetings_latest	37k	21k	389k	0.999	[0,10]	[3]
movietweetings_recsys2014	25k	15k	211k	0.999	[0,10]	[3]
tripadvisor	778k	13k	1.5M	0.999	[1,5]	[23]
yahoo-music	6k	10k	364k	0.994	[1,5]	[25]

a threshold $t \in [0.6, 0.9]$ with increments of 0.5. This decreased the number of features from 74 to the interval $[11, 28]$, depending on the threshold used.

Each set of meta-features originated seven meta-level datasets, one per each CFS threshold. Each metadataset is associated with 1 of the 7 recommendation targets, creating 49 metadatasets. As the model selection problem is approached here as a classification task, 11 classification algorithms representing several biases were chosen: ctree, C4.5, C5.0, kNN, LDA, Naive Bayes, SVM (linear, polynomial and radial kernels), random forest and a baseline algorithm: majority vote. Since the metadatasets have a reduced number of examples, the algorithms were evaluated for accuracy in a leave one out strategy and no tuning was performed. The goal is to reduce the potential overfitting of the meta-models.

5 Experimental Results

5.1 Base-level results

The results at the base-level are presented in Table 2. This table presents the best algorithm for each dataset and metric. Each metric is applicable only to a suitable type of recommendation algorithm: rating prediction or item recommendation. These results are used as the target attributes in the meta-datasets.

Regarding the rating prediction experiments, it can be observed that most datasets have for best algorithm either a baseline or BMF. In fact, only 6 datasets do not follow this process. Furthermore, the results show that, for the metrics MAE and NMAE, the best algorithms are almost always the same. Since the metrics are very similar, this behavior is expected.

In the item recommendation experiments, the distribution of best algorithm for each dataset is fairly distributed, although it is noticeable that these algorithms have the tendency to not change according to the different metrics. This is also expected since all of them evaluate the ranking accuracy of the algorithms. However, since AUC values are more concerned with accuracy assessment regardless of the ranking, it produces different results.

It is important to observe that the baseline algorithms often perform best on the largest datasets, regardless of the recommendation scope: IA, UA and GA in rating prediction and MP in item recommendation. This relates to the sparsity problem in CF and how difficult it is to make predictions in a cold start environment.

Another important observation is that there are few algorithms that are never chosen as the best in any pair dataset/metric. This may be a consequence of the lack of tuning on the base level methods. These are the cases of SUAFM and UIB in rating prediction and SMRMF in item recommendation. This means that it is not possible to extract useful knowledge from these algorithms in the meta-level. This can change if we can increase the number and diversify the nature of the datasets in order to expand the search space.

5.2 Meta-level results: rating prediction

Figures 1 and 2 show the meta-models performance across several CFS thresholds for the MAE and RMSE metrics, respectively. Each threshold was used to understand the effect of correlation in our metafeature framework. The NMAE analysis was discarded in the paper due to space restrictions. However, the performance is similar to the MAE metric.

The accuracy values are clearly different: while most algorithms, concerning MAE, performed always above the baseline, on the RMSE meta-level problem, only 2 of them achieve this goal. This is a consequence of the bias in the meta-dataset towards the BMF algorithm. Since this algorithm wins most of the times, the metalearning strategy becomes obsolete for this scope. Hopefully, using more and diversified datasets will enable to study this specific problem in further detail. This experiment shows that the meta-models created with our metafeature framework are useful for solving the algorithm selection problem for CF.

Table 2: Best models on multiple evaluation metrics for each dataset

dataset	Rating Prediction			Item Recommendation			
	MAE	NMAE	RMSE	MAP	MRR	NDCG	AUC
amazon-apps	BMF	BMF	BMF	MP	MP	MP	MP
amazon-automotive	IA	IA	BMF	MP	MP	MP	MP
amazon-baby	IA	IA	BMF	MP	MP	MP	MP
amazon-beauty	UA	UA	BMF	MP	MP	MP	MP
amazon-cd	UA	UA	BMF	MBPRMF	MBPRMF	MBPRMF	MBPRMF
amazon-digital-music	UA	UA	BMF	BPRMF	MP	MP	MP
amazon-food	IA	IA	BMF	MP	MP	MP	MP
amazon-games	BMF	BMF	BMF	MP	MP	MP	MP
amazon-garden	IA	IA	BMF	MP	MP	MP	MP
amazon-home	IA	IA	BMF	MBPRMF	MBPRMF	MBPRMF	MBPRMF
amazon-instant-video	IA	IA	BMF	MP	MP	MP	MP
amazon-instruments	IA	IA	BMF	MP	MP	MP	MP
amazon-movies	BMF	BMF	BMF	WBPRMF	WBPRMF	WBPRMF	MBPRMF
amazon-office	IA	IA	BMF	MP	MP	MP	MP
amazon-pet-supplies	IA	IA	BMF	MP	MP	MP	MP
amazon-phones	BMF	BMF	BMF	BPRMF	BPRMF	BPRMF	MBPRMF
amazon-sports	IA	IA	BMF	BPRMF	MBPRMF	MBPRMF	MBPRMF
amazon-tools	IA	IA	BMF	MP	MP	MP	MP
amazon-toys	IA	IA	BMF	MP	MP	MP	MP
flixter	BMF	BMF	BMF	MP	MBPRMF	MP	MBPRMF
jester1	SVD++	SVD++	SVD++	MP	MP	MP	MP
jester2	SVD++	SVD++	SVD++	MP	MP	MP	MP
jester3	SIAFM	SIAFM	SIAFM	MP	MP	MP	MP
movielens_latest	BMF	BMF	BMF	WRMF	WRMF	WRMF	MBPRMF
movielens100k	BMF	BMF	BMF	WRMF	WRMF	WRMF	WRMF
movielens10m	MF	MF	BMF	WRMF	WRMF	WRMF	WRMF
movielens1m	MF	MF	MF	WRMF	WRMF	WRMF	MBPRMF
movielens20m	BMF	BMF	BMF	WRMF	WRMF	WRMF	MBPRMF
movietweetings_latest	SCAFM	SCAFM	SCAFM	WRMF	WRMF	WRMF	MBPRMF
movietweetings_recsys2014	UA	GA	GA	MP	MP	MP	MBPRMF
tripadvisor	SIAFM	SIAFM	SIAFM	WBPRMF	WBPRMF	WBPRMF	MBPRMF
yahoo-music	SVD++	SVD++	LFLLM	WRMF	WRMF	WRMF	WRMF

One important point lies in the fact that the performances are fairly constant across the thresholds. This was not expected beforehand and points to the fact that the metafeatures used are very different in nature, despite having for basis the same rating distribution. Therefore, the CFS analysis does not have sufficient impact on selecting the best meta-models. This means that, in this experimental setup, if a meta-model beats the baseline, it is of low importance which is the CFS threshold used to build it.

The strategy to select the best algorithms follows the principle that the average accuracy across thresholds must be always better than the baseline. To ensure this principle, the algorithms whose average accuracy for all thresholds minus the standard deviation is above the performance of the baseline algorithm (majority voting) were selected as the best ones. Thus, the best algorithms for the MAE metric are all except the ctree. For the RMSE target, only the SVM with polynomial kernel satisfies this principle.

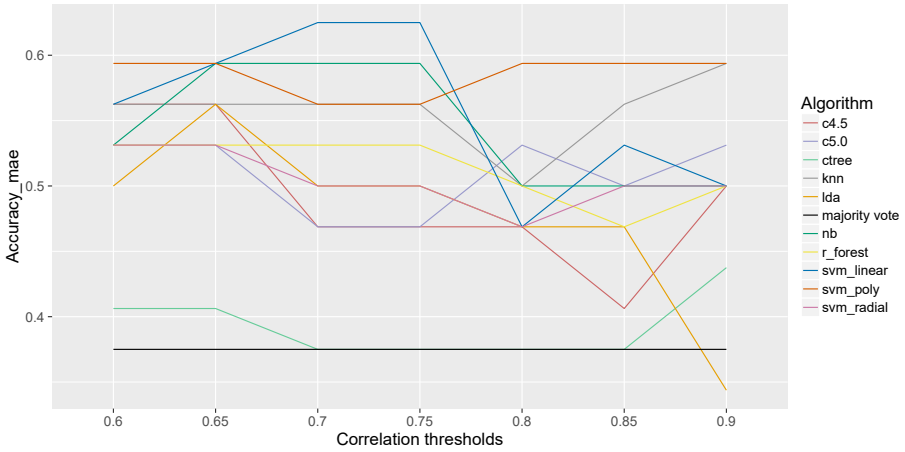

Fig. 1: Results of MAE meta-dataset on CFS thresholds

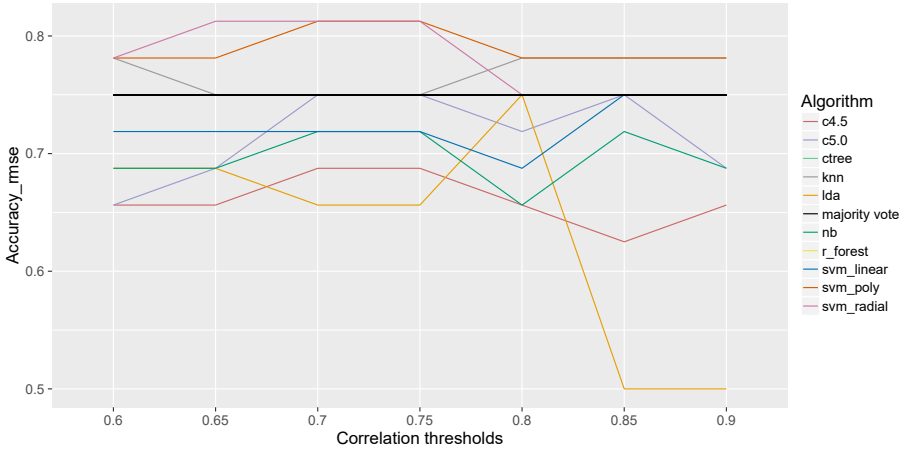

Fig. 2: Results of RMSE meta-dataset on CFS thresholds

5.3 Meta-level results: item recommendation

Figures 3 and 4 present the accuracy results for the NDCG and AUC metrics, respectively. MRR and MAP were discarded due to space restrictions. However, the performances on these targets are also fairly similar to performance obtained with the NDCG metric. First of all, one notices that there are several algorithms whose performance was better than the baseline and remained stable across the CFS thresholds. This behavior is similar to the one found in the rating prediction problem. The only exception found shows that in both metrics, the Naive Bayes algorithm presents a poor predictive performance, scoring always below the baseline accuracy. The only explanation available is that the class

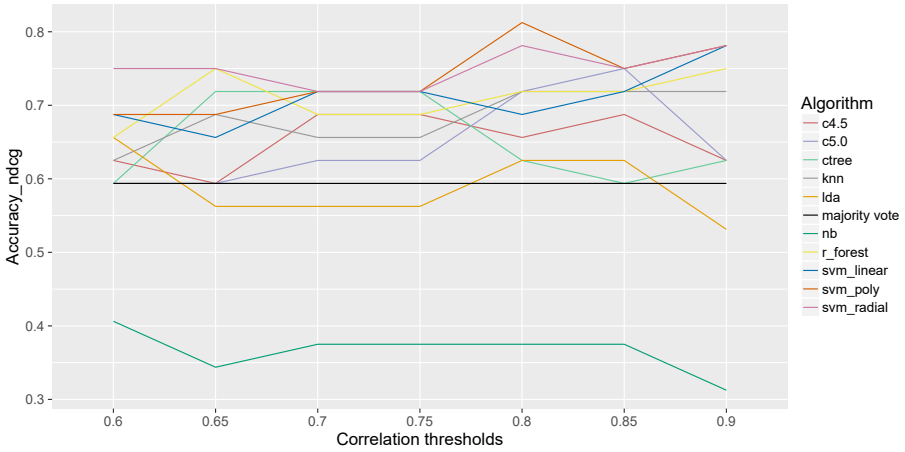

Fig. 3: Results of NDCG meta-dataset on CFS thresholds

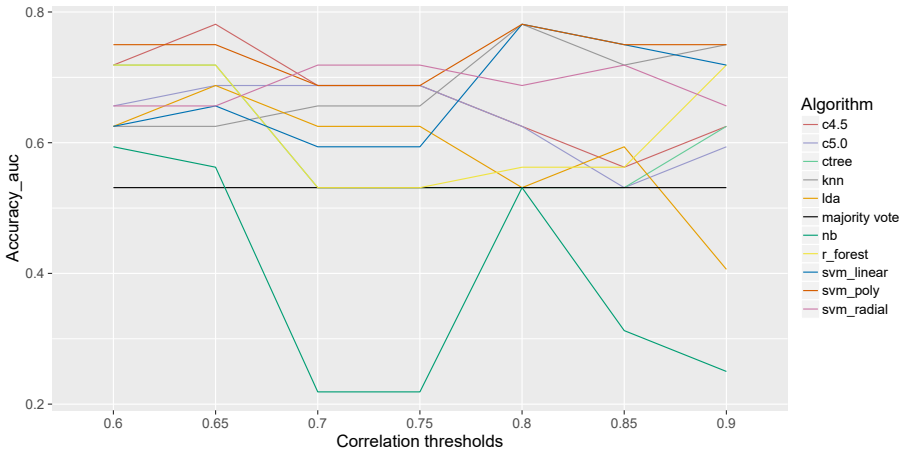

Fig. 4: Results of AUC meta-dataset on CFS thresholds

distribution is more balanced than the first problem and this affects the Naive Bayes algorithm specially. However, on the overall analysis, these meta-models perform better than in the rating prediction problem.

Following the previous strategy to select the best models, for the NDCG target, the algorithms with the best predictive performance were SVM (linear, polynomial and radial kernels), random forest, kNN and C4.5. For the AUC metric, the best algorithms are almost the same, with the difference that the random forest algorithm was replaced with the C5.0 algorithm.

5.4 Meta-knowledge

To extract meta-knowledge from the previous MtL experiments, variable importance analysis was performed on all the algorithms previously identified with the best performance. Two different analysis were carried out: with and without model information. Thus, in the first case, the trained model characteristics are used to infer the most important variables, unlike in the second case.

The first analysis was conducted by assessing the feature frequency in the best models for all CFS thresholds. Next, the meta-features present in most meta-datasets (i.e., 5 meta-datasets must contain the feature) were selected. The results extract 11 meta-features: *number of ratings*, *dataset.ratings.mode*, *dataset.ratings.gini*, *row.mean.median*, *row.mean.entropy*, *row.mean.skewness*, *row.count.kurtosis*, *column.count.gini*, *column.count.skewness*, *col.mean.min* and *column.sum.kurtosis*. The features are distributed as follows: 3 features about the entire dataset, and 4 for each the user and item. This highlights the fact that not only the original ratings distribution holds important characteristics to solve the algorithm selection problem for CF. When these metafeatures are compared with the related work (Section 2.3), it is observed that few have already been used and their importance is confirmed in this study (for instance the number of ratings and the ratings gini index). However, there are others proposed that have not been used so far and that hold important value. Also, *column.count* and *row.mean* are found to be the most relevant distributions to be analyzed in this problem. Although the related work has some metafeatures related to the *row.mean* distribution (i.e., average of user ratings), the depth level on which they were used does not compare to our experimental work. This leads us to the conclusion that our metafeature framework is able to propose novel and important metafeatures which are useful for the problem of algorithm selection.

A second analysis was carried out using the method RELIEF.[8] It finds weights of attributes based on the distance between instances, using only the dataset. The results obtained show that, for each meta-dataset, a subset of the previously mentioned meta-features is selected. These tests assure the validity of the set of most important features found.

The main pattern found upon model inspection is that a low global number of ratings leads to the selection of a baseline algorithm. This is expressed in several ways by a combination of the previous meta-features or simply by the number of ratings. Other meta-features, despite being important in discerning the algorithms, are difficult to interpret. This has more impact if the functions or post-functions are themselves not easily understandable.

One important consideration lies in the fact that the meta-dataset has very few instances, which prevents a more detailed analysis of the meta-knowledge. Still, the fact remains that the meta-features proposed are informative and that help tackling the problem of algorithm selection for CF.

[8] https://cran.r-project.org/web/packages/FSelector/index.html

6 Conclusions

In this study, we have proposed a Metalearning approach to select Matrix Factorization algorithms on two scopes of the CF problem: rating prediction and item recommendation. The meta-features proposed follow a thorough analysis of the feature space and are based on combinations of the original rating distribution and generalize the meta-features used in recent studies. Each base-learner is trained on a collection of real-world datasets and evaluated on a range of suitable metrics, which serve as different targets in the meta-level. The meta-models induced have performed well above the baseline algorithm, even when the meta-dataset has very few examples. Furthermore, variable importance analysis has shown that the proposed meta-features provide added knowledge when compared with the usage of characteristics of only the original rating distribution. Future work may focus on increasing the number of datasets, perform dimensionality reduction to expand the range of algorithms available, proposal of meta-features related to the models characteristics, the extension of the meta-targets to label ranking problems and tuning of both the base and meta level algorithms.

Acknowledgments This work is financed by the ERDF Fund through the Operational Programme for Competitiveness and Internationalization - COMPETE 2020 of Portugal 2020 through the National Innovation Agency (ANI) as part of the project 3506 and also through project «POCI-01-0145-FEDER-006961» via National Funds through the FCT – Fundação para a Ciência e a Tecnologia as part of project UID/EEA/50014/2013. The research was also funded from the ECSEL Joint Undertaking, the framework programme for research and innovation horizon 2020 (2014-2020) under grant agreement 662189-MANTIS-2014-1.

References

1. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowledge-Based Systems 46, 109–132 (2013)
2. Brazdil, P., Giraud-Carrier, C., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Springer Publishing Company, Incorporated, 1 edn. (2009)
3. Dooms, S., De Pessemier, T., Martens, L.: MovieTweetings: a Movie Rating Dataset Collected From Twitter. In: Workshop on Crowdsourcing and Human Computation for Recommender Systems, CrowdRec at RecSys 2013 (2013)
4. Gantner, Z., Rendle, S., Freudenthaler, C., Schmidt-Thieme, L.: MyMediaLite: A Free Recommender System Library. In: ACM Conference on Recommender Systems. pp. 305–308 (2011)
5. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval 4(2), 133–151 (2001)
6. Gomes, T.A., Prudêncio, R.B., Soares, C., Rossi, A.L., Carvalho, A.: Combining meta-learning and search techniques to select parameters for support vector machines. Neurocomputing 75(1), 3 – 13 (2012)
7. Griffith, J., O'Riordan, C., Sorensen, H.: Investigations into user rating information and predictive accuracy in a collaborative filtering domain. In: ACM Symposium on Applied Computing. pp. 937–942 (2012)

8. GroupLens: MovieLens datasets (2016), http://grouplens.org/datasets/movielens/
9. Herlocker, J.L., Konstan, J.a., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems 22(1), 5–53 (2004)
10. Hu, Y., Koren, Y., Volinsky, C.: Collaborative Filtering for Implicit Feedback Datasets. In: IEEE International Conference on Data Mining. pp. 263 – 272 (2008)
11. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 426–434 (2008)
12. Koren, Y.: Factor in the neighbors: Factor in the Neighbors: Scalable and Accurate Collaborative Filtering. ACM Transactions on Knowledge Discovery from Data 4(1), 1–24 (2010)
13. Lü, L., Medo, M., Yeung, C.H., Zhang, Y.C., Zhang, Z.K., Zhou, T.: Recommender systems. Physics Reports 519(1), 1–49 (2012)
14. Matuszyk, P., Spiliopoulou, M.: Predicting the Performance of Collaborative Filtering Algorithms. In: International Conference on Web Intelligence, Mining and Semantics. pp. 38:1–38:6 (2014)
15. McAuley, J., Leskovec, J.: Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text. In: ACM Conference on Recommender Systems. pp. 165–172 (2013)
16. Menon, A.K., Elkan, C.: A log-linear model with latent features for dyadic prediction. In: Proceedings - IEEE International Conference on Data Mining, ICDM. pp. 364–373 (2010)
17. Paterek, A.: Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop. pp. 2–5 (2007)
18. Pinto, F., Soares, C., Mendes-Moreira, J.: Towards automatic generation of metafeatures. In: Advances in Knowledge Discovery and Data Mining, pp. 215–226. Springer (2016)
19. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-thieme, L.: BPR: Bayesian Personalized Ranking from Implicit Feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. pp. 452–461 (2009)
20. Salakhutdinov, R., Mnih, A.: Probabilistic Matrix Factorization. In: Advances in Neural Information Processing Systems (NIPS '08). pp. 1257–1264 (2008)
21. Serban, F., Vanschoren, J., Bernstein, A.: A survey of intelligent assistants for data analysis. ACM Computing Surveys V(212), 1–35 (2013)
22. Vanschoren, J.: Understanding machine learning performance with experiment databases. Ph.D. thesis, Katholieke Universiteit Leuven (2010)
23. Wang, H., Lu, Y., Zhai, C.: Latent Aspect Rating Analysis Without Aspect Keyword Supervision. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 618–626. ACM (2011)
24. Weimer, M., Karatzoglou, A., Smola, A.: Improving Maximum Margin Matrix Factorization. Machine Learning 72(3), 263–276 (2008)
25. Yahoo!: Webscope datasets (2016), https://webscope.sandbox.yahoo.com/
26. Yang, X., Guo, Y., Liu, Y., Steck, H.: A survey of collaborative filtering based social recommender systems. Computer Communications 41, 1–10 (2014)
27. Zafarani, R., Liu, H.: Social Computing Data Repository at {ASU} (2009), http://socialcomputing.asu.edu
28. Zapata, A., Menéndez, V.H., Prieto, M.E., Romero, C.: Evaluation and selection of group recommendation strategies for collaborative searching of learning objects. International Journal of Human-Computer Studies 76, 22–39 (2015)