Dataset Morphing to Analyze the Performance of Collaborative Filtering

André Correia[1], Carlos Soares[1,2,3], and Alípio Jorge[3,4]

[1] Faculdade de Engenharia da Universidade do Porto, Porto, Portugal
{up200706629,csoares}@fe.up.pt
[2] LIACC, Porto, Portugal
[3] LIAAD-INESC TEC, Porto, Portugal
[4] Faculdade de Ciências da Universidade do Porto, Porto, Portugal
amjorge@fc.up.pt

Abstract. Machine Learning algorithms are often too complex to be studied from a purely analytical point of view. Alternatively, with a reasonably large number of datasets one can empirically observe the behavior of a given algorithm in different conditions and hypothesize some general characteristics. This knowledge about algorithms can be used to choose the most appropriate one given a new dataset. This very hard problem can be approached using metalearning. Unfortunately, the number of datasets available may not be sufficient to obtain reliable metaknowledge. Additionally, datasets may change with time, by growing, shrinking and editing, due to natural actions like people buying in a e-commerce site. In this paper we propose dataset morphing as the basis of a novel methodology that can help overcome these drawbacks and can be used to better understand ML algorithms. It consists of manipulating real datasets through the iterative application of gradual transformations (morphing) and by observing the changes in the behavior of learning algorithms while relating these changes with changes in the meta features of the morphed datasets. Although dataset morphing can be envisaged in a much wider framework, we focus on one very specific instance: the study of collaborative filtering algorithms on binary data. Results show that the proposed approach is feasible and that it can be used to identify useful metafeatures to predict the best collaborative filtering algorithm for a given dataset.

Keywords: Recommender Systems · Metalearning

1 Introduction

In this paper, we propose an empirical methodology for improved understanding of the behavior of algorithms that combines a novel data manipulation approach, *dataset morphing*, with a Metalearning (MtL) approach. MtL consists of relating data characteristics (*metafeatures*) to the performance of learning algorithms. These metafeatures are expected to contain some useful information about the performance of the algorithms. To generalize the extracted (meta) knowledge and, therefore, make it applicable to new problems, MtL approaches require a large collection of datasets, which is often not the case. Dataset morphing addresses this issue by iteratively transforming (morphing) one real dataset into another. If the two datasets display interesting contrasting behavior of algorithms (e.g. algorithm A is better than B on one dataset but not on the other) then interesting metaknowledge can be obtained (e.g. determine the turning point on the performance of the algorithms, and carefully analyzing what happens around the performance boundary in terms of data characteristics). The proposed methodology can be used to select the most appropriate algorithm for a new problem or to analyze algorithm behaviour with evolving data.

As a an example of application for Recommender Systems (RS), we instantiate the above proposed method to study two popular Collaborative Filtering (CF) algorithms: item-based and user-based neighborhood approaches. We focus on item recommendation (*top-N*), where the aim is to recommend ordered lists of items in a binary setting [12]. To automatically identify the most adequate CF algorithm for a given data set has proven challenging [2]. Empirical results about algorithm behavior are limited due to the absence of a large number of datasets [2] and purely artificially generated data does not entirely solve the problem because it is unlikely that it reflects real world distributions.

This work extends existing studies [3,2] by proposing dataset morphing as a process of generating multiple realistic datasets (viewed as meta-examples), that could be useful to enrich the metadata and, therefore, to improve the results of MtL processes that learn the relationship between the performance of RS algorithms and data characteristics. Despite the provided example with CF, dataset morphing is virtually applicable to any Machine Learning domain.

2 Metalearning

MtL studies how Machine Learning (ML) can be employed to understand the learning process and improve the use of ML in future applications [11]. A successful MtL approach can provide a solution to the problem of selecting an algorithm for a given dataset [2]. It allows the extraction of knowledge that explains why a suggested algorithm is a good choice. It uses ML techniques to obtain predictive models, which associate data characteristics to algorithm performance. The methodology involves extracting characteristics, named *metafeatures*, from multiple datasets and assessing the performance, which will be used as *metalabels*, of a group of algorithms. Afterwards, this data is used to induce a predictive model to represent the relationship between the metafeatures and metalabels. After obtaining an accurate MtL model we can predict the most promising algorithm without running a full-fledged empirical evaluation and also explain why an algorithms performs better or worse [3].

MtL has been used for algorithm selection in RS [2,3,6]. These authors manually define metafeatures, which aggregate information from datasets into single number statistics. For example, the number of instances in the dataset is a *simple* metafeature, the mean or kurtosis of a column is a *statistical* metafeature.

They then use supervised ML to learn the relationships between the metafeatures and the performance of recommendation algorithms on datasets, measured by standard metrics. Although the use of MtL for the selection of CF algorithms has already been investigated, the approaches proposed have limited scope: the set of datasets, recommendation algorithms and metafeatures studied was rather restricted An extensive overview of their positive and negative aspects can be seen in a recent survey [3].

3 Morphing Recommendation Datasets

Image morphing has proven to be a powerful tool for visual effects in film and television, enabling the fluid transformation of one digital image into another [14]. We believe that the principle behind image morphing can be applied to generate datasets that can be used to study the behavior of RS algorithms. Actually, we can generalize the morphing technique to any type of ML problem. Thus, dataset morphing can be defined, in general, as a process of gradually transforming a source dataset into a target dataset. That way, we can explore the space of datasets along trajectories and study the behavior of algorithms in regions of that space that are not currently available, particularly in regions where algorithms' performances change.

The approach proposed here consists in starting with two datasets — the source (D_s) and the target (D_t). The operational goal is to analyze the evolution in the behavior of one or more algorithms between two points of interest. In particular, we will pick up pairs of datasets where two RS algorithms A and B have contrasting relative performances. A is better than B in one dataset and vice-versa. This set up will originate two regions of the space of datasets. We can study those two regions, their boundary and the trajectory that crosses that boundary. We get from one dataset to the other by sequentially applying transformations $\{T_1, T_2, ..., T_{n-1}, T_n\}$ (Figure 1). The initial datasets have contrasting algorithm performance. The color gradient, illustrated in Space D, means that during the transformation process, (intermediate) datasets $\{D_1, D_2, ..., D_{n-2}, D_{n-1}\}$ will gradually become more similar to the target dataset (D_t). As previously mentioned it is important to keep datasets as realistic as possible. To have that, intermediate datasets $\{D_1, D_2, ..., D_{n-2}, D_{n-1}\}$ are a mixture of real — source and target — datasets. In short, considering source (D_s) and target (D_t) datasets and a transformation function (τ), we define dataset morphing as a process of iteratively getting intermediate datasets (D_j) such that:

$$\mathcal{D}_{morph} : \{D_j \mid D_0 = D_s, D_n = D_t, D_j = \tau(D_{j-1})\}, 1 \leq j < n \tag{1}$$

where $\mathcal{D}_{morph}$ is the set of datasets and n is the number of transformations needed to get from source (D_s) to target (D_t). The function (τ) is guaranteed to converge.

The upper layer of Figure 1 illustrates the trajectory in the dataset space. The feature space $\mathcal{F}$, represented with vectors of metafeatures $\{MF_1, MF_2, ..., MF_f\}$, is the middle layer. These metafeatures are important to characterize the relative

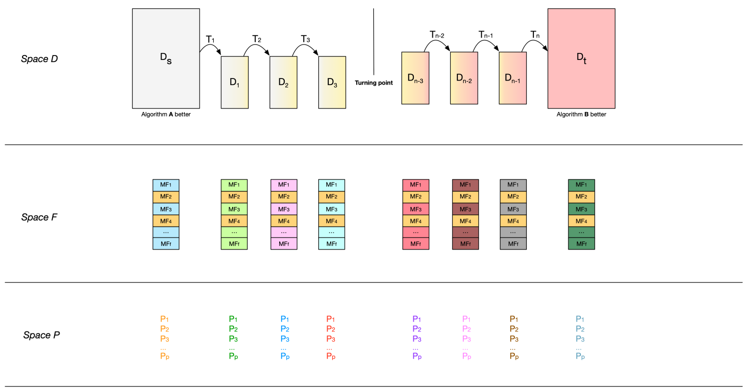

Fig. 1: Analysis of algorithm behavior between two points of interest using a metalearning approach based on dataset morphing. D is the data space; F is the metafeatures space; and P is the algorithm performance space.

performance of the algorithms. As transformations are applied, some metafeatures may change a lot, others only slightly and others not at all. These different types of changes are illustrated in Figure 1 with the magnitude of change in the colors between adjacent vectors. Finally, in the bottom layer, the performance space $\mathcal{P}$ with $\{P_1, P_2, ..., P_p\}$ is represented. The performance of the algorithms in the datasets of space $\mathcal{D}$ may be evaluated using several standard metrics. Once again, colors are intended to represent performance variation. To learn about algorithm behaviour, our focus will be on metafeatures that vary according to the changes in algorithm performance.

3.1 Dataset transformations

Source and target datasets could be selected based on a specific property of interest. In this paper we focus on contrasting algorithm performance, where a given algorithm A outperforms an algorithm B in (D_s) and the opposite happens in (D_t). Other possibilities can be considered, such as different performances of the same algorithm, comparison with baselines or effects of parameters.

One of the key issues in the methodology is the definition of the transformation function. One important property it should have is *convergence*. After each application the resulting dataset is more similar to the target and less similar to the source. Another important property is *smoothness*. The difference between two consecutive datasets should be small both in terms of metafeatures and of performance. Other characteristics may depend on the task and the type of data available. In this paper, we will focus on CF with binary data. Transformations may be simple (e.g., random bit flipping, random rows/ columns switching) or more complex (e.g., switch the most similar row/column first). They can also be applied in batches (e.g., flipping a portion of all bits, switching a portion of all rows/columns).

We also have natural morphing processes when users' preferences have a very volatile nature [9]. In real world RS, it is reasonable to approach ratings data as evolving datasets: ratings are continuously being generated, and we have no control over the data rate or the ordering of the arrival of new ratings. Actually, adding or removing a row means a new customer application or disassociation, respectively. Likewise, adding or removing a column denotes a new item arrival or removal, respectively.

4 Empirical Evaluation

The main aim of these experiments is to show that dataset morphing can be useful for identifying predictive metafeatures of the relative performance of CF algorithms using a limited number of original datasets. We intend to illustrate how dataset morphing enriches the metadata and improves the results of MtL processes.

4.1 Base-Level

In this study, we focus on the item recommendation CF task and evaluate $top-1$, 3, 5, 10, 15 and 20 recommendation lists. CF algorithms are evaluated using a 10-fold cross-validation scheme with the all-but-1 protocol to collect data about the behavior of two CF algorithms: user-based and item-based. As both are k Nearest Neighbors (NN) algorithms, we considered $k = 20$ and $k = 50$ for user-based and item-based, respectively. The performance of these algorithms is estimated on each dataset, using *precision*@k. In terms of implementation we use on the *recommenderlab* package [5] since the comparison of recommender algorithms is readily available [7]. Other algorithms, parameters or platform could have been chosen, without loss of generality. Table 1 lists the 3 real-world datasets selected for this study. They were binarized by making items with a rating of 1, or higher, a positive rating. Due to the very large number of experiments needed for meta learning, to have feasible computational times we have used subsets of the original datasets, obtaining 60000 random samples from each dataset. Each sample has 250 rows (users) and 1000 columns (items).

Table 1: Datasets used in the base-level experiments.

Dataset	#users	#items	#ratings	ratings scale	ref.
amazon-movies	73 k	4 k	111 k	[1;5]	[10]
movielens1m	6 k	4 k	1 M	[0;5]	[5]
palcoprincipal-playlist	4 k	5 k	37 k	[0;1]	[4]

Regarding the source (D_s) and target (D_t) datasets selection, we followed two main approaches: selecting two subsets from the same dataset or select-

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

ing source and target from different datasets. The first approach was applied both on *amazon-movies* and *palcoprincipal-playlist* datasets. The second approach was applied both on *movielens1m* and *palcoprincipal-playlist* datasets, selecting source (D_s) datasets from *movielens1m*, and target (D_t) datasets from *palcoprincipal-playlist*. As mentioned, the criterion considered in the experiments was the contrasting algorithm performance. This means that for the source dataset (D_s) a given algorithm A outperforms another given algorithm B and, in target dataset (D_t), B outperforms A. In the experiments, the algorithms A and B are user-based CF and item-based CF, respectively. For each dataset, we evaluated both algorithms on each sample (out of the 60000 samples). For each algorithm, we selected the top-100 samples with the highest difference in precision (*delta*). Therefore, for each dataset selection approach described above, we formed 100 pairs, selecting, for each algorithm, the 100 samples with the highest *delta* values.

Regarding the dataset transformations, we decided to use random one row replacements. We iteratively replace rows in the source dataset (D_s) with rows from the target (D_t). This enforces smoothness and trivially guarantees convergence in a pre-defined number of steps. By way of illustration, to obtain dataset D_1 we start with source dataset (D_s). We randomly sample, without replacement, one row index and copy that row from (D_t) to D_1. Likewise, dataset D_2 has all but one row from dataset D_1. This procedure is repeated until intermediate and target datasets match. In the experiments, for each pair created we obtain 250 intermediate datasets. However, the wide variety of possibilities to get from source (D_s) to target (D_t) datasets, deserves future exploration. To minimize time and computational resources, in the experiments, for each pair created, we sampled 10 different trajectories between source (D_s) to target (D_t) datasets. This means that, for each pair, we applied the random row-wise transformations in 10 different ways.

4.2 Meta-Level

One of the most important factors in the success of a MtL approach is the definition of a set of metafeatures that contain information about the performance of algorithms [1]. Part of the metafeatures used in this study are obtained procedurely [2] and are based on two different perspectives on their distribution: users and items. These distributions are aggregated, by row and by column, using simple, standard statistical functions (count and mean) and post-processing functions: 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* (e.g., *column.mean.entropy*). Other metafeatures used in this study are based on [13]. That work organizes metafeatures in five groups i.e., subsets of data characterization measures that share similarities among them [1]. Since this study focuses only on binary rating-based CF datasets, we only considered metafeatures from the Simple and Information-theoretic groups [13]. From the Information-theoretic group, we used the attributes concentration (*attrConc*) and the attributes entropy (*attrEnt*) [13]. These metafeatures were implemented using the *mfe* package.[6] Lastly, we used two other metafeatures: number of zeros of the entire dataset and sparsity [8].

The techniques used in the meta-level are usually either classification or regression. This study focuses on classification tasks. Considering the 100 identified pairs of subset datasets, the 10 different trajectories between each pair and the 250 intermediate datasets for each trajectory as a result of dataset morphing process, we created 18 meta-datasets with 250.000 meta-examples. One for each of the 3 selected original dataset (Table 1), and for each of the 6 top-N. The algorithm selection problem is formulated as a classification task, where the class label is the best algorithm, according to the precision metric. Either IB (item-based) or UB (user-based). The predictive attributes are the metafeatures described above. We did some exploratory experiments with a set of classification algorithms: Adaboost, C5.0, Gradient Boosting Machine, Logistic Regression, Naive Bayes, Random Forest, rpart and XGBoost. Regarding the classification problem, we chose the following error measures: accuracy, recall for item-based class ($Recall_{IB}$), recall for user-based class ($Recall_{UB}$) and area under the curve (AUC). We performed tuning on algorithms, optimizing the AUC metric i.e., for each meta-level algorithm we considered different values for its hyperparameters. Despite considering many trajectories for each pair, intermediate datasets of same pair are very similar to each other. Therefore, the algorithms were evaluated in a leave 20 pairs out strategy. This means that we use 80 pairs of datasets for training and leave the remaining 20 pairs for testing. Meta-learning was done using the *caret* package,[7].

5 Experimental Results

Base-level: as an example of the algorithm performance evaluation at the base-level, Figure 2 illustrates the results for *palcoprincipal-playlist* dataset, for one pair and trajectory. The performance of user-based CF is represented in blue colour and the results of item-based CF are represented in red colour. We can observe that for top-3 and top-5 tasks item-based CF starts presenting higher values at approximately (intermediate) dataset D_{100}. This means that D_{100} is on an interesting boundary and is worth looking into.
Meta-level: The exploratory data analysis allows to identify metafeatures that are good indicators of the relative performance of the algorithms. From the vast number of experiments we have performed, we show one figure, where the winning algorithm for each intermediate dataset is illustrated in different colours. Once again, user-based CF is in blue and item-based CF is in red. The presented results are only for one pair created and one trajectory of *palcoprincipal-playlist* dataset, and for the top-3 task. As an example, the metafeature illustrated in Figure 3 seems to contain useful information about the relative performance of the algorithms i.e., it varies accordingly with algorithm performance. In fact, higher

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

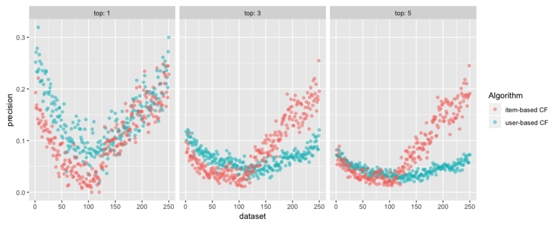

Fig. 2: Performance of IB and UB through morphing.

values of this metafeature indicate that user-based performs best. Item-based CF algorithm seems to be the winner when this metafeature has lower values. This is true for the following metafeatures: *attrConc.mean*, *column.count.mean*, *row.count.entropy*, *row.count.kurtosis* and *row.count.max*.

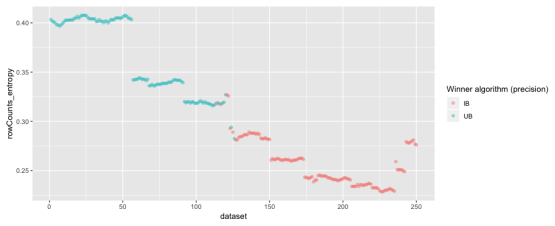

Fig. 3: Results of meta-level evaluation on *palcoprincipal-playlist* dataset — Entropy of row count.

Regarding the meta-learning results, we firstly tested the meta-models against test sets obtained within the same dataset used for training. Then, in order to ensure that the extracted metaknowledge could be generalizable, we also tested each meta-model against the test set of remaining domains. Lastly, we present performance results of meta-models created with random samples. To serve as baseline, we did some experiments training meta-models with only the source and target datasets i.e., without the intermediate datasets.

Results on Table 2 show that the meta-models created help in intra-domain algorithm selection. The meta-models obtained from *amazon-movies* and *palco principal-playlist* datasets, seem to clearly identify the item-based CF instances. On the other hand, the *movielens1m/ palcoprincipal-playlist* dataset yields a meta-model that seems to identify reasonably well both classes. The inter-domain test set results, partially support the conclusions of intra-domain results. In fact, meta-models obtained from *amazon-movies* and *palco principal-playlist* datasets, presented high values for $Recall_{IB}$ i.e., they seem to clearly identify the item-based CF instances. On the other hand, unlike the intra-domain results, the meta-model obtained from *movielens1m/ palcoprincipal-playlist* dataset presented low values for $Recall_{IB}$ and high values for $Recall_{UB}$. This seems to mean that it clearly identifies the user-based CF instances, unlike the item-based CF ones. In short, the results of these experiments show that the extracted meta-knowledge could be generalized and transferred across the studied domains.

Concerning the performance results of meta-models trained without the intermediate datasets, and based on intra-domain results, it is possible to conclude that the meta-data obtained from morphing datasets leads to better meta-knowledge, when compared to meta-data obtained from random samples. Nevertheless, considering the inter-domain results, some performance metrics (e.g., $Recall_{IB}$) presented better overall results, on meta-models trained with trajectories, and some others don't (e.g., AUC).

Table 2: Summary of results. Green (red) dots indicate that meta-models from trajectories beat (loose against) random samples. Delta is the winning margin.

Model/Test set	amazon-movies			mlens/ palco			palco		
	Metric		Delta	Metric		Delta	Metric		Delta
amazon-movies	AUC	🟢	0.21	AUC	🔴	0.13	AUC	🔴	0.01
	$Recall_{IB}$	🟢	0.36	$Recall_{IB}$	🟢	0.07	$Recall_{IB}$	🟢	0.33
	$Recall_{UB}$	🔴	0.19	$Recall_{UB}$	🔴	0.277	$Recall_{UB}$	🔴	0.45
movielens1m/ palcoprincipal-playlist	AUC	🟢	0.07	AUC	🟢	0.07	AUC	🔴	0.03
	$Recall_{IB}$	🔴	0.29	$Recall_{IB}$	🔴	0.13	$Recall_{IB}$	🔴	0.45
	$Recall_{UB}$	🟢	0.18	$Recall_{UB}$	🟢	0.17	$Recall_{UB}$	🟢	0.37
palcoprincipal-playlist	AUC	🔴	0.01	AUC	🔴	0.01	AUC	🟢	0.11
	$Recall_{IB}$	🟢	0.20	$Recall_{IB}$	🟢	0.02	$Recall_{IB}$	🟢	0.04
	$Recall_{UB}$	🔴	0.22	$Recall_{UB}$	🟢	0.25	$Recall_{UB}$	🟢	0.22

To extract meta-knowledge we assess features frequency in the best models. The best metafeatures are: *row.count.entropy*, *row.count.max*, *row.count. kurtosis*, *attrConc.mean* and *attrEnt.mean*. We observe that both rows and columns hold important characteristics to solve the algorithm selection problem. This confirms previous studies [3,2]. Nevertheless, *row.count* is the most relevant distribution to be analyzed here. Actually, the choice between user-based and item-based depends on the ratio between the number of rows and the items.

6 Conclusions

In this study, we have proposed a methodology that generates new datasets by manipulating existing ones, for understanding algorithm behavior using MtL approaches. In the experiments, the proposed methodology was used to select CF algorithms. The algorithm selection problem was formulated as a classification task, where the target attribute is the best CF algorithm, according to precision metric. The results show that the proposed approach is feasible and that it can be used to identify useful metafeatures to predict the best collaborative filtering algorithm for a given dataset. Considering the majority of the scenarios studied, the results support the importance of dataset morphing to enrich the metadata and, therefore, to improve the results of MtL processes. As future work we intend to explore the multiple avenues offered to machine learning by dataset morphing.

Acknowledgments. This work is funded by ERDF through the Operational Programme of Competitiveness and Internationalization — COMPETE 2020 — of Portugal 2020 within project PushNews — POCI-01-0247-FEDER-0024257.

References

1. Brazdil, P., Carrier, C.G., Soares, C., Vilalta, R.: Metalearning: Applications to Data Mining. Springer (2008)
2. Cunha, T., Soares, C., de Carvalho, A.C.P.L.F.: Selecting Collaborative Filtering Algorithms Using Metalearning. In: Frasconi, P., Landwehr, N., Manco, G., Vreeken, J. (eds.) Machine Learning and Knowledge Discovery in Databases. ECML PKDD 2016, pp. 393–409. Springer, Cham (sep 2016)
3. Cunha, T., Soares, C., de Carvalho, A.C.: Metalearning and Recommender Systems: A literature review and empirical study on the algorithm selection problem for Collaborative Filtering. Information Sciences **423**, 128–144 (jan 2018)
4. Domingues, M.A., Gouyon, F., Jorge, A.M., Leal, J.P., Vinagre, J., Lemos, L., Sordo, M.: Combining usage and content in an online recommendation system for music in the Long Tail. International Journal of Multimedia Information Retrieval (2013)
5. F.Maxwell, Konstan, J.A.: The MovieLens Datasets: History and Context. ACM Transactions on Intelligent Systems and Technology (TIST) (2015)
6. García-Saiz, D., Zorrilla, M.: A meta-learning based framework for building algorithm recommenders: An application for educational arena. In: Journal of Intelligent and Fuzzy Systems (2017)
7. Hahsler, M.: recommenderlab: A Framework for Developing and Testing Recommendation Algorithms. Nov (2011)
8. Massa, P., Avesani, P.: In: Proceedings of the 2007 ACM conference on Recommender systems - RecSys '07
9. Matuszyk, P., Vinagre, J., Spiliopoulou, M., Jorge, A.M., Gama, J.: Forgetting methods for incremental matrix factorization in recommender systems. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15 (2015)
10. McAuley, J., Leskovec, J.: Hidden factors and hidden topics: understanding rating dimensions with review text. Proceedings of the 7th ACM conference on Recommender systems - RecSys '13 (2015)
11. Prudêncio, R.B., Soares, C., Ludermir, T.B.: Combining Meta-learning and Active Selection of Datasetoids for Algorithm Selection. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (2011)
12. Ricci, F., Shapira, B., Rokach, L.: Recommender systems: Introduction and challenges. In: Recommender Systems Handbook, Second Edition (2015)
13. Rivolli, A., Garcia, L.P.F., Soares, C., Vanschoren, J., de Leon Ferreira de Carvalho, A.C.P.: Towards Reproducible Empirical Research in Meta-Learning. CoRR **abs/1808.1** (2018)
14. Wolberg, G.: Image morphing: A survey. Visual Computer (1998)