HAL
open science

ActivMetaL: Algorithm Recommendation with Active Meta Learning

Lisheng Sun-Hosoya, Isabelle Guyon, Michèle Sebag

To cite this version:

Lisheng Sun-Hosoya, Isabelle Guyon, Michèle Sebag. ActivMetaL: Algorithm Recommendation with Active Meta Learning. IAL 2018 workshop, ECML PKDD, Sep 2018, Dublin, Ireland. , 2018. hal-01931262

HAL Id: hal-01931262

https://hal.science/hal-01931262

Submitted on 22 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ActivMetaL: Algorithm Recommendation with Active Meta Learning

Lisheng Sun-Hosoya[1], Isabelle Guyon[1,2], and Michèle Sebag[1]

[1] UPSud/CNRS/INRIA, Univ. Paris-Saclay. [2] ChaLearn

Abstract. We present an *active meta learning* approach to model selection or algorithm recommendation. We adopt the point of view "collaborative filtering" recommender systems in which the problem is brought back to a missing data problem: given a sparsely populated matrix of performances of algorithms on given tasks, predict missing performances; more particularly, predict which algorithm will perform best on a new dataset (empty row). In this work, we propose and study an active learning version of the recommender algorithm CofiRank algorithm and compare it with baseline methods. Our benchmark involves three real-world datasets (from StatLog, OpenML, and AutoML) and artificial data. Our results indicate that CofiRank rapidly finds well performing algorithms on new datasets at reasonable computational cost.

Keywords: Model Selection · Recommender · Active Meta Learning.

1 Introduction

While Machine Learning and Artificial Intelligence are taking momentum in many application areas ranging from computer vision to chat bots, selecting the best algorithm applicable to a novel task still requires human intelligence. The field of AutoML (Automatic Machine Learning), aiming at automatically selecting best suited algorithms and hyper-parameters for a given task, is currently drawing a lot of attention. Progress in AutoML has been stimulated by the organization of challenges such as the AutoML challenge series[1]. Among the winning AutoML approaches are AutoWeka and Auto-SkLearn, developed by the Freiburg team [6,5,7] (more in Section 2). These approaches, taking inspiration from Bayesian optimization [4], alternatively learn an inexpensive estimate of model performance on the current dataset, and use this estimate to reduce the number of model candidates to be trained and tested using the usual expensive cross-validation procedure. A novel ingredient of AutoSkLearn, referred to as "meta-learning", takes in charge the initialization of the Bayesian optimization process, with a predictor using "meta-features" describing the datasets. Meta-learning reportedly yields significant improvements over random initializations.

[1] http://automl.chalearn.org

Another approach targeting AutoML is based on recommender systems (RS), popularized by the Netflix challenge [2]. RS approaches seek the item best suited to a given user, based on historical user-item interactions and user preferences. By analogy [15] proposed first to treat algorithm selection as a recommender problem in which datasets "prefer" algorithms solving their task with better performance. Along this line, the "Algorithm Recommender System" ALORS [9], combines a recommender system and an estimate of model performance based on predefined meta-features, to achieve AutoML (more in Section 2).

In this paper, we propose an *active meta-learning* approach inspired by AUTOSKLEARN and ALORS. Formally (Section 3), given a matrix of historical algorithm performance on datasets, we aim at finding *as fast as possible* the model with best performance on a new dataset. The originality compared to the former approaches lies in the coupled search for the meta-features describing the dataset, the model performance based on these meta-features, and the selection of a candidate model to be trained and tested on the dataset.

This paper is organized as follows: After briefly reviewing the SOTA in Section 2, we formalize our problem setting in Section 3. We then describe the benchmark data in Section 4 and provide an empirical validation of the approach in Section 5. While the validation considers only the "classical" machine learning settings, it must be emphasized that the proposed approach does not preclude of any type of tasks or algorithms, hence is applicable to a broader range of problems.

2 State of the art

It is notorious that the success of model search techniques can be dramatically improved by a careful initialization. In AUTOSKLEARN, the search is improved by a sophisticated initialization using a form of transfer learning [10] called "meta-learning". The meta-data samples include all the datasets of openml.org [12] (a platform which allows to systematically run algorithms on datasets). Systematically launching AUTOSKLEARN on each dataset yields the best (or near best) models associated with each dataset.

Independently, each dataset is described using so-called meta-features. Meta-features are generally of two kinds: i) simple statistics of the dataset such as number of training examples, number of features, fraction of missing values, presence of categorical variables, etc.; ii) performance on the current dataset of "landmark algorithms", namely a well-chosen set of algorithms that can be trained and tested with moderate computational effort such as one nearest neighbor (1NN) or decision trees.

When considering a new dataset, AUTOSKLEARN first determines its nearest neighbors in the meta-feature space, and initializes the search using the best models associated with these neighbors. Other meta-learning formalisms, not considered further in this paper, are based on learning an estimate of the model performance from meta-features [11], or learning to predict the best performing algorithm, as a multi-class classification problem [17].

The delicate issue is to control the cost of the initialization step: considering many landmark algorithms comes with an expected benefit (a better initialization of the search), and with a cost (the computational cost of running the landmarks).

As said, recommender systems (RS) aim at selecting the item best suited to a particular user, given a community of users, a set of items and some historical data of past interactions of the users with the items, referred to as "collaborative matrix" [16,3], denoted S (for "score") in this paper. As first noted by [15], algorithm selection can be formalized as a recommender problem, by considering that a dataset "likes better" the algorithms with best performances on this dataset. Along this line, one proceeds by i) estimating all algorithm performances on this dataset (without actually evaluating them by training and testing); and ii) recommending the algorithm(s) with best estimated performance on this dataset.

The merits of RS approaches regarding algorithm selection are twofold. Firstly, **RS approaches are frugal** (like other methods, e.g. co-clustering). RS proceeds by estimating the value associated with each (user, item) pair − here, the performance associated with each (algorithm, dataset) − from a tiny fraction of the (user, item) ratings, under the assumption that the collaborative matrix is of low rank k. More precisely the (usually sparse) matrix S of dimensions (p, N) is approximated by UV', with U a (p, k) matrix and V a (N, k) matrix, such that $\langle U_{i,\cdot}, V_{j,\cdot}\rangle$ is close to $\mathrm{S}_{i,j}$ for all pairs i, j (e.g. using maximum margin matrix factorization in [14]). U (respectively V) is referred to as latent representation of the users (resp. the items). In the model selection context, RS approaches are thus frugal: they can operate even when the performance of a model on a dataset is known on a tiny fraction of the (model, dataset) pairs. Secondly, most-recent **RS approaches are ranking methods.** Estimating algorithm performance is a harder problem than ranking them in order of merit. A second benefit of RS is that they can rank items conditionally to a given user. The CofiRank algorithm [18] accordingly considers the rank matrix (replacing $\mathrm{S}_{i,j}$ with the rank of item j among all items user i has rated) and minimizes the Normalized Discounted Cumulative Gain (NDCG) in which correctness in higher ranked items is more important. As optimizing NDCG is non-convex, CofiRank thus instead optimizes a convex upper-bound of NDCG.

In counterpart for these merits, mainstream RS is not directly applicable to AutoML, as it focuses on recommending items to known users (warm-start recommendation). Quite the contrary, AutoML is concerned with recommending items (models) to new users (new datasets), a problem referred to as cold-start recommendation [13,8]. This drawback is addressed in the general purpose ALORS system [9], where external meta-features are used to estimate the latent representation $\hat{U}_{\cdot}$ of the current dataset; this estimated latent representation is used together with the latent representation of any model to estimate the model performance (as $\langle \hat{U}_{\cdot}, V_j\rangle$) and select the model with best estimated performance. The novel active meta-learning approach presented in this paper proposes a different approach to warm start, not requiring external meta-features: Previously evaluated algorithm scores are themselves used as meta-features (see Section 3).

3 Problem setting and algorithms

We define the **active meta-learning problem** in a **collaborative filtering recommender** setting as follows:

GIVEN:

- An ensemble of **datasets** (or tasks) $\mathcal{D}$ of elements d (not necessarily finite);
- A finite ensemble of n **algorithms** (or machine learning models) $\mathcal{A}$ of elements $a_j, j = 1, \cdots, N$;
- A **scoring program** $\mathcal{S}(d, a)$ calculating the performance (score) of algorithm a on dataset d (*e.g.* by cross-validation). Without loss of generality we will assume that **the larger $\mathcal{S}(d, a)$, the better**. The evaluation of $\mathcal{S}(d, a)$ can be computationally expensive, hence we want to limit the number of times S is invoked.
- A **training matrix** S, consisting of p lines (corresponding to example datasets $d_i, i = 1, \cdots p$ drawn from $\mathcal{D}$) and n columns (corresponding to all algorithms in $\mathcal{A}$), whose elements are calculated as $\mathrm{S}_{ij} = \mathcal{S}(d_i, a_j)$, but may contain missing values (denoted as NaN).
- A **new test dataset** $d_t \in \mathcal{D}$, NOT part of training matrix S. This setting can easily be generalized to test matrices with more than one line.

GOAL: Find "as quickly as possible" $j_* = argmax_j(S(d_t, a_j))$.

For the purpose of this paper "as quickly as possible" shall mean by evaluating as few values of $S(d_t, a_j), j = 1, \cdots, n$ as possible. More generally, it could mean minimizing the total computational time, if there is a variance in execution time of $S(d_t, a_j)$ depending on datasets and algorithms. However, because we rely in our experimental section on archival data without information of execution time, we reserve this refinement for future studies. Additionally, we assume that the computational cost of our meta-learning algorithm (excluding the evaluations of $\mathcal{S}$) is negligible compared to the evaluations of $\mathcal{S}$, which has been verified in practice.

In our setting, we reach our goal iteratively, in an **Active Meta Learning** manner (ActivMetaL), see Algorithm 1. The variants that we compare differ in the choices of InitializationScheme(S) and SelectNext($S, \mathbf{t}$), as described in Algorithms 2-5: Given a new dataset (an empty line), we need to initialize it with one or more algorithm performances, this initialization is done by InitializationScheme(S) and is indispensable to fire CofiRank. Algorithms 2-5 show 2 initialization methods: randperm in Algorithm 2 (the first algorithm is selected at random) and median in Algorithms 3-5 (the algorithms are sorted by their median over all datasets in training matrix S and the one with highest median is selected as the first algorithm to evaluate). Once we have evaluated the first algorithm, the next algorithms can be chosen with or without active learning, this is done by SelectNext($S, \mathbf{t}$): Algorithm 2-3 without active meta learning select next algorithms at random or according to median over training datasets, i.e. the knowledge from evaluated algorithms on the new dataset is not taken into account; Algorithm 4-5 run CofiRank for active meta learning, which, initialized with performances of evaluated algorithm, returns a ranking of algorithms on the new dataset. The difference is that in Alg. 4 we run CofiRank for

each selection of next algorithm, i.e. CofiRank is initialized with more and more known values. In Alg. 5 CofiRank is run only once at the beginning, initialized with 3 landmark values.

Algorithm 1 ActivMetaL

```
procedure ActivMetaL(𝒜, 𝒮, S, d_t, n_max)
    n ← size(S, 2)                          ▷ Number of algorithms to be evaluated on d_t
    t ← NaNvector(n)                        ▷ Algorithm scores on d_t are initialized w. missing values
    j_+ ← InitializationScheme(S)           ▷ Initial algorithm a_{j+} ∈ 𝒜 is selected
    while n < n_max do
        t[j_+] ← 𝒮(d_t, a_{j+})             ▷ Complete t w. one more prediction score of a_{j+} on d_t
        j_+ = SelectNext(S, t)
        n ← length(notNaN(t))               ▷ number of algorithms evaluated on d_t
    return j_+
```

Algorithm 2 Random

```
procedure InitializationScheme(S)
    r ← randperm(size(S, 2))                ▷ Replaced by something more clever elsewhere
    return j_+ ← argmax(r)
procedure SelectNext(S,t)
    evaluated ← notNaN(t)
    r ← randperm(size(S, 2))                ▷ Replaced by something more clever elsewhere
    r(evaluated) ← −Inf
    return j_+ ← argmax(r)
```

Algorithm 3 SimpleRankMedian

```
procedure InitializationScheme(S)
    r ← median(S, 2)                        ▷ Column-wise median
    return j_+ ← argmax(r)
procedure SelectNext(S,t)
    evaluated ← notNaN(t)
    r ← median(S, 2)                        ▷ Column-wise median
    r(evaluated) ← −Inf
    return j_+ ← argmax(r)
```

Algorithm 4 ActiveMetaLearningCofiRank

```
procedure InitializationScheme(S)
    r ← median(S, 2)                        ▷ Column-wise median
    return j_+ ← argmax(r)
procedure SelectNext(S,t)
    evaluated ← notNaN(t)
    r ← CofiRank(S,t)                       ▷ Collaborative filtering on [S; t] returning last line
    r(evaluated) ← −Inf
    return j_+ ← argmax(r)
```

Algorithm 5 MedianLandmarks1CofiRank

```
procedure InitializationScheme(S)
    r ←median(S, 2)                                   ▷ Column-wise median
    return j+ ← argmax(r)
procedure SelectNext(S,t)
    evaluated ←notNaN(t)
    if length(evaluated) < num_landmarks then
        r ←median(S, 2)                               ▷ Column-wise median
    else if length(evaluated) == num_landmarks then
        static r ← CofiRank(S,t)        ▷ Keep the CofiRank predictions thereafter
    r(evaluated) ← −Inf
    return j+ ← argmax(r)
```

4 Benchmark data

To benchmark our proposed method, we gathered datasets from various sources (Table 1). Each dataset consists of a matrix S of performances of algorithms (or models) on tasks (or datasets). Datasets are in lines and algorithms in columns. The performances were evaluated with a single training/test split or by cross-validation. The tasks were classification or regression tasks and the metrics quasi-homogeneous for each S matrix (*e.g.* Balanced Accuracy a.k.a. BAC for classification and R^2 for regression). We excluded data sources for which metrics were heterogeneous (a harder problem that we are leaving for further studies). Although ActivMetaL lends itself to using sparse matrices S (with a large fraction of missing values), these benchmarks include only full matrices S.

The artificial dataset was constructed from a matrix factorization to create a simple benchmark we understand well, allowing to easily vary the problem difficulty. Matrix S is simply obtained as a product of three matrices $U\Sigma V$, U and V being orthogonal matrices and Σ a diagonal matrix of "singular values", whose spectrum was chosen to be exponentially decreasing, with $\Sigma_{ii} = exp(-\beta i)$, $\beta = 100$ in our experiments. The other benchmarks were gathered from the Internet or the literature and represent the performances of real algorithms on real datasets. We brought back all metrics to scores that are "the larger the better". In one instance (StatLog), we took the square root of the performances to equalize the distribution of scores (avoid a very long distribution tail). For AutoML, many algorithms were aborted due to execution time constraints. We set the corresponding performance to 0. To facilitate score comparisons between benchmark datasets, all S matrices were globally standardized (*i.e.* we subtracted the global mean and divided by the global standard deviation). This scaling does not affect the results.

We conducted various exploratory data analyses on the benchmark data matrices, including two-way hierarchical clustering, to visualize whether there were enough similarities between lines and columns to perform meta-learning. See our supplemental material referenced at the end of this paper.

Table 1: **Statistics of benchmark datasets used**. *#Datasets=number of datasets, #Algo=number of algorithms, Rank=rank of the performance matrix.*

	Artificial	Statlog	OpenML	AutoML
#Dataset	50	21	76	30
#Algo	20	24	292	17
Rank	20	21	76	17
Metric	None	Error rate	Accuracy	BAC or R^2
Preprocessing	None	Take square root	None	Scores for aborted algo. set to 0
Source	Generated by authors	Statlog Dataset in UCI database	Alors [9] website	AutoML1 (2015-2016)

5 Results

Table 2: **Results of meta-learning methods for all 4 meta-datasets**. *Performances of meta-learning algorithms are measured as the area under the meta learning curve (AUMLC) normalized by the area of the best achievable curve. Active_Meta_Learning w. CofiRank (our proposed method) performs always best, although not significantly considering the 1-sigma error bars of the leave-one-dataset-out procedure.*

	Artificial	Statlog	OpenML	AutoML
Active_Meta_Learning w. CofiRank	**0.91** (±**0.03**)	**0.802** (±**0.117**)	**0.96** (±**0.04**)	**0.84** (±**0.11**)
Random	0.81 (±0.05)	0.77 (±0.05)	0.95 (±0.03)	0.79 (±0.07)
SimpleRank w. median	0.7 (±0.2)	0.798 (±0.102)	0.95 (±0.04)	0.82 (±0.12)
Median_LandMarks w. 1-CofiRank	0.88 (±0.04)	0.795 (±0.099)	0.92 (±0.08)	0.83 (±0.11)

In this section, we analyze the experimental results of Table 2 and Figure 1. The graphs represent meta-learning curves, that is the performance of the best algorithm found so far as a function of the number of algorithms tried.[2] The ground truth of algorithm performance is provided by the values of the benchmark matrices (see Section 4).

We remind the reader that in a meta-learning problem, each sample is a dataset. To evaluate meta-learning we use the equivalent of a leave-one-out estimator, *i.e.* leave-one-dataset-out. Hence, we use as meta-learning training data

[2] In the future, when we have meta-learning datasets for which the computational run time of algorithms is recorded, we shall tackle the harder and more interesting problem of meta-learning performance as a function of "total" computational time rather than number of algorithms tried.

all datasets but one, then create the learning curve for the left-out dataset. Thus, given a benchmark data matrix, we generate meta-learning curves using as matrix S a sub-matrix with one line left out (held out), which serves as target vector $\mathbf{t}$ for the dataset tested. Subsequently, we average all meta-learning curves, step-by-step. Thus the result shown in Figure 1 are the averaged learning curves obtained with the leave-one-dataset-out scheme, *i.e.* averaged over all datasets, for a given benchmark dataset.

To evaluate the significance of the efficiency of our proposed method, we ran 1000 times the Random search algorithm, in which algorithms are ran in a random sequence. We drew the curves of median performance (blue curves) and showed as blue shadings various quantiles. The fact that the red curves, corresponding to the proposed algorithm Active Meta Learning w. CofiRank is generally above the blue curve comforts us that the method is actually effective. It is not always significantly better than the median of Random search. However, this is a very hard benchmark to beat. Indeed, the median of Random search is not a practical method, it is the average behavior of random search over many runs. Thus, performing at least as good as the median of Random search is actually pretty good.

We also compared our method with two other baselines. (1) The SimpleRank w. median (green curves) uses the median performance of algorithms on all but the left-out dataset. Thus it does not perform any *active* meta-learning. (2) The Median Landmark w. 1 CofiRank (pink curves) makes only one call of CofiRank to reduce computational expense, based on the performance of only 3 Landmark algorithms, here simply picked based on median ranking.

The first benchmark using artificial data (Figure 1(a)) a relative position of curves that we intuitively expected: *SimpleRankw.median* (in green) does not perform well and Active Meta Learning w. CofiRank (in red) is way up in the upper quantiles of the random distribution, close to the ideal curve that goes straight up at the first time step (selects right away the best algorithm). Median Landmark w. 1 CofiRank (in pink) quickly catches up with the red curve: this is promising and shows that few calls to CofiRank might be needed, should this become a computational bottleneck.

However, the analysis of the results on real data reveals a variety of regimes. The first benchmark using the datasets of the AutoML challenge (Figure 1(b)) gives results rather similar to artificial data in which Active Meta Learning w. CofiRank still dominates, though SimpleRank w. median performs surprisingly well. More surprisingly, Active Meta Learning w. CofiRank does not beat SimpleRank w. median on the StatLog benchmark and beats it with difficulty (after more than 10% of the algorithms have been trained/tested) on the OpenML benchmark. Also, the cheap algorithm calling CofiRank just once (Median Landmark w. 1 CofiRank, performing no active learning) which looked promising on other benchmark datasets, performs poorly on the OpenML dataset. This is unfortunate since this is the largest dataset, on which running active-learning is most computationally costly. We provide a discussion of computational considerations in Section 6.

Table 2 sums up the results in terms of area under the meta-learning curves (AUMLC). Active Meta Learning w. CofiRank consistently outperforms other methods, although not significantly according to the error bars.

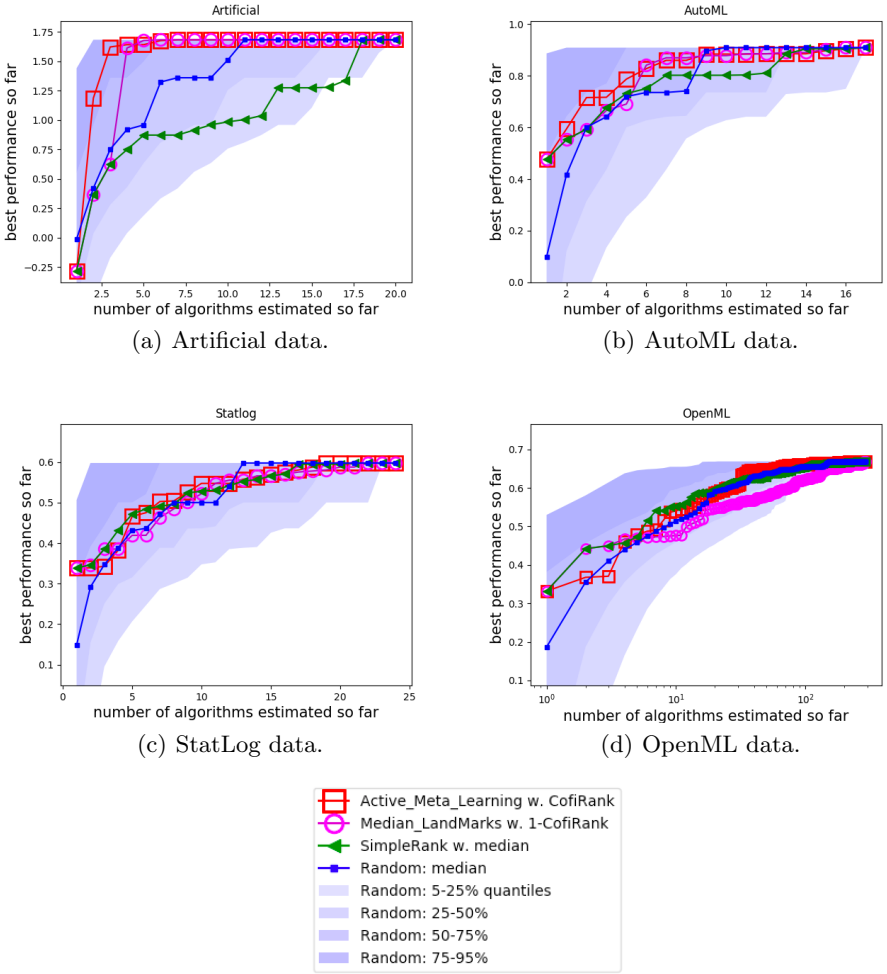
6 Discussion and conclusion

We have presented an approach to algorithm recommendation (or model selection) based on meta-learning, capitalizing on previous runs of algorithms on a variety of datasets to rank candidate algorithms and rapidly find which one will perform best on a new dataset. The originality of the paper lies in its active learning approach based on a collaborative-filtering algorithm: CofiRank. Collaborative filtering is a technique to fill in missing data in a collaborative matrix of scores, which in our case represents performances of algorithms on datasets. Starting from the evaluation of a single algorithm on a new dataset of interest, the CofiRank method ranks all remaining algorithms by completing the missing scores in the collaborative matrix for that new dataset. The next most promising algorithm is then evaluated and the corresponding score added to the collaborative matrix. The process is iterated until all missing scores are filled in, by trying all algorithms, or until the allotted time is exhausted.

We demonstrated that Active Meta Learning w. CofiRank performs well on a variety of benchmark datasets. Active Meta Learning w. CofiRank does not always beat the naive SimpleRank w. median baseline method, but it consistently outperforms the "hard-to-beat" median of Random ranking, while SimpleRank w. median does not.

We also investigated whether the (meta-) active learning aspect is essential or can be replaced by running CofiRank a single time after filling in a few scores for Landmark algorithms. This technique (called Median Landmark w. 1 CofiRank) seemed promising on the smallest benchmark datasets, but gives significantly worse results that Active Meta Learning w. CofiRank on the largest benchmark dataset on which it would help most (computationally). One avenue of future research would be to put more effort in the selection of better Landmarks.

Further work also includes accounting for the computational expense of model search in a more refined way. In this work, we neglected the cost of performing meta-learning compared to training and testing the algorithms. This is justified by the fact that their run time is a function of the volume of training data, which is considerably smaller for the collaborative matrix (of dimension usually $\simeq 100$ datasets times $\simeq 100$ algorithms) compared to modern-times "big data" datasets (tens of thousands of samples times thousands of features). However, as we acquire larger meta learning datasets, this cost may become significant. Secondly, we assumed that all algorithms had a comparable computational time (to be able to use meta-learning datasets for which this information was not recorded). In the future, we would like to take into account the duration of each algorithm to better trade-off accuracy and computation. It is also worth noting that ActivMetaL does not optimize the exploration/exploitation trade-off. It is more geared toward exploitation than exploration since the next best algorithm



(a) Artificial data. (b) AutoML data.

(c) StatLog data. (d) OpenML data.

Fig. 1: **Meta-learning curves.** *We show results of 4 methods on 4 meta-learning datasets, using the leave-one-dataset-out estimator. The learning curves represent performance of the best model trained/tested do far, as a function of the number of models tried. The curves have been averaged over all datasets held-out. The method Active Meta Learning w. CofiRank (red curve) generally dominates other methods. It always performs at least as well as the median of random model selection (blue curve), a hard-to-beat benchmark. The more computationally economical Median Landmark w. 1 CofiRank consisting in training/testing only 3 models (Landmarks) to rank methods using only 1 call to CofiRank (pink curve) generally performs well, except on OpenML data for which it would be most interesting to use it, since this is the largest meta learning datasets. Thus active learning cannot easily be replaced by the use of Landmarks, lest more work is put into Landmark selection. The method SimpleRank w. median that ranks algorithm with their median performance (green curve) is surprisingly a strong contender to Active Meta Learning w. CofiRank for the StatLog and OpenML datasets, which are cases in which algorithms perform similarly on all datasets.*

is chosen at every step. Further work may include incorporating monitoring the exploration/exploitation trade-off. In particular, as said, so far we have not taken into account the computational cost of running algorithms. When we have a total time budget to respect, exploring first using faster algorithms then selecting slower (but better) algorithms may be a strategy that ActivMetal could adopt (thus privileging first exploration, then exploitation).

At last, the experiments performed in this paper assumed that, except to the new dataset being tested, there were no other missing values in the collaborative matrix. One of the advantages of collaborative filtering techniques is that they can handle matrices sparsely populated. This deserves further investigation.

Supplemental material, data and code

For full reproducibility of our results, datasets and code are available on Github. To run it, CofiRank must be installed. We recommend using the Docker [1] image we built for this purpose. Please refer to the Github repository for all instructions. Our repository also includes a Jupyter-notebook with additional graphs referred to in the text.

References

1. Docker. https://www.docker.com/
2. Bennett, J., Lanning, S., Netflix: The Netflix prize. KDD Cup and Workshop in conjunction with ACM SIGKDD p. 201–206 (2007)
3. Bobadilla, J., Ortega, F., Hernando, A., Gutiérrez, A.: Recommender systems survey. Knowledge-Based Systems **46**, 109–132 (2013)
4. Eggensperger, K., Feurer, M., Hutter, F., Bergstra, J., Snoek, J., Hoos, H., Leyton-Brown, K.: Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In: NIPS workshop on Bayesian Optimization in Theory and Practice (2013)
5. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Proceedings of the Neural Information Processing Systems, pp. 2962–2970 (2015), https://github.com/automl/auto-sklearn
6. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Methods for improving bayesian optimization for automl. In: Proceedings of the International Conference on Machine Learning 2015, Workshop on Automatic Machine Learning (2015)
7. Feurer, M., Springenberg, J., Hutter, F.: Initializing bayesian hyperparameter optimization via meta-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 1128–1135 (2015)
8. Gunawardana, A., Meek, C.: Tied boltzmann machines for cold start recommendations. In: Proceedings of the 2008 ACM conference on Recommender systems. pp. 19–26. ACM (2008)
9. Mısır, M., Sebag, M.: Alors: An algorithm recommender system. Artificial Intelligence **244**, 291–314 (2017)

10. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2010)
11. Rice, J.: The algorithm selection problem. Advances in computers **15**, 65–118 (1976)
12. van Rijn, J., Bischl, B., Torgo, L., Gao, B., Umaashankar, V., Fischer, S., Winter, P., Wiswedel, B., Berthold, M., Vanschoren, J.: OpenML: A collaborative science platform. In: Blockeel, H., Kersting, K., Nijssen, S., Železný, F. (eds.) Proceedings of the Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD Part III, LNCS, vol. 8190, pp. 645–649. Springer (2013)
13. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 253–260. ACM (2002)
14. Srebro, N., Rennie, J., Jaakkola, T.: Maximum-margin matrix factorization. Advances in neural information processing systems **17**(5), 1329–1336 (2005)
15. Stern, D., Herbrich, R., Graepel, T., Samulowitz, H., Pulina, L., Tacchella, A.: Collaborative expert portfolio management. In: AAAI. pp. 179–184 (2010)
16. Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Advances in artificial intelligence **2009**, 4 (2009)
17. Sun-Hosoya, L., Guyon, I., Sebag, M.: Lessons learned from the automl challenge. In: Conférence sur l'Apprentissage Automatique 2018. Rouen, France (June 2018)
18. Weimer, M., Karatzoglou, A., Le, Q., Smola, A.: CofiRank-maximum margin matrix factorization for collaborative ranking. In: Proceedings of the 21st Annual Conference on Neural Information Processing Systems (NIPS). pp. 222–230 (2007)