



Using Result Profiles to Drive Meta-learning

Krzysztof Grąbczewski(✉)

Institute of Engineering and Technology, Faculty of Physics,
Astronomy and Informatics, Nicolaus Copernicus University,
ul. Grudziądzka 5, 87-100 Toruń, Poland
kg@is.umk.pl

Abstract. Knowledge gained by meta-learning processes is valuable when it can be successfully used in solving algorithm selection problems. There is still strong need for automated tools for learning from data, performing model construction and selection with little or no effort from human operator. This article provides evidence for efficacy of a general meta-learning algorithm performing validations of candidate learning methods and driving the search for most attractive models on the basis of an analysis of learning results profiles. The profiles help in finding similar processes performed for other datasets and pointing to promising learning machines configurations. Further research on profile management is expected to bring very attractive automated tools for learning from data. Here, several components of the framework have been examined and an extended test performed to confirm the possibilities of the method. The discussion also touches on the subject of testing and comparing the results of meta-learning algorithms.

Keywords: Meta-learning · Meta-search · Algorithm selection · Algorithm ranking · Result profiles

1 Introduction

Many learning algorithms have been proposed by scientific community to solve problems like classification, approximation, clustering, time series prediction and so on. Large amount of algorithms and their different performance in particular applications have risen questions about ways to select the most adequate and the most successful learning methods for particular tasks. Such algorithm selection problems, also solved with machine learning, are tasks of *meta-level learning* (or just *meta-learning*, ML) and by analogy to the vocabulary used in logic, the former group of learning approaches, can be referred to as *object-level learning* (though more often called *base-level learning*).

In fact, the term *meta-learning* has been used in many different contexts, so that it encompasses the whole spectrum of techniques aiming at gathering meta-knowledge and exploiting it in learning processes. Although many different particular goals of meta-learning have been defined, the ultimate goal is to

use meta-knowledge for finding more accurate models at object-level and/or to find them with as little resources as possible. This article presents a general algorithm for meta-learning based on results profiles, and examines the gains of its particular implementation. The method facilitates arbitrary profile management for discovering datasets in available knowledge base characterized by similar results, to draw conclusions about which learning machines are likely to provide good results.

Some selected, interesting approaches to meta-learning, more related to the approach proposed here, are superficially reviewed in Sect. 2. Section 3 presents the motivation and general idea of the proposed algorithm, which is presented in detail in Sect. 4. Its several incarnations are tested on 77 datasets and 212 object-level learning machines. The test is described in Sect. 5 and its results analyzed in Sect. 6.

2 Meta-learning for Ranking Learning Machines

Many different approaches have been proposed as tools for meta-learning. Here, we focus on meta-learning techniques capable of reliable prediction which methods would provide best results for a given learning task. In other words, we need tools to generate accurate rankings of learning algorithms for particular problems. The task of ranking algorithms can be regarded as equivalent to the *algorithm selection problem* (ASP) discussed already in 1970s by Rice [16]. He presented an abstract model of the problem as in Fig. 1. The goal is to find a mapping $S : \mathcal{D} \rightarrow \mathcal{A}$, such that for given data $D \in \mathcal{D}$, $A = S(D)$ is an algorithm maximizing some norm of performance $\|p(A, D)\|$. The problem has been addressed by many researchers and undertaken from different points of view [9, 17]. Often, the task gets reduced to the problem of assigning optimal algorithm to a vector of features describing data. Such approaches are certainly easier to handle, but the conclusions they may bring are also limited. Separating meta-learning (ranking) and object-level learning processes simplifies the task, but implies resignation from on-line exploitation of meta-knowledge resulting from object-learners validation.

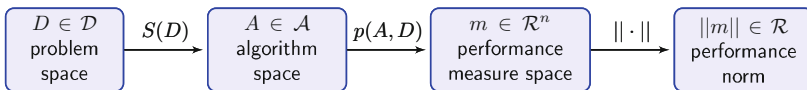


Fig. 1. Rice's model of algorithm selection problem

Building successful tools for automated selection of learning methods, most suitable for particular tasks, requires integration of meta-level and object-level learning in a single search process with built-in validation of object-level learning machines and meta-knowledge acquisition and exploitation. A general algorithm for such kind of meta-learning based on learning machine complexity estimation

has been proposed [11], but it can not be successful in cases, where the complexity estimation can not successfully drive the meta-search.

The most popular approach to meta-learning was initiated by the meta-learning project *MetaL*. It was focused on learning rankings of algorithms from simple descriptions of data (collections of *meta-attributes*). In first approaches, meta-attributes were basic data characteristics like the number of instances in the dataset, the number of features, the types of features (continuous or discrete, how many of which), data statistics, some numbers obtained with some more advanced statistical analysis and information theory [5] and so on. Provided a description of a dataset, rankings were generated by *meta-learners* in such a way that for each pair of algorithms to be ranked, a classification algorithm was trained on two-class datasets describing wins and losses of the algorithms on some collection of datasets and after that, decisions of the meta-classifiers were combined to build final rankings.

An interesting step forward was *landmarking* [15]. The idea was to use meta-features measuring the performance of some simple and efficient learning algorithms (*landmarkers*) like linear discriminant learners, naive Bayesian learner or C5.0 decision tree inducer. The results provided by the landmarkers were used as features describing data for further meta-learning.

Interesting further improvements were: *relative landmarking* that introduced meta-attributes describing relations between results [7], *typed higher-order inductive learning* [2], deriving data characteristics from the structures of C5.0 decision trees [14], and many others [1].

Many other interesting algorithms of similar goal are also worth mentioning here. Some of them merge model selection and hyperparameter optimization, often relying on Bayesian optimization with different internal solutions. Random forests are successfully applied in Auto-WEKA [10,19] and AUTO-SKLEARN [6]. Gaussian processes find application in a meta-search for appropriate kernels in model space, treating model evidence as a function to be maximized [12]. An evolutionary algorithm was used within Tree-based Pipeline Optimization Tool (TPOT) to automatically design and optimize machine learning pipelines [13].

Algorithm Ranking Criteria. To generate rankings of learning algorithms, the results of meta-learners have usually been combined with such criteria as *average ranks* (AR), *success rate ratios* (SRR), *significant wins* (SW, counting statistically significant differences between pairs of algorithms) or *adjusted ratio of ratios* (ARR, combining ratios of accuracy and time) and *relative landmark* (RL, proposed for cases involving $n > 2$ algorithms) [18]:

$$ARR_{i,j}^d = \frac{\frac{A_i^d}{A_j^d}}{1 + \log\left(\frac{T_i^d}{T_j^d}\right) * X}, \quad (1)$$

$$RL_i^d = \frac{\sum_{j \neq i} ARR_{i,j}^d}{n - 1}. \quad (2)$$

where A_i^d and T_i^d are accuracy and time of the i 'th landmarker on data d and X is a parameter interpreted as "the amount of accuracy we are willing to trade for 10-times speed-up".

A serious disadvantage of methods like SRR, ARR and SW is their quadratic complexity, so when the number of candidate algorithms is large, they are not applicable. Moreover, they are strongly dependent on the collection of methods included in the comparison – when we need to test one more algorithm, all the indices must be recalculated.

The quality of rankings generated by meta-learning algorithms has usually been measured by means of its similarity to the ideal ranking calculated for the data. The similarity has been measured with some statistical methods like Spearman’s rank correlation coefficient, Friedman’s significance test and Dunn’s multiple comparison technique. Such measures pay the same attention to rank differences at the beginning and at the end of the ranking, which is not what we really expect from meta-learning algorithms. We need the best methods at the top of the ranking while it is not so important whether the configuration is ranked 50000th or 100000th.

3 Motivation – From Passive to Active Ranking

A possibility of ranking algorithms ideally, without running them on the data at hand, would be found with great enthusiasm by many data analysts. But the task is very hard, even if significantly restricted to particular kind of data or/and small set of algorithms to be ranked. Very naive approaches basing on simple data characteristics are doomed to failure, because the information about the methods’ eligibility is usually hidden deeply in complex properties of the data like the shapes of decision borders, not in simple characteristics of the current form of the data.

Landmarking is definitely an interesting direction, because it tries to take advantage of the knowledge extracted by landmarkers in their learning processes. But it is still passive in the sense that the selection of landmarkers and of meta-features is done once at the beginning of the process. The result is also static, because the ranking is not verified—no feedback is expected and no adaptation is performed.

To provide a trustworthy decision support system the rankings should be validated. No human expert would blindly believe in raw rankings. Instead they repeatedly construct and test complex models that encompass data preprocessing and final target learners. Meta-learning tools should proceed in a similar way. Therefore it seems reasonable to organize meta-learning approaches as search processes, exploring the space of complex learning machine configurations [4] augmented with heuristics created and adjusted according to proper meta-knowledge coming from human experts and artificial processes, protecting against spending time on learning processes of poor promise and against combinatorial explosion [11].

The Idea of Profiles. Profile analysis proposed here follows the idea that the relative differences between the results obtained by different machines may point the directions to more attractive machine configurations. Such informative results

are gathered into collections called *profiles* and are used to predict the areas in the machine space worth further investigation. The idea is similar to relative landmarking, but in contrast to it, profiles are handled in an active way—there is no a priori definition of landmarks, but a dynamic profile of results is created—its contents may change in time, when the feedback shows that the current profile predictions are not satisfactory. The profiles can contain the results of arbitrary selection of learning machines (any machine can be a landmarker), and on the basis of profile similarity between learning tasks, the machines most successful in solving similar problems are selected as candidates for best results providers. The details on the possibilities of profile definitions, ranking generation and the whole active search process are described in the following section.

4 Result Profile Driven Meta-learning

Result *profiles* are collections of results obtained in validation processes for selected machine configurations. The configurations are not fixed a priori as in classical landmarking methods, but are updated on-line during the search process. Thus, the profiles can be called *active*. Within the search, subsequent candidates are validated to verify which machines are really efficient in learning from the data at hand. Using profiles for ranking other methods may take advantage of relations between results to point promising directions of machine parameter changes. Because of the similarity to relative landmarking, the technique can also be called an *adaptive (or active) relative landmarking*.

The Algorithm. Result profile driven meta-learning (RPDML) searches for successful learning algorithms within a given set of candidate machine configurations \mathcal{C} . It runs a procedure called *validation scenario* (VS) for selected machine configurations to get some measure of the quality of the machine, in the form of an element of an ordered set \mathcal{R} . An important part of RPDML is a profile manager (PM) responsible for the profile contents and for using the profile for selection of subsequent candidate machine configurations for validation. The list of parameters includes also the time deadline for the whole process. The procedure of RPDML is written formally as Algorithm 1. It operates on four collections of machine configurations:

- C_R – a collection of pairs $\langle c, r \rangle \in \mathcal{C} \times \mathcal{R}$ of machine configurations c validated with results r ,
- $C_P \subseteq C_R$ – the profile – a collection of selected results,
- C_Q – a sequence of candidate configurations (the queue) ordered with respect to estimated qualities and step numbers at which they were added,
- C_B – used temporarily in the algorithm to represent current ranking of candidate machine configurations.

To avoid long but fully formal statements, some shortcuts will be taken below, for example, the expression “a configuration in the profile” should be understood as an appropriate pair and so on. Although slightly informal, they will not introduce ambiguities and will significantly simplify descriptions.

Algorithm 1 (Result profile driven ML)

Input: set C of machine configurations, validation scenario (VS), profile manager (PM), time deadline.

Output: Machine configuration ranking.

The algorithm:

1. $C_R \leftarrow \emptyset, C_Q \leftarrow \emptyset$
 2. $step \leftarrow 0$
 3. Initialize PM
 4. While the time deadline is not exceeded:
 - (a) If $step == 0$ or PM changed the profile C_P since last time:
 - i. $C_B \leftarrow$ new ranking for current profile C_P
 - ii. For each $c \in C_B$,
if c does not occur in C_R then add $\langle c, step + \frac{rank\ of\ c\ in\ C_B}{length(C_B)} \rangle$ to C_Q
 - (b) If C_Q is empty then break the loop
 - (c) Pop an item c with maximum rank from C_Q
 - (d) $r \leftarrow VS(c)$ (perform VS for c and get the result r)
 - (e) Add $\langle c, r \rangle$ to C_R
 - (f) Adjust the profile C_P within PM with $\langle c, r \rangle$
 - (g) $step \leftarrow step + 1$
 5. Return the configurations from C_R in the order of decreasing results
-

New ranking is generated each time the profile is changed. Machine configurations are added to the queue with a rank in which the step number also plays an important role—first the most recent ranking is considered, and only if the ranking is fully served, the next configuration is taken from the newest of the older rankings.

Validation Scenario. The validation scenario defines what needs to be done to estimate the quality of a given machine configuration. It contains a test scenario, specifying the details of the test to perform for a candidate machine configuration (for example a cross-validation test) and a *result query* extracting the quality of machine configuration from the test (for example mean accuracy or a collection of accuracy results).

Time Limit. Usually, new algorithms are analyzed with respect to computational complexity – the question is how much time and/or memory it takes to reach the goal. In case of meta-learning, we deal with an inverse problem – how good model can be found in a given time, as it is natural that all possible models can not be tested and estimated, because that would take many orders of magnitude larger time than we can supply. Because of that, the main loop of the algorithm is limited by time deadline and the analysis of its efficiency has the form of empirical examination of the accuracy of models returned within that time. Different approaches are given the same amount of time for their work and their gains are compared.

Profile Management. Two important aspects of profile management are: care about the contents of the profile and the method of using it to determine advantageous directions of further search for machine configurations. It is not obvious what results should be kept in the profile and even what size of the profile can be most successful. There is much space for in-depth analysis of such dependencies. Given a profile, ranking configurations can also be done in a number of ways. The most natural one is to collect the results obtained for the profile configurations on other data files, determine the most similar profile, or more precisely, the data for which the profile is the most similar, and generate a ranking corresponding to the most successful configurations for that data. The similarity can also be used to combine results for several similar datasets into the final scores. So, the behaviour of PM may be determined by:

- profile similarity measure,
- configuration selection strategy (deciding how many and which configurations are included in the ranking),
- knowledge base used for calculations (the collection of datasets D_1, \dots, D_k and results obtained for these datasets with the methods of interest, that is a set of functions $f_{D_i} : \mathcal{C} \rightarrow \mathcal{R}$).

From practical, object-oriented point of view, it may be advantageous to design the methods of profile adjustment and ranking generation together, to let them interact better in their tasks. Therefore, they are enclosed in a single PM object. It can also be very profitable to use the feedback sent to PM not only for profile adjustment, but also for learning how to generate rankings on the basis of the profiles (to exploit the information about how successful the previous rankings were). Investigations on these subjects are very interesting direction of further research, but significant advantages will be possible only when we have gathered a huge database of results obtained with large number of learning methods on large number of datasets in experiments performed in systematic, unified manner. Therefore, the experiments presented here, though not small, are just the beginning. Nevertheless, they show that meta-learning driven by result profiles really works.

5 The Experiment Settings

To analyze the RPDML framework in action, its configuration described in Subsect. 5 was tested on 77 datasets from the UCI repository [3], also available from OpenML [20], representing classification problems and 212 configurations of object-level learning machines. A list of names identifying the datasets is given in Table 1. The sets represent various domains and pose different requirements to the learning methods. The feature spaces consist of from 2 to 10000 features. The instance counts range from 42 to almost 5 millions. The numbers of classes of objects to recognize are between 2 and 23. Such collection of datasets seems to provide a sufficient field for reliable meta-learning experiments.

Table 1. Summary of 77 UCI datasets, used in the experiment.

balone3c	census-income-all	heart	ml-prove-h2	shuttle-all	vehicle
acute1	contraceptive-method-choice	hepatitis	ml-prove-h3	sonar	vertebral-column-3c
acute2	credit-screening	image	ml-prove-h4	soybean-large	vote
anneal-all	ctg-1	ionosphere-all	ml-prove-h5	soybean-small	wall-following-2
appendicitis	ctg-2	iris	movement-libras	spect-all	wall-following-24
arcene-trn-val	cylinder-bands	kddcup99-10	mushroom	splice	wall-following-4
australian	dermatology	kddcup99	nursery	steel-plates-faults	waveform2
balance-scale	ecoli	kr-vs-kp	page-blocks	tae	wdbc
breast-cancer-wisconsin	flag	lbreast	parkinsons	telugu-vowel	wine
breast-tissue	german	lung-cancer	pima-indians-diabetes	thyroid-all	wdbc
bridges-mat	glass	mammographic-masses	qsar-biodeg	tic-tac-toe	yeast
bridges-rel	hayes-roth	ml-prove-h0	sat-all	transfusion	zoo
car	heart-270	ml-prove-h1	seeds	user-knowledge-all	

Some of the UCI datasets are originally split into parts (usually training and test sample). Here, all such datasets were joined into a single set to be analyzed with cross-validation. In Table 1, suffix “-all” informs about such joins.

RPDML Configuration. Each parameter of the Algorithm 1 (set \mathcal{C} of candidate machine configurations, validation scenario, profile manager and time deadline) may significantly influence the gains of the method in particular application. Their values used for the experiment are specified below.

Object-Level Learning Machine Configurations. 212 learning methods of four types were applied: 102 decision tree (DT) induction algorithms, 40 k-nearest neighbor (kNN) methods, 5 naive Bayesian (NB) classifiers, 65 support vector machines (SVM). Decision tree algorithms were constructed with different split criteria, pruning methods and/or the way of validation. The 40 kNN methods were obtained with 10 different values of k (from 1 to 51) and 4 “metrics” used for distance calculations (Square Euclidean, Manhattan, Chebychev, Canberra). Bayesian classification used 5 settings of Naive Bayesian Classifier: one using no corrections, one applied with Laplace correction and 3 instances equipped with m-estimate corrections ($m \in \{1, 2, 5\}$). The 65 Support Vector Machines contained 5 configurations with linear kernel (with $C \in \{.5, 2, 8, 32, 128\}$) and 60 with Gaussian kernels: all combinations of $\sigma \in \{.001, .01, .1, .5, 1, 10\}$, square Euclidean or Canberra “metric” and $C \in \{.5, 2, 8, 32, 128\}$.

Application of all the classifiers but DTs was preceded by data standardization. For kNN and SVM methods, because of their computational complexity, each dataset containing more than 4000 vectors was filtered randomly to leave more or less 1000 vectors (each vector was kept with probability equal to 1000 divided by the original vectors count). The largest datasets (census-income-all and the two KDD cup datasets) were also filtered in a similar way for DTs, to keep around 10000 vectors.

More details on the methods configuration, experiment design and results can be found at <http://www.is.umk.pl/~kg/papers/EMCIS21-RPDML>.

Validation Scenario. Algorithm selection in the approach of RPDML is based on validation of learning machines, as shown in Sect. 4. In the experiment, this role

was entrusted to 10-fold cross-validation. Since we are interested in not only high mean accuracy, but also in its small variance (i.e. stable results), the criterion used here was the mean accuracy reduced by its standard deviation.

Time Limit. In more sophisticated applications, it may be very important to comply with the time limit, to check some faster methods first and then, engage more complex ones [8]. Here, we have just 212 machine configurations, so instead of the time limit it was chosen to allow for validation of 12 configurations, before the final decision had to be taken.

Profile Management. Proper profile handling is a crucial idea of the RPDML. The rationale behind it is that similar relations between selected results for two datasets may be precious pointers of meta-search directions. We expect that the winners of the most similar tasks will perform well also on the new dataset being analyzed. A thorough analysis of profile solutions should take into consideration such aspects as profile construction and maintenance, profiles similarity measures, ranking candidates on the basis of profile similarity.

The experiment, presented here, is based on a knowledge base consisting of results of 212 learning machines obtained for 77 datasets, so possibilities of profile maintenance analysis are limited. To draw reliable conclusions, one needs to collect thousands or even millions of results. Such analysis has not been done yet. Since the task was to check just 12 of 212 configurations (the “time” limit) and point the most promising one, the profiles were composed of all the results obtained by the validation procedures.

Profile similarity was measured in one of two ways: with (the square of) the Euclidean metric and with the Pearson’s correlation coefficient. Current rankings used to nominate the most promising configurations were calculated with respect to four indices:

- average accuracy reduced by its standard deviation,
- the same but expressed in the units of standard deviations in relation to the best of 212 results,
- average p-value of paired t-test comparing the method with the best one,
- average rank of the method.

All the averages were calculated over 5 or 10 most similar datasets from the knowledge base and were weighted with the similarity measures. These create 16 instances of RPDML configurations: all combinations of 2 similarity measures, 4 ranking indices and 2 sizes of the set of similar datasets.

State-of-the-Art Ranking Methods for Comparison. State-of-the-art methods for ranking algorithms usually prepare data descriptions to measure similarity between datasets and construct rankings on the basis of ranked methods accuracies obtained for the most similar data. They do not perform any validation of the methods on the data for which the ranking is being created, hence they can be called *passive*.

Since pairwise comparisons are possible only when the number of candidate machine configurations is extremely small, we can not base ranking on measures like ARR with RL, SRR or SW. Instead, the passive rankings, used for reference here, are based on less computationally complex formulae (3, 4, 5, 6).

$$AA(c) = \frac{1}{n} \sum_{i=1}^n A_c(i) \quad (3) \quad AR(c) = \frac{1}{n} \sum_{i=1}^n Rank(c, i) \quad (5)$$

$$AD(c) = \frac{1}{n} \sum_{i=1}^n \frac{A_c(i) - A_{B(i)}(i)}{\sigma_{B(i)}(i)} \quad (4) \quad AP(c) = \frac{1}{n} \sum_{i=1}^n p(c, i), \quad (6)$$

where c denotes the candidate configuration, n the number of datasets, $A_c(i)$ is the accuracy of the machine c tested on i 'th dataset, $\sigma_c(i)$ – the standard deviation of this accuracy estimated in the test, $B(i)$ stands for a configuration that has reached the highest accuracy for i 'th dataset, $Rank(c, i)$ is the rank of configuration c in the ordered results for i 'th dataset and $p(c, i)$ is the p-value of the paired t-test comparing results obtained for c and $B(i)$. Each formula has its advantages and drawbacks. Some of them are pointed out further, in the discussion on the results.

To reliably compare active ranking methods and passive approaches, the latter are also validated. It means that the passive rankings are subject to step-by-step validation of top machine configurations to return the configurations in new order, corresponding to the validation results (the same validation scenario as in Algorithm 1).

Goals of the Experiment. The main goal of the test was to examine whether the profile approach of RPDML is capable of finding more attractive learning machines than other approaches devoid of profile analysis. Therefore, the first reference method for the comparison is a meta-search based on random ranking of methods. It is important to realize that it is not the same as random selection of a model. Here, a number of randomly selected machines go through the validation process which prevents from selecting a method that fails.

Another goal of the experiment was to examine the advantages of the profiles themselves. To achieve this, a comparison of the profile-based methods to static rankings based on the same criteria has been conducted.

Since the goal here is to check advantages of profiles, they are compared to similar algorithms with no profile based decisions. We do not compare to other complete solutions like Auto-WEKA or AUTO-SKLEARN, because that would not provide desired information on profiles and would require significant modifications of the systems to compensate the differences in goals and architectures.

Testing algorithm selection should not be performed with the use of a knowledge base containing information about the test data. That would be a sort of cheating. Thus, a leave-one-out style tests were executed (each RPDML instance was trained on the results obtained for 76 datasets and tested on the 77th).

Table 2. Summary of RPDML results for 77 UCI datasets.

Rank	Method	Mean - s.d.	Loss	p value	Wins	Mean
28.21	Corr-5NN-PV	0.82791	0.522	0.1474	28	0.8615
28.23	Corr-10NN-PV	0.82268	0.520	0.1582	28	0.8621
28.32	SqEuclid-10NN-PV	0.82590	0.410	0.1847	37	0.8638
28.65	Corr-5NN-Ranks	0.82080	0.569	0.1927	31	0.8603
28.88	SqEuclid-5NN-PV	0.82570	0.448	0.1957	35	0.8629
30.15	SqEuclid-5NN-Ranks	0.81911	0.657	0.1596	27	0.8581
30.24	Corr-5NN-AccStdDiff	0.82030	0.564	0.1904	28	0.8594
31.98	Corr-10NN-AccDiff	0.82017	0.641	0.1356	26	0.8588
32.12	Corr-5NN-AccDiff	0.81972	0.599	0.1577	28	0.8594
33.33	SqEuclid-5NN-AccDiff	0.81729	0.672	0.1394	30	0.8566
33.43	Static PV	0.82181	0.551	0.1873	36	0.8600
33.91	SqEuclid-5NN-AccStdDiff	0.81711	0.674	0.1621	28	0.8559
34.08	Corr-10NN-AccStdDiff	0.81858	0.641	0.1406	23	0.8578
34.19	Corr-10NN-Ranks	0.81884	0.637	0.1844	31	0.8583
34.42	SqEuclid-10NN-Ranks	0.81701	0.655	0.1571	34	0.8572
36.95	SqEuclid-10NN-AccStdDiff	0.81498	0.732	0.1618	30	0.8545
37.25	Static Ranks	0.81394	0.765	0.1159	22	0.8524
39.1	SqEuclid-10NN-AccDiff	0.81433	0.751	0.1544	25	0.8545
48.4	Random	0.80997	1.050	0.0526	15	0.8514
53.05	Static AccStdDiff	0.80243	1.098	0.0752	20	0.8420
55.16	Static AccDiff	0.80234	1.096	0.0813	19	0.8423

* s.d. stands for standard deviation

Table 3. Top ranked results of object-level classifiers for 77 UCI datasets.

Rank	Method	Mean - s.d.	Loss	p value	Wins	Mean	S.d.
46.77	M 1	0.80480	1.109	0.1198	19	0.8427	0.03793
52.64	M 2	0.80062	1.155	0.1458	22	0.8390	0.03837
53.4	M 3	0.80176	1.164	0.0988	15	0.8418	0.04005
53.48	M 4	0.80003	1.281	0.1078	13	0.8384	0.03835
53.52	M 5	0.79971	1.198	0.1517	19	0.8410	0.04131
54.25	M 6	0.80123	1.242	0.0883	14	0.8404	0.03913
55.84	M 7	0.80240	1.514	0.0668	14	0.8407	0.03828
56.36	M 8	0.79706	1.434	0.1021	12	0.8366	0.03953
56.66	M 9	0.79695	1.364	0.0962	12	0.8375	0.04051
58.68	M 10	0.79522	1.364	0.1242	14	0.8372	0.04198
59.13	M 11	0.79915	1.379	0.0904	9	0.8389	0.03971
59.28	M 12	0.80040	1.574	0.1019	15	0.8417	0.04132
59.34	M 13	0.79606	1.353	0.1209	19	0.8382	0.04212
59.48	M 14	0.79748	1.301	0.1228	15	0.8359	0.03839
59.53	M 15	0.79869	1.337	0.0985	12	0.8405	0.04180
59.58	M 16	0.79465	1.513	0.0963	14	0.8356	0.04100
...
72.15	M 44	0.79308	1.590	0.0423	11	0.8364	0.04330
72.47	M 45	0.79075	1.494	0.1615	22	0.8343	0.04357
72.6	M 46	0.78725	3.019	0.0662	8	0.8292	0.04191
73.71	M 47	0.78916	2.352	0.0886	15	0.8324	0.04323
74.27	M 48	0.78517	3.023	0.0504	9	0.8273	0.04216
74.88	M 49	0.78566	3.035	0.0518	8	0.8258	0.04016
75.08	M 50	0.78598	2.855	0.0601	8	0.8295	0.04356
...

6 Results and Analysis

Some of the most reasonable indicators of algorithm quality that can serve as comparison criteria, and have been calculated in the experiment, are:

- average test accuracy or the accuracy reduced by its standard deviation,
- average loss with respect to the best available result, represented in the unit of (the winner’s) standard deviation,
- average rank obtained among other algorithms,
- average p-value of paired t-test comparisons with the winner methods,
- the count of wins defined as obtaining results not statistically significantly worse than those of the winners (examined with paired t-test with $\alpha = 0.05$).

There is no single best criterion and each of those listed above has some drawbacks. Average test accuracy (also reduced by standard deviation) may favor datasets with results of relatively larger variance. Calculating accuracy differences (loss) with respect to the winner and rephrasing them in the units of winner’s standard deviations is not adequate in case of zero variance (happens e.g. when 100% accuracy is feasible). Ranks may be misleading when families of similar methods are contained in the knowledge base – a small difference in the accuracy may make big difference in rank. Average p-values are not perfect either – it is possible that a better model gets lower p-value when compared to the winner then a worse model (e.g. when large variance of the latter disturbs statistical significance). Naturally, the count of wins suffers from the same drawbacks as p-values.

To facilitate multicriteria comparison, all these scores are presented in Table 2. For reference, also the top-ranked results of the 212 object-level algorithms are shown in Table 3. Full tables of results (overall and particular for each dataset) are available online at <http://www.is.umk.pl/~kg/papers/EMCIS21-RPDML>.

First Glance Analysis. Even a short glance at the result tables makes clear impression that most of the values in Table 2 are more attractive the ones in Table 3. The mean accuracies of the best RPDML scores are higher by more than 2% and average ranks get reduced by almost half in relation to the best object-level algorithms.

It may seem surprising that the best average ranks are so large (46.77 for the object-level winner and 28.21 for RPDML). Their nominal values may be a bit misleading because of the strategy of assigning the ranks following the methods of rank-based statistical tests, where in case of a draw, all the methods with the same result obtain the average rank corresponding to their positions. As a result, in the most extreme case, for the “accute1” dataset, the best classifiers performing with 100% accuracy get the rank 59, because 117 of the 212 classifiers classify the data perfectly. Other examples are “accute2” with the top rank of 49.5, “steel-plates-faults” with 35 or two “wall-following” datasets, where 49 perfect classifiers share the rank of 25.

Average accuracies, average loss and mean rank show similar properties, so they are quite correlated. More different rankings are obtained with the criteria of p-values and numbers of wins (naturally, the two are also correlated). It can be observed in the tables by means of the bold-printed numbers which show the best scores according to the criteria.

Detailed analysis of the results confirms that none of the comparison criteria is perfect. Even the statistical methods like p-values and counts of wins (or more precisely the count of datasets for which the results can not be confirmed with a statistical test to be significantly less accurate than the best results) sometimes show peculiar effects. For example, the results for the “vertebral-columns-3c” data show that a method ranked 126th, according to the mean reduced by standard deviation, is not significantly worse than the winner while 70 methods ranked higher, including the one ranked at position 20, lose significantly to the winner. The winner’s accuracy is 0.8323 ± 0.05832 , while the algorithms ranked at positions 126 and 20 reach 0.8032 ± 0.08185 and 0.8086 ± 0.05148 respectively. The one ranked 20th seems better, because it is more stable (less variance), but paradoxically, the stability makes it lose to the winner, while larger variance of the 126th one grants it with a win. Another example is “wall-following-24”, where the meta-learning based on random ranking gets accuracy of 0.9937 ± 0.01059 (hypothesis about losing to the winner is not rejected with paired t-test with $\alpha = 0.05$) while the others provide slightly higher accuracy and 4 times smaller deviation (e.g. 0.9964 ± 0.00248) and are confirmed with the t-test to lose.

Similar facts cause that the largest average p-value (among the object-level learning methods) was obtained by a method building full greedy DTs, which means maximum possible accuracy for the training data and large variance for the test data. As a result of the larger variance, the number of wins is the highest, but at the same time its mean accuracy is about 1% lower on average than that of other methods with lower win counts.

Meta-learning vs Single Object-Level Classifiers. Comparison of meta-learning approaches with single learners should be done with special care. The meta-learning algorithms are allowed to perform multiple validation tests and select one of many (here 12) models. Thus, it is natural that they can be more accurate. The difference can be clearly seen with the naked eye from the results. For example:

- average ranks for the RPDML methods range from 28.21 to 39.1 (most less than 34) while the top-ranked object-level method reaches the value of 46.77,
- average mean accuracy minus standard deviation ranges from 0.81433 to 0.82590 for RPDML while the best object-level result is 0.8048.

An interesting observation is that for some datasets, the accuracies of some RPDML machines are higher than the best object-level classifier’s. Although RPDML just selects a machine from the 212 object-level algorithms, it sometimes happens, that in some folds of cross-validation, it recognizes that a machine, less accurate in general, can be more successful in application to this particular data sample. Obviously, it can not be treated as a regular feature, as it happened 8 times for 4 datasets (ctg-2, ml-prove-h1, nursery and sat-all), but it is interesting that such improvement is at all possible. In such cases, the lower p-value of the t-test, the better, so the index has been modified to $2 - p\text{-value}$, to be a greater reward instead of a penalty.

Profile-Based vs Static Ranking Methods. The most important analysis of this experiment concerns the advantages of the function of profiles. To facilitate the discussion, the methods based on static rankings have been introduced and tested. Table 2 presents the results of the static methods among those of RPDML and the random ranking. According to all the criteria, two of the static rankings (based on accuracy) perform slightly worse or slightly better than random ranking. The other two (based on ranks and p-values) do better, but they are outperformed by most of the RPDML configurations in most comparisons. Only the static ranking based on p-values is located higher than some of its RPDML counterparts in comparison with respect to p-values and wins, but keeping in mind the disadvantages of these criteria mentioned above, we can reliably claim that the profiles are precious tools in meta-learning.

7 Conclusions and Future Perspectives

RPDML is an open framework, that facilitates easy implementation of many meta-learning algorithms integrating meta-knowledge extraction and its exploitation with object-level learning. Different kinds of problems may be solved with appropriate implementation of the framework modules like validation scenario and profile manager. The implementations presented here have proven their value in the experiments. The framework facilitates active management of learning results profiles, leading to more adequately adapted meta-learning algorithms.

Since the framework is open, a lot of research can be done with it, hopefully leading to many successful incarnations of RPDML. First of all, there is a lot of space for development of intelligent methods for profile management. The way profile is maintained with the feedback coming from experiments, can be crucial for final efficiency of profile-based meta-learning machines. Some experiments with simple methods of profile size reduction have shown that it is very easy to spoil the results by inaccurate profile maintenance. For example, the idea of handling the profile by keeping the results with as high accuracy as possible can lead to profile degeneration. Experiments, not described here in detail, have shown that such technique may at some point generate poor ranking, which would be kept and followed to the end of learning time, because once machines of poor accuracy appear at the top, subsequent validations end up with weak results, not eligible for the profile, so the profile does not change and the poor ranking is followed on and on. The lack of changes in the profile implies no changes in ranking and inversely. A sort of dead lock appears. Hence, profile management must be equipped with techniques aimed at avoiding such situations. The profiles should be diverse, and continuously controlled, whether they result in rankings providing accurate models at the top. Otherwise, it can be more successful to return to some old but more suitable profile, instead of losing time for validation of many machines from a degenerate ranking.

Profile management includes adaptive methods of dataset similarity measurement for more suitable ranking generation. In parallel to profile analysis, knowledge base properties may also be examined in order to prepare knowledge bases most eligible for meta-learning. The knowledge bases may be equipped with additional information providing specialized ontologies useful not only in meta-learning but also in testing object-level learners.

References

1. Balte, A., Pise, N., Kulkarni, P.: Meta-learning with landmarking: a survey. *Int. J. Comput. Appl.* **105**(8), 47–51 (2014)
2. Bensusan, H., Giraud-Carrier, C., Kennedy, C.J.: A higher-order approach to meta-learning. In: Cussens, J., Frisch, A. (eds.) *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, pp. 33–42 (2000)
3. Dua, D., Graff, C.: UCI machine learning repository (2017). <http://archive.ics.uci.edu/ml>
4. Duch, W., Grudziński, K.: Meta-learning via search combined with parameter optimization. In: Rutkowski, L., Kacprzyk, J. (eds.) *Advances in Soft Computing*, vol. 17, pp. 13–22. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-7908-1777-5_2
5. Engels, R., Theusinger, C.: Using a data metric for preprocessing advice for data mining applications. In: *Proceedings of the European Conference on Artificial Intelligence (ECAI-98)*, pp. 430–434. Wiley (1998)
6. Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F.: Efficient and robust automated machine learning. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds.) *Advances in Neural Information Processing Systems*, vol. 28, pp. 2962–2970. Curran Associates, Inc. (2015)

7. Fürnkranz, J., Petrak, J.: An evaluation of landmarking variants. In: Giraud-Carrier, C., Lavra, N., Moyle, S., Kavsek, B. (eds.) Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (2001)
8. Grąbczewski, K., Jankowski, N.: Saving time and memory in computational intelligence system with machine unification and task spooling. *Knowl.-Based Syst.* **24**, 570–588 (2011)
9. Guyon, I., Saffari, A., Dror, G., Cawley, G.: Model selection: beyond the Bayesian/frequentist divide. *J. Mach. Learn. Res.* **11**, 61–87 (2010)
10. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 507–523. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-25566-3_40
11. Jankowski, N., Grąbczewski, K.: Universal meta-learning architecture and algorithms. In: Jankowski, N., Duch, W., Grąbczewski, K. (eds.) Meta-Learning in Computational Intelligence. Studies in Computational Intelligence, vol. 358, pp. 1–76. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20980-2_1
12. Malkomes, G., Schaff, C., Garnett, R.: Bayesian optimization for automated model selection. In: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, pp. 2900–2908. Curran Associates Inc., Red Hook (2016)
13. Olson, R.S., Bartley, N., Urbanowicz, R.J., Moore, J.H.: Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016, GECCO '16, pp. 485–492. Association for Computing Machinery, New York (2016)
14. Peng, Y., Flach, P.A., Soares, C., Brazdil, P.: Improved dataset characterisation for meta-learning. In: Lange, S., Satoh, K., Smith, C.H. (eds.) DS 2002. LNCS, vol. 2534, pp. 141–152. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36182-0_14
15. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: Proceedings of the Seventeenth International Conference on Machine Learning, pp. 743–750. Morgan Kaufmann (2000)
16. Rice, J.R.: The algorithm selection problem - abstract models. Technical report, Computer Science Department, Purdue University, West Lafayette, Indiana (1974). cSD-TR 116
17. Smith-Miles, K.A.: Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Comput. Surv.* **41**(1), 6:1–6:25 (2009)
18. Soares, C., Petrak, J., Brazdil, P.: Sampling-based relative landmarks: systematically test-driving algorithms before choosing. In: Brazdil, P., Jorge, A. (eds.) EPIA 2001. LNCS (LNAI), vol. 2258, pp. 88–95. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45329-6_12
19. Thornton, C., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Auto-WEKA: combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13, pp. 847–855. Association for Computing Machinery, New York (2013)
20. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: networked science in machine learning. *SIGKDD Explor.* **15**(2), 49–60 (2013)