Prediction of Classifier Training Time including Parameter Optimization

Matthias Reif, Faisal Shafait, and Andreas Dengel

German Research Center for Artificial Intelligence,
Trippstadter Str. 122, 67663 Kaiserslautern, Germany
{matthias.reif,faisal.shafait,andreas.dengel}@dfki.de

Abstract. Besides the classification performance, the training time is a second important factor that affects the suitability of a classification algorithm regarding an unknown dataset. An algorithm with a slightly lower accuracy is maybe preferred if its training time is significantly lower. Additionally, an estimation of the required training time of a pattern recognition task is very useful if the result has to be available in a certain amount of time.
Meta-learning is often used to predict the suitability or performance of classifiers using different learning schemes and features. Especially landmarking features have been used very successfully in the past. The accuracy of simple learners are used to predict the performance of a more sophisticated algorithm.
In this work, we investigate the quantitative prediction of the training time for several target classifiers. Different sets of meta-features are evaluated according to their suitability of predicting actual run-times of a parameter optimization by a grid search. Additionally, we adapted the concept of landmarking to time prediction. Instead of their accuracy, the run-time of simple learners are used as feature values.
We evaluated the approach on real world datasets from the UCI machine learning repository and StatLib. The run-time of five different classification algorithms are predicted and evaluated using two different performance measures. The promising results show that the approach is able to reasonably predict the training time including a parameter optimization. Furthermore, different sets of meta-features seem to be necessary for different target algorithms in order to achieve the highest prediction performances.

1 Introduction

Selecting the best classifier for a given problem instance is an important part of developing a pattern recognition system. The choice can be made by different points of views. Typically, the achieved classification performance is the most important aspect. However, the No-Free-Lunch theorem [19] tells us that there is no uniformly best algorithm.

Furthermore, the computation time is often a significant factor when choosing an algorithm, too. If the expected performance values of several algorithms are the same, the algorithm with a lower run-time is usually preferred. Also slight decreases in performance may be acceptable if the reduction in run-time is significant. Sometimes, only a limited amount of time is available for computing the results. Furthermore, if the user has to pay for the computation time, he might not want to start a possibly time-consuming process without any idea about its duration. All these considerations make the estimation about the expected time needed a valuable information, especially for the increasing amount of large datasets.

Usually, an algorithm is described by a general statement about its complexity. For example, Multilayer Perceptrons (MLP) are known to have a rather high training time compared to a k-Nearest Neighbor approach. Nevertheless, the actual run-time often depends on the dataset and the exact parameter values of the algorithm. Furthermore, categorical time estimations like "high" or "low" do not give the user the same amount of information like actual time values using real units. For example, a "high" run-time could mean "several hours" to "several weeks" or even longer. Such nominal values are only limitedly useful for comparisons of multiple classifiers. In contrast to this, the prediction of real numbers can be much more precise. Additionally, values of actual time units make the estimation much more useful for the user. The theoretical computational complexity is also known for many algorithms. Since constant terms are neglected in computational complexity theory, the practical usefulness of such indications is limited, too.

However, a challenge in predicting run-times is the dependency of the hardware the algorithm runs on. A faster computer will always decrease the run-time for the same dataset and the same algorithm. This makes the estimation of real time values harder if the prediction model was trained on time data that was gathered on a different machine. The goal is to predict the time by taking the users machine into account as well.

In this paper, a method for predicting actual run-times of classification algorithms is presented. Since most algorithms contain parameters that influence their performance, they are typically optimized. Therefore, we do not predict the time of one training or one application of a classifier but the time needed for a grid search of the most important parameters. This includes multiple training and application phases.

The presented approach uses concepts of meta-learning including features of datasets. A regression model is learned that is able to predict the run-time of a particular algorithm regarding an unknown dataset. We investigated traditional features of datasets as well as features that are more specialized for predicting time values. Therefore, we adapted the successfully used concept of landmarking to the time domain. This new type of features enables the presented approach to take the performance of the users machine into account. The learned model becomes more independent of the actual computing power and is able to predict the run-time more precisely.

The rest of the paper is structured as follows. In the next section, we describe the approach of meta-learning including related work. The challenges of predicting the run-time of a grid search are presented in Section 3. Section 4 contains the detailed description of the presented approach. The evaluation and its results are given in Section 5. The last section comprises of the conclusion.

2 Meta-Learning

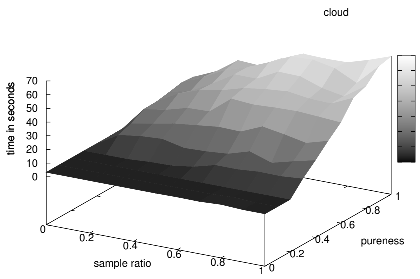
Meta-learning uses knowledge about already solved problem instances to gain information about unknown problems regarding one or multiple learning schemes. Typically, meta-learning is based on features of datasets. These features are often called meta-features. They describe properties of a dataset by using different approaches. Simple meta-features use directly accessible properties like the number of samples, the number of attributes or the number of classes. More sophisticated features are statistical measures [8, 17], which are based on statistical analysis of the data. Typical measures of this group are the kurtosis and the skewness. The group of information theoretic features basically use the entropy values of the attributes and the class label [16].

Two more recently proposed groups are landmarking and model-based meta-features. Both approaches utilize other classification algorithms. Landmarking [15, 4, 2, 9] applies simple and fast computable algorithms on the dataset and uses the achieved classification performance as feature value. The model-based features [14, 3] also create a classification model of the data, but use several properties of this model instead of its performance. Model-based features are e.g. the width or height of an unpruned decision tree that was build on the dataset.

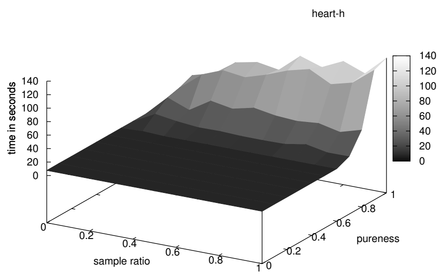
A typical application of meta-learning is the prediction of the best classifier for an unknown dataset. The used knowledge consists of the meta-features of known datasets and the information about what algorithm worked best for each of the datasets. Based on such a meta-dataset, a learning scheme creates a classification model that can be used for predicting the best classifier of an unknown dataset.

A slightly different approach creates a ranking of all considered target algorithms. The correct ranking is known for several datasets and the predicted ranking of the new dataset is typically gained by a nearest-neighbor approach. The distance measure is based on the meta-features. Brazdil et al. [6] presented a method for ranking algorithms that also includes computation times. A multi-criteria measure involves a ratio of accuracies and a ratio of times between two considered algorithms. The run-time is not directly predicted but only influences the ranking score. The strength of the influence is a user defined parameter.

Regression was also used for meta-learning [10, 11, 5]. Instead of predicting the best algorithm using a classification approach, this method predicts quantitative performance values. Therefore, for each target classifier whose performance should be predicted, a separate regression model has to be trained. Again, the features are the meta-features of the datasets, but the label is the actual performance value of one single target classifier. Various meta-features and regression algorithms have been used to predict different performance measures of classification algorithms, but none of these methods predicted their run-time.



(a) Different parameter combinations require different run-times.



(b) Additionally, the distribution of the run-time differs for other datasets.

Fig. 1. The parameters *sample ratio* and *pureness* of the Ripper classifier against the required run-time for the two datasets *cloud* and *heart-h*.

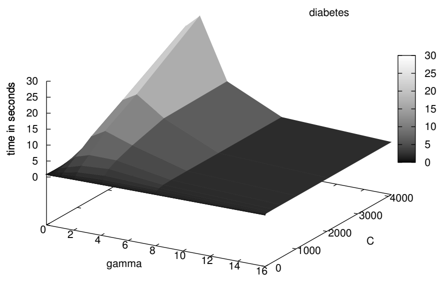
Lindner and Studer [12] used a case-based reasoning approach for algorithm recommendation by meta-learning that also considers time aspects. Training and testing time are treated separately. Each algorithm is generally assigned to one of five categories from "very fast" to "very slow" in order to include requirements of the user into the recommendation. Different run-times of the algorithms for different datasets were not investigated.

In this paper, a regression approach for predicting the time of a grid search for diverse classifiers is presented. The expected run-time is predicted in a quantitative way instead of predicting categories or comparisons between algorithms only. The user gets an actual estimation of the time required for optimizing one particular algorithm on his data. Additionally, the prediction can be used to choose the most suitable algorithm for a problem respecting run-time conditions as well.

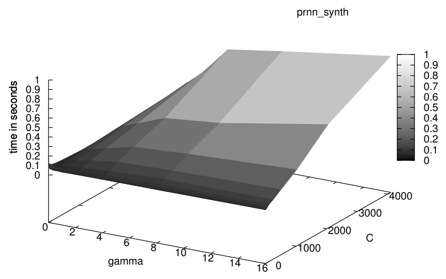
3 Run-Time of a Grid Search

Since the performance of most classifiers depends on parameter values, the parameters are usually optimized. A simple and often used method for parameter optimization is a grid search. All predefined combinations of parameter values are evaluated to determine the best of them. The advantage of a grid search is that it usually delivers very good results. However, the drawback of this brute force approach is the rather high run-time compared to other optimization strategies.

For the time prediction of a grid search, one may think of predicting the run-time for one single evaluation of the classifier using one defined parameter combination and afterwards multiply the result with the size of the grid. However, as visible for the *cloud* dataset in Figure 1(a), different parameter combinations require different amounts of time. The plot shows the run-time of training the Ripper classifier for different combinations of its two parameters *sample ratio* and *pureness*. If, for example, the time of the lowest values of the parameters have been used, the total run-time will be significantly underesti-



(a) A high value of γ lead to a shorter run-time for the *diabetes* dataset.



(b) In contrast to the *diabetes* dataset, a higher value of γ results in a longer run-time for the *prnn_ synth* dataset.

Fig. 2. Run-time of training a Support Vector Machine (SVM) with different combinations of its parameters γ and C on the *diabetes* dataset and the *prnn_ synth* dataset.

mated. Also other parameters of widely used classifiers obviously influence their run-time, e.g. the maximal depth of a decision tree or the learning rate of a Multilayer Perceptron (MLP).

A solution might be using the estimation for a defined parameter combination and a constant factor in addition to the grid size in order to compensate these dependencies. However, the distributions of the run-times within the parameter space differs for different datasets as well. For example, the increase of run-time for the *heart-h* dataset is significantly different than for the *cloud* dataset, as visible in Figure 1(b).

Furthermore, the measured run-times can differ even more between multiple datasets. Figure 2 shows the run-time for the typically optimized parameters γ and C of a Support Vector Machine (SVM). As visible in Figure 2(a), a higher value of the kernel parameter γ leads to a shorter run-time for the *diabetes* dataset, whereas for the *prnn_ synth* dataset in Figure 2(b), a higher value of γ results in a longer run-time. Obviously, the distribution of the run-time depends on the dataset itself. Therefore, meta-features, which describe the data, are additionally necessary for a precise prediction of the run-time.

4 Methodology

For each target classifier, whose run-time should be predicted, a separate regression model is trained. The training data for the learning scheme consists of the knowledge about known datasets. Each instance of the training set describes one dataset. It contains the meta-features of the dataset and the measured run-time of the considered target classifier. The run-time is used as the target variable. After the learning phase, the resulting model can be used to predict the run-time of an unknown dataset. The model is applied on the meta-features of this dataset. The overall approach is illustrated in Figure 3.

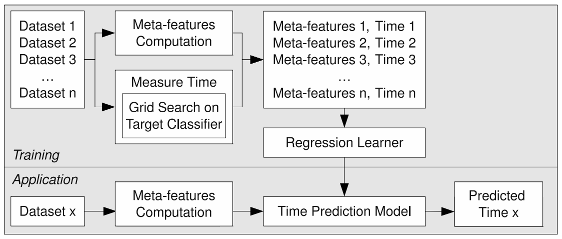


Fig. 3. Training of a time prediction model for one target classifier using a regression learner (top) and application of the model to an unknown dataset (bottom). The target value is the run-time of a parameter optimization.

4.1 Traditional Meta-Features

As a first set of meta-features, we used typical measures from the previously mentioned groups. This set includes the following 34 meta-features:

Simple meta-features: number of samples, number of classes, number of attributes, number of nominal attributes, number of numerical attributes, dimensionality (number of attributes divided by number of samples)

Statistical meta-features: kurtosis, skewness, canonical discriminant correlation, first normalized eigenvalues of canonical discriminant matrix, absolute correlation

Information theoretic meta-features: normalized class entropy, normalized attribute entropy, joint entropy, mutual information, noise-signal-ratio, equivalent number of attributes

Model-based meta-features: For these features, a decision tree is trained without pruning. Different properties of this tree are used as feature values: number of leaves, number of nodes, nodes per attribute, nodes per sample, leaf corrobation.
Additionally the minimum, maximum, mean value and standard deviation of the following measures are used: length of a branch, number of nodes in a level, number of occurrences of attributes in a split

The previously successfully used concept of landmarking uses only the performance of several classification algorithms. Since these values are obviously unrelated to the run-time of an algorithm, we used landmarking in an adapted way that is described in the next section.

4.2 Time-Based Meta-Features

Landmarking have been successfully used in the past for different meta-learning approaches [15, 4, 2, 9]. The approach uses performance values of simple classifiers for predicting the performance of more sophisticated algorithms. Analogically, we use the run-time of the same simple learners for predicting the run-time of a sophisticated classifier. The used classifiers are Naive Bayes, One-Nearest Neighbor, and Decision Stumps.

Additionally, we also included several times required for computing the other groups of meta-feature. The measured computation times of the following steps are used:

– calculating the statistical meta-features
– calculating the information theoretic meta-features
– calculating the model-based features including the creation of the decision tree

If the traditional meta-features are calculated anyway, e.g. for predicting the accuracy of classifiers, these measures do not require any additional computational effort.

In a practical scenario, these time-specific meta-features may be able to compensate the difference in performance of the computer the regression model was created on and the computer the regression model is used on. The features take the actual machine where the prediction is performed on into account because they are computed on the users machine. Therefore, more realistic and more useful predictions are possible. Since the approach estimates actual run-times, it depends on the actual hardware and features describing the environmental aspects should be used. To summarize, a time prediction approach should include meta-features that are able to describe two aspects: the dataset and the performance of the users computer. Using traditional meta-features and time-based measures, both requirements are fulfilled.

5 Evaluation

We evaluated the presented approach on real world datasets from the UCI machine learning repository [1] and StatLib [18]. The run-time of a grid search for five different classifiers are investigated. The used classifiers as well as their optimized parameters are listed in Table 1.

The complete evaluation was done using RapidMiner [13]. It is an open source data mining and pattern recognition framework implemented in Java. All times have been measured on an AMD Opteron using a single-threaded program.

The number of datasets is different for the classifiers since we only considered datasets with a run-time of the grid search in a defined interval. We excluded datasets with a run-time smaller than 10 seconds since the time measurements in this small time scales are more error-prone. Especially the landmarking times are too small for reliable results, but very low run-times are not important in a practical sense either. Additionally, datasets with a run-time of the algorithm greater than 24 hours have been neglected as well because of the computational effort. The exact number of datasets used for the single algorithms are shown in Table 2.

The presented approach was evaluated by a leave-one-out cross-validation for every algorithm. We used the regression variant of a Support Vector Machine,

Classifier	Parameter	Interval	Steps	Scale
k-NN	k	$[1, 1000]$	100	logarithmic
	weighted vote	{yes, no}		
SVM	γ	$[2^{-10}, 2^4]$	15	quadratic
	C	$[2^{-2}, 2^{12}]$	15	quadratic
MLP	learning rate	$[0.01, 1.0]$	10	logarithmic
	momentum	$[0.0, 1.0]$	10	linear
	decay	{yes, no}		
Ripper	sample ratio	$[0.1, 1.0]$	10	linear
	prune benefit	$[0.1, 1.0]$	10	linear
	pureness	$[0.1, 0.9]$	9	linear
	criterion	{inf. gain, acc.}		
Decision Tree	min split size	$[2, 100]$	10	logarithmic
	min leaf size	$[1, 100]$	10	logarithmic
	min gain	$[0.05, 5]$	10	logarithmic
	max depth	$[5, 100]$	10	logarithmic
	confidence	$[0.05, 5]$	10	linear

Table 1. The investigated classifiers and their parameters that have been optimized by a grid search.

Classifier	k-NN	SVM	MLP	Ripper	Decision Tree
Samples	68	123	123	58	118

Table 2. The number of samples (datasets) used within the evaluation for the different target classifiers.

the ϵ-SVR, as meta-learning scheme. The parameters γ and C of the ϵ-SVR have been optimized by a grid search. LibSVM [7] was used as implementation.

We investigated different subsets of the described meta-features. The basic subsets are as follows:

simple: simple meta-features only
normal: all traditional meta-features
time: time measures of the traditional meta-features
***t*LM:** time-landmarking

All meta-features were calculated using R and were normalized to the interval $[0, 1]$. The sets of features and several combinations of them were evaluated using two common performance measures of regression models: the correlation coefficient and the normalized absolute error.

Classifier	simple	normal	time	tLM	simple + time	simple + tLM	time + tLM	simple + time + tLM
k-NN	0.817	0.702	0.803	0.867	0.883	0.905	0.875	**0.933**
SVM	0.766	0.653	0.569	**0.876**	0.764	0.863	0.873	0.863
MLP	0.810	0.814	0.647	0.733	0.799	0.825	0.768	**0.816**
Ripper	0.737	**0.814**	0.544	0.700	0.731	0.718	0.756	0.713
Decision Tree	0.850	0.844	0.575	0.821	0.833	**0.878**	0.864	0.874

Table 3. The Pearson product moment correlation coefficients of different sets of meta-features: feature sets that include the proposed time-landmarking features (tLM) achieve higher correlation values for four out of five classifiers.

5.1 Correlation

The Pearson product moment correlation coefficient (PMCC) of the actual run-time and the predicted run-time was calculated. The correlation between two variables X and Y is defined as

$$\rho_{X,Y} = \frac{E\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y}. \tag{1}$$

The results are values in the interval $[-1, 1]$. A value of one indicates a perfect positive relationship whereas minus one means the inverse, perfect negative relationship. If the correlation is zero, the two input variables are independent.

Table 3 shows the correlation coefficients for all five target classifiers and the investigated sets of meta-features. It is visible that good correlation values could be achieved. The correlation is at least 0.8 for all classifiers. Besides the simple meta-features, the time-landmarking approach seems to be particularly suitable for the task. The traditional meta-features achieved clearly better results only for the Ripper classifier.

5.2 Normalized Absolute Error

In addition to the correlation measure, the normalized absolute error was determined that serves as a comparison to a baseline. The absolute error of the prediction by the presented approach is divided by the absolute error of the prediction by a baseline method:

$$e = \frac{|t_m - t_p|}{|t_m - t_b|} \tag{2}$$

where t_m is the actual measured time, t_p the predicted time of the presented approach, and t_b the time predicted by the baseline method. For the baseline method, the predicted run-time is simply the average run-time of the classifier. Hence, the baseline method predicts the same run-time for every dataset.

If the normalized absolute error is smaller than one, the approach is better than the baseline. A value greater than one would indicate that predicting

Classifier	simple	normal	time	tLM	simple + time	simple + tLM	time + tLM	simple + time + tLM
k-NN	0.619	1.107	0.673	1.283	**0.596**	0.601	1.142	0.611
SVM	0.562	0.772	0.727	0.482	0.553	0.521	0.512	**0.467**
MLP	0.458	0.516	0.756	0.671	0.480	**0.440**	0.625	0.448
Ripper	0.562	0.564	0.796	0.584	0.579	0.581	**0.557**	0.584
Decision Tree	0.535	0.625	0.860	0.459	0.542	**0.434**	0.468	0.464

Table 4. The normalized absolute errors of different sets of meta-features. Like for the correlation, features sets including the time-landmarking features (tLM) achieve lower error rates for four target classifiers.

the average run-time is better than using meta-learning. Table 4 shows the normalized absolute errors. For this evaluation, the parameters of the ϵ-SVR were optimized according to this performance measure.

Error rates clearly below the baseline have been reached. Surprisingly, the error rates for the k-NN classifier are greater than one for certain meta-feature groups although the achieved correlation values are rather high.

5.3 Correlation between Features and Target Variable

Finally, the correlation between the different timing meta-features and the actual run-time was determined for each target classifier. Thereby, we wanted to evaluate the usefulness of the single time features and we wanted to determine how strong they are connected to the target time. In this part of the evaluation, no meta-learner is trained.

Figure 4 shows the correlation values between the six investigated time-based meta-features and the five considered target classifiers. It is visible that especially the run-times of the Naive Bayes and the One-Nearest Neighbor landmarkers are highly correlated to the actual run-time. Surprisingly, the time of the model-based features that include a creation of a decision tree is not related to the time of the decision tree target classifier. The reason for this is that the two methods are quite different. In contrast to the target classifier, the decision tree of the model-based features is unpruned and does not include any parameter optimization. Moreover, different implementations were used.

It is also noticeable that all six timing meta-features achieve high correlation values for the k-NN target classifier. One reason for this is probably the simplicity of the algorithm.

6 Conclusion

A method for predicting the run-time of a classification algorithm was presented. Meta-learning was used to estimate the time needed for a grid search over multiple parameters. Therefore, independent regression models have been learned for multiple target classifier. Since the run-time depends on the parameters of the

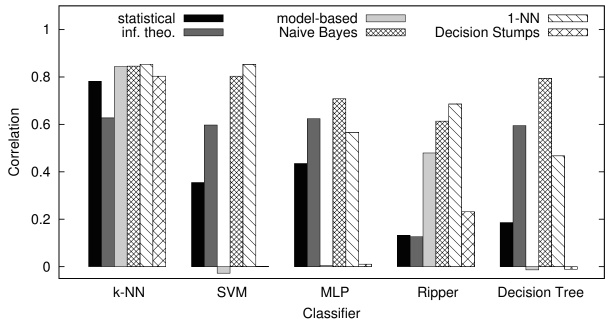


Fig. 4. The correlation between the time-based meta-features and the five target classifiers: Especially the Naive Bayes and the 1-Nearest Neighbor time-landmarkers achieve high correlation values.

classifier and the dataset itself, meta-features are used for creating the prediction model. In addition to traditional meta-features, measures that are specialized for time prediction were proposed. The run-time of simple learners as well as the time needed for calculating the traditional meta-features were used.

The presented approach was evaluated on real world datasets from the UCI machine learning repository and StatLib. Different subsets and combinations of the meta-features were evaluated according to their suitability for the task. Therefore, the correlation coefficient and the normalized absolute error were used.

The results show that the approach is able to reasonably predict the run-time of different algorithms. Especially the proposed time-landmarking measures improved the results. The investigated correlation values between time-based meta-features and the target run-time also show that the 1-Nearest Neighbor and the Naive Bayes time-landmarkers are suitable for this task.

References

1. Asuncion, A., Newman, D.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (2007), `http://www.ics.uci.edu/~mlearn/MLRepository.html`
2. Bensusan, H., Giraud-Carrier, C.: Casa batló is in passeig de gràcia or how landmark performances can describe tasks. In: Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination. pp. 29–46 (2000)
3. Bensusan, H., Giraud-Carrier, C., Kennedy, C.: A higher-order approach to meta-learning. In: Proceedings of the ECML'2000 workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination. pp. 109–117 (June 2000)
4. Bensusan, H., Giraud-Carrier, C.G.: Discovering task neighbourhoods through landmark learning performances. In: PKDD '00: Proceedings of the 4th European

Conference on Principles of Data Mining and Knowledge Discovery. pp. 325–330. Springer-Verlag, London, UK (2000)
5. Bensusan, H., Kalousis, A.: Estimating the predictive accuracy of a classifier. In: De Raedt, L., Flach, P. (eds.) Machine Learning: ECML 2001, Lecture Notes in Computer Science, vol. 2167, pp. 25–36. Springer Berlin / Heidelberg (2001)
6. Brazdil, P., Soares, C., da Costa, J.P.: Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results. Machine Learning 50(3), 251–277 (2003)
7. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines (2001), software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`
8. 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. John Wiley & Sons (1998)
9. Fürnkranz, J., Petrak, J.: An evaluation of landmarking variants. In: Giraud-Carrier, C., Lavrač, N., Moyle, S., Kavšek, B. (eds.) Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001). pp. 57–68. Freiburg, Germany (2001)
10. Gama, J., Brazdil, P.: Characterization of classification algorithms. In: Pinto-Ferreira, C., Mamede, N. (eds.) Progress in Artificial Intelligence, Lecture Notes in Computer Science, vol. 990, pp. 189–200. Springer Berlin / Heidelberg (1995)
11. Köpf, C., Taylor, C., Keller, J.: Meta-analysis: From data characterisation for meta-learning to meta-regression. In: Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support, Meta-Learning and ILP (2000)
12. Lindner, G., Studer, R.: Ast: Support for algorithm selection with a cbr approach. In: Recent Advances in Meta-Learning and Future Work. pp. 418–423 (1999)
13. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Ungar, L., Craven, M., Gunopulos, D., Eliassi-Rad, T. (eds.) KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 935–940. ACM, New York, NY, USA (August 2006)
14. Peng, Y., Flach, P., Soares, C., Brazdil, P.: Improved dataset characterisation for meta-learning. In: Lange, S., Satoh, K., Smith, C. (eds.) Discovery Science, Lecture Notes in Computer Science, vol. 2534, pp. 193–208. Springer Berlin / Heidelberg (2002)
15. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Meta-learning by landmarking various learning algorithms. In: In Proceedings of the Seventeenth International Conference on Machine Learning. pp. 743–750. Morgan Kaufmann (2000)
16. Segrera, S., Pinho, J., Moreno, M.: Information-theoretic measures for meta-learning. In: Corchado, E., Abraham, A., Pedrycz, W. (eds.) Hybrid Artificial Intelligence Systems, Lecture Notes in Computer Science, vol. 5271, pp. 458–465. Springer Berlin / Heidelberg (2008)
17. Sohn, S.Y.: Meta analysis of classification algorithms for pattern recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on 21(11), 1137 –1144 (11 1999)
18. Vlachos, P.: StatLib Datasets Archive. Department of Statistics, Carnegie Mellon University (1998), `http://lib.stat.cmu.edu`
19. Wolpert, D.H.: The lack of a priori distinctions between learning algorithms. Neural Comput. 8(7), 1341–1390 (1996)