Experiment Databases: Creating a New Platform for Meta-Learning Research

Keywords: meta-learning, databases, model selection, ontologies

Joaquin Vanschoren JOAQUIN.VANSCHOREN@CS.KULEUVEN.BE
Hendrik Blockeel HENDRIK.BLOCKEEL@CS.KULEUVEN.BE
Department of Computer Science, K.U.Leuven, Leuven, Belgium

Bernhard Pfahringer BERNHARD@CS.WAIKATO.AC.NZ
Geoff Holmes GEOFF@CS.WAIKATO.AC.NZ
Department of Computer Science, University of Waikato, Hamilton, New Zealand

Abstract

Many studies in machine learning try to investigate what makes an algorithm succeed or fail on certain datasets. However, the field is still evolving relatively quickly, and new algorithms, preprocessing methods, learning tasks and evaluation procedures continue to emerge in the literature. Thus, it is impossible for a single study to cover this expanding space of learning approaches. In this paper, we propose a community-based approach for the analysis of learning algorithms, driven by sharing meta-data from previous experiments in a uniform way. We illustrate how organizing this information in a central database can create a practical public platform for any kind of exploitation of meta-knowledge, allowing effective reuse of previous experimentation and targeted analysis of the collected results.

1. Introduction

1.1. Sharing Meta-Data

Research in machine learning is inherently empirical. Researchers, as well as practitioners, seek a deeper understanding of learning algorithm performance by performing large numbers of learning experiments. Whether the goal is to develop better learning algorithms or to select useful approaches to analyze new sources of data, collecting the right meta-data and correctly interpreting it is crucial for a thorough understanding of learning processes.

Despite an abundance of empirical studies (and meta-data), much remains to be learned about what makes an algorithm succeed or fail on certain datasets. Several comprehensive empirical studies, such as StatLog (Michie et al., 1994), MetaL (Brazdil et al., 2003) and, more recently, Ali and Smith (2006) and Caruana and Niculescu (2006) try to provide an overview of the state-of-the-art, but as new algorithms, preprocessing methods, learning tasks, and evaluation metrics are introduced at a constant rate, it is impossible for a single study to cover this continuously expanding space of learning approaches. Moreover, the meta-data generated by these and thousands of other machine learning studies is usually collected and stored differently and therefore hard to share and reuse. Collecting this information in public repositories would create a resource that could be tapped at any time to retrieve up-to-date results from a wide range of prior studies.

Furthermore, especially from a practitioner's perspective, data mining is not a one-shot operation. Generally, many different preprocessing and modeling techniques have to be applied in order to gain a deeper understanding of the data at hand, and sharing meta-data about previous studies could greatly help practitioners to build on previous experience and check which methods might be particularly useful.

1.2. A Community-Based Approach

In this paper, we propose a community-based approach for the analysis of learning algorithms, driven by collecting and sharing meta-data about learning

Appearing in *Proceedings of the ICML/COLT/UAI 2008 Planning to Learn Workshop (PlanLearn)*, Helsinki, Finland, 2008.

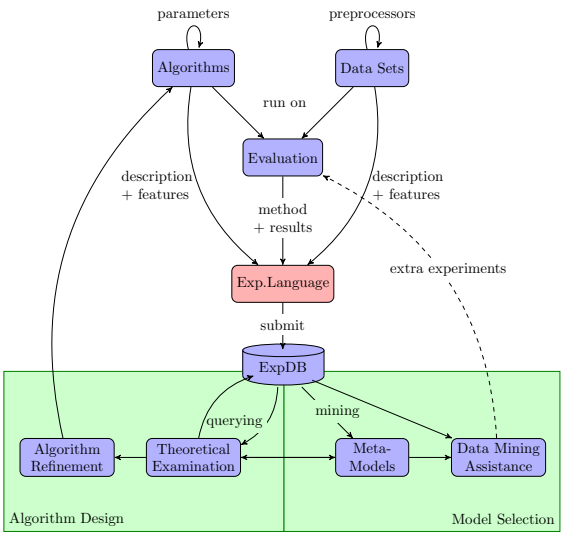

Figure 1. Using experiment databases[1]

methods, preprocessing methods and datasets in a uniform way, as illustrated in Fig. 1.

To share the results of any empirical machine learning study, they are first expressed in a standard experiment description language, capturing the details of the algorithms used, their parameter settings, the datasets on which they were run, the employed preprocessing steps, the evaluation methodology used and the results of that evaluation. Known features of the algorithms (e.g. its bias-variance profile) and datasets (e.g. landmarker evaluations) can also be added at any time. Next, the experiments are submitted to a repository, where they are automatically organized to make this information easily accessible and searchable. To this effect, each experiment is associated with previously stored experiments and linked to all known properties of the algorithms and datasets.

There are various ways to make use of such a resource. When evaluating or designing new methods, the repository could be queried to compare different algorithms or to test specific aspects of an algorithm's behavior, for instance how data properties or parameter settings affect its performance. The returned results can then be theoretically interpreted, possibly leading to new insights and/or improved algorithms. Furthermore, given the amount of available meta-data, it is also possible to mine the repository for patterns in algorithms behavior, or to use the shared data in data mining assistance tools to propose interesting approaches to new problems, possibly setting up more experiments to ascertain their validity.

[1]Figure adapted from a framework by Smith (2008).

The remainder of this paper is organized as follows. In Section 2, we discuss how experiment databases can presently serve as a useful repository for machine learning information. In Section 3, we show how to use such databases to answer various interesting research questions about the behavior of learning algorithms, and in Section 4 we discuss meta-models generated by mining the data. Finally, we consider some interesting future applications in Section 5, including possible uses in data mining assistance tools. Section 6 concludes.

2. Experiment Databases

An experiment database, as described in (Blockeel & Vanschoren, 2007), is a database specifically designed to store learning experiments in an organized fashion, including all details about the algorithms used, parameter settings, datasets, preprocessing methods, evaluation procedure and results. Once stored, all information can be easily accessed by writing the right database query, in standard SQL, providing a very versatile means to investigate large amounts of experimental results, both under very specific and very general conditions. With all details publicly available, they also ensure experiments can be easily reproduced and reused in future analysis.

Our implementation of such a database currently contains about 500,000 experiments of 54 classification algorithms on 87 datasets with 50 dataset characterizations, each evaluated on 36 evaluation metrics. It also contains some characterizations of the algorithms, like the model type, a bias-variance profile, and their compatibility with different types of data. It is available online at `http://expdb.cs.kuleuven.be`. This website also hosts a description of a standard experiment description language, available tools for uploading experiments, a gallery of SQL queries (including the ones used in the next section), and a query interface including visualization tools for displaying returned results.

3. Querying for Answers

To illustrate the use of such databases in various stages of the knowledge discovery process, we try a number of example queries, increasingly making use of the available meta-level descriptions of algorithms and datasets. While the first queries only use the recorded performance evaluations of specific algorithms on specific datasets, subsequent queries also use the stored data characteristics and algorithm features, offering increasingly generalizable results. A wider range of interesting queries can be found in Vanschoren et al. (2008).

3.1. Ranking Algorithms

When selecting interesting learning approaches, or when testing the effectiveness of a new algorithm, one is often interested in a ranking of the available methods. For instance, to investigate whether some algorithms consistently rank high over various problems, we can query for their average rank (using each algorithm's optimal observed performance) over a large number of datasets. Fig. 2 shows the result of a single query over all UCI datasets[2]. To check which algorithms perform significantly different, we used the Friedman test, as discussed in Demšar (2006). The right axis shows the average rank divided by the *critical difference*, meaning that two algorithms perform significantly different if the average ranks of two algorithms differ by at least that value (one unit)[3].

This immediately shows that indeed, some algorithms rank much higher on average than others on the UCI datasets. Boosting and Bagging, if used with the correct base-learners, perform significantly better than SMO (an SVM trainer), and SMO in turn performs better than J48 (C4.5). We cannot yet say that SMO is also better than the MultilayerPerceptron or RandomForests: more datasets (or fewer datasets in the comparison) are needed to reduce the critical difference. Note that the average rank of Bagging and Boosting is close to two, suggesting that a (theoretical) meta-algorithm that reliably chooses between the two approaches (and the underlying base-learner) would yield a very high rank[4].

Of course, to be fair, we should differentiate between different base-learners and kernels. We can drill down through the previous results by adjusting the query, additionally asking for the base-learners and kernels involved, yielding Fig. 3. Bagged naive Bayes trees seem to come in first overall, but the difference is not significant compared to that of SMO with a polynomial kernel (although it is compared to the RBF kernel). Also note that bagging and boosting PART and NBTrees seem to yield big performance boosts, while boosting random trees proves particularly ineffective.

Many more types of rankings can be generated with similar queries. For instance, if datasets are labeled as belonging to a specific task (e.g. image recognition), more specific queries could rank all algorithms for that particular task.

[2]We selected 18 algorithms to limit the amount of statistical error generated by using 'only' 87 datasets

[3]The critical difference was calculated using the Nemenyi test with p=0.1, 18 algorithms and 87 datasets

[4]Rerunning the query while joining Bagging and Boosting yields an average rank of 1.7, down from 2.5.

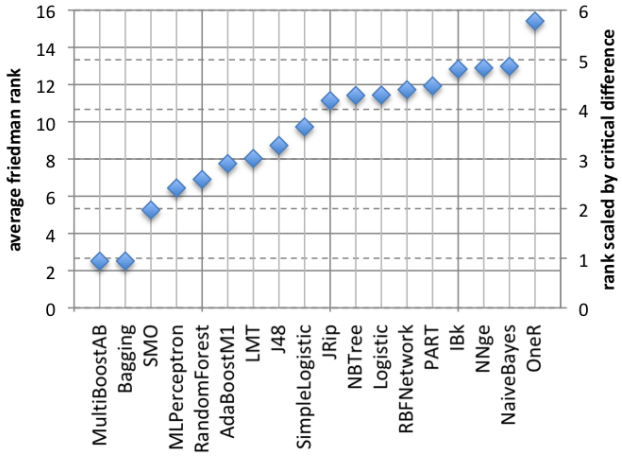

Figure 2. Algorithm ranking over all UCI datasets.

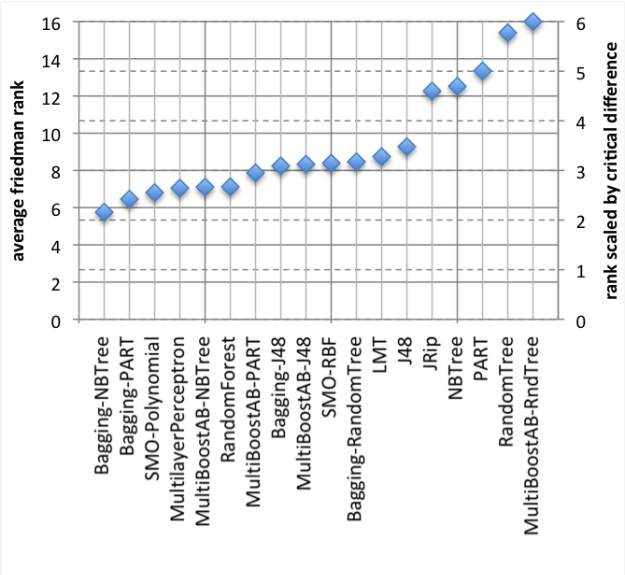

Figure 3. Algorithm ranking over all UCI datasets, including base-learners and kernels.

3.2. Tuning Parameters

Next to performance, there may be other reasons for selecting a certain algorithm. In that case, one still needs to tune the method's parameters to fit the given data. To learn from previously stored data, one could query for the effect of those parameters on other datasets. For instance, Fig. 4 shows the effect of the gamma parameter of the RBF kernel on a number of different datasets. Note that on some datasets, increasing the gamma value will steeply increase performance, until it reaches a maximum and then slowly declines, while on other datasets, performance immediately starts decreasing up to a point, after which it quickly drops to default accuracy. Querying for the number of attributes in each dataset (shown in brack-

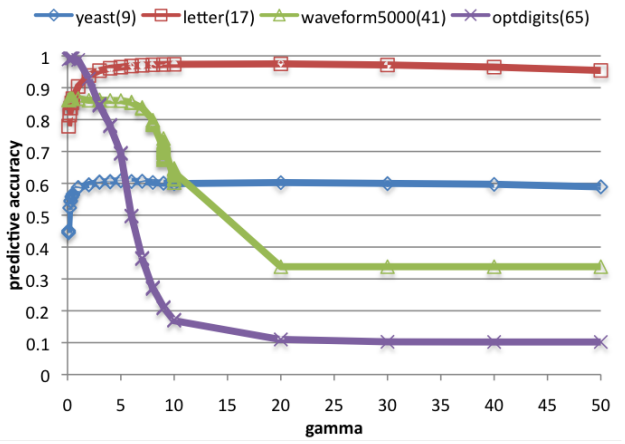

Figure 4. Effect of the RBF-kernel's gamma parameter on different datasets.

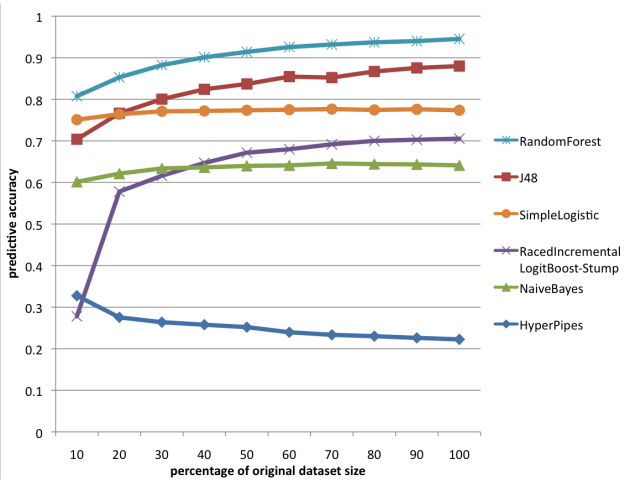

Figure 5. Learning curves on the Letter-dataset.

ets), suggests that only datasets with few attributes benefit from large gamma values[5]. This is one example where a query suggests ways to improve an algorithm: by taking the number of attributes into account, the kernel can be made less sensitive to this characteristic.

3.3. Applying Preprocessors

In many cases, data needs to be preprocessed before a certain learning algorithm can be used effectively. Since the database can, to a certain extent, also store preprocessing methods, we can investigate their effect on the performance of learning algorithms. For instance, to investigate how much data a certain algorithm needs to perform optimally, we can query for the results on downsampled versions of the dataset, yielding a learning curve for each algorithm, as shown in Fig. 5. Note that the learning curves cross, indicating that the ranking of algorithms depends on the size of the sample. While logistic regression is initially stronger than J48, the latter keeps on improving when given more data[6]. Also note that RandomForest is consequently better, improving steadily when given more data, that RacedIncrementalLogitBoost (with decision stumps as its base-learner) has a particularly steep learning curve, crossing two other curves, and that the performance of the HyperPipes algorithm actually worsens given more data.

However, to be truly useful for investigating the effect of preprocessing methods, the database model will need to be extended further, as discussed in Section 5.

[5]This relationship is investigated in more detail in Vanschoren et al. (2008).

[6]This confirms earlier analysis by Perlich et al. (2003), even though in that study, the dataset was transformed to a binary problem.

3.4. Generalizing over Algorithms

Instead of querying for a specific algorithm to use on certain types of data, we may also want to investigate which general properties of an algorithm might be desirable when approaching a certain task. One very interesting property of an algorithm is its bias-variance profile (Kalousis & Hilario, 2000). Since the database contains a large number of bias-variance decomposition experiments[7], we can give a realistic, numerical assessment of how capable each algorithm is in reducing bias and variance error. The average percentage of bias-related error, compared to the total error, is stored in the database as an algorithm property.

In a final query, we investigate the claim by Brain & Webb (2002) that on large datasets, the bias-component of the error becomes the most important factor, and that we should use algorithms with high bias management to tackle them. To verify this, we look for a connection between the dataset size and the proportion of bias error in the total error of a number of algorithms, selecting algorithms with very different bias-variance profiles. Averaging the bias-variance results over datasets of similar size for each algorithm produces the result shown in Fig. 6. It shows that bias error is of varying significance on small datasets, but steadily increases in importance on larger datasets, for all algorithms. This validates the previous study on a larger set of datasets. In this case (on UCI datasets), bias becomes the most important factor on datasets larger than 50000 examples, no matter which algorithm is used. As such, it is indeed advisable to look to algorithms with good bias management when dealing with large datasets.

[7]The database stores both Kohavi-Wolpert's and Webb's definition of bias/variance. We use the former.

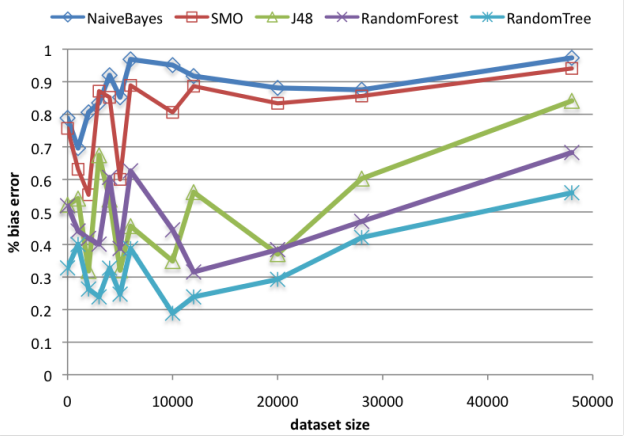

Figure 6. The average percentage of bias-related error in algorithms as a function of dataset size.

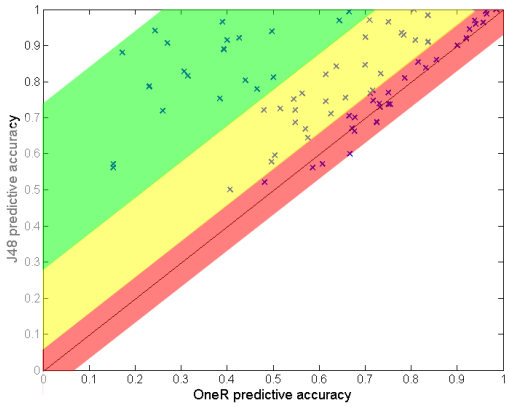

Figure 7. Performance comparison of J48 and OneR on all UCI datasets, discretized in 3 classes: draw (red), win_J48 (yellow), large_win_J48 (green).

4. Mining for Patterns

Instead of studying different dataset properties independently, we could also use data mining techniques to relate the effect of many different properties to an algorithm's performance. For instance, when looking at Fig. 2, we see that OneR ranks, on average, much lower than the other algorithms, while earlier studies, most notably one by Holte (1993), found very little performance difference between OneR and the more complex J48. In fact, when querying for the default performance of OneR and J48 on all UCI datasets, and plotting them against each other, we get Fig. 7, showing that on a large number of datasets, their performance is indeed about the same (their performances cross near the diagonal). Still, J48 works much better on many other datasets.

To automatically learn under which conditions J48 clearly outperforms OneR, we queried for the characteristics of each dataset, and discretized the data into three class values: "draw", "win_J48" (>4% gain), and "large_win_J48" (>20% gain). The tree returned by J48 on this meta-dataset is shown in Fig. 8, showing that a high number of class values often leads to a large win of J48 over 1R.

Probably, many more meta-models remain to be discovered by simply querying the database and and mining the results.

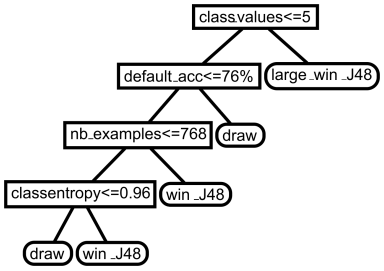

Figure 8. Meta-decision tree showing on which data J48 is better than OneR

5. Future Work

Although the system discussed above readily allows users to share and explore machine learning meta-data, there are several opportunities for further development.

First of all, the current support for preprocessing methods is rather limited as the database can currently only store sequences of preprocessing steps. In reality, complex workflows are used to prepare the data before modeling, and the database model should be extended to capture such workflows. Unfortunately, there is, to our knowledge, no standard language to describe KDD workflows. Still, this is a challenge, not a limitation. One interesting approach using ontologies to describe preprocessing steps in bio-informatics can be found in Pérez-Rey et al. (2006).

Secondly, next to querying and mining the repository, it would be interesting to use the available meta-data and querying possibilities in data mining assistance tools. A number of such tools have been proposed before, some of which could be enhanced by making use of the repository. One very interesting approach is the Intelligent Discovery Assistant proposed by Bernstein et al. (2005), which defines an ontology for the data mining process and uses it to generate a ranking of interesting approaches to handle a certain data mining

task. However, the effects and properties of preprocessors and algorithms are hard-coded. Using an experiment repository could engender a much more flexible system, querying the database to see, for instance, how fast an algorithm would be on a certain dataset, or what the preconditions are for applying newly entered algorithms.

New installments of algorithm ranking tools, like the MetaL ranker (Brazdil et al., 2003) can also be envisaged. For instance, such a ranker would be able to use any stored evaluation metric, or a combination thereof, take parameter variations into account and include viable preprocessing steps. Furthermore, it could be given a certain amount of time to optimize its predictions by automatically running additional experiments.

6. Conclusions

We propose a community-based approach for the analysis of learning algorithms in which both machine learning researchers and practitioners can share metadata from learning experiments in a uniform way, and upload this information to a searchable experiment database. We illustrate the use of such databases in various stages of the knowledge discovery process by discussing a number of example queries, increasingly making use of the available meta-level descriptions of algorithms and datasets. These include ranking all algorithms on a group of datasets, comparing parameter effects on different types of datasets, using preprocessors to draw learning curves, and plotting the average bias-variance ratio of several algorithms against the dataset size, each in a single query. Next, we illustrated how the available meta-data can also be automatically mined for patterns in algorithm behavior. Finally, we discussed possibilities for future work, including applications in data mining assistance tools.

Acknowledgements

Hendrik Blockeel is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (F.W.O.-Vlaanderen), and this research is further supported by GOA 2003/08 “Inductive Knowledge Bases”.

References

Ali, S., & Smith, K. A. (2006). On learning algorithm selection for classification. *Applied Soft Computing*, *6*, 119–138.

Bernstein, A., Provost, F., & Hill, S. (2005). Toward intelligent assistance for a data mining process: An ontology-based approach for cost-sensitive classification. *IEEE Transactions on Knowledge and Data Engineering*, *17*, 503–518.

Blockeel, H., & Vanschoren, J. (2007). Experiment databases: Towards an improved experimental methodology in machine learning. *Lecture Notes in Computer Science 4702: PKDD 2007 Proceedings* (pp. 6–17). Springer.

Brain, D., & Webb, G. (2002). The need for low bias algorithms in classification learning from large data sets. *Lecture Notes in Computer Science 2431: PKDD 2002 Proceedings* (pp. 62–73).

Brazdil, P., Soares, C., & Pinto da Costa, J. (2003). Ranking learning algorithms: Using ibl and meta-learning on accuracy and time results. *Machine Learning*, *50*, 251–277.

Caruana, R., & Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. *ICML '06: Proceedings of the 23rd international conference on Machine learning* (pp. 161–168). Pittsburgh, Pennsylvania.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J.Mach.Learn.Res.*, *7*, 1–30.

Holte, R. (1993). Very simple classification rules perform well on most commonly used datasets. *Machine Learning*, *11*, 63–91.

Kalousis, A., & Hilario, M. (2000). Building algorithm profiles for prior model selection in knowledge discovery systems. *Engineering Intell. Syst.*, *8(2)*.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (1994). *Machine learning, neural and statistical classification*. Ellis Horwood.

Pérez-Rey, D., Anguita, A., & Crespo, J. (2006). Ontodataclean: Ontology-based integration and preprocessing of distributed data. *ISBMDA* (pp. 262–272).

Perlich, C., Provost, F., & Simonoff, J. S. (2003). Tree induction vs. logistic regression: a learning-curve analysis. *J. Mach. Learn. Res.*, *4*, 211–255.

Smith-Miles, K. (2008). Cross-disciplinary perspectives on the algorithm selection problem: recent advances and emerging opportunities. *ACM Computing Surveys*, to appear.

Vanschoren, J., Pfahringer, B., & Holmes, G. (2008). *Learning from the past with experiment databases.* (Technical Report Working Paper Series 08/2008). Computer Science Department, University of Waikato.