Working Paper Series
ISSN 1177-777X

LEARNING FROM THE PAST WITH EXPERIMENT DATABASES

Joaquin Vanschoren & Bernhard Pfahringer & Geoff Holmes

Working Paper: 08/2008
June 24, 2008

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

LEARNING FROM THE PAST WITH EXPERIMENT DATABASES

Joaquin Vanschoren[1] & Bernhard Pfahringer[2] & Geoff Holmes[2]
[1] Katholieke Universiteit Leuven, Leuven, Belgium
[2] University of Waikato, Hamilton, New Zealand
joaquin.vanschoren@cs.kuleuven.be
{bernhard,geoff}@cs.waikato.ac.nz

June 24, 2008

Abstract

Thousands of Machine Learning research papers contain experimental comparisons that usually have been conducted with a single focus of interest, and detailed results are usually lost after publication. Once past experiments are collected in experiment databases they allow for additional and possibly much broader investigation. In this paper, we show how to use such a repository to answer various interesting research questions about learning algorithms and to verify a number of recent studies. Alongside performing elaborate comparisons and rankings of algorithms, we also investigate the effects of algorithm parameters and data properties, and study the learning curves and bias-variance profiles of algorithms to gain deeper insights into their behavior.

1 Introduction

"Study the past", Confucius said, "if you would divine the future". Also in machine learning, studying the results of earlier analysis is essential to gain a deeper understanding of learning methods.

As learning algorithms are typically heuristic in nature, learning experiments are needed to investigate the (combined) effects of different kinds of data, different preprocessing methods and, in many cases, different parameter settings. With so many factors influencing an algorithm's behavior, these experiments are (or should be) quite general, which means that they probably have more uses than originally intended.

To keep these learning experiments for future use, they can be submitted to an experiment database [1]. This is a database specifically designed to store learning experiments, including all details about the algorithms used, parameter settings, datasets, preprocessing methods, evaluation procedures and results. When new learning experiments, as well as meta-level descriptions of its

components, are submitted to the database, they are automatically stored in a well-organized way, associating them with all other experiments and theoretical properties.

All this information can then be accessed by writing the right database query (e.g. in SQL). As we will demonstrate, this provides a very versatile means to investigate large amounts of experimental results, both under very specific and very general conditions. Furthermore, storing learning experiments in a unified fashion allows the creation of experiment repositories in which large amounts of experimental data can be gathered and reused for many future studies.

In this paper, we make use of such an experiment database, reusing its experiments to gain new insights into a range of machine learning questions. More specifically, we distinguish between three types of studies, increasingly making use of the available meta-level descriptions, and offering increasingly generalizable results[1]:

1. Model-level analysis. These studies evaluate the produced models through a range of performance measures, but typically consider only individual datasets and algorithms. They typically try to identify HOW a specific algorithm performs, either on average or under specific conditions.

2. Data-level analysis. These studies investigate how known or measured data properties, not individual datasets, affect the performance of specific algorithms. They identify WHEN (on which kinds of data) an algorithm can be expected to behave a certain way.

3. Method-level analysis. These studies don't look at individual algorithms, but take general properties of the algorithms (eg. their bias-variance profile) into account, using these properties to identify WHY an algorithm behaves a certain way.

In the next section, we first give a short overview of the experiments available in the database. The three ensuing sections cover the different types of studies mentioned above. Section 6 concludes.

2 A Repository of Learning Experiments

The experiment database used in this study contains about 500.000 experiments of supervised classification. It stores 54 classification algorithms from the WEKA platform[10], together with all their parameters. It also holds 86 commonly used classification datasets taken from the UCI repository, described by 56 data characteristics, most of which are mentioned in [7]. Moreover, it contains a range of preprocessed datasets created by sampling the original datasets by removing 10%, 20%,... of their instances, and by applying feature selection with Correlation-based Feature Subset Selection using the default best-first search method [4].

[1] A similar distinction is identified by Van Someren [9].

As for the available experiments, it contains the results of running all algorithms, with default parameter settings, on all datasets. Furthermore, the algorithms SMO (a support vector machine trainer), MultilayerPerceptron, J48 (C4.5), 1R, Random Forests, Bagging and Boosting are varied over their most important parameter settings[2]. For all these algorithms, 20 sensible values were defined for each parameter, and the algorithms were run using those values while keeping the other parameters on their default value. In the case of J48, Bagging and 1R, parameters were additionally varied randomly to achieve at least 1000 experiments per algorithm on each dataset. For all randomized algorithms, each experiment was repeated 20 times with different random seeds. All experiments where evaluated with 10-fold cross-validation, using the same folds on each dataset, and a large subset was additionally evaluated with a bias-variance analysis.

The database is publicly available on `http://expdb.cs.kuleuven.be/`. All SQL queries used in this paper (not printed here because of space constraints) are also available there, and most of the graphs can be instantly generated online as well.

3 Model-level analysis

In the first type of study, we are interested in how individual algorithms perform on specific datasets. This is the most common type of study, typically used to benchmark, compare or rank algorithms, but also to investigate how specific parameter settings affect performance.

3.1 Comparing algorithms

To compare the performance of all algorithms on one specific dataset, we could select the name of the algorithm used and the predictive accuracy recorded in all experiments on, for instance, the dataset 'letter'. For more detail, we can also select the kernel in the case of a SVM and the base-learner in the case of an ensemble. We order the results by their performance and plot the results in Fig. 1.

Since the returned results are always as general as the query allows, we now have a complete overview of how each algorithm performed. Next to their optimal performance, it is also immediately clear how much variance is caused by suboptimal parameter settings (at least for those algorithms whose parameters were varied). For instance, when looking at SVMs, it is clear that especially the RBF-kernel is of great use here, while the polynomial kernel is much less interesting[3]. Still, there is still much variation in the performance of the SVM's, so it might be interesting to investigate this in more detail. Also, while most algorithms vary smoothly as their parameters are altered, there seem to be large

[2]For the ensemble methods, all non-ensemble learners were used as possible base-learners, each with default parameter settings.

[3]RBF kernels are popular in letter recognition problems.

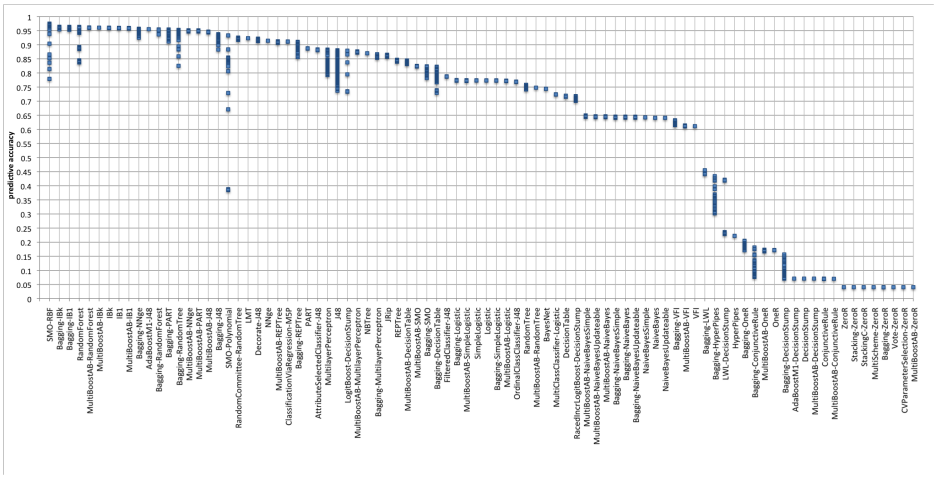

Figure 1: Performance of all algorithms on dataset 'letter', including base-learners and kernels.

jumps in the performances of SVMs and RandomForests, which are, in all likelihood, caused by parameters that heavily affect their performance. Moreover, when looking at bagging and boosting, it is clear that some base-learners are much more interesting than others. For instance, it appears that while bagging and boosting give an extra edge to the Nearest Neighbor algorithms, the effect is rather limited, and the same holds for Logistic Regression. Conversely, bagging RandomTree seems to be hugely profitable, but this does not hold for boosting. It also seems more rewarding to fine-tune RandomForests, MultiLayerPerceptrons and SVMs than to bag or boost their default setting. Still, this is only one dataset, further querying is needed. Given the generality of the returned results, each query is likely to highlight things we were not expecting, providing interesting cases for further study.

3.2 Investigating parameter effects

First, we examine the effect of the parameters of the RBF kernel. Based on the first query, we can zoom in on the SVM's results by adding a constraint, and additionally ask for the value of the parameter we are interested in. Selecting the value of the gamma parameter and plotting the results, we obtain Fig. 2. While we are doing that, we can just as easily ask for the effect of this parameter on a number of other datasets as well.

When comparing the effect of gamma to the variation in RBF-kernel performance in the previous plot, we see that the variation corresponds exactly with the variation caused by this parameter. On the 'letter' dataset, performance increases when increasing gamma up to value 20, after which it slowly declines. The other curves show that the effect on other datasets is very different. On some datasets, performance steeply increases until reaching a maximum and

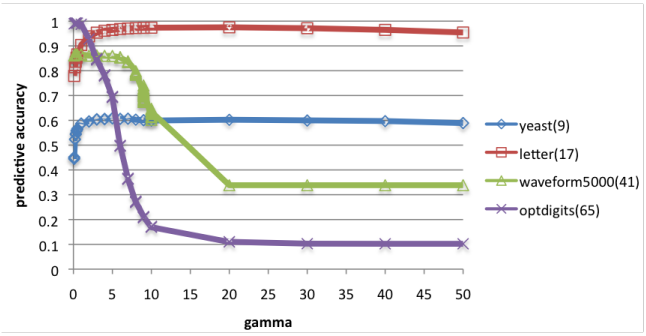

Figure 2: The effect of parameter gamma of the RBF-kernel in SVMs on a number of different datasets.

then slowly declines, while on other datasets, performance immediately starts decreasing up to a point, after which it quickly drops down to default accuracy. Looking at the number of attributes in each dataset (shown in brackets) seems to show some correlation, which we will investigate further in Sect.4.1.

3.3 Ranking algorithms

Previous queries investigated the performance of algorithms under rather specific conditions. Yet, by just dropping the constraints on the datasets used, we can query for their performance over a large number of different problems.

An interesting and sizable comparison of supervised learning algorithms was performed by Caruana and Niculescu-Mizil [3]. Most interestingly, this study compares across different performance measures by normalizing all performance metrics between the baseline performance and the best observed performance on each dataset. Using the aggregation functions of SQL, we can do this normalization right inside a query[4].

To verify the conclusions of [3], we select all datasets, and all algorithms whose parameters were varied (see Sect.2). We also added naive bayes, logistic regression and 1NN, but only as a point of comparison, their ranking should not be interpreted as optimal. As for the performance metrics, we used predictive accuracy, F-score, precision and recall, the last three of which were averaged over all classes. We then queried for the maximal (normalized) performance of each algorithm for each metric, averaged over all datasets, and then averaged over all metrics to obtain the overall score for each algorithm. The results are shown in Fig.3.

Taking care not to overload the figure, we compacted groups of similar and similarly performing algorithms, indicated with an asterix (*). The overall best performing algorithms are mostly bagged and, to a lesser extent, boosted ensem-

[4]We normalized between the performance of the algorithm ZeroR and the maximum observed performance over all algorithms on each dataset.

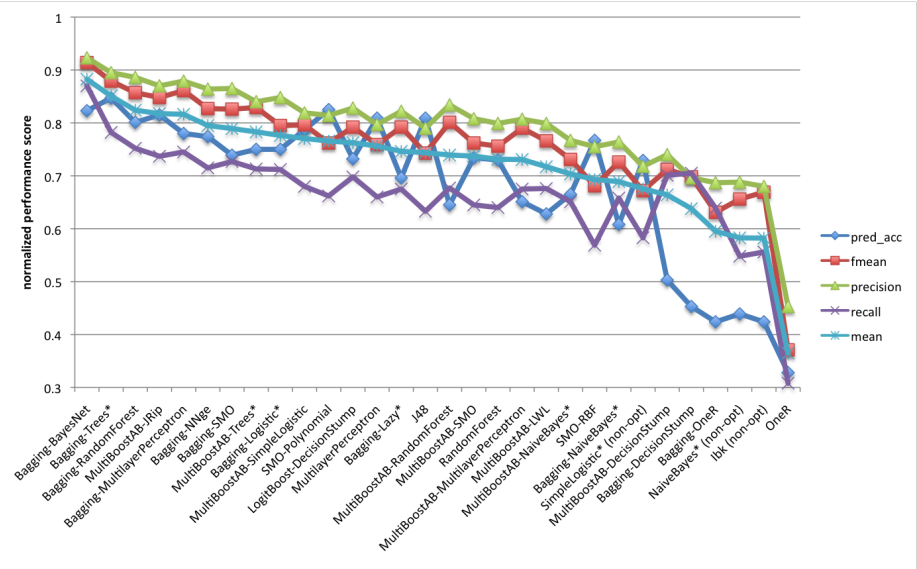

Figure 3: Ranking of algorithms over all datasets and over different performance metrics.

bles. Especially bagged trees[5] perform very well, which corresponds nicely to [3]. Another shared conclusion is that boosting full trees performs dramatically better than boosting stumps. One notable difference is that RandomForests and MultiLayerPerceptrons perform much worse in our study, while J48 seems to perform better. A possible explanation lies in the fact that we use a somewhat different set of performance measures in the ranking, although it may also depend on the (non-binary) datasets used. Furthermore, this study contains many more algorithms, in particular, the bagged versions of other strong learners (BayesNet, RandomForest, MultiLayerPerceptron,etc.) which perform very well in this ranking. Note however that these bagged versions primarily improve on precision and recall, while the original base-learners perform better on accuracy.

While this is a very comprehensive comparison of learning algorithms, each such comparison is still only a snapshot in time. However, as new algorithms, datasets and experiments are added to the database, one can at any time rerun the query and immediately see how things have evolved.

4 Data-level analysis

While the queries in the previous section allow the examination of the behavior of learning algorithms to a high degree of detail, they give no indication of exactly *when* (on which kind of datasets) a certain behavior is to be expected.

[5]These are PART, LMT, NBTree, J48 and similar tree-based learners.

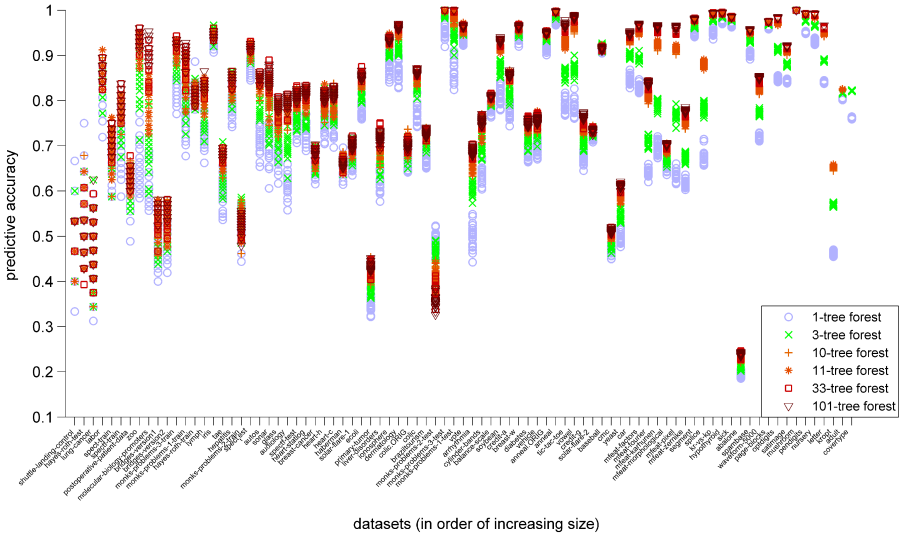

Figure 4: The effect of dataset size and the number of trees for random forests.

In order to obtain results that generalize over different datasets, we need to look at the properties of each individual dataset, and investigate how they affect learning performance.

4.1 Investigating the effect of specific data properties

In a first such study, we examine what causes the 'performance jumps' that we noticed with the Random Forest algorithm in Fig. 1. Querying for the effect of the number of trees in the forest on all datasets, ordering from small to large yields Fig. 4. This shows that predictive accuracy increases with the number of trees, usually leveling off between 33 and 101 trees[6]. We also see that as dataset size increases, the accuracies for a given forest size vary less as trees become more stable on large datasets. For very small datasets, the benefit of using more trees is overpowered by the randomness in the trees. For very large datasets, stable trees result in performance jumps between different forest sizes, which also indicates that the trees must still be quite different from each other.

A second effect we can investigate is whether the optimal value for the gamma-parameter of the RBF-kernel is indeed linked to the number of attributes in the dataset. After querying for the relationship between the gamma-value corresponding with the optimal performance and the number of attributes in the dataset used, we get Fig. 5.

Although the number of attributes and the optimal gamma-value are not directly correlated, it is clear that high optimal gamma values only occur on

[6] `monks-problems-2_test` is a notable exception: obtaining less than 50% accuracy on a binary problem, it actually performs worse as more trees are included.

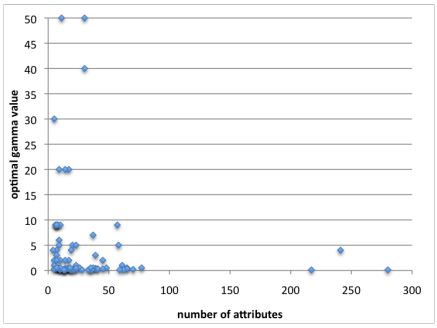

Figure 5: The effect of the number of attributes on the optimal gamma-value.

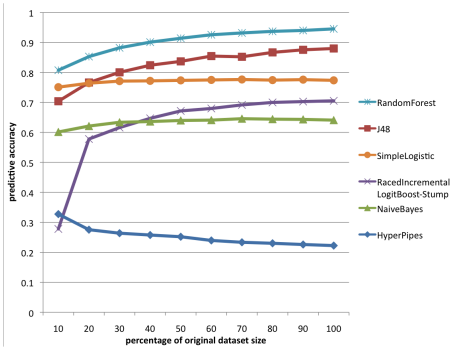

Figure 6: Learning curves on the Letter-dataset.

datasets with a small number of attributes. A possible explanation for this lies in the fact that SMO normalizes all attributes into the interval [0,1]. Therefore, the maximal squared distance between two examples, $\sum (a_i - b_i)^2$ for every attribute i, is equal to the number of attributes. Since the RBF-Kernel computes $e^{(-\gamma * \sum (a_i - b_i)^2)}$, the kernel value will go to zero very quickly for large gamma-values and a large number of attributes, making the non-zero neighborhood around a support vector very small. Consequently, SMO will overfit these support vectors, resulting in low accuracies. This suggests that the RBF kernel should take the number of attributes into account to make the default gamma value more suitable across a range of datasets. It also illustrates how the experiment database allows the investigation of algorithms in detail and assist their development.

4.2 Investigating the effect of preprocessing methods

Since the database can also store preprocessing methods, we can investigate their effect on the performance of learning algorithms. For instance, to investigate if the results in Fig. 2 are also valid on smaller samples of the 'letter' dataset, we can query for the results on downsampled versions of the dataset, yielding a learning curve for each algorithm, as shown in Fig. 6. It is now clear that the ranking of algorithms also depends on the size of the sample. While logistic regression is initially stronger than J48, the latter keeps on improving when given more data[7]. Also note that RandomForest is consequently better, that RacedIncrementalLogitBoost has a particularly steep learning curve, crossing two other curves, and that the performance of the HyperPipes algorithm actually worsens given more data.

[7]This confirms earlier analysis by Perlich et al. [8], even though in that study, the dataset was transformed to a binary problem.

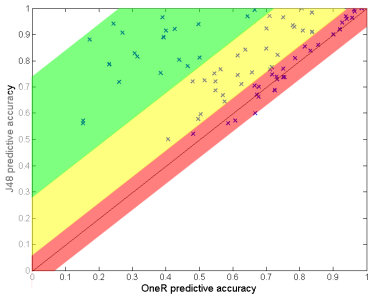

Figure 7: J48 vs. 1R on all datasets, discretized in draw (red), win_J48 (yellow) and large_win_J48 (green).

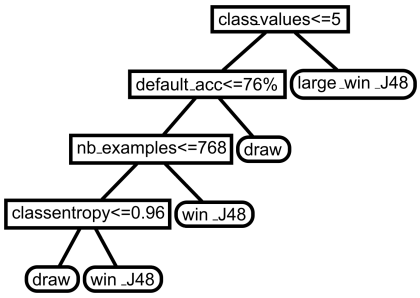

Figure 8: A meta-decision tree predicting J48's superiority over OneR based on dataset characteristics.

4.3 Mining for patterns in learning behavior

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. 3, we see that OneR performs obviously much worse than the other algorithms. Still, some earlier studies, most notably one by Holte [5], found very little performance differences 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 3 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. When we query for the date (the date it was entered into the UCI repository) and number of classes of each dataset (see Fig. 9), we see a possible explanation for the earlier reported results. The datasets in [5] were all from the period 1988-1989, and few of those datasets have many class values. Fig. 10 displays the average gain of J48 over OneR per year, showing that, in general, datasets have become harder for OneR over time.

5 Method level analysis

While the results in the previous section are clearly more generalizable towards the datasets used, they don't explain *why* algorithms behave a certain way. They only consider individual algorithms and thus do not generalize over different techniques. Hence, we need to extend the description of learning algorithms with a range of algorithm properties, and include these in our queries.

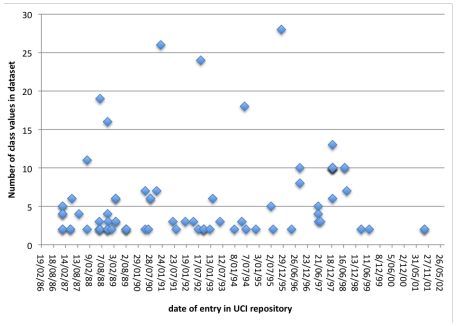

Figure 9: Number of classes in UCI datasets over time.

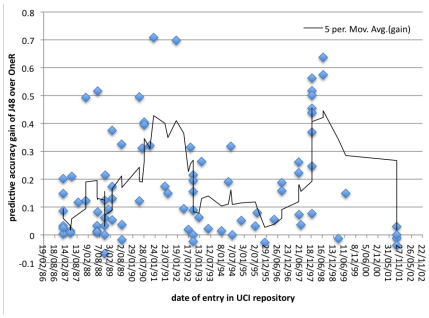

Figure 10: Gain of J48 over OneR over time and moving average.

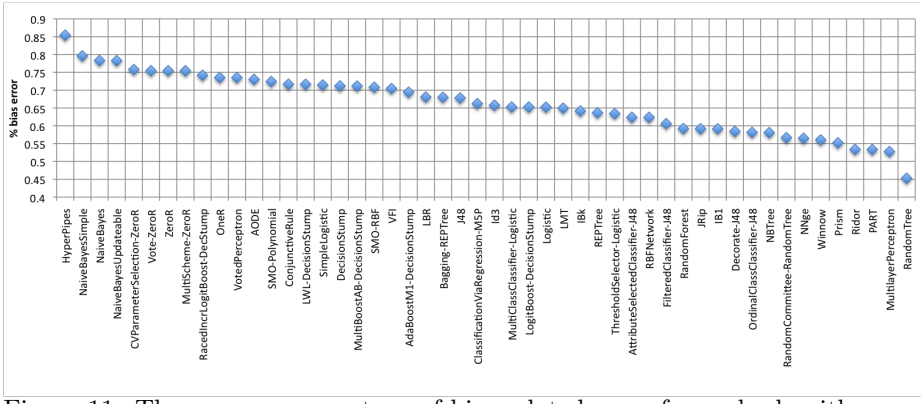

Figure 11: The average percentage of bias-related error for each algorithm averaged over all datasets.

5.1 Bias-variance profiles

One very interesting property of an algorithm is it's bias-variance profile[6]. Since the database contains a large number of bias-variance decomposition experiments[8], we can give a realistic, numerical assessment of how capable each algorithm is in reducing bias and variance error. In Fig. 11 we show, for each algorithm, the proportion of the total error that can be attributed to bias error, using default parameter settings and averaged over all datasets.

The algorithms are ordered from large bias (low variance), to low bias (high variance). NaiveBayes is, as expected, one of the algorithms with the strongest variance management, but poor bias management, while RandomTree has very good bias management, but generates more variance error. When looking at the ensemble methods, it also shows that Bagging is a variance-reduction method, as

[8]The database stores both Kohavi-Wolpert's and Webb's definition of bias/variance, but we use the former in our queries.

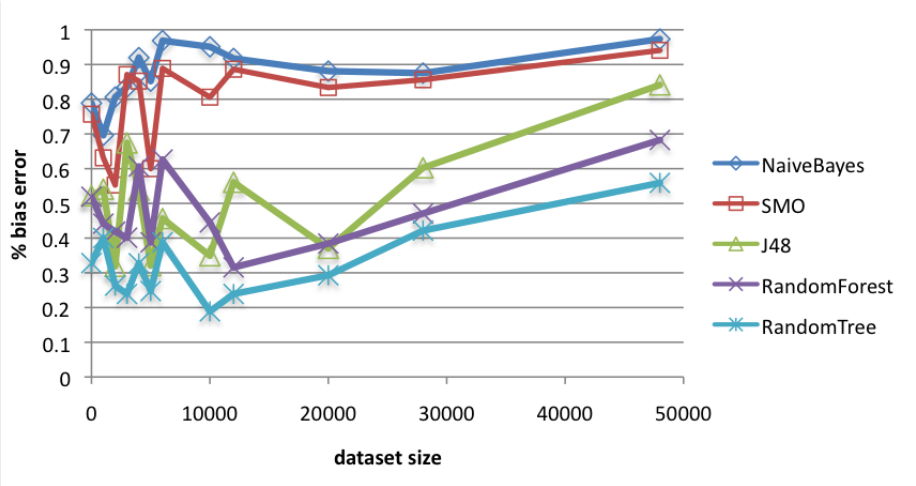

Figure 12: The average percentage of bias-related error in algorithms as a function of dataset size.

it causes REPTree to shift significantly to the left. Conversely, Boosting reduces bias, shifting DecisionStump to the right in AdaBoostM1 and LogitBoost.

5.2 Investigating bias-variance effects

As a final study, we investigate the claim by Brain and Webb [2] 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, using the previous figure to select 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. 12. 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.

6 Conclusions

Much can be learned by looking at past learning experiments, and the creation of repositories of learning experiments provides an effective way of tapping into this information, often yielding surprising new insights or generating interesting research questions. In a series of increasingly in-depth studies, we first used such a repository to perform an elaborate comparison and ranking of supervised classification algorithms. Next, the available data characteristics were used to investigate their effects on learning performance and we discovered relationships that suggest further improvements on learning algorithms, as well as meta-models of algorithm performance. Taking preprocessing methods into account, we also found crossing learning curves for several algorithms. Finally, we studied the bias-variance profiles of learning algorithms, and provided further evidence that managing bias error is particularly important on large datasets. We are confident that many more interesting results can be discovered by learning from past experiments. In the words of Albert Einstein, "Learn from yesterday, live for today, hope for tomorrow. The important thing is not to stop questioning."

References

[1] Blockeel, H. and Vanschoren, J.: Experiment databases: Towards an improved experimental methodology in machine learning. PKDD '07: Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases. Lecture Notes in Computer Science **4702** (2007) 6-17

[2] Brain D, Webb G.: The Need for Low Bias Algorithms in Classification Learning from Large Data Sets. PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery (2002) 62–73

[3] Caruana R. and Niculescu-Mizil A.: An empirical comparison of supervised learning algorithms. ICML '06: Proceedings of the 23rd international conference on Machine learning (2006) 161–168

[4] Hall, M. A. Correlation-based Feature Selection for Machine Learning. Ph.D diss. Hamilton, NZ: Waikato University, Department of Computer Science (1998)

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

[6] Kalousis, A. and Hilario, M.: Building Algorithm Profiles for prior Model Selection in Knowledge Discovery Systems. Engineering Intelligent Systems **8(2)** (2000)

[7] Peng, Y. et al.: Improved Dataset Characterisation for Meta-Learning. Lecture Notes in Computer Science **2534** (2002) 141–152

[8] Perlich, C. and Provost, F. and Siminoff, J.: Tree induction vs. logistic regression: A learning curve analysis. Journal of Machine Learning Research **4** (2003) 211–255

[9] Van Someren, M.: Model Class Selection and Construction: Beyond the Procrustean Approach to Machine Learning Applications. Lecture Notes in Computer Science **2049** (2001) 196–217

[10] Witten, I.H. and Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques (2nd edition). Morgan Kaufmann (2005)