Investigating Classifier Learning Behavior with Experiment Databases

Joaquin Vanschoren and Hendrik Blockeel

Computer Science Dept., K.U.Leuven, Celestijnenlaan 200A, 3001 Leuven, Belgium

Abstract. Experimental assessment of the performance of classification algorithms is an important aspect of their development and application on real-world problems. To facilitate this analysis, large numbers of such experiments can be stored in an organized manner and in complete detail in an *experiment database*. Such databases serve as a detailed log of previously performed experiments and a repository of verifiable learning experiments that can be reused by different researchers. We present an existing database containing 250,000 runs of classifier learning systems, and show how it can be queried and mined to answer a wide range of questions on learning behavior. We believe such databases may become a valuable resource for classification researchers and practitioners alike.

1 Introduction

Supervised classification is the task of learning from a set of classified training examples $(x, c(x))$, where $x \in X$ (the instance space) and $c(x) \in C$ (a finite set of classes), a classifier function $f : X \rightarrow C$ such that f approximates c (the target function) over X. Most of the existing algorithms for learning f are heuristic in nature, and try to (quickly) approach c by making some assumptions that may or may not hold for the given data. They assume c to be part of some designated set of functions (the hypothesis space), deem some functions more likely than others, and strictly consider consistency with the observed training examples (not with X as a whole). While there is theory relating such heuristics to finding c, in many cases this relationship is not so clear, and the utility of a certain algorithm needs to be evaluated empirically.

As in other empirical sciences, experiments should be performed and described in such a way that they are easily verifiable by other researchers. However, given the fact that the exact algorithm implementation used, its chosen parameter settings, the used datasets and the experimental methodology all influence the outcome of an experiment, it is practically not self-evident to completely describe such experiments. Furthermore, there exist complex interactions between data properties, parameter settings and the performance

of learning algorithms. Hence, to thoroughly study these interactions and to assess the generality of observed trends, we need a sufficiently large sample of experiments, covering many different conditions, organized in a way that makes their results easily accessible and interpretable.

For these reasons, Blockeel (2006) proposed the use of *experiment databases*: databases describing a large number of learning experiments in complete detail, serving as a detailed log of previously performed experiments and an (online available) repository of learning experiments that can be reused by different researchers. Blockeel and Vanschoren (2007) provide a detailed account of the advantages and disadvantages of experiment databases, and give guidelines for designing them. As a proof of the concept, they present a concrete implementation that contains a full description of the experimental conditions and results of 250,000 runs of classifier learning systems, together with a few examples of its use and results that were obtained from it.

In this paper we provide a more detailed discussion of how this database can be used in practice to store the results of many learning experiments and to obtain a clear picture of the performance of the involved algorithms and the effects of parameter settings and dataset characteristics. We believe that this discussion may be of interest to anyone who may want to use this database for their own purposes, or set up a similar databases for their own research.

We describe the structure of the database in Sect. 2 and the experiments in Sect. 3. In Sect. 4 we illustrate the power of this database by showing how SQL queries and data mining techniques can be used to investigate classifier learning behavior. Section 5 concludes.

2 A Database for Classification Experiments

To efficiently store and allow queries about all aspects of previously performed classification experiments, the relationships between the involved learning algorithms, datasets, experimental procedures and results are captured in the database structure, shown in Fig. 1. Since many of these aspects are parameterized, we use *instantiations* to uniquely describe them. As such, an `Experiment` (central in the figure) consists of instantiations of the used learner, dataset and evaluation method.

First, a `Learner instantiation` points to a learning algorithm (`Learner`), which is described by the algorithm name, version number, a url where it can be downloaded, and some generally known or calculated properties (Van Someren (2001), Kalousis & Hilario (2000)), like the used approach (e.g. neural networks) or how susceptible it is to noise. Then, if an algorithm is parameterized, the parameter settings used in each learner instantiation (one of which is set as default) are stored in table `Learner_parval`. Because algorithms have different numbers and kinds of parameters, we store each parameter value assignment in a different row (in Fig. 1 only two are shown). A

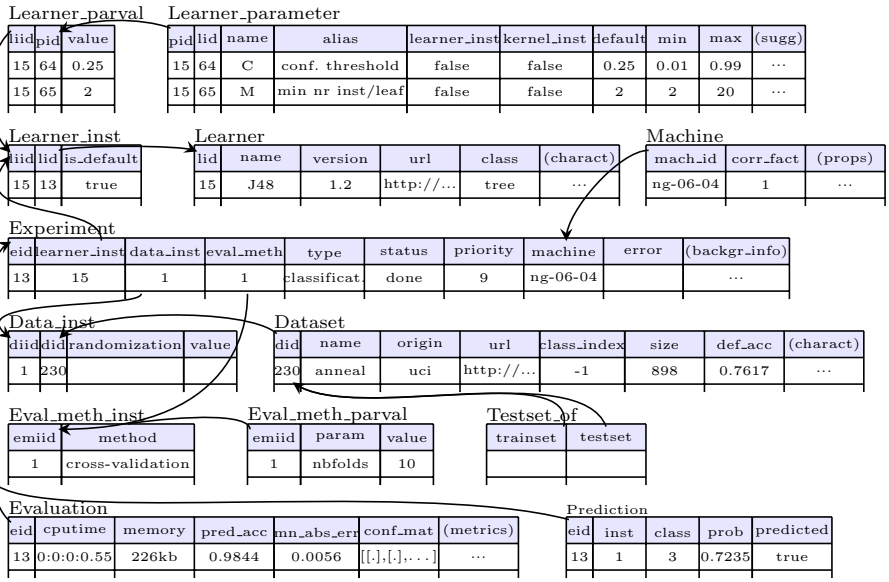


Fig. 1. A simplified schema of the experiment database.

`Learner_parameter` is described by the learner it belongs to, its name and a specification of sensible or suggested values, to facilitate experimentation.

Secondly, the used `Dataset`, which can be instantiated with a randomization of the order of its attributes or examples (e.g. for incremental learners), is recorded by its name, download url(s), the index of the class attribute and some information on its origin (e.g. to which repository it belongs or how it was generated artificially). In order to investigate whether the performance of an algorithm is linked to certain kinds of data, a large set of dataset characterization metrics is stored, most of which are described in Peng et al. (2002). These can be useful to help gain insight into an algorithm's behavior and, conversely, assess a learner's suitability for handling new learning problems[1].

Finally, we must store an evaluation of the experiments. The evaluation method (e.g. cross-validation) is stored together with its parameters (e.g. the number of folds). If a dataset is divided into a training set and a test set, this is defined in table `Testset_of`. The result of the evaluation of each experiment is described in table `Evaluation` by a wide range of evaluation metrics for classification, including the contingency tables. To compare cpu times, a factor describing the relative speed of the used `Machine` is stored as part of the machine description. The last table in Fig. 1 stores the (probabilities of the) predictions returned by each experiment, which may be used to calculate new performance measures without rerunning the experiments.

[1] New data and algorithm characterizations can be added at any time by adding more columns and calculating the characterizations for all datasets or algorithms.

3 The Experiments

To populate the database with experiments, we selected 54 classification algorithms from the WEKA platform (Witten and Frank (2005)) and inserted them together with all their parameters. Also, 86 commonly used classification datasets were taken from the UCI repository and inserted together with their calculated characteristics. Then, to generate a sample of classification experiments that covers a wide range of conditions, while also allowing to test the performance of some algorithms under very specific conditions, some algorithms were explored more thoroughly than others.

First, we ran all experiments with their default parameter settings on all datasets. Secondly, we defined sensible values for the most important parameters of the algorithms SMO (a kernel method), MultilayerPerceptron, J48 (C4.5), 1R and Random Forests and varied each of these parameters one by one, while keeping all other parameters at default. Finally, we further explored the parameter spaces of J48 and 1R by selecting random parameter settings until we had about 1000 experiments on each dataset. For all randomized algorithms, each experiment was repeated 20 times with different random seeds. All experiments (about 250,000 in total) where evaluated using 10-fold cross-validation, using the same folds for each dataset.

An online interface is available at `http://www.cs.kuleuven.be/~dtai/expdb/` for those who want to reuse experiments for their own purposes, together with a full description and code which may be of use to set up similar databases, for example to store, analyse and publish the results of large benchmark studies.

4 Using the database

We will now illustrate how easy it is to use this experiment database to investigate a wide range of questions on the behavior of learning algorithms by simply writing the right queries and interpreting the results, or by applying data mining algorithms to model more complex interactions.

4.1 Comparing different algorithms

A first question may be "How do all algorithms in this database compare on a specific dataset D?" To investigate this, we query for the learning algorithm name and evaluation result (e.g. predictive accuracy), linked to all experiments on (an instance of) dataset D, which yields the following query:

```
SELECT l.name, v.pred_acc
FROM experiment e, learner_inst li, learner l, data_inst di,
dataset d, evaluation v
WHERE v.eid = e.eid and e.learner_inst = li.liid and li.lid = l.lid
and e.data_inst = di.diid and di.did = d.did and d.name='D'
```

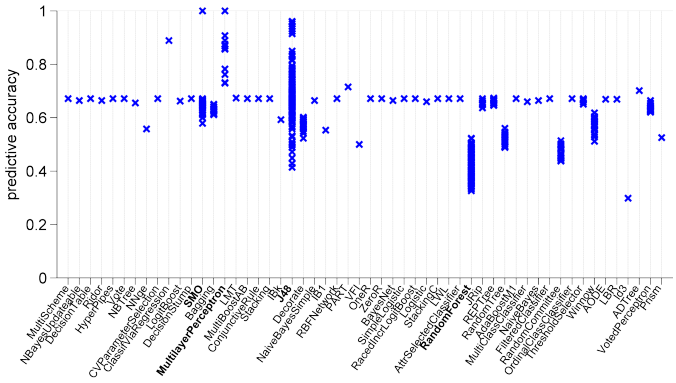
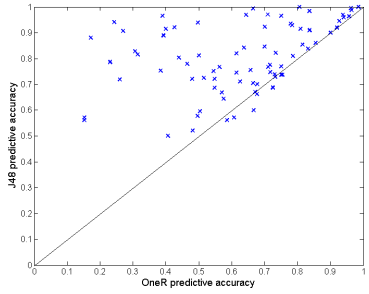


Fig. 2. Algorithm performance comparison on the `monks-problems-2_test` dataset.

We can now interpret the returned results, e.g. by drawing a scatterplot. For dataset `monks-problems-2` (a near-parity problem), this yields Fig. 2, giving a clear overview of how each algorithm performs and (for those algorithms whose parameters where varied) how much variance is caused by different parameter settings. Only a few algorithms surpass default accuracy (67%) and while some cover a wide spectrum (like J48), others jump to 100% accuracy for certain parameter settings (SMO with higher-order polynomial kernels and MultilayerPerceptron when enough hidden nodes are used).

We can also compare two algorithms A1 and A2 on *all* datasets by joining their performance results (with default settings) on each dataset, and plotting them against each other, as shown in Fig. 3. Moreover, querying also allows us to use aggregates and to order results, e.g. to directly build rankings of all algorithms by their *average* error over all datasets, using default parameters:

```
SELECT l.name, avg(v.mn_abs_err) AS avg_err
FROM experiment e, learner l, learner_inst li, evaluation v
WHERE v.eid = e.eid and e.learner_inst = li.liid and li.lid = l.lid
and li.default = true GROUP BY l.name ORDER BY avg_err asc
```




```
SELECT s1.name, avg(s1.pred_acc) AS
A1_acc, avg(s2.pred_acc) AS A2_acc
FROM (SELECT d.name, e.pred_acc FROM ..
WHERE l.name = 'A1' ... ) AS s1
JOIN (SELECT d.name, e.pred_acc FROM ..
WHERE l.name = 'A2' ... ) AS s2
ON s1.name = s2.name
GROUP BY s1.name
```

Fig. 3. Comparing relative performance of J48 and OneR with a single query.

Similar questions can be answered in the same vein. With small adjustments, we can query for the variance,... of each algorithm's error (over all or a single dataset), study how much error rankings differ from one dataset to another, or study how parameter optimization affects these rankings.

4.2 Querying for parameter effects

Previous queries generalized over all parameter settings. Yet, starting from our first query, we can easily study the effect of a specific parameter P by "zooming in" on the results of algorithm A (by adding this constraint) and selecting the value of P linked to (an instantiation of) A, yielding Fig. 4a:

```
SELECT v.pred_acc, lv.value
FROM experiment e, learner_inst li, learner l, data_inst di,
dataset d, evaluation v, learner_parameter lp, learner_parval lv
WHERE v.eid = e.eid and e.learner_inst = li.liid and li.lid = l.lid
and l.name='A' and lv.liid=li.liid and lv.pid = lp.pid and lp.
name='P' and e.data_inst = di.diid and di.did = d.did and d.name='D'
```

Sometimes the effect of a parameter P may be dependent on the value of another parameter. Such a parameter P2 can however be controlled (e.g. by demanding its value to be larger than V) by extending the previous query with a constraint requiring that the learner instances additionally are amongst those where parameter P2 obeys those constraints.

```
WHERE ... and lv.liid IN
(SELECT lv.liid FROM learner_parval lv, learner_parameter lp
WHERE lv.pid = lp.pid and lp.name='P2' and lv.value>V)
```

Launching and visualizing such queries yield results such as in Fig. 4, clearly showing the effect of the selected parameter and the variation caused by other parameters. As such, it is immediately obvious how general an observed trend is: all constraints are explicitly mentioned in the query.

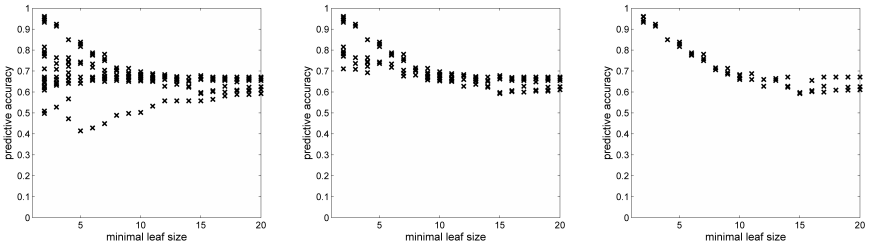


Fig. 4. The effect of the minimal leafsize of J48 on `monks-problems-2_test` (a), after requiring binary trees (b), and after also suppressing reduced error pruning (c)

4.3 Querying for the effect of dataset properties

It also becomes easy to investigate the interactions between data properties and learning algorithms. For instance, we can use our experiments to study the effect of a dataset's size on the performance of algorithm A[2]:

```
SELECT v.pred_acc, d.nr_examples
FROM experiment e, learner_inst li, learner l, data_inst di,
dataset d, evaluation v
WHERE v.eid = e.eid and e.learner_inst = li.liid and li.lid = l.lid
and l.name='A' and e.data_inst = di.diid and di.did = d.did
```

4.4 Applying data mining techniques to the experiment database

There can be very complex interactions between parameter settings, dataset characteristics and the resulting performance of learning algorithms. However, since a large number of experimental results are available for each algorithm, we can apply data mining algorithms to model those interactions.

For instance, to automatically learn which of J48's parameters have the greatest impact on its performance on `monks-problems-2_test` (see Fig. 4), we queried for the available parameter settings and corresponding results. We discretized the performance with thresholds on 67% (default accuracy) and 85%, and we used J48 to generate a (meta-)decision tree that, given the used parameter settings, predicts in which interval the accuracy lies. The resulting tree (with 97.3% accuracy) is shown in Fig. 5. It clearly shows which are the most important parameters to tune, and how they affect J48's performance.

Likewise, we can study for which dataset characteristics one algorithm greatly outperforms another. Starting from the query in Fig. 3, we additionally queried for a wide range of data characteristics and discretized the performance gain of J48 over 1R in three classes: "draw", "win_J48" (4% to 20% gain), and "large_win_J48" (20% to 70% gain). The tree returned by J48 on this meta-dataset is shown in Fig. 6, and clearly shows for which kinds of datasets J48 has a clear advantage over OneR.

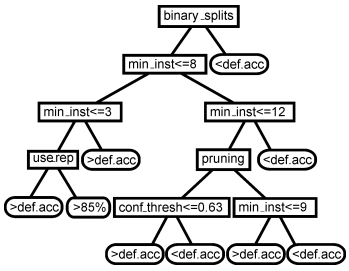


Fig. 5. Impact of parameter settings.

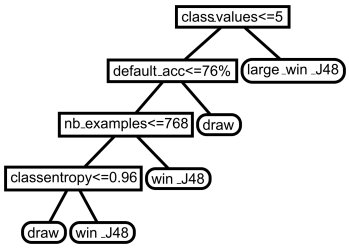


Fig. 6. Impact of dataset properties.

[2] To control the value of additional dataset properties, simply add these constraints to the list: `WHERE ... and d.nr_attributes>5`.

4.5 On User-Friendliness

The above SQL queries are relatively complicated. Part of this is however a consequence of the relatively complex structure of the database. A good user interface, including a graphical query tool and an integrated visualization tool, would greatly improve the usability of the database.

5 Conclusions

We have presented an experiment database for classification, providing a well-structured repository of fully described classification experiments, thus allowing them to be easily verified, reused and related to theoretical properties of algorithms and datasets. We show how easy it is to investigate a wide range of questions on the behavior of these learning algorithms by simply writing the right queries and interpreting the results, or by applying data mining algorithms to model more complex interactions. The database is available online and can be used to gain new insights into classifier learning and to validate and refine existing results. We believe this database and underlying software may become a valuable resource for research in classification and, more broadly, machine learning and data analysis.

Acknowledgements

We thank Anneleen Van Assche and Celine Vens for their useful comments and help building meta-decision trees and Anton Dries for implementing the dataset characterizations. Hendrik Blockeel is Postdoctoral Fellow of the Fund for Scientific Research - Flanders (Belgium) (FWO-Vlaanderen), and this research is further supported by GOA 2003/08 “Inductive Knowledge Bases”.

References

Blockeel, H. (2006): Experiment databases: A novel methodology for experimental research. *Lecture Notes in Computer Science, 3933, 72-85.*

Blockeel, H. and Vanschoren J. (2007): Experiment Databases: Towards an Improved Experimental Methodology in Machine Learning. *Lecture Notes in Computer Science, 4702, to appear.*

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

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

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

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