Fusion of Meta-knowledge and Meta-data for Case-Based Model Selection

Melanie Hilario and Alexandros Kalousis

CUI - University of Geneva, CH-1211 Geneva 4
{hilario,kalousis}@cui.unige.ch

Abstract. Meta-learning for model selection, as reported in the symbolic machine learning community, can be described as follows. First, it is cast as a purely data-driven predictive task. Second, it typically relies on a mapping of dataset characteristics to some measure of generalization performance (e.g., error). Third, it tends to ignore the role of algorithm parameters by relying mostly on default settings. This paper describes a case-based system for model selection which combines knowledge and data in selecting a (set of) algorithm(s) to recommend for a given task. The knowledge consists mainly of the similarity measures used to retrieve records of past learning experiences as well as profiles of learning algorithms incorporated into the conceptual meta-model. In addition to the usual dataset characteristics and error rates, the case base includes objects describing the evaluation strategy and the learner parameters used. These have two major roles: they ensure valid and meaningful comparisons between independently reported findings, and they facilitate replication of past experiments. Finally, the case-based meta-learner can be used not only as a predictive tool but also as an exploratory tool for gaining further insight into previously tested algorithms and datasets.

1 Issues and Objectives

Broadly speaking, the *model selection* problem concerns the choice of an appropriate model for a given learning task. However, the term has been used with varying nuances, or at least shifting emphases, among the different communities now involved in data mining. The divergence seems to concern the level of generality at which one situates the search for the appropriate model. Among statisticians, model selection takes place *within* a given model family: it typically refers to the task of creating a fully specified instance of that family, called the fitted model, whose complexity has been fine-tuned to the problem at hand [9]. This convention has been carried over to neural network (NN) learning, where model complexity is a function of parameters specific to a family of network architectures – e.g., the number of hidden layers and units for feedforward NNs or the number of centers for Radial Basis Function Networks [15]. In the symbolic machine learning (ML) community, the term *model selection* designates the task of selecting a learning algorithm for a specific application; thus search is conducted in the space of all known or available model families and the selection of a learning algorithm circumscibes a model class or family from which the learned model will eventually emerge. However, the task of finding the most appropriate model instantiation from the selected family has been relatively neglected. In other words, ML researchers tend to end where statisticians and NN researchers tend to start. Thus, while the latter seldom envisage model selection beyond the frontiers of a specific model family (e.g. multilayer perceptrons), the former often overlook the role of model parameters when doing cross-algorithm comparisons; the Statlog [12] and Metal projects [11], for instance, have adopted the expedient of systematically evaluating learning algorithms with their default parameter settings. In the broader perspective of data mining, however, these diverse definitions should be seen as partial and complementary perspectives on a complex multifaceted task. *The first research objective of this work is to integrate the choice of learning methods as well as model parameters in a unified framework for model selection.*

Given that no learning algorithm can systematically outperform all others [17], the model selection problem arises anew with each learning task. The most common approach consists in experimenting with a number of alternative methods and models and then selecting one which maximizes certain performance measures. However, with the increasing number and diversity of learning methods available, exhaustive experimentation is simply out of the question. There is a need to limit the initial set of candidate algorithms on the basis of the given task or data, and one can legitimately hope that algorithms which have proved useful for a certain class of tasks will confirm their utility on other, similar tasks. Hence the idea that by examining results of past learning experiences, one might determine broad mappings between learning methods and task classes so that model selection does not start from scratch each time. Meta-learning is an attempt at automating the realization of this idea.

The meta-learning approach to model selection has been typically based on characterizations of the application dataset. In Statlog, meta-learning is a classification task which maps dataset attributes to a binary target variable indicating the applicability or non-applicability of a learning algorithm. Data characteristics can be descriptive summary statistics (e.g., number of classes, number of instances, average skew of predictive variables) or information-theoretic measures (e.g., average joint entropy of predictive and target variables). There have since been a number of attempts to extend or refine the Statlog approach. In the Metal project, for instance, datasets have also been characterized by the error rates observed when they are fed into simple and efficient learning algorithms called landmarkers (e.g., Naive Bayes or linear discriminants) [13]. However, the basic idea underyling the Statlog approach remains intact – i.e., that meta-attributes describing the application task/dataset suffice to predict the applicability or the relative performance of the candidate learning algorithms.

To broaden the range of meta-level predictors, we propose algorithm profiling as a complement to dataset characterization in general and to landmarking in particular. Landmarking uses specially selected learners to uncover information about the nature of the learning task or data. By contrast, algorithm profiling uses specially designed datasets to deliver information about a learning algorithm

– its bias/variance profile, scalability, tolerance to noise, irrelevant variables or missing data. While landmarking attempts to describe a dataset in terms of the areas of expertise to which it belongs (as witnessed by the landmarkers which perform well on it), algorithm profiling strives to describe in concrete, quantitative terms what makes up the region of expertise of a learning algorithm. *The second objective is to complement dataset characterizations with algorithm profiles as predictors in the meta-learning process.*

There is a fundamental difference between dataset and algorithm characterizations behind their apparent symmetry. Dataset characteristics are *meta-data* extracted from individual datasets whereas algorithm profiles embody *meta-knowledge* about learning algorithms which can be brought to bear on model selection over different datasets. The essential difference lies in the fact that algorithm characteristics have been derived via a process of abstraction and/or generalization. This generalized knowledge may be borrowed from the collective store of expertise in the domain or alternatively abstracted via controlled experimentation, as in the case of meta-attributes concerning sensitivity to missing or irrelevant data. In addition to prior meta-knowledge about learning/modelling tools, a domain expert's background knowledge of her application domain can be expressed in the form of constraints that should be taken into account in the search for an appropriate tool. *The third objective is to strike an effective and efficient balance between meta-learning and the use of prior (base- and meta-level) knowledge in the model selection process.*

With the introduction of prior knowledge about learning algorithms and application domains, meta-learning becomes a multi-relational task which calls for greater expressive power than that offered by attribute value vectors. In this paper we describe an object-oriented case-based meta-learning assistant which addresses the issues and objectives described above. Section 2 describes the knowledge embedded in the system's underlying conceptual (meta) model. Section 3 describes the current implementation – the extensive case base gathered to date as well as the different ways in which it can be exploited – and proposes a strategy for evaluating its incremental meta-learning capabilities. Section 4 summarizes and argues for its possible utility as a long-term meta-memory of machine learning experiments.

2 The Embedded Knowledge

This section focuses on the knowledge embedded in the meta-model of the learning process. A simplified view of the conceptual schema is shown in Fig. 1.

2.1 Modelling Processes

The core of the meta-model is the *modelling process*, which depicts a specific learning episode or experiment. It is described by a number of performance measures such as the error rate, the training/testing time, and the size of the learned model or hypothesis. More importantly, the ModProcess object is the hub

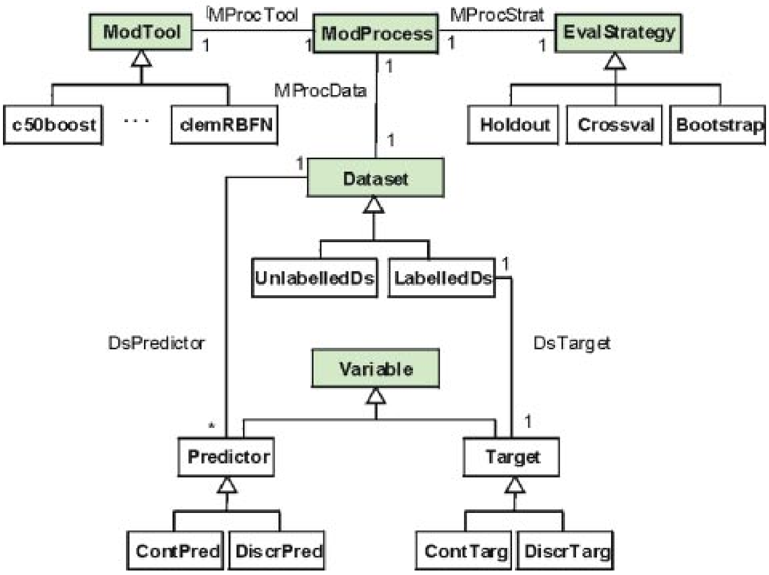

Fig. 1. The conceptual metamodel (see Sect. 2 for explanation)

which links together three main components according to a precise semantics: a modelling or learning tool (ModTool) is trained and tested on the given data (Dataset) following a particular evaluation strategy (EvalStrat). The structure and attributes of these three components comprise the background information which will be brought to bear in the meta-learning and model selection process.

2.2 Datasets

The object depicting a dataset can be seen as a simple, albeit extended, importation of the Statlog dataset characteristics. These will not be described here (see [12] for a detailed discussion); rather, we present several major extensions which have been made possible by the structured representation adopted. In the Statlog formalism, only summary statistics over all predictive variables of a dataset could be recorded; the consequence was that atypical variables (e.g., irrelevant variables) were impossible to detect since the symptoms of this atypicality (e.g., an extremely low measure of mutual information with the target variable) somehow got dissolved in the overall statistics (e.g., average mutual information). Such problems disappear in a multi-relational representation, where a dataset can be characterized more thoroughly by the collection of objects that describe its component variables.

In addition, both variables and datasets can be divided into subclasses and thus described only by features that are certain to make sense for the specific category in question. One persistent problem of meta-learning in Statlog and other propositional approaches is that different types of datasets are forced into a single attribute vector, with the result that many meta-attributes have missing values when they turn out to be non applicable to a certain data type. For instance, summary statistics of continuous variables (eg mean skewness, departure from normality) are not applicable to categorical variables; also, information theoretic measures are often not computed for continuous variables. The result is that comparisons between symbolic, numeric, and mixed datasets become highly problematical. This difficulty is circumvented quite naturally in a typed multi-relational setting. Variables can be divided into subclasses along two dimensions. According to their role in the dataset, they are either predictors or targets; according to the data type, they are either continuous or discrete. Similarly, datasets may be symbolic, numeric, or mixed, depending on the types of their component variables. Along a different dimension, datasets are either labelled (e.g., for classification) or unlabelled (e.g., for association). Labelled datasets contain a number of attributes such as average joint entropy or average mutual information which make sense only for supervised learning tasks.

2.3 Modelling Tools

The ModTool class subsumes any fully implemented modelling or learning method which can be used for supervised or unsupervised knowledge discovery from data. Each tool is formalized as a ModTool sublcass. The tools used in our initial study are C5.0 in its tree (c50tree) and rule (c50rule) versions, an oblique decision tree (ltree) [4], a sequential covering rule inducer (ripper) [3], the MLC++ implementations of Naive Bayes (mlcNB) and IB1 (mlcIB1) [7], Joao Gama's implementation of Fisher's linear discriminant (lindiscr), and the Clementine implementations of radial basis function networks (clemRBFN) and backpropagation in multilayer perceptrons (clemMLP)[1]. All learning tools inherit the meta-features defined for the ModTool class; in addition, each subclass has its specific attributes corresponding to the tool's user-tunable model and search parameters – e.g., pruning severity for (c50tree), the number of centers and their overlap for (clemRBFN). Each application of a learning tool is recorded as an instance of the corresponding subclass, and the actual parameters override the default values predefined in the conceptual schema.

There are no intermediate subhierarchies of modelling tools according to learning paradigm or computational approach. This design option has been taken deliberately: since the aim of our meta-learner is to discover mappings of data and task types onto classes of learners, we have avoided a priori classifications that may hinder the discovery of novel or unexpected affinities or clusters among learning methods. On the other hand, we have tried to embed as much knowledge as possible about the biases, strengths and weaknesses of each learning tool.

Representation and Approach. The simplest form of knowledge concerns the basic requirements, capabilities and limitations of each tool, which have

[1] The parenthesized names will be used throughout the rest of this paper to identify the specific implementations of the learning algorithms studied.

Table 1. Characterizing representation and approach of modeling tools

ModTool	Data	Inc	CH	VH	Par	Meth	Strat	Cum
c50rules	NS	N	Y	Seq	Sym	Logic	E	L
c50tree	NS	N	Y	Seq	Sym	Logic	E	L
clemMLP	NS	N	N	Par	NN	Thresh	E	M
clemRBFN	NS	N	N	Par	NN	Comp	E	M
lindiscr	NS	N	N	Par	Stat	Thresh	E	H
ltree	NS	N	N	SP	Sym	Logic	E	M
mlcib1	NS	Y	N	Par	Sym	Comp	L	M
mlcnb	NS	N	N	Par	Sym	Comp	E	H
ripper	NS	N	N	Seq	Sym	Logic	E	L

generally been gathered from algorithm specifications or instruction manuals of the implemented tool. Attributes in this group indicate the type of data (Data in Table 1) supported by a learning tool (N for numeric/continuous, S for symbolic/discrete, NS for both), whether the tool learns incrementally or not (Inc), or whether it can handle externally assigned costs (CH). These characteristics can be determined in a straightforward manner and they are usually invariant for all instantiations of a given learning tool.

As tool specifications provide only a minimal characterization of learner functionality, knowledge of less obvious features (see last five columns of Table 1) has been gathered mainly from cumulative results of past research. We borrowed the paradigm-based categorization of learning algorithms (Par) as symbolic, statistical, or connectionist as well as Langley's distinction between logical, competitive, and threshold-based learning approaches (Meth). From the point of view of learning strategy (Strat), modelling tools are either lazy or eager, depending on whether they simply store data, deferring learning until task execution time, or use given data to create a model in view of future task requests. Another dimension is the way a learner handles input variables in the generalization process (VH). Sequential algorithms (e.g., decision trees) examine one input variable at a time, while parallel algorithms examine all input variables simultaneously [14]. Neural networks are clearly parallel; so are instance-based learners and Naive Bayes classifiers, which aggregate distances or probabilities over all variable values simultaneously. A third, hybrid category includes algorithms such as oblique decision trees which alternate between sequential and simultaneous processing of variables depending on the data subset examined at a node.

An additional aspect of learning bias which has been brought to light by recent research is what Blockeel [2] calls cumulativity (Cum). This is a generalization of the statistical concept of additivity: two features are cumulative if their effects are mutually independent, so that their combined effect is the trivial composition of their separate effects. The cumulativity of learning algorithms is nothing more than their ability to handle cumulativity of features, which can be discretized roughly on a three-step scale. Linear regressors and discriminants are naturally situated on the high end of the scale; so is Naive Bayes with its assumption of the class-conditional independence of predictors (the product of

Table 2. Characterizing resilience of modeling tools

ModTool	Var	ErrSIrr	TimeSIrr	MCAR	MAR
c50rules	0.4503	0.0324	4.76	0.2475	0.2527
c50tree	0.4548	0.0337	4.26	0.2098	0.2094
clemMLP	0.4230	0.1292	378.25	?	?
clemRBFN	0.3626	0.0910	9595.93	?	?
lindiscr	0.2308	0.0351	4.92	0.1913	0.1624
ltree	0.4154	0.0413	14.57	0.1111	0.1251
mlcib1	0.3868	0.1347	56.98	0.2283	0.2292
mlcnb	0.2273	0	3.64	0.1073	0.0940
ripper	0.3862	0.0173	98.67	0.1310	0.1444

likelihoods and priors it maximizes translates directly to addivity of their respective logarithms). At the other extreme, sequential learners like decision trees and rule induction systems allow for maximal interaction between variables and therefore have low cumulativity. Instance-based learners and neural networks occupy the midpoint of the scale. Neural nets handle cumulativity by means of linear combinations while handling interaction by superposing multiple layers and using nonlinear threshold functions.

Resilience. A second group of characteristics concerns the resilience of a modelling tool, i.e., its capability of ensuring reliable performance despite variations in training conditions and especially in the training data. Resiliency characteristics reflect the sensitivity or tolerance of an algorithm to data characteristics or pathologies that are liable to affect performance adversely. Examples are stability, scalability, and resistance to noise, missing values, and irrelevant or redundant features. Contrary to representational and methodological meta-attributes, there is no consensus regarding the resilience of the ten learning tools included in our initial knowledge base. We thus undertook extensive experimental studies concerning their bias/variance trade-off and their sensivity to missing values and irrelevant features.

Table 2 shows the results of these experiments. The column labelled Var gives the proportion of variance in the generalization error. This was obtained by applying Kohavi and Wolpert's bias-variance decomposition method for zero-one loss [8] to each tool, averaged over 40 datasets from the UCI Machine Learning Repository. Note that the variance measures given here concern each tool as applied with its default parameter settings. For instance, 0.4548 is the mean variance observed for Clementine RBF networks with the default number of hidden units, i.e., 20; variances have also been recorded as the number of hidden units is varied from 5 to 150. The next two columns quantfy a learner's sensitivity to irrelevant attributes as measured by its mean increase in generalization error (ErrSIrr) or in training time (TimeSIrr) for each additional percent of irrelevant attributes. These measures were obtained in a series of 10-fold cross-validation experiments on 43 UCI datasets; each learning algorithm was run with default parameters on the original datasets, then on 6 corrupted versions containing respectively 5%, 10%, 20%, 30%, 40%, and 50% irrelevant features. A full discussion of these experiments and the results is given in [5]. The last two columns depict mean increase in generalization error with each per cent increase in missing values – either values that are missing completely at random (MCAR) or missing at random (MAR). A value is said to be missing completely at random if its absence is completely independent of the dataset; it is missing at random if its absence depends, not on its own value, but on the value of some other attribute in the dataset [10]. Here again, we followed a strategy of "progressive corruption" to observe how learners cope with incomplete data. Generalization error was estimated for each learner on the original datasets, then on five increasingly incomplete versions from which 5%, 10%, 20%, 30%, and 40% of the feature values were deleted. For the MCAR series, feature values were deleted randomly following a uniform distribution; for the MAR series, values of selected features were deleted conditional on the values of other attributes. The interested reader is referred to [6] for details.

It should be stressed that all characteristics as well as any conclusions drawn about a modelling tool concern *the specific software implementation under study rather than the generic learning algorithm or method*. It is well known that the specification of an algorithm leaves considerable flexibility in implementation, and differences between implementations of the same algorithm can turn out to be as significant as differences between distinct methods or algorithms. Thus, while we have taken pains to include a wide variety of learning approaches in our study, the findings reported should not be extrapolated from the individual tool implementation to the generic method without utmost precaution.

Practicality. Finally, other features concern more practical issues of usability. They do not impact a learner's generalization performance but can be used to pre-select tools on the basis of user preferences. Examples of such characteristics are the comprehensibility of the method, the interpretability of the learned model, or the degree to which model and search parameters are handled automatically. Since values of these meta-characteristics are qualitative and highly subjective, we assigned 5-level ordinal values on what we deemed to be intuitively obvious grounds. For instance, parameter handling is rated very high for lindiscr, ltree, mlcib1, and mlcnb – tools which require absolutely no user-tuned parameter. When an algorithm involves user-tunable parameters, its rating depends on how well the algorithm performs without user intervention, that, when run with its default parameter settings. Thus c50tree and c50rules are marked high, clemMLP medium, and clemRBFN low (the default of 20 centers often leads to poor performance and even to downright failure to converge). As for method comprehensibility and result interpretability, we relied heavily on the Statlog characterization, since these are among the few characteristics that are intrinsic to the method and vary little across implementations. For instance, the neural network tools rate very low on both comprehensibility and interpretability; decision trees and rules rate high on both counts whereas for mlcib1, the method (learning by similarity) is easier to grasp than the 'model' (distance measures of nearest neighbors).

2.4 Evaluation Strategies

To ensure that all recorded learning episodes conform to the elementary rules of tool evaluation, each modelling process is associated with a fully specified evaluation strategy. Examples of generic strategies are simple holdout, cross-validation, bootstrap, and subsampling without replacement. Each has its own particular set of parameters: the proportion of the training set for holdout, the number of folds for cross-validation, or the number of replicates for bootstrap. Attributes common to all strategies are the number of trials (complete runs of the selected strategy) and the random seed used. Such detailed accounts have a two-fold motivation. First, we all know that the same method applied to the same dataset can lead to widely different performance measures, depending on whether these were estimated using simple holdout or leave-one-out cross-validation. Many of the cross-experimental comparisons reported in the literature deserve little credence for lack of evidence that performance measures were obtained under identical or at least comparable learning conditions. Secondly, information about the evaluation strategy followed in an experiment should be sufficiently precise and complete to allow for replication and take-up by other researchers.

3 The Implementation

3.1 The Case Base

The conceptual schema described in the preceding section has been implemented using CBR-Works Professional. Given the sheer volume of the collected metadata, interactive data entry was out of the question. A set of scripts was implemented to automate the translation of data characterization files as well as results of learning experiments into CBR-Works' Case Query Language. The current case base contains objects representing:

– more than 1350 datasets for classifications tasks (98 UCI and other benchmarks plus semi-artificial datasets generated from these for irrelevance and missing values experiments)
– around 37500 variables belonging to the above datasets
– around 11700 experiments involving the training and evaluation of 9 classification algorithms on the above datasets.

3.2 Exploitation Scenarios

The basic scenario follows the standard CBR cycle consisting of the four R's: retrieve, reuse, revise, retain [1]. A query case is entered in the form of a ModProcess object whose minimal content is a set of links to a three objects representing a dataset, a modelling tool, and an evaluation strategy. While the last two can be left completely unspecified, the application dataset should be fully characterized in the query case's Dataset object, itself linked to objects describing its component variables. Optionally, users can fill out slots of all the other objects of the query case in order to specify a set of constraints based on the nature of the application task or their own preferences. For instance, they can impose preferences on the incrementality, comprehensibility, or stability of a modelling tool by filling out the relevant slot of the ModTool object. In such cases, learner characteristics serve in prior model selection by restricting from the outset the space of tools to consider. Users can also use the ModProcess object to specify what they consider acceptable performance (e.g., a lower bound on the accuracy gain ratio or an upper bound on the error rate or the time to be spent in training or testing). The system retrieves k most similar cases (where k is a user-specified parameter), each of which can be taken as a combined recommendation for a modelling tool, its parameter settings, and an evaluation strategy. The user runs the recommended algorithms and the results are integrated into the case memory as k new cases and their associated objects. This basic scenario illustrates the use of standard CBR to incrementally improve model selection by learning from experience.

For case retrieval to work properly, additional domain-specific knowledge has been embedded in the similarity measures. While standard symmetric criteria work well with boolean or categorical meta-attributes such as learner incrementality, method, or cost handling ability, ordered features usually call for asymmetrical similarity criteria. Ordinal (including real) values specified in a query case are often meant as lower or upper bounds on the corresponding attribute. For instance, a user who requires medium interpretability of learned models would be even happier with high or very high interpretability. Similarly, an error specification of 0.2 in the query case should be taken to mean the highest acceptable error, with errors <0.2 getting proportionally higher similarity scores as they approach 0. On the contrary, an accuracy gain ratio of 0.1 should be interpreted as a minimum, with higher values becoming progressively more "similar " as they approach 1.

The reverse exploitation scenario illustrates the use of the system as an exploratory workbench which the user (who may happen to be a KDD researcher) can use to gain insights about learning tools of interest. In problem-solving cum learning mode, the goal is: Given this dataset, which modelling tool(s) should I use to get best results? In exploratory mode, the goal can be stated thus: Given this modelling tool, for which class(es) of tasks/data is it most appropriate? To chart the region of expertise of a learning algorithm, the user enters a query case consisting mainly of the learning algorithm and a bound on some performance measure (e.g., an error rate <learning algorithm's region of competence as defined by the performance criteria used.

3.3 Validating the System

We have described an initial implementation of a case-based assistant which recommends modelling tools for a given learning task. It provides decision support by incorporating meta-knowledge of the model selection problem into its basic learning mechanism. The goal is to develop a workbench for incremental meta-learning which is on the agenda for the third year of the Metal project.

To validate the system we need to set a baseline, i.e., measure the performance of the system with its initial knowledge and case base, and then evaluate its ability to improve performance with experience. We propose the following experimental setup: First, divide the exising meta-dataset into 3 roughly equal subsets (around 4500 learning episodes each) and prime the learner with subset 1. Second, use subset 2 to "grow " the system. Enter each case without the target performance measure (e.g., accuracy gain ratio) as a query, then compare system recommendations with known results and add the query case to the growing base. After addition of a fixed number of cases (e.g., 200), test the case base on subset 3 in view of plotting performance variation with experience.

4 Summary

We presented a case-based assistant for model selection which combines three major features. First, it combines knowledge of learning algorithms with dataset characteristics in order to facilitate model selection by focusing on the most promising tools on the basis of user specified constraints and preferences. Second, it incorporates a mechanism for distinguishing between different parameterizations of learning tools, thus extending model selection to cover both the choice of the learning algorithm and its specific parameter settings. Third, it integrates meta-data gathered from different learning experiences with meta-knowledge not only of learning algorithms but also of modelling processes, performance metrics and evaluation strategies. We are aware of no other system that has all these features simultaneously. As pointed out in the introduction, mainstream meta-learning for model selection has focused mainly on dataset characteristics as predictors of the appropriateness of candidate learning algorithms. Todorowski [16] has tried to go beyond summary dataset statistics and examine characteristics of the individual variables in the data in order to learn first-order model selection rules. However he does not incorporate knowledge of learning tools or their parameters.

We believe that the proposed system is not just useful for meta-learning but can also evolve into some kind of meta-memory of machine learning research. There is a need for a system that manages and maintains meta-knowledge induced from experimentation together with information about the experiments themselves. First, such a long-term memory would allow reliable cross-experimental comparisons. It is common practice among machine learning researchers to compare new observations on tool performance and efficiency with past findings; however, in the absence of clear indications concerning experimental conditions (parameter settings of the learning tools, evaluation strategies used, training and test sample sizes, statistical significance of results, etc.), there is no guarantee that the measures being compared are indeed comparable. Second, it would avoid redundant effort, as researchers pursuing an idea or hypothesis could first consult the store of accumulated knowledge before designing new experiments.

Acknowledgements. We are very grateful to Maria Malek, ESTI (France), for several fruitful discussions on the subject of CBR modelling. We also thank Johann Petrak of OFAI (Vienna) for his most useful experimentation scripts. The work reported in this paper was partly supported by a grant from the Swiss OFES in the framework of ESPRIT LTR Project METAL.

References

1. A. Agnar and E. Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications,* 7(1), 1994.
2. H. Blockeel. Cumulativity as inductive bias. In *Data Mining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions,* pages 61–70, Lyon, France, July 2000.
3. W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proc. of the 11th International Conference on Machine Learning,* pages 115–123, Tahoe City, CA, 1995. Morgan Kaufmann.
4. J. Gama and P. Brazdil. Linear tree. *Intelligent Data Analysis,* 3:1–22, 1999.
5. M. Hilario and A. Kalousis. Quantifying the resilience of inductive classification algorithms. In *Principles of Data Mining and Knowledge Discovery. Proceedings of the 4th European Conference,* pages 106–115, Lyon, France, 2000. Springer-Verlag.
6. A. Kalousis and M. Hilario. Supervised knowledge discovery from incomplete data. In *International Conference on Data Mining,* Cambridge, UK, 2000.
7. R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A machine learning library in C++. Technical report, CSD, Stanford University, August 1994. An abridged version of this report appears in AI'94: Tools in AI.
8. R. Kohavi and D. Wolpert. Bias plus variance decomposition for zero-one loss functions. In L. Saitta, editor, *Proc. of the 13th International Conference on Machine Learning,* pages 275–283, Bari (Italy), 1996. Morgan Kaufmann.
9. H. Linhart and W. Zucchini. *Model Selection.* J. Wiley, NY, 1986.
10. R. J. Little and D. B. Rubin. *Statistical Analysis with Missing Data.* Wiley, 1987.
11. MetaL Consortium. Project Homepage. `http://www.metal-kdd.org/`.
12. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine learning, neural and statistical classification.* Ellis-Horwood, 1994.
13. Bernhard Pfahringer, Hilan Bensusan, and Christophe Giraud-Carrier. Metalearning by landmarking various learning algorithms. In *Proc. Seventeenth International Conference on Machine Learning, ICML'2000,* pages 743–750, San Francisco, California, June 2000. Morgan Kaufmann.
14. J. R. Quinlan. Comparing connectionist and symbolic learning methods. In S. J. Hanson, G. A. Drastal, and R. L. Rivest, editors, *Computational Learning Theory and Natural Learning Systems,* volume I, chapter 15, pages 446–456. MIT Press, 1994.
15. B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge U. Press, 1996.
16. L. Todorowski. Experiments in meta-level learning with ILP. In *International Workshop on Inductive Logic Programming,* Bled, Slovenia, 1999. Springer-Verlag.
17. D. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation,* 8(7):1381–1390, 1996.