THE 20TH EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE (ECAI 2012)

ECAI 2012
20TH EUROPEAN CONFERENCE ON ARTIFICIAL INTELLIGENCE
MONTPELLIER, FRANCE
AUGUST 27-31, 2012
CONFERENCE CHAIR: DIDIER DUBOIS
PROGRAM CHAIR: LUC DE RAEDT
ORGANIZING COMMITTEE CHAIR: CHRISTIAN BESSIERE
www.lirmm.fr/ecai2012

5th PLANNING TO LEARN WORKSHOP WS28 AT ECAI 2012

PlanLearn-2012

August 28, 2012

Montpellier, France

Edited by

Joaquin Vanschoren, Pavel Brazdil, Jörg-Uwe Kietz

Preface

We are living in an age of great opportunity. There is an ever-rising flood of digital data from many sources, from tweets, photographs and social network communications to high-resolution sensor data across nearly all sciences. The great opportunity, and also the challenge, lies in making order out of chaos and deliver more usable information.

Not that there is a shortage of great data analysis tools. The fields of Knowledge Discovery in Databases (KDD) and Machine Learning (ML) have reached a mature stage, offering plenty of techniques to solve complex data analysis tasks on all types of data. The problem is that, with the increased sophistication of data analysis techniques, there is a greater than ever need for clear guidance on how to use these techniques.

Indeed, data needs to be analysed in often non-trivial refinement processes, which require technical expertise about methods and algorithms, skill in successful workflow design, experience with how a precise analysis should proceed, and knowledge about an exploding number of analytic approaches.

The underlying idea of this workshop is that constructing such workflows could be greatly facilitated by leveraging machine learning techniques. In other words, can't we *learn*, based on examples, how to select algorithms and build workflows? Research over the last decades has shown that this is in fact an inherently multidisciplinary problem, requiring an multidisciplinary solution. We can distinguish the following themes that have emerged in the literature:

Planning, from the field of Artificial Intelligence, provides a principled approach to generate and build workflows. Indeed, building a knowledge discovery workflow can be cast as a planning problem in which (sub)tasks such as data preprocessing and modelling must be achieved by devising an optimal sequence of operators.

Ontologies, from the field of Information Science, allow us to express knowledge about data, algorithms and model in such a way that systems can interpret and reason about these concepts. It yields precise descriptions of data format requirements so that workflows don't break, and express the inner working of algorithms so we can reason about how to tune them.

Meta-learning, from the field of Machine Learning, allows us to build models about algorithm behavior under various conditions, and thus to predict which (combinations of) techniques are most useful. Meta-learning approaches have grown ever more sophisticated, building on ontological information about algorithms, large databases of machine learning experiments, and domain-specific data features.

It is interesting to see how these fields interact: ontologies lead to better meta-learning techniques, which in turn allow for more accurate workflow planning, in turn providing more examples from which to learn. This workshop, therefore, is particularly aimed at exploring the possibilities of integrating these fields. It offers a forum for exchanging ideas and experience concerning the state-of-the-art from these different areas and outline new directions for research.

These proceedings include 7 contributions on the latest advances in this area, many of them integrating planning, ontologies and meta-learning as discussed above. Moreover, they include several demonstrations of working systems in this area, which will be of great use to end users. We thank everybody for their sincere interest and their contributions, and especially thank our invited speakers:

Ross King (Manchester University): *The potential of automated workflow planning in the Robot Scientist.*
Filip Zelezny (Czech Technical University): *Planning to learn: Recent developments & future directions.*

We hope you will find it an interesting and inspiring workshop, leading to great new collaborations.

Leiden, August 2012

Joaquin Vanschoren
Pavel Brazdil
Jörg-Uwe Kietz

Main areas covered by the workshop

Of particular interest are methods and proposals that address the following issues:

- Planning to construct workflows
- Exploitation of ontologies of tasks and methods
- Representation of learning goals and states in learning
- Control and coordination of learning processes
- Experimentation and evaluation of learning processes
- Recovering / adapting sequences of DM operations
- Meta-learning and exploitation of meta-knowledge
- Layered learning
- Multi-task learning
- Transfer learning
- Active learning
- Multi-predicate learning (and other relevant ILP methods)
- Learning to learn
- Learning to plan
- Intelligent design and learning

A Brief History

PlanLearn-2012 is the 5th workshop in the series, which has been collocated with a range of conferences:

- PlanLearn-2007 was associated with ECML/PKDD-07 in Warsaw, Poland. The invited speaker was Larry Hunter (Univ. of Colorado at Denver and Health Sciences Center) who presented a talk entitled *Historical Overview of the area Planning to Learn.*
- PlanLearn-2008 was associated with ICML/COLT/UAI in Helsinki, Finland. The invited speaker was Raymond J. Mooney (University of Texas at Austin) who presented a talk on *Transfer Learning by Mapping and Revising Relational Knowledge.*
- PlanLearn-2010 was associated with ECAI in Helsinki, Finland. Invited speakers were Michele Sebag (LRI, U. Paris-Sud) on *Monte-Carlo Tree Search: From Playing Go to Feature Selection*, and Luc de Raedt (Katholieke Universiteit Leuven) on *Constraint Programming for Data Mining*.
- PlanSoKD-2011 was held in conjunction with the Service-Oriented Knowledge Discovery workshop at ECML/PKDD in Athens, Greece.

More information on the workshop and previous editions can be found at:
`http://datamining.liacs.nl/planlearn.html`

Acknowledgements

Many thanks to Rui Leite for typesetting these proceedings.

Workshop Organization

Workshop Chairs

Joaquin Vanschoren (LIACS, Leiden University)
Pavel Brazdil (LIAAD-INESC Porto L.A. / FEP, University of Porto)
Jörg-Uwe Kietz (Department of Informatics, University of Zürich)

ECAI Workshop Chair

Jérôme Lang and Michèle Sebag

Workshop Program Committee

Michael Berthold, U. Konstanz, Germany, and KNIME
Pavel Brazdil, LIAAD-INESC Porto L.A. / FEP, University of Porto, Portugal
André Carlos Ponce de Carvalho, USP, Brasil
Claudia Diamantini, Università Politecnica delle Marche, Italy
Johannes Fuernkranz, TU Darmstadt, Germany
Christophe Giraud-Carrier, Brigham Young University, USA
Krzysztof Grabczewski, Nicolaus Copernicus University, Poland
Melanie Hilario, U Geneva, Zwitserland
Norbert Jankowski, Nicolaus Copernicus University, Poland
Alexandros Kalousis, U Geneva, Zwitserland
Pance Panov, Jozef Stefan Institute, Slovenia
Bernhard Pfahringer, U Waikato, New Zealand
Vid Podpecan, Jozef Stefan Institute, Slovenia
Ricardo Prudêncio, Universidade Federal de Pernambuco Recife (PE), Brasil
Stefan Rüping, FhG-IAIS, Germany
Floarea Serban, University of Zurich, Switzerland
Carlos Soares, FEP, University of Porto, Portugal
Maarten van Someren, University of Amsterdam, The Netherlands
Vojtech Svatek, Univ. of Economics, Prague, Czech Republic
Ljupco Todorovski, Jozef Stefan Institute, Slovenia
Joaquin Vanschoren, U. Leiden / KU Leuven
Michel Verleysen, UC Louvain, Belgium
Ricardo Vilalta, University of Houston, USA
Filip Zelezny, Czech Technical University, Czech Republic

Technical support

Management of EasyChair: Joaquin Vanschoren, Pavel Brazdil and Jörg-Uwe Kietz

Table of Contents

Combining Meta-Learning and Optimization Algorithms for Parameter Selection

T. Gomes and **P. Miranda** and **R. Prudêncio**[1] and **C. Soares**[2] and **A. Carvalho**[3]

Abstract. In this article we investigate the combination of meta-learning and optimization algorithms for parameter selection. We discuss our general proposal as well as present the recent developments and experiments performed using Support Vector Machines (SVMs). Meta-learning was combined to single and multi-objective optimization techniques to select SVM parameters. The hybrid methods derived from the proposal presented better results on predictive accuracy than the use of traditional optimization techniques.

1 Introduction

The induction of a machine learning model with a good predictive accuracy to solve a learning problem is influenced by a variety of aspects, such as data pre-preprocessing, algorithm selection, parameter optimization and training procedure. The study presented in this paper focuses on a specific and relevant step of modeling: *parameter selection*. Once a learning algorithm is chosen, the user has to define its parameter values. Learning performance is usually affected by a poor selection of these values. For instance, the performance of SVMs depends on the adequate choice of its kernel function, kernel parameters, regularization constant, among other aspects [2].

Parameter selection is treated by many authors as an optimization problem in which a search technique is employed to find the configuration of parameters which maximizes the learning performance estimated on the problem at hand. There is an extensive literature applying optimization algorithms for parameter selection, especially for Artificial Neural Networks. Although it represents a systematic approach to parameter selection, this approach can be very expensive, since a large number of candidate parameter configurations must be evaluated to ensure that an optimal, or at least reasonably good, set of parameters is found [6].

Meta-learning, originally proposed for algorithm selection, has also been adapted to parameter selection (e.g., for SVM [6, 1]). In this approach, the choice of parameter values for a given task is based on parameter values sucessfully adopted in similar problems. Each *meta-example* in this solution includes: (1) a set of characteristics (called *meta-features*) describing a learning problem; and (2) the best configuration of parameters (among a set of candidates) tested on that problem. A *meta-learner* is used to acquire knowledge from a set of such meta-examples in order to recommend (predict) adequate parameters for new problems based on their meta-features.

Compared to the optimization approach, meta-learning tends to be computationally more efficient, at least at the moment when the recommendation of parameters is made. It must be observed that meta-learning however is very dependent on the quality of its meta-examples. It is usually difficult obtaining good results since meta-features are in general very noisy and the number of problems available for meta-example generation is commonly limited.

As discussed in [4], good solutions to a particular search problem can be used to indicate promising regions of the search space for similar problems. Related ideas have been applied to improve optimization tasks but in very different contexts (e.g. job shop scheduling [4]). The positive results in these contexts motivated us to apply similar ideas for optimizing learning parameters. Here, we present the combination of optimization techniques and meta-learning for the problem of parameter selection. Meta-learning is used to suggest an initial set of solutions, which are then refined by a search technique. In previous work, the search process starts by evaluating random solutions from the parameter space. In the proposed hybrid approach, the search process starts with successful solutions from previous similar problems. Hence, we expect that meta-learning guides the search directly to promising regions of the search space, thus speeding up the convergence to good solutions.

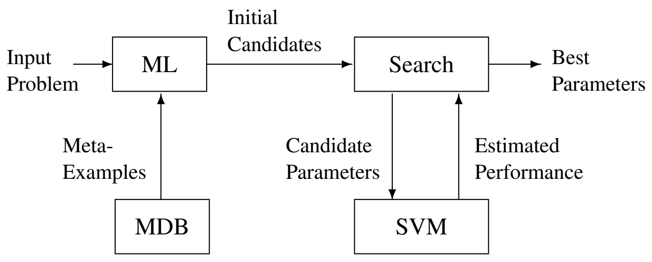


Figure 1. General Architecture

2 Developed Research

Figure 1 shows the general architecture of the proposed solution. Initially, the Meta-Learner (ML) module retrieves a predefined number of past meta-examples stored in a Meta-Database (MDB), selected on the basis of their similarity to the input problem. Next, the Search module adopts as initial search points the configurations of successful parameter values on the retrieved meta-examples. In the Search module, a search process iteratively generates new candidate values for the SVM parameters. The final solution which is recommended by the system is the best one generated by the Search module up to its convergence or other stopping criteria.

[1] Universidade Federal de Pernambuco, Brazil, email: {tafg,pbcm,rbcp}@cin.ufpe.br
[2] Universidade do Porto, Portugal, email: csoares@fep.up.pt
[3] Universidade de São Paulo, São Carlos, Brazil, email: andre@icmc.usp.br

In [3], we performed experiments that evaluated the proposed hybrid method using Particle Swarm Optimization (PSO) in the Search module. The system was empirically tested on the selection of two parameters for SVMs on regression problems: the γ parameter of the RBF kernel and the regularization constant C, which may have a strong influence in SVM performance. A database of 40 meta-examples was produced from the evaluation of a set of 399 configurations of (γ, C) on 40 different regression problems. Each meta-example refers to a single regression problem, which was described in our work by 17 meta-features (see [3] for details).

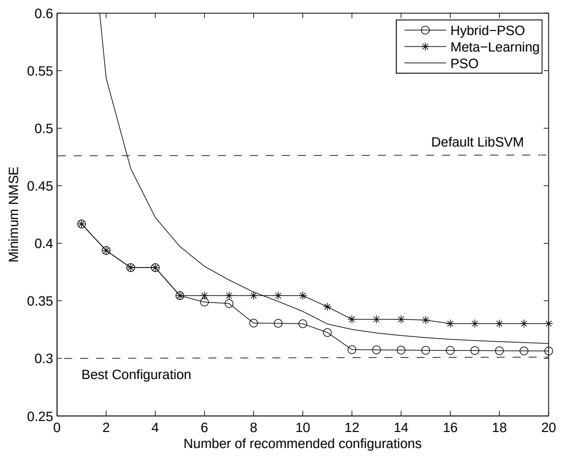


Figure 2. NMSE result obtained at each recommended configuration

Figure 2 compares the minimum NMSE (averaged over the 40 problems) obtained by SVM using the parameters suggested by combining meta-learning and PSO, referred to as Hybrid-PSO (using 5 initial solutions recommended by meta-learning), and the two methods individually, PSO (with random initialization, population size = 5) and meta-learning (which recommends the best configuration of each retrieved meta-example). We also present in Figure 2 the average NMSE achieved by the default heuristic adopted by the LibSVM tool (γ = inverse of the number of attributes and C=1). Finally, Figure 2 shows the average NMSE that would be achieved if the best parameter configuration had been chosen on each problem.

By comparing PSO and meta-learning, we identified a trade-off in their relative performances. Meta-learning is better than PSO for a small number of recommended parameter configurations. It is also better than the default LibSVM parameters. Hence, meta-learning alone would be indicated in situations in which the SVM user had strong resources constraints. In these situations, meta-learning could recommend a lower number of configurations with intermediate performance levels. PSO in turn is able to find better configurations along its search and then it is more adequate if a higher number of configurations can be tested.

The Hybrid-PSO was able to combine the advantages of its components. The performance of the Hybrid-PSO in the initial five recommended configurations is of course the same as the performance of meta-learning (since the initial configurations are recommended by meta-learning). From that point of the curve, the Hybrid-PSO consistently achieves better results compared to both the PSO and the meta-learning. It converges earlier to solutions with similar NMSE values compared to the best configurations observed in the 40 problems. There is an additional cost in recommending the configurations by the hybrid approach which is the cost of the meta-learning initialization (specially the cost of computing the meta-features). However, we deployed meta-features with a low computational cost.

In [5], we extended the previous work to perform Multi-Objective Optimization (MOO) of SVM parameters. The Multi-Objective PSO (MOPSO) algorithm was used to optimize the parameters (γ, C) regarding two conflicting objectives: complexity (number of support vectors) and success rate. We evaluated the MOPSO in two different versions: (1) MOPSO with initial population suggested by ML (Hybrid MOPSO) and (2) MOPSO with random initial population. In the meta-learning module, for each similar problem retrieved, we generated a Pareto Front (a set of non-dominated solutions) by applying the dominance evaluation to the 399 configurations of SVM parameters considered. In order to suggest an initial population, we select one random solution of each produced Pareto Front.

In our experiments, the final Pareto Fronts optimized by the MOPSO and the Hybrid MOPSO were evaluated using three metrics for MOO problems: Spacing, Hypervolume and Spread. The proposed hybrid approach was able to generate better comparative results, considering the Spacing and Hypervolume metrics. Regarding the maximum Spread, our approach lost in first generations, but was similar to MOPSO in the last generations.

3 Conclusion

The combination of meta-learning and optimization techniques showed promising results for SVM parameter values selection. The proposed approach can be easily adapted to other learning algorithms (e.g., Artificial Neural Networks). A number of aspects need to be investigated in our proposed solution such as alternative strategies to integrate meta-learning in the optimization process. For instance, not only the best solutions to similar problems can be considered, but also diverse solutions in the search space. Additionally, the limitations of the individual components (as usual in hybrid systems) need to be dealt with. For instance, new strategies to augment the number of datasets for meta-learning can improve the learning performance in our context.

Acknowledgments: The authors would like to thank CNPq, CAPES, FAPESP and FACEPE (Brazilian Agencies) and FCT (Portuguese Agency) for their financial support.

REFERENCES

[1] S. Ali and K. Smith-Miles, ‘A meta-learning approach to automatic kernel selection for support vector machines’, *Neurocomputing*, **70**(1-3), 173–186, (2006).

[2] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, Cambridge University Press, 2000.

[3] T. Gomes, R. B. C. Prudêncio, C. Soares, A. Rossi, , and A. Carvalho, ‘Combining meta-learning and search techniques to select parameters for support vector machines’, *Neurocomputing*, **75**, 3–13, (2012).

[4] S. Louis and J. McDonnell, ‘Learning with case-injected genetic algorithms’, *IEEE Trans. on Evolutionary Computation*, **8**, 316–328, (2004).

[5] P. Miranda, R. B. C. Prudêncio, C. Soares, and A. Carvalho, ‘Multi-objective optimization and meta-learning for svm parameter selection’, in *IJCNN 2012 (to appear)*, (2012).

[6] C. Soares, P. Brazdil, and P. Kuba, ‘A meta-learning approach to select the kernel width in support vector regression’, *Machine Learning*, **54**(3), 195–209, (2004).

Formal Frame for Data Mining with Association Rules – a Tool for Workflow Planning

Jan Rauch and **Milan Šimůnek** [1]

1 INTRODUCTION

The goal of this extended abstract is to contribute to the forum for research on construction of data mining workflows. We briefly introduce a formal framework called *FOFRADAR* (FOrmal FRAmework for Data mining with Association Rules) and then we outline how it can be used to control a workflow of data mining with association rules. We consider this relevant to associative classifiers that use association rule mining in the training phase [3].

We deal with association rules $\varphi \approx \psi$ where φ and ψ are general Boolean attributes derived from columns of analyzed data matrices. Symbol $\approx$ is called 4ft-quantifier and it stands for a condition concerning a contingency table of φ and ψ [6]. Such rules are more general than rules introduced in [1]. We consider data mining process as described by the well known CRISP-DM methodology.

The FOFRADAR is introduced in [5]. Its goal is to formally describe a data mining process such that domain knowledge can be used both in formulation of reasonable analytical questions and in interpretation of resulting set of association rules. No similar approach to dealing with domain knowledge in data mining is known to the authors. An application of the FOFRADAR in data mining workflows is outlined here for the first time.

2 FOFRADAR

FOFRADAR is based on a **logical calculus $\mathcal{LC}$ of association rules**. Formulas of $\mathcal{LC}$ correspond to the association rules $\varphi \approx \phi$ [4]. Such rules are evaluated in data matrices rows of which correspond to observed objects $o_1, \ldots, o_n$ and columns correspond to observed attributes $A_1, \ldots, A_K$. We assume that A_i has a finite number $t_i \geq 2$ of possible values $1, \ldots, t_i$ (i.e. *categories*) and $A_i(o_j)$ is a value of A_i in row o_j for $i = 1, \ldots, K$ and $j = 1, \ldots, n$.

Boolean attributes φ, ϕ are derived from basic Boolean attributes i.e expressions $A_i(\alpha)$ where $\alpha \subset \{1, \ldots, t_i\}$. A basic Boolean attribute $A_i(\alpha)$ is true in a row o_j of a given data matrix $\mathcal{M}$ if $A_i(o_j) \in \alpha$, otherwise it is false. Thus, we do not deal only with Boolean attributes - conjunctions of attribute-value pairs $A_i(a)$ where $a \in \{1, \ldots, t_i\}$ but we use general Boolean attributes derived by connectives $\wedge, \vee, \neg$ from columns of a given data matrix.

The *4ft-table* $4ft(\varphi,\psi, \mathcal{M})$ of φ and ψ in a data matrix $\mathcal{M}$ is a quadruple $\langle a, b, c, d\rangle$ where a is the number of rows of $\mathcal{M}$ satisfying both φ and ψ, b is the number of rows satisfying φ and not satisfying ψ, c is the number of rows not satisfying φ and satisfying ψ and d is the number of rows satisfying neither φ nor ψ. A $\{0, 1\}$-valued associated function $F_{\approx}(a, b, c, d)$ is defined for each 4ft-quantifier $\approx$. The rule $\varphi \approx \psi$ is true in a data matrix $\mathcal{M}$ if $F_{\approx}(a, b, c, d) = 1$ where $\langle a, b, c, d\rangle = 4ft(\varphi,\psi, \mathcal{M})$, otherwise it is false in $\mathcal{M}$.

Expression $A_1(1, 2, 3) \vee A_2(4, 6) \Rightarrow_{p,B} A_3(8, 9) \wedge A_4(1)$ is an example of association rule, $\Rightarrow_{p,B}$ is a 4ft-quantifier of founded implication. It is $F_{\Rightarrow_{p,B}}(a, b, c, d) = 1$ if and only if $\frac{a}{a+b} \geq p \wedge a \geq B$ [2]. There are various additional 4ft-quantifiers defined in [2, 4].

A deduction rule $\frac{\varphi \approx \psi}{\varphi' \approx \psi'}$ is correct if the following is true for each data matrix $\mathcal{M}$: *if $\varphi \approx \psi$ is true in $\mathcal{M}$ then also $\varphi' \approx \psi'$ is true in* $\mathcal{M}$. There are reasonable criteria making possible to decide if $\frac{\varphi \approx \psi}{\varphi' \approx \psi'}$ is a correct deduction rule [4].

FOFRADAR consists of a logical calculus $\mathcal{LC}$ of association rules and of several mutually related languages and procedures used to formalize both items of domain knowledge and important steps in the data mining process. They are shortly introduced below, relations of some of them to the CRISP-DM are sketched in Fig. 1.

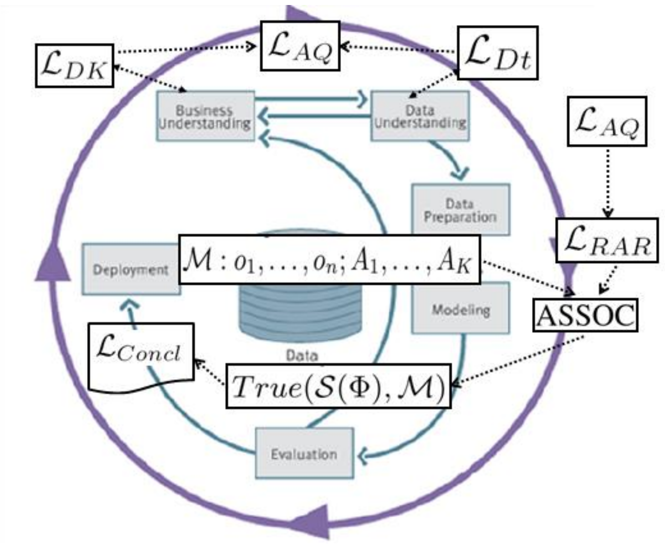


Figure 1. FOFRADAR framework and CRISP-DM methodology

Language $\mathcal{L}_{DK}$ of domain knowledge – formulas of $\mathcal{L}_{DK}$ correspond to items of domain knowledge. A formula $A_1 \uparrow\uparrow A_{11}$ meaning that if A_1 increases then A_{11} increases too is an example. We consider formulas of $\mathcal{L}_{DK}$ as results of business understanding.

***Language $\mathcal{L}_{Dt}$* of data knowledge** – its formulas can be considered as results of data understanding. An example is information that 90 per cent of observed patients are men.

***Language $\mathcal{L}_{AQ}$* of analytical questions** – the expression

[1] University of Economics, Prague, Czech Republic, email: rauch@vse.cz and simunek@vse.cz

$[\mathcal{M} : A_1, \ldots, A_{10} \approx^? A_{11}, \ldots, A_{20}; \not\to A_1 \uparrow\uparrow A_{11}]$ is an example of a formula of $\mathcal{L}_{AQ}$. It corresponds to a question $\mathcal{Q}_1$: *In data matrix $\mathcal{M}$, are there any relations between combinations of values of attributes $A_1, \ldots, A_{10}$ and combinations of values of attributes $A_{11}, \ldots, A_{20}$ which are not consequences of $A_1 \uparrow\uparrow A_{11}$?*

***Language* $\mathcal{L}_{RAR}$ of sets of relevant association rules** – each formula Φ of $\mathcal{L}_{RAR}$ defines a set $\mathcal{S}(\Phi)$ of rules relevant to a given analytical question. The set $\mathcal{S}(\Phi)$ relevant to $\mathcal{Q}_1$ can consist of rules $\varphi \Rightarrow_{0.9,100} \psi$ where φ is a conjunction of some of basic Boolean attributes $A_1(\alpha_1), \ldots, A_{10}(\alpha_{10})$, similarly for ψ and $A_{11}, \ldots, A_{20}$. Here α_1 can be e.g. any interval of maximal 3 consecutive categories, similarly for additional basic Boolean attributes.

Procedure ASSOC – its input consists of a formula Φ of $\mathcal{L}_{RAR}$ and of an analyzed data matrix $\mathcal{M}$. Output of the ASSOC procedure is a set $True(\mathcal{S}(\Phi), \mathcal{M})$ of all rules $\varphi \approx \psi$ belonging to $\mathcal{S}(\Phi)$ which are true in $\mathcal{M}$. The procedure 4ft-Miner [6] is an implementation of ASSOC. It deals with a very sophisticated language $\mathcal{L}_{RAR}$.

Procedure Cons – this procedure maps a formula Ω of $\mathcal{L}_{DK}$ to a set $Cons(\Omega, \approx)$ of association rules $\varphi \approx \psi$ which can be considered as consequences of Ω. The set $Cons(A_1 \uparrow\uparrow A_{11}, \Rightarrow_{p,B})$ is a set of all rules $\varphi \Rightarrow_{p,B} \phi$ for which $\frac{\omega \Rightarrow_{p,B} \tau}{\varphi \Rightarrow_{p,B} \phi}$ is a correct deduction rule and $\omega \Rightarrow_{p,B} \tau$ is an atomic consequence of $Cons(A_1 \uparrow\uparrow A_{11})$. Rules $A_1(\mathit{low}) \Rightarrow_{p,B} A_{11}(\mathit{low}))$ and $A_1(\mathit{high}) \Rightarrow_{p,B} A_{11}(\mathit{high})$ are examples of atomic consequences of $A_1 \uparrow\uparrow A_{11}$, *low* and *high* are suitable subsets of categories of A_1 and A_{11}. Some additional rules can also be considered as belonging to $Cons(A_1 \uparrow\uparrow A_{11}, \Rightarrow_{p,B})$, see [5] for details.

***Language* $\mathcal{L}_{Concl}$** – formulas of this language correspond to conclusions which can be made on the basis of the set $True(\mathcal{S}(\Phi), \mathcal{M})$ produced by the ASSOC procedure. Two examples of such conclusions follow. (1): *All rules in $True(\mathcal{S}(\Phi), \mathcal{M})$ can be considered as consequences of known items of domain knowledge $A_1 \uparrow\uparrow A_{11}$ or $A_2 \uparrow\uparrow A_{19}$*. (2): *Lot of rules from $True(\mathcal{S}(\Phi), \mathcal{M})$ can be considered as consequences of yet unknown item of domain knowledge $A_9 \uparrow\uparrow A_{17}$*.

There are additional procedures belonging to FOFRADAR, they transform formulas of a particular language to formulas of another language of FOFRADAR [5].

3 FOFRADAR and Workflow of Data Mining

To keep things simple and the explanation concise we assume that the analyzed data matrix $\mathcal{M}$ is given as a result of necessary transformations. In addition, we assume that a set DK of formulas of the language $\mathcal{L}_{DK}$ and a set DtK of formulas of the language $\mathcal{L}_{Dt}$ are given. A workflow of data mining with association rules can be then described according to Fig. 2.

The first row in Fig. 2 means that an analytical question $\mathcal{Q}$ which can be solved by the procedure ASSOC is formulated using set DK of formulas of the language $\mathcal{L}_{DK}$. The set DtK of formulas of the language $\mathcal{L}_{DK}$ can also be used to formulate reasonable analytical questions.

A solution of $\mathcal{Q}$ starts with a definition of a set $\mathcal{S}(\Phi)$ of relevant association rules which have to be verified in $\mathcal{M}$, see row 2 in Fig. 2. The set $\mathcal{S}(\Phi)$ is given by a formula Φ of language $\mathcal{L}_{RAR}$. The formula Φ is a result of application of a procedure transforming formulas of $\mathcal{L}_{AQ}$ to formulas of $\mathcal{L}_{RAR}$.

Then, the procedure ASSOC is applied, see row 3 in Fig. 2. Experience with the procedure 4ft-Miner, which is an enhanced implementation of the ASSOC procedure, are given in [6, 7]. The application of ASSOC results into a set $True(\mathcal{S}(\Phi), \mathcal{M})$ of all association

```
 1   1 Formulate_Analytical_Question
 2     Define_Set_of_Relevant_Rules
 3   2 Apply ASSOC
 4     Apply CONCL
 5     IF Continue_ASSOC THEN
 6        BEGIN
 7        Modify Set_of_Relevant_Rules
 8        GOTO 2
 9        END
10     IF Continue_Analysis THEN GOTO 1
11     STOP
```

Figure 2. Association rule data mining workflow based on FOFRADAR

rules $\varphi \approx \psi$ which belong to $\mathcal{S}(\Phi)$ and which are true in $\mathcal{M}$.

A next step is interpretation of the set $True(\mathcal{S}(\Phi), \mathcal{M})$. This is realized by the procedure $CONCL$, see row 4 in Fig. 2. Consequences of particular items of domain knowledge are used which means that the procedure $Cons$ is applied and several formulas of the language $\mathcal{L}_{Concl}$ are produced by the procedure $CONCL$.

One of formulas produced by $CONCL$ is a simple Boolean variable $Continue\_ASSOC$. If its value is setup as $true$, then the set $\mathcal{S}(\Phi)$ of relevant association rules which have to be verified is modified and the process of solution of the analytical question $\mathcal{Q}$ continues, ses rows 5 – 9 in Fig. 2. The modification of the set $\mathcal{S}(\Phi)$ is done by the procedure $ModifySet\_of\_Relevant\_Rules$ which uses experience from applications of the 4ft-Miner procedure.

If the value of $Continue\_ASSOC$ is setup to $false$, then the process of solution of the particular analytical question is terminated. Then a procedure $Continue\_Analysis$ is used to decide if an additional analytical question will be generated and solved.

There are first experiences with "manually driven" processes corresponding to procedures used in Fig. 2. The most important is experience with process corresponding to the $CONCL$ procedure, see [7]. We believe to get enough experience to run the whole data workflow process automatically as outlined in Fig. 2.

ACKNOWLEDGEMENTS

The work described here has been supported by Grant No. 201/08/0802 of the Czech Science Foundation.

REFERENCES

[1] R. Aggraval et al., *Fast Discovery of Association Rules*, 307–328, Advances in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, 1996.

[2] P. Hájek and T. Havránek, *Mechanising Hypothesis Formation - Mathematical Foundations for a General Theory*, Springer, Berlin, 1978

[3] M. Jalali-Heravi and R. Zaiane, *A study on interestingness measures for associative classifiers*, 1039–1046, Proceedings of the 2010 ACM Symposium on Applied Computing, ACM New York, 2010

[4] J. Rauch: Logic of association rules, *Applied Intelligence*, **22**, 9–28, 2005

[5] J. Rauch, *Consideration on a Formal Frame for Data Mining*, 562–569, Proceedings of Granular Computing 2011, IEEE Computer Society, Piscataway, 2011.

[6] J. Rauch and M. Šimůnek, *An Alternative Approach to Mining Association Rules*, 219–238, Data Mining: Foundations, Methods, and Applications, Springer-Verlag, Berlin, 2005.

[7] J. Rauch and M. Šimůnek, *Applying Domain Knowledge in Association-Rules Mining Process - First Experience*, 113–122, Foundations of Intelligent Systems, Springer-Verlag, Berlin, 2011.

Designing KDD-Workflows via HTN-Planning for Intelligent Discovery Assistance

Jörg-Uwe Kietz[1] and **Floarea Serban**[1] and **Abraham Bernstein**[1] and **Simon Fischer**[2]

Abstract. Knowledge Discovery in Databases (KDD) has evolved a lot during the last years and reached a mature stage offering plenty of operators to solve complex data analysis tasks. However, the user support for building workflows has not progressed accordingly. The large number of operators currently available in KDD systems makes it difficult for users to successfully analyze data. In addition, the correctness of workflows is not checked before execution. Hence, the execution of a workflow frequently stops with an error after several hours of runtime.

This paper presents our tools, eProPlan and eIDA, which solve the above problems by supporting the whole life-cycle of (semi-) automatic workflow generation. Our modeling tool eProPlan allows to describe operators and build a task/method decomposition grammar to specify the desired workflows. Additionally, our Intelligent Discovery Assistant, eIDA, allows to place workflows into data mining (DM) tools or workflow engines for execution.

1 Introduction

One of the challenges of Knowledge Discovery in Databases (KDD) is assisting users in creating and executing KDD workflows. Existing KDD systems such as the commercial IBM SPSS Modeler[3] or the open-source KNIME[4] and RapidMiner[5] support the user with nice graphical user interfaces. Operators can be dropped as nodes onto the working pane and the data-flow is specified by connecting the operator-nodes. This works very well as long as neither the workflow becomes too complicated nor the number of operators becomes too large.

However, in the past decade, the *number of operators* in such systems has been growing fast. All of them contain over 100 operators and RapidMiner, which includes Weka, R, and several pluggable operator sets (such as anomaly detection, recommendation, text and image mining) now has around 1000. It can be expected that the transition from closed systems (with a fixed set of operators) to open systems that can also use Web services as operators (which is especially interesting for domain specific data access and transformations) will further accelerate the rate of growth *resulting in total confusion about what operators to use for most users*.

In addition to the number of operators also the *size of the KDD workflows* is growing. Today's workflows easily contain hundreds of operators. Parts of the workflows are applied several times (e.g. the preprocessing sub-workflow has to be applied on training, testing, and application data) implying that the users either need to copy/paste or even repeatedly design the same sub-workflow[6] several times. As none of the systems maintain this "copy"-relationship, it is left to the user to maintain the relationship in the light of changes.

Another weak point is that workflows are not checked for *correctness* before execution. As a consequence, the execution of the workflow oftentimes stops with an error after several hours runtime due to small syntactic incompatibilities between an operator and the data it should be applied on.

To address these problems several authors [1, 3, 18] propose the use of planning techniques to automatically build such workflows. However, all these approaches are limited. First, they only model a very small number of operations and were only demonstrated to work on very short workflows (less than 10 operators). Second, none of them models operations that work on individual columns of a data set: they only model operations that process all columns of a data set in the same way. Lastly, the approaches cannot scale to large amounts of operators and large workflows: their planning approaches fail in the large design space of "correct" (but nevertheless most often unwanted) solutions. A full literature review about IDAs (including these approaches) can be found in our survey [13].

In this paper we describe the first approach for designing KDD workflows based on ontologies and Hierarchical Task Network (HTN) planning [5]. *Hierarchical task decomposition knowledge* available in DM (e.g. CRISP-DM [2] and CITRUS [15]) can be used to significantly reduce the number of generated unwanted correct workflows. The main scientific contributions of this paper, hence, are: First, we show how KDD workflows can be designed using ontologies and HTN-planning in eProPlan. Second, we exhibit the possibility to plug in our approach in existing DM-tools (as illustrated by RapidMiner and Taverna). Third, we present an evaluation of our approach that shows significant improvement and simplification of the KDD-workflow design process. Thus, the KDD researchers can easily model not only their DM and preprocessing operators but also their DM tasks that is exploited to guide the workflow generation. Moreover less experienced users can use our RM-IDA plugin to automatically generate workflows in only 7-clicks. Last but not least, the planning community may find it interesting to see this real world problem powered by planning techniques and may also find some of the problems we faced and solved rather pragmatically inspiring for

[1] University of Zurich, Department of Informatics, Dynamic and Distributed Information Systems Group, Binzmühlestrasse 14, CH-8050 Zurich, Switzerland {kietz|serban|bernstein}@ifi.uzh.ch
[2] Rapid-I GmbH, Stockumer Str. 475, 44227 Dortmund, Germany fischer@rapid-i.com

[3] http://www.ibm.com/software/analytics/spss/
[4] http://www.knime.org/
[5] http://rapid-i.com/content/view/181/190/

[6] Several operators must be exchanged and cannot be reapplied. Consider for example training data (with labels) and application data (without labels). Label-directed operations like feature-selection cannot be reapplied. But even if there is a label on separate test data, redoing feature selection may result in selecting different features. To apply and test the model, however, exactly the same features have to be selected.

further research.

The rest of this paper is structured as follows: Section 2 describes the knowledge used for planning, then Section 3 details the components of our system. Section 4 describes several evaluation methods and, finally, we conclude with Section 5.

2 The Planning Knowledge

We modeled our Data Mining Workflow planning problem as a Data Mining Worlkflow ontology (DMWF),[7] which we succinctly illustrate here (a more detailed description of it can be found in [8, 9]).

The DMWF contains *input/output objects* and their *meta-data* (Sec.2.1), which are sufficiently detailed to enable column-wise operations – a feature that is not available in any of the previous approaches. In addition, *tasks and methods* are used to guide and simplify the planning process. Each method consists of a sequence of steps, where each step is a (sub-) task or an operator (Sec. 2.2 shows some of the methods). In total, the DMWF contains *more than 100 operators* from RapidMiner (Sec. 2.3 illustrates one). The amount of operators and their partial redundancy made it favorable to structure them in an inheritance hierarchy starting with abstract operators until the basic operators which can be applied on the data. The number of operators is not a limitation of the HTN-planning approach, but a limitation set by the effort to model them and to keep them consistent with changes introduced by new releases of RapidMiner. To get a significantly higher number of modeled operators, semi-automatic modeling or at least verification methods need to be developed.

Besides the main contribution of supporting users in designing KDD workflows, this paper may also be interesting to the planning community because it shows the successful usage of planning techniques to solve the problem of workflow design and more generally problem-specific software-configuration. It may stipulate further research on planning as we solved some problems that did not get much attention in planning so far. First, in our domain it is usually easy to find a correct plan. The simplest correct plan for prediction uses the default model (mean-value for regression, modal-value for classification). This is a correct solution for all predictive modeling problems, but it is only the baseline that DM wants to improve and not the wanted solution. We tackle that problem by excluding such unwanted workflows from our HTN. The real problem is not finding a solution, but handling the large amount of solutions[8]. We handle this by grouping of plans based on meta-level equivalent output (similar characteristics of the output) [9] and by using the probabilistic pattern generated by meta-learning (see Sec. 2.4) not only to rank the enumerated workflows, but also for a heuristic beam search in the space of possible solutions. Another interesting problem is the large number of relevant operators that we handled by embedding conditions and effects into an ontological operator hierarchy with inheritance. This is supported by our eProPlan plugin into the popular ontology editor Protégé. Furthermore, RapidMiner (and in DM in general) has several special purpose control/loop operators like cross-validation. They are parametrized operators (the number of folds and the sampling strategy for cross validation). In contrast to other operators, it is not sufficient to choose the dominating operators since they contain one or more subtasks that have to be planned as well. This is similar to planning general control structures like if-then-else or loops, but the problem is also easier to solve as these dominating operators and their subtasks have a special purpose and, therefore, task/method decompositions as all other tasks. Figure 1a shows a cross-validation workflow which uses first a preprocessing task and then it applies cross-validation to produce and test the model as seen in Figure 1b. During the training step it selects the important features and then it trains the model. The produced model is then applied on the testing data. The output is an average of the n runs (where n is the number of folds, usually set to 10).

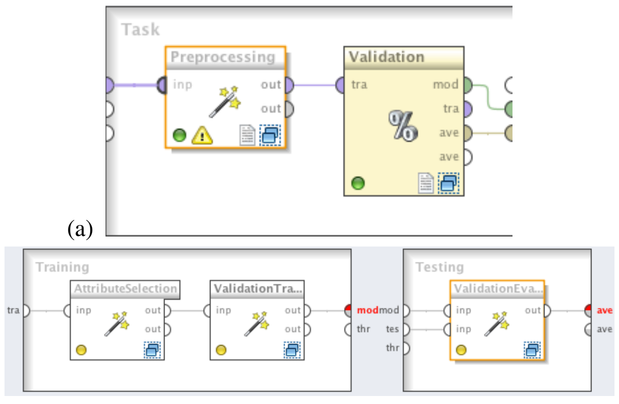


Figure 1: (a) Cross Validation as Operator (labeled as “Validation” in the Figure) in a workflow; (b) Subtasks of Cross Valdation

2.1 Meta-Data to describe Input/Output Objects

The planner recognizes the *IO-Objects* (e.g. *DataTable*, *DocumentCollection* and *ImageCollection*), *Background Knowledge*, *Model* (e.g. *PreprocessingModel* and *PredictionModel*, recording required, modified, added and deleted attributes such that the conditions and effects of applying these models on the data can be computed by the planner), and *Report* (e.g. *PerformanceVector* and *LiftChart*). As an example Table 1 shows the meta-data of a *DataTable*. The meta-data for the user-data is generated by RapidMiner's/RapidAnalytic's meta-data analyzer and passed to the planner. During the planning process the planner generates the meta-data of an operator's output objects from the operator's effect-specification (see Sec. 2.3).

Attribute	#Attr	Type	Role	#Diff	#Miss	Values	Min	Max	Mean	Modal	Std.
age	1	Scalar	input		0	[]	20.0	79.0	50.95		16.74
genLoad	1	Nominal	input	2	36	[0,1]				1	
label	1	Nominal	target	2	0	[+,-]				+	
meas1	1	Scalar	input		0	[]	0.10	3.93	1.845		0.861
meas2	1	Scalar	input		30	[]	0.12	4.35	1.979		0.902
meas3	1	Scalar	input		0	[]	0.33	5.35	2.319		1.056
sex	1	Nominal	input	2	0	[f,m]				m	

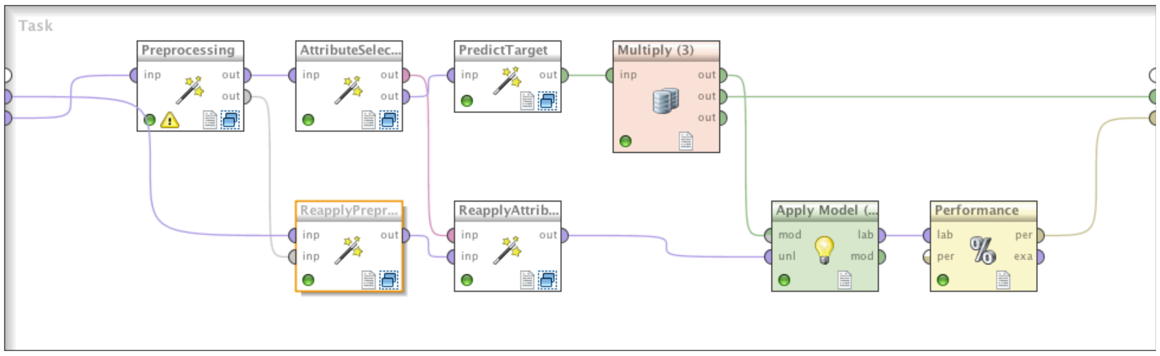
Table 1: Meta-Data for a Data Table

One of the strengths of the IDA is the ability to plan workflows with attribute-wise operations – a feature no other previous approach had so far. Especially biological micro-array data can easily contain several thousand columns, turning this possibility into a major performance bottleneck. Looking at a number of such analyses we observed that these columns often have very similar (if not identical)

[7] It is public available from `http://www.e-LICO.eu/ontologies/dmo/e-Lico-eProPlan-DMWF-HTN.owl`. The best way to open this ontology is: download Protégé 4.0 or 4.1 from `http://protege.stanford.edu/` and eProPlan from `http://elico.rapid-i.com/eproplan.html` (2.0.1 for Protégé4.0 and 2.1.0 for 4.1).

[8] With column-wise operations this may be very large, just consider having 5 alternative methods to discretize 100 attributes. This results in $5^{100} \approx 10^{70}$ possible correct plans.

[9] 'Meta-level equivalent output' can be defined as the IOOs-descriptions produced by the planner are equivalent up to the names of individuals.

(a) 

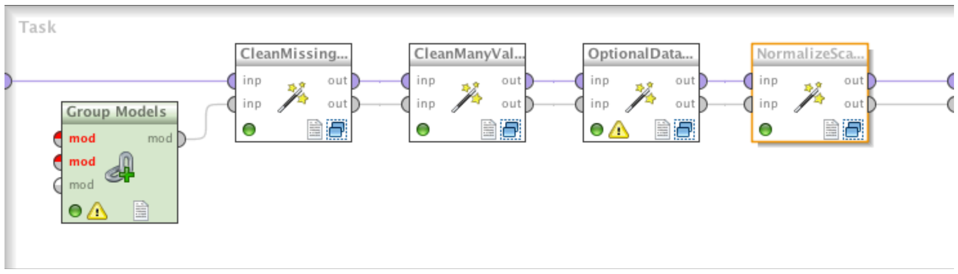
(b) 

Figure 2: Methods for (a) Modeling with Test-Set Evaluation and (b) Preprocessing

meta-data and data characteristics, effectively eliminating the need to differentiate between them during planning. To keep the strength and avoid the performance bottleneck of several thousand attributes that are identical on the meta-data level, we introduced attribute groups that collect all attributes together where the operator conditions would not discriminate between them anyway. This worked well as all column-wise operators were able to handle not only single columns but also specified sets of columns. Therefore, we are able to generate workflows with attribute-group-wise operations for micro-array and collaborative filtering data with even 50.000 attributes.

There are various approaches which have used meta-data to suggest the best algorithm for a given problem [12, 4, 7] in the context of meta-learning.

2.2 The Task/Method decomposition

The top-level task of the HTN is the *DM* task. It has six methods: *Clustering*, *Association Rule Mining*, *Predictive Modeling with Test Set Evaluation* (external given separation), *Predictive Modeling with Cross Validation*, *Predictive Modeling with Test Set Split Evaluation* (random 70:30 split), and *Simple Modeling with Training Set Performance*.

The selection is directed by the (required) main-goal, i.e. *Pattern Discovery* enforces *Association Rule Mining*, *Descriptive Modeling* enforces *Clustering*, and *Predictive Modeling* forces the choice of one of the others. If test data are provided *Modeling with Test Set Evaluation* has to be chosen, otherwise the choice can be influenced by an (optional) evaluation-subgoal (not possible with the current GUI). If there are still several methods possible, they are enumerated in the rank-order provided by the probabilistic planner (see Sec. 2.4). Each method consists of a sequence of steps, where each step is a (sub-)task or an operator (it can also be an abstract operator subsuming several basic or dominating operators). Planning occurs in the order of steps, however the resulting data flow is not necessarily linear, as the grammar allows the specification of port mapping. Figure 2a shows the method *Modeling with Test Set Evaluation* and its flow of IO-Objects in RapidMiner.[10] The white nodes are tasks to be planned and the other nodes are operators. Operators in RapidMiner are grouped in a way similar to the DMWF. Some of the top nodes have colors. The greenish ones are operators that deal with *Model* as well as objects that inherit from *Model*. This includes all learners and the *Apply Model* operator. The more purple ones are data transformation operators. The RapidMiner's plan-interpreter adds a Multiply node whenever an IO-Object is used by several operators.

The *Preprocessing* task has only one method (Fig. 2b). First an empty preprocessing model is created using the operator "Group Models". It is then extended by the next steps. All its sub-tasks have optional "Nothing to Do" methods (as shown for *CleanMissingValues* in Fig. 3c). Most of the preprocessing methods are recursive, handling one attribute at a time until nothing is left to be done. *CleanMissingValues* has two different recursive methods, the choice is made by the planner depending on the amount of missing values. If there are more than 30% values missing, the attribute can be dropped (Figure 3b). When there are less than 50% missing, it can be filled with mean or modal value (Figure 3a). If $30 - 50\%$ of the values are missing, both methods can be used and plans for both are enumerated in the order provided by the probabilistic planner. The usage of column-wise operations is illustrated in Figure. 3, which shows how depending on the amount of missing values per column the planner chooses to fill or drop the column.

Figure 4 shows a generated workflow for the UCI data set labor-negotiations, which has 16 attributes with various amounts of missing values. Note that in its output the planner compresses recursive tasks into a single task to simplify the browsing of the workflow by the user. Such column-wise handling can greatly improve the results of workflows. For users, however, it is usually too much manual effort (placing and connecting the 22 operations and setting their parameters in the workflow below).

In total the HTN of the DMWF now contains 16 tasks with 33

[10] Note that the figures illustrate the methods with structurally equivalent workflows in RapidMiner and do not show the more complicated method definitions in the ontology.

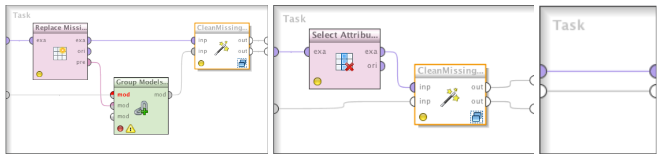


Figure 3: The (a) "Fill Missing Values", (b) "Drop Missing Values", and (c) "Empty"/"Nothing to Do" Method that can be used in Fig. 2b

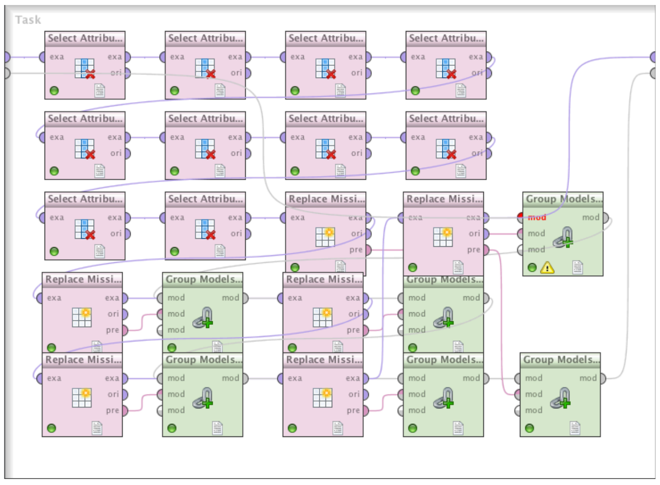


Figure 4: Resulting Example Workflow for Missing Value Cleaning

methods, such that it handles general DM from single table data sets with adequate preprocessing. However, we believe that this will show its real power in customer/application area specific grammars designed and/or extended in eProPlan, resulting in adaptive customer templates for workflows. The complexity of the produced workflows depends on the characteristics of the dataset (if it has missing values, if it needs to be normalized, etc.) and on the number of modeled operators for each steps of the KDD process (*FeatureSelection* and *DataMining* have more operators). Due to space limitations, we only illustrate some parts of the grammar here. The full grammar can be inspected in the public available DMWF-Ontology[11].

2.3 The Operator Models

To be able to express the operators' conditions and effects for planning we stored them as annotations in the ontology. Conditions and effects can contain concept expressions, SWRL-rules,[12] and some extended-SWRL-like built-ins. We have introduced a set of special *built-ins* needed for planning (e.g., *new*, *copy*, *copyComplex*, etc.). These built-ins allow to create, copy, and destroy objects during planning (e.g., models, produced IO-objects, weights, etc.). eProPlan allows to define new built-ins which are stored as subclasses of the *Built-in* concept. Each built-in can have types/parameters and the corresponding implementation in Flora2 [16]. But users who want to add new built-ins need to have some Flora-2 knowledge. They have the possibility to define new functions/operations on the data and introduce them in the conditions and effects. The built-ins' definition with parameters and implementation are stored as class annotations.

[11] Use an OWL2 Ontology Editor like Protege to "Open OWL ontology from URI" with `http://www.e-lico.eu/ontologies/dmo/e-Lico-eProPlan-DMWF-HTN.owl`

[12] `http://www.w3.org/Submission/SWRL/`

Inputs and outputs of an operator are defined as concept expressions and are either stored as superclasses or as equivalent classes. The parameters and the corresponding RapidMiner operator name are stored in equivalent classes. Fig. 5 exemplifies the abstract operator for a classification learner operator with its corresponding inputs/outputs, condition and effect.

"ClassificationLearner":	
Equiv. class:	*PredictiveSupervisedLearners* **and**
	(*uses* **exactly 1** *DataTable*) **and**
	(*produces* **exactly 1** *PredictionModel*) **and**
	(*operatorName* **max 1** Literal)
Condition:	[*DataTable* **and** (*targetAttribute* **exactly 1** *Attribute*) **and**
	(*inputAttribute* **min 1** *Attribute*) **and**
	(*targetColumn* **only** (*DataColumn* **and**
	columnsHasType **only** (*Categorial*))) **and**
	(*inputColumn* **only** (*DataColumn* **and**
	columnsHasType **only** (*Scalar* **or** *Categorial*)))
	](?D)
	→ *new*(**?this**), *ClassificationLearner*(**?this**), *uses*(**?this**,?D)
Effect:	*uses*(**?this**,?D), *ClassificationLearner*(**?this**),
	inputColumn(?D,?IC),*targetColumn*(?D,?TC),
	→ *copy*(?M,?D, {*DataTable(?D)*, *containsColumn*(?D,?_),
	amountOfRows(?D,?_)}),*produces*(**?this**,?M), *PredictionModel*(?M),
	needsColumn(?M,?IC), *predictsColumn*(?M,?TC)

Figure 5: An abstract operator: ClassificationLearner

A basic classification learner operator inherits all the characteristics of the classification learner. In addition, it can define more refined conditions or effects and more parameters. Fig. 6 shows the refinement of the general class of all classification learners to a specific Support Vector Machine implementation in RapidMiner. It has an additional condition (binary target and scalar input attribute and no attribute is allowed to have missing values), but it does not contain a refined effect. Its effect is the one used for all classification learners (it builds a predictive model that requires all input attributes to be present to be applicable and predicts the target attribute).

"RM_Support_Vector_Machine_LibSVM_C_SVC_linear":	
Equiv. class:	RM_Operator **and**
	(*usesData* **exactly 1** *DataTable*) **and**
	(*producesPredictionModel* **exactly 1** *PredictionModel*) **and**
	(*simpleParameter_kernel_type* **value** "linear") **and**
	(*simpleParameter_svm_type* **value** "minimal_leaf_size") **and**
	(*operatorName* **exactly 1** {"support_vector_machine_libsvm"})
Condition:	[*MissingValueFreeDataTable* **and**
	(*targetColumn* **exactly 1** *CategorialColumn*) **and**
	(*inputColumn* **min 1** *Thing*) **and**
	(*inputColumn* **only** (*ScalarColumn*))
	](?D)
	→ *RM_Support_Vector_Machine_LibSVM_C_SVC_linear*(**?this**),
	simpleParameter_svm_type(**?this**,"C-SVC"),
	simpleParameter_kernel_type(**?this**,"linear")

Figure 6: A basic classification learner operator

Each input/output class expression (e.g., *usesData exactly 1 DataTable*) has an annotation which defines its port mapping to its corresponding RapidMiner operator port (e.g., "training set"). Both conditions and effects are rules. Conditions check the applicability (lhs) and infer the parameter settings (rhs); different solutions can infer that the operator can be applied with different parameter settings. Effects compute the variable-bindings (lhs) for the assertion to be made (rhs); all different solutions are asserted as the effect of one operator application.

2.4 Probabilistic Ranking of Workflows

The HTN-planning method described in this paper enumerates all possible workflows for a given problem. The operator models ensure that they are executable without error. Furthermore, the HTN-Grammar prevents senseless operator combinations. For example, first normalizing the data and then discretizing it does not make sense since the normalization effect is absorbed by the discretization one. Also, converting the scalar data to nominal and then converting it back is a useless operation. Another example is dropping attributes without a reason. Nonetheless, the planner can still generate a very large number of correct candidate workflows and we are unaware of any analytical knowledge available to decide which of them will perform well on the current data. Meta-learning tries to learn relations between data characteristics and method performance. Hence, the IDA uses such meta-learned [11] patterns to order the enumeration of M candidate workflows (heuristic search) and to finally select the N best plans and present them to the user. The ranking approach works as follows: whenever several operators ('meta-level NON equivalent output') or methods are applicable, the PP is asked for a (local, up-to now) ranking and delivers the plans in this order. This determines which space is enumerated if the planner is asked for a limited number of solutions. In the end, all generated alternatives are ranked (with left and right context available) for the final presentation to the user.

The planning knowledge described above also does not know about the execution time of operators. This is caused by the fact that the actual runtime of a DM method cannot be predicted easily because of the complexity of the generated models. It can be worst-case bounded in the number of examples and attributes, but its actual size is strongly affected by statistical properties (e.g. noise-level) of the data. The runtime prediction of a DM method was first introduced in [6]. The ranking in our planner, therefore, relies on a meta-learning based method to predict the runtime of modeling and feature selection operators [19].

3 The Overall System

Our system has two main components as illustrated in Fig. 7: eProPlan, our modeling support tool for new operators and new tasks to be solved by the planner, and eIDA, which generates and deploys workflows into DM-suites. eProPlan is the modeling environment for the DM Workflow Ontology (DMWF). It allows to model new operators and uses a task-method decomposition grammar to solve DM problems. Designed as a plugin for the open-source ontology-editor Protégé 4,[13] eProPlan exploits the advantages of the ontology as a formal model for the domain knowledge. Instead of employing the ontological inferences for planning (as done in [3, 17]) we extend the ontological formalism with the main components of a plan, namely operator conditions and effects for classical planning and tasks-methods decomposition grammar for HTN-planning. This allowed us to cleanly separate the inheritance from the planing mechanisms in our systems.

[13] http://protege.stanford.edu/

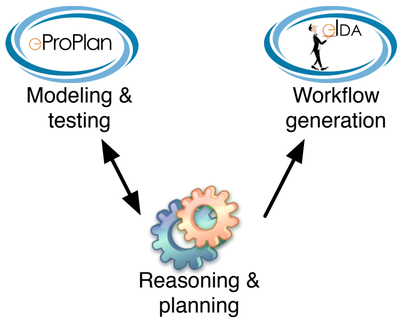


Figure 7: The eProPlan architecture

The planner is implemented in Flora2/XSB [16] and uses different parts from the workflow ontology for different purposes (see Figure 8). Specifically, it employs both a specialized ABox (assertional box—individual assertions) reasoner that relies on an external TBox (terminological box: classes and properties) reasoner (e.g. Pellet[14] or FaCT++ [14]) as a subroutine. Since the TBox reasoner can only partially handle OWL2 (Web Ontology Language[15]), we filter all expressions that are not supported from the ontology. The resulting inferred/completed TBox and its possibly inconsistent class definitions are passed to our ABox reasoner. The ABox reasoner, implemented in Flora2/XSB, first compiles the classified TBox obtained from Pellet on the initial ontology. Then, we process the operators together with their inputs, outputs, preconditions, and effects that are stored as OWL annotations. Tasks and methods are handled analogously. Finally, we finish with the compilation of the problem definition, which is represented by a set of individuals. The problem description has two elements: the input description in terms of meta-data (characteristics of the data like attributes, types, median, etc.) and the goals/hints entered by the user. Both are stored as a set of ABox assertions. Having specified the planning domain and the problem description, one can start planning DM workflows.

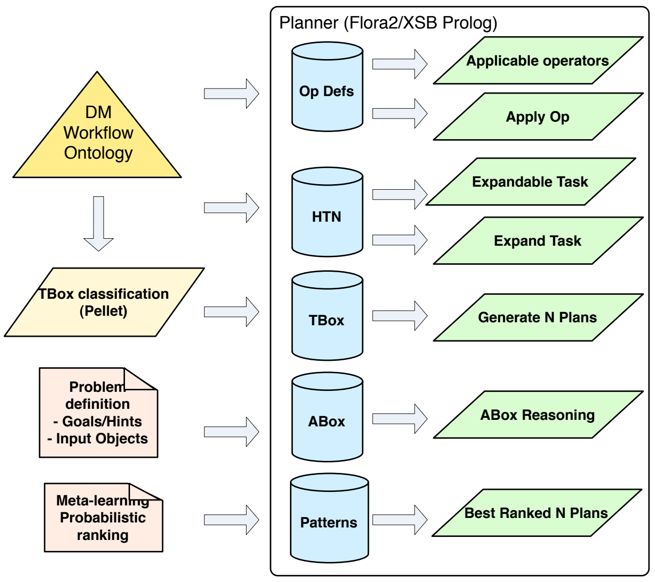


Figure 8: Workflow Ontology and AI Planner capabilities.

eIDA is a programming interface to the reasoner & planner used

[14] http://clarkparsia.com/pellet/
[15] http://www.w3.org/TR/owl2-profiles/

to plugin such an Intelligent Discovery Assistant (IDA), based on the services of the planner, into existing systems (so far into RapidMiner and Taverna).[16] It provides methods for retrieving the plans starting from the data set meta-data and the selection of a main goal. To improve the user experience with the RM-IDA plugin we have developed a simple installer based on precompiled binaries. It works on Linux, Mac OS X 10.5/6, Windows 7 and Windows XP systems.

The RapidMiner IDA Extension can be downloaded (or even auto-installed) from the Rapid-I Marketplace. [17] So far it was downloaded over 150 times during the first two months.

Both presented tools (eProPlan, eIDA) are open source and available on request.

An alternative implementation of RapidMiner IDA Extension exists for Taverna[18]. Taverna can execute all workflows composed of web-services. It can execute the workflows generated by the IDA[19] using any RapidAnalytics [20] server that provides all RapidMiner operators as web-services. Extensions for other KDD tools (e.g., KNIME, Enterprise Miner, etc.) would require two steps: first modeling their corresponding operators in the DMWF, second an implementation of the GUI and the plan-converter using the IDA-API.

4 Evaluation of the IDA

We tested the IDA on 108 datasets from the UCI repository of Machine Learning datasets. [21] It produced executable plans for all 78 classification and 30 regression problems. These datasets have between three and 1558 attributes, being all nominal (from binary too many different values like ZIP), all scalar (normalized or not), or mixed types. They have varying degrees of missing values. We are not aware of any other Machine Learning or DM approach that is able to adapt itself to so many different and divergent datasets. The IDA also works for less well prepared datasets like the KDD Cup 1998 challenge data (370 attributes, with up to 50% missing values and nominal data, where it generates plans of around 40 operators. Generating and ranking 20 of these workflows took 400 sec. on a 3.2 GHz Quad-Core Intel Xeon.

4.1 Ease of Use

Without an IDA data mining is typically achievable by specialized highly-trained professionals such as DM consultants. They have to know a lot about DM methods and how they are implemented in tools. They have to inspect the data and combine the operators into an adequate workflow.

The IDA reduces the technical burden, it now offers "DM with 7 clicks" (see Figure 5). (1) Show the IDA-Perspective of the tool; (2) drag the data to be analyzed from the repository to the view or import (and annotate) your data; (3) select your main goal in DM; (4) ask the IDA to generate workflows for data and goal; (5) evaluate all plans by executing them in RapidMiner; (6) select the plan you like most to see a summary of the plan (the screenshot in Figure 6 is made after this step); and finally, (7) inspect the plan and its results. Note that these steps do not require detailed technical knowledge anymore. Still a user should be aware of what (s)he is doing when (s)he uses

[16] http://www.taverna.org.uk/
[17] http://rapidupdate.de:8180/UpdateServer/faces/product_details.xhtml?productId=rmx_ida
[18] http://e-lico.eu/taverna-ida.html
[19] http://e-lico.eu/taverna-rm.html
[20] http://rapid-i.com/content/view/182/196/
[21] http://archive.ics.uci.edu/ml/

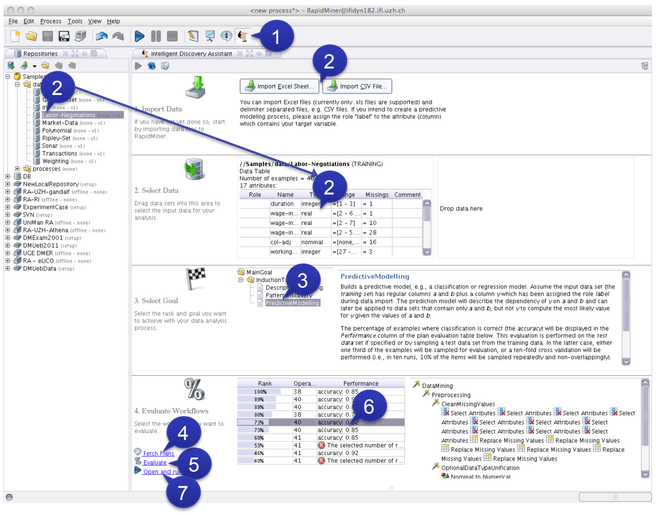


Figure 9: IDA Interface in RapidMiner

DM, i.e. (s)he should know the statistical assumptions underlying DM (e.g., a user should know what it means to have a sample that is representative, relevant, and large enough to solve a problem with DM/statistics). But this is knowledge required in any experimental science.

4.2 Speedup of Workflow Design

Besides making DM easier for inexperienced users, our main goal in building the IDA was to speed-up the design of DM workflows. To establish a possible speed-up we compared the efficiency of computer science students after attending a DM class to a person using the IDA. The study comprises in total 24 students (9 in 2011 and 15 in 2012). They had to solve the following DM problems:

- Take the UCI "Communities and Crime" data from `http://archive.ics.uci.edu/ml/datasets/Communities+and+Crime` and
  - **a)** generate a fine clustering of data that allows me to look for very similar communities
  - **b)** generate a description of the clusters (learn a model to predict the cluster label build in task a)).
  - **c)** generate a function to predict "ViolentCrimesPerPop" and evaluate it with 10-fold cross-validation.
- Take the UCI "Internet Advertisement" data from `http://archive.ics.uci.edu/ml/datasets/Internet+Advertisements` and generate an evaluated classification for the attribute "Ad/Non-Ad".

All data are provided as already imported into RapidMiner (a local RapidAnalytics server they could access). All students took/needed the full 3 hours.

The study confirmed that the standard DM problems they had to solve (clustering and prediction tasks on complex UCI data) can be sped-up by the using an IDA whilst maintaing comparable quality: it took the students 3 hours (designing and executing the workflows) to solve the tasks, whereas a non-specialist using the IDA accomplished the same tasks in 30 minutes (IDA planning and minimal manual adaptation and execution of the workflows) with a comparable output. Table 10 shows how many students managed to solve successfully the given problems and how the IDA solved it.

Task	#Students Succeeded	#Students partial success	IDA's solution
Crime Data			
Clean Missing Values	22/24	1/24	drop & fill
Many Valued Nominals	11/24	6/24	drop
Normalization	-	-	yes
Clustering	9/24	-	2/10-means
Cluster Description	8/24	-	DT
Regression	RMSE$<$0.2:11/24, 0.22$\leq$RMSE$\leq$0.244: 3/11, Best RMSE=0.136 $\pm$0.010 : 4/24	-	k-NN RMSE=0.2
Evaluation	15/24	2/24	10-fold X-Val
Advertisement Data			
Clean Missing Values	17/24	1/24	fill
Feature Selection	no	no	no
Classification	acc$>$96%:13/24, 86%$\leq$acc$<$96%:6/24, Best Acc=97.32%:1/24	-	DT, Acc=96.34% $\pm$ 0.65
Evaluation	5/24	7/24	10-fold X-Val

Figure 10: Features of the designed solutions by students and IDA

Both datasets had missing values which could be ignored ($-$), the attribute could be all dropped or all filled or depending on the amount of missing values individually dropped & filled. Here, both students and IDA did a good job. The ”Communities and Crime” data had key-like attributes (many valued nominals) which are likely to disturb the DM results and should be dropped or marked as (to be) ignore(d). Here, only around half of the students handled it correctly. Numerical attributes with different scales cause unwanted weighting of attributes in distance based similarity computation. Therefore, they should be normalized[22]. There are several clustering methods in RapidMiner. The best way to solve the clustering task is by using hierarchical top-down 2-means (k-means with $k = 2$) clustering till the grouping is fine enough. Only one student used this approach. The rest of the students and the IDA used k-means with different values for k (k<20 is successful, larger values make the prediction very difficult).The IDA sets $k = 2$ and there is no way to specify the goal of a ”fine clustering”. This can be solved by opening the returned workflow and changing the parameter (re-running it is not much effort). We manually choose $k = 10$ as we had the next task in mind and knew it is difficult to predict a nominal with too many different values[23], but many students chose a too fine k and failed the cluster description task (using different methods like Naive Bayes (NB), Decision Tree (DT) or jRIP), most only build bad models using the not dropped key-like attributes). For the ”ViolentCrimesPerPop”, most students used linear regression (linReg). The probabilistic ranking of the IDA preferred k-nearest neighbor. Common mistakes for this task were: regression applied on the wrong attribute, converting numeric data to nominal and applying NaiveBayes (bad accuracy), or converting it to binary: no crime (3 examples), crime (595 examples) the resulting model of course predicted crime everywhere. The DM step should have used a 10-fold cross validation, but some students delivered a training/test set split (maybe to save execution time, maybe because that was used in the group exercise). ”Internet Advertisement” data has many attributes, so Feature Selection would have been an option, but no one did it, also the IDA had none in the top 5 ranked plans. The task was a simple binary prediction, non-ads are much more frequent (2820 non-ad, 459 ad), solved by all students by different methods. One balanced the data, but that worsened the results. One learned a decision tree but did not do an evaluation of the results. Some students even failed to produce a model or plot the data incorrectly (20:80).

This user evaluation provides a strong indication about the strength of the IDA. Note that the students are an optimal user-group for the IDA, as they have limited DM experience but understand the principles of DM.

[22] In fact the initial data is 0-1 range normalized, so the students did not do that step, but preprocessing operations like filling missing values change the column statistics. The planner cannot predict the results very well for column groups, so it ensures normalization of the data at the end of preprocessing.

[23] The meta-data analysis returns categorial for $\leq$ 10 different values and nominal otherwise. Prediction requires a categorial target (or numeric) target, i.e. the IDA refuses to build a plan to predict a target with more than 10 nominal values.

4.3 Performance of the generated workflows

The performance of the generated workflows depends strongly on the ranking. The baseline strategy is to rank the workflows simply based on the popularity of the operators. RapidMiner automatically collects operator usage-frequencies from all the users who accept to submit it. A workflow is ranked *better*, if it contains more frequently used operators. This already produces workflows comparable to user-designed workflows and was used in the speedup-experiments. Our e-LICO project partners[24] used the planner to systematically generate workflows, executed them to get the performance data, and applied meta-learning to these experiments [11].

In [10] they evaluated the meta-mining module and the resulting plan ranking on 65 biological datasets. These datasets are high dimensional with few instances/samples. For their experiments they cross-validated all performance by holding out a dataset. The resulting meta-model was then used to rank the IDA-generated workflows. They found that the meta-learned rankings significantly outperformed the default, frequency-based strategy. Hence, their ranker was able to improve on our ranking to find DM workflows that maximize predictive performance.

5 Conclusion

We presented our Intelligent Discovery Assistant (eIDA and eProPlan) for planning KDD workflows. eIDA can be easily integrated into existing DM-suites or workflow engines. eProPlan is a user-friendly environment for modeling DM operators and defining the HTN grammar for guiding the planning process. Furthermore, it is able to plan attribute-wise operations. The main scientific contribution of the IDA is the ability to build complex workflows out of a much larger set of operations than all previous systems. The demo presents how planning-based KDD workflow design can significantly help KDD practitioners to make their daily work more efficient.

ACKNOWLEDGEMENTS

The work described in this paper is partially supported by the European Community 7^{th} framework program ICT-2007.4.4 under grant number 231519 “e-Lico: An e-Laboratory for Interdisciplinary Collaborative Research in DM and Data-Intensive Science” and the result of collaborative work within this project. We especially like to thank Phong Nguyen, Alexandros Kalousis, Melanie Hilario, and Tom Smuc for their work on meta-learning and ranking, sketched here to show the overall picture.

[24] `http://www.e-LICO.eu/`

REFERENCES

[1] A. Bernstein, F. Provost, and S. Hill, 'Towards Intelligent Assistance for a Data Mining Process: An Ontology-based Approach for Cost-sensitive Classification', *IEEE Transactions on Knowledge and Data Engineering*, **17**(4), 503–518, (April 2005).

[2] P. Chapman, J. Clinton, R. Kerber, T. Khabaza, T. Reinartz, C. Shearer, and R. Wirth, 'Crisp–dm 1.0: Step-by-step data mining guide', Technical report, The CRISP–DM Consortium, (2000).

[3] C. Diamantini, D. Potena, and E. Storti, 'KDDONTO: An Ontology for Discovery and Composition of KDD Algorithms', in *Proceedings of the SoKD-09 Workshop at ECML/PKDD*, (2009).

[4] J. Gama and P. Brazdil, 'Characterization of classification algorithms', *Progress in Artificial Intelligence*, 189–200, (1995).

[5] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory & Practice*, Morgan Kaufmann, San Francisco, CA, USA, 2004.

[6] N. Jankowski and K. Grabczewski, 'Building meta-learning algorithms basing on search controlled by machine complexity', in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2008*, pp. 3601–3608. IEEE, (2008).

[7] A. Kalousis and M. Hilario, 'Model selection via meta-learning: a comparative study', in *International Journal on Artificial Intelligence Tools, ICTAI 2000*, pp. 406–413. IEEE, (2000).

[8] J.-U. Kietz, F. Serban, A. Bernstein, and S. Fischer, 'Towards cooperative planning of data mining workflows', in *Proceedings of the SoKD-09 Workshop at ECML/PKDD*, pp. 1–12, (2009).

[9] J.-U. Kietz, F. Serban, A. Bernstein, and S. Fischer, 'Data mining workflow templates for intelligent discovery assistance and auto-experimentation', in *Proceedings of the SoKD-10 Workshop at ECML/PKDD*, (2010).

[10] P. Nguyen and A. Kalousis, 'Evaluation report on meta-miner'. Deliverable 7.2 of the EU-Project e-LICO, January 2012.

[11] P. Nguyen, A. Kalousis, and M. Hilario, 'A meta-mining infrastructure to support kd workflow optimization', in *Proc. of the PlanSoKD-11 Workshop at ECML/PKDD*, (2011).

[12] L. Rendell, R. Seshu, and D. Tcheng, 'Layered concept learning and dynamically-variable bias management', in *Proceedings of International Joint Conference on Artificial Intelligence, IJCAI-87*, pp. 308–314. Citeseer, (1987).

[13] F. Serban, J. Vanschoren, J.-U. Kietz, and A. Bernstein, 'A Survey of Intelligent Assistants for Data Analysis', *ACM Computing Surveys*, (to appear 2012).

[14] D. Tsarkov and I. Horrocks, 'FaCT++ description logic reasoner: System description', *Lecture Notes in Computer Science*, **4130**, 292, (2006).

[15] R. Wirth, C. Shearer, U. Grimmer, T. P. Reinartz, J. Schlösser, C. Breitner, R. Engels, and G. Lindner, 'Towards process-oriented tool support for knowledge discovery in databases', in *Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery, PKDD '97*, pp. 243–253, London, UK, (1997). Springer-Verlag.

[16] G. Yang, M. Kifer, and C. Zhao, 'Flora-2: A Rule-Based Knowledge Representation and Inference Infrastructure for the Semantic Web', *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, 671–688, (2003).

[17] M. Žaková, P. Křemen, F. Železný, and N. Lavrač, 'Automating knowledge discovery workflow composition through ontology-based planning', *IEEE Transactions on Automation Science and Engineering*, **8**(2), 253 –264, (2011).

[18] M. Žáková, V. Podpečan, F. Železný, and N. Lavrač, 'Advancing data mining workflow construction: A framework and cases using the orange toolkit', in *Proceedings of the SoKD-09 Workshop at ECML/PKDD*, (2009).

[19] M. Znidarsic, M. Bohanec, N. Trdin, T. Smuc, M. Piskorec, and M. Bosnjak, 'Non-functional workflow assessment'. Deliverable D7.3 of the EU Project e-LICO, November 2011.

Experimental Evaluation of the e-LICO Meta-Miner

(Extended Abstract)

Phong Nguyen and **Alexandros Kalousis** and **Melanie Hilario** [1]

1 Introduction

Operator selection is the task of selecting the right operator for building not only valid but also optimal data mining (DM) workflows in order to solve a new learning problem. One of the main achievements of the EU-FP7 e-LICO project[2] has been to develop an *Intelligent Data-Mining Assistant* (IDA) to assist the DM user in the construction of such DM workflows following a cooperative AI-planning approach [2] coupled with a new meta-learning approach for mining past DM experiments, referred as the e-LICO meta-miner [3]. The idea of meta-mining [1] is to build meta-mined models from the full knowledge discovery process by analysing learning problems and algorithms in terms of their characteristics and core components within a declarative representation of the DM process, the Data Mining OPtimization ontology (DMOP)[3].

In this paper, we provide experimental results to validate the e-LICO meta-miner's approach to the operator selection task. We experimented on a collection of real-world datasets with feature selection and classification workflows, comparing our tool with a default strategy based on the popularity of DM workflows. The results show the validity of our approach; in particular, that our selection approach allows to rank appropriately DM workflows with respect to the input learning problem. In the next section, we briefly review the meta-miner. In section 3, we present our results. And in section 4, we conclude.

2 The e-LICO Meta-Miner

The role of the AI-planner is to plan valid DM workflows by reasoning on the applicability of DM operators at a given step i according to their pre/post-conditions. However, since several operators can have equivalent conditions, the number of resulting plans can be in the order of several thousands. The goal of the meta-miner is to select at a given step i among a set of candidate operators A_i the k best ones that will optimize the performance measure associated with the user goal g and its input meta-data $\mathbf{m}$ in order to gear the AI-planner toward optimal plans. For this, the meta-miner makes use of a quality function Q which will score a given plan w by the quality q of the operators that form w as:

$$Q(w|g,\mathbf{m}) = q^*(o_1|g,\mathbf{m}) \prod_{i=2}^{|\mathcal{T}(w)|} q(o_i|\mathcal{T}(w_{i-1}), g, \mathbf{m}) \qquad (1)$$

where $\mathcal{T}(w_{i-1}) = [o_1, .., o_{i-1}]$ is the sequence of previous operators selected so far, and q^* is an initial operator quality function.

[1] University of Geneva, Switzerland, email: Phong.Nguyen@unige.ch
[2] http://www.e-lico.eu
[3] The DMOP is available at http://www.dmo-foundry.org

Thus the meta-miner will qualify a candidate operator by its conditional probability of being applied given all the preceding operators, and select those that have maximum quality to be applied at a step i. In order to have reliable probabilities, the meta-miner makes use of frequent workflow patterns extracted from past DM processes with the help of the DMOP ontology such that the operator quality function q is approximated as:

$$q(o|\mathcal{T}(w_{i-1}), g, \mathbf{m}) \approx \text{aggr}\left\{\frac{\text{supp}(f_i^o|g,\mathbf{m})}{\text{supp}(f_{i-1}|g,\mathbf{m})}\right\}_{f_i^o \in F_i^o} \qquad (2)$$

where aggr is an aggregation function, F_i^o is the set of frequent workflow patterns that match the current candidate workflow w_i^o built with a candidate operator o, and f_{i-1} is the pattern prefix for each pattern $f_i^o \in F_i^o$. More importantly, the quality of a candidate workflow w_i^o will depend on the support function $\text{supp}(f_i^o|g,\mathbf{m})$ of its matching patterns. As described in [3], this support function is defined by learning a dataset similarity measure which will retrieve a dataset's nearest neighbors Exp_N based on the input meta-data $\mathbf{m}$. We refer the reader to [3] for more details. In the next section, we will deliver experimental results to validate our meta-mining approach.

3 Experiments

To meta-mine real experiments, we selected 65 high-dimensional biological datasets representing genomic or proteomic microarray data. We applied on these bio-datasets 28 feature selection plus classification workflows, and 7 classification-only workflows, using tenfold cross-validation. We used the 4 following feature selection algorithms: Information Gain, *IG*, Chi-square, *CHI*, ReliefF, *RF*, and recursive feature elimination with SVM, *SVMRFE*; we fixed the number of selected features to ten. For classification we used the 7 following algorithms: one-nearest-neighbor, *1NN*, the *C4.5* and *CART* decision tree algorithms, a Naive Bayes algorithm with normal probability estimation, *NBN*, a logistic regression algorithm, *LR*, and SVM with the linear, *SVM_l* and the rbf, *SVM_r*, kernels. We used the implementations of these algorithms provided by the RapidMiner data mining suite with their default parameters. We ended up with a total of $65 \times (28 + 7) = 2275$ base-level DM experiments, on which we gathered all experimental metadata; folds predictions and performance results, dataset metadata and workflow patterns, for meta-mining [1].

We constrain the AI-planner so that it generates feature selection and/or classification workflows only. We did so in order for the past experiments to be really relevant for the type of workflows we want to design. Note that the AI-planner can also select from operators with which we have not experimented. These are for feature selection, Gini Index, *Gini*, and Information Gain Ratio, *IGR*. For classification, we used a Naive Bayes algorithm with kernel-based probability estimation, *NBK*, a Linear Discriminant Analysis algorithm, *LDA*, a Rule Induction algorithm, *Ripper*, a Random Tree algorithm, *RDT*, and a Neural Network algorithm, *NNet*.

3.1 Baseline Strategy

In order to assess how well our meta-miner performs, we need to compare it with some baseline. To define this baseline, we will use as the operators quality estimates simply their frequency of use within the community of the RapidMiner users. We will denote this quality estimate for an operator o by $q_{def}(o)$. Additionaly, we will denote the quality of a DM workflow, w, computed using the $q_{def}(o)$ quality estimations by $Q_{def}(w)$, thus:

$$Q_{def}(w) = \prod_{o_i \in \mathcal{T}(wf)} q_{def}(o_i) \tag{3}$$

The score $q_{def}(o)$ focuses on the individual frequency of use of the DM operators, and does not account for longer term interactions and combinations such as the ones captured by our frequent patterns. It reflects thus simply the popularity of the individual operators. In what concerns the most frequently used classification operators, these were *C4.5*, followed by *NBN*, and *SVM_l*. For the feature selection algorithms, the most frequently used were *CHI* and *SVM-RFE*.

3.2 Evaluation and Comparison Strategy

The evaluation will be done in a leave-one-dataset-out manner, where we will use our selection strategies on the remaining 64 datasets to generate workflows for the dataset that was left out. On the left-out dataset, we will then determine the K best workflows using the baseline strategy as well as using the meta-miner selection strategy. To compare the performance of the ordered set of workflows constructed by each strategy, we will use the average estimated performance of the K workflows on the given dataset, which we will denote by ϕ_a. We will report the average of ϕ_a over all the datasets. Additionally, we will estimate the statistical significance of the number of times over all the datasets that the meta-miner strategy has a higher ϕ_a than the baseline strategy; we will denote this by ϕ_s. We estimated the neighborhood Exp_N of a dataset using $N = 5$ nearest neighbors. We will compare the performance of the baseline and of the meta-miner for $K = 1, 3, 5$ generated workflows in order to have a large picture of their overall performance.

3.3 Performance Results and Comparisons

K=1. The top-1 workflow selected by the baseline strategy is *CHI-C4.5*. When we compare its performance against the performance of the top-1 workflow selected by the meta-miner given in the first row of table 1, we can see that the meta-mining strategy gives an average performance improvement of around 6% over the baseline strategy. In addition, its improvement over the baseline is statistically significant in 53 datasets over 65, while the baseline wins only on 11 datasets.

K=3. The two other workflows selected by the baseline strategy additionally to the top-1 are *CHI-NBN* and *CHI-SVM_l*. When we extend the selection to the three best workflows, we obtain the results given in the second row of table 1, where we see that the average predictive performance improvement over the baseline strategy is around 2%. As before, the meta-miner achieves significantly better performance than the baseline in a larger number of baselines datasets than vice-versa.

K=5. The two other workflows selected by the baseline strategy additionally to the top-3 are *SVMRFE-C4.5* and *SVMRFE-SVM_l*. We give the results of the five best workflows selected by the meta-miner in the last row of table 1, where we observe similar trends as before; 2% of average performance improvement and statistical difference in the number of improvement in favor of the meta-mining strategy.

		ϕ_a	ϕ_s	
$K = 1$	Q_{def}	71.92%	11/65	
	Q	77.68%	53/65	p=2e-7
$K = 3$	Q_{def}	75.04%	22/65	
	Q	77.28%	41/65	p=0.046
$K = 5$	Q_{def}	75.18%	18/65	
	Q	77.14%	44/65	p=0.006

Table 1. Performance results and comparisons for the top-K workflows.

3.4 Selected Workflows

We will briefly discuss the top-K workflows selected by the meta-miner. For $K = 1$, we have on a plurality of datasets the selection of the *LDA* classifier, an algorithm we have not experimented with. This happens because within the DMOP ontology this algorithm is related both with the linear, *SVM_l*, and with the NaiveBayes algorithm, both of which perform well on our dataset collection. For $K = 3$ and $K = 5$, we have additionally the selection of the previously unseen *NNet* and *Ripper* classifiers. These operator selections demonstrate the capability of the meta-miner to select new operators based on their algorithm similarities given by the DMOP with past ones.

4 Conclusion and Future Works

This is a preliminary study, but already we see that we are able to deliver better workflow suggestions, in terms of predictive performance, compared to the baseline strategy, while at the same time being able to suggest workflows consisting of operators with which we have never experimented. Future works include more detailed experimentation and evaluation, and the construction of similarity measures combining both the dataset characteristics and the workflow patterns.

ACKNOWLEDGEMENTS

We would like to thank Jörg-Uwe Kietz and Simon Fischer for their contribution in the development and evaluation of the e-LICO meta-miner.

REFERENCES

[1] Melanie Hilario, Phong Nguyen, Huyen Do, Adam Woznica, and Alexandros Kalousis, 'Ontology-based meta-mining of knowledge discovery workflows', in *Meta-Learning in Computational Intelligence*, eds., N. Jankowski, W. Duch, and K. Grabczewski, Springer, (2011).

[2] Jörg-Uwe Kietz, Floarea Serban, Abraham Bernstein, and Simon Fischer, 'Towards Cooperative Planning of Data Mining Workflows', in *Proc of the ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-oriented Knowledge Discovery (SoKD-09)*, (2009).

[3] Phong Nguyen, Alexandros Kalousis, and Melanie Hilario, 'A meta-mining infrastructure to support kd workflow optimization', in *Proc. of the PlanSoKD-2011 Workshop at ECML/PKDD-2011*, (2011).

Selecting Classification Algorithms with Active Testing on Similar Datasets

Rui Leite[1] and **Pavel Brazdil**[2] and **Joaquin Vanschoren**[3]

Abstract. Given the large amount of data mining algorithms, their combinations (e.g. ensembles) and possible parameter settings, finding the most adequate method to analyze a new dataset becomes an ever more challenging task. This is because in many cases *testing* all possibly useful alternatives quickly becomes prohibitively expensive. In this paper we propose a novel technique, called *active testing*, that intelligently selects the most useful cross-validation tests. It proceeds in a tournament-style fashion, in each round selecting and testing the algorithm that is *most likely to outperform the best algorithm of the previous round* on the new dataset. This 'most promising' competitor is chosen based on a history of prior duels between both algorithms on *similar* datasets. Each new cross-validation test will contribute information to a better estimate of dataset similarity, and thus better predict which algorithms are most promising on the new dataset. We also follow a different path to estimate dataset similarity based on data characteristics. We have evaluated this approach using a set of 292 algorithm-parameter combinations on 76 UCI datasets for classification. The results show that active testing will quickly yield an algorithm whose performance is very close to the optimum, after relatively few tests. It also provides a better solution than previously proposed methods. The variants of our method that rely on cross-validation tests to estimate dataset similarity provides better solutions than those that rely on data characteristics.

1 Background and Motivation

In many data mining applications, an important problem is selecting the best algorithm for a specific problem. Especially in classification, there are hundreds of algorithms to choose from. Moreover, these algorithms can be combined into composite learning systems (e.g. ensembles) and often have many parameters that greatly influence their performance. This yields a whole spectrum of methods and their variations, so that *testing* all possible candidates on the given problem, e.g., using cross-validation, quickly becomes prohibitively expensive.

The issue of selecting the right algorithm has been the subject of many studies over the past 20 years [17, 3, 23, 20, 19]. Most approaches rely on the concept of *metalearning*. This approach exploits characterizations of datasets and past performance results of algorithms to recommend the best algorithm on the current dataset. The term *metalearning* stems from the fact that we try to learn the function that maps *dataset characterizations* (meta-data) to *algorithm performance estimates* (the target variable).

The earliest techniques considered only the dataset itself and calculated an array of various simple, statistical or information-theoretic properties of the data (e.g., dataset size, class skewness and signal-noise ratio) [17, 3]. Another approach, called *landmarking* [2, 12], ran simple and fast versions of algorithms (e.g. decision stumps instead of decision trees) on the new dataset and used their performance results to characterize the new dataset. Alternatively, in *sampling landmarks* [21, 8, 14], the complete (non-simplified) algorithms are run on small samples of the data. A series of sampling landmarks on increasingly large samples represents a partial learning curve which characterizes datasets and which can be used to predict the performance of algorithms significantly more accurately than with classical dataset characteristics [13, 14]. Finally, an 'active testing strategy' for sampling landmarks [14] was proposed that actively selects the most informative sample sizes while building these partial learning curves, thus reducing the time needed to compute them.

Motivation. All these approaches have focused on dozens of algorithms at most and usually considered only default parameter settings. Dealing with hundreds, perhaps thousands of algorithm-parameter combinations[4], provides a new challenge that requires a new approach. First, distinguishing between hundreds of subtly different algorithms is significantly harder than distinguishing between a handful of very different ones. We would need many more data characterizations that relate the effects of certain parameters on performance. On the other hand, the latter method [14] has a scalability issue: it requires that pairwise comparisons be conducted between algorithms. This would be rather impractical when faced with hundreds of algorithm-parameter combinations.

To address these issues, we propose a quite different way to characterize datasets, namely through the *effect that the dataset has on the relative performance of algorithms run on them.* As in landmarking, we use the fact that each algorithm has its own learning bias, making certain assumptions about the data distribution. If the learning bias 'matches' the underlying data distribution of a particular dataset, it is likely to perform well (e.g., achieve high predictive accuracy). If it does not, it will likely under- or overfit the data, resulting in a lower performance.

As such, we *characterize a dataset based on the pairwise performance differences between algorithms run on them*: if the same algorithms win, tie or lose against each other on two datasets, then the data distributions of these datasets are likely to be similar as well, at least in terms of their effect on learning performance. It is clear that the more algorithms are used, the more accurate the characterization

[1] LIAAD-INESC Porto L.A./Faculty of Economics, University of Porto, Portugal, rleite@fep.up.pt

[2] LIAAD-INESC Porto L.A./Faculty of Economics, University of Porto, Portugal, pbrazdil@inescporto.pt

[3] LIACS - Leiden Institute of Advanced Computer Science, University of Leiden, Nederlands, joaquin@liacs.nl

[4] In the remainder of this text, when we speak of *algorithms*, we mean *fully-defined algorithm instances* with fixed components (e.g., base-learners, kernel functions) and parameter settings.

will be. While we cannot run all algorithms on each new dataset because of the computational cost, we can run a fair amount of CV tests to get a reasonably good idea of which prior datasets are most similar to the new one.

Moreover, we can use these same performance results to establish which (yet untested) algorithms are likely to perform well on the new dataset, i.e., those algorithms that outperformed or rivaled the currently best algorithm on similar datasets in the past. As such, we can intelligently select the most promising algorithms for the new dataset, run them, and then use their performance results to gain increasingly better estimates of the most similar datasets and the most promising algorithms.

Key concepts. There are two key concepts used in this work. The first one is that of the *current best candidate algorithm* which may be challenged by other algorithms in the process of finding an even better candidate.

The second is the pairwise performance difference between two algorithms running on the same dataset, which we call *relative landmark*. A collection of such relative landmarks represents a history of previous 'duels' between two algorithms on prior datasets. The term itself originates from the study of landmarking algorithms: since absolute values for the performance of landmarkers vary a lot depending on the dataset, several types of *relative* landmarks have been proposed, which basically capture the relative performance difference between two algorithms [12]. In this paper, we extend the notion of relative landmarks to *all* (including non-simplified) classification algorithms.

The history of previous algorithm duels is used to select the most promising challenger for the current best candidate algorithm, namely the method that most convincingly outperformed or rivaled the current champion on prior datasets *similar* to the new dataset.

Approach. Given the current best algorithm and a history of relative landmarks (duels), we can start a tournament game in which, in each round, the current best algorithm is compared to the next, most promising contender. We select the most promising challenger as discussed above, and run a CV test with this algorithm. The winner becomes the new current best candidate, the loser is removed from consideration. We will discuss the exact procedure in Section 3.

We call this approach *active testing* (AT)[5], as it actively selects the most interesting CV tests instead of passively performing them one by one: in each iteration the *best competitor* is identified, which determines a new CV test to be carried out. Moreover, the same result will be used to further characterize the new dataset and more accurately estimate the similarity between the new dataset and all prior datasets.

Evaluation. By intelligently selecting the most promising algorithms to test on the new dataset, we can more quickly discover an algorithm that performs very well. Note that running a selection of algorithms is typically done anyway to find a suitable algorithm. Here, we optimize and automate this process using historical performance results of the candidate algorithms on prior datasets.

While we cannot possibly guarantee to return the absolute best algorithm without performing all possible CV tests, we can return an algorithm whose performance is either identical or very close to the truly best one. The difference between the two can be expressed in terms of a *loss*. Our aim is thus to *minimize* this loss using a *minimal number of tests*, and we will evaluate our technique as such.

In all, the research hypothesis that we intend to prove in this paper is: *Relative landmarks provide useful information on the similarity of datasets and can be used to efficiently predict the most promising algorithms to test on new datasets.*

We will test this hypothesis by running our active testing approach in a leave-one-out fashion on a large set of CV evaluations testing 292 algorithms on 76 datasets. The results show that our AT approach is indeed effective in finding very accurate algorithms with a limited number of tests.

We also present an adaptation of method AT that uses data characteristics to define the similarity of the datasets. Our purpose was to compare the relative landmark versus data measures approaches to select classification algorithms.

Roadmap. The remainder of this paper is organized as follows. First, we formulate the concepts of relative landmarks in Section 2 and active testing in Section 3. Next, Section 4 presents the empirical evaluation and Section 5 presents an overview of some work in other related areas. The final section presents conclusions and future work.

2 Relative Landmarks

In this section we formalize our definition of relative landmarks, and explain how are used to identify the most promising competitor for a currently best algorithm.

Given a set of classification algorithms and some new classification dataset d_{new}, the aim is to identify the potentially best algorithm for this task with respect to some given performance measure M (e.g., accuracy, AUC or rank). Let us represent the performance of algorithm a_i on dataset d_{new} as $M(a_i, d_{new})$. As such, we need to identify an algorithm $a*$, for which the performance measure is maximal, or $\forall a_i M(a*, d_{new}) \geq M(a_i, d_{new})$. The decision concerning $\geq$ (i.e. whether $a*$ is at least as good as a_i) may be established using either a statistical significance test or a simple comparison.

However, instead of searching exhaustively for $a*$, we aim to find a near-optimal algorithm, $\hat{a*}$, which has a high probability $P(M(\hat{a*}, d_{new}) \geq M(a_i, d_{new}))$ to be optimal, ideally close to 1.

As in other work that exploits metalearning, we assume that $\hat{a*}$ is likely better than a_i on dataset d_{new} if it was found to be better on a similar dataset d_j (for which we have performance estimates):

$$P(M(\hat{a*}, d_{new}) \geq M(a_i, d_{new})) \sim P(M(\hat{a*}, d_j) \geq M(a_i, d_j)) \tag{1}$$

The latter estimate can be maximized by going through all algorithms and identifying the algorithm $\hat{a*}$ that satisfies the $\geq$ constraint in a maximum number of cases. However, this requires that we know which datasets d_j are most similar to d_{new}. Since our definition of similarity requires CV tests to be run on d_{new}, but we cannot run all possible CV tests, we use an iterative approach in which we repeat this scan for $\hat{a*}$ in every round, using only the datasets d_j that seem most similar at that point, as dataset similarities are recalculated after every CV test.

Initially, having no information, we deem all datasets to be similar to d_{new}, so that $\hat{a*}$ will be the globally best algorithm over all prior datasets. We then call this algorithm the *current best algorithm* a_{best} and run a CV test to calculate its performance on d_{new}. Based on this, the dataset similarities are recalculated (see Section 3), yielding a possibly different set of datasets d_j. The best algorithm on this new

[5] Note that while the term 'active testing' is also used in the context of actively selected sampling landmarks [14], there is little or no relationship to the approach described here.

set becomes the *best competitor* a_k (different from a_{best}), calculated by counting the number of times that $M(a_k, d_j) > M(a_{best}, d_j)$, over all datasets d_j.

We can further refine this method by taking into account how large the performance differences are: the larger a difference was in the past, the higher chances are to obtain a large gain on the new dataset. This leads to the notion of relative landmarks RL, defined as:

$$RL(a_k, a_{best}, d_j) = i(M(a_k, d_j) > M(a_{best}, d_j)) * \\ *(M(a_k, d_j) - M(a_{best}, d_j)) \tag{2}$$

The function $i(test)$ returns 1 if the $test$ is true and 0 otherwise. As stated before, this can be a simple comparison or a statistical significance test that only returns 1 if a_k performs significantly better than a_{best} on d_j. The term RL thus expresses how much better a_k is, relative to a_{best}, on a dataset d_j. Experimental tests have shown that this approach yields much better results than simply counting the number of wins.

Up to now, we are assuming a dataset d_j to be either similar to d_{new} or not. A second refinement is to use a gradual (non-binary) measure of similarity $Sim(d_{new}, d_j)$ between datasets d_{new} and d_j. As such, we can weigh the performance difference between a_k and a_{best} on d_j by how similar d_j is to d_{new}. Indeed, the more similar the datasets, the more informative the performance difference is. As such, we aim to optimize the following criterion:

$$a_k = \arg\max_{a_i} \sum_{dj \in D} RL(a_i, a_{best}, d_j) * Sim(d_{new}, d_j)) \tag{3}$$

in which D is the set of all prior datasets d_j.

To calculate the similarity $Sim()$, we use the outcome of each CV test on d_{new} and compare it to the outcomes on d_j.

In each iteration, with each CV test, we obtain a new evaluation $M(a_i, d_{new})$, which is used to recalculate all similarities $Sim(d_{new}, d_j)$. In fact, we will compare four variants of $Sim()$, which are discussed in the next section. With this, we can recalculate equation 3 to find the next best competitor a_k.

3 Active Testing

In this section we describe the active testing (AT) approach, which proceeds according to the following steps:

1. Construct a global ranking of a given set of algorithms using performance information from past experiments (metadata).
2. Initiate the iterative process by assigning the top-ranked algorithm as a_{best} and obtain the performance of this algorithm on d_{new} using a CV test.
3. Find the most promising competitor a_k for a_{best} using relative landmarks and all previous CV tests on d_{new}.
4. Obtain the performance of a_k on d_{new} using a CV test and compare with a_{best}. Use the winner as the current best algorithm, and eliminate the losing algorithm.
5. Repeat the whole process starting with step 3 until a stopping criterium has been reached. Finally, output the current a_{best} as the overall winner.

Step 1 - Establish a Global Ranking of Algorithms. Before having run any CV tests, we have no information on the new dataset d_{new} to define which prior datasets are similar to it. As such, we naively assume that all prior datasets are similar. As such, we generate a global ranking of all algorithms using the performance results of all algorithms on all previous datasets, and choose the top-ranked algorithm as our initial candidate a_{best}. To illustrate this, we use a toy example involving 6 classification algorithms, with default parameter settings, from Weka [10] evaluated on 40 UCI datasets [1], a portion of which is shown in Table 1.

As said before, our approach is entirely independent from the exact evaluation measure used: the most appropriate measure can be chosen by the user in function of the specific data domain. In this example, we use *success rate (accuracy)*, but any other suitable measure of classifier performance, e.g. *AUC* (area under the ROC curve), precision, recall or F1 can be used as well.

Each accuracy figure shown in Table 1 represents the mean of 10 values obtained in 10-fold cross-validation. The ranks of the algorithms on each dataset are shown in parentheses next to the accuracy value. For instance, if we consider dataset $abalone$, algorithm MLP is attributed rank 1 as its accuracy is highest on this problem. The second rank is occupied by $LogD$, etc.

The last row in the table shows the *mean rank* of each algorithm, obtained by averaging over the ranks of each dataset: $R_{a_i} = \frac{1}{n}\sum_{d_j=1}^{n} R_{a_i,d_j}$, where R_{a_i,d_j} represents the rank of algorithm a_i on dataset d_j and n the number of datasets. This is a quite common procedure, often used in machine learning to assess how a particular algorithm compares to others [5].
The mean ranks permit us to obtain a global ranking of candidate algorithms, CA. In our case, $CA = \langle MLP, J48, JRip, LogD, IB1, NB \rangle$. It must be noted that such a ranking is not very informative in itself. For instance, statistical significance tests are needed to obtain a truthful ranking. Here, we only use this global ranking CA are a starting point for the iterative procedure, as explained next.

Step 2 - Identify the Current Best Algorithm. Indeed, global ranking CA permits us to identify the top-ranked algorithm as our initial best candidate algorithm a_{best}. In Table 1, $a_{best} = MLP$. This algorithm is then evaluated using a CV test to establish its performance on the new dataset d_{new}.

Step 3 - Identify the Most Promising Competitor. In the next step we identify a_k, the *best competitor* of a_{best}. To do this, all algorithms are considered one by one, except for a_{best} and the eliminated algorithms (see step 4).

For each algorithm we analyze the information of past experiments (meta-data) to calculate the relative landmarks, as outlined in the previous section. As equation 3 shows, for each a_k, we sum up all relative landmarks involving a_{best}, weighted by a measure of similarity between dataset d_j and the new dataset d_{new}. The algorithm a_k that achieves the highest value is the most promising competitor in this iteration. In case of a tie, the competitor that appears first in ranking CA is chosen.

To calculate $Sim(d_{new}, d_j)$, the similarity between d_j and d_{new}, we have explored six different variants, AT0, AT1, ATWs, ATx, ATdc, ATWs_k described below.

AT0 is a base-line method which ignores dataset similarity. It always returns a similarity value of 1 and so all datasets are considered similar. This means that the best competitor is determined by summing up the values of the relative landmarks.

AT1 method works as AT0 at the beginning, when no tests have been carried out on d_{new}. In all subsequent iterations, this

Table 1. Accuracies and ranks (in parentheses) of the algorithms 1-nearest neighbor (IB1), C4.5 (J48), RIPPER (JRip), LogisticDiscriminant (LogD), MultiLayerPerceptron (MLP) and naive Bayes (NB) on different datasets and their mean rank.

Datasets	IB1	J48	JRip	LogD	MLP	NB
abalone	.197 (5)	.218 (4)	.185 (6)	.259 (2)	.266 (1)	.237 (3)
acetylation	.844 (1)	.831 (2)	.829 (3)	.745 (5)	.609 (6)	.822 (4)
adult	.794 (6)	.861 (1)	.843 (3)	.850 (2)	.830 (5)	.834 (4)
...	...	...	...	...	...	...
Mean rank	4.05	2.73	3.17	3.74	2.54	4.78

method estimates dataset similarity using only the most recent CV test. Consider the algorithms listed in Table 1 and the ranking CA. Suppose we started with algorithm MLP as the current best candidate. Suppose also that in the next iteration $LogD$ was identified as the best competitor, and won from MLP in the CV test: ($M(LogD, d_{new}) > M(MLP, d_{new})$). Then, in the subsequent iteration, all prior datasets d_j satisfying the condition $M(LogD, d_j) > M(MLP, d_j)$ are considered similar to d_{new}. In general terms, suppose that the last test revealed that $M(a_k, d_{new}) > M(a_{best}, d_{new})$, then $Sim(d_{new}, d_j)$ is 1 if also $M(a_k, d_j) > M(a_{best}, d_j)$, and 0 otherwise. The similarity measure determines which RL's are taken into account when summing up their contributions to identify the next best competitor.

Another variant of AT1 could use the difference between $RL(a_k, a_{best}, d_{new})$ and $RL(a_k, a_{best}, d_j)$, normalized between 0 and 1, to obtain a real-valued (non-binary) similarity estimate $Sim(d_{new}, d_j)$. In other words, d_j is *more similar* to d_{new} if the relative performance difference between a_k and a_{best} is about as large on both d_j and d_{new}. We plan to investigate this in our future work.

ATWs is similar to AT1, but instead of only using the last test, it uses *all* CV tests carried out on the new dataset, and calculates the Laplace-corrected ratio of corresponding results. For instance, suppose we have conducted 3 tests on d_{new}, thus yielding 3 pairwise algorithm comparisons on d_{new}. Suppose that 2 tests had the same result on dataset d_j (i.e. $M(a_x, d_{new}) > M(a_y, d_{new})$ and $M(a_x, d_j) > M(a_y, d_j)$), then the frequency of occurrence is 2/3, which is adjusted by Laplace correction to obtain an estimate of probability $(2 + 1)/(3 + 2)$. As such, $Sim(d_{new}, d_j) = \frac{3}{5}$.

ATx is similar to ATWs, but requires that all pairwise comparisons yield the same outcome. In the example used above, it will return $Sim(d_{new}, d_j) = 1$ only if all three comparisons lead to same result on both datasets and 0 otherwise.

ATdc is similar to ATWs, but uses a different similarity function. The idea of using this variant was to test if data characteristics (e.g. *number of examples*) could provide better information to identify the most similar datasets (w.r.t. d_{new}). We define the similarity between two datasets d_{new} and d_j using the following expression:

$$Sim(d_{new}, d_j) = 1 - \frac{1}{|Z|} \sum_{z \in Z} \frac{|z(d_{new}) - z(d_j)|}{max(z) - min(z)} \tag{4}$$

The symbol z represents a generic data measure used to characterize the datasets ($z(d)$ is the value of characteristic z for dataset d). Z is the set of measures used.

ATWs_k is similar to ATWS, but only consider the k most similar datasets (those with highest Sim values). The similarity for all the other datasets are set to 0. The idea is to avoid the situation where the similarity values show very small variations, making unusefull the information in the relative landmarks.

Step 4 - Determine which of the Two Algorithms is Better. Having found a_k, we can now run a CV test and compare it with a_{best}. The winner (which may be either the current best algorithm or the competitor) is used as the new current best algorithm in the new round. The losing algorithm is eliminated from further consideration.

Step 5 - Repeat the Process and Check the Stopping Criteria. The whole process of identifying the best competitor (step 3) of a_{best} and determining which one of the two is better (step 4) is repeated until a stopping criterium has been reached. For instance, the process could be constrained to a fixed number of CV tests: considering the results presented further on in Section 4, it would be sufficient to run at most 20% of all possible CV tests. Alternatively, one could impose a fixed CPU time, thus returning the best algorithm in h hours, as in budgeted learning. In any case, until aborted, the method will keep choosing a new competitor in each round: there will always be a next best competitor. In this respect our system differs from, for instance, hill climbing approaches which can get stuck in a local minimum.

Discussion - Comparison with Active Learning: The term *active testing* was chosen because the approach shares some similarities with *active learning* [7]. The concern of both is to speed up the process of improving a given performance measure. In active learning, the goal is to select the most informative data point to be labeled next, so as to improve the predictive performance of a supervised learning algorithm with a minimum of (expensive) labelings. In active testing, the goal is to select the most informative CV test, so as to improve the prediction of the best algorithm on the new dataset with a minimum of (expensive) CV tests.

4 Empirical Evaluation

4.1 Evaluation Methodology and Experimental Set-up

The proposed method AT was evaluated using a *leave-one-out* method [18]. The experiments reported here involve D datasets and so the whole procedure was repeated D times. In each cycle, all performance results on one dataset were left out for testing and the results on the remaining $D - 1$ datasets were used as metadata to determine the best candidate algorithm.

This study involved 292 algorithms (algorithm-parameter combinations), which were extracted from the experiment database for machine learning (ExpDB) [11, 22]. This set includes many different algorithms from the Weka platform [10], which were varied by assigning different values to their most important parameters. It includes SMO (a support vector machine, SVM), MLP (Multilayer Perceptron), J48 (C4.5), and different types of ensembles, including RandomForest, Bagging and Boosting. Moreover, different

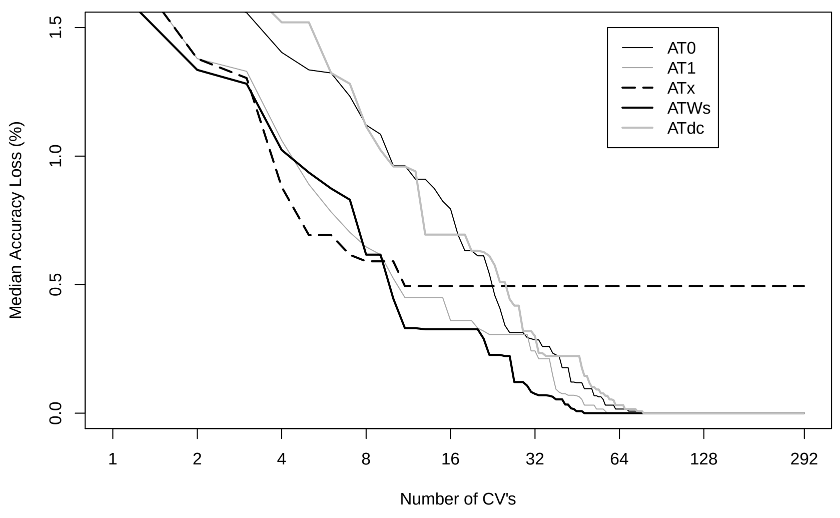


Figure 1. Median loss as a function of the number of CV tests.

SVM kernels were used with their own parameter ranges and all non-ensemble learners were used as base-learners for the ensemble learners mentioned above. The 76 datasets used in this study were all from UCI [1]. A complete overview of the data used in this study, including links to all algorithms and datasets can be found on `http://expdb.cs.kuleuven.be/ref/blv11`.

The data measures used to characterize the datasets for variant ATdc were *number of examples*, *proportion of nominal attributes*, *proportion of missing values*, *class entropy* and *mean mutual information*.

Regarding ATWs_k we set $k = 20$ (this value was set by exploring only the alternatives 30, 20 and 10).

The main aim of the test was to prove the research hypothesis formulated earlier: relative landmarks provide useful information for predicting the most promising algorithms on new datasets. Therefore, we also include two baseline methods:

TopN has been described before (e.g. [3]). It also builds a ranking of candidate algorithms as described in step 1 (although other measures different from mean rank could be used), and only runs CV tests on the first N algorithms. The overall winner is returned.

Rand simply selects N algorithms at random from the given set, evaluates them using CV and returns the one with the best performance. It is repeated 10 times with different random seeds and the results are averaged.

Since our AT methods are iterative, we will restart TopN and Rand N times, with N equal to the number of iterations (or CV tests).

To evaluate the performance of all approaches, we calculate the *loss* of the currently best algorithm, defined as $M(a_{best}, d_{new}) - M(a*, d_{new})$, where a_{best} represents the currently best algorithm, $a*$ the best possible algorithm and $M(.)$ represents the performance measure (success rate).

4.2 Results

By aggregating the results over D datasets, we can track the *median loss* of the recommended algorithm as a function of the number of CV tests carried out. The results are shown in Figure 1. Note that the number of CV tests is plotted on a logarithmic scale.

First, we see that ATWs and AT1 perform much better than AT0, which indicates that it is indeed useful to include dataset similarity. If we consider a particular level of loss (say 0.5%) we note that these variants require much fewer CV tests than AT0. The results also indicate that the information associated with relative landmarks obtained on the new dataset is indeed valuable. The median loss curves decline quite rapidly and are always below the AT0 curve. We also see that after only 10 CV tests (representing about 3% of all possible tests), the median loss is less than 0.5%. If we continue to 60 tests (about 20% of all possible tests) the median loss is near 0.

Also note that ATWs, which uses all relative landmarks involving a_{best} and d_{new}, does not perform much better than AT1, which only uses the most recent CV test. However in Figure 2 we show the performance of variant ATWs_k is much better than ATWs.

Method ATx, the most restrictive approach, only considers prior datasets on which *all* relative landmarks including a_{best} obtained similar results. As shown in Figure 1, this approach manages to reduce the loss quite rapidly, and competes well with the other variants in the initial region. However, after achieving a minimum loss in the order of 0.5%, there are no more datasets that fulfill this restriction, and consequently no new competitor can be chosen, causing it to stop. The other three methods, ATWs, ATdc and AT1, do not suffer from this shortcoming. We can see that the variant that uses data characteristics (ATdc) is generally worse than the other variants. Before the 23^{rd} CV test all variants AT1, ATx and ATWs need about half the CV tests needed by ATdc.

AT0 was also our best baseline method. To avoid overloading Figure 1, we show this separately in Figure 2. Indeed, AT0 is clearly better than the random choice method *Rand*. Comparing AT0 to

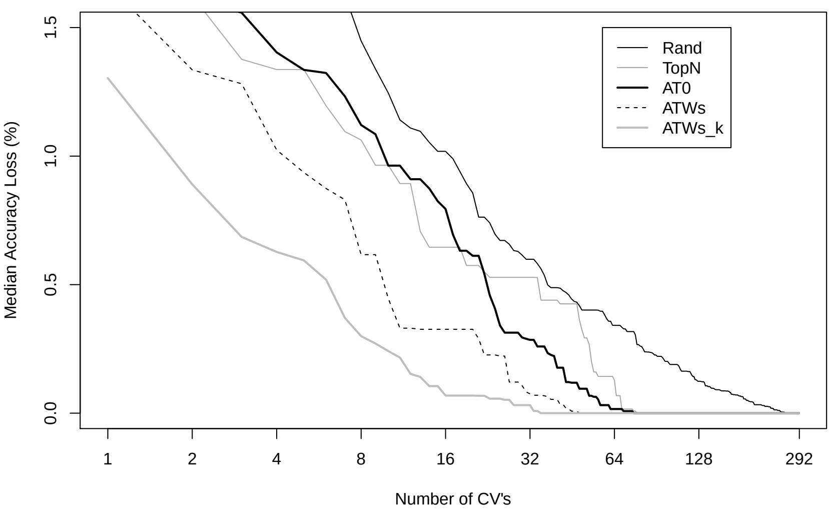


Figure 2. Median loss of AT0, ATWs, ATWs_k and the two baseline methods.

TopN, we cannot say that one is clearly better than the other overall, as the curves cross. However, it is clear that TopN looses out if we allow more CV tests, and that it is not competitive with the more advanced methods such as AT1, ATWs and ATWs_k. Is clear in Figure 2 that the best method variant is ATWs_k (using $k = 20$). We can see clearly in the curves that ATWs_k typically needs about half the CV tests needed by ATWs. The comparison with all the other variants is even more impressive.

The curves for mean loss (instead of median loss) follow similar trends, but the values are 1-2% worse due to outliers (see Fig. 3 relative to method ATWs_k). Besides, this figure shows also the curves associated with quartiles of 25% and 75% for ATWs_k. As the number of CV tests increases, the distance between the two curves decreases and approaches the median curve. Similar behavior has been observed for ATWs and AT1 but we skip the curves in this text.

Algorithm trace. It is interesting to trace the iterations carried out for one particular dataset. Table 2 shows the details for method AT1, where $abalone$ represents the new dataset. Column 1 shows the number of the iteration (thus the number of CV tests). Column 2 shows the most promising competitor a_k chosen in each step. Column 3 shows the index of a_k in our initial ranking CA, and column 4 the index of a_{best}, the *new best* algorithm after the CV test has been performed. As such, if the values in column 3 and 4 are the same, then the most promising competitor has won the duel. For instance, in step 2, $SMO.C.1.0.Polynomial.E.3$, i.e. SVM with complexity constant set to 1.0 and a 3rd degree polynomial kernel, (index 96) has been identified as the best competitor to be used (column 2), and after the CV test, it has won against $Bagging.I.75..100.PART$, i.e. Bagging with a high number of iterations (between 75 and 100) and PART as a base-learner. As such, it wins this round and becomes the new a_{best}. Columns 5 and 6 show the *actual* rank of the competitor and the winner on the $abalone$ dataset. Column 7 shows the loss compared to the optimal algorithm and the final column shows the number of datasets whose similarity measure is 1.

We observe that after only 12 CV tests, the method has identified an algorithm with a very small loss of 0.2%: $Bagging.I.25..50.MultilayerPerceptron$, i.e. Bagging with relatively few iterations but with a MultiLayerPerceptron base-learner.

Incidentally, this dataset appears to represent a quite atypical problem: the truly best algorithm, $SMO.C.1.0.RBF.G.20$, i.e. SVM with an RBF kernel with kernel width (gamma) set to 20, is ranked globally as algorithm 246 (of all 292). AT1 identifies it after 177 CV tests.

5 Related Work in Other Scientific Areas

In this section we briefly cover some work in other scientific areas which is related to the problem tackled here and could provide further insight into how to improve the method.

One particular area is *experiment design* [6] and in particular *active learning*. As discussed before, the method described here follows the main trends that have been outlined in this literature. However, there is relatively little work on active learning for ranking tasks. One notable exception is [15], who use the notion of *Expected Loss Optimization (ELO)*. Another work in this area is [4], whose aim was to identify the most interesting substances for drug screening using a minimum number of tests. In the experiments described, the authors have focused on the top-10 substances. Several different strategies were considered and evaluated. Our problem here is not ranking, but rather simply finding the best item (algorithm), so this work is only partially relevant.

Another relevant area is the so called *multi-armed bandit problem (MAB)* studied in statistics and machine learning [9, 16]. This problem is often formulated in a setting that involves a set of traditional slot machines. When a particular lever is pulled, a reward is provided from a distribution associated with that specific lever. The bandit problem is formally equivalent to a one-state Markov decision process. The aim is to minimize *regret* after T rounds, which is defined as a difference between the reward sum associated with an optimal strategy and the sum of collected rewards. Indeed, pulling

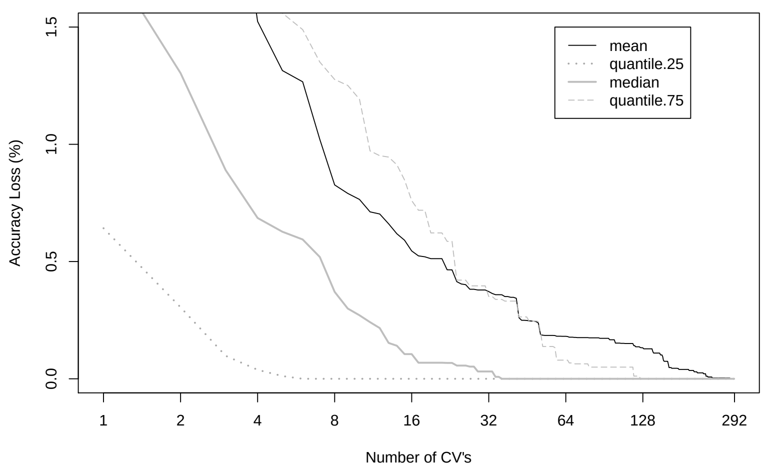


Figure 3. Loss of ATWs_k (k=20) as a function of the number of CV tests.

Table 2. Trace of the steps taken by AT1 in the search for the supposedly best algorithm for the *abalone* dataset

CV test	Algorithm used (current best competitor, a_k)	CA a_k	CA new a_{best}	abalone a_k	abalone new a_{best}	Loss (%)	D size
1	Bagging.I.75..100.PART	1	1	75	75	1.9	75
2	SMO.C.1.0.Polynomial.E.3	96	96	56	56	1.6	29
3	AdaBoostM1.I.10.MultilayerPerceptron	92	92	47	47	1.5	34
4	Bagging.I.50..75.RandomForest	15	92	66	47	1.5	27
...	...	...	...	...	...	...	...
10	LMT	6	6	32	32	1.1	45
11	LogitBoost.I.10.DecisionStump	81	6	70	32	1.1	51
12	Bagging.I.25..50.MultilayerPerceptron	12	12	2	2	0.2	37
13	LogitBoost.I.160.DecisionStump	54	12	91	2	0.2	42
...	...	...	...	...	...	...	...
177	SMO.C.1.0.RBF.G.20	246	246	1	1	0	9

a lever can be compared to carrying out a CV test on a given algorithm. However, there is one fundamental difference between MAB and our setting: whereas in MAB the aim is to maximize the sum of collected rewards, our aim it to maximize *one* reward, i.e. the reward associated with identifying the best algorithm. So again, this work is only partially relevant.

To the best of our knowledge, no other work in this area has addressed the issue of how to select a suitable algorithm from a large set of candidates.

6 Significance and Impact

In this paper we have addressed the problem of selecting the best classification algorithm for a specific task. We have introduced a new method, called *active testing*, that exploits information concerning past evaluation results (metadata), to recommend the best algorithm using a limited number of tests on the new dataset.

Starting from an initial ranking of algorithms on previous datasets, the method runs additional CV evaluations to test several competing algorithms on the new dataset. However, the aim is to reduce the number of tests to a minimum. This is done by carefully selecting which tests should be carried out, using the information of both past and present algorithm evaluations represented in the form of relative landmarks.

In our view this method incorporates several innovative features. First, it is an iterative process that uses the information in each CV test to find the most promising next test based on a history of prior 'algorithm duels'. In a tournament-style fashion, it starts with a current best (parameterized) algorithm, selects the most promising rival algorithm in each round, evaluates it on the given problem, and eliminates the algorithm that performs worse. Second, it continually focuses on the most similar prior datasets: those where the algorithm duels had a similar outcome to those on the new dataset.

Four variants of this basic approach, differing in their definition of dataset similarity, were investigated in a very extensive experiment setup involving 292 algorithm-parameter combinations on 76 datasets. Our experimental results show that particularly versions ATWs and AT1 provide good recommendations using a small number of CV tests. When plotting the median loss as a function of the number of CV tests (Fig. 1), it shows that both outperform all other variants and baseline methods. They also outperform AT0, indicating that dataset similarity is an important aspect.

We also see that after only 10 CV tests (representing about 3% of all possible tests), the median loss is less than 0.5%. If we continue to 60 tests (about 20% of all possible tests) the median loss is near 0. Similar trends can be observed when considering mean loss.

The results support the hypothesis that we have formulated at the outset of our work, that relative landmarks are indeed informative and can be used to suggest the best contender. If this is procedure is used iteratively, it can be used to accurately recommend a classification algorithm after a very limited number of CV tests.

Still, we believe that the results could be improved further. Classical information-theoretic measures and/or sampling landmarks could be incorporated into the process of identifying the most similar datasets. This could lead to further improvements and forms part of our future plans.

ACKNOWLEDGEMENTS

This work is funded (or part-funded) by the ERDF European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP - 01-0124-FEDER-022701.

REFERENCES

[1] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[2] B.Pfahringer, H.Bensussan, and C. Giraud-Carrier, 'Meta-learning by landmarking various learning algorithms', in *Proceedings of the 17th Int. Conf. on Machine Learning (ICML-2000*, Stanford,CA, (2000).

[3] P. Brazdil, C. Soares, and J. Costa, 'Ranking learning algorithms: Using IBL and meta-learning on accuracy and time results', *Machine Learning*, **50**, 251–277, (2003).

[4] K. De Grave, J. Ramon, and L. De Raedt, 'Active learning for primary drug screening', in *Proceedings of Discovery Science*. Springer, (2008).

[5] J. Demsar, 'Statistical comparisons of classifiers over multiple data sets', *The Journal of Machine Learning Research*, **7**, 1–30, (2006).

[6] V. Fedorov, *Theory of Optimal Experiments*, Academic Press, New York, 1972.

[7] Y. Freund, H. Seung, E. Shamir, and N. Tishby, 'Selective sampling using the query by committee algorithm', *Machine Learning*, **28**, 133–168, (1997).

[8] Johannes Fürnkranz and Johann Petrak, 'An evaluation of landmarking variants', in *Proceedings of the ECML/PKDD Workshop on Integrating Aspects of Data Mining, Decision Support and Meta-Learning (IDDM-2001)*, pp. 57–68. Springer, (2001).

[9] J. Gittins, 'Multi-armed bandit allocation indices', in *Wiley Interscience Series in Systems and Optimization*, John Wiley & Sons, Ltd., (1989).

[10] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten, 'The WEKA data mining software: an update', *SIGKDD Explor. Newsl.*, **11**(1), 10–18, (2009).

[11] H.Blockeel, 'Experiment databases: A novel methodology for experimental research', in *Leacture Notes on Computer Science 3933*. Springer, (2006).

[12] J.Fürnkranz and J.Petrak, 'An evaluation of landmarking variants', in *Working Notes of ECML/PKDD 2000 Workshop on Integration Aspects of Data Mining, Decision Support and Meta-Learning*, eds., C.Carrier, N.Lavrac, and S.Moyle, (2001).

[13] R. Leite and P. Brazdil, 'Predicting relative performance of classifiers from samples', in *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pp. 497–503, New York, NY, USA, (2005). ACM Press.

[14] Rui Leite and Pavel Brazdil, 'Active testing strategy to predict the best classification algorithm via sampling and metalearning', in *Proceedings of the 19th European Conference on Artificial Intelligence - ECAI 2010*, (2010).

[15] B. Long, O. Chapelle, Y. Zhang, Y. Chang, Z. Zheng, and B. Tseng, 'Active learning for rankings through expected loss optimization', in *Proceedings of the SIGIR'10*. ACM, (2010).

[16] A. Mahajan and D. Teneketzis, 'Multi-armed bandit problems', in *Foundations and Applications of Sensor Management*, eds., D. A. Castanon, D. Cochran, and K. Kastella, Springer-Verlag, (2007).

[17] D. Michie, D.J.Spiegelhalter, and C.C.Taylor, *Machine Learning, Neural and Statistical Classification*, Ellis Horwood, 1994.

[18] Tom M. Mitchell, *Machine Learning*, McGraw-Hill, New York, 1997.

[19] John R. Rice, 'The algorithm selection problem', volume 15 of *Advances in Computers*, 65 – 118, Elsevier, (1976).

[20] Kate A. Smith-Miles, 'Cross-disciplinary perspectives on meta-learning for algorithm selection', *ACM Comput. Surv.*, **41**(1), 1–25, (2008).

[21] Carlos Soares, Johann Petrak, and Pavel Brazdil, 'Sampling-based relative landmarks: Systematically test-driving algorithms before choosing', in *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001)*, pp. 88–94. Springer, (2001).

[22] J. Vanschoren and H. Blockeel, 'A community-based platform for machine learning experimentation', in *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2009*, volume LNCS 5782, pp. 750–754. Springer, (2009).

[23] Ricardo Vilalta and Youssef Drissi, 'A perspective view and survey of meta-learning', *Artif. Intell. Rev.*, **18**(2), 77–95, (2002).

Predicting the Accuracy of Regression Models in the Retail Industry[1]

Fábio Pinto[2] and **Carlos Soares**[3]

Abstract. Companies are moving from developing a single model for a problem (e.g., a regression model to predict general sales) to developing several models for sub-problems of the original problem (e.g., regression models to predict sales of each of its product categories). Given the similarity between the sub-problems, the process of model development should not be independent. Information should be shared between processes. Different approaches can be used for that purpose, including metalearning (MtL) and transfer learning. In this work, we use MtL to predict the performance of a model based on the performance of models that were previously developed. Given that the sub-problems are related (e.g., the schemas of the data are the same), domain knowledge is used to develop the metafeatures that characterize them. The approach is applied to the development of models to predict sales of different product categories in a retail company from Portugal.

1 Introduction

The retail industry is a world of extreme competitiveness. Companies struggle on a daily basis for the loyalty of their clients through diverse marketing actions, while providing better products, better prices and better services. The growing need for analytic tools that enhance retailers performance is unquestionable, and Data Mining (DM) is central in this trend [2].

Sales prediction is one of the main tasks in retail. The ability to assess the impact that a sudden change in a particular factor will have on the sales of one or more products is a major tool for retailers. DM is one of the approaches for this task.

In early approaches to predict sales, a single model could be used for a whole business. As more detailed data becomes available, retail companies are dividing the problem into several sub-problems (predict the sales of each of its stores or product categories). The same trend can be observed in several industries [6].

In this approach, there are obviously many similarities between the sub-problems. Not only is the structure of the data typically the same across all sub-problems (e.g., the variables are the same and their domains are similar) but also the patterns in that data may have similarities (e.g., the most important variables across different problems are also similar). Therefore, the process of model building should not be independent. The knowledge obtained from generating a model for one sub-problem can and should be applied to the process of developing the model for the other sub-problems. Different approaches can be used for that purpose, two of them being MtL and transfer learning [1].

Our goals with this work is to use metalearning (MtL) to predict the performance of one model based on the performance of models that were previously developed to predict sales of product categories in a retail company in Portugal, and unveil the attributes that are more important for that prediction. The paper is organised as follows. In Section 2, we briefly survey the concept of MtL and the importance of metafeatures. Our case study is presented in Section 3. Finally, in Section 4 we expose some conclusions.

2 Metafeatures for Metalearning

MtL can be defined as the use of data about the performance of machine learning algorithms on previous problems to predict their performance on future ones [1]. For more information on MtL, we refer the reader to [5, 1].

One of the essential issues about MtL are the metafeatures that characterize the problem. Which metafeatures contain useful information to predict the performance of an algorithm on a given problem? Much work has been done on this topic (e.g., [3]). Typically the work on MtL includes problems from different domains, so the metafeatures need to be very generic (e.g., number of attributes and mutual information between symbolic attributes and the target). However, in more specific settings, metafeatures should encode more particular information about the data, which probably contain useful information about the performance of the algorithms.

3 Case Study

The base-level data used in this study was collected to model monthly sales by product category in a Portuguese retail company. We also gather 9 variables that describe store layout, store profile, client profile and seasonality. Six regression methods from R packages were tested: Cubist, NN, SVM, Generalized Boosted Regression, MARS and Random Forests (RF). The DM algorithm with the most robust performance was RF.[4] The models for the 89 categories were evaluated using the mean percentage error (Eq. 1) where f_i is the predicted value and a_i the real value.

[1] This work is partially funded by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project "FCOMP - 01-0124-FEDER-022701"

[2] Faculdade de Economia, Universidade do Porto, Portugal, email: fabiohscpinto@gmail.com

[3] INESC TEC/Faculdade de Economia, Universidade do Porto, Portugal, email: csoares@fep.up.pt

[4] The *randomForest* package was used to fit RF models.

$$MPE = \frac{\sum_{i=1}^{n} |\frac{f_i - a_i}{a_i}|}{n} \quad (1)$$

The estimates were obtained using a sliding window approach where the base-level data spans two years. For the majority of the models, one and a half year (approximately 75% of the data) was used as training set and the remaining half year was used as test set. For categories containing just one year of data, the first 9 months were used as training set and the remaining instances as test set. The results are summarized in Table 1.

Table 1. Summary of base-level results (MPE)

Min.	1st Q.	Median	Mean	3rd Q.	Max.
0.072	0.091	0.116	0.212	0.211	1.209

Modelling the variance in results across different product categories is important not only to predict the performance of models but also to understand it. A better understanding of the factors affecting the performance of the algorithm may lead us to better results. For that purpose, we used a MtL approach with the following problem-specific metafeatures:

- Number of instances
- Type of sliding window
- Variables that capture the amount of variation in store layout
- Variables that capture the diversity of store profile in the data
- Variables that capture the amplitude[5] of sales in the test set
- Variables that capture the amplitude of store layout attributes in the training and test sets

The metadata contained 89 examples (corresponding to the 89 categories) described by 14 predictors. The meta-level error of RF for regression was estimated using 10-fold cross-validation.[6] The number of trees was set to 500, as improvement in performance was not found for larger values; and 3 values where tested for the m_{try} parameter, which controls the number of variables randomly sampled as candidates at each split [4]. The results are summarized in Table 2.

Table 2. Meta-level results obtained with all metafeatures in terms of Relative Mean Squared Error and the R^2 and their standard-deviations (SD).

m_{try}	RMSE	R^2	RMSE SD	R^2 SD
2	0.148	0.66	0.0847	0.215
8	0.146	0.663	0.0877	0.234
14	0.15	0.65	0.0871	0.224

The best results were obtained with the m_{try} parameter set to 8. Even with a standard deviation of 0.23, a value of 0.66 for R^2 gives us confidence about the capacity of the regression metamodel in predicting the performance of future RF models.

The next step is to identify the metafeatures that contributed the most to this result. The RF implementation that we used includes a function to measure the importance of predictors in the classification/regression model. We applied the algorithm on all 89 instances.

[5] We calculate "amplitude" of a variable by dividing the largest value by the smallest.

[6] The package *caret* for R was used for 10-fold cross validation estimation. Size of each sample: 81, 80, 80, 81, 80, 79, 80, 80, 79 and 81.

Table 3. Importance of Variables.

Rank	Variable
1st	Max(sales)/Min(sales) in training set
2nd	Mean number of changes in shelf size of category
3rd	Number of instances

The R^2 obtained was 0.93. The importance of the variables for this model is summarized in Table 3. These results show evidence that the metafeatures that represent the amplitude of the dependent variable in the training set and the amount of variance in data are very informative for predicting the accuracy of regression models.

Finally, we used these results for meta-level variable selection. We re-executed the meta-level experiments, again estimating the performance of the RF for regression with 10-fold cross-validation, with only the three most informative metafeatures. The results are summarized in Table 4.

Table 4. Meta-level results obtained with three selected metafeatures in terms of Relative Mean Squared Error and the R^2 and their standard-deviations (SD).

m_{try}	RMSE	R^2	RMSE SD	R^2 SD
2	0.138	0.678	0.0552	0.234
3	0.14	0.687	0.0571	0.23

The results shown in Table 4 are even better than those obtained previously. However, they must be interpreted carefully, as the same dataset was used to do metafeature selection and test its effectiveness, thus increasing the potential for overfitting. Nevertheless, this evidence let us believe that some of the metafeatures used before were carrying noise and that results can improve by metafeature selection.

4 Conclusions

We successfully used MtL with domain-specific metafeatures to predict the accuracy of regression models in predicting sales by product category in the retail industry. Our work shows evidence that, when possible, using domain knowledge to design metafeatures is advantageous.

We plan to extend this approach to predict the performance of multiple algorithms. Additionally, we will compare these results with the results of MtL using traditional metafeatures.

REFERENCES

[1] Pavel Brazdil, Christophe Giraud-Carrier, Carlos Soares, and Ricardo Vilalta, *Metalearning: Applications to Data Mining*, Cognitive Technologies, Springer, Berlin, Heidelberg, 2009.

[2] Thomas H Davenport, 'Realizing the Potential of Retail Analytics', *Working Knowledge Research Report*, (2009).

[3] Alexandros Kalousis, João Gama, and Melanie Hilario, 'On Data and Algorithms: Understanding Inductive Performance', *Machine Learning*, **54**(3), 275–312, (2004).

[4] A Liaw and M Wiener, 'Classification and Regression by randomForest', *R News*, **II/III**, 18–22, (2002).

[5] F. Serban, J. Vanschoren, J.U. Kietz, and A. Bernstein, 'A survey of intelligent assistants for data analysis', *ACM Computing Surveys*, (2012). in press.

[6] Françoise Soulié-Fogelman, 'Data Mining in the real world: What do we need and what do we have?', in *Proceedings of the KDD Workshop on Data Mining for Business Applications*, eds., R Ghani and C Soares, pp. 44–48, (2006).

The experiment database for machine learning (Demo)

Joaquin Vanschoren[1]

Abstract. We demonstrate the use of the experiment database for machine learning, a community-based platform for the sharing, reuse, and in-depth investigation of the thousands of machine learning experiments executed every day. It is aimed at researchers and practitioners of data mining techniques, and is publicly available at `http://expdb.cs.kuleuven.be`. This demo gives a hands-on overview of how to share novel experimental results, how to integrate the database in existing data mining toolboxes, and how to query the database through an intuitive graphical query interface.

1 Introduction

Experimentation is the lifeblood of machine learning (ML) research. A considerable amount of effort and resources are invested in assessing the usefulness of new algorithms, finding the optimal approach for new applications or just to gain some insight into, for instance, the effect of a parameter. Yet in spite of all these efforts, experimental results are often discarded or forgotten shortly after they are obtained, or at best averaged out to be published, which again limits their future use. If we could collect all these ML experiments in a central resource and make them publicly available in an organized (searchable) fashion, the combined results would provide a highly detailed picture of the performance of algorithms on a wide range of data configurations, speeding up ML research.

In this paper, we demonstrate a community-based platform designed to do just this: the *experiment database for machine learning*. First, experiments are automatically transcribed in a common language that captures the exact experiment setup and all details needed to reproduce them. Then, they are uploaded to pre-designed databases where they are stored in an organized fashion: the results of every experiment are linked to the exact underlying components (such as the algorithm, parameter settings and dataset used) and thus also integrated with all prior results. Finally, to answer any question about algorithm behavior, we only have to write a query to the database to sift through millions of experiments and retrieve all results of interest. As we shall demonstrate, many kinds of questions can be answered in one or perhaps a few queries, thus enabling fast and thorough analysis of large numbers of collected results. The results can also be interpreted unambiguously, as all conditions under which they are valid are explicitly stored.

1.1 Meta-learning

Instead of being purely empirical, these experiment databases also store known or measurable properties of datasets and algorithms. For datasets, this can include the number of features, statistical and information-theoretic properties [7] and landmarkers [10], while algorithms can be tagged by model properties, the average ratio of bias or variance error, or their sensitivity to noise [3].

As such, all *empirical* results, past and present, are immediately linked to all known *theoretical* properties of algorithms and datasets, providing new grounds for deeper analysis. For instance, algorithm designers can include these properties in queries to gain precise insights on how their algorithms are affected by certain kinds of data or how they relate to other algorithms.

1.2 Overview of benefits

We can summarize the benefits of this platform as follows:

Reproducibility The database stores all details of the experimental setup, resulting in truly reproducible research.

Reference All experiments, including algorithms and datasets, are automatically organized in one resource, creating an overview of the state-of-the-art, and a useful 'map' of all known approaches, their properties, and their performance. This also includes *negative results*, which usually do not get published.

Querying When faced with a question on the performance of learning algorithms, e.g., 'What is the effect of the training set size on runtime?', we can answer it in seconds by writing a query, instead of spending days (or weeks) setting up new experiments. Moreover, we can draw upon many more experiments, on many more algorithms and datasets, than we can afford to run ourselves.

Reuse It saves time and energy, as previous experiments can be readily reused. For instance, when benchmarking a new algorithm, there is no need to benchmark the older algorithms over and over again as well: their evaluations are likely stored online, and can simply be downloaded.

Larger studies Studies covering many algorithms, parameter settings and datasets are very expensive to run, but could become much more feasible if a large portion of the necessary experiments are available online. Even when all the experiments have yet to be run, the automatic storage and organization of experimental results markedly simplifies conducting such large scale experimentation and thorough analysis thereof.

Visibility By using the database, users may learn about (new) algorithms they were not previously aware of.

Standardization The formal description of experiments may catalyze the standardization of experiment design, execution and exchange across labs and data mining tools.

The remainder of this paper is organized as follows. Sect. 2 outlines how we constructed our pilot experiment database and the underlying models and languages that enable the free exchange of experiments. In Sect. 3, we demonstrate how it can be used to quickly discover new insights into a wide range of research questions and to verify prior studies. Sect. 4 concludes.

[1] LIACS, Leiden University, The Netherlands, email: joaquin@liacs.nl

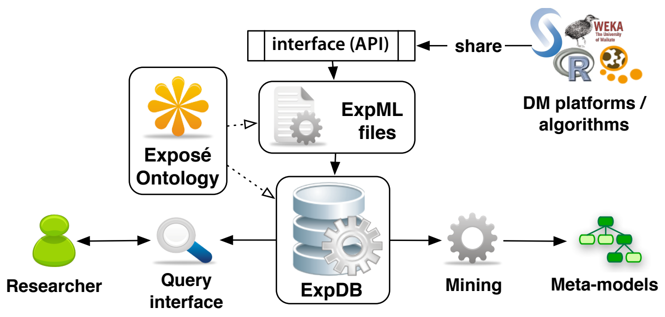


Figure 1. Components of the experiment database framework.

2 Framework description

In this section, we outline the design of this collaborative framework, outlined in Fig. 1. We first establish a controlled vocabulary for data mining experimentation in the form of an open ontology (Exposé), before mapping it to an experiment description language (called ExpML) and an experiment database (ExpDB). These three elements (boxed in Fig. 1) will be discussed in the next three subsections. Full versions of the ontologies, languages and database models discussed below will be available on `http://expdb.cs.kuleuven.be`.

Experiments are shared (see Fig. 1) by entering all experiment setup details and results through the framework's interface (API), which exports them as ExpML files or directly streams them to an ExpDB. Any data mining platform or custom algorithm can thus use this API to add a 'sharing' feature that publishes new experiments. The ExpDB can be set up locally, e.g., for a single person or a single lab, or globally, a central database open to submissions from all over the world. Finally, the bottom of the figure shows different ways to tap into the stored information:

Querying. Querying interfaces allow researchers to formulate questions about the stored experiments, and immediately get all results of interest. We currently offer various such interfaces, including graphical ones (see Sect. 2.3.2).

Mining. A second use is to automatically look for patterns in algorithm performance by mining the stored evaluation results and theoretical meta-data. These *meta-models* can then be used, for instance, in algorithm recommendation [1].

2.1 The Exposé Ontology

The Exposé ontology describes the concepts and the structure of data mining experiments. It establishes an unambiguous and machine-interpretable (semantic) vocabulary, through which experiments can be automatically shared, organized and queried. We will also use it to define a common experiment description language and database models, as we shall illustrate below. Ontologies can be easily extended and refined, which is a key concern since data mining and machine learning are ever-expanding fields.

2.1.1 Collaborative Ontology Design

Several other useful ontologies are being developed in parallel: OntoDM [8] is a top-level ontology for data mining concepts, EXPO [11] models scientific experiments, DMOP [4] describes learning algorithms (including their internal mechanisms and models) and workflows, and the KD ontology [13] and eProPlan ontology [5] describe large arrays of DM operators, including information about their use to support automatic workflow planning.

To streamline ontology development, a 'core' ontology was defined, and an open ontology development forum was created: the Data Mining Ontology (DMO) Foundry[2]. The goal is to make the ontologies interoperable and orthogonal, each focusing on a particular aspect of the data mining field. Moreover, following best practices in ontology engineering, we reuse concepts and relationships from established top-level scientific ontologies: BFO,[3] OBI,[4] IAO,[5] and RO.[6] We often use subproperties, e.g. *implements* for *concretizes*, and *runs* for *realizes*, to reflect common usage in the field. Exposé is designed to integrate or be similar to the above mentioned ontologies, but focusses on aspects related to experimental evaluation.

2.1.2 Top-level View

Fig. 2 shows Exposé's high-level concepts and relationships. The full arrows symbolize *is-a* relationships, meaning that the first concept is a subclass of the second, and the dashed arrows symbolize other common relationships. The most top-level concepts are reused from the aforementioned top-level scientific ontologies, and help to describe the exact semantics of many data mining concepts. For instance, when speaking of a 'data mining algorithm', we can semantically distinguish an abstract *algorithm* (e.g., C4.5 in pseudo-code), a concrete *algorithm implementation* (e.g., WEKA's `J48` implementation of C4.5), and a specific *algorithm setup*, including *parameter settings* and subcomponent setups. The latter may include other algorithm setups, e.g. for base-learners in ensemble algorithms, as well as mathematical *functions* such as kernels, distance functions and evaluation measures. A *function setup* details the implementation and parameter settings used to evaluate the function.

An algorithm setup thus defines a deterministic function which can be directly linked to a specific result: it can be *run* on a *machine* given specific input data (e.g., a *dataset*), and produce specific output data (e.g., new datasets, *models* or *evaluations*). As such, we can trace any output result back to the inputs and processes that generated it (data provenance). For instance, we can query for evaluation results, and link them to the specific algorithm, implementation or individual parameter settings used, as well as the exact input data.

Algorithm setups can be combined in *workflows*, which additionally describe how data is passed between multiple algorithms. Workflows are hierarchical: they can contain sub-workflows, and algorithm setups themselves can contain internal workflows (e.g., a cross-validation setup may define a workflow to train and evaluate learning algorithms). The level of detail is chosen by the author of an experiment: a simple experiment may require a single algorithm setup, while others involve complex scientific workflows.

Tasks cover different data mining (sub)tasks, e.g., supervised classification. *Qualities* are known or measurable properties of algorithms and datasets (see Sect. 1.1), which are useful to interpret results afterwards. Finally, algorithms, functions or parameters can play certain *roles* in a complex setup: an algorithm can sometimes act as a *base-learner* in an ensemble algorithm, and a dataset can act as a *training set* in one experiment and as a *test set* in the next.

[2] The DMO Foundry: `http://dmo-foundry.org`
[3] The Basic Formal Ontology (BFO): `http://www.ifomis.org/bfo`
[4] The Ontology for Biomedical Investigations (OBI): `http://obi-ontology.org`
[5] The Information Artifact Ontology (IAO): `http://bioportal.bioontology.org/ontologies/40642`
[6] The Relation Ontology (RO): `http://www.obofoundry.org/ro`

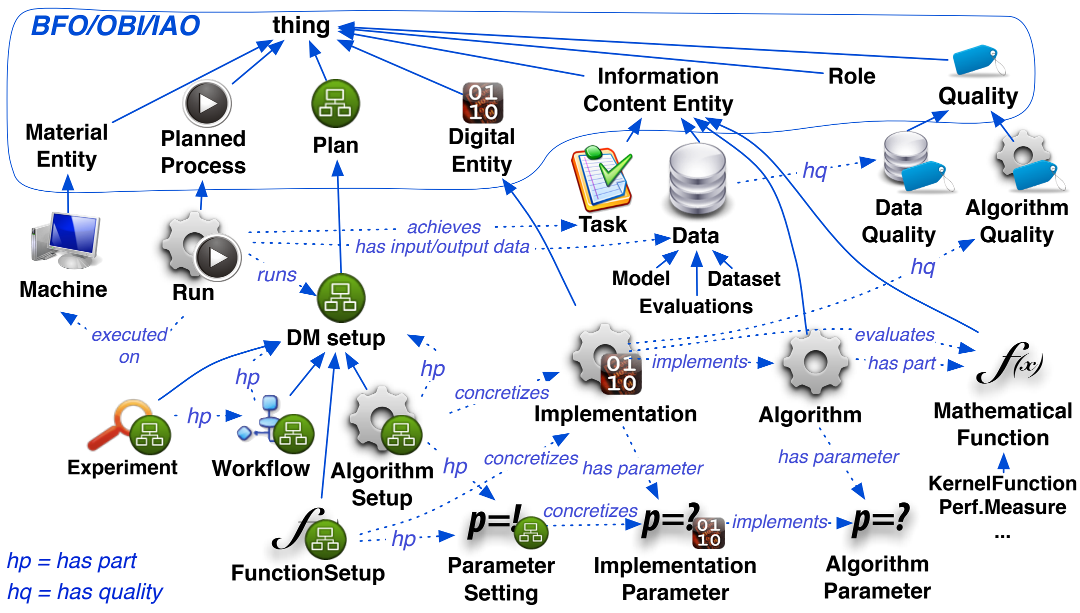


Figure 2. An overview of the top-level concepts in the Exposé ontology.

2.1.3 Experiments

An *experiment* tries to answer a question (in exploratory settings) or test a hypothesis by assigning certain values to these input variables. It has *experimental variables*: *independent variables* with a range of possible values, *controlled variables* with a single value, or *dependent variables*, i.e., a monitored output. The *experiment design* (e.g., *full factorial*) defines which combinations of input values are used.

One experiment run may generate several workflow runs (with different input values), and a workflow run may consist of smaller algorithm runs. *Runs* are triples consisting of input data, a setup and output data. Any sub-runs, such as the 10 algorithm runs within a 10-fold CV run, could also be stored with the exact input data (folds) and output data (predictions). Again, the level of detail is chosen by the experimenter. Especially for complex workflows, it might be interesting to afterwards query the results of certain sub-runs.

2.2 ExpML: A Common Language

Returning to our framework in Fig. 1, we now use this ontology to define a common language to describe experiments. The most straightforward way to do this would be to describe experiments in Exposé, export them in RDF[7] and store everything in RDF databases (triplestores). However, such databases are still under active development, and many researchers are more familiar with XML and relational databases, which are also widely supported by many current data mining tools. Therefore, we will also map the ontology to a simple XML-based language, ExpML, and a relational database schema. Technical details of this mapping are outside the scope of this paper. Below, we show a small example of ExpML output to illustrate our modeling of data mining workflows.

[7] Resource Description Framework: http://www.w3.org/RDF

2.2.1 Workflow Runs

Fig. 3 shows a *workflow run* in ExpML, executed in WEKA [2] and exported through the aforementioned API, and a schematic representation is shown in Fig. 4. The workflow has two inputs: a dataset URL and parameter settings. It also contains two algorithm setups: the first loads a dataset from the given URL, and then passes it to a cross-validation setup (10 folds, random seed 1). The latter evaluates a Support Vector Machine (SVM) implementation, using the given parameter settings, and outputs evaluations and predictions. Note that the workflow is completely concretized: all parameter settings and implementations are fixed. The bottom of Figure 3 shows the workflow run and its two algorithm sub-runs, each pointing to the setup used. Here, we chose not to output the 10 per-fold SVM runs.

The final output consists of *Evaluations* and *Predictions*. As shown in the ExpML code, these have a predefined structure so that they can be automatically interpreted and organized. Evaluations contain, for each evaluation function (as defined in Exposé), the evaluation value and standard deviation. They can also be labeled, as for the per-class precision results. Predictions can be probabilistic, with a probability for each class, and a final prediction for each instance. For storing models, we can use existing formats such as PMML.

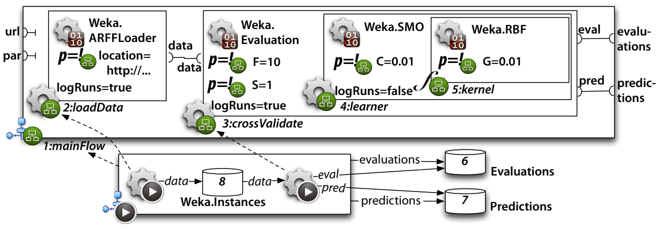


Figure 4. A schematic representation of the run.

```
<Run machine=" " timestamp=" " author=" ">
 <Workflow id="1:mainflow" template="10:mainflow">
   <AlgorithmSetup id="2:loadData" impl="Weka.ARFFLoader(1.22)" logRuns="true">
    <ParameterSetting name="location" value="http://.../lymph.arff"/>
   </AlgorithmSetup>
   <AlgorithmSetup id="3:crossValidate" impl="Weka.Evaluator(1.25)" logRuns="true" role="CrossValidation">
    <ParameterSetting name="F" value="10"/>
    <ParameterSetting name="S" value="1"/>
    <AlgorithmSetup id="4:learner" impl="Weka.SMO(1.68)" logRuns="false" role="Learner">
     <ParameterSetting name="C" value="0.01"/>
     <FunctionSetup id="5:RBFKernel" impl="Weka.RBF(1.3.1)" role="Kernel">
      <ParameterSetting name="G" value="0.1"/>
     </FunctionSetup>
    </AlgorithmSetup>
   </AlgorithmSetup>
   <Input name="url" dataType="Tuples" value="http://.../lymph.arff"/>
   <Input name="par" dataType="Tuples" value="[name:G,value:0.1]"/>
   <Output name="evaluations" dataType="Evaluations"/>
   <Output name="predictions" dataType="Predictions"/>
   <Connection source="2:loadData" sourcePort="data" target="3:crossValidate" targetPort="data" dataType="Weka.Instances"/>
   <Connection source="3:crossValidate" sourcePort="evaluations" target="1:mainflow" targetPort="evaluations" dataType="
       Evaluations"/>
   <Connection source="3:crossValidate" sourcePort="predictions" target="1:mainflow" targetPort="predictions" dataType="
       Predictions"/>
  </Workflow>
 <OutputData name="evaluations">
  <Evaluations id="6">
   <Evaluation function="PredictiveAccuracy" value="0.831081" stDev="0.02"/>
   <Evaluation function="Precision" label="class:normal" value="0" stDev="0"/>
  ... </Evaluations>
 </OutputData>
  <Run setup="2:loadData">
  <OutputData name="data">
   <Dataset id="8" name="lymph" url="http://.../lymph.arff" dataType="Weka.Instances"/>
  </OutputData>
 </Run>
 <Run setup="3:crossValidate">
  <InputData name="data"> <Dataset ref="8"/> </InputData>
  <OutputData name="evaluations"> <Evaluations ref="6"/> </OutputData>
  <OutputData name="predictions"> <Predictions ref="7"/> </OutputData>
 </Run>
</Run>
```

Figure 3. A *workflow run* in ExpML.

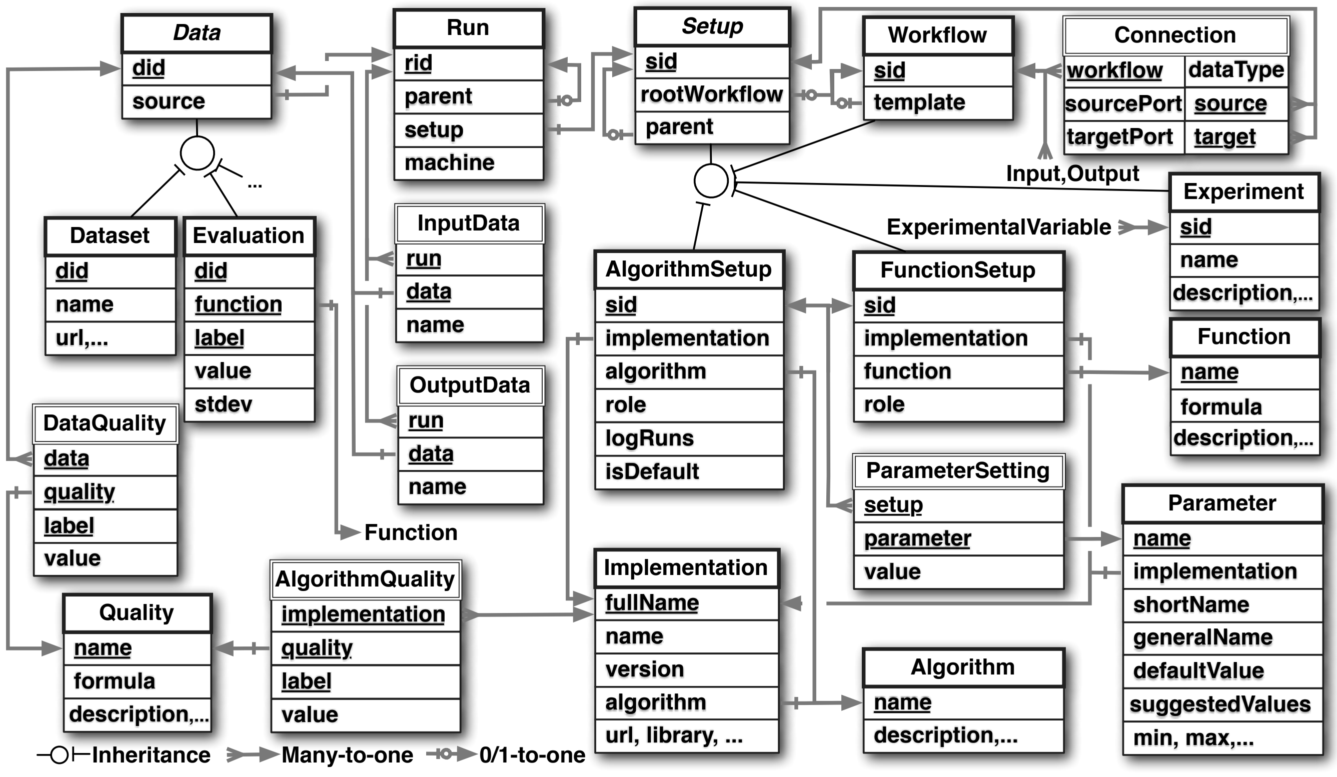


Figure 5. The general structure of the experiment database. Underlined columns indicate primary keys, the arrows denote foreign keys. Tables in italics are abstract: their fields only exist in child tables.

2.3 Organizing Machine Learning Information

The final step in our framework (see Fig. 1) is organizing all this information in searchable databases such that it can be retrieved, rearranged, and reused in further studies. This is done by collecting ExpML descriptions and storing all details in a predefined database. To design such a database, we mapped Exposé to a relational database model. In this section, we offer a brief overview of the model to help interpret the queries in the remainder of this paper.

2.3.1 Anatomy of an Experiment Database

Fig. 5 shows the most important tables, columns and links of the database model. `Runs` are linked to their input- and output data through the join tables `InputData` and `OutputData`, and data always has a *source* run, i.e., the run that generated it. Runs can have *parent* runs, and a specific `Setup`: either a `Workflow` or `AlgorithmSetup`, which can also be hierarchical. `AlgorithmSetups` and `FunctionSetups` can have `ParameterSettings`, a specific `Implementation` and a general `Algorithm` or `Function`. `Implementations` and `Datasets` can also have `Qualities`, stored in `Algorithm Quality` and `DataQuality`, respectively. Data, runs and setups have unique id's, while algorithms, functions, parameters and qualities have unique names defined in Exposé.

2.3.2 Accessing the Experiment Database

The experiment database is available at `http://expdb.cs.kuleuven.be`. A graphical query interface is provided (see the examples below) that hides the complexity of the database, but still supports most types of queries. In addition, it is possible to run standard SQL queries (a library of example queries is available. Several video tutorials help the user to get started quickly. We are currently updating the database, query interface and submission system, and a public submission interface for new experiments (described in ExpML) will be available shortly.

3 Example Queries

In this section, we illustrate the use of the experiment database.[8] In doing this, we aim to take advantage of the theoretical information stored with the experiments to gain deeper insights.

3.1 Comparing Algorithms

To compare the performance of all algorithms on one specific dataset, we can plot the outcomes of cross-validation (CV) runs against the algorithm names. In the graphical query interface, see Fig. 6, this can be done by starting with the *CrossValidation* node, which will be connected to the input *Dataset*, the outputted *Evaluations* and the underlying *Learner* (algorithm setup). Green nodes represent data, blue nodes are setups and white nodes are qualities (runs are hidden). By clicking a node it can be expanded to include other parts of the workflow setup (see below). For instance, 'Learner' expands into the underlying implementation, parameter settings, base-learners and sub-functions (e.g. kernels). By clicking a node one can also add a selection (in green, e.g. the used learning algorithm) or a constraint (in red, e.g. a preferred evaluation function). The user is always given a list of all available options, in this case a list of all evaluation functions present in the database. Here, we choose a specific input dataset and a specific evaluation function, and we aim to plot the evaluation value against the used algorithm.

Running the query returns all known experiment results, which are scatterplotted in Fig. 7, ordered by performance. This immediately provides a complete overview of how each algorithm performed. Because the results are as general as allowed by the constraints written in the query, the results on sub-optimal parameter settings are shown as well (at least for those algorithms whose parameters were varied), clearly indicating the performance variance they create. As expected, ensemble and kernel methods are dependent on the selection of the correct kernel, base-learner, and other parameter settings. Each of them can be explored by adding further constraints.

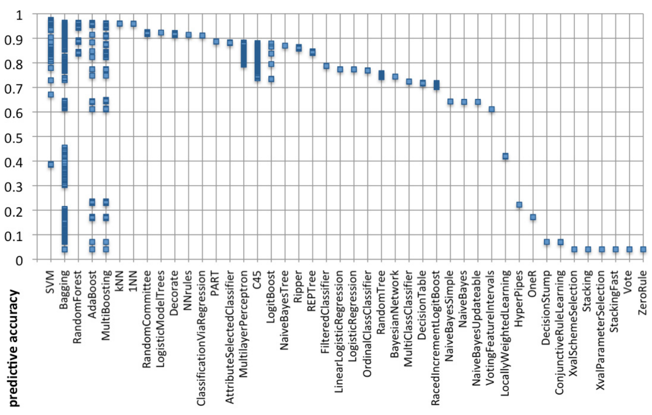


Figure 7. Performance of all algorithms on dataset 'letter'.

3.2 Investigating Parameter Effects

For instance, we can examine the effect of the used kernel, or even the parameters of a given kernel. Building on our first query, we *zoom in* on these results by adding two constraints: the algorithm should be an SVM[9] and contain an RBF kernel. Next, we select the value of the 'gamma' parameter (kernel width) of that kernel. We also relax the constraint on the dataset by including three more datasets, and ask for the number of features in each dataset.

The result is shown in Fig. 10. First, note that much of the variation seen for SVMs on the 'letter' dataset (see Fig. 7) is indeed explained by the effect of this parameter. We also see that its effect on other datasets is markedly different: on some datasets, performance increases until reaching an optimum and then slowly declines, while on other datasets, performance decreases slowly up to a point, after which it quickly drops to default accuracy, i.e., the SVM is simply predicting the majority class. This behavior seems to correlate with the number of features in each dataset (shown in brackets). Further study shows that some SVM implementations indeed tend to overfit on datasets with many attributes [12].

3.3 Preprocessing Effects

The database also stores workflows with preprocessing methods, and thus we can investigate their effect on the performance of learning

[8] See [12] for a much more extensive list of possible queries

[9] Alternatively, we could ask for a specific implementation, i.e., 'implementation=weka.SMO'.

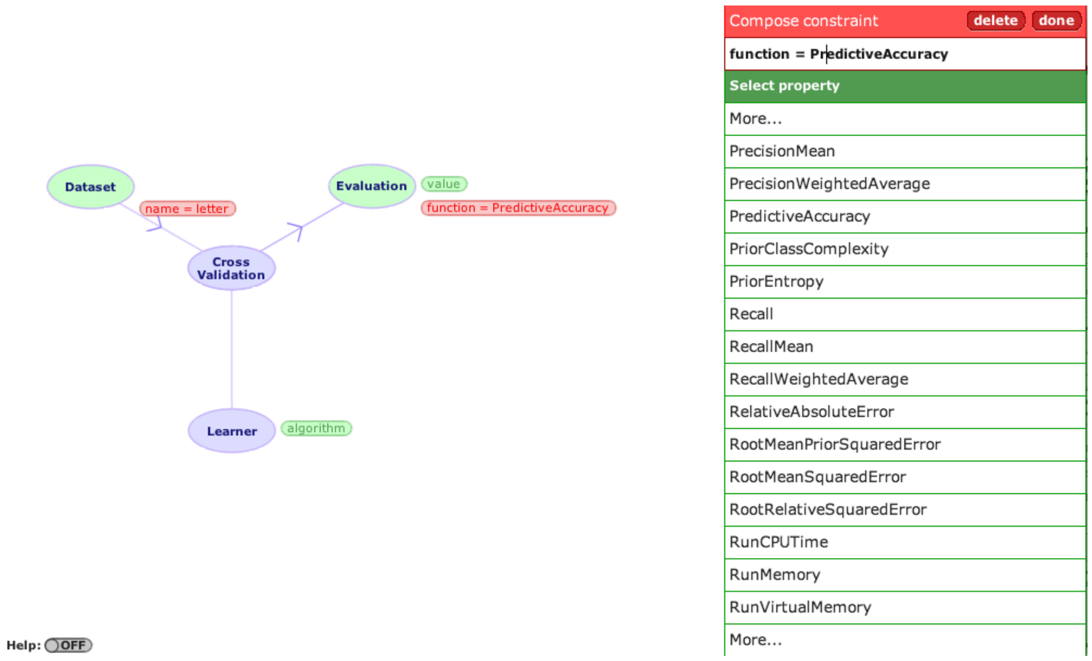


Figure 6. The graphical query interface.

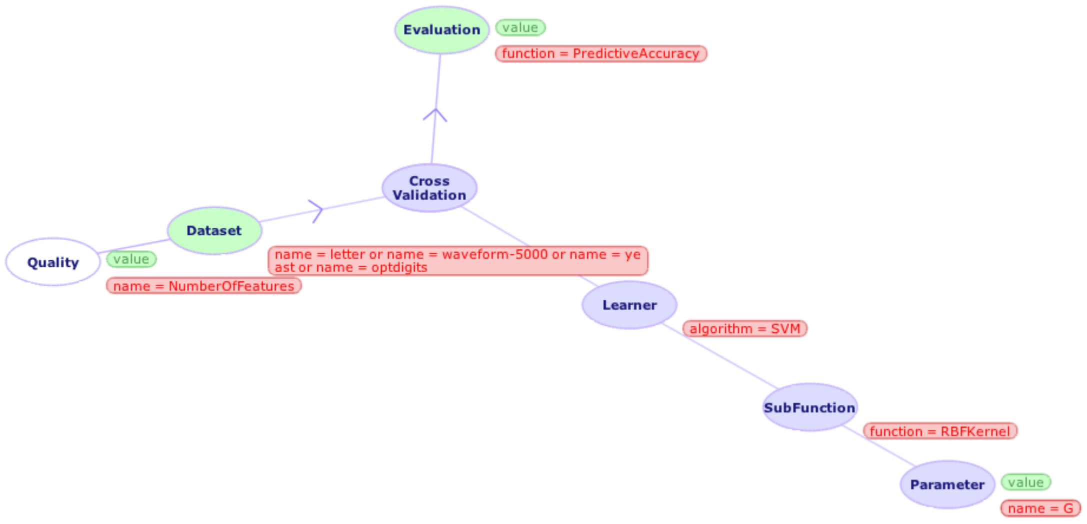


Figure 8. Querying the performance of SVMs with different kernel widths on datasets of different dimensionalities.

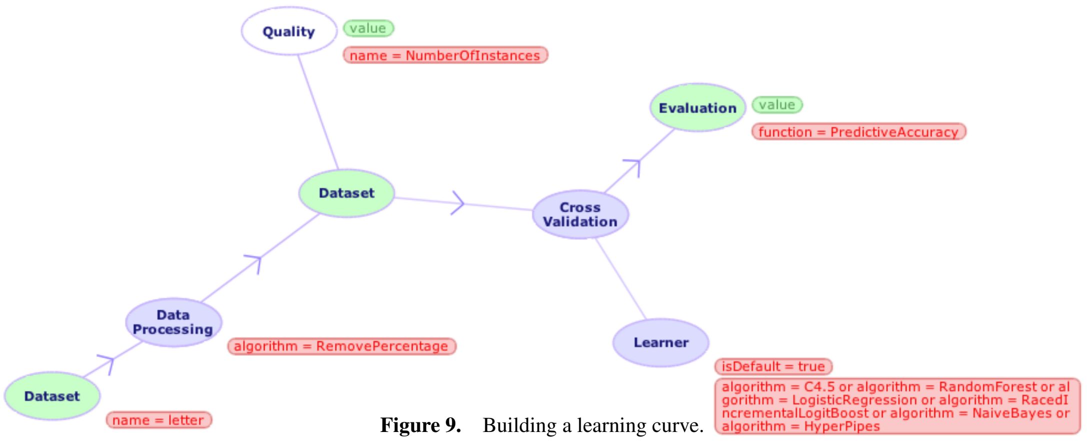

Quality
value
name = NumberOfInstances
Evaluation
value
function = PredictiveAccuracy
Dataset
Cross Validation
Data Processing
algorithm = RemovePercentage
Learner
isDefault = true
algorithm = C4.5 or algorithm = RandomForest or al
gorithm = LogisticRegression or algorithm = RacedI
ncrementalLogitBoost or algorithm = NaiveBayes or
algorithm = HyperPipes
Dataset
name = letter

Figure 9. Building a learning curve.

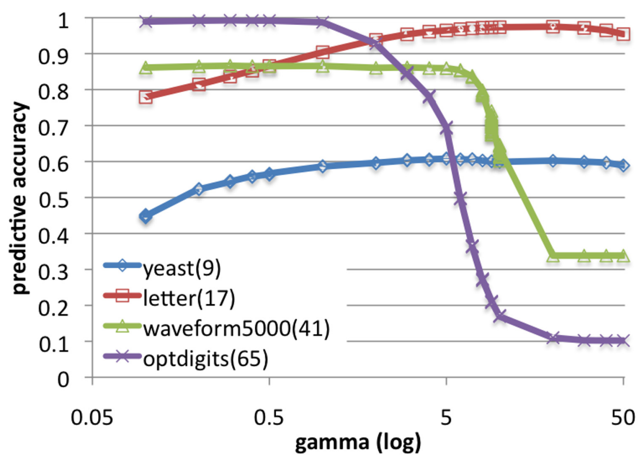


Figure 10. The effect of parameter gamma of the RBF kernel in SVMs on a number of different datasets (number of attributes shown in brackets).

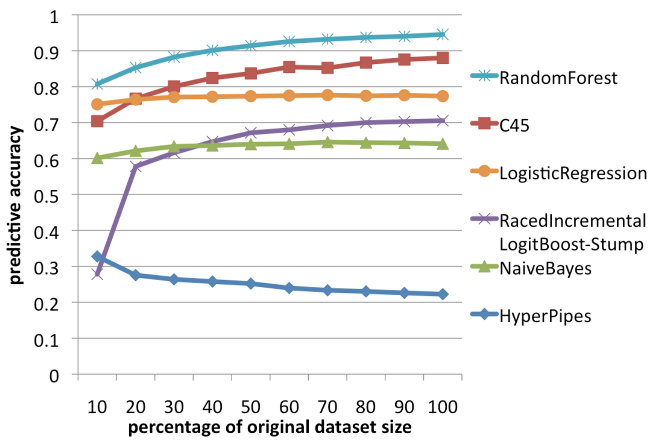


Figure 11. Learning curves on the Letter-dataset.

algorithms. For instance, when querying for workflows that include a downsampling method, we can draw learning curves by plotting learning performance against sample size. Fig. 9 shows the query: a preprocessing step is added and we query for the resulting number of instances, and the performance of a range of learning algorithms (with default parameter settings). The result is shown in Fig. 11. From these results, it is clear that the ranking of algorithm performances depends on the size of the sample: the curves cross. While logistic regression is initially stronger than C4.5, the latter keeps improving when given more data, confirming earlier analysis [9]. Note that RandomForest performs consistently better for all sample sizes, that RacedIncrementalLogitBoost crosses two other curves, and that HyperPipes actually performs worse when given more data, which suggests that its initially higher score was largely due to chance.

3.4 Bias-Variance Profiles

The database also stores a series of algorithm properties, many of them calculated based on large numbers of experiments. One interesting algorithm property is its bias-variance profile. Because the

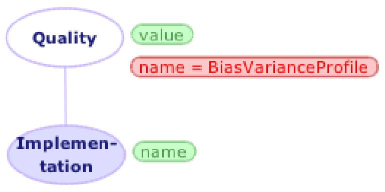


Figure 12. Query for the bias-variance profile of algorithms.

database contains a large number of bias-variance decomposition experiments, we can give a realistic numerical assessment of how capable each algorithm is in reducing bias and variance error. Fig. 13 shows, for each algorithm, the proportion of the total error that can be attributed to bias error, calculated according to [6], using default parameter settings and averaged over all datasets. The simple query is shown in Fig. 12. The algorithms are ordered from large bias (low variance), to low bias (high variance). NaiveBayes is, as expected, one of the algorithms whose error consists primarily of bias error, whereas RandomTree has relatively good bias management, but generates more variance error than NaiveBayes. When looking at the ensemble methods, Fig. 13 shows that bagging is a variance-reduction method, as it causes REPTree to shift significantly to the left. Conversely, boosting reduces bias, shifting DecisionStump to the right in AdaBoost and LogitBoost (additive logistic regression).

3.5 Further queries

These are just a few examples the queries that can be answered using the database. Other queries allow algorithm comparisons using multiple evaluation measures, algorithm rankings, statistical significance tests, analysis of ensemble learners, and especially the inclusion of many more dataset properties and algorithm properties to study how algorithms are affected by certain types of data. Please see [12] and the database website for more examples.

4 Conclusions

Experiment databases are databases specifically designed to collect all the details on large numbers of experiments, performed and shared by many different researchers, and make them immediately available to everyone. They ensure that experiments are repeatable and automatically organize them such that they can be easily reused in future studies.

This demo paper gives an overview of the design of the framework, the underlying ontologies, and the resulting data exchange formats and database structures. It discusses how these can be used to share novel experimental results, to integrate the database in existing data mining toolboxes, and how to query the database through an intuitive graphical query interface. By design, the database also calculates and stores a wide range of known or measurable properties of datasets and algorithms. As such, all *empirical* results, past and present, are immediately linked to all known *theoretical* properties of algorithms and datasets, providing new grounds for deeper analysis. This results in a great resource for meta-learning and its applications.

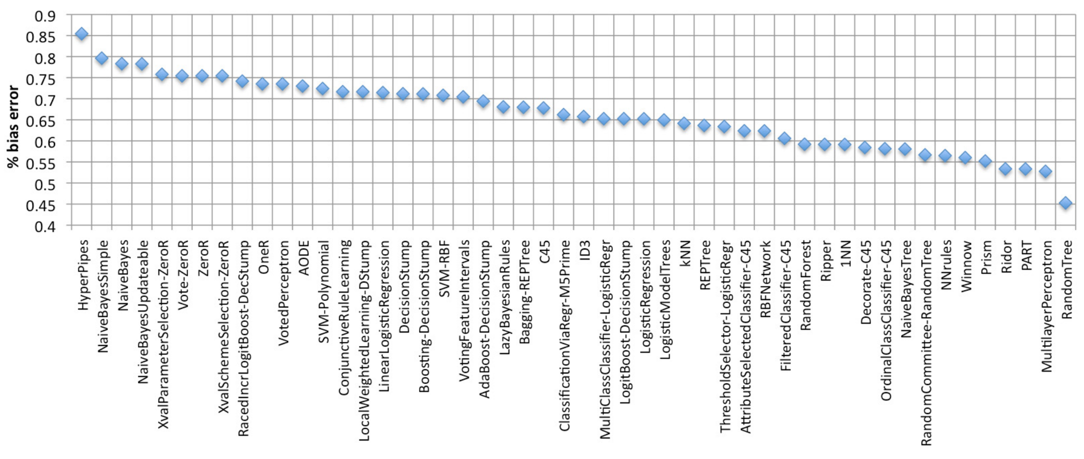


Figure 13. The average percentage of bias-related error for each algorithm averaged over all datasets.

Acknowledgements

We acknowledge the support of BigGrid, the Dutch e-Science Grid, supported by the Netherlands Organisation for Scientific Research, NWO. We like to thank Larisa Soldatova and Pance Panov for many fruitful discussions on ontology design.

REFERENCES

[1] P Brazdil, C Giraud-Carrier, C Soares, and R Vilalta, ‘Metalearning: Applications to data mining’, Springer, (2009).

[2] MA Hall, E Frank, G Holmes, B Pfahringer, P Reutemann, and IH Witten, ‘The WEKA data mining software: An update’, SIGKDD Explorations, **11**(1), 10–18, (2009).

[3] M Hilario and A Kalousis, ‘Building algorithm profiles for prior model selection in knowledge discovery systems’, Engineering Intelligent Systems, **8**(2), 956–961, (2000).

[4] M Hilario, A Kalousis, P Nguyen, and A Woznica, ‘A data mining ontology for algorithm selection and meta-mining’, Proceedings of the ECML-PKDD’09 Workshop on Service-oriented Knowledge Discovery, 76–87, (2009).

[5] J Kietz, F Serban, A Bernstein, and S Fischer, ‘Towards cooperative planning of data mining workflows’, Proceedings of the ECML-PKDD’09 Workshop on Service-oriented Knowledge Discovery, 1–12, (2009).

[6] R Kohavi and D Wolpert, ‘Bias plus variance decomposition for zero-one loss functions’, Proceedings of the International Conference on Machine Learning (ICML), 275–283, (1996).

[7] D Michie, D Spiegelhalter, and C Taylor, ‘Machine learning, neural and statistical classification’, Ellis Horwood, (1994).

[8] P Panov, LN Soldatova, and S Džeroski, ‘Towards an ontology of data mining investigations’, Lecture Notes in Artificial Intelligence, **5808**, 257–271, (2009).

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

[10] B Pfahringer, H Bensusan, and C Giraud-Carrier, ‘Meta-learning by landmarking various learning algorithms’, Proceedings of the International Conference on Machine Learning (ICML), 743–750, (2000).

[11] LN Soldatova and RD King, ‘An ontology of scientific experiments’, Journal of the Royal Society Interface, **3**(11), 795–803, (2006).

[12] J Vanschoren, H Blockeel, B Pfahringer, and G Holmes, ‘Experiment databases: A new way to share, organize and learn from experiments’, Machine Learning, **87**(2), (2012).

[13] M Zakova, P Kremen, F Zelezný, and N Lavrač, ‘Planning to learn with a knowledge discovery ontology’, Proceedings of the ICML/UAI/COLT’08 Workshop on Planning to Learn, 29–34, (2008).

Author Index