A Data Mining Ontology for Algorithm Selection and Meta-Mining*

Melanie Hilario, Alexandros Kalousis, Phong Nguyen, Adam Woznica

University of Geneva, Department of Computer Science
Artificial Intelligence Laboratory
CUI - 7, route de Drize, CH-1227 Carouge, Switzerland

Abstract. Given a learning task, the standard approach is to experiment with a broad range of algorithms and parameter settings, and select the model which performs best according to some performance criterion. One of the aims of meta-learning is to at least restrict the space of candidate models by exploiting insights gained from previous experiments. This has been done over the years by correlating dataset characteristics with the observed performance of algorithms viewed as black boxes. We have started to pry open these black boxes to sort out salient algorithm features such as the structure and parameters of the models built, the data partitions effected in data space, the cost function used and the optimization strategy adopted to minimize this cost function. The immediate goal is to build a data mining ontology formalizing the key components that together compose an algorithm’s inductive bias. Based on this ontology, a meta-learner could infer algorithm selection guidelines by correlating an algorithm’s intrinsic bias with empirical evidence of its performance.

1 Introduction

The medium-term goal of the work reported in this paper is to build an e-Laboratory for Interdisciplinary COllaborative research (e-LICO) in data mining and data-intensive sciences. The proposed e-lab comprises three layers: the e-science and data mining layers form a generic research environment that can be adapted to different scientific domains by customizing the application layer. The e-science infrastructure integrates semantic web technologies for resource sharing and integration to support collaborative scientific research. The main innovation of the data mining (DM) layer is a self-improving, planner-based DM assistant. Improvement with experience is ensured by a meta-miner that thrives on data and meta-data collected from groups of committed scientists. The term meta-mining specifically designates meta-learning applied not only to the learning

* To appear in: ECML/PKDD09 Workshop on Third Generation Data Mining: Towards Service-Oriented Knowledge Discovery (SoKD-09)
phase, but to the complete knowledge discovery process, in particular to all
tasks that require search in the space of applicable methods. The DM assistant
draws its intelligence not only from its planning and meta-mining capabilities
but also from the wealth of domain-specific and domain-independent knowledge
at its disposal.

A key source of domain-independent knowledge is the data mining ontology
(DMO), which can be viewed as the repository of the intelligent assistant’s data
mining expertise. The DMO plays a major role throughout the lifecycle of the
DM e-lab. As a compendium of knowledge about DM tasks, algorithms, data
and models, it will be used: 1) to plan the DM process using hierarchical task
networks and generate alternative workflows; 2) to guide algorithm and model
selection for critical tasks such as learning and dimensionality reduction; 3) to
meta-mine experimentation records in order to improve algorithm and model
selection; 4) to provide a controlled vocabulary for semantic annotation of DM
tools and services offered in e-LICO. Use of the DMO to plan the knowledge
discovery process is discussed in [17]. This paper describes how the DMO has
been designed to support algorithm selection and meta-learning. Section 2 pro-
poses a novel approach to the algorithm selection problem; Section 3 discusses
related work concerning algorithm selection, meta-learning, and DM ontologies.
Sections 4 and 5 give an overview of the DMO and its conceptualization of
tasks and methods to support algorithm selection and meta-learning. Section 6
concludes with a discussion of major open issues and future work.

2 Algorithm selection and meta-learning

It is now a matter of consensus that no learning algorithm can outperform all
others across broad classes of problems and domains [26]. Thus an essential step
in any machine learning experiment is selecting the algorithm that will perform
best for a given task and data set. As pointed out in a recent survey [24], research
on algorithm selection finds its origins outside machine learning, in a broader
framework that cuts across diverse areas of mathematics and computer science.
In 1976 a seminal paper by John Rice [23] proposed a formal model comprising
four components: a problem space \( \mathcal{X} \) or collection of problem instances describ-
able in terms of features defined in feature space \( \mathcal{F} \), an algorithm space \( \mathcal{A} \) or
set of algorithms considered to address problems in \( \mathcal{X} \), and a performance space
\( \mathcal{P} \) representing metrics of algorithm efficacy in solving a problem. Algorithm
selection can then be formulated as follows: Given a problem \( x \in \mathcal{X} \) characterized
by \( f(x) \in \mathcal{F} \), find an algorithm \( \alpha \in \mathcal{A} \) via the selection mapping \( S(f(x)) \) such
that the performance mapping \( p(\alpha(x)) \in \mathcal{P} \) is maximized. A schematic diagram
of the abstract model is given in Fig. 1.

In Rice’s model, selection mapping from problem space \( \mathcal{X} \) onto algorithm space \( \mathcal{A} \)
is based solely on features \( f \in \mathcal{F} \) over the problem instances. In machine learning
terms, the choice of the appropriate induction algorithm is conditioned solely on
the characteristics of the learning problem and data. Strangely, meta-learning research has independently abided by the same restriction from its inception to the present. Learned meta-rules are generally of the form: if the given dataset has characteristics $C_1, C_2, \ldots, C_n$, then use algorithm $A_1$. Sometimes the conclusion can take other forms such as "don't use algorithm $A_2"$ or "prefer $A_1$ to $A_2"$; in all cases, however, these rules represent mappings from data set features to algorithms viewed essentially as black boxes.

So far no attempt has been made to correlate dataset and algorithm characteristics, in other words to understand which aspects of a given algorithm explain its expected performance given the features of the data to be modelled. As a consequence, current meta-learners cannot generalize over algorithms as they do over data sets. To illustrate this problem, suppose that three algorithms are observed to achieve equivalent performance on a collection of datasets representing a task family. Meta-learning would yield three disjunctive rules with identical conditions and distinct recommendations. There would be no way of characterizing in more abstract terms the class of algorithms that would perform well on the given task domain. In short, no amount of meta-learning would reap fresh insights into the commonalities underlying the disconcerting variety of algorithms.

To overcome this difficulty, we propose to extend the Rice framework and pry open the black box of algorithms. To be able to differentiate similar algorithms as well as detect deeper commonalities among apparently unrelated ones, we propose to characterize them in terms of components such as the model structure built, the objective functions and search strategies used, or the type of data partitions produced. This compositional approach is expected to have two far-reaching consequences. Through a systematic analysis of all the ingredients that constitute an algorithm’s inductive bias, meta-learning systems (and data miners in the first instance) will be able to infer not only which algorithms work for specific data/task classes but—more importantly—who. In the long term, they should be able to operationalize the insights thus gained in order to combine algorithms purposefully and perhaps design new algorithms. This novel approach to algorithm selection is not limited to the induction phase; it should be applicable to other data and model processing tasks that require search in the space of candidate algorithms. The proposed approach will also be adapted to model selection, i.e., finding the specific parameter setting that will allow a
Fig. 2. Proposed model for algorithm selection

given algorithm to achieve acceptable performance on a given task. This will require an extensive study of the parameters involved in a given class of algorithms, their role in the learning process or their impact on the expected results (e.g., on the complexity of the learned model for induction algorithms), and their formalization in the data mining ontology.

The proposed revision of Rice’s model for algorithm selection is visualized in Fig. 2. It includes an additional feature space $G$ representing the space of features extracted to characterize algorithms; selection mapping is now a function of both problem and algorithm features. The revised problem formulation now is: Given a problem $x \in \mathcal{X}$ characterized by $f(x) \in \mathcal{F}$ and algorithms $a \in \mathcal{A}$ characterized by $g(a) \in G$, find an algorithm $\alpha \in \mathcal{A}$ via the selection mapping $S(f(x), g(a))$ such that the performance mapping $p(\alpha(x)) \in \mathcal{P}$ is maximized.

3 Related work

Of the few data mining ontologies reported in the literature, the majority focus on planning the DM process and building workflows [4, 28, 27], sometimes in the specific context of Grid computing [8, 7]. We shall not delve into their content which is not directly relevant to the focus of this paper. A recent paper proposes a data mining ontology aimed at "the unification of the field of data mining" [20] but defines no specific use case that it is intended to support. To our knowledge, the DMO is the first data mining ontology that has been designed to support, among other tasks, algorithm/model selection and meta-learning.

Algorithm and model selection in data mining has been the object of intensive experimentation and large-scale comparative studies, a comprehensive review of which is outside the scope of this paper. (e.g., [19, 14, 18, 9, 12]). More interestingly, choosing the right algorithm and parameter setting has been cast as a learning problem in itself: meta-learning for algorithm and model selection has been an active area of investigation for the past two decades [22, 1, 6, 16, 13, 2]. As pointed out in Section 2, most research on this topic has been implicitly done within the bounds of Rice’s framework, where black-box algorithms are selected
based solely on problem/data descriptions. An important body of meta-learning research has been devoted to dataset characterization. The Statlog project [19] yielded several dozen dataset features grouped into three categories: simple counts (e.g., number of instances, features or classes), statistical measures (e.g., feature covariance) and information-theoretic measures (e.g., feature entropy). Thereafter, other researchers have tried to expand this set by exploring new features that might yield clues on which algorithms work best for which dataset characteristics [11,10].

The use of landmarking [21,3] in the METAL project gave new impetus to the study of algorithm performance on datasets. This approach uses two sets of algorithms: so-called landmarkers and the actual candidate algorithms. Landmarkers are simple and fast learners (preferably with different inductive biases) whose performance on a set of different learning tasks serve to chart the space of learning problems. To generate meta-rules, both landmarkers and candidates are trained and evaluated on a given set of datasets. Each learning task/dataset then becomes a meta-learning instance which is characterized, in addition to standard predictive features, by the different landmarkers’ performance scores. The label of each meta-instance is the candidate algorithm with the best performance measure. The meta-learner is then trained to predict the winning algorithm by identifying tasks in which landmarkers’ performance correlate with that of a particular candidate. An example of a learned meta-rule is: If error-LINEAR-DISCR ≤ 0.0652 ∨ (num-inst ≥ 10 ∧ num-classes ≥ 5 ∧ maxclass ≥ 0.547) then choose LTREE, else choose RIPPER [21]. However, despite the use of learners to landmark areas of expertise of other learners, no attempt is made to explain observed performance of algorithms on the basis of landmarkers’ or their own characteristics. In landmarking, as before, learners remain black boxes.

### 4 The data mining ontology

As indicated in Section 1, the DMO is meant to support a number of use cases. This section presents a specific view of DMO based on the algorithm selection use case. The most important competency questions that the ontology should be able to answer include the following: Given a data mining task/data set, what is the set of potentially applicable methods/algorithms? Given a set of candidate methods/algorithms for a given task/data set, which data set characteristics should be taken into account in order to select the most appropriate one? Given a set of candidate methods/algorithms for a given task/data set, which method/algorithm characteristics should be taken into account in order to select the most appropriate one?

The DMO is currently being developed in OWL2 using the Protegé 4 editor. To support algorithm selection, it provides a conceptualization of data mining tasks, methods/algorithms and datasets. The task hierarchy is divided into two major subtrees: the first represents the user task which is more relevant to the planning use case described in [17], while the concept of GenericDMTask subsumes
four major task classes: data processing, modelling, model transformation, and model evaluation. Since the focus of this paper is on algorithm selection for classification, Fig. 3 shows an extract of the Modelling Task hierarchy where PredictiveModellingTask subsumes three subclasses distinguished by the data type of their output: categories for classification, scalars for regression, and complex objects (e.g., tuples, trees) for structured prediction.

Fig. 3. The Modelling Task subtree

For each leaf class of the task hierarchy, there is a corresponding Method subtree whose branches represent broad classes of methods that address the task. For instance, classification methods can be divided into three broad categories [5] that form the main branches of the ClassificationMethod subtree (Fig. 4). Generative methods compute the class-conditional densities \( p(x|C_k) \) and the priors \( p(C_k) \) for each class \( C_k \), then use Bayes' theorem to find posterior class probabilities \( p(C_k|x) \). They can also model the joint distribution \( p(x,C_k) \) directly and then normalize to obtain the posteriors. In both cases, they use statistical decision theory to determine the class for each new input. Examples of generative methods are normal discriminant analysis and Naive Bayes. Discriminative methods such as logistic regression compute posteriors \( p(C_k|x) \) directly to determine class membership. Discriminant functions build a direct mapping \( f(x) \) from input \( x \) onto a class label; neural networks and support vector machines (SVMs) are examples of discriminative methods.

5 Algorithm characterization in the DMO

The DMO’s conceptualization of learning algorithms hinges on the 4-tuple of concepts (Task, Data Set, Method, Model). For instance, a Classification Task is
achieved by applying a ClassificationMethod to a LabelledDataSet, producing a ClassificationModel. As we go down the classification method subtree in Fig. 4, the broad approaches described in the previous section split into more specialized methods which in turn give rise to formally specified algorithms such as those on the right side of the figure. In ontological terms, these specific method subclasses are simultaneously declared as instances of the Algorithm meta-class; their subclasses represent operators, defined as concrete software implementations of algorithms. In the same vein, these subclasses are themselves instances of the Operator meta-class. For example, DiscriminantFunctionMethod subsumes RecursivePartitioning which in turn subsumes algorithms LTREE, CART and C4.5. The black triangle to the right of C4.5 depicts its (hidden) subclasses, operators Weka-J48 and RapidMiner-DecisionTree.

We now zoom in on the key components of classification algorithms; these are represented by datatype and object properties of Algorithm instances. For the purposes of this paper, we focus on characterizing how algorithms work and ignore shallow algorithm characteristics such as ease of implementation, computational cost or readability. To do this, we must first characterize the models they were designed to produce.

A ClassificationModel is defined by its ModelStructure and by the ModelParameters that instantiate this basic structure. It is the ModelStructure that distinguishes the major classification models: a GenerativeModel's basic structure is
a JointProbabilityDistribution, that of a DiscriminativeModel is a PosteriorProbabilityDistribution. DiscriminantFunctionMethods produce diverse model structures such as decision trees and neural networks, depending on the nature of the mapping function. Within each model family, a variety of models are produced by coupling the model structure with different types/values of model parameters. To see this, consider the difference between linear and quadratic discriminant analysis under the Gaussian assumption. The NormalQuadraticDiscriminantModel has the same model structure and first model parameter as NormalLinearDiscriminantModel shown in Fig. 5. However, its second model parameter is not a single SharedCovarianceMatrix, but as many class-specific covariance matrices as there are classes in the given dataset. The outcome is a major difference in the geometry of the resulting models: one draws a linear (value of the doesDataSplit property, Fig. 5) and the other a quadratic boundary between the classes.

---

**NormalLinearDiscriminantModel**
- hasStructure: JointProbabilityDistribution
- hasParameter: ClassCondMeanMatrix
- hasParameter: SharedCovarianceMatrix
- hasDensityEstimation: Gaussian
- doesDataSplit: Linear

**NaiveBayesKernelModel**
- hasStructure: JointProbabilityDistribution
- hasParameter: TrainingSample
- hasDensityEstimation: KernelBased
- doesDataSplit: Nonlinear

---

**Fig. 5.** Characterization of two generative algorithms and models

In probabilistic (generative and discriminative) models, another property that further specifies the model structure to yield diverse models is the DensityEstimationMethod used. For instance, although the two generative models in Fig. 5 use a JointProbabilityDistribution structure, the NormalLinearDiscriminantModel uses a Gaussian distribution whereas NaiveBayesKernel [13] estimates a non-parametric distribution by fitting a Gaussian kernel around each training instance. This entails clear differences in the type of model parameters: the sufficient statistics of the estimated Gaussian distribution are the NormalLinearDiscriminantModel’s mean and covariance matrices, whereas those of NaiveBayesKernelModel are all the training instances.

Given a model structure and its parameters, the learning process is nothing more or less than the automated adjustment of these parameters to produce a fully specified, operational model. This is the task of the learning algorithm. The goal is to determine the set of parameter values that will maximize classification performance as gauged by some criterion. Independently of the manner in which the learned model will be evaluated after the learning process, the learner should
define a cost (or objective) function, which quantifies how close the current parameter values are to the optimum. Learning stops when the cost function is minimized. In its simplest version, the cost function can be simply some measure of error or more generally of loss (e.g. misclassification rate or sum of squared errors). However, minimizing training set error can lead to overfitting and generalization failure. The more general concept of CostFunction used in the DMO can be formalized as $F = \epsilon + \lambda c$, where $\epsilon$ is a measure of loss, $c$ is a measure of model complexity, and $\lambda$ is a regularization parameter which controls the trade-off between loss and complexity. The components of the cost function used in SVM learning are shown in Fig. 6.

The search for the right setting can be cast as an optimization problem that consists in minimizing the cost function. Hence an OptimizationStrategy is another essential component of a learning algorithm. In certain cases, optimization is straightforward. This is the case of NormalLinearDiscriminantAnalysis (Fig. 5), where the cost function is the log likelihood, and the maximum likelihood estimates of the model parameters have a closed form solution: it suffices to take the derivatives of the log likelihood with respect to the different parameters, set them to 0, and solve for the parameters. Logistic regression, on the other hand, estimates the maximum likelihood parameters using methods such as Newton-Raphson. SVMs use Sequential Minimal Optimization (SMO), a quadratic programming method rendered necessary by the quadratic complexity component of the cost function ($L_2$ norm in Fig. 6).

A learning algorithm's model structure and its strategy for finding the optimal model parameters are essential ingredients of its inductive bias, without which no generalization is possible. Despite such design options that restrict the space of target functions that a learning algorithm can explore, the combinatorics of search remains daunting. Thus many algorithms allow the user to restrict further the space of considered models or steer the search in regions deemed promising. This is the role of hyperparameters: they allow the user to reinforce an algorithm's built-in inductive bias by specifying choices that might be informed
by prior knowledge. In SVMs, for instance, a single generic algorithm can give rise to a number of different models based on the hyperparameter values selected by users. One such hyperparameter is the kernel function, which is defined by the kernel type (e.g., polynomial, Gaussian) and its associated parameters: the order or degree of a polynomial kernel, or the bandwidth of a Gaussian kernel. The kernel function selected by the user (depicted as \texttt{<Kernel>} in Fig.6) specifies the \texttt{LinearCombinationOfKernels} that comprises the model structure. Adjustment of the model parameters (the kernel coefficients) is controlled by yet another hyperparameter called C. As shown in the figure, the value of C becomes the regularization parameter that controls the trade-off between error (measured by Hinge Loss) and model complexity (quantified by the $L_2$ norm of the kernel coefficients). This is expressed in OWL through the SWRL rule: \texttt{If SVM(?x) \land hasCostFunction(?x, ?y) \land hasHyperparameter(?x, ?z) \land hasValue(?z, ?c) -> hasRegularizationParameter(?y, ?c)}.

### 6 Conclusion

In this paper we presented our vision of a data mining ontology designed to support meta-learning for algorithm (and subsequently model) selection. Previous research has focused obsessively on aligning experiments and performance metrics while little effort has gone into explaining observations in terms of the internal logic and mechanisms of learning algorithms. In this sense, meta-learning research has remained within the strict bounds of the Rice framework, which relates dataset descriptions to performance of algorithms viewed mainly as black boxes. We propose to extend the Rice model by adding algorithm features to dataset features as parameters of the algorithm selection function. To do this, we need to investigate the building blocks that comprise algorithms in order to reveal commonalities underlying their apparent diversity; more ambitiously, the goal is to identify the components of inductive bias that characterize each algorithm and algorithm family. Key components are: the structure and parameters of the models produced, the cost function used to quantify the appropriateness of a model, and the optimization strategy adopted to find the model parameter values that minimize this cost function.

Ongoing work involves two broad groups of issues. First, we should sort out a number of ontology engineering problems. The main hurdle we face concerns the limitations of description logic: we need the power of first-order logic to formulate the underlying mathematics of learning in an ontological framework. However, we must weigh the trade-off between expressive power and interoperability with OWL-based e-science platforms. Collaboration with specialists in formal ontologies is crucial at this point. Second, the priority data mining issue is identifying other components of bias for learning algorithms, in addition to those described in this paper. This task concerns classification in the first instance, but could be fruitfully extended to other predictive and descriptive data mining tasks.
This two-pronged research agenda is clearly beyond the reach of a single research group or even of a small-scale European project. The short-term goal is to gather interested data miners and ontology engineers to consolidate the core concepts and orientation of the DMO. The next step will be to show how the DMO can be used to improve algorithm selection through meta-learning. Here again, it is indispensable to establish broad collaborations and leverage the results of teams working actively in the area. For instance, the wealth of meta-data gathered in extensive empirical comparisons \[9\] and community-based experimentation platforms \[25\] will certainly help to overcome the well-known bottleneck of meta-data sparsity that has always hindered meta-learning research.

Acknowledgements

This work was supported by the European Union within FP7 ICT project eLICO (Grant No 231519).

References


