A Proposal for Meta-learning through a MAS
(Multi-agent System)

Juan A. Botía *, and Antonio F. Gómez-Skarmeta *, Juan R. Velasco +, and Mercedes Garjo +
Departamento de Informática, Inteligencia Artificial y Electrónica. Universidad de Murcia.
{juanbot, skarmeta}@um.es
Departamento de Ingeniería Telemática. Universidad Politécnica de Madrid.
{juanra, mga}@gpsi.dit.upm.es

Abstract. The meta-learning problem has become an important issue in the recent years. This has
been caused by the growing role of datamining applications in the global information systems of big
companies which want to obtain benefits from the analysis of its data. It is necessary to obtain faithfull
application rules that guide the datamining process in order to achieve the best possible models that
explain the databases. We follow an inductive approach to discover these kind of rules. This paper ex-
plains the MAS-based information system we use for mining and meta-learning, and how the scalability
problem is solved in order to support a community of many software agents.

1 Introduction

The problem we tackle in the present work has become very important the last years. This is the problem
of automating the process of deciding the most suitable learning algorithm for a concrete intelligent data
analysis task. We call it the Meta-learning problem. Maybe, the most important reason for the occurence
of this fact is a certain stability situation in both machine learning and datamining research fields. Concerning
the machine learning discipline, lots of algorithms have been developed that try to create approximated
theories from a number of observed examples (for a good introduction to the field you can see [6]). In the
other hand, algorithms coming from the former discipline have been successfully applied in many real world
datamining applications. But a very important problem still remains open and that is that of the lack of
methodologies to guide the process of the correct design of a intelligent data analysis experiment.
In a typical datamining session, once we are familiar with the problem domain, and have the data ready to
get it automatically analised, two important decisions have to be taken by the data miner and these are
(1) choosing a data mining task and then (2) deciding which data analysis algorithm to apply to the data.
Those decision have to be taken sequentially as (2) strongly depends on decision (1).
The data mining phase is the closest to the concrete applicability of the KDD (Knowledge Discovery in
Databases) process because it answers the question of "What do we want to do with our customer's data?" Some
datamining tasks could be classification (i.e. learning a function that maps a data example into one of
several predefined classes), regression (i.e. learning a function that maps a data item to a real-value
variable), clustering (i.e. grouping behaviours not known a priori in data), etc. Others are summarization,
dependency modeling and change and/or deviation detection.
To model the possible relations between a concrete datamining task and a learning algorithm is not easy. In
order to get deeper into the problem, we shall introduce the factors that influence decision labeled as (1).
Choosing a data mining task depends on three factors:

1. User requirements: we have clearly identified three dimensions on user requirements, the user's primary
goal (i.e. why is he using a data analysis tool), its interactivity needs, and the desired quality level of
the final data model.
2. Nature of the source data: meta-data parameters of source data like number of available tuples, number
of features, nulls percentage, etc. have to be taken into account in order to obtain a data theory with an
optimum quality.
3. Type of model to obtain: in some cases, obtaining a model that the user can easily understand and
interpretate is an important thing. However, in other cases the user is more interested in accuracy of the
model. Besides, the data mining expert could have more skills with certain types of models than with
others so he could introduce a bias in the use of the application.

Hence, the problem of choosing the datamining task is reduced to the selection of one or a few of the above
introduced types of mining techniques.
There are many automatic learning approaches and each one owns its own pletora of automatic learning
algorithms. However, all of those can be featured in terms of three concrete parameters. The first parameter,
model representation, is defined by the manner in which knowledge is represented. The second, model evaluation, refers to the strategy for evaluating learned models. The third, the search method, is the heuristic that guides the search for the optimal model.

If we found our strategy in searching in a problem space compound of these three dimensions, it is obvious that the objective datamining system will not be very useful if the learning algorithm found as optimal, for a concrete KDD session, were not present in the system. One of the directions of this work is the proposal of a multi-agent architecture that provides a repository of ready to go datamining algorithms that are encapsulated into learning agents. Each one of them offering a common interface so all of them can be managed in the same manner no matter the type of datamining algorithm. Besides, these algorithms must be distributed in different machines; perhaps with different operating systems and hardware architectures. In this way we can offer both a powerful datamining tool (in terms of performance) and a scalable scheme of integration of datamining algorithms as the system can be extended through as many hosts as we would need.

The other direction is that this system is being used, at the same time, to study the problem of meta-learning, under an inductive approach. That is, we maintain the idea of using all datamining session results as a feedback for the system in order to automatically learn what could be done in the next session. In this sense the approach is inductive if we see each datamining session results as a tuple of a meta-learning data set.

Section 2 is dedicated to introduce and explain the part of the multi-agent architecture devoted to offer a repository of intelligent analysis algorithms. The following section, 3 shows our approach to afford scalability. Section 4 gets deeper into the meta-learning issue and give details on the inductive approach we use. In section 5 related works are introduced and both conclusions and future works are pointed out.

2 A Multi-agent Architecture for Intelligent Data Analysis

In this section we will concentrate our attention on the software engineering process of the encapsulation of a machine learning algorithm into a software agent.

In order to better understand what could be a typical scenario of the final system we are proposing, that we call GEMINIS (GEneric MINing System), in this work we will begin showing a possible configuration of the system, that appears in figure 1. This figure shows a host named A that runs three different data analysis processes. There are two of them that work with decision rules, (i.e. AQ11 and ID3) and another one that is an artificial neural networks that learns with the backpropagation rule. The host named as B is running FOIL that learns first order logic predicates, and C4.5 that generates regression decision trees. At least, in the host named C there is a single algorithm, ANFIS that generates fuzzy rules from data. There are two important issues that must be taken into account after having a look at the figure:

![Diagram](image)

Fig. 1. Typical scenario of GEMINIS. Notice that there is a middleware that works as an IPC (Inter Process Communication) layer and it is based on CORBA.

- All algorithms are seen in the same manner by a possible client (i.e. they offer the same interface in terms of service calls).
- There is a middleware that offers distribution services and holds communication between clients and learning agents based on CORBA (Common Object Request Broker Arquitecture).