Knowledge-Based Systems 24 (2011) 570–588

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Saving time and memory in computational intelligence system with machine unification and task spooling

Krzysztof Grąbczewski, Norbert Jankowski *

Department of Informatics, Nicolaus Copernicus University, ul. Grudziądzka 5, 87-100 Toruń, Poland

ARTICLE INFO

Article history:
Received 4 August 2010
Received in revised form 17 November 2010
Accepted 6 January 2011
Available online 15 January 2011

Keywords:
Knowledge-based systems
Data mining
Data mining tools
Computational intelligence
Meta-learning
Machine learning

ABSTRACT

There are many knowledge-based data mining frameworks and it is common to think that new ones cannot come up with anything new. This article refutes such claims. We propose a sophisticated unification mechanism and two-tier machine cache system aimed at saving time and memory. No machine is run twice. Instead, machines are reused wherever they are repeatedly requested (regardless of request context). We also present an exceptional task spooler. Its unique design facilitates efficient automated management of large numbers of tasks with natural adjustment to available computational resources. Dedicated task scheduler cooperates with machine unification mechanism to save time and space. The solutions are possible thanks to very general and universal design of machine, configuration, machine context, unique machine life cycle, machine information exchange, configuration templates and other necessary concepts. Results gained by machines are stored in a uniform way, facilitating easy results exploration and collection by means of a special query system and versatile analysis with series transformations. No knowledge about internals of particular machines is necessary to extensively explore the results. The ideas presented here, have been implemented and verified inside Intemi framework for data mining and meta-learning tasks. They are general engine-level mechanisms that may be fruitful in all aspects of data analysis, all applications of knowledge-based data mining, computational intelligence, machine learning or neural networks methods.

1. Introduction

Automation of advanced data analysis exploiting knowledge-based systems or Computational Intelligence (CI) has recently become a very important challenge. The community has formulated many algorithms for data transformation and for solving classification, approximation and other optimization problems (for a compact review see [1] or see some handbooks [2–11]). The algorithms may be combined in many ways, so that the tasks of finding optimal solutions are very hard and require sophisticated tools. Nontriviality of model selection is evident when browsing the results of NIPS 2003 Challenge in Feature Selection [12,13], WCCI Performance Prediction Challenge [14] in 2006 or other similar contests. The competitions show that in real applications, optimal solutions are often complex models and require atypical ways of learning. Problem complexity is even clearer when solving more difficult problems in text mining or bioinformatics, where only good cooperation between different machines may provide a competitive solution. This means that before application of a final decision learner (for example a classifier) we have to prepare some transformations (and/or their ensembles) which facilitate success in further decision making.

To perform successful learning from data in an automated manner, we need some meta-knowledge i.e. knowledge about how to build efficient learning machines providing accurate solutions to the problem being solved. Our interest is to provide tools for automated meta-level analysis, to support finding the most appropriate (usually complex) models for particular problems. The meta-task is independent from particular object-level tasks within data mining and computational intelligence. We present a general approach, applicable to any kind of learning problems.

The term *meta-learning* encompasses the whole spectrum of techniques aiming at gathering meta-knowledge and exploiting it in learning processes. Although many different particular goals of meta-learning have been defined, the superior goal is to use meta-knowledge to create more accurate models and/or to find them sooner. To reach such goal, a robust CI framework that efficiently manages time and memory must be used. Here, we show some aspects of our approach to time and memory efficiency as one of the pillars of successful meta-learning.

* Corresponding author. Tel.: +48 56 6113307; fax: +48 56 6221543.
E-mail addresses: kg@is.umk.pl (K. Grąbczewski), norbert@is.umk.pl (N. Jankowski).

0950-7051/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2011.01.003

Some meta-learning approaches [15–18] based mainly on data characterization techniques (characteristics of data like number of features/vectors/classes, features variances, information measures on features, also from decision trees etc.) or on *landmarking* (machines are ranked on the basis of simple machines performances before starting the more power consuming ones). Although the projects are really interesting, they still suffer from significant limitations. The whole space of possible and interesting models is not browsed so thoroughly, thereby some types of solutions cannot be found with this kind of approaches.

We do not believe that on the basis of some simple and inexpensive description of data, it is possible to predict the structure and configuration of the most successful learner. Thus, in our approach the term *meta-learning* encompasses the whole complex process of model construction including adjustment of training parameters for different parts of the model hierarchy, construction of hierarchies, combining miscellaneous data transformation methods and other adaptive processes, performing model validation and complexity analysis, etc. So in fact, our approach to meta-learning is a search process, driven by heuristics (created and adjusted according to proper meta-knowledge) protecting from spending time on learning processes of poor promise and from the danger of combinatorial explosion. The problem of driving the search resembles the problems of context-aware design agents, where context is understood in a very broad sense including experience [19].

Meta-learning can be regarded as successful only if it efficiently uses the time it is given. It must be realized within as efficient CI environment as possible. Therefore, we have designed and implemented a general environment for complex machines learning and analysis. This article describes some of the crucial elements of the system with special emphasis on efficiency of time and memory usage. We introduce some completely new concepts in the realm of knowledge based systems including unprecedented machine unification system and dedicated approach to task spooling and running. Section 2 presents more detailed justification of the need for a new architecture and, at the same time, presents the main features of our system. Section 3 describes some of the substantial aspects of kernel design. Sections 4 and 5 present two aspects of the system that predispose it for meta-learning: task management and machine unification. Deep analysis of learning and testing results is possible thanks to the query subsystem presented in Section 7. In Section 8 we present our meta parameter search machine and its basic applications that illustrate the mechanisms described in preceding sections. All the solutions and applications have been realized in practice—they are not parts of a future project but of an existing and already working system. We do not include exhaustive tables of results obtained with the system, because the aim of this article is not to discuss particular results but to present some interesting mechanisms and illustrate how they work. The final Section 9 summarizes and discusses future perspectives of the environment.

2. Why yet another data mining system was indispensable

In order to conduct robust meta-learning, we need a universal, versatile, but also efficient and easy to use framework. It must facilitate unhampered manipulation of complex machine configurations and learning results. Because in meta-learning we must perform huge amounts of tests and compare them reliably, we need a framework capable of avoiding multiple calculations of the same tests and facilitating robust comparisons between old and new calculations, so learning must be performed for the same data samples etc. Therefore, we need a system providing all of the following features:

1. Engine-level architecture:
   (a) A unified encapsulation of most aspects of handling CI models like learning machines creation, running and removal, defining inputs and outputs of adaptive methods and their connections, adaptive processes execution, etc.
   (b) The same way of handling and operation of simple learning machines and complex, heterogeneous structures. Easy definition, configuration and running of machine hierarchies (submachines creation and management).
   (c) Easy and uniform access to learners' parameters.
   (d) Easy and uniform mechanisms for representation of machine inputs and outputs and for universal information interchange.
2. Task management:
   (a) Efficient and transparent multitasking environment for processes queuing/spooling and running on local and remote CPUs.
   (b) Versatile time and memory management for optimal usage of the computational resources.
3. Results acquisition and analysis:
   (a) Easy and uniform access to exhaustive browsing and analysis of the machine learning results.
   (b) Simple and efficient methods of validation of the learning processes, conducive to fair validation i.e. not prone to testing embezzlements.
   (c) Variety of tools for estimation of model *relevance*, analysis of reliability, complexity and statistical significance of differences [20].
4. Machine management efficiency:
   (a) Mechanisms of machine unification and a machine cache system preventing repeated calculations (significant in so large scale calculations like meta-learning).
   (b) Multilevel random seed management to facilitate providing the same learning environment and data for different learning machines.
5. Others:
   (a) Templates for creation and manipulation of complex–structure machines, equipped with exchangeable parts, instantiated during meta-learning.
   (b) Rich library of fundamental methods (especially learning machines) providing high functionality, versatility and diversity.
   (c) Simple and highly versatile Software Development Kit (SDK) for programming system extensions.

There are plenty of data mining or knowledge-based systems available today [21–24]. Software packages or systems like Weka, RapidMiner, Knime, SPSS Modeler (former Clementine), GhostMiner etc. are designed to prepare and validate different computational intelligence models, but we have found none, that would have implemented all the features mentioned above or would facilitate an extension to support the features without significant rearrangements within the architecture of the kernel. Moreover, some of the features, we propose, have not been provided by any of the systems mentioned above. These features are not just attractive—they are necessary to provide advanced and efficient data analysis using advanced meta-learning techniques.

Commercial products (like SPSS Modeler), are addressed rather to business users than to academic researchers, so they are designed as tools for data analysis on the basis of obtained models and as such they often do not reveal their internals in a satisfactory way for efficient meta-analysis. The fundamental purposes are different in the case of open source packages like Weka, RapidMiner or Knime, but they do not support the features listed above in a satisfactory way either. Some systems are limited to one research field, e.g. SNNS [25] is limited to some algorithms around neural